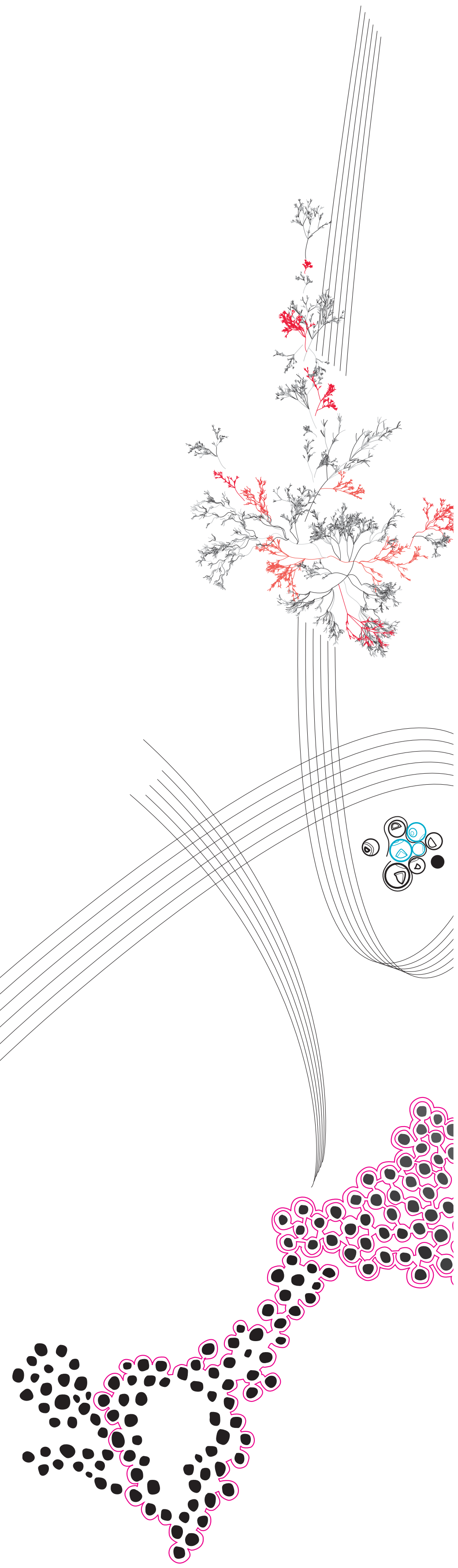
MSc Business Information Technology
Final Project



Explainable Artificial Intelligence in Job Recommendation Systems

Thi Hoang Anh Tran

Supervisor:
Dr. F.A. Bukhsh
Dr. M. Amir Haeri
Ankit Shukla (Accenture B.V.)

August, 2023

Faculty of Electrical Engineering,
Mathematics and Computer Science,
University of Twente

UNIVERSITY OF TWENTE.

Preface

This master's thesis was conducted in parallel with my internship at Accenture - Oracle Business Group (AOBG). My inspiration for building a complete explainable job recommendation system comes from one SaaS product provided by AOBG (Oracle Human Capital Management) and my growing interest in explainable AI. The content of this paper, including the experiment source code is developed using a public dataset and purely from an R&D perspective. A tailored concept note for practical implications based on the design of the Oracle HCM Demo is provided separately for the Accenture team.
The graduation project has equipped me with both technical skills and precious lessons in managing multi-stakeholder expectations and self-motivation. The findings of this project would contribute to bridging the gap between academia and industry.

This project would not have been successful without my supervisors' support from the University of Twente and Accenture.

- I would like to thank Dr. Faiza Bukhsh and Dr. Maryam Amir Haeri from the University of Twente for their guidance on methodology, experiment design, and facilitation at the university to make the project run smoothly.
- I also appreciate the support of my mentors at Accenture, Mr. Ankit Shukla and Mr. Jeroen Broecheler, for introducing the demo of Oracle HCM and for giving me feedback from industry perspectives.

Equally important, I would like to thank my family, especially my dear sister, Tran Thi Hoang Yen, for encouraging me to step out of my comfort zone and pursue a new career direction. Without her unconditional support, I could not get to the final stage. Despite the geographical distance, I never felt alone on my two-year journey in the Netherlands.

Contents

Abstract

This study takes an empirical approach to explainable AI in job recommendation systems (JRS). Using a real-world Kaggle's Career Builder 2012 dataset, the study compares the recommendations generated by different JRS designs and then decomposes the factors that contribute to the ranking results. Two post hoc techniques: LIME and SHAP, and the Explainable Boosting Machines (EBM) algorithm are implemented for explanation purposes. The results of the experiment confirm the potential of using inherently explainable models (EBM) to mitigate the trade-off between performance and explainability in recommendation tasks. JRSs using the EBM algorithm can perform on par with complex counterparts, using the Factorization Machines algorithm. These explainable JRS can generate both global and local explanations. JRS using post hoc analysis could retrieve most of the critical features of the original model when explaining at a local scope. Equally important, the explanations from both approaches can reflect the bias in the recommendation systems, which links to data quality in this case. Finally, the study proposes some use cases to illustrate the solution to challenges in explaining embedded features in job recommendation systems.

Keywords: explainable AI, recommendation systems, job recommendation systems, data-centric explanation, biases, fairness.

List of Figures

List of Tables

Acronyms

ACM RecSys Challenge ACM Recommendation System Challenge.

BoW Bag of Words.

BPR-FM Bayesian personalized ranking-Factorization Machines.

CBF Content-based filtering.

CF Collaborative Filtering.

DPEBM Differentially Private Explainable Boosting Machines.

DT Decision Trees.

e-LSTM explainable-Long Short Term Memory.

EBM Explainable Boosting Machines.

FM Factorization Machines.

GAM Generalized Additive Model.

GAMMLI Generalized Additive Models with Manifest and Latent Interactions.

GBDT Gradient Boosting Decision Trees.

JRS Job Recommendation System.

KNN K-Nearest Neighbors.

LDA Latent Dirichlet Allocation.

LIME Local Interpretable Model-agnostic Explanations.

LnDA Linear Discriminant Analysis.

LogReg Logistic Regression.

MPR Mean Reciprocal Rank.

NB Naive Bayes.

nDCG normalized Discounted Cumulative Gain.

PJFNN	Person-Job Fit Neural Network.
QDA	Quadratic Discriminant Analysis.
SHAP	SHapley Additive exPlanation.
SOTA	State-Of-The-Art.
SVM	Support Vector Machine.
XAI	Explainable Artificial Intelligence.
XGBT	XGBoost.

Chapter 1

Introduction

1.1 Introduction

The emerging application of artificial intelligence (AI) raises the same need to explain the decisions generated by opaque algorithms. Explainable Artificial Intelligence (XAI) is gaining attention as a means of improving transparency in AI systems and thus enabling trust from various stakeholders.

This project focuses on explainable AI in Job Recommendation System (JRS). While numerous research contributions have been made to improve the performance of JRS algorithms, there is still limited focus on providing comprehensible reasons behind these systems. The best-performing models in recommendation challenges (such as ACM RecSys challenges) are often ensemble models or models based on neural networks, which present challenges in providing intuitive explanations to users.

This project tackles XAI's challenges in JRS using model-specific and model-agnostic approaches. Model fidelity was used as one metric to evaluate the explanation for a model-specific approach. In addition, the target audience is lay users (i.e. job seekers) who are assumed to have limited knowledge about AI to understand the mechanism for making recommendations. Given this assumption, the final output of XAI pipelines must be human-digestible explanations, which minimize unnecessary cognitive processes for the target audience to understand these explanations.

The project aims to answer four research questions:

1. How well do State-Of-The-Art (SOTA) algorithms perform on job recommendations?
2. Which features contribute mainly to the ranking results?
3. How to evaluate the explanations generated by different XAI techniques?
4. How to make explanations digestible for lay users?

1.2 Thesis structure

The remaining content of this project is divided into five chapters as below:

- Chapter 2 presents background studies for this project. This includes an overview of explainable AI (XAI), job recommendation systems and the need for explanation, and methods of evaluating explanations.

- Chapter 3 presents the methodology of this project. It starts with a general mapping of the experiment design to answer research questions, followed by the experiment design. Key aspects of the experiment including data collection, the choice of algorithms for recommendation systems, and explanation approaches are discussed in this chapter.
- Chapter 4 explains in detail how the experiment was carried out following the design in chapter 3. The results of each phase are interpreted in this chapter.
- Chapter 5 discusses the results with respect to the proposed research questions and comparison with previous findings. In addition to a model-centric explanation, this chapter provides a data-centric explanation tailored to the data set used in this project for implications in further research.
- Chapter 6 summarizes key contributions of the project, limitations of the project, and future research opportunities.

Chapter 2

Literature Review

This chapter presents background knowledge for this project. It starts with an overview of explainable AI (XAI), job recommendation systems, and the methods of evaluating explanations.

2.1 Overview of Explainable AI

According to DARPA, AI systems are considered explainable when they communicate to human users their rationale, strengths, and weaknesses, and how they will behave in the future [4]. XAI emphasizes generating explainable models while maintaining a high level of performance (which are normally measured by accuracy), thus, enabling human to understand, appropriately trust, and effectively manage the emerging generation of AI system [1, 4].

To achieve the ultimate goal of XAI, the first step is to identify the need for explainability or the target audience of XAI models. The demand for explainability is perceived differently by various stakeholders in the AI system. For example, stakeholders can be end users who are affected by AI system decisions or take actions from AI system [4]. The group of people who "take action by the AI system", could be divided into domain experts, data scientists, and developers, managers, and regulatory bodies, as shown in figure 2.1.

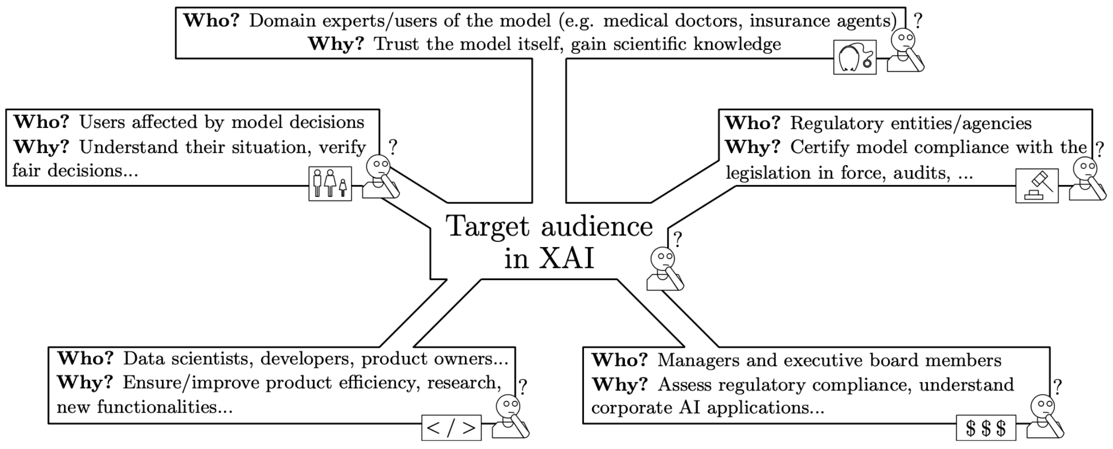

Who? Domain experts/users of the model (e.g. medical doctors, insurance agents)
Why? Trust the model itself, gain scientific knowledge
Who? Users affected by model decisions
Why? Understand their situation, verify fair decisions...
Who? Regulatory entities/agencies
Why? Certify model compliance with the legislation in force, audits, ...
Target audience in XAI
Who? Data scientists, developers, product owners...
Why? Ensure/improve product efficiency, research, new functionalities...
< / >
Who? Managers and executive board members
Why? Assess regulatory compliance, understand corporate AI applications...
$ $ $
?

FIGURE 2.1: XAI audience profile, figure from paper [1]

The second consideration of XAI could be the scope of explainability, which is often divided into global and local interpretability [1, 19, 8]. Global interpretability provides the target audience with an understanding of the input (data) and their modeled/holistic relationship with the model's output. Thus, this scope requires trained models and knowledge of algorithms and data [19]. On the contrary, local interpretability explains the predictions for one data instance or a group of similar rows [1, 8]. According to Molnar (2019), in some situations, it might be easier for the audience to look at the individual prediction of complex models, such as being linearly dependent on some features, rather than the complex dependence [19].

Third, to answer the question of "How" to explain, there is a variety of taxonomy for XAI techniques. At a general level, we could divide XAI techniques based on whether we need to use the pre-trained models for an explanation. This refers to the two approaches: model-specific and model-agnostic. Model-agnostic techniques assume that all models are black-box; thus, they do not consider the inner mechanism of these models. Instead, they take the trained outputs of the black-box models as targets of their surrogate models, and the explanations are interpreted based on the surrogate models [8, 19]. On the other hand, the model-specific [1, 8] or model-intrinsic approach [35, 19] looks at the internal decision mechanism of the models, thus, depending on different classes of algorithms. Model-specific approaches aim to approximate the black-box counterparts with an interpretable version so that it can achieve both goals of model performance and explainability.

According to Hall et al. (2019), an XAI technique can be local or global, depending on how much explanation can be generated [8]. For example, Shapley value-based techniques (such as the method [17]) by design are a local XAI technique as they compute the additive feature contribution for each prediction. Hall et al. (2019) argue that the addition of SHAP can provide a generalized explanation for the whole model. SHAP method can also be classified as model-agnostic, unlike other taxonomies such as in [1, 19], when the calculation of Shapley values is based on a class of algorithms, such as tree-based models (TreeSHAP).

Equally important, XAI only meets its goals if it can be delivered to the target audience with suitable technical knowledge. Instead of top-down approaches for XAI taxonomy as [1], Liao et al. (2022) proposed a mapping XAI question bank to map the information needs of the audience. There are 9 categories of questions: How, Why, Why not, How to be that, How to be this still, What if, Performance, Data, Output [13].

2.2 Job Recommendation System

2.2.1 Background of recommendation system

Early recommendation systems mostly use Content-based filtering (CBF) or Collaborative Filtering (CF) approaches.
In CBF recommendation systems, the profile of users and items are used as features to predict the outcome of the interaction between users and items. Thus, the recommendations are perceived as explainable when profile features are literally explainable themselves [35].
CF systems take another approach by making suggestions based on past interactions between users and items, which can be implicit or explicit. Thus, CF recommendation systems have a "domain-free" advantage, meaning that they do not rely on the profile of users and items, which are not always available. CF recommendation systems can be divided further into two approaches: neighborhood models and latent factor models. Recommendations generated by the first approach can be explainable as they are based on either neighboring items by the same user or on neighboring users for the same item. The level of complexity increases for latent models as they characterize both items and users inferred from a rating pattern. The matrix factorization-based model by Koren et al. (2009) from the Netflix competition was a well-performed candidate for this approach [6, 10].

Despite high accuracy, CF recommendation systems commonly face cold-start challenges when users and items are brand new to the system. In this aspect, content filtering is superior. Additionally, latent factor models were generally less intuitive to explain compared to content-based filtering due to the matrix factorization mechanism [35].

2.2.2 SOTA job recommendation systems

Research contribution to () could be highlighted with three competitions: ACM Recommendation System Challenge (ACM RecSys Challenge) in 2016 [1] and 2017 [2] and Kaggle's job recommendation challenge in 2012 [2]. ACM RecSys Challenges formulate the problem as a personalized top-k recommendation by computing 30 suitable jobs for a user, and user interaction with one of the top-k is counted as a success. The Kaggle challenge aims to predict the jobs that users will apply for in a given window. These competitions attracted business-oriented social networks as data contributors such as XING.com (in RecSys challenges) and CareerBuilder (in Kaggle challenge). These data sets were reused in many studies post-competition, making them an essential contribution to the field, according to Ruijt et al. (2021) [3].

Table 2.1 summarizes the selected JRS using three datasets. State-Of-The-Art (SOTA) models vary between ensemble models, hybrid recommendation systems, and deep learning models. For example, the Mim-Solution developed by Pacuk et al. (2016) consists of two stages: first, select the most relevant job posts for each user using a learning-to-rank approach; second, for each pair (users, job post), predict the probability that users will interact with the job posting using the GBDT algorithm [24].

The complexity in SOTA job recommendations also comes from the feature engineering process. For example, the winning model of RecSys 2017 by Volkovs et al. (2017) focuses on extracting features from the (user, item) pairs, including five categories: user content, item content, user-item content, user-item interaction, and user-item temporal. A total of

[1]https://recsys.acm.org/recsys16/challenge/
[2]http://www.recsyschallenge.com/2017/

TABLE 2.1: SOTA JRS using three datasets: CB12, RS2016, RS2017

No	Authors	Data	Highlights of JRS design
1	Zhang et al. (2016) [34]	RS2016	Two-stage ensemble ranking system: - First: CBF method to select 100 relevant items for each user. Algorithms: Using word2vec to compute the similarity between tag words, and build a doc2vec model to judge the content similarity between a pair of items. - Second: use the CF method to re-rank the list and select a top-n list. Algorithms: Latent Semantic model (LSI) to capture similar users in interactions
2	Mishra et al. (2016) [18]	RS2016	- CF: K-mean clustering on users and items & Cosine Similarity - Scoring: Heuristic scoring, based on impression frequency, interaction score, matching between items and users), Linear Regression - Gradient Boosting Decision Trees (GBDT)
3	Liu et al. (2016) [15]	RS2016	Ensemble ranking system, using: - Temporal-based ranking on history - Temporal re-weighted Matrix Factorization - Encoder-Decoder Sequence Model (RNNs)
4	Pacuk et al. (2016) [24]	RS2016	Two-stage system: - Candidate selection: Using a set of criteria for similarity: e.g., common tags, the popularity of items, Jaccard coefficients - Candidate ranking: using the predicted probability that the user will positively interact with the job offer. Algorithms: GBDT
5	Xiao et al. (2016) [32]	RS2016	Hierarchical ranking model with ensemble learning: - Feature representations: 3 categories: user features, job post features, user-job post interaction features - Logistic Regression (LogReg), XGBoost, GBDT, and Temporal Intensity are used to compute element scores - Ensemble GBDT for final score
6	Lian et al. (2017) [12]	RS2017	Deep negative sampling for users with negative reaction (deleted, unclicked) - Feature engineering: Profile matching, Historical Matching, Bag of Words (BoW) - Algorithms: LogReg, Support Vector Machine (SVM), Factorization Machines (FM), GBDT, LambdaMART for the ranking model
7	Volkovs et al. (2017) [30]	RS2017	Combination of content and neighbor-based models: - Intensive feature engineering: 5,416 features of five categories - Data augmentation: validation set: Randomly selected a subset of 20K unique items that had at least one positive interaction - Algorithms: XGBoost
8	Guo et al. (2014) [5]	CB2012	- Content-based recommenders using BoW and TF-IDF cosine-based similarity, entities and Jaccard similarity, social tags and Jaccard similarity; - Case-based recommenders using categories of entities, categories of entities with the explicit features; - A hybrid recommender using both the categories of entities features and the bag-of-words feature
9	Lacic et al. (2020) [11]	CB2012, RS2017, Studo Jobs	- Data augmentation: Generate sessions for CareerBuilder data. - Session-based recommendation methodology: Inferred latent session representations in a k-nearest neighbor. - Algorithms: Autoencoder, Denoising Autoencoder, and Variational Autoencoder

5,416 features were used to train the XGBoost model to predict the probability of positive interaction [30].

2.2.3 The influence of data on JRS performance

Finding high-quality public datasets for recommendation research in the HR context is a challenging task for two reasons. First, scraping persons' profiles from social network platforms such as LinkedIn is now prohibited. Second, the interaction between persons and job posts normally cannot be scraped and is heavily dependent on data providers. For these reasons, most of the existing studies in job recommendation systems leverage datasets provided in public competitions, as mentioned in the previous section [3].

Lacic et al. (2020) implemented three variants of autoencoders for session-based job recommendations and evaluated their performance on three datasets: ACM RecSys 2017, CareerBuilder 2012, and Studo - a private dataset collected from the online platform Studo Jobs [11]. The authors compared the performance of autoencoder models against different SOTA models, including the popularity-based approach, neural networks, session-based nearest neighbor models (all of these models were adapted from Hidasi et al. (2015) [9]) and matrix factorization methods for implicit feedback (Bayesian personalized ranking-Factorization Machines (BPR-FM) by Rendle et al. (2009) [26]).
Experiment results showed a great variation in model performance depending on the datasets and the metrics selected for evaluation. For example, session KNN-based models outperformed other model classes with respect to accuracy (measured by normalized Discounted Cumulative Gain (nDCG) or Mean Reciprocal Rank (MPR)). The variational auto-encoder model (VAE) developed by Lacic et al. (2020) performed second best in the ACM RecSys 2017 dataset but worse in the Studo dataset and medium performance in the CareerBuilder dataset. Matrix factorization models performed best in the ACM RecSys 2017 dataset but were ranked second worst in terms of nDCG in the other two datasets.

It is noted from the study by Lacic et al. (2020) that CareerBuilder data were augmented to generate sessions and more types of interaction to be comparable with other datasets. Thus, apart from the nature of the dataset, a data-centric explanation is necessary for users in order to draw a fair conclusion of which models are best performers in which situation.

2.3 Explanation challenges in job recommendation systems

2.3.1 Overview of explainable recommendation system

Explainability in recommendation systems can be either in the methods or in the results or both aspects [14]. Zhang and Cheng (2020) used two dimensions to classify explainable recommendation systems: "information/style of explanations" and “models for explainable recommendations” [35]. When benchmarking to the criteria of Section 2.1, most of the models in the taxonomy by Zhang and Cheng (2020) follow model-intrinsic approaches, as explanations depend on the model classes. These include neighbor-based, matrix factorization, topic modeling, graph-based, deep learning, knowledge-based, and rule-mining models. The post hoc explanation methods had fewer representatives in this survey.

In personalized recommendation systems, local explanation per user or group of users is preferred over global explanation. Integrating post hoc techniques into the component of the black-box recommendation system is a common approach to generating local explanations.

Using the MovieLens dataset, Peake and Wang (2018) used association rules trained on the output of a matrix factorization-based recommendation system to generate explanations [25]. The explanations were formed with the template "Because you watched X, I recommend Y". Morisawa and Yamana (2021) and Caio et al.(2019) incorporated LIME to generate explanations for movie recommendations. LIME-RS by Morisawa et al. (2021) set an optimal number of explanations instead of selecting all feature importance as an explanation. The excluded features include negative features that may cause misleading interpretations or unmatched features with the target users [20]. Using model fidelity@10 and model fidelity@50, LIME-RS by Morisawa et al. (2021) outperformed both XAI models presented in [21, 25].

Regarding the model-specific approach, Guo et al. (2020) developed explainable recommendation systems based on Generalized Additive Model (GAM). The algorithm called Generalized Additive Models with Manifest and Latent Interactions (GAMMLI), decomposed recommendations into three components: (1) user and item main effects, (2) manifest user-item interactions based on the observed features, and (3) latent effects of residual interactions [6]. These effects were quantified into importance ratio (IR), i.e. the percentage of variation explained by each component, and visualized using line plots and bivariate heat maps. GAMMLI has been applied to three real-world datasets: MovieLens, Santander Bank, and University Recommendation.

Regardless of using model-specific or model-agnostic approaches, the above examples of explainable recommendation systems were all built on structured datasets. In addition, the features are literally explainable such as genres of movies in MovieLens; or GPA, GRE, and university ranking as in Kaggle’s University Recommendation. Thus, even raw feature importance scores make sense to the audience.

2.3.2 Explaination challenges in job recommendation systems

There is a limited contribution toward explainable job recommendation systems. Most SOTA JRS in previous competitions aim for accuracy in recommendation rather than explainability. Two major challenges in explaining job recommendations are identified from existing studies.

First, anonymity in the data sets could be attributed to the lack of an intuitive explanation. For example, in ACM RecSys Challenge, all information about users and jobs was encoded by index terms. As a result, the importance of global characteristics is the most feasible explanation that can be derived from JRS developed on these datasets. The winning model of the ACM RecSys Challenge 2017 by Volkovs et al. (2017) integrated the XGBoost model to display the top five features within each feature category. Among 5,416 features in five categories, the most important category is user-item interaction. Given a pair of (user, item) (u, v), the interaction between the user and the item was calculated by taking the average similarities of all items with which the user (u) interacted, except the item v [30].

Second, the feature engineering process adds more challenges for model interpretation. Given the huge number of features such as Volkovs et al. (2017), feature importance scores need further filtering and the final outputs were perhaps useful only for system developers. For JRS built on text features (such as job descriptions and candidate CVs), text features normally require embedding or transformation. Embedded features are numbers in latent vectors, and thus cannot directly infer the profile of users and items.

Person-Job Fit Neural Network (PJFNN) developed by Zhu et al. (2018) is an example of an explainable job recommendation using visualization to explain embedded features. This model was based on the assumption that both job postings and resumes can be projected onto a shared latent representation. Thus, suitable candidates for a given job post were determined by the similarity of their latent representations to that of the job post. Latent vectors were plotted in the corresponding heat maps, in which darker colors mean greater similarity [36]. This study did not use data sets from ACM RecSys and CareerBuilder.

2.4 Evaluation explainable recommendation systems

This section begins with a discussion of a common belief in the trade-off between model performance and explainability. Following that, we look at different approaches to evaluate XAI models and the importance of communicating explanations in a human-understandable manner.

2.4.1 Trade-off between model performance and explainability

The trade-off between model performance and explainability has been a controversial topic. According to the authors who support the trade-off, the best-performing methods (such as deep learning) are generally less explainable, but outperform explainable models (such as decision trees) [1, 19, 4].

Rudin (2018) criticized this perception, especially when the input data were clearly structured with meaningful features. In such situations, there were no significant differences in performance between complex models such as deep neural networks, boosted decision trees, and much simpler models such as logistic regression [28]. In addition, Rudin (2018) did not support post hoc explainable methods in high-stakes decisions such as criminal justice, healthcare, and computer vision. The author argued that the surrogate models did not have perfect fidelity with respect to the original models, leading to unfaithful explanations. A proposed solution was to use models that were inherently interpretable.

Rudin's (2018) criticism of the model-agnostic approach possibly comes from a different perception of the term "explainability". The author draws a distinction between explaining the mechanism of calculation (inherently in the model, to answer the question "How") and causal inference (which is often explained by feature importance in the model-agnostic techniques to answer the question "Why").

Based on empirical studies, both model-specific and post hoc analysis approaches could be used to increase interpretability without sacrificing the original model's performance. For example, Schellingerhout et al. (2022) use a model-intrinsic approach to explain career path prediction. All models in this study were neural network-based. The finding of this study supports Rudin (2018)'s claim. Explainable neural network models performed on par with their counterparts, and the explainable-Long Short Term Memory (e-LSTM) model even outperformed LSTM for one performance metric. [29]. Post hoc analysis can help minimize the trade-off between model performance and interoperability, as can be seen from the Local Interpretable Model-agnostic Explanations (LIME)-RS case studies in section 2.3. For example, in Peak and Wang's study [25], combining association rules with recommendation systems achieved the same precision as black-box models while gaining interpretability (84% of the model's recommendations could be explained).

2.4.2 Evaluation of XAI for recommendation systems

Evaluation explanations can come from either the algorithm's perspective or the user's perspective, according to Zhang et al. (2020). Evaluation from the user's perspective directly considers the quality of the explanation. This approach had various terms for measurement, persuasiveness, effectiveness, efficiency, transparency, trustworthiness, and user satisfaction [35]. Normally, additional surveys/experiments on the target audience of the explanation are required, such as in the study [29].

Evaluation from algorithm perspectives assesses the methods used to generate explanations. As explanations can be generated globally (through a model-agnostic approach) and/or locally (through a model-agnostic approach), the evaluation should be adjusted based on the scope of the explanation.

Peak and Wang (2018) proposed the first metric to evaluate interpretability in the recommendation system: model fidelity 2.1. The authors stress the distinction between evaluating the recommendation models and evaluating the explanation models. When using the surrogate models in post hoc explainability, we aim to approximate the original black-box models. Thus, we can measure how close our approximation is by using the model fidelity [25]. It is interpreted as the percentage of explainable items (i.e., the set of retrieved items by the interpretable model) over the recommended items (the set of retrieved items by the complex recommender model). Thus, the higher the metric, the better [21].

$$\text{Model Fidelity} = \frac{|\text{Explainable items} \cap \text{Recommended items}|}{|\text{Recommended items}|} \tag{2.1}$$

Molnar (2020) also pointed out the importance of model fidelity in relation to accuracy. "If the black-box model has high accuracy and the explanation has high fidelity, the explanation also has high accuracy." [19]

Model fidelity can be easily calculated on the whole dataset as the underlying assumption is that explainable models are able to perform exact machine learning tasks as the black-box models on the whole dataset (thus, generating a global explanation).

Evaluation of a local explanation from an algorithm perspective is more challenging because the explanation by nature is case by case and may not be representative of how the black-box system works. Therefore, human evaluation was often selected for this evaluation.

From previous work by Ribero et al. [27], simulating human experiments can be used to implement such an evaluation. The general design of these experiments was to have a benchmark of **gold features** generated by self-explainable models and to check the fraction of gold features that can be recovered by explanations. I define this concept as "Feature Importance Fidelity" which is calculated by the following formula 2.2.

$$\text{Feature Importance Fidelity} = \frac{|\text{Important local features} \cap \text{Important global features}|}{|\text{Important global features}|} \tag{2.2}$$

2.4.3 Generation of human-digestible explanations

XAI involves machine learning, social science [1], and psychology. Hadash et al. (2022) argued that the ultimate output of the XAI system should be comprehensible for users as much as possible. Using the example of (visual) explanation generated by the LIME and SHapley Additive exPlanation (SHAP) methods for credit loans, the authors criticized these methods for potentially misleading interpretation without an additional cognitive process. Accordingly, a good explanation should be self-explanatory and depend as little as possible on implied knowledge [7].

The criticism of XAI techniques in study [7] is perhaps too harsh. I would attribute the potential of being misleading to the way to communicate explanations to the target audience, not the XAI techniques themselves. The following two empirical studies support the importance of communicating explanations.

In study [29], the direct explanation generated by the neural network models needed further transformation to become digestible to the audience (recruiters). Raw explanations including feature importance, attention gradient per feature, attention score per time step, and spatio-temporal attention were visualized in the diagram and combined with interpretation instructions. Although the target audience was unfamiliar with the technical terms in the raw explanations, they could easily parse the terms with additional instructions on the visualization platform and perceived the explanation as useful.

Another example to support the importance of communication "post-explanation" is the study by Yang et al. (2022). The authors designed CrystalCandle, a model-explainer interface representing a complete XAI process from raw input to the generation of human-digestible explanation [33]. Human-digestible narratives were generated based on feature hierarchy and narrative templates.

- The feature hierarchy helped to interpret the meaning of a feature, which was helpful when the original data used to train the model had a complex tabular structure.
- Narrative templates were manually constructed by model owners and then translated into appropriate code and will be substituted by feature values in Narrative Generator.

Furthermore, additional tuning was performed on the list of narratives, so only the top narratives (with the highest importance score) were presented to users to avoid overwhelming. This system was applied to the LinkedIn sales team and helps to improve the adoption rate of recommendations, user satisfaction, and revenue [33].

Chapter 3

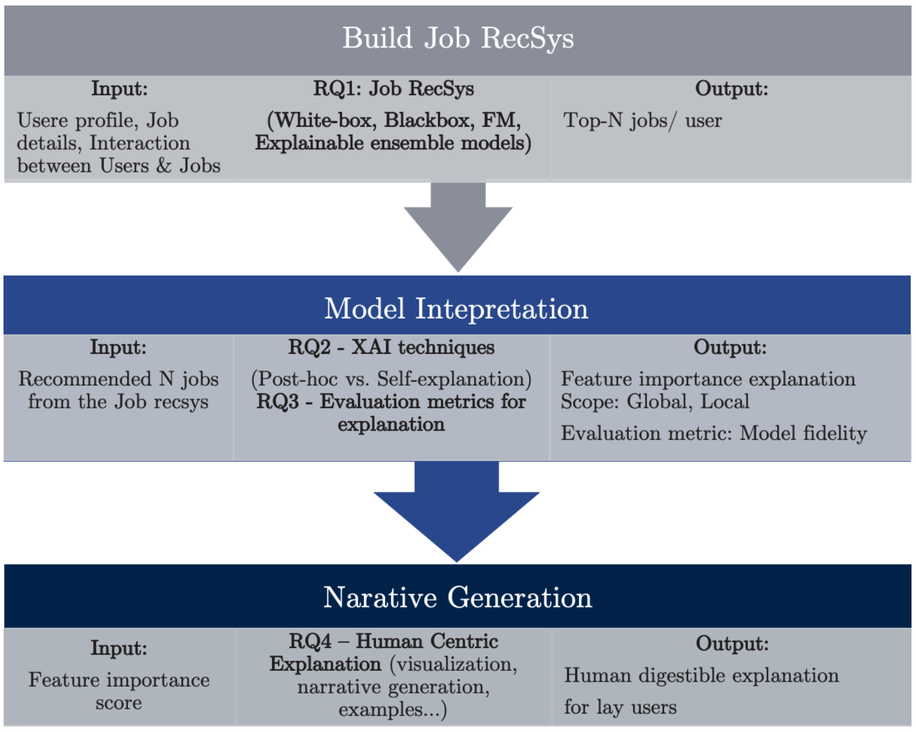
Methodology & Experiment Design

This chapter presents the methodology and experiment design to answer four research questions. It starts with an overall pipeline mapping to each research question, followed by data collection, selection of algorithms for the recommendation systems, and XAI techniques. The chapter concludes with evaluation metrics.

3.1 Methodology

This project uses an empirical approach to answer research questions. The Kaggle CareerBuilder 2012 data set [1] was used for the experiment. A detailed description of this dataset is provided in Section 3.2.

The general experiment design is mapped to research questions, as shown in figure 3.1



Build Job RecSys
Input:
Usere profile, Job details, Interaction between Users & Jobs
RQ1: Job RecSys
(White-box, Blackbox, FM, Explainable ensemble models)
Output:
Top-N jobs/ user
Model Intepretation
Input:
Recommended N jobs from the Job recsys
RQ2 - XAI techniques
(Post-hoc vs. Self-explanation)
RQ3 - Evaluation metrics for explanation
Output:
Feature importance explanation
Scope: Global, Local
Evaluation metric: Model fidelity
Narative Generation
Input:
Feature importance score
RQ4 – Human Centric Explanation (visualization, narrative generation, examples...)
Output:
Human digestible explanation for lay users

FIGURE 3.1: Mapping methodology to research questions

[1] https://www.kaggle.com/competitions/job-recommendation/data

The first step is to generate job recommendation systems formulated by a top k recommendation problem as in ACM RecSys Challenge. Complexity in recommendation systems is simulated based on a variety of model classes and feature engineering techniques. For this phase, the inputs are profiles of users, items, and interactions between users and items. The results of this phase are complex recommendations. White-box models are also used as a baseline to compare recommendation performance. The performance of recommendation systems is evaluated using quantitative metrics such as accuracy, f1, recall, and hit rate. More details of the algorithm selection and system design are provided in Section 3.3.

The second step, corresponding to two research questions (RQ2 and RQ3), focuses on interpreting the recommendation using both model-agnostic and model-specific approaches. Model-specific explanations, either locally or globally, would address both research questions quantitatively, using global feature importance and model fidelity. Model-agnostic explanation aimed at local explanation and will be evaluated based on several use cases. More details of the explanation methods are discussed in Section 3.3.

The final step is to generate human-digestible explanations (RQ4). As discussed in Section 2.3, human-digestible explanations for embedded text features require additional processing, such as visualization or text explanation. The heat map used in the Person-Job Fit Neural Network (PJFNN) study by Zhu et al. (2018) was feasible if both users and jobs had comparable profile features (e.g., skills for users and requirements for jobs). In this project, the data were not sufficient to replicate such explanations. However, visualization would be used for a general explanation.

In addition to model-centric explanations, data-centric explanations are provided for future implications.

3.2 Data Collection

When choosing a data set for implementation, both ACM RecSys data and Kaggle's CareerBuilder data have their own advantages and limitations.

ACM RecSys data 2017[2] (provided by XING.com) has various interactions between a job seeker and a job post including explicit (such as users' click on the reply and application button or deleting the recommendation) and implicit (such as users' click on the site, bookmark, impression, or recruiters' clicks on user profile). For this reason, the ACM Recsys data are more popular for building SOTA models. The only limitation of ACM RecSys data is that all profile information is anonymized, such as job role, description, or requirements for confidentiality purposes. Therefore, no verbal explanation can be generated based on these features.

CareerBuilder 2012 dataset, on the other hand, has text features such as job titles, descriptions, and requirements, which support the generation of explanations. However, it only has one interaction type which is the historical application of users for job posts. As the ultimate research goal was to generate explanation job recommendations, the CareerBuilder dataset is more suitable. This dataset contains four types of information:

- User data: Information about job seekers includes the user profile (`user.tsv`) and their work history (`user_history.tsv`). The user profile information includes categorical and numeric features. The user history only provides a list of job titles that a user has done in chronological order. This may be insufficient to represent the skills or qualifications of job seekers. Detailed descriptions of the data sets can be found in tables, as can be seen in A.1 and A.2.
- Item data: Details of a job posting. Three text features are provided, including job title, job description, and job requirement.
- Interaction data: is the job application made by users. As mentioned above, only positive interaction data is available. Thus, for the implementation, we need to perform negative sampling to have both apply and no-apply actions. Details of this process are discussed in Section 4.1

[2]http://www.recsyschallenge.com/2017/

3.3 Job recommendation system design

3.3.1 General JRS pipeline

In general, the job recommendation systems in this project follow a four-stage pipeline: (1) feature engineering, (2) shortlisting potential applications, (3) ranking applications, and (4) returning the best N results, as shown in table 3.1. Complexity elements of the recommendation systems were adapted from the previous empirical studies reviewed in section 2.2.

Step	Summary	Complexity factor	Reference (if available)
1	Feature Engineering: transform text features into lower dimension matrix	(1) TF-IDF or (2) Latent Dirichlet Association	Guo et al. (2014) [5], Zhang et al. (2016) [34] Zhu et. al. (2018) [36], Lacic et. al. (2020) [11]
2	Shortlist potential job applications for each user.	(1) Random sampling with control on positive labels and (2) Unsupervised K-Nearest Neighbors (KNN)	Volkov et al. (2017) [31], Lacic et. al. (2020) and Hidashi et. al (2015)
3	Ranking job applications, using binary classification models as scoring method	Various model classes: white-box, ensemble, factorization machines, and explainable ensemble*	Zhu et. al. (2018) [36], BPR-FM model in the study by [32], [12], Lacic et. al. (2020)
4	Finalizing result	Derived from step 3. The scoring criteria are the probability of class 1.	

Table 3.1: Overview of complexity in each step of job recommendation system explainable ensemble models in step 3 is not 100% complex as it is considered model-specific approach for explanation.

The first step in the job recommendation system is feature engineering. Text features for job details and user profiles cannot be used directly to train models. Thus, the features of the raw text are transformed into a matrix of a lower dimension, adding complexity to the recommendation systems. TF-IDF and Latent Dirichlet Allocation (LDA) were implemented for this process, which will be discussed in detail in Section 4.1. After transformation, different scenarios of features and ranking algorithms were formulated, as shown in Figures 3.2 and 3.3.

The second step is to shortlist potential applications (i.e., ranking data). Each job application is represented by a pair of (UserID, JobID). Due to the huge number of available jobs and users, forming all possibilities of combination would lead to computation complexity. In addition, the ranking ground truth is not available in the original dataset. Therefore, I will simplify this step by shortlisting a maximum of 100 applications for each user. Two approaches to generating ranking data were implemented: random sampling with control of positive labels and unsupervised KNN. Please refer to Section 4.2 for more details.

Third, potential applications are ranked for a given user (in this case, 100 applications). In this project, the interaction between Users and Jobs only has two outcomes: apply or not apply. Thus, binary classification models were used as the ranking method and the probability of class 1 was used as the ranking criteria. These machine learning-based classification methods also add complexity to recommendation systems.

The final output, job recommendations, includes four pieces of information: JobID, predicted label (Y_pred), predicted probability of the label (Y_prob), and rank (based on Y_prob). 20 recommended jobs for each user in the test sets are sorted descendingly according to the scoring criteria in the previous steps.

3.3.2 Algorithm selection for ranking methods

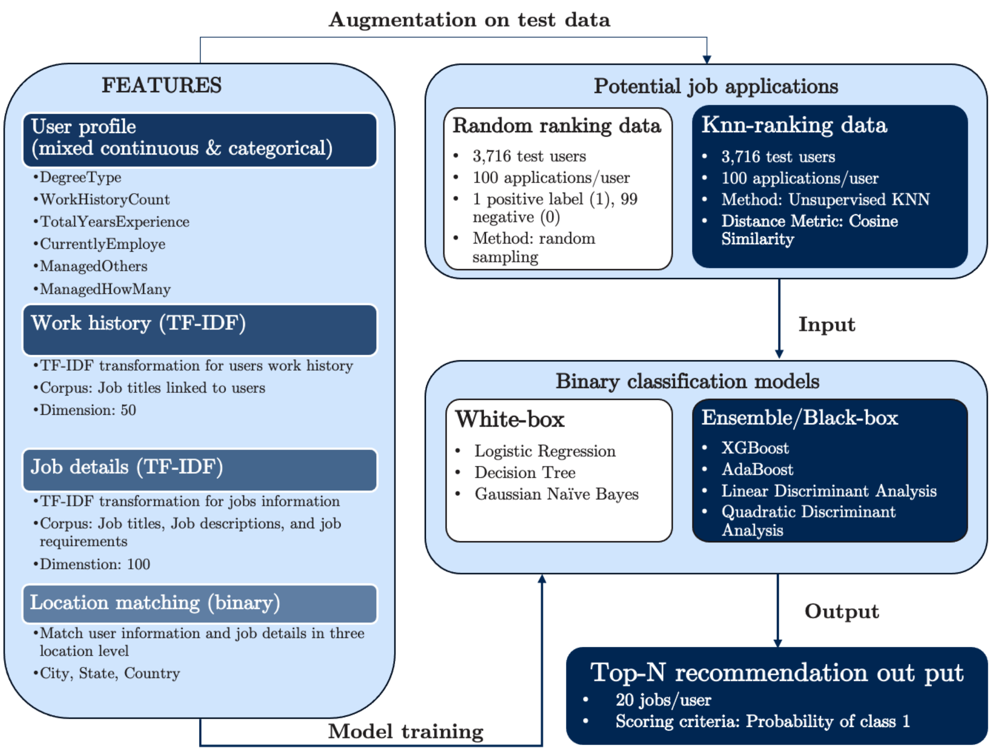

Augmentation on test data
FEATURES
User profile (mixed continuous & categorical)
•DegreeType
•WorkHistoryCount
•TotalYearsExperience
•CurrentlyEmploye
•ManagedOthers
•ManagedHowMany
Work history (TF-IDF)
•TF-IDF transformation for users work history
•Corpus: Job titles linked to users
•Dimension: 50
Job details (TF-IDF)
•TF-IDF transformation for jobs information
•Corpus: Job titles, Job descriptions, and job requirements
•Dimenstion: 100
Location matching (binary)
•Match user information and job details in three location level
•City, State, Country
Potential job applications
Random ranking data
• 3,716 test users
• 100 applications/user
• 1 positive label (1), 99 negative (0)
• Method: random sampling
Knn-ranking data
• 3,716 test users
• 100 applications/user
• Method: Unsupervised KNN
• Distance Metric: Cosine Similarity
Input
Binary classification models
White-box
• Logistic Regression
• Decision Tree
• Gaussian Naïve Bayes
Ensemble/Black-box
• XGBoost
• AdaBoost
• Linear Discriminant Analysis
• Quadratic Discriminant Analysis
Output
Top-N recommendation out put
• 20 jobs/user
• Scoring criteria: Probability of class 1
Model training

FIGURE 3.2: Job Recommendation System design: TF-IDF in feature engineering features and White-box & Black-box algorithms for ranking.

White-box models

Three white box models: LogReg, Decision Trees (DT), and Gaussian Naive Bayes (NB) were selected as baseline models, similar to the study by Zhu et al. (2018) [36]. According to the taxonomy of the Arietta et al. (2019) class, these models are transparent in three aspects: simulatability, decomposability, and algorithmic transparency, which may not require post hoc analysis. Specifically:

- Simulatability denotes the ability of a model to be simulated or thought about strictly by a human, hence complexity takes a dominant place in this class.
- The decomposability aims to explain the explainability of each component of the models, such as the input, the parameters, and the calculations.
- Algorithmic transparency refers to the ability of the user to understand the process followed by the model to produce any given output from its input data.

Intrinsic explainability in these models is attributed to feature importance scores (for DT) and coefficients (for logistic regression). Gaussian NB predicts the class based on conditional and unconditional probability; thus, it does not provide a direct explanation of the importance of the characteristic. As a result, a post hoc explanation is still needed for this model.

Ensemble & Complex models

Three ensemble/black-box models adapted from the baseline models in Zhu et al. (2018) study, that is, XGBoost (XGBT), Linear Discriminant Analysis (LnDA), and Quadratic Discriminant Analysis (QDA). Additionally, XGBoost has demonstrated outstanding performance in many ACM RecSys Challenges SOTA models. Therefore, the baseline XGBoost of ACM Challenge in 2017 was incorporated into this project, with 25 trees and a maximum depth of 2.

The seven above models were combined with TF-IDF in feature engineering, as shown in figure 3.2. All of these models were trained using `sklearn` library.

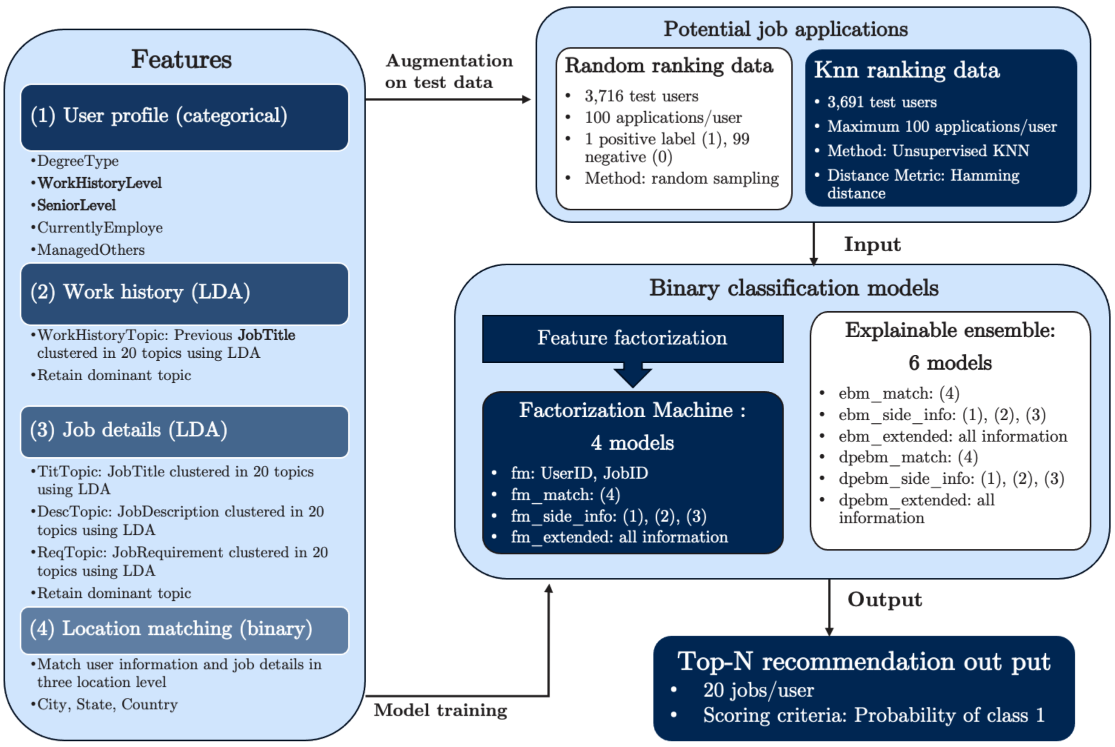

Features
(1) User profile (categorical)
•DegreeType
•WorkHistoryLevel
•SeniorLevel
•CurrentlyEmploye
•ManagedOthers
(2) Work history (LDA)
•WorkHistoryTopic: Previous JobTitle clustered in 20 topics using LDA
•Retain dominant topic
(3) Job details (LDA)
•TitTopic: JobTitle clustered in 20 topics using LDA
•DescTopic: JobDescription clustered in 20 topics using LDA
•ReqTopic: JobRequirement clustered in 20 topics using LDA
•Retain dominant topic
(4) Location matching (binary)
•Match user information and job details in three location level
•City, State, Country
Augmentation on test data
Model training
Potential job applications
Random ranking data
• 3,716 test users
• 100 applications/user
• 1 positive label (1), 99 negative (0)
• Method: random sampling
Knn ranking data
• 3,691 test users
• Maximum 100 applications/user
• Method: Unsupervised KNN
• Distance Metric: Hamming distance
Input
Binary classification models
Feature factorization
Factorization Machine : 4 models
• fm: UserID, JobID
• fm_match: (4)
• fm_side_info: (1), (2), (3)
• fm_extended: all information
Explainable ensemble: 6 models
• ebm_match: (4)
• ebm_side_info: (1), (2), (3)
• ebm_extended: all information
• dpebm_match: (4)
• dpebm_side_info: (1), (2), (3)
• dpebm_extended: all information
Output
Top-N recommendation out put
• 20 jobs/user
• Scoring criteria: Probability of class 1

FIGURE 3.3: Job Recommendation System design:
Latent Dirichlet Allocation in feature engineering - Factorization machines and Explainable ensemble models for the ranking algorithm.

Hybrid Factorization Machines models

The FM model developed by Rendle (2010) [26] combines the advantages of Support Vector Machines (SVM) with matrix factorization models. Instead of using dense parametrization like in SVM, FM models use factorized parametrization which allows estimating parameters under very spare data where SVMs fail. In other words, data from one interaction between a user and a job can help predict the parameters for related interactions. This experiment implements Bayesian Factorization Machines for classification, using `myfm`[3] library.

[3]https://pypi.org/project/myfm/

FM models were combined with LDA in feature engineering as illustrated in Figure 3.3 with four variations.

- fm: Pure matrix factorization using only UserID and JobID.
- fm_match: In addition to UserID and JobID, this model uses three location-matching features: City, State, and Country.
- fm_side_info: In addition to UserID and JobID, this model uses side information from users and jobs. These are categorical features as described in section 4.1. However, this model does not include location-matching features.
- fm_extended: This model uses all information available for users and jobs, including side information and location matching information.

3.3.3 Evaluation recommendation systems

Binary classification models used for ranking applications were evaluated using accuracy, precision, recall, and f1-score. Using recall and f1-score indicates how well the ranking models can detect augmented negative labels in the interactions.

The final topN-recommendation systems were evaluated using **hit_rate@k**. As the ranking ground truth was not available in the original dataset, a relevant recommendation was defined as an actual application by test users. As a result, revised hit_rate@k was defined as the proportion of users for whom at least one of the top k recommended jobs (from augmented potential applications) matches with the actual positive jobs (from interaction data) as illustrated in function 1.

Algorithm 1 Function to calculate hit_rate@k

```
Input: recommendation_data, evaluation_data, $k$
Output: hit_rate
1 $total\_users \leftarrow$ {unique users in recommendation_data}
  $hit\_count \leftarrow 0$
2 foreach $user\_id$ in $total\_users$ do
3     $user\_recommendations \leftarrow \{job\_i$ in recommendation data for $user\_id$ | i =1, 2, ... k}
      $actual\_jobs \leftarrow \{job\_j$ in evaluation data for $user\_id$ | label = 1}
4     foreach $rec\_job\_id$ in $user\_recommendations$ do
5         if $rec\_job\_id$ exists in $actual\_job$ then
6             $hit\_count \leftarrow hit\_count + 1$
              break
7 $hit\_rate \leftarrow \frac{hit\_count}{total\_users}$
  return $hit\_rate$
```

For each recommendation system, hit_rate were calculated at three level: 5, 10, and 20. Interpretation of hit_rate@5, for example, is the ability to recommend jobs that users will apply (ie. relevant) within the first five jobs. Thus, the higher the hit_rate, the better. In addition, a high hit_rate@5 is more desirable than a high hit_rate@10 as it illustrates the ability to retrieve relevant jobs with fewer attempts.

3.4 Selection of XAI techniques

We implemented both post-hoc (model-agnostic) and model-specific approaches for explaining the ranking of recommended jobs. White box models introduced in section 3.3 can also be considered model-specific as they have inherited explanations (i.e. feature importance). From the literature review, black-box models in general perform better than their white-box counterparts. Thus, we propose a model-specific approach using explainable resemble models as a hybrid of two groups. It has algorithm complexity yet is explainable, thus, can minimize the trade-off.

3.4.1 Post-hoc approach: LIME and Kernel SHAP

The two popular model-agnostic techniques, namely Local Interpretable Model-agnostic Explanations (LIME) [27] and SHapley Additive exPlanation (SHAP) [17] were selected for post-hoc explanation. Both techniques provide a local explanation. In other words, these techniques could explain the recommendations on a subset of data points (in this case, interactions formed in the potential application space), and per single data record (in this case, per application).

As the final output of the job recommendation is 20 jobs, the scope for local explanation is 20 applications. Both LIME and SHAP provide feature importance as the explanation, using two different approaches, as below:

LIME method

LIME, developed by Ribeiro et al. (2017) [27] is based on the general idea that all decisions generated by black-box models can be explained by a simpler model on smaller data points (locally). Thus, the process of explanation for LIME can be summarized in six steps:

1. Permuted data: Create fake data for the data point that needs to be explained.
2. Calculate the distance between the permuted data and the original observation, to measure how similar the permuted data is.
3. Make prediction using the permuted data, using the black-box model.
4. Pick m features that best describe the black-box model outcome from the permuted data. LIME uses the smallest number of features that can produce the closest outcome to the original outcome.
5. Use a simple model to fit the permuted data with m features and similarity scores as weights.
6. The feature weights from the simple models will be used to explain feature importance in the black-box model.

For the experiment, LIME was implemented using two libraries, namely interpretml[4] and the original `lime` library[5].

SHAP

SHAP developed by Lundberg et al. (2017) makes an explanation of marginal feature contribution using Shapley values. Using a game-theory approach, each feature is considered as a player, and the model will predict the outcome given all possible coalition of features.

[4]https://interpret.ml/docs/lime.html
[5]https://lime-ml.readthedocs.io/en/latest/lime.html

Shapley values attribute to each feature the change in the expected model prediction when conditioning on that feature. They explain how to get from the base value E[f(z)] that would be predicted without knowing any features of the current output f(x). Since calculating Shapley values requires forming all possible combinations of features, classical Shapley has a disadvantage in computational complexity, especially in the situation of high dimensional data. There are variations of SHAP for specific model classes, such as TreeSHAP of GPUTreeSHAP.

This project implemented the Kernel SHAP algorithm, using SHAP library[6]. Kernel SHAP is a combination of Linear LIME (i.e. LIME using linear regression as the surrogate model) and Shapley values that improve computation efficiency as well as the accuracy of the estimate [17].

3.4.2 Model-specific approach: EBM and DPEBM

Explainable Boosting Machines (EBM) developed by Lou et al. (2012) [16] and Noris et al. (2019) [23] is a tree-based cyclic gradient boosting Generalized Additive Model (GAM) with automatic interaction detection. The general form of EBM is as below:

$$g(E[y]) = \beta_0 + \Sigma f_i(x_i) + \Sigma f_{i,j}(x_i, x_j) \tag{3.1}$$

where g is the link function that adapts the model to different settings such as classification or regression; f_i is the feature function, and $f_{i,j}$ is the function of pairwise interaction between features.

The underlying theory of EBM is GAM but with more improvements. Standard GAM models the dependent variable as the sum of univariate models (feature models), making it understandable for users. However, it has a lower accuracy compared to complex models where interaction between predictors is allowed. GA2M improves GAM by adding selected terms of interacting pairs of features. EBM inherits the advantage of GA2M and improves further with cyclic gradient boosting techniques to learn the feature function f.

Differentially Private Explainable Boosting Machines (DPEBM) also developed by Noris et al. (2021) is an improved version of EBM to minimize the risk of leaking training data by machine learning models. DP-EBM ensures both performance and privacy compared to EBM by modifying three aspects: (1) Generate differentially private bins in preprocessing data, (2) Randomly select split in each epoch instead of best split (EBM) to avoid privacy budget on selecting which feature to include in each tree, and (3) Add a Gaussian noise in each split.

A comparison in pseudo-codes between EBM and DP-EBM is available in the appendix B.

3.4.3 Evaluation explanation for job recommendations

The average fidelity of the model (2) is calculated for the ensemble explainable models to evaluate how close the output of the explainable model is to the black-box model. To evaluate how well local explanations work, the fidelity rate of the importance of the characteristic was calculated using the alorithm 3.

[6]https://shap-lrjball.readthedocs.io/en/latest/generated/shap.KernelExplainer.html

Algorithm 2 Calculate model fidelity rate for each user

```
Data: source_rec, xai_rec, u_id
Result: model_fidelity_rate
1 $source\_rec \leftarrow \{job\_i$ generated by black-box JRS | where UserID = $u\_id\}$
2 $xai\_rec \leftarrow \{job\_j$ generated by explainable JRS | where UserID = $u\_id\}$
3 $fidelity \leftarrow \{source\_rec\} \cap \{xai\_rec\}$
4 $model\_fidelity\_rate \leftarrow \frac{\#\text{jobs in } fidelity}{\#\text{jobs in } source\_rec}$
5 return $model\_fidelity\_rate$
```

Algorithm 3 Calculate feature importance fidelity rate for local explanation method. Gold features are based on the global feature important score (nonzero), and local features are based on the average magnitude of the feature contribution.

```
Data: gold_features, local_features
Result: feature_importance_fidelity_rate
1 $gold\_features \leftarrow \{feature_i$ from self-explainable model | i = 1, 2, ... k $\}$
2 $local\_features \leftarrow \{feature_j$ from local explanation sample | j = 1, 2, ... m$\}$
3 $fidelity \leftarrow \{gold\_features\} \cap \{local\_features\}$
4 $feature\_importance\_fidelity\_rate \leftarrow \frac{\#\text{features in } fidelity}{\#\text{features in } gold\_features}$
5 return $feature\_importance\_fidelity\_rate$;
```

Chapter 4

Implementation and Results

This chapter presents the implementation of the experiment and the interpretation of the results of the experiment, following the methodology in chapter 3. In general, the experiment includes three phases:

- Build the JRS and generate top N recommendations: Sections 4.1, 4.2, 4.3 and 4.4.
- Explain the recommendations using two XAI approaches: Section 4.5 and 4.6
- "Post explanation" to generate human-digestible explanation: Section 4.7

The source code, detailed analysis, and high-resolution figures are available in Github repository.

4.1 Data pre-processing and Feature Engineering

4.1.1 Data cleaning

Regarding job-related features, HTML tags were cleaned from text features such as Job Titles, Job Descriptions, and Job Requirements. Jobs that do not have these attributes were crossed out.

Regarding user profiles, duplicated records and records missing important features such as "Major", "TotalYearsExperience", and "CurrentlyEmployed" for user profiles were excluded. Character-type attributes in the user information were encoded in numbers: DegreeType (1-5), ManagedOthers (0/1), and CurrentlyEmployed (0/1). Nonrelevant features were dropped, including Zip5, and ManagedHowMany (this feature did not add more information as we already had ManagedOthers).

Job applications were filtered using two following conditions:

- Users must have a work history.
- Users with more than 10 applications are excluded.
- Applications of new users who lack profile or work history information are excluded.

4.1.2 Data augmentation: Negative sampling for interaction data

The action of applying for a job is considered a positive interaction, and the CareerBuilder 2012 dataset only has a positive class. Thus, to train binary classification models, we need to generate at least another "negative" class. In this case, a negative class means that users do not apply for the job.

Negative sampling was performed using the methodology of the study by Zhu et al. (2018) [36]. Specifically, for each user in the application dataset:

- Get the list of jobs that they have applied for n JobID(s)
- Get the list of candidates' jobs: candidates jobs = all jobs (N) - applied jobs (n)
- Randomly selected n JobID(s) from the candidate jobs. Each application between (UserID, JobID) from this list is assigned label 0.

The final result of this process is an **equal ratio** of positive and negative applications for all users. At the same time, user-related features remained the same in augmented job applications.

4.1.3 Data splitting

The original data was collected in 7 windows. Each 13-day window was divided into two parts: the first 9 days were the training period and the last 4 days were the test period. Thus, I split the data using the "Split" column in the dataset.

4.1.4 Feature Engineering: Generate location matching features

For each pair of (UserID, JobID) in the application dataset, we check whether user information and job information match in location, including City, State, and Country. This resulted in three binary features where 1 = match and 0 = not match. Location-matching features were used in most of the models in Section 4.3

4.1.5 Feature Engineering: Transform the text features

The text features in include two groups: (1) Job details: Title, Description, Requirements in the dataset `jobs.tsv`, and (2) User work history: Title in the `user_history.tsv`.

Two approaches of text transformation were employed to convert text features to numeric representation: TF-IDF and LDA.

TF-IDF

TF-IDF was used to encode job details and user history. A document in the job details corpus was a combination of job titles, job descriptions, and job requirements, while a document for the work history only contained joined job titles. Thus, the TF-IDF vector for job detail had 100 features, while the TF-IDF vector for working history had 50 features.

As LIME-Tabular was implemented for post hoc explanation, the vectors were flattened into columns before training ranking models. The column names follow template"job_matrix_[index]" or "work_history_[index]" . TF-IDF flatten features were used to train white-box, black box/ensemble models.

Latent Dirichlet Allocation (LDA) - Topic Modelling

LDA was used for clustering text features for job details and users' work history. We use the same hyperparameters of the LDA models in the study by Lacic et al. (2020) [11], which resulted in 20 topics for each LDA model.

The dominant topic of each document was used as a representation of Job Titles, Job Requirements, and Job Descriptions (corresponding variable names are TitTopic, ReqTopic, and DescTopic)
For users, each document was defined as all job titles linked to that user. I used the same model as Job features, thus the outcome was a column named WorkHistoryTopic. LDA features were used in hybrid FM models, EBM models, and DPEBM models.

4.1.6 Feature Engineering: Discretizing user profile features

For white-box and ensemble black-box models, numeric features of user profiles were reused for training models, such as TotalYearsExperience, WorkHistoryCount, etc. However, for models requiring categorical data such as factorization machines, using continuous features will result in extra computation for one-hot encoding. Thus, numeric features were discretized into categorical, using a customer bin similar to the ACM RecSys data: Senior level (corresponding to TotalYearsExpence) and WorkHistoryLevel (corresponding to WorkHistoryCount). These features are used in hybrid FM models, EBM models, and DPEBM models, as shown in figure 3.3.

4.2 Generating potential applications

As discussed in the Data Overview section 3.2, there were no available target job lists or ranking ground truth that could be used to formulate potential applications. Thus, ranking data were augmented by shortlisting a maximum of 100 applications for each user before ranking, using two approaches: random sampling and unsupervised KNN. With 3,716 test users, the complete ranking data will have a maximum of 371,600 applications.

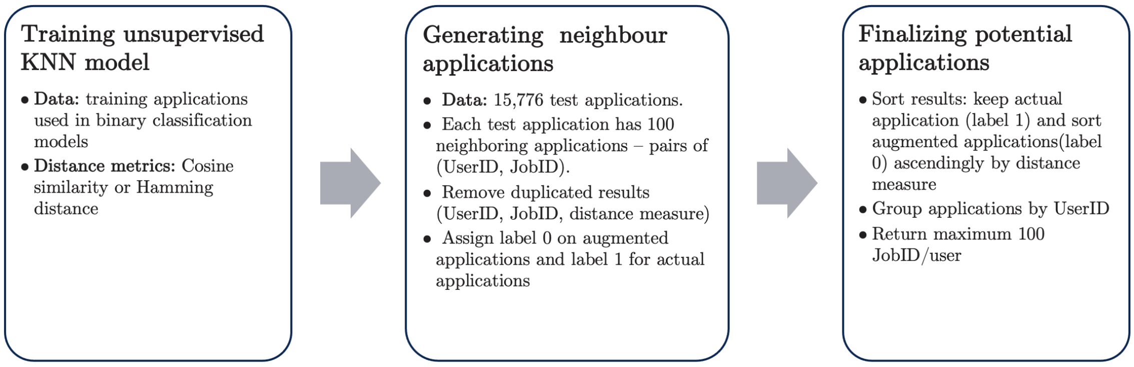

Training unsupervised KNN model
• Data: training applications used in binary classification models
• Distance metrics: Cosine similarity or Hamming distance
Generating neighbour applications
• Data: 15,776 test applications.
• Each test application has 100 neighboring applications – pairs of (UserID, JobID).
• Remove duplicated results (UserID, JobID, distance measure)
• Assign label 0 on augmented applications and label 1 for actual applications
Finalizing potential applications
• Sort results: keep actual application (label 1) and sort augmented applications(label 0) ascendingly by distance measure
• Group applications by UserID
• Return maximum 100 JobID/user

FIGURE 4.1: Potential applications generated by unsupervised KNN models

4.2.1 Potential application generation by random sampling with control on positive label

The random sampling process for test users is quite similar to negative sampling for application data, as described in Section 4.1.2. For a given user:

1. Get the list of actual jobs that users have applied (m) and keep the first application.
2. Get the list of candidate jobs: candidate jobs = all jobs (N) - applied jobs (m).
3. Randomly selected 99 jobs from the list of candidate jobs and assigned them label 0.

In the end, the ranking data contain applications for 3,716 test users, each of them having 1 successful application and 99 failed applications.

4.2.2 Potential application generation by unsupervised KNN models

Unsupervised KNN models were used to obtain 100 potential applications for each user. KNN is also considered a Collaborative Filtering recommendation because it takes into account the similarity of user-item interaction using distance measures.

The process of generating potential applications (KNN ranking data) is illustrated in Figure 4.1. Application data used in training binary classification models were used to train unsupervised KNN models. The distance metrics depended on the type of features. Specifically:

- For systems using a mixture of numeric, categorical, and TF-IDF flattening features as in 3.2, cosine similarity was used as the distance metric.
- For systems that use only categorical features and LDA dominant topics 3.3, the Hamming distance was used as the distance metric.

4.3 Training ranking models

17 binary classification models of four groups were trained using augmented interaction data, following the design in figure 3.2 and figure 3.3. Models were trained on 563,889 applications made by 146,577 users, corresponding to 373,227 jobs. After training, models were evaluated on test applications (15,736 applications) made by 3,716 users to 15,289 jobs. Four metrics were used for evaluations: accuracy, precision, recall, and f1 score.

Figure 4.2 shows performance metrics of white-box and ensemble/black-box models on binary classification tasks. Overall, all models have high performance in all four metrics. The high recall (above 80%) in all models also indicates that models can detect the augmented negative labels.

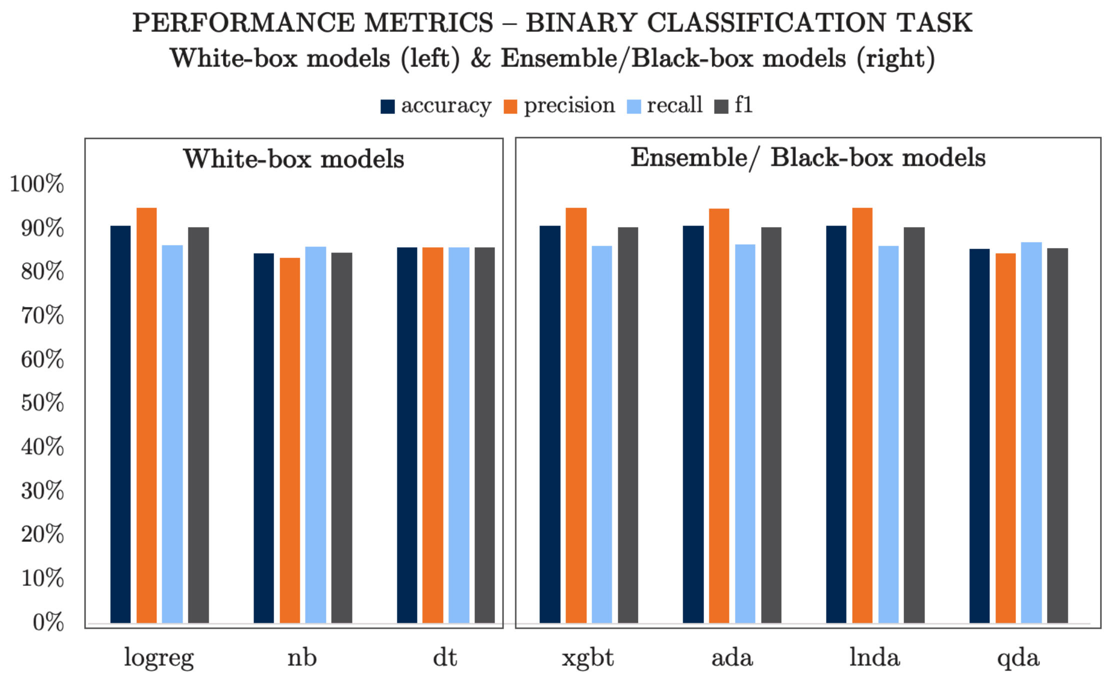


FIGURE 4.2: Performance metrics on binary classification task: White-box vs. Black-box models. All models were combined with TF-IDF for transforming text features before training.

Figure 4.3 shows the performance of the hybrid factorization machine models (FM) and the explainable ensemble models (EBM and DBEBM). These models used only categorical input to train the models. In general, explainable ensemble models perform equally well as hybrid FM models in all performance metrics. For example, fm_side_info, ebm_side_info, and dpebm_side_info used the same features: user profiles, LDA dominant topic in the user's work history, and LDA dominant topic in job details. All of these models performed at medium levels (70%).

Another observation from figure 4.3 is the consistent influence of side information on the performance of the models. The extended models (fm_extended, ebm_extended, dpebm_extended) achieved 90% in four performance metrics, similar to the white-box and black-box models analyzed above.

Analysis within variations of each model class confirmed this observation. Excluding side information and relying only on historical interaction between users and jobs (as in model fm) resulted in medium accuracy (71%). However, adding only side information

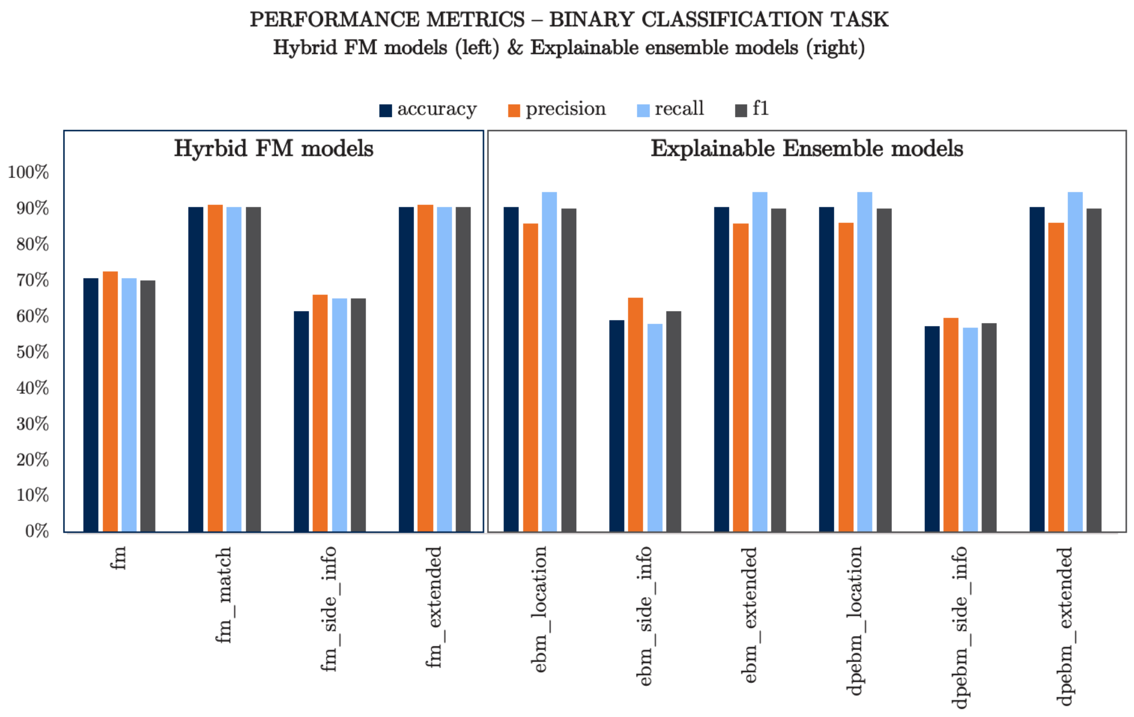


FIGURE 4.3: Performance metrics on binary classification task: FM models vs. Explainable ensemble models (EBM, DPEBM). All models were combined with LDA for transforming text features before training.

about users and jobs does not improve the performance. It even decreases the performance slightly (fm_side_info vs. fm, ebm_side_info, dpebm_side_info). In contrast, adding location information matching improves significantly the performance of the models (from 71% in fm to 91% for fm_location). Thus, the LDA dominant topic features do not contribute much to the binary classification performance.

Regarding computation efficiency, hybrid factorization machine models took the longest time for computation among the four model classes. This was due to the process of one-hot encoding features and matrix factorization. As a result, all FM models have high feature dimensions. The fm_extended model had a factorized feature dimension of 521,962.
After training, models were exported to pickle format for reuse in the recommendation tasks. sklearn-based and interpretml-based pre-trained models are generally lightweight (less than 1 GB/model). Hybrid FM pre-trained models are large (greater than 10 GB/model), taking much longer time to load the model.

4.4 Finalize ranking results and evaluation recommendation systems - RQ1

This section presents the implementation of the final step for topN-recommendation. Potential applications in step 4.2 were ranked using pre-trained models from 4.3 and returned the top 20 jobs for each user. The workflow of generating recommendations for each user is illustrated in figure 4.4

Final results from 17 job recommendation systems were evaluated using hit_rate@5, hit_rate@10, and hit_rate@20. The findings answer research question 1: **How well do the SOTA models perform on recommendation system**.

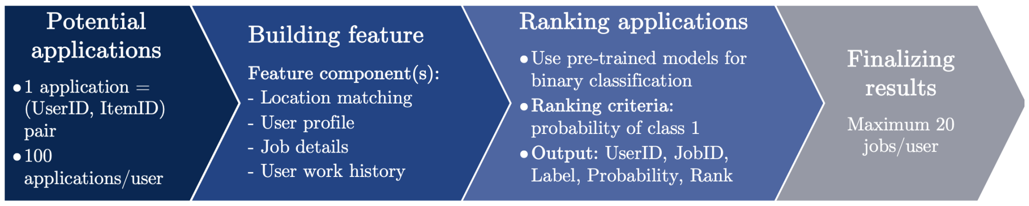


FIGURE 4.4: Top-N recommendation work-flow (for a single user)

Evaluation on recommendation system using random ranking data

Figure 4.5 summarises the hit rate calculated on different JRS. For JRS using white-box and black-box ranking models, the hit rates remained stably high across three levels (5, 10, 20). JRS using XGBoost models had the highest performance with 100% correct positive recommended jobs.

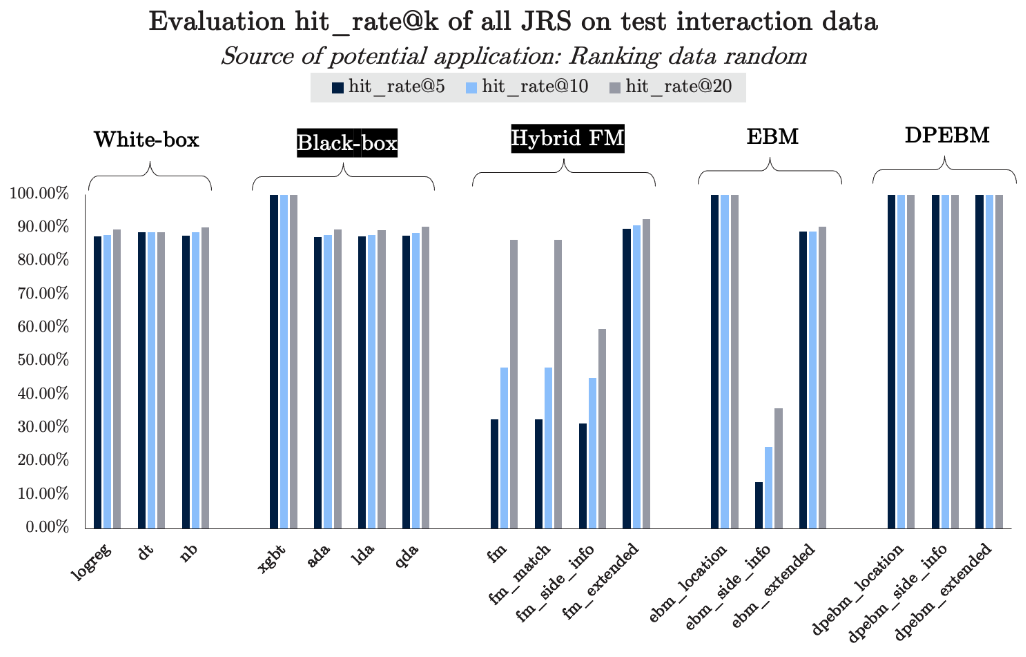


FIGURE 4.5: Evaluating hit rate of JRS using different ranking algorithms. Source of potential applications: random ranking data. White-box and black-box ranking algorithms were combined with TF-IDF in feature engineering. The remaining algorithms were combined with LDA in feature engineering.

For JRS using factorization machines as ranking methods, the hit rates increased gradually when adding more side information. fm_extended, the model with all side information from users, jobs, and location matching performed equally well as white-box and black-box ranking methods. Unlike in binary classification tasks, LDA dominant topics in user work history and job details contributed to the recommendation performance. For example, comparing fm_match (only including location matching features) to fm_extended, the latter model achieved a remarkably higher hit rate at the first 5 and 10 recommended jobs.

Finally, for JRS using explainable ensemble models as ranking methods (EBM and DPEBM), the side information seems not to influence the performance. All JRSs using the DPEBM ranking algorithm achieve perfectly correct hit rates regardless of which side information was included.

Evaluation on recommendation system using unsupervised ranking data

Figure 4.6 summarises the hit rate calculated on JRS using potential applications from unsupervised KNN models (as in section 4.2). In general, JRS using KNN ranking data performs much worse than those using random ranking data in the previous section. The highest hit rate is 4.25%. In addition, hit_rates are increasing when relaxing the top k recommended jobs. Among all model classes, only XGBT has a stable hit_rate across three levels of top-k.

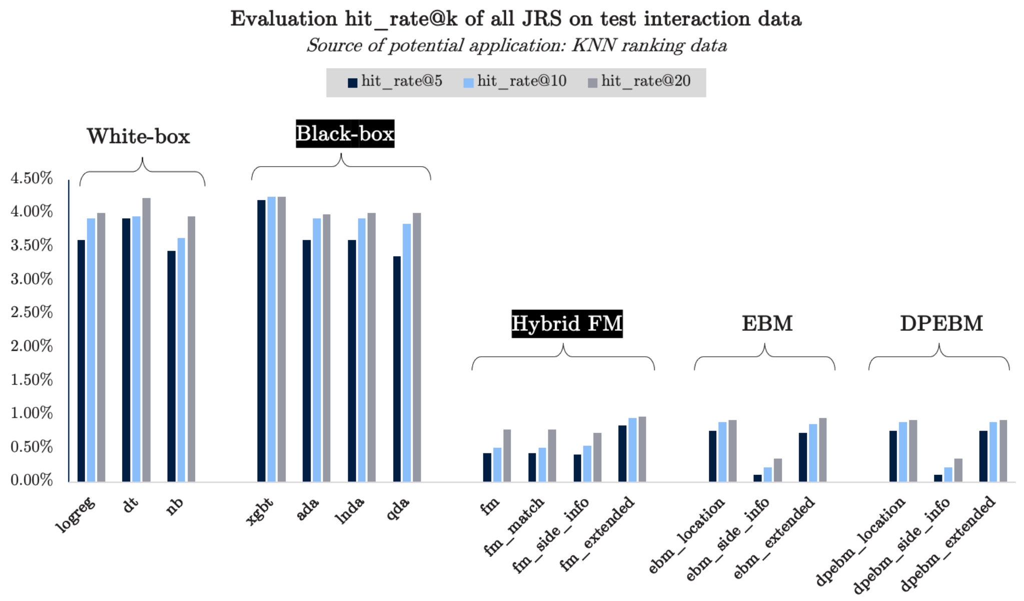


FIGURE 4.6: Evaluating hit rate of JRS using different ranking algorithms. Source of potential applications: unsupervised KNN model. White-box and black-box ranking algorithms were combined with TF-IDF in feature engineering. The remaining algorithms were combined with LDA in feature engineering.

It can be seen from the figure that JRS using white-box and black-box ranking algorithms performs better than JRS using hybrid-FM, EBM, and DPEBM algorithms. This means that feature engineering has influenced the performance of job recommendations. Specifically, the white-box and black-box ranking models used numeric and TF-IDF transformation matrices, so the features have more details than categorical features in the hybrid FM, EBM, and DPEBM ranking models.

Influence of source of potential applications on JRS performances

The low hit rates of JRS using KNN ranking data raise the question of whether using model-based to shortlist applications is worse than random ranking data. According to previous studies, the use of collaborative filtering before ranking job applications contributes to the personalization of the recommendation. In other words, the hit rates are likely to be higher compared to ranking on randomly selected applications.

Figure 4.7 shows the comparison in the average probability of two classes (0: Not apply and 1: Apply) on different JRS combined with different sources of potential applications. The classification algorithms in all JRS use the same threshold of 50% to determine the final class. The higher the predicted probability, the greater the certainty in the predicted class. For example, given the same input, if model A predicts class 1 with 55% and model B also predicts class 1 with 80%, then the decision by model B has a higher certainty than model A. Thus, comparing the average of probability may reveal the level of certainty in decision-making by different JRS.

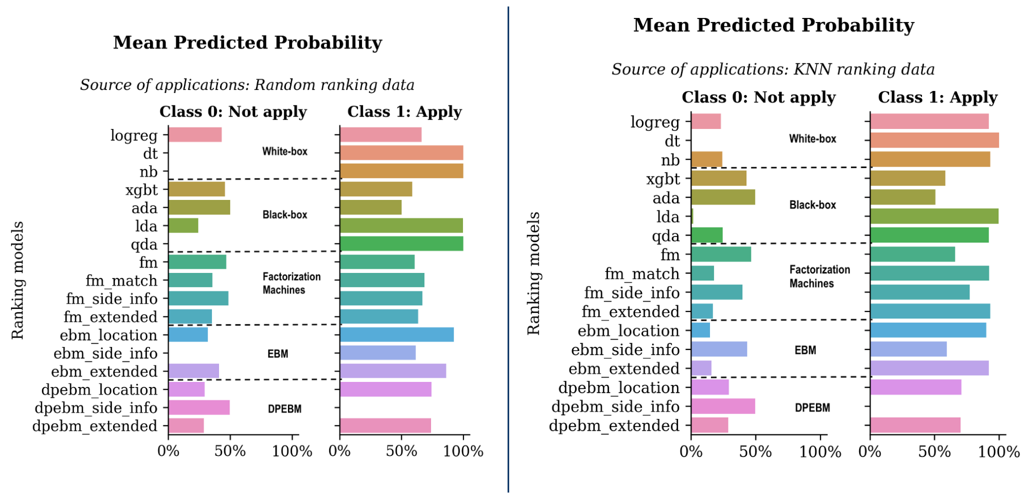


FIGURE 4.7: Comparison average predicted probability. JRS uses different sources of potential applications: random ranking data (left) and unsupervised KNN ranking data (right).

It can be seen from the right figure that the average probability of class 1 is clearly higher than that in class 0, especially for factorization machine models. In other words, the use of potential applications from the KNN model helps the JRS predict positive labels with higher certainty.

In the figure on the left, most FM models have an average probability of class 0 close to 50%. This means that the final decision contains more uncertainty since it just passes the threshold of 50%.

This analysis indicates that adding context by using unsupervised KNN to shortlist applications may contribute to higher certainty in final job-ranking decisions. The low recommendation of JRS using KNN ranking data may come from other causes, for example, sparsity in the data. A data-centric explanation is provided in Chapter 5.

4.5 Explaining recommendations - RQ2

4.5.1 Summary experiment

In this part of the experiment, different explainers need to decompose the features that contribute to the recommendations. The explanations can be global or local. The findings answer **Research Question 2: Which features contribute mainly to the ranking results?**.

The input of the explainers are the application labels (0/1) and/or the predicted probability generated by the recommendation systems. These "black-box inputs" vary depending on the design of the recommendation systems. Thus, to evaluate the utility of the explanation by different model interpreters (local vs. global), the black-box inputs should be fixed. As a result, test applications are used to evaluate ranking algorithms in section 4.4 that was used for this purpose.

The global explanations were generated using all test applications. Local explanation, on the other hand, took a subset of applications. The sample size in this case is 20 applications, as each user would have a maximum of 20 recommended jobs. The detail of the sample used to generate an explanation is available in table F.1. The labels of these applications were assumed to be recommendations generated by different models in Section 4.4.

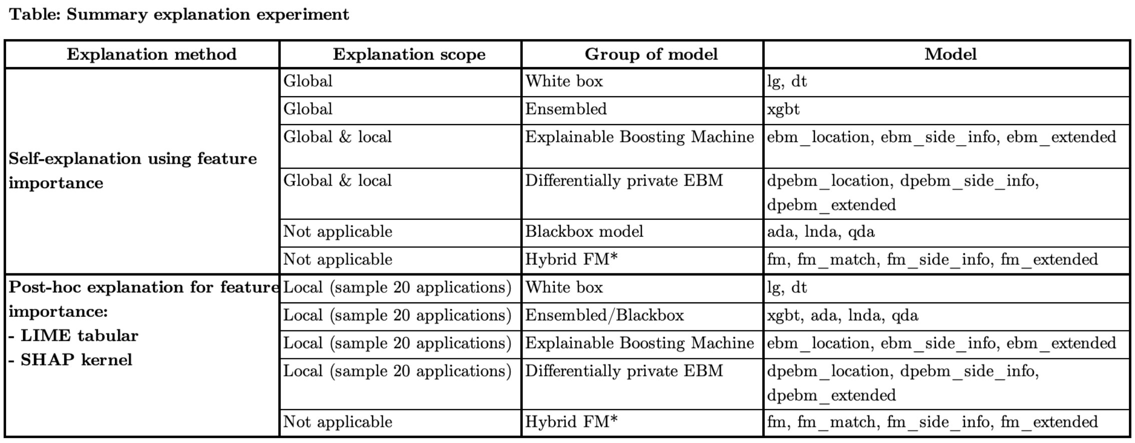


Table: Summary explanation experiment

Explanation method	Explanation scope	Group of model	Model
Self-explanation using feature importance	Global	White box	lg, dt
	Global	Ensembled	xgbt
	Global & local	Explainable Boosting Machine	ebm_location, ebm_side_info, ebm_extended
	Global & local	Differentially private EBM	dpebm_location, dpebm_side_info, dpebm_extended
	Not applicable	Blackbox model	ada, lnda, qda
	Not applicable	Hybrid FM*	fm, fm_match, fm_side_info, fm_extended
Post-hoc explanation for feature importance: **- LIME tabular** **- SHAP kernel**	Local (sample 20 applications)	White box	lg, dt
	Local (sample 20 applications)	Ensembled/Blackbox	xgbt, ada, lnda, qda
	Local (sample 20 applications)	Explainable Boosting Machine	ebm_location, ebm_side_info, ebm_extended
	Local (sample 20 applications)	Differentially private EBM	dpebm_location, dpebm_side_info, dpebm_extended
	Not applicable	Hybrid FM*	fm, fm_match, fm_side_info, fm_extended

FIGURE 4.8: Model interpretation experiment setting

The explanation experiment setting is presented in figure 4.8. First, global self-explanations in terms of feature importance were obtained in the following models:

- Ensemble explainable models (i.e. **model-specific explanation**): EBM and DPEBM models (section 4.5.2, 4.5.3)
- White-box models: LogReg, DT (section 4.5.4)
- Black-box model having feature importance score: XGBT (section 4.5.4)

Regarding the local post hoc explanation, LIME and Kernel SHAP took the labels of the applications and used different ranking algorithms to generate the contribution scores of the features. It was noted that the current LIME and SHAP libraries could not perform calculations on FM models due to the high feature dimensions. The feature importance plots of the local post hoc explanations were compared with their global counterparts (Section 4.5.5, 4.5.6)

Finally, local self-explanations could be obtained from the EBM and DPEBM models. Therefore, to evaluate the consistency of the explanation, the feature importance plots of one application (UserID = 13, JobID = 821691) were compared by different explainers.

4.5.2 Global explanation by model-specific approach: EBM models

Figure 4.9 presents the term importance for the whole model for three variations of EBM models. The "term importance" is the average absolute contribution (i.e., score) that each term (i.e., feature or interaction of features) makes to predictions averaged across the training dataset.

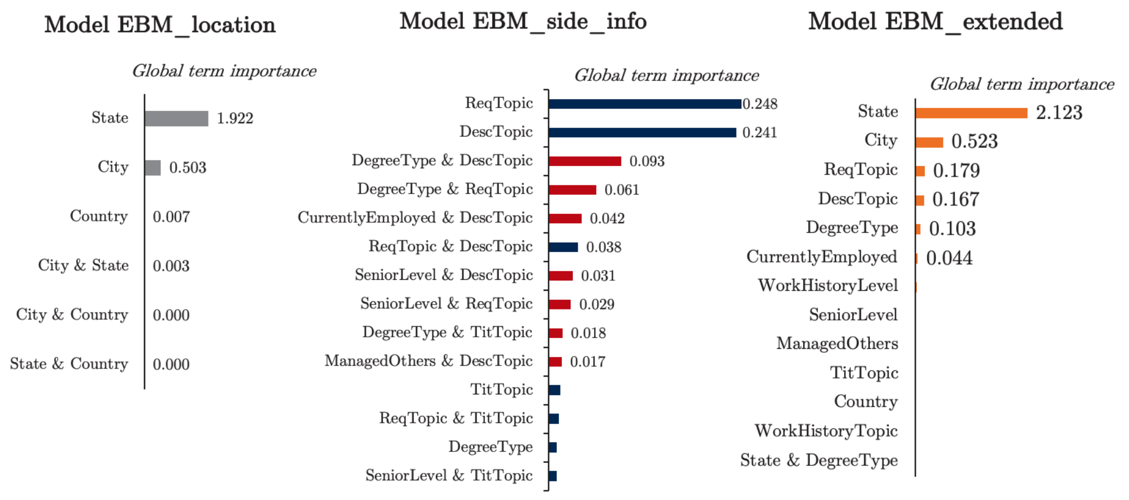


FIGURE 4.9: Global feature importance explanation by EBM models. Features with zero-term importance are included in the plot. Interaction between user profile features and job features are highlighted in red colors.

As can be seen from figure 4.9 when considering only location for ranking application, the matching state is more important than the matching city and matching country. This could be explained by the nature of the CareerBuilder dataset, where most of the applications were in the US.

The middle plot of figure 4.9 illustrates the important contribution of interaction between users and jobs' to the application ranking, when excluding location information. The interaction terms could be interpreted as a matching degree between the user profile and the job details. Some examples include users' education level and job description (DegreeType & DescTopic), education level and job requirement (DegreeType & ReqTopic), users' number of working experience and job requirement (SeniorLevel & ReqTopic), and number of working experience and job description (SeniorLevel & DescTopic)

4.5.3 Global explanation by model-specific approach: DPEBM models

Figure 4.10 presents the term importance for the whole white-box model for three variations of DPEBM models. The term importance is calculated similarly to those in EBM models.

As can be seen from these figures, there were no interactions between features that can be obtained from DPEBM models. When considering individual feature contributions to ranking applications, the patterns were similar to the observation in EBM models. Specifically, matching in State and City contributes more to the final ranking than side

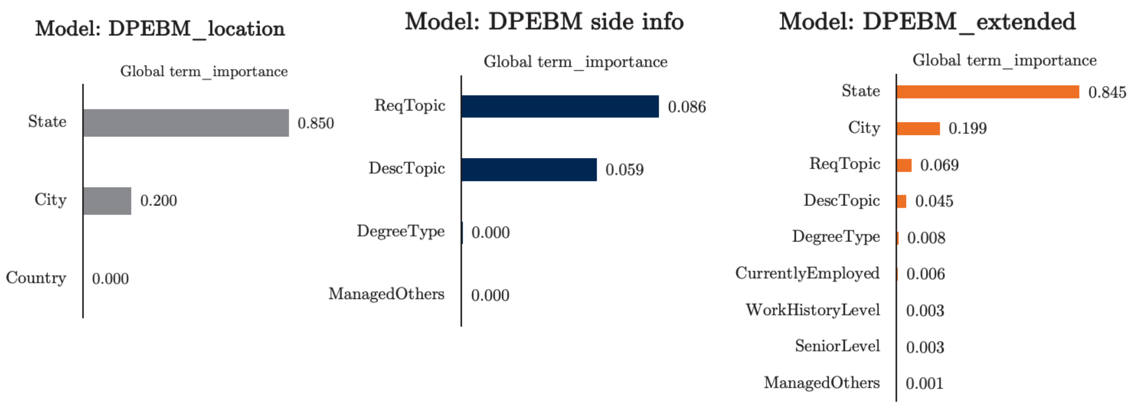


FIGURE 4.10: Global feature importance explanation by DPEBM models. Features with zero-term importance are excluded from the plot.

information (ReqTopic, DescTopic). Information from user profiles does not contribute much to the final ranking compared to other types of features.

4.5.4 Global self-explanation by white-box models and XGBoost

Figure 4.11 presents the top 20 feature importance for white-box models and the XGBoost model. The feature importance differentiates a lot for each model.

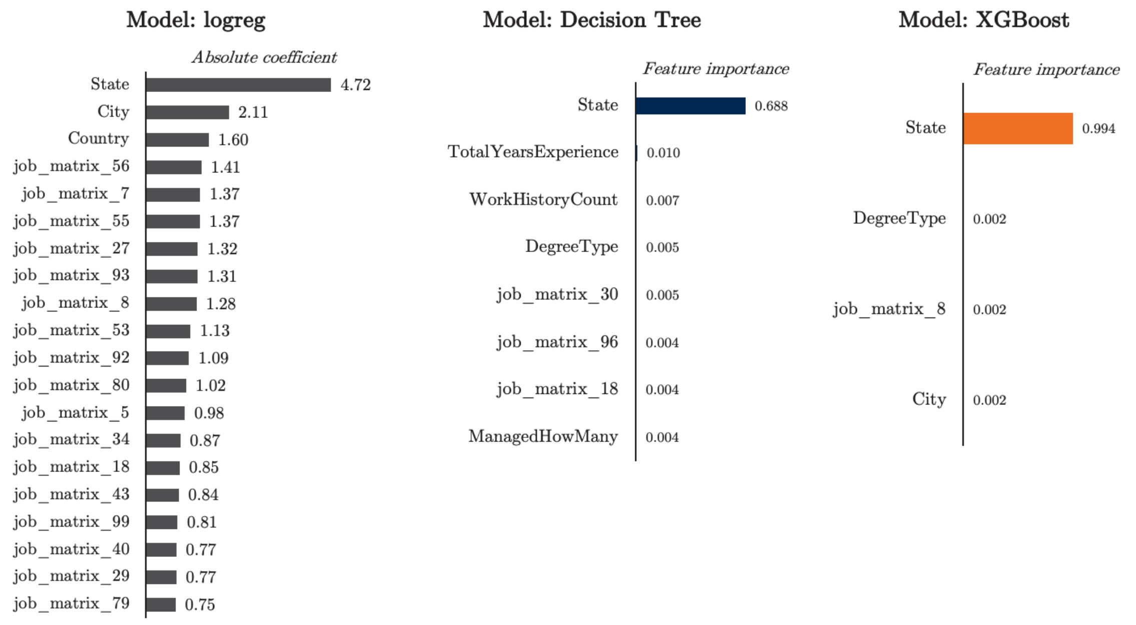


FIGURE 4.11: Global feature importance by white-box models and XGboost model. Features with zero-term importance are included in the plot.

For logistic regression, the level of importance is illustrated by the absolute value of the coefficients. Accordingly, location matching in State, City, and Country are the top three important features. The remaining features in the top 20 important features belong to the TF-IDF values of the terms in the job details.

Decision Tree determined ranking applications mostly based on whether applicant is in the same state with the job position. Although other features have much lower importance scores compared to State, the features about user profiles (TotalYearExperience, WorkHistoryCount, and DegreeType) are slightly more important than job details.

For XGBoost, the decision is mainly based on matching the state between users and jobs.

4.5.5 Kernel SHAP: Local feature importance

Kernel SHAP was implemented on 3 white-box models, 4 black-box models, and 6 explainable ensemble models in the same sample of the first 20 applications in the test application data for comparison. The current Kernel SHAP explainer in SHAP library **did not support** explaining directly factorization machine models.

For a given instance (x), the Kernel SHAP explainer calculates the expected model output, and the Shapley value of all the features that contribute to that output, according to equation 4.1. Shapley values can be either positive or negative. A positive Shapley value means that the feature pushes the prediction higher and a negative Shapley value means that the feature pushes the prediction lower.

$$E[f(x)] = \sum_{i=1}^{n} \text{ShapleyValue}(x_i) \tag{4.1}$$

where:

- $E[f(x)]$ represents the expected model output for a given instance x.
- n is the number of features in the instance x.
- $\text{ShapleyValue}(x_i)$ denotes the Shapley value for the i-th feature in the instance x.

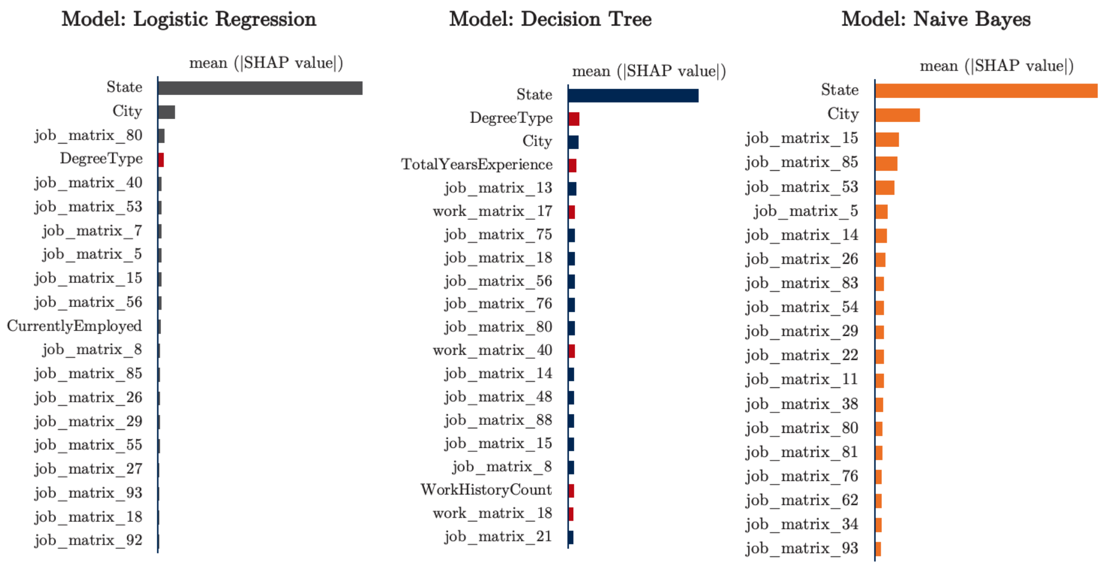


FIGURE 4.12: Local feature importance by Kernel SHAP for output of white-box models. The length of the bar indicates the magnitude of importance.
(Sample: 20 applications - Model: White-box)

For comparison of the consistency of local explanation without knowing the model classes (white-box or black-box or explainable ensemble), the sign of Shapley values was eliminated. In other words, the overall magnitude of the influence of the feature was calculated by taking an average of absolute Shapley values for each instance of the sample. The results are shown in the figures: 4.12 and 4.13

Kernel SHAP local explanation on white-box models

It can be seen from figure 4.12 that the importance of the dominant features is location matching (City or State) as they appear in the top three features. However, when considering local explanation in the Decision Tree model, user profile features appear more important than job details: DegreeType, TotalYearExperience, work_matrix features. DegreeType is also equally important as job_matrix_80 in Logistic Regression. For the Naive Bayes model, only location matching and job detail features appear in the top 20.

Kernel SHAP local explanation on black-box models

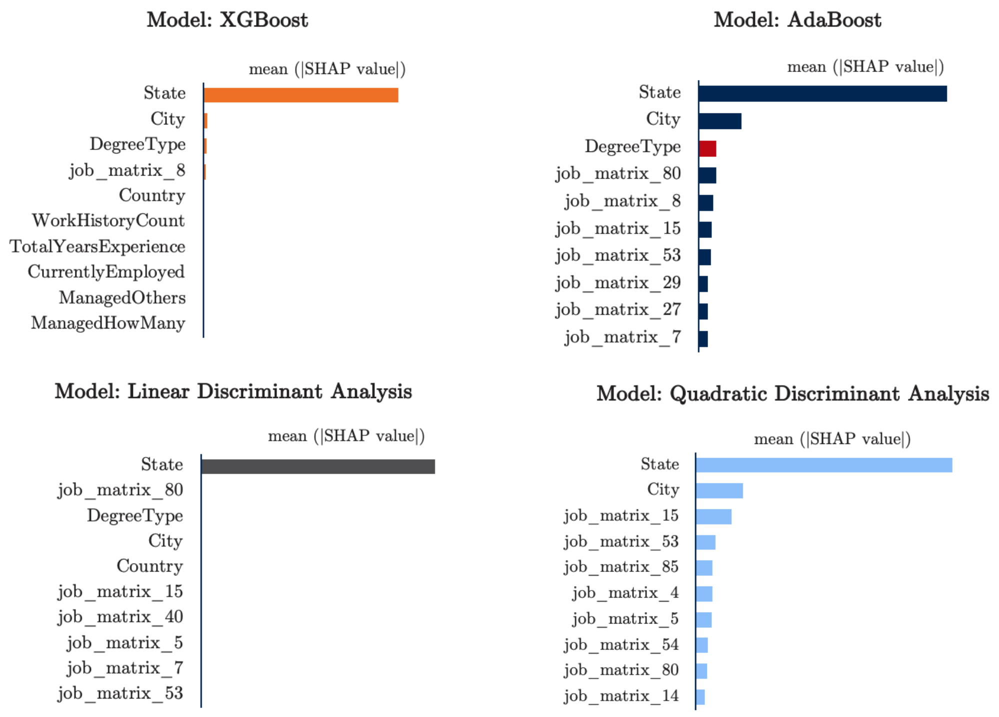


FIGURE 4.13: Local feature importance of top 10 features by Kernel SHAP. The length of the bar indicates the magnitude of importance.
(Sample: 20 applications - Model: Black-box)

For black-box models, matching the State between the user and job contributes the most to the final outcome of the application (apply or not apply). XGBoost and LnDA seem to depend on only location criteria to predict the outcome of applications. In the AdaBoost model, user education was among the top 3 important features. For the QDA model, job details ("job_matrix" features) are the second important factor group.

Kernel SHAP local explanation on explainable ensemble models

For the ensemble explainable model, we only compute Kernel SHAP for EBM models to check the consistency between local explanation by Kernel SHAP (post-hoc) versus the self-explanation by the EBM model.

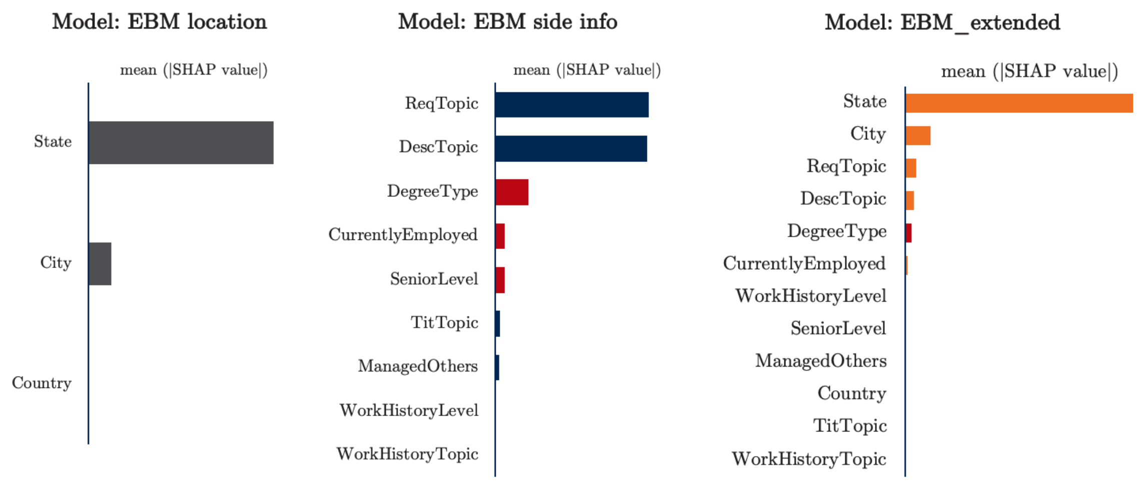


FIGURE 4.14: Local feature importance generated by Kernel SHAP. The length of the bar indicates the magnitude of importance. Important features of the user profile are highlighted in red colors.
(Sample: 20 applications - Model: EBM)

As can be seen from plot 4.14 local post hoc explanations by Kernel SHAP can capture only individual feature contribution, while the local self-explanation by the EBM model can capture both individual and feature interaction.

For a deeper analysis, I took the first job application in the test data and compared the explanation by EBM_side_info model and Kernel SHAP. The explanations are illustrated in Figure 4.15. This application has predicted label 1.

- In the top figure, the term importance score by the EBM model shows how features contributed to label 0 and label 1. Accordingly, DecsTopic and the interaction between CurrentlyEmployed & DescTopic and the interaction between SeniorLevel & ReqTopic are three contributing factors to positive labels. Meanwhile, the combination of SeniorLevel & DecsTopic and DegreeType & ReqTopic contributes more towards the negative label.

- In the bottom figure (Kernel SHAP), the Shapley values highlighted in red push higher predicted probability, or contribute to a positive label. The top three features of this type include DescTopic, DegreeType, and ReqTopic. Meanwhile, SeniorLevel (in blue) pushes the predicted probability to decrease (i.e. Not apply) with the strongest magnitude. It means that for this application, the SeniorLevel negatively influences on "Apply" decision.

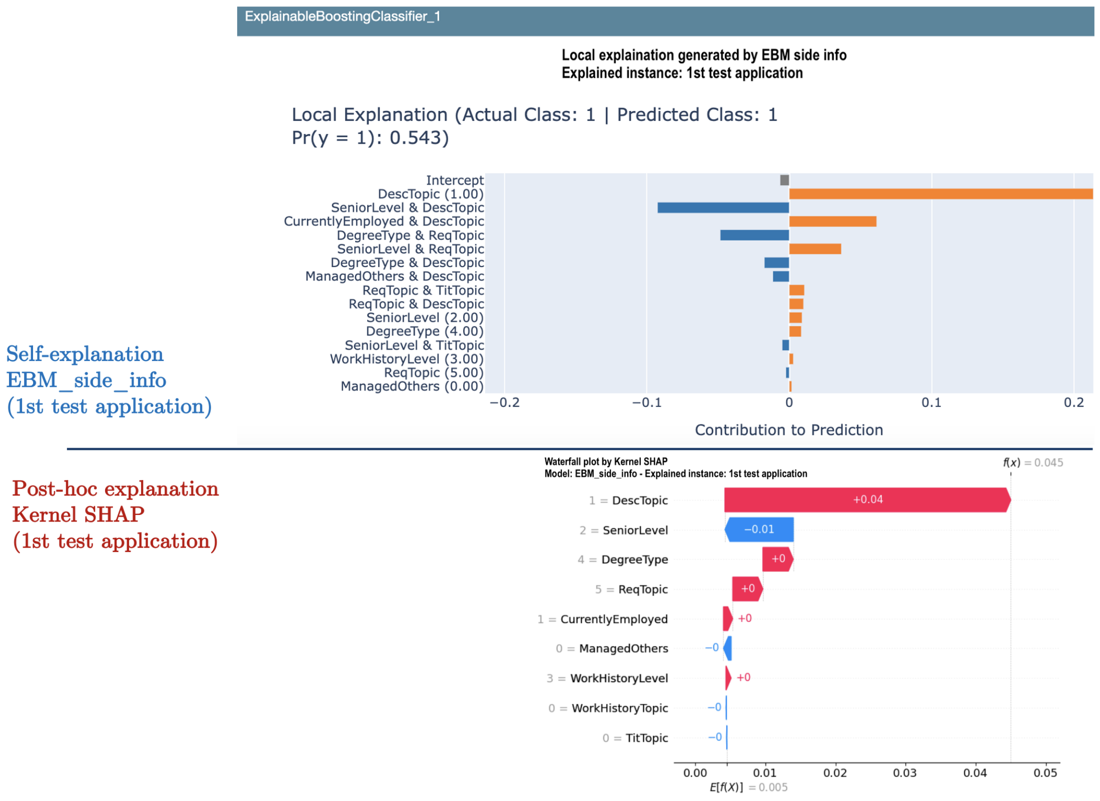


FIGURE 4.15: Comparison local explanation: Self-explanation generated by EBM_side_info model (top) and Post-hoc explanation generated by Kernel SHAP (bottom) on the same single job application.

4.5.6 LIME: Local feature importance

LIME local explanation on white-box models

Figure 4.16 shows the local explanation generated by LIME on 20 applications from white-box models. The top 20 important features are plotted based on the magnitude of feature contribution, measured by taking an average of the absolute value of feature weights across 20 applications in the sample.

Compared to the local explanation generated by Kernel SHAP for the same instance in figure 4.12, the LIME explanation put more favor on the user profile, highlighted in red bars. Interestingly, in LIME's explanation for Naive Bayes, all location-matching features (City/State/Country) are not presented.

LIME local explanation on black-box models

LIME's explanation of black-box models (as shown in figure 4.17) shares the same pattern as Kernel SHAP's explanations (4.13). The dominant feature is matching State and City.

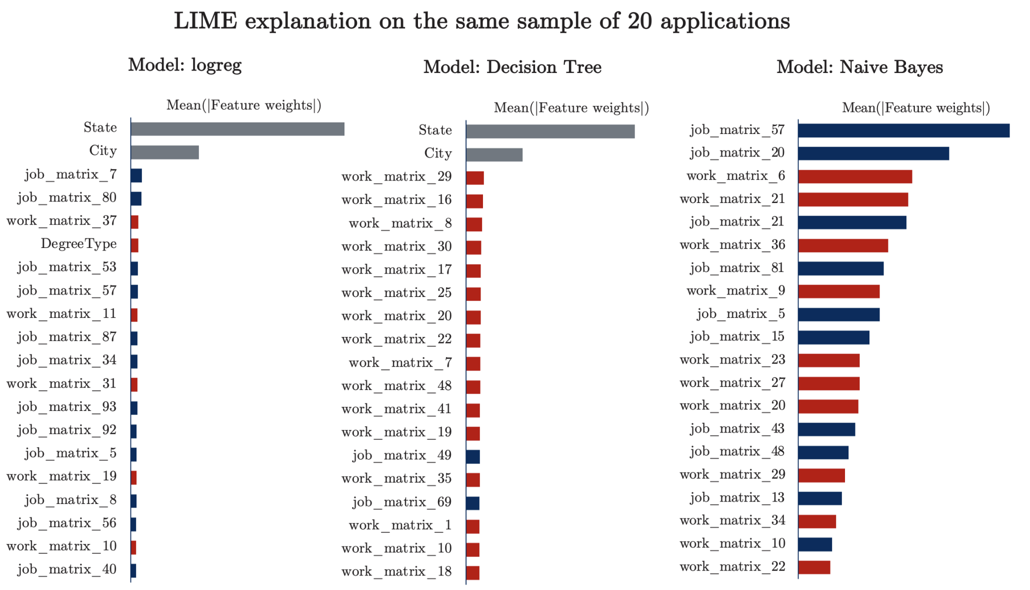


FIGURE 4.16: Local feature importance of top 20 features by LIME. The length of the bar indicates the magnitude of importance. Features relating to user profiles are highlighted in red color.
(Sample: 20 applications - Model: White-box)

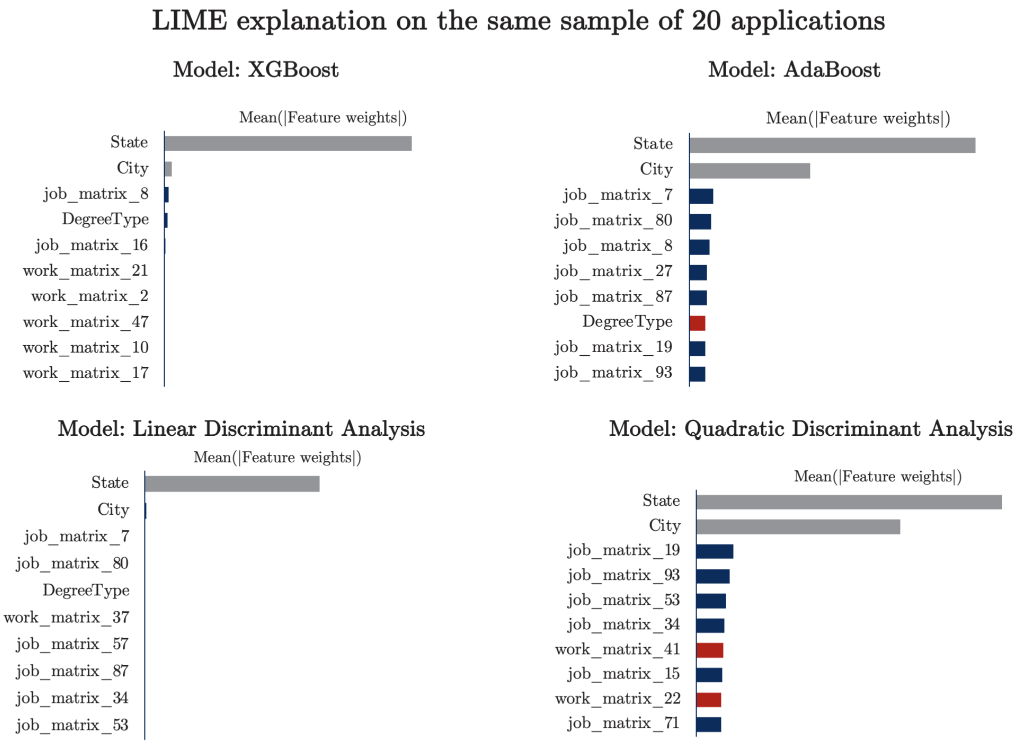


FIGURE 4.17: Local feature importance of top 10 features by LIME. The length of the bar indicates the magnitude of importance. Features relating to user profiles are highlighted in red color.
(Sample: 20 applications - Model: Black-box)

LIME local explanation on explainable ensemble models

Similarly to Kernel SHAP, LIME could not capture the interaction between features. The magnitude of individual feature contribution for three variations of EBM models is shown in figure 4.18.

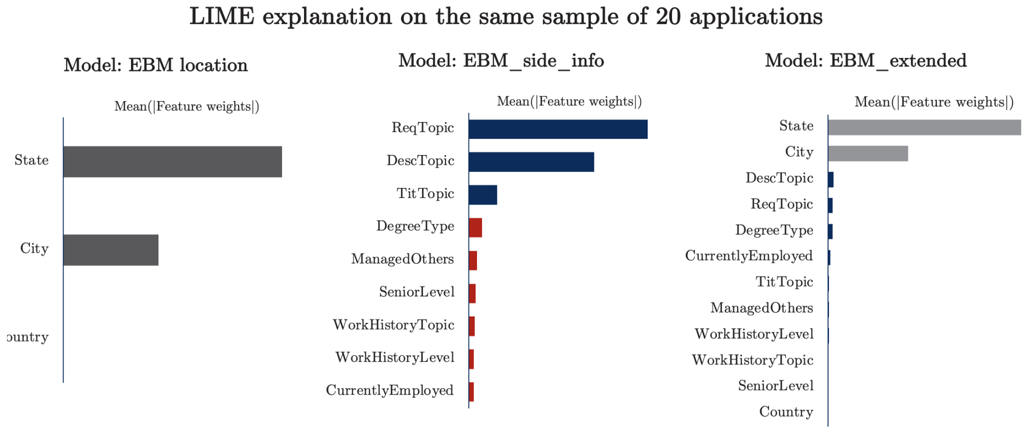
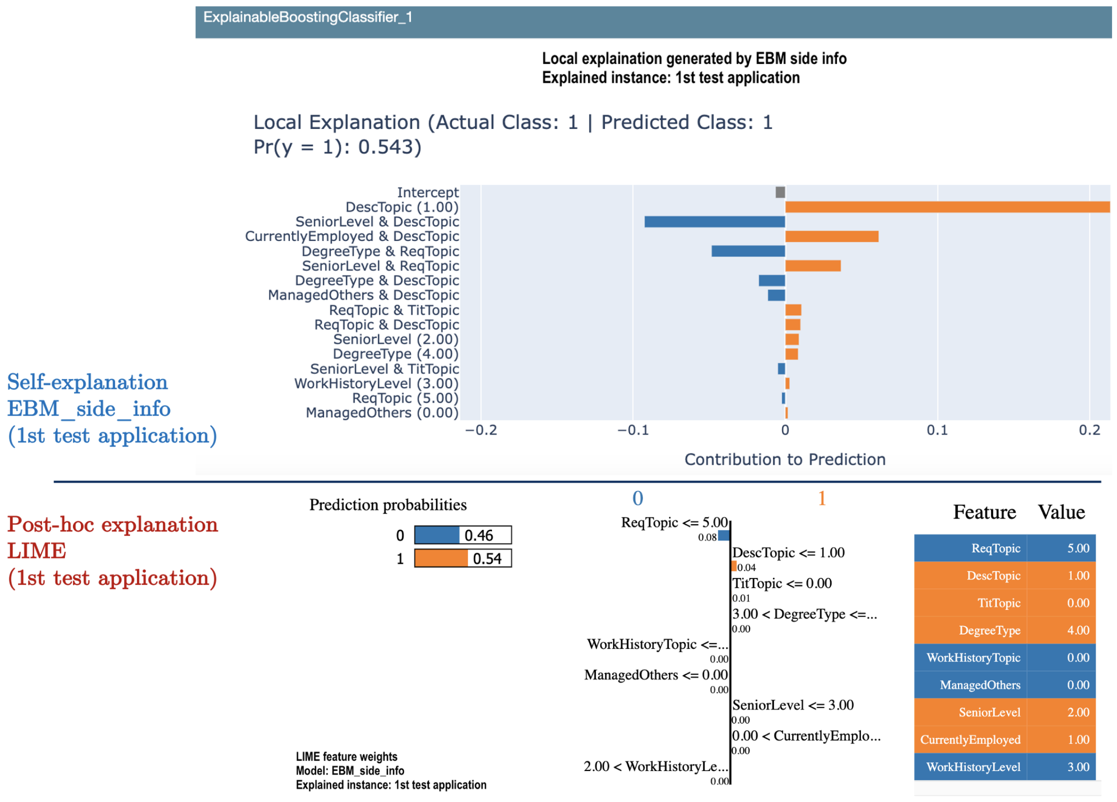


FIGURE 4.18: Local feature importance features by LIME. The length of the bar indicates the magnitude of importance. Features relating to user profiles are highlighted in red color.
(Sample: 20 applications - Model: Explainable ensemble)

Comparison between post-hoc local explanation and local self-explanation

Figure 4.19 compares the local self-explanation using the EBM_side_info model versus the post hoc explanation using LIME. This application was made represented by (UserID=13, JobID = 821691).

Although both explanations make the same decision (class 1, with probability 54%), the LIME explanation (bottom figure) ignores the contribution of ReqTopic on the positive label. ReqTopic is ranked as the most important topic that contributes to a negative label. Meanwhile, in the self-explanation by the EBM side info model (top figure), ReqTopic is placed at the bottom in terms of individual contribution towards a negative label. This is because the interaction of ReqTopic and other features of the user profile, such as SeniorLevel & ReqTopic have a stronger magnitude towards the positive label.
This example again highlights the necessity of interaction effects between features to explain the ranking result of an application.



Feature	Value
ReqTopic	5.00
DescTopic	1.00
TitTopic	0.00
DegreeType	4.00
WorkHistoryTopic	0.00
ManagedOthers	0.00
SeniorLevel	2.00
CurrentlyEmployed	1.00
WorkHistoryLevel	3.00

FIGURE 4.19: Comparison local explanation: Self-explanation generated by EBM_side_info model (top) and post-hoc explanation generated by LIME (bottom) on the same single job application.

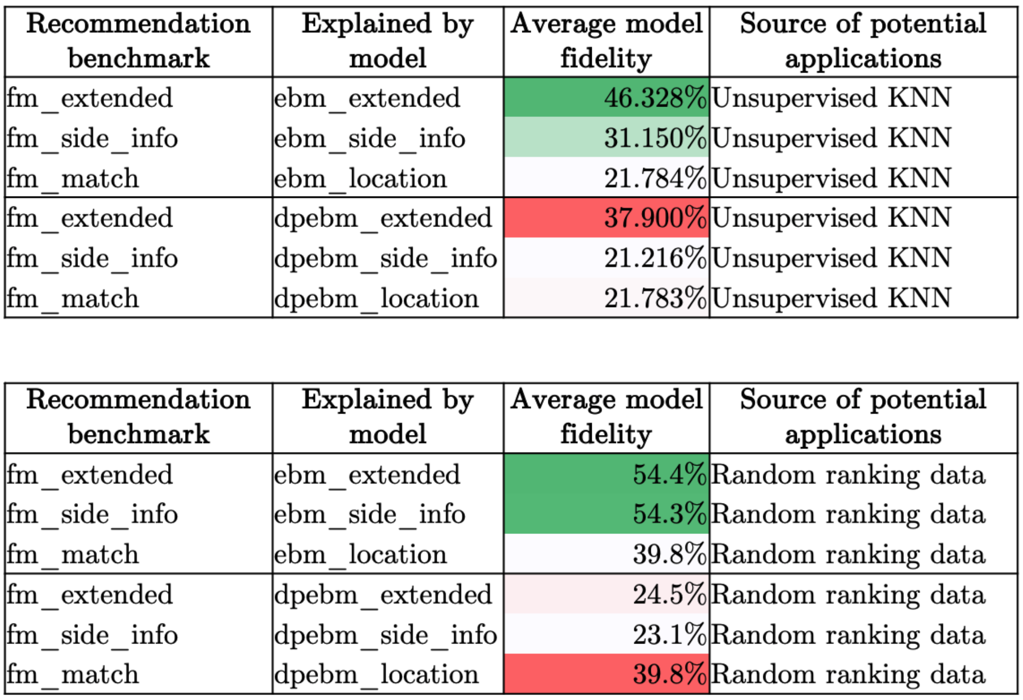
4.6 Evaluation explanation - RQ3

This section evaluates the utility of the explanation using two metrics: model fidelity rate and feature importance fidelity rate. The findings answer **Research question 3: How to evaluate the explanations generated by different XAI techniques?**

4.6.1 Model fidelity rate - Global explanation

The model fidelity was computed on JRS using EBM and DPEBM as ranking algorithms. The fidelity rate of the model was calculated on each test user, using the algorithm 2, and then taking the average fidelity of all test users for evaluation. The purpose of this evaluation is to measure how well explainable models can retrieve recommended jobs using the benchmark recommended jobs from hybrid FM models.

Figure 4.20 shows the results of the model fidelity for two groups of explainable ensemble models: EBM and DPEBM. A high model fidelity rate implies that the explainable ensemble model can be used for recommendation and at the same time reduce the complexity in matrix factorization.



Recommendation benchmark	Explained by model	Average model fidelity	Source of potential applications
fm_extended	ebm_extended	46.328%	Unsupervised KNN
fm_side_info	ebm_side_info	31.150%	Unsupervised KNN
fm_match	ebm_location	21.784%	Unsupervised KNN
fm_extended	dpebm_extended	37.900%	Unsupervised KNN
fm_side_info	dpebm_side_info	21.216%	Unsupervised KNN
fm_match	dpebm_location	21.783%	Unsupervised KNN

Recommendation benchmark	Explained by model	Average model fidelity	Source of potential applications
fm_extended	ebm_extended	54.4%	Random ranking data
fm_side_info	ebm_side_info	54.3%	Random ranking data
fm_match	ebm_location	39.8%	Random ranking data
fm_extended	dpebm_extended	24.5%	Random ranking data
fm_side_info	dpebm_side_info	23.1%	Random ranking data
fm_match	dpebm_location	39.8%	Random ranking data

FIGURE 4.20: Average model fidelity rate on explainable ensemble models

In general, explainable ensemble models can only retrieve a maximum of half of the jobs recommended by hybrid factorization models. EBM performed better than the DPEBM models. In addition, the source of potential applications does not greatly influence the fidelity of the model of the EBM models. EBM_extended models perform best as they use all available information.

However, for DPEBM models the ability to retrieve close results as black-box models depends on side information and the source of potential applications. Specifically, DPEBM models using unsupervised KNN applications show a preference for using all sides of the information. On the contrary, DPEBM models that use random ranking applications only need location information to achieve the same performance.

4.6.2 Feature importance fidelity rate - Local explanation

First, a maximum of 20 important features from self-explainable models: Logistic Regression, Decision Tree, and three variations of EBM models with nonzero feature importance. These are called gold features.

Second, from the local explanation by LIME and Kernel SHAP, the top 20 features were extracted based on the magnitude of the contribution. The fraction of gold features that appeared in local features was used for the feature-important fidelity rate. For example, if a City appears in both the gold feature and local feature set (regardless of its rank), it is counted in the fidelity set.

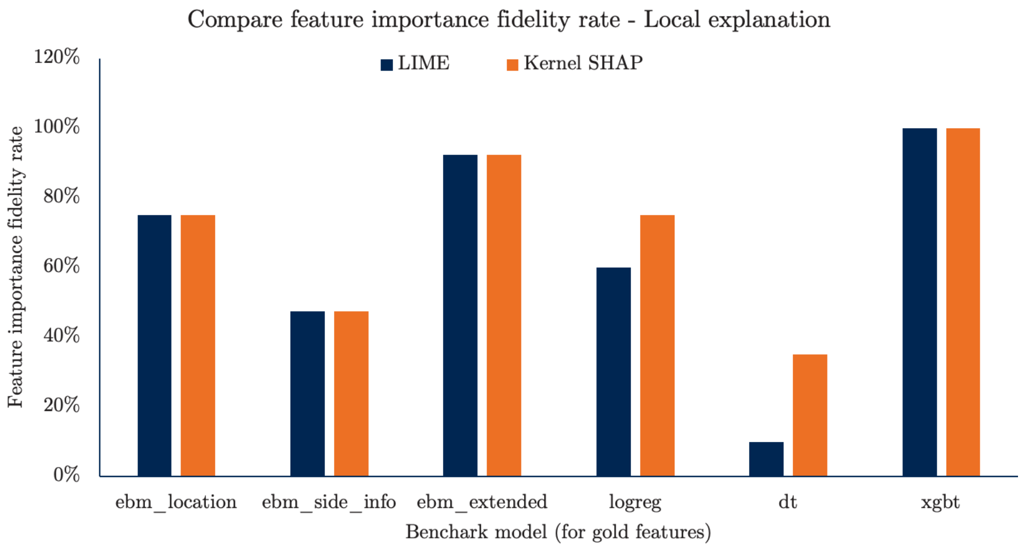


FIGURE 4.21: Figure importance fidelity rate for two local explanations by two techniques: LIME and Kernel SHAP. Detailed results are available at appendix F.2

Figure 4.21 shows a comparison between LIME and Kernel SHAP using the same set of gold features as a benchmark. Accordingly, there is no difference between LIME and Kernel SHAP feature fidelity rates when benchmarking the gold features by three EBM models and by the XGBoost model. However, when using the gold features from white-box models (Logistic Regression and Decision Tree), Kernel SHAP performs better in terms of retaining the most important features.

4.7 Generating human-digestible explanation - RQ4

4.7.1 Summary experiment

The last part of the experiment approach aims to answer **research question 4: How to make explanations digestible for lay users?**. The "Post-explanation" experiment focuses on communicating the explanations from different model interpretation techniques (as in section 4.5) to the target audience in the most human-digestible way. The target audience in this case is lay users who have limited knowledge of machine learning. Using a human-computer interaction approach and user experience approach, the cognitive process of understanding a job recommendation is composed of several elements.

1. Why does job A have a higher ranking than job B in the list of recommended items?
2. What factors lead to the higher ranking of the job?

First, the probability of the predicted class (i.e., score) can serve as the first explanation element. For example, the explanation can be presented in the template "This job matches [score]% to your profile." Job A has a higher probability of being applied by the user than job B means that it matches the user better.

Second, the raw explanations in section 4.5 are feature names and their contribution to the outcome, so they could answer the second question. The challenge, however, lies in feature names as they lack semantics due to the feature engineering process. To gain semantics, we could use either visualization or text explanation, depending on how text features are embedded. Specifically:

For feature names generated by JRS in figure 3.2, the TF-IDF column index of the term can be extracted from feature names as they follow the template "job_matrix_[idx]" or "word_matrix_[idx]". The document index is the corpus that will be the row index in the TF-IDF matrix. TF-IDF values could be converted into raw terms in the original document, using the pseudocode 4. For local post hoc explanations, using text features instead of embedded TF-IDF features may provide more context to the users.

Algorithm 4 Convert TF-IDF feature names to raw text: apply to 1 job application

```
Input: UserID, JobID
local explainer (Kernel SHAP/ LIME)
local explanation
corpus (job details, user work history)
Output: converted features
1 $top\_features \leftarrow$ ExtractTopFeatures($local\_explanation, n = 20$)
2 $top\_matrix\_features \leftarrow$ ExtractTFIDFFeatures($top\_features, corpus$)
3 $matrix\_index\_list \leftarrow$ GetMatrixColumnIndices($top\_matrix\_features$)
4 $doc\_index \leftarrow$ GetDocIndex($JobID$ or $UserID, corpus$)
5 $converted\_matrix\_terms \leftarrow$ InverseTransformTFIDF($top\_matrix\_features, corpus,$
    $matrix\_index\_list, doc\_index$)
6 $converted\_features \leftarrow$ ReplaceFeatures($local\_explanation, top\_features,$
    $converted\_matrix\_terms$)
7 return $converted\_features$
```

Features name generated by JRS in 3.3, the LDA topic contribution or keywords can provide an intuitive explanation to the user, especially when combined with visualization.

Words cloud, for example, helps users capture the theme of each category. This can be an industry or a discipline of employment. The word clouds of all LDA models are available in appendix C.2.
Keywords contributing to a dominant topic could be extracted, using pseudocode 5. This pseudocode is based on the assumption that keywords are the words in the document that have the highest probability under that dominant topic.

Algorithm 5 Extract keywords contributing to the dominant topic of a given document - Standard LDA

```
Input: $lda\_model, feature\_names$ : the pre-trained LDA model and feature names
$dtm$ : document-term-matrix
$doc\_idx$ : document index in the corpus
$topic\_idx$ : dominant topic index
$top\_n$ : number of top words
Output: $top\_words$
1 $topic\_word\_distributions \leftarrow$ ExtractTopicWordDistributions($lda\_model, dtm$)
2 $doc\_topic\_distributions \leftarrow$ ExtractDocTopicDistributions($lda\_model, dtm$)
3 $doc\_word\_indices \leftarrow$ ExtractWordsFromDocument($dtm, doc\_idx$)
4 $doc\_words \leftarrow$ [feature_names[idx] for idx in doc_word_indices]
5 $word\_probs \leftarrow$ CalculateWordProbabilities($topic\_word\_distributions$,
   $doc\_words$,
   $doc\_word\_indices$,
   $topic\_idx$)
6 $sorted\_words \leftarrow$ SortWordsByProbabilities($word\_probs$)
7 $top\_words \leftarrow$ GetTopNWords($sorted\_words, top\_n$)
8 return $top\_words$
```

The next two sections present two use cases for the illustration post-explanation process.

4.7.2 Usecase Post-explanation: extract raw term from TF-IDF features

This use case focuses on interpreting post hoc local explanations for JRS using TF-IDF features. The use case includes 20 recommended jobs for UserID 577106 using a logistic regression ranking model. The full result of the recommendation is available in Table G.1. This user has an education background in Computer Science and has previously worked as a "Help Desk Analyst" role.

The first recommended result is JobID 355928, "Technical Support Specialist" with the positive predicted label. For the full job description and job requirements, see Appendix G.

First, feature information is retrieved based on the UserID and JobID of the recommendation, including location matching, user profile, job details (TF-IDF), and user work history (TF-IDF). Local explainers (LIME and Kernel SHAP) use this information to predict their own labels and compute feature contributions to those outcomes. The local explanation is shown to the left of two figures 4.22 and 4.23. The original model had 159 features, but in the plot, only the top features are presented.

Interpretation of local explanation by Kernel SHAP

The final probability of application is 0.969 which means that the label predicted by the SHAP explainer is positive (label 1). This is the same result as the label predicted by the original JRS.

After extracting the main features and converting the name of the features to raw terms, using the algorithm 4, the local explanation is re-plotted using the converted text features, as shown on the right side of two figures 4.22 and 4.23. Accordingly, most of the important features come from the job details ("job_matrix_[index]"). For this application, keywords such as "degree, plus, excellent, computer, high, clients" have positive Shapley values (highlighted in red). These terms also have non-zero TF-IDF values. Thus, the audience can interpret that job detail with these terms push the probability of applying for the job to increase. The term "technical" has negative Shapley values (highlighted in blue). This means that the term reduces the probability that users will apply for the job. In terms of location matching, "State", "Country", and "City" all appear in the top features. However, their influence is different. Matching in state contributes positively, matching in country contributes negatively to the probability of application, while mismatching in city does not influence the final outcome of the application.

Interpretation raw explanation by LIME

For the same use case, the local interpretation of LIME separates two labels by two colors and plots the feature that contributes to each label, as shown in Figure 4.23. The length of the bar indicates the importance of the feature. Accordingly, the final result generated by the LIME explainer is also positive: the probability of label 1 is 0.97. Among the top 20 features plotted, the most influencing features of this decision are more about location matching than job detail. Matching in "State" is more important than not matching in "City" in driving the user to apply for the job. Although many job-related terms appear in the top features supporting positive labels, only those with nonzero values (>0) actually appear in the job description: "office, technical, excellence, design, and company". The user profile feature, such as "DegreeType", is also among the top 10 important features to support users applying to the job.

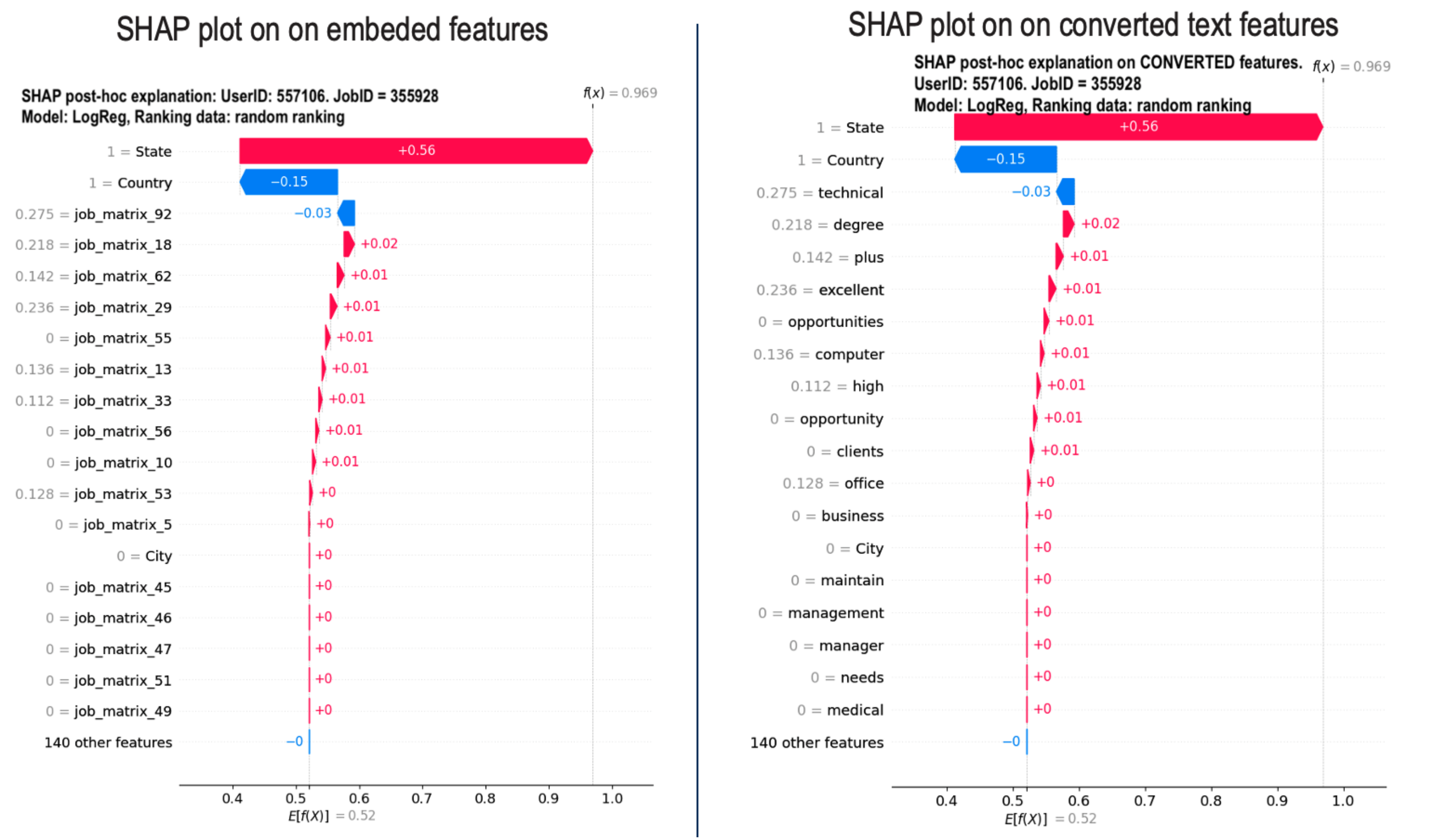


FIGURE 4.22: Use case: UserID = 577106, JobID = 355928. Compare the SHAP explanation before and after converting feature names. Left: Raw explanation, right: Figure the converted feature names.

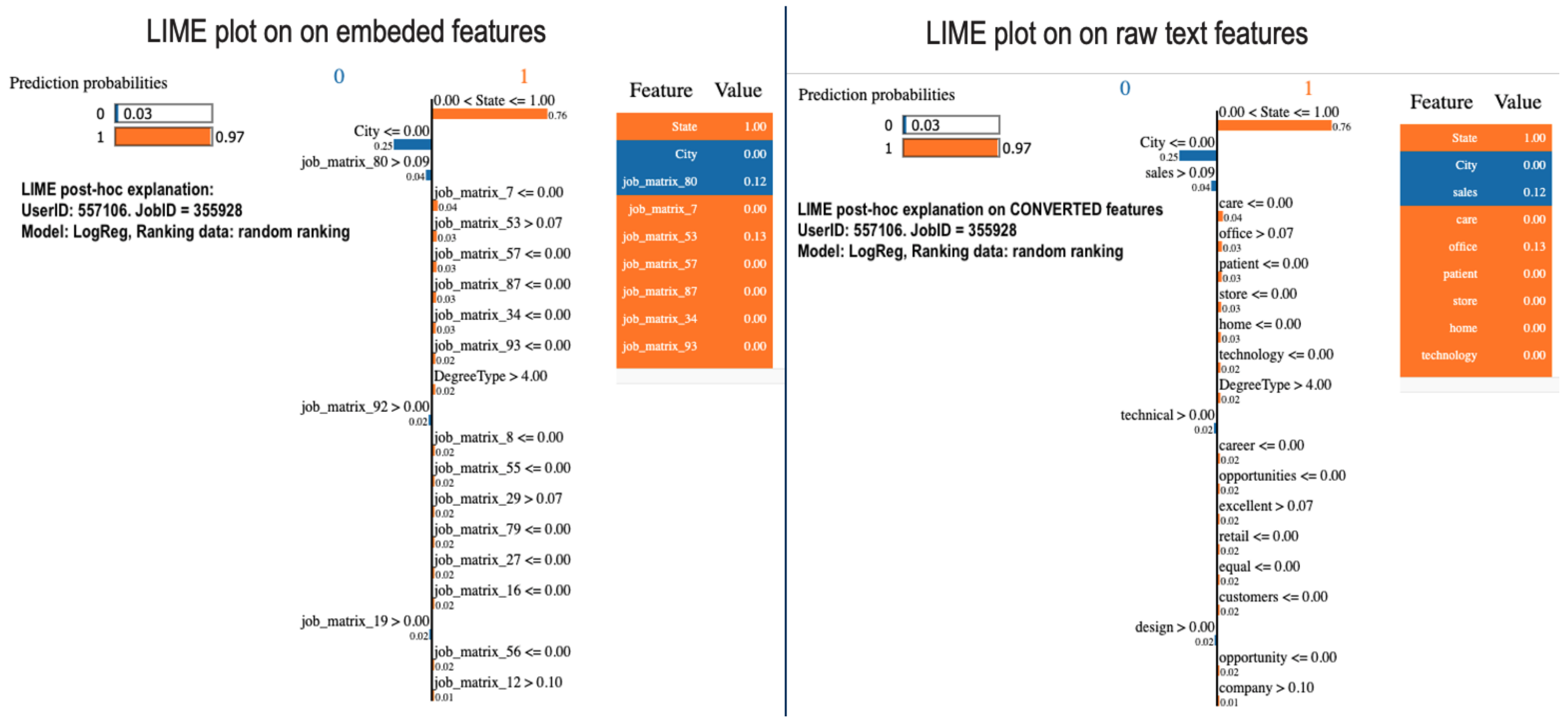


FIGURE 4.23: Use case: UserID = 577106, JobID = 355928. Compare the LIME explanation before and after converting the feature names.: Raw explanation, right: Figure the converted feature names.

4.7.3 Usecase Post-explanation: LDA topic contribution visualization

This use case focuses on interpreting self-local explanations using the EBM ranking model. The local explanation sample includes 20 recommended jobs for the user with UserID = 577106. Only categorical features were used to generate this result, using the EBM_side_info model. The full result of the recommendation is available in table G.2. This user has an education background in Computer Science and has previously worked as a "Help Desk Analyst" role.

The first job recommended for this user is JobID = 92240, Outside Sales Representative - B2B. This application has a positively predicted label by the EBM model and will be explained in more detail after computing the contribution of the feature.

First, complete feature information was retrieved based on the UserID and JobID of the recommendation, including user profile, job details (TitTopic, ReqTopic, DescTopic), and user work history (WorkHistoryTopic) (EBM_side_info does not use location matching information). EBM_side_info will use this information to predict labels but in a **local** manner. Figure 4.24 shows a local explanation of this application. Actual (1) is the label predicted by the recommendation system, and predicted (1) is the label predicted by the local EBM model. In fact, the user applied for this job based on the original data.

In the feature importance plot, both individual features and interaction between features are plotted. The left side represents features supporting negative labels (DegreeType & ReqTopic, ManagedOthers & DescTopic, CurrentlyEmployed & DescTopic, SeniorLevel ReqTopic) and the right side represents figures supporting positive labels (DescTopic, ReqTopic, DegreeType & DescTopic, etc.). The interaction between user profile features and job details features is worth investigating to understand whether the user is likely to apply for the job (based on a local explanation).

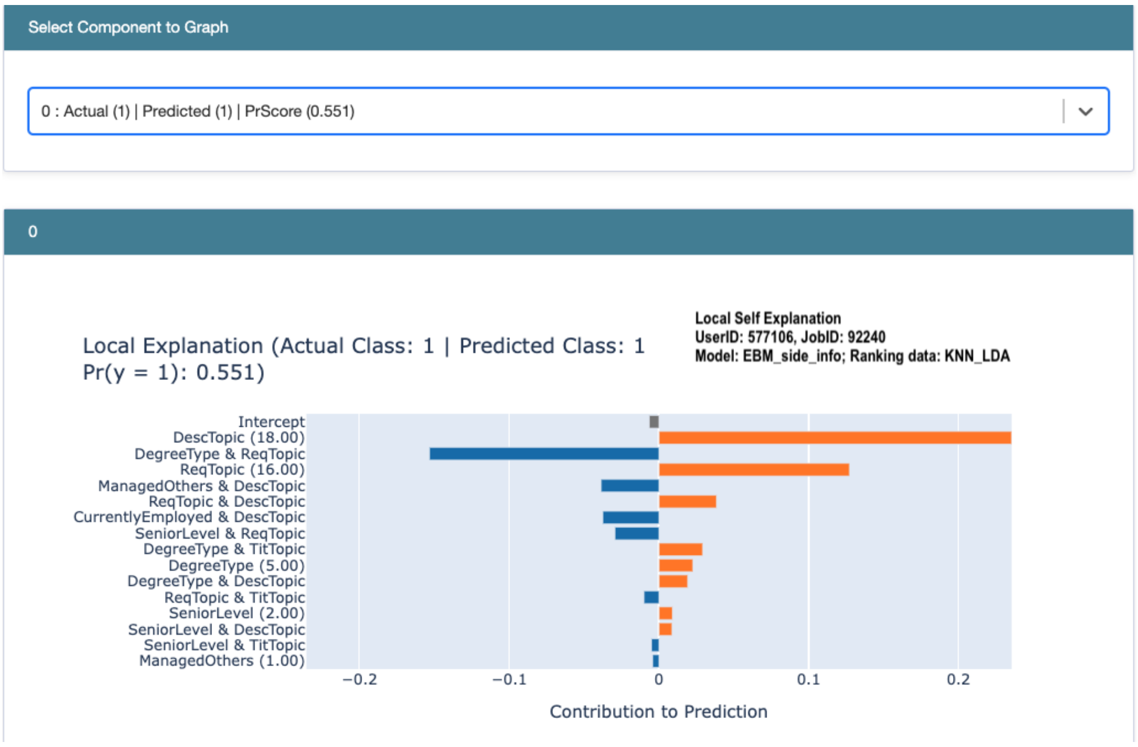


FIGURE 4.24: Use-case "post-explanation": UserID = 577106, JobID = 92240. EBM_side_info. Local self-explanation

The first step to transform the above explanation into understandable words is to "decode" the value of the features, as all features have been discretized during feature engineering. Table 4.1 shows the interpretation of this case. LDA topic modeling features (WorkTopicHistory, ReqTopic, DescTopic, and TitTopic) represent the dominant topic model for the given user and job. They need further interpretation.

Features name	**Discretized value**	**Intepretation**
City	0	Not match between User & Job
State	0	Not match between User & Job
Country	1	Match between User & Job
DegreeType	5	Master's level
CurrentlyEmployed	1	Yes
ManagedOthers	1	Yes
WorkHistoryTopic	0	Required further explanation
WorkHistoryLevel	1	1-2 previous jobs
SeniorLevel	2	3-5 years working experience
ReqTopic	16	Required further explanation
DescTopic	18	Required further explanation
TitTopic	0	Required further explanation

Table 4.1: Use-case: Generate narrative on EBM_side_info local explanation. Feature interpretation.

The audience can capture the theme of each dominant topic by using word clouds. Figure 4.25 shows the keywords in the dominant topic for the job title (Topic 0), the job requirement (Topic 16), the job description (Topic 18), and the user's work history (Topic 0). These keywords may not appear in the given application, but they give the user an overview of the theme of each topic.

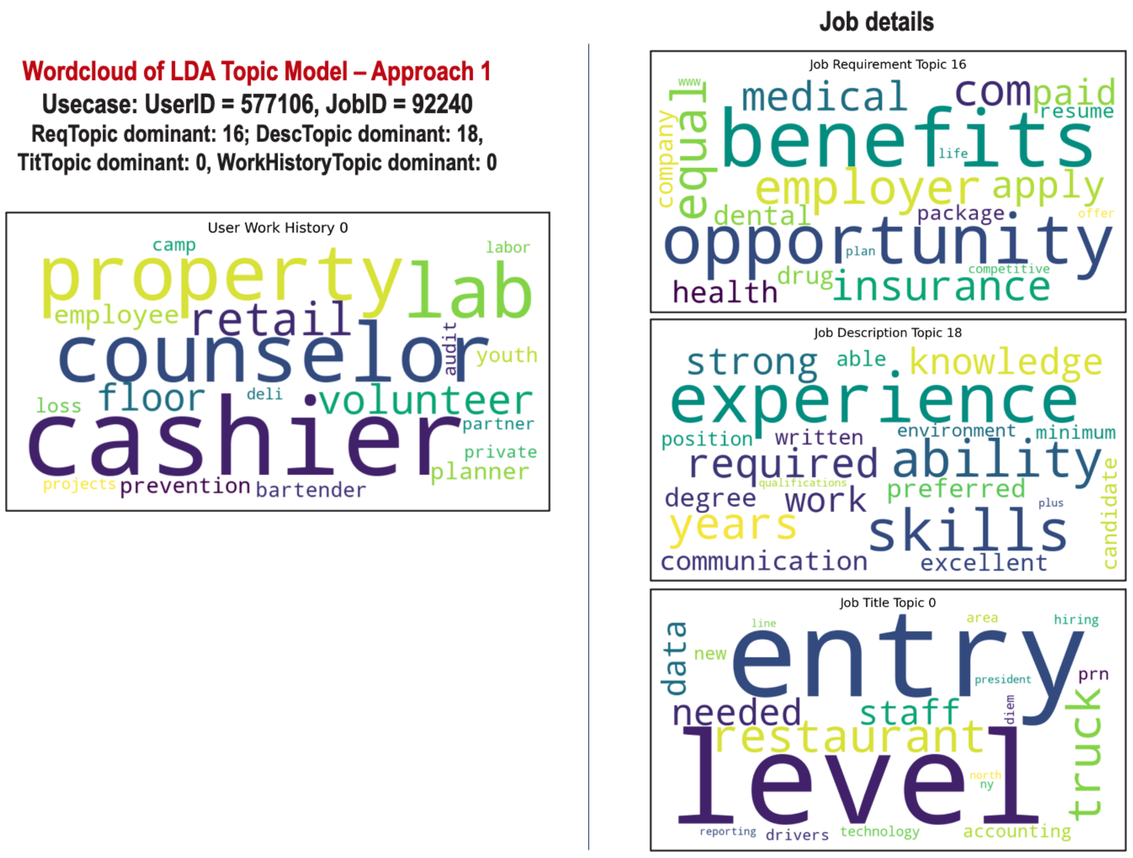

Wordcloud of LDA Topic Model – Approach 1
Usecase: UserID = 577106, JobID = 92240
ReqTopic dominant: 16; DescTopic dominant: 18,
TitTopic dominant: 0, WorkHistoryTopic dominant: 0
User Work History 0
property lab counselor cashier retail floor volunteer employee camp labor audit youth loss deli partner private planner projects prevention bartender
Job details
Job Requirement Topic 16
medical com paid resume benefits equal company employer apply dental package offer opportunity plan drug health insurance competitive life www
Job Description Topic 18
strong able knowledge experience position written environment minimum required ability degree work preferred qualifications plus years skills communication excellent candidate
Job Title Topic 0
data line new area hiring entry president prn needed staff diem restaurant truck level north ny reporting drivers technology accounting

FIGURE 4.25: Use-case: Generate narrative on EBM_side_info local explanation. Wordcloud of dominant topic calculated by Approach 1. Wordcloud after re-calculating dominant topics by approach 2 is available in the appendix G.1

Based on the LDA methodology, all topics (20 topics) are assigned to each document. However, only the dominant topic was used in the training model, thus excluding the 19 remaining topics. It is noted that the method to compute the dominant topic in feature engineering (by Lacic et al. 2020 [11]) focuses on the overlap between words in the document and words in each topic to determine the dominant topic. This approach does not make use of the probability distribution of words in each topic by the standard LDA model. The latter approach would result in more consistency and the dominant topic would better capture the user's previous job titles. I also recalculated the dominant topics and compared the results. See the Appendix G.1 for more details.

- Approach 1: Based on word-topic distribution, adapt from Lacic et al. 2020 [11]
- Approach 2: Based on standard LDA topic-document distribution.

Figure 4.26 shows the contribution of the topic by two approaches. The plot of Approach 1 (left side) shows the dominant topics for Job Requirements and for Job Descriptions because of nonzero values in topic contribution. Zero values for Job Title and Work History may be explained by no overlap in words between the document and the topic. This is the limitation of Approach 1.
On the right side of the figure, the contribution of the topic of Approach 2 shows the full results for 4 types of documents. The dominant topics are different from those computed in Approach 1, for example, ReqTopic (7 vs. 18), TitTopic (3 vs. 0), and WorkHistory-Topic (13 vs. 0). Using this plot, the audience may notice a similar contribution between

Topic 11 and Topic 7 for Job Description and check the word cloud or extract keywords contributing to this topic for further investigation.

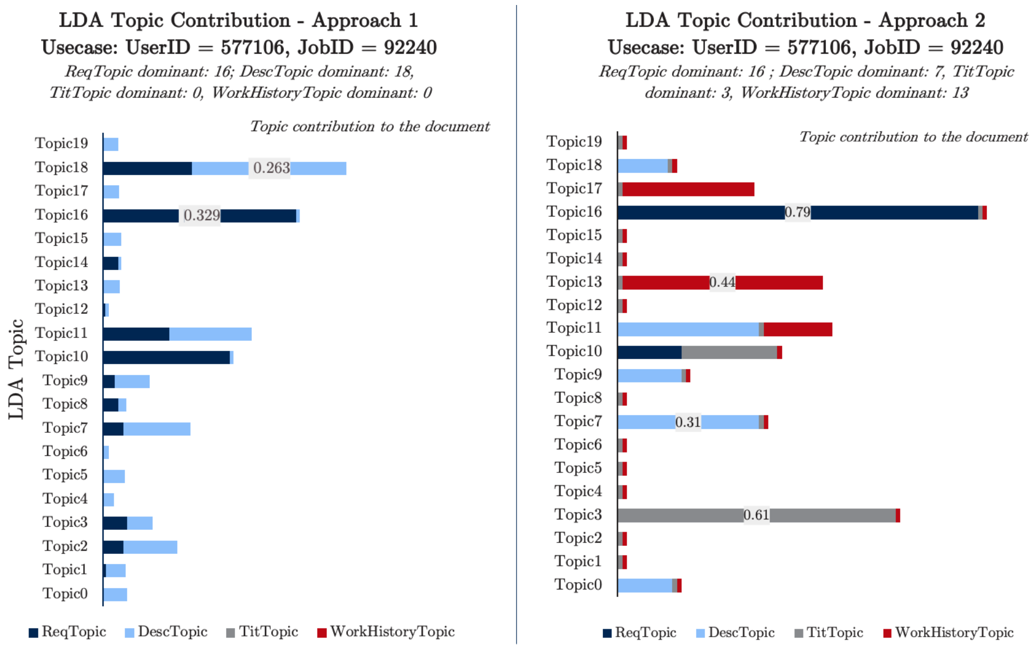
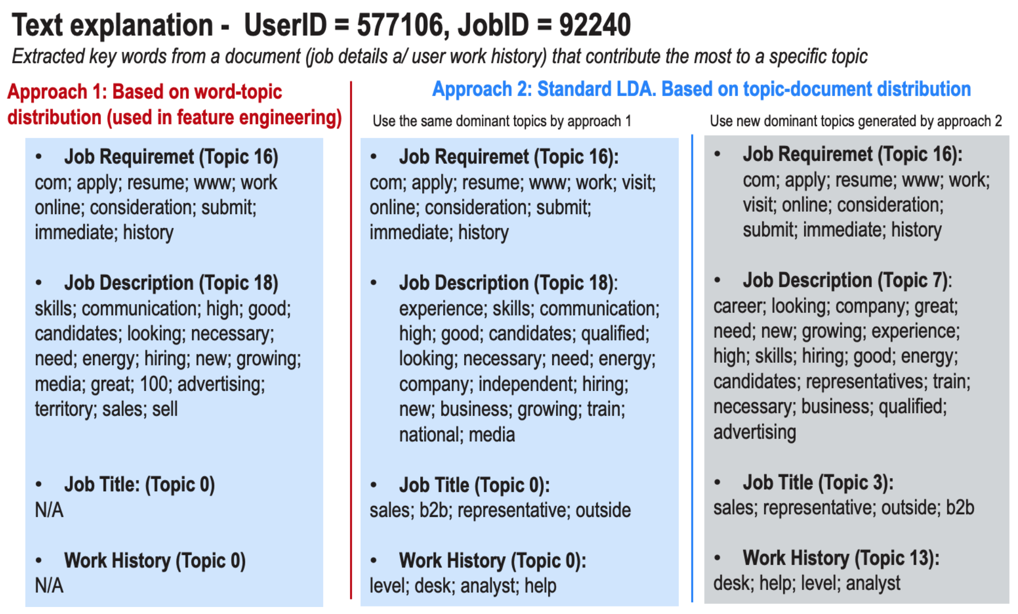


FIGURE 4.26: Use-case: Generate narrative on EBM_side_info local explanation. Compare topic contribution calculation by two approaches.

Similarly to computing the topic contribution, there are two methods of extracting keywords from a document contributing to a specific topic. The algorithm 5 adapts approach 2, using the probability property of LDA. This approach is more consistent compared to using only the presence of the word (Approach 1).
Figure 4.27 shows the results of the keyword extraction using two approaches. Under approach 1, the words in the Title and Work History in this use case do not overlap with keywords from the feature name of LDA models. As a result, no keywords can be extracted.

Keyword extraction using approach 2 can be done on the original dominant topics (calculated by approach 1), or on the new dominant topics recalculated by approach 2. The results are quite similar regardless of how to generate dominant topics. The extracted keywords still need a further filter for non-meaningful terms, such as "com, www" or general terms such as "company, job, history,..."



Text explanation - UserID = 577106, JobID = 92240

Extracted key words from a document (job details a/ user work history) that contribute the most to a specific topic

Approach 1: Based on word-topic distribution (used in feature engineering)

- **Job Requiremet (Topic 16)**
com; apply; resume; www; work online; consideration; submit; immediate; history
- **Job Description (Topic 18)**
skills; communication; high; good; candidates; looking; necessary; need; energy; hiring; new; growing; media; great; 100; advertising; territory; sales; sell
- **Job Title: (Topic 0)**
N/A
- **Work History (Topic 0)**
N/A

Approach 2: Standard LDA. Based on topic-document distribution

Use the same dominant topics by approach 1

- **Job Requiremet (Topic 16):**
com; apply; resume; www; work; visit; online; consideration; submit; immediate; history
- **Job Description (Topic 18):**
experience; skills; communication; high; good; candidates; qualified; looking; necessary; need; energy; company; independent; hiring; new; business; growing; train; national; media
- **Job Title (Topic 0):**
sales; b2b; representative; outside
- **Work History (Topic 0):**
level; desk; analyst; help

Use new dominant topics generated by approach 2

- **Job Requiremet (Topic 16):**
com; apply; resume; www; work; visit; online; consideration; submit; immediate; history
- **Job Description (Topic 7):**
career; looking; company; great; need; new; growing; experience; high; skills; hiring; good; energy; candidates; representatives; train; necessary; business; qualified; advertising
- **Job Title (Topic 3):**
sales; representative; outside; b2b
- **Work History (Topic 13):**
desk; help; level; analyst

FIGURE 4.27: Use-case: Generate narrative on EBM_side_info local explanation. Keywords extracted by two approaches.

It is also necessary to compare the extracted keywords with the original text input. For example, for the job requirement, the raw text (quoted below) does not contain much information about the skills or experience requirements for the jobs. Thus, extracted keywords might not be helpful.

> *For immediate consideration call (248) 514-1991 and ask for Andre. To submit your resume or work history, please apply via Online. Visit us online at www.rtui.com*

Another example is the job description. The raw job description (quoted below) is still quite short, and half of it contains an introduction about the company. However, extracted keywords contain meaningful terms that capture the nature of this job: "representative, advertising, communication, media, energy".

> *Independent Sales Representatives! Register Tapes Unlimited (RTUI) is hiring! Business is good. We are growing. We need you now! LOOKING FOR A NEW CAREER?* ***National Media Company is looking for sales candidates to sell dramatic new advertising media. Qualified candidates will have enthusiasm, high energy and great communication skills. No experience necessary.*** *WILL TRAIN 100%. You pick territory.*

The extracted keywords may help interpret the interaction between user-related features and job-related features in the local explanation 4.24. For example, the interaction between "ManageOthers & DescTopic" and "CurrentlyEmployed & DescTopic" does not support this user in applying for this job. Specifically, the job description states "No experience necessary", thus, a candidate who has already worked as a manager might not be motivated to apply for the job. However, in this case, the user actually applied for this job probably because of (s)he wants to change career direction from IT to Business and this job is a good starting point.

Chapter 5

Discussion of findings

5.1 Research Question 1: JRS performance

The performance of JRS is evaluated on the basis of two aspects: the ability to rank an application and the ability to retrieve relevant jobs for users. Most of the ranking algorithms used in the experiment perform well in the first aspect, which means that the choice of algorithms may not have much influence on this task. The high prediction performance could be attributed to the feature engineering and large amount of data. As user information and job information are well represented (using all details of the TF-IDF matrix 3.2) and balance of the target label (by negative sampling for interaction labels), using simple models like Logistic Regression or Naive Bayes can also perform well like the ensemble counterparts (such as XGBoost, AdaBoost).

However, the ability to retrieve relevant jobs depends on several factors.
The most important factor is the representation of positive labels in the potential application space. In the experiment, most job recommendation systems using random ranking data perform much better than those using KNN-generated ranking data because the percentage of positive labels in the random ranking data is higher.

Labels	**Sampling with controlled label**	**KNN - LDA**	**KNN - TF-IDF**
#"Apply" (1)	3,716	147	174
#"Not apply" (0)	367,884	368,951	371,426
# applications	371,600	369,098	371,600
Percentage of label (1)	1.000%	0.040%	0.047%

Table 5.1: Data-centric explanation: Percentage of positive labels in augmented potential applications

Table 5.1 shows the percentage of positive labels on three augmented application datasets. For two datasets generated by unsupervised KNN models, even though the algorithm learns better from the context of the application (similarity measured by distance metric), if the original data have a low percentage of positive application for each user, it will likely suggest "similarly negative label" applications. Thus, ranking models will continue learning from this negative pattern and suggest applications with users are not likely to apply (label 0). Therefore, the final recommendation performance was a very low hit rate (below 5%). Increasing the fraction of positive labels to only 1% can boost the recommendation significantly, even in random potential applications: most of the JRS on this dataset get a hit_rate higher than 80%.

Another data-centric explanation for the poor performance of unsupervised KNN models in augmenting potential applications was the **cold-start** situation. In this experiment, applications were not split randomly but following the original design of the dataset, using column "Split". As a result, after data cleaning, 100% of the test users are cold-start users. In other words, no previous applications by test users were made in the training application data for the unsupervised KNN model to learn the context. Thus, it learns more from the context of the jobs (the same jobs applied by different users, rather than the same users applied to different jobs). This issue could be improved by using a different splitting strategy to minimize the number of users in the cold start.

5.2 Research Question 2 & 3: Explanation Utility

Global explanation

It can be seen from both local and global explanations that location-matching features play a significant role in predicting the outcome of an application. However, this contribution may be attributed to the nature of the dataset. The CareerBuilder dataset was collected entirely in the US; thus, it lacks diversity in the representation of matching in location features.

When dropping location-matching features from the model, the role of the user profile and job detail was more visible, as illustrated by global and local explanations of the EBM_side_info model.

Regarding the contribution of text features toward the final decision, job features (either "job_matrix" or "ReqTopic", "TitTopic", "DescTopic") in general contribute more to the decision than User features ("work_matrix" or "WorkHistoryTopic"). This is again due to the nature of the datasets. Specifically, the user work history was represented only by job titles linked to a user, so the amount of information from this text is much less than the amount of information for a job post. The titles may not cover skills or qualifications that users have while doing previous jobs. The lack of information about users' skills was attributed to the design of the Kaggle challenge (using CareerBuilder data), which is to predict the outcome of the applications rather than a recommendation problem.

Post-hoc local explanation

High dimensionality can influence the feasibility of implementing post hoc explanation techniques. For example, using factorization machines resulted in enormous dimensions, combined with large datasets that both current LIME and SHAP-Kernel libraries could not handle.

Regarding the utility of local explanations, Kernel SHAP shows better performance in retaining important features (based on a small sample of explanations). Kernel SHAP already combined LIME and classical Shapley values to optimize the sampling for computing feature contribution, which could be attributed to the observation in this experiment. This performance comes with computation cost as Kernel SHAP still took a considerably longer execution time than LIME. However, the result might not hold if we change the explanation samples or perform on the whole test application. Thus, having the best representative of the sample for an explanation would validate the observation.

Finally, when comparing local self-explanation (by EBM_side_info) and local post hoc explanation use cases, both LIME and Kernel SHAP failed to capture the interaction between features. Given the nature of the system is matching jobs to users, such interaction effects may give a more intuitive explanation compared to considering individual factors.

The only limitation of EBM in the `interpretml` library is that it does not support extracting local explanations. The output of the local explainer is wrapped in an object for interactive visualization; thus, no contribution weights could be extracted for evaluating the magnitude of contribution like (SHAP and LIME).

Which explanation is more suitable?

From the perspective of the audience (job seeker), using a local explanation is preferable to a global explanation, as it is tailored to a specific user and the recommended jobs. However, the local explainer may not always come up with the same conclusion (whether the user will apply for the job) because it makes predictions based on the surrogate model.

Choosing a suitable sample size for the local explainer can help improve the fidelity of the prediction. All use cases in this LIME and SHAP experiment in this project use a sample size of 500, which can guarantee a consistent prediction between surrogate and black-box models. Using a smaller sample size (e.g. 100) resulted in different predicted labels. Furthermore, the poor explanation may also be attributed to the poor performance of the black-box model. In other words, the low "accuracy" in prediction by the local surrogate model is one indicator of the low performance of the original black-box models.

5.3 Research Question 4: Generating an intuitive explanation for lay users

From the two use cases in Section 4.7 the challenge of explaining job recommendations comes from the feature engineering process rather than the ranking models. Human-digestible explanations should interpret the embedded features. Thus, the overall XAI pipeline is still model-specific, as it cannot provide a complete intuitive explanation without knowing the model mechanism. Post hoc techniques such as LIME and Kernel SHAP can be used to generate a raw explanation.

As discussed in section 5.2, the user's work history text data are less informative than the job's text data. Thus, it can lead to misleading explanations by matching only the job titles the user previously has with the job titles of the open position. In previous studies, such as [36], user information is represented only by the skills section in their CVs, and job information is filtered using only job requirements. Having such a comparable filter of the raw text will improve the explanation.

Using visualization combined with text can help the audience understand the explanation. From the two use cases, the extracted keywords or converted feature names still need further filtering or processing to form a narrative explanation. Equally important, the raw text (job description, job requirement) can also influence the quality of the explanation. If the raw documents are too vague or contain too little information, the keywords extracted might not be helpful.

Chapter 6

Conclusion

6.1 Reseach limitation

First, the data set used for the experiment was only in the US and does not provide ranking ground truth. This may lead to bias in favor of matching the location of job seekers and the job post. In addition, text data from users are less informative compared to that on the jobs, which may lead to bias in explanation towards embedded job features.

The second limitation also comes from the dataset, including the sparsity and the cold start situation by design. A better augmentation strategy should combine the context (as in the KNN ranking data) and the control presence of positive labels (as random ranking data) for the best results.

Third, the experiment design only considers applications made by users with existing work histories. This means that for users without work history information (for example, fresh graduates), the findings from this project may not be applied.

Fourth, the experiment only simulates the complexity of job recommendation systems, instead of fully reproducing a SOTA system as introduced in the ACM RecSys Challenge. This is due to the mismatch between user characteristics and job characteristics in the dataset. Specifically, in the ACM data, users and jobs have many compatible tabular features. Baseline models of ACM RecSys 2017 show at least five types of matching that can be inferred from comparing users and table features: matching in the current user title and job role, discipline, career level, industry and location (country and region).

Finally, all explanation evaluation in this study is based on an algorithm approach rather than human evaluation. The interpretation of the explanation in the last experiment use cases is from the author's perspective. User experiment surveys would add a comprehensive evaluation of the quality of the explanation.

6.2 Future work

Future research may investigate the opening issues or limitations of the current experiment design. First, in terms of data enhancement for interaction or ranking ground truth, one might consider using an algorithm to simulate job applications or using different negative sampling strategies, as in the study [3]. The sampling methods introduced in RecBole library currently support random negative sampling and popularity-biased negative sampling. Having a dataset with more geographical diversity may add implications to job recommendations, especially when hybrid and remote work is becoming more popular. Implicit feedback also adds more diversity and better reflects reality, as users may show their interest in a job before actually applying to the position (such as implicit feedback in ACM RecSys datasets).

Equally important, evaluation recommendation systems should consider both relevance (i.e., via hit rate metrics) and the level of certainty in decision-making (i.e., via the probability of predicted class). Raising the threshold for decisions higher than 50% may help eliminate ambiguous decisions. However, in the case of users changing career direction, a recommendation with lower certainty is still helpful as long as it explains the ambiguity. For example, job seekers can use the explanation to identify their skills gap.

Third, the complexity of the algorithm used for recommendation systems can be extended with neural network-based models or multilayer ensemble models. Tuning hyperparameters for surrogate models in post hoc explanation techniques can help improve the accuracy of local prediction.

Finally, narrative generation is not fully automated and scalable. Due to time limitations, in the current setting, each component of the narrative generation pipeline is already automated, such as extracting raw terms and plotting explanations with converted features. These can be input for building a narrative generation platform, such as the Crystal Candle in the LinkedIn study [33].

6.3 Conclusion

This research focuses on explainable AI in recommendation systems using Kaggle's CareerBuilder 2012 dataset. It builds an end-to-end process that includes the generation of recommendations, the generation of explanations, and the communication of explanations in an intuitive manner.

Empirical results from extensive comparison and use cases bring three main findings:
(1) The performance of JRS depends on the selection of potential applications before ranking them. Combining the context of existing applications and the controlled positive sampling during the shortlisting process will help improve the relevance of recommendations. The ranking performance was similar among four groups of models: white-box, black-box, Factorization Machines (FM), and Explainable Boosting Machines (EBM).
(2) Two explanation approaches, model-agnostic and model-specific, show mixed results in explanation utility. Model-agnostic models (EBM and DPEBM) can retrieve a maximum of half of the recommendations of the original black-box systems. EBM models show potential in both performance and self-explanation (globally and locally). Post hoc explanation techniques, LIME and kernel SHAP, can retrieve most of the important features of the original model at a local explanation scope. Equally important, the explanations from both approaches can reflect the bias in the recommendation systems, which in this case links to data quality. This is helpful for system developers.
(3) Communicating human-digestible explanations requires a deeper understanding of feature engineering as the systems used transform text data. Mixed use of text explanation and visualization has been illustrated as a use case for the "post-explanation process".

The findings of this project contribute to the research community in explainable AI and explainable recommendation systems in three aspects: (1) The project validates the performance of the factorization machine algorithm ([26]) in recommendation systems using a new kind of input: text data. The project also tackles explanation challenges in job recommendation systems when combined with post-hoc XAI techniques. (2) The project validates the ensemble explainable algorithms in [23, 16, 22] in terms of prediction performance and explainability, for systems that require complicated feature engineering. (3) Finally, the study provides detailed data-centric explanations for future implications.

Appendix A

Detailed description of dataset - CareerBuilder 2012

user.tsv: contains information about the users. Each row of this file describes a user. The UserID column contains a user's unique id number, the WindowID column contains which of the 7 windows the user is assigned to, and the Split column tells whether the user is in the Train or Test groups. The remaining columns contain demographic and professional information about the users.

#	Column	Non-Null Count	Dtype
0	UserID	389,708 non-null	int64
1	WindowID	389,708 non-null	int64
2	Split	389,708 non-null	object
3	City	389,708 non-null	object
4	State	389,218 non-null	object
5	Country	389,708 non-null	object
6	ZipCode	387,974 non-null	object
7	DegreeType	389,708 non-null	object
8	Major	292,468 non-null	object
9	GraduationDate	269,477 non-null	object
10	WorkHistoryCount	389,708 non-null	int64
11	TotalYearsExperience	375,528 non-null	float64
12	CurrentlyEmployed	347,632 non-null	object
13	ManagedOthers	389,708 non-null	object
14	ManagedHowMany	389,708 non-null	int64

Table A.1: Dataset description - user.tsv - 389,708 entries

user_history.tsv: contains information about a user's work history. Each row of this file describes a job that a user held. The UserID, WindowID, and Split columns have the same meaning. The JobTitle column represents the job title, and the Sequence column represents the order in which the user held that job, with smaller numbers indicating more recent jobs.

jobs.tsv: contains information about job postings. Each row of this file describes a job post. The JobID column contains the job posting's unique id number, and the WindowID column contains which of the 7 windows the job was assigned to. The other columns contain information about the job posting. Two of these columns deserve special

#	**Column**	**Dtype**
0	UserID	int64
1	WindowID	int64
2	Split	object
3	Sequence	int64
4	JobTitle	object

Table A.2: Dataset description - user_history.tsv - 1,753,901 entries

attention, the StartDate and EndDate columns. These columns indicate the period this job posting was visible on careerbuilder.com. Each job was visible for part of its 13-day window, but not necessarily for the entire 13 days.

#	**Column**	**Non-Null Count**	**Dtype**
0	JobID	1,091,923 non-null	int64
1	WindowID	1,091,923 non-null	int64
2	Title	1,091,916 non-null	object
3	Description	1,091,913 non-null	object
4	Requirements	1,050,521 non-null	object
5	City	1,091,921 non-null	object
6	State	1,091,922 non-null	object
7	Country	1,091,920 non-null	object
8	Zip5	685,724 non-null	object
9	StartDate	1,091,923 non-null	object
10	EndDate	1,091,914 non-null	object

Table A.3: Dataset description - jobs.tsv - 1,091,923 entries

apps.tsv: contains information about applications made by users to jobs. Each row describes an application. The UserID, WindowID, Split, and JobID columns have the same meanings as above, and the ApplicationDate column indicates the date and time at which UserID applied to JobId.

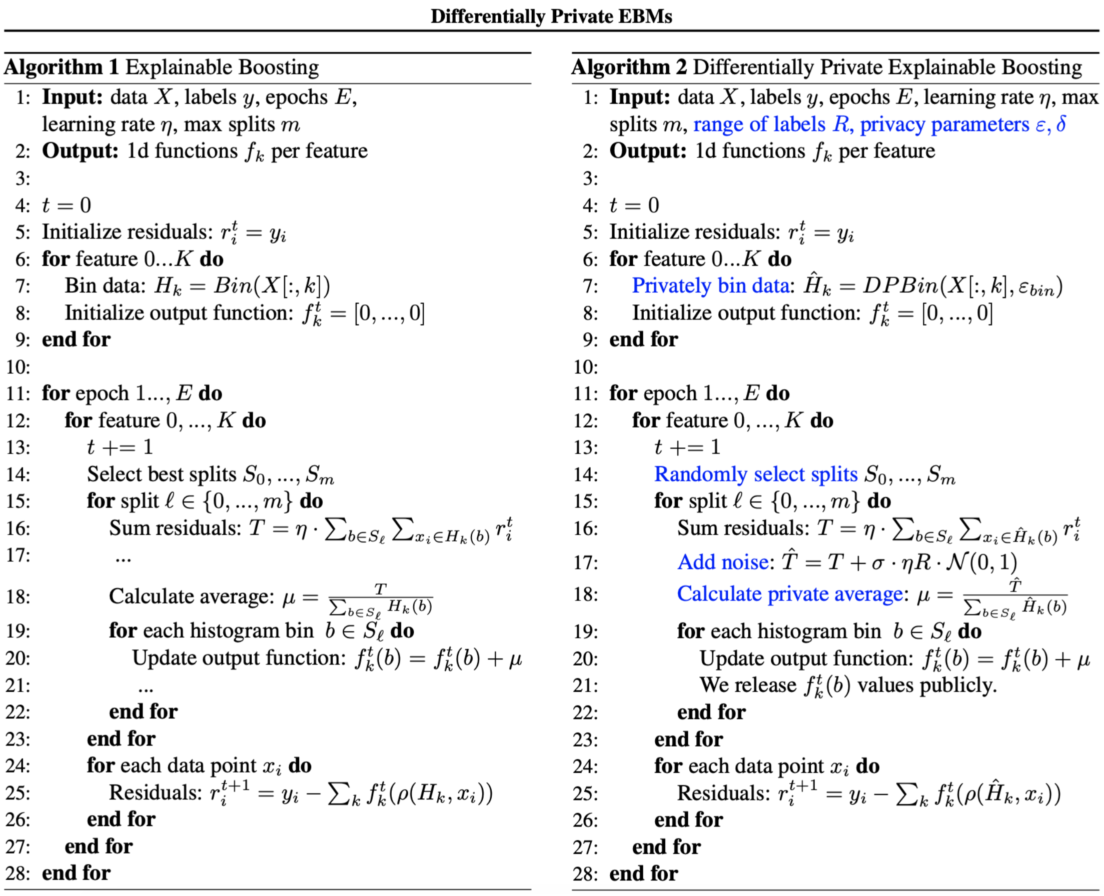
#	**Column**	**Non-Null Count**	**Dtype**
0	UserID	1,603,111 non-null	int64
1	WindowID	1,603,111 non-null	int64
2	Split	1,603,111 non-null	object
3	ApplicationDate	1,603,111 non-null	object
4	JobID	1,603,111 non-null	int64

Table A.4: Dataset description - apps.tsv - 1,603,111 entries

Appendix B

Pseudo code of explainable ensemble models

The figure below summarizes pseudo code in ensemble explainable models: EBM and DPEBM [22].



Differentially Private EBMs

Algorithm 1 Explainable Boosting

```
1: Input: data $X$, labels $y$, epochs $E$,
   learning rate $\eta$, max splits $m$
2: Output: 1d functions $f_k$ per feature
3:
4: $t = 0$
5: Initialize residuals: $r_i^t = y_i$
6: for feature $0...K$ do
7:     Bin data: $H_k = Bin(X[:, k])$
8:     Initialize output function: $f_k^t = [0, ..., 0]$
9: end for
10:
11: for epoch $1..., E$ do
12:     for feature $0, ..., K$ do
13:         $t += 1$
14:         Select best splits $S_0, ..., S_m$
15:         for split $\ell \in \{0, ..., m\}$ do
16:             Sum residuals: $T = \eta \cdot \sum_{b \in S_\ell} \sum_{x_i \in H_k(b)} r_i^t$
17:             ...
18:             Calculate average: $\mu = \frac{T}{\sum_{b \in S_\ell} H_k(b)}$
19:             for each histogram bin $b \in S_\ell$ do
20:                 Update output function: $f_k^t(b) = f_k^t(b) + \mu$
21:                 ...
22:             end for
23:         end for
24:         for each data point $x_i$ do
25:             Residuals: $r_i^{t+1} = y_i - \sum_k f_k^t(\rho(H_k, x_i))$
26:         end for
27:     end for
28: end for
```

Algorithm 2 Differentially Private Explainable Boosting

```
1: Input: data $X$, labels $y$, epochs $E$, learning rate $\eta$, max
   splits $m$, range of labels $R$, privacy parameters $\varepsilon, \delta$
2: Output: 1d functions $f_k$ per feature
3:
4: $t = 0$
5: Initialize residuals: $r_i^t = y_i$
6: for feature $0...K$ do
7:     Privately bin data: $\hat{H}_k = DPBin(X[:, k], \varepsilon_{bin})$
8:     Initialize output function: $f_k^t = [0, ..., 0]$
9: end for
10:
11: for epoch $1..., E$ do
12:     for feature $0, ..., K$ do
13:         $t += 1$
14:         Randomly select splits $S_0, ..., S_m$
15:         for split $\ell \in \{0, ..., m\}$ do
16:             Sum residuals: $T = \eta \cdot \sum_{b \in S_\ell} \sum_{x_i \in \hat{H}_k(b)} r_i^t$
17:             Add noise: $\hat{T} = T + \sigma \cdot \eta R \cdot \mathcal{N}(0, 1)$
18:             Calculate private average: $\mu = \frac{\hat{T}}{\sum_{b \in S_\ell} \hat{H}_k(b)}$
19:             for each histogram bin $b \in S_\ell$ do
20:                 Update output function: $f_k^t(b) = f_k^t(b) + \mu$
21:                 We release $f_k^t(b)$ values publicly.
22:             end for
23:         end for
24:         for each data point $x_i$ do
25:             Residuals: $r_i^{t+1} = y_i - \sum_k f_k^t(\rho(\hat{H}_k, x_i))$
26:         end for
27:     end for
28: end for
```

Figure B.1: Pseudo codes of Explainable Boosting Machines (EBM) and Differentially Private EBM model

Appendix C

Feature Engineering

C.1 Transform text features: user details and job details

Hyper-parameters of TF-IDF transformation on job details
Each document in the job detail corpus includes the job title, job description, and job requirements. The corpus has a total: 1,050,509 documents.

- ngram_range = (1, 2)
- min_df = 5
- max_features = 100
- stop_words= 'english'

Hyper-parameters of TF-IDF transformation on user work history
Each document in the corpus was all job titles linked to a user (identified by UserID). The corpus has a total: 152,292 documents.

- ngram_range = (1, 2)
- min_df = 5
- max_features = 50
- stop_words= 'english'

Hyper-parameters of LDA models

LDA models adopt the same hyper-parameters from the study by Lacic et. al. (2020) [11]. Lacic et. al. performed LDA only on job details while in this experiment, LDA was performed on both jobs and users information.

- n_components = 20
- max_iter = 5
- learning_method = 'online'
- learning_offset = 50

C.2 LDA Topic model visualization

Wordclouds of LDA Topic Model for Job details

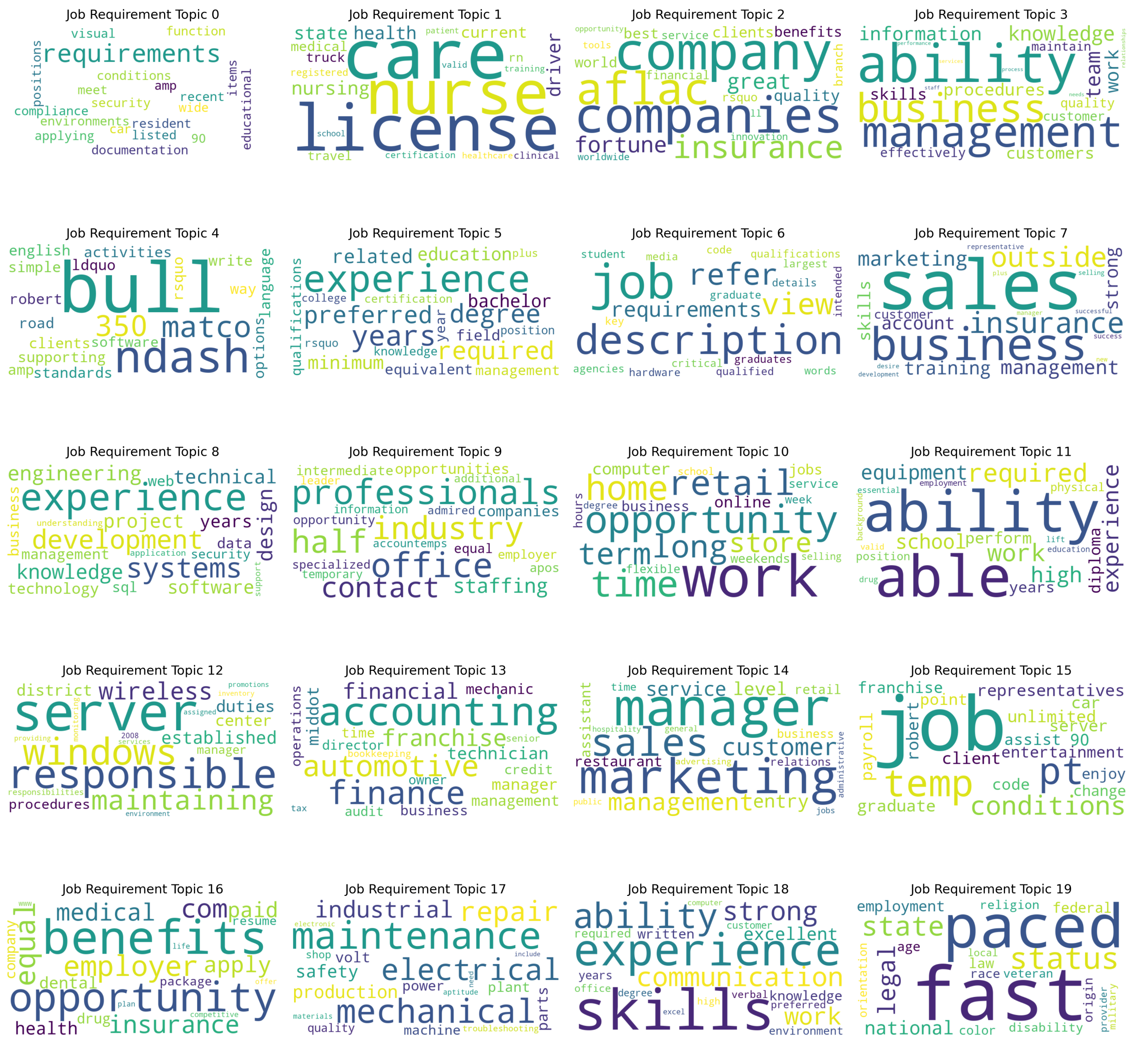


FIGURE C.1: Wordcloud-Job Requirement LDA Topic Modelling

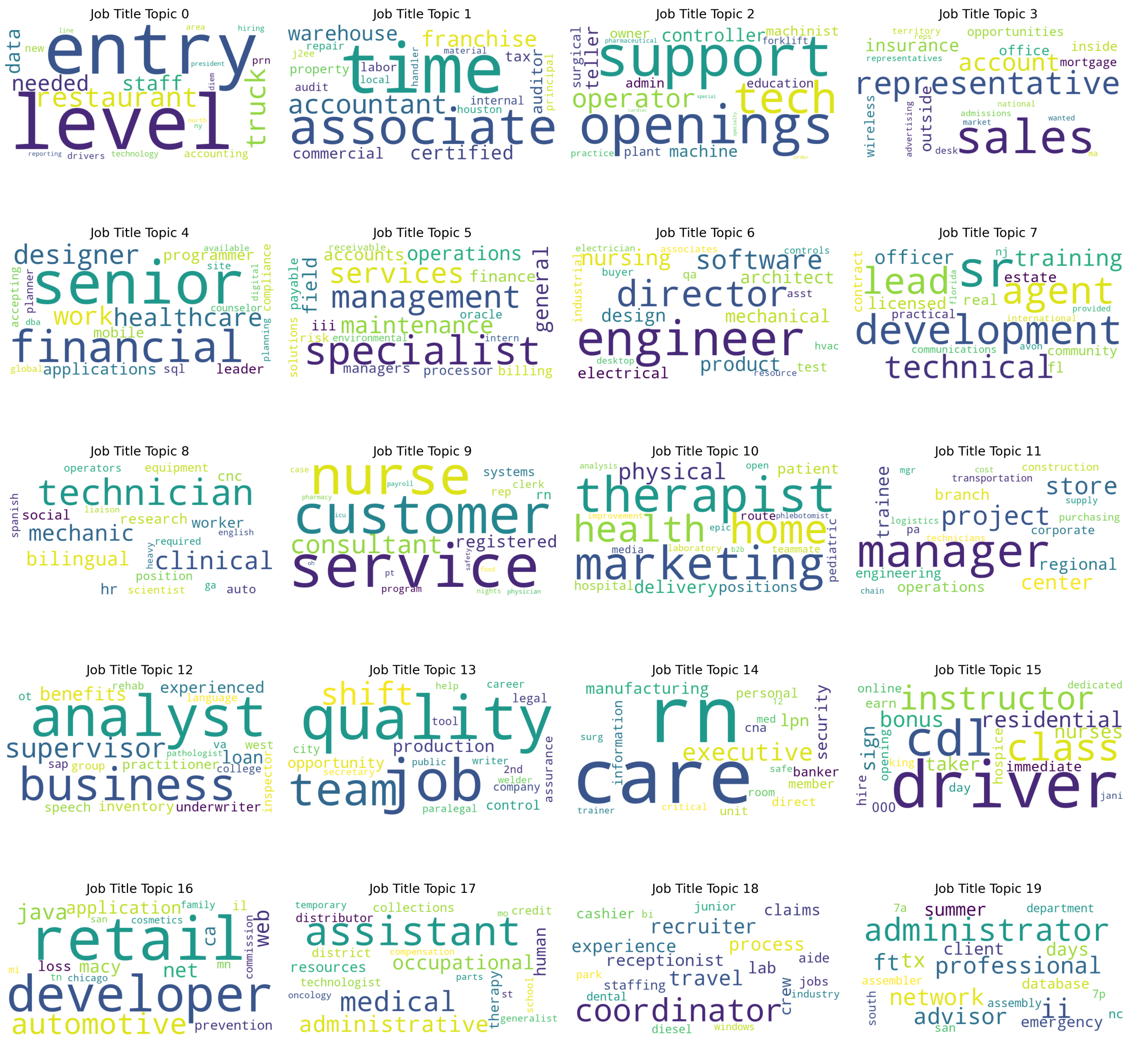


FIGURE C.2: Wordcloud-Job Title LDA Topic Modelling

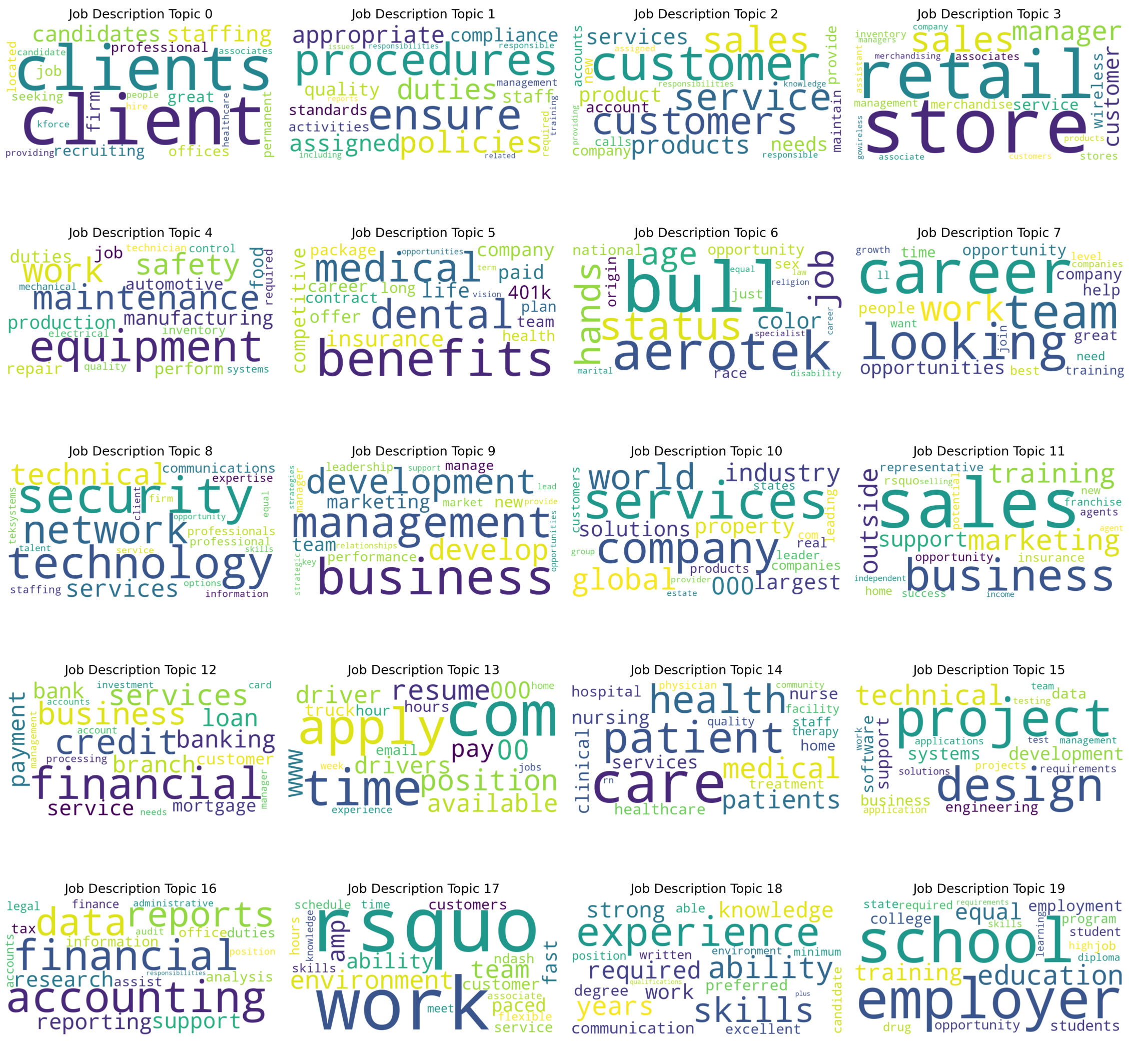


Figure C.3: Wordcloud-Job Description LDA Topic Modelling

Wordclouds of LDA Topic Model for User Work history

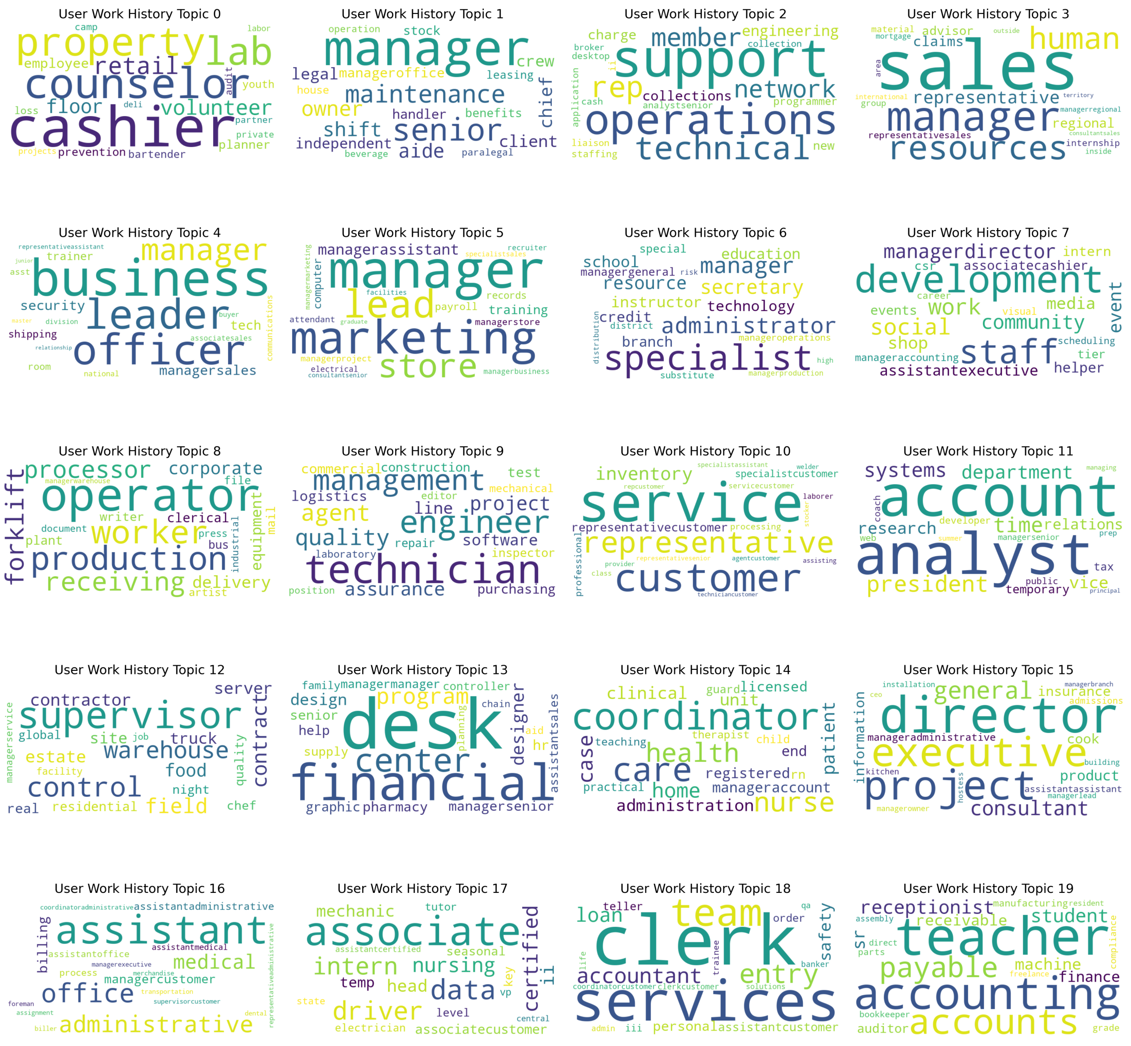


Figure C.4: Wordcloud-User work history LDA Topic Modelling

C.3 Discretization features

Discretization converts numeric features into categorical datatypes. These features train hybrid FM, EBM, and DPEBM models. The bin sizes for WorkHistoryLevel, and YearOfExperience level adopt from ACM RecSys 2017.

Table C.1: Discretization details for user side information

Features	**Discretized values**
UserID:	Unique User ID
DegreeType:	0 = None 1 = High school 2 = Vocational 3 = Associate's 4 = Bachelor's 5 = Master's 6 = PhD
CurrentlyEmployed:	0 = No, 1 = Yes
ManagedOthers:	0 = No, 1 = Yes
WorkHistoryTopic: LDA dominant topic	0-19
WorkHistoryLevel: Converted from numeric feature "WorkHistoryCount"	0 = less than 1 job 1 = 1-2 jobs 2 = 3-4 jobs 3 = 5 or more jobs
SeniorLevel: Converted from numeric feature "YearsOfExperience"	0 = less than 1 year 1 = 1-3 years 2 = 3-5 years 3 = 5-10 years 4 = 10-15 years 5 = 16-20 years 6 = more than 20 years

Table C.2: Discretization details for job side information

Features	**Value**
JobID	Unique Job ID
TitTopic: LDA dominant topic	0-19
DescTopic: LDA dominant topic	0-19
ReqTopic: LDA dominant topic	0-19

Appendix D

Detailed experiment design

D.1 Feature and ranking model

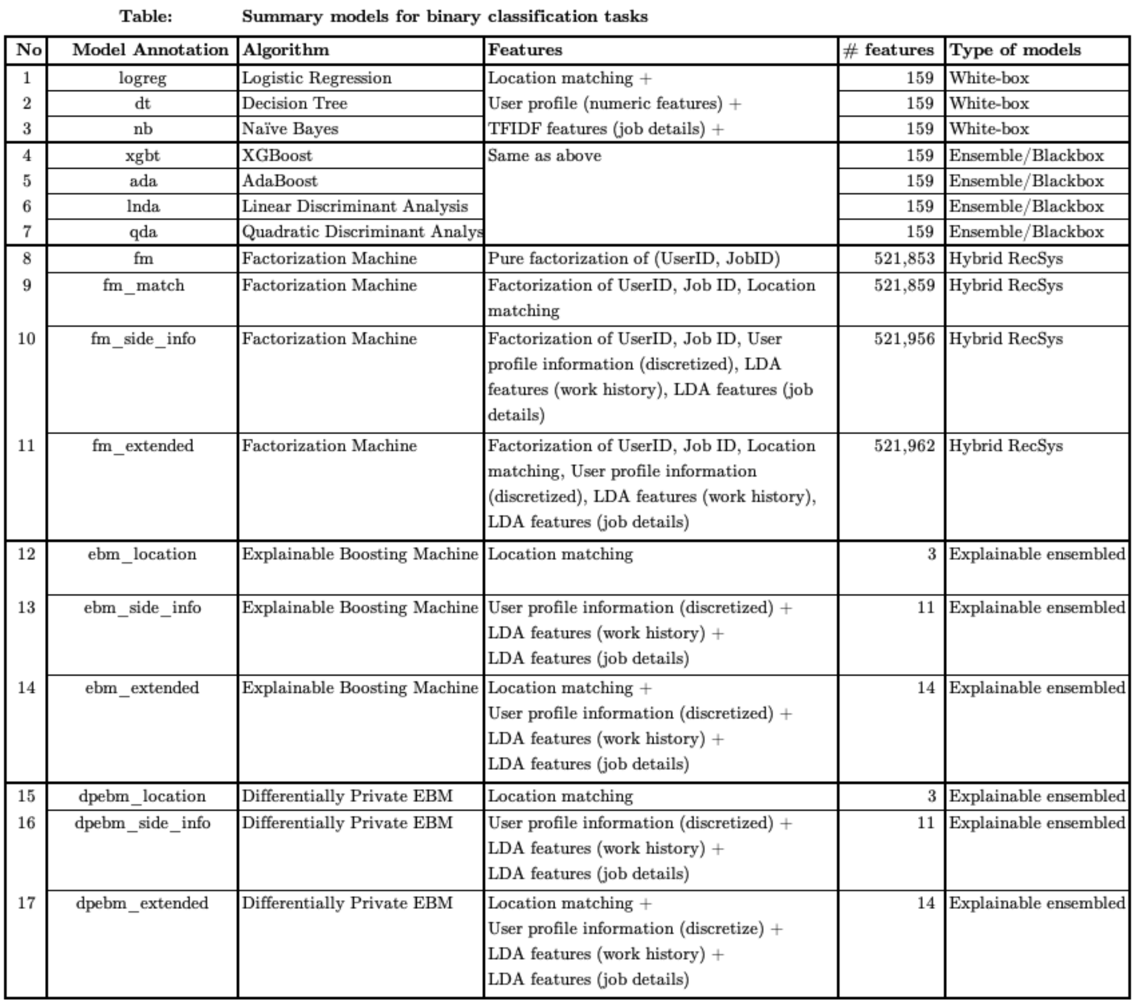


Table: Summary models for binary classification tasks

No	Model Annotation	Algorithm	Features	# features	Type of models
1	logreg	Logistic Regression	Location matching +	159	White-box
2	dt	Decision Tree	User profile (numeric features) +	159	White-box
3	nb	Naïve Bayes	TFIDF features (job details) +	159	White-box
4	xgbt	XGBoost	Same as above	159	Ensemble/Blackbox
5	ada	AdaBoost		159	Ensemble/Blackbox
6	lnda	Linear Discriminant Analysis		159	Ensemble/Blackbox
7	qda	Quadratic Discriminant Analys		159	Ensemble/Blackbox
8	fm	Factorization Machine	Pure factorization of (UserID, JobID)	521,853	Hybrid RecSys
9	fm_match	Factorization Machine	Factorization of UserID, Job ID, Location matching	521,859	Hybrid RecSys
10	fm_side_info	Factorization Machine	Factorization of UserID, Job ID, User profile information (discretized), LDA features (work history), LDA features (job details)	521,956	Hybrid RecSys
11	fm_extended	Factorization Machine	Factorization of UserID, Job ID, Location matching, User profile information (discretized), LDA features (work history), LDA features (job details)	521,962	Hybrid RecSys
12	ebm_location	Explainable Boosting Machine	Location matching	3	Explainable ensembled
13	ebm_side_info	Explainable Boosting Machine	User profile information (discretized) + LDA features (work history) + LDA features (job details)	11	Explainable ensembled
14	ebm_extended	Explainable Boosting Machine	Location matching + User profile information (discretized) + LDA features (work history) + LDA features (job details)	14	Explainable ensembled
15	dpebm_location	Differentially Private EBM	Location matching	3	Explainable ensembled
16	dpebm_side_info	Differentially Private EBM	User profile information (discretized) + LDA features (work history) + LDA features (job details)	11	Explainable ensembled
17	dpebm_extended	Differentially Private EBM	Location matching + User profile information (discretize) + LDA features (work history) + LDA features (job details)	14	Explainable ensembled

FIGURE D.1: Feature engineering and ranking model settings [22]

Appendix E

Hit rate of job recommendation systems

Detail hit rate results for job recommendation systems using potential applications from random ranking data.

Table E.1: Hit_rate@k of all JRS on test interaction data
Source of potential application: random ranking data

Ranking model	**hit_rate@5**	**hit_rate@10**	**hit_rate@20**
logreg	0.8754	0.8808	0.8959
dt	0.8875	0.8875	0.8891
nb	0.8778	0.8875	0.9029
xgbt	1.0000	1.0000	1.0000
ada	0.8743	0.8805	0.8961
lnda	0.8749	0.8808	0.8953
qda	0.8773	0.8864	0.9055
fm	0.3275	0.4836	0.8654
fm_match	0.3275	0.4836	0.8654
fm_side_info	0.3154	0.4525	0.5988
fm_extended	0.8987	0.9095	0.9268
ebm_location	1.0000	1.0000	1.0000
ebm_side_info	0.1389	0.2449	0.3614
dpebm_location	1.0000	1.0000	1.0000
dpebm_side_info	1.0000	1.0000	1.0000
dpebm_extended	1.0000	1.0000	1.0000

Table E.2: Hit_rate@k of all JRS on test interaction data
Source of potential application: unsupervised KNN ranking data

Ranking model	**hit_rate@5**	**hit_rate@10**	**hit_rate@20**
logreg	0.0361	0.0393	0.0401
dt	0.0393	0.0396	0.0422
nb	0.0344	0.0363	0.0396
xgbt	0.0420	0.0425	0.0425
ada	0.0361	0.0393	0.0398
lnda	0.0361	0.0393	0.0401
qda	0.0336	0.0385	0.0401
fm	0.0043	0.0051	0.0079
fm_match	0.0043	0.0051	0.0079
fm_side_info	0.0041	0.0054	0.0073
fm_extended	0.0084	0.0095	0.0098
ebm_location	0.0076	0.0089	0.0092
ebm_side_info	0.0011	0.0022	0.0035
ebm_extended	0.0073	0.0087	0.0095
dpebm_location	0.0076	0.0089	0.0092
dpebm_side_info	0.0011	0.0022	0.0035
dpebm_extended	0.0076	0.0089	0.0092

Appendix F

Local explanation

The table below provides information about the application sample used in evaluation post-hoc explanation approaches in section 4.5.

UserID	JobID	label
13	821691	1
13	329572	0
514	131166	1
514	620304	0
681	654542	1
681	625758	1
681	15081	1
681	291467	0
681	1035056	0
681	2083	0
767	85377	1
767	385582	0
883	663823	1
883	136012	0
1006	655183	1
1006	466034	0
1066	709999	1
1066	1096020	1
1066	700868	1
1066	726478	0

TABLE F.1: Test application sample for model interpretation. The "label" column represents the actual outcome: 1 - Apply, 0 - Not apply. The green row is the selected application for comparison in local explanation.

Usecase for comparison in local self-explanation versus post-hoc local explanation

User profile:

UserID	13
WindowID	6
Split	Test
City	Philadelphia
State	PA
Country	US
ZipCode	19143
DegreeType	Bachelor's
Major	Psychological & Social Sciences
GraduationDate	40878
WorkHistoryCount	6
TotalYearsExperience	5

User work history: Job titles:

- Pennsylvania Mentor
- Student Worker
- Internship in Adoption Unit
- Student Worker - Continuing Education
- Sales Associate

Job detail:

- **JobID: 821691 - Job Title: Group Counselor**
- **Job Description**: The Group Counselor provides social skill interventions within a camp environment to children presenting with social, emotional, and behavioral problems. The Group Counselor is responsible for modeling appropriate social skill interactions for campers and providing a safe environment where campers can learn and practice positive social skills. Key Areas of Responsibilities: Provide age-appropriate recreational and therapeutic activities within the camp environment. Provide crisis management techniques and de-escalation strategies. Provide a safe and structured environment for campers to engage in therapeutic activities. Encourage positive peer social interactions throughout all activities. Create problem-solving and positive decision-making activities. Create activities related to improving social skills and interactions. Introduce and practice social decision making social problem-solving concepts with campers. Provide campers opportunity for leadership and various roles within each activity and group. Model appropriate social interactions and decision making skills for campers. Provide encouragement and praise for campers efforts in each activity. Participate in all activities with campers. Access resources to help appropriately address campers mental health needs. Collaborate with directors to access needed materials and resources for behavioral plans and interventions. Ensure cleanliness of camp environment. Responsible for daily goal sheets to document camperrsquo;s behaviors. Will provide safety techniques and physical emergency intervention due to safety concerns when needed. Consults with: The Group counselor will consult with directors at camp site. The group counselor will have daily interactions with all counselors at camp. Group counselor will consult with and collaborate with all STAP staff to ensure the safety and all campers and assist with behavioral strategies. Term of employment: Seasonal; summer position (June ndash; August) Hours: 8:30 ndash; 4:00. Must be flexible with work hours, will work day, and evenings as necessary.
- **Job Requirement**: Bachelors degree or non-degreed candidates accepted. Experience providing services to children with behavioral problems and social skill deficits. Relevant experience providing services to children in a camp setting.

F.1 Data use for evaluation local explanation

F.2 Feature importance fidelity - Local explanation

Table F.2: Feature importance fidelity for Kernel SHAP local explaination

Original Model	**# features**	**# gold features**
ebm_location	3	4
ebm_side_info	11	19
ebm_extended	14	13
logreg	159	20
dt	159	20
xgbt	159	4

Table F.3: Feature importance fidelity for LIME local explanation

Original Model	**# features**	**# gold features**
ebm_location	3	4
ebm_side_info	11	19
ebm_extended	14	13
logreg	159	20
dt	159	20
xgbt	159	4

Appendix G

Appendix: Use Case - Interpret explanation for recommendations

G.0.1 User information

User profile:

UserID	577106
WindowID	5
Split	Test
City	Chicago
State	IL
Country	US
ZipCode	60628
DegreeType	Master’s
Major	Computer Information Systems
GraduationDate	01/10/2009 0:00

Work History:

- Help Desk Analyst Level 3
- Help Desk Level 3

G.0.2 Case 1: Interpret post hoc explanation by Kernel SHAP and LIME

The table below shows the result of the JRS recommendation using logistic regression as a ranking algorithm, the source of potential application is the random ranking data. Column "Y_pred" is the predicted label by the recommendation system. Column "Y_pred_posthoc" is the predicted label by the post-hoc explainers, including LIME and Kernel SHAP. The column "label" is the actual interaction between the user and the jobs from the original data.

UserID	JobID	Y_prob	Y_pred	rank	label	Y_pred_posthoc
577106	355928	0.950842121	1	0	1	1
577106	514855	0.534474817	1	1	0	1
577106	296141	0.512052627	1	2	0	1
577106	596860	0.499821843	0	4	0	1
577106	1057551	0.499680993	0	5	0	1
577106	404437	0.488222237	0	9	0	0
577106	115760	0.478346461	0	13	0	0
577106	838051	0.475534386	0	15	0	0
577106	1076079	0.47423476	0	17	0	0

TABLE G.1: "Post-explanation" use case: JRS ranking algorithm: LogReg; Source of applications: Random ranking data; UserID = 577106. The first application was selected for detailed post-hoc explanation and interpretation in Figures 4.22 and 4.23

The post hoc explanation of the first application: UserID = 577106, JobID = 355928 is illustrated in Figs. 4.22 and 4.23. The following are the raw text features of JobID: 355928.

- **Job Title**: Technical Support Specialist
- **Job Description**: Technical Support Specialist Mount Prospect, IL Company Description: Since 1887, Cummins Allison has produced a wide range of products. Today we manufacture a variety of equipment serving financial, gaming and commercial marketplaces across the U.S. and worldwide. Our factory direct sales of top quality products include: High speed coin and currency processing equipment as well as other office products. For more information, visit our web site at www.cumminsallison.com. Responsibilities: Currency pattern development for International currencies both in house and onsite worldwide. Working with mechanical, software and electronic design engineers to identify and resolve problems relating to currency pattern development. The ability to travel once or twice a month for up to one week or longer to international locations.
- **Job Requirement**: Qualifications: Applicants must be skilled in Microsoft Word, Excel and basic network communications for the PC and servers. An excellent command of English is essential, with other languages a plus. Minimum requirements include an Associate's Degree in Electronic or Computer Technologies or a certificate of completion from an accredited Technical Degree program. Field experience for electro mechanical devices is also beneficial. Reply information: We provide excellent benefits including a successful 401k plan and tuitionassistance. Please send e-mail responses, including salary history, to jobs@cumminsallison.com EOE m f d v

G.0.3 Case 2: Interpret local self-explanation by EBM side info model

The table below shows the recommendation generated by JRS using EBM_side_info as a ranking algorithm. The source of potential application is KNN ranking data. Column "Y_pred" is the predicted label by the recommendation system. The column "Y_pred_local" is the predicted label generated by the model itself but in the local scope. The column "label" is the actual interaction between the user and the jobs from the original data.

UserID	JobID	Y_prob	Y_pred	rank	label	Y_pred_local
577106	92240	0.551190765	1	0	1	1
577106	16606	0.551190765	1	1	1	1
577106	367673	0.551190765	1	2	1	1
577106	14404	0.506421643	1	3	1	1
577106	927790	0.50383071	1	4	1	1
577106	766469	0.480923437	0	5	0	0
577106	649469	0.480923437	0	6	0	0
577106	356244	0.480923437	0	7	0	0
577106	500312	0.480923437	0	8	0	0
577106	1017568	0.480923437	0	9	0	0
577106	610171	0.480923437	0	10	0	0
577106	251074	0.473254323	0	11	0	0
577106	281238	0.472712644	0	12	0	0
577106	453343	0.472712644	0	13	0	0
577106	24530	0.472712644	0	14	0	0
577106	476420	0.472712644	0	15	0	0
577106	1052122	0.472712644	0	16	0	0
577106	899538	0.472712644	0	17	0	0
577106	480445	0.472712644	0	18	0	0
577106	96236	0.472712644	0	19	0	0

TABLE G.2: "Post-explanation" use case: JRS ranking algorithm EBM_side_info model; Source of applications: KNN-ranking data; UserID = 577106. The explanation for the first application (green row) was illustrated by figures 4.24

The local self-explanation of the first application: UserID = 577106, JobID = 92240 was illustrated in figure 4.24. Below are the raw text features of JobID: 92240.

- **Job ID:** 92240. - **Job Title**: Outside Sales Representative - B2B
- **Job Description**: Independent Sales Representatives! Register Tapes Unlimited (RTUI) is hiring! Business is good. We are growing. We need you now! LOOKING FOR A NEW CAREER? National Media Company is looking for sales candidates to sell dramatic new advertising media. Qualified candidates will have enthusiasm, high energy, and great communication skills. No experience is necessary. WILL TRAIN 100%. You pick territory.
- **Job Requirement**: For immediate consideration call (248) 514-1991 and ask for Andre. To submit your resume or work history, please apply via Online. Visit us online at www.rtui.com

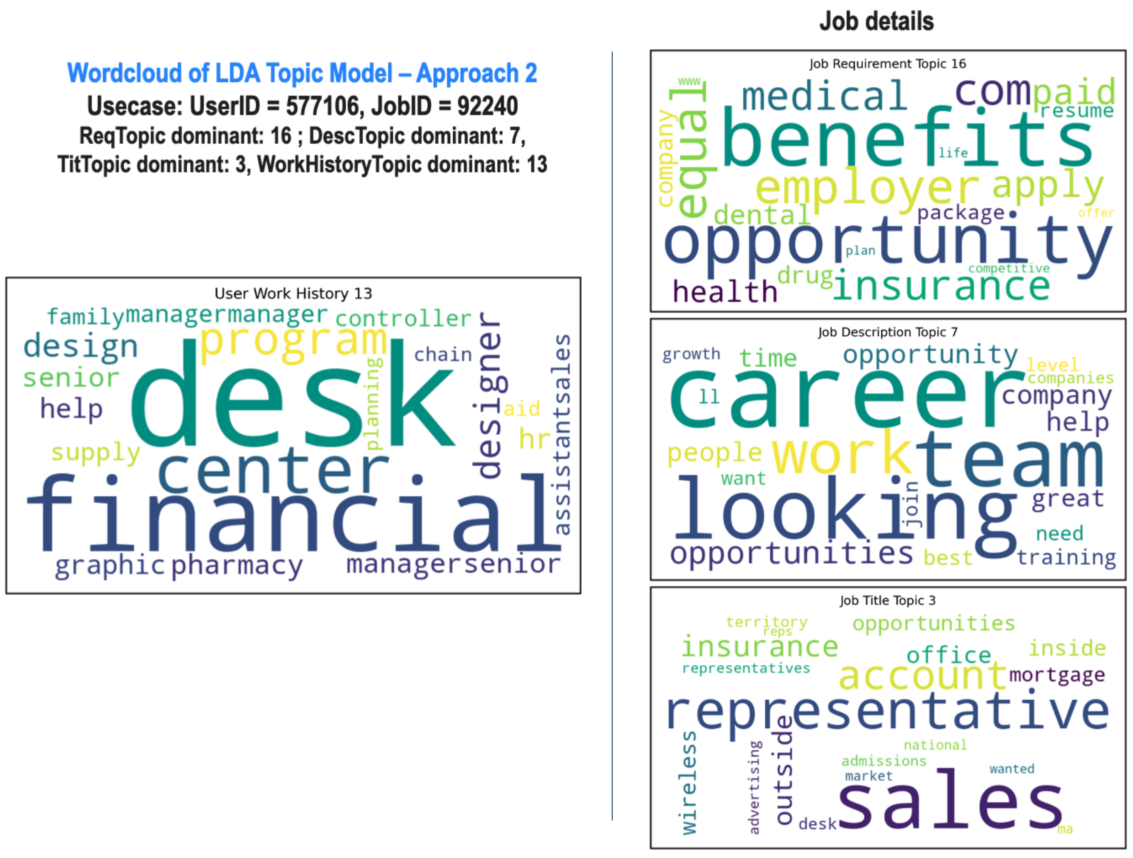


FIGURE G.1: Use-case: Generate narrative on EBM_side_info local explanation. Wordcloud of LDA topic model for the use case. The dominant topic was recalculated based on the standard topic-document distribution approach. Thus, results may be different from the original features.

Bibliography

[1] Alejandro Barredo Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador Garcia, Sergio Gil-Lopez, Daniel Molina, Richard Benjamins, Raja Chatila, and Francisco Herrera. Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai. *Information Fusion*, 58:82–115, 2020. URL: https://www.sciencedirect.com/science/article/pii/S1566253519308103, doi:https://doi.org/10.1016/j.inffus.2019.12.012.

[2] Wojciech Krupa Ben Hamner, Road Warrior. Job recommendation challenge, 2012. URL: https://kaggle.com/competitions/job-recommendation.

[3] Corné de Ruijt and Sandjai Bhulai. Job recommender systems: A review, 2021. arXiv:2111.13576.

[4] David Gunning and David Aha. Darpa's explainable artificial intelligence (xai) program. *AI Magazine*, 40(2):44–58, Jun. 2019. URL: https://ojs.aaai.org/aimagazine/index.php/aimagazine/article/view/2850, doi:10.1609/aimag.v40i2.2850.

[5] Xingsheng Guo, Houssem Jerbi, and Michael P O'Mahony. An analysis framework for content-based job recommendation. In *22nd International Conference on Case-Based Reasoning (ICCBR), Cork, Ireland*, volume 29, 2014.

[6] Yifeng Guo, Yu Su, Zebin Yang, and Aijun Zhang. Explainable recommendation systems by generalized additive models with manifest and latent interactions, 2020. arXiv:2012.08196.

[7] Sophia Hadash, Martijn C. Willemsen, Chris Snijders, and Wijnand A. IJsselsteijn. Improving understandability of feature contributions in model-agnostic explainable ai tools. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*, CHI '22, New York, NY, USA, 2022. Association for Computing Machinery. doi:10.1145/3491102.3517650.

[8] Patrick Hall and Navdeep Gill. *An introduction to machine learning interpretability*. O'Reilly Media, Incorporated, 2019. URL: https://www.oreilly.com/library/view/an-introduction-to/9781492033158/.

[9] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. Session-based recommendations with recurrent neural networks, 2016. arXiv:1511.06939.

[10] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009. doi:10.1109/MC.2009.263.

[11] Emanuel Lacic, Markus Reiter-Haas, Dominik Kowald, Manoj Reddy Dareddy, Junghoo Cho, and Elisabeth Lex. Using autoencoders for session-based job recommendations. *User Modeling and User-Adapted Interaction*, 30(4):617–658, Sep 2020. doi:10.1007/s11257-020-09269-1.

[12] Jianxun Lian, Fuzheng Zhang, Min Hou, Hongwei Wang, Xing Xie, and Guangzhong Sun. Practical lessons for job recommendations in the cold-start scenario. In *Proceedings of the Recommender Systems Challenge 2017*, RecSys Challenge '17, New York, NY, USA, 2017. Association for Computing Machinery. doi:10.1145/3124791.3124794.

[13] Q. Vera Liao and Kush R. Varshney. Human-centered explainable ai (xai): From algorithms to user experiences, 2022. arXiv:2110.10790.

[14] Zachary C. Lipton. The mythos of model interpretability. *Commun. ACM*, 61(10):36–43, sep 2018. doi:10.1145/3233231.

[15] Kuan Liu, Xing Shi, Anoop Kumar, Linhong Zhu, and Prem Natarajan. Temporal learning and sequence modeling for a job recommender system. In *Proceedings of the Recommender Systems Challenge*, RecSys Challenge '16, New York, NY, USA, 2016. Association for Computing Machinery. doi:10.1145/2987538.2987540.

[16] Yin Lou, Rich Caruana, Johannes Gehrke, and Giles Hooker. Accurate intelligible models with pairwise interactions. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '13, page 623–631, New York, NY, USA, 2013. Association for Computing Machinery. doi:10.1145/2487575.2487579.

[17] Scott M. Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 4768–4777, Red Hook, NY, USA, 2017. Curran Associates Inc.

[18] Sonu K. Mishra and Manoj Reddy. A bottom-up approach to job recommendation system. In *Proceedings of the Recommender Systems Challenge*, RecSys Challenge '16, New York, NY, USA, 2016. Association for Computing Machinery. doi:10.1145/2987538.2987546.

[19] C. Molnar. *Interpretable Machine Learning*. Leanpub, 2020. URL: https://books.google.nl/books?id=jBm3DwAAQBAJ.

[20] Shun Morisawa and Hayato Yamana. *Faithful Post-hoc Explanation of Recommendation Using Optimally Selected Features*, pages 159–173. Springer International Publishing, Cham, 2021. doi:10.1007/978-3-030-89385-9_10.

[21] Caio Nóbrega and Leandro Marinho. Towards explaining recommendations through local surrogate models. In *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*, SAC '19, page 1671–1678, New York, NY, USA, 2019. Association for Computing Machinery. doi:10.1145/3297280.3297443.

[22] Harsha Nori, Rich Caruana, Zhiqi Bu, Judy Hanwen Shen, and Janardhan Kulkarni. Accuracy, interpretability, and differential privacy via explainable boosting, 2021. arXiv:2106.09680.

[23] Harsha Nori, Samuel Jenkins, Paul Koch, and Rich Caruana. Interpretml: A unified framework for machine learning interpretability, 2019. arXiv:1909.09223.

[24] Andrzej Pacuk, Piotr Sankowski, Karol Wundefinedgrzycki, Adam Witkowski, and Piotr Wygocki. Recsys challenge 2016: Job recommendations based on preselection of offers and gradient boosting. In *Proceedings of the Recommender Systems Challenge*, RecSys Challenge '16, New York, NY, USA, 2016. Association for Computing Machinery. doi:10.1145/2987538.2987544.

[25] Georgina Peake and Jun Wang. Explanation mining: Post hoc interpretability of latent factor models for recommendation systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery amp; Data Mining*, KDD '18, page 2060–2069, New York, NY, USA, 2018. Association for Computing Machinery. doi:10.1145/3219819.3220072.

[26] Steffen Rendle. Factorization machines. In *Proceedings of the 2010 IEEE International Conference on Data Mining*, ICDM '10, page 995–1000, USA, 2010. IEEE Computer Society. doi:10.1109/ICDM.2010.127.

[27] Marco Ribeiro, Sameer Singh, and Carlos Guestrin. "why should I trust you?": Explaining the predictions of any classifier. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, pages 97–101, San Diego, California, June 2016. Association for Computational Linguistics. URL: https://aclanthology.org/N16-3020, doi:10.18653/v1/N16-3020.

[28] Cynthia Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1(5):206–215, May 2019. doi:10.1038/s42256-019-0048-x.

[29] Roan Schellingerhout, Volodymyr Medentsiy, and Maarten Marx. Explainable career path predictions using neural models. In *Recommender Systems for Human Resources 2022*, volume 3218 of *CEUR Workshop Proceedings*, page 7, 2022.

[30] Maksims Volkovs, Guang Wei Yu, and Tomi Poutanen. Content-based neighbor models for cold start in recommender systems. In *Proceedings of the Recommender Systems Challenge 2017*, RecSys Challenge '17, New York, NY, USA, 2017. Association for Computing Machinery. doi:10.1145/3124791.3124792.

[31] Maksims Volkovs, Guang Wei Yu, and Tomi Poutanen. Content-based neighbor models for cold start in recommender systems. In *Proceedings of the Recommender Systems Challenge 2017*, RecSys Challenge '17, New York, NY, USA, 2017. Association for Computing Machinery. doi:10.1145/3124791.3124792.

[32] Wenming Xiao, Xiao Xu, Kang Liang, Junkang Mao, and Jun Wang. Job recommendation with hawkes process: An effective solution for recsys challenge 2016. In *Proceedings of the Recommender Systems Challenge*, RecSys Challenge '16, New York, NY, USA, 2016. Association for Computing Machinery. doi:10.1145/2987538.2987543.

[33] Jilei Yang, Diana Negoescu, and Parvez Ahammad. Crystalcandle: A user-facing model explainer for narrative explanations, 2022. arXiv:2105.12941.

[34] Chenrui Zhang and Xueqi Cheng. An ensemble method for job recommender systems. In *Proceedings of the Recommender Systems Challenge*, RecSys Challenge '16, New York, NY, USA, 2016. Association for Computing Machinery. doi:10.1145/2987538.2987545.

[35] Yongfeng Zhang and Xu Chen. Explainable recommendation: A survey and new perspectives. *Foundations and Trends® in Information Retrieval*, 14(1):1–101, 2020. URL: http://dx.doi.org/10.1561/1500000066, doi:10.1561/1500000066.

[36] Chen Zhu, Hengshu Zhu, Hui Xiong, Chao Ma, Fang Xie, Pengliang Ding, and Pan Li. Person-job fit: Adapting the right talent for the right job with joint representation learning. *ACM Trans. Manage. Inf. Syst.*, 9(3), sep 2018. doi:10.1145/3234465.