Web Scraping as a Data Source for Machine Learning Models and the Importance of Preprocessing Web Scraped Data

MAXIM FRAŇO, University of Twente, The Netherlands

The general concept of machine learning (ML) cannot work without large amounts of data. There are many methods of data gathering, ranging from writing down data manually to using complex algorithms. This research specifically focuses on web scraping as a method for data extraction, and its effect on ML models. The approach in this research is split into two parts, theoretical and practical. First, the connection between web scraping and ML is observed through literature analysis. Later an experiment focused on using non-preprocessed web scraped data in a dataset for training an ML model is carried out using a tool made in Python. The results of the literature review show that web-scraped data can be greatly varied, gathered through a wide range of tools, and used in a large number of different ML models, while the experiment shows the importance of preprocessing web-scraped data to achieve high performance in these models.

Additional Key Words and Phrases: Machine Learning, Web Scraping, Artificial Intelligence, Python, Sentiment Analysis

1 INTRODUCTION

Machine learning (ML) is more and more at the forefront of innovations within the field of artificial intelligence (AI), for example, due to the popular resurgence of large language models, such as ChatGPT[1]. To go more in-depth, ML is a field of study within the field of AI that aims to create algorithms to emulate human intelligence by learning from its environment [9]. Machine learning can be applied in almost any field, as long as the work in that field generates data. Finance, engineering, or more specific use cases such as pattern recognition or computer vision are all use cases for ML [31, 37]. The previously mentioned environment that ML models learn from is in this case the data the algorithm is provided. This data can be "labeled" or "unlabeled", depending on whether the model utilizes supervised or unsupervised learning, and can be in the form of text, numbers, images, videos, etc [36]. The question is how this data can be obtained. There are many possible approaches to obtaining data for ML models, for example utilizing public databases or pre-made datasets. However, if the data that one is looking for cannot be easily found, other methods have to be employed to obtain such data. For example, relevant scans from magnetic resonance imaging (MRI) for a specific disease might be difficult to find, so to train an ML model, a sufficient number of MRI scans have to be made of both healthy individuals and individuals suffering from the disease that the ML models want to predict [27].

In this research, the focus is on a method of data acquisition called web scraping. Web scraping or web crawling is a form of data extraction that utilizes querying public websites and extracting the HTML (HyperText Markup Language) code of which these websites consist [17]. Then, only necessary parts from this HTML code are extracted, in a process called preprocessing, and lastly, the data is stored in a proprietary system, such as Excel [10, 17]. The aim of this research is to look into the use of web-scraped data in the field of ML through the means of a literature review, in order to find common practices and a deeper understanding of combining web-scraped data and ML models. The findings from the literature review then supplement an experiment that is carried out using a tool programmed in Python which is used to get technical insight into the use of web-scraped data for training ML models. Through the combination of both of these approaches, conclusions are drawn to answer the research question at hand.

[1]ChatGPT is a natural language processing model that generates human-like responses to user prompts [2]

TScIT 41, July 5, 2024, Enschede, The Netherlands

© 2024 University of Twente, Faculty of Electrical Engineering, Mathematics and Computer Science.
Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

2 PROBLEM STATEMENT

Currently, there are a number of papers that look at the concept of web scraping as a whole, including its applications, or papers solely focused on the applications of web scraping [17, 38]. A closely related paper [39] already touches on some high-level aspects of web-scraping as a data source for machine learning (ML) models in the form of a review. However, there is a lack of papers that focus specifically on the application of web scraping in the domain of ML from both a theoretical and technical perspective. This paper consequently aims to fill this gap in research and examine the connection between web-scraping and ML from both perspectives. To fill the gap, research papers that combine web scraping and ML as a methodology to achieve research objectives are examined in the way of an unsystematic literature review. This research outlines different fields of application mentioned in the examined papers, the type of data extracted, the software used, and the choice of ML models. Consequently, after the results of the literature review, an experiment is carried out that puts the combination of web scraping and ML into practice to provide a technical perspective.

2.1 Research Question

The main research question is as follows:
To what extent can web scraped data be used as a complete or complementary data source for machine learning models?

The following sub-questions have been derived to help answer the main research question:

(1) *How has the combination of web-scraping and ML been utilized in a selection of other research papers?*
(2) *To what extent is the performance of ML models affected when using non-preprocessed web-scraped data in a dataset used for training an ML model?*

To answer the first sub-question we make use of a literature review to explore different applications of the combination of web scraping and ML and focus on the extracted data, software, and ML models in each paper. To answer the second sub-question we make use of use of a web scraping tool to test how non-preprocessed data affects the performance of ML models.

3 PRACTICAL CONTRIBUTION

The takeaway from this paper should be a deeper understanding of how web-scraped data interacts with ML models. While a multitude of papers provide a high-level description of either the applications of web scraping or its technical aspects, to the author's best knowledge, none provide a more in-depth look into how the theoretical information applies to practice. This research consequently allows for the outlining of a more technical approach to the high-level theoretical step-by-step process of using web-scraped data to train an ML model. By applying a more technical approach, it is also possible to outline the hurdles and examine the consequences of not adhering to proper practice through the results of an experiment. This paper should serve to some extent as a guideline for practitioners or researchers who are considering using web-scraped data as a data source for ML models.

4 METHODOLOGY

For easier navigation, this paper is split into two parts. In the first part, a literature review is carried out. In the second part, an experiment is carried out, where a web scraping tool is developed alongside a machine learning (ML) model. The methodology for the experiment follows the CRISP-ML (Cross-Industry Standard Process for Machine Learning) approach [18]. The CRISP-ML methodology consists of seven stages: Project initiation and planning (1), data exploration and cleansing (2), data predictive potential evaluation (3), data enrichment (4), model building and evaluation (5), deriving business insights (6), model deployment and reporting (7). From this approach, only stages (1), (2), (3), and (5) are followed. Stage (4) is omitted because it is concerned with identifying sources for improving the quality of the data, which is not the concern of this experiment. Stages (6) and (7) are excluded because the experiment does not provide business insight and the ML model is not deployed for practical use.

The conclusions from both of these parts are combined to form a discussion where the research question can be answered. In the following subsections, a more elaborate explanation of both of the structural parts can be found.

4.1 Part 1: Literature Review

In the literature review part, five papers are chosen as a representative sample. This number of papers is chosen primarily due to the scope of this research. Each of these papers should be concerned with different application areas, such as medicine or finance, however, the methodology that was used to carry out the research in these papers should remain the same. More specifically, the methodology in these papers should include the combination of the use of web-scraped data for training ML models. Using these papers, information regarding which software or other tools were used for web scraping, which ML models were utilized, and how the combination of web scraping and ML benefited the research carried out within the individual papers is extracted. To search for the papers, the database Web of Science is to be utilized, with the search terms "web scraping" and "machine learning". Papers with higher citation counts are prioritized.

4.2 Part 2: Web scraping tool and Dataset used

The web scraping tool should provide a deeper insight into the technical side of the combination of web scraping and ML. In the experiment carried out using this tool, both preprocessed and non-preprocessed web-scraped data are used in a dataset for training an ML model. Preprocessed data can be imagined as data that is ready to be analyzed either by a human or a computer [10]. Non-preprocessed data on the other hand contains problems that could interfere with an analysis such as gaps in data or incorrect format [10]. In practice, ML models are trained on a dataset that is split into two parts, the first being called the training data, and the second testing data [42]. The testing data is always part of the original dataset, given that the ML model should be tested on similar, yet unfamiliar data. In the experiment to be carried out, both the training and testing portion of the original dataset are replaced, both partially and completely, by non-preprocessed web-scraped data, which does not bear complete similarity to the original preprocessed dataset. Through this, it is intended to find the effect of introducing web-scraped data into the dataset, primarily on the accuracy of the model. The tool consists of two parts. A web scraping and data cleaning tool that is used to gather data from the web for the ML model, and the ML model itself. The ML algorithm that is used belongs to the category of "classifiers", which are part of the "supervised learning" category of algorithms. The Python library that facilitates the model is Sci-Kit Learn. The dataset used for the ML model is concerned with sentiment analysis of articles about sustainability and sustainable practices. This means that the ML model tries to classify whether the article written about this topic falls into two categories, i.e. a positive stance toward sustainability, or a negative. The main idea of the dataset is to contain as sanitized data as possible, so when non-preprocessed web-scraped data is introduced, the difference between using preprocessed data and non-preprocessed web-scraped data can be observed.

5 RELATED WORK AND ADDITIONAL KNOWLEDGE

As mentioned in Section 2, there are multiple articles concerned with applications of web scraping. In [17], alongside methods of web scraping using different libraries, the authors also outline applications of web scraping in fields such as cybersecurity, data science, business intelligence, and more. Another article [38] similarly to the previous article outlines some common practices when it comes to web scraping however, it also goes more in-depth into the technical aspects of the practice. The article also lists yet more applications, such as for research or marketing. As mentioned in Section 2, these articles were deemed insufficient when it comes to outlining both theoretical and technical details of the use of web-scraped data in the field of ML, however, they are still invaluable in getting insight

into applications of web scraping in general as well as some technical properties of it. The article [39] provides comparatively the most relevant information on the combination of web scraping and machine learning for this research, however, it does not go into technical details of the practice. To guide the search for literature and the literature review itself, [20] is used as a guideline. Similar to the guidelines on literature review, for the part of the research in which the web scraping tool is developed, technical literature about both web scraping and ML needs to be utilized. Starting with web scraping, for the choice of web scraping software and approach, alongside all examined articles in the literature review part, the article [17] is used. Consequently, for further guidance with the chosen software, in this case, Python, [29] is used. For applying ML techniques and creating an ML model, a more general literature is chosen [42] and [41], as well as a more specific one about Sci-Kit Learn [32], which is the library that is used in the web scraping tool.

6 FINDINGS

This section encompasses the findings from both parts outlined in Section 4. The steps taken in each part are explained in more detail as well as the findings per each part. Lastly, a discussion follows to relate the output from both parts.

6.1 Literature Review

As mentioned in subsection 4.1, five papers were chosen as a representative sample. These papers are a representative sample because they to a sufficient degree explain how both web scraping and machine learning (ML) were utilized in the research carried out by the authors of the papers. When the desired number of articles was reached, no other articles beyond that point for this search query were examined. The following are short descriptions of each article (summary in Table 1). [8] is concerned with skills required for Big Data professions. These skills are extracted from online job postings and classified using ML. [3] focuses on fake review detection on the website Yelp and the authors utilize Python and multiple different ML algorithms. [25] looks into identifying illicit drug trade on the social media platform Instagram with the help of web scraping and four supervised ML algorithms. [28] introduces a tool called SEOpinion that uses web scraping and deep learning to summarize aspects of products in e-commerce stores and identify opinions about them from customer reviews. Lastly, [13] concerns evaluating different ML techniques and their suitability for detecting Ponzi schemes within Ethereum smart contracts. For context, a Ponzi scheme in the world of smart contracts works the same as it would with other investments, i.e. early investors are paid with the money of new investors, or in this case using the cryptocurrency Ethereum [5]. The following subsections will first examine the different use cases, the software and libraries used, and the ML models used for the combination of web scraping and ML, per article. The information gathered in each of these subsections will be later used in the discussion to answer the first research sub-question.

6.1.1 Web Scrapped Data. From the analyzed articles, as well as from the principle of web scraping itself [17], we know that web scraping is used to gather data from publicly available websites. The examined articles show that the data that can be gathered is varied in theme as well as type. The theme, in this case, would refer to the area of concern of the data, such as reviews from e-commerce websites [3, 28], or required skills for job positions from online job posting websites [8]. When it comes to the type of data, all examined articles scrape textual information, or in other words plain text in a string format. However, web scraping also allows for the scraping of images, forms, or tables [29]. Web-scrapped data, however, also has its limitations. While publicly accessible websites on the web can be web scraped, some websites or organizations prohibit the extraction of data through their "Terms and Conditions" [22]. These conditions should always be adhered to when extracting data. This means that some data, for example, the scraping of academic articles, might not always be possible to extract.

Table 1. Summary of examined research papers during literature review

Reference	Description
[8]	Skills for Big Data professions
[3]	Fake review detection on Yelp
[25]	Identification of illicit drug dealers on Instagram
[28]	Framework for summarizing aspects of products in e-commerce
[13]	Detection of Ponzi schemes in smart contracts

6.1.2 Software and Libraries for Web Scraping. In this subsection, the software or programming languages and their libraries will be examined. Firstly, from the articles, we can see that Python was the most popular solution used by the authors [3, 13, 25, 28], while the authors of article [8] used Portia, a web-based solution. Python is a well-known language and is widely utilized for purposes of ML, primarily due to its simplicity [33]. However, other solutions, such as Portia, offer easier access to web scraping, potentially without the users being required to have any programming knowledge [16]. However, upon closer inspection, it is possible to find that the solution Portia provides has been programmed using Python using a popular library called Scrapy. This means that currently there are solutions on the market that offer easy-to-use interfaces and abstract what would normally need to be programmed manually, such as Portia, however, these solutions are still often based on Python [38]. Python is, however, not the only language that can be used for web scraping, and other notable languages include for example Java [38].

Given that Python was the most popular programming language used in the examined articles, it is also necessary to look at the libraries that the authors utilized. Out of the four papers that utilized Python, two papers used Scrapy [3, 28], one paper used Beautiful Soup [13], and paper [25] does not specify which library was used. Given that Scrapy and Beautiful Soup are among the most popular libraries [16], it is important to mention that these libraries are fundamentally different. Scrapy, being a full web scraping framework library, contains most tools a user needs to scrape without installing other libraries or packages [19]. On one hand, This means that Scrapy is focused more on large-scale web scraping, however, some

downsides include slower execution compared to other libraries as well as functions a user might never utilize [19]. On the other hand, Beautiful Soup is only an HTML or XML (eXtensible Markup Language) parser, which means that other libraries are required to allow for other necessary steps such as retrieving websites, for example, the Requests library [17]. This means that the choice between these two libraries depends on the use case, as well as the proficiency of a person's programming knowledge. Similarly, other libraries follow the same path, whereas they are either a full-scale solution, with another example being Selenium, or a library more specialized for facilitating only certain parts of the web scraping process, such as Urllib3 which is an alternative to the Requests library or Lxml, an alternative to Beautiful Soup [4].

6.1.3 Use of Machine Learning Models. The use of ML models in the analyzed articles varies greatly. Firstly, given that some articles used deep learning [25, 28], which is a subset of ML, we need to differentiate between these two concepts. ML as we currently know it comprises many different algorithms, some simple and some more complex, which fall into certain categories, such as supervised, semi-supervised, or unsupervised, depending on how data is provided to the algorithm [36]. Deep learning is a relatively new trend and tries to go a step further and emulate how the human brain learns using a model called Artificial Neural Network (ANN). In short, the main difference between these concepts is that deep learning is just a small subset of models out of the relatively large collection of ML models, however, it has become exceedingly more relevant in recent times. [24, 37, 41, 42]

Table 2 summarizes which models were used per examined paper as well as lists which model performed the best. The following is a more in-depth description of each model used. In [13], three algorithms were used, namely Decision Tree (DT), Support Vector Machine (SVM), and Multinomial Naive Bayes (MNB). A Decision Tree is based on deciding an outcome on the basis of a tree structure, for example, several yes or no questions are posed, and based on the answers to these questions, the algorithm ends at a different root of the tree, which in other words is the outcome [42]. An SVM tries to find the best boundary between different classes in data. This can be imagined as a linear line on a graph that separates points of class A and class B (if the classification is binary, otherwise more classes are possible)[42]. The Multinomial Naive Bayes algorithm utilizes the frequency of features, for example, words in a text, to make classifications based on learned probabilities [42]. [28] utilized SVM and Deep Learning (DL) using Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN), both of which are explained in more detail in [24]. [25] used a Decision Tree (DT), Random Forest (RF), SVM, and a custom deep learning model (DL). The Random Forest algorithm combines the output of multiple Decision Tree algorithms which are trained on randomly sampled subsets of the original training data [41]. In [3] Logistic Regression (LR), Decision Tree (DT), Random Forest (RF), Gaussian Naive Bayes (GNB), and AdaBoost (AB) were utilized. Logistic Regression is used for binary classification problems and uses a sigmoid function which is a function that maps a number in the range between 0 and 1. It is possible to also set a decision boundary, for example, 0.5, where anything above 0.5 is classified as positive and vice-versa. Gaussian Naive Bayes differs from Multinomial Naive Bayes only by the use of a Gaussian distribution instead of a multinomial distribution [42]. A Gaussian distribution uses continuous features rather than discrete as used in multinomial distribution. AdaBoost (Adaptive Boosting) is an ensemble learning algorithm that creates multiple weak learners to create a strong learner. This means that a strong classifier is created on the basis of the results of a number of weak classifiers, such as decision trees with only one level [11]. [8] used Latent Dirichlet Allocation (LDA) which works on the basis of a "bag-of-words" where the words are then allocated to a topic, which is a concept represented by a series of related words [42].

In a number of articles, the article authors utilized multiple algorithms to create machine-learning models. This is a common practice, called the "Ensemble Method", which trains multiple learners to solve the same problem [41]. For instance, the random forest algorithm is a good example of an ensemble method, given that it combines the output of multiple decision tree algorithms. The authors of the examined articles also used primarily supervised models since the data the authors worked with had clear labels. Next, it is important to observe which metrics the authors used to measure the effectiveness of their models. In [13], the most common metrics associated with ML were used, namely accuracy, precision, recall, and F-Score. In the case of [13], these metrics were above 95% in most cases, which should be the aim when training an ML model. However, it is important to mention, that when it comes to recall and precision, in all but simple problems, these metrics are contradictory, as in that if one is high the other one should be low [42]. Similarly, [25] scored high on both precision and recall metrics of all models, with the custom-made deep learning model scoring the highest. The reasons for high precision and recall at the same time can be many, however, in the case of [13] and [25] they are most likely caused by a selection of complex models, such as the deep learning model in [25], alongside a thoroughly preprocessed dataset. [8] utilized LDA for clustering instead of classification and provided no relevant metrics. Next, F-Score is a metric that also indicates whether more importance is placed upon recall or precision [42]. Given that recall and precision in the articles [13] and [25] were similar, it made sense to use the Fβ-Score where β is equal to 1, which indicates that recall and precision have the same level of importance [42]. Lastly, a confusion matrix is a useful tool to visualize how each label (in the case of this research positive or negative) is classified or misclassified as the other label [36] which was for example utilized in [28].

6.2 Web Scraping Experiment

Subsection 4.2 outlined the high-level approach for the experiment in this paper. In this subsection, the approach is outlined in more detail. This includes how the experiment is carried out, how the dataset is constructed, how the web scraping process works, which models are used and their development, and lastly the results of the experiment. The purpose of this experiment is to examine how non-preprocessed web-scraped data affects ML models in terms of performance metrics. Consequently, three different models are

Table 2. Summary of used ML models per article and which model achieved the highest accuracy ([8] does not provide metrics, [3]) uses F-score instead of accuracy

Reference	Models	Highest Accuracy(%) + Model
[8]	LDA	Not Applicable
[3]	LR, DT, RF, GNB, AB	82% - RF, AB (Used F-Score Instead)
[25]	DT, RF, SVM, DL	99% - DL
[28]	SVM, DL	83% - DL
[13]	DT, SVM, MNB	99% - MNB & SVM

trained to examine the effect of non-preprocessed data on the performance metrics of each model. Lastly, a comparison is drawn between the three different models and similarly to Subsection 6.1, the findings from this part, primarily from Subsection 6.2.4, are used in a discussion at the end of Section 6 to answer the second research sub-question.

6.2.1 Approach. To facilitate the experiment, several steps need to be taken. Firstly, data needs to be collected out of which a dataset can be created. This data is collected using an API (Application Programming Interface), which provides a quick way of gathering news articles in a JavaScript Object Notation (JSON) format. The use of an API is still considered web scraping in the context of this research. Then, this data needs to be preprocessed. This means that the text needs to be cleaned from any special characters and potential stop words, tokenized, which means split into phrases or words, normalized, such as all words being lowercased, and lastly labeled, which in our case would be the stance of the article on sustainable practices (i.e. positive or negative) [10]. For labeling, the TextBlob library is used, which uses an already pre-trained algorithm for sentiment analysis [26]. The ML model is developed using the Scikit-learn Python library and utilizes the Naive Bayes classifier which is an appropriate choice of algorithm for text classification [42]. When the ML model is prepared, training and testing take place, where the split between training and testing data is 70% training and 30% testing [42]. To put this into perspective, Fig 1 shows a framework that outlines all the required steps in this experiment.
When it comes to the experiment itself, first, a model is trained only using preprocessed data in the dataset, called the preprocessed model. After the preprocessed model, a mixed model is created, where the dataset consists of half non-preprocessed data and half preprocessed data. Non-preprocessed data in the context of this research means data in a raw HTML format, which is textual information that includes everything present on the website, such as advertisements, image text, headers and footers, navigation menus, and HTML tags such as which defines a section within an HTML document [21]. Lastly, a non-preprocessed model is trained and tested, which includes only non-preprocessed data. The purpose of each of the models is to classify whether an article can be considered positive or negative, and consequently, the aim is to track how each model performed at this task. For each of the three models, performance metrics are recorded, including accuracy, recall, precision, and F1-score. A confusion matrix is also provided.

For the experiment, Python version 3.12 is used. The IDE (Integrated Development Environment), an environment that provides developers with the tools required for programming, is PyCharm by JetBrains [23]. The libraries that are used to facilitate the solution are: "requests" used for accessing websites through Python. "json" allows the reading and writing of JSON files. "re" or regular expressions, used for text cleaning using patterns. "nltk" for its stop words package, which already includes predefined English stop words. "TextBlob" to assign sentiment to an article. "pandas" can convert JSON data into DataFrame objects which are used for the training of the ML model. "sklearn" facilitates the necessary tools to train the ML model as well as provides performance metrics and the values for the confusion matrix. "matplotlib" provides a visual representation for the confusion matrix based on values from sklearn.

6.2.2 Web Scraping and Dataset. The initial API used for creating a dataset of articles works on the basis of inputting a theme, which in this case would be "sustainability" or "sustainable practices", the language of the articles, in this case, English, and lastly the polarity of the articles, either positive, negative, or neutral. After inputting all parameters, a Python code is generated, which can be input into a Python file. The dataset size for this experiment is 2000 samples, with 1000 positive and 1000 negative articles. The choice of this sample size was inspired by the use of two datasets with the same sample size to also train a Naive Bayes model in chapter three of [1], as well as by other papers [14, 15].
As already mentioned, the API itself allows picking whether retrieved articles should be positive or negative. This was initially considered to be used instead of the TextBlob library for sentiment labeling, however, after manual inspection and processing of 1000 negative samples (as labeled by the API) through TextBlob which deemed only 30% to be actually negative, it was decided that this feature of the API would not be used and instead, TextBlob would relabel each article.
The output from the API was already in a sanitized form in a JSON format, which means that, unlike regular web scraping, there were no HTML elements to remove. The API provided large amounts of metadata about each article, however for the purpose of this experiment, only the title, body, and link to the article are relevant. The title primarily served as an identifier for articles. The body of the article was the main input for training the ML model. The body of an article consists in this case of an excerpt of the whole article with a length of around two short paragraphs. Lastly, the link to the article has been used to retrieve the article again, yet this time manually using the Requests library. This second retrieval was necessary to retrieve the same article in a non-preprocessed form, given that the API already does most of the preprocessing. This means that for both the positive and the negative labels, there are 1000 preprocessed and 1000 non-preprocessed articles, which are saved in four different JSON files. Examples of preprocessed and non-preprocessed article lines can be found in Figures 3 and 4 respectively. Lastly, it is important to mention the hurdles encountered during the web-scraping process. Web scraping works on the same principle as when a person accesses a website through their browser [38], which means that web scraping software can encounter the same problems. A common problem is a 404 error, which means

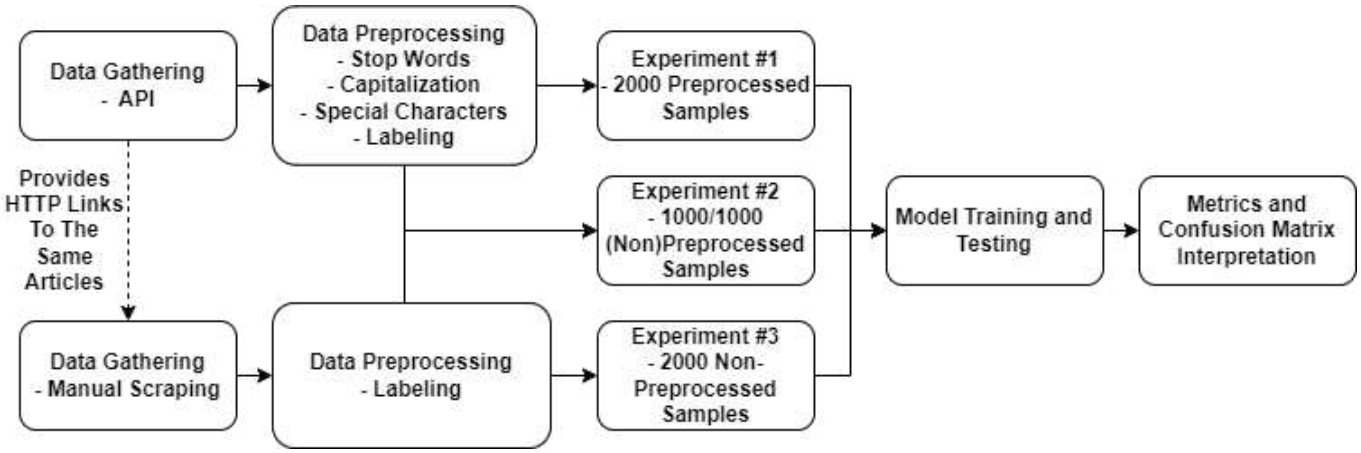

Fig. 1. The framework includes a high-level overview of the steps of the experiment, starting with data gathering, data preprocessing, model training and testing, and interpretation of the results. First articles are gathered through an API. 2000 of these articles are preprocessed and labeled as either positive or negative. The API also provides links to these articles, which are then accessed by the web scraping tool, and 2000 of the same articles are retrieved in a non-preprocessed form and only labeled either positive or negative. Then three experiments are carried out with preprocessed and non-preprocessed data taking up either a full or partial portion of the dataset used to train and test the model at hand. Lastly, the results are recorded in the form of metrics (accuracy, recall, precision, F1-score) and a confusion matrix.

Attendees can learn about sustainable living from central Minnesota businesses and organizations

Fig. 2. Original line from an article. It is missing some preprocessing steps, such as the removal of capitalization and stop words

attendees learn sustainable living central minnesota businesses organizations

Fig. 3. Article line as seen in Fig 2 in a preprocessed form. Contains no capitalization, special characters, or stop words

<p class="paragraph">Attendees can learn about sustainable living from central Minnesota businesses and organizations<p>

Fig. 4. Web scraped version of the article line as seen in Fig 2. Apart from the features of the original, it also includes HTML tags. In this case, the <p> HTML tag signifies the start of a paragraph

that the website for the given URL (Uniform Resource Locator) does not exist [7]. This error occurred during the experiment, given that the API retrieved articles up to 90 days from the past, which means that in the meantime, the article could have gotten deleted. Another common error is a 403 error, which most of the time does not happen to a regular user, however, a web scraping software might experience this given that a website might flag its connection attempt as automated, denying it access [7, 29]. A possible way to avoid 403 errors is the inclusion of proper headers in an HTTP (HyperText Transfer Protocol) request, where headers are metadata that contains more information about the sender of the request, making the request seem more human [7]. Next, if websites from different countries are scraped, which happened during this experiment with some websites being hosted in for example Asia, a 451 error may occur, which states that the access has been denied due to legal reasons [35]. It is important to mention that during web scraping, the IP (Internet Protocol) address of the person running the web scraping software is used by default [16]. Through an IP address, it is possible to obtain the geographical location of a person, or at minimum its internet provider [30]. This means that if a website in Asia uses geoblocking for European countries, which is a practice of limiting access based on geographical location, and is accessed through a European IP address, the access will be denied [40]. A solution to this is the use of a VPN (Virtual Private Network) to route the connection through a server located in a different country, effectively changing the IP address of the user [35]. Lastly, a very common error is a 429 error, where the server hosting the website has received too many requests from the same source, effectively blocking access for some period of time [34].

6.2.3 Model development. The Naive Bayes algorithm is chosen as a basis for the model in this experiment. This algorithm works on the principle of independence between variables [36, 42]. In the case of this experiment, where text classification is the main concern, the variables are words. However, according to [36], complete word independence is rarely the case, with an example being the phrase "first quarter" which has more occurrences in business articles than multiplying the probabilities of words "first" and "quarter" would suggest.

The Naive Bayes algorithm works on the basis of the Bayes Theorem. This theorem is useful for computing the probability of an event based on probabilities of events related to it. For example, in text classification, it is known that articles about sports include words such as "football", however, what is the probability of an article containing the word "football" to be actually about sports? In mathematical terms, this would mean deriving $P(topic|words)$ using

$P(words|topic)$, where P is a conditional probability. In simpler terms, first, what is the probability of an article being of a certain topic given the observed words, and second what is the probability of certain words being observed given the topic. [36, 42]
The Bayes Theorem formula computes the conditional probability of class y given the feature vector X ($P(y|X)$) by multiplying the conditional probability of feature vector X given class y ($P(X|y)$) by the probability of occurrence of class y ($P(y)$) and dividing by the marginal likelihood of feature vector X ($P(X)$) [36, 42]. Returning to the example of text classification of articles about sports, the formula would be as follows. To calculate the probability of an article being about sports based on a set of words, we need to multiply the probability of a set of words belonging to the topic of sports by the probability of the topic of sports occurring and divide by the probability of the set of words (Fig 5). It is also important to mention that the Bayes Theorem can be adjusted to different types of attributes, such as nominal (as used in this experiment), discrete, or continuous [42].

$$P(sports \mid "football") = \frac{P("football" \mid sports) \cdot P(sports)}{P("football")} \quad (1)$$

Fig. 5. The Bayes Theorem adjusted to the sports example

Several steps had to be taken in the Python implementation of the Naive Bayes algorithm. First, JSON data is normalized into a DataFrame object. Then, the "*test_train_split*" function is used, where the body of the articles and the label (after normalization either 0 for negative and 1 for positive) are the input arrays, and the parameter "*test_size*" is set to 0.3 so the test subset would be 30% of the dataset. Lastly the parameter "*random_state*" is set to 1, this parameter randomizes data distribution before data is split between training and testing, and the use of an arbitrary integer ensures that the results are always the same, in other words, the results are reproducible. Next, both the training and testing datasets are vectorized [32]. Vectorization means converting text data into numerical features. The vectorization method used is TF-IDF (Term Frequency - Inverse Document Frequency) which evaluates the importance of words by giving words with higher importance a higher weight and a lower weight to common words [6]. Next, a Multinomial Naive Bayes classifier is initialized and trained using the vectorized training data and its accompanying labels, by calling the "*fit*" method. Lastly, predictions are made by calling the "*predict*" function and inputting the vectorized testing data. These predictions are stored in a variable, which can be used to print the metrics, such as the accuracy of these predictions, as well as the confusion matrices.

6.2.4 Results. After all three models (preprocessed, mixed, and non-preprocessed) are trained and tested, it is possible to observe the resulting metrics and the confusion matrices. Metrics from all models can be found in Table 3. From the metrics of the preprocessed model (Table 3) and the associated confusion matrix (Fig 6), it is possible to observe that the model did well at classifying the articles into negative and positive classes, with an accuracy of 89%. [15] and [14] are a good reference point given a similar use of the Naive Bayes algorithm for sentiment analysis of posts on the social media site X (formerly Twitter) and product reviews respectively. The Naive Bayes model in the case of [15] achieved an accuracy of around 85%, while [14] averaged an accuracy of roughly 92%. Even though it might seem that the model in this experiment performs better than in [15] and worse than in [14], this is not entirely the case. In Table 3, the "Support" column represents how many samples from each category have been used for testing. In this case, out of the total 600 samples provided for testing, 301 of them were negative and 299 positive. This distribution is caused by the "*random_state*" parameter in the Python implementation. The results in Table 3 are the outcome of setting "*random_state*" to the value of 1. If the value is changed to 4, the distribution becomes 310 negative and 290 positive samples, which yields an accuracy of 92% for the preprocessed model. On the opposite side of the spectrum, setting the value of "*random_state*" to 2 yields an accuracy of 86% for the preprocessed model with a distribution of 318 negative and 282 positive samples. It is then possible to conclude, that depending on the distribution of the data, the variance of this model can be +/-3%. Consequently, using papers [15] and [14] as a reference we can conclude that the

Table 3. Metrics of the preprocessed, mixed, and non-preprocessed model with random_state = 1. The accuracy of a model is approximated by taking the average of the F1-score of the positive and the negative label

	Precision	Recall	F1-Score	Support
Preprocessed				
Negative	0.98	0.79	0.87	301
Positive	0.82	0.99	0.90	299
Accuracy			0.89	600
Mixed				
Negative	0.85	0.69	0.76	301
Positive	0.74	0.88	0.80	299
Accuracy			0.78	600
Non-Preprocessed				
Negative	0.72	0.72	0.72	301
Positive	0.72	0.72	0.72	299
Accuracy			0.72	600

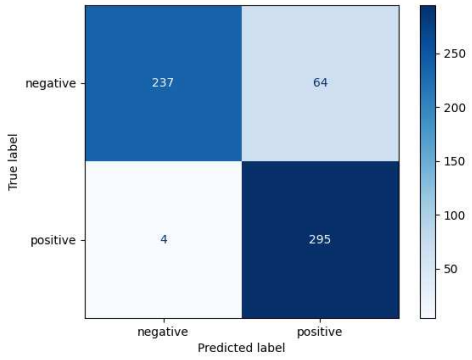

Fig. 6. Confusion matrix of the preprocessed model (random_state = 1)

accuracy of the preprocessed model including its possible deviations fits within the spectrum of possible results of using the Naive Bayes algorithm for sentiment analysis.

The confusion matrix (Fig 6) shows two axes. The X-axis is the predicted label, which is a label the model thinks the article is. The Y-axis is the true label, which is the actual label of the article. From this matrix, it is possible to see that the misclassification of positive articles as negative (bottom left of Fig 6) is a rare occurrence. The recall of the positive label of 99% and the precision of the negative label of 98% also confirm this. Consequently, most misclassifications happened with negative articles being classified as positive (top right of Fig 6). Even though the number of positive and negative samples was equivalent, the dataset was still imbalanced. This can be for example caused by positive words being more common than negative, however, there are other possibilities that are not discussed in this research due to them not being the area of concern. For further research, [12] outlines some of these possibilities.

The mixed model, as outlined in Section 6.2.1, contains 1000 preprocessed and 1000 non-preprocessed samples (500 positive and 500 negative articles per each). Table 3 and Fig 7 show the results of this model. As can be observed from the metrics in Table 3, the

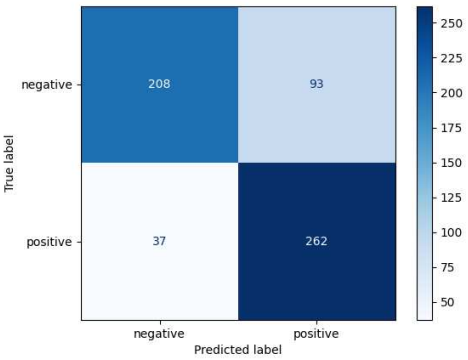

Fig. 7. Confusion matrix of the mixed model (random_state = 1)

accuracy decreased drastically by more than 10% compared to the preprocessed model. This shows that introducing web-scraped data into a dataset of preprocessed data has a negative impact on the performance of the model. From the confusion matrix (Fig 7), it is also possible to observe that the number of misclassifications both in false negatives and false positives rose by approximately the same amount, in this case, 30 samples. Another interesting observation is when the "*random_state*" variable is set to 2. This, in the preprocessed model, caused a drop in accuracy of approximately 3%. In the mixed model, the value of "*random_state*" of 2 again causes the distribution of the samples in the testing subset to be 318 negative and 282 positive, which compared to the preprocessed model has a significantly larger impact, lowering the accuracy to 70%. A confusion matrix for this case (Fig 8) has also been provided and shows that the inclusion of more negative samples than positive ones almost completely removes the model's ability to distinguish whether a negative article is positive or negative. Lastly, the non-

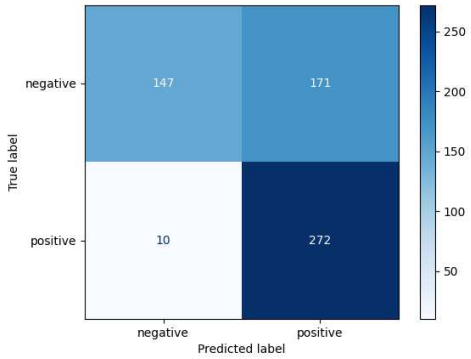

Fig. 8. Confusion matrix of the mixed model (random_state = 2)

preprocessed model is examined using the metrics in Table 3 and Fig 9. This model includes 1000 non-preprocessed positive and 1000 non-preprocessed negative articles. It is possible to observe from

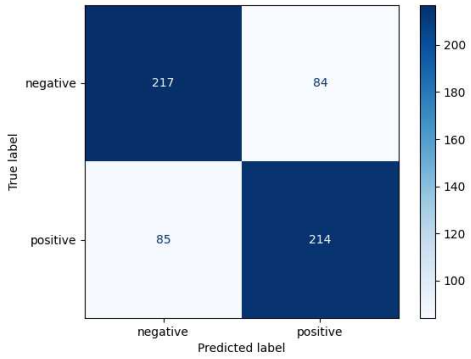

Fig. 9. Confusion matrix of the non-preprocessed model (random_state = 1)

Table 3 that the accuracy is 72% which is lower than both the preprocessed and mixed model. An interesting observation, however, is that the amount of false negatives and false positives is equivalent, at roughly 85. Given that in the preprocessed model, the number of false positives was overwhelmingly higher than false negatives, it is possible to conclude that apart from low accuracy, the model also cannot reflect imbalances correctly. An assumption can be made that this is caused primarily by raw HTML data sharing similarities, such as each HTML document starting with <!DOCTYPE HTML> tag, however, a further investigation is outside of the scope of this research.

6.3 Discussion

The objective of this research has been to investigate the use of web-scraped data in the domain of ML models. This has been achieved to a satisfactory degree, both through the findings in the literature review part, as well as through the experiment. In the literature

review, it was possible to find that web-scraped data can be used for many different use cases, with the only limitation being the presence of the data on the web in a sufficient amount and quality, as well as the content not being restricted either by law or terms and conditions or, as observed in the experiment, through geoblocking as discussed in Subsection 6.2.2. The data that can be extracted can also be varied, even though mostly textual, the context behind it often varies, and because of this, different forms of preprocessing could also be necessary. For instance, using [13] and [25] as an example, [13] extracted among other things the source code of smart Ethereum contracts, while [25] extracted hashtags from Instagram captions. While both of these, the source code and the hashtags, are textual information, they had to be preprocessed in a different way, for example, [13] mentioned replacing logic operators such as "=" with the word "equals", while [25] mentioned that simply removing the hashtag (#) symbol increased the performance of the model. To facilitate web-scraping, it has been found in the literature review that Python is a popular choice, however, it is certainly not the only choice, and the choice of software available for the purposes of web-scraping is varied and catered to both more and less technologically skilled users. The literature review also shows the variety of models that can be used alongside web-scraped data. It is important to acknowledge however that the choice of ML models is not directly linked to web-scraped data. Given that web-scraped data in most cases provides textual data, it is not different from retrieving textual data through other ways such as surveys or interview transcripts. This means that the choice of ML models depends significantly on the format of the data, though the source of the data can also influence the model selection and preprocessing requirements. This collection of information answers the first research sub-question (1).

The experiment provided a more technical and concrete perspective on the use of web-scraped data for training ML models. It helped outline the step-by-step process of training an ML model using web-scraped data, the tools necessary, and an example implementation in Python. Primarily, however, the experiment brought to attention the importance of preprocessing web-scraped data. As could have been observed, the accuracy of the model decreased with the introduction of non-preprocessed web-scraped data. This could have been observed best from the results of the mixed model, which delivered results consistent with the imbalance of the original dataset. The non-preprocessed model delivered the worst results that were both inaccurate as well as did not reflect the imbalance of the original dataset. Thus, to answer the second research sub-question (2), the introduction of non-preprocessed web-scraped data into a dataset used for training an ML model has a negative impact on the performance of the model.

To answer the research question at hand, we combine the knowledge gained from both parts of the research. From the literature review part, it was possible to observe that the authors successfully utilized web-scraped data as a complete data source for training their respective ML models. The authors did not outline whether there have been issues with web-scraped data, however, each article outlined some necessary preprocessing steps to achieve desired results. Thus, it is possible to conclude from the literature review that web-scraped data can be used as a data source for training ML models, and in some application cases such as user reviews on e-commerce websites, web scraping might be a requirement [3, 28], however, the literature also indicates that a more important aspect is how the data is handled after web scraping. The experiment utilized solely web-scraped data, both preprocessed and non-preprocessed, and it showed not only that web-scraped data is adequate for training an ML model but also exacerbated the importance of data preprocessing when it comes to web-scraped data. From these findings, the answer to the research question is that web-scraped data can be used as a complete or complementary source of data for training ML models, however, web-scraping-specific data preprocessing is required to achieve high performance in these models.

7 CONCLUSION

In conclusion, this paper provides a deeper look into the use of web-scraped data to train ML models. The paper outlines that web-scraped data is a sufficient data source for ML models, provided that the data is preprocessed properly. Through a literature review, web scraping is found to be able to extract data from the web that is rich in variety, using many different tools such as Python, and be used to train a wide range of models, whether the data is textual or visual. More importantly, through an experiment carried out in this research, the necessity for proper preprocessing of web-scraped data has been identified. This necessity is also supported by the examined articles in the literature review, where the authors carried out some form of preprocessing before the data was ready to be used for training. Overall, web scraping, as well as web-scraped data, bring unique challenges into the world of data gathering and training of ML models, however, a deeper understanding of this combination can enhance one's ability to rapidly gather quality data from the web.

7.1 Limitations and Future Research

Due to the scope, both parts of this research experienced some limitations. In the literature review, the number of articles examined is low compared to traditional literature reviews. Currently, the literature review serves more as a brief look into the practice of using web-scraped data as a source for ML models. Future research into this topic, if a literature review is to be part of it, should examine a larger number of articles to add more information to the main points of the literature review carried out in this research, namely the type of data, use of web-scraping software, and use of ML models. Alternatively, a more beneficial addition to this topic would be examining the approaches to preprocessing data in a larger number of papers with a focus on different fields. It has already been mentioned that data had to be preprocessed in its own unique way in each paper examined in this research. What further research in this direction could yield is a systematic overview of how web-scraped data needs to be preprocessed per its source of origin, i.e. online reviews, comments on videos, news articles, images from social media posts, etc.

When it comes to the limitations of the experiment, a clear limitation is the use of only one ML algorithm. As discussed in the literature review, practitioners often use the ensemble method to combine output from different ML algorithms. This has not been

utilized in this research. For further research, a solution would be to examine other papers conducting sentiment analysis using ML for inspiration regarding other suitable ML algorithms, such as in [15] where a decision tree algorithm has been used or [14] where SVM was utilized. The effect of introducing non-preprocessed data on the speed of training and testing the model has also not been recorded in this research. Recording the speed could serve as another metric to compare the effect of non-preprocessed and preprocessed data. Lastly, this research should inspire further research into preprocessing techniques for web-scraped data. The experiment in this research showed the impact of non-preprocessed web-scraped data on the performance of ML models as being negative, and the articles examined in this research also showed unique ways of preprocessing web-scraped data depending on the source. This means that the area of preprocessing web-scraped data could be a potential topic for further examination, and could also serve a practical purpose for practitioners looking to utilize web-scraped data. This examination could include running experiments where individual steps of preprocessing web-scraped data are applied to observe their individual impact on the performance of different ML models and track which preprocessing steps carry the largest effect on the performance.

8 APPENDIX

8.1 AI Notice

During the preparation of this work, the author used:
- Grammarly in order to fix spelling mistakes and sentence formation
- ChatGPT to fact-check simplified explanations of complex topics written by the author
After using this tool/service, the author reviewed and edited the content as needed and takes full responsibility for the content of the work.

REFERENCES

[1] Basant Agarwal and Namita Mittal. 2016. Prominent Feature Extraction for Sentiment Analysis. *Prominent Feature Extraction for Sentiment Analysis* i (2016).

[2] Jiafu An, Wenzhi Ding, and Chen Lin. 2023. ChatGPT: tackle the growing carbon footprint of generative AI. , 586 pages. https://doi.org/10.1038/d41586-023-00843-2

[3] Rodrigo Barbado, Oscar Araque, and Carlos A. Iglesias. 2019. A framework for fake review detection in online consumer electronics retailers. *Information Processing and Management* 56, 4 (2019). https://doi.org/10.1016/j.ipm.2019.03.002

[4] Anish. Chapagain. 2019. *Hands-on web scraping with Python perform advanced scraping operations using various Python libraries and tools such as Selenium, Regex, and others.*

[5] Weili Chen, Zibin Zheng, Jiahui Cui, Edith Ngai, Peilin Zheng, and Yuren Zhou. 2018. Detecting ponzi schemes on ethereum: Towards healthier blockchain technology. In *The Web Conference 2018 - Proceedings of the World Wide Web Conference, WWW 2018*. https://doi.org/10.1145/3178876.3186046

[6] Hans Christian, Mikhael Pramodana Agus, and Derwin Suhartono. 2016. Single Document Automatic Text Summarization using Term Frequency-Inverse Document Frequency (TF-IDF). *ComTech: Computer, Mathematics and Engineering Applications* 7, 4 (2016). https://doi.org/10.21512/comtech.v7i4.3746

[7] David Gourley and Brian Totty. 2002. *HTTP: The Definitive Guide.*

[8] Andrea De Mauro, Marco Greco, Michele Grimaldi, and Paavo Ritala. 2018. Human resources for Big Data professions: A systematic classification of job roles and required skill sets. *Information Processing and Management* 54, 5 (2018). https://doi.org/10.1016/j.ipm.2017.05.004

[9] Wolfgang Ertel. 2017. *Introduction to Artificial Intelligence (Undergraduate Topics in Computer Science).*

[10] A. Famili, Wei Min Shen, Richard Weber, and Evangelos Simoudis. 1997. Data preprocessing and intelligent data analysis. *Intelligent Data Analysis* 1, 1 (1997). https://doi.org/10.3233/IDA-1997-1102

[11] Yoav Freund and Robert E. Schapire. 1997. A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. *J. Comput. System Sci.* 55, 1 (1997). https://doi.org/10.1006/jcss.1997.1504

[12] Doaa Mohey El Din Mohamed Hussein. 2018. A survey on sentiment analysis challenges. *Journal of King Saud University - Engineering Sciences* 30, 4 (2018). https://doi.org/10.1016/j.jksues.2016.04.002

[13] Giacomo Ibba, Giuseppe Antonio Pierro, and Marco Di Francesco. 2021. Evaluating Machine-Learning Techniques for Detecting Smart Ponzi Schemes. In *Proceedings - 2021 IEEE/ACM 4th International Workshop on Emerging Trends in Software Engineering for Blockchain, WETSEB 2021.* https://doi.org/10.1109/WETSEB52558.2021.00012

[14] Rajkumar S. Jagdale, Vishal S. Shirsat, and Sachin N. Deshmukh. 2019. Sentiment analysis on product reviews using machine learning techniques. In *Advances in Intelligent Systems and Computing*, Vol. 768. https://doi.org/10.1007/978-981-13-0617-4{_}61

[15] Anuja P. Jain and Padma Dandannavar. 2017. Application of machine learning techniques to sentiment analysis. In *Proceedings of the 2016 2nd International Conference on Applied and Theoretical Computing and Communication Technology, iCATccT 2016.* https://doi.org/10.1109/ICATCCT.2016.7912076

[16] Jarmul Katharine and Lawson Richard. 2017. *Python Web Scraping* (2 ed.). Packt Publishing Ltd.

[17] Moaiad Ahmad Khder. 2021. Web scraping or web crawling: State of art, techniques, approaches and application. *International Journal of Advances in Soft Computing and its Applications* 13, 3 (2021). https://doi.org/10.15849/ijasca.211128.11

[18] Inna Kolyshkina and Simeon Simoff. 2019. Interpretability of machine learning solutions in industrial decision engineering. In *Communications in Computer and Information Science*, Vol. 1127 CCIS. https://doi.org/10.1007/978-981-15-1699-3{_}13

[19] Dimitrios Kouzis-Loukas. 2016. *Learning Scrapy.* Vol. 1.

[20] Sascha Kraus, Matthias Breier, Weng Marc Lim, Marina Dabić, Satish Kumar, Dominik Kanbach, Debmalya Mukherjee, Vincenzo Corvello, Juan Piñeiro-Chousa, Eric Liguori, Daniel Palacios-Marqués, Francesco Schiavone, Alberto Ferraris, Cristina Fernandes, and João J. Ferreira. 2022. Literature reviews as independent studies: guidelines for academic practice. *Review of Managerial Science* 16, 8 (2022). https://doi.org/10.1007/s11846-022-00588-8

[21] Jörg Krause. 2016. *Introducing Web Development.* https://doi.org/10.1007/978-1-4842-2499-1

[22] Vlad Krotov, Leigh Johnson, and Leiser Silva. 2020. Tutorial: Legality and ethics of web scraping. *Communications of the Association for Information Systems* 47, 1 (2020). https://doi.org/10.17705/1CAIS.04724

[23] Kati Kuusinen. 2015. Software developers as users: Developer experience of a cross-platform integrated development environment. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Vol. 9459. https://doi.org/10.1007/978-3-319-26844-6{_}40

[24] Y. LeCun, Y. Bengio, and G. Hinton. 2015. Deep learning. *Nature* 521 (5 2015), 436–444.

[25] Jiawei Li, Qing Xu, Neal Shah, and Tim K. Mackey. 2019. A machine learning approach for the detection and characterization of illicit drug dealers on instagram: Model evaluation study. *Journal of Medical Internet Research* 21, 6 (2019). https://doi.org/10.2196/13803

[26] Steven Lorla. 2020. TextBlob Documentation. *TextBlob* (2020).

[27] Alexander Selvikvåg Lundervold and Arvid Lundervold. 2019. An overview of deep learning in medical imaging focusing on MRI. https://doi.org/10.1016/j.zemedi.2018.11.002

[28] Alhassan Mabrouk, Rebeca P.Díaz Redondo, and Mohammed Kayed. 2021. Seopinion: Summarization and exploration of opinion from e-commerce websites. *Sensors (Switzerland)* 21, 2 (2021). https://doi.org/10.3390/s21020636

[29] Ryan Mitchell. 2015. *Web Scraping with Python: Collecting Data from the Modern Web.* Vol. 53.

[30] James A. Muir and Paul C. Van Oorschot. 2009. Internet geolocation: Evasion and counterevasion. *Comput. Surveys* 42, 1 (2009). https://doi.org/10.1145/1592451.1592455

[31] Issam El Naqa and Martin J. Murphy. 2022. What Are Machine and Deep Learning? In *Machine and Deep Learning in Oncology, Medical Physics and Radiology, Second Edition.* https://doi.org/10.1007/978-3-030-83047-2{_}1

[32] Fabio Nelli. 2023. Machine Learning with scikit-learn. In *Python Data Analytics: With Pandas, NumPy, and Matplotlib*. Apress, Berkeley, CA, 259–287. https://doi.org/10.1007/978-1-4842-9532-8{_}8

[33] Sebastian Raschka and Vahid Mirjalili. 2019. *Python Machine Learning. Machine Learning and Deep Learning with Python, scikit-learn, and TensorFlow 2.* Number January 2010.

[34] RFC-6585. 2012. HTTP – Additional HTTP Status Codes. *IETF* (2012).

[35] RFC-7725. 2016. An HTTP status code to report legal obstacles.

[36] Stuart Russell and Peter Norivg. 2021. Artificial Intelligence: A Modern Approach (Global Edition). *Artificial Intelligence: A Modern Approach* (2021).

[37] Pramila P. Shinde and Seema Shah. 2018. A Review of Machine Learning and Deep Learning Applications. In *Proceedings - 2018 4th International Conference on Computing, Communication Control and Automation, ICCUBEA 2018.* https://doi.org/10.1109/ICCUBEA.2018.8697857

[38] Vidhi Singrodia, Anirban Mitra, and Subrata Paul. 2019. A Review on Web Scrapping and its Applications. In *2019 International Conference on Computer Communication and Informatics, ICCCI 2019*. https://doi.org/10.1109/ICCCI.2019.8821809

[39] Scm C.M.D.S. Sirisuriya. 2023. Importance of Web Scraping as a Data Source for Machine Learning Algorithms - Review. In *2023 IEEE 17th International Conference on Industrial and Information Systems, ICIIS 2023 - Proceedings*. https://doi.org/10.1109/ICIIS58898.2023.10253502

[40] Marketa Trimble. 2016. Geoblocking, Technical Standards and the Law. *Scholarly Works* (2016).

[41] Zhi Hua Zhou. 2012. *Ensemble methods: Foundations and algorithms*. https://doi.org/10.1201/b12207

[42] Zhi Hua Zhou. 2021. *Machine Learning*. https://doi.org/10.1007/978-981-15-1967-3