American University of Armenia

College of Science and Engineering

Capstone project

Travel Advisor

Authors: Liana Avagyan, Artyom Ghazaryan
Supervisor: Anna Tshngryan

Spring 2024

Abstract

Traveling has always been popular among people from all around the world. Sometimes people face hardships traveling to their desired places because they lack knowledge about where to travel to. They do not have time to look through different cities and parts of the world, review the feedback from other travelers in order to be able make decisions. Travel Advisor allows future travelers to get a prediction for their perfect destination to travel to having a simple input of several words. Being trained on about 20,000 distinct and unique text data points, the Travel Advisor is capable to predict a travel destination from 12 different famous cities among travelers. Travel Advisor predicts based on a text input of activities, words that a future traveler inputs. Usually the input includes activities that travelers would like to do while being on a vacation.

Contents

Chapter 1

Introduction

Travel Advisor is a data-driven project that predicts destinations for future travelers with an input of just a few words. It uses data points scraped from the TripAdvisor's website and predicts one of the top 12 famous cities from the same website. The project utilizes data collected from 12 top cities from TripAdvisor's website list on Traveler's Choice. Project dataset is collected by scraping the titles from the website "attractions" section for each of the cities. Datasets for each of the cities include around 2000 values. The project will be able to predict a destination from 12 different popular cities. The travel agent's final product is a tool, where users input text and the travel agent gives them the city recommendation based on the input.

Data cleaning process includes several steps:

a. Removing punctuation and numbers
b. Lowercasing everything
c. Removing stop words
d. Creating a document term matrix grouped by city
e. Visualizing the most frequent words

1. Different vectorization strategies are used in the project, after which visualizations are created with word clouds. Several functions are created to make the pre-processing steps easier as well as make word clouds with each city's country form on the map.
2. Modeling part of the project includes:
    a. Baseline Naive Bayes Model with 4 different iterations
    b. Random Forest Model (with and without lemmatization)

Chapter 2

Introduction to Data

The data used for this project is fully scraped from the TripAdvisor's website. When visiting the website and navigating to the top attractions list for each of the 12 cities, the website pulls up the most famous attractions to visit for the specific inputted city. Several pages of attractions per each of the cities are being scraped to collect the attractions as data points.
Table 2.1 shows the list of all 12 cities where the top attractions are collected from.

Top Attractions	**City, Country**
`Attraction list`	London, UK
`Attraction list`	Paris, France
`Attraction list`	Crete, Greece
`Attraction list`	Bali, Indonesia
`Attraction list`	Rome, Italy
`Attraction list`	Phuket, Thailand
`Attraction list`	Sicily, Italy
`Attraction list`	Majorca, Balearic Islands
`Attraction list`	Barcelona, Spain
`Attraction list`	Istanbul, Turkey
`Attraction list`	Dubai, United Arab Emirates
`Attraction list`	Vienna, Austria

Table 2.1: For each of the 12 cities the number of top attractions is different, but the function is created to pull up values from the same number of pages of each city attractions.

The table shows the number of top attractions collected/scraped for each of the cities. The function used for scraping data points is universal for all the cities.

2.1 Exploratory Data Analysis

The project's most relevant part is to collect relevant data and to figure out which models can be tested on the scraped data points.

N	City	Country	Total Number of Attractions Collected
1	London	United Kingdom	1620
2	Paris	France	1620
3	Crete	Greece	950
4	Bali	Indonesia	1620
5	Rome	Italy	1560
6	Phuket	Italy	1511
7	Sicily	Italy	1560
8	Majorca	Balearic Islands	1137
9	Barcelona	Spain	1620
10	Istanbul	Turkey	1680
11	Dubai	United Arab Emirates	1590
12	Vienna	Austria	1200

The final dataset combines all the scraped data points from the 12 cities and here are some features of the final dataset: total number of data points is 17668 and total number of duplicated values is 1200.

Attractions per city for the initial datasets are shown below.

To visualize the dataset containing data points from all the 12 cities, Figure 2.1 shows the information. It also displays the class weights imbalance between cities.

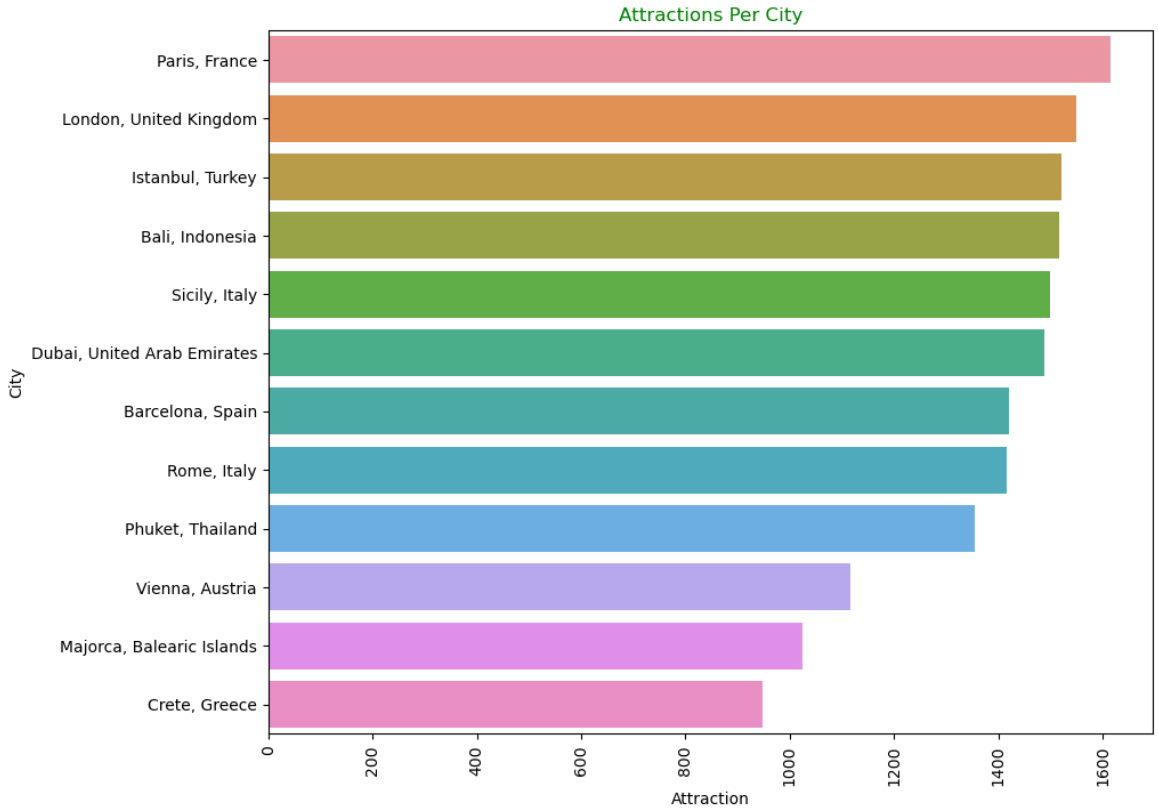

Figure 2.1: The figure visualizes the data points collected for each of the cities.

The next steps that have been done include the processes of:

1. Data Cleaning 2. Preprocessing 3. Deeper Exploration.

These processes include removing punctuation and numbers, turning all the data points into lower cased ones, removing stop words from the dataset, creating a document term matrix grouped by city, lemmatizing, performing count vectorization, performing tf-idf vectorization, working with bi-grams, visualizing most frequent words, visualizing data points with word clouds, bar plots and histograms.

Figure 2.2 below is showing a single word cloud created for Bali, Indonesia with TF-IDF Vectorization. The same word clouds are created for each and every city.

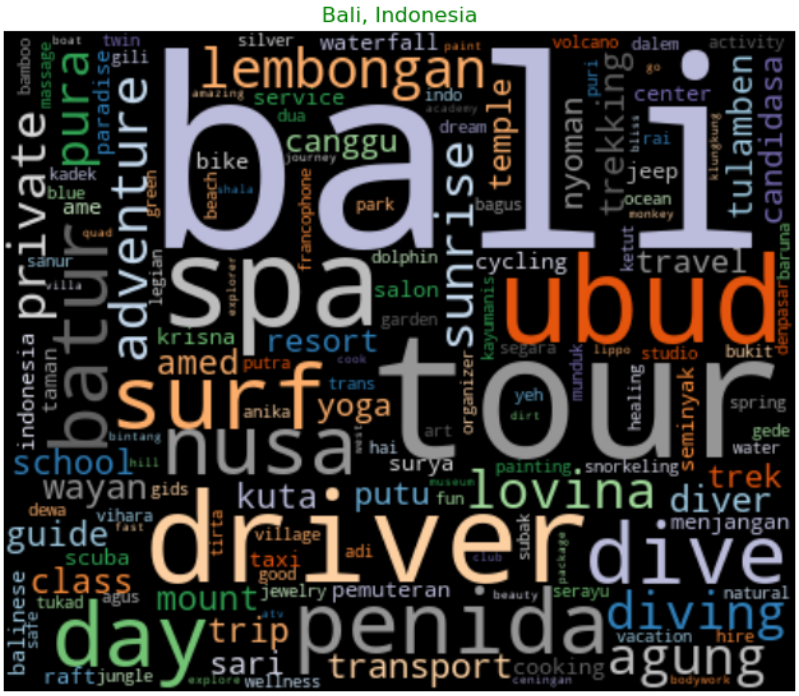

Figure 2.2: The word cloud representing the figure above is for highlighting our finding that tf-idf is a better vectorization technique compared with count vectorization, since it is finding more words that are unique to the cities. This is why it is going to be used while modeling in order to better predict the cities.

Figure 2.3: The final word cloud created for Dubai, United Arab Emirates is shown below. The word cloud has the form of Burj Al Arab Jumeirah, Dubai, a palace recognised by Lartisien for its excellence and one of the landmarks of Dubia.

Figure 2.3 For creating the figure, we are removing noise from the data. Since there are still lots of words in the word clouds like ’private’, ’airport’ and ’transfer’, we want to try to take those attractions for airport transfers out because they are causing noise in the data. After final pre processing steps, the final word clouds are being created for each of the 12 cities. The final figures (word clouds) for each of the 12 cities are created with the forms of famous landmarks of those cities. As an example, the word cloud of Paris, France is created with the form of The Eiffel Tower, the most famous landmark of Paris. Some of the word clouds are created with the maps of the cities to better illustrate the visualization.

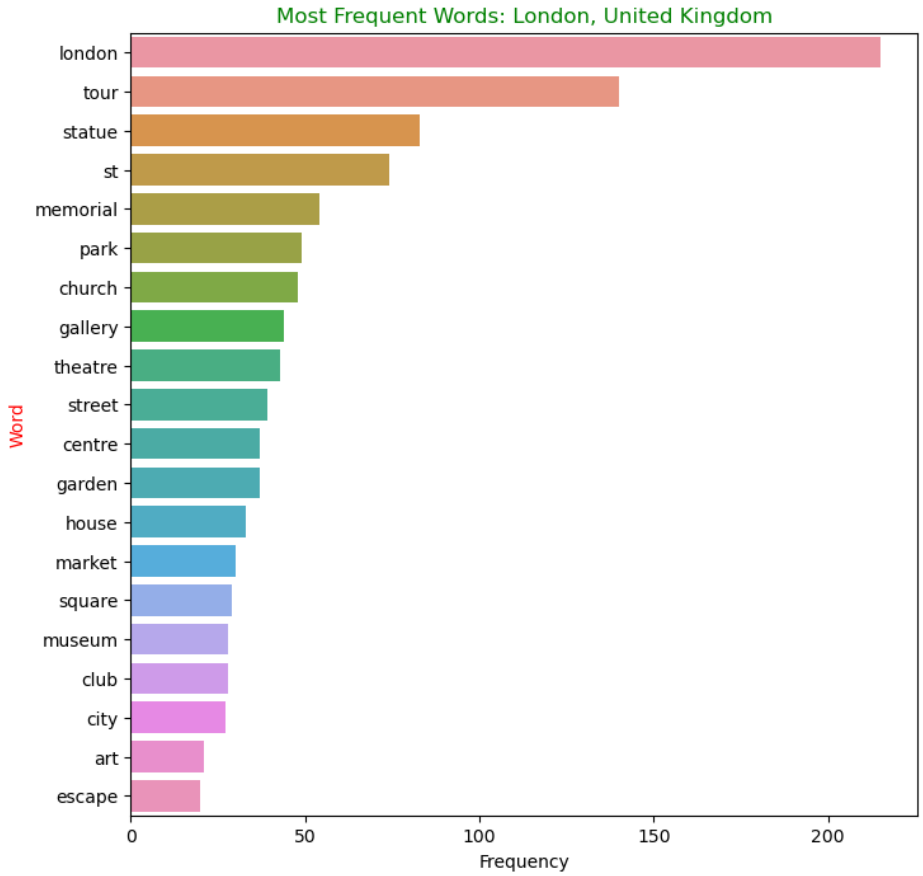

Figure 2.4: The figure is the result of making a graph for the most frequent words occurring in the dataset of London, United Kingdom. The figure shows the most frequent words with their frequencies.

From Figure 2.4, it can be concluded that the word ”London” appears in the dataset for London city with the most frequency. This is true for most of the cities and it is expected since the names of attractions include city names in their text. This figure illustrates just the example of London, but 11 more such graphs are created for each of the cities. For further steps of modeling, the names of each of the cities are removed from the datasets to make sure the city names do not affect the modeling processes.

Chapter 3

Literature Review

According to Will Koehrsen, Senior Machine Learning Engineer, usually, when training a model on a dataset, we need to randomly split the dataset into two separate sets: training and testing sets. After having the real answers for the test set, we can go ahead and compare the predictions with the actual values to understand how accurately the model performs. This same procedure is followed in the project, using the steps followed in the source and with difference of using TFIDF features with the classifier [Koe17].

According to the source on Naïve Bayes Classification in Python and its author Shuvrajyoti Debroy, a Data Science graduate from the Maryland University, the Naive Bayes is a classification technique which is based on Bayes' theorem. Understanding the basis of Bayes' theorem helps to understand the model used in the project. The source notes that according to this classification algorithm, we are trying to find out the class which is holding the most likely appearing set of features and attributes [Deb23]. According to the same source, the training set of the dataset is used to train the model. The model is tested afterwards, but this time on the test split set to understand its performance and accuracy. This is a relevant point, as when the model is trained and tested on the exact same data, it is possible to face over-fit on the data and the model may perform poorly on new, not seen data [Deb23]. The Naive Bayes Classification used with TfidfVectorizer and stop words used in the project supports the ideas presented in the source.

Jason Brownlee, PhD in Machine Learning, mentions that instead of trying to figure out the probabilities of each attribute, we make an assumption of them to be conditionally independent having the class value

[Bro19]. This assumption is important when implementing different iterations with Naive Bayes Classifier. In addition, another source implementing ML | Naive Bayes Scratch Implementation using Python mentions that Naive Bayes shows to be a very basic and effective algorithm working perfectly with classification tasks [Nbg]. This is true for the project as well since using some techniques as class imbalance, count vectorization and bi-grams, the model iterations show to be simple and efficient.

According to Priyanka Charak, who presents a full guide of Selenium and Selenium WebDriver with Python for Web Automation Testing, Selenium is considered to be one of the most famous and flexible web automation testing techniques [Cha23]. In addition, the author notes that Selenium WebDriver, which is the used in the project for collecting data from TripAdvisor's website, is the most preferable open-source tool because it supports many programming languages.

The source presenting the background and implementation of Vectorization, Multinomial Naive Bayes Classifier and Evaluation acts as a full guide on model building, vectorization, evaluation of Naive Bayes taking into consideration the class imbalance [Vec]. In the project, a separate iteration of using class weights for improving the class imbalance is performed.

Web Program Testing Using Selenium Python: Best Practices and Effective Approaches is a source provides with a real life example of implementing Selenium with WebDriver [Sel]. It illustrates the use cases, challenges and benefits of using this web program testing approach.

In addition, Baiju Muthukadan, in their official documentation and full guide presents a thorough explanation of the process, using Selenium, WebDriver API, Alerts, different browser WebDriver options and many more sections [Mut]. They support the technique which is used in this project for scraping the TripAdvisor's website and getting the necessary data points.

Chapter 4

Applied Models

Initially, the project implements preprocessing of the data to make sure models are applied on clean data. The preprocessing steps include:

- Removing duplicates from the data points.
- Removing punctuation and numbers
- Making the data points lowercase
- Using lemmatization for preprocessing the train and test dataset splits
- Removing stop words (Natural Language Processing (NLP) is used with its aspect of handling stop words with the help of Natural Language Toolkit. The reason why stop words are removed is that words such as 'I', 'and', 'a', and 'the' and more do not really introduce relevant information about the topic. By getting rid of these words from the dataset, it is easier to identify unique and relevant words)
- Adding city names into the list of stop words (this is done to make sure any occurrence of the city name in the attractions column is removed, so models work correctly)
- Visualizing the most frequent words in the datasets

4.1 Baseline Naive Bayes Model

Naive Bayes methods represent a collection of supervised algorithms that are based on using Bayes' theorem along with a "naive" as-

sumption about the conditional independence between each pair of features that are given the value of the class variable. Bayes' theorem states the following relationship, given class variable y and dependent feature vector x_1 through x_n:

$$P(y|x_1,...,x_n) = \frac{P(y) \times P(x_1,...,x_n|y)}{P(x_1,...,x_n)} \tag{4.1}$$

Using the naive conditional independence assumption that

$$P(x_i|y,x_1,...,x_{i-1},x_{i+1},...,x_n) = P(x_i|y) \tag{4.2}$$

Starting with the models, the initial model iteration used in the project is the Baseline Naive Bayes Model. This model ensures the dataset is preprocessed, uses train/test splits of the dataset, uses TfidfVectorizer and eliminates stop words including city names to make predictions.

4.2 Multinomial Naive Bayes Model with class imbalance

Moving forward, the next iteration used in the project presents the Naive Bayes Iteration 2, which uses class weights to improve the available class imbalance. After computing the sample weight, the model is implemented on the already created train and test splits of the datasets.

4.3 Multinomial Naive Bayes Model with TF-IDF vectorization

The next model iteration used in the project presents the Naive Bayes Iteration 3. This iteration is done after finding out that in many of the cities' attractions list, the name of the city is already included. Considering this as a potential issue, the project introduces new text for this model, which does not include the city names. The same list of stop words is used for this iteration as in the first iteration.

4.4 Multinomial Naive Bayes Model with Count Vectorization

The next model iteration used in the project is the Multinomial Naive Bayes with Count Vectorization, sample weights and the same list of stop words.

4.5 Multinomial Naive Bayes Model with Bi-Grams

The next model iteration used in the project is the Multinomial Naive Bayes Using Bi-Grams and the list of stop words.

4.6 Random Forest Model

Iteration 6 uses the Random Forest Model for predictions along with Tfidf Vectorizer and initial train and test datasets. Here the project also explores the feature importance of features. Figuring out the top 20 features from the full dataset, it becomes obvious that a lot of city-specific words, such as 'eiffel', the name of a tower in Paris, or general words coming from different city datasets such as 'tour', 'travel', 'guide' and more are among these features. In future, these kind of repeating words can be found and eliminated.

4.7 Multinomial Naive Bayes without lemmatization

The final iteration uses the third iteration but without lemmatization of the train and test dataset splits. This iteration is performed as lemmatization version of the model is more usually more computationally expensive than using the lemmatization technique.

Chapter 5

Results

Accuracy is the first metric to be discussed, and, the simplest one. It can be calculated by the following formula:

$$\text{Accuracy} = \frac{\text{\#correctly classified items}}{\text{\#all classified items}} \tag{5.1}$$

According to the scikit-learn.org website about accuracy and F1 scores [F1s], ”F1-score shows the harmonic mean of precision and recall, where F1 score reaches its best value at 1 and worst score at 0”. The formula for the F1 score is:

$$\text{F1} = \frac{2 * \text{TP}}{2 * \text{TP} + \text{FP} + \text{FN}} \tag{5.2}$$

Where TP is the number of true positives, FN is the number of false negatives, and FP is the number of false positives.

5.1 Baseline Naive Bayes Model Results

The initial model iteration used in the project called Baseline Naive Bayes Model, results in training accuracy of 0.816 and testing accuracy of 0.593. Training F1 score of the model shows the value of 0.815 and the testing F1 shows 0.585. Thus, this model performs well showing around 59% accuracy score, however, the 3 classes with the lowest accuracy and F1 scores are the cities Majorca, Rome and Vienna. In the next model, the project ensures that the class imbalance is not affecting the model. This issue is fixed by using class weights in the next iteration.

5.2 Multinomial Naive Bayes Model with class imbalance

The next model iteration used in the project called Multinomial Naive Bayes Model with class imbalance, results in training accuracy of 0.826 and testing accuracy of 0.594. Training F1 score of the model shows the value of 0.826 and the testing F1 shows 0.592. Thus, this model performs better showing around 60% accuracy score, however, in many of these cities' attractions text, the name of the city is included. This may become an issue in the future. This issue is fixed by taking city names out in the next iteration.

5.3 Multinomial Naive Bayes Model with TF-IDF vectorization

The next model iteration used in the project called Multinomial Naive Bayes Model without relevant stop words, results in training accuracy of 0.826 and testing accuracy of 0.594. Training F1 score of the iteration shows the value of 0.826 and the testing F1 shows 0.592. Thus, this iteration performs similar to the previous iteration.

5.4 Multinomial Naive Bayes Model with Count Vectorization

The next model iteration used in the project called Multinomial Naive Bayes Model with Count Vectorization, results in training accuracy of 0.786 and testing accuracy of 0.581. Training F1 score of the iteration shows the value of 0.787 and the testing F1 shows 0.577. Thus, with count vectorization, the scores are very similar, but still lower than with TF-IDF vectorization, therefore we will keep the TF-IDF vectorization strategy.

5.5 Multinomial Naive Bayes Model with Bi-Grams

The next model iteration used in the project called Multinomial Naive Bayes Model with Bi-Grams, results in training accuracy of 0.849 and testing accuracy of 0.397. Training F1 score of the iteration shows the value of 0.863 and the testing F1 shows 0.429. This iteration did well for the training accuracy, but not good for the testing accuracy. Thus, the TF-IDF vectorization remains the best vectorization strategy for this dataset.

5.6 Random Forest Model

The next model iteration used in the project called Random Forest Model, results in training accuracy of 0.973 and testing accuracy of 0.592 (showing just 59 % accuracy). Training F1 score of the iteration shows the value of 0.973 and the testing F1 shows 0.597. This model still does not perform very well with the test set. It is important to note that random forest model has 2 major flaws that will affect this model for its specific use-case:

1. It is more computationally expensive than Naive Bayes models (meaning takes longer to train and predict).
2. It uses a greedy algorithm, so it often favors the bigger class.

Thus, iteration 3 remains the best model.

5.7 Multinomial Naive Bayes without lemmatization

The final iteration uses the third iteration, using the cleaned text data without lemmatizing it. Ultimately, this model performs very similar to the lemmatized version of it. But here the model performs with accuracy score of more than 60 %. Here the project shows training accuracy of 0.843 (84.3 %), testing accuracy 0.608 (60.8 %). Training F1 score for the iteration displays 0.843 (84.3 %) and testing F1 is 0.606 (60.6 %).

Chapter 6

Conclusion

The final model is the Multinomial Naive Bayes Model with TF-IDF vectorization and without lemmatization, which provides the best accuracy. It is important to note that there are some small differences that contribute to choosing Multinomial Naive Bayes Model without lemmatization version as the final model:

- The test accuracy and F1 scores are higher for this model compared to the lemmatized version.
- Lemmatization version is more computationally expensive than getting rid of the lemmatization.

Therefore, the final model is the iteration 3 (Multinomial Naive Bayes Model with TF-IDF vectorization) without lemmatizing the text. This can predict a destination with 60% accuracy and around 60% F1 score. The text data based on which thus model was implemented is not lemmatized, but is lowercased with stopwords and city names removed from it.

Chapter 7

Future Work

The project plans include some future work to make the best use of Travel Advisor. The main points of adding future work include:

- Include even more data of more than 12 popular cities from the TripAdvisor's website and have the models work on more data points. This is for the thorough application of the project including more than 50 different famous cities to visit across the world, so people have better advice
- Include reviews left by people about attraction places of cities and take keywords from these reviews to add them as features of the cities. This is for users who look for places to visit and can rely on others' review about the city being for example dangerous or safe. This way the users will get more information

Bibliography

[Koe17] Will Koehrsen. *Random Forest in Python*. 2017. URL: https://towardsdatascience.com/random-forest-in-python-24d0893d51c0.

[Vec] *Vectorization, Multinomial Naive Bayes Classifier and Evaluation*. 2017. URL: https://www.ritchieng.com/machine-learning-multinomial-naive-bayes-vectorization/.

[F1s] *Vectorization, Multinomial Naive Bayes Classifier and Evaluation*. 2017. URL: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html.

[Bro19] Jason Brownlee. *Naive Bayes Classifier From Scratch in Python*. 2019. URL: https://machinelearningmastery.com/naive-bayes-classifier-scratch-python/.

[Cha23] Priyanka Charak. *Selenium WebDriver with Python for Web Automation Testing: Getting Started*. 2023. URL: https://www.pcloudy.com/blogs/selenium-webdriver-with-python-for-web-automation-testing/.

[Deb23] Shuvrajyoti Debroy. *Naïve Bayes Classification in Python*. 2023. URL: https://medium.com/@shuv.sdr/naïve-bayes-classification-in-python-f869c2e0dbf1.

[Nbg] *ML | Naive Bayes Scratch Implementation using Python*. 2023. URL: https://www.geeksforgeeks.org/ml-naive-bayes-scratch-implementation-using-python/.

[Sel] "Web Program Testing Using Selenium Python: Best Practices and Effective Approaches". In: *researchgate.net* (2024).

[Mut] Baiju Muthukadan. *Selenium with Python*. URL: https://selenium-python.readthedocs.io.