A Final report for a dissertation that will be submitted in partial fulfilment of a University of Greenwich Master's Degree

A MACHINE LEARNING SOLUTION FOR PRODUCT MATCHING IN E-COMMERCE

Name: Hassaan Abdullah

Student ID: 001203332

Program of Study: MSc. Data Science

Final Report

Submission Date: 01/17/2023

Supervisor: Dr. Timothy Reis

Table of Contents

List of Tables

List of Figures

Acknowledgement

First and foremost, I am extremely grateful to my supervisor, Dr. Timothy Reis for their invaluable advice, continuous support, and patience during my MSc dissertation. Their immense knowledge and plentiful experience have encouraged me in all the time of my academic research and daily life. I would also like to thank my fellow data scientists for their technical support in my study. I would like to thank all of my friends for their kind help and support that have made my study and life in the UK a wonderful time. Finally, I would like to express my gratitude to my mother (Tahira Deep), my father (Dr. Noor Ahmad) and my siblings (Dr. Uswa Ahmad, Dr. Saud Ahmad, Bisma Ahmad, and Shiza Ahmad). Without their tremendous guidance, understanding and encouragement throughout my life, it would have been impossible for me to complete my studies.

Abstract

Online shopping has grown in popularity in recent years, resulting in increased wealth on online platforms. Generally, many retailers offer the same items on various platforms. Thus, product matching across different platforms became a basic requirement for both buyers and sellers. However, recognizing similar items across numerous platforms is challenging since product descriptions often vary. In this project, I have implemented various machine learning models to solve this problem. I have used product title, product brand and product description, which are the most widely used attributes to advertise products on different online platforms. Experiments were carried out on a real-world data set containing thousands of products from online e-commerce platforms. Lastly, models were evaluated on the basis of confusion matrix, mean squared error and accuracy score.

Chapter 1: Introduction

1.1 Background

Product matching is a process of matching identical products from different internet sources using machine learning technologies. Retailers are compelled to have a better understanding of their goods, pricing, features, and more in a competitive environment where the customer's buying habits have changed significantly. Additionally, merchants are finding it increasingly difficult to maintain the accuracy of their product data due to growing data quantities. Retailers find it challenging to monitor the quality of their data since issues with data flowing in from several vendors and suppliers occur all too frequently. This is why the future of retail operations lies in automated, machine learning-based product data management systems.

Retailers often manually match items using characteristics like stock keeping units, product name, and product brand. This is not only time-consuming, but it is also ineffective since product data is unstructured and with variable information. Brands want their items to be universally displayed under the appropriate category, yet merchants frequently fail to do so or neglect to highlight a crucial aspect of the product. Retailers must extract characteristics, remove duplicates, standardize information, and categorize it in order to offer the best possible customer experience when matching products. Without these safeguards, a user may, for example, see a brand-new laptop when searching for a secondhand laptop. Retailers may categorize variants using product matching, preventing them from becoming a front-end issue. Insufficiently specified categories and qualities may lead to a user purchasing the incorrect item.

Product matching now extends beyond the fundamentals. Retailers can now clean up, standardize, and combine disparate information from many sources to generate unified product information thanks to modern technologies. The hardest part of product matching is extracting and categorizing attributes while making sure that even the smallest differences are found.

(Li et al., 2019) An innovative neural matching model is used in this study to address the product matching problem. This strategy considers the two types of descriptions—product names and attributes—that are frequently published on online platforms. On two online e-commerce platforms, a real-world data collection with hundreds of goods is used to run experiments. The experimental findings demonstrate that the strategy greatly outperforms the most recent matching algorithms by using the product information included in both titles and characteristics. Users enter product queries to look for product entities in this approach. Product name, product attributes are common examples of product entities. Between user queries and product entities, matching is done. These methods for textual query and product matching are used. However, because the matching process takes place between users' queries and product entities rather than between product entities, they are unable to directly address the issue of product matching. When items are matched across platforms, things get considerably more challenging. Numerous approaches, including record linking, duplication detection, and entity resolution, have been suggested to address this issue. These ideas, however, have not yet been put into practice.

(More, 2020) In this study, a method is developed that relies on the product data considered to be the most reliable—title, description, images, and price—in order to make a matching assessment. The system consists of a number of components, any, or all of which may be utilized based on the particular use cases. Same title is a measure of the degree of similarity between two product titles. Same product image is the measure of the degree of similarity between two product photos. Key attribute extraction is identifying important characteristics for each product, such as its brand, color, make, model, and price, then calculating the variations in the values of those characteristics. Based on the results of these individual components, matched and mismatched decisions are made.

Both consumers and retailers can benefit from this product matching application. Product matching gives consumers the chance to compare all of their alternatives and select the best deal. Let's say a consumer wishes to purchase a work desk. The price will be the primary justification for choosing a certain deal. They visit a certain marketplace and begin weighing their possibilities. Meanwhile, product matching can help a retailer in creating a reasonable pricing strategy and maintain business competitiveness. Businesses learn more about their rivals and how to differentiate their items when they compare product costs across several platforms. An illustration of how a retailer might stand out is the unique offer where the work desk and the desk clock are sold together.

1.2 Problem area that the project addresses

Online shopping is becoming more and more common as Internet and mobile technology advances. Large-scale offline transactions have been migrated online, resulting in the success of online marketplaces like Amazon, eBay, ASOS, and Sports Direct. It is typical for vendors to sell their items on many internet platforms at once since different buyers prefer various platforms. Comparing costs, features, and user reviews helps shoppers decide whether to buy a product from a certain platform. For instance, when identical items are offered on several platforms, customers often choose the least expensive option. They like the product given by a vendor with better service when both price and quality are comparable. In the meanwhile, platform management as well as sellers can modify pricing and sales strategies by comparing their items to those of rivals. Therefore, it is important to research the issue of distinguishing between similar items on various platforms.

Products are often characterized on online e-commerce platforms by their product names and product features. The most important details of a product are typically contained in the title, which is typically a short piece of free language. More specific information about items may be specified using attributes, which are often presented as name-value pairs from catalogue data or from structured tables on web sites. Still, it is challenging to determine whether two items are identical or not given their titles and attributes. One issue is that names and characteristics are formed differently, and items are described in a variety of ways depending on seller needs. Synonyms present another challenge in the product matching process. In addition to appearing in product names, synonyms also show up in product characteristics. As a result, it is quite difficult to match items while taking both titles and qualities into account. Through this application, I have tried to solve this problem.

1.3 Project aim and objectives

In this project, my aim is to present a machine learning solution for product matching in e-commerce. The application will handle the product matching as a binary classification problem where it will predict if the two of the given products are same (matching) or different (not matching). It depends on the several features of the product such as product's title, product's description, product's category, product's brand, sizes, colors etc. Product matching enables price optimization, item-to-item product linkage, search aggregation, and product suggestions, as well as increased catalogue quality. Building a product matching pipeline of high-quality that is scalable and has appropriate match performance guarantees is a complex task. A lot of research is done to completely define the scope of the project. Lastly, if I talk about building the application. I have used python programing language along with JSON documents which will contain the data of the products. Binary classification algorithms are used to classify the products. I have used various algorithms and implemented whichever gives the highest accuracy. To help me achieve my targets and successfully complete my project, I have taken the courses of Applied Machine Learning, which will help me to implement different machine learning algorithms and choose whichever gives highest accuracy, big data will help me to handle huge amount of data, clean it and use it for training and test of my machine learning model. Lastly, graph and modern databases will help me to iterate e-commerce databases and JSON document of given products.

1.4 Requirement for the proposed system

Product matching is the process of finding products deals from many websites that pertain to the same physical product. Product offers and deals are advertised on the web using product title, textual description, category and other attributes like product image, size, color, price, country of origin etc. However, product matching is difficult due to the syntactic, structural, and semantic heterogeneity across the offerings.

1.4.1 Dataset:

Product matching data is available on Kaggle. Data is available in the form of JSON documents. Data has 6 attributes which contains information regarding a product. ID describes the identity of the product, product category, title of the product, description of the product, brand, and price of the product.

1.4.2 Data pre-processing:

Real-world data contains noise, errors, partial information, missing values, and other forms of inconsistencies. In order to achieve the maximum accuracy of a machine learning model and to get most out of the data we need data pre-processing. For that purpose, I performed following steps:

- Remove duplicate values.
- Remove tangential data.
- Standardize capitalization.
- Data type conversion.
- Data formatting.
- Error fixing.
- Handle missing values.

1.4.3 Model Architecture:

The major goal is to build a machine learning model that has the ability to make precise predictions if the two of the given products are identical (matching) or different (not matching). As part of this research, the objective is to create a model that is able to take into consideration a variety of factors that are associated with the prediction of product matching.

1.4.4 Model Selection:

Fitting models is fairly a simple process, but the real difficulty in using machine learning is choosing which models to use. Given the unexplained variability in the data, the incompleteness of the data sample, and the restrictions of each unique model type, all models have some prediction inaccuracy. As a result, the idea of a perfect or optimal model is useless. Instead, we should look for a model which makes the most accurate prediction. The suggested approach employs not one, not two, but three distinct types of machine learning models in order to generate accurate prediction: logistic regression, support vector machine, and random forest. The model with the highest degree of precision is selected.

1.4.5 Prediction:

Predictions can be made using the selected model. Regardless of the success of the predictions made for each class or label, an accuracy score is used to determine a model's ability to accurately categories data items. It provides us with an intuitive sense of whether the data we currently have is appropriate for our classification problem. Furthermore, I have used mean squared error and confusion matrix to evaluate the model.

1.4.6 Visualization:

Python includes a number of useful tools for plotting data, but Matplotlib is by far the most well-known. A wonderful utility that produces a far more pleasing plot is Seaborn, which also makes use of matplotlib as its foundation. When all of the data is stored in a pandas dataframe, there are several more comparable types of plots that are also accessible. I have used matplotlib to visualize the data of product matching.

Chapter 2: Literature Review

2.1 Data pre-processing:

(Gonzalez Zelaya, 2019) As computers are expected to make accurate predictions that can have a significant influence on people's lives, the concerns of transparency and fairness in how machine learning (ML) algorithms function have become increasingly important. While transparency relates to understanding how predictions are made, fairness refers to comparable people receiving similar results. There have been several formal definitions of fairness offered. In general, decisions can be class assignments, grades, or ranking spots on a list. These two notions are intrinsically tied, because understanding how a prediction was made is essential for determining whether it was fair or not. The way data is preprocessed for a machine learning model has an impact on its ability to make accurate predictions.

Data preprocessing involves getting raw data ready to be used with a machine learning model. The first and most crucial step in creating a machine learning model is this one. We don't always find clean, prepared data while building a machine learning model. Furthermore, data must be cleaned up and formatted before being used for training purposes. It contains following steps:

2.1.1 Getting a dataset:

The most important thing I needed to build a machine learning model was a dataset, because a machine learning model is completely reliant on data. The dataset is the collection of data for a certain topic in the right format. I have used Product matching data which is available on Kaggle. Data is available in the form of JSON documents. It has 6 attributes which contain information regarding a product. ID describes the identity of the product, product category, title of the product, description of the product, brand, and price of the product.

2.1.2 Data Cleaning:

Data cleaning involves removal of missing values, removal of redundant data, standardize capitalization, conversion of data types, error handling and consistently formatting the data. There are several ways to handle incomplete or missing values in a dataset. In the first method used to deal with missing values. We simply remove the individual row or column that contains null data. However, this method is inefficient, and deleting data may result in information loss, resulting in an inaccurate output. A more efficient way is by calculating the mean. I computed the mean of the column or row containing any missing values and substitute it in place of the missing value. Next up, I removed duplicate data from the dataset using python built-in function. Data is converted to appropriate data types and formatted according to the need of machine learning model.

2.2 Splitting data into training and testing sets:

(Vrigazova, 2021) This study looks at what ratio should be used to divide the dataset into the training and testing sets so that the machine learning model can work well with other prediction techniques in terms of accuracy. To assess the performance of various classification algorithms, training and testing data is split into different proportions and utilized using the following resampling methods: the bootstrap, leave one out cross validation method, tenfold cross validation method, and random repeated train/test split. The logistic regression, random forest, and support vector machine are among the categorization methods employed. The findings imply that utilizing a different test set structure (for example 30/70, 20/80) might further enhance the bootstrap performance when used to logistic regression and random forest. Tenfold cross validation method with a 70/30 train/test splitting ratio is recommended for support vector machine. The bootstrap can increase the classification problem's accuracy depending on the features and preliminary transformations of the variables.

2.2.1 Training set:

It is the collection of data which is used to train a machine learning model and teach it to discover the hidden attributes and patterns within the data. The machine learning model is given training data periodically, and the model continues to learn the features of the data. A wide range of data should be included in the training dataset so that the model could well be trained in all conditions and anticipate any future data sample that has not yet been observed.

2.2.2 Testing set:

The test set is a distinct collection of data that is used to test the model after it has been trained. It gives impartial final model performance metrics like accuracy, precision, and so on.

2.3 Feature Selection:

(Cai et al., 2018) Machine learning and data mining academics and engineers struggle with high-dimensional data analysis. Feature selection is an effective solution to this problem since it eliminates unnecessary and redundant data, reducing computation time, improving prediction accuracy, and allowing for a better understanding of the learning model or data. We discuss numerous commonly used feature selection assessment metrics in this study before examining supervised, unsupervised, and semi-supervised feature selection methodologies utilized in machine learning tasks such as binary classification issues and clustering tasks. Feature selection is the process of constructing a subset from an initial feature set based on a feature selection criterion that selects the relevant properties of the dataset. It helps to compress the data processing scale by removing duplicate and useless information. A sophisticated feature selection method can improve model accuracy, save computing time, and simplify forecast results. Furthermore, two strategies for lowering dimensionality are feature selection and feature extraction. Unlike feature selection, feature extraction frequently necessitates the transformation of the original data into features with high pattern recognition capacity, whereas the original data may be viewed as features with low recognition ability.

According to the type of training data, feature selection methods are characterized as supervised, unsupervised, or semi-supervised (labelled, unlabeled, or partially labelled). A unified framework for selecting features in supervised, unsupervised, and semi-supervised contexts. Based on how they interact with learning methods, feature selection approaches are classified as filter, wrapper, or embedding models. Depending on the evaluation criterion, feature selection procedures may be produced from correlation, Euclidean distance, consistency, dependence, and information measure. Based on search strategies, feature selection approaches may be divided into forward increase, backward deletion, random, and hybrid models. Feature selection approaches are classified into feature rank (weighting) and subset selection models based on the kind of output.

(Khalid et al., 2014) High-dimensional data contains elements that may be misleading, redundant, or unnecessary, which widens the search area, making it difficult to evaluate the data further and impeding the learning process. Feature subset selection refers to the process of selecting the best characteristics from among all the features that may be used to differentiate classes. Filters, wrappers, and embedded/hybrid techniques are the three categories of feature selection methods. Because the feature selection procedure is tailored for the classifier to be employed, wrappers techniques perform better than filter methods. Wrapper techniques, on the other hand, are expensive to utilize for large feature spaces due to high computational costs and the requirement that each feature set be assessed using the trained classifier, which makes the feature selection process lengthy. In comparison to wrapper techniques, filter methods are quicker and have lower computing costs, but they are less reliable at classifying data and are better suited for high-dimensional data sets. Recently, hybrid and embedded solutions that combine the best elements of the wrappers and filters approaches have been created. A hybrid strategy makes use of both an independent test and the feature subset's performance assessment function. Two other categories for filtering techniques are feature weighting algorithms and subset search algorithms. The relevance of each feature to the target idea is weighted and ranked by feature weighting algorithms.

The process of features extraction involves changing the original features in order to produce more important new features. The following definition of feature extraction is given: "Feature extraction is commonly used to denote the generation of linear combinations of continuous features with excellent discriminating power between classes." In the study of neural networks and other disciplines like artificial intelligence, finding a suitable representation of multivariate data is a key challenge. By considering each variable in the feature space as a linear combination of the original input variable, features extraction may be used in this case to reduce complexity and simplify the data representation. Principle Component Analysis PCA, developed by Karl, is the most well-known and often applied feature extraction technique. There have been several principal component analysis variations proposed. Principle component analysis is a straightforward non-parametric technique used to isolate the most important details from a collection of duplicated or noisy data. principle component analysis is a linear data transformation that optimizes information while minimizing redundancy (measured by covariance) (measured through the variance).

To study the interaction between various dimensionality reduction strategies, including feature subset selection using information gain (IG) and wrapper methods, and feature extraction using several flavors of principle component analysis methods, and the effects of these methods on classification accuracy, empirical tests were performed on two distinct types of datasets. Results have demonstrated that the kind of data has a significant impact on feature extraction (transformation) using principle component analysis. Method for choosing features for both types of data, wrapper exhibits a reasonable impact on classification accuracy compared to information gain. The outcomes of the experiment highlight the significance of dimensionality reduction. In comparison to feature extraction approaches, wrappers methods for feature selection frequently yield the smallest feature subsets with relatively competitive classification accuracy. Wrappers, however, cost a lot more to compute than traditional approaches for feature extraction.

2.4 Model Selection:

(Schönleber, 2021) In the context of machine learning, the term "model selection" can apply to various degrees of abstraction. For instance, we may be looking for the best hyperparameters for a certain machine learning strategy. The learning technique's hyperparameters must be defined before a model can be fitted. On the other hand, model parameters are parameters that come as an output of the fit. For instance, in a logistic regression model, the fitted model coefficients are model parameters, but the regularization type and strength are hyperparameters that must be defined before fitting. The performance of a model on a set of data may be significantly impacted by choosing the right hyperparameters. Another scenario is that we want to select from a list of available machine learning techniques the best learning method (together with its associated "optimal" hyperparameters). This is referred to as algorithm selection in the following. When faced with a classification challenge, we could question if a logistic regression model or a random forest classifier produces the greatest classification performance on the particular job. Before delving into the specifics of various techniques for model selection and when to utilize them, there is "one more thing" to cover: model evaluation. Model evaluation is to determine the generalization error of the selected model, or how well the selected model performs on unidentified data. Naturally, an effective machine learning model must also perform well on data that has not been observed during training; otherwise, the model would merely memorize the training data. Therefore, before putting a model into use, we should be pretty certain that its performance won't suffer when exposed to the most recent data. But why is it important to distinguish between model evaluation and model selection? The problem is overfitting. Why? Let's do a series of experiments. Assume you are asked to select the best-performing classifier from a selection of black-box classifiers. Since every classifier generates a (fixed) sequence of zeros and ones, they are all worthless. You test each classifier using your data and find that, on average, they correctly classify 50% of the instances. By accident, one classifier performs 8% better on the data than the rest. As a rough indication of your classifier's performance on unknown data, you utilize this performance (i.e., as an estimate of the generalization error). You claim to have found a classifier that performs better than random guessing. If you had used a really independent test set to estimate the generalization error, you would have quickly discovered the fraud! We need completely independent data to calculate a model's generalization error in order to avoid such issues.

(Ding et al., 2018) To learn from and forecast data, typical statistical inference or machine-learning procedures employ fitted parametric or nonparametric models (in a broad sense). However, no model is universally applicable to all facts and objectives. A bad choice of model or approach may result in wholly noise-based discoveries, seriously incorrect conclusions, or unimpressive forecasting skills. As a result, selecting the best model from a pool of candidates (known as the model class) is a critical stage in any data analysis. In other terms, model selection is the process of selecting a statistical model from a model class given a set of data. To select a good model, we need to look into these parameters. Variables for the basis terms of linear regression, including polynomials, splines, or wavelets in function estimation, the number of components in a machine learning model, the best parametric family among several candidates, the number of change points in time series models, and the number of neurons and layers in neural networks.

In order to learn from data, there are two major goals. One is for interpreting the nature of the data, comprehending the data-generation process, and making scientific discoveries. A scientist may use the data, for instance, to back up a physical model or pinpoint genes that unmistakably encourage the early beginning of a disease. Predictive learning, or using data to statistically represent future observations, is another goal of data learning. In this case, the data scientist may not be concerned with establishing a precise probabilistic representation of the data. Of course, it's possible to have dual interests. Model selection can go in one of two directions—model selection for inference or model selection for prediction—to correspond with the two distinct aims. The goal of the first is to choose the optimal model for the data, which should give scientists useful information and context for interpreting sources of uncertainty. The second is picking a model as a vehicle to get to a model or approach that performs at its best. The chosen model must not be very sensitive to the sample size in order to achieve the first aim. In the latter case, the chosen model could only be the lucky victor over a few close rivals, but the prediction performance might still be (almost) the greatest that could be.

2.4.1 Logistic Regression

(Kambria, 2019) Logistic regression is a binary classification algorithm which is when the target variables consist of categorical values. When the data in question has a binary output, or when it falls into one of two classifications or is either 0 or 1, logistic regression is most frequently used. Similar to how linear regression assumes that the data follows a linear distribution, logistic regression models the data using the sigmoid function. When plotted on a graph, the sigmoid function/logistic function resembles an “S" shaped curve. It takes numbers between 0 and 1 and squishes them towards the top and bottom edges, marking them as 0 or 1.

The equation of sigmoid function is as follows:

$$y = \frac{1}{(1 - e^{-x})}$$

It is critical to remember that logistic regression should only be used when the target variables are discrete and that it should not be utilized if the target variable contains a range of continuous values. Logistic regression does not become a classification method unless a decision threshold is included in the equation. A key component of logistic regression is the threshold value, which is established by the classification problem itself. Determining the threshold value is heavily influenced by the accuracy and recall levels. Although it would be ideal, accuracy and recall are rarely equal to 1. In the context of a Precision-Recall tradeoff, the following arguments are used to determine the threshold:

2.4.1.1 Low Precision/High Recall:

When we want to reduce the proportion of false negatives without necessarily reducing the number of false positives, we select a decision value with a low Precision or a high Recall.

2.4.1.2 High Precision/Low Recall:

When we want to cut down on false positives without necessarily cutting down on false negatives, we select a decision value with a high Precision value and a low Recall value.

2.4.2 Support Vector Machine (SVM):

(Gandhi, 2018) Support Vector Machine (SVM) is a supervised machine learning method appropriate for regression and classification problems. However, categorization problems make considerable use of it. Each piece of data is represented as a point in an n-dimensional space, where n is the total number of features, and each feature's value corresponds to a certain position in the support vector machine. Then, classification is achieved by identifying the hyperplane that most effectively separates the two classes.

Decision boundaries known as hyperplanes help classify data elements. The data points on either side of the hyperplane belong to different classes. Additionally, the quantity of features affects how big the hyperplane is. The hyperplane is essentially a line when there are two input characteristics. The hyperplane changes into a two-dimensional plane when there are three input features. Support vectors are closer-to-the-hyperplane data points that influence the position and orientation of the hyperplane. We increase the margin of the classifier by utilizing these support vectors. Changing the support vectors will change where the hyperplane is located. These are the ideas that will help us create our support vector machine. In logistic regression, the output of the linear function was compressed between 0 and 1 using the sigmoid function. The value is labelled 1 if it exceeds a threshold value (0.5), else it is labelled 0. If the result of the linear function is more than 1, I used support vector machine to classify it into one class, and if it is less than 1, I used a different class. Because the SVM threshold values are changed to 1 and -1, I obtained the margin-effective reinforcing range of values [-1,1].

2.4.3 Random Forest

(Yiu, 2021) To discuss random forest, first we have to understand decision trees. A decision tree can be used in decision analysis to describe decisions and decision making graphically and explicitly. It employs a decision-tree-like model, as the name implies. Though a typical technique in data mining for developing a strategy to achieve a specific goal. The random forest technique is a classification system comprised of a large number of decision trees. It utilizes bagging and feature randomization while building each individual tree in an attempt to produce an uncorrelated forest of trees whose prediction by committee is more accurate than that of any individual tree. The forest's trees, and more crucially, their projections, must be uncorrelated or have minimal correlations with one another. While the algorithm attempts to build these low correlations for us using feature randomization, the features, and hyper-parameters we choose will also have an influence on the final correlations.

2.5 Model evaluation

(Saravanan, 2021) One of the most difficult aspects of Machine Learning is evaluating the Model's performance. So, how should the success of a machine learning model be measured? How would we determine when to call it quits on the training and evaluation? And what metric should be used to assess the model? In this article, the author has attempted to respond to these questions.

The quality of a statistical or machine learning model is measured using evaluation metrics. Building machine learning models is based on the notion of constructive feedback. The performance of a model is explained by evaluation metrics. The capacity of assessment metrics to discern between model outputs is a crucial feature. Evaluation allows you to learn more about your Model. It indicates if your model memorized or learned. This is critical if your model has only remembered rather than learned, as the model works well on known data, rendering the model wasteful. To ensure that your model learns, employ various assessment criteria to evaluate the model. Because a model may perform well using one assessment metric while performing poorly using another evaluation metric. Evaluation metrics are crucial in verifying that your model is working correctly and appropriately.

Overfitting and underfitting are the two most common reasons for poor machine learning algorithm performance. Overfitting happens when a model excels at fitting a certain set of known data and may therefore struggle to fit additional unknown data or make accurate predictions about the future. Underfitting happens when the model fails to capture the fundamental structure of the data accurately.

2.5.1 Evaluation matrix

There are several metrics for classification, regression, ranking, clustering, topic modelling, and so forth. The following are some of the metrics:

2.5.1.1 Confusion matrix:

A table known as a confusion matrix is frequently used to illustrate how a classification model, performed on a set of test data for which the real values were known. Confusion matrix classifies the predicted values into four classes. True positives, values which are predicted correctly in positive class. True negative, values which are correctly predicted in negative class. False positive, values which are wrongly classified in positive class and False negatives are the values which are wrongly classified in negative class.

2.5.1.2 Mean Squared Error:

It is probably the most frequently utilized indicator for regression tasks. It is the average of squared error between the expected values and actual values. A better value for the MSE, which is always non-negative and measures an estimator's quality, is one that is closer to zero. Mean squared error can be calculated using the following equation.

$$\mathrm{MSE} = \frac{1}{\mathrm{n}}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)$$

Here n is number of data points, Y_i is the label value and $\hat{Y}_i$ is the value predicted by our model. MSE only indicates how well a regression line resembles a set of points. This is accomplished by squaring the distances between the points and the regression line (also known as the "errors"). To simplify the intricacy with negative signs, squaring is essential. The model may need to be more precise in order to reduce MSE, which would bring the model closer to the real data. The least square approach, which assesses if a linear regression model is adequate for a bivariate dataset, is one example of a linear regression that makes use of this technique. However, this method has limitations relating to the data's known distribution.

2.5.1.3 Mean Absolute Error:

It calculates the average error size over a group of forecasts without taking direction into account. All individual differences are equally weighted in the test sample's average of the absolute disparities between prediction and observation. MAE can be calculated using the following equation as well:

$$\mathrm{MAE} = \frac{\sum_{i=1}^{n}|y_i - x_i|}{n}$$

Here n is number of data points, y_i is value predicted by our model and x_i is the label value. Compared to MSE, MAE is reported to be more resilient to outliers. The fundamental reason is because the outliers—which often have bigger mistakes than other samples—get more attention and dominance in the final error and have an influence on the model parameters because of the way MSE squares the errors.

Chapter 3: Requirement analysis and System design

3.1 Requirement Analysis:

3.1.1 Problem statement:

Product names and product attributes are frequently used to describe products on online e-commerce platforms. Product titles, which are usually a brief bit of free language, contain the most significant data about a product. Product attributes, which are commonly supplied as name-value pairs from catalogue data or structured tables on web sites, can be used to specify more particular information about products.

Nonetheless, given their titles and qualities, it is difficult to establish if two objects are the same or not. One difficulty is that names and attributes are constructed differently, and products are described differently based on the seller's demands. Another problem in the product matching process is synonyms. Synonyms can be found in product attributes as well as product names. As a result, matching objects while considering both titles and attributes is tough. I attempted to address this problem with this software.

3.1.2 Functional Requirements:

It refers to product features or functions that developers must create in order for consumers to complete their duties. As a result, it's critical to make them apparent to both the development team and the stakeholders. In general, functional requirements explain how a system behaves under certain situations. For this project, I have implemented following functional requirements:

3.1.2.1 Data Collection:

Product matching data is available on Kaggle. Data is available in the form of JSON documents. Data has 6 attributes which contains information regarding a product. ID describes the identity of the product, product category, title of the product, description of the product, brand, and price of the product.

3.1.2.2 Data pre-processing:

It involves getting raw data ready to be used with a machine learning model. The first and most crucial step in creating a machine learning model is this one. We don't always find clean, prepared data while building a machine learning model. Furthermore, data must be cleaned up and formatted before being used in any way.

3.1.2.3 Training a machine learning model:

Divide the dataset into the training and testing sets so that the machine learning model can be trained and work well with other prediction techniques in terms of accuracy. To assess the performance of various classification algorithms, different train and test split proportions are utilized. Model with the maximum accuracy score is selected.

3.1.2.4 Model Evaluation:

The quality of a statistical or machine learning model is measured using evaluation metrics. Building machine learning models is based on the notion of constructive feedback. The performance of a model is explained by evaluation metrics. The capacity of assessment metrics to discern between model outputs is a crucial feature. Evaluation allows you to learn more about your Model. It indicates if your model memorized or learned.

3.1.3 Non-functional Requirements:

(Altexsoft, 2020) They are a collection of specifications that outline the system's operational capabilities and limits, as well as their attempts to improve its functioning. These are the specifications that define how well it will function, such as performance, interoperability, scalability, recoverability, etc. For this project I have met the following non-requirements, which were very helpful in running the application successfully.

3.1.3.1 Performance:

It is a quality attribute that describes the system's response to various user inputs. Poor performance results in poor user experience. When the system is overwhelmed, it also jeopardizes its safety.

3.1.3.2 Scalability:

In order for the system to grow without deteriorating performance, it must be scalable. More users will be serviced, more data will be processed, and more transactions will be completed as a result. Scalability impacts both hardware and software. Expanding RAM, servers, or storage capacity, for example, may improve scalability. On the other hand, you may improve algorithms, compress data, and so on.

3.1.3.3 Reliability:

Reliability represents the likelihood that the software will run without interruption for a certain period of time. Programming errors, hardware issues, and problems with other system components all degrade reliability. To evaluate software dependability, compute the proportion of tasks completed correctly or measure the average time the system functions before failing.

3.1.3.4 Security and privacy:

When leveraging large data to help machine learning, efforts have been made to address privacy problems. I attempted to include privacy-preserving protocols in various machine learning methodologies that I have implemented, while openly acknowledging the trade-off in terms of algorithm speed when rewriting privacy strategies.

Chapter 4: Implementation

4.1 Methodology:

4.1.1 Data Analysis and preprocessing:

This section describes how data is preprocessed and analyzed with the help of python libraries such as pandas, numpy, seaborn etc. For the implementation of my project, I have used a dataset of 1500 products.

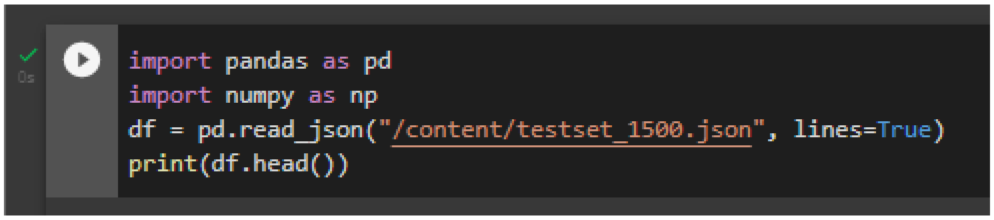
The original dataset is split into two subsets (left and right). Each subset contains information regarding the products, having 6 columns each:

Left subset of the data	Right subset of the data	Description
id_left	id_right	ID of the product
category_left	category_right	Category of the product
title_left	title_right	Title of the product
description_left	description_right	Description of the product
brand_left	brand_right	Brand of the product
price_left	price_right	Price of the product

Table 1: Data subsets

4.1.1.1 Reading the data:

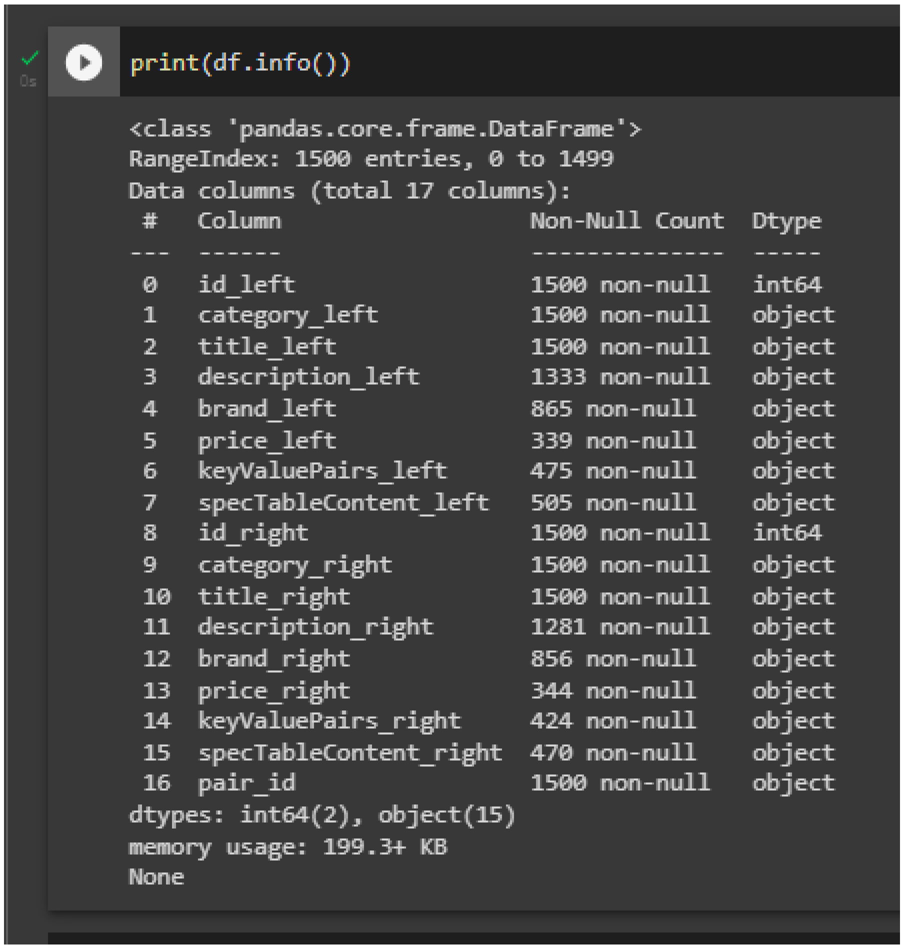
Data is available in the form of JSON documents. Data has 6 attributes which contains information regarding a product. ID describes the identity of the product, product category, title of the product, description of the product, brand, and price of the product. I have used pandas and numpy libraries to read the dataset and stored it in a dataframe. Below is the screenshot of reading the dataset:



```
import pandas as pd
import numpy as np
df = pd.read_json("/content/testset_1500.json", lines=True)
print(df.head())
```

Figure 1: Reading the data

To explore the dataset, I have used info() function which provides the information about the dataframe. The data includes the number of columns, column names, column data types, memory use, range index, and cell count for each column (non-null values). Through this function we can easily identify inconsistencies in our data using cell count of each column.



```
print(df.info())
```

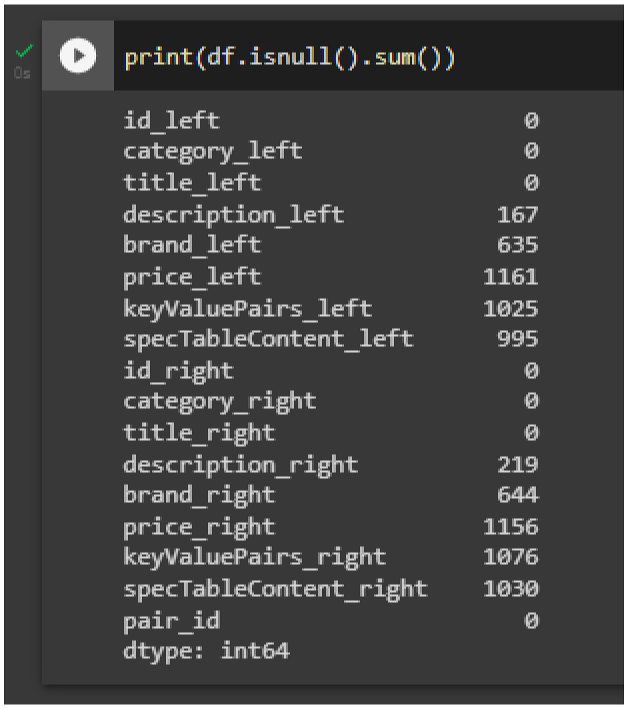
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1500 entries, 0 to 1499
Data columns (total 17 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   id_left                 1500 non-null   int64
 1   category_left           1500 non-null   object
 2   title_left              1500 non-null   object
 3   description_left        1333 non-null   object
 4   brand_left              865 non-null    object
 5   price_left              339 non-null    object
 6   keyValuePairs_left      475 non-null    object
 7   specTableContent_left   505 non-null    object
 8   id_right                1500 non-null   int64
 9   category_right          1500 non-null   object
 10  title_right             1500 non-null   object
 11  description_right       1281 non-null   object
 12  brand_right             856 non-null    object
 13  price_right             344 non-null    object
 14  keyValuePairs_right     424 non-null    object
 15  specTableContent_right  470 non-null    object
 16  pair_id                 1500 non-null   object
dtypes: int64(2), object(15)
memory usage: 199.3+ KB
None
```

Figure 2: Data information

4.1.1.2 Data Cleaning:

Data cleaning is the act of repairing or deleting inaccurate, erroneous, or unneeded data from a data set. To expand on this fundamental description, data cleaning, also known as data scrubbing, and data preprocessing, is the process of converting dirty, possibly problematic data into clean data.

I used the isnull() method to find the missing value. It finds missing values in a dataset. This function takes a scalar or array-like object as input and returns NaN in numeric arrays, None or NaN in object arrays, and NaT in datetimelike objects. Our result is a collection of Boolean values. The list can provide us with a lot of information. The first inquiry concerns the location of the missing data; any 'True' reading beneath a column indicates that there is missing data in that column's category for that data file.

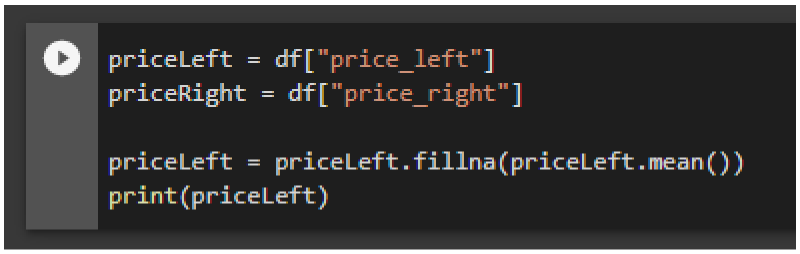


```
print(df.isnull().sum())

id_left                      0
category_left                0
title_left                   0
description_left           167
brand_left                 635
price_left                1161
keyValuePairs_left        1025
specTableContent_left      995
id_right                     0
category_right               0
title_right                  0
description_right          219
brand_right                644
price_right               1156
keyValuePairs_right       1076
specTableContent_right    1030
pair_id                      0
dtype: int64
```

Figure 3: Finding missing values

After determining the missing values, the data cleaning process starts. I have used mean imputation in which missing values are replaced by the mean values of the entire column. Below is the screenshot of implementation:



```
priceLeft = df["price_left"]
priceRight = df["price_right"]

priceLeft = priceLeft.fillna(priceLeft.mean())
print(priceLeft)
```

Figure 4: Data imputation

Remaining data is cleaned manually and using various python functions such as dropna().

4.1.2 Exploratory Data Analysis:

Data preprocessing is already done on the data which includes, data cleaning, removing all the missing and invalid values. Data is stored in two dataframes i.e., dataset_left and dataset_right.

Now that the data has been pre-processed, I started data analysis. First, I looked into the categories of product which are available in the left subset of the data:

- In total six product categories are available: computer and accessories, video games, office products, other electronics, camera and photo, luggage, and travel gear.
- Computer and accessories is the biggest category which covers more than 80% of the products available in the dataset.
- Video games is the second most popular category having 12 products and office products is the third category in line having 11 products. Categories and number of products in each category are depicted in the following bar chart:

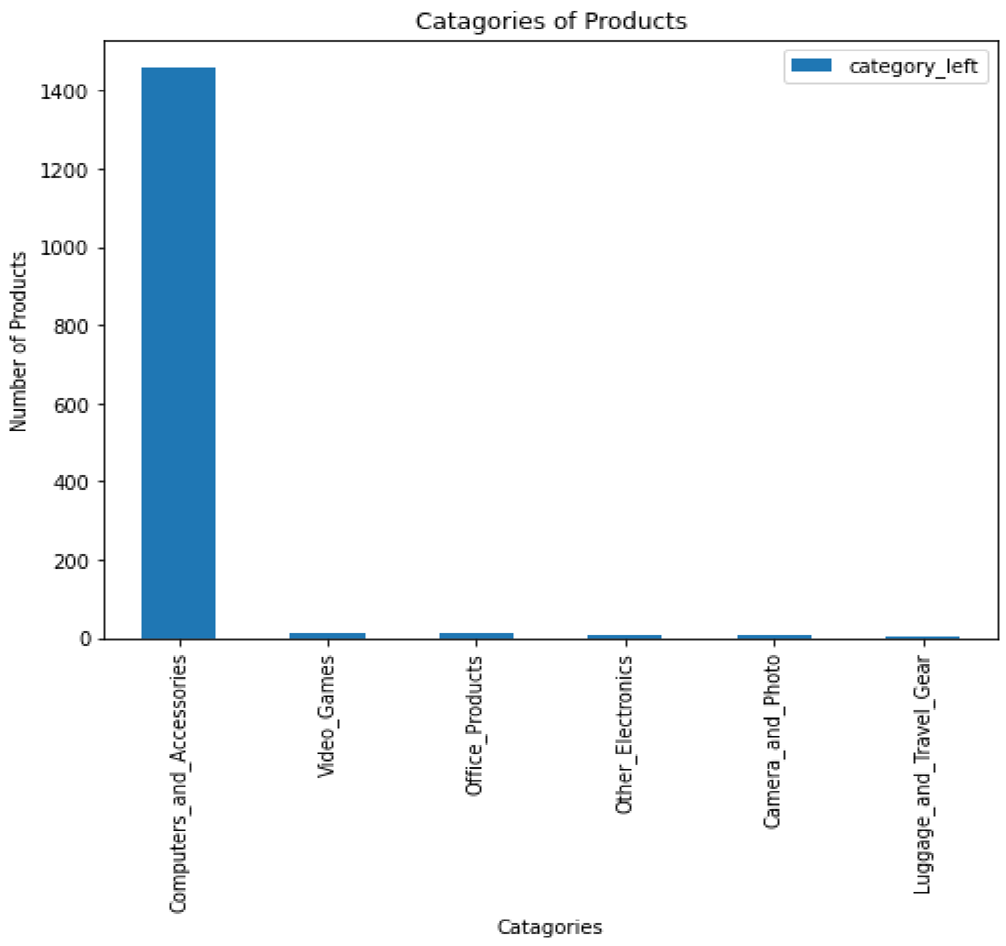


Figure 5: Top categories (left data subset)

If I talk about the categories in data subset right, computer and accessories dominate here as well, covering more than 80% of the total dataset. Second most popular category is other electronics and video games are on the third as shown in the figure below:

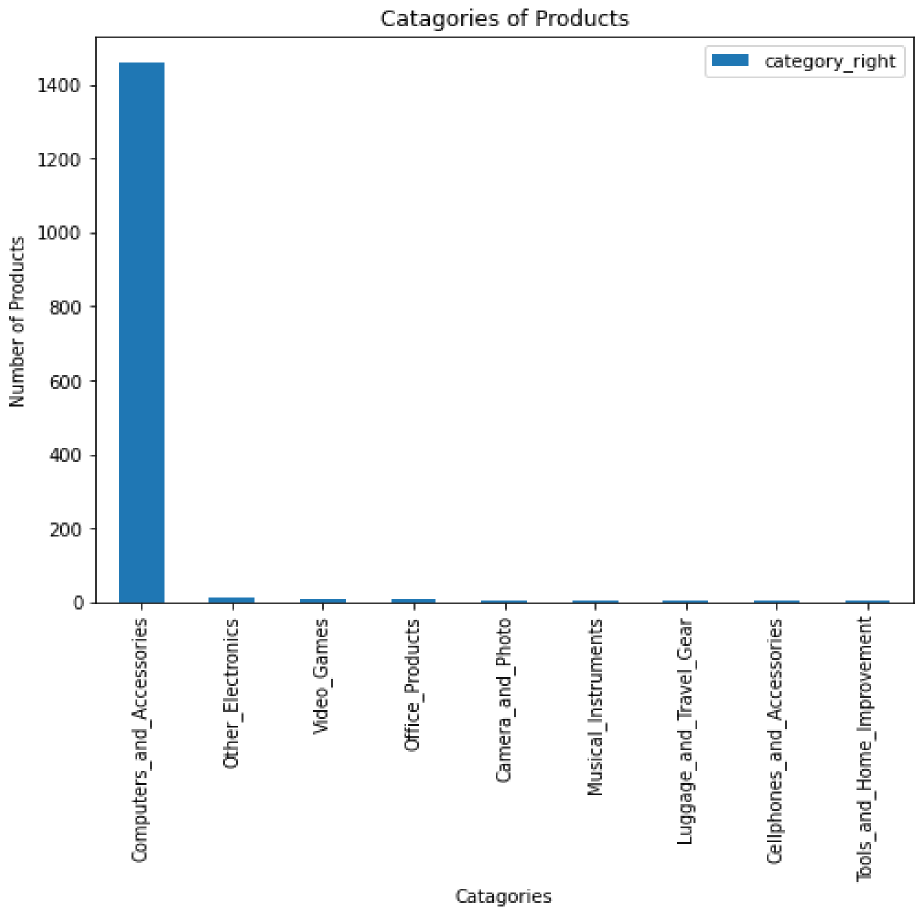


Figure 6: Top categories (right data subset)

Second attribute which I looked into is product brands. A Total of 92 unique brands are available in the left subset of the dataset. Here are some of the insights:

- HP enterprise is the most popular brand within our dataset. All the products of HP enterprise are listed under the category of computer and accessories. An exact number of 210 products of HP enterprises are available in the dataset.
- Corsair is the second most popular brand with 60 unique products. Corsair largely contributed to the category of video games and other electronics.
- Asus is third on the list with 41 unique products. AMD and Intel are fourth and fifth respectively.

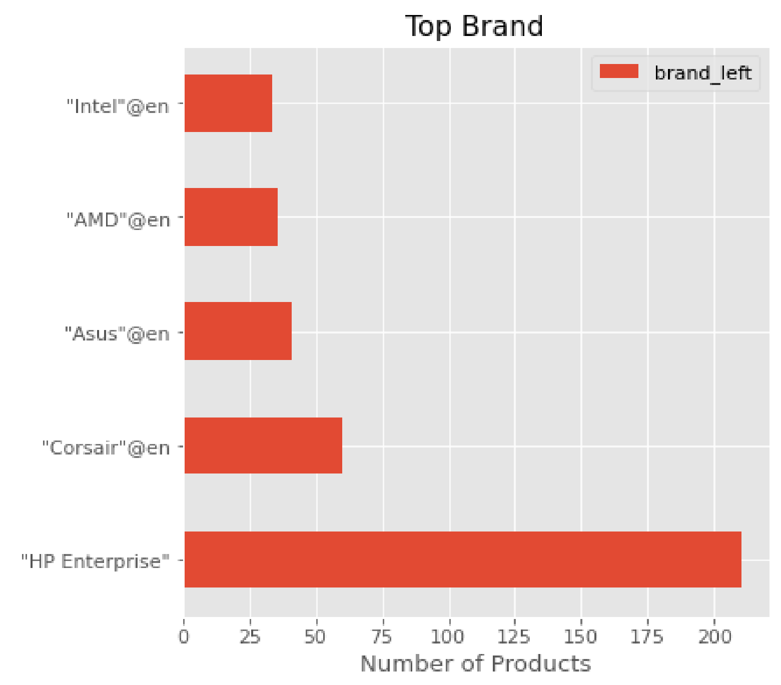


Figure 7: Top brands (left data subset)

If I talk about right subset of the dataset. HP is the most popular brand with 256 unique products. Corsair is second in line with 45 unique products. Intel with 40 unique products is third most popular brand, AMD and Intel are forth and fifth respectively. Product data of top 5 brands is visualized below:

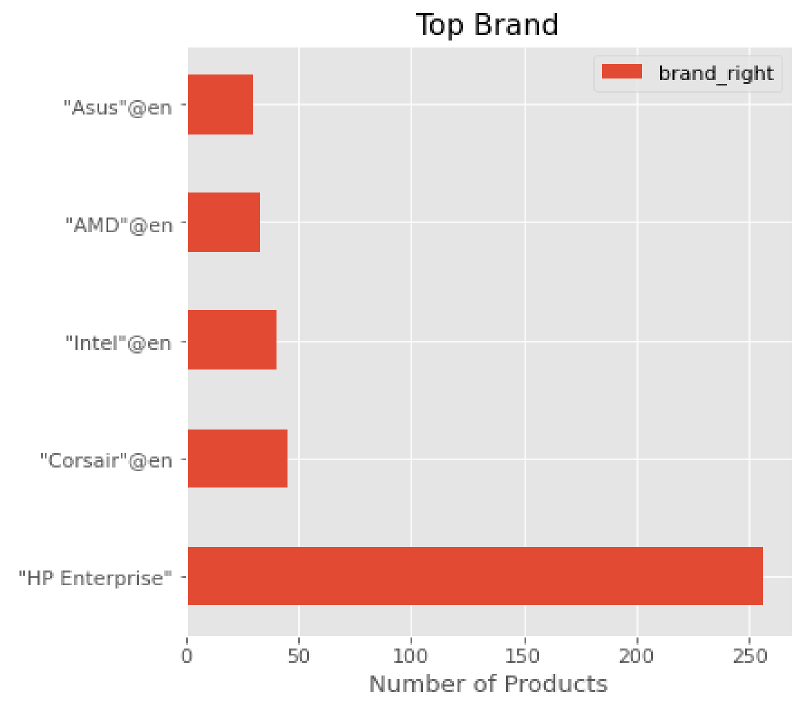


Figure 8: Top brands (right data subset)

Third attribute is number of unique products. A total of 598 unique products are available in the left subset of the dataset. Some insights are as follow:

- HyperX Fury is the most popular product in our dataset. It has two variants, one with DDR3 8 GB RAM and the other with DDR4 4 GB RAM.
- Apple MAC mini is one of the top products along with AMD Ryzen 7 and Kingston USB.
- Samsung 850 EVO internal SSD is one of the last popular products in our dataset.

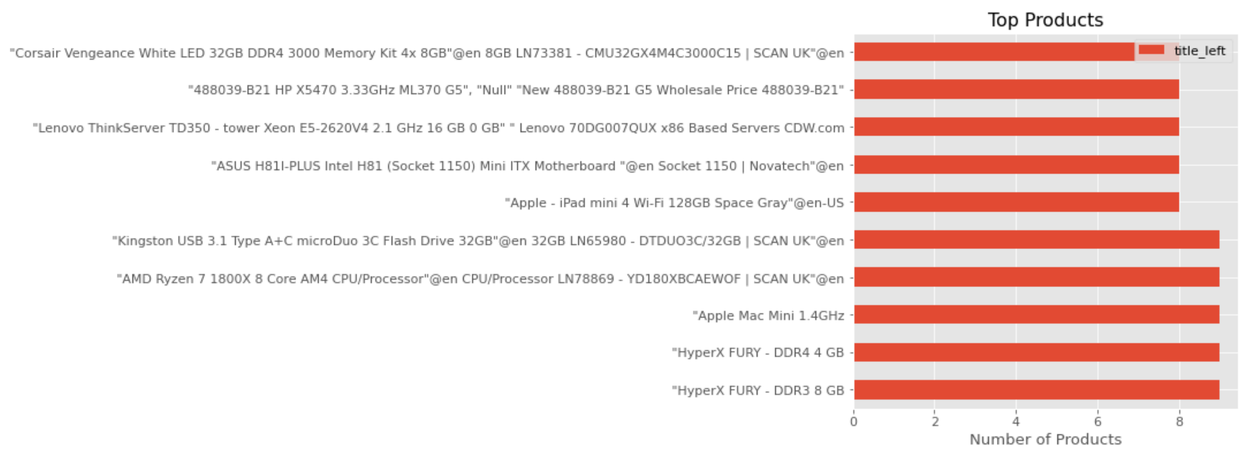


Figure 9: Top products (left data subset)

Meanwhile, a total of 1312 products are available in the right subset of the dataset. Top 10 products are visualized in the chart below:

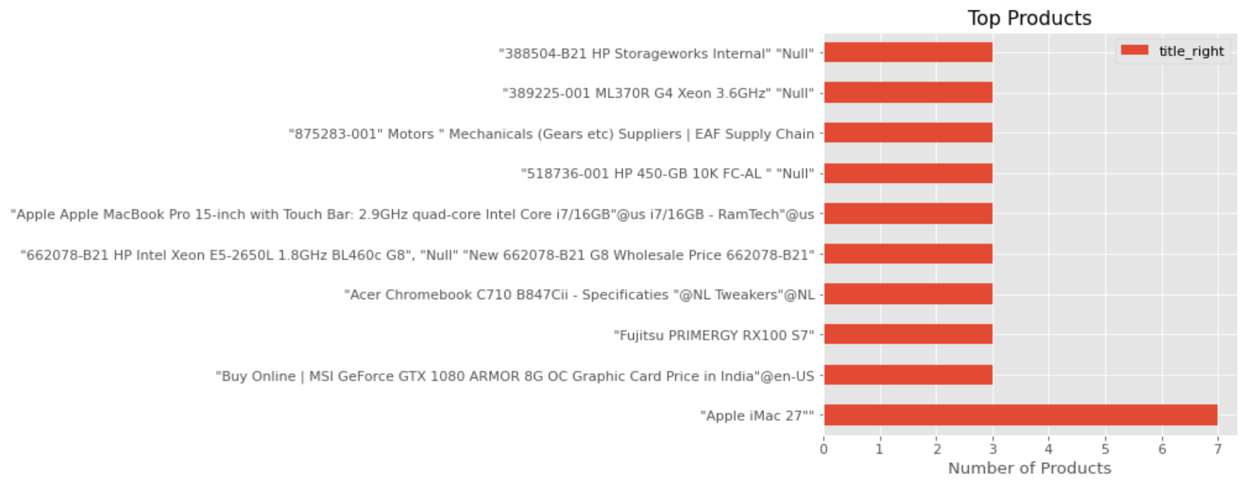


Figure 10: Top products (right data subset)

(Sonal, 2021) Exploratory data analysis makes it easier to comprehend the structure of a dataset, which facilitates data modelling. Data should be "clean," which implies it should be devoid of redundancies, according to the fundamental goal of EDA. It makes it easier to find false data points, so they can be quickly removed, and the data can be cleaned. Additionally, it aids in our comprehension of the relationship between the variables, providing us with a clearer image of the data and enabling us to expand on it by making use of the interactions between the variables. It also makes it easier to evaluate the dataset's statistical measures. Outliers or unusual events in a dataset can have an influence on machine learning model accuracy. There may also be some missing or duplicate values in the dataset. EDA may be used to remove or resolve all of the dataset's unwanted characteristics.

4.1.3 Data Transformation:

The term "data transformation" refers to changing the format, structure, or values of data. When utilized for data analytics projects, data can be altered twice along the data pipeline. Data transformation serves as the intermediary stage in the ETL (extract, transform, load) process used by businesses using on-premises data warehouses. The processing and storage capacity of cloud-based data warehouses may be expanded with a delay measured in seconds or minutes. Organizations may load raw data directly into data warehouses using ELT (extract, load, transform), without making any preload changes, and convert the data when a query is received. Data migration, warehousing, integration, and wrangling are a few examples of operations where data transformation may be applied. Furthermore, it is essential for any corporation wishing to use its data to offer pertinent business insights. As the volume of data has expanded, organizations want a trustworthy way for utilizing it in order to use it effectively for their operations. Utilizing this data requires data transformation since, when done correctly, it guarantees that the information is available, consistent, secure, and finally acknowledged by the intended business users.

After data exploration, data has been transformed to make it suitable for training of machine learning model. I have labelled the data; matching products are labelled as 1 and products which are not matching are labelled as 0. I have used product id, product title, product brand and product description as features which are used to train the machine learning model. As we know that, machine learning models can only understand numerical values, for that I converted all the textual features into numerical values. I have used CountVectorizer class from the sklearn.feature_extraction.text library. There are several critical arguments that must be given to the class's function. The first argument is set to 1500, which is the maximum number of features. This is because when you utilize the bag of words technique to convert words to numbers, all of the unique words in all of the documents are turned into features. Tens of thousands of unique terms can be found in each document. However, words with a very low frequency of occurrence are not an effective criterion for categorizing manuscripts. As a result, we set the max features option to 1500, indicating that we want to train our classifier using the top 1500 most often occurring terms. The next argument is mindf, which is set to 5. This is the minimum number of rows that should have this functionality. As a result, we only list terms that appear in at least 5 rows. Similarly, the value for the maxdf feature is set to 0.7, where the fraction equates to a percentage. In this case, 0.7 suggests that we should only consider terms that appear in at least 70% of all rows. Words that appear in nearly every text are typically unsuitable for categorization since they give no distinctive information about the material. Finally, we eliminate the stop words from our text because they may not contain any valuable information in the case of sentiment analysis. We send the stopwords object from the nltk.corpus library to the stop words parameter to remove the stop words. The CountVectorizer class's fit transform method turns text documents into numerical characteristics.

The bag of words approach works effectively for converting text to numbers. However, it does have one flaw. It assigns a score to a word based on where it occurs in a document. It does not account for the potential that the term appears often in other works as well. TFIDF solves this problem by increasing the term frequency of a word by the inverse document frequency. TF and IDF are abbreviations for "Term Frequency" and "Inverse Document Frequency," respectively. The TFIDF value of a phrase in a specific document is higher if it occurs more frequently in that row but less frequently in all other rows.

4.1.4 Splitting data into training and testing sets:

The train test split() method is used to divide our data into train and test sets. To begin, we must divide our data into features (X) and labels (y). There are four segments in the dataframe: X train, X test, y train, and y test. Model training and fitting are done with the x and y train sets. The X and Y test sets are used to determine if the model correctly predicts the outputs/labels. We can test the train and test set sizes explicitly. It is suggested that we keep our train sets larger than our test sets.

The training dataset is a set of data that is used to fit the model. The dataset on which the model was trained. This data is seen and learned by the model. The test dataset is a subset of the training dataset that is used to assess the final model fit properly.

4.1.5 Model Selection:

Model selection is the process of picking one final machine learning model for a training dataset from a group of candidate machine learning models. Model selection is a method that may be used to models of various types (e.g., random forest, logistic regression, support vector machine, and so on) as well as models of the same type of set with alternative model hyperparameters (for example, different kernels in a support vector machine). Given the statistical noise in the data, the incompleteness of the data sample, and the limits of each model type, all models have some prediction inaccuracy. As a result, the concept of a perfect or best model is useless. Instead, we must look for an adequate model. I have used three machine learning models and done a comparative analysis. Those machine learning models are discussed as follow:

4.1.5.1 Random Forest:

It is an algorithm for supervised machine learning. Classification and regression can both be used. The algorithm is also the most customizable and user-friendly. Trees comprise a forest. It is believed that the greater the number of trees in a forest, the stronger it gets. Random forests create decision trees from randomly selected data samples, then get forecasts from each tree and vote for the best choice. It's also a great indicator of importance of a feature.

Technically speaking, it is an ensemble approach of decision trees built using the divide-and-conquer strategy on a dataset that was randomly divided. Another term for this group of decision tree classifiers is the forest. For each feature, a unique decision tree is built using an attribute selection indicator such the Gini index, gain ratio, or information gain. Based on a distinct random sample, each tree. Each tree casts a vote on a classification problem, and the winning class is determined by the number of votes it receives. The average of all tree outputs is what is ultimately determined in regression. In comparison to earlier non-linear classification algorithms, it is both easier and more effective.

4.1.5.2 Logistic Regression:

(Kambria, 2019) Logistic regression is a classification algorithm that is employed when the target variable's value is categorical. Logistic regression is most commonly employed when the data in question has a binary output, falls into one of two classes, or is either 0 or 1. Similar to how linear regression assumes that the data follows a linear distribution, logistic regression models the data using the sigmoid function. When plotted on a graph, the sigmoid function/logistic function resembles an “S" shaped curve. It takes numbers between 0 and 1 and squishes them towards the top and bottom edges, marking them as 0 or 1. The equation of sigmoid function is as follows:

$$y = \frac{1}{(1 - e^{-x})}$$

It is critical to remember that logistic regression should only be used when the target variables are discrete and that it should not be utilized if the target variable contains a range of continuous values. Logistic regression does not become a classification method unless a decision threshold is included in the equation. A key component of logistic regression is the threshold value, which is established by the classification problem itself. Determining the threshold value is heavily influenced by the accuracy and recall levels. Although it would be ideal, accuracy and recall are rarely equal to 1. In the context of a precision recall tradeoff, the following arguments are used to determine the threshold:

Low Precision/High Recall:

We choose a decision value with a low Precision or a high recall when we wish to minimize the proportion of false negatives without necessarily decreasing the number of false positives.

High Precision/Low Recall:

We choose a decision value with a high Precision value and a low Recall value when we wish to reduce the number of false positives but not necessarily lowering the number of false negatives.

4.1.5.3 Support Vector Machine (SVM):

(Gandhi, 2018) Support Vector Machine (SVM) is a supervised machine learning algorithm which is most suitable for classification and regression problems. It is, however, extensively applied in categorization tasks. Each data element is represented as a point in an n-dimensional space, where n is the total number of features, and each feature's value corresponds to a certain position in the support vector machine algorithm. Then, classification is achieved by identifying the hyperplane that most effectively separates the two classes.

Decision boundaries known as hyperplanes help classify data elements. The data points on either side of the hyperplane belong to different classes. Additionally, the quantity of features affects how big the hyperplane is. The hyperplane is essentially a line when there are two input characteristics. The hyperplane changes into a two-dimensional plane when there are three input features. Support vectors are closer-to-the-hyperplane data points that influence the position and orientation of the hyperplane. We increase the margin of the classifier by utilizing these support vectors. Changing the support vectors will change where the hyperplane is located. These are the ideas that will help us create our support vector machine. In logistic regression, the output of the linear function was compressed between 0 and 1 using the sigmoid function. The value is labelled 1 if it exceeds a threshold value (0.5), else it is labelled 0. If the result of the linear function is more than 1, I used support vector machine to classify it into one class, and if it is less than 1, I used a different class. Because the SVM threshold values are changed to 1 and -1, I obtained the margin-effective reinforcing range of values [-1,1].

Chapter 5: Evaluation and Testing

5.1 Confusion matrix:

It is a table which is commonly used to illustrate the performance of a classification model on a set of test data for which the real values were known. Confusion matrix classifies the prediction results into four classes. True positives, values which are predicted correctly in positive class. True negative, values which are correctly predicted in negative class. False positive, values which are wrongly predicted in positive class and False negatives are the values which are wrongly predicted in negative class.

If I compare the models based in confusion matrix. Random forest algorithm has predicted true positives 1%. The percentage of true negatives is an astonishing 48.33%. The percentage of false positives is 50.17% and lastly the percentage of false negatives is 0.50%. Confusion matrix of random forest algorithms is given below:

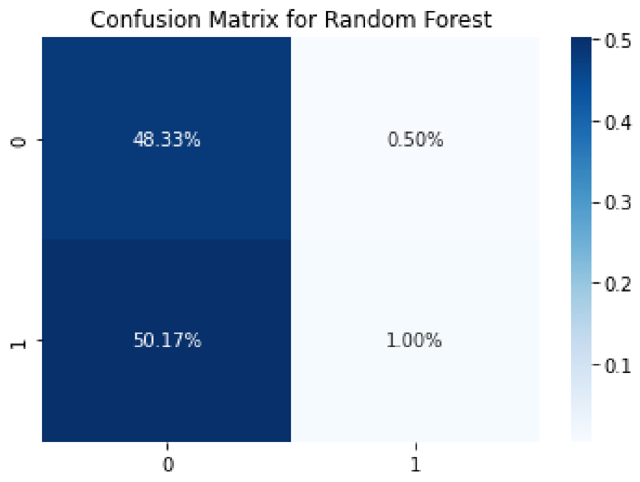


Figure 11: Confusion matrix for random forest

Logistic Regression algorithm has also production quite similar results with confusion matrix. It has predicted true positives of 1.22%. The percentage of true negatives is an astonishing 47.56%. The percentage of false positives is 50.33% and lastly the percentage of false negatives is 0.89%. Confusion matrix of random forest algorithms is given below:

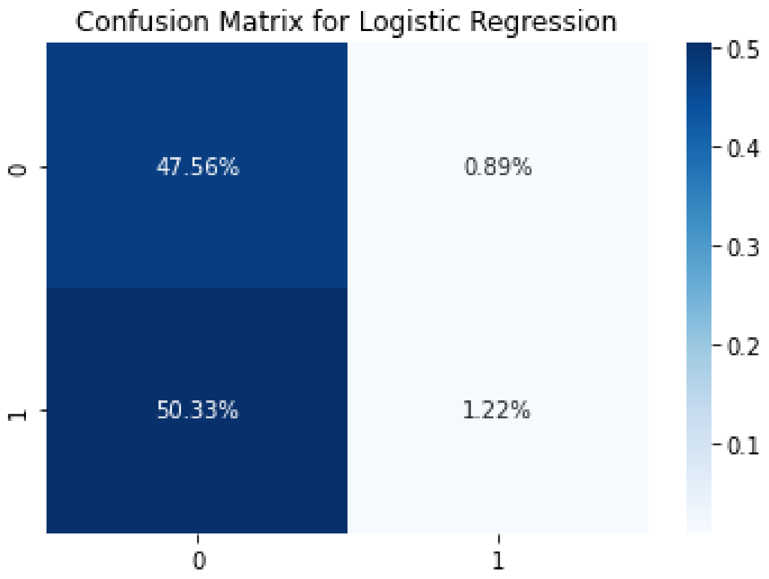


Figure 12: Confusion matrix for logistic regression

Lastly, support vector machine has predicted true positives of 0.83%. The percentage of true negatives is an astonishing 48.33%. The percentage of false positives is 50.33% and lastly the percentage of false negatives is 0.50%. Confusion matrix of random forest algorithms is given below:

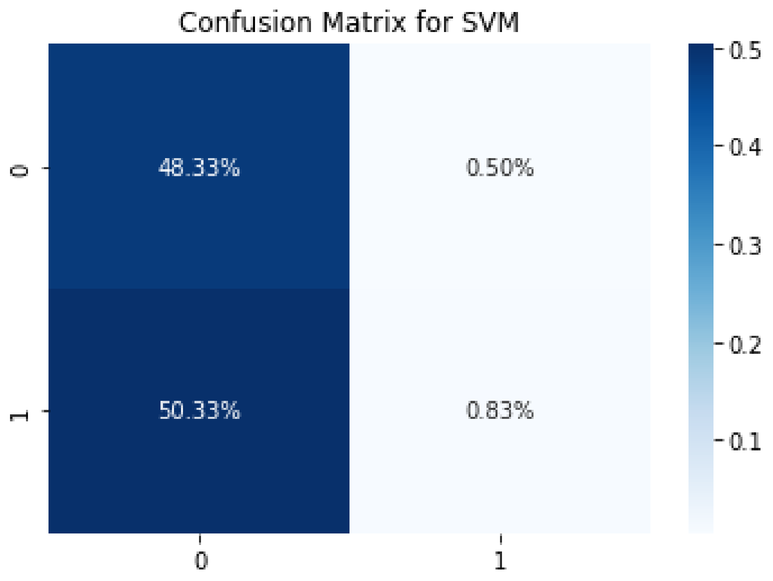


Figure 12: Confusion matrix for SVM

5.2 Mean Squared Error:

It is probably the most frequently utilized indicator for regression tasks. It is the average of squared error between the expected values and actual values. A better value for the mean squared error, which is always non-negative and measures an estimator's quality, is one that is closer to zero. Mean squared error can be calculated using the following equation.

$$\text{MSE} = \frac{1}{\text{n}} \sum_{i=1}^{n} (Y_i - \hat{Y}_i)$$

Here n is number of data points, Y_i is the label value and $\hat{Y}_i$ is the value predicted by our model. Mean squared error only indicates how well a regression line resembles a set of points. This is accomplished by squaring the distances between the points and the regression line (also known as the "errors"). To simplify the intricacy with negative signs, squaring is essential. The model may need to be more precise in order to reduce mean squared error, which would bring the model closer to the real data. The least square approach, which assesses if a linear regression model is adequate for a bivariate dataset, is one example of a linear regression that makes use of this technique. However, this method has limitations relating to the data's known distribution. These comparison of selected on bases of mean squared error is given in the table below:

Algorithm	*Mean Squared Error*
Random Forest	0.507
Logistic Regression	0.512
Support Vector Machine	0.512

Table 2: MSE comparison

5.3 Accuracy Score:

Accuracy is a popular statistic for calculating the performance of classification models. The percentage of labels correctly predicted by our model is denoted by accuracy. For example, if our model correctly identified 70 out of 100 labels, its accuracy would be 0.70. These comparison of selected on bases of mean accuracy score is given in the table below:

Algorithm	*Accuracy Score*
Random Forest	0.493
Logistic Regression	0.488
Support Vector Machine	0.488

Table 3: Accuracy score comparison

Chapter 6: Conclusion

To examine the problem of discovering the same products across numerous platforms, I built a machine learning product matching model that considers the respective aspects of product names and attributes, and I modelled the problem as ranking and classification scenarios. In order to combine our matching traits with a few other variables, I also tested out various conventional classification and ranking algorithms. The results showed that our model performed very well compared to earlier techniques in both classification and ranking scenarios. The experiments were conducted on data obtained from real-world web marketplaces. We'll look into a better method in the future to determine how relevant these criteria are, as well as how to add more product information into product matching, such product photographs and in-depth product descriptions.

References:

Adeeba, S. et al. (2022) “A comparative study of machine learning algorithms for predicting weight range of neonate,” 2022 International Conference on Decision Aid Sciences and Applications (DASA) [Preprint]. Available at: https://doi.org/10.1109/dasa54658.2022.9765164.

AltexSoft (2020) Non-functional requirements: Examples, types, how to approach, AltexSoft. Available at: https://www.altexsoft.com/blog/non-functional-requirements/ (Accessed: January 9, 2023).

Cai, J. et al. (2018) “Feature selection in Machine Learning: A new perspective,” Neurocomputing, 300, pp. 70–79. Available at: https://doi.org/10.1016/j.neucom.2017.11.077.

C. V. Gonzalez Zelaya, "Towards Explaining the Effects of Data Preprocessing on Machine Learning," *2019 IEEE 35th International Conference on Data Engineering (ICDE)*, 2019, pp. 2086-2090, doi: 10.1109/ICDE.2019.00245.

Ding, J., Tarokh, V. and Yang, Y. (2018) “Model selection techniques: An overview,” IEEE Signal Processing Magazine, 35(6), pp. 16–34. Available at: https://doi.org/10.1109/msp.2018.2867638.

Gandhi, R. (2018) Support Vector Machine - introduction to machine learning algorithms, Medi-um. Towards Data Science. Available at: https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47 (Accessed: December 26, 2022).

Jeremy Jordan (2018) Evaluating a machine learning model., Jeremy Jordan. Jeremy Jordan. Available at: https://www.jeremyjordan.me/evaluating-a-machine-learning-model/ (Accessed: January 9, 2023).

Khalid, S., Khalil, T. and Nasreen, S. (2014) “A survey of feature selection and feature extraction techniques in machine learning,” 2014 Science and Information Conference [Preprint]. Available at: https://doi.org/10.1109/sai.2014.6918213.

Kleimann, S.G.A. (2020) Logistic regression for binary classification, Medium. Towards Data Science. Available at: https://towardsdatascience.com/logistic-regression-for-binary-classification-56a2402e62e6 (Accessed: January 9, 2023).

Kluge, R. et al. (2008) “An approach for matching functional business requirements to standard application software packages via ontology,” 2008 32nd Annual IEEE International Computer Software and Applications Conference [Preprint]. Available at: https://doi.org/10.1109/compsac.2008.147.

Li, J. et al. (2019) “Deep cross-platform product matching in e-commerce,” Information Retrieval Journal, 23(2), pp. 136–158. Available at: https://doi.org/10.1007/s10791-019-09360-1.

Navlani, A. (2019) Scikit-learn SVM tutorial with Python (Support Vector Machines), DataCamp. DataCamp. Available at: https://www.datacamp.com/tutorial/svm-classification-scikit-learn-python (Accessed: January 9, 2023).

Schönleber, D. (2021) A "short" introduction to model selection, Medium. Towards Data Science. Available at: https://towardsdatascience.com/a-short-introduction-to-model-selection-bb1bb9c73376 (Accessed: December 24, 2022).

TIBOACA, M.E., COSTINAS, S. and RADAN, P. (2021) “Design of short-term wind production forecasting model using machine learning algorithms,” 2021 12th International Symposium on Advanced Topics in Electrical Engineering (ATEE) [Preprint]. Available at: https://doi.org/10.1109/atee52255.2021.9425247.

Vrigazova, B. (2021) “The proportion for splitting data into training and test set for the bootstrap in Classification problems,” Business Systems Research Journal, 12(1), pp. 228–242. Available at: https://doi.org/10.2478/bsrj-2021-0015

Yiu, T. (2021) Understanding random forest, Medium. Towards Data Science. Available at: https://towardsdatascience.com/understanding-random-forest-58381e0602d2 (Accessed: January 7, 2023).