**9. Project Technical Implementation**

**9. 1 IMPORTING THE DATASET IN JUPYTER NOTEBOOK:**

Graphical user interface, text, application

Description automatically generated

I opened the "Customer data" data set and used the pandas library to determine the shape of the data. Pandas provide incredibly straightforward data representation. This enhances understanding and data analysis. Simpler data representation in data science activities leads to better results. The opened dataset's data shape is: (2240, 11), It means the dataset consists of 2240 rows and 11 columns.

**9.2 DATA TYPES:**

A variable's data type and the kinds of mathematical, relational, and logical operations that can be performed on it without producing an error are classified as data types in programming. For instance, a data type called a string is used to categorise text, while a data type called an integer is used to categorise whole numbers.

| **Variables** | **Data Types** |
| --- | --- |
| Income | float64 |
| Kidhome | int64 |
| Teenhome | int64 |
| Recency | int64 |
| MntFruits | int64 |
| MntMeatProducts | int64 |
| MntSweetProducts | int64 |
| NumWebVisitsMonth | int64 |
| AcceptedCmp1 | int64 |
| AcceptedCmp2 | int64 |
| Coupon | int64 |

According to the table above, all variables are of the integer and float data types, with the exception of the variable Income. It is important to have all the variable as integer because machine learning models are only relevant for numbers and not strings. Though Income column is float type, it comes under integer pattern only.

**9.3 Missing Values:**

The data value that is not kept for a variable in the relevant observation is known as missing data (or missing values). Nearly all research encounters the issue of missing data, which can significantly affect the inferences that can be made from the data.

| **Variables** | **Data Types** |
| --- | --- |
| Income | 0 |
| Kidhome | 0 |
| Teenhome | 0 |
| Recency | 0 |
| MntFruits | 0 |
| MntMeatProducts | 0 |
| MntSweetProducts | 0 |
| NumWebVisitsMonth | 0 |
| AcceptedCmp1 | 0 |
| AcceptedCmp2 | 0 |
| Coupon | 0 |

However, our dataset does not contain any missing values.

**9.4 Duplicated data**

Any record that unintentionally shares data with another record in a database is considered to have duplicate data. Most duplicate data happens while transmitting data across systems, and it is simple to detect. The most popular occurrence of duplicate data is a complete carbon copy of a record.

We have 186 duplicate records accessible, according to our data. I got rid of the redundant rows to ensure that I was building a solid model. The data shape has finally changed to (2054, 11) post removing the duplicates

**9.5 Statistical summary:**

The DataFrame's columns' statistics are compiled by the describe() method. The mean, standard deviation, and interquartile range (IQR) values are returned by this function. Furthermore, the method eliminates character columns and provides a summary of numeric columns.

Graphical user interface

Description automatically generated

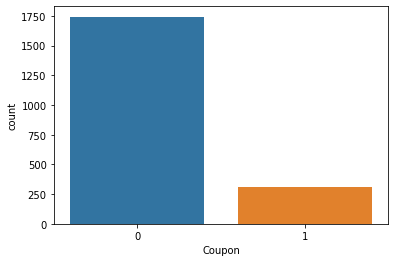
The table indicates that there are 2054 data frequencies. Additionally, we can see all the columns have different standard deviation. Hence, it is necessary for us to normalise the data before feeding the data into the model. The target column coupon clearly states that the minimum value is 0 and 1. Hence, we can confirm it is a classification problem.

**9.6 Exploratory data analysis:**

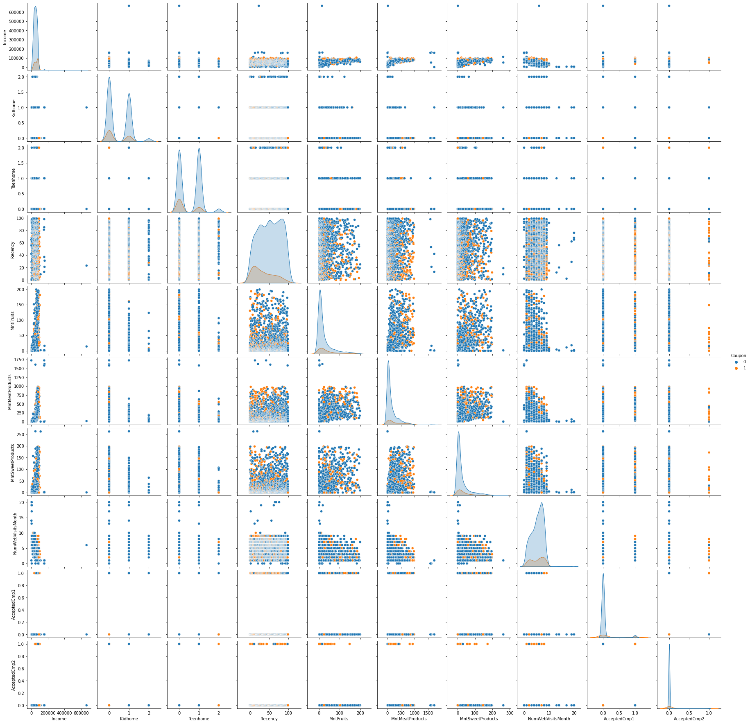
Exploratory visualisation is the process of creating maps and other graphics while working with geographical data that is not completely known. These maps frequently have a specific function and assist the expert in trying to resolve a (geo) problem.

Python's seaborn and matplotlib modules were used to summarise the data's main features in order to undertake an exploratory data analysis that would help users better comprehend the dataset. This phase of the research is crucial because it aids in getting a sense of what's contained in the data set, transforming it into a visual medium to quickly identify its features, such as interesting curves, lines, trends, or anomalous outliers, and it's used to find any intriguing storylines in the data.

**Histogram** - Information is graphically represented in a bar graph. To represent value, it makes use of bars that stretch to various heights. Vertical bars, horizontal bars, grouped bars (several bars that compare values in a category), and stacked bars can all be used to construct bar graphs (bars containing multiple types of information).



By looking at the graph we can confirm that there are some data imbalance in terms of the target column which would lead the model to overfit only for the target value 0 since the 0 has more frequency than 1. Imbalanced data typically refers to a classification problem in which the classes are not equally represented.

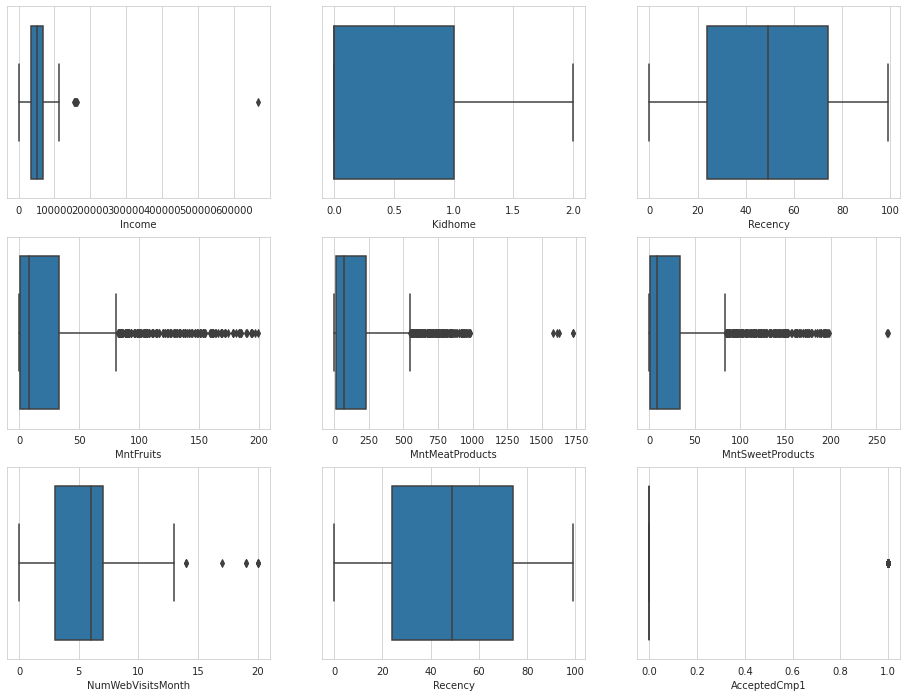


**Pair Plot** - The relationship between the given data, where the variables may be continuous or categorical, is visualised using a pair plot. In a data set, plot pairwise relationships. The seaborn library's Pair plot module offers a high-level interface for creating visually appealing and educational statistics visuals. From the above pair plot, we can confirm that none of the feature is separating the data perfectly. However, if we see the difference by seeing the colour of the target class 0 and 1

A picture containing crossword puzzle, shoji, text

Description automatically generated

A **histogram** of all the variables shows clearly which ones are continuous and which ones are categorical. From the above graph, we can see that the column recency, sweet products,number of websites of a month, mntmeats, mntfruits, mnt sweets are continuous and others are discreate data. The frequency of the coupoun 1 and coupon 2 data are also low.



**Bar Chart** - You may discover that you require more details than just the measures of central tendency for specific distributions or data sets (median, mean and mode). You must have knowledge of the data's variability or dispersion. A boxplot is a graph that effectively illustrates how the values in the data are distributed. Boxplots have the benefit of taking up less room than a histogram or density plot, which is helpful for comparing distributions across numerous groups or data sets. Despite the fact that they may appear simple in comparison. Boxplots can reveal information about your outliers, including their values. Additionally, it can inform you of the degree of grouping, symmetry, and skewness of your data.

The boxplot shown above makes the outliers in every column very visible. The outliers for the column’s Income, MntMeatProducts, and MntSweetProducts can be eliminated because the outliers caused in the extreme outside of the data points. Other outliers cannot be eliminated because even though it has outliers, they can be the important points on the dataset.

**Graphical user interface, application

Description automatically generated**

Plots of **correlation** can be utilised to quickly discover insights. It is applied to highlight the most associated variables in a data table and to study the simultaneous dependence between numerous variables. The hue of the correlation coefficients in this graph reflects their value. Additionally, the correlation matrix may be rearranged in accordance with the strength of the correlation between variables or grouped using the hierarchical clustering technique. This graphic is easy to use and intuitive.

The process is as follows:

* Define the quantitative parameters to be investigated (two or more columns)
* Utilize a variety of formatting options to enhance the plot's visual appeal.

The Corelation plot shown above provides a clearer picture of the relationship between each variable and the others. The variable Coupon has a co-related relationship with the column Acceptedcmp1 of 29%, the column MntMeatProducts of 26%, and the column Acceptedcmp2 of 18%. The three columns mentioned above are therefore essential for Coupon prediction. However, Kidhome and NumWebVisitsMonth, which have a -0.8% and 0.7% correlation with Coupon respectively, are the least important in terms of price prediction. However, since the dataset is not larger in terms of the computer's efficiency of use, we are not eliminating.

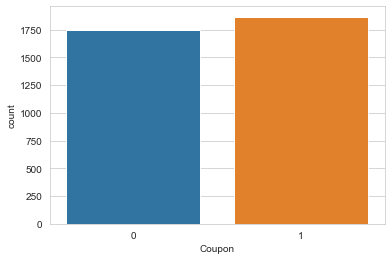
**9.7 Upsampling the data:**

Upsampling is the process of inserting synthetic and real data sets (representing the minority class) into a dataset. Following this procedure, the counts for both labels are nearly identical. This equalisation procedure keeps the model from favouring the majority class. Furthermore, the interaction (boundary line) between the target classes does not change. Furthermore, the extra details introduced by the upsampling mechanism introduces bias into the system.

From the above analysis, we found that the target column has the imbalanced data. If we create a model with the same data, the model will get biased for the class which has higher frequency. Hence, we are upscaling the data so that the model does not give biased results.



The above image of the code says that the frequency of the coupon variable 1 is just 311 and the coupon variable 0 is 1743. To make sure both the frequency of the variable is same, we have multiplied the coupon 1 with 6. Hence, both together got the value of 3609.



Post upsampling the data, we can see that the coupon column category has similar frequencies. Hence, the model will throw out the good accuracy if the data is good.

**9.8 Train Test Split**

A method for evaluating the effectiveness of a machine learning algorithm is the train-test split. It can be applied to classification or regression tasks and can be utilised for any supervised learning technique.

Two subgroups are created by splitting a dataset. The model is fitted using the first subset, also referred to as the training dataset. The model is given the input element from the dataset instead of the second subset, and predictions are made and compared to actual values. The test dataset is the second dataset.

**Train Dataset**: The machine learning model is fitted using the train dataset.

**Test Dataset**: Used to evaluate how well the machine learning model fits the data.

The objective is to evaluate the machine learning model's performance using fresh data that was not utilised in model training.

We intend to apply the paradigm in practise in this manner. Or to put it another way, we want to adapt it to the data that already exists with known inputs and outputs and then make predictions on new cases that will arise in the future without the goal or expected output values. The train-test method is suitable when a sizable enough dataset is available.

**9.8.1 WHEN TO USE TRAIN-TEST SPLIT**

Each predictive modelling assignment defines "sufficiently large" differently. It means that there is enough data to split the dataset into a train and test dataset and that each train and test dataset accurately captures the problem domain. This demands that the initial dataset be an accurate reflection of the problem domain.

A problem domain is adequately represented if there are enough entries to account for both common and uncommon occurrences. This could be a reference to actual input variable combinations. There could be a need for thousands, hundreds of thousands, or even millions of instances.

The train-test method is useless when there is a minimal amount of data available. This is because the training dataset will not contain the test set when the dataset is split into train and test sets.

enough information to teach the model an effective way to map inputs to outputs. Additionally, there won't be enough information in the test set to properly assess the model's performance. The performance estimate could be overly optimistic (great) or incredibly pessimistic (poor) (bad).

The k-fold cross-validation strategy is an useful alternate model assessment method if you don't have enough data. In addition to dataset size, another justification for using the train-test split assessment method is computational effectiveness.

Some models are very expensive to train, making it impractical to do regular assessments as used in other techniques. One such is deep learning models. The train-test strategy is frequently used in this situation. As an alternative, a project can have an effective model and a sizable dataset but still require a quick evaluation of the model's effectiveness. The train-test split method is applied here once more.

Samples from the initial training dataset are split into two groups using random selection. To ensure that the train and test datasets accurately reflect the original dataset, this is done.

**9.8.2 HOW TO CONFIGURE THE TRAIN-TEST SPLIT**

The main customizable parameter for the technique is the size of the train and test sets. The most common way to express this is as a percentage between 0 and 1 for either the train dataset or the test dataset. For instance, if the size of the training set is 0.67 (67%), the test set will receive the 0.33 (33%), or leftover, percentage.

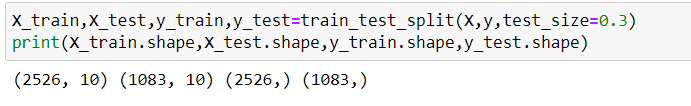
There is no such thing as a proper split percentage. You must choose a split percentage that aligns with the goals of your project, considering things like:

* The expense of computing when training the model.
* The cost of computing when evaluating the model.
* The training set's representativeness
* Test set representativeness

On the other hand, common split percentages are as follows:

* 80% for teaching, 20% for testing
* 33% test, 67% train
* 50% testing and 50% training

Let's look at how we may use the train-test split model assessment technique in Python now that we've covered it.



According to the figure above, I divided the data with a test size of 0.3, which indicates that 70% of the data went to the training set and 30% went to the testing set. The training set's shape is 2526, whereas the test set's is 1083.

**9.9 Standardising the Data**

Standardization assumes that your data has a Gaussian (bell curve) distribution. This is not required, but the technique works better if your attribute distribution is Gaussian. Standardization is useful when your data has varied scales and your algorithm, such as linear regression, logistic regression, and linear discriminant analysis, makes assumptions about your data having a Gaussian distribution.

From the above visualisation analysis, we can say that the data does not form a gaussian distribution, hence we are standardising the data with the python function minmaxscalaer()

Text

Description automatically generated

From the above image, we can say that we have done the fit\_transform for training data and transform for the test data. When you call MinMaxScaler.fit(X train), it computes the mean and variance based on the values in X train. Then, using.transform(), all of the features are transformed by subtracting the mean and dividing by the variance. Fit transform allows you to perform these two functions calls in a single step for convenience ().

Because you don't want to bias your model with information from the test data, you should fit the scaler using only the training data.

Fit() would compute a new mean and variance for each feature based on your test data. In theory, these values should be very similar if your test and train sets have the same distribution, but this is rarely the case in practise.

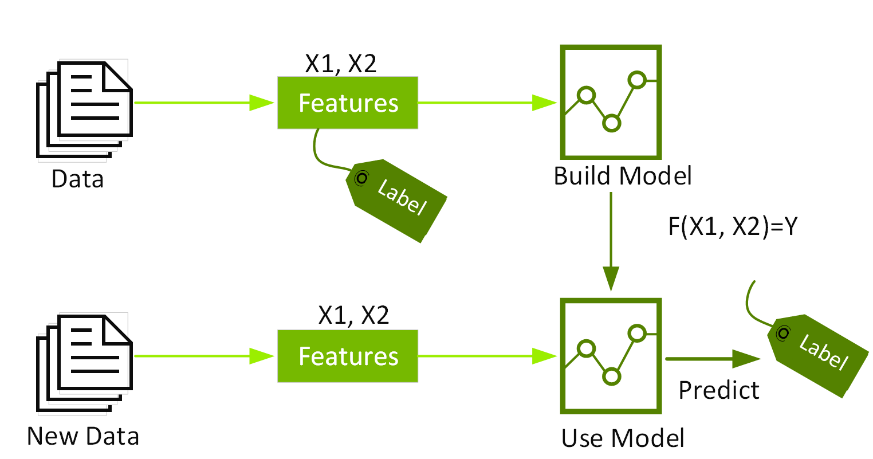
**9.10 Creating Models**

**9.10.1 XgBoosting**

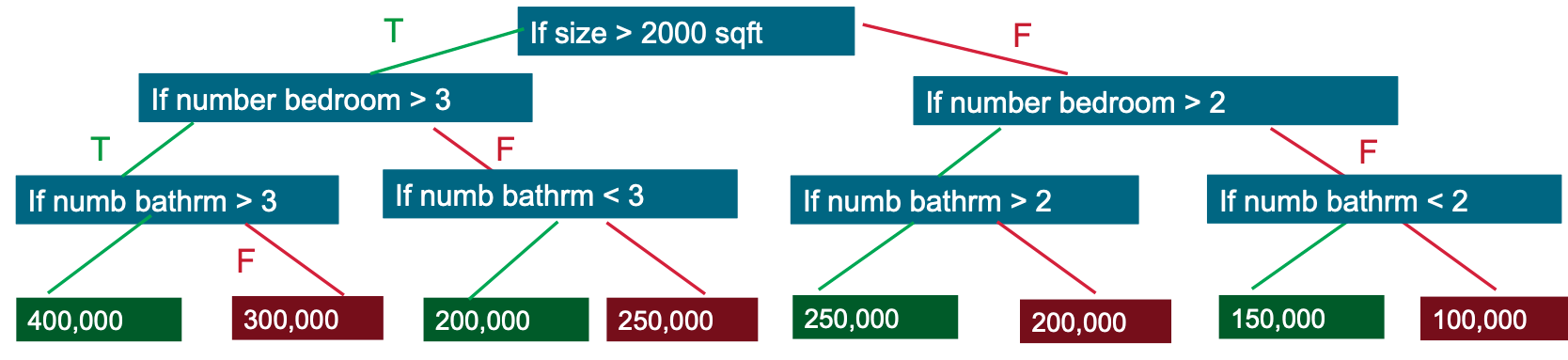
Extreme Gradient Boosting (XGBoost) is a scalable, distributed gradient-boosted decision tree (GBDT) machine learning library. It is the leading machine learning library for regression, classification, and ranking problems, and it supports parallel tree boosting.

To understand XGBoost, you must first understand the machine learning concepts and algorithms on which it is based: supervised machine learning, decision trees, ensemble learning, and gradient boosting.

Supervised machine learning employs algorithms to train a model to find patterns in a dataset with labels and features, and then employs the trained model to predict the labels on the features of a new dataset.



By evaluating a tree of if-then-else true/false feature questions and estimating the minimum number of questions required to assess the probability of making a correct decision, decision trees generate a model that predicts the label. Decision trees can be used to predict a category or a continuous numeric value using classification or regression. A decision tree is used in the simple example below to estimate a house price (the label) based on the size and number of bedrooms (the features).



**Classification report:**

A classification report is used to assess the accuracy of a classification algorithm's predictions. How many predictions are correct and how many are incorrect? True Positives, False Positives, True Negatives, and False Negatives are specifically used to predict the metrics of a classification report, as shown below.

**Precision - How accurate were your predictions?**

Precision is a classifier's ability to avoid labelling a negative instance as positive. It is defined for each class as the ratio of true positives to the sum of true and false positives.

TP – True Positives

FP – False Positives

Precision is the accuracy with which positive predictions are made.

Precision = TP/(TP + FP)

**Recall** - **What percentage of positive cases did you identify?**

A classifier's recall is its ability to find all positive instances. It is defined for each class as the ratio of true positives to the sum of true positives and false negatives.

FN – False Negatives

Recall is the percentage of positives that were correctly identified.

Recall = TP/(TP+FN)

**F1 score - What percentage of correct positive predictions were there?**

The F1 score is a weighted harmonic mean of precision and recall, with 1.0 being the best and 0.0 being the worst. F1 scores are generally lower than accuracy measures because they incorporate precision and recall into their computation. To compare classifier models, use the weighted average of F1, rather than global accuracy.

F1 Score = 2\*(Recall \* Precision) / (Recall + Precision)

Test Classification Report for model XG

**Confusion Matrix**

Chart, treemap chart

Description automatically generated

**Accuracy score for training data (XG) 0.8607142857142858**

**Accuracy score for testing data (XG): 0.8020351526364478**

| **Target coupon** | **Precision** | **Recall** | **F1-score** | **Support** |
| --- | --- | --- | --- | --- |
| 0 | 0.81 | 0.77 | 0.79 | 526 |
| 1 | 0.79 | 0.83 | 0.81 | 555 |
|  |  |  |  |  |
| accuracy |  |  | 0.80 | 1081 |
| macro avg | 0.80 | 0.80 | 0.80 | 1081 |
| weighted avg | 0.80 | 0.80 | 0.80 | 1081 |

If you see the accuracy of the training and testing data is 86% and 80% which means the model is overfitting because the training accuracy is more than the testing accuracy. Additionally, the precision, recall, and f1 score of the category 0 (No coupon available) is 0.81, 0.77 and 0.79 which is lesser than the category 1 (Coupon Available) which is 0.79, 0.83 and 0.79.

A **confusion matrix** is a summary of prediction results for a classification problem. The number of correct and incorrect predictions is summarised with count values and broken down by class. This is the key to the confusion matrix. The confusion matrix depicts the various ways in which your classification model can be confused. When making predictions, it becomes perplexed. It informs you not only about the errors made by your classifier, but also about the types of errors made. This breakdown overcomes the limitation of using classification accuracy alone.

From the above graph, we can say that

* Correct Detection of Coupon not given (0) is 409 out of 526 data
* Correct Detection of Coupon given (1) is 461 out of 555 data

This means that the XgBoosting model has found 120 of (0) class and 94 of (1) class wrongly.

**Roc Curve and AUC**

Chart, line chart

Description automatically generated

The AUC - ROC curve is a performance metric for classification problems at various threshold levels. AUC represents the degree or measure of separability, while ROC is a probability curve. It indicates how well the model can distinguish between classes. The greater the AUC, the better the model predicts 0 classes as 0 and 1 classes as 1. Similarly, the higher the AUC, the better the model distinguishes between patients with and without the disease.

The ROC curve is plotted with TPR versus FPR, with TPR on the y-axis and FPR on the x-axis.

An excellent model has an AUC close to one, indicating that it has a high level of separability. A poor model has an AUC close to zero, indicating that it has the worst measure of separability. In fact, it means that the result is being reciprocated. It predicts 0s to be 1s and 1s to be 0s. When the AUC is 0.5, the model has no class separation capacity at all.

From, the above roc curve, we can say that the AUC is 0.88 which is a good model to create a web application. However, we can try with other models to check whether we can improve the accuracy more.

**9.10.2 KNN model**

K-nearest Neighbors (KNN) is a supervised machine learning algorithm used for classification, regression, and outlier detection. In its most basic form, it is extremely simple to implement but can perform fairly complex tasks. It is a lazy learning algorithm because it lacks a specialised training phase. Rather, it trains on all of the data while classifying (or regressing) a new data point or instance.

KNN is a non-parametric learning algorithm, which means it makes no assumptions about the data. This is a very useful feature because most real-world data do not adhere to any theoretical assumptions, such as linear separability, uniform distribution, and so on.

KNN can be used to predict regression and classification problems. However, when it comes to industrial problems, it is mostly used in classification because it performs well across all parameters considered when determining a technique's usability.

* Prediction Capability
* Calculation Period
* The output is simple to interpret.

The KNN algorithm performs well across all parameters of consideration. However, it is mostly used because of its ease of interpretation and short calculation time.

To get the right K, run the KNN algorithm several times with different values of K and choose the one with the fewest errors. The right K must be able to accurately predict data that it has never seen before.

**Things to consider when deciding on the value of K**

Your prediction becomes less stable as K approaches 1. Because of most voters, your prediction becomes more stable as your K value increases. When you notice an increase in the number of errors, you know you're pushing your K too far.

To have a tiebreaker when taking a majority vote among labels, K must be an odd number.

**KNN Advantages**

* Time to calculate quickly
* To interpret, a simple algorithm is used.
* Versatile in that it can be used for regression and classification.
* High precision - no need to compare to better-supervised learning models.
* There are no data assumptions, so there is no need to make additional assumptions, tune several parameters, or build a model. This is especially important in the case of nonlinear data.

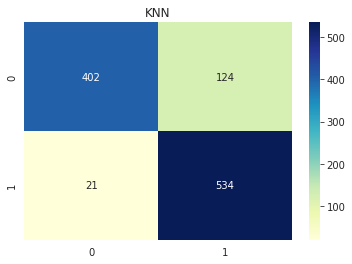
**KNN Disadvantages**

* The accuracy is determined by the quality of the data.
* The prediction stage may be slow when dealing with large amounts of data.
* Sensitive to data scale and irrelevant features
* High memory is required because all training data must be stored.
* It can be computationally expensive because it stores all the training.

**Accuracy score for training data (KNN) 0.9357142857142857**

**Accuracy score for testing data (KNN): 0.8658649398704903**

| **Target coupon** | **Precision** | **Recall** | **F1-score** | **Support** |
| --- | --- | --- | --- | --- |
| 0 | 0.95 | 0.76 | 0.85 | 526 |
| 1 | 0.81 | 0.96 | 0.88 | 555 |
|  |  |  |  |  |
| accuracy |  |  | 0.87 | 1081 |
| macro avg | 0.88 | 0.86 | 0.86 | 1081 |
| weighted avg | 0.88 | 0.87 | 0.86 | 1081 |



Chart, line chart

Description automatically generated

The model KNN gave the training data accuracy as 93% and the testing data accuracy is 86%. KNN gave the better accuracy than the Xgboost. However, the model is overfit since the training accuracy is more than the testing accuracy. To make sure the model is robust, we need to have similar accuracy in both training and testing samples.

The **confusion matrix** says that

* Correct Detection of Coupon not given (0) is 402 out of 526 data
* Correct Detection of Coupon given (1) is 534 out of 555 data

This means that the KNN model has found 124 of (0) class and 21 of (1) class wrongly. We can say that we have decreased the amount of the prediction wrongly in the class (1) when compared to the Xgboost model.

From, the above roc curve, we can say that the AUC is 0.91 which is a good model to create a web application. However, we can try with other models to check whether we can improve the accuracy more.

**9.10.3 Logistic Regression model**

Logistic regression is a classifier. It predicts a binary outcome from a collection of independent variables.

So, what exactly does this mean? A binary outcome has only two possible outcomes: the event occurs (1) or it does not occur (0). Independent variables are those that can influence the outcome (or dependent variable).

So, when dealing with binary data, you should use logistic regression. When the output or dependent variable is dichotomous or categorical in nature (such as "yes" or "no," "pass" or "fail"), you're dealing with binary data.

The independent variables, on the other hand, can fall into any of the following categories:

1. **Continuous** -Temperature in degrees Celsius or weight in grammes are examples of continuous data. In technical terms, continuous data is classified as either interval data (the intervals between each value are equally split) or ratio data (the intervals are equally split and there is a true or meaningful "zero"). For instance, temperature in degrees Celsius would be categorised as interval data; the distance between 10 and 11 degrees C is equal to the distinction between 30 and 31 degrees C, but there is no genuine zero—a temperature of zero degrees does not indicate there is "no temperature". Weight in grammes, on the other hand, is considered as ratio data since it contains equal intervals and a genuine zero. In other terms, if something has a weight of 0 grammes, it literally has no weight.
2. **Discrete, ordinal**—data that may be arranged in some form of order on a scale. If you are asked to rate your happiness on a scale of 1 to 5, the points on the scale indicate ordinal data. A score of 1 suggests less happiness than a score of 5, however there is no method to calculate the numerical value between each point on the scale. Ordinal data is the type of information obtained from a customer satisfaction survey.
3. **Discrete, nominal**—data that fits into designated groupings but does not indicate any sort of order or scale. For example, eye colour may fall into the categories "blue," "brown," or "green," yet these categories have no order.

**Assumptions of Logistic regression:**

* The dependent variable is binary or dichotomous, which means that it falls into one of two distinct groups. This is true for binary logistic regression, the form of logistic regression we've examined thus far. In part five, we'll look at several different forms of logistic regression.
* There should be no or very little multicollinearity between the predictor variables—that is, the predictor variables (or independent variables) should be independent of one another. This suggests that the independent variables should not have a high correlation. Certain tests in statistics may be used to measure the correlation between the predictor variables; if you want to learn more about them, just Google "Spearman's rank correlation coefficient" or "the Pearson correlation coefficient."
* The independent variables should be connected to the log chances in a linear fashion. We've provided a quick explanation of log odds below if you're unfamiliar with them.
* Logistic regression necessitates relatively large sample sizes—the greater the sample size, the more dependable (and strong) the findings of your study.

**Accuracy score for training data (log) 0.7634920634920634**

**Accuracy score for testing data (log): 0.7141535615171137**

| **Target coupon** | **Precision** | **Recall** | **F1-score** | **Support** |
| --- | --- | --- | --- | --- |
| 0 | 0.71 | 0.71 | 0.71 | 526 |
| 1 | 0.72 | 0.72 | 0.72 | 555 |
|  |  |  |  |  |
| accuracy |  |  | 0.72 | 1081 |
| macro avg | 0.71 | 0.71 | 0.71 | 1081 |
| weighted avg | 0.72 | 0.72 | 0.72 | 1081 |

Chart, treemap chart

Description automatically generated

The model Logistic regression model gave the training data accuracy as 76% and the testing data accuracy is 71% which is lower when compared to previous all both models. However, the model is quite weak in finding the coupon available and coupon not available. To make sure the model is robust, we need to build the accuracy towards 100%.

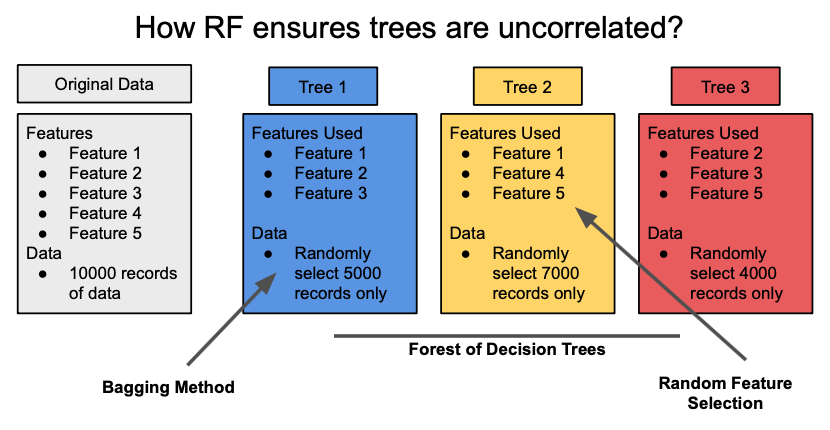
The **confusion matrix** says that

* Correct Detection of Coupon not given (0) is 371 out of 526 data
* Correct Detection of Coupon given (1) is 402 out of 555 data

This means that the Logistic regression model has found 155 of (0) class and 153 of (1) class wrongly. We can say that we have decreased the amount of the prediction wrongly in the class (1) and (2) when compared to the both the models created.

**9.10.4 Random Forest model**

Random Forest is among the most well-known and widely utilised algorithms in real-world data science projects and competitions. The objective of this narrative is to explain this popular algorithm in simple words. The Random Forest Algorithm is an ensemble model, which means that more than one model is constructed during the prediction process. In the case of the random forest, numerous decision tree models are constructed, and the final forecast is based on the average of those independent decision tree projections.

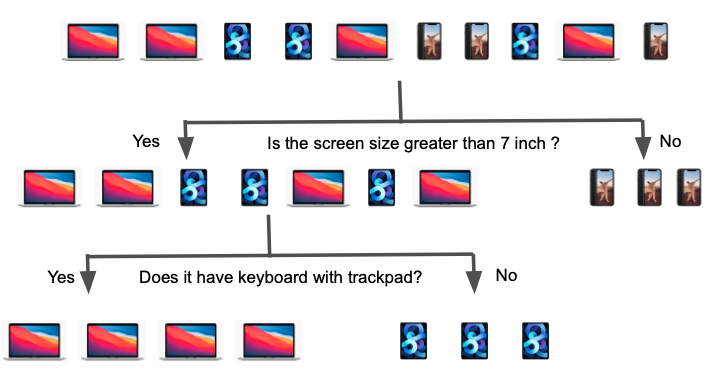


The random forest approach will employ various subsets of data and characteristics for each decision tree to improve the performance of the model output. This guarantees that the model is free of bias since no single segment of data or group of characteristics dominates the prediction outcomes. It also guarantees that each model is distinct from the others; otherwise, having so many models would be pointless. If the individual models were developed using the same dataset and attributes, the predictions would be quite similar no matter how many separate models we built. Because it employs diverse subsets of data and characteristics, the random forest technique prevents overfitting and produces distinct uncorrelated models.

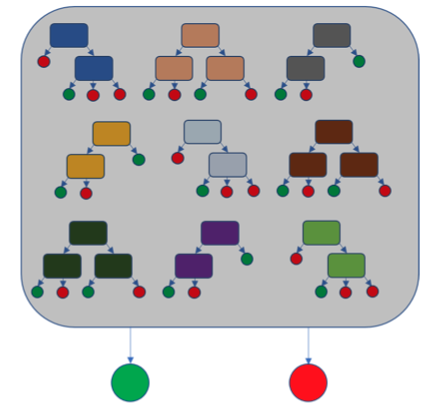
**Decision Trees:**

A basic decision tree algorithm is shown below. Let’s use a decision tree algorithm in the classification of iPhones, iPads, and MacBooks. A decision tree is a straightforward rule-based algorithm. It classifies a dataset through a series of questions, that is called the decision nodes.

In the below example, the screen size is first used to isolate the iPhones from iPads and MacBooks. Then we can separate the iPads from the MacBooks by checking for the presence of a keyboard and trackpad.



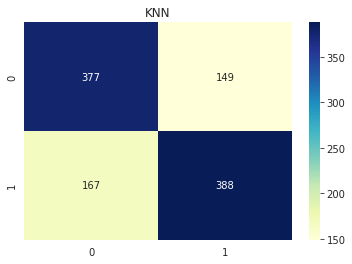
In the case of a classification challenge, we would also use a series of questions to categorise the dataset into distinct groups. To produce a better forecast, we will construct several such decision trees in a random forest. The random selection of data subsets and attributes ensures that each individual decision tree in the random forest is unique. The graphic below is a representation of how the various trees might appear in a random forest algorithm.



**Accuracy score for training data (Randon Forest) 0.7507936507936508**

**Accuracy score for testing data (Randon Forest): 0.7076780758556892**

| **Target coupon** | **Precision** | **Recall** | **F1-score** | **Support** |
| --- | --- | --- | --- | --- |
| 0 | 0.69 | 0.72 | 0.70 | 526 |
| 1 | 0.72 | 0.70 | 0.71 | 555 |
|  |  |  |  |  |
| accuracy |  |  | 0.71 | 1081 |
| macro avg | 0.71 | 0.71 | 0.71 | 1081 |
| weighted avg | 0.71 | 0.71 | 0.71 | 1081 |

****

Chart, line chart

Description automatically generated

When compared to prior models, the Random Forest model had a training data accuracy of 75% and a testing data accuracy of 70%. However, the model is rather poor at determining which coupons are accessible and which are not. Furthermore, we can observe that the training accuracy is larger than the testing accuracy, indicating that the model is overfit in attempting to predict the training data. To ensure the model's robustness, we must aim for 100% accuracy while avoiding overfitting.

The **confusion matrix** says that

* Correct Detection of Coupon not given (0) is 377 out of 526 data
* Correct Detection of Coupon given (1) is 388 out of 555 data

This suggests that the Random Forest model incorrectly identified 167 of the (0) class and 149 of the (1) class. We may state that we incorrectly reduced the quantity of prediction in the classes (1) and (2) as compared to all prior models. To ensure that the model is accurate, we must limit the number of false positives and false negatives. Even the roc graph indicates that the auc curve has a value of 0.77. However, the graph's curve begins at 0.4. As a result, we cannot use random forest to deploy the model.

**9.10.5 Support Vector machine**

Support Vector Machine (SVM) is a supervised machine learning technique that may be used for classification as well as regression. Though we call them regression issues, they are best suited for categorization. The SVM algorithm's goal is to identify a hyperplane in an N-dimensional space that clearly classifies the input points. The size of the hyperplane is determined by the number of features. If there are just two input characteristics, the hyperplane is simply a line. If the number of input characteristics is three, the hyperplane becomes a two-dimensional plane. When the number of characteristics exceeds three, it becomes impossible to imagine.

Consider two independent variables, x1 and x2, and one dependent variable, either a blue or a red circle**.**

Chart, bubble chart

Description automatically generated

The graphic above clearly shows that there are several lines (our hyperplane here is a line because we are just examining two input characteristics x1, x2) that separate or classify our data points into red and blue circles.

**Kernel for SVM:**

The SVM kernel is a function that changes a low-dimensional input space into a higher-dimensional space, i.e. it converts a non-separable issue into a separable problem. It is highly effective in nonlinear separation situations. Simply said, the kernel does some fairly sophisticated data transformations before determining how to split the data depending on the labels or outputs specified.

**SVM advantages:**

* In high-dimensional scenarios, it is effective.
* It saves memory by using a subset of training points in the decision function known as support vectors.
* Different kernel functions can be given for the decision functions, as well as bespoke kernels.

**Accuracy score for training data (SVM) 0.7670634920634921**

**Accuracy score testing data (SVM): 0.7150786308973173**

| **Target coupon** | **Precision** | **Recall** | **F1-score** | **Support** |
| --- | --- | --- | --- | --- |
| 0 | 0.71 | 0.71 | 0.71 | 526 |
| 1 | 0.72 | 0.72 | 0.72 | 555 |
|  |  |  |  |  |
| accuracy |  |  | 0.71 | 1081 |
| macro avg | 0.71 | 0.71 | 0.71 | 1081 |
| weighted avg | 0.72 | 0.72 | 0.72 | 1081 |

**Chart, treemap chart

Description automatically generated**

When compared to previous models, the SVM model exhibited 76% training data correctness and 71% testing data accuracy. However, the model is weak at distinguishing which coupons are and are not available. Furthermore, the training accuracy is greater than the testing accuracy, indicating that the model is overfit when attempting to predict the training data. We must aim for 100% accuracy while avoiding overfitting to ensure the model's durability.

The **confusion matrix** says that

* Correct Detection of Coupon not given (0) is 371 out of 526 data
* Correct Detection of Coupon given (1) is 402 out of 555 data

This implies that the Support vector machine model misidentified 155 members of the (0) class and 153 members of the (1) class. We may say that we underestimated the amount of prediction in classes (1) and (2) when compared to all past models. We must restrict the number of false positives and false negatives to guarantee that the model is accurate.

**9.10.6 Voting classifier**

Voting Classifier is a machine-learning technique that data scientists frequently employ to improve the performance of their models and move up the rank ladder. Voting Classifier may also be used to increase performance on real-world datasets, although it has significant limitations. The model's interpretability suffers since it cannot be interpreted using the shap or lime packages.

Unlike previous models, Scikit-learn does not include a solution for computing the top-performing features for the voting classifier, but I have devised a workaround. The feature significance may be calculated by adding the important scores of each of the estimation methods depending on the weights.

A voting classifier is a machine learning predictor that trains many base models or estimators and predicts by aggregating their results. Aggregating criteria can be coupled voting decisions for each estimator output. There are two sorts of voting criteria:

1. Hard Voting: Voting is determined based on the projected output class.
2. Soft voting: Voting is based on the output class's projected probability.

**How can a Voting Classifier increase performance?**

The voting classifier collects the projected class or probability based on hard or soft voting. So, if we provide a choice of base models to the voting classifier, it ensures that the mistake is resolved by any model.

Diagram

Description automatically generated

**Implementation**

We are training four basic estimators for our Coupon classification dataset: Logistic Regression, Random Forest, KNN, and Support Vector machines Classifier.

The voting='soft' or voting='hard' parameters allow developers to choose between hard and soft vote aggregators. Users can adjust the parameter weight to overshadow some of the better performing base estimators. The weighting sequence used to weight the occurrences of anticipated class labels in hard voting or class probabilities prior to averaging in soft voting.

We're employing a soft voting classifier with a weight distribution of [1,1,2,1], with the Random Forest model getting twice the weight. Now consider the benchmark performance of each of the basic estimators in comparison to the voting classifier.

| Model | Training Accuracy | Testing Accuracy | Precision | Recall | F1-score |
| --- | --- | --- | --- | --- | --- |
| Knn | 0.92 | 0.89 | 0.96 | 0.8 | 0.87 |
| Logistic Regression | 0.74 | 0.76 | 0.74 | 0.77 | 0.75 |
| Random Forest | 0.71 | 0.71 | 0.67 | 0.78 | 0.72 |
| SVM | 0.74 | 0.76 | 0.74 | 0.76 | 0.75 |
| Voting Classifier | 0.86 | 0.84 | 0.84 | 0.83 | 0.83 |

According to the lovely table above, the voting classifier improves performance over the base estimator.

Chart, treemap chart

Description automatically generated

When compared to earlier models, the voting classifier model performed 86% correctly on training data and 84% correctly on testing data. However, the model performed better than the prior model's analysis. Furthermore, with just 2% difference, the training accuracy outperforms the testing accuracy, demonstrating that the model is not overfit or underfit when attempting to predict the training data. Despite our greatest efforts, we are just 14% of the way to achieving 100% accuracy. As a result, for the following model, we'll attempt the grid search strategy.

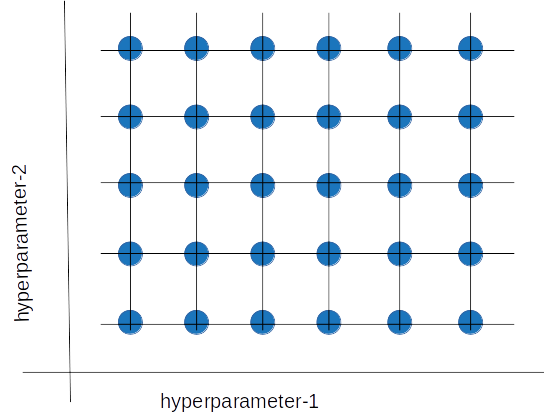
The **confusion matrix** says that

* Correct Detection of Coupon not given (0) is 411 out of 526 data
* Correct Detection of Coupon given (1) is 472 out of 555 data

This means that the Voting Classifier model misidentified 115 (0) class members and 83 (1) class members. When compared to all previous models, the model has provided the highest accuracy in quantity of prediction in classes (1) and (2). To ensure that the model is correct, we must limit the number of false positives and false negatives.

**9.10.7 Grid search/ Hyper parameters tunings and Voting classifier**

**Hyper Parameters tuning -** Finding a set of ideal hyperparameter values for a learning algorithm and applying this improved algorithm to every data set is what hyperparameter tuning entails. This set of hyperparameters optimises the model's performance by minimising a predetermined loss function in order to deliver better outcomes with fewer mistakes.



**Methods of Hyperparameter tuning**

**Grid Search -** One way is to test out several numbers and then select the value with the highest score. This is referred to as a grid search. If we had to choose values for two or more parameters, we would consider all possible combinations of the sets of values, generating a grid of values.

Grid search is a "brute force" approach for tweaking hyperparameters. We generate a grid of discrete hyperparameter values and then fit the model with every feasible combination. We record the model performance for each set and then choose the combination that generated the greatest results.

Grid search is an exhaustive technique that can identify the optimal hyperparameter combination. The disadvantage is that it is sluggish. Fitting the model with every potential combination generally necessitates a large amount of computer capability and time, which may not be accessible.

**Random Search:**

The random search technique, as the name suggests, selects values at random rather than utilising a preset set of values like the grid search method does.

In each cycle, random search attempts a different combination of hyperparameters and records the model performance. It returns the combination that gave the greatest outcome after multiple iterations.

When we have numerous hyperparameters with somewhat big search domains, random search is acceptable. We can build discrete ranges (for example, [5-100] in 5 step increments) and still have a reasonable collection of choices.

The advantage is that random search usually takes less time than grid search to provide equivalent results. It also assures that we don't wind up with a model that is skewed toward value sets picked at random by users. Its disadvantage is that the resulting hyperparameter combination may not be the best possible.

We utilised grid search to train the data in our model, and the hyperparameters we used are,

| Models | Hyper parameters | Best Hyper Parameter |
| --- | --- | --- |
| KNN | k Value 1 -5 | KNN = {'n\_neighbors': 1} |
| Random Forest | 'n\_estimators': [50, 100,200,250,300,350,400] | 'n\_estimators': 200 |
| Logistic Regression | {'C':[0.001,.009,0.01,.09,1,5,10,25,50,70]},scoring = 'recall' | C=0.001 |
| SVM | {'C': [0.1,1, 10], 'gamma': [1,0.1,0.01],'kernel': ['rbf', 'poly', 'sigmoid']} | SVC(C=10, gamma=1, probability=True) |

If you see the above table, we used 4 models for the grid search and the best hyper parameter that the grid search gave us for the model s

* KNN - K value = 1
* Random Forest - 'n\_estimators': 200
* Logistic Regression – C = 0.001
* SVM - SVC(C=10, gamma=1, probability=True)

The model I propose will take the best hyper parameters from each model and train the data. Following that, each trained model is fed into the voting classifier to obtain the average predictions and then tested with the testing data, yielding better outcomes than the other model.

Accuracy score for the training data after grid search(Voting\_model 0.9932539682539683

Accuracy score for the testing data after grid search(Voting\_model): 0.9167437557816837

| **Target coupon** | **Precision** | **Recall** | **F1-score** | **Support** |
| --- | --- | --- | --- | --- |
| 0 | 0.98 | 0.85 | 0.91 | 526 |
| 1 | 0.87 | 0.98 | 0.92 | 555 |
|  |  |  |  |  |
| accuracy |  |  | 0.92 | 1081 |
| macro avg | 0.92 | 0.91 | 0.92 | 1081 |
| weighted avg | 0.92 | 0.92 | 0.92 | 1081 |

Chart, treemap chart

Description automatically generated

Chart, line chart

Description automatically generated

When compared to all the models, the Grid search voting classifier model performed 99% correctly on training data and 92% correctly on testing data which is better than all the model which we built before. Furthermore, the precision, recall and F1 score is also progressed than compared to the previous models. Even though, the training accuracy outperforms the testing accuracy, the testing accuracy has performed 91% which is very rare in failing the prediction of coupon available or not available.

The **confusion matrix** says that

* Correct Detection of Coupon not given (0) is 447 out of 526 data
* Correct Detection of Coupon given (1) is 544 out of 555 data

This means that the Grid search voting classifier model misidentified 11 (0) class members and 79 (1) class members. When compared to all previous models, the model has provided the highest accuracy in quantity of prediction in classes (1) and (2). Also, the roc curve says the value of auc is 0.97 which is very near to 1.

Furthermore, the accompanying visualisation indicates that the Grid search classifier is superior in terms of accuracy, precision, recall, and F1 score. As a result, we may apply that strategy to launch and develop a web app for the organisation.

### **10. Choosing the best Model**

According to the research observations, the performance of the Logistic regression, SVM, and Xg boosting, KNN, Random Forest, and Voting classifiers is between 72 and 86%, which is adequate for creating and deploying the model. However, by reviewing the accuracy, recall, and F1 score, we can see that the Grid search classifier performs admirably for this dataset. Furthermore, using the model grid search classifier, we can state with 99% confidence if the consumer receives the coupon or not. As a result, I'm storing the model as a pickle file in order to deploy it on the web.

**11. Model Deployment and Web application in flask**

We often focus a lot on the EDA, Model Building part, but there is one big element that we tend to overlook, which is attempting to create an end-to-end implementation or deploying the model. After all of your hard work and efforts in developing your model, it is equally crucial that you provide it in an useful form that can be consumed directly by the end user. Furthermore, creating an application based on the code you've written allows you to exhibit your work more effectively and display it more effectively.

**Flask:**

Flask is a Python online application framework that allows end users to work with your Python script (in our example, our ML algorithms) straight from their internet browser without the need for any libraries, code files, etc.

Flask makes it simple to construct web apps, allowing you to spend your efforts on other key aspects of the ML lifecycle like as EDA, feature engineering, and so on. In this blog, I will guide you through the process of creating a small web application from your ML Model and finally deploying it.

**DIRECTORY STRUCTURE:**

First, let's look at our directory structure, which will provide us a better understanding of the whole project and is also important when dealing with flask. I've stored the project under a primary directory named **Coupon Prediction**.

Diagram

Description automatically generated

**Templates**: This folder includes the html code (index.html, predict.html) that will be utilised by our main file (app.py) to construct our application's front end.

**app.py**:- This is the primary application file that contains all of our code and connects together everything.

**model.pkl:**- This is the classification model that we will use; in this example, it is a Grid search Classifier Model that I had previously trained and provided us the best results of all the models.

**Model\_prediction.ipynb:** This is the Python environment in which we created the model, and we saved the best model as a pickle file from this environment as model.pkl.

**11.1 Steps to create the web app.**

**11.1.1 Importing the flask library and opening the model saved**

Graphical user interface, text

Description automatically generated

At the end of the day, this app is nothing more than an object of class Flask that does all the heavy lifting for us, such as handling incoming browser requests (in our case, the question that the user enters) and providing adequate answers (in this case, our model forecast) in some lovely form using html,css. All of this will be revealed gradually, as will how this app item fit into the picture. Additionally, the model saved in the pickle file is saved into the variable called model.

**11.1.2 Mapping with html and prediction**

Text

Description automatically generated

The first line @app.route ('/index') is a decorator, which simply maps the method described below it to the URL given within the decorator, i.e. anytime a user visits that URL '/' (the whole address would also contain an ip address and a port number, something like http://127.0.0.1:5000/), the index() method is automatically called, and the index() function returns our main HTML page named index.html (in our case index.html provides a text box to the user where he could enter his question)

The flask.render template() function looks for this index.html document in the templates folder that we generated in our current directory and automatically generates/renders an HTML page for the end user, which I will discuss in more detail later.

(Note: Because the word predict is used in several situations, I will refer to predict() as the forecast method created in our script, predict.html as the forecast template placed in our template folder, and /predict as the forecasting URL to prevent any confusion.)

Now we add another decorator @app.route ('/predict'), which maps the predict() function to the /predict URL. As the name implies, the predict() method takes the user's input, does all the pre-processing, builds the final feature representation, runs the model on it, and returns the final prediction.

Now, let us focus on the rendering dynamic HTML page part. Sometimes we need to put information dynamically in the HTML page, such as in this case, our predict.html page would require the prediction value to render the page properly, which is only available when the user enters the text and clicks the submit button in our index.html page, this is where the render template comes into play, in the predict.html page we already have created placeholders (will discuss this later).

Now, let's take a quick look at the predict() method in our main code. The first line inside the method, to predict list = request.form.to dict(), takes the consumer entered text, when user click on the submit button in our form, a json is returned containing key value pairs with responses, when the submit button is clicked, this /predict URL is called, and reactions are sent to the page correlating to this URL as a json.

Finally, we mentioned the render template within the predict() function, which states that if the target value 1 is returned as output, it will be replaced with "Hurrah..you have received a 25% discount," otherwise the output will be "Sorry..You will not receive any discount."

**11.1.3 OUTPUT and DEPLOYMENT**

Let us now have a look at the eventual result, how everything will appear if we deploy it, first on my local machine and then on a public server. Deployment is very simple; simply set up your Python environment and run the app.py application; because I already have Pycharm installed on my computer, I will run the app.py file from there; otherwise, you can create your own virtual environment, install all the dependencies, and run it from there; you do not need the entire Anaconda package to run it; you only need python and the libraries that were used in the code.

Text

Description automatically generated

Once you've started Pycharn, navigate to the main folder Coupon prdeiction in where everything is stored and execute the main python script file (app.py), just examine the final line that states running on http://127.0.0.1:5000/. Here is where our web app is running, you can copy this URL and paste it in your web browser to see the web application, here 127.0.0.1 is an ip address for locahost, i.e your own machine (just like we have a house address, an address is a unique address to recognise your machine on the internet or a local network), and 5000 is the interface that connects, this is where your web application resides on the server (think of it as a doorway to access your application). Let us now paste this URL into our browser and observe how the site appears.



Graphical user interface, website

Description automatically generated

Graphical user interface, application

Description automatically generated

The above images are the prediction pages in which all the main html, model.pkl, and app.py bind each other and give us the forum in which we can give our independent variable values and predict whether a customer is eligible to get the coupon. Additionally, I have given some sample values inside all the variable boxes and the web app gave us the results of that customer got the coupon which is mentioned in the above image.

### **12. CONCLUSION:**

**12.1 RESEARCH PROJECT OBJECTIVES**

All the objectives were completed with the completion of this study, and the project will contribute to the current research on the use of data to anticipate the coupon.

**12.2 PROJECT PROBLEM STATEMENT**

This research study was able to investigate the contribution of the new digital era to the issue of reputation measurement in market research. The research project provided information on the viability of using new data gathering and analysis technologies as alternatives to traditional research methods like surveys, in-depth interviews, focus group discussions, and so on. The study project's techniques and methodologies addressed all the issues encountered during market research, including such deploying a web app for machine-learning models.

### **12.3 EVALUATION**

as the Amazon dataset. It allows us to study the link between each feature and the variables and then use the correlations to make more accurate forecasts. In this work, we trained a variety of models, including logistic regression, random forest, Knn, and decision tree models.

To improve efficiency, we also employed grid search techniques to tune the hyper parameters in our features. Finally, we compared the performance of the models on the test dataset and discovered that the grid search classifier produced the best results because it integrates certain reasonable qualities extracted from the dataset's rich information. Furthermore, the flask-based web application worked well in terms of effectively executing the model. Additionally, we deployed the flask application on the web to get the predictions.

**13. Future Work**

1. <https://muthu.co/understanding-the-classification-report-in-sklearn/#:~:text=A%20Classification%20report%20is%20used,classification%20report%20as%20shown%20below>. – classification
2. <https://www.nvidia.com/en-us/glossary/data-science/xgboost/#:~:text=XGBoost%2C%20which%20stands%20for%20Extreme,%2C%20classification%2C%20and%20ranking%20problems>. – Xgboosting
3. <https://machinelearningmastery.com/confusion-matrix-machine-learning/> - confusion matrix
4. <https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5#:~:text=ROC%20is%20a%20probability%20curve,and%201%20classes%20as%201>. – roc curve
5. <https://stackabuse.com/k-nearest-neighbors-algorithm-in-python-and-scikit-learn/> - Knn
6. <https://www.mygreatlearning.com/blog/knn-algorithm-introduction/> - Knn
7. <https://careerfoundry.com/en/blog/data-analytics/what-is-logistic-regression/> - Logistic regression
8. <https://towardsdatascience.com/random-forest-in-simple-english-why-is-it-so-popular-3ba04d0374d> - random forest model
9. <https://www.geeksforgeeks.org/support-vector-machine-algorithm/> - SVM
10. <https://towardsdatascience.com/use-voting-classifier-to-improve-the-performance-of-your-ml-model-805345f9de0e> - voting classifer
11. <https://www.anyscale.com/blog/what-is-hyperparameter-tuning> - grid search and hyper parameter tuning
12. <https://towardsdatascience.com/model-deployment-using-flask-c5dcbb6499c9#:~:text=Deployment%20is%20very%20simple%20you,from%20there%2C%20you%20don't> – model development