Purchasing Intention: Building a Predictive Model Using Machine learning

Report
Mohamed Shabanpour

Content

Executive Summary

In recent years, extensive studies from different viewpoints have been conducted to understand online consumer behavior owing to the exponential growth of online retailing in both developed and developing economies. However, one of the crucial problems of online retailers remains unanswered: why do most visitors abandon their shopping carts and leave stores without generating revenue. One way to cope with this problem is to evaluate online visitors' behavior so that managers and marketers reconsider their decisions to have a greater conversion rate. This study aims to investigate online users' behavior based on an online retailer's Clickstream and Web Analytics data. First, the data were visualized to provide valuable insights for marketers. Subsequently, machine learning classification algorithms were applied to build a high-performance predictive model. In refining the model, oversampling technique, feature selection, model evaluations, and model selection were employed. The results suggest that a Random Forest model with a cross-validated recursive feature elimination (RFE) has the highest performance.

Introduction

Online retailing has experienced exponential growth Since the commercialization of the Internet in the early 1990s. According to United Nations Conference on Trade and Development, B2C e-commerce sales were estimated at $4.9 trillion in 2019, up 11% over 2018. Moreover, the 2020 covid-19 pandemic has shown the great potential of online retailing and risen the number of online shoppers around the globe. To gain the utmost advantage of this opportunity, marketers undertake costly activities such as segmentation, targeting, positioning, and use the four Ps to sell their products and services in a competitive environment (Rajamma, et al., 2009). However, many customers still experience frustrations through their online shopping and thus hesitate to make online product purchases (Cho, et al., 2006).

Unfortunately, online retailers have not been able to solve some of the significant issues that plague the market, like high exit rates, abandoning shopping carts, and online shopping hesitation. Cho et al. have defined online shopping hesitation as "postponing or deferring product purchases by having additional processing time before making final product-purchase decisions on the Internet." (Cho, et al., 2006)

There is an overwhelming consensus among researchers and managers that online businesses cannot use offline retailing strategies and require novel approaches. Compared to online shopping, where the interaction only occurred between a customer and a machine, in physical retailing, a salesperson diminishes perceived risk by interpersonal interaction, offering customized alternatives, facilitating the product return process, and increasing trust and security, which induce effective sales. Therefore, to tackle this problem, many e-commerce companies invest in behavioral prediction systems that imitate a salesperson's behavior in a virtual shopping environment (Sakar, et al., 2019) (Zhou, et al., 2007).

Moreover, the widespread availability of clickstream data enables practitioners to examine consumer search behavior in a large-scale field setting (Moe, 2003). The ability to know these behaviors has allowed online websites to optimize their contents, subdue online shopping hesitation, and generate more revenue. It is, therefore, essential to know whether a user visiting an e-commerce website will buy or abandon the store.

This project aims to build a predictive model using machine learning algorithms to determine online shoppers purchasing intention for an online retailer in Sweden. The data was provided by the website visitors from February 2018 to January 2020 inclusive, which involves both numerical and categorical values. Based on prior literature (Sakar, et al., 2019), variables are defined as follows:

Table1. Numerical features definitions (Sakar, et al., 2019)

Feature name	Feature description
Administrative	Number of pages visited by the visitor about account management
Administrative duration	Total amount of time (in seconds) spent by the visitor on account management related page
Informational	Number of pages visited by the visitor about Web site, communication and address information of the shopping site
Informational duration	Total amount of time (in seconds) spent by the visitor on informational pages
Product related	Number of pages visited by visitor about product related pages
Product related duration	Total amount of time (in seconds) spent by the visitor on product related pages
Bounce rate	Average bounce rate value of the pages visited by the visitor
Exit Rate	Average exit rate value of the pages visited by the visitor
Page Value	Average page value of the pages visited by the visitor
Special Day	Closeness of the site visiting time to a special day

Table2. Categorical features definitions (Sakar, et al., 2019)

Feature name	Feature description
OperatingSystems	Operating system of the visitor
Browser	Browser of the visitor
Region	Geographic region from which the session has been started by the visitor
Traffic Type	Traffic source by which the visitor has arrived at the Web site (e.g., banner, SMS, direct)
Visitor Type	Visitor type as ‘‘New Visitor,’’ ‘‘Returning Visitor,’’ and ‘‘Other”
Weekend	Boolean value indicating whether the date of the visit is weekend
Month	Month value of the visit date
Revenue	Class label indicating whether the visit has been finalized with a transaction

The proposed model gives the necessary insights into consumer behavior and buying patterns to the firm managers and enables them to use proper marketing strategies to generate more revenue.

Exploratory data analysis

Exploratory data analysis (EDA) empowers researchers to detect anomalies, get insights on various features, and substantiate assumptions using summary statistics and graphical representations. In the following sections, the author presents the EDA results using different techniques to obtain the most insights into the data.

Data Type:

The dataset of this study contains 599130 data categorized in 33284 unique sessions based on visitors' IP addresses, which are described by 18 features. Data types are given in the following table.

Table 3. Data Type

```
RangeIndex: 33284 entries
Data columns (total 18 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   Administrative           33284 non-null  int64
 1   Administrative_Duration  33284 non-null  float64
 2   Informational            33284 non-null  int64
 3   Informational_Duration   33284 non-null  float64
 4   ProductRelated           33284 non-null  int64
 5   ProductRelated_Duration  33282 non-null  float64
 6   BounceRates              33284 non-null  float64
 7   ExitRates                33284 non-null  float64
 8   PageValues               33284 non-null  float64
 9   SpecialDay               33284 non-null  float64
 10  Month                    33284 non-null  object
 11  OperatingSystems         33284 non-null  int64
 12  Browser                  33284 non-null  int64
 13  Region                   33284 non-null  int64
 14  TrafficType              33284 non-null  int64
 15  VisitorType              33284 non-null  object
 16  Weekend                  33284 non-null  bool
 17  Revenue                  33284 non-null  bool
dtypes: bool(2), float64(7), int64(7), object(2)
```

Summary Statistics:

Summary statistics are displayed in the following table to provide simple insight into the dataset. Standard deviation values imply high dispersion among some features. Also, comparing the mean, standard deviation, and max column of administrative duration, informational duration, Product-related duration, and page values reveals the presence of outliers, which must be considered in the analysis.

Table 4. Summary Statistics

	count	mean	std	min	25%	50%	75%	max
Administrative	33284.0	2.250721	3.227397	0.0	0.000000	1.000000	3.000000	32.00000
Administrative_Duration	33284.0	80.326337	172.547820	0.0	0.000000	14.000000	100.000000	3502.39000
Informational	33284.0	0.502734	1.269416	0.0	0.000000	0.000000	0.000000	22.00000
Informational_Duration	33284.0	35.011617	141.705620	0.0	0.000000	0.000000	0.000000	3001.00000
ProductRelated	33284.0	31.568592	44.961744	0.0	7.000000	17.000000	37.000000	834.00000
ProductRelated_Duration	33282.0	1199.989650	1959.346666	0.0	182.503571	598.828571	1460.489029	74053.52223
BounceRates	33284.0	0.023054	0.049480	0.0	0.000000	0.003704	0.018182	0.85100
ExitRates	33284.0	0.045283	0.051503	0.0	0.014286	0.026194	0.050000	0.85452
PageValues	33284.0	9.838049	807.331141	0.0	0.000000	0.000000	0.000000	147258.00000
SpecialDay	33284.0	0.066781	0.193133	0.0	0.000000	0.000000	0.000000	1.00000
OperatingSystems	33284.0	2.029173	0.786164	1.0	2.000000	2.000000	2.000000	6.00000
Browser	33284.0	2.111826	1.212904	1.0	1.000000	2.000000	2.000000	9.00000
Region	33284.0	2.383127	2.403755	1.0	1.000000	1.000000	3.000000	12.00000
TrafficType	33284.0	3.835927	3.503519	1.0	2.000000	2.000000	4.000000	18.00000

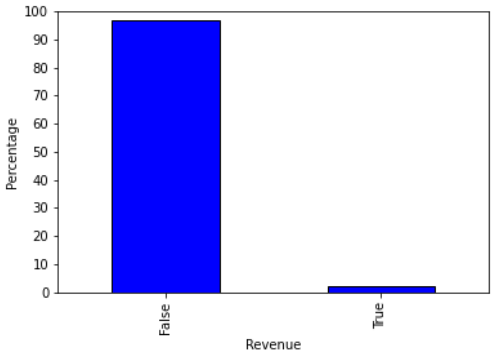
Data Distribution

In this section, data distribution of all features was visualized. Subsequently, the relationships among relevant independent variables and Revenue, which is the target variable of this study, was examined.

Conversion rate:

The following figure reveals that the conversion rate of the online retailing website is 1.20%. Moreover, it shows that the dataset is highly imbalanced, which must be addressed before building machine learning models to increase their performance.

Figure 1



Visitor Type:

It is apparent in the following figures that returning visitors make the most significant proportion of visitor types' population. More precisely, 86.01% of the visitors are returning ones, while the two other types only account for 13.99% of the visitors.

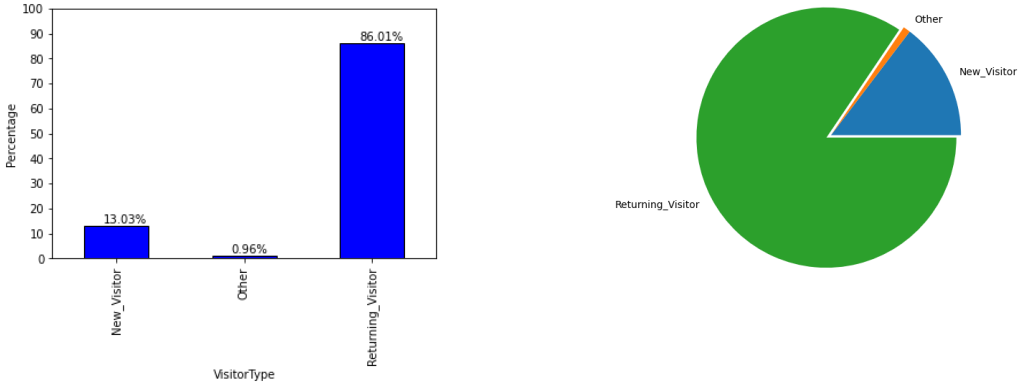
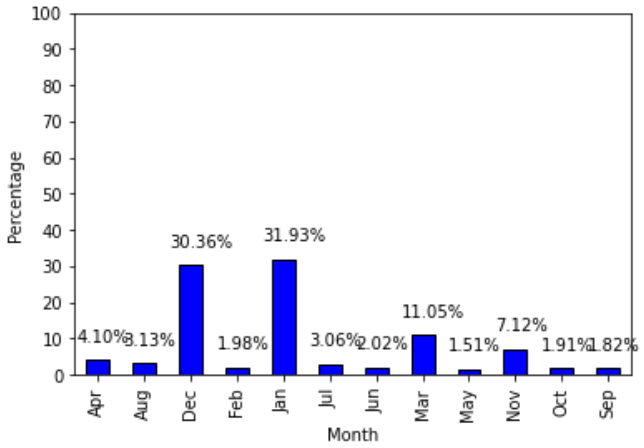


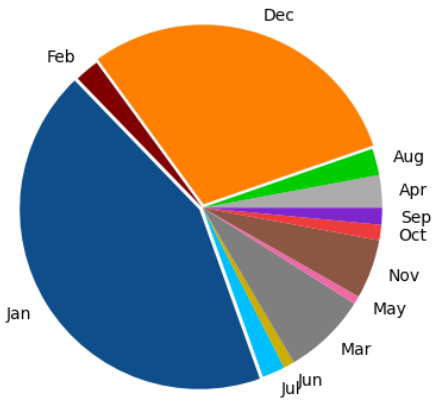
Figure 2

Month:

Figure 3 reports that visitors go to the online store in January and December more often than the other months. It also reveals that January, December, and March constitute more than 73% of the months when visitors look through the store.

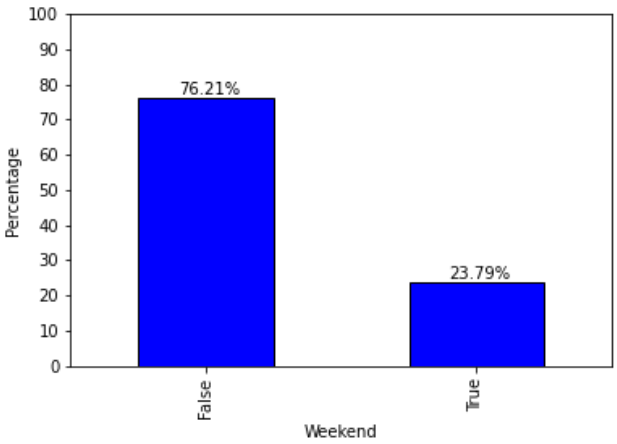
Figure 3





Weekend:

Based on the subsequent figures, more than three-quarters of the potential customers visit the website on weekdays.



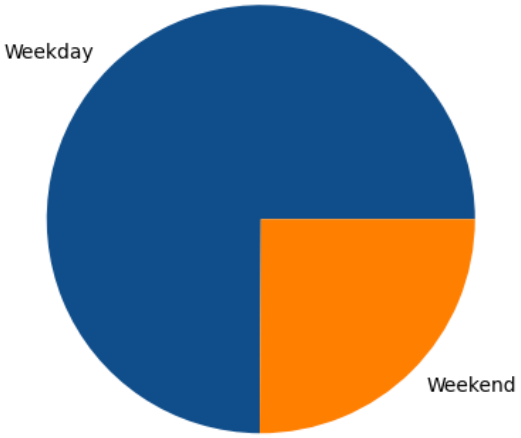
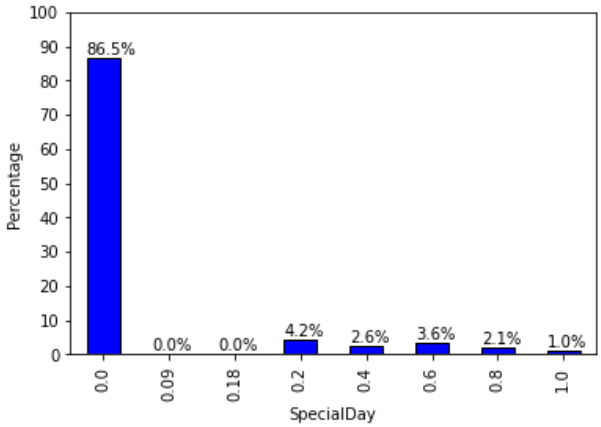


Figure 4

Special Day:

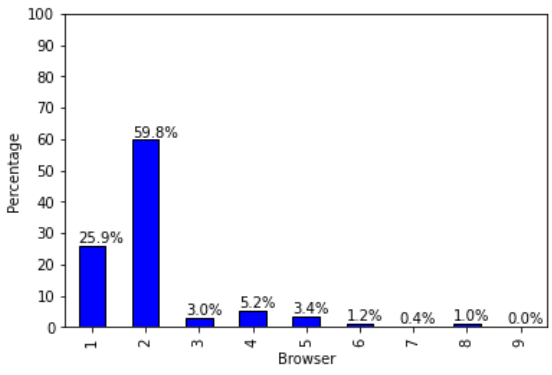
The Special Day feature indicates the closeness of the session visit to a particular day, such as Valentine's Day, Norooz Holliday, and Mother's day. One can be discerned from Figure 6 that 86.5% of sessions are not related to a Special Day, 6.8% are barely close to them, and 6.7% are near or on a Special Day. This observation implies that this feature may not be a compelling incentive for visitors to explore the online store.

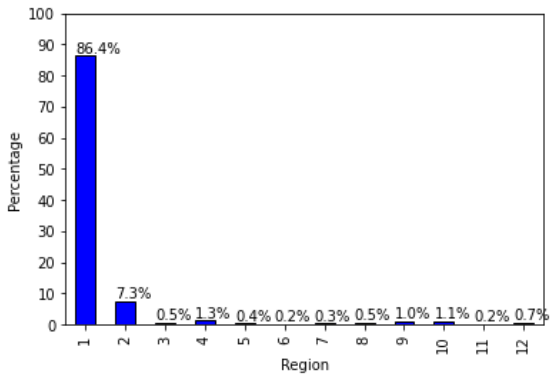
Figure 5



Browser, Region, Operating Systems, and Traffic Type:

For the sake of confidentiality, the retailer asked the author not to use the actual name of these features. Consequently, the variables are solely visualized by simple bar charts and introduced by numeric labels.





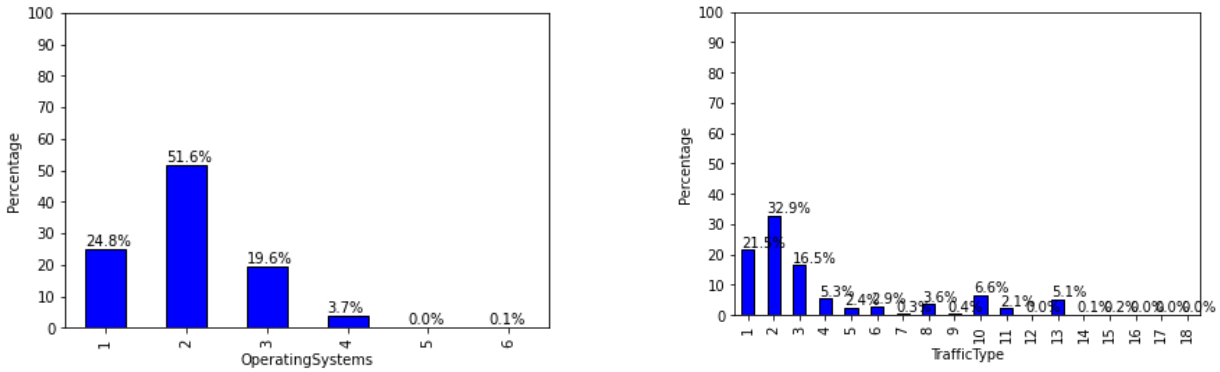


Figure 6

Density Plots:

To study the distribution of continuous numerical predictors, density plots were drawn.

Administrative page:

Figure 7 depicts the decreasing trend in the number of administrative page visits. It can be concluded that it is improbable for a visitor to explore many account-related pages.

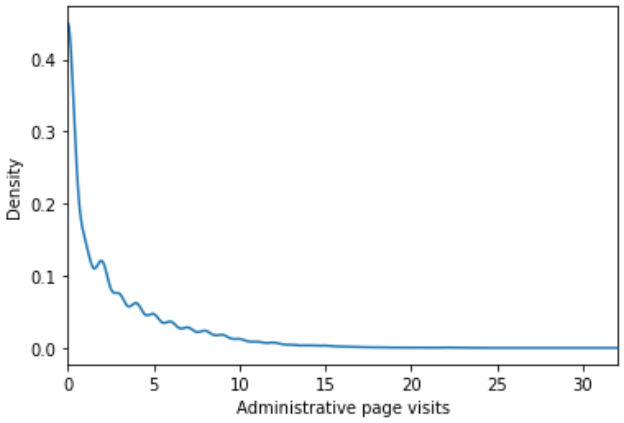


Figure 7

Informational page:

The following ripple figure illustrates the density of visited pages that provide information on the online retailer. It represents that the number of informational page visits undulates for a small number of visits. Also, its right-skewed shape demonstrates a decline in density as Informational Page visits rise.

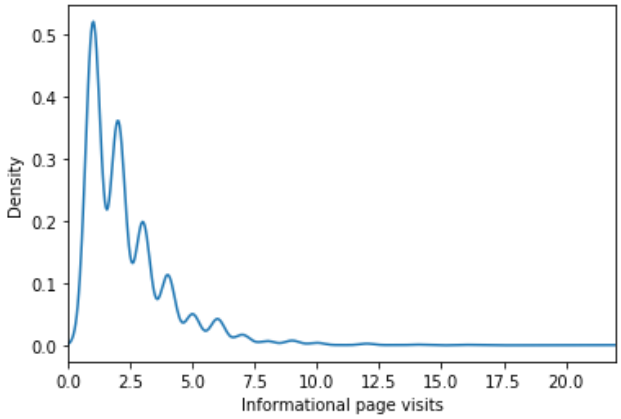


Figure 8

Product related page:

It is evident from Figure 9 that the number of Product-Related page visits is a decreasing trend. However, this sharp downtrend is not a serious issue for the online retailer, as there are significant outliers—a fair share of visitors views about ten Product-Related pages.

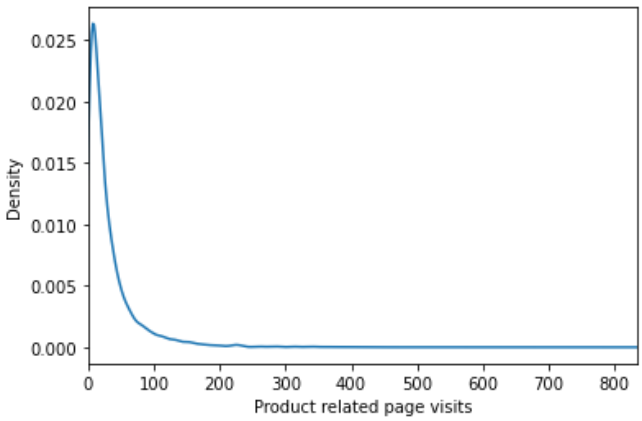


Figure 9

Bounce rate

The bounce rate plot depicts a relatively decreasing trend. The high density of zero bounce rate reveals that shoppers often visit a page, which is not the landing page. It could also be explained by visiting the online retailer's inner pages through different advertising intermediaries or word-of-mouth marketing rather than the landing page. Moreover, there may be some technical problems; according to Google Analytics, a zero bounce rate might occur if the web pages receive traffic from external sources, redirect

issues, third-party plugins, or duplicate tags. Although the density is very low, it should be noted that there are slight upticks near 0.1, 0.2, and 0.45 values, which indicates a fair percentage of bounce rate.

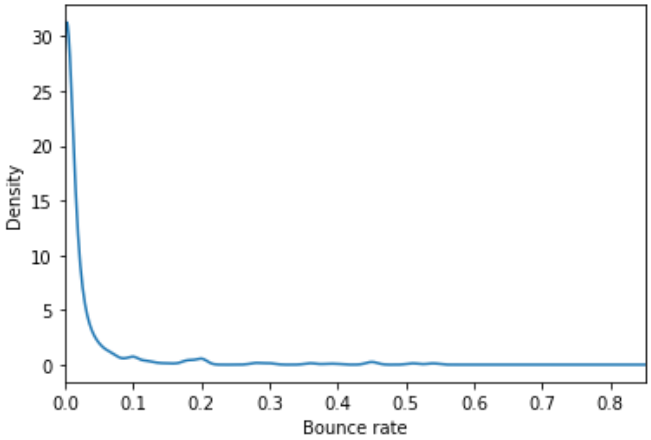


Figure 10

Exist rate:

For all pageviews to a page, Exit Rate is the percentage of times the page was the last in a session. As the resulting figure suggests, lower Exit Rates are pretty widespread among visitors. Also, there are some upticks, the highest of which is around 0.1, remarking that some pages have a higher Exit Rate than others. These upticks are related to “order confirmation” and “thank you” pages.

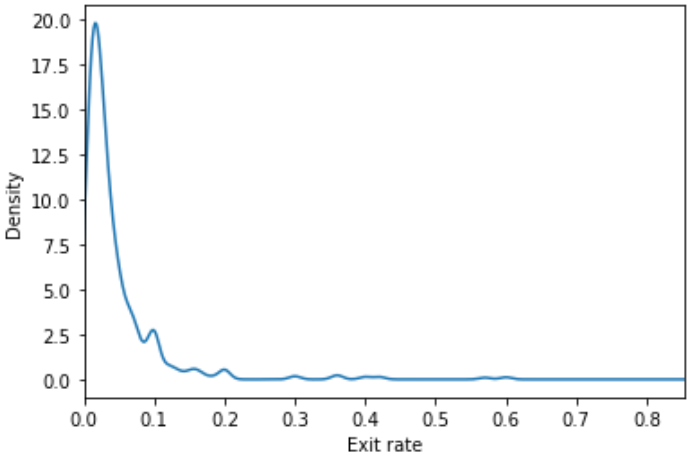


Figure 11

Customers Purchasing Intentions Determinants:

In this study, customers purchasing intentions are examined by generating Revenue for the company. Thus, in this section, the author visualizes the impacts of distinct predictors on Revenue.

Visitor Type and Revenue:

It can be deduced from Figure 12 that most of the sessions are Returning Visitors. 1.32% of Returning Visitors made the purchase, as compared to 0.68% of New Visitors. Also, Table 5 explains that more than 80% of the buyers are Returning Visitors. Therefore, considering the high potential buyers in this category, it is necessary for the online retailer to redesign its marketing strategy in order to motivate the non-buyer Returning Visitors to order products and build customer loyalty and retention.

Figure 12

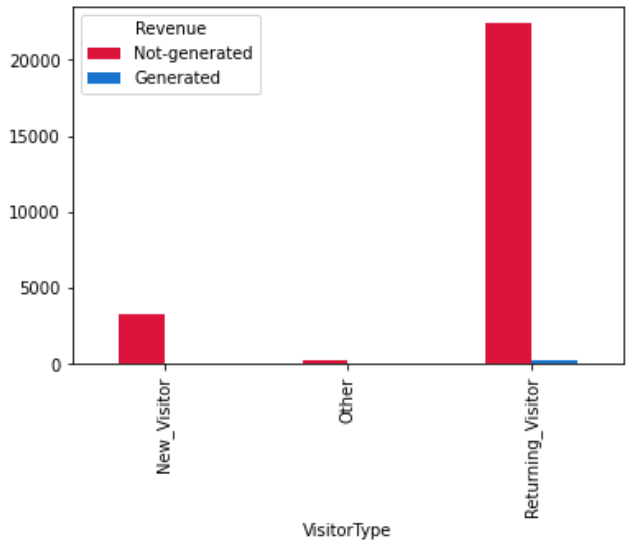


Table 5

Visitor Type	Percentage of total buyers
New Visitor	14.64 %
Other	0.89 %
Returning Visitor	84.47 %

Weekend and Revenue:

Of the 25366 visitors on Weekdays, 1.39% complete their shopping. Also, 7918 shoppers visit the website on Weekends, 0.63% of which made a purchase. Moreover, Table 6 states that almost three-quarters of the total buyers do shopping on Weekdays. Therefore, the retailer should consider various strategies regarding Weekdays to enhance the conversion rate and boost customer satisfaction and loyalty. Also, by different discount strategies, gamification, and coupon marketing, the company can motivate customers to increase the conversion rate on Weekends.

Figure 13

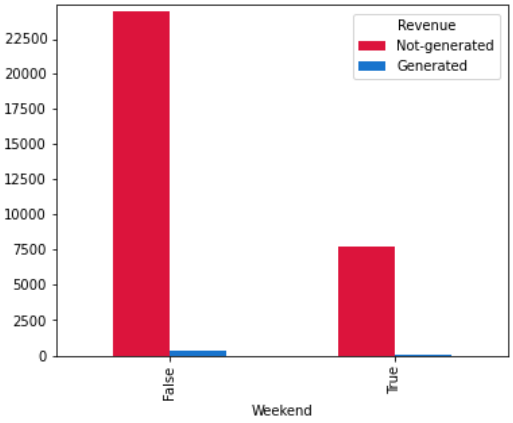


Table 6

Weekend	Percentage of total buyers
Weekday	74.92 %
Weekend	25.08 %

Month and Revenue:

Most of the customers visited the online store in December, January, and March, respectively. The highest number of orders have been made in January. In percentage terms, 1.62% of January, 1.15% of December, and 0.81% of March visitors buy products. The Month of May has the least number of sessions and, along with June, has a nominal purchase rate compared to other months. Also, Table 7 indicates that January and December are account for 72.95% of total buyers. Thus, the retailer should optimize its marketing strategies for these Month's users and offer various incentives to customers such as discounts, freebies, and reward points in other months.

Figure 14

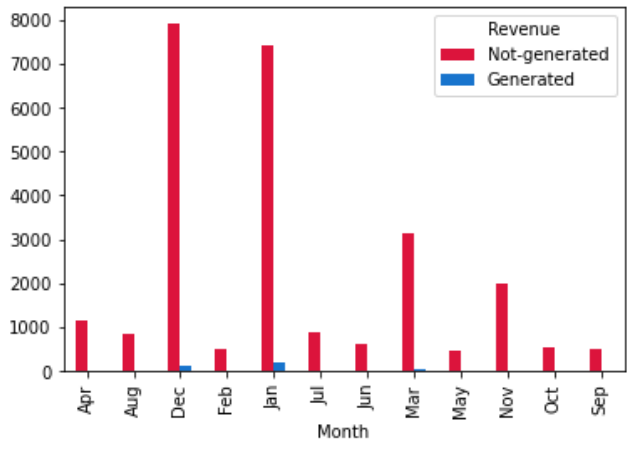


Table 7

Month	Percentage of total buyers
Jan	43.24 %
Feb	2.19 %
Mar	7.54 %
Apr	2.92 %
May	0.73 %
June	0.97 %
Jul	2.06 %
Aug	2.40 %
Sep	1.53 %
Oct	1.38 %
Nov	5.33 %
Dec	29.71 %

Region and Revenue:

Figure 15 depicts that most of the revenue generation occurred in Region 1 and 2, respectively. Out of 86.4% of total visitors in Region 1, 1.22% order products. Moreover, Region 2 accounts for 7.3% of total visitors, 1.15% of which contributes to revenue generation. Also, except Regions 4 and 10, all other Regions have negligible participation in revenue generation. In addition, based on the information in Table 8, almost 88% of total buyers live in Region 1, which implies that target customers live there. Hence, the retailer needs to study this Region in-depth to find its occupants' needs and desired delivery system.

Figure 15

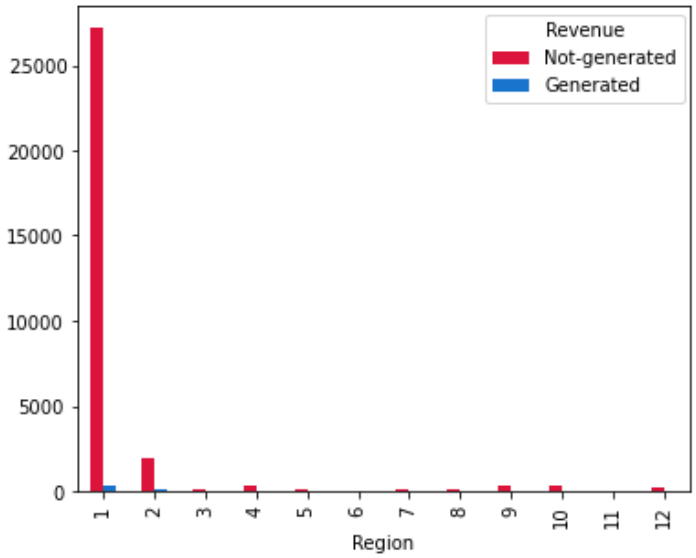


Table 8

Region	Percentage of total buyers
1	88.20 %
2	6.92 %
3	0.49 %
4	1.15 %
5	0.27 %
6	0.22 %
7	0.29 %
8	0.59 %
9	0.69 %
10	0.88 %
11	0.15 %
12	0.15 %

Traffic Type and Revenue:

Most revenue is generated by Traffic Type 2, followed by 1 and 3. 1.19% of Traffic Type 2 users, 0.84% of Type 1, and 1.17% of Type 3 made a purchase. It should also be noted that 2.51% of 2191 visitors who use Traffic Type 10 complete their shopping. Traffic Types 9, 12, 16, 17, and 18 have no participation in revenue generation.

Figure 16

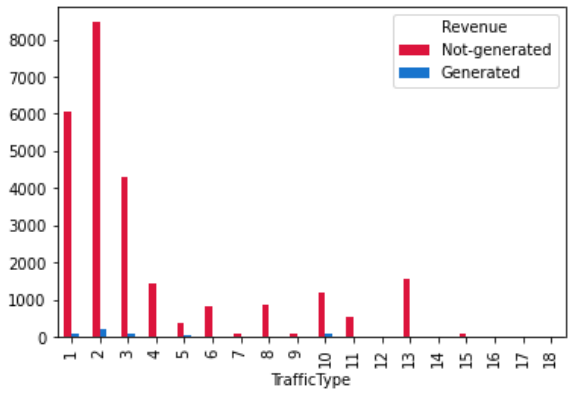


Table 9

Traffic Type	Percentage of total buyers
1	14.82 %
2	33.4 %
3	16.07 %
4	4.28 %
5	5.93 %
6	1.99 %
7	0.61 %
8	4.72 %
9	0.24 %
10	13.81 %
11	2.12 %
12	0.00 %
13	1.91 %
14	0.04 %
15	0.03 %
16	0.03 %
17	0.00 %
18	0.00 %

Operating systems and Revenue:

Operating system 2 has the highest number of users, 1.16% of which generate revenue for the retailer. Also, 1.66% of operating system 1 order products. Operating systems 5 and 6 almost have no contribution in revenue generation based on Figure 17. Moreover, considering the escalating usage of mobile phones, App marketing is essential for the retailer. This information enables the retailer to determine the most desirable operating system for investing in and designing its application.

Figure 17

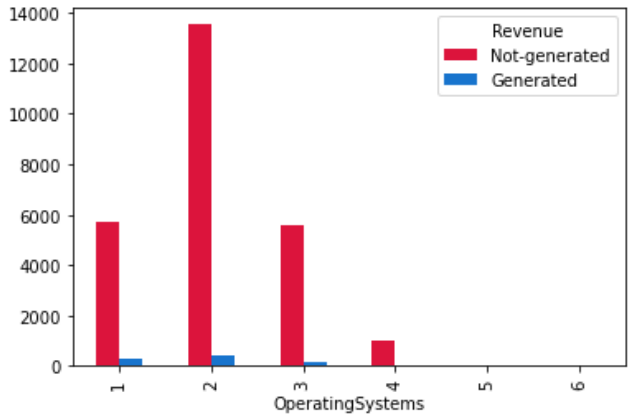


Table 10

Operating Systems	Percentage of total buyers
1	34.43 %
2	49.38 %
3	12.86 %
4	3.2 %
5	0.08 %
6	0.05 %

Browser and Revenue:

Browser 2 is the most popular browser among the online stores' visitors. 0.97% of 19893 sessions which utilize Browser 2 purchase. Also, 1.47% of browser 1 users generate revenue.

Figure 18

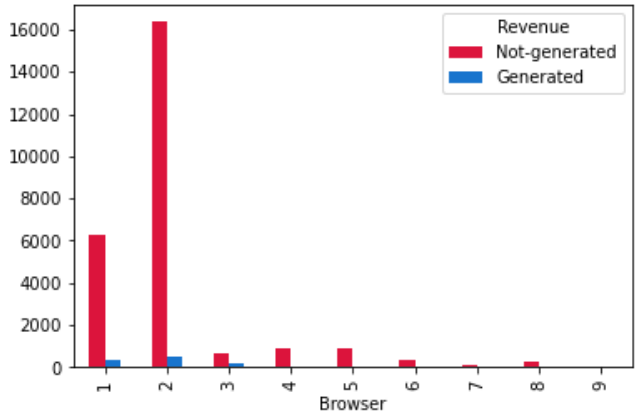


Table 11

Browser	Percentage of total buyers
1	31.63 %
2	47.85 %
3	10.38 %
4	4.83 %
5	3.3 %
6	0.77 %
7	0.36 %
8	0.87 %
9	0.01 %

Pair Plot:

A pair plot was drawn to get more insights into data and examine the correlation among independent variables, which is displayed in the following graph. The results are:

1- Visitors are more likely to purchase if the Page value is less than 150.
2- Exit Rate and Bounce Rate have a positive correlation. The probability of revenue generation is higher when the Exit Rate and Bounce Rate are low.

3- If product-related duration increase in a session, the number of pages related to products might also increase.
4- Exit Rate and Bounce rate are negatively associated with informational page duration.
5- Special Days may not have a significant impact on visitors' purchasing intention.
6- As we identified from the summary statistics table, there are several outliers in the variables.
7- Page Values have an inverse relationship with both Exit Rate and Bounce Rate.

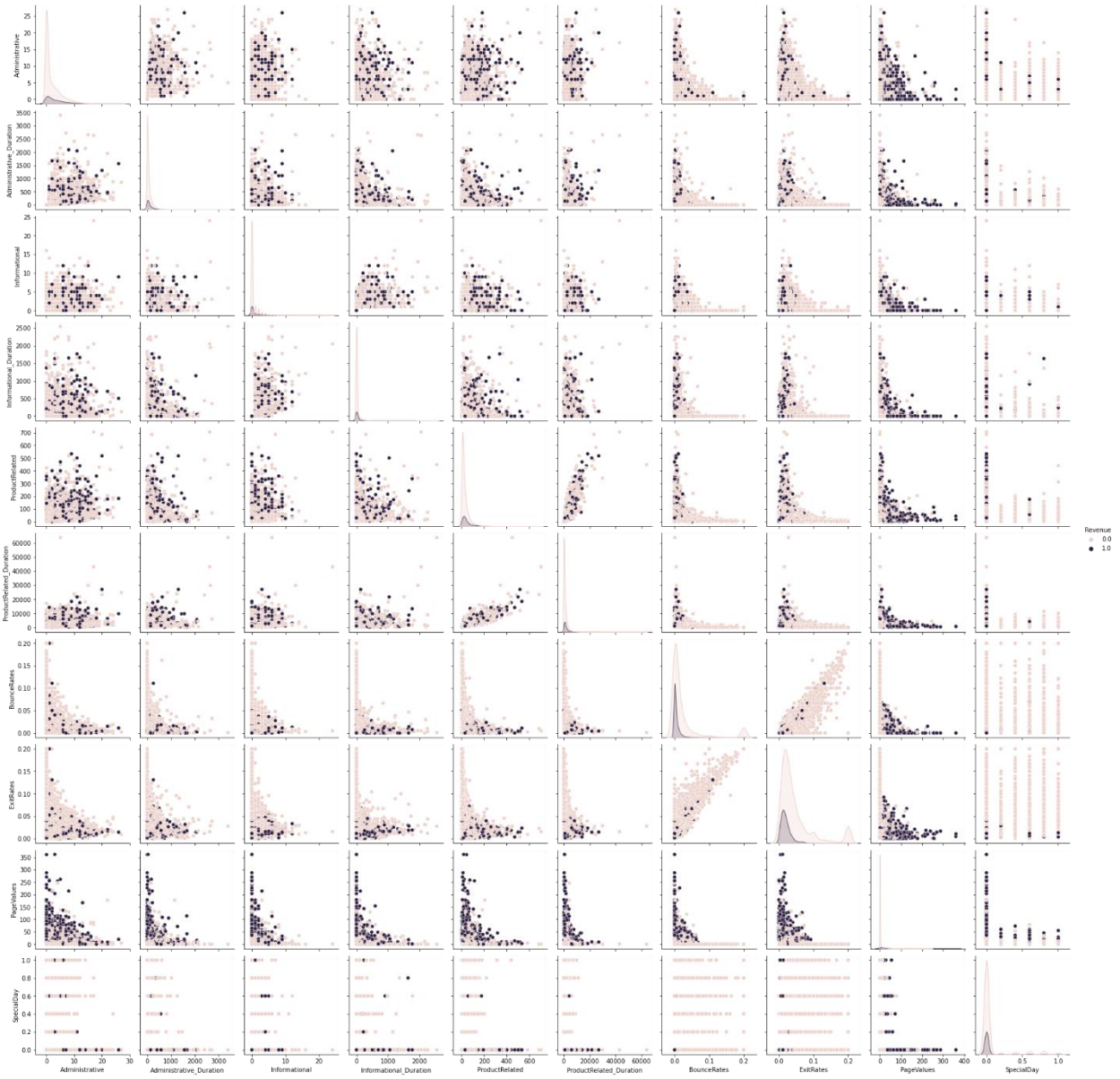


Figure 19

Data Preprocessing:

Categorical Variable:

Prior to dealing with outliers, the categorical data should be converted into identical numerical values so that it becomes interpretable for the machine learning models. The factorization technique was adopted to operate this conversion.

Outlier Treatment:

The author employs two different techniques to deal with the outliers. First, Box-Cox transformation was utilized to remove outliers and create a relatively normal distribution of the data. However, the result was not sufficiently satisfying. As a result, the second method, the interquartile rule, was applied to find outlier variables and treat them.

Imbalance Treatment:

Machine learning algorithms make more robust predictions after being trained with balanced data. In other words, standard machine learning techniques cause biased results when one class in the training set dominates the other. In the EDA section, a highly imbalanced distribution was recognized in the dependent variable, which gives rise to misinterpretation; out of the 33284 unique sessions, only 401 led to generating revenue. Therefore, to simultaneously balance the data and not lose any valuable information, the author oversamples the data using Synthetic Minority Oversampling Technique (SMOTE). SMOTE synthesizes new training records by randomly chooses a vector among a K nearest neighbors and a given data point of the minority class. Subsequently, it generates a unique data point by multiplying that vector by a random number between 0 and 1 and adding it to the given data point. This process results in a new balanced dataset with matching records for both classes in the refined feature.

Removing Junk Data

Pages in the online store are categorized into three groups: Administrative, Informational, and Product-Related. Hence, the author considers all sessions with Administrative Duration, Informational Duration, and Product-Related duration simultaneously equal to zero as junk data and eliminates them from the dataset.

Feature Selection

To develop a predictive model, it is reasonable to get rid of noisy data and redundant variables and find the best subset of predictors to decrease the computational cost and, at the same time, enhance the quality of the model. Feature selection process is applied to achieve the mentioned purposes. It improves the generalization of the model by lowering the probability of overfitting and avoid dimensionality. It should be conceded that a feature could be determining in one machine learning algorithm and useless in another. There is not an absolute proper feature selection technique for all predictive modeling problems. Hence, based on the dataset and the problem, the author employs three different feature selection techniques and compares the results to find the best.

In this research, two supervised and one unsupervised feature selection technique was used. First, Kbest feature, which is an unsupervised filter-based method, was employed using χ^2values. In this method, the relevance of the predictors outside of the predictive models are evaluated, and subsequently, determine the predictors that pass a specific criterion (Kuhn & Johnson, 2013). Here, χ^2 statistics between every feature and the Revenue were calculated, and their level of dependency was observed. Only were features chosen that rejected the null hypothesis, which is the independence of the feature and target variable.

Secondly, another supervised feature selection, Recursive Feature Elimination (RFE), a wrapper method selection, was exercised. Wrapper methods evaluate multiple models using procedures that add and/or remove predictors to find the optimal combination that maximizes model performance (Kuhn & Johnson, 2013). RFE is a greedy optimization algorithm that excludes the weakest features by repeatedly eliminating a few per loop after ranking them based on their attributes, such as feature importance.

In the end, correlations among the predictors were compared to decide the best subset. High correlations express multicollinearity, which brings about high standard errors of affected variables and subsequently causes overfitting. Therefore, Variance Inflation Factor (VIF) was used to detect this issue. Also, the target variable is ignored during the elimination of predictors; thus, the technique counts as unsupervised (Kuhn & Johnson, 2013).

Model Evaluation:

Machine learning models are helpful in the quality of their prediction. Therefore, every model needs to be evaluated to see how well it can predict unobserved data. Residual evaluation and cross-validation are the two most widely accepted techniques to assess

the model. Residual evaluation is not the best choice, for it is solely based on the observed data. Thus, in this study, two types of cross-validation are employed to check the quality of the predictions.

First, the hold-out method is operated. It is the simplest type of cross-validation by which the data set splits into two subsets: training and test set. The training set is what a given model is trained on, and the test set is utilized to understand how well the model performs on unseen data. The most commonly used split is 80% of the data for the training set and 20% percent allocate to the test set. The problem with this method is that it only relies on one split. That being said, when the timing is of the highest priority, the hold-out approach could be a choice.

Secondly, K-fold cross-validation, which is way more potent than other methods, is applied. In this method, the data set splits into k subsets, and then the hold-out method is employed k times. Every time it operates, one of the k subsets is considered the test set, and the other k-1 sets are put together as the training set and scored on the test set. After all k splits have been analyzed, the average of the scores will be computed. It is more accurate yet more time-consuming than other methods. In this research, the author uses a ten K-fold cross-validation.

Model Selection:

The various models built must be evaluated based on specific performance measures to identify the most robust one. Choosing the right model performance measures is critical since the dataset is highly imbalanced, and the conversion rate is 1.20%. Model accuracy alone may not be sufficient to assess a model. Hence, based on the confusion matrix, the following model performance measures have been used to build the most promising model.

		Predicted	
		Negative (N)	**Positive (P)**
Actual (Observation)	**Negative**	True Negative (TN)	False Positive (FP)
	Positive	False Negative (FN)	True Positive (TP)

Figure 20. Confusion Matrix

- **Accuracy:** It is the proportion of the total number of correct predictions. It is not the most valid measure to compare the models because of the low conversion rate of the data set.

$$(TP + TN)/(TP + FP + FN + TN)$$

That is, even if all the records are predicted as non-generated (0) by a model, it has an accuracy of 98.80%! As a result, other performance indicators must be taken into account.

- **Precision:** It is the ratio of all correctly predicted positive observations to all predicted positive observations.

$$TP/(TP + FP)$$

Here, Precision tells us what proportion of customers who are predicted to be revenue generators actually generate revenue. A lower Precision score implies higher wrong predictions.

- **Recall(TPR):** It is the ratio of all correctly predicted positive observations to the total observations in the actual class.

$$TP/(TP + FN)$$

In this report, Recall shows the ratio of actual buyers by the number of visitors who are correctly predicted as a revenue generator or not. The higher Recall score implies a better prediction of actual visitors who generate revenue.

- **F1 Score:** It is the weighted average of precision and recall. In other words, it is the harmonic average of the two mentioned indicators and gives better insights into the model performance.

$$2 * (Recall * Precision) / (Recall + Precision)$$

Here, a higher F1 Score implies that the model predicted fewer false positives and fewer false negatives, expressing better overall performance.

- **Specificity(TNR):** It is the proportion of the true negatives that are correctly identified.

$$TN / (TN + FP)$$

In this study, Specificity shows the ratio of visitors who are actually will not purchase by the number of visitors who are predicted as a non-revenue generator. The higher Specificity score, the more accurate the model is, for it predicts negatives more accurately.

- **FPR:** It is the proportion of the negative class that got incorrectly predicted by the model.

$$(FP) \ / \ (FP \ + \ TN)$$

Thus, a lower FPR rate implicates better performance.

- **FNR:** It is the proportion of the positive class that got incorrectly predicted by the model.

$$(FN) \ / \ (FN \ + \ TP)$$

Hence, a lower FNR score implies better performance to predict actual buyers or non-buyers.

- **AUC:** Receiver operating characteristic curve (ROC) is a graphical plot representing the diagnostic ability of a predictive model at all classification thresholds. The area under this curve is called AUC. In general, the higher AUC offers better performance.

- **MCC:** Matthews Correlation Coefficient (MCC) is an indicator that considers all the Confusion Matrix cells into consideration in one formula, making it so valuable.

$$((TP * TN) - (FP * FN))/\sqrt{(TP + FP) * (TP + FN) * (TN + FP) * (TN + FN)}$$

MCC values range between –1 and 1, where -1 means total disagreement between prediction and observation, and 1 signifies the complete agreement. Therefore, a higher MCC value indicates better performance.

The author runs the following machine learning algorithms:

1. Logistic regression
2. SVM
3. KNN
4. Gaussian Naive Bayes
5. Decision Tree
6. Random forest
7. Xgboost

Results:

Following data preprocessing, the baseline and feature-selected models are first created and evaluated using the hold-out method. RFE Random Forest is chosen as the best-performed model. Subsequently, the Kfold cross-validation method is applied to the models, where the random forest model with a cross-validated RFE has the highest performance rate. In the end, these two models were compared, and the latter is determined as the best model. The results are presented in the following tables.

Model evaluation results using Hold-out Method

Table 12. Baseline model

Model	Accuracy	F1 Score	Precision	Recall(TPR)	Specificity(TNR)	FPR	FNR	AUC	MCC
Logistic Regression	0.8761	0.4786	0.7190	0.3587	0.9736	0.0264	0.6413	0.6661	0.4497
SVM	0.8889	0.5564	0.7582	0.4394	0.9736	0.0264	0.5606	0.7065	0.5222
KNN	0.8731	0.4902	0.6750	0.3848	0.9651	0.0349	0.6152	0.6749	0.4457
Naive Bayes	0.7669	0.4687	0.3669	0.6485	0.7892	0.2108	0.3515	0.7188	0.3559
Decission Tree	0.8750	0.6019	0.6077	0.5962	0.9275	0.0725	0.4038	0.7618	0.5278
Random Forest	0.8956	0.6741	0.7638	0.6295	0.9651	0.0331	0.3705	0.7982	0.6527
XGBOOST	0.8901	0.6667	0.7451	0.6033	0.9453	0.0349	0.3967	0.7842	0.6277

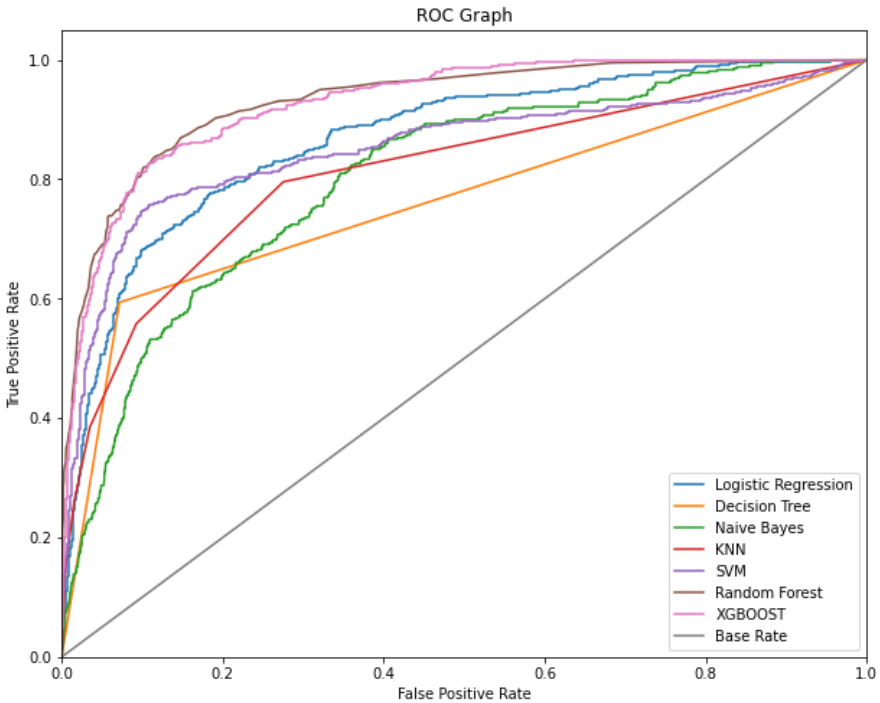


Figure 21

Table 13. Model results whit Kbest feature selection

Model	Accuracy	F1Score	Precision	Recall(TPR)	Specificity(TNR)	FPR	FNR	AUC	MCC
Logistic Regression_KB	0.8674	0.6263	0.5662	0.7007	0.8988	0.1012	0.2993	0.7998	0.5514
SVM_KB	0.8546	0.6148	0.5301	0.7316	0.8778	0.1222	0.2684	0.8047	0.5384
KNN_KB	0.8019	0.5362	0.4264	0.7221	0.8169	0.1831	0.2779	0.7695	0.4442
Naive Bayes_KB	0.7247	0.4491	0.3289	0.7078	0.7278	0.2722	0.2922	0.7178	0.3357
Decision Tree_KB	0.8621	0.6013	0.5553	0.6556	0.9011	0.0989	0.3444	0.7783	0.5213
Random Forest_KB	0.8938	0.7032	0.6314	0.7933	0.9127	0.0873	0.2067	0.8530	0.6457
XGBOOST_KB	0.8896	0.6651	0.6410	0.6912	0.9270	0.0730	0.3088	0.8091	0.5998

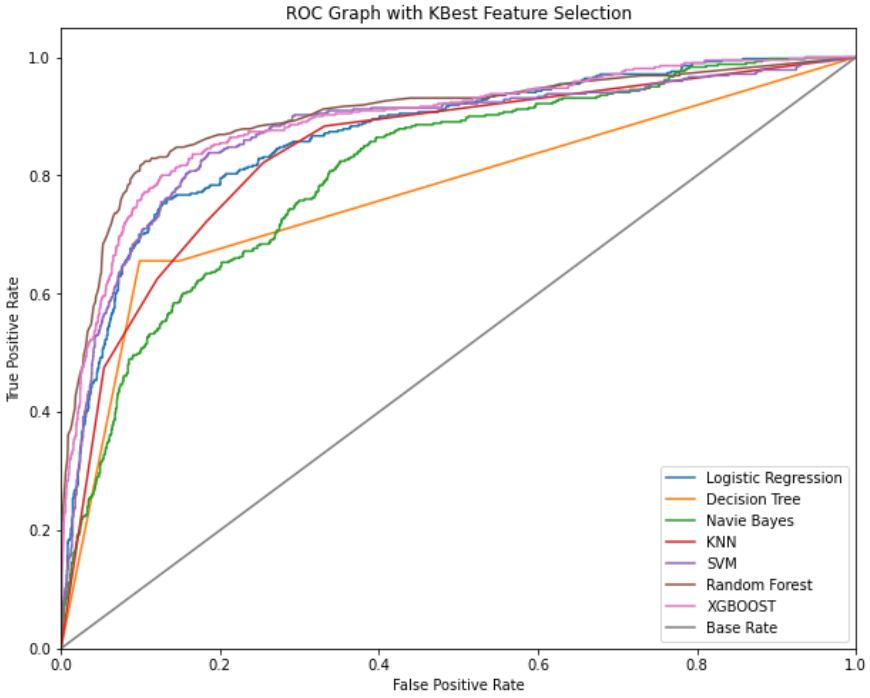


Figure 22

Table 14. Model results whit RFE feature selection

Model	Accuracy	F1Score	Precision	Recall(TPR)	Specificity(TNR)	FPR	FNR	AUC	MCC
Logistic Regression_RFE	0.8682	0.6212	0.5706	0.6817	0.9033	0.0967	0.3183	0.7925	0.5453
SVM_RFE	0.8685	0.6307	0.5687	0.7078	0.8988	0.1012	0.2922	0.8033	0.5568
KNN_RFE	0.8128	0.5510	0.4446	0.7245	0.8295	0.1705	0.2755	0.7770	0.4622
Naive Bayes_RFE	0.5043	0.3661	0.2296	0.9026	0.4293	0.5707	0.0974	0.6659	0.2502
Decission Tree_RFE	0.8674	0.5982	0.5758	0.6223	0.9136	0.0864	0.3777	0.7680	0.5195
Random Forest_RFE	0.9087	0.7218	0.6994	0.7458	0.9702	0.0698	0.2447	0.8780	0.6539
XGBOOST_RFE	0.8949	0.6706	0.6667	0.6746	0.9364	0.0636	0.3254	0.8055	0.6081

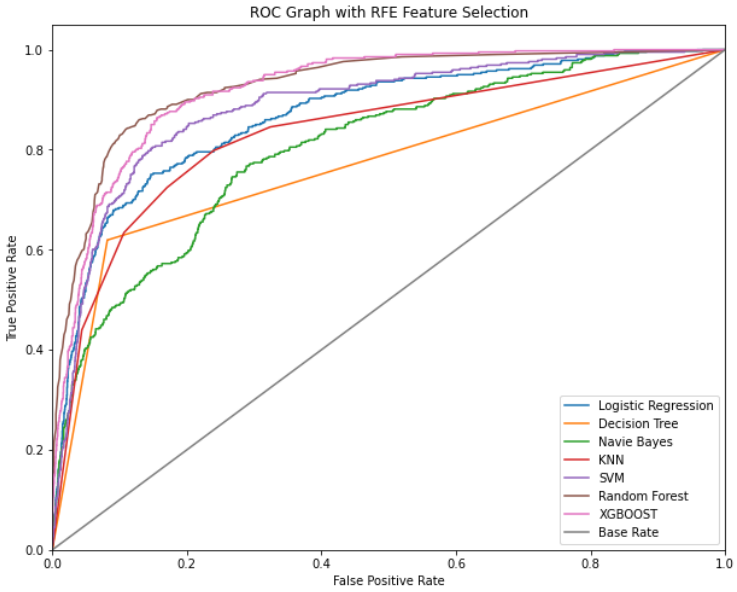


Figure 23

Table 15. Model results whit VIF feature selection

Model	Accuracy	F1 Score	Precision	Recall(TPR)	Specificity(TNR)	FPR	FNR	AUC	MCC
Logistic Regression_VIF	0.8692	0.6409	0.5842	0.7099	0.9005	0.0995	0.2901	0.8052	0.5659
SVM_VIF	0.8631	0.6384	0.5646	0.7344	0.8885	0.1115	0.2656	0.8114	0.5630
KNN_VIF	0.8064	0.5467	0.4446	0.7099	0.8254	0.1746	0.2901	0.7676	0.4509
Navie Bayes_VIF	0.7047	0.4659	0.3316	0.7832	0.6892	0.3108	0.2168	0.7362	0.3593
Decission Tree_VIF	0.8523	0.5806	0.5448	0.6214	0.8978	0.1022	0.3786	0.7596	0.4930
Random Forest_VIF	0.8920	0.6968	0.6474	0.7542	0.9191	0.0809	0.2458	0.8367	0.6343
XGBOOST_VIF	0.8945	0.6809	0.6778	0.6840	0.9360	0.0640	0.3160	0.8100	0.6177

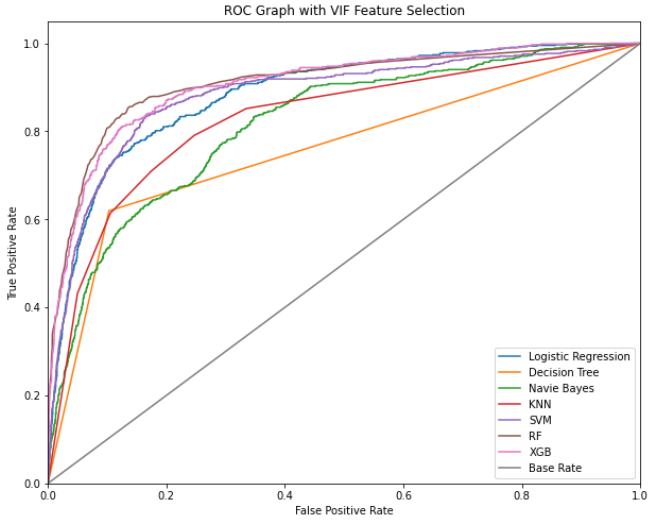


Figure 24

Table 16. Comparing all models using Hold-out Method

Model	Accuracy	F1 Score	Precision	Recall(TPR)	Specificity(TNR)	FPR	FNR	AUC	MCC
Logistic Regression	0.8761	0.4786	0.7190	0.3587	0.9736	0.0264	0.6413	0.6661	0.4497
SVM	0.8889	0.5564	0.7582	0.4394	0.9736	0.0264	0.5606	0.7065	0.5222
KNN	0.8731	0.4902	0.6750	0.3848	0.9651	0.0349	0.6152	0.6749	0.4457
Naive Bayes	0.7669	0.4687	0.3669	0.6485	0.7892	0.2108	0.3515	0.7188	0.3559
Decission Tree	0.8750	0.6019	0.6077	0.5962	0.9275	0.0725	0.4038	0.7618	0.5278
Random Forest	0.8956	0.6741	0.7638	0.6295	0.9651	0.0331	0.3705	0.7982	0.6527
XGBOOST	0.8901	0.6667	0.7451	0.6033	0.9453	0.0349	0.3967	0.7842	0.6277
Logistic Regression_KB	0.8674	0.6263	0.5662	0.7007	0.8988	0.1012	0.2993	0.7998	0.5514
SVM_KB	0.8546	0.6148	0.5301	0.7316	0.8778	0.1222	0.2684	0.8047	0.5384
KNN_KB	0.8019	0.5362	0.4264	0.7221	0.8169	0.1831	0.2779	0.7695	0.4442
Naive Bayes_KB	0.7247	0.4491	0.3289	0.7078	0.7278	0.2722	0.2922	0.7178	0.3357
Decision Tree_KB	0.8621	0.6013	0.5553	0.6556	0.9011	0.0989	0.3444	0.7783	0.5213
Random Forest_KB	0.8938	0.7032	0.6314	0.7933	0.9127	0.0873	0.2067	0.8530	0.6457
XGBOOST_KB	0.8896	0.6651	0.6410	0.6912	0.9270	0.0730	0.3088	0.8091	0.5998
Logistic Regression_RFE	0.8682	0.6212	0.5706	0.6817	0.9033	0.0967	0.3183	0.7925	0.5453
SVM_RFE	0.8685	0.6307	0.5687	0.7078	0.8988	0.1012	0.2922	0.8033	0.5568
KNN_RFE	0.8128	0.5510	0.4446	0.7245	0.8295	0.1705	0.2755	0.7770	0.4622
Naive Bayes_RFE	0.5043	0.3661	0.2296	0.9026	0.4293	0.5707	0.0974	0.6659	0.2502
Decission Tree_RFE	0.8674	0.5982	0.5758	0.6223	0.9136	0.0864	0.3777	0.7680	0.5195
Random Forest_RFE	0.9087	0.7218	0.6994	0.7458	0.9702	0.0698	0.2447	0.8780	0.6539
XGBOOST_RFE	0.8949	0.6706	0.6667	0.6746	0.9364	0.0636	0.3254	0.8055	0.6081
Logistic Regression_VIF	0.8692	0.6409	0.5842	0.7099	0.9005	0.0995	0.2901	0.8052	0.5659
SVM_VIF	0.8631	0.6384	0.5646	0.7344	0.8885	0.1115	0.2656	0.8114	0.5630
KNN_VIF	0.8064	0.5467	0.4446	0.7099	0.8254	0.1746	0.2901	0.7676	0.4509
Navie Bayes_VIF	0.7047	0.4659	0.3316	0.7832	0.6892	0.3108	0.2168	0.7362	0.3593
Decission Tree_VIF	0.8523	0.5806	0.5448	0.6214	0.8978	0.1022	0.3786	0.7596	0.4930
Random Forest_VIF	0.8920	0.6968	0.6474	0.7542	0.9191	0.0809	0.2458	0.8367	0.6343
XGBOOST_VIF	0.8945	0.6809	0.6778	0.6840	0.9360	0.0640	0.3160	0.8100	0.6177

As can be seen, Random Forest with RFE as a feature selection has the highest F1 Score, AUC, accuracy, and Specificity. Moreover, it has an excellent MCC score regarding other models. Thus, this model is selected as the best one in this setting.

Model evaluation results using Hold-out Method

Table 17. Baseline model

	Accuracy	F1	Recall	Precision	AUC
LogisticRegression	0.8556	0.6224	0.7510	0.5324	0.8862
SVC	0.8861	0.6464	0.6628	0.6312	0.8697
KNeighborsClassifier	0.8464	0.5711	0.6436	0.5163	0.8441
GaussianNB	0.7966	0.5116	0.6768	0.4071	0.8228
DecisionTreeClassifier	0.8561	0.5913	0.6664	0.5342	0.7801
RandomForestClassifier	0.8941	0.6931	0.7554	0.6452	0.9193
XGBClassifier	0.8931	0.6697	0.6894	0.6514	0.9094

Table 18. Model results whit Kbest feature selection

	Accuracy	F1	Recall	Precision	AUC
LogisticRegression	0.8529	0.6182	0.7613	0.5221	0.8871
SVC	0.8881	0.6481	0.6518	0.6474	0.8828
KNeighborsClassifier	0.8393	0.5417	0.5893	0.4965	0.8168
GaussianNB	0.7112	0.4688	0.8036	0.3321	0.8228
DecisionTreeClassifier	0.8681	0.6154	0.6831	0.5542	0.7908
RandomForestClassifier	0.9054	0.7150	0.7421	0.6851	0.9373
XGBClassifier	0.8984	0.6832	0.7007	0.6743	0.9300

Table 19. Model results whit RFE feature selection

	Accuracy	F1	Recall	Precision	AUC
LogisticRegression	0.8519	0.6199	0.7601	0.5208	0.8872
SVC	0.8881	0.6476	0.6505	0.6453	0.8838
KNeighborsClassifier	0.8395	0.5381	0.5878	0.4931	0.8171
GaussianNB	0.7141	0.4689	0.8043	0.3322	0.8247
DecisionTreeClassifier	0.8649	0.6145	0.6838	0.5606	0.7932
RandomForestClassifier	0.9061	0.7125	0.7491	0.6814	0.9381
XGBClassifier	0.9001	0.6864	0.6928	0.6779	0.9297

Table 20. Model results whit VIF feature selection

	Accuracy	F1	Recall	Precision	AUC
LogisticRegression	0.8399	0.5961	0.7421	0.4978	0.8762
SVC	0.8863	0.6452	0.6526	0.6391	0.8862
KNeighborsClassifier	0.8431	0.5522	0.6053	0.5019	0.8277
GaussianNB	0.6446	0.4317	0.8543	0.2902	0.8281
DecisionTreeClassifier	0.8593	0.6019	0.6652	0.5495	0.7845
RandomForestClassifier	0.8991	0.6925	0.7292	0.6654	0.9309
XGBClassifier	0.8914	0.6673	0.6753	0.6508	0.9244

Final results of Model evaluation using Kfold cross-validation

Subsequent to comparing the scores, one can infer that the Random Forest Classifier and XGBOOST Classifier trained more properly than other models. Thus, these models, alongside with baseline model, were collected and operated on the test data. The results are given in the following table.

Table 21. Comparing best models using Kfold cross-validation

Model	Accuracy	F1Score	Precision	Recall(TPR)	Specificity(TNR)	FPR	FNR	AUC	MCC
RandomForest	0.8879	0.6921	0.6129	0.7947	0.9054	0.0946	0.2053	0.9019	0.6328
RF_CVRFE	0.9108	0.7324	0.7567	0.7096	0.9610	0.0390	0.2555	0.9325	0.6471
XGBOOST	0.8879	0.6921	0.6129	0.7947	0.9054	0.0946	0.2053	0.9190	0.6328
XGBOOST_CVRFE	0.8993	0.6647	0.7043	0.6293	0.9502	0.0498	0.3707	0.9315	0.6070

The random forest model with a cross-validated RFE has the highest F1 Score, MCC, Accuracy, AUC, Specificity, the lowest FPR, and a good FNR score. Consequently, this model was compared to the Hold-out RFE Random forest, which is shown in the following table. It can be concluded from the result that the Random forest model with a cross-validated RFE is the best model for predictive online visitors' purchasing intention due to its higher F1 score, AUC, and accuracy. However, when time is of great importance for the online retailer, the other RFE Random Forest can also perform for the prediction, for it has excellent and very close scores to the best model.

Table 22. Final result

Model	Accuracy	F1Score	Precision	Recall(TPR)	Specificity(TNR)	FPR	FNR	AUC	MCC
RF_RFE	0.9087	0.7218	0.6994	0.7458	0.9702	0.0698	0.2447	0.8780	0.6539
RF_CVRFE	0.9108	0.7324	0.7567	0.7096	0.9610	0.0390	0.2555	0.9325	0.6471

Conclusion:

In this project, the author aims to create a predictive model in order to foretell the purchasing intention of an online retailer's visitors by employing machine learning algorithms. The Clickstream and Web analytics data of the online retailer was utilized for training the models and subsequently testing them. First, EDA was conducted to understand the dataset properly and acquire valuable insights. Based on the findings, the retailer should redesign its marketing strategy and provide incentives to encourage visitors to overcome their hesitation and generate revenue.

In the next step, Preprocessing techniques were used to modify the categorical value into numerical, remove outliers and junk data, and oversample data to treat the imbalanced distribution of the target variable. Three features selection, namely, Kbest, RFE, and VIF, were applied to determine the most beneficial subsets of training data. To evaluate the models, Hold-out and Kfold cross-validation methods were implemented. In the end, the performance indicators were compared to find the most robust model.

The results show that the Random forest model with a cross-validated RFE has the highest prediction performance among all other models. It indicates that RFE feature selection enhances the models' performance, and Kfold cross-validation made the models more reliable. The proposed predictive model enables the retailer to specifically carry out its novel marketing strategies to the visitors who are predicted as non-buyers.

The findings also confirm the idea that clickstream data during the visit of an online store comprises important information of online shoppers' behavior. Moreover, the outcomes reveal that choosing a minimal subset of data results in a more accurate and scalable model and lessens computational costs. Therefore, considering the significance of the time used by the predictive model in the real world, the e-commerce company should scrutinize the topic and collect more reliable data to find better predictive models.

References

Cho, C., Kang, J. & Cheon, H., 2006. Online Shopping Hesitation. *CyberPsychology & Behavior,* 9(3), pp. 261 - 274.

Kuhn, M. & Johnson, K., 2013. *Applied Predictive Modeling.* 1st ed. s.l.:Springer.

Moe, W., 2003. Buying, Searching, or Browsing: Differentiating Between Online Shoppers Using In-Store Navigational Clickstream. *Consumers in Cyberspace,* 13(1-2), pp. 29 - 39.

Rajamma, R. K., Paswan, A. K. & Hossain, M. M., 2009. Why do shoppers abandon shopping cart? Perceived waiting time, risk, and transaction inconvenience. *Journal of Product & Brand Management,* 18(3), pp. 188 - 197.

Sakar, C., Polat, S. & Katircioglu, K. Y., 2019. Real-time prediction of online shoppers' purchasing intention using multilayer perceptron and LSTM recurrent neural networks. *Neural Computing and Applications,* Volume 31, p. 6893–6908.

Zhou, L., Dai, L. & Zhang, D., 2007. Online Shopping Acceptance Model: A Critical Survey of Consumer Factors in Online Shopping. *Journal of Electronic Commerce Research,* 8(1), pp. 41-63.

Python Codes

Importing required libraries

```
import pandas as pd
import numpy as np
from scipy import stats
import matplotlib.pyplot as plt
import seaborn as sns
import imblearn
import math

from sklearn.model_selection import cross_val_score
from sklearn.model_selection import RepeatedStratifiedKFold
from sklearn.model_selection import StratifiedKFold
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
from sklearn.feature_selection import RFE
from sklearn.model_selection import train_test_split
from sklearn.model_selection import RandomizedSearchCV


from statsmodels.stats.outliers_influence import variance_inflation_factor

from imblearn.over_sampling import BorderlineSMOTE
from imblearn.over_sampling import SMOTE
from imblearn.under_sampling import RandomUnderSampler
from imblearn.pipeline import Pipeline

from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB
from sklearn.datasets import load_wine
from sklearn.metrics import confusion_matrix, accuracy_score, recall_score, f1_score, precision_score, roc_auc_score, matthews_corrcoef
from xgboost import XGBClassifier

data = pd.read_csv("theonlineRetailer.csv")
```

Data Type

```
data.info()
```

Summary Statistics

```
ds = data.describe()
ds.T
```

Data Distribution

Conversion Rate

```
plt.figure(figsize=(6,4))

plt.xlabel('Revenue')
plt.ylabel('Percentage')

ax = (data.Revenue.value_counts()/len(data)*100).sort_index().plot.bar(color='blue', edgecolor='black')
ax.set_yticks(np.arange(0, 110, 10))
```

```
for p in ax.patches:
   ax.annotate('{:.2f}%'.format(p.get_height()), (p.get_x()+0.15, p.get_height()+1))
```

Visitor Type:

```
plt.figure(figsize=(6,4))

plt.xlabel('VisitorType')
plt.ylabel('Percentage')

ax = (data.VisitorType.value_counts()/len(data)*100).sort_index().plot.bar(color='blue', edgecolor='black')
ax.set_yticks(np.arange(0, 110, 10))

for p in ax.patches:
   ax.annotate('{:.2f}%'.format(p.get_height()), (p.get_x()+0.15, p.get_height()+1))

#PIEPLOT

plt.figure(figsize=(6,6), dpi=100)

explode=[0,0,0.04]
sums1 = data.Revenue.groupby(data.VisitorType).sum()
plt.pie(sums1, labels=sums1.index, explode=explode)
plt.show()
```

Month:

```
plt.figure(figsize=(6,4))

plt.xlabel('Month')
plt.ylabel('Percentage')

ax = (data.Month.value_counts()/len(data)*100).sort_index().plot.bar(color='blue', edgecolor='black')
ax.set_yticks(np.arange(0, 110, 10))

for p in ax.patches:
   ax.annotate('{:.2f}%'.format(p.get_height()), (p.get_x()-0.2, p.get_height()+5))

######

#PIEPLOT
plt.figure(figsize=(5,5), dpi=100)


colors = ['#ABABAB', '#00CD00', '#FF8000', '#800000','#104E8B', '#00BFFF', '#CDAD00', '#7F7F7F','#EE6AA7', '#8B5742', '#EE3B3B', '#7D26CD']
explode=[0,0,0.02,0,0.02,0,0,0,0,0,0,0]
sums = data.Revenue.groupby(data.Month).sum()
plt.pie(sums, labels=sums.index, colors=colors, explode=explode)
plt.show()
```

Weekend:

```
plt.figure(figsize=(6,4))

plt.xlabel('Weekend')
plt.ylabel('Percentage')

ax = (data.Weekend.value_counts()/len(data)*100).sort_index().plot.bar(color='blue', edgecolor='black')
ax.set_yticks(np.arange(0, 110, 10))

for p in ax.patches:
   ax.annotate('{:.2f}%'.format(p.get_height()), (p.get_x()+0.15, p.get_height()+1))

#PIEPOLT
plt.figure(figsize=(5,5), dpi=100)


colors = ['#104E8B', '#FF8000']
mylabels = ["Weekday", "Weekend"]
sums1 = data.Revenue.groupby(data.Weekend).sum()

plt.pie(sums1, labels=mylabels, colors=colors)
plt.show()
```

Special Day:

```
plt.figure(figsize=(6,4))

plt.xlabel('SpecialDay')
plt.ylabel('Percentage')
```

```
ax = (data.SpecialDay.value_counts()/len(data)*100).sort_index().plot.bar(color='blue', edgecolor='black')
ax.set_yticks(np.arange(0, 110, 10))

for p in ax.patches:
   ax.annotate('{:.0001f}%'.format(p.get_height()), (p.get_x()+0.00001, p.get_height()+1))
```

Browser, Region, Operating Systems, and Traffic Type:

```
plt.figure(figsize=(6,4))

plt.xlabel('Browser')
plt.ylabel('Percentage')

ax = (data.Browser.value_counts()/len(data)*100).sort_index().plot.bar(color='blue', edgecolor='black')
ax.set_yticks(np.arange(0, 110, 10))

for p in ax.patches:
   ax.annotate('{:.0001f}%'.format(p.get_height()), (p.get_x()+0.00001, p.get_height()+1))
###########
plt.figure(figsize=(6,4))

plt.xlabel('Region')
plt.ylabel('Percentage')

ax = (data.Region.value_counts()/len(data)*100).sort_index().plot.bar(color='blue', edgecolor='black')
ax.set_yticks(np.arange(0, 110, 10))

for p in ax.patches:
   ax.annotate('{:.0001f}%'.format(p.get_height()), (p.get_x()+0.00001, p.get_height()+1))

############

plt.figure(figsize=(6,4))

plt.xlabel('OperatingSystems')
plt.ylabel('Percentage')

ax = (data.OperatingSystems.value_counts()/len(data)*100).sort_index().plot.bar(color='blue', edgecolor='black')
ax.set_yticks(np.arange(0, 110, 10))

for p in ax.patches:
   ax.annotate('{:.0001f}%'.format(p.get_height()), (p.get_x()+0.00001, p.get_height()+1))

#############

plt.figure(figsize=(6,4))

plt.xlabel('TrafficType')
plt.ylabel('Percentage')

ax = (data.TrafficType.value_counts()/len(data)*100).sort_index().plot.bar(color='blue', edgecolor='black')
ax.set_yticks(np.arange(0, 110, 10))

for p in ax.patches:
   ax.annotate('{:.0001f}%'.format(p.get_height()), (p.get_x()+0.00001, p.get_height()+1))
```

Density Plots:

Administrative page:

```
data['Administrative'].plot(kind='density', xlim=(0,32))
plt.xlabel('Administrative page visits')
plt.show()
```

Informational page:

```
informdf=data.loc[lambda data: data['Informational']>0]
informdf['Informational'].plot(kind='density', xlim=(0,22))
plt.xlabel('Informational page visits')
plt.show()
```

Product related page:

```
data['ProductRelated'].plot(kind='density', xlim=(0,834))
plt.xlabel('Product related page visits')
plt.show()
```

Bounce rate

```
data['BounceRates'].plot(kind='density', xlim=(0,0.851))
plt.xlabel('Bounce rate')
plt.show()
```

Exist rate:

```
data['ExitRates'].plot(kind='density', xlim=(0,0.85452))
plt.xlabel('Exit rate')
plt.show()
```

Customers' Purchasing Intentions Determinants:

Visitor Type and Revenue:

```
vr = pd.crosstab(data.VisitorType,data.Revenue)
vr.plot(kind='bar',color=['#DC143C','#1874CD'])
l= ['Not-generated','Generated']
plt.legend(title = 'Revenue' ,loc='best', labels = l)
```

Weekend on Revenue:

```
wr = pd.crosstab(data.Weekend,data.Revenue)
wr.plot(kind='bar',color=['#DC143C','#1874CD'])
l= ['Not-generated','Generated']
plt.legend(title = 'Revenue' ,loc='best', labels = l)
```

Month on Revenue:

```
mr = pd.crosstab(data.Month,data.Revenue)
mr.plot(kind='bar',color=['#DC143C','#1874CD'])
l= ['Not-generated','Generated']
plt.legend(title = 'Revenue' ,loc='best', labels = l)
```

Region on Revenue:

```
rr = pd.crosstab(data.Region,data.Revenue)
rr.plot(kind='bar',color=['#DC143C','#1874CD'])
l= ['Not-generated','Generated']
plt.legend(title = 'Revenue' ,loc='best', labels = l)
```

Traffic Type on Revenue:

```
tr = pd.crosstab(data.TrafficType,data.Revenue)
tr.plot(kind='bar',color=['#DC143C','#1874CD'])
l= ['Not-generated','Generated']
plt.legend(title = 'Revenue' ,loc='best', labels = l)
```

Operating systems on Revenue:

```
osr = pd.crosstab(data.OperatingSystems,data.Revenue)
osr.plot(kind='bar',color=['#DC143C','#1874CD'])
l= ['Not-generated','Generated']
plt.legend(title = 'Revenue' ,loc='best', labels = l)
```

Browser on Revenue:

```
br = pd.crosstab(data.Browser,data.Revenue)
br.plot(kind='bar',color=['#DC143C','#1874CD'])
l= ['Not-generated','Generated']
plt.legend(title = 'Revenue' ,loc='best', labels = l)
```

Pair Plot:

```
sns.pairplot(data, vars = ['Administrative', 'Administrative_Duration', 'Informational',
      'Informational_Duration', 'ProductRelated', 'ProductRelated_Duration',
```

```
'BounceRates', 'ExitRates', 'PageValues', 'SpecialDay'],hue = 'Revenue')
```

Data Preprocessing:

Null value:

```
data.isnull().sum()
```

Categorical Variable:

```
le = preprocessing.LabelEncoder()

le.fit(data['Month'])
list(le.classes_)
data['Month']=le.transform(data['Month'])

le.fit(data['VisitorType'])
list(le.classes_)
data['VisitorType']=le.transform(data['VisitorType'])

le.fit(data['Revenue'])
list(le.classes_)
data['Revenue']=le.transform(data['Revenue'])

le.fit(data['Browser'])
list(le.classes_)
data['Browser']=le.transform(data['Browser'])

le.fit(data['Region'])
list(le.classes_)
data['Region']=le.transform(data['Region'])

le.fit(data['Weekend'])
list(le.classes_)
data['Weekend']=le.transform(data['Weekend'])

le.fit(data['OperatingSystems'])
list(le.classes_)
data['OperatingSystems']=le.transform(data['OperatingSystems'])

le.fit(data['TrafficType'])
list(le.classes_)
data['TrafficType']=le.transform(data['TrafficType'])
```

Outlier Treatment:

BOX-COX

```
data_boxcox = data.copy()

data_boxcox['Informational_Duration'] = stats.boxcox(data_boxcox.Informational_Duration.values+1)[0]
data_boxcox['Administrative_Duration'] = stats.boxcox(data_boxcox.Administrative_Duration.values+1)[0]
data_boxcox['ProductRelated_Duration'] = stats.boxcox(data_boxcox.ProductRelated_Duration.values+1)[0]
data_boxcox['PageValues'] = stats.boxcox(data_boxcox.PageValues.values+1)[0]

data_boxcox.iloc[:,:-2].boxplot(figsize=(24,24));

fig, ax = plt.subplots(figsize=(15,15))
data_boxcox.iloc[:,:-2].hist(ax=ax)
plt.show()
```

IQR

```
num_cols=['Administrative', 'Administrative_Duration', 'Informational','Informational_Duration', 'ProductRelated', 'ProductRelated_Duration','BounceRates', 'ExitRates', 'PageValues',
'SpecialDay']

Q1=data[num_cols].quantile(0.25)
Q3=data[num_cols].quantile(0.75)
IQR=Q3-Q1
print(Q1)
print(Q3)
IQR
```

```
num_cols=['Administrative', 'Administrative_Duration', 'Informational','Informational_Duration', 'ProductRelated', 'ProductRelated_Duration','BounceRates', 'ExitRates', 'PageValues',
'SpecialDay']

lower = Q1 - 1.5* IQR
upper = Q3 + 1.5* IQR
print("lower limit",'\n',lower,'\n')
print("upper limit",'\n',upper,'\n')

outliers = ((data[num_cols]< (Q1 - 1.5 * IQR)) | (data[num_cols] > (Q3 + 1.5 * IQR))).sum()
data = data.drop(outliers)
```

Imbalance Treatment:

```
sm = SMOTE(random_state = 30)
X_train_new, y_train_new = sm.fit_sample(xtrain, ytrain.ravel())
X_train_new=pd.DataFrame(X_train_new)
X_train_new.columns=columns
y_train_new=pd.DataFrame(y_train_new)
y_train_new.columns=['Revenue']
```

Removing Junk Data

```
ju = (data.ProductRelated_Duration != 0.0) | (data.Administrative_Duration != 0.0) | (data.Informational_Duration != 0.0)
data = data[ju]
len(data.index)
```

Feature Selection:

Kbest

```
X=data.drop(columns=['Revenue'])
y=data['Revenue']
bestfeatures = SelectKBest(score_func=chi2, k=15)
bf = bestfeatures.fit(X,y)
dfscores = pd.DataFrame(bf.scores_)
dfcolumns = pd.DataFrame(X.columns)

featureScores = pd.concat([dfcolumns,dfscores],axis=1)
featureScores.columns = ['Specs','Score']
print(featureScores.nlargest(10,'Score'))

featureScores = pd.concat([dfcolumns,dfscores],axis=1)
featureScores.columns = ['Specs','Score']
print(featureScores.nlargest(10,'Score'))
```

RFE

```
model = LogisticRegression(solver='lbfgs')
rfe = RFE(model,10 )
fit = rfe.fit(X, y)
print("Num Features: %d" % fit.n_features_)
print("Selected Features: %s" % fit.support_)
print("Feature Ranking: %s" % fit.ranking_)

num_ls = np.arange(1,14)
num = 0
high_score = 0
score_list = []
for i in range(len(num_ls)):
    x_train,x_test,y_train,y_test = train_test_split(X,y,test_size=0.3,random_state=0)
    LR = LogisticRegression()
    rfe = RFE(LR,num_ls[i])
    x_train_rfe = rfe.fit_transform(x_train,y_train)
    x_test_rfe = rfe.transform(x_test)
    LR.fit(x_train_rfe,y_train)
    score = LR.score(x_test_rfe,y_test)
    print(rfe.support_)
    print(rfe.ranking_)
    print(score)
    if(score>high_score):
        high_score = score
        num = num_ls[i]
print("The highest r score is {}".format(high_score))
print("{} number of features give the best score of {}".format(num,high_score)).

cols = list(X.columns)
model = LinearRegression()
#Initializing RFE model
rfe = RFE(model, 12)
```

```
#Transforming data using RFE
X_rfe = rfe.fit_transform(X,y)
#Fitting the data to model
model.fit(X_rfe,y)
temp = pd.Series(rfe.support_,index = cols)
selected_features_rfe = temp[temp==True].index
print(selected_features_rfe)
```

VIF

```
[variance_inflation_factor(X.values, j) for j in range(1, X.shape[1])]

def calculate_vif(x):
    thresh = 5.0
    output = pd.DataFrame()
    k = x.shape[1]
    vif = [variance_inflation_factor(x.values, j) for j in range(x.shape[1])]
    for i in range(1,k):
        print("Iteration no.",i)
        print(vif)
        a = np.argmax(vif)
        print("Max VIF is for variable no.:",a)
        if vif[a] <= thresh :
            break
        if i == 1 :
            output = x.drop(x.columns[a], axis = 1)
            vif = [variance_inflation_factor(output.values, j) for j in range(output.shape[1])]
        elif i > 1 :
            output = output.drop(output.columns[a],axis = 1)
            vif = [variance_inflation_factor(output.values, j) for j in range(output.shape[1])]
    return(output)
```

Model Evaluation:

Hold-out

```
d1 = data.drop('Revenue',axis=1)
df_scaled = d1.apply(zscore)
x = df_scaled[cols]
y = data.Revenue

xtrain,xtest,ytrain,ytest = train_test_split(x,y,test_size = 0.20,random_state = 0)
(xtrain.shape,xtest.shape,ytrain.shape,ytest.shape)
```

Kfold cross-validation

```
over=BorderlineSMOTE(sampling_strategy=0.3, k_neighbors=6)
under=RandomUnderSampler(sampling_strategy=0.4)

def cv_comparison(models, x, y, cv=RepeatedStratifiedKFold(n_splits=10, n_repeats=3, random_state=24)):
    cv_accuracies = pd.DataFrame()


    for model in models:
        steps=[('o',over),('u',under),('model',model)]
        pipe=Pipeline(steps=steps)
        accuracies = cross_val_score(pipe, x, y, scoring='accuracy', cv=cv)
        accuracy_score = np.mean(accuracies).round(4)
        f1s = cross_val_score(pipe, x, y, scoring='f1', cv=cv)
        f1_score = np.mean(f1s).round(4)
        recalls = cross_val_score(pipe, x, y, scoring='recall', cv=cv)
        recall_score = np.mean(recalls).round(4)
        precisions = cross_val_score(pipe, x, y, scoring='precision', cv=cv)
        precision_score = np.mean(precisions).round(4)
        aucs = cross_val_score(pipe, x, y, scoring='roc_auc', cv=cv)
        auc_score = np.mean(aucs).round(4)


        cv_accuracies[str(model).split('(')[0]] = [ accuracy_score, f1_score, recall_score, precision_score, auc_score]
    cv_accuracies.index = ['Accuracy','F1', 'Recall', 'Precision','AUC']
    return cv_accuracies


lg = LogisticRegression(max_iter = 10000, class_weight='balanced', random_state=42)
rf = RandomForestClassifier(random_state=42)
xgb = XGBClassifier(random_state=42)
dt = DecisionTreeClassifier(random_state=42)
knn = KNeighborsClassifier()
```

```
svc = SVC()
nb = GaussianNB()

models = [lg, svc, knn, nb, dt, rf, xgb]


cv_comparison(models, xtrain, ytrain)
```

Model Selection:

Model evaluation results using Hold-out Method

```
Model = []
Accuracy= []
F1Score = []
Precision = []
Recall = []
Specificity = []
FPR = []
FNR = []
AUC = []
MCC = []


#pate the code for RFE Hold-out. The codes for baseline Kbest, and VIF are almost the same


lr = LogisticRegression()
lr.fit(X_train_new, y_train_new)
y_pred_rfe = lr.predict(xtest)

cm = confusion_matrix(ytest,y_pred_rfe)

TP = cm[1,1]
TN = cm[0,0]
FP = cm[0,1]
FN = cm[1,0]

Model.append('Logistic Regression_RFE')
Accuracy.append(accuracy_score(ytest,y_pred_rfe).round(4))
F1Score.append(f1_score(ytest,y_pred_rfe).round(4))
Precision.append(precision_score(ytest,y_pred_rfe).round(4))
AUC.append(roc_auc_score(ytest,y_pred_rfe).round(4))
Recall.append(recall_score(ytest,y_pred_rfe).round(4))
Specificity.append((TN/(TN+FP)).round(4))
FPR.append((FP / (FP + TN)).round(4))
FNR.append((FN / (FN + TP)).round(4))
MCC.append((((TP*TN)-(FP*FN))/(math.sqrt((TP+FP)*(TP+FN)*(TN+FP)*(TN+FN)))).round(4))


####


svc = SVC()
svc.fit(X_train_new, y_train_new)
y_pred_rfe = svc.predict(xtest)

cm = confusion_matrix(ytest,y_pred_rfe)

TP = cm[1,1]
TN = cm[0,0]
FP = cm[0,1]
FN = cm[1,0]

Model.append('SVM_RFE')
Accuracy.append(accuracy_score(ytest,y_pred_rfe).round(4))
F1Score.append(f1_score(ytest,y_pred_rfe).round(4))
Precision.append(precision_score(ytest,y_pred_rfe).round(4))
AUC.append(roc_auc_score(ytest,y_pred_rfe).round(4))
Recall.append(recall_score(ytest,y_pred_rfe).round(4))
Specificity.append((TN/(TN+FP)).round(4))
FPR.append((FP / (FP + TN)).round(4))
FNR.append((FN / (FN + TP)).round(4))
MCC.append((((TP*TN)-(FP*FN))/(math.sqrt((TP+FP)*(TP+FN)*(TN+FP)*(TN+FN)))).round(4))

#####

knn = KNeighborsClassifier()
knn.fit(X_train_new, y_train_new)
```

```
y_pred_rfe = knn.predict(xtest)

cm = confusion_matrix(ytest,y_pred_rfe)

TP = cm[1,1]
TN = cm[0,0]
FP = cm[0,1]
FN = cm[1,0]

Model.append('KNN_RFE')
Accuracy.append(accuracy_score(ytest,y_pred_rfe).round(4))
F1Score.append(f1_score(ytest,y_pred_rfe).round(4))
Precision.append(precision_score(ytest,y_pred_rfe).round(4))
AUC.append(roc_auc_score(ytest,y_pred_rfe).round(4))
Recall.append(recall_score(ytest,y_pred_rfe).round(4))
Specificity.append((TN/(TN+FP)).round(4))
FPR.append((FP / (FP + TN)).round(4))
FNR.append((FN / (FN + TP)).round(4))
MCC.append((((TP*TN)-(FP*FN))/(math.sqrt((TP+FP)*(TP+FN)*(TN+FP)*(TN+FN)))).round(4))

######

nb = GaussianNB()
nb.fit(X_train_new, y_train_new)
y_pred_rfe = nb.predict(xtest)

cm = confusion_matrix(ytest,y_pred_rfe)

TP = cm[1,1]
TN = cm[0,0]
FP = cm[0,1]
FN = cm[1,0]

Model.append('Naive Bayes_RFE')
Accuracy.append(accuracy_score(ytest,y_pred_rfe).round(4))
F1Score.append(f1_score(ytest,y_pred_rfe).round(4))
Precision.append(precision_score(ytest,y_pred_rfe).round(4))
AUC.append(roc_auc_score(ytest,y_pred_rfe).round(4))
Recall.append(recall_score(ytest,y_pred_rfe).round(4))
Specificity.append((TN/(TN+FP)).round(4))
FPR.append((FP / (FP + TN)).round(4))
FNR.append((FN / (FN + TP)).round(4))
MCC.append((((TP*TN)-(FP*FN))/(math.sqrt((TP+FP)*(TP+FN)*(TN+FP)*(TN+FN)))).round(4))

######

dt = DecisionTreeClassifier()
dt.fit(X_train_new, y_train_new)
y_pred_rfe = dt.predict(xtest)

cm = confusion_matrix(ytest,y_pred_rfe)

TP = cm[1,1]
TN = cm[0,0]
FP = cm[0,1]
FN = cm[1,0]

Model.append('Decission Tree_RFE')
Accuracy.append(accuracy_score(ytest,y_pred_rfe).round(4))
F1Score.append(f1_score(ytest,y_pred_rfe).round(4))
Precision.append(precision_score(ytest,y_pred_rfe).round(4))
AUC.append(roc_auc_score(ytest,y_pred_rfe).round(4))
Recall.append(recall_score(ytest,y_pred_rfe).round(4))
Specificity.append((TN/(TN+FP)).round(4))
FPR.append((FP / (FP + TN)).round(4))
FNR.append((FN / (FN + TP)).round(4))
MCC.append((((TP*TN)-(FP*FN))/(math.sqrt((TP+FP)*(TP+FN)*(TN+FP)*(TN+FN)))).round(4))

######


rfc = RandomForestClassifier()
rfc.fit(X_train_new, y_train_new)
y_pred_rfe = rfc.predict(xtest)

cm = confusion_matrix(ytest,y_pred_rfe)



TP = cm[1,1]
TN = cm[0,0]
FP = cm[0,1]
```

```
FN = cm[1,0]

Model.append('Random Forest_RFE')
Accuracy.append(accuracy_score(ytest,y_pred_rfe).round(4))
F1Score.append(f1_score(ytest,y_pred_rfe).round(4))
Precision.append(precision_score(ytest,y_pred_rfe).round(4))
AUC.append(roc_auc_score(ytest,y_pred_rfe).round(4))
Recall.append(recall_score(ytest,y_pred_rfe).round(4))
Specificity.append((TN/(TN+FP)).round(4))
FPR.append((FP / (FP + TN)).round(4))
FNR.append((FN / (FN + TP)).round(4))
MCC.append((((TP*TN)-(FP*FN))/(math.sqrt((TP+FP)*(TP+FN)*(TN+FP)*(TN+FN)))).round(4))

#####


xgb = XGBClassifier()
xgb.fit(X_train_new, y_train_new)
y_pred_rfe = xgb.predict(xtest)

cm = confusion_matrix(ytest,y_pred_rfe)

TP = cm[1,1]
TN = cm[0,0]
FP = cm[0,1]
FN = cm[1,0]

Model.append('XGBOOST_RFE')
Accuracy.append(accuracy_score(ytest,y_pred_rfe).round(4))
F1Score.append(f1_score(ytest,y_pred_rfe).round(4))
Precision.append(precision_score(ytest,y_pred_rfe).round(4))
AUC.append(roc_auc_score(ytest,y_pred_rfe).round(4))
Recall.append(recall_score(ytest,y_pred_rfe).round(4))
Specificity.append((TN/(TN+FP)).round(4))
FPR.append((FP / (FP + TN)).round(4))
FNR.append((FN / (FN + TP)).round(4))
MCC.append((((TP*TN)-(FP*FN))/(math.sqrt((TP+FP)*(TP+FN)*(TN+FP)*(TN+FN)))).round(4))


######

result_RFE = pd.DataFrame({'Model':Model,'Accuracy':Accuracy,'F1Score':F1Score,'Precision':Precision, 'Recall(TPR)':Recall, 'Specificity(TNR)':Specificity,'FPR':FPR,'FNR':FNR,
'AUC':AUC,'MCC':MCC})
result_RFE

#####

fpr, tpr, thresholds = roc_curve(ytest, lr.predict_proba(xtest)[:,1])
dt_fpr, dt_tpr, dt_thresholds = roc_curve(ytest, dt.predict_proba(xtest)[:,1])
nb_fpr, nb_tpr, nb_thresholds = roc_curve(ytest, nb.predict_proba(xtest)[:,1])

svc = SVC(probability=True) # Platt calibration
svc.fit(X_train_new, y_train_new)


svc_fpr, svc_tpr, svc_thresholds = roc_curve(ytest, svc.predict_proba(xtest)[:,1])

rfc_fpr, rfc_tpr, rfc_thresholds = roc_curve(ytest, rfc.predict_proba(xtest)[:,1])
knn_fpr, knn_tpr, knn_thresholds = roc_curve(ytest, knn.predict_proba(xtest)[:,1])
xgb_fpr, xgb_tpr, xgb_thresholds = roc_curve(ytest, xgb.predict_proba(xtest)[:,1])

plt.figure(figsize=(10,8))
plt.plot(fpr,tpr, label='Logistic Regression')
plt.plot(dt_fpr, dt_tpr, label='Decision Tree')
plt.plot(nb_fpr, nb_tpr, label='Navie Bayes')
plt.plot(knn_fpr, knn_tpr, label='KNN')
plt.plot(svc_fpr, svc_tpr, label='SVM')
plt.plot(rfc_fpr, rfc_tpr, label='Random Forest')
plt.plot(xgb_fpr, xgb_tpr, label='XGBOOST')
plt.plot([0,1], [0,1],label='Base Rate')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Graph with RFE Feature Selection')
plt.legend(loc="lower right")
plt.show()
```

Model evaluation results using Hold-out Method

```
rf = RandomForestClassifier(random_state=42, n_jobs=-1, class_weight='balanced')
over=BorderlineSMOTE(sampling_strategy=0.3, k_neighbors=6)
under=RandomUnderSampler(sampling_strategy=0.4)

n_estimators = [int(x) for x in np.linspace(start = 200, stop = 1000, num = 10)]
max_features = ['auto', 'sqrt']
max_depth = [int(x) for x in np.linspace(10, 110, num = 11)]
max_depth.append(None)
min_samples_split = [2, 5, 10]
min_samples_leaf = [1, 2, 4]
bootstrap = [True, False]

random_grid = {'n_estimators': n_estimators,
           'max_features': max_features,
           'max_depth': max_depth,
           'min_samples_split': min_samples_split,
           'min_samples_leaf': min_samples_leaf,
           'bootstrap': bootstrap}



cv=RepeatedStratifiedKFold(n_splits=7, n_repeats=2, random_state=42)

grid_search=RandomizedSearchCV(estimator=rf
                      ,param_distributions=random_grid
                      ,n_jobs=-1
                      ,scoring='f1'
                      ,cv=cv
                      ,n_iter = 50
                     )

pipe=Pipeline([('o',over),('u',under), ('model',grid_search)])


pipe.fit(xtrain,ytrain)



result_rfecv_rf = pd.DataFrame({'Model':Model,'Accuracy':Accuracy,'F1Score':F1Score,'Precision':Precision, 'Recall(TPR)':Recall, 'Specificity(TNR)':Specificity,'FPR':FPR,'FNR':FNR,
'AUC':AUC,'MCC':MCC})
result_rfecv_rf

#######

xgb = XGBClassifier(random_state=42)
random_grid = {'n_estimators': range(50,150,10),
           'algorithm': ['SAMME','SAMME.R']
          }

grid_search=GridSearchCV(estimator=xgb
                      ,param_grid=random_grid
                      ,n_jobs=-1
                      ,scoring='f1'
                      ,cv=cv
                     )

pipe_xgb=Pipeline([('o',over),('u',under), ('model',grid_search)])


pipe_xgb.fit(xtrain,ytrain)



result_rfecv_xgb = pd.DataFrame({'Model':Model,'Accuracy':Accuracy,'F1Score':F1Score,'Precision':Precision, 'Recall(TPR)':Recall, 'Specificity(TNR)':Specificity,'FPR':FPR,'FNR':FNR,
'AUC':AUC,'MCC':MCC})
result_rfecv_xgb
```