

**Project Report**  
**Machine Learning [CSE2023]**

**E-COMMERCE PURCHASE PREDICTION**

*By*

*Ridit Jain*  
*210471*

*Ashi Jain*  
*210451*



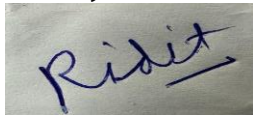
**Department of Computer Science and Engineering**  
**School of Engineering and Technology**  
**BML Munjal University**

**April 2023**

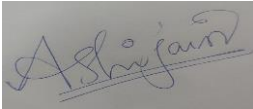
## Declaration by the Candidates

We hereby declare that the project entitled "***E-COMMERCE PURCHASE PREDICTION***" has been carried out for fulfilling the partial requirements for completion of the course on **Machine Learning** offered at the 4<sup>th</sup> Semester of the Bachelor of Technology (B. Tech) program in the Department of Computer Science and Engineering during AY-2022-23 (even semester). This experimental work has been carried out by us and submitted to the course instructor *Dr. Soharab Hossain Shaikh*. Due acknowledgments have been made in the text of the project to all other materials used. This project has been done in full compliance with the requirements and constraints of the prescribed curriculum.

Ridit Jain



Ashi Jain



**Place:** BML Munjal University

**Date:** 23 April 2023

## **ABSTRACT**

With today's shift from visiting physical stores to shopping online, approximately consumer behaviours in e-commerce have become increasingly important. By making it easier to do more personal work, it increases customer satisfaction and sales, leading to more conversions and better competitive advantage. By leveraging clickstream and adding customer data, models can be built to predict customer behaviour. This project analyses machine learning models to predict purchases using data from major clothing stores. Next, to compare the models, this study takes the data described by and trains the model on different data to gain more insight into performance about the structure of the extension line and static customer data. The results show that the Random Forest algorithm is the best for prediction tasks, showing the best performance, reasonable latency, providing insight and robustness. In terms of disparate data, the model trained on the speech data array outperformed the models trained on customer data so far. The best results are obtained when combining two files.

## **ACKNOWLEDGEMENT**

We are highly grateful to Dr. Soharab Hossain Shaikh BML Munjal University, Gurugram, for providing supervision to carry out the project from March-April 2023.

Dr. Soharab Hossain Shaikh has been of tremendous assistance in carrying out this study and are gratefully thanked. It'd have been difficult to finish the course in this way without the sensible advice and capable direction.

We wish to thank Dr. Soharab Hossain Shaikh for his encouragement from time to time. We would also like to thank the whole BML Munjal University team. We would also want to thank my friends for devoting their important time and assisting me in whatever way they could to ensure my progress.

Ashi Jain

Ridit Jain

# Contents

	Page No.
1: Introduction	1
2: Problem Statement	2
3: Literature Review	3-6
3.1: Research paper review	
3.2: Different models	
3.2.1: Logistic Regression	
3.2.2: K Nearest Neighbour	
3.2.3: Naïve Bayes	
3.2.4: Decision Tree	
3.2.5: Random Forest	
4: Description of the Dataset	7-9
5: Methodology	10-12
6: Experimental Results	13-17
7: Conclusions	18
8: References	19
9: Appendix:	20-38

## 1: Introduction

According to today's business world information and business media, e-commerce has become popular all over the world. The most common type of business is the business-to-consumer business, often in the form of online stores, that replaces physical stores.

The rapid growth of e-business has changed the entire purchasing process and with it the traditional one. buyer relationship This change comes with challenges businesses need to solve. These challenges require more competition and unstable relationships between consumers and merchants because consumers and their interests are not known individually, resulting in lower customers. For this reason, attracting customers, gaining customer trust, and retaining customers have become the main goals of e-commerce today.

Online shoppers are more extroverted than ever before. Collect, connect, and store large volumes of personal data and clickstream data recorded on each visit to the online store for analysis using data mining. The insights gained from these analyses can increase customer satisfaction and make the purchasing process more efficient, engaging, and personalized. The analysis of buyer behaviour goes back to the fundamentals of e-commerce and has many applications today. For example, recommending products to customers, converting customers into buyers, visitors, etc. to better serve customers who are more likely to buy.

The project is like predicting whether a particular customer will make a purchase based on their browsing behaviour on an e-commerce website. For example, protection can be provided by displaying product recommendations to encourage purchase. Customer behaviour predictions are often based on customer data, clickstream data, and additional data from other sources. A lot of machine learning is involved in classifying clickstream data. Examples are general learning models such as logistic regression (LR), support vector machine (SVM), decision trees (DT), random forests (RF).

Due to the importance of categorizing consumer behaviour and the abundance of predictive models and data, this project aims to use and compare appropriate models learned on the same data to determine the most appropriate model for website visitors to purchase. Hence the binary classification of visitors belonging to the buy or not buy category.

## 2: Problem Statement

In this project, a machine learning model was identified and used to solve the task of classifying website visitors as buying or not buying. This model works on different data, which are clickstream data generated by each visitor to the online store, and customer data that can identify the visitor. This was done to determine which model and data are best for predicting the probability of purchase for an online business based on performance, latency, and insight. Latency is important because, in use, estimates must be made between the time in use and the time the comprehension model is calculated, as the need to explain the process determined by the machine learning model increases. The support tree model currently used by clothing stores will provide a basis for comparing different algorithms. An exploratory data analysis of the clickstream and static customer data is performed to offer more explanation and reasons for the differing performances on different datasets. After getting the results of different models of each dataset, test the robustness of the model for various conditions. This review shows how the model will perform in the real world after deployment.

## 3: Literature Review

### 3.1: Research paper review

E-commerce platforms have become a significant source of revenue for businesses around the world. The ability to predict customer purchasing behaviour has become a critical area of research in recent years. The use of machine learning algorithms to analyse large data sets generated by e-commerce platforms has led to significant improvements in the accuracy of purchase predictions. This literature review examines existing research on e-commerce purchase prediction using machine learning.

#### **"Predicting Customer Purchase in E-Commerce Using Machine Learning Techniques" by S. K. Thakur et al.**

This paper proposes a machine learning approach to predict customer purchases in e-commerce. The authors used a dataset containing customer browsing and purchasing behaviour to train the model. The study evaluated the performance of several machine learning algorithms, including decision trees, random forests, and support vector machines. The results showed that the random forest algorithm outperformed the other models in terms of prediction accuracy.

#### **"E-commerce Product Recommendation Using Machine Learning Techniques" by S. S. Kulkarni et al."**

This paper proposes a recommendation system for e-commerce platforms using machine learning algorithms. The authors used a dataset containing customer purchase history and product attributes to train the model. The study evaluated the performance of several machine learning algorithms, including collaborative filtering and content-based filtering. The results showed that the collaborative filtering algorithm outperformed the other models in terms of recommendation accuracy.



## 3.2: Different Models

For a broader understanding of the problem and its various algorithms, we describe the algorithms used to solve it, and consider its applications and performance in the literature.

### 3.2.1 Logistic Regression

LR, also known as logistic regression, belongs to the general linear model and is used to predict the target variable. This is done by the Logic function, which is in the form of an S-handle and uses values between 0 and 1. This model by linearly combining the input values with the coefficients. where  $y$  is the output,  $b_0$  is the bias term and  $b_1$  is the coefficient of the input value  $x$ .

$$y = e^{b_0 + b_1 x} / 1 + e^{b_0 + b_1 x}$$

Each column of the input vector learns a coefficient by training the data from the maximum value. LR is very fast in terms of prediction and training time, so it is one of the most popular machine learning algorithms for binary classification. However, it requires engineering and coding of categorical variables. It is also sensitive to noise, so outliers should be removed before training. Additionally, LR underperforms on high-value strategies. Therefore, regression with the main points (linearly uncorrelated variables) will help. This can be done through principal component analysis (PCA), a method to extract key components from related processes.

### 3.2.2 K-nearest Neighbour

KNN is a lazy learning algorithm where the training data is.

is simply saved and a test data point is waited for classification.

All stored training instances correspond to points in an  $n$ -dimensional element.

space. The nearest neighbours of a point are defined by distance measurements, mostly, usually Euclidean distance. An unlabelled test data point will be assigned the label most common among its nearest neighbours. Advantages of this method

are its robustness to noisy data and very high training speed. There are disadvantages.

increased complexity of dimensionality through irrelevant features and therefore a reduced performance, highlighting the importance of feature engineering, longer.

prediction times compared to eager learning models and low intelligibility.

with high-dimensional input.

The KNN model is regularly applied in e-commerce regarding recommendation systems, where products are recommended to the visitor of the online store based on preferences of their nearest neighbours.

### **3.2.3 Naive Bayes**

The machine learning method Naive Bayes is straightforward but effective for classification applications. It is founded on the Bayes theorem, a cornerstone of probability theory. Since the algorithm assumes that each feature in the data set is independent of every other feature, which is frequently false, it is known as a "naive" algorithm. Naive Bayes can achieve excellent accuracy in many applications despite this oversimplifying assumption, and it is frequently employed in many different industries.

Calculating the probability of each class for a specific collection of features is how the Naive Bayes algorithm operates. It determines the conditional probability of each feature given each class after estimating the prior probability of each class based on the frequency of occurrence in the training data.

The posterior probability of each class given the observed features is then calculated using Bayes' theorem utilising these probabilities.

Naive Bayes has several benefits, including simplicity, scalability, and the capacity to effectively handle high-dimensional data. It can handle both numerical and categorical data and is also comparatively indifferent to insignificant aspects. The assumption of feature independence and the vulnerability to overfitting when the training data is scarce are some of its drawbacks.

### **3.2.4 Decision Trees**

DTs consist of a set of split conditions which divide a heterogeneous population into smaller, more homogeneous subgroups regarding a certain variable. The aim is to create the most homogeneous subgroups. Simple DTs have the advantage of being convertible to simple, understandable classification rules.

This comprehensibility decreases as the models grow larger and more unbalanced.

. In general, DTs offer a relatively fast learning and prediction speed. Disadvantages are the required feature engineering, the inability to implicitly model time sequences and an increased complexity regarding trees with categorical variables consisting of many categories.

### 3.2.5 Random Forest

The bag is another example of a tree with many large trees that fit the bootstrap resampled version of the data and are classified by majority vote. RF improved Bagging by breaking the association of trees. After each tree split, an attribute instance is selected and only those are considered for the next tree split. The result is again by voting a majority of a tree. Improvement of DT deficiency weaknesses such as robustness and overfitting with using multiple classifiers, bagging, and RF. They train faster than the support tree but spend more time making predictions.

DT has shown good results throughout the literature when applied to problems like this study. For example, use different DT algorithms to classify segments into buying and non-buying segments. In addition to the clickstream data, the configuration includes additional data about the inventory statistics used, which improves performance. Using only the clickstream data, which is closest to our reference data, Bagging RepTree shows the best results.

## 4: Description of the Dataset

The dataset that we have used in our project is freely available UCI Machine Learning Repository

<b>Data Set Characteristics:</b>	Multivariate	<b>Number of Instances:</b>	12330	<b>Area:</b>	Business
<b>Attribute Characteristics:</b>	Integer, Real	<b>Number of Attributes:</b>	18	<b>Date Donated</b>	2018-08-31
<b>Associated Tasks:</b>	Classification, Clustering	<b>Missing Values?</b>	N/A	<b>Number of Web Hits:</b>	223864

### Source:

1. C. Okan Sakar

Department of Computer Engineering, Faculty of Engineering and Natural Sciences, Bahcesehir University, 34349 Besiktas, Istanbul, Turkey

2. Yomi Kastro

Inveon Information Technologies Consultancy and Trade, 34335 Istanbul, Turkey

### Data Set Information:

The dataset consists of feature vectors belonging to 12,330 sessions. The dataset was formed so that each session would belong to a different user in a 1-year period to avoid any tendency to a specific campaign, special day, user profile, or period.

## Attribute Information:

The dataset consists of 10 numerical and 8 categorical attributes.

- **Informational:** This feature indicates the number of pages of the website that are related to providing information about the products or services offered by the website.
- **Informational Duration:** This feature indicates the time spent by the user on informational pages of the website.
- **Product Related:** This feature indicates the number of pages of the website that are related to the specific products offered by the website.
- **Product Related Duration:** This feature indicates the time spent by the user on product-related pages of the website.
- **Bounce Rate:** This feature indicates the percentage of users who leave the website after viewing only one page.
- **Exit Rate:** This feature indicates the percentage of users who leave the website after viewing multiple pages.
- **Page Value:** This feature indicates the average value of a page that a user visited before completing a transaction on the website.
- **Special Day:** This feature indicates the proximity of the visit to a special day, such as Mother's Day or Valentine's Day.
- **Month:** This feature indicates the month in which the visit occurred.
- **Operating System:** This feature indicates the operating system used by the user to browse the website.
- **Browser:** This feature indicates the browser used by the user to browse the website.
- **Region:** This feature indicates the geographical region from which the user accessed the website.

- **Traffic Type:** This feature indicates the type of traffic used by the user to access the website, such as direct, referral, or search engine.
- **Visitor Type:** This feature indicates whether the user is a returning or new visitor.
- **Weekend:** This feature indicates whether the visit occurred on a weekend or not.
- **Revenue:** This is the target variable and indicates whether a transaction was completed or not.

## 5: Methodology

Every machine learning project has some basic steps and procedures that we need to follow in a specific order to arrive at the desired result or output. In this project, we have applied the same procedure and have divided the complete project into six steps. The detailed description of these steps is as follows: -

### 1. Exploratory Data Analysis (EDA): -

Data scientists use exploratory data analysis (EDA), which frequently makes use of data visualization techniques, to examine and study data sets and summarize their key properties. It makes it simpler for data scientists to find patterns, identify anomalies, test hypotheses, or verify assumptions by determining how to modify data sources to achieve the answers they need.

EDA's major goal is to encourage data analysis before making any assumptions. It can assist in finding glaring errors, better understanding data patterns, spotting outliers or unusual occurrences, and discovering intriguing relationships between the variables.

### 2. Feature Selection and Feature Engineering: -

With the use of just pertinent data and the elimination of irrelevant data, feature selection is a technique for lowering the input variable for your model. It involves automatically selecting features for your machine learning model that are pertinent to the problem you are attempting to solve. We accomplish this by adding or removing significant features without altering them. It assists in minimizing the amount of noise in our data and the quantity of our input data.

The act of choosing, modifying, and converting unprocessed data into features that can be applied in supervised learning is known as feature engineering. It may be necessary to create and train improved features for machine learning to perform well on new jobs.

### 3. Model building: -

In machine learning, generalizing, and learning from training data results in the creation of a mathematical representation. The resulting machine learning model is then used to analyze new data and derive predictions.

Based on the target variable, also known as the Y variable, the model you create may be either a regression model or a classification model. Create a regression model if the target variable has a numerical value. You should create a classification model if the target variable's data type is qualitative.

There are two main types of machine learning models: -

- Supervised model

Machine learning models look for the best dependencies and correlations between the input data and the targets in supervised learning. For supervised learning, you must supply a labelled input training dataset. From that data set, the machine learning algorithms find patterns. The process then creates a model that can be applied to test data, which is brand-new data.

- Unsupervised model

Unsupervised learning creates clusters using input data. So, the machine learning model can figure out patterns, rules, or summaries of similar data points.

### 4. Model selection: -

In this section, we will compare the performances of various machine learning models using various factors like accuracy, F1 score, error calculation, calculating the R square values, etc. This step is very important as after this step we can conclude that out of all the machine learning models that we have made in the previous step which model gives the best performance on our dataset. After selecting that one model we can move on to the model deployment step of the project.



## **5. Hyperparameter Tuning: -**

A single training task is used to execute several trials for hyperparameter tweaking. Each trial represents the full execution of your training application, with the values of the selected hyperparameters set within the boundaries you define.

The AI Platform Training service and your training application need to explicitly communicate to do hyperparameter tuning. The information that your model requires is all defined in your training application. You must specify the hyperparameters (variables) you wish to modify as well as the target value for each.

## **6. Model deployment: -**

The process of putting a fully operational machine learning model into use, where it can make predictions based on data, is known as model deployment." These predictions are then used by users, developers, and systems to generate useful business decisions. Although models can be used in many ways, they are typically coupled with apps using APIs to provide universal access.

For data scientists, this step is typically the most challenging because it requires time and resources. Before a model is ready to be deployed into a production setting, it often goes through a great deal of change. Additionally, if models don't achieve the specified goals, they may never even reach the deployment stage.

## 6: Experimental Results

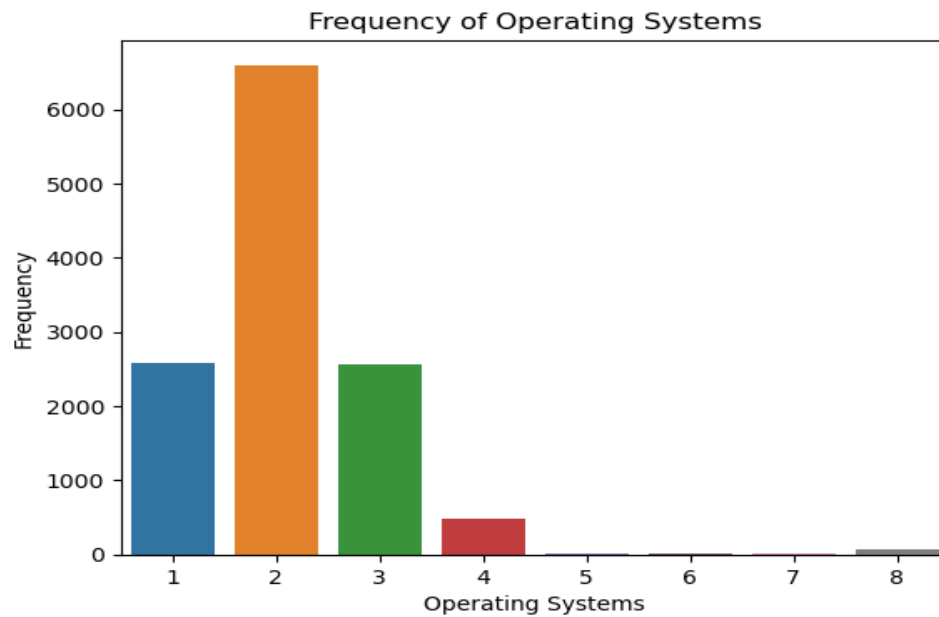
The aim of the project was to predict whether a customer will make a purchase or not.

1: import and load the dataset

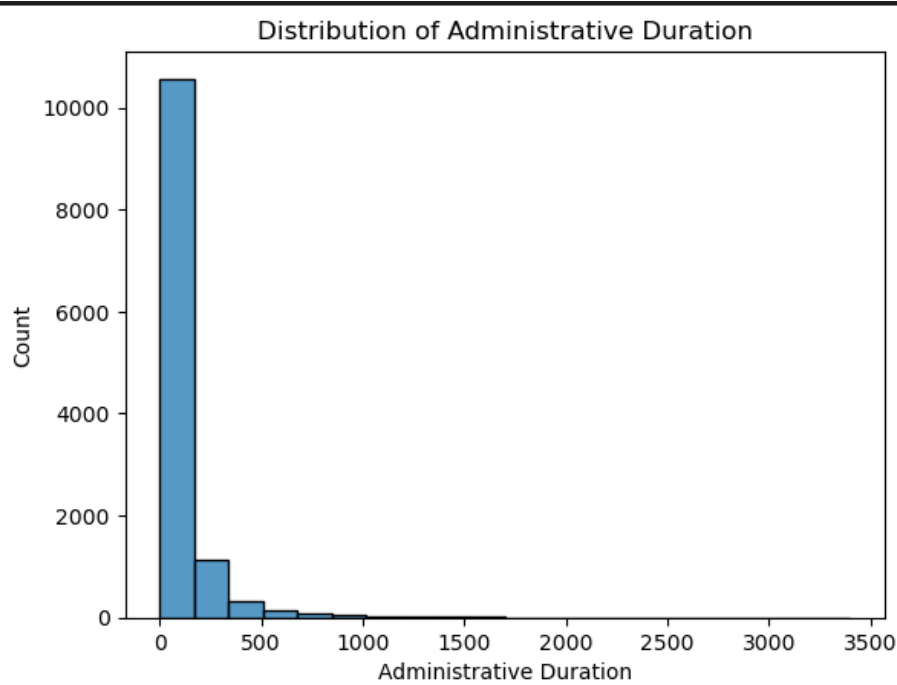
2: Exploratory data analysis

3: doing data visualization

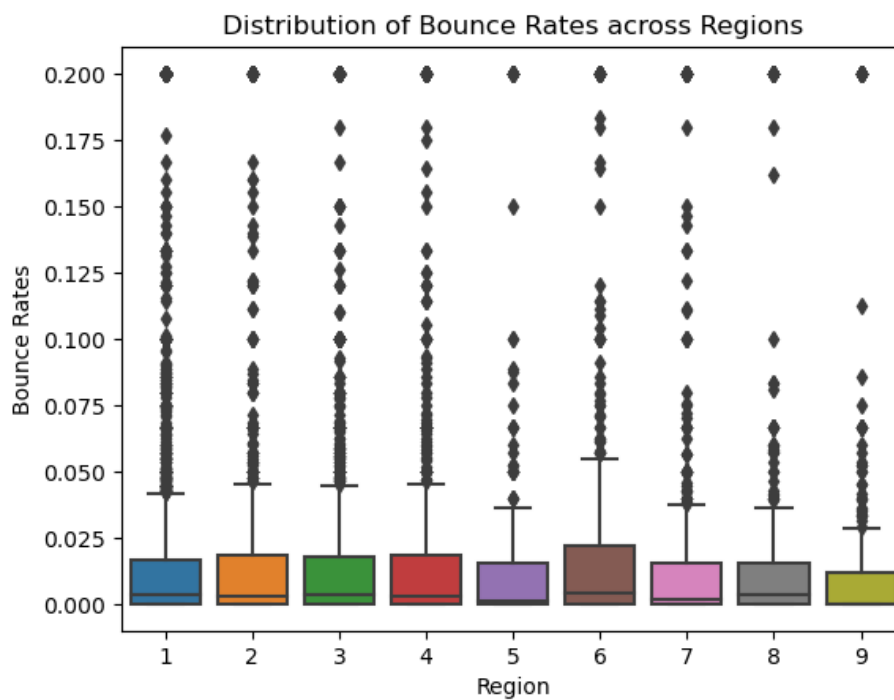
1: A bar plot showing the frequency of each operating system used by the website visitor



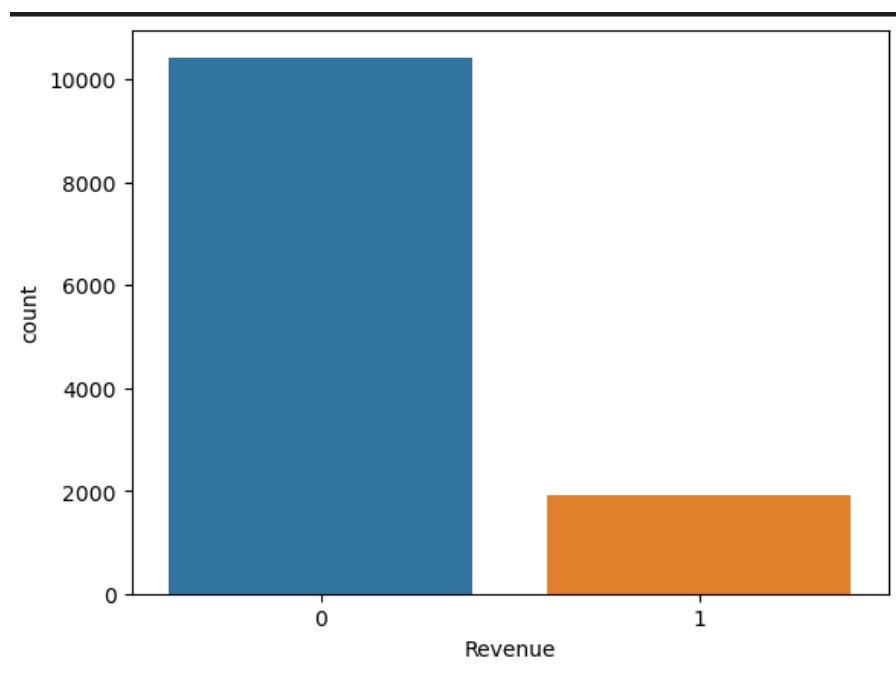
2: a histogram showing the distribution of the time spent on the website by the visitor



3: a box plot showing the distribution of the bounce rates across different regions

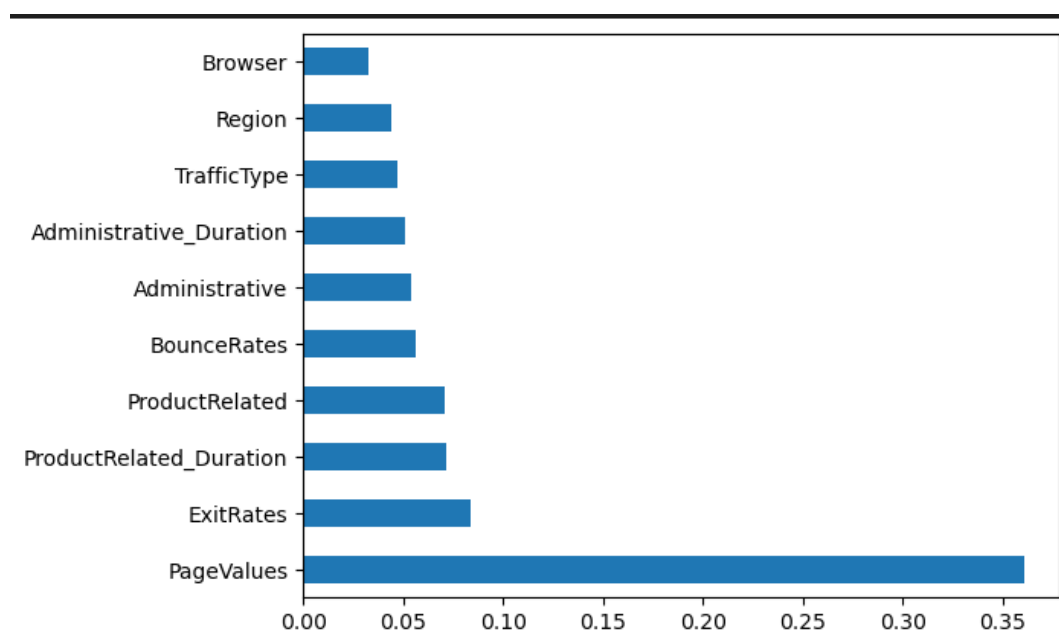


#### 4: Countplot of target variable



#### 4: Principal Component Analysis

##### Plotting feature importance graph



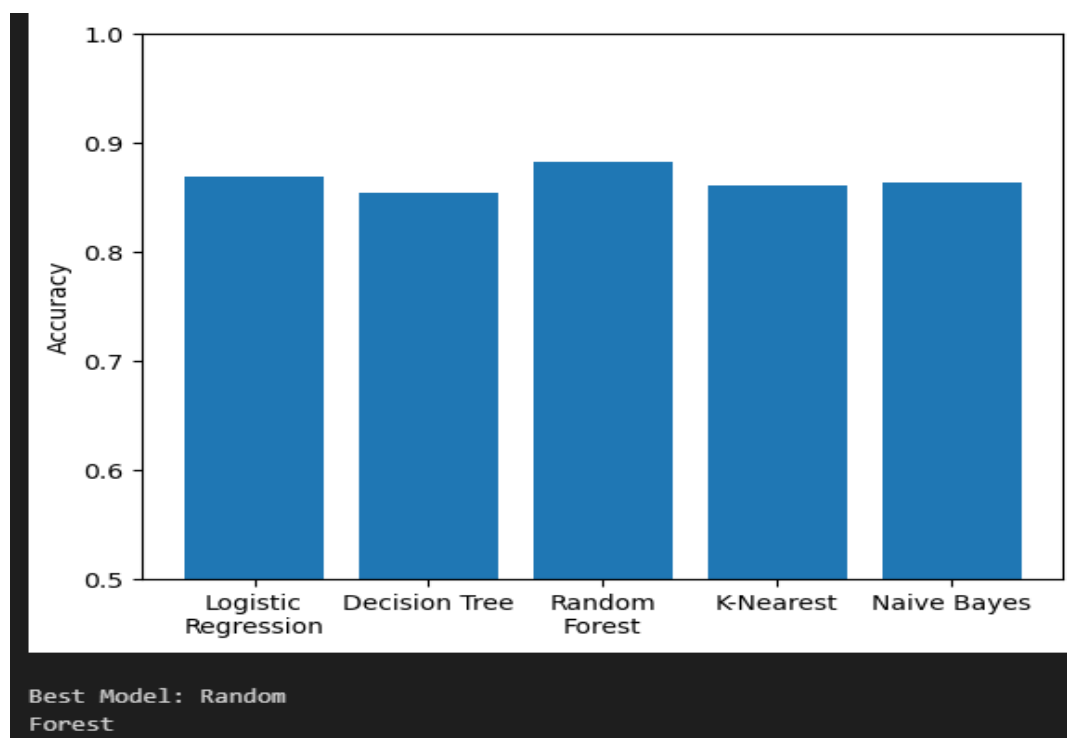
Here we see that column page value has the height importance and is greater as

compared to others there is a large difference after the 4th feature hence selecting top four features and we'll apply feature scaling to normalize the data point.

5: Building different models and predicting there accuracy

```
Logistic Regression Accuracy: 0.8694241686942417
Decision Tree Accuracy: 0.8499594484995945
Random Forest Accuracy: 0.8913219789132197
K-Nearest Neighbors Accuracy: 0.8584752635847527
Naive Bayes Classifier Accuracy: 0.7879156528791565
```

6: Plotting graph for model selection



## 7: Hyper Parameter Tunning

### 1: Finding the best parameter

```
Best Parameters: {'max_depth': 5, 'min_samples_leaf': 4, 'min_samples_split': 5, 'n_estimators': 200}
```

### 2: Classification report

Tuned Random Forest:

Accuracy: 0.8909164639091647

Confusion Matrix: [[1983 72]

[ 197 214]]

Classification Report:

				precision	recall	f1-score	support
	0	0.91	0.96	0.94	2055		
	1	0.75	0.52	0.61	411		
	accuracy			0.89	2466		
	macro avg	0.83	0.74	0.78	2466		
	weighted avg	0.88	0.89	0.88	2466		

## 7: Conclusions

The aim of this project was to develop a model to identify check whether a customer will make purchase or not on an e-commerce platform. Predicting the purchase is a complex process. Many factors affect the prediction and the outcome. By training on different datasets this project was able to emphasize that session-based data, generated from the customer clickstream, is most important for predicting purchase probabilities. This indicates that personal customer information, often associated with privacy concerns and regulations, is not necessarily needed to predict customer behaviour well. The project starts the prediction by exploratory data analysis in which it checked for the null values and there were no null values. After that we checked for the categorical values, we found two columns 'revenue' and 'weekend', so we converted them in unique values. Then it was found that 'other' in visitor type does not make sense, so replaced 'other' to 'returning visitor'. Then dropping the non-numeric column such as 'month'. After that plotting the graph of feature importance for better visualization then printing the feature of top four feature we found 'page value' has the height importance and is greater as compared to others, there is a huge difference after the 4th feature hence selecting top four feature and will apply feature scaling to normalize the data points. Further in model building we used models like logistic regression, Knn, Naïve Bayes, Decision Tress, Random Forest. The accuracy of all the models were: Logistic Regression Accuracy: 0.8694241686942417, Decision Tree Accuracy: 0.8499594484995945, Random Forest Accuracy: 0.8913219789132197, K-Nearest Neighbors Accuracy: 0.8584752635847527, Naive Bayes Classifier Accuracy: 0.7879156528791565. Then plotting the graph, we found that Random Forest was the best model for prediction. In Hyperparameter Tunning the best parameters are max\_depth, min\_samples\_leaf, min\_samples\_split, n\_estimators and at last whenever we will make a prediction of whether a customer will make a purchase or not. The model makes a prediction with 89% accuracy which is reliable.

## References

1. Jia, R., Li, R., Yu, M. and Wang, S., 2017, July. E-commerce purchase prediction approach by user behaviour data. In *2017 international conference on computer, information and telecommunication systems (CITS)* (pp. 1-5). IEEE.
2. Cirqueira, Douglas, et al. "Customer purchase behavior prediction in e-commerce: A conceptual framework and research agenda." *New Frontiers in Mining Complex Patterns: 8th International Workshop, NFMCP 2019, Held in Conjunction with ECML-PKDD 2019, Würzburg, Germany, September 16, 2019, Revised Selected Papers*. Cham: Springer International Publishing, 2020.
3. Kahng, A. B. (2022). Machine learning for CAD/EDA: The road ahead. *IEEE Design & Test*.
4. LaValley, Michael P. "Logistic regression." *Circulation* 117.18 (2008): 2395-2399.
5. Wang, Tiesheng, Irene YH Gu, and Pengfei Shi. "Object tracking using incremental 2D-PCA learning and ML estimation." In *2007 IEEE International Conference on Acoustics, Speech and Signal Processing-ICASSP'07*, vol. 1, pp. I-933. IEEE, 2007.
6. Webb GI, Keogh E, Miikkulainen R. Naïve Bayes. *Encyclopedia of machine learning*. 2010;15:713-4.
7. Mantovani, Rafael G., et al. "Hyper-parameter tuning of a decision tree induction algorithm." *2016 5th Brazilian Conference on Intelligent Systems (BRACIS)*. IEEE, 2016.
8. Hsu, P.F., Ray, S. and Li-Hsieh, Y.Y., 2014. Examining cloud computing adoption intention, pricing mechanism, and deployment model. *International Journal of Information Management*, 34(4), pp.474-488.
9. Quinlan, J.R., 1996. Learning decision tree classifiers. *ACM Computing Surveys (CSUR)*, 28(1), pp.71-72.



## Appendix

Following is the link for Streamlit app

<https://ridit07-ml-project-app-f3zc6r.streamlit.app/>

Github code link

<https://github.com/Ridit07/ml-project.git>

Following is the code of ipnb Machine learning file

### Importing The Dataset

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')

# URL of the data set
url = "https://archive.ics.uci.edu/ml/machine-learning-
databases/00468/online_shoppers_intention.csv"

# Read the CSV file into a pandas DataFrame
df = pd.read_csv(url)

# Print the first 5 rows of the DataFrame
df.head()
```

	Administrative	Administrative_Duration	Informational	\
0	0	0.0	0	
1	0	0.0	0	
2	0	0.0	0	
3	0	0.0	0	
4	0	0.0	0	

	Informational_Duration	ProductRelated	ProductRelated_Duration	\
0	0.0	1	0.000000	
1	0.0	2	64.000000	
2	0.0	1	0.000000	
3	0.0	2	2.666667	
4	0.0	10	627.500000	

	BounceRates	ExitRates	PageValues	SpecialDay	Month	OperatingSystems	\
0	0.20	0.20	0.0	0.0	Feb	1	
1	0.00	0.10	0.0	0.0	Feb	2	
2	0.20	0.20	0.0	0.0	Feb	4	
3	0.05	0.14	0.0	0.0	Feb	3	
4	0.02	0.05	0.0	0.0	Feb	3	

	Browser	Region	TrafficType	VisitorType	Weekend	Revenue
0	1	1	1	Returning_Visitor	False	False
1	2	1	2	Returning_Visitor	False	False
2	1	9	3	Returning_Visitor	False	False
3	2	2	4	Returning_Visitor	False	False
4	3	1	4	Returning_Visitor	True	False

## Basic information on the Dataset

df.describe()

	Administrative	Administrative_Duration	Informational	\
count	12330.000000	12330.000000	12330.000000	
mean	2.315166	80.818611	0.503569	
std	3.321784	176.779107	1.270156	
min	0.000000	0.000000	0.000000	
25%	0.000000	0.000000	0.000000	
50%	1.000000	7.500000	0.000000	
75%	4.000000	93.256250	0.000000	
max	27.000000	3398.750000	24.000000	

	Informational_Duration	ProductRelated	ProductRelated_Duration	\
count	12330.000000	12330.000000	12330.000000	
mean	34.472398	31.731468	1194.746220	
std	140.749294	44.475503	1913.669288	
min	0.000000	0.000000	0.000000	
25%	0.000000	7.000000	184.137500	
50%	0.000000	18.000000	598.936905	
75%	0.000000	38.000000	1464.157214	
max	2549.375000	705.000000	63973.522230	

	BounceRates	ExitRates	PageValues	SpecialDay \
count	12330.000000	12330.000000	12330.000000	12330.000000
mean	0.022191	0.043073	5.889258	0.061427
std	0.048488	0.048597	18.568437	0.198917
min	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.014286	0.000000	0.000000
50%	0.003112	0.025156	0.000000	0.000000
75%	0.016813	0.050000	0.000000	0.000000
max	0.200000	0.200000	361.763742	1.000000

	OperatingSystems	Browser	Region	TrafficType
count	12330.000000	12330.000000	12330.000000	12330.000000
mean	2.124006	2.357097	3.147364	4.069586
std	0.911325	1.717277	2.401591	4.025169
min	1.000000	1.000000	1.000000	1.000000
25%	2.000000	2.000000	1.000000	2.000000
50%	2.000000	2.000000	3.000000	2.000000
75%	3.000000	2.000000	4.000000	4.000000
max	8.000000	13.000000	9.000000	20.000000

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
```

```
RangeIndex: 12330 entries, 0 to 12329
```

```
Data columns (total 18 columns):
```

#	Column	Non-Null Count	Dtype
0	Administrative	12330 non-null	int64
1	Administrative_Duration	12330 non-null	float64
2	Informational	12330 non-null	int64
3	Informational_Duration	12330 non-null	float64
4	ProductRelated	12330 non-null	int64
5	ProductRelated_Duration	12330 non-null	float64
6	BounceRates	12330 non-null	float64
7	ExitRates	12330 non-null	float64
8	PageValues	12330 non-null	float64
9	SpecialDay	12330 non-null	float64
10	Month	12330 non-null	object
11	OperatingSystems	12330 non-null	int64
12	Browser	12330 non-null	int64
13	Region	12330 non-null	int64
14	TrafficType	12330 non-null	int64
15	VisitorType	12330 non-null	object
16	Weekend	12330 non-null	bool
17	Revenue	12330 non-null	bool

```
dtypes: bool(2), float64(7), int64(7), object(2)
```

```
memory usage: 1.5+ MB
```

# EDA (Exploratory Data Analysis)

## Check for Null Values

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

```
Administrative      0
Administrative_Duration  0
Informational      0
Informational_Duration  0
ProductRelated     0
ProductRelated_Duration  0
BounceRates        0
ExitRates          0
PageValues         0
SpecialDay         0
Month              0
OperatingSystems   0
Browser            0
Region             0
TrafficType        0
VisitorType        0
Weekend            0
Revenue            0
dtype: int64
```

## Categorical value conversion

```
df["Revenue"] = df["Revenue"].astype(int)

df["Weekend"] = df["Weekend"].astype(int)

from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
label = le.fit_transform(df['VisitorType'])
label

array([2, 2, 2, ..., 2, 2, 0])

cnt = 0
visited = []
for i in range(0, len(df['VisitorType'])):

    if df['VisitorType'][i] not in visited:

        visited.append(df['VisitorType'][i])

        cnt += 1

print("No.of.unique values :",
      cnt)
```

```

print("unique values :",
      visited)

No.of.unique values : 3
unique values : ['Returning_Visitor', 'New_Visitor', 'Other']

# 'Other' in Visitor type doesn't make sense, so replacing 'others' to
'Returning_Visitor'
df['VisitorType'] = df['VisitorType'].replace(['Other'],'Returning_Visitor')

le = LabelEncoder()
label = le.fit_transform(df['VisitorType'])
label

array([1, 1, 1, ..., 1, 1, 0])

df.drop("VisitorType", axis=1, inplace=True)

# Appending the array to our dataframe
df["VisitorType"] = label

```

## Dropping non-numeric columns

```
df.drop("Month", axis=1, inplace=True)
```

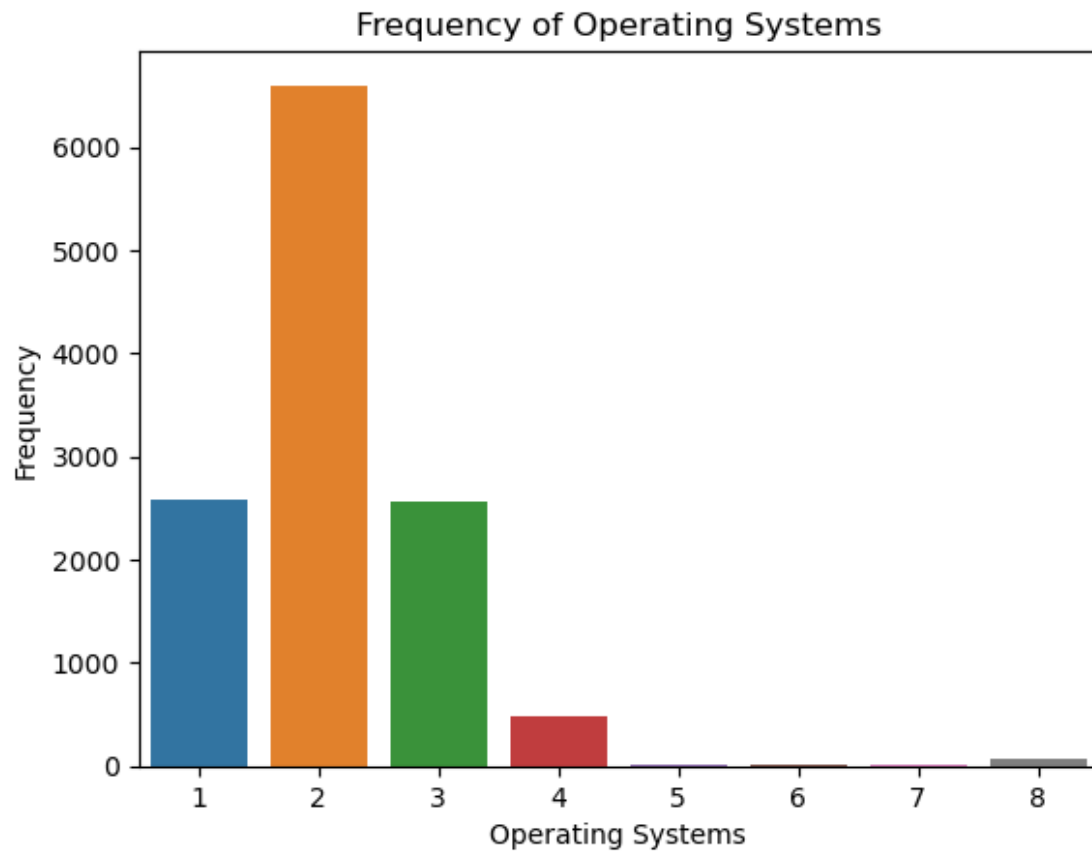
## Data Visualization

**A bar plot showing the frequency of each operating system used by the website visitors**

```

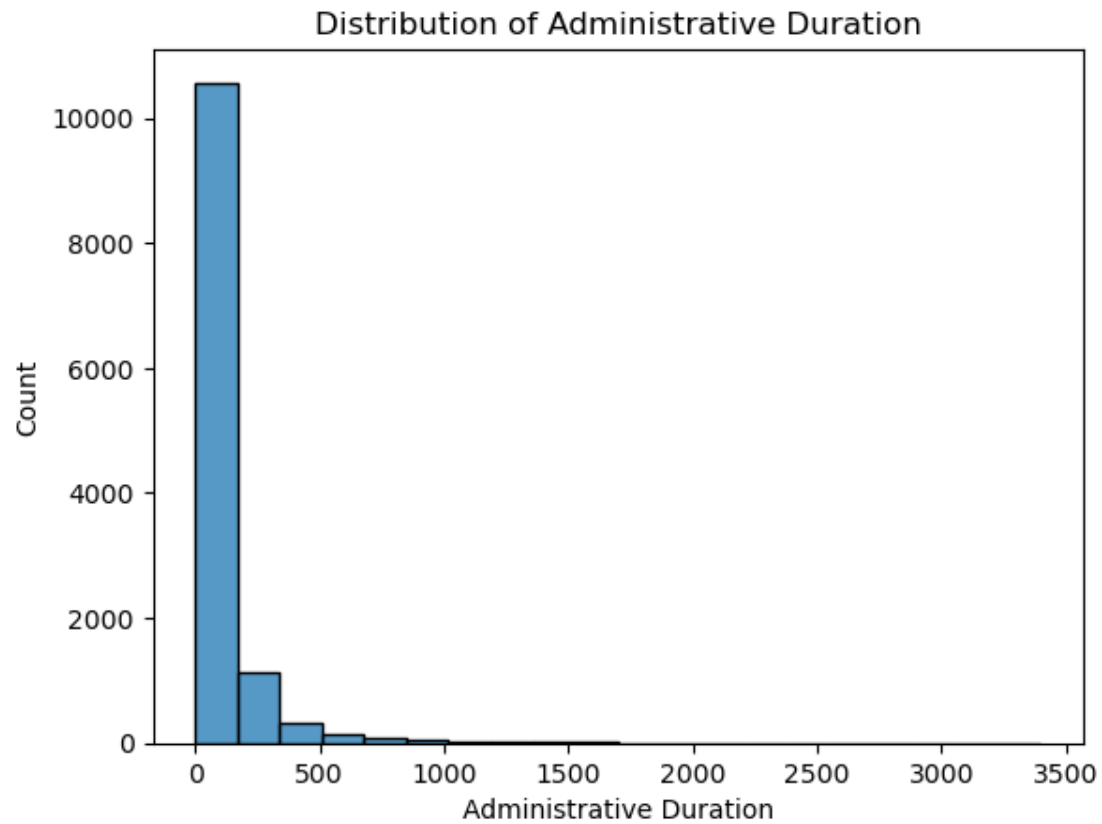
sns.countplot(x='OperatingSystems', data=df)
plt.title('Frequency of Operating Systems')
plt.xlabel('Operating Systems')
plt.ylabel('Frequency')
plt.show()

```



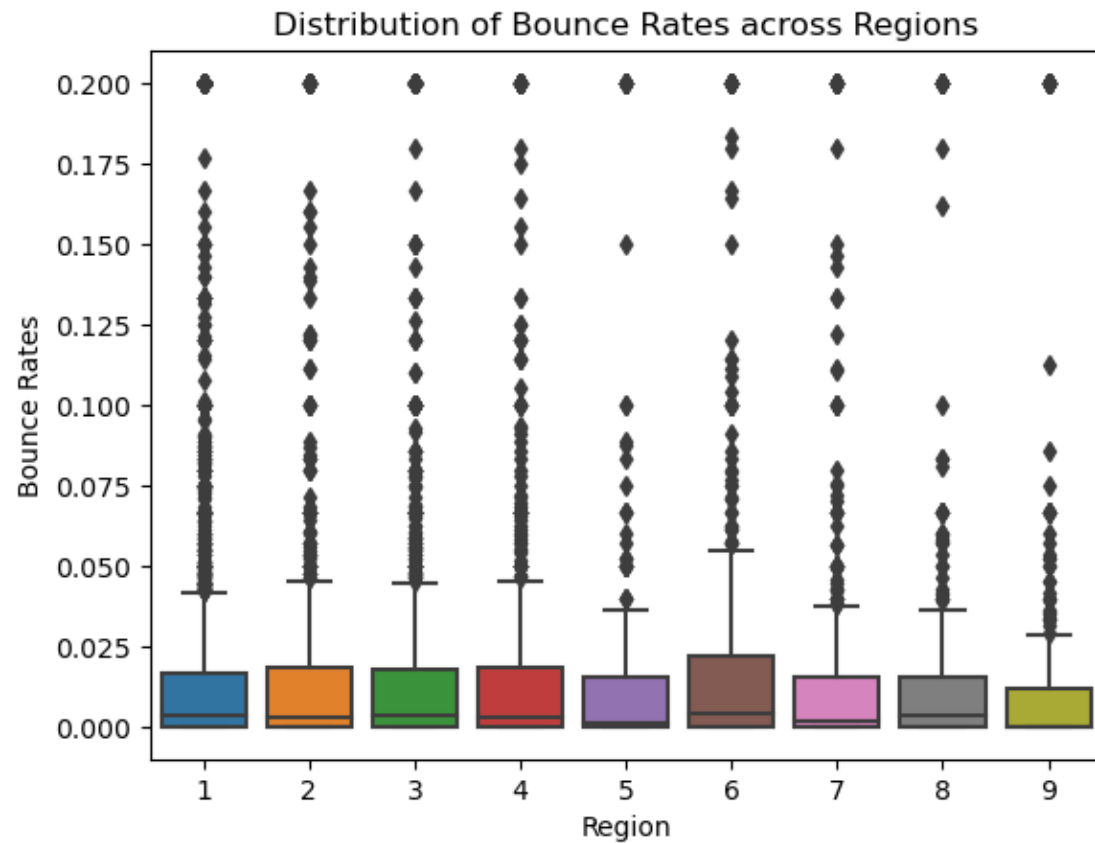
**A histogram showing the distribution of the time spent on the website by the visitors**

```
sns.histplot(x='Administrative_Duration', data=df, bins=20)
plt.title('Distribution of Administrative Duration')
plt.xlabel('Administrative Duration')
plt.ylabel('Count')
plt.show()
```



**A box plot showing the distribution of the bounce rates across different regions**

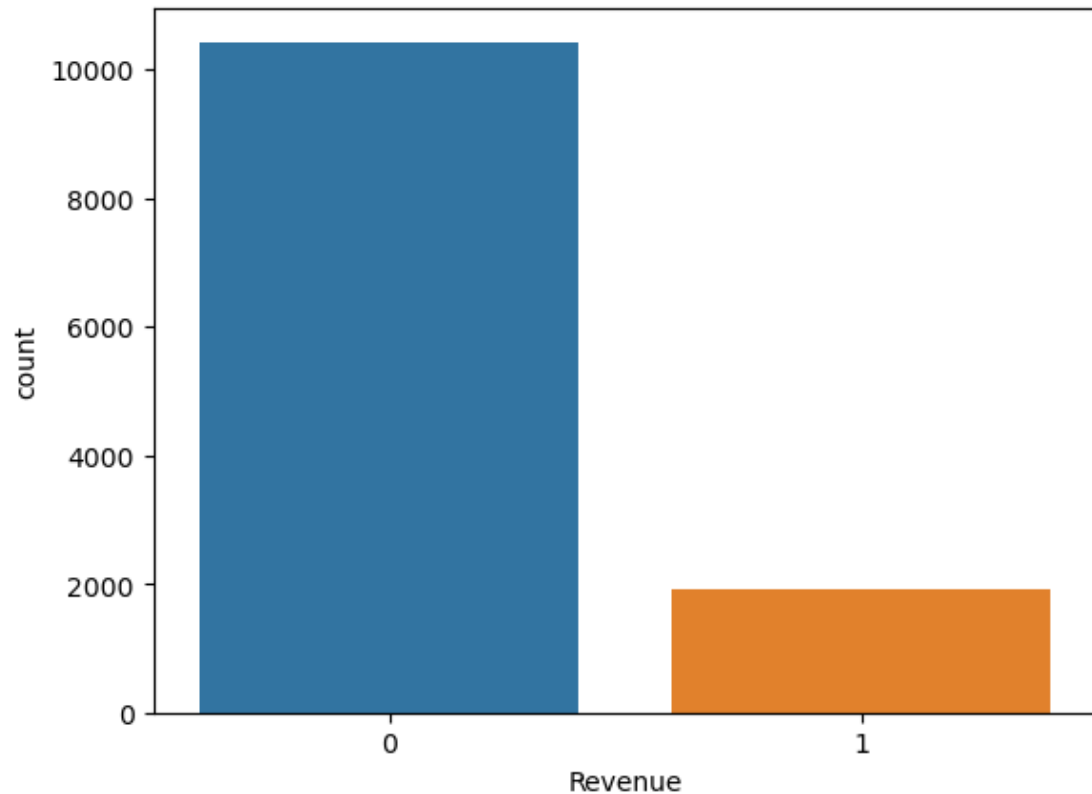
```
sns.boxplot(x='Region', y='BounceRates', data=df)
plt.title('Distribution of Bounce Rates across Regions')
plt.xlabel('Region')
plt.ylabel('Bounce Rates')
plt.show()
```



### Countplot of the Target Variable

```
sns.countplot(x='Revenue', data=df)  
plt.show()
```





# Feature Selection and Feature Engineering

## Dropping the Target Variable from df

```
df.drop('Revenue', axis=1)
```

	Administrative	Administrative_Duration	Informational	\
0	0	0.0	0	
1	0	0.0	0	
2	0	0.0	0	
3	0	0.0	0	
4	0	0.0	0	
...	...	...	...	
12325	3	145.0	0	
12326	0	0.0	0	
12327	0	0.0	0	
12328	4	75.0	0	
12329	0	0.0	0	

	Informational_Duration	ProductRelated	ProductRelated_Duration	\
0	0.0	1	0.000000	
1	0.0	2	64.000000	
2	0.0	1	0.000000	
3	0.0	2	2.666667	
4	0.0	10	627.500000	

```

...
12325      0.0      53      1783.791667
12326      0.0      5      465.750000
12327      0.0      6      184.250000
12328      0.0     15      346.000000
12329      0.0      3      21.250000

      BounceRates  ExitRates  PageValues  SpecialDay  OperatingSystems \
0      0.200000    0.200000    0.000000      0.0      1
1      0.000000    0.100000    0.000000      0.0      2
2      0.200000    0.200000    0.000000      0.0      4
3      0.050000    0.140000    0.000000      0.0      3
4      0.020000    0.050000    0.000000      0.0      3
...
12325      0.007143    0.029031    12.241717      0.0      4
12326      0.000000    0.021333     0.000000      0.0      3
12327      0.083333    0.086667     0.000000      0.0      3
12328      0.000000    0.021053     0.000000      0.0      2
12329      0.000000    0.066667     0.000000      0.0      3

      Browser  Region  TrafficType  Weekend  VisitorType
0      1      1      1      0      1
1      2      1      2      0      1
2      1      9      3      0      1
3      2      2      4      0      1
4      3      1      4      1      1
...
12325      6      1      1      1      1
12326      2      1      8      1      1
12327      2      1     13      1      1
12328      2      3     11      0      1
12329      2      1      2      1      0

```

[12330 rows x 16 columns]

## Selecting X and y columns

```

X = df.drop('Revenue', axis=1)
y = df['Revenue']

```

# PCA(Principal Component Analysis)

## Feature Scaling

```

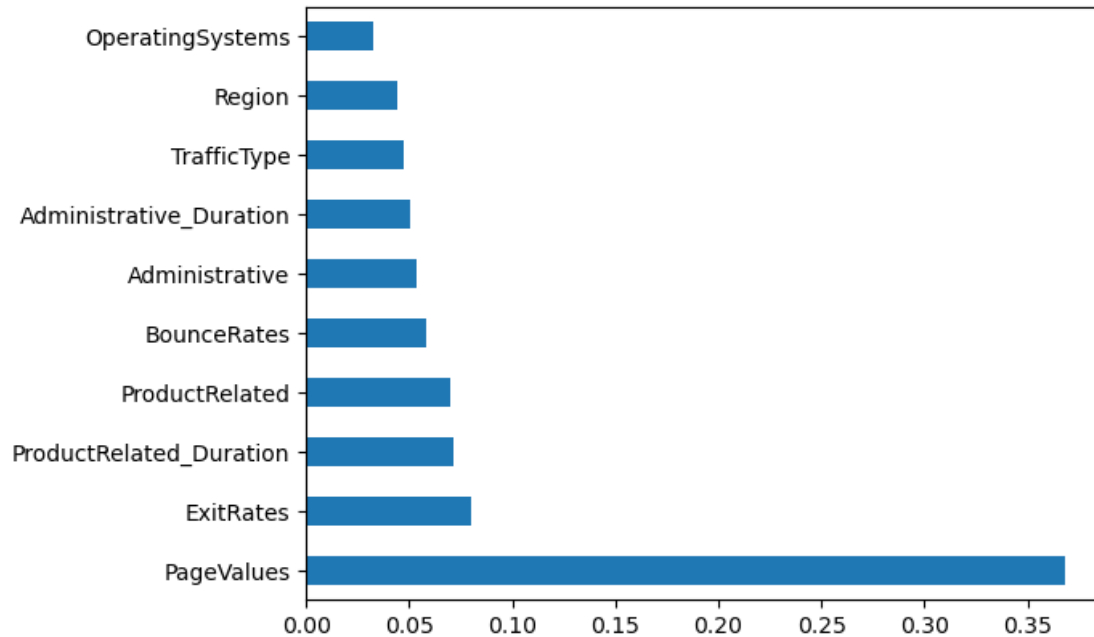
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X = pd.DataFrame(StandardScaler().fit_transform(X), columns=X.columns, index=X.index)

```

## Plotting feature importance graph

```
import pandas as pd
import numpy as np
from sklearn.ensemble import ExtraTreesClassifier
import matplotlib.pyplot as plt
model = ExtraTreesClassifier()
model.fit(X,y)
print(model.feature_importances_) #use inbuilt class feature_importances of tree based classifiers
#plot graph of feature importances for better visualization
feat_importances = pd.Series(model.feature_importances_, index=X.columns)
feat_importances.nlargest(10).plot(kind='barh')
plt.show()

[0.05349017 0.05046701 0.02870005 0.02927102 0.06956014 0.07148338
 0.05831459 0.08009879 0.36793201 0.0083054 0.0323917 0.03230736
 0.04460658 0.04756339 0.01348036 0.01202805]
```



```
top4 = feat_importances.nlargest(4).index
```

```
# Print the feature importances of the top 4 features
print(feat_importances[top4])
```

```
PageValues          0.367932
ExitRates           0.080099
ProductRelated_Duration 0.071483
ProductRelated      0.069560
dtype: float64
```

Here we see that 'PageValues' has the height importance and is greater as compared to others, There is a huge difference after the fourth feature hence selecting top 4 features and will apply feature scaling to normalize the data points

### Dropping unnecessary columns

```
X =
X.drop(['Administrative', 'Administrative_Duration', 'Informational', 'Informational_Duration', 'BounceRates', 'SpecialDay', 'OperatingSystems', 'Browser', 'Region', 'TrafficType', 'VisitorType', 'Weekend'], axis=1)
```

## Model Building and Model Selection

### Importing necessary libraries and performing train test split

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
```

### Building Various ML Models

```
# train logistic regression model
lr = LogisticRegression()
lr.fit(X_train, y_train)
lr_pred = lr.predict(X_test)
lr_accuracy = accuracy_score(y_test, lr_pred)
```

```
# train decision tree model
dt = DecisionTreeClassifier()
dt.fit(X_train, y_train)
dt_pred = dt.predict(X_test)
dt_accuracy = accuracy_score(y_test, dt_pred)
```

```
# train random forest model
rf = RandomForestClassifier()
rf.fit(X_train, y_train)
rf_pred = rf.predict(X_test)
rf_accuracy = accuracy_score(y_test, rf_pred)
```

```
#K-Nearest Neighbors (KNN) Classifier
```

```
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(X_train, y_train)
knn_pred = knn.predict(X_test)
knn_accuracy = accuracy_score(y_test, knn_pred)
```

```
#Naive Bayes Classifier
```

```
gnb = GaussianNB()
gnb.fit(X_train, y_train)
gnb_pred = gnb.predict(X_test)
gnb_accuracy = accuracy_score(y_test, gnb_pred)
```

```
# print model accuracies
```

```
print('Logistic Regression Accuracy:', lr_accuracy)
print('Decision Tree Accuracy:', dt_accuracy)
print('Random Forest Accuracy:', rf_accuracy)
print('K-Nearest Neighbors Accuracy:', knn_accuracy)
print('Naive Bayes Classifier Accuracy:', gnb_accuracy)
```

```
Logistic Regression Accuracy: 0.8690186536901865
Decision Tree Accuracy: 0.85117599351176
Random Forest Accuracy: 0.884022708840227
K-Nearest Neighbors Accuracy: 0.8767234387672344
Naive Bayes Classifier Accuracy: 0.8661800486618005
```

## Plotting Graph for model selection

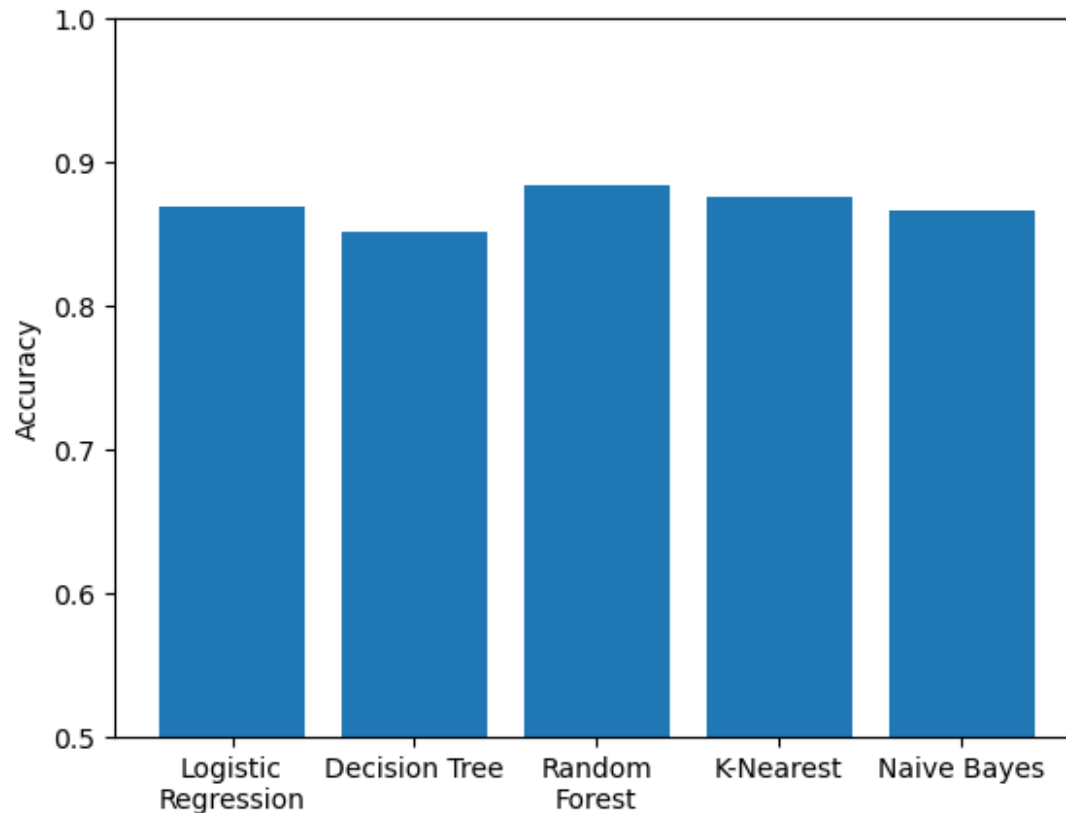
```
# plot model accuracies
```

```
models = ['Logistic\nRegression', 'Decision Tree', 'Random\nForest', 'K-Nearest', 'Naive\nBayes']
accuracies = [lr_accuracy, dt_accuracy, rf_accuracy, knn_accuracy, gnb_accuracy]
```

```
plt.bar(models, accuracies)
plt.ylim(0.5, 1)
plt.ylabel('Accuracy')
plt.show()
```

```
# select best model based on accuracy
```

```
best_model = models[accuracies.index(max(accuracies))]
print('Best Model:', best_model)
```



Best Model: Random Forest

**We find out that best model to work on is Random forest**

## Hyper-parameter Tuning

### Making Grid for Gridsearch

```
from sklearn.model_selection import GridSearchCV
```

```
param_grid = {
    'n_estimators': [100,150,200],
    'max_depth': [2,3,5],
    'min_samples_split': [ 3,7,10],
    'min_samples_leaf': [2, 4,5]
}
```

### Finding out the best parameters

```
from joblib import parallel_backend
```

```
with parallel_backend('threading', n_jobs=10):
```

```
    rf = RandomForestClassifier()
    rf_tuned = GridSearchCV(rf, param_grid, cv=5)
    rf_tuned.fit(X_train, y_train)
    best_params = rf_tuned.best_params_
    print(f"Best Parameters: {best_params}")
```

```
Best Parameters: {'max_depth': 5, 'min_samples_leaf': 4, 'min_samples_split': 7,
'n_estimators': 200}
```

## Making a classification report of your tuned model

```
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
```

```
rf_tuned = RandomForestClassifier(**best_params)
rf_tuned.fit(X_train, y_train)
y_pred = rf_tuned.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"Tuned Random Forest:")
print(f"Accuracy: {accuracy}")
print(f"Confusion Matrix: {confusion_matrix(y_test, y_pred)}")
print(f"Classification Report: {classification_report(y_test, y_pred)}")
```

Tuned Random Forest:

Accuracy: 0.8925385239253852

Confusion Matrix: [[1980 75]  
[ 190 221]]

Classification Report:		precision	recall	f1-score	support
0	0.91	0.96	0.94	2055	
1	0.75	0.54	0.63	411	
accuracy			0.89	2466	
macro avg	0.83	0.75	0.78	2466	
weighted avg	0.88	0.89	0.89	2466	

## Model Deployment

### Dumping our tuned model for deployment

```
import pickle
```

```
with open('C:/Users/ridit/Downloads/aaaa/project.pkl', 'wb') as f:
    pickle.dump(rf_tuned, f)
```

## Selecting Range of Parameters using min-max and histograms

```
df['PageValues'].max()
```

```
361.7637419
```

```
df['ExitRates'].max()
```

```
0.2
```

```
df['ProductRelated_Duration'].max()
```

```
63973.52223
```

```
df['ProductRelated'].max()
```

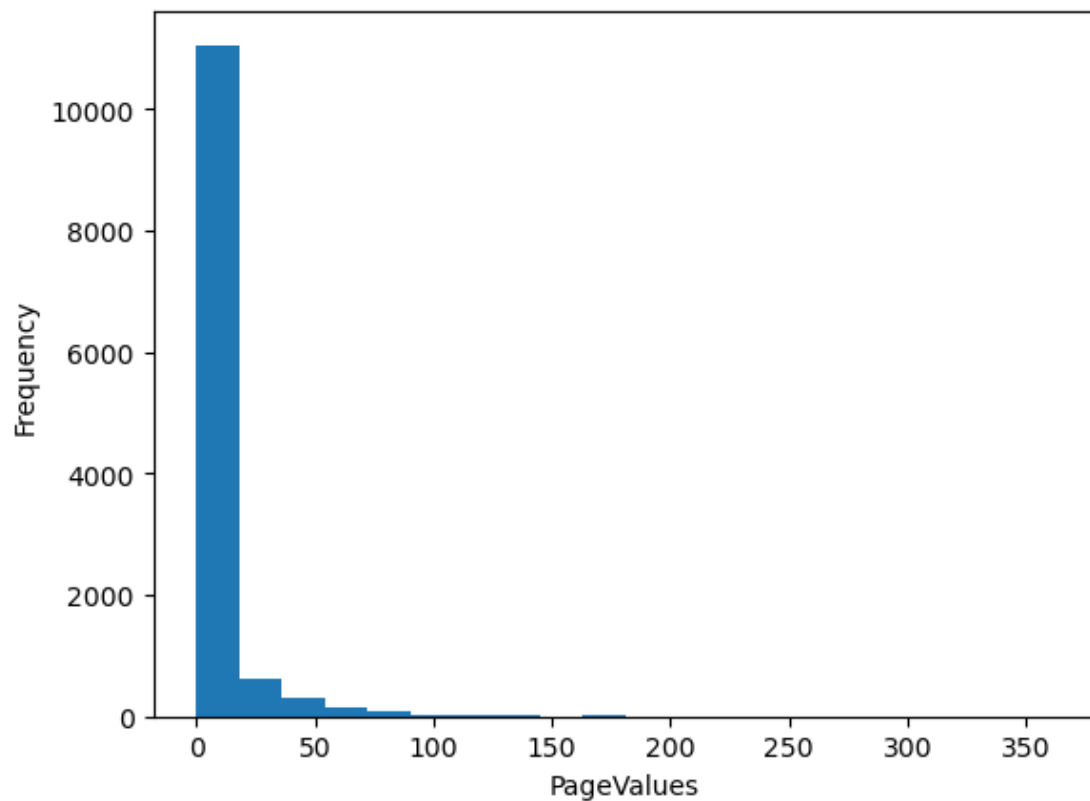
```
705
```

```
plt.hist(df["PageValues"], bins=20)
```

```
plt.xlabel("PageValues")
```

```
plt.ylabel("Frequency")
```

```
plt.show()
```



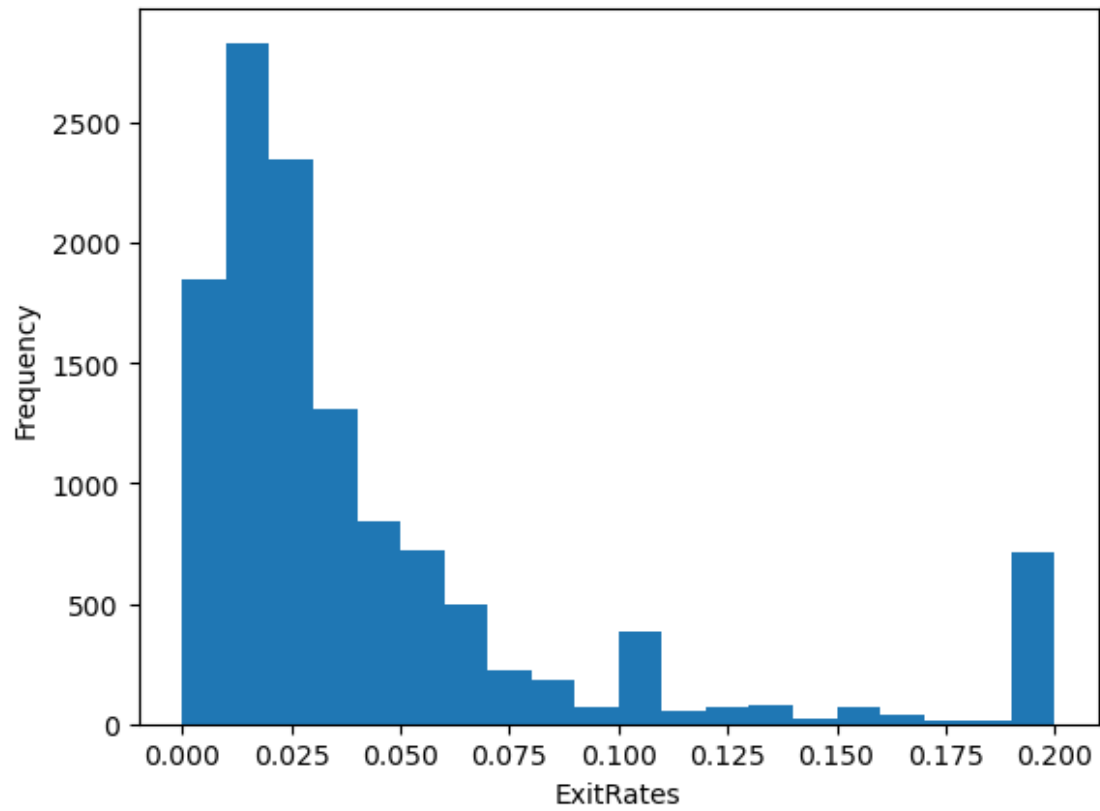
```
plt.hist(df["ExitRates"], bins=20)
```

```
plt.xlabel("ExitRates")
```

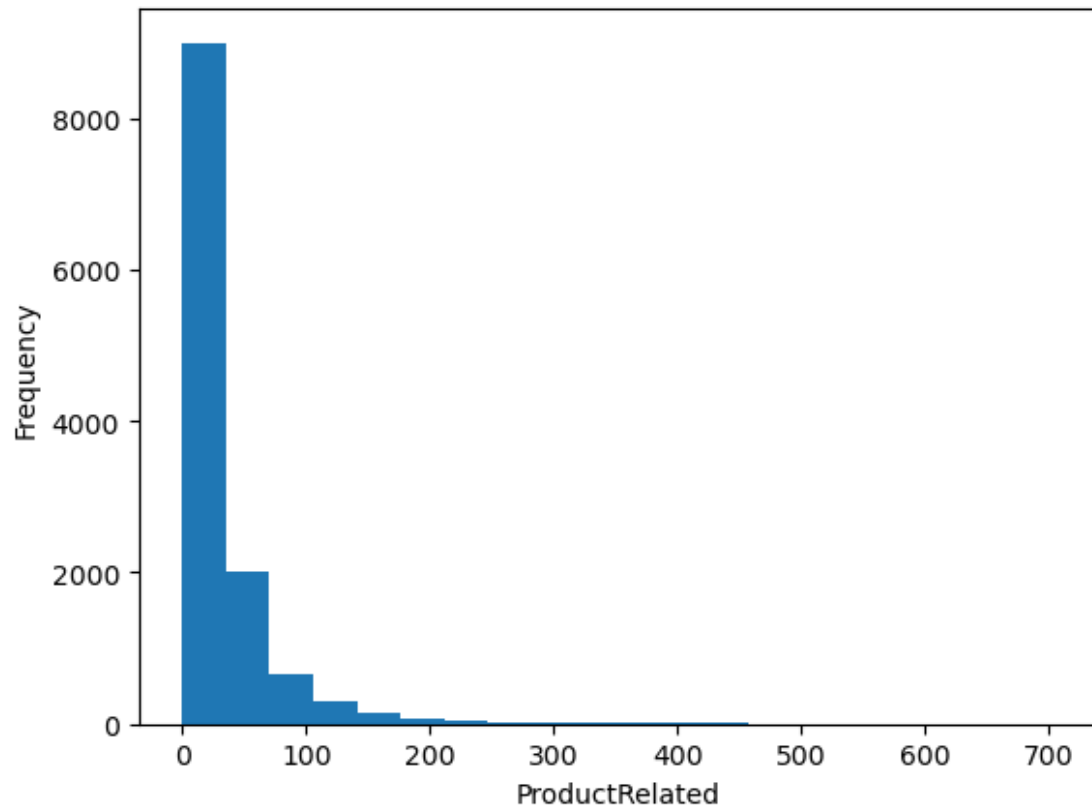
```
plt.ylabel("Frequency")
```

```
plt.show()
```





```
plt.hist(df["ProductRelated"], bins=20)
plt.xlabel("ProductRelated")
plt.ylabel("Frequency")
plt.show()
```



```
plt.hist(df["ProductRelated_Duration"], bins=20)
plt.xlabel("ProductRelated_Duration")
plt.ylabel("Frequency")
plt.show()
```

## Following is the code of API file (app.py)

```
import streamlit as st
import numpy as np
import pickle

#Importing our pickle file
model = pickle.load(open('project.pkl', 'rb'))

#Title of app
st.title('Will the person make a purchase or not')

#Range of paramenters
PageValues = st.slider("PageValues",0.00,200.76)
ExitRates = st.slider("ExitRates",0.000,0.200,step=0.001,format="%.3f")
ProductRelated_Duration = st.slider("ProductRelated_Duration",0.00,15000.00)
ProductRelated = st.slider("ProductRelated",0,200)

#Pridiction function
def predict():
    float_features = [float(x) for x in [PageValues, ExitRates, ProductRelated_Duration,
ProductRelated]]
    final_features = [np.array(float_features)]
    prediction = model.predict(final_features)
    label = prediction[0]

    print(type(label))
    print(label)

#Printing the output
    if(int(label)==1):
        st.success('Hureyyyy!! The costomer will make a purchase ' + ' :thumbsup:')
    else:
        st.success('Ohhh The costomer wont make a purchase ' + ' :thumbsup:')

trigger = st.button('Predict', on_click=predict)
```