# E-SHOP CLOTHING DATA ANALYSIS

Pradyumna Kombethota Ramgopal

Department of Computer Science, Virginia Tech

CS 5805: Machine Learning I

Dr. Reza Jafari

## Table of Contents

## Table of Figures and Tables

**Table**



**Figures**

| | |
|---|---|
| **17** | Condition Number - Numerical Target |
| **18** | SVD - Numerical Target |
| **19** | SVD - Reduced Matrix |
| **20** | Feature Importance - Random Forest Regressor - Numerical Target |
| **21** | Features Based on Threshold - Numerical Target |
| **22** | VIF - Numerical Target |
| **23** | Covariance Matrix - Numerical Target |
| **24** | Pearson Correlation Coefficient - Numerical Target |
| **25** | PCA for the categorical target with ten components |
| **26** | Explained Variance Ratio - Categorical Target - Ten components |
| **27** | PCA for the categorical target with nine components |
| **28** | Explained Variance Ratio - Categorical Target - Nine Components |
| **29** | Condition Number - Categorical Target |
| **30** | SVD - Categorical Target |
| **31** | SVD - Reduced Matrix |
| **32** | Feature Importance - Random Forest Regressor - Categorical Target |
| **33** | Features - Based on Threshold - Categorical Target |
| **34** | VIF - Categorical Target |
| **35** | Covariance Matrix - Categorical Target |
| **36** | Pearson Correlation Coefficient - Categorical Target |

| | |
|---|---|
| **55** | ROC and AUC - Post-Pruned Decision Tree |
| **56** | Post-Pruned Decision Tree |
| **57** | Metrics - Logistic Regression |
| **58** | Confusion Matrix - Logistic Regression |
| **59** | ROC and AUC - Logistic Regression |
| **60** | Metrics - K-Nearest Neighbors |
| **61** | Confusion Matrix - K-Nearest Neighbors |
| **62** | ROC and AUC - K-Nearest Neighbors |
| **63** | Metrics - Naive Bayes Classifier |
| **64** | Confusion Matrix - Naive Bayes Classifier |
| **65** | ROC and AUC - Naive Bayes Classifier |
| **66** | Metrics - Support Vector Machine (Linear) |
| **67** | Confusion Matrix - Support Vector Machine (Linear) |
| **68** | ROC and AUC - Support Vector Machine (Linear) |
| **69** | Metrics - Support Vector Machine (Polynomial) |
| **70** | Confusion Matrix - Support Vector Machine (Polynomial) |
| **71** | ROC and AUC - Support Vector Machine (Polynomial) |
| **72** | Metrics - Support Vector Machine (Radial Basis Function) |
| **73** | Confusion Matrix - Support Vector Machine (Radial Basis Function) |
| **74** | ROC and AUC - Support Vector Machine (Radial Basis Function) |

**Abstract**

This project utilizes the E-Shop Clothing dataset, containing clickstream data from online stores catering to pregnant women. Its aim is to apply a range of machine-learning algorithms to this real-world dataset. Initially, feature engineering and exploratory data analysis techniques like PCA, SVD, VIF, Condition Number, and Random Forest Analysis were employed to reduce dimensions and address collinearity. In the subsequent phase, stepwise regression was applied to eliminate irrelevant features based on a predefined threshold and make predictions on the dataset's continuous variable. Phase three involved utilizing various machine learning classifiers to forecast the dependent variable, ultimately recommending the best-performing classifier. Finally, clustering and association rule mining algorithms were employed to unveil valuable trends and patterns within the dataset. This project enhanced a deeper grasp of feature engineering, exploratory data analysis, regression, classification, clustering, and association rule mining.

*Keywords:* Random Forest Analysis, PCA, Condition Number, SVD, VIF, Label Encoding, IQR, Stepwise Regression, Logistic Regression, KNN, SVM, Decision Tree, Naive Bayes, Random Forest, Neural Network, K-Means, DBSCAN, and Apriori.

## Introduction

The project utilizes the E-Shop Clothing dataset, spanning five months of 2008, to forecast product prices within specific categories. This dataset comprises details such as country, color, model imagery, orders, session IDs, page views, and pricing information. With a balanced target variable, both categorical and numerical features are analyzed. The project aims to uncover insights such as the top-selling clothing items, the correlation between clicks and sales volume, monthly sales trends, and the development of a model for predicting product prices based on various features. Additionally, a classification model is constructed to determine whether a product's price exceeds or falls below the average price for its category.

The project is structured into four parts:

**Phase One: Feature Engineering and EDA**

During this phase, we implemented data preprocessing techniques on the dataset. This involved eliminating missing data and duplicate entries. We aggregated the data for exploratory data analysis purposes. To address collinearity and select significant features, we applied various dimensionality reduction techniques such as principal component analysis, condition number analysis, random forest analysis, singular value decomposition, and variance inflation factor assessment. Categorical features underwent label encoding as part of discretization. Anomalies were identified and removed using the interquartile range (IQR) method. Finally, we assessed whether the target feature was balanced or imbalanced.

**Phase Two: Regression Analysis**

In this phase we made predictions on the continuous variable ‘price’ using backward stepwise regression. T-test analysis, F-test analysis, and confident interval analysis were performed to compare the actual and predicted value. In addition to that, we developed a table showing the R-squared, Adjusted R-square, AIC, BIC, and MSE.

**Phase Three: Classification Analysis**

In this phase we make use of decision trees, logistic regression, support vector machine, naive bayes, k-nearest neighbor, random forest, and neural network to train and test the model. We made use of grid search for hyper parameter search for each classifier. Here ‘price 2’ is used as the dependent variable. The performance of the model is evaluated through confusion matrix, precision, recall, specificity, F1-score, ROC, and AUC curve.

**Phase Four: Clustering and Association Rule Mining**

The last phase focuses on clustering and association rule mining. We used K-Means clustering to find the clusters within our dataset. Silhouette and Within Cluster Sum of Square were plotted to determine the optimal value of K. Apriori algorithm used for frequent set mining and association rule learning over the dataset.

## Description of the Dataset

The dataset of our interest E-Shop clothing dataset. It contains information on clickstream from an online store which offers clothing for pregnant women. The dataset is multivariate and sequential. We can perform regression, classification, clustering, and association rule mining on the datasets. It contains both numerical and categorical values with a total of 165,474 observations. It has fourteen features namely year, month, day, order, country, session id, page one, page two, color, location, model photography, price one, price two, and page.

**Feature Description**

- Year - The entire dataset has the value of year as 2008.
- Month - Categorical feature - The value ranges from April to August.
- Day - Categorical feature - The day number of the month.
- Order - Numerical feature - Sequence of clicks in a session.
- Country - Categorical feature - Country of origin of the IP address and this can take up to 47 values.
- Session Id - Numerical feature - Feature indicating session id.
- Page 1 - Categorical feature - It's concerned with the main category. 1=trousers, 2=skirts, 3=blouses, 4=sales.
- Page 2 - Categorical feature - It contains information about code for each product (217 products)
- Color - Categorical feature - The color of the product. 1=beige, 2=black, 3=blue,

4=brown, 5=burgundy, 6=gray, 7=green, 8=navy blue, 9=of many colors, 10=olive, 11=pink, 12=red, 13=violet, and 14=white.

- Location - Categorical feature - The location of the photo on the page. The screen has been divided into six parts. 1=top left, 2=top in the middle, 3=top right, 4=bottom left, 5=bottom in the middle, and 6= bottom right.
- Model photography - Categorical feature - Feature with two categories. 1=en face, 2=profile.
- Price - Numerical feature - Price of the product.
- Price 2 - Categorical feature - The variable informing whether the price of a particular product is higher than the average price for the entire product category.

Our target variable is ‘price’ for the regression and the rest are the independent variable. For the classification we use ‘price 2’ as the dependent variable and the rest as the independent variable. The categorical feature  page 2 is label encoded and the rest of them have already been label encoded. It provides insights into consumer behavior, such as browsing patterns, product preferences, and purchasing habits. This information can be valuable for market analysis, helping businesses understand their target audience and tailor marketing strategies accordingly. Overall, the E-Shop Clothing Dataset provides valuable insights into online consumer behavior and can be leveraged by businesses for strategic decision-making, product optimization, and enhancing the overall shopping experience.

## Phase 1 - Feature Engineering

### Data Cleaning

In the first part we loaded the dataset and looked at the features and their values. There are 165,474 observations in the dataset. We analyzed every aspect of the features through the describe method.

**Figure 1**

*Information about features*

More info about the dataframe:

| | year | month | day | order | country |
|---|---|---|---|---|---|
| count | 165474.0 | 165474.000000 | 165474.000000 | 165474.000000 | 165474.000000 |
| mean | 2008.0 | 5.585887 | 14.524554 | 9.817476 | 26.952621 |
| std | 0.0 | 1.328160 | 8.830374 | 13.478411 | 7.150691 |
| min | 2008.0 | 4.000000 | 1.000000 | 1.000000 | 1.000000 |
| 25% | 2008.0 | 4.000000 | 7.000000 | 2.000000 | 29.000000 |
| 50% | 2008.0 | 5.000000 | 14.000000 | 6.000000 | 29.000000 |
| 75% | 2008.0 | 7.000000 | 22.000000 | 12.000000 | 29.000000 |
| max | 2008.0 | 8.000000 | 31.000000 | 195.000000 | 47.000000 |

| | session ID | page 1 (main category) | colour | location |
|---|---|---|---|---|
| count | 165474.000000 | 165474.000000 | 165474.000000 | 165474.000000 |
| mean | 12058.417056 | 2.400842 | 6.227655 | 3.258198 |
| std | 7008.418903 | 1.144420 | 4.235606 | 1.713206 |
| min | 1.000000 | 1.000000 | 1.000000 | 1.000000 |
| 25% | 5931.000000 | 1.000000 | 3.000000 | 2.000000 |
| 50% | 11967.500000 | 2.000000 | 4.000000 | 3.000000 |
| 75% | 18219.000000 | 3.000000 | 9.000000 | 5.000000 |
| max | 24026.000000 | 4.000000 | 14.000000 | 6.000000 |

| | model photography | price | price 2 | page |
|---|---|---|---|---|
| count | 165474.000000 | 165474.000000 | 165474.000000 | 165474.000000 |
| mean | 1.260071 | 43.802507 | 1.488167 | 1.710166 |
| std | 0.438674 | 12.548131 | 0.499861 | 0.982412 |
| min | 1.000000 | 18.000000 | 1.000000 | 1.000000 |
| 25% | 1.000000 | 33.000000 | 1.000000 | 1.000000 |
| 50% | 1.000000 | 43.000000 | 1.000000 | 1.000000 |
| 75% | 2.000000 | 52.000000 | 2.000000 | 2.000000 |
| max | 2.000000 | 82.000000 | 2.000000 | 5.000000 |

Then we looked for any missing observations in the dataset and found out that there are no missing entries.

**Figure 2**

*Count of missing values in features*

```
The missing values of the dataframe is:
year                        0
month                       0
day                         0
order                       0
country                     0
session ID                  0
page 1 (main category)      0
page 2 (clothing model)     0
colour                      0
location                    0
model photography           0
price                       0
price 2                     0
page                        0
dtype: int64
```

**Duplication Removal**

The next step is to check for any duplicate observation in the dataset and we used the duplicated method in the pandas to check that. After operation we found that there are no duplicate rows in the dataset.

**Figure 3**

*Count of duplicates in the dataset*

```
Number of duplicate rows: 0
```

**Aggregation**

As part of the aggregation we applied group-by operation on month and aggregated based on order with summation operation. We found an interesting trend after the aggregation operation.

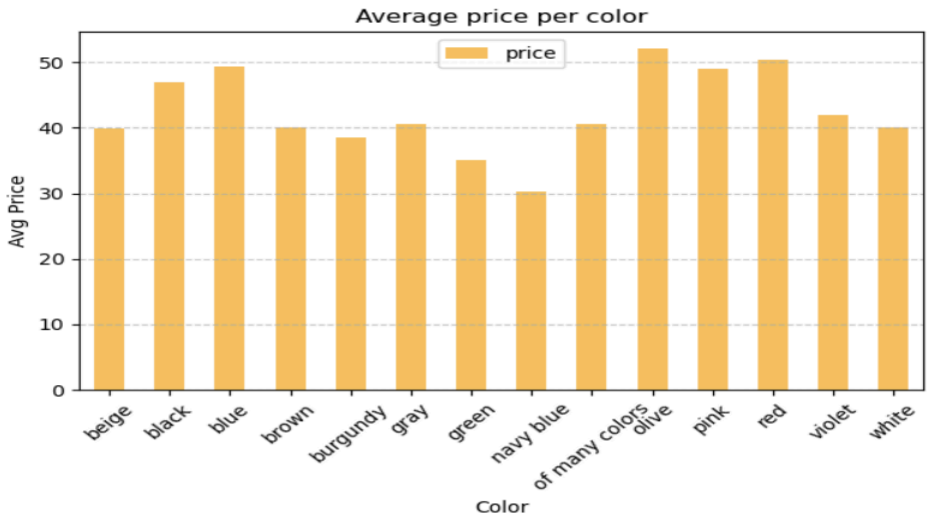
**Figure 4**

*Total order over months*


We also used the features price and color to find out the average price per color.

**Figure 5**

*Average price per color*




**Downsampling**

We downsampled the data using sample operation to choose 60,000 observations for future needs. While performing the downsampling we need to reset the index as it chooses randomly from the dataset.

**Figure 6**

*Downsampling*

```
downSampledData = dataFrame.sample(n=60000,random_state=5805).reset_index(drop=True)
```

**Label Encoding**

In this process we used LabelEncoder from sklearn to convert categorical variables to numerical variables. Here all the features are already label encoded by default except page 2 feature. So we encode only page 2 to numerical form.

**Figure 7**

*Label Encoding Page 2 feature*

```
labelEncoder = LabelEncoder()
dataFrame['page 2 (clothing model)'] = labelEncoder.fit_transform(dataFrame['page 2 (clothing model)'])
```

**Outlier Detection and Removal**

Outlier detection and removal is the crucial step in feature engineering. The dataset with outliers perform poorly in the classification and regression. So it is very important to remove outliers from the dataset before proceeding further. The approach we opted for outlier removal is the IQR method. In this approach we calculate the 25th and 75th quantile for the selected features. Then we calculate the IQR and use it to calculate upper bound and lower bound. Based on the upper bound and lower bound we removed the values that are beyond and below it..

```
Q1 = dataFrame[numerical_columns].quantile(0.25)
Q3 = dataFrame[numerical_columns].quantile(0.75)
IQR = Q3 - Q1
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

condition = (dataFrame[numerical_columns] >= lower_bound) & (dataFrame[numerical_columns] <= upper_bound)
```

We plotted before and after  box plots on the selected features to check whether outliers have been removed.

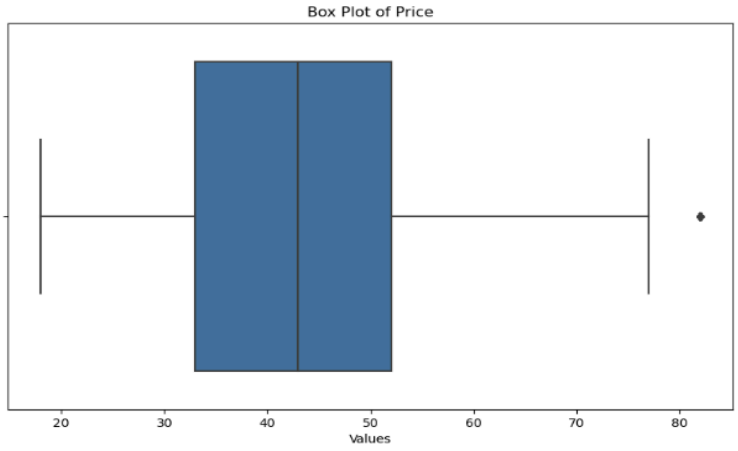
**Figure 8**

*Box plot of order before outlier removal*


We can notice the outliers present in the Order feature. These outliers might hurt the performance of our model.

**Figure 9**

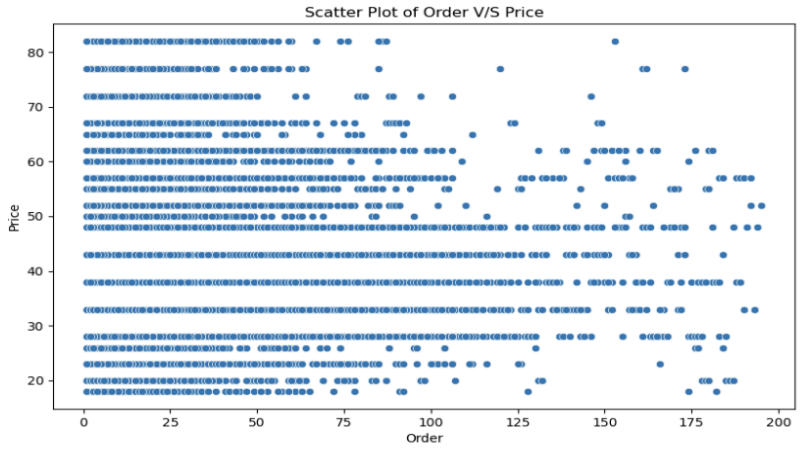
*Box plot of price before outlier removal*




Outliers are also available in the price feature and it should be removed before moving ahead.

**Figure 10**

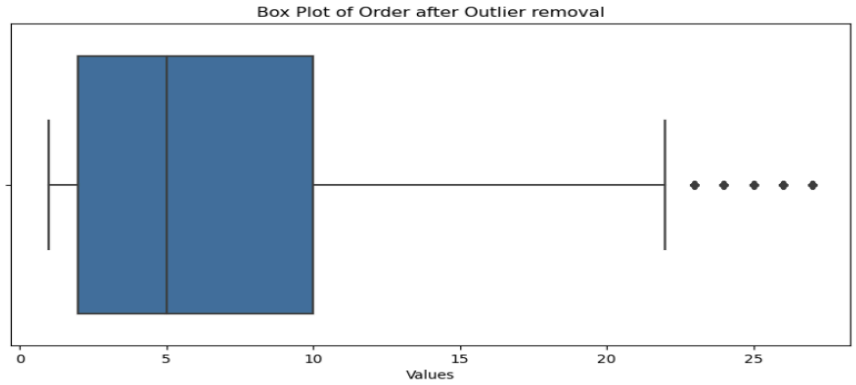
*Scatter plot of order versus price*




After looking into the scatter plot we can conclude that we have fewer observations for high orders and more observations when the order is low.

**Figure 11**

*Box plot of order after removal of outliers*

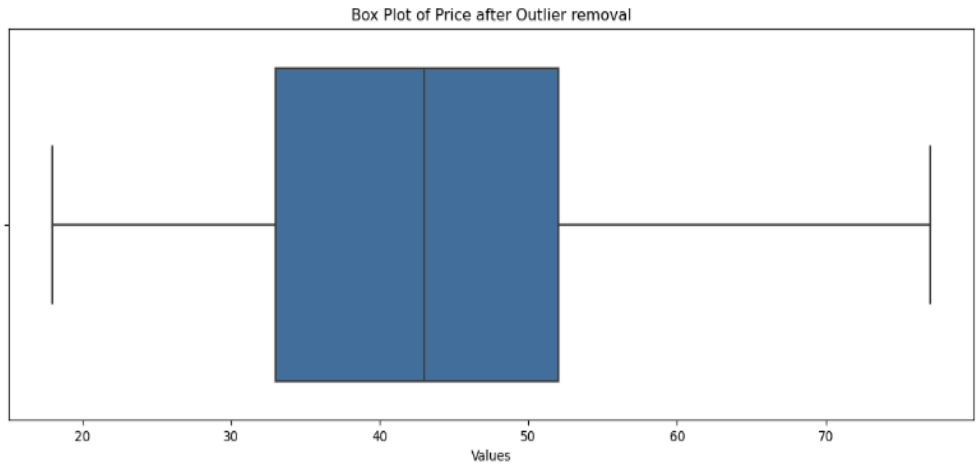



After outliers removal the dataset looks cleaner but there is still a small amount of outliers in the dataset. Exposing the model to outliers during training can improve its robustness and generalization performance. By learning to accommodate outliers, the model becomes more

resilient to unexpected or noisy data points. Removing outliers could result in the loss of important information, leading to a less informative model. Retaining outliers allows the model to capture the full range of variation present in the data.

**Figure 12**

*Box plot of price after outlier removal*




All the outliers are removed for the price feature after applying the IQR method.

We perform two different feature reduction techniques both for regression and classification analysis.

**Dimensionality reduction/Feature selection - Numerical Target**

***PCA and Condition Number***

We performed PCA and condition number on the standardized dataset and found the following observations.

- We need nine components to explain more than 95% of variance in the entire dataset.
- The condition number for the standardized dataset is infinity.

**Figure 13**

*PCA for the Numerical Target with thirteen components*

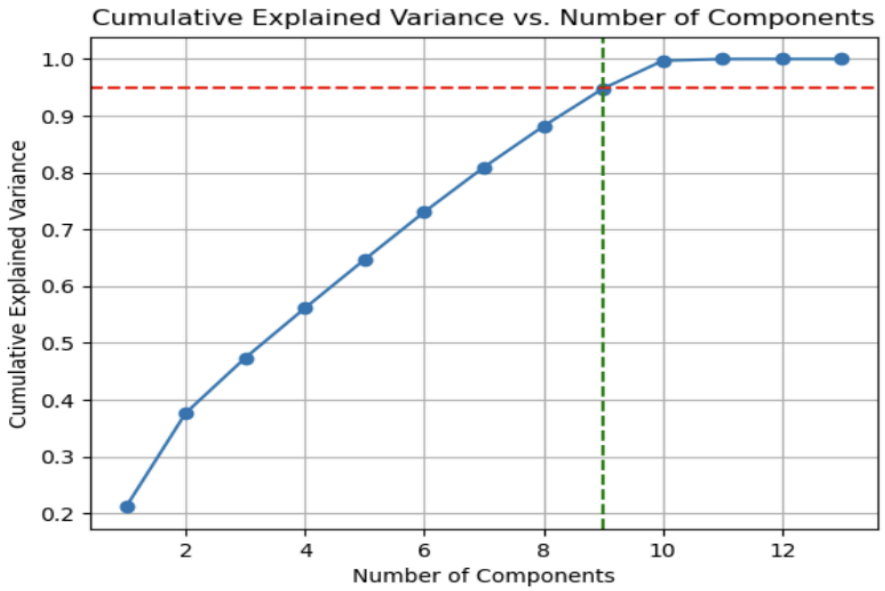



**Figure 14**

*Explained Variance Ratio*

```
The Explained Variance Ratio is: [2.11380973e-01 1.65277096e-01 9.71985966e-02 8.76067230e-02
 8.54199558e-02 8.37894504e-02 7.87844992e-02 7.23537813e-02
 6.62897600e-02 4.88336620e-02 2.98170715e-03 8.37963992e-05
 0.00000000e+00]
```

Then we transformed the feature matrix to contain ten features and calculated the PCA and explained variance ratio for that as well.

**Figure15**

*PCA for the Numerical Target with ten components*

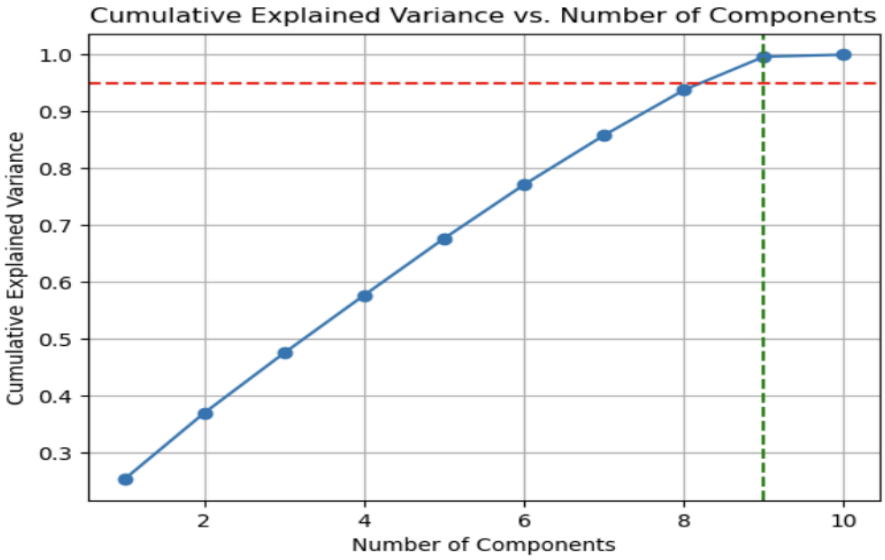



**Figure 16**

*Explained Variance Ratio for ten components*

```
The Explained Variance Ratio is: [0.25309012 0.11679154 0.10552205 0.1015245  0.09977816 0.09458505
 0.08683162 0.0796398  0.05865891 0.00357825]
```

We came up with the following conclusions for ten components.

- We need nine components to explain 95% variance in our data.
- The condition number drastically dropped to 8.4101

**Figure 17**

*Condition number on the original and reduced feature matrix*

```
The condition number for the original feature matrix:inf

The condition number for the reduced feature matrix:8.410121903284699
```

### *Singular Value Decomposition*

Singular value decomposition of the matrix was calculated and we saw a significant drop in the values for the last values.

**Figure 18**

*SVD of the original matrix*

```
The Singular Value Decomposition values are: [621.15648624 549.25550035 421.2094206  399.88654926 394.86419009
 391.07742736 379.21758489 363.41152402 347.84943479 298.55732101
  73.77353128  12.36746692   0.        ]
```

We selected only the important features from the dataset and again calculated the singular value decomposition of the matrix.

**Figure 19**

*SVD of the reduced matrix*

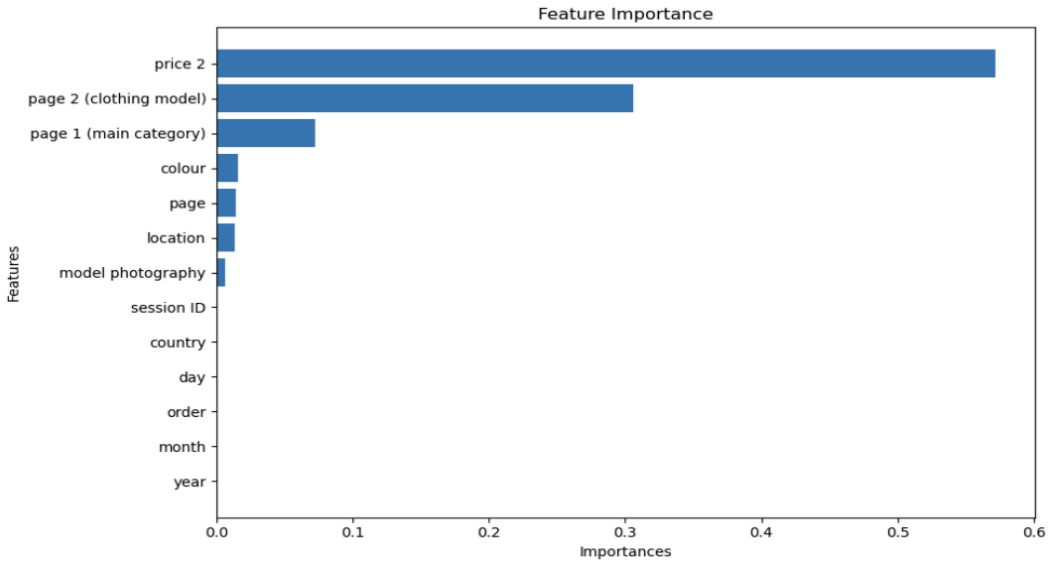
```
The Reduced Singular Value Decomposition values are: [620.46180539 421.48599675 400.63516998 392.97315734 389.57868921
 379.3051168  363.42633944 348.05071751 298.70635827  73.77560189]
```

### *Random Forest Analysis*

We ran random forest analysis on the features along with the dependent variable and found out that only few features are more important in the dataset. We plotted the graph for feature importance.

**Figure 20**

*Feature importance for the random forest regressor*




We set the threshold of 0.001 and found out that seven features are important according to random forest analysis.

**Figure 21**

*Selected features based on threshold*

```
The selected features are: ['price 2', 'page 2 (clothing model)', 'page 1 (main category)', 'colour', 'page', 'location', 'model photography']
```

### *Variance Inflation Factor*

VIF, or Variance Inflation Factor, is a statistical measure used to assess multicollinearity in regression analysis. Upon applying VIF we found that year, session id and month have very high VIF. We then used the threshold of five to drop the features which are contributing more to multicollinearity.

**Figure 22**

*Result of VIF on features*

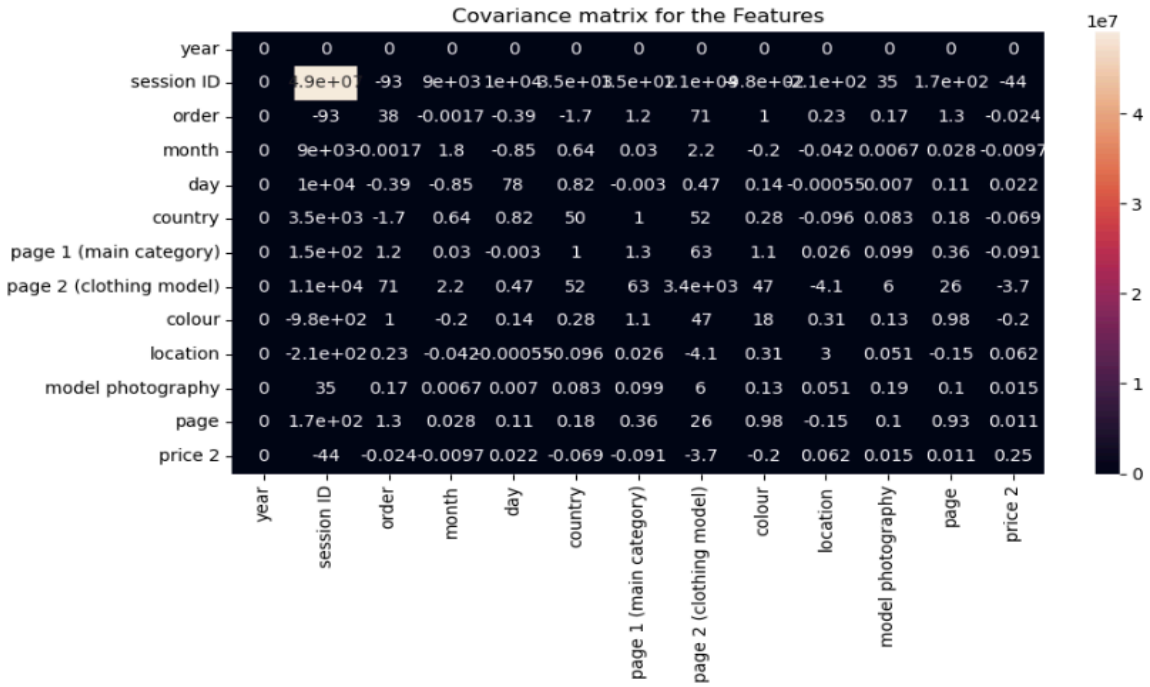
```
The VIFs are:
                     Feature          VIF
0                       year  3984.194388
1                 session ID   489.122123
2                      order     1.072189
3                      month   478.467384
4                        day    28.367786
5                    country     1.026899
6    page 1 (main category)    13.301469
7   page 2 (clothing model)    14.832735
8                     colour     1.144969
9                   location     1.050649
10        model photography     1.102100
11                     page     1.726990
12                  price 2     1.057023

The features selected by VIF are:
['order', 'country', 'colour', 'location', 'model photography', 'page', 'price 2']
```

### *Covariance Matrix*

A covariance matrix is a fundamental concept in statistics and machine learning that provides insights into the relationships between variables in a dataset. We plotted the covariance matrix for the dataset to understand the relation among features and dropped the features which are not important for our regression analysis.

**Figure 23**

*Covariance matrix of the features*

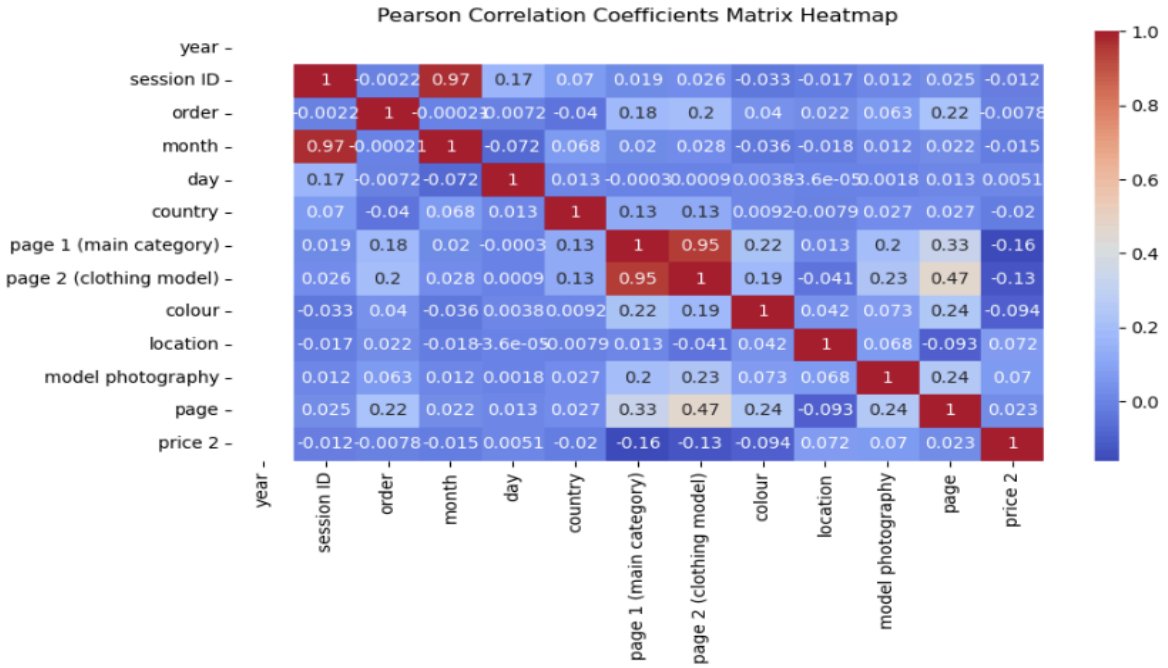



***Pearson Correlation Coefficient***

The Pearson correlation coefficient, often denoted as *r*, is a measure of the linear correlation between two variables. It quantifies the strength and direction of the linear relationship between variables. It is essential for evaluating relationships between variables, identifying dependencies, and making informed decisions in data analysis. We plotted the heatmap for it and understood the relation between features.

**Figure 24**

*Pearson correlation coefficient for the selected features.*




At the end after applying PCA, condition number, SVD, Random Forest Analysis, VIF, plotting covariance matrix and correlation coefficient matrix we found out that only few features are contributing in the dataset. The rest is causing multicollinearity and they are irrelevant. So we dropped the features which were considered redundant by the end of the feature reduction phase.

At last we selected order, day, country, page 1, page 2, color, location, model photography, page and price 2 for the regression analysis.

**Dimensionality reduction/Feature selection - Categorical target**

***PCA and Condition Number***

After performing PCA and condition number on the given standardized dataset we came up with the following conclusions.

- We need ten components to explain more than 95% variance in our data.
- The condition number for the standardized dataset is infinity

**Figure 25**

*PCA for the standardized data (Categorical target)*

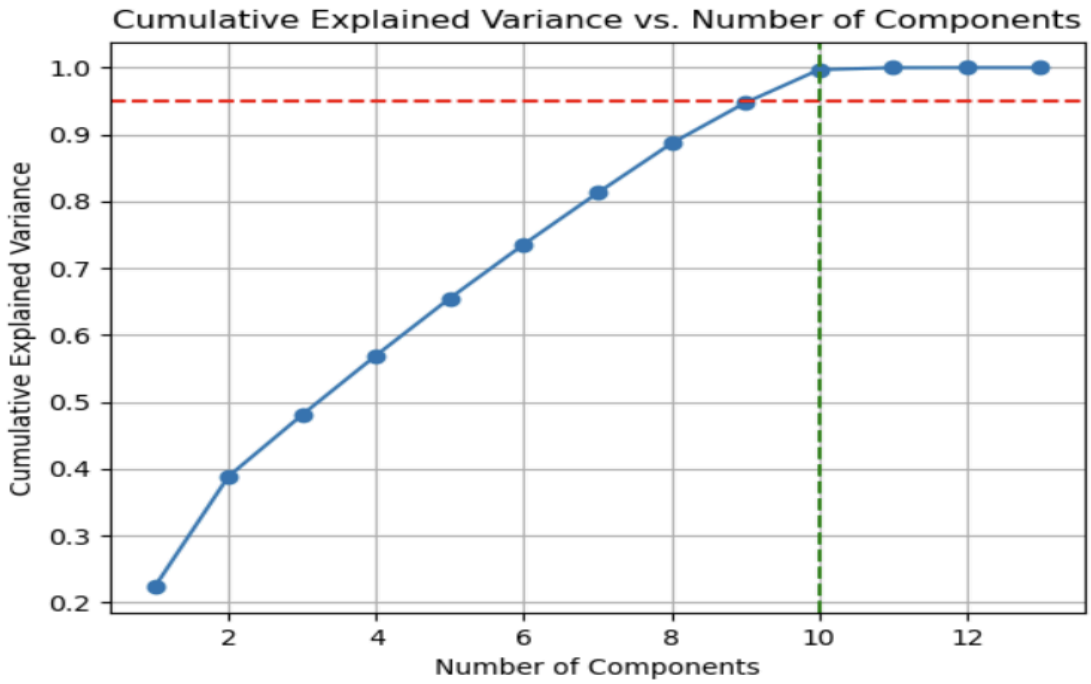


**Figure 26**

*Explained Variance Ratio (Categorical target)*

```
The Explained variance ratio for the categorical target is: [2.23072921e-01 1.65526772e-01 9.22313461e-02 8.88385718e-02
 8.55031420e-02 8.05385629e-02 7.68952791e-02 7.40728681e-02
 6.07783893e-02 4.94682784e-02 2.99006285e-03 8.38060312e-05
 0.00000000e+00]
```

We then dropped the irrelevant three features and performed PCA on it. We found that we need nine components to explain more than 95% variance in our data.

**Figure 27**

*PCA on the reduced standardized data (Categorical target)*

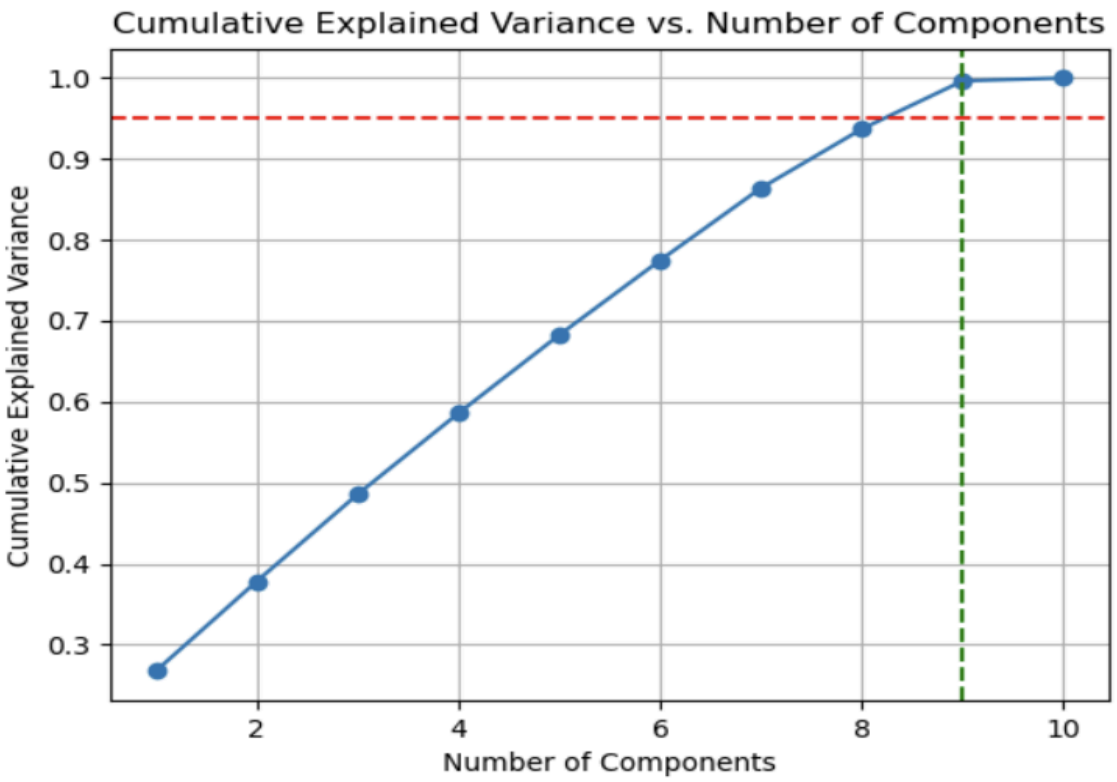


**Figure 28**

*Explained Variance Ratio of the reduced data (Categorical target)*

```
The Explained variance ratio for categorical target is: [0.26735972 0.11068061 0.10755109 0.1002082  0.09670932 0.09226791
 0.089274   0.07298017 0.05938079 0.0035882 ]
```

We saw a drastic drop in condition number after dropping some features. The new condition number of the data is 8.631

**Figure 29**

*Condition number on the original and reduced data (Categorical target)*

```
The conditional number of the features is: inf

The conditional number of the features is: 8.631960467161196
```

***Singular Value Decomposition***

We calculated the singular value decomposition on the data and we observed a drastic drop after certain values.

**Figure 30**

*SVD on the original data (Categorical target)*

```
The SVD values of the features are: [638.10405717 549.67021172 410.30551276 402.6881619  395.05641252
 383.41580966 374.64325445 367.70341139 333.07548731 300.49100538
  73.87682749  12.36817768   0.        ]
```

After dropping a few insignificant features we performed SVD on the reduced data as well.

**Figure 31**

*SVD on the reduced data (Categorical target)*

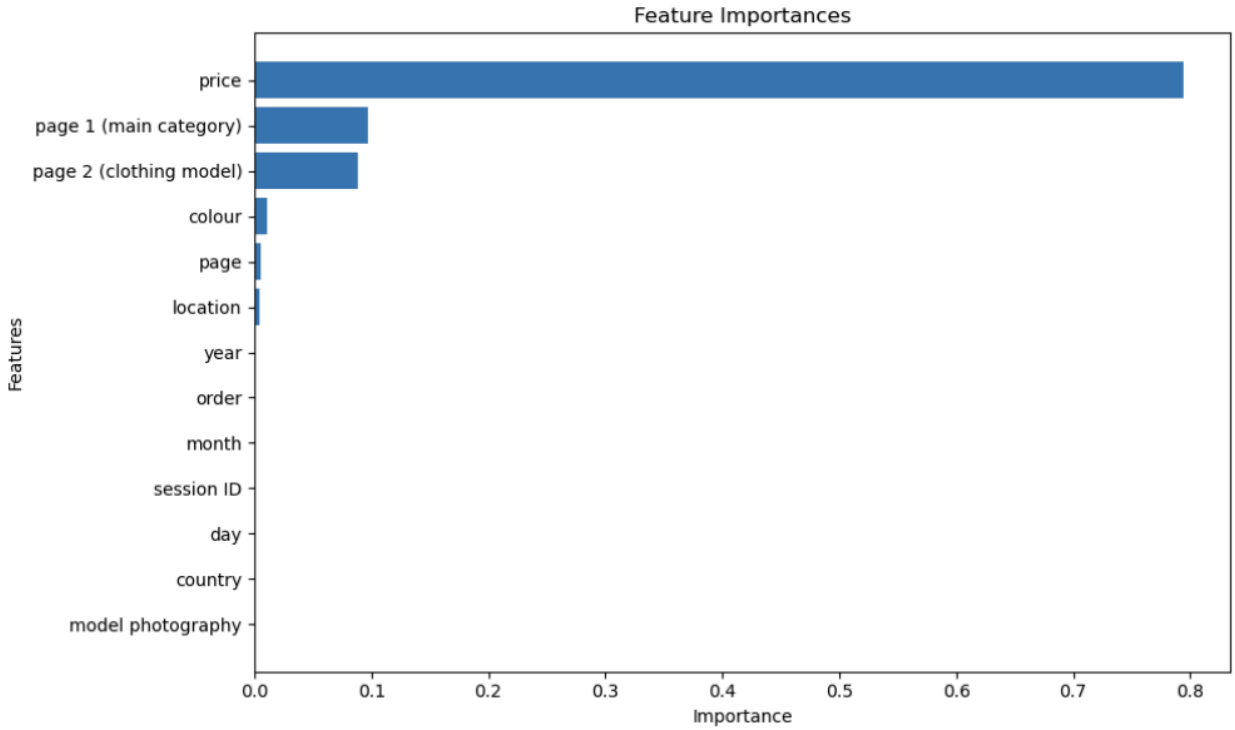
```
The SVD values of the features are: [637.71325566 410.31105655 404.46865482 390.41732552 383.54083877
 374.63021365 368.50208612 333.18073177 300.53871522  73.87814832]
```

### *Random Forest Analysis*

Random Forest provides a natural way to assess the importance of features in a dataset. We selected the categorical dependent variable along with the independent features for analysis. We plotted the important features based on analysis.

**Figure 32**

*Feature importance based on random forest analysis*




We set the threshold to 0.01 and found that only four features are important according to random forest analysis.

**Figure 33**

*Features based on importance (Categorical target)*

```
The selected features are: ['price', 'page 1 (main category)', 'page 2 (clothing model)', 'colour']
```

### *Variance Inflation Factor*

VIF, or Variance Inflation Factor, is a metric used to detect multicollinearity in analysis. Multicollinearity occurs when two or more predictor variables in a classification model are highly correlated, which can lead to issues such as unstable coefficient estimates and inflated standard errors. We calculated VIF on the given data and removed the features which have a VIF value more than 5.

**Figure 34**

*Result of VIF on features (Categorical target)*

```
The VIF values of the features:
                    Feature          VIF
0                      year  3984.139530
1                     order     1.072188
2                     month   478.412209
3                 session ID   489.066464
4                       day    28.364697
5                   country     1.026908
6    page 1 (main category)    13.277203
7   page 2 (clothing model)    14.815979
8                    colour     1.138723
9                  location     1.048766
10       model photography     1.116115
11                    page     1.722146
12                   price     1.160635
```

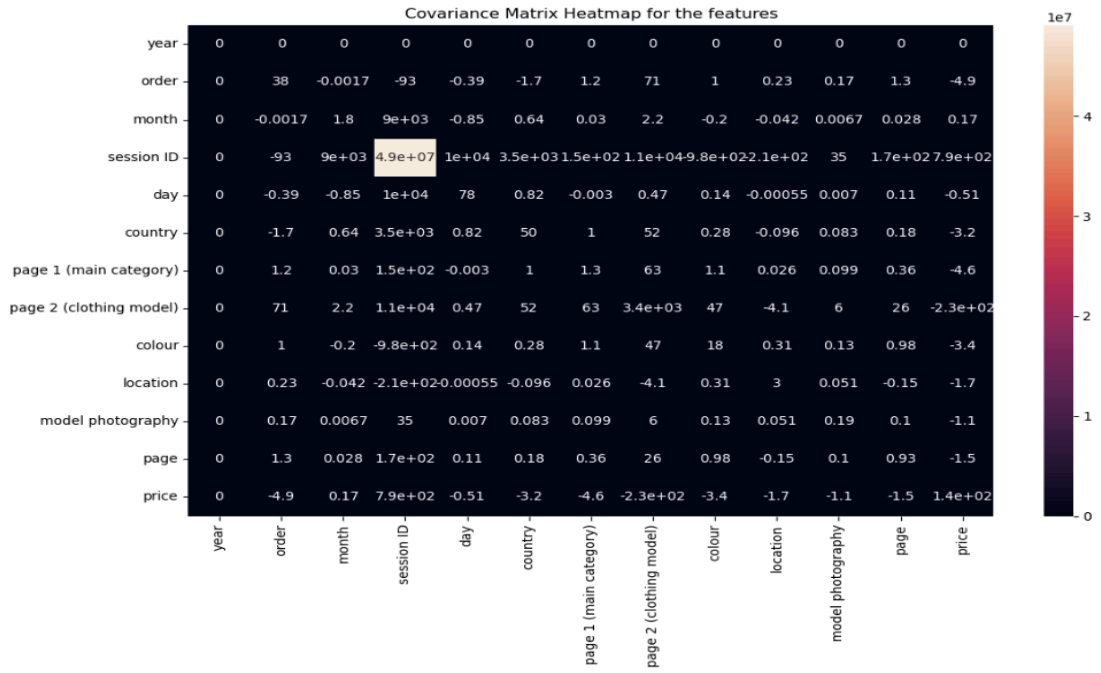
```
The selected features are: ['order', 'country', 'colour', 'location', 'model photography', 'page', 'price']
```

### *Covariance Matrix*

We plotted the covariance matrix to understand the covariance relationships between multiple variables in the dataset. It provides insights into how the variables change together, whether they move in the same direction or opposite directions.

**Figure 35**

*Covariance matrix of the features (Categorical Target)*

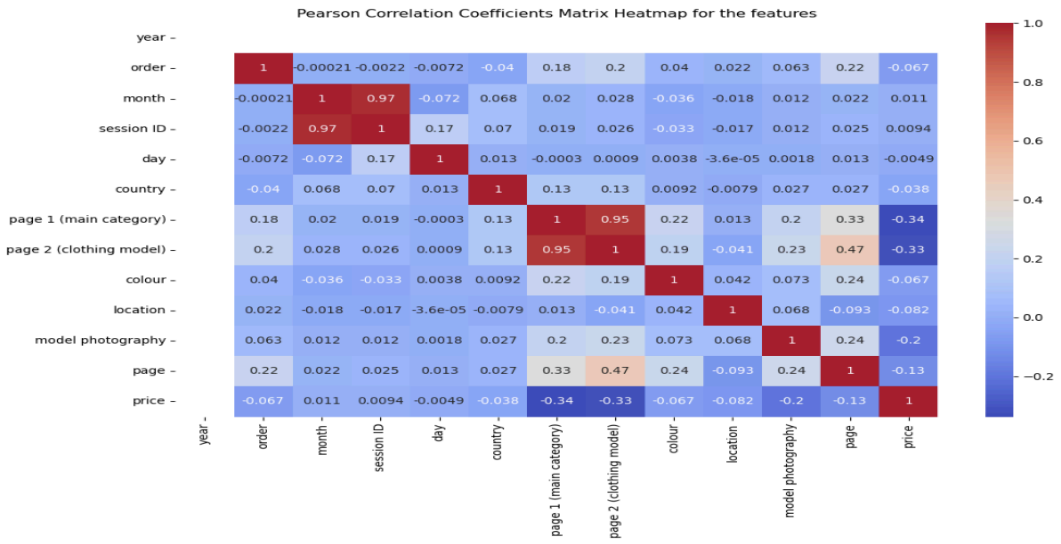



### *Pearson Correlation Coefficient*

Pearson correlation coefficient is essential for evaluating relationships between variables, identifying dependencies, and making informed decisions in data analysis. Pearson Correlation Coefficient was also performed to understand the feature importance.

**Figure 36**

*Pearson Correlation Coefficient of the features*




After performing PCA, Condition number, SVD, random forest analysis, VIF, sample covariance matrix, pearson correlation coefficient we dropped insignificant features year, session id, and page 2 from the dataset. We used the rest of the independent variables for the classification.

**Balanced or Imbalanced**

The dataset is balanced with almost equal numbers of positive and negative classes.

**Figure 37**

*Value count of the target feature*

```
 The number of positive and negative classes are:
1    77020
2    75089
```

## Phase 2 - Regression Analysis

In this phase of the analysis, we employed the backward stepwise regression model to predict a continuous numerical feature. It is a statistical technique used to understand the relationship between multiple predictor variables and a single continuous response variable. In the process we remove insignificant features to build a model. We used a threshold of 0.01 and removed the features having p value greater than the threshold value. In the process we watched the value of adjusted r-square and stopped fitting the model whenever there is a drastic drop in the value.Our objective is to predict the price of the product given the input features.

**Figure 38**

*Backward Stepwise Regression*

```
model = sm.OLS(y, sm.add_constant(X.iloc[:, included_features])).fit()
```

OLS Regression Results

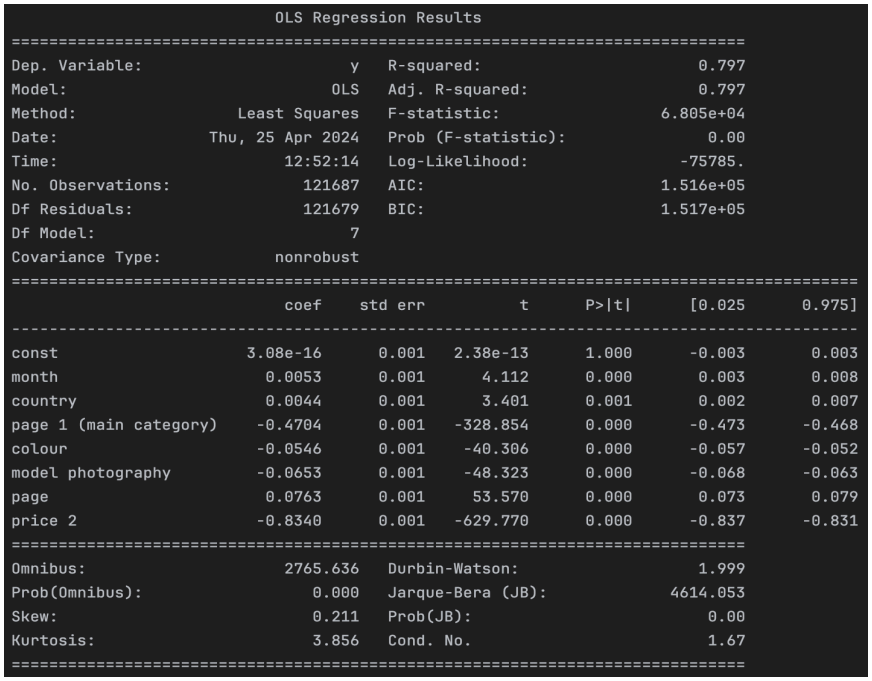
| | | | |
|---|---|---|---|
| Dep. Variable: | y | R-squared: | 0.797 |
| Model: | OLS | Adj. R-squared: | 0.797 |
| Method: | Least Squares | F-statistic: | 4.764e+04 |
| Date: | Thu, 25 Apr 2024 | Prob (F-statistic): | 0.00 |
| Time: | 12:52:14 | Log-Likelihood: | -75784. |
| No. Observations: | 121687 | AIC: | 1.516e+05 |
| Df Residuals: | 121676 | BIC: | 1.517e+05 |
| Df Model: | 10 | | |
| Covariance Type: | nonrobust | | |

| | coef | std err | t | P>\|t\| | [0.025 | 0.975] |
|---|---|---|---|---|---|---|
| const | 3.08e-16 | 0.001 | 2.38e-13 | 1.000 | -0.003 | 0.003 |
| order | -0.0003 | 0.001 | -0.210 | 0.833 | -0.003 | 0.002 |
| month | 0.0052 | 0.001 | 3.998 | 0.000 | 0.003 | 0.008 |
| day | -0.0015 | 0.001 | -1.169 | 0.242 | -0.004 | 0.001 |
| country | 0.0044 | 0.001 | 3.391 | 0.001 | 0.002 | 0.007 |
| page 1 (main category) | -0.4703 | 0.001 | -325.991 | 0.000 | -0.473 | -0.468 |
| colour | -0.0545 | 0.001 | -40.128 | 0.000 | -0.057 | -0.052 |
| location | -0.0014 | 0.001 | -1.100 | 0.272 | -0.004 | 0.001 |
| model photography | -0.0652 | 0.001 | -48.082 | 0.000 | -0.068 | -0.063 |
| page | 0.0761 | 0.001 | 52.185 | 0.000 | 0.073 | 0.079 |
| price 2 | -0.8338 | 0.001 | -627.586 | 0.000 | -0.836 | -0.831 |

| | | | |
|---|---|---|---|
| Omnibus: | 2766.772 | Durbin-Watson: | 1.999 |
| Prob(Omnibus): | 0.000 | Jarque-Bera (JB): | 4613.821 |
| Skew: | 0.211 | Prob(JB): | 0.00 |
| Kurtosis: | 3.856 | Cond. No. | 1.76 |

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.797
Model:                            OLS   Adj. R-squared:                  0.797
Method:                 Least Squares   F-statistic:                 5.293e+04
Date:                Thu, 25 Apr 2024   Prob (F-statistic):               0.00
Time:                        12:52:14   Log-Likelihood:                -75784.
No. Observations:              121687   AIC:                         1.516e+05
Df Residuals:                  121677   BIC:                         1.517e+05
Df Model:                           9
Covariance Type:            nonrobust
============================================================================================
                               coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------------------
const                    3.08e-16      0.001   2.38e-13      1.000      -0.003       0.003
month                      0.0052      0.001      3.999      0.000       0.003       0.008
day                       -0.0015      0.001     -1.167      0.243      -0.004       0.001
country                    0.0045      0.001      3.410      0.001       0.002       0.007
page 1 (main category)    -0.4704      0.001   -328.555      0.000      -0.473      -0.468
colour                    -0.0545      0.001    -40.147      0.000      -0.057      -0.052
location                  -0.0015      0.001     -1.109      0.267      -0.004       0.001
model photography         -0.0652      0.001    -48.083      0.000      -0.068      -0.062
page                       0.0761      0.001     52.973      0.000       0.073       0.079
price 2                   -0.8338      0.001   -627.591      0.000      -0.836      -0.831
==============================================================================
Omnibus:                     2766.562   Durbin-Watson:                   1.999
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             4612.516
Skew:                           0.211   Prob(JB):                         0.00
Kurtosis:                       3.855   Cond. No.                         1.71
==============================================================================
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.797
Model:                            OLS   Adj. R-squared:                  0.797
Method:                 Least Squares   F-statistic:                 5.955e+04
Date:                Thu, 25 Apr 2024   Prob (F-statistic):               0.00
Time:                        12:52:14   Log-Likelihood:                -75785.
No. Observations:              121687   AIC:                         1.516e+05
Df Residuals:                  121678   BIC:                         1.517e+05
Df Model:                           8
Covariance Type:            nonrobust
============================================================================================
                               coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------------------
const                    3.08e-16      0.001   2.38e-13      1.000      -0.003       0.003
month                      0.0052      0.001      4.014      0.000       0.003       0.008
day                       -0.0015      0.001     -1.165      0.244      -0.004       0.001
country                    0.0045      0.001      3.421      0.001       0.002       0.007
page 1 (main category)    -0.4704      0.001   -328.855      0.000      -0.473      -0.468
colour                    -0.0546      0.001    -40.305      0.000      -0.057      -0.052
model photography         -0.0653      0.001    -48.324      0.000      -0.068      -0.063
page                       0.0763      0.001     53.582      0.000       0.074       0.079
price 2                   -0.8340      0.001   -629.768      0.000      -0.837      -0.831
==============================================================================
Omnibus:                     2763.758   Durbin-Watson:                   1.998
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             4609.964
Skew:                           0.211   Prob(JB):                         0.00
Kurtosis:                       3.855   Cond. No.                         1.67
==============================================================================
```



```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.797
Model:                            OLS   Adj. R-squared:                  0.797
Method:                 Least Squares   F-statistic:                 6.805e+04
Date:                Thu, 25 Apr 2024   Prob (F-statistic):               0.00
Time:                        12:52:14   Log-Likelihood:                -75785.
No. Observations:              121687   AIC:                         1.516e+05
Df Residuals:                  121679   BIC:                         1.517e+05
Df Model:                           7
Covariance Type:            nonrobust
============================================================================================
                               coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------------------
const                    3.08e-16      0.001   2.38e-13      1.000      -0.003       0.003
month                      0.0053      0.001      4.112      0.000       0.003       0.008
country                    0.0044      0.001      3.401      0.001       0.002       0.007
page 1 (main category)    -0.4704      0.001   -328.854      0.000      -0.473      -0.468
colour                    -0.0546      0.001    -40.306      0.000      -0.057      -0.052
model photography         -0.0653      0.001    -48.323      0.000      -0.068      -0.063
page                       0.0763      0.001     53.570      0.000       0.073       0.079
price 2                   -0.8340      0.001   -629.770      0.000      -0.837      -0.831
==============================================================================
Omnibus:                     2765.636   Durbin-Watson:                   1.999
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             4614.053
Skew:                           0.211   Prob(JB):                         0.00
Kurtosis:                       3.856   Cond. No.                         1.67
==============================================================================
```

**Figure 39**

*Stepwise Regression Results*

```
Backward Stepwise Regression Results:

  Eliminated Feature   P-value             AIC             BIC  \
0              order  0.833332  151589.932130  151696.733412
1           location  0.267314  151587.976415  151685.068489
2                day  0.244109  151587.207002  151674.589869

   Adjusted R-squared
0            0.796531
1            0.796532
2            0.796532
```

We can see we've dropped order, location, and day feature while performing the fit as these features have high values of p. From the above picture we can also notice that the adjusted R-squared is constant while removing the insignificant features.

**Figure 40**

*Final selected features*

```
Final Selected Features:
Index(['month', 'country', 'page 1 (main category)', 'colour',
       'model photography', 'page', 'price 2'],
      dtype='object')
```

We've selected seven at the end of regression analysis and dropped three features which weren't contributing much to the model. The final equation of the model with features and coefficients can be seen below.

**Figure 41**

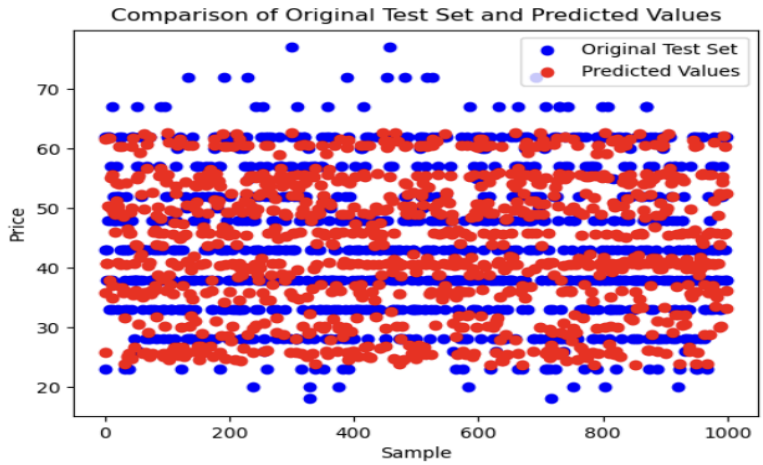
*Stepwise Regression Equation*

```
The final equation is:  (0.0053366507154653945 * month) + (0.004444628588451841 * country) + (-0.47042444828400876 * page 1 (main category)) + (-0.054593628007116216 *
 colour) + (-0.06527332401265237 * model photography) + (0.07627254278806014 * page) + (-0.8339717815755833 * price 2)
```

The scatter and the normal plot for the thousand test observations are seen. In the result we can see there is an overlap between the test and the predicted values .

**Figure 42**

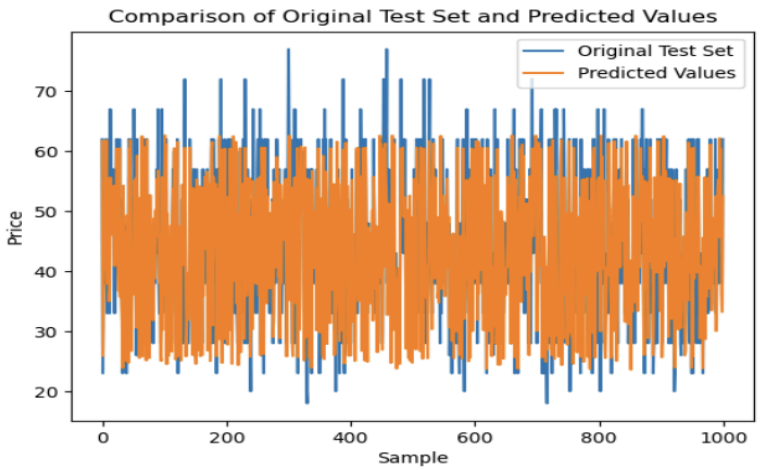
*Scatter plot for test versus predicted values*




For the normal plot the price target variable almost follows the same pattern which exists in the actual test set. We can see only for sparse observations the model failed to predict actual price value.

**Figure 43**

*Comparison of the actual versus predicted value*



We also calculated the confidence intervals for the coefficients of the regression model.

**Figure 44**

*Confidence Interval*

```
month                      0.002793  0.007881
country                    0.001883  0.007006
page 1 (main category) -0.473228 -0.467621
colour                 -0.057248 -0.051939
model photography      -0.067921 -0.062626
page                       0.073482  0.079063
price 2                -0.836567 -0.831376
```

**Figure 45**

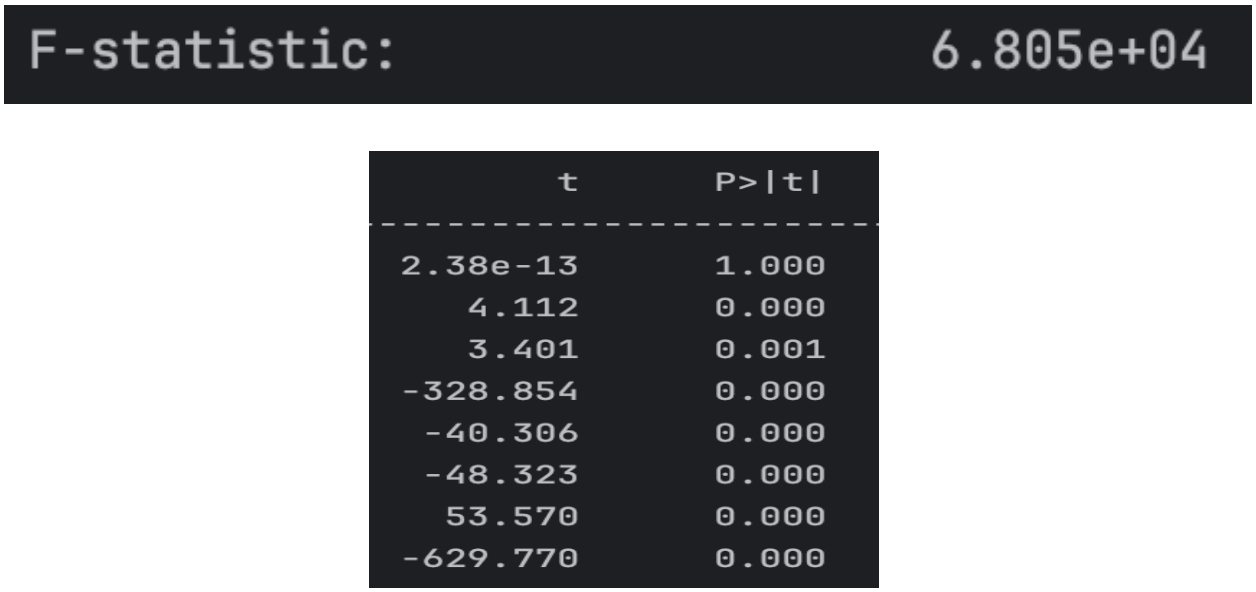
*Actual versus predicted values*

```
Comparison of Original Test Set and Predicted Values

   Price  Predicted_Price
0     62        61.695983
1     23        25.869013
2     43        35.729324
3     38        40.678778
4     43        40.669270
```

For the statistical analysis we will refer to figure 38. In the figure we are able to see f-statistic, t-statistic and p values of the final model.

**Figure 46**

*F-statistic, T-statistic, and P value of the final model*




A high f-statistic value confirms the usefulness and validity of the regression analysis in explaining the relationship between variables and making predictions. A high t-statistic value in regression analysis indicates that the coefficient of the corresponding predictor variable is statistically significant and meaningful in explaining the variation in the dependent variable. p-value of zero, it typically indicates that the corresponding coefficient is highly statistically significant.

At last we can see that the model has significant low mse which is .204 with the high value for R-squared and Adjusted R-squared.

**Figure 47**

*R-Squared, Adjusted R-Squared and MSE of the model*

```
The final model is:
    Rsquared  Rsquared-Adj            AIC            BIC       MSE
0  0.796543      0.796531  151586.563814  151664.237474  0.204058
```

## Phase 3 - Classification Analysis

The classification analysis has been done on the target variable price 2. Predicting the price 2 can help us in the following situations.

- Customer Behavior Understanding: Classifying Price 2 (higher or lower than the average price of the entire product category) can help in understanding customer preferences and behaviors. Knowing whether a product's price is perceived as higher or lower than average by customers can inform pricing strategies and product positioning.
- Segmentation and Targeting: By classifying products based on Price 2, businesses can segment their customer base more effectively. Products with higher perceived prices may appeal to a different customer segment compared to products with lower perceived prices. This segmentation can aid in targeted marketing and product positioning strategies.

**Decision Tree (Pre Pruning)**

We ran the decision tree classifier on the data to predict the target with the hyper parameters max depth, min samples split, min sample leaf, max features, splitter, and criterion. We used GridSearCV which helps us to pick the best parameter for the model.

```
clf = DecisionTreeClassifier(random_state=5805)
tuned_parameters = [{'max_depth': range(2,15,1),
                     'min_samples_split': range(2,10,1),
                     'min_samples_leaf': range(1,5,1),
                     'max_features': range(1,5,1),
                     'splitter': ['best', 'random'],
                     'criterion': ['gini', 'entropy', 'log_loss']}]

try:
    gridSearch = joblib.load('decision_tree_pre.pkl')
    print("\nLoaded checkpoints successfully")

except FileNotFoundError:
    print("No previous checkpoint found. Starting new training...")
    gridSearch = GridSearchCV(clf, tuned_parameters, cv=5, scoring='accuracy',verbose=2)
    gridSearch.fit(X_train,y_train)

print('\nBest parameters set found on development', gridSearch.best_params_)
best_estimator = gridSearch.best_estimator_
```

```
Best parameters set found on development {'criterion': 'gini', 'max_depth': 12, 'max_features': 3, 'min_samples_leaf': 1, 'min_samples_split': 4, 'splitter': 'best'}
```

The accuracy of our pre-pruned model is one. The accuracy for each fold is also calculated.

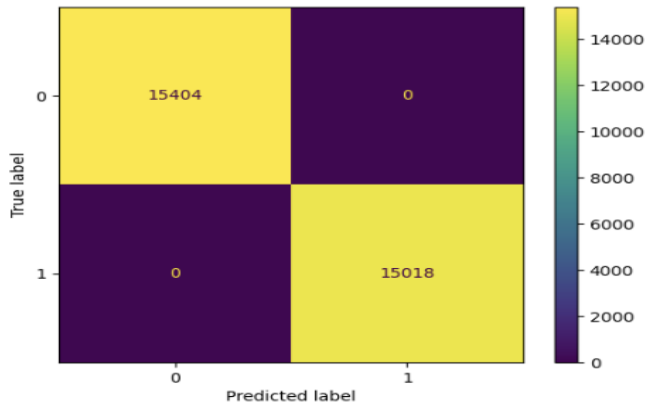
**Figure 48**

*Metrics of decision tree pre pruned model*

```
The accuracy of the pre-pruned model is:  1.0

The confusion matrix is:
 [[15404     0]
 [    0 15018]]

The precision of the pre-pruned model is:  1.0

The recall of the pre-pruned model 1.0

The specificity of the pre-pruned model is:  1.0

The f1 score of the pre-pruned model is:  1.0

The accuracy for each fold is:
Accuracy: 1.0
Accuracy: 0.9999671290513444
Accuracy: 0.9999342581026889
Accuracy: 0.998356452567221
Accuracy: 1.0
```

**Figure 49**

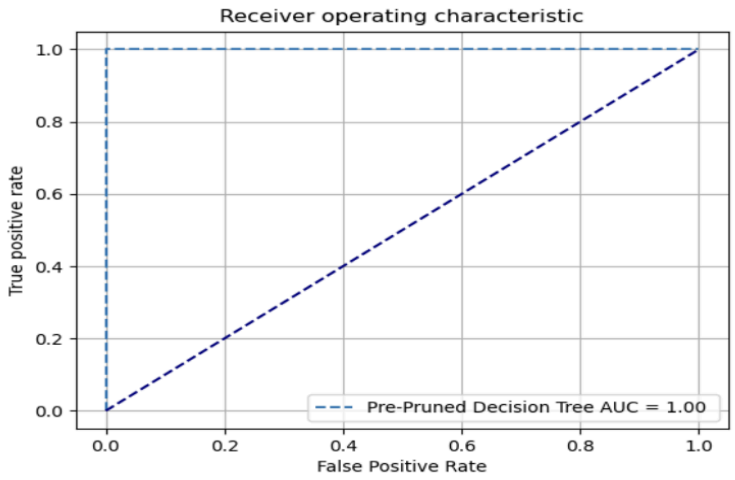
*Confusion Matrix (Pre Pruned Decision Tree)*



In the confusion matrix we don't have any false positive and false negative values.

**Figure 50**

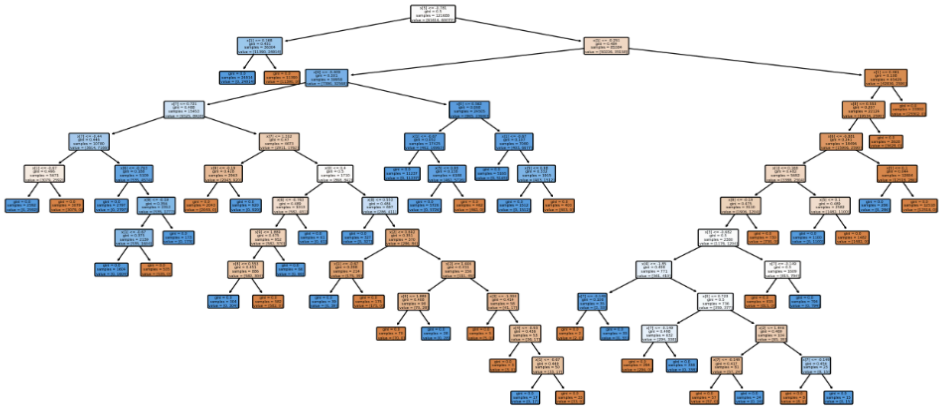
*ROC and AUC for the pre pruned decision tree*




The AUC of the pre pruned decision tree found to be 1.00. The final pre pruned decision tree is also plotted.

**Figure 51**

*Pre Pruned Decision Tree*



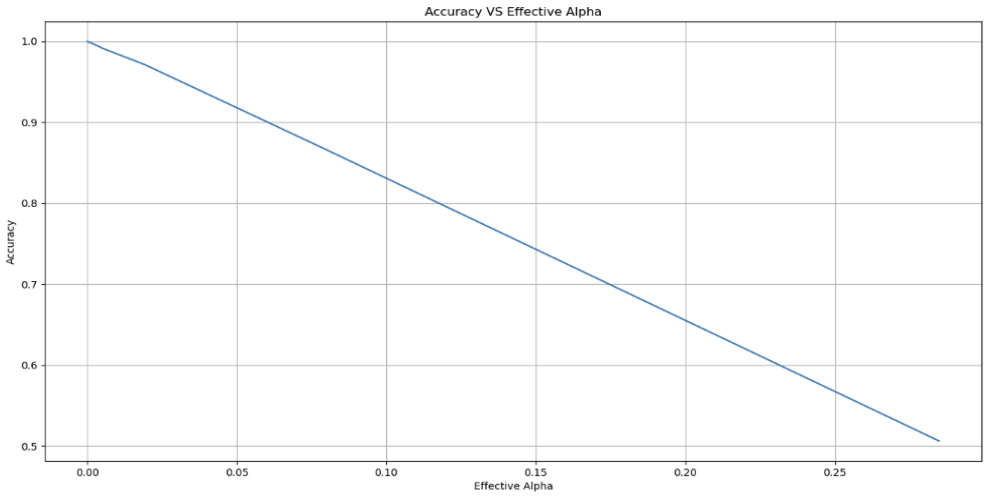
**Decision Tree (Post Pruning)**

In this approach we use cost complexity pruning path to determine the best value for alpha.

```
clf = DecisionTreeClassifier(random_state=5805)
path = clf.cost_complexity_pruning_path(X_train, y_train)
ccp_alphas, impurities = path.ccp_alphas, path.impurities

clfs = []
for alpha in ccp_alphas:
    clf = DecisionTreeClassifier(random_state=5805, ccp_alpha=alpha)
    clf.fit(X_train, y_train)
    clfs.append(clf)
```

**Figure 52**

*Accuracy versus Effective Alpha*




We found that the optimal value for the alpha to be zero.

```
Optimal Alpha:  0.0
```

**Figure 53**

*Metrics of decision tree post pruned model*

```
The accuracy of the post-pruned model is:  1.0

The confusion matrix is:
[[15404     0]
[    0 15018]]

The precision score of the post-pruned model is:  1.0

The recall score of the post-pruned model is:  1.0

The specificity of the post-pruned model is:  1.0

The f1 score of the post-pruned model is:  1.0

The accuracy for each fold is:
Accuracy: 1.0
Accuracy: 1.0
Accuracy: 1.0
Accuracy: 1.0
Accuracy: 1.0
```

We found that accuracy of the post pruned model to be one and k fold cross validation shows a consistent value of one as well.

**Figure 54**

*Confusion Matrix (Post Pruned Decision Tree)*

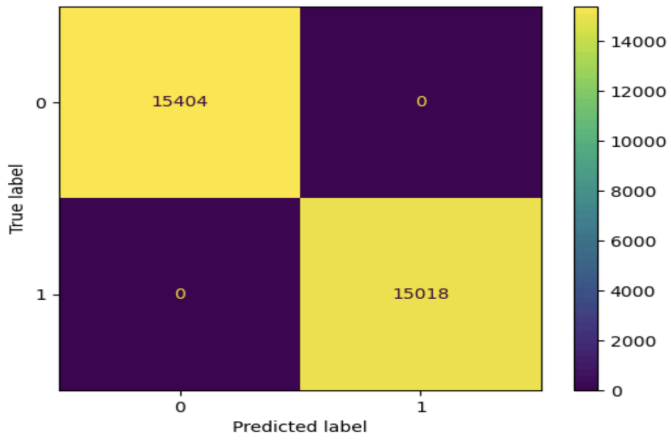
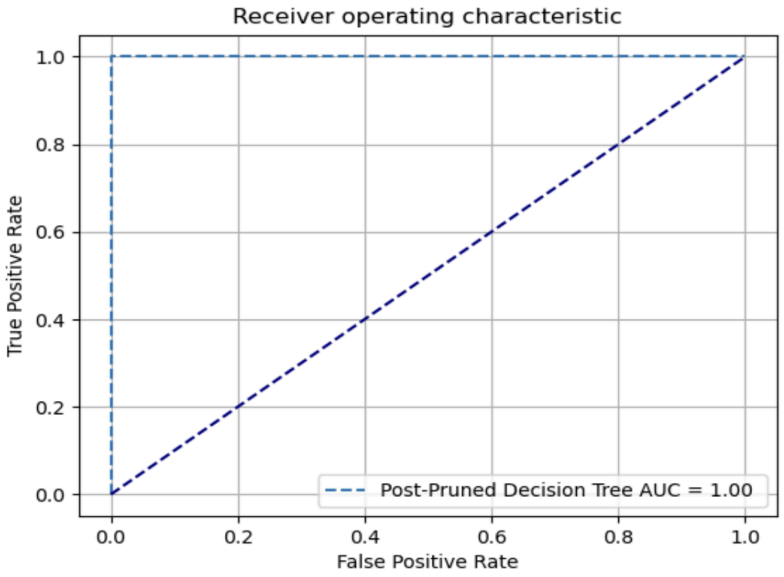


**Figure 55**

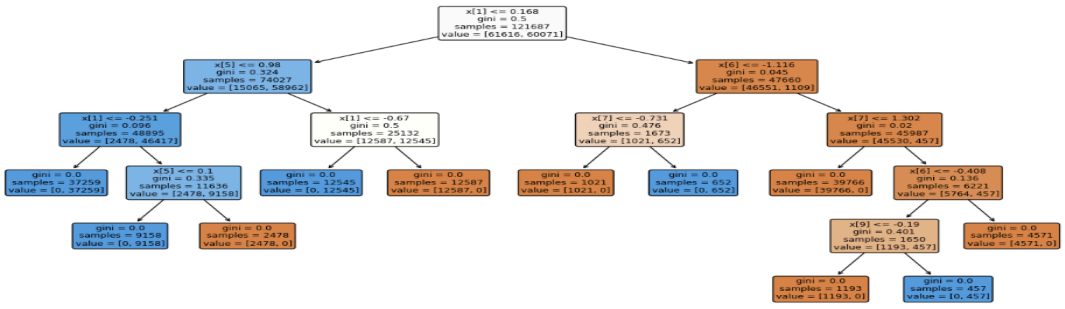
*ROC and AUC for the post pruned decision tree*




The AUC of the post pruned decision tree found to be 1.00. The final post pruned decision tree is also plotted.

**Figure 56**

*Post Pruned Decision Tree*



**Logistic Regression**

We ran the logistic regression on the data to predict the target with the hyper parameters C and penalty. We used GridSearCV which helps us to pick the best parameter for the model.

```
clf = LogisticRegression()

tuned_parameters = [{'C': [0.001, 0.01, 0.1, 1, 10, 100],
                     'penalty': ['l1', 'l2', 'elasticnet']
                    }]

gridSearch = GridSearchCV(clf, tuned_parameters, cv=5, scoring='accuracy')
gridSearch.fit(X_train,y_train)

print("Best Parameters for the Logistic Regression:", gridSearch.best_params_)
```

```
Best Parameters for the Logistic Regression: {'C': 100, 'penalty': 'l2'}
```

**Figure 57**

*Metrics for the logistic regression model*

```
The test accuracy of the Logistic Regression model:  0.9936559069094734

The confusion matrix is:
 [[15344    60]
 [  133 14885]]

The precision score of the Logistic Regression model:  0.9959852793576447

 The recall score of the Logistic Regression model:  0.9911439605806366

The specificity of the Logistic Regression model:  0.9961049078161517

The f1 Score of the Logistic Regression model:  0.9935587224243234

The accuracy for each fold is:
Accuracy: 0.993064229833673
Accuracy: 0.9933600683715732
Accuracy: 0.992439681809217
Accuracy: 0.993688777858129
Accuracy: 0.9931954899575951
```

We found that accuracy of the logistic regression model to be 0.9936 and k fold cross validation shows a consistent value of .993 as well.

**Figure 58**

*Confusion Matrix (Logistic Regression)*

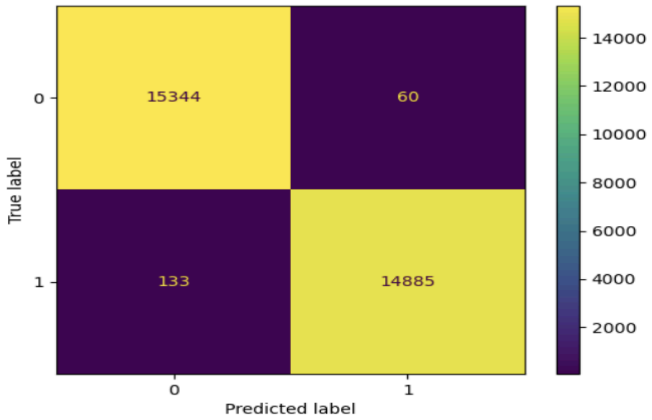
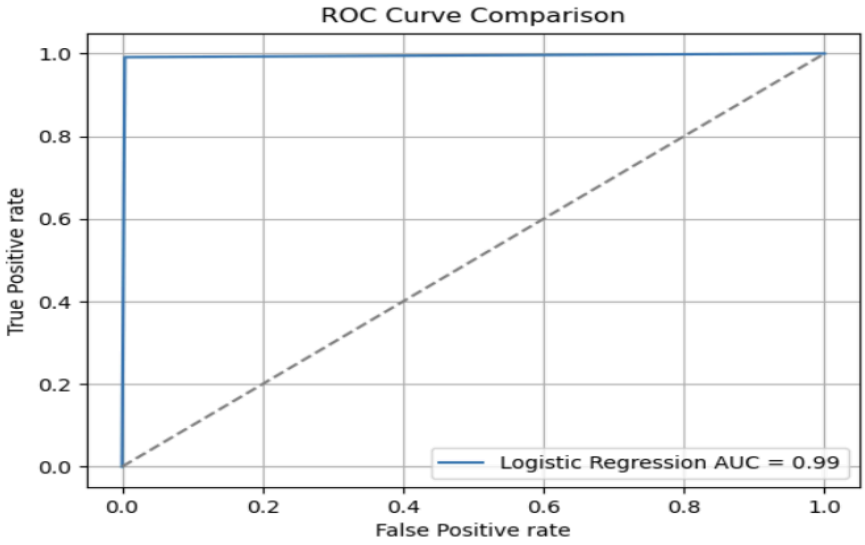



**Figure 59**

*ROC and AUC for the Logistic Regression*



## K-Nearest Neighbors

We ran the K-Nearest Neighbors on the data to predict the target with the hyper parameters n_neighbors, p, weights, and algorithm. We used GridSearchCV to find the best parameter for the model.

```
parameters = [{'n_neighbors':range(1,11,1),
               'p':[1,2],
               'weights' : ['uniform','distance'],
                'algorithm' : ['auto','ball_tree','kd_tree']}]
clf = KNeighborsClassifier()

grid_search = GridSearchCV(estimator=clf, param_grid=parameters, cv=5, scoring='accuracy',verbose=2)
grid_search.fit(X_train, y_train)
```

```
Best Parameters for the KNN classifier is:  {'algorithm': 'auto', 'n_neighbors': 9, 'p': 1, 'weights': 'distance'}
```

**Figure 60**

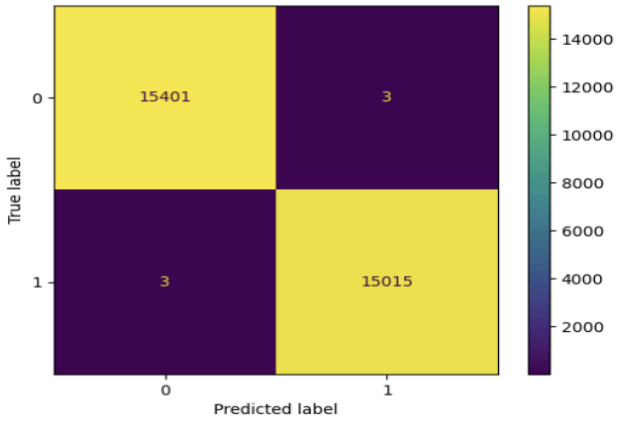
*Metrics of the KNN classifier*

```
The accuracy of the KNN classifier model is:  0.9998027743080665

The confusion matrix of the KNN classifier is :
 [[15401     3]
 [    3 15015]]

The precision of the KNN classifier is :  0.9998002397123452

The recall of the KNN classifier is :  0.9998002397123452

The specificity of the KNN classifier is :  0.9998052453908076

The f1 score of the KNN classifier is :  0.9998002397123452

The accuracy for each fold is:
Accuracy: 0.999769903359411
Accuracy: 0.9998027743080665
Accuracy: 0.9999342581026889
Accuracy: 0.9997041614620998
Accuracy: 0.9997041517372868
```

We found that accuracy of the KNN classifier to be 0.9980 and k fold cross validation shows a consistent value of .999 as well.

**Figure 61**

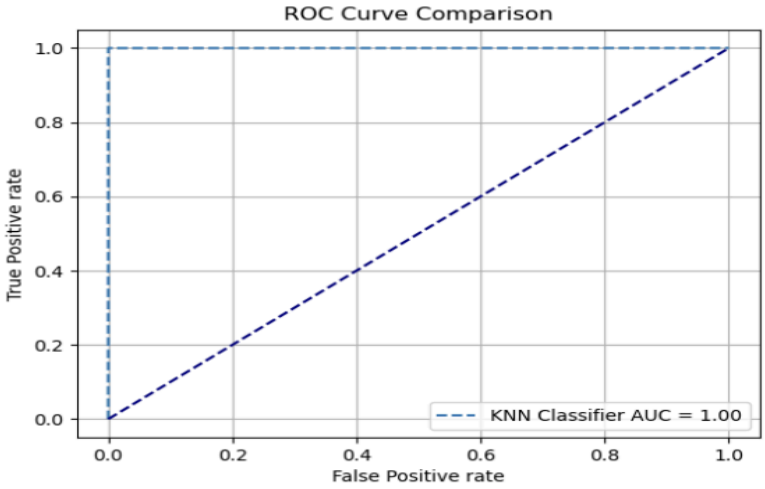
*Confusion Matrix (KNN Classifier)*




We noticed that the confusion matrix got the least number of false positive and false negative values.

**Figure 62**

*ROC and AUC for the KNN Classifier*



We noticed that AUC for the KNN classifier to be one which shows a good fit and the model is able to distinguish between two classes.

**Naive Bayes Classifier**

We trained the naive bayes classifier on the dataset. Here we didn't use GridSearchCV as the classifier doesn't take any parameter values.

```
clf = GaussianNB()
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
```

**Figure 63**

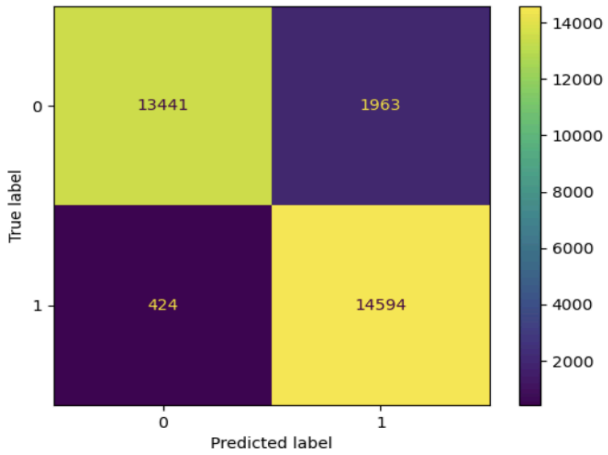
*Metrics of the Naive Bayes Classifier*

```
The test accuracy of the Naive Bayes model:  0.9215370455591348

The confusion matrix of the Naive Bayes:
 [[13441  1963]
 [  424 14594]]

The precision score of the Naive Bayes: 0.8814398743733768

The recall score of the Naive Bayes: 0.9717672126781196

The specificity of the Naive Bayes:  0.8725655673850948

The f1 score of the Naive Bayes:  0.9244022169437845

The accuracy for each fold is:
Accuracy: 0.9212412070212347
Accuracy: 0.9230162382486359
Accuracy: 0.9196305305371113
Accuracy: 0.9249227532706594
Accuracy: 0.9216988264685579
```

We found that accuracy of the naive bayes classifier to be 0.921 and k fold cross validation shows a consistent value of .92 as well.

**Figure 64**

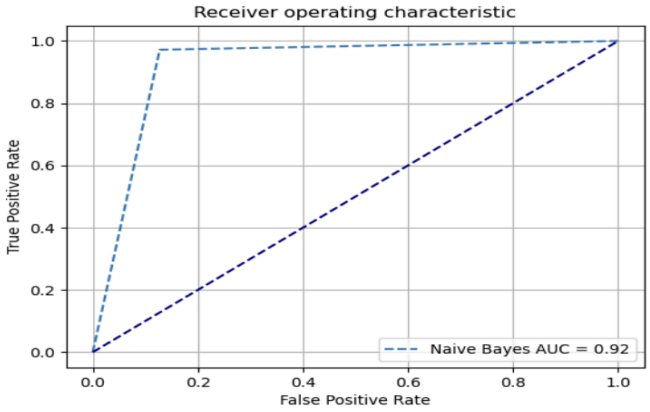
*Confusion Matrix (Naive Bayes Classifier)*




We noticed that the confusion matrix has more false positive and false negative values compared to other models.

**Figure 65**

*ROC and AUC for the Naive Bayes Classifier*



We noticed that AUC for the naive bayes classifier to be .92 which shows a good fit and the model is able to distinguish between two classes.

**Support Vector Machine (Linear)**

We trained SVM with the linear kernel on the dataset with the parameters C and gamma. We used GridSearchCV to find the best parameter for the model.

```
clf = SVC(kernel='linear', random_state=5805)

param_grid = {'C': [0.01, 0.1, 1, 10],
              'gamma': ['scale','auto']}
try:
    grid_search = joblib.load('svm_model.pkl')
    print("Loaded checkpoints successfully...")

except FileNotFoundError as e:
    print("No previous checkpoint found. Starting new training...")
    grid_search = GridSearchCV(clf, param_grid, scoring='accuracy', cv=5, verbose=2)
    grid_search.fit(X_train, y_train)
```

```
The best parameter for SVM (linear kernel) is:  {'C': 1, 'gamma': 'scale'}
```

**Figure 66**

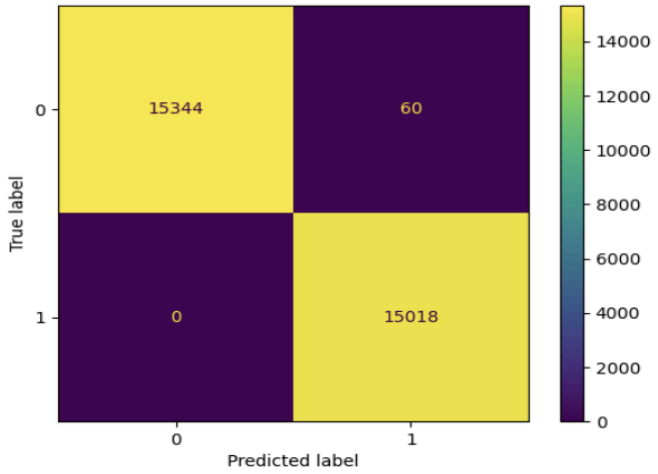
*Metrics of the Support Vector Machine (Linear)*

```
The best parameter for SVM (linear kernel) is:  {'C': 1, 'gamma': 'scale'}

The test accuracy of SVM (linear kernel) is:  0.9980277430806653

The confusion matrix of SVM (linear kernel) is:
 [[15344    60]
 [    0 15018]]

The precision of SVM (linear kernel) is:  0.9960206923995225

The recall of SVM (linear kernel) is:  1.0

The specificity of SVM (linear kernel) is:  0.9961049078161517

The f1 score of SVM (linear kernel) is:  0.998006379585327

The accuracy for each fold is:
Accuracy: 0.9980934849779765
Accuracy: 0.9982907106699099
Accuracy: 0.9983893235158766
Accuracy: 0.9981920978239432
Accuracy: 0.9983235265112915
```

We found that accuracy of the support vector machine (linear) classifier to be 0.998 and k fold cross validation shows a consistent value of .998 as well.

**Figure 67**

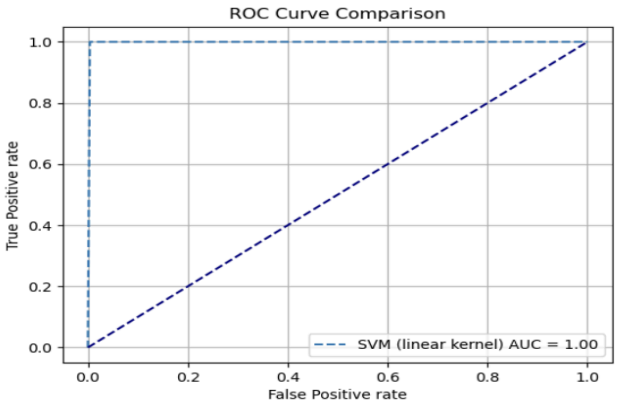
*Confusion Matrix (Support Vector Machine - Linear)*




We noticed that the confusion matrix got a few false positive and false negative values.

**Figure 68**

*ROC and AUC for the Support Vector Machine (Linear)*



We noticed that AUC for the Support Vector Machine (Linear) classifier to be one.

**Support Vector Machine (Polynomial)**

We trained SVM with the polynomial kernel on the dataset with the parameters C, degree, and gamma.

```
clf = SVC(kernel='poly', random_state=5805)

param_grid = {'C':[1, 10],
              'degree':[2, 3, 4],
              'gamma':['auto', 'scale']}

grid_search = GridSearchCV(clf, param_grid, scoring='accuracy', cv=5, verbose=2)
grid_search.fit(X_train, y_train)
```

```
The best parameter for SVM (polynomial kernel) is:  {'C': 10, 'degree': 3, 'gamma': 'auto'}
```

**Figure 69**

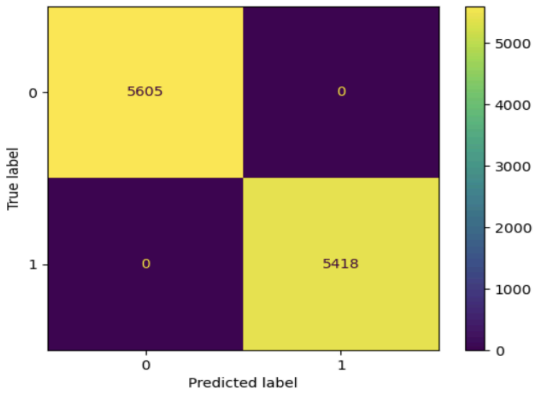
*Metrics of the Support Vector Machine (Polynomial)*

```
The test accuracy of SVM (polynomial kernel) is:  1.0

The confusion matrix of SVM (polynomial kernel) is:
 [[5605    0]
 [   0 5418]]

The precision of SVM (polynomial kernel) is:  1.0

The recall of SVM (polynomial kernel) is:  1.0

The specificity of SVM (polynomial kernel) is:  1.0

The f1 score of SVM (polynomial kernel) is:  1.0

The accuracy for each fold is:
Accuracy: 0.9999092805951193
Accuracy: 1.0
Accuracy: 1.0
Accuracy: 1.0
Accuracy: 1.0
```

We found that accuracy of the support vector machine (polynomial) classifier to be 1.0 and k fold cross validation shows almost a consistent value of  1.0 as well.

**Figure 70**

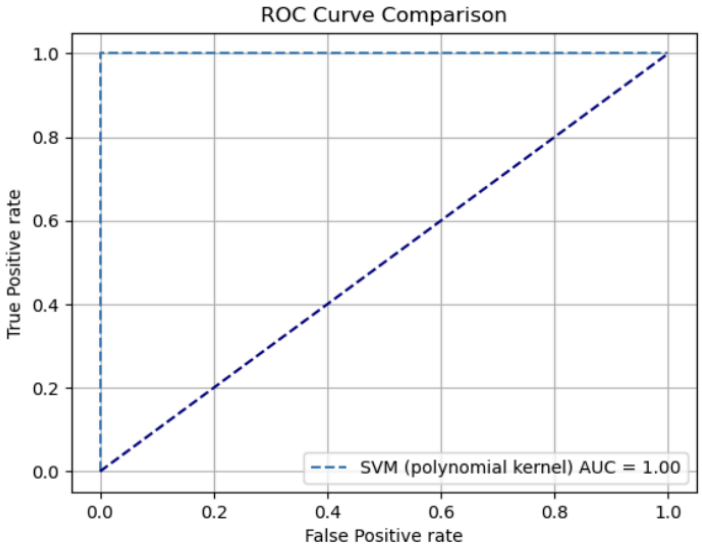
*Confusion Matrix (Support Vector Machine - Polynomial)*




We noticed that the confusion matrix got zero false positive and false negative values.

**Figure 71**

*ROC and AUC for the Support Vector Machine (Polynomial)*



We noticed that AUC for the Support Vector Machine (Polynomial) classifier to be one.

**Support Vector Machine (Radial Basis Function)**

We trained SVM with the RBF kernel on the dataset with the parameters C, and gamma.

```
clf = SVC(kernel='rbf', random_state=5805)

param_grid = {'C': [0.1, 1, 10],
              'gamma': ['scale', 'auto']}

grid_search = GridSearchCV(clf, param_grid, scoring='accuracy', cv=5, verbose=2)
grid_search.fit(X_train, y_train)
```

```
The best parameter for SVM (rbf kernel) is:  {'C': 10, 'gamma': 'scale'}
```

**Figure 72**

*Metrics of the Support Vector Machine (RBF)*

```
The test accuracy of SVM (rbf kernel) is:  1.0

The confusion matrix of SVM (rbf kernel) is:
 [[15404     0]
 [    0 15018]]

The precision of SVM (rbf kernel) is:  1.0

The recall of SVM (rbf kernel) is:  1.0

The specificity of SVM (rbf kernel) is:  1.0

The f1 score of SVM (rbf kernel) is:  1.0

The accuracy for each fold is:
Accuracy: 0.9999671290513444
Accuracy: 1.0
Accuracy: 1.0
Accuracy: 1.0
Accuracy: 1.0
```

We found that accuracy of the support vector machine (RBF) classifier to be 1.0 and k fold cross validation shows almost a consistent value of 1.0 as well.

**Figure 73**

*Confusion Matrix (Support Vector Machine - RBF)*

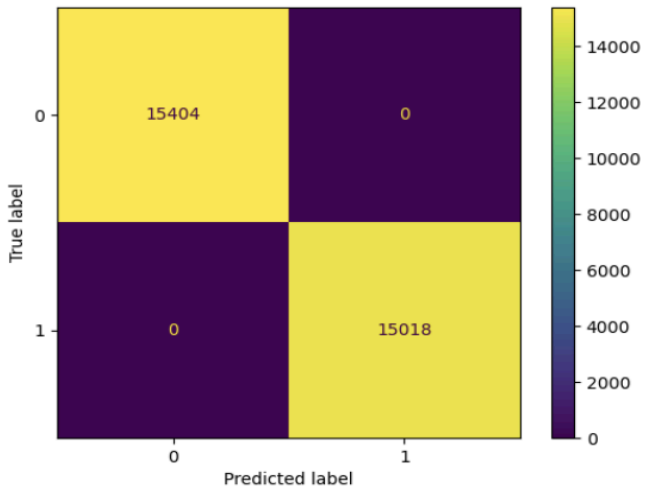
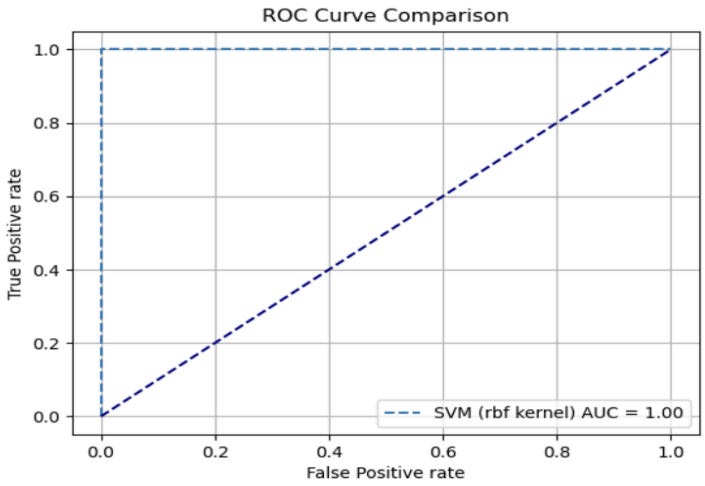



**Figure 74**

*ROC and AUC for the Support Vector Machine (RBF)*



We noticed that AUC for the Support Vector Machine (RBF) classifier to be one.

**Random Forest Classifier (Bagging)**

We ran a combined Bagging classifier and Random Forest classifier to build our model with the parameters n_estimators, max samples, and max features.

```
base_classifier = RandomForestClassifier(random_state=5805)

clf = BaggingClassifier(estimator=base_classifier, random_state=5805)

param_grid = {'n_estimators': [5, 10, 15],
              'max_samples': [0.5, 0.75, 0.9],
              'max_features': [2, 5, 8]
              }

try:
    grid_search = joblib.load('rf_bagging.pkl')
    print("\nLoaded checkpoints successfully")

except FileNotFoundError:
    print("No previous checkpoint found. Starting new training...")
    grid_search = GridSearchCV(clf, param_grid, scoring='accuracy', cv=5, verbose=2)
    grid_search.fit(X_train, y_train)
```

```
The best parameter for Random Forest (Bagging) is:  {'max_features': 8, 'max_samples': 0.5, 'n_estimators': 5}
```

**Figure 75**

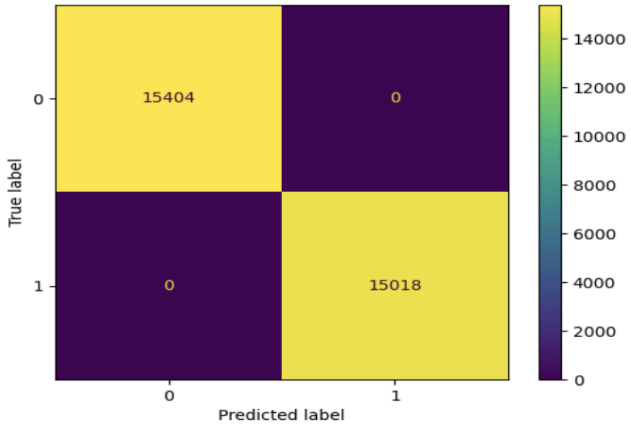
*Metrics for the Random Forest Classifier (Bagging)*

```
The test accuracy of Random Forest (Bagging) is:  1.0

The confusion matrix of Random Forest (Bagging) is:
 [[15404     0]
 [    0 15018]]

The precision of Random Forest (Bagging) is:  1.0

The recall of Random Forest (Bagging) is:  1.0

The specificity of Random Forest (Bagging) is:  1.0

The f1 score of Random Forest (Bagging) is:  1.0

The accuracy for each fold is:
Accuracy: 1.0
Accuracy: 1.0
Accuracy: 1.0
Accuracy: 1.0
Accuracy: 1.0
```

We found that accuracy of the random forest classifier (Bagging) classifier to be 1.0 and k fold cross validation shows a consistent value of 1.0 as well.

**Figure 76**

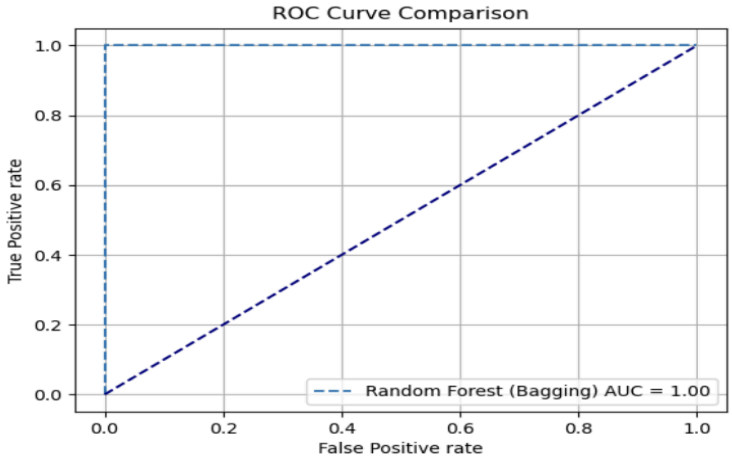
*Confusion Matrix (Random Forest - Bagging)*




Confusion matrix has zero false positive and false negative values.

**Figure 77**

*ROC and AUC for the Random Forest (Bagging)*



Random forest classifier (Bagging) has the perfect AUC score of one.

**Random Forest Classifier (Boosting)**

We ran a random forest classifier (boosting) using AdaBoost classifier to build our model with the parameters n_estimators, and learning rate.

```
base_estimator = RandomForestClassifier(random_state=5805)

clf = AdaBoostClassifier(random_state=5805, estimator=base_estimator)

param_grid = {'n_estimators': [5, 10, 15],
              'learning_rate': [0.1, 1, 10],
              'base_estimator__n_estimators': [5, 10, 15]}

try:
    grid_search = joblib.load('rf_boosting.pkl')
    print("\nLoaded checkpoints successfully")

except FileNotFoundError:
    print("No previous checkpoint found. Starting new training...")
    grid_search = GridSearchCV(clf, param_grid, scoring='accuracy', cv=5, verbose=2)
    grid_search.fit(X_train, y_train)
```

```
The best parameter for Random Forest (Boosting) is:  {'base_estimator__n_estimators': 5, 'learning_rate': 0.1, 'n_estimators': 5}
```

**Figure 78**

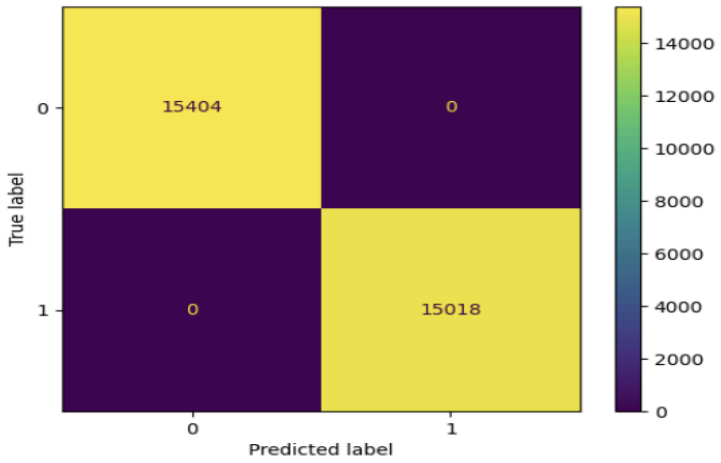
*Metrics for the Random Forest Classifier (Boosting)*

```
The test accuracy of Random Forest (Boosting) is:  1.0

The confusion matrix of Random Forest (Boosting) is:
 [[15404     0]
 [    0 15018]]

The precision of Random Forest (Boosting) is:  1.0

The recall of Random Forest (Boosting) is:  1.0

The specificity of Random Forest (Boosting) is:  1.0

The f1 score of Random Forest (Boosting) is:  1.0

The accuracy for each fold is:
Accuracy: 1.0
Accuracy: 1.0
Accuracy: 1.0
Accuracy: 1.0
Accuracy: 1.0
```

We found that accuracy of the random forest classifier (Boosting) classifier to be 1.0 and k fold cross validation showed a consistent value of 1.0 as well.

**Figure 79**

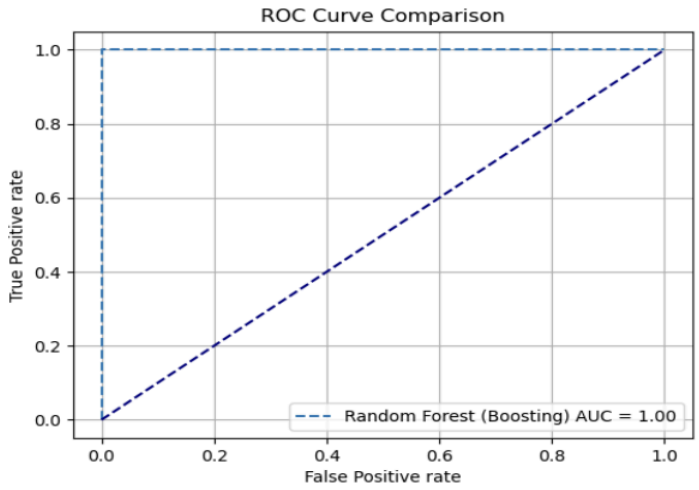
*Confusion Matrix (Random Forest - Boosting)*




Confusion matrix has zero false positive and false negative values.

**Figure 80**

*ROC and AUC for the Random Forest (Boosting)*



Random forest classifier (Boosting) has the perfect AUC score of one.

**Random Forest Classifier (Stacking)**

We ran a random forest classifier (stacking) using GaussianNB and Logistic Regression to build our model with the parameters n_estimators, max features, and max samples.

```
base_classifiers = [
    ('gnb', GaussianNB()),
    ('lr', LogisticRegression(random_state=5805))
]

clf = StackingClassifier(estimators=base_classifiers,final_estimator=RandomForestClassifier(random_state=5805))

param_grid = {
    'final_estimator__n_estimators': [5, 10, 15],
    'final_estimator__max_features': [2, 5, 8],
    'final_estimator__max_samples': [.7, .9]
}

try:
    grid_search = joblib.load('rf_stacking.pkl')
    print("\nLoaded checkpoints successfully")

except FileNotFoundError:
    print("No previous checkpoint found. Starting new training...")
    grid_search = GridSearchCV(clf, param_grid, scoring='accuracy', cv=5, verbose=2)
    grid_search.fit(X_train, y_train)
```

```
The best parameter for Random Forest (Stacking) is:  {'final_estimator__max_features': 2, 'final_estimator__max_samples': 0.7, 'final_estimator__n_estimators': 5}
```

**Figure 81**

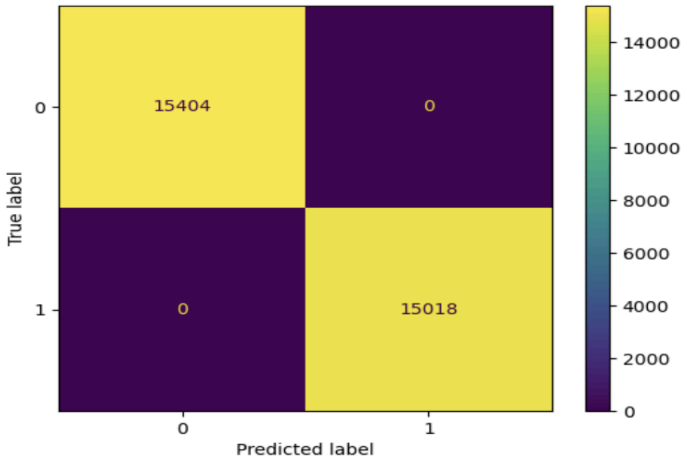
*Metrics for the Random Forest Classifier (Stacking)*

```
The test accuracy of Random Forest (Stacking) is:  1.0

The confusion matrix of Random Forest (Stacking) is:
 [[15404     0]
 [    0 15018]]

The precision of Random Forest (Stacking) is:  1.0

The recall of Random Forest (Stacking) is:  1.0

The specificity of Random Forest (Stacking) is:  1.0

The f1 score of Random Forest (Stacking) is:  1.0

The accuracy for each fold is:
Accuracy: 1.0
Accuracy: 0.9999671290513444
Accuracy: 1.0
Accuracy: 0.9999671290513444
Accuracy: 0.9999671279708097
```

We found that accuracy of the random forest classifier (Stacking) classifier to be 1.0 and k fold cross validation almost showed a consistent value of  1.0 as well.

**Figure 82**

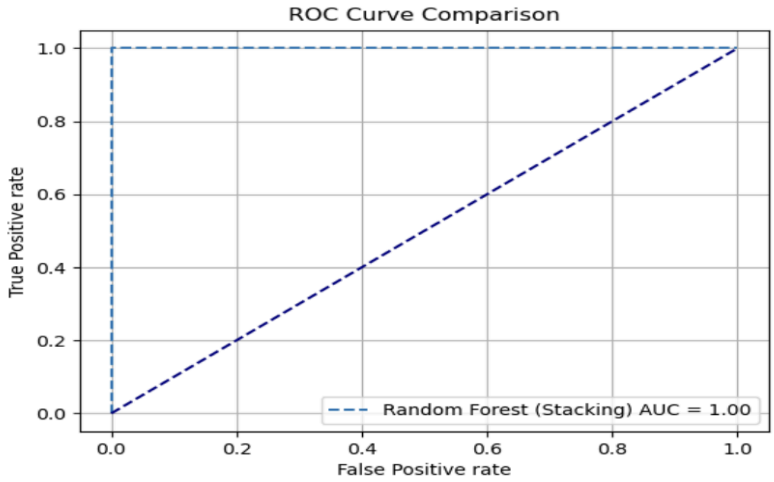
*Confusion Matrix (Random Forest - Stacking)*




Confusion matrix has zero false positive and false negative values.

**Figure 83**

*ROC and AUC for the Random Forest (Stacking)*



Random forest classifier (Stacking) has the perfect AUC score of one.

**Neural Networks - Multi Layer Perceptron**

We built a multi layer perceptron model with the parameters activation function and alpha.

```
clf = MLPClassifier(random_state=5805)

param_grid = {'activation': ['logistic', 'tanh', 'relu'],
              'alpha': [0.0001, 0.001, 0.01]}

grid_search = GridSearchCV(clf, param_grid, scoring='accuracy', cv=5, verbose=2)
grid_search.fit(X_train, y_train)

print("\nThe best parameter for MLP is: ", grid_search.best_params_)
best_model = grid_search.best_estimator_
```

```
The best parameter for MLP is:  {'activation': 'logistic', 'alpha': 0.0001}
```

**Figure 84**

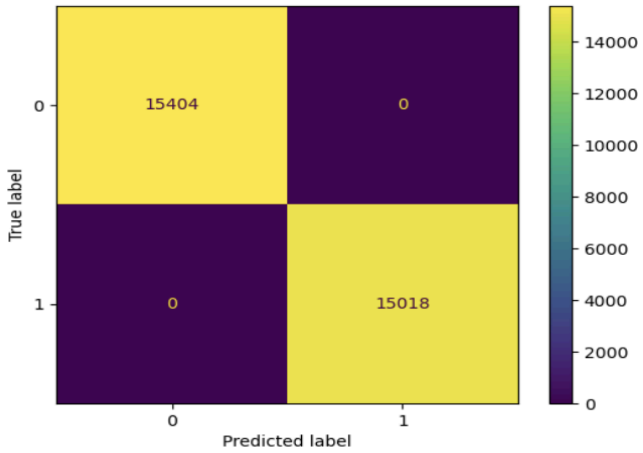
*Metrics - Multi Layer Perceptron*

```
The test accuracy of MLP is:  1.0

The confusion matrix of MLP is:
 [[15404     0]
 [    0 15018]]

The precision of MLP is:  1.0

The recall of MLP is:  1.0

The specificity of MLP is:  1.0

The f1 score of MLP is:  1.0

The accuracy for each fold is:
Accuracy: 1.0
Accuracy: 1.0
Accuracy: 1.0
Accuracy: 1.0
Accuracy: 1.0
```

We found that accuracy of the MLP classifier to be 1.0 and k fold cross validation almost showed a consistent value of  1.0 as well.

**Figure 85**

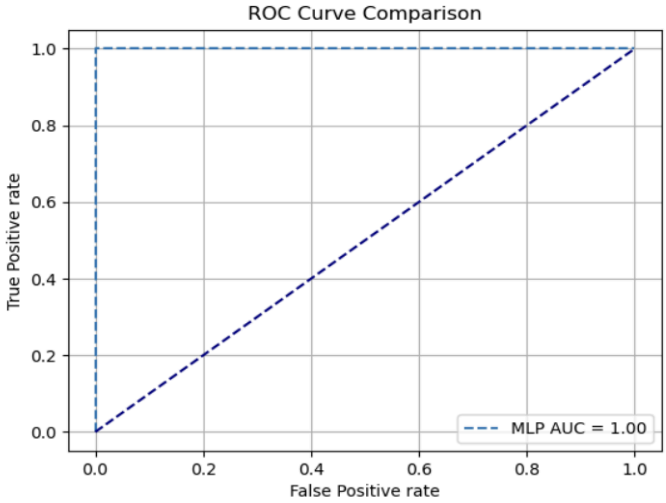
*Confusion Matrix (Multi Layer Perceptron)*




Confusion matrix has zero false positive and false negative values.

**Figure 86**

*ROC and AUC for the  MLP*



Multi Layer Perceptron has a perfect AUC score of one.

All the classifiers performed well on the given dataset. The accuracy, recall, precision, specificity, F1 score, ROC, and AUC were excellent for all the classifiers. The confusion matrix was plotted for all the classifiers with ROC curve. Every classifier is also validated through stratified K fold validation. The naive bayes classifier had the least value in the metrics. The below table summarizes the accuracy of all the classifiers in this phase.

**Table 1**

*Overview of Classifier Performance*

| **Classifier** | **Accuracy** |
|---|---|
| Decision Tree (Pre Pruned) | 1.0 |
| Decision Tree (Post Pruned) | 1.0 |
| Logistic Regression | 0.99 |
| K-Nearest Neighbors | 0.99 |
| Naive Bayes Classifier | 0.92 |
| Support Vector Machine (Linear) | 0.99 |
| Support Vector Machine (Polynomial) | 1.0 |
| Support Vector Machine (RBF) | 1.0 |
| Random Forest Classifier (Bagging) | 1.0 |
| Random Forest Classifier (Boosting) | 1.0 |
| Random Forest Classifier (Stacking) | 1.0 |
| Multi Layer Perceptron | 1.0 |

In the above figure we can notice that Pre Pruned Decision Tree, Post Pruned Decision Tree, Support Vector Machine (Polynomial), Support Vector Machine (RBF), Random Forest (Bagging, Boosting, and Stacking), and MLP have a perfect accuracy score of one. Ultimately, customers typically care about getting good value for their money. If "Price 2" is 1 ("yes"), this might be less appealing to customers who are price-conscious or looking for budget-friendly options. Conversely, if "Price 2" is 2 ("no"), this can be attractive to customers seeking bargains or discounts. For the retailers, products marked with 1 ("yes") might be perceived as premium or high-end offerings, potentially targeting customers willing to pay more for quality or unique features. Products labeled with 2 ("no") could be seen as budget-friendly options aimed at price-sensitive consumers. Shop owners might use this information to attract more cost-conscious shoppers and compete effectively in the market.

At last after taking everything into consideration we recommend Decision Tree (Post Pruned), Random Forest (Boosting), Random Forest (Stacking), or MLP can be used for the given task.

## Phase 4 - Clustering and Association Rule Mining

In this phase we use unsupervised machine learning algorithms which is a category of methods used to uncover patterns, structures, or relationships in data without explicit supervision or labeled outcomes.

### K-Means Clustering

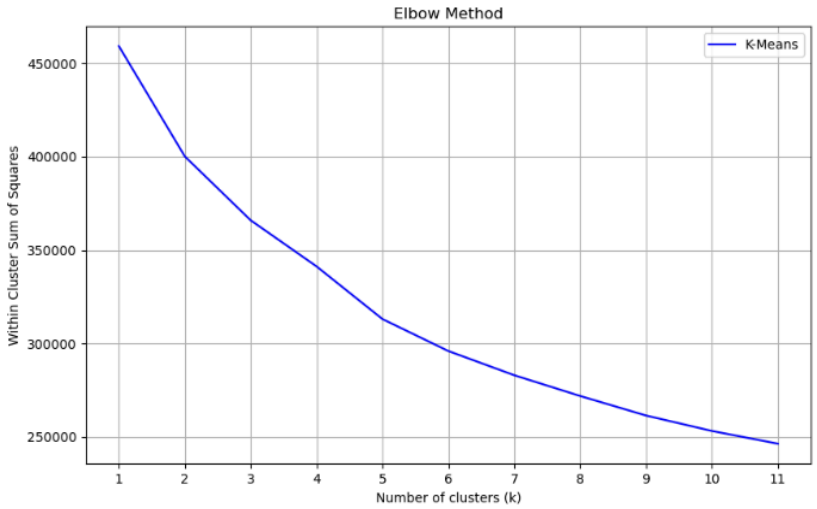
K-means clustering can provide valuable insights into the structure of your data, uncover hidden patterns, and facilitate data-driven decision-making in areas such as customer segmentation, market analysis, and personalized marketing strategies.

```
wcss = []

for k in range(1,11):
    model = KMeans(n_clusters=k, random_state=5805, verbose=2)
    model.fit(X)
    wcss.append(model.inertia_)
```

At first we apply the elbow method that involves plotting the within-cluster sum of squares (WCSS) against different values of K (number of clusters). WCSS represents the sum of squared distances between each point and its assigned cluster centroid.

**Figure 87**

*Elbow Method*



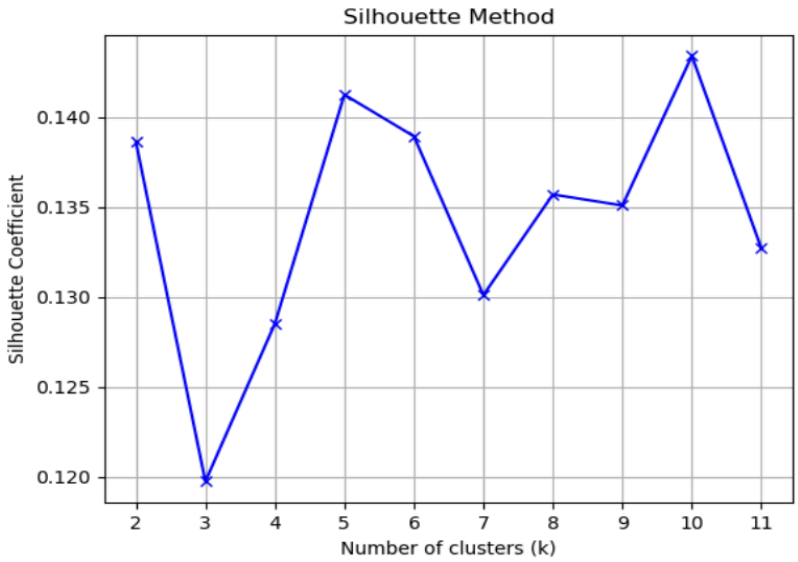

The plot displays the WCSS values against the number of clusters (K), allowing us to identify the "elbow" point where the rate of decrease in WCSS starts to slow down significantly. We can notice that the elbow point in the above graph is K equals 10.

```
slc = []
for k in range(2, 12):
    kmeans = KMeans(n_clusters=k, random_state=5805, verbose=2).fit(X)
    score = silhouette_score(X, kmeans.labels_, random_state=5805)
    slc.append(score)
```

The Silhouette Method evaluates clustering quality using the average silhouette score across data points for different values of K.

**Figure 88**

*Silhouette Method*




The plot shows the silhouette scores against the number of clusters (K), helping us identify the K value that maximizes clustering quality. We can see that the Silhouette coefficient is maximum when K equals 10.

Based on the findings that both the Elbow Method and Silhouette Method suggest 10 clusters for our dataset, the dataset likely exhibits distinct patterns or behaviors that can be categorized into 10 meaningful customer segments. Each cluster represents a group of customers with similar characteristics, preferences, or shopping behaviors.

**Apriori Algorithm**

The Apriori algorithm aims to identify frequent itemsets in a transaction database, where an itemset refers to a collection of items that appear together in transactions.

```
transactions = X.values.tolist()
trans = TransactionEncoder()
trans_array = trans.fit(transactions).transform(transactions)

encoded_df = pd.DataFrame(trans_array, columns=trans.columns_)
print(encoded_df)
print("\n")

data = apriori(encoded_df, min_support = 0.2, use_colnames = True, verbose = 1)
print(data)
print("\n")

data_ar = association_rules(data, metric = "confidence", min_threshold = 0.6)
data_ar = data_ar.sort_values(['confidence','lift'], ascending = [False, False])
print(data_ar)
print("\n")
```

In this the dataset is prepared for association rule mining by converting it into a binary-encoded format suitable for Apriori analysis. The result of the transformation is as follows.

**Figure 89**

*Result of encoding*

```
        April  August  Australia  Austria  Belgium  BritishVirginIslands  \
0        True   False      False    False    False                 False
1        True   False      False    False    False                 False
2        True   False      False    False    False                 False
3        True   False      False    False    False                 False
4        True   False      False    False    False                 False
...       ...     ...        ...      ...      ...                   ...
165469  False    True      False    False    False                 False
165470  False    True      False    False    False                 False
165471  False    True      False    False    False                 False
165472  False    True      False    False    False                 False
165473  False    True      False    False    False                 False

        CaymanIslands  ...  top-in-the-middle  top-left  top-right  trousers  \
0               False  ...              False     False      False      True
1               False  ...              False     False      False      True
2               False  ...               True     False      False     False
3               False  ...              False     False      False     False
4               False  ...              False     False       True     False
...               ...  ...                ...       ...        ...       ...
165469          False  ...              False     False      False     False
165470          False  ...              False     False      False      True
165471          False  ...              False      True      False      True
165472          False  ...              False      True      False     False
165473          False  ...              False      True      False     False
```

```
        unidentified  violet  white
0              False   False  False
1              False   False  False
2              False   False  False
3              False   False  False
4              False   False  False
...              ...     ...    ...
165469         False   False  False
165470         False   False  False
165471         False   False  False
165472         False   False  False
165473         False   False  False

[165474 rows x 83 columns]
```

Apriori algorithm applied on the encoded data with the min support 0.2.

**Figure 90**

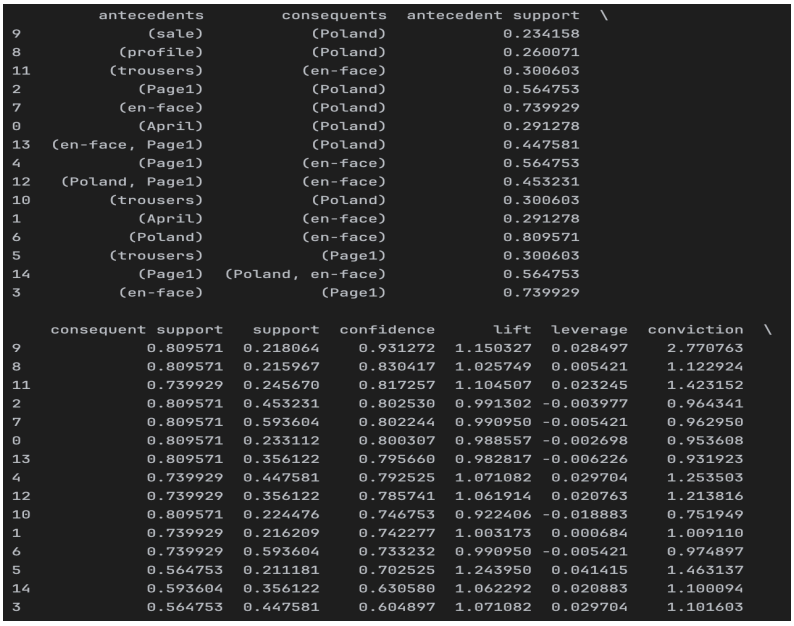
*Result of Apriori Algorithm*

| | support | itemsets |
|---|---|---|
| 0 | 0.291278 | (April) |
| 1 | 0.212910 | (July) |
| 2 | 0.215466 | (May) |
| 3 | 0.564753 | (Page1) |
| 4 | 0.247997 | (Page2) |
| 5 | 0.809571 | (Poland) |
| 6 | 0.233130 | (blouses) |
| 7 | 0.739929 | (en-face) |
| 8 | 0.260071 | (profile) |
| 9 | 0.234158 | (sale) |
| 10 | 0.232109 | (skirts) |
| 11 | 0.201742 | (top-in-the-middle) |
| 12 | 0.208685 | (top-left) |
| 13 | 0.300603 | (trousers) |
| 14 | 0.233112 | (Poland, April) |
| 15 | 0.216209 | (en-face, April) |
| 16 | 0.453231 | (Poland, Page1) |

Using the frequent itemsets, association rules are generated based on the specified confidence threshold.

**Figure 91**

*Result of Association Rule Mining*



| | antecedents | consequents | antecedent support | consequent support | support | confidence | lift | leverage | conviction |
|---|---|---|---|---|---|---|---|---|---|
| 9 | (sale) | (Poland) | 0.234158 | 0.809571 | 0.218064 | 0.931272 | 1.150327 | 0.028497 | 2.770763 |
| 8 | (profile) | (Poland) | 0.260071 | 0.809571 | 0.215967 | 0.830417 | 1.025749 | 0.005421 | 1.122924 |
| 11 | (trousers) | (en-face) | 0.300603 | 0.739929 | 0.245670 | 0.817257 | 1.104507 | 0.023245 | 1.423152 |
| 2 | (Page1) | (Poland) | 0.564753 | 0.809571 | 0.453231 | 0.802530 | 0.991302 | -0.003977 | 0.964341 |
| 7 | (en-face) | (Poland) | 0.739929 | 0.809571 | 0.593604 | 0.802244 | 0.990950 | -0.005421 | 0.962950 |
| 0 | (April) | (Poland) | 0.291278 | 0.809571 | 0.233112 | 0.800307 | 0.988557 | -0.002698 | 0.953608 |
| 13 | (en-face, Page1) | (Poland) | 0.447581 | 0.809571 | 0.356122 | 0.795660 | 0.982817 | -0.006226 | 0.931923 |
| 4 | (Page1) | (en-face) | 0.564753 | 0.739929 | 0.447581 | 0.792525 | 1.071082 | 0.029704 | 1.253503 |
| 12 | (Poland, Page1) | (en-face) | 0.453231 | 0.739929 | 0.356122 | 0.785741 | 1.061914 | 0.020763 | 1.213816 |
| 10 | (trousers) | (Poland) | 0.300603 | 0.809571 | 0.224476 | 0.746753 | 0.922406 | -0.018883 | 0.751949 |
| 1 | (April) | (en-face) | 0.291278 | 0.739929 | 0.216209 | 0.742277 | 1.003173 | 0.000684 | 1.009110 |
| 6 | (Poland) | (en-face) | 0.809571 | 0.739929 | 0.593604 | 0.733232 | 0.990950 | -0.005421 | 0.974897 |
| 5 | (trousers) | (Page1) | 0.300603 | 0.564753 | 0.211181 | 0.702525 | 1.243950 | 0.041415 | 1.463137 |
| 14 | (Page1) | (Poland, en-face) | 0.564753 | 0.593604 | 0.356122 | 0.630580 | 1.062292 | 0.020883 | 1.100094 |
| 3 | (en-face) | (Page1) | 0.739929 | 0.564753 | 0.447581 | 0.604897 | 1.071082 | 0.029704 | 1.101603 |

We mine association rules from a dataset using a specified confidence threshold (0.6) and the resulting association rules are then sorted based on confidence and lift values in descending order.

## Recommendations

We had many key learnings across the phases. In the first phase we gained insights into our dataset's structure, relationships, and distributions and learned best practices for cleaning, transforming, and encoding data. In the second and third phase we understood the strengths and weaknesses of various regression and classification algorithms. We also identified which features are most relevant for prediction. In the end of the third phase we used various metrics to gauge model performance and choose the most suitable model for our dataset. In the last phase we gained experience in visualizing data and model results to communicate findings effectively.

We found that day, month, country, order, price, page one, color, location, model photography and page were the features associated with the target variable.

The Decision Tree (post pruned), Random Forest (Boosting and Stacking) or MLP can be used on this dataset as these classifiers were best at predicting the price2 of the given product as they had the best metric values.

We can improve the performance of the classification by exploring more sophisticated feature engineering techniques to extract additional insights from the dataset. This could involve creating new features based on existing ones, such as interaction terms between different categorical variables.

We performed K-means clustering using both the elbow method and silhouette method, and both methods suggested that the optimal number of clusters is k = 10. This analysis uncovered valuable insights into the structure and relationships present in our dataset.

# Appendix

Backward Stepwise Regression

```
def backward_stepwise_regression(X, y, threshold=0.01):
    num_features = X.shape[1]
    included_features = list(range(num_features))
    results = []

    while True:
        model = sm.OLS(y, sm.add_constant(X.iloc[:, included_features])).fit()
        p_values = model.pvalues[1:]
        print(model.summary())
        max_p_value = p_values.max()
        if max_p_value > threshold:
            excluded_feature_idx = p_values.idxmax()
            idx = X.columns.to_list().index(excluded_feature_idx)
            included_features.remove(idx)
            results.append((excluded_feature_idx, max_p_value, model.aic, model.bic, model.rsquared_adj))
        else:
            break

    return included_features, results, model


selected_features, results, model = backward_stepwise_regression(standardized_train, standardized_y_train)
```

Stratified K-fold cross validation

```
skf = StratifiedKFold(n_splits=5, shuffle=True, random_state=5805)
print("\nThe accuracy for each fold is: ")
for train_index, test_index in skf.split(X_CV, y):
    X_train_k, X_test_k = X_CV.iloc[train_index], X_CV.iloc[test_index]
    y_train_k, y_test_k = y.iloc[train_index], y.iloc[test_index]
    clf.fit(X_train_k, y_train_k)
    accuracy = clf.score(X_test_k, y_test_k)
    print("Accuracy:", accuracy)
```

## References

Clickstream Data for Online Shopping Retrieved from https://archive.ics.uci.edu/dataset/553/clickstream+data+for+online+shopping

Avcontentteam. (2023, December 19) How to Perform Label Encoding in Python? Retrieved from https://www.analyticsvidhya.com/blog/2023/07/label-encoding-in-python/

Krish Naik. (2019) Finding an outlier in a dataset using Python Retrieved from https://www.youtube.com/watch?v=rzR_cKnkD18&t=10s

Emmanuella Budu Bagging, Boosting, and Stacking in Machine Learning Retrieved from https://www.baeldung.com/cs/bagging-boosting-stacking-ml-ensemble-models#:~:text=Bagging%20trains%20multiple%20weak%20models,to%20obtain%20a%20meta%2Dmodel.

Ankita Banerji. K-Mean: Getting the Optimal Number of Clusters Retrieved from https://www.analyticsvidhya.com/blog/2021/05/k-mean-getting-the-optimal-number-of-clusters/

Apriori: Frequent itemsets via the Apriori algorithm Retrieved from https://rasbt.github.io/mlxtend/user_guide/frequent_patterns/apriori/