Customer Behavioural Analysis and Purchase Prediction Model

Problem Statement

To understand and predict customer behaviour using behavioural signals from customers on an e-commerce platform. Each row in the dataset represents a single user session, including details like time spent on the site, pages viewed, and ad interactions.

The objective is to predict whether a user made a purchase (1) or not (0) based on their behaviour during the session.

Exploratory Data Analysis

The training data set contains 1800 individual data points (user sessions) and 8 columns (7 features + 1 target).

```
train.shape
```

```
(1800, 8)
```

```
train.sample(5)
```

	Time_on_site	Pages_viewed	Clicked_ad	Cart_value	Referral	Browser_Refresh_Rate	Last_Ad_Seen	Purchase
1296	8.93	24.68	0	89.96	Facebook	94.29	A	1
1067	5.86	9.11	0	44.73	Facebook	192.97	A	0
1202	8.33	25.72	1	10.00	Facebook	152.34	A	1
833	2.08	3.36	0	38.85	Facebook	60.33	A	0
1096	6.53	18.97	1	63.66	Facebook	86.86	D	1

There are no missing or duplicate data points.

```
train.isnull().sum()
```

```
Time_on_site           0
Pages_viewed           0
Clicked_ad             0
Cart_value             0
Referral               0
Browser_Refresh_Rate   0
Last_Ad_Seen           0
Purchase               0
dtype: int64
```

```
train.duplicated().sum()
```

```
np.int64(0)
```

There are 6 numerical features out which 4 are float type and 2 are integer type. There are 2 string features.

```
train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1800 entries, 0 to 1799
Data columns (total 8 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Time_on_site          1800 non-null   float64
 1   Pages_viewed          1800 non-null   float64
 2   Clicked_ad            1800 non-null   int64
 3   Cart_value            1800 non-null   float64
 4   Referral              1800 non-null   object
 5   Browser_Refresh_Rate  1800 non-null   float64
 6   Last_Ad_Seen          1800 non-null   object
 7   Purchase              1800 non-null   int64
dtypes: float64(4), int64(2), object(2)
memory usage: 112.6+ KB
```

We can infer using train.nunique() that there are 4 numerical columns and 4 categorical columns.

```
train.nunique()
```

```
Time_on_site             739
Pages_viewed            1239
Clicked_ad                 2
Cart_value              1353
Referral                   4
Browser_Refresh_Rate    1735
Last_Ad_Seen               4
Purchase                   2
dtype: int64
```

We check for correlation between the different feature columns and our target variable column. Since the correlation between Browser_Refresh_Rate and Purchase is very low (magnitude less than 0.05) we can say that Browser_Refresh_Rate does not affect the decision of whether the user decides to purchase or not. Note that Time_on_site and Pages_viewed have high correlation.

```
train.corr(numeric_only=True)
```

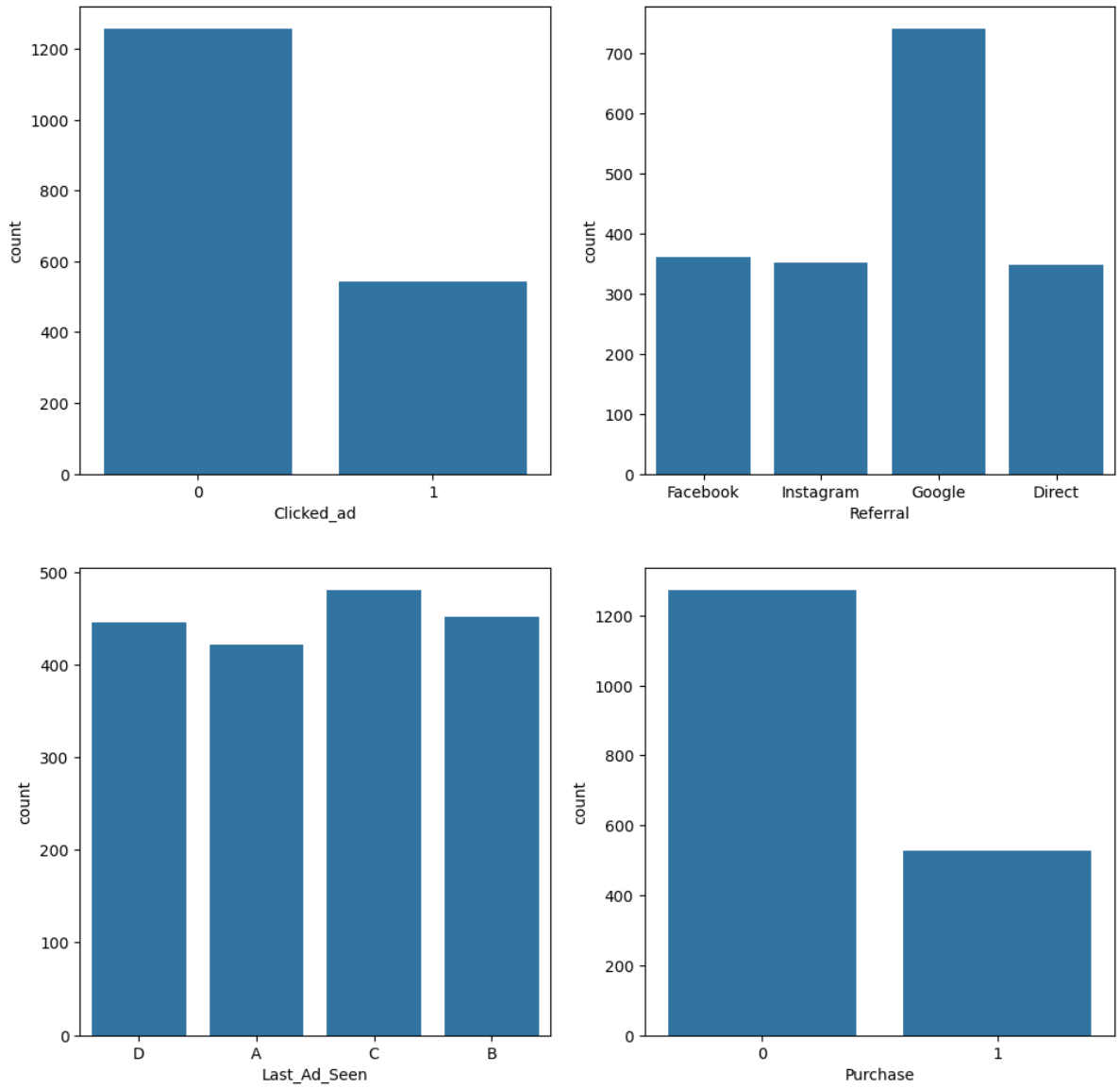
	Time_on_site	Pages_viewed	Clicked_ad	Cart_value	Browser_Refresh_Rate	Purchase
Time_on_site	1.000000	0.906873	-0.019618	0.021560	0.011260	0.344128
Pages_viewed	0.906873	1.000000	-0.011645	0.005392	0.019562	0.358996
Clicked_ad	-0.019618	-0.011645	1.000000	0.048734	0.008641	0.283907
Cart_value	0.021560	0.005392	0.048734	1.000000	-0.042083	0.137851
Browser_Refresh_Rate	0.011260	0.019562	0.008641	-0.042083	1.000000	-0.005686
Purchase	0.344128	0.358996	0.283907	0.137851	-0.005686	1.000000

Univariate Analysis

We have four categorical features:

1. Clicked_ad
2. Referral
3. Last_Ad_Seen
4. Purchase

We create their bar plots.

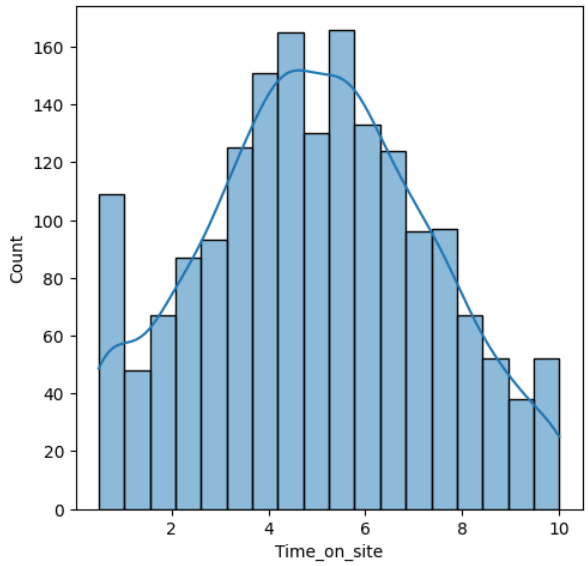


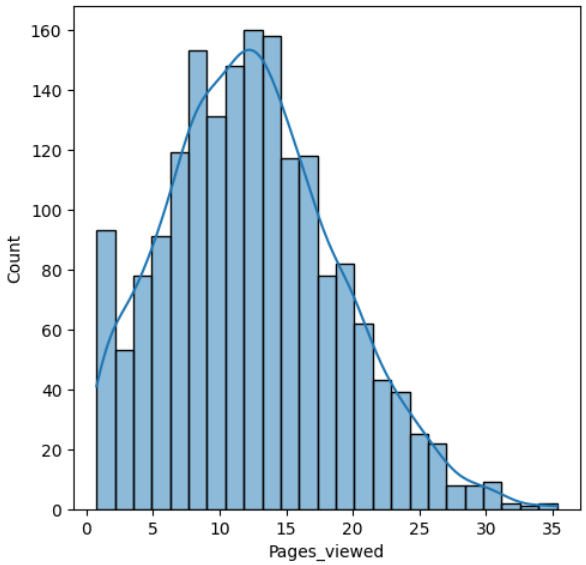
It is important to note that the target variable (Purchase) is highly imbalanced. The model might favour predicting “0” as it is the major class. This might affect the scoring metrics.

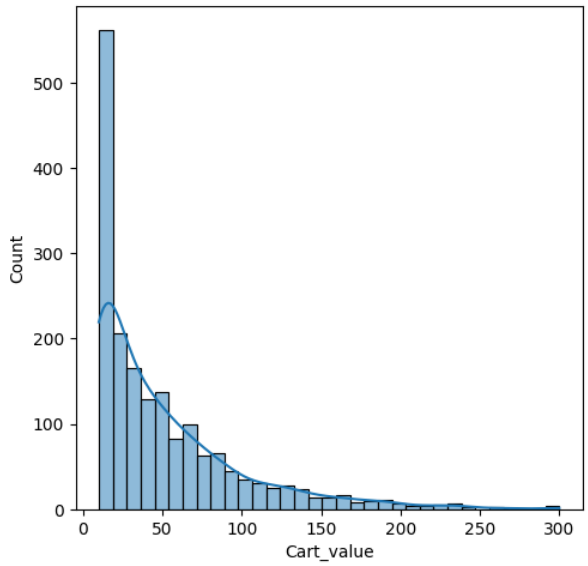
We have three numerical features:

1. Time_on_site
2. Pages_viewed
3. Cart_value

Their Histograms:







The Cart_value (skew = 1.780) has a highly right skewed distribution as compared to Time_on_site (skew = 0.031) and Pages_viewed (skew = 0.375) which are fairly normally distributed.

Therefore, we need to apply a log transformation on Cart_value but before that we will check for outliers as removing outliers may hinder our transformation.

We detect outliers using the IQR Method:

1) For a dataset we arrange it in ascending order and find the Q1 (25^{th} percentile) and Q3 (75^{th} percentile) points.

2) We calculate the IQR (Interquartile Range):

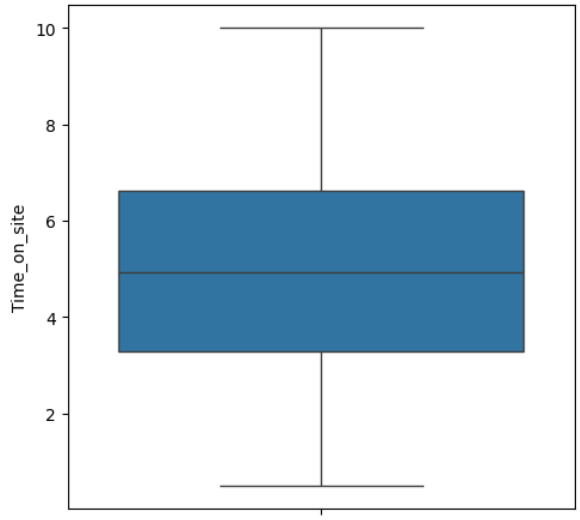
$$IQR = Q3 - Q1$$

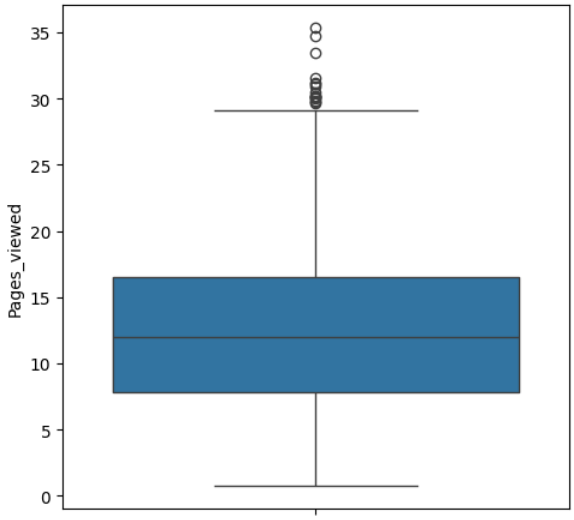
3) We define the upper and lower limits:

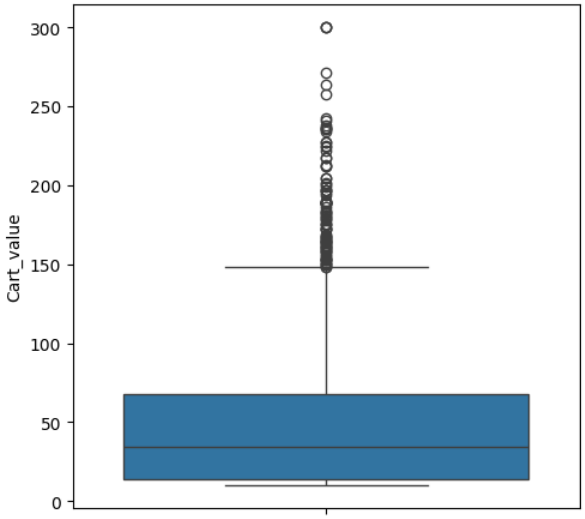
$$Upper\ Limit\ =\ Q3\ +1.5\ \times\ IQR$$
$$Lower\ Limit\ =\ Q1\ -\ 1.5\ \times\ IQR$$

4) Any data point below the lower limit or above the upper limit is then considered an outlier.

Using The IQR method we detect the outliers for Time_on_site, Pages_viewed and Cart_value.

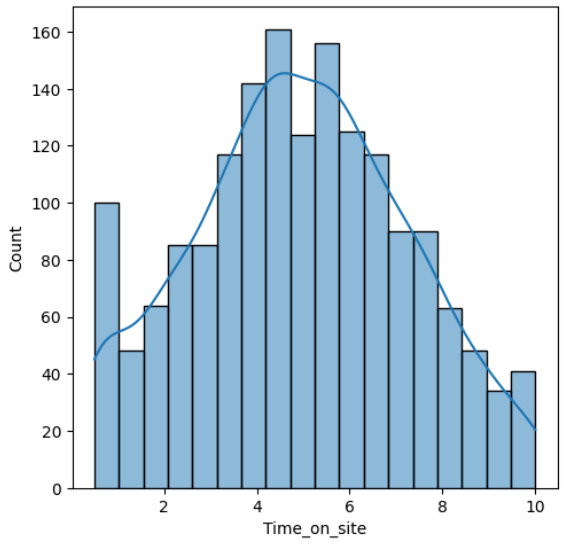


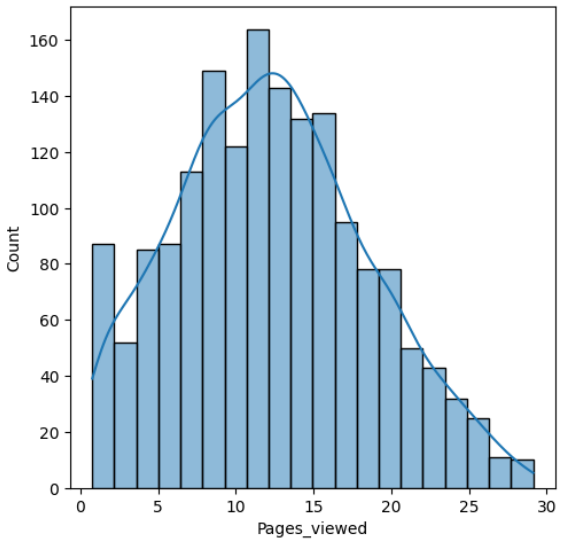


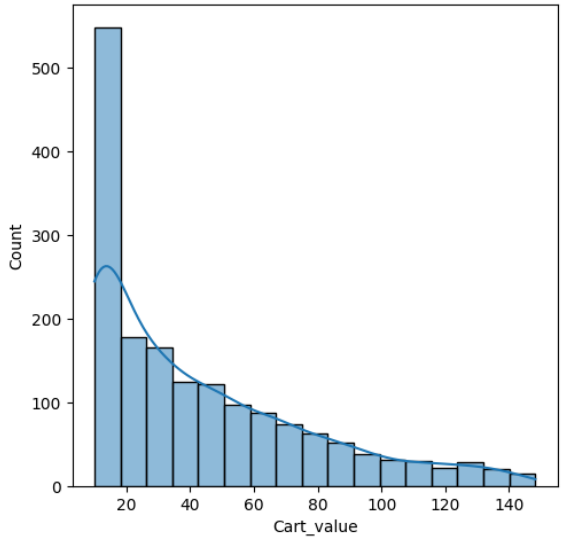


And then after detection, we remove the data points outside the set limit bounds.

Now the histogram looks like:





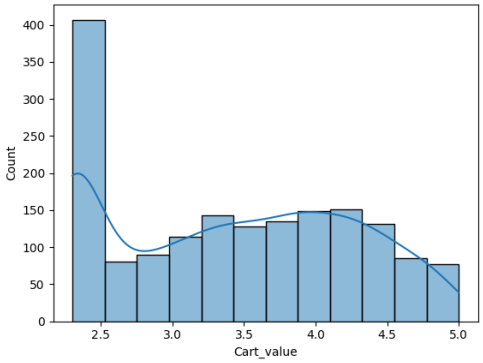



```
train.shape
```

```
(1690, 8)
```

1690 columns remain from the original 1800, therefore 110 points have been removed as outliers.

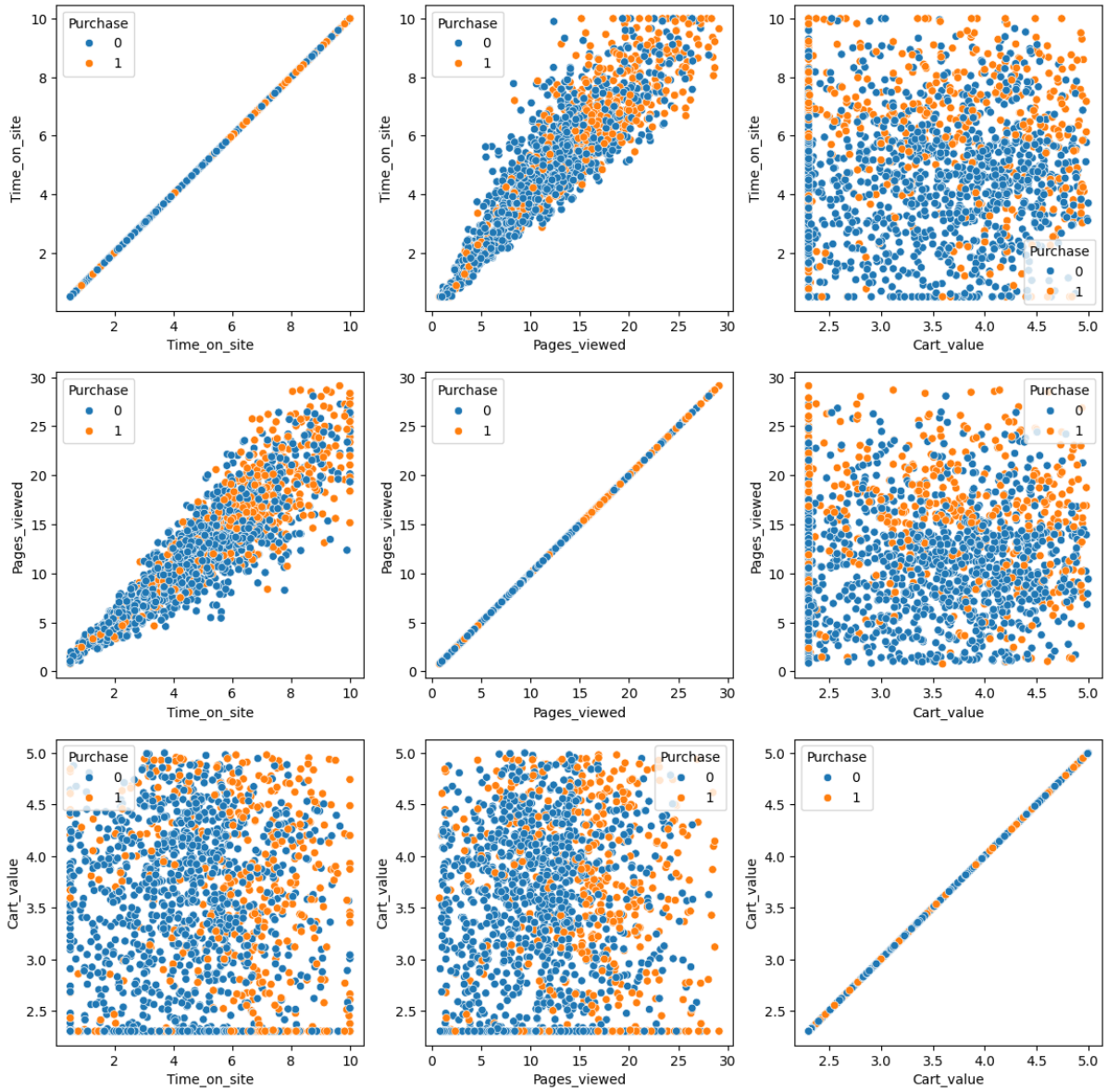
Now we perform the log transformation on Cart_value:



After log transformation the skew is 0.041.

Bivariate Analysis

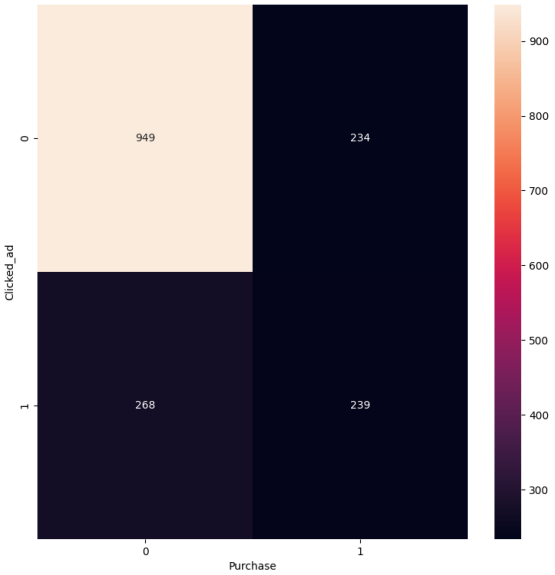
We observe the scatterplot of all the various numerical columns with each other.

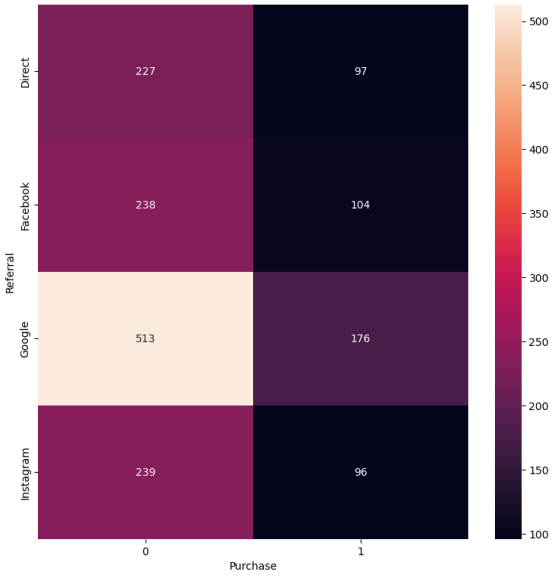


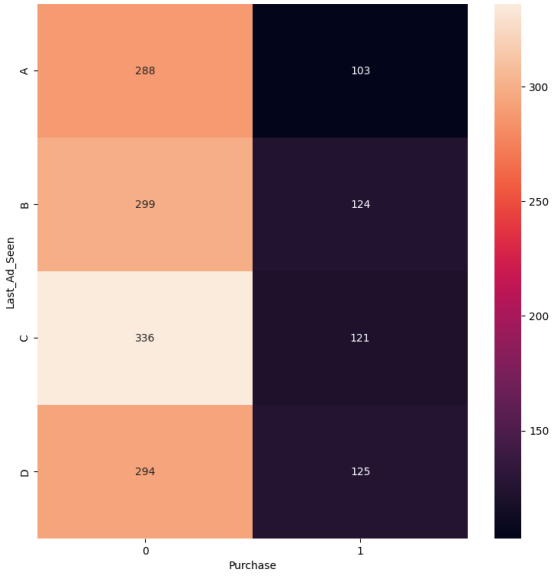
The linear relationship between Pages_viewed and Time_on_site shows us that these features are highly correlated and to avoid multi-collinearity, its good to drop one feature. In this case, Pages_viewed is dropped.

Also, across all scatterplots, the orange and blue dots overlap heavily implying absence of a clear boundary separating the two values (0 and 1) of Purchase. This might affect the accuracy of the model.

We also observe the heatmaps of the categorical features with Purchase (The Target).







1) Among the users who clicked an ad, purchases (239) are equal to non-purchases (268). But among those who did not click an ad, there were far more non-purchases (949) than purchases (234). Hence, Clicked_ad has a high correlation with Purchases.

2) Referrals from all sources have almost 2-3 times more non-purchases than purchases. Maximum Referrals and the highest purchases come from Google.

3) Ad D has a slightly higher purchase ratio (0.42) than the other ads, but overall, the purchase ratio remains in the similar range.

We will one hot encode all the categorical columns which will help the model predict accurate outcomes.

Model Selection and Explanation

We will use Logistic Regression and Random Forest Classifiers as our two main models to train, predict and classify.

Logistic Regression

Logistic Regression is a supervised learning algorithm used for classification tasks, in our case, for binary classification (purchased (1) or not purchased (0)). Logistic Regression estimates the probability of any given input, in which particular class they belong to, by applying a sigmoid function to a linear combination of its features.

The linear combination of the features is given by: $\theta^{\mathrm{T}}X$ where $\theta^{\mathrm{T}}X = \theta_{\mathrm{o}} + \sum_{\mathrm{j}=1}^{\mathrm{d}} \theta_{\mathrm{j}}\mathrm{x}_{\mathrm{j}}$

The sigmoid function is given by $\sigma(z) = \frac{1}{1+e^{-z}}$

Therefore, our predicted output or our hypothesis is $h_\theta(X) = \sigma(\theta^T X) = \frac{1}{1+e^{-\theta^T X}}$

The range of $h_\theta(X) \in (0,1)$

We endow our model with a set of probabilistic assumptions, and then fit the parameters using the maximum likelihood of θ. We can define our probabilistic assumptions as:

$$P(y = 1 \mid x;\theta) = h_\theta(X)$$
$$P(y = 0 \mid x;\theta) = 1 - h_\theta(X)$$

Which is equal to:

$$P(y \mid x;\theta) = \left(h_\theta(X)\right)^y \left(1 - h_\theta(X)\right)^{1-y}$$

Now we maximize the Likelihood of θ which is given as:

$$L(\theta) = \prod_{i=1}^{n} P\left(y^{(i)} \mid x^{(i)};\theta\right)$$

It is much more convenient to maximize the log of Likelihood of θ:

$$l(\theta) = \log\left(L(\theta)\right) = \sum_{i=1}^{n} y^{(i)} \log\left(h\left(x^{(i)}\right)\right) + \left(1 - y^{(i)}\right) \log\left(1 - h\left(x^{(i)}\right)\right)$$

The negative of this log of likelihood of θ is the Loss Function of Logistic Regression and we to minimize this Loss Function using gradient descent.

$$Loss(\theta) = -l(\theta)$$

Gradient descent is given by:

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} Loss(\theta)$$

After constructing the model, Logistic Regression works on the training dataset by find the value of $h_\theta(X)$ of each new input and if its value is greater than the threshold (0.5), it gives the output as (1) and if its value is lesser than (0.5), it gives the output as (0).

Logistic regression is suitable for this problem as it interprets the output off probabilities, which is useful in decision making. It is a go to model for binary classification with a clear loss function built on the probabilistic interpretation based on the maximum likelihood of the parameters.

Random Forest Classifier

Random Forest is a collection of many randomized decision trees. It is an Ensemble Technique in which we take many different random subsets of the training data (where data points can be repeated) or we take random subsets of features or both. We train many decision trees based on these randomized data sets.

The final outcome for any testing data point is given by the most votes received from the many different decision trees we trained. This method is an implementation of Bagging. The term "Bagging" comes from the words Bootstrapping, which essentially describes the randomization of the various decision trees the model creates, and Aggregation, which describes the final outcome based on the voting process of these various decision trees.

Random Forest is a powerful classifier as it solves the problem of the Bias Variance trade-off. While individual decision trees tend to have low bias but high variance (they often overfit the training data), Random Forest reduces this variance by averaging the predictions from many diverse trees. This ensemble approach smoothen outs the predictions, making the model more stable and less sensitive to fluctuations in the training data.

Model Training and Evaluation

The two models are trained using the scikit-learn module. The evaluation of the models is done using the confusion matrix, accuracy and F1 scores. Since the target variable, Purchase, in our training data is imbalanced and has a high bias toward 0 than 1, we find the weighted F1 score to fairly evaluate the models' performances across all classes.

The confusion matrix is given by:

$$Confusion\ Matrix = \begin{bmatrix} TN & FP \\ FN & TP \end{bmatrix}$$

Where,

1) TN [True Negative] : Predicted and Actual value is 0
2) TP [True Positive] : Predicted and Actual value is 1
3) FN [False Negative] : Predicted value is 0 but Actual value is 1
4) FP [False Positive] : Predicted value is 1 but Actual value is 0

The Accuracy is given by:

$$Accuracy = \frac{TN + TP}{TN + TP + FN + FP}$$

The F1 score is given by the harmonic mean of Precision and Recall:

$$F1\ score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

Where,

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

Evaluation of Logistic Regression model

```
print("Confusion Matrix using Logistic Regression is:\n", confusion_matrix(y_test, y_pred_lr), "\n")
print("Accuracy score using Logistic Regression:\n", accuracy_score(y_test, y_pred_lr), "\n")
print("F1 Score using Logistic Regression:\n", f1_score(y_test, y_pred_lr ,average = 'weighted'))
```

```
Confusion Matrix using Logistic Regression is:
 [[281  33]
 [ 62  74]]

Accuracy score using Logistic Regression:
 0.7888888888888889

F1 Score using Logistic Regression:
 0.7809509492580691
```

1) Accuracy = 78.88%
2) F1 Score = 78.09%

Evaluation using Random Forest Classifier

```
print("Confusion Matrix using Random Forest is:\n", confusion_matrix(y_test, y_pred_rf), "\n")
print("Accuracy score using Random Forest:\n", accuracy_score(y_test, y_pred_rf), "\n")
print("F1 Score using Random Forest:\n", f1_score(y_test, y_pred_rf, average = 'weighted'))
```

```
Confusion Matrix using Random Forest is:
 [[264  50]
 [ 62  74]]

Accuracy score using Random Forest:
 0.7511111111111111

F1 Score using Random Forest:
 0.7477008547008547
```

1) Accuracy = 75.11%
2) F1 Score = 74.77%

Hyperparameter Tuning

Grid Search is performed on relevant hyperparameters for both the models. Grid Search is a systematic method used for tuning the hyperparameters of a machine learning model to find the combination that gives the best performance. It works by exhaustively trying out all possible combinations of specified values for each hyperparameter, training and evaluating the model for each one using cross-validation.

Logistic Regression Hyperparameter Tuning

```
param_grid_lr = {
    'logisticregression__C': [0.01, 0.1, 1, 10, 100],
    'logisticregression__penalty': ['elasticnet'],
    'logisticregression__solver': ['saga'],
    'logisticregression__l1_ratio': [0.1, 0.5, 0.9],
    'logisticregression__max_iter': [100, 200]
}
```

Best Hyperparameters found after Grid search across 30 unique candidates cross validating 5 times are:

1. C (Inverse of Regularization Strength) = 1
2. penalty = elasticnet
3. solver = saga
4. L1_ratio = 0.1
5. max_iter = 100

New F1 Score = 73.45%

Random Forest Hyperparameter Tuning

```
param_grid_rf = {
    'randomforestclassifier__max_depth': [2, 5, 10, 15],
    'randomforestclassifier__n_estimators': [100, 300, 500],
    'randomforestclassifier__min_samples_split': [2, 5],
}
```

Best Hyperparameters found after Grid search across 24 unique candidates cross validating 5 times are:

1. max_depth = 5
2. n_estimators = 500
3. min_samples_split = 2

New F1 Score = 72.53%

Hyperparameter tuning using Grid Search is expected to produce better outputs. But it is observed that there is actually a reduction in the F1 scores in both the models than before. This can be due to the reason that the hyperparameters and their values we chose do not cover the entire scope of the search for optimal results and the default values of the hyperparameters already are optimized to provide good results for our data set.

Comparative Analysis and Conclusion

In this Project, Logistic regression outperforms Random Forest in both Accuracy and F1 scores. This could be due to good preprocessing that was done on the data. Logistic Regression tends to perform very well on datasets that have good feature engineering done on them. There could also be an issue of overfitting in Random Forest as the data was highly scattered.

In conclusion, this project focused on understanding and predicting customer purchasing behaviour on an e-commerce platform by analyzing behavioural signals. The initial dataset, comprising of 1800 individual user sessions, was found to be free of missing or duplicate data points, indicating a good foundation for analysis. Exploratory Data Analysis revealed the dataset's structure, encompassing 6 numerical and 2 object features, further categorized into 4 numerical and 4 categorical columns. A strategic decision was made to drop 'Pages_viewed' due to its high correlation with 'Time_on_site', a measure to mitigate multicollinearity and enhance model performance. 'Browser_refresh_rate' was also dropped due to its low correlation with 'Purchase'. The identification of the imbalanced target variable, 'Purchase', underscored the necessity for careful evaluation metrics like the weighted F1 score. One Hot Encoding on the categorical columns and Standard Scaling on the numerical was done. While log transformation was applied on 'Cart_value' as it had a highly positive skewed distribution.

Both Logistic Regression and Random Forest models were employed for classification. While Logistic Regression generally demonstrated superior performance in terms of accuracy and F1 scores, potentially attributable to the efficacy of the data preprocessing steps, hyperparameter tuning unexpectedly led to a reduction in F1 scores for both models. This suggests that the chosen hyperparameter ranges might not have fully encompassed the optimal parameter space, or that the default settings were already well-optimized for the given dataset.

By Adwaiy Sakolkar

24IM10014