CIS 550: ADVANCED MACHINE LEARNING

Final Project Summary

Vaishnavi Paineni

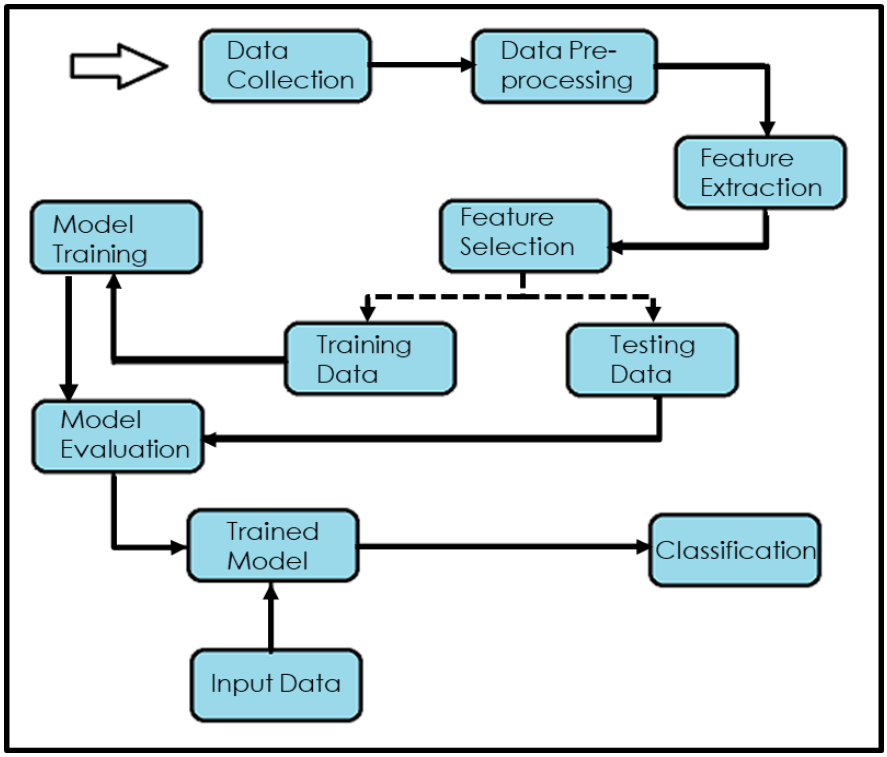
Predicting Customer Churn in Banking: A Binary Classification Approach

Introduction

Customer churn, in the context of businesses like banking, refers to the phenomenon where customers discontinue their relationship with a service or company. Predicting customer churn is crucial as it allows businesses to proactively identify customers who are likely to leave. By leveraging machine learning models for predicting churn, businesses gain the ability to foresee potential attrition and implement targeted retention strategies. These strategies may include personalised offers, improved customer service, or other interventions aimed at preventing customer departure. Predicting churn not only helps in retaining valuable customers but also contributes to optimising resource allocation, enhancing customer satisfaction, and maintaining overall business sustainability. It enables businesses to take proactive measures, reduce revenue loss, and foster long-term customer loyalty.

Machine Learning Pipeline

For this binary classification project, we've used the standard machine learning pipeline where the first step was Data Collection. The data was collected from Kaggle. The following steps were Data Cleaning and preprocessing, Feature Engineering, Splitting the Data for the Train and Test Dataset, Training the Model, and Evaluating the Model.



Dataset Overview

This project aims to predict customer churn in the banking industry based on the dataset from Kaggle (https://www.kaggle.com/datasets/radheshyamkollipara/bank-customer-churn/). The dataset contains information about the customers such as their CustomerId, Surname, Geography, Gender, Age, etc, and banking information such as Tenure, Balance, NumOfProducts, HasCrCard, IsActiveMember, EstimatedSalary, Exited, etc.

Attributes

1. RowNumber—corresponds to the record (row) number and does not affect the output.
2. CustomerId—contains random values and does not affect customers leaving the bank.
3. Surname—the surname of a customer has no impact on their decision to leave the bank.
4. CreditScore—can affect customer churn, since a customer with a higher credit score is less likely to leave the bank.
5. Geography—a customer's location can affect their decision to leave the bank.
6. Gender—it's interesting to explore whether gender plays a role in a customer leaving the bank.

7. Age—this is certainly relevant since older customers are less likely to leave their bank than younger ones.

8. Tenure—refers to the number of years that the customer has been a client of the bank. Normally, older clients are more loyal and less likely to leave a bank.

9. Balance—is also a very good indicator of customer churn, as people with a higher balance in their accounts are less likely to leave the bank compared to those with lower balances.

10. NumOfProducts—refers to the number of products that a customer has purchased through the bank.

11. HasCrCard—denotes whether or not a customer has a credit card. This column is also relevant since people with credit cards are less likely to leave the bank.

12. IsActiveMember—active customers are less likely to leave the bank.

13. EstimatedSalary—as with balance, people with lower salaries are more likely to leave the bank compared to those with higher salaries.

14. Exited—whether or not the customer left the bank. (Target Column)

15. Complain—customer has a complaint or not.

16. Satisfaction Score—Score provided by the customer for their complaint resolution.

17. Card Type—the type of card held by the customer.

18. Points Earned—the points earned by the customer for using a credit card.

```
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 18 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   RowNumber           10000 non-null  int64
 1   CustomerId          10000 non-null  int64
 2   Surname             10000 non-null  object
 3   CreditScore         10000 non-null  int64
 4   Geography           10000 non-null  object
 5   Gender              10000 non-null  object
 6   Age                 10000 non-null  int64
 7   Tenure              10000 non-null  int64
 8   Balance             10000 non-null  float64
 9   NumOfProducts       10000 non-null  int64
 10  HasCrCard           10000 non-null  int64
 11  IsActiveMember      10000 non-null  int64
 12  EstimatedSalary     10000 non-null  float64
 13  Exited              10000 non-null  int64
 14  Complain            10000 non-null  int64
 15  Satisfaction Score  10000 non-null  int64
 16  Card Type           10000 non-null  object
 17  Point Earned        10000 non-null  int64
```

Data Cleaning & Preprocessing

As the dataset has several attributes that do not affect the churn prediction, some columns are dropped from the dataset such as Customer ID, Surname, RowNumber, Complain, SayisfactionScore, Card Type, and Points Earned. After cleaning the data, data preprocessing is done by encoding the categorical features and standardizing the scales for the numerical features. The label encoding is done to features such as Geography and Gender. For geography, 'France' is encoded as '0', 'Germany' as '1', and 'Spain' as '2'. For gender, 'Female' is encoded as '0', and 'Male' is encoded as '1'. Finally, Normalization was performed on numerical features such as Salary, Balance, Credit Score, and Age.

```
#Label Encoding
label_encodor = LabelEncoder()
lstforle = ['Geography', 'Gender']
for i in lstforle :
    df[i] = label_encodor.fit_transform(df[i])
    print(i,' : ',df[i].unique(),' = ',label_encodor.inverse_transform(df[i].unique()))
```

```
#normalization
sc = StandardScaler()
for col in num_Col:
    df[col] = sc.fit_transform(df[[col]])
df.head()
```

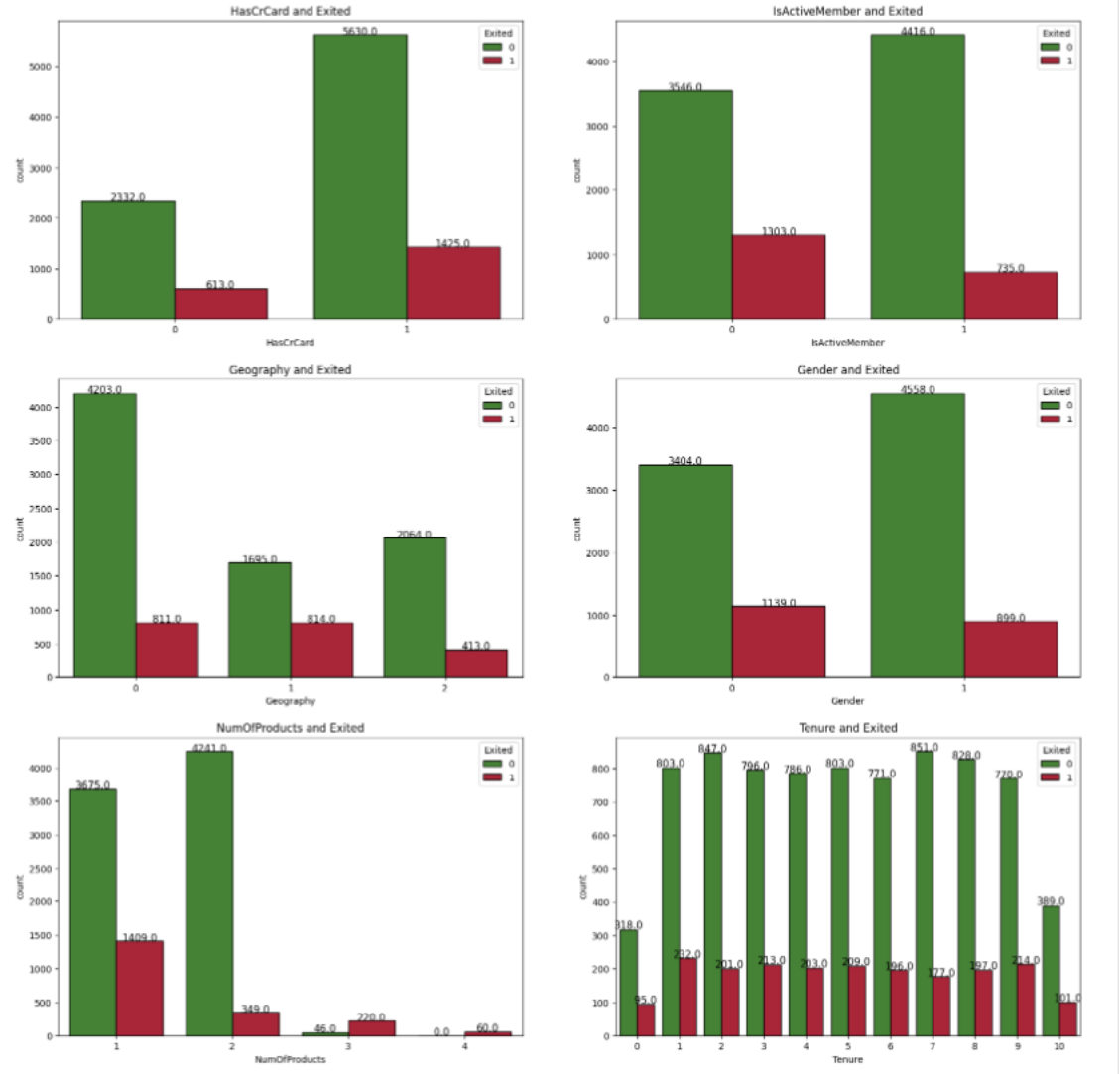
Data Visualization

To gain a better understanding of categorical and numerical features, we visualise data using different types of plots.

Analysis of categorical features

'HasCrCard', 'IsActiveMember', 'Geography', 'Gender', 'NumOfProducts', and 'Tenure' are individually explored through visual representations showcasing their impact on churn. Here we have utilised seaborn's **count plot** and matplotlib's **subplots**. Insights present churn percentages within each category of these features, providing a clear understanding of their influence on customer churn.

```
                horizontalalignment='center', fontsize = 11)
    title = cat_Col[i] + ' and Exited'
    plt.title(title);
```



From the above plot, we can gain the following insights

- The HasCrCard feature did not show class differentiation.
- The Geography feature turned out to be underestimated by the Chi-square. Country 1 (Germany) shows the highest customer churn, while the other two have an equal distribution of classes.
- The IsActiveMember feature predictably indicates that the chance of losing an active customer is less than that of an inactive one.
- The Gender feature also shows a significant difference between classes: women are 70% more likely to leave the bank.
- The NumOfProducts feature shows quite significant differences between classes. The smallest churn rate is among customers with two bank products. However, after two products, the churn increases.

- The Tenure feature shows a slow decline until 8 years. At 8-9 years, churn rates show an increase.

Analysis of numerical features

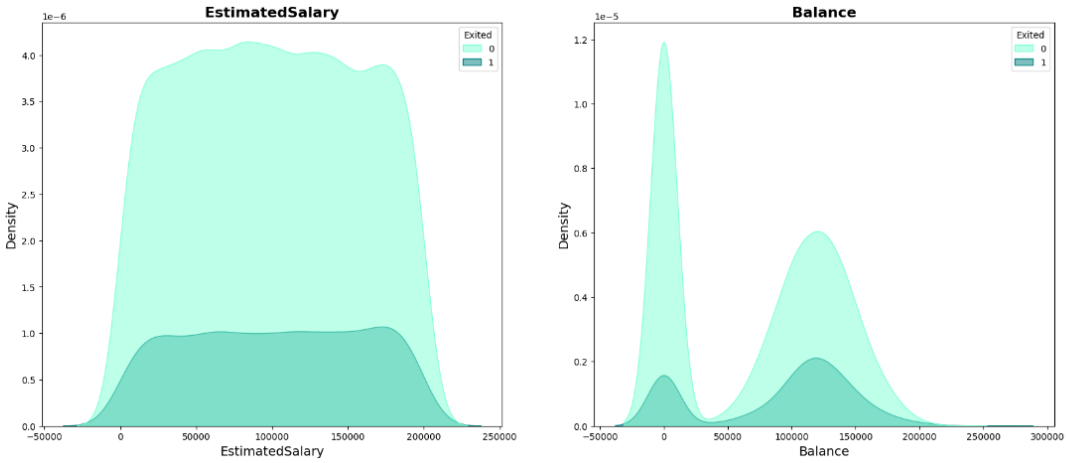
Examined key numerical features including 'EstimatedSalary', 'Balance', 'CreditScore', and 'Age' via histograms and kernel density estimations (KDE).

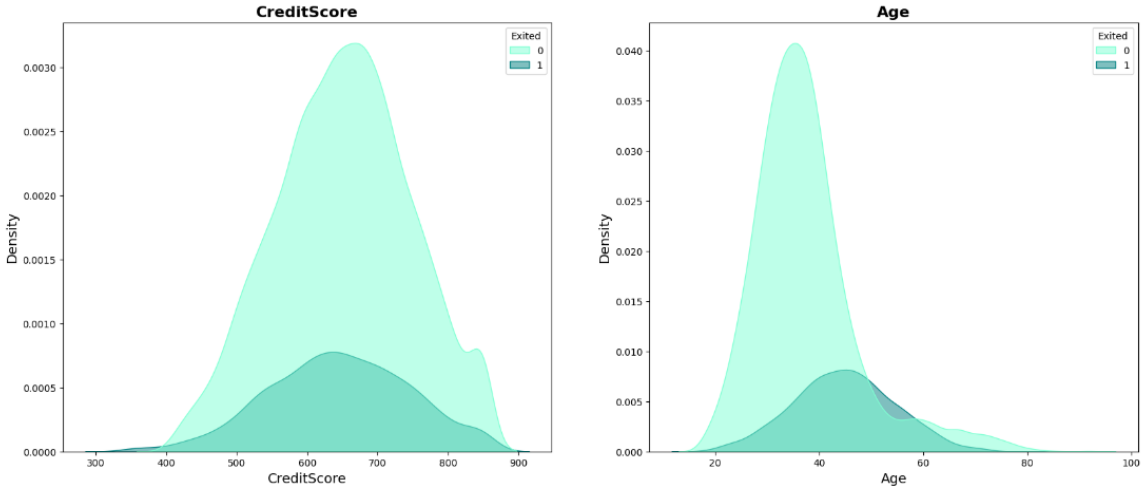
The light green represents the customer remained and the dark green represents the customer churned.

In [ ]:

```
fig, axes = plt.subplots(2,2, figsize = (20,20))
plt.subplots_adjust(hspace=0.5)
columns = df[num_Col]
for i, column in enumerate(columns):
    ax = axes[i // 2, i % 2]
    sns.kdeplot(data = df,
                x = column,
                fill = True,
                alpha = 0.5,
                hue = 'Exited',
                palette = [ '#7FFFD4', '#008080'],
                ax = ax)

    ax.set_title(column, fontweight = 'bold', fontsize = 16)
    ax.set_xlabel(column, fontsize = 14)
    ax.set_ylabel('Density', fontsize = 14)
plt.show()
```





The key insights that we can get from the above plots are:

- With the increase in EstimatedSalary, the churn rate shows a slight linear increase.
- When the balance is zero, the churn rate is lower compared to other cases.
- Customers with a high credit score are less likely to leave the bank.
- There is a significant difference in the age distributions.

Normalization

To ensure that all numerical features have a common scale we have used the Normalization technique **StandardScalar**. This process prevents features with larger scales from dominating or influencing the machine learning model more than those with smaller scales.

In [ ]:
```
#normalization
sc = StandardScaler()
for col in num_Col:
    df[col] = sc.fit_transform(df[[col]])
df.head()
```

Out[22]:

	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	-0.326221	0	0	0.293517	2	-1.225848	1	1	1	0.021886	1
1	-0.440036	2	0	0.198164	1	0.117350	1	0	1	0.216534	0
2	-1.536794	0	0	0.293517	8	1.333053	3	1	0	0.240687	1
3	0.501521	0	0	0.007457	1	-1.225848	2	0	0	-0.108918	0
4	2.063884	2	0	0.388871	2	0.785728	1	1	1	-0.365276	0

In the provided code, StandardScaler() scales each feature column independently by removing the mean and scaling it to unit variance. This approach transforms the data so that the mean of each feature is 0 and the variance is 1.

Oversampling

The uneven distribution within our target column (Exited) leads to an imbalance, which can introduce bias in our machine learning model.

```
#Exited  (l[0]-Remained  1[1]-Churned)
l = list(df['Exited'].value_counts())
print(f"Remained: {l[0]} ; Churned: {l[1]}")
```

```
Remained: 7962 ; Churned: 2038
```

Oversampling is a technique used in machine learning to address class imbalance by increasing the number of instances in the minority class(es) to prevent bias toward the majority class in the model.

```
#oversampling
over = SMOTE(sampling_strategy = 1)

f1 = df.drop(columns=['Exited']).iloc[:,:].values
t1 = df.iloc[:, -1].values

f1, t1 = over.fit_resample(f1, t1)
Counter(t1)
```

```
Counter({1: 7962, 0: 7962})
```

Here we use **SMOTE**(Synthetic Minority Over Sampling Technique) to achieve this.

SMOTE works by generating synthetic samples for the minority class by interpolating between existing instances, thus creating a more balanced representation of both classes in the dataset. With a sampling_strategy set to 1, SMOTE generated synthetic samples equivalent to the majority class, allowing us to address the imbalance and create a more robust dataset for model training.

Splitting the data

In [ ]:
```
#spliting the data into training and testing
x_train, x_test, y_train, y_test = train_test_split(f1, t1, test_size = 0.22, random_state = 100)
```

Having achieved a balanced class distribution using SMOTE, we partitioned our dataset into training and testing subsets for model development. The split allocated 22% of the data for testing purposes and reserved 78% for model training.

Model Training and Evaluation

In this section, we train various machine learning models with our dataset for the customer churn prediction. To understand and compare the performances of these models, we design two functions namely - **'RunModel'** and **'model_evaluation'**

RunModel

```
def RunModel(classifier,x_train,y_train,x_test,y_test):

    classifier.fit(x_train,y_train)
    pred = classifier.predict(x_test)
    cv = RepeatedStratifiedKFold(n_splits = 10,n_repeats = 3,random_state = 1)
    cr_v_sc = cross_val_score(classifier,x_train,y_train,cv = cv,scoring = 'roc_auc')
    r_a_s = roc_auc_score(y_test,pred)
    print("Cross Validation Score : ",'{0:.2%}'.format(cr_v_sc.mean()))
    print("ROC_AUC Score : ",'{0:.2%}'.format(r_a_s))
```

The function has the following parameters:-

- *classifier* - The machine learning model to be trained
- *x_train, y_train* - Training data features and labels
- *x_test, y_test* - Testing data features and labels

The function is responsible for training the classifier and evaluating the performance by Cross-Validation and ROC_AUC score.

Cross-Validation score is a performance metric to assess the model's ability to generalise by averaging its performance across multiple subsets of the dataset. We can under how well the model might perform on an independent dataset by this metric.

$$\textit{Cross-Validation Score} = 1/k\ \Sigma^{k}_{i=1}\ \textit{Score}_i$$

where k is the number of folds in the cross-validation which is 10 in our case and repeats 3 times.

Receiver Operating Characteristic Area Under the Curve (ROC AUC) score is a performance metric that evaluates the trade-off between true positive rate (sensitivity) and false positive rate (1 - specificity) for different threshold values. It is commonly used for binary classification tasks.

$$\textit{ROC\_AUC Score} = ½(\textit{True Positive Rate} + \textit{True Negative Rate})$$

This score ranges from 0 to 1, where 1 indicates perfect classification, and 0.5 represents random chance. A higher ROC AUC score signifies better discrimination performance of the model. We have used it as a percentage of 100 in our project.

model_evaluation

```
def model_evaluation(classifier,x_test,y_test):

    cm = confusion_matrix(y_test,classifier.predict(x_test))
    names = ['True Neg','False Pos','False Neg','True Pos']
    counts = [value for value in cm.flatten()]
    percentages = ['{0:.3%}'.format(value) for value in cm.flatten()/np.sum(cm)]
    labels = [f'{v1}\n{v2}\n{v3}' for v1, v2, v3 in zip(names,counts,percentages)]
    labels = np.asarray(labels).reshape(2,2)
    sns.heatmap(cm,annot = labels,cmap = ['#20B2AA','#48D1CC'],fmt ='')

    print(classification_report(y_test,classifier.predict(x_test)))
```

The function has the following parameters:-

- *classifier* - The machine learning model to be trained
- *x_test, y_test* - Testing data features and labels

The function is responsible for generating a confusion matrix and a heatmap for visualizing model predictions. Additionally, it prints a comprehensive classification report, including precision, recall, F1-score, and support for each class. This helps us understand the strengths and weaknesses of the model.

Confusion matrix visualizes the performance of a classification algorithm. It compares the predicted labels of a model with the actual labels of the dataset, providing insights into true positive, true negative, false positive, and false negative predictions. We have used a heatmap for visualizing these predictions.

TrueNegative(TN) *False Positive(FP)*

False Negative(FN) *True Positive(TP)*

- *True Negatives (TN):* Instances correctly predicted as the negative class.
- *False Positives (FP):* Instances incorrectly predicted as the positive class.
- *False Negatives(FP):* Instances incorrectly predicted as the negative class.
- *True Positives (FP):* Instances correctly predicted as the positive class.

The Classification report presents precision, recall, F1-score, and support for each class.

Precision measures the accuracy of positive predictions.

Precision = True Positives (TP) / (True Positives (TP) + False Positives (FP))

Higher precision values signify a lower rate of false positives and, therefore, better accuracy in positive predictions.

Recall assesses the model's ability to capture all positive instances

Recall = True Positives (TP) / (True Positives (TP) + False Negatives (FN))

Higher recall values signify a lower rate of false negatives and better coverage of positive instances by the model.

F1-score represents the balance between precision and recall.

F1 Score = 2 x Precision x Recall / (Precision + Recall)

Support indicates the number of actual occurrences of each class.

Model Comparison

Next, we compare the performance of various machine learning models trained for customer churn prediction.

Logistic Regression

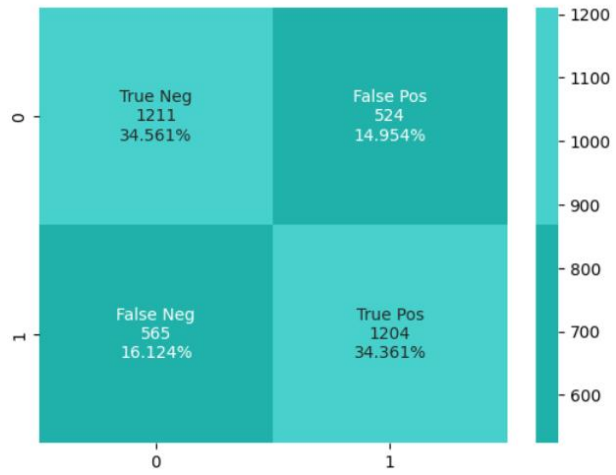
Logistic Regression is a foundational model for binary classification tasks. It uses the logistic function to transform the output into a probability, making it suitable for binary classification. The model minimizes a logistic loss function during training.

```
LR = LogisticRegression()

RunModel(LR,x_train,y_train,x_test,y_test)
```

```
Cross Validation Score :  77.04%
ROC_AUC Score :  68.93%
```

	precision	recall	f1-score	support
0	0.68	0.70	0.69	1735
1	0.70	0.68	0.69	1769
accuracy			0.69	3504
macro avg	0.69	0.69	0.69	3504
weighted avg	0.69	0.69	0.69	3504



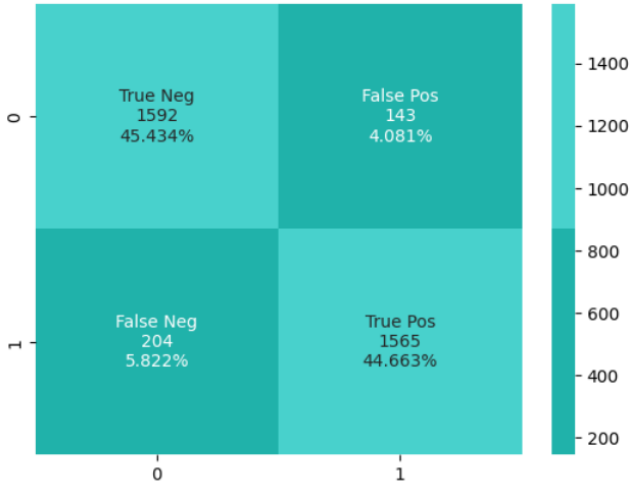
Random Forest Classifier

Random Forest is an ensemble model that combines multiple decision trees to improve predictive performance. It excels in capturing complex relationships within the data.

```
RanForCla = RandomForestClassifier()
RunModel(RanForCla,x_train,y_train,x_test,y_test)
```

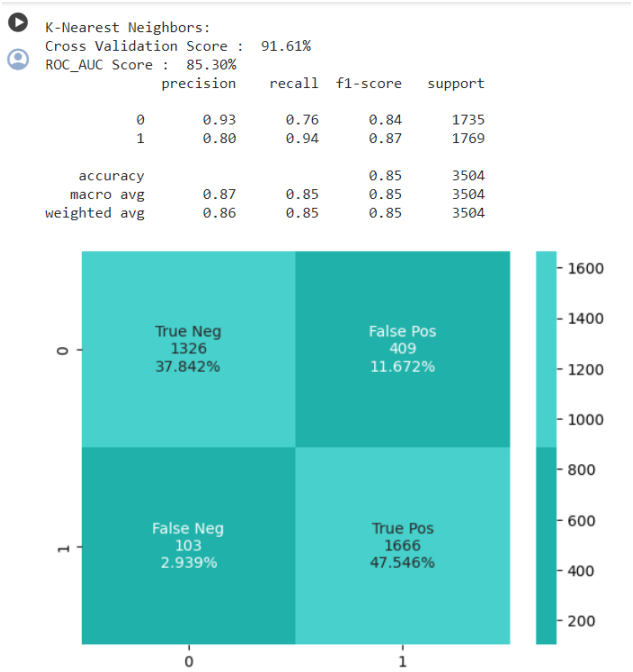
```
Cross Validation Score :  96.17%
ROC_AUC Score :  90.09%
```

	precision	recall	f1-score	support
0	0.89	0.92	0.90	1735
1	0.92	0.88	0.90	1769
accuracy			0.90	3504
macro avg	0.90	0.90	0.90	3504
weighted avg	0.90	0.90	0.90	3504



K-Nearest Neighbors

KNN is a non-parametric and instance-based learning algorithm. It classifies a data point based on the majority class of its k-nearest neighbors. It relies on measuring distances between data points to determine similarity. The class label is assigned based on the majority class among the k-nearest neighbors.



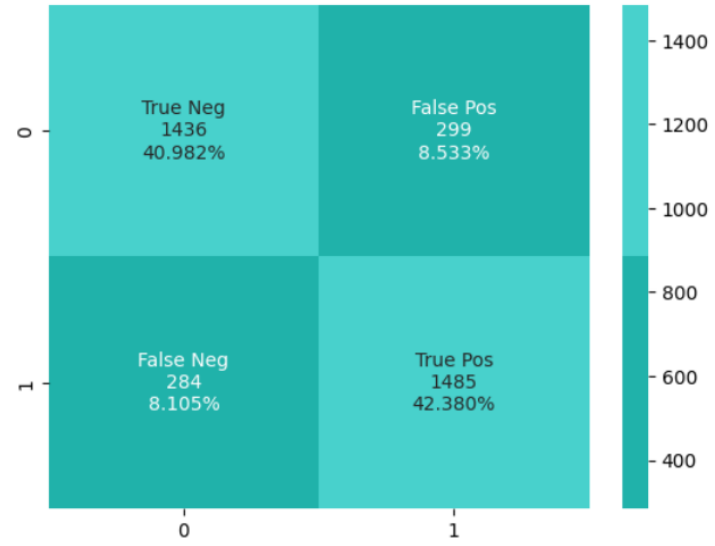
Decision Tree Classifier

Decision Trees are intuitive models that make decisions based on feature values. They are easy to interpret and provide insights into feature importance. The tree structure allows for clear visualization of decision-making paths.

```
Decision Tree:
Cross Validation Score :  84.77%
ROC_AUC Score :  83.36%
              precision    recall  f1-score   support

           0       0.83      0.83      0.83      1735
           1       0.83      0.84      0.84      1769

    accuracy                           0.83      3504
   macro avg       0.83      0.83      0.83      3504
weighted avg       0.83      0.83      0.83      3504
```



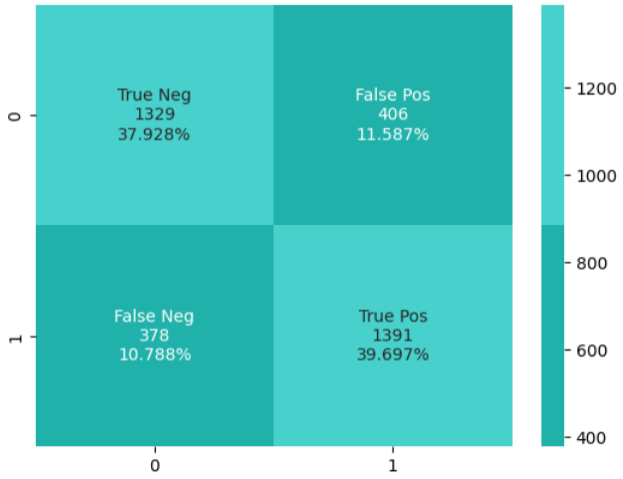
Support Vector Machine

SVM maps data into a higher-dimensional space and finds the hyperplane that maximally separates classes. The model is effective in high-dimensional spaces.

```
Support Vector Machine:
Cross Validation Score :  85.01%
ROC_AUC Score :  77.62%
              precision    recall  f1-score   support

           0       0.78      0.77      0.77      1735
           1       0.77      0.79      0.78      1769

    accuracy                           0.78      3504
   macro avg       0.78      0.78      0.78      3504
weighted avg       0.78      0.78      0.78      3504
```



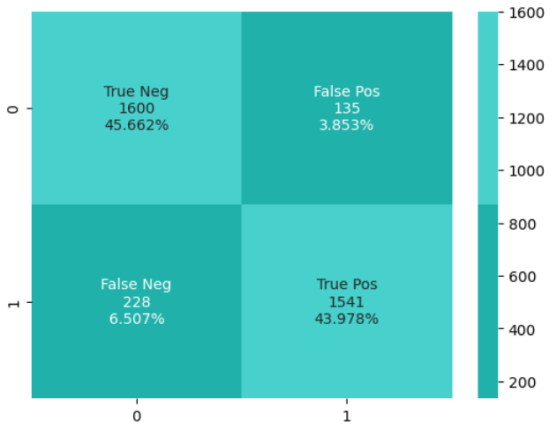
XGBoost

XGBoost is a gradient-boosting algorithm that builds a series of weak learners to create a strong predictive model.

```
XGBoost:
Cross Validation Score :  95.79%
ROC_AUC Score :  89.67%
              precision    recall  f1-score   support

           0       0.88      0.92      0.90      1735
           1       0.92      0.87      0.89      1769

    accuracy                           0.90      3504
   macro avg       0.90      0.90      0.90      3504
weighted avg       0.90      0.90      0.90      3504
```



Conclusion

Model	Cross-Validation Score	ROC-AUC Score	Accuracy
Logistic Regression	77.15%	68.93%	69%
Random Forest	96.24%	90.96%	91%
KNN	91.56%	84.61%	85%
Decision Tree	84.76%	84.50%	85%
SVM	84.97%	76.94%	77%
XGBoost	95.87%	89.81%	90%

- Considering the diverse nature of customer churn prediction, where both precision and recall are crucial, Random Forest and XGBoost emerge as the top-performing models. These models exhibit high accuracy, a balanced trade-off between precision and recall, and robust predictive capabilities. The choice between them may depend on specific business objectives, interpretability, and computational resources.
- Further fine-tuning and optimization can be explored to enhance the performance of the chosen model for real-world deployment.