Customer Churn Prediction

Introduction

About the problems

Customer churn is the loss of clients or customers. In order to avoid losing customers, a company needs to examine why its customers have left in the past and which features are more important to determine who will churn in the future. Our task is therefore to predict whether customers are about to churn and which are the most important features to get that prediction right. As in most prediction problems, we will use machine learning.

About the data

- Customers who left within the last month – the column is called Churn
- Services that each customer has signed up for – phone, multiple lines, internet, online security, online backup, device protection, tech support, and streaming TV and movies
- Customer account information – how long they've been a customer, contract, payment method, paperless billing, monthly charges, and total charges
- Demographic info about customers – gender, age range, and if they have partners and dependents

Main Objective

Our objective is to create a model to detect customers, including those who will churn or not, so that preventive measures can be taken to customers who have the possibility of churning to keep using the company's products.

This analysis focuses on prediction to obtain the highest accuracy from the model rather than interpretation.

Data Importing and First Lookup

```
RangeIndex: 7043 entries, 0 to 7042
Data columns (total 21 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   customerid        7043 non-null   object
 1   region            7043 non-null   object
 2   gender            7043 non-null   object
 3   seniorcitizen     7043 non-null   int64
 4   partner           7043 non-null   object
 5   dependents        7043 non-null   object
 6   tenure            7043 non-null   int64
 7   phoneservice      7043 non-null   object
 8   multiplelines     7043 non-null   object
 9   internetservice   7043 non-null   object
 10  onlinesecurity    7043 non-null   object
 11  onlinebackup      7043 non-null   object
 12  deviceprotection  7043 non-null   object
 13  techsupport       7043 non-null   object
 14  streamingtv       7043 non-null   object
 15  streamingmovies   7043 non-null   object
 16  paperlessbilling  7043 non-null   object
 17  paymentmethod     7043 non-null   object
 18  monthlycharges    7043 non-null   float64
 19  totalcharges      7043 non-null   object
 20  churn             7043 non-null   object
dtypes: float64(1), int64(2), object(18)
memory usage: 1.1+ MB
```

Our data contains 7043 observations and 21 variable:

- 20 independent variables
- 1 dependent variable with 2 classes

It has 18 object types (categorical variables), 2 int type (discrete variables), and 1 float type (continuous variable).

And it looks like we don't have missing values.

There are 2 variables with incorrect type:

1. seniorcitizen : must be an object type because it is categorical variable.
2. totalcharges : must be numeric type

We have to change the 1 and 0 values in seniorcitizen to 'Yes' and 'No' accordingly (because the other categorical variables has this value) to have better interpretation when we make visualization.

Next, we want to examine totalcharges why it has an object type.

```
# Check whether columns value contains  other than numeric and alphabet
for col in data.columns:
  print(col, data.loc[~data[col].astype(str).str.contains(pat='[0-9a-zA-Z]'), col].tolist())
```

```
customerid []
region []
gender []
seniorcitizen []
partner []
dependents []
tenure []
phoneservice []
multiplelines []
internetservice []
onlinesecurity []
onlinebackup []
deviceprotection []
techsupport []
streamingtv []
streamingmovies []
paperlessbilling []
paymentmethod []
monthlycharges []
totalcharges [' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ']
churn []
```

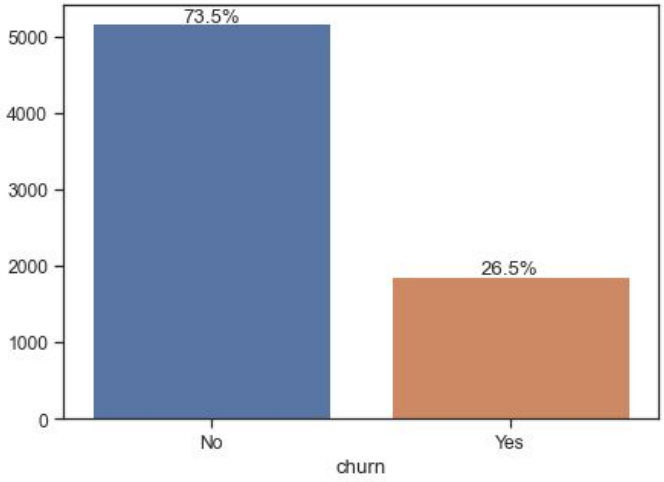
It turns out that the totalcharges column has a value containing ' '(space). This is why the data type is object.

It is necessary to change the value containing ' '(space) and change the data type to numeric.

Exploratory Data Analysis

Univariate Analysis

Target Class



Our data have slightly imbalance class distribution. It will cause a model hard to identify the minority class.

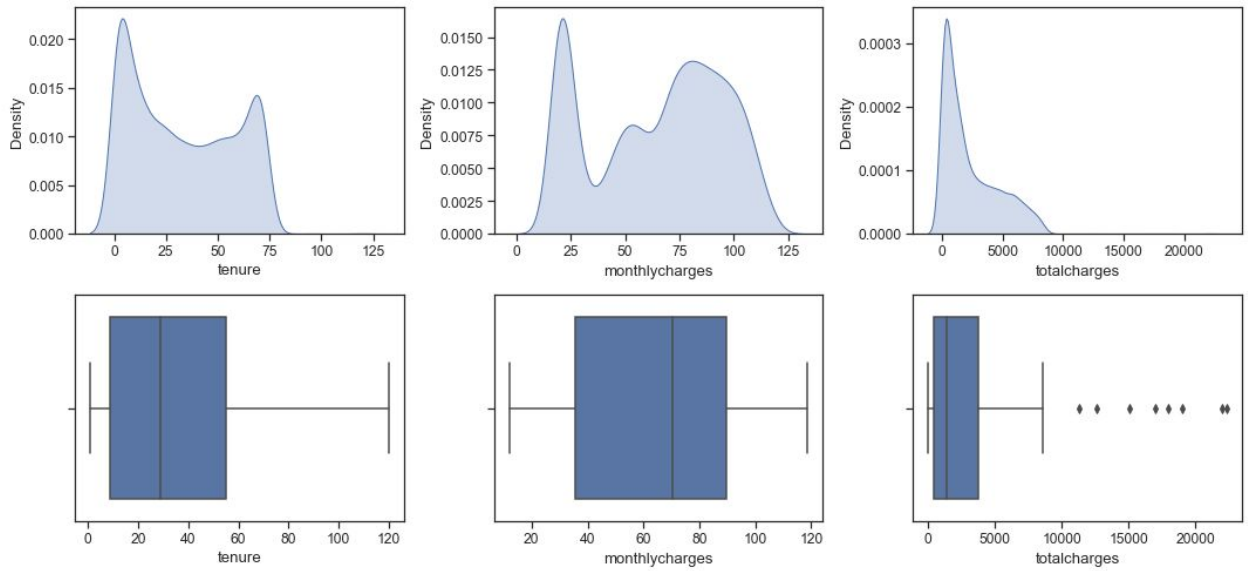
- No (Non Churn) : 73.5%
- Yes (Churn) : 26.5%

Consideration :

Required to tune the model or do resampling to meet our objective (detect the minority class).

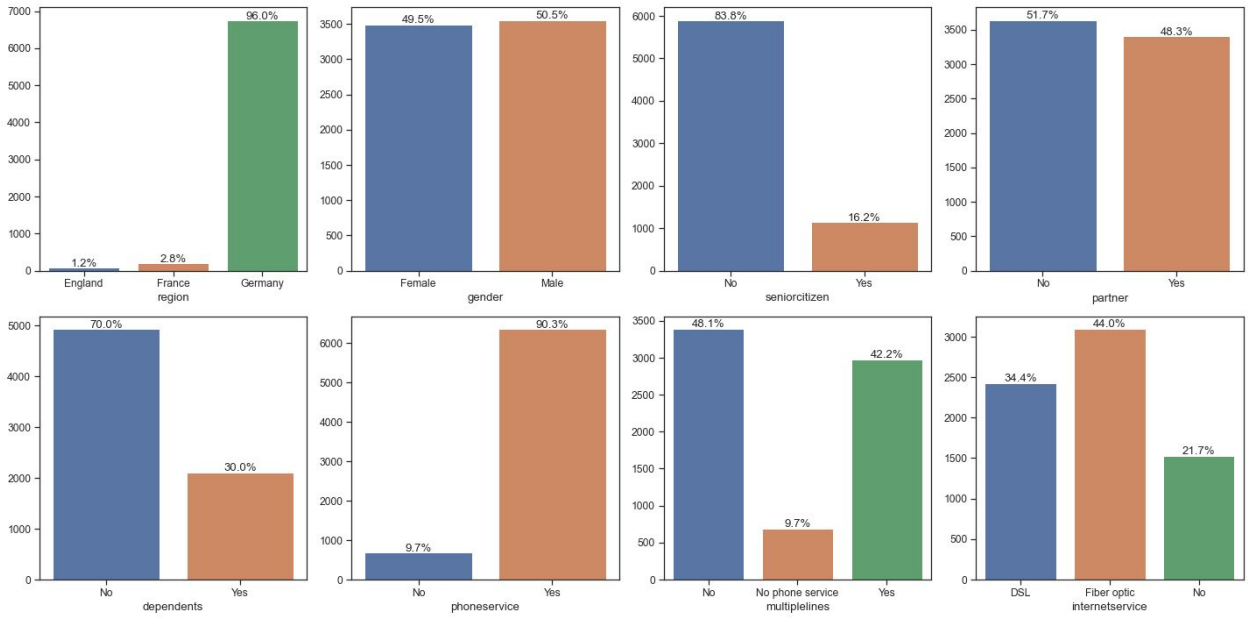
Numerical Features

	tenure	monthlycharges	totalcharges
count	7043.000000	7043.000000	7032.000000
mean	32.401107	64.751874	2294.275919
std	24.606849	30.097858	2316.761157
min	1.000000	12.000000	12.000000
25%	9.000000	35.500000	401.450000
50%	29.000000	70.350000	1397.475000
75%	55.000000	89.850000	3801.400000
max	120.000000	118.750000	22345.600000



- Tenure and monthly charges have a bimodal distribution. Meanwhile, total charges feature has highly right skewed distribution.
- There are many outliers on total charges feature.

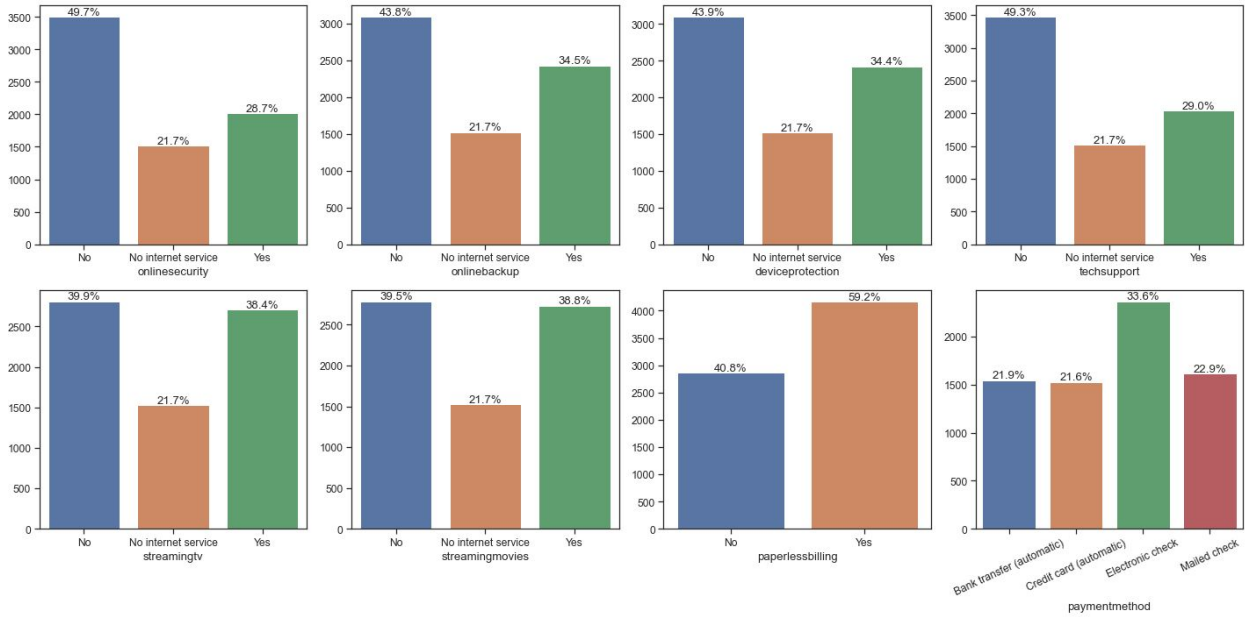
Categorical Features



From this visualization we got much information about our data.

- Most of customer are from Germany (96%).
- Proportion of Male and Female customer is about the same.
- Senior citizen is minority here (just 16.2%).
- Customer who do not have partner slighly have more proportion.

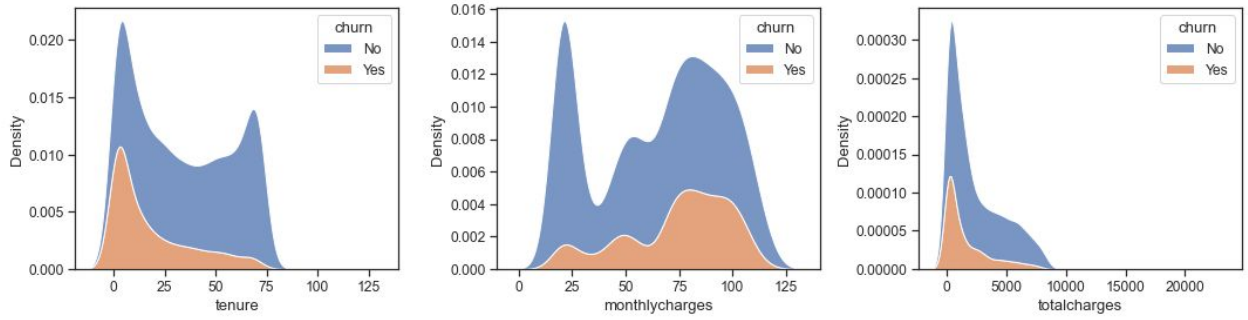
- Most of customer do not have dependents (70%).
- Most of customer has a phone service (90.3%).
- Customer who has multiple lines are slighly lower tha who don't.
- 44% of customers registered to Fiber Optic, 34.4% to DSL, and 21.7% who do not register to internet service.



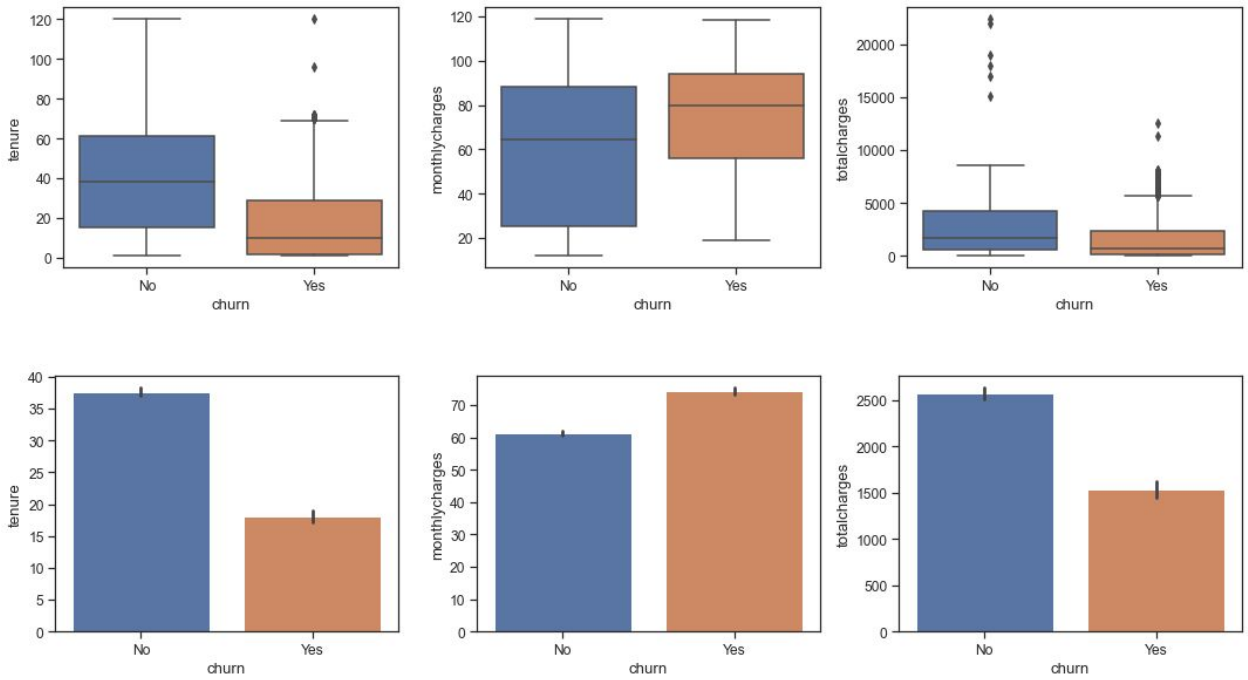
- Many customers don't have online security, online backup, device protection and tech support compared to those who do.
- Most of the customers use paper less billing and pay by electronic check.

Multivariate Analysis

Numerical Features



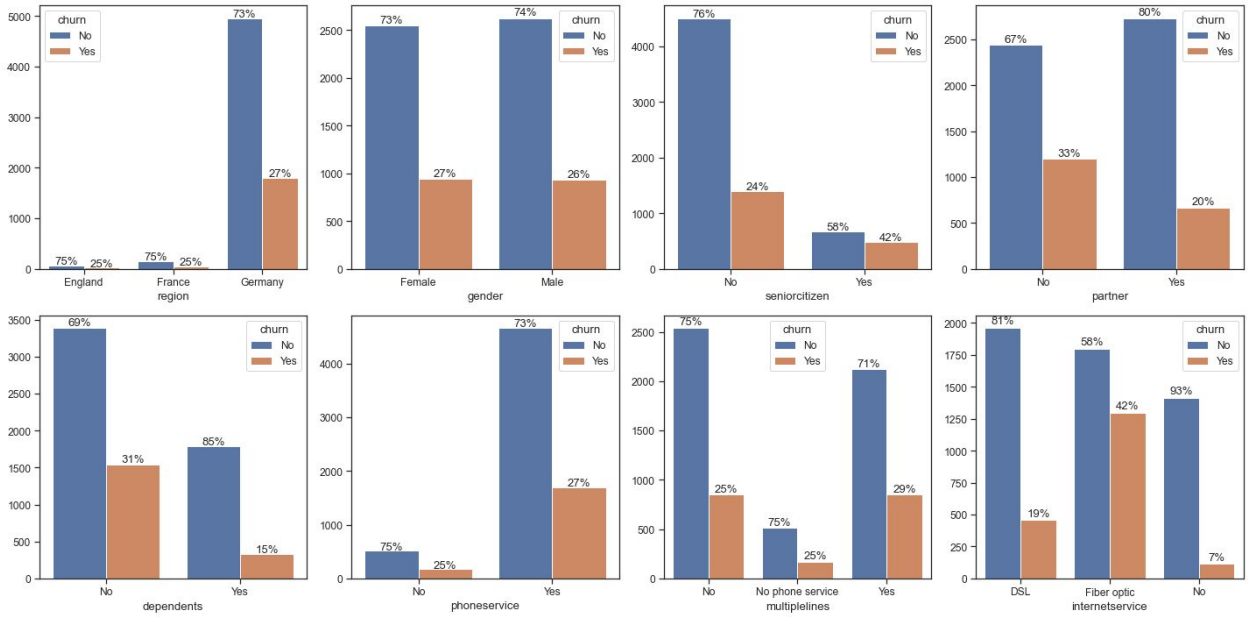
- In feature tenure and monthly charges, negative (non churn) class has a bimodal distribution, while positive class does not.
- As for total charges, the distribution between the two classes is similar.



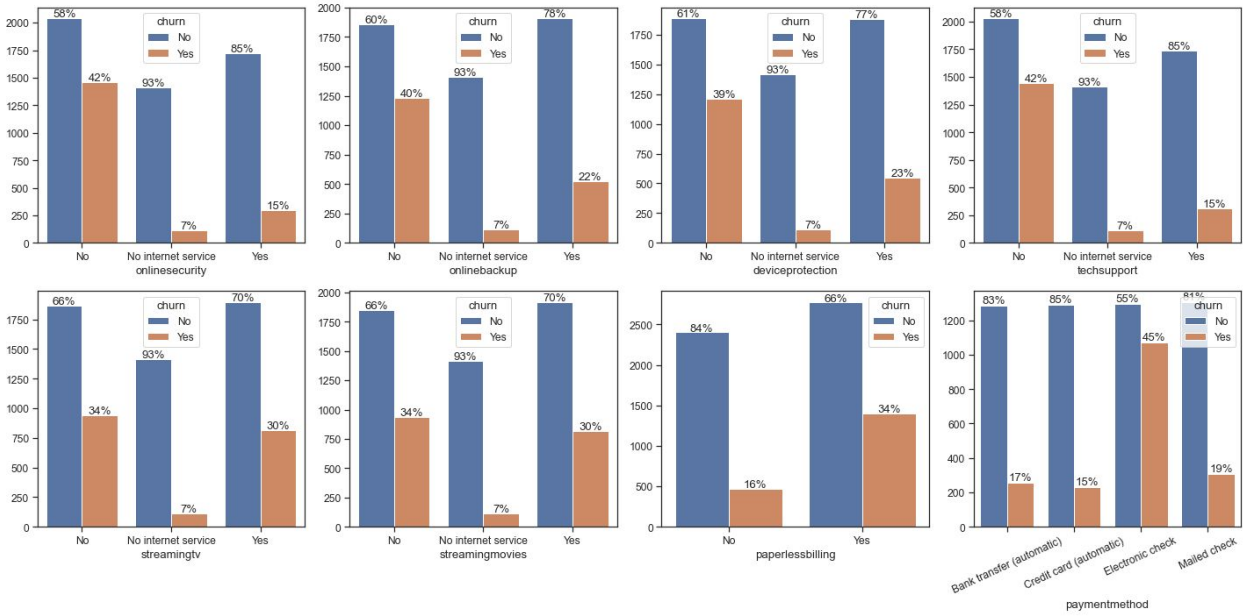
From the comparison of the independent and dependent numerical variables above on average it can be concluded:

- Customers with low tenure tend to have higher churn
- Customers with high monthly charges tend to have higher churn
- Customers with lower total charges tend to have higher churn

Categorical Features

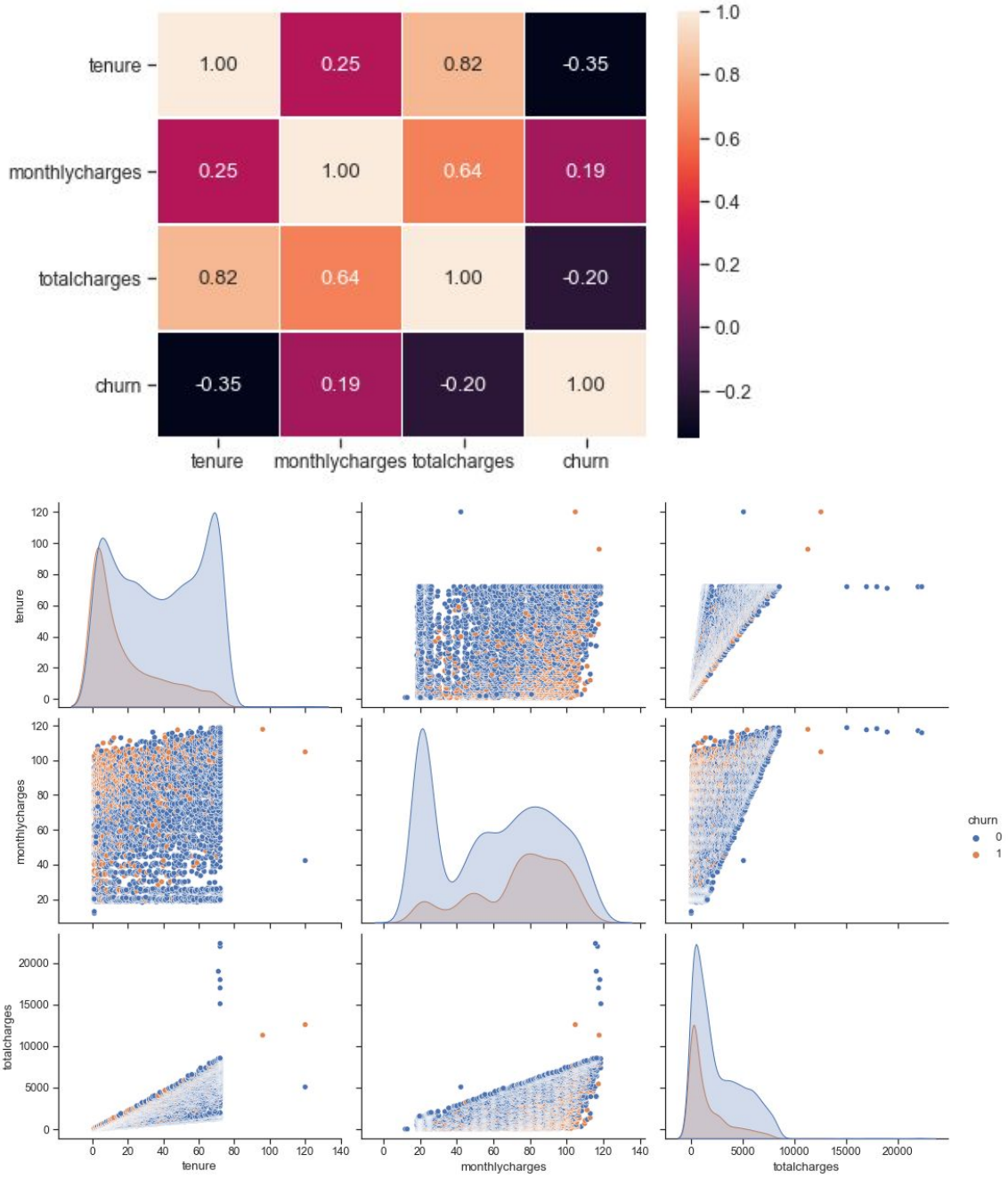


- From this comparation, the tendency for churn are more occurs on customers from the senior citizens, who do not have a partner, and who do not have dependents.
- In addition, customers who subscribe to fiber optic have a higher churn rate when compared to other services.



- Customers who do not subscribe to online security, online backup, device protection, and tech support have a higher churn rate when compared to customers who subscribe to this service.
- In streaming tv and streaming movies services, there is no significant difference between customers who churn and those who don't. (4% different)
- In terms of payment, it appears that customers who using paper less billing and electronic check payment methods have higher churn.

Features Correlation



- Tenure and total charges have a strong positive correlation.
- The monthly charges and total charges features have a positive correlation.
- Meanwhile, tenure and monthly charges have a weak positive correlation.

We can see some outliers that are very far away from the data set.

Judging from the data distribution, it is possible that these outliers were caused by input errors.

Outlier handling will be carried out at the data processing step after this.

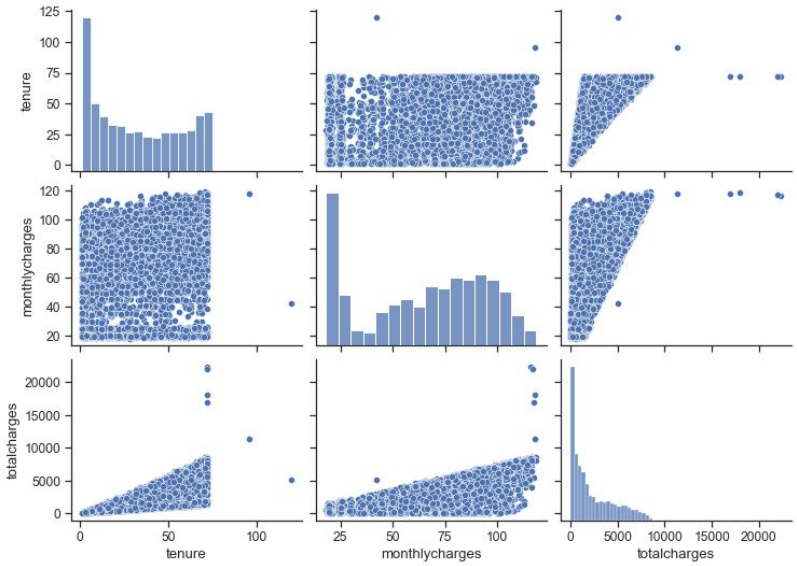
Data Processing

Outlier Handling

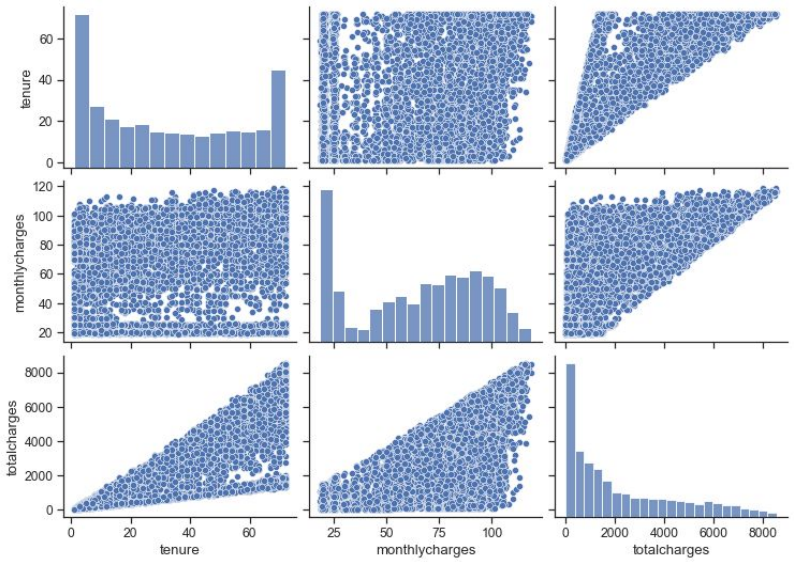
We will remove this outlier with a threshold of 10000 for total charges and 80 for tenure.

Outliers above those values will be discarded from the data set.

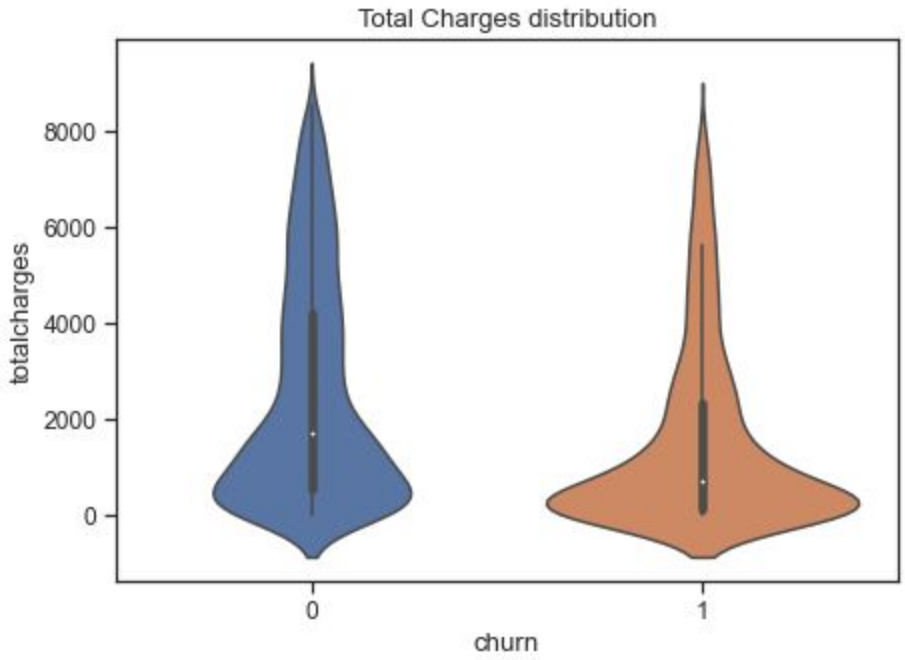
Before remove outlier



After remove outlier



Missing Value Handling



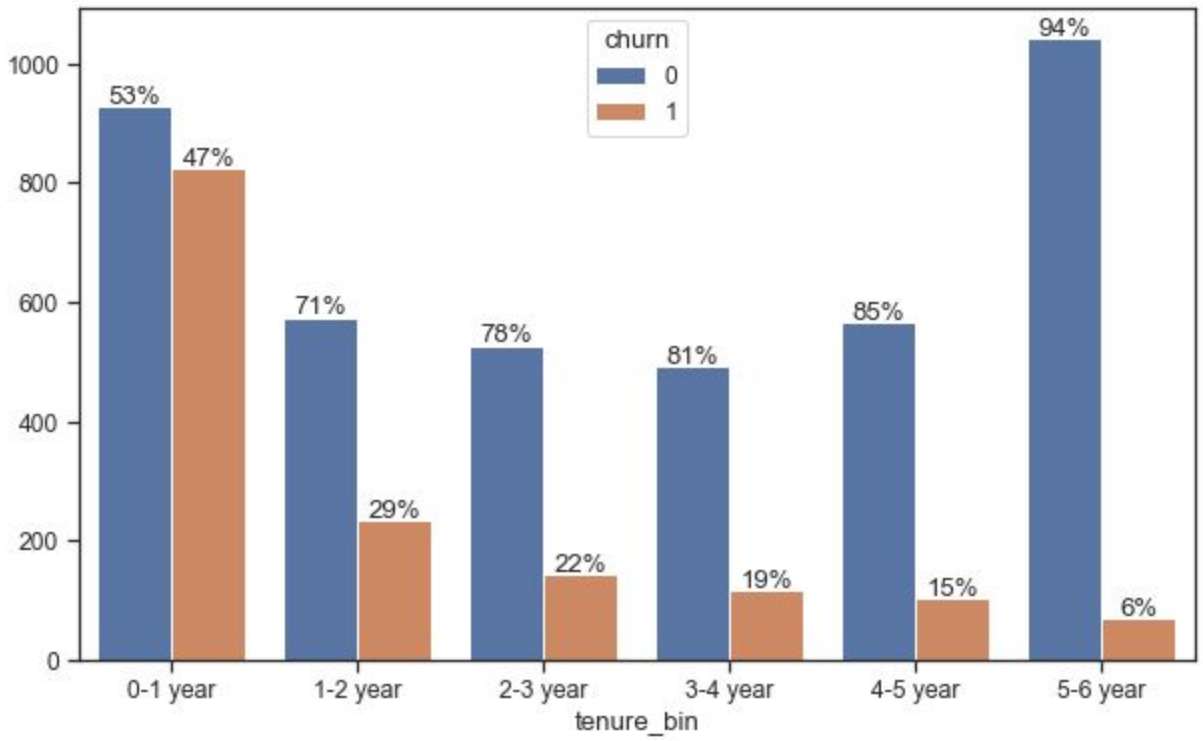
Because the total charges feature has a highly right skewed distribution, missing value imputation will be done with median value.

Feature Engineering

Make categorize (binning) for tenure features.

```
def tenure_bin(df):
  if df['tenure'] <= 12:
    return '0-1 year'
  elif df['tenure'] > 12 and df['tenure'] <= 24:
    return '1-2 year'
  elif df['tenure'] > 24 and df['tenure'] <= 36:
    return '2-3 year'
  elif df['tenure'] > 36 and df['tenure'] <= 48:
    return '3-4 year'
  elif df['tenure'] > 48 and df['tenure'] <= 60:
    return '4-5 year'
  elif df['tenure'] > 60:
    return '5-6 year'
```

```
train['tenure_bin'] = train.apply(lambda x: tenure_bin(x), axis=1)
test['tenure_bin'] = test.apply(lambda x: tenure_bin(x), axis=1)
```



With binning, we can more clearly see the distribution comparison on tenure features. Customers with tenure less than 1 year tend to have higher churn compared to the other.

Categorical Encoding

- Use binary encoding for columns with “Yes” and “No” value and gender column.
- Use ordinal encoding for tenure.
- Use one hot encoding for the rest.

Scaling

Use MinMaxScaler for scaling purposes because two of our models are linear base and distance base machine learning.

Modeling

At this modeling step we will use 4 machine learning, namely:

1. Logistic Regression
2. Decision Tree
3. K-Nearest Neighbors
4. XGBoost Classifier

We will compare the performance of several models.

Because we have imbalanced classes, we will focus on roc auc score with higher recall.

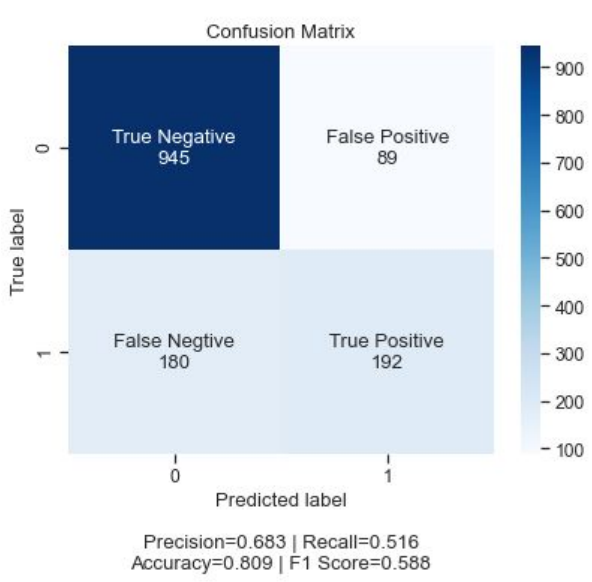
In situations where we want to detect instances of a minority class, we are usually concerned more so with recall than precision, as in the context of detection, it is usually more costly to miss a positive instance than to falsely label a negative instance.

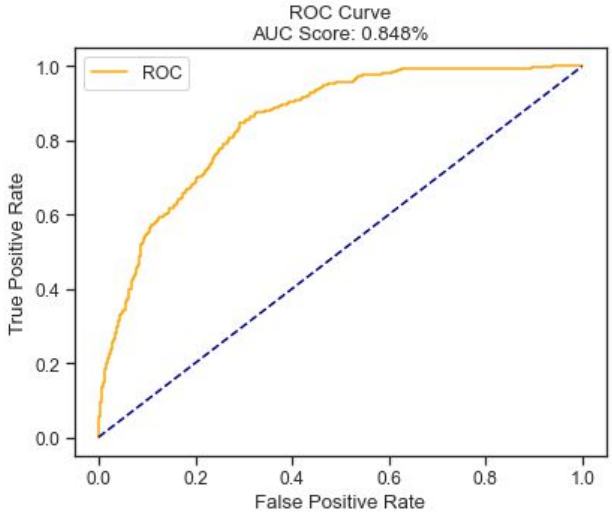
Model Fitting and Evaluation (Base Model)

Logistic Regression

Model Report on Train and CV Set:

Train Accuracy: 0.805615
Train AUC Score: 0.841049
CV AUC Score: Mean - 0.837321 | Std - 0.003221 | Min - 0.831449 | Max - 0.841139





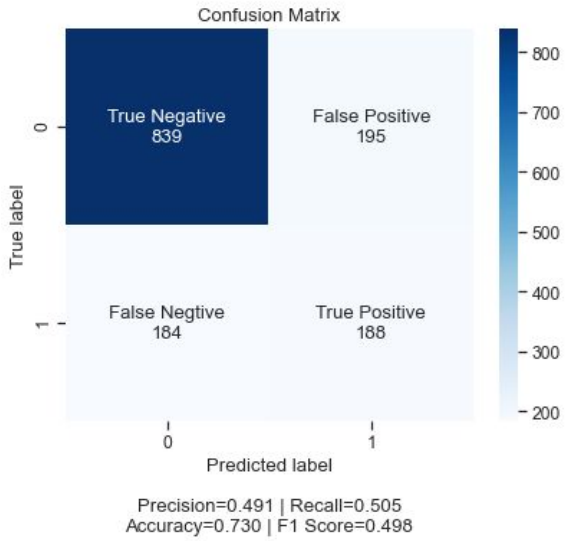
Decision Tree

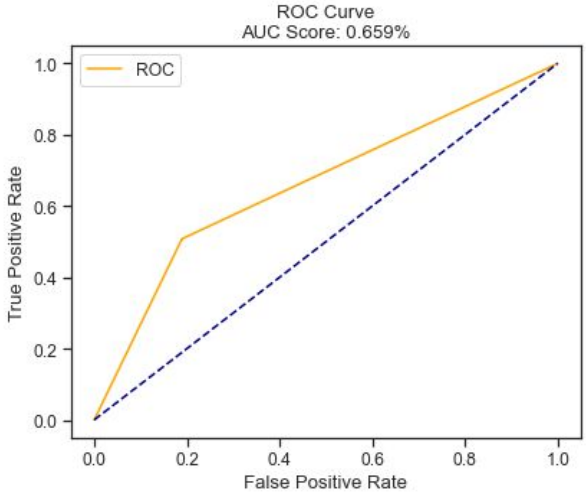
Model Report on Train and CV Set:

Train Accuracy: 0.998223

Train AUC Score: 0.999991

CV AUC Score: Mean - 0.647861 | Std - 0.016105 | Min - 0.625327 | Max - 0.667479





Precision=0.491 | Recall=0.505
Accuracy=0.730 | F1 Score=0.498

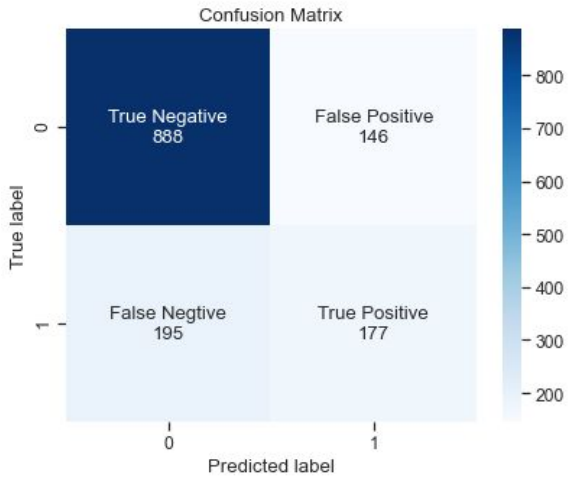
K-Nearest Neighbors

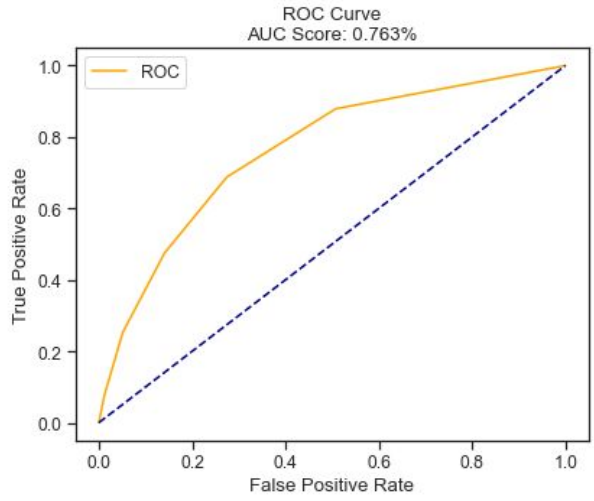
Model Report on Train and CV Set:

Train Accuracy: 0.832623

Train AUC Score: 0.890344

CV AUC Score: Mean - 0.751474 | Std - 0.004456 | Min - 0.744370 | Max - 0.757839





Precision=0.548 | Recall=0.476
Accuracy=0.757 | F1 Score=0.509

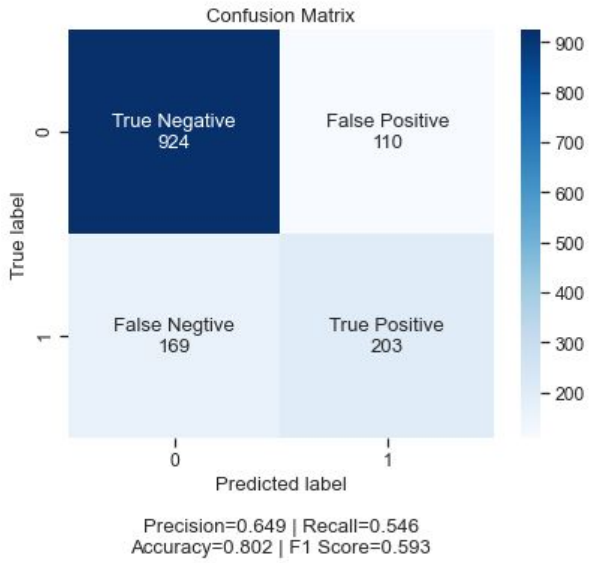
XGBoost Classifier

Model Report on Train and CV Set:

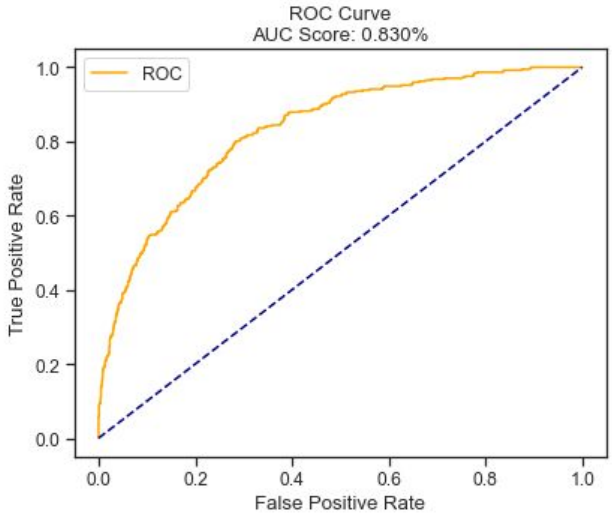
Train Accuracy: 0.936389

Train AUC Score: 0.983966

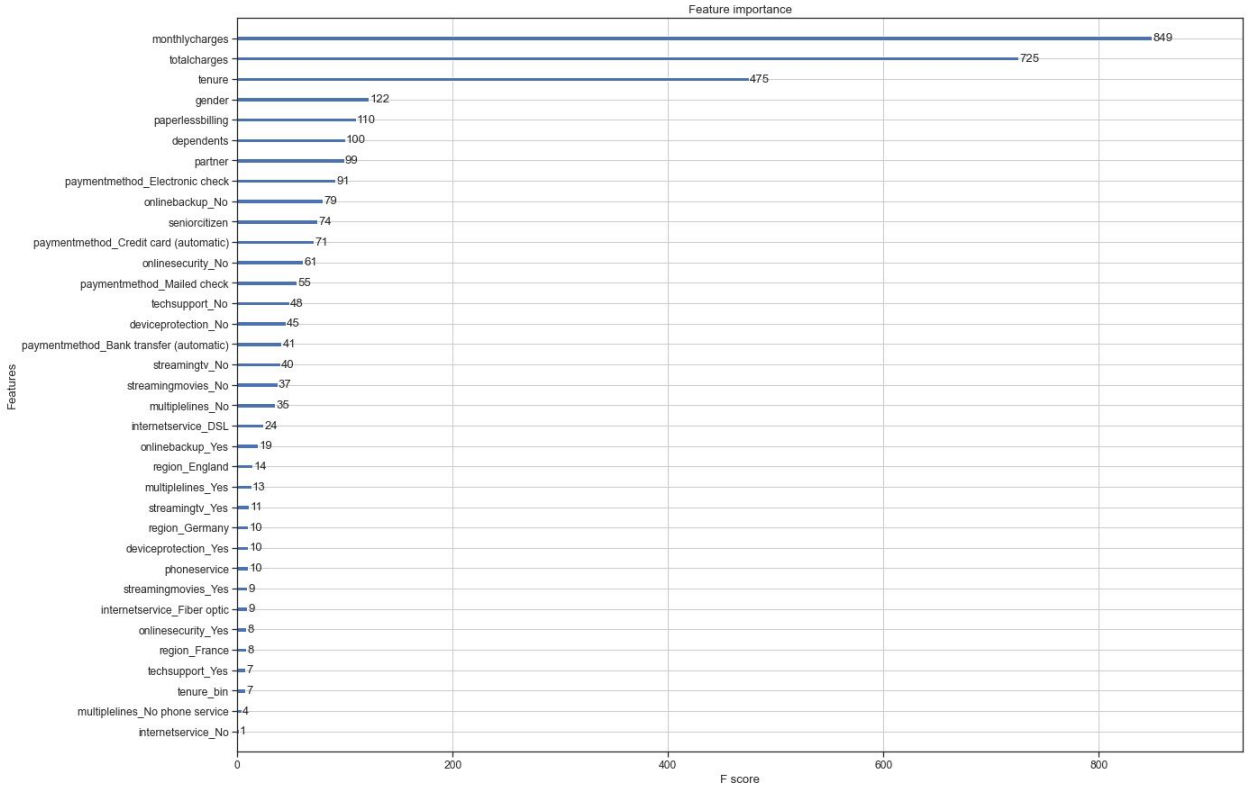
CV AUC Score: Mean - 0.808600 | Std - 0.009825 | Min - 0.794575 | Max - 0.820530



Precision=0.649 | Recall=0.546
Accuracy=0.802 | F1 Score=0.593



Feature Importance



Base Model Summary:

- Logistic Regression has pretty good results, neither overfit nor underfit (CV AUC 0.837), but recall is only 0.52
- Decision Tree is too overfit in the training set
- KNN results are quite good (slightly overfit on the training set), but the AUC score is still far below the Logistic Regression
- XGBoost results are similar to Logistic Regression with slightly superior CV AUC score of 0.839

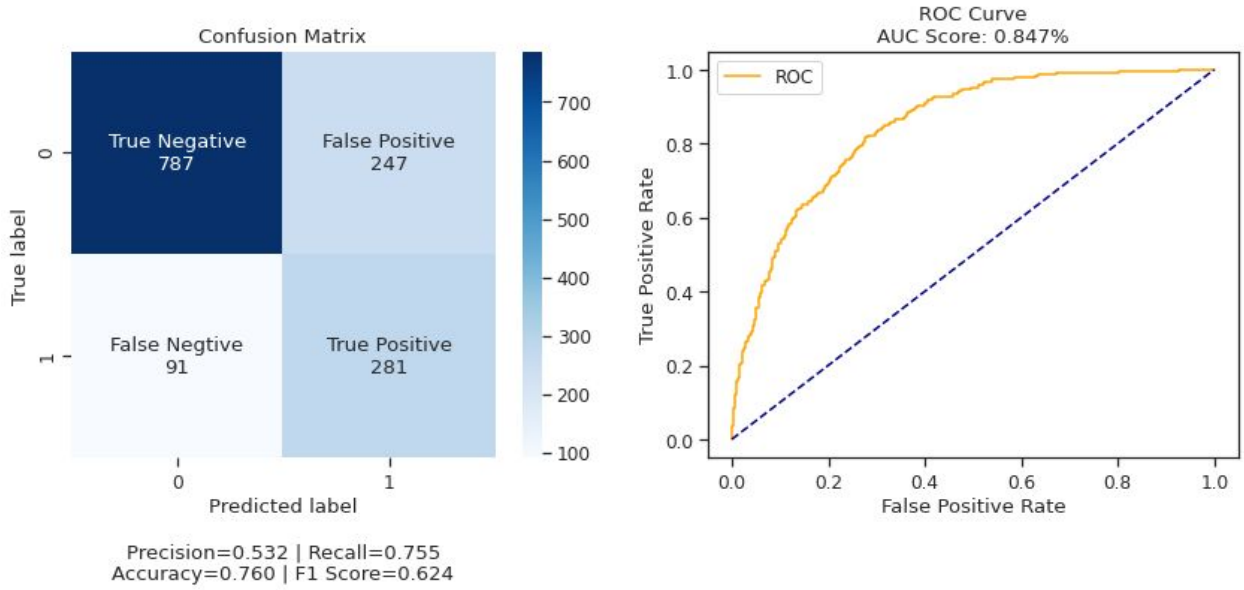
Without tuning, all models struggled to detect positive classes with an average recall around 0.5

Model with Weighted Parameter

Logistic Regression

Model Report on Train and CV Set:

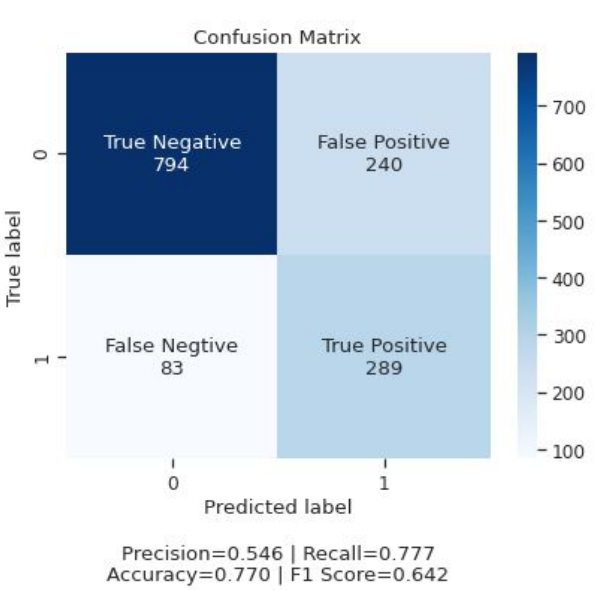
Train Accuracy: 0.756397
Train AUC Score: 0.842698
CV AUC Score: Mean - 0.839119 | Std - 0.003256 | Min - 0.833625 | Max - 0.843816

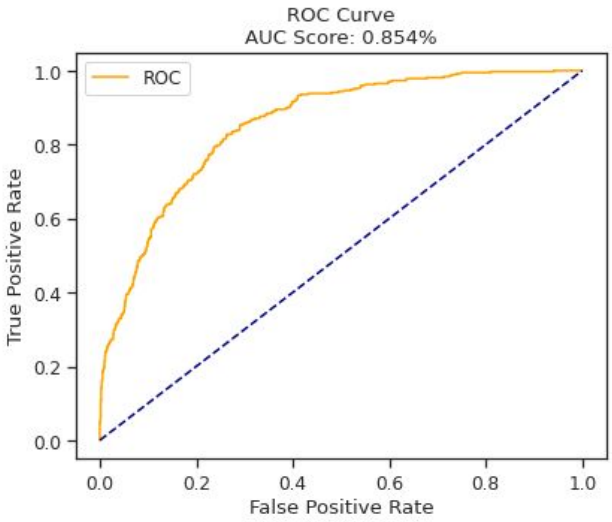


XGBoost Classifier

Model Report on Train and CV Set:

Train Accuracy: 0.767413
Train AUC Score: 0.864499
CV AUC Score: Mean - 0.840313 | Std - 0.005340 | Min - 0.830511 | Max - 0.846386





Summary:

By tuning the class weight parameter, we get a model that can give a higher recall but with a decrease in Accuracy.

- The results of Logistic Regression evaluation on the test set obtained Recall 0.755 and AUC Score 0.847.
- XGBoost was slightly higher with Recall 0.777 and AUC Score 0.854 on the test set.

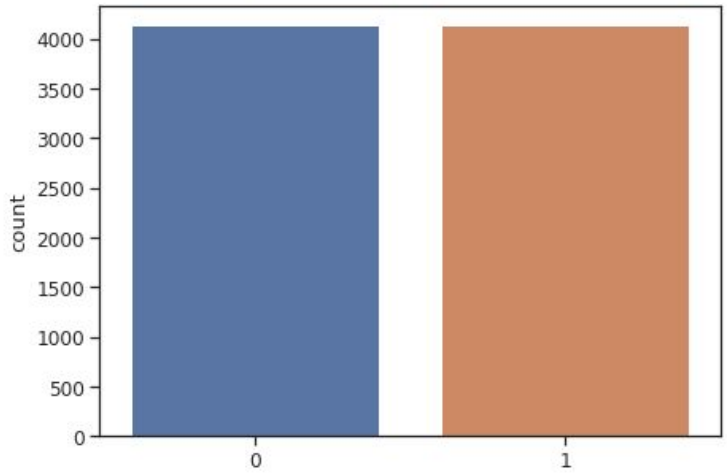
Model with Oversampling using SMOTE

```
from imblearn.over_sampling import SMOTE

oversample = SMOTE()
X_over, y_over = oversample.fit_resample(X_train.values, y_train.values)
```

```
sns.countplot(x=y_over)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f164c13b630>
```



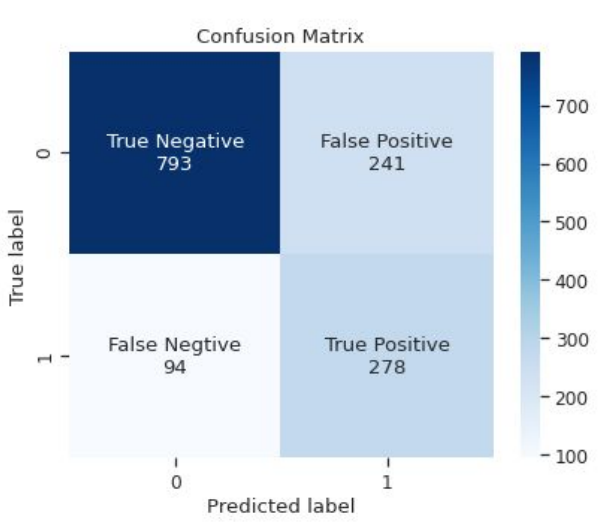
Logistic Regression

Model Report on Train and CV Set:

Train Accuracy: 0.764457

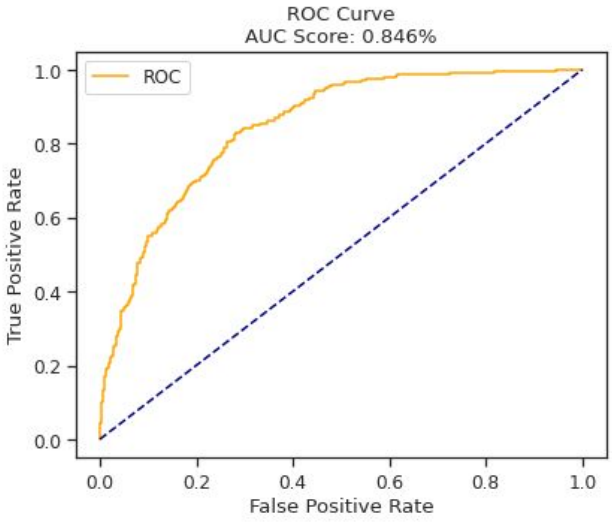
Train AUC Score: 0.846790

CV AUC Score: Mean - 0.844929 | Std - 0.011031 | Min - 0.832237 | Max - 0.859103



Precision=0.536 | Recall=0.747

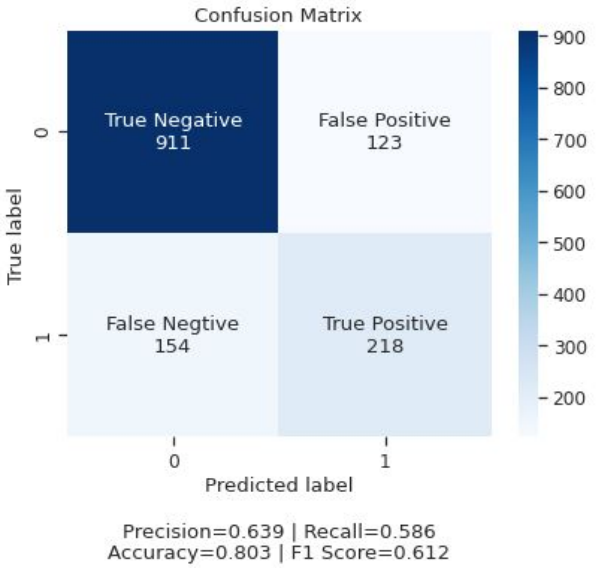
Accuracy=0.762 | F1 Score=0.624

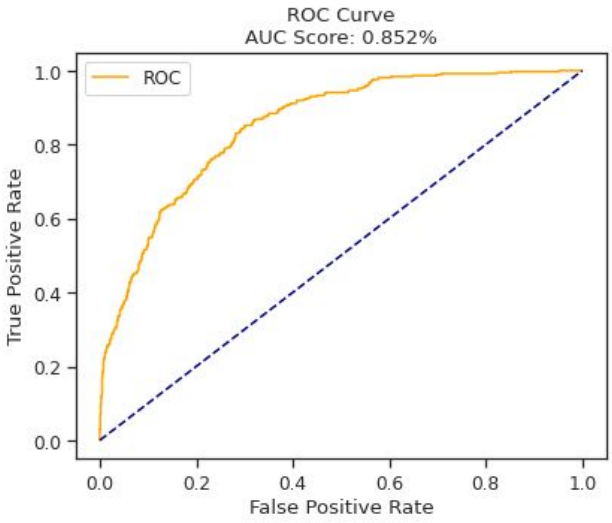


XGBoost Classifier

Model Report on Train and CV Set:

Train Accuracy: 0.866804
Train AUC Score: 0.945238
CV AUC Score: Mean - 0.936870 | Std - 0.069216 | Min - 0.831444 | Max - 0.993631





Summary:

- With the oversampling method, similar results are obtained for Logistic Regression compared to previous class weighted models.
- However, the XGBoost results were too overfit for the training and cross validation sets. The metrics result is less than what we expected because Recall is only 0.59

Final Model Recommendation

Our final model recommendation is XGBoost with Weighted Parameter, It has the highest scores with Recall 0.777 and AUC Score 0.854 on the test set.

Suggestions for next steps

- Try a different resampling method.
- Perform additional engineering features.
- Make modeling with different machine learning such as LightGBM or CatBoost.
- Perform stacking and advance hyperparameter tuning with bayesian optimization.

.