

Springboard Data Science Bootcamp – Capstone 2 – Predict Customer Churn



Objective:

Customer Churn measures the loss of customers and service provider companies use this metric to understand the customer retention. The objective is to predict behavior to retain customers by analyzing all relevant customer data and develop focused customer retention programs.

Problem:

Is the customer going churn?

Outcome:

When a customer stops service or company losing customer is referred to as Customer Churn. This is an important measure for any service-based company. The model predictions can provide the propensity of churning and gives the companies with the feature's importance that leads the customer to churn. With the list of potential customers who are likely to churn, the marketing/retention teams can then take measure to reduce their churn probability. This project helps companies in identifying customer who are at risk of churning and we have used this IBM sample data set provided for a telecom company. We will be using statistical analysis to understand variables that are associated with customer churn.

Dataset:

- 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

Data set contains 7043 rows and 21 columns, see below for more information:

- *customerID*: Customer ID
- *genderCustomer*: gender (female, male)
- *SeniorCitizen*: Whether the customer is a senior citizen or not (1, 0)
- *PartnerWhether*: the customer has a partner or not (Yes, No)
- *Dependents*: Whether the customer has dependents or not (Yes, No)
- *tenure*: Number of months the customer has stayed with the company
- *PhoneService*: Whether the customer has a phone service or not (Yes, No)
- *MultipleLines*: Whether the customer has multiple lines or not (Yes, No, No phone service)
- *InternetService*: Customer's internet service provider (DSL, Fiber optic, No)
- *OnlineSecurity*: Whether the customer has online security or not (Yes, No, No internet service)
- *OnlineBackup*: Whether the customer has online backup or not (Yes, No, No internet service)
- *DeviceProtection*: Whether the customer has device protection or not (Yes, No, No internet service)
- *TechSupport*: Whether the customer has tech support or not (Yes, No, No internet service)
- *StreamingTV*: Whether the customer has streaming TV or not (Yes, No, No internet service)
- *StreamingMovies*: Whether the customer has streaming movies or not (Yes, No, No internet service)
- *Contract*: The contract term of the customer (Month-to-month, One year, Two year)
- *PaperlessBilling*: Whether the customer has paperless billing or not (Yes, No)
- *PaymentMethod*: The customer's payment method (Electronic check, mailed check, Bank transfer (automatic), Credit card (automatic))
- *MonthlyCharges*: The amount charged to the customer monthly
- *TotalCharges*: The total amount charged to the customer
- *Churn*: Whether the customer churned or not (Yes or No)

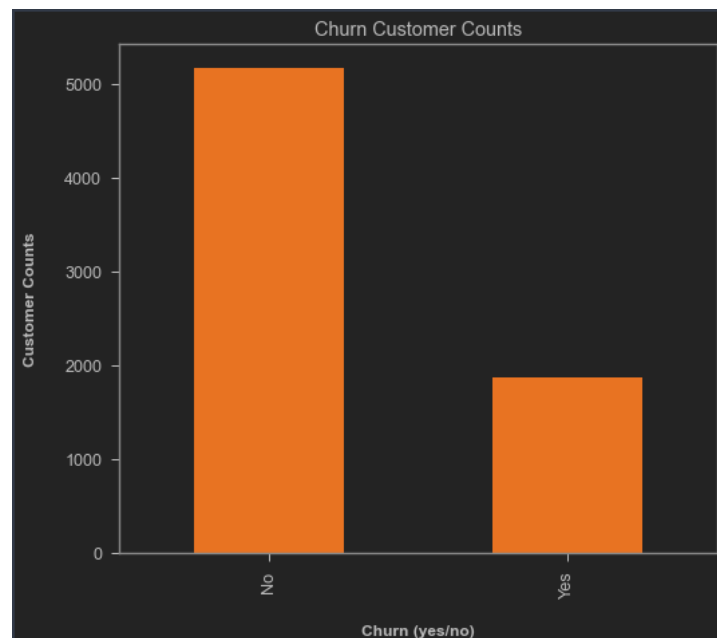
There are many categorical variables in this data set. The numerical features are Tenure, MonthlyCharges and TotalCharges.

Descriptive Stats:

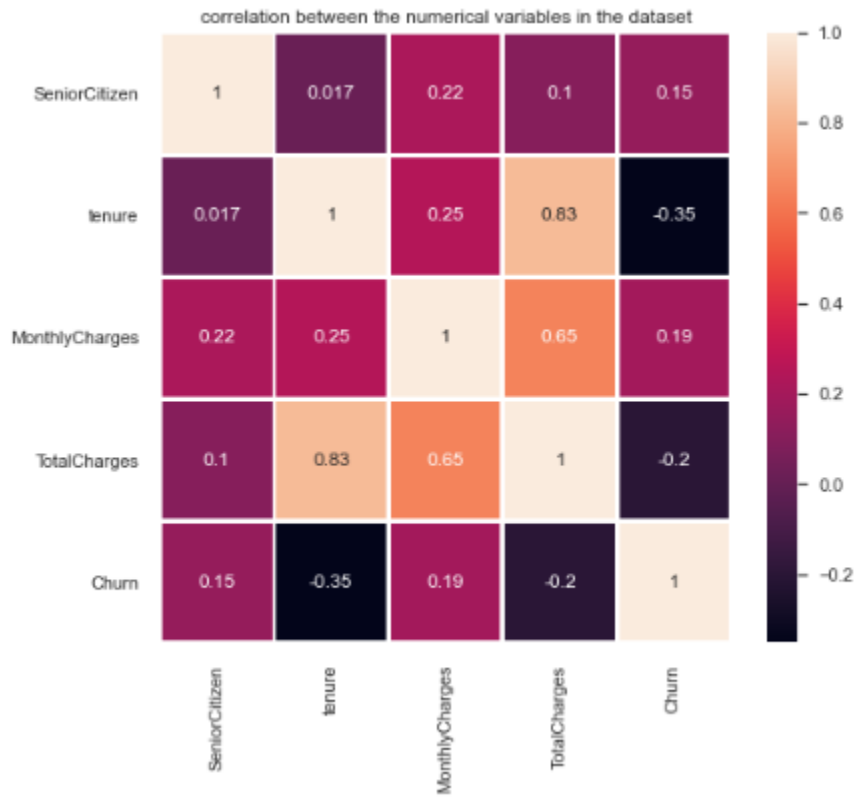
We see that Tenure ranges from 0 (new customer) to 6 years, Monthly charges range from \$18 to \$118, etc

tenure	MonthlyCharges
7043.000000	7043.000000
NaN	NaN
NaN	NaN
NaN	NaN
32.371149	64.761692
24.559481	30.090047
0.000000	18.250000
9.000000	35.500000
29.000000	70.350000
55.000000	89.850000
72.000000	118.750000

Roughly a quarter of the customers have churned in this data set.



Looking at the coloration matrix, there seems to be some positive correlation between Monthly Charges and Churn and some negative correlation between tenure and Churn.



Data Wrangling:

Replacing the Churn string value (yes/no) to numbers (0 – no and 1 – yes)

```
# converting churn to numerical variable for analyzing
df.loc[df.Churn=='No', 'Churn'] = 0
df.loc[df.Churn=='Yes', 'Churn'] = 1|
df['Churn'] = df['Churn'].astype(int)
```

Total Charges column was read as object, to fix this, we are converting its data type.

```
#Total charges is object in the data types converting to number
totalCharges = df.columns.get_loc("TotalCharges")
new_col = pd.to_numeric(df.iloc[:, totalCharges], errors='coerce')
df.iloc[:, totalCharges] = pd.Series(new_col)
```

Then investigate for missing values, it looks like TotalCharges has missing values.

```
print(df.isnull().values.any())
df.isnull().sum()
# Looks like Total charges has missing values
```

True

```
: customerID      0
   gender         0
   SeniorCitizen  0
   Partner        0
   Dependents     0
   tenure         0
   PhoneService   0
   MultipleLines   0
   InternetService 0
   OnlineSecurity  0
   OnlineBackup   0
   DeviceProtection 0
   TechSupport    0
   StreamingTV    0
   StreamingMovies 0
   Contract       0
   PaperlessBilling 0
   PaymentMethod  0
   MonthlyCharges 0
   TotalCharges   11
   Churn          0
dtype: int64
```

Applying imputation to fix the issue with missing values with means of TotalCharges using SimpleImputer from sklearn.impute.

```
# Handle missing values for nan_column (TotalCharges)
from sklearn.impute import SimpleImputer

# Find the column number for TotalCharges (starting at 0).
total_charges_idx = df.columns.get_loc("TotalCharges")
imputer = SimpleImputer(missing_values=np.nan, strategy='mean')

df.iloc[:, total_charges_idx] = imputer.fit_transform(df.iloc[:, total_charges_idx].values.reshape(-1, 1))
df.iloc[:, total_charges_idx] = pd.Series(df.iloc[:, total_charges_idx])
```

Identified the categorical, numerical and continuous features in the data set.

```
columns_idx = np.s_[0:] # Slice of first row(header) with all columns.
first_record_idx = np.s_[0] # Index of first record

string_fields = [fld for fld in df.iloc[first_record_idx, columns_idx]] # All string fields
all_features = [x for x in df.columns if x != 'Churn']
categorical_columns = list(np.array(df.columns)[columns_idx][string_fields])
categorical_features = [x for x in categorical_columns if x != 'Churn']
continuous_features = [x for x in all_features if x not in categorical_features]

print('All Features: ', all_features)
print('\nCategorical Features: ', categorical_features)
print('\nContinuous Features: ', continuous_features)
print('\nAll Categorical Columns: ', categorical_columns)

All Features: ['customerID', 'gender', 'SeniorCitizen', 'Partner', 'Dependents', 'tenure', 'PhoneService', 'MultipleLines', 'InternetService', 'OnlineSecurity', 'OnlineBackup', 'DeviceProtection', 'TechSupport', 'StreamingTV', 'StreamingMovies', 'Contract', 'PaperlessBilling', 'PaymentMethod', 'MonthlyCharges', 'TotalCharges']

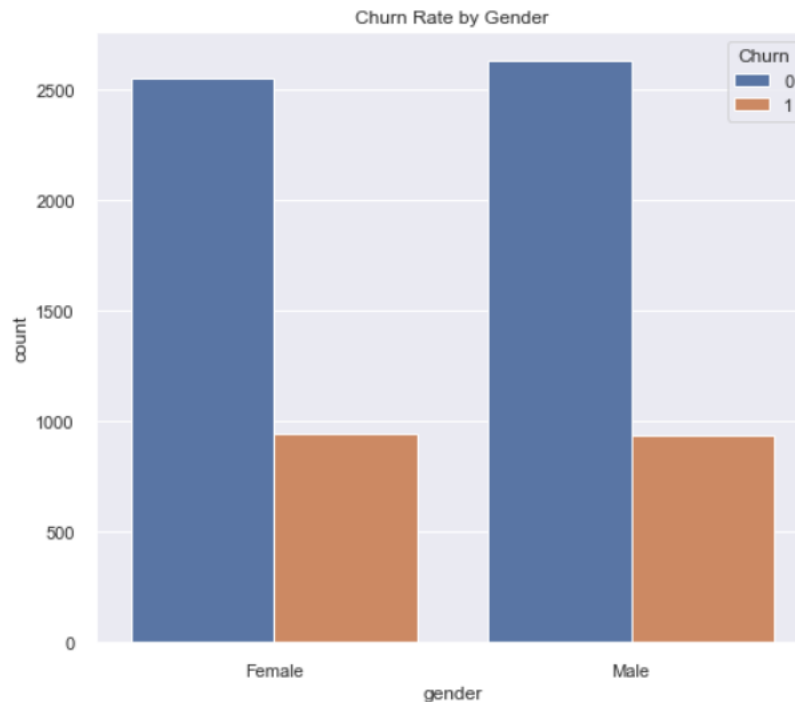
Categorical Features: ['customerID', 'gender', 'Partner', 'Dependents', 'PhoneService', 'MultipleLines', 'InternetService', 'OnlineSecurity', 'OnlineBackup', 'DeviceProtection', 'TechSupport', 'StreamingTV', 'StreamingMovies', 'Contract', 'PaperlessBilling', 'PaymentMethod']

Continuous Features: ['SeniorCitizen', 'tenure', 'MonthlyCharges', 'TotalCharges']

All Categorical Columns: ['customerID', 'gender', 'Partner', 'Dependents', 'PhoneService', 'MultipleLines', 'InternetService', 'OnlineSecurity', 'OnlineBackup', 'DeviceProtection', 'TechSupport', 'StreamingTV', 'StreamingMovies', 'Contract', 'PaperlessBilling', 'PaymentMethod']
```

EDA - Data visualization:

Looking at the churn rate by gender there seems to be a very little higher % of female customer churning than compared to male customers, however, I don't think this is a big enough difference to be considered.



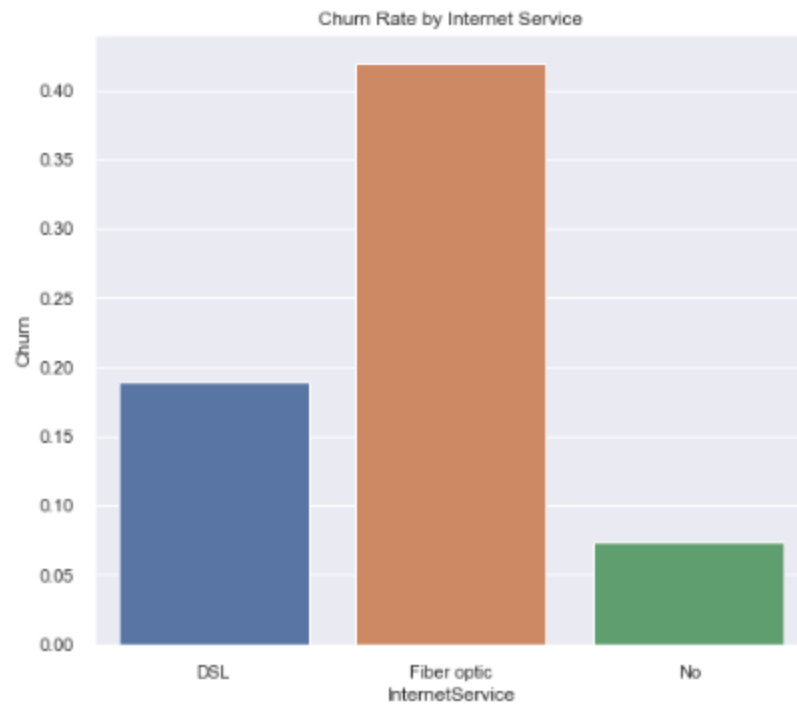
```
# plotting gender
mean_churn = df[['gender', 'Churn']].groupby('gender').mean()
print(mean_churn)
print("Female churn is little higher, but not that big difference")
```

gender	Churn
Female	0.269209
Male	0.261603

Female churn is little higher, but not that big difference

Looking at the churn rate by Internet service type there seems to be a high % of customers churning that are Fiber Optic service. This also could be due to Fiber optic being the most used service type.

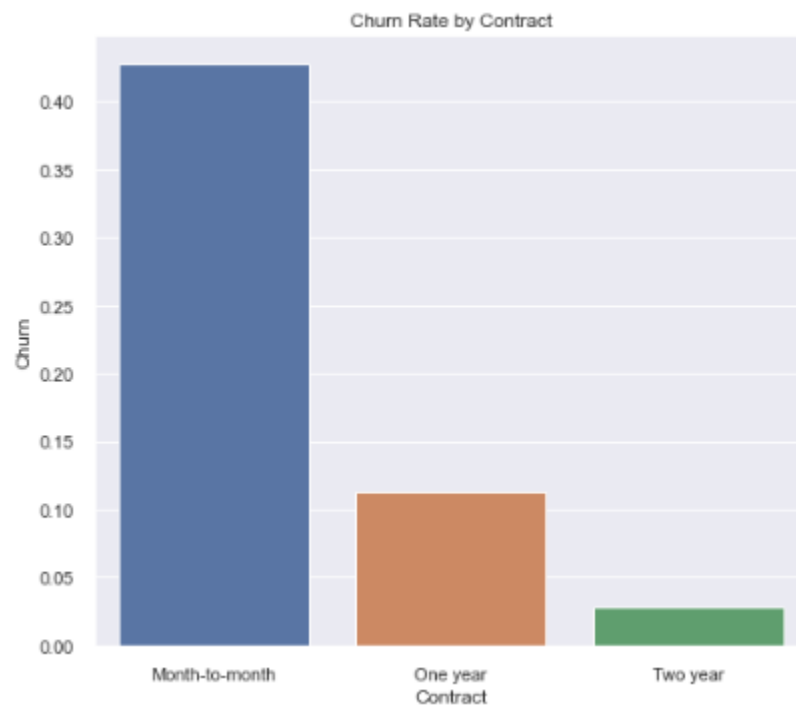
Fiber Optic customers are churning at a higher rate:



```
# plotting Internet Service type
isp_churn = df[['InternetService', 'Churn']].groupby('InternetService').mean()
print(isp_churn)
print("Fiber Optic customers are churning at a higher rate")
```

```
Churn
InternetService
DSL          0.189591
Fiber optic  0.418928
No           0.074050
Fiber Optic customers are churning at a higher rate
```

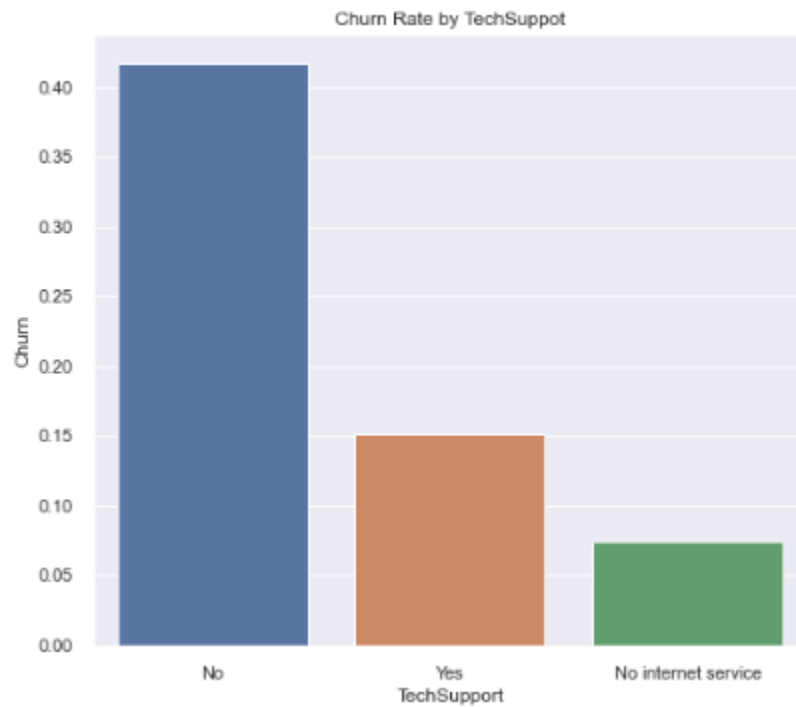
Month to Month subscribers are churning at a higher rate



```
# plotting Churn rate by Contract type|
cont_churn = df[['Contract','Churn']].groupby('Contract').mean()
print(cont_churn)
print("Month to Month subscribers are churning at a higher rate")
```

```
Churn
Contract
Month-to-month  0.427097
One year        0.112695
Two year        0.028319
Month to Month subscribers are churning at a higher rate
```

Customers who did not use Tech Support are churning at a higher rate

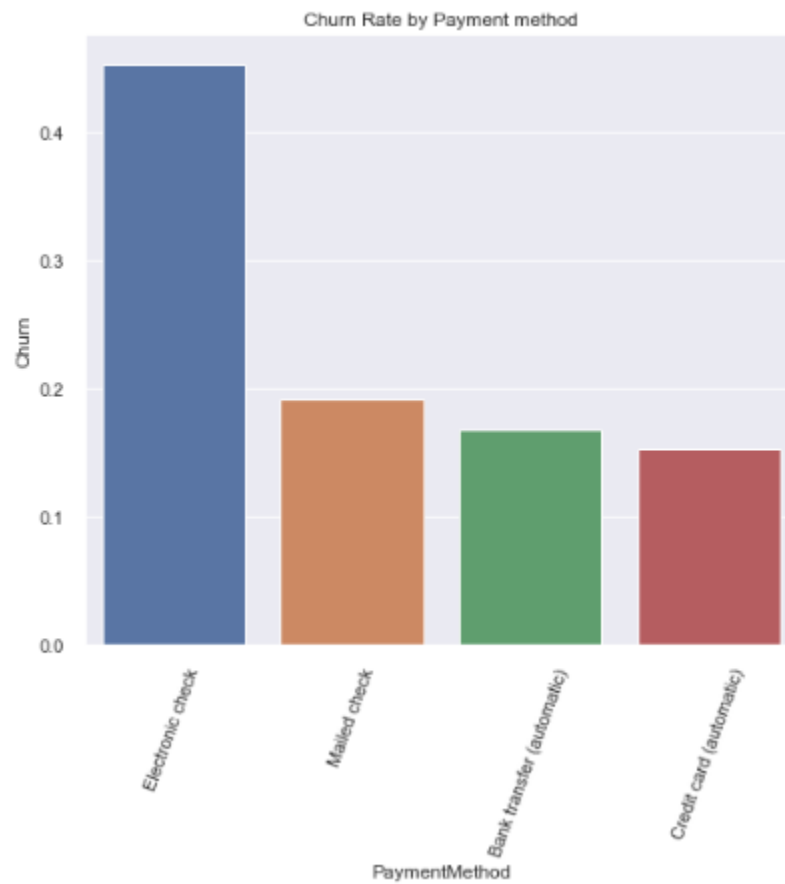


```
# plotting Churn rate by TechSupport
tech_churn = df[['TechSupport', 'Churn']].groupby('TechSupport').mean()
print(tech_churn)
print("Customers who did not use Tech Support are churning at a higher rate")
```

	Churn
TechSupport	
No	0.416355
No internet service	0.074050
Yes	0.151663

Customers who did not use Tech Support are churning at a higher rate

Customers who use checks as payment method are churning at higher rate

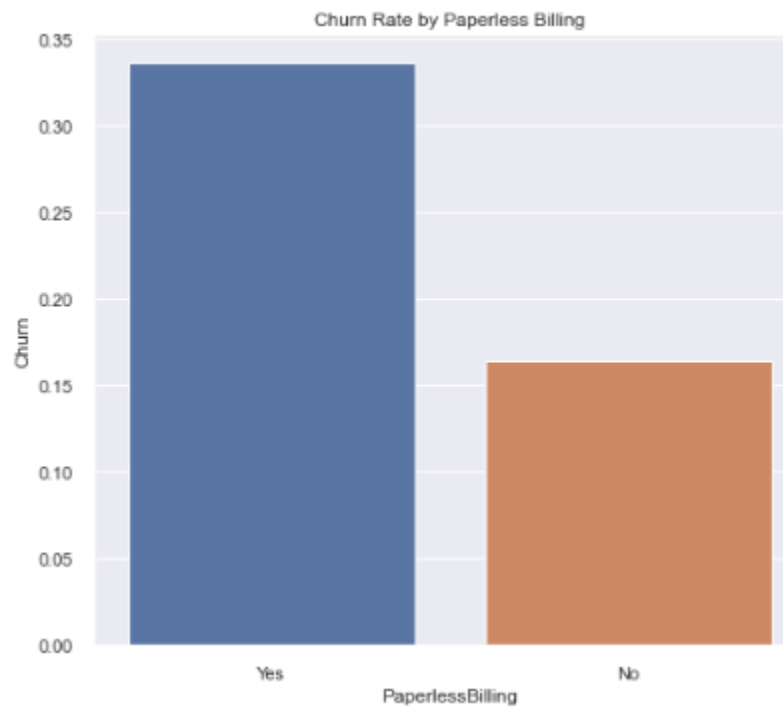


```
# plotting PaymentMethod
paym_churn = df[['PaymentMethod', 'Churn']].groupby('PaymentMethod').mean()
print(paym_churn)
print("Customers who use checks as payment method are churning at higher rate")
```

PaymentMethod	Churn
Bank transfer (automatic)	0.167098
Credit card (automatic)	0.152431
Electronic check	0.452854
Mailed check	0.191067

Customers who use checks as payment method are churning at higher rate

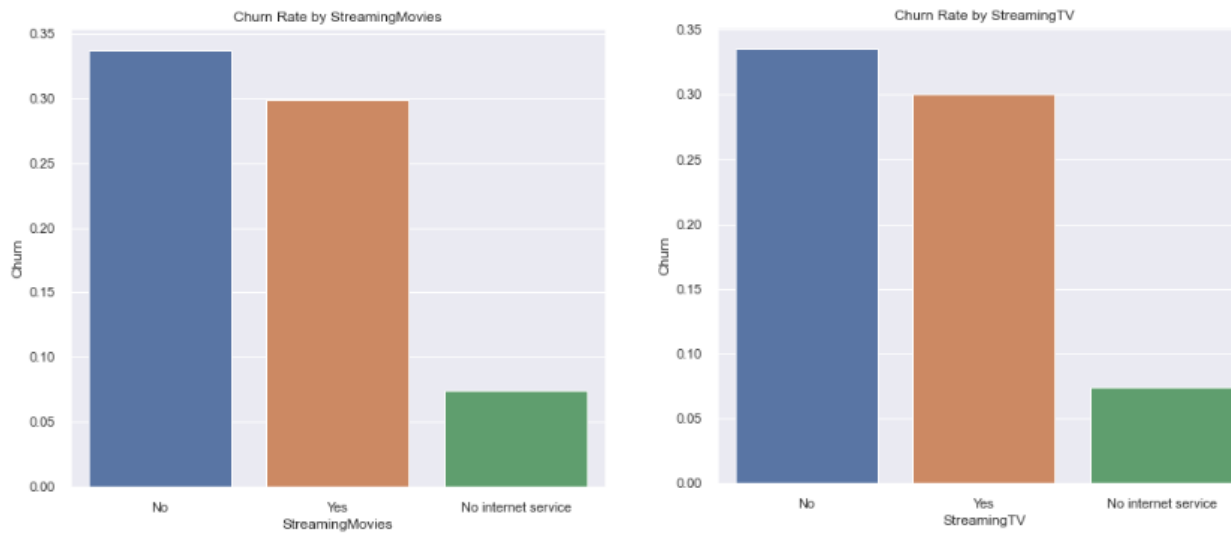
Customers who are on paperless billing have a higher churn rate. But this is could be very well because a lot of customers are enrolled into paperless billing.



```
# plotting PaperlessBilling
ebill_churn = df[['PaperlessBilling', 'Churn']].groupby('PaperlessBilling').mean()
print(ebill_churn)
print("Customers who are on paperless billing are churning at higher rate")
```

```
Churn
PaperlessBilling
No          0.163301
Yes         0.335651
Customers who are on paperless billing are churning at higher rate
```

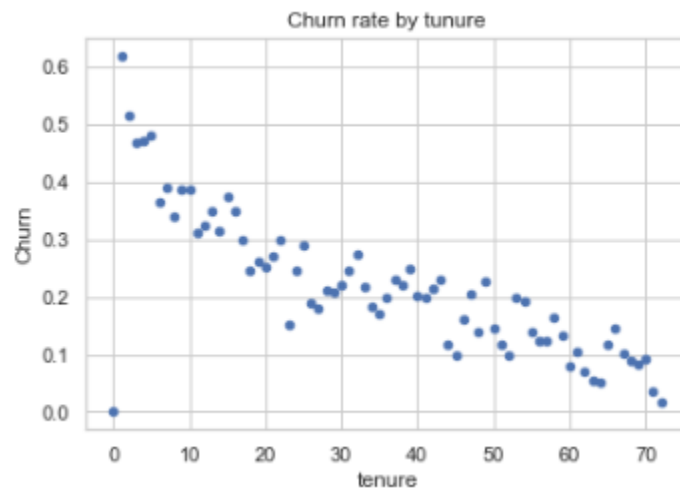
Customer who are Steaming movies vs Steaming tv seems to have very similar Churn rates:



Let's look at some summary stats for tenure variable:

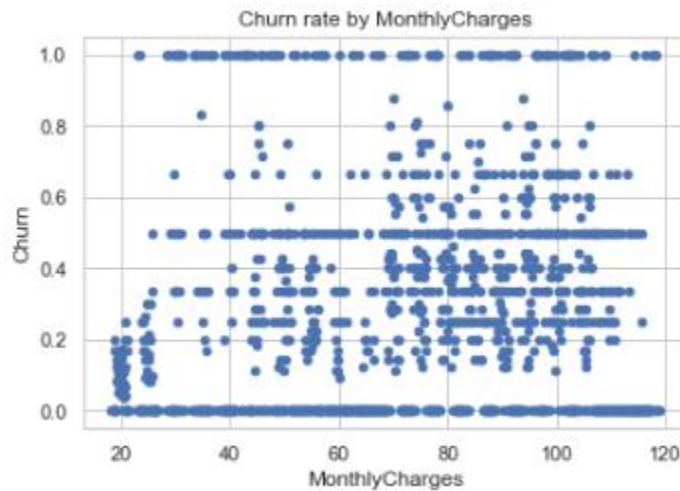
```
count    7043.000000
mean      32.371149
std       24.559481
min        0.000000
25%        9.000000
50%       29.000000
75%       55.000000
max       72.000000
Name: tenure, dtype: float64
```

Now, we will take a look at how mean churn rates are doing when compared with tenure.



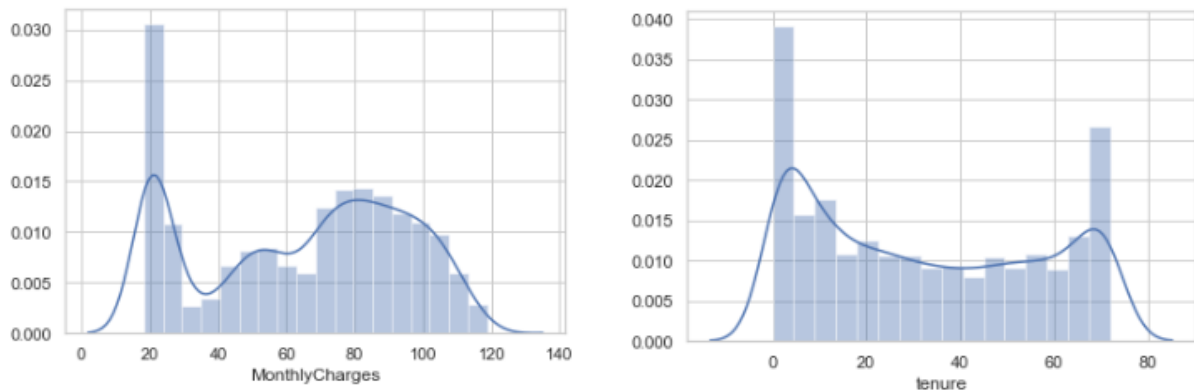
It shows that if the customer's tenure is long then churn rate is low.

Let's also look at other numerical variables such as MonthlyCharges:



There seems to be no relation for Churn and MonthlyCharges

Distribution plots for MonthlyCharges and tenure:



Monthly Charges seems to be roughly normal distribution and Tenure Distribution seems to be high at the ends, so a portion of the customers have either had lowest and highest tenure periods.

EDA Summary:

- More customers are using Fiber Optic for Internet Service have left the company than compared to DSL.
- Customers who do not use online security have left the company.
- Customers not using technical support have left the company.
- Customers who pay month to month are the most who leave the company.
- Customer's gender has almost equal rates of churn between them.
- The Monthly Charges for customers who churned tends to pay higher monthly fees than those that stay.
- Customers that churn tend to be relatively new customers when looking at tenure distribution.

Data Pre-processing:

- Categorical data label encoding
- Machine learning works with only numerical values. Therefore, we need to convert our categorical values to numerical values. By using the Pandas function “get_dummies()”, we can replace the gender column with “gender_Female” and “gender_Male”. We will use df.info() to show us which ones are categorical and numerical.

```
#import Label Encoder
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
dummy_columns = [] #array for multiple value columns

for column in df.columns:
    if df[column].dtype == object and column != 'customerID':
        if df[column].nunique() == 2:
            #apply Label Encoder for binary ones
            df[column] = le.fit_transform(df[column])
        else:
            dummy_columns.append(column)

#apply get dummies for selected columns
df_clean_data = pd.get_dummies(data = df, columns = dummy_columns)
```

- Dropping unwanted variables and target variable
- In our dataset, we can see that customer ID is not needed for our model, so we drop the variable. We do not need to treat missing values as there are none in this dataset.

```
# Dropping variables that are not needed and also target variables
X = df_clean_data.drop(['Churn', 'customerID'], axis=1)

# target variable
y = df_clean_data['Churn']
```

- Train on one set and test and measure on the test set.
- Our model needs to be trained; second our model needs to be tested. Therefore, it is best to have two different datasets.

- Splitting the data 70% for training and 30% for testing.

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    test_size=0.3,
                                                    random_state=123,
                                                    stratify=y)
```

- Realized that the data set is facing a class imbalance so apply Under Sampling technique to standardize the dataset.

```
# import the StandardScaler
from sklearn.preprocessing import StandardScaler

# initiate a StandardScaler and fit to training feature set.
scaler = StandardScaler().fit(X_train)

# Scale the training data using transform
X_train = scaler.transform(X_train)

from imblearn.under_sampling import RandomUnderSampler

# initiate a random under sampler
rus = RandomUnderSampler()

# Store in new random undersampled variables
X_train_rus, y_train_rus = rus.fit_sample(X_train, y_train)
```

Machine Learning:

The customer churn is a classification problem and have choose the following classification models to predict if a customer churns by fitting the training data set and testing on the test set. Used scikit-learn and other libraries in the python machine learning world.

1. Logistic Regression
2. Decision Tree model
3. Random Forest Classifier
4. Extreme gradient boosting (xGBoost)

Model performance metrics

Accuracy = Pct of correctly classified predictions (i.e, churn and non-churn)

Precision = Pct of total positive class predictions that were correctly classified

Recall = Pct of total positive class samples that were correctly classified

Logistic Regression:

This type of statistical classification modelling is commonly used for predicting binary variables. It models the log odds of the probability of the target variable.

The accuracy train accuracy and the testing accuracy was around 80%

True positives (TP) - Predicted correctly that customer will churn - 446

True negative (TN) - Predicted correctly that customer would not churn - 1145

False positives (FP) - The model incorrectly predicted that they would churn when in reality they did not - 407

False negative (FN) - the model incorrectly predicted that customer will not churn when in reality they did churn - 115

The models seem to have maintained recall rate between training and test models.

Model:

```
#Initialize the log reg model instance
lr = LogisticRegression()

# fit the training data
lr.fit(X_train_rus, y_train_rus)

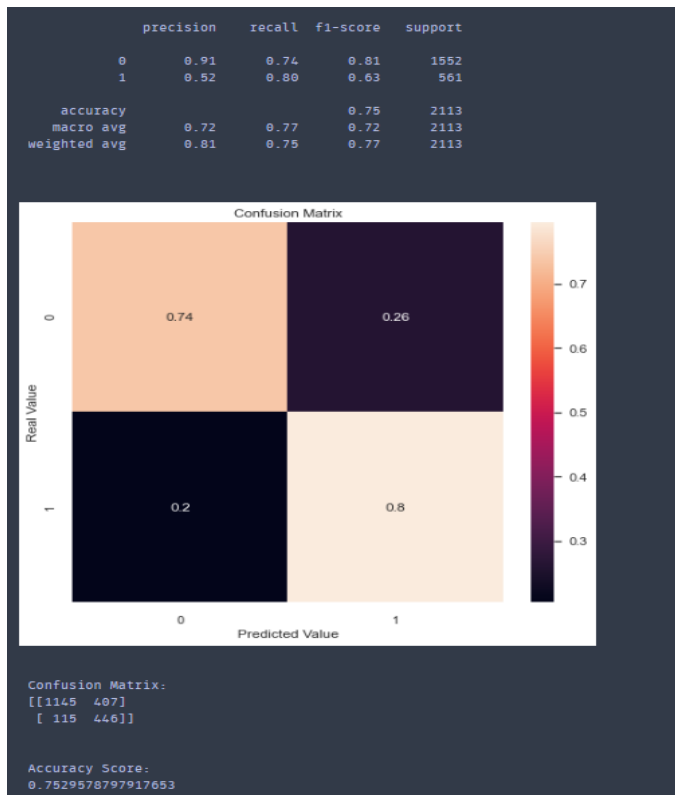
# Measure the model accuracy
X_test_lr = scaler.transform(X_test)
y_pred_test = lr.predict(X_test_lr)

# Generate Classification report
print(classification_report(y_test, y_pred_test))

# confusion matrix
fig, ax = plt.subplots()
sns.heatmap(confusion_matrix(y_test, y_pred_test, normalize='true'), annot=True, ax=ax)
ax.set_title('Confusion Matrix')
ax.set_ylabel('Real Value')
ax.set_xlabel('Predicted Value')
plt.show()

print("Confusion Matrix:")
print(metrics.confusion_matrix(y_test, y_pred_test))
print('\n')
print("Accuracy Score:")
```

Evaluation metrics:



Decision Tree model:

First tree-based model that splits the data multiple times based on the cutoff values defined in the features.

Model parameters: criterion = gini; max_depth=5;random_state=1

The accuracy train accuracy and the testing accuracy was around 79%

TP=1374

TN=303

FP=178

FN=258

The models seem to decent accuracy and precision and recall rates between train and test data set.

```

#from sklearn.tree import DecisionTreeClassifier
#from sklearn.metrics import accuracy_score
# # Instantiate a DecisionTreeClassifier 'dt' with a maximum depth of 6
dt = DecisionTreeClassifier(criterion = 'gini', max_depth=5, random_state=1)

# Fit dt to the training set
dt.fit(X_train_rus,y_train_rus)

# Predict test set labels
X_test_dt = scaler.transform(X_test)
y_pred_dt_test = dt.predict(X_test_dt)

# classification report
print(classification_report(y_test, y_pred_dt_test))

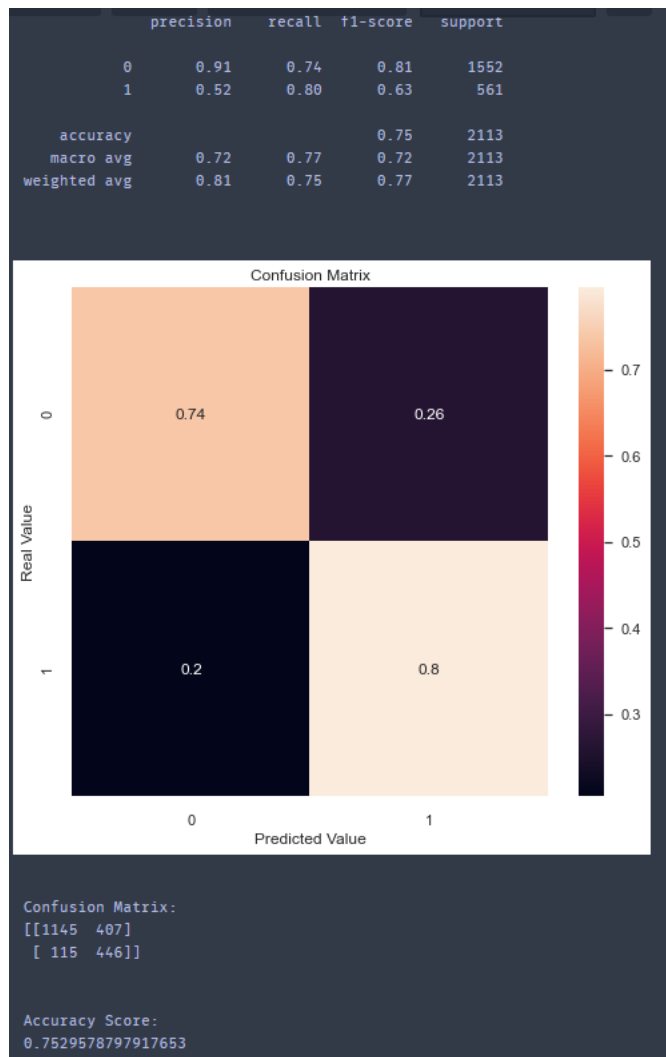
# confusion matrix
fig, ax = plt.subplots()
sns.heatmap(confusion_matrix(y_test, y_pred_dt_test, normalize='true'), annot=True, ax=ax)
ax.set_title('Confusion Matrix')
ax.set_ylabel('Real Value')
ax.set_xlabel('Predicted Value')

plt.show()

print("Confusion Matrix:")
print(metrics.confusion_matrix(y_test, y_pred_dt_test))
print('\n')
print("Accuracy Score:")
print(metrics.accuracy_score(y_test, y_pred_test))

```

Evaluation metrics:



Random Forest

This is the second tree-based classification models we have experiments in the model phase. The difference between the previous model is it using bagging to define the boundaries in Random Forest models.

Model parameters include `n_estimators=50`; `criterion=entropy`
`min_samples_split=285`

The accuracy train accuracy and the testing accuracy was around 79%

The models seem to have low recall rate and higher precession rate.

The model predicted more False Negatives compared to the previous Decision tree model.

```
#from sklearn import metrics
#from sklearn.ensemble import RandomForestClassifier

# Start an instance of RandomForestClassifier
RF = RandomForestClassifier(n_estimators = 50, criterion = 'entropy', min_samples_split = 285)

# fit the model to training data
RF.fit(X_train, y_train)

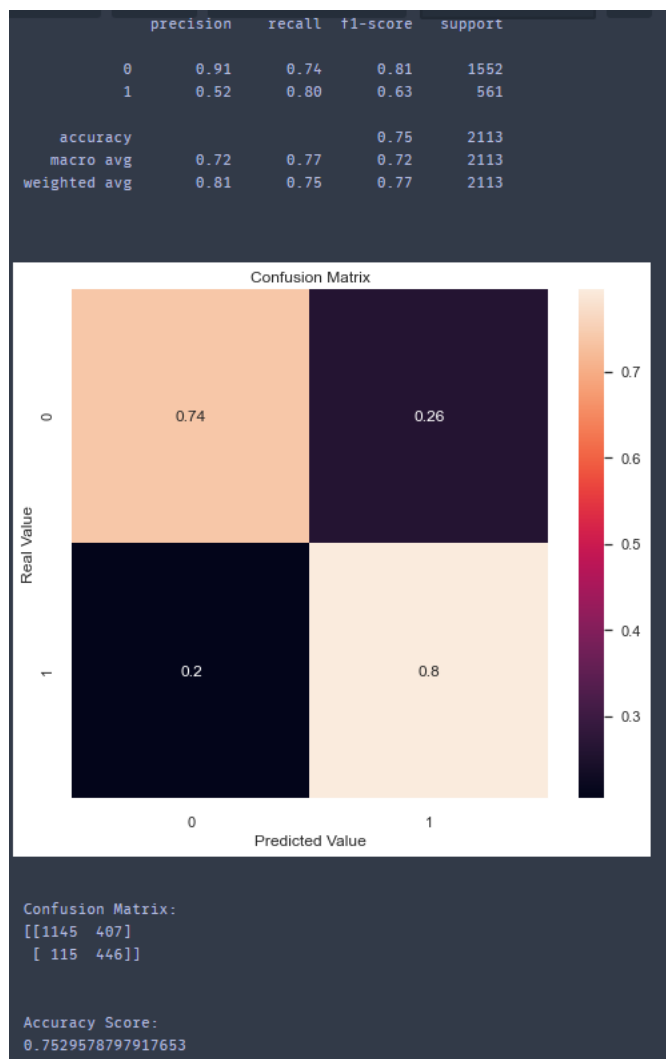
# predict the labels
X_test_rf = scaler.transform(X_test)
y_pred_rf_test = RF.predict(X_test_rf)

# classification report
print(metrics.classification_report(y_test, y_pred_rf_test))

# confusion matrix
fig, ax = plt.subplots()
sns.heatmap(confusion_matrix(y_test, y_pred_rf_test, normalize='true'), annot=True, ax=ax)
ax.set_title('Confusion Matrix')
ax.set_ylabel('Real Value')
ax.set_xlabel('Predicted Value')

plt.show()
```

Evaluation metrics:



Feature importance:

```
(0.1879207270248737, 'Contract_Month-to-month')
(0.11284650561596918, 'Contract_Two year')
(0.10806257606188054, 'OnlineSecurity_No')
(0.10471715403124655, 'tenure')
(0.07692005102945117, 'TotalCharges')
(0.06968666921458776, 'TechSupport_No')
(0.05234377185200279, 'MonthlyCharges')
(0.039030635347902326, 'InternetService_Fiber optic')
(0.03319961592145481, 'PaymentMethod_Electronic check')
(0.030193193411427354, 'OnlineBackup_No')
(0.026137524973344362, 'Contract_One year')
(0.02172726731833357, 'InternetService_DSL')
(0.020071852660975175, 'InternetService_No')
(0.014122446762522518, 'StreamingMovies_No internet service')
(0.009771629958891456, 'OnlineBackup_No internet service')
(0.009043407444789216, 'OnlineSecurity_No internet service')
(0.008323410990405023, 'OnlineSecurity_Yes')
(0.008133091349550465, 'TechSupport_Yes')
(0.007402325649217218, 'OnlineBackup_Yes')
(0.006298065650423189, 'DeviceProtection_No internet service')
(0.006004197845796403, 'TechSupport_No internet service')
(0.005833233955297278, 'StreamingTV_No internet service')
(0.004656246491838335, 'SeniorCitizen')
(0.004613171514422696, 'StreamingMovies_Yes')
(0.004499559319886553, 'Partner')
(0.003827242713864532, 'DeviceProtection_No')
(0.0033673289736262427, 'StreamingTV_Yes')
(0.0032246318604002375, 'PaperlessBilling')
(0.0030617464634349087, 'Dependents')
(0.0026909529825503847, 'DeviceProtection_Yes')
(0.0025497907785194517, 'StreamingMovies_No')
(0.0022121197062000136, 'PaymentMethod_Mailed check')
(0.0016962921997061052, 'PaymentMethod_Credit card (automatic)')
(0.001641964225790931, 'MultipleLines_Yes')
(0.0012197791463175382, 'StreamingTV_No')
(0.0008576520544336781, 'PaymentMethod_Bank transfer (automatic)')
(0.0008308913869217064, 'MultipleLines_No')
(0.00046103869229996837, 'gender')
(0.0004506833882795072, 'MultipleLines_No phone service')
(0.0003495540311650109, 'PhoneService')
```

xGBoost (Extreme Gradient Boosting)

This is the second tree-based classification models we have experiments in the model phase. The difference between the previous model is it using bagging to define the boundaries in Random Forest models.

Model parameters include n_estimators=50; criterion=entropy'
min_samples_split=285

The accuracy train accuracy and the testing accuracy was around 79%

The models seem to have low recall rate and higher precession rate.

The model predicted more False Negatives compared to the previous Decision tre
Model:

```

# instantiate xgb classifier
xg_cl = xgb.XGBRFClassifier(learning_rate=0.0001,objective='binary:logistic', n_estimators=25,seed=123)

# Fit on train
xg_cl.fit(X_train, y_train)

#Predict on Test
X_test_xgb = scaler.transform(X_test)
y_pred_xgb_test = xg_cl.predict(X_test_xgb)

# Classification report
print(metrics.classification_report(y_test, y_pred_xgb_test))

# confusion matrix
fig, ax = plt.subplots()
sns.heatmap(confusion_matrix(y_test, y_pred_xgb_test, normalize='true'), annot=True, ax=ax)
ax.set_title('Confusion Matrix')
ax.set_ylabel('Real Value')
ax.set_xlabel('Predicted Value')

plt.show()

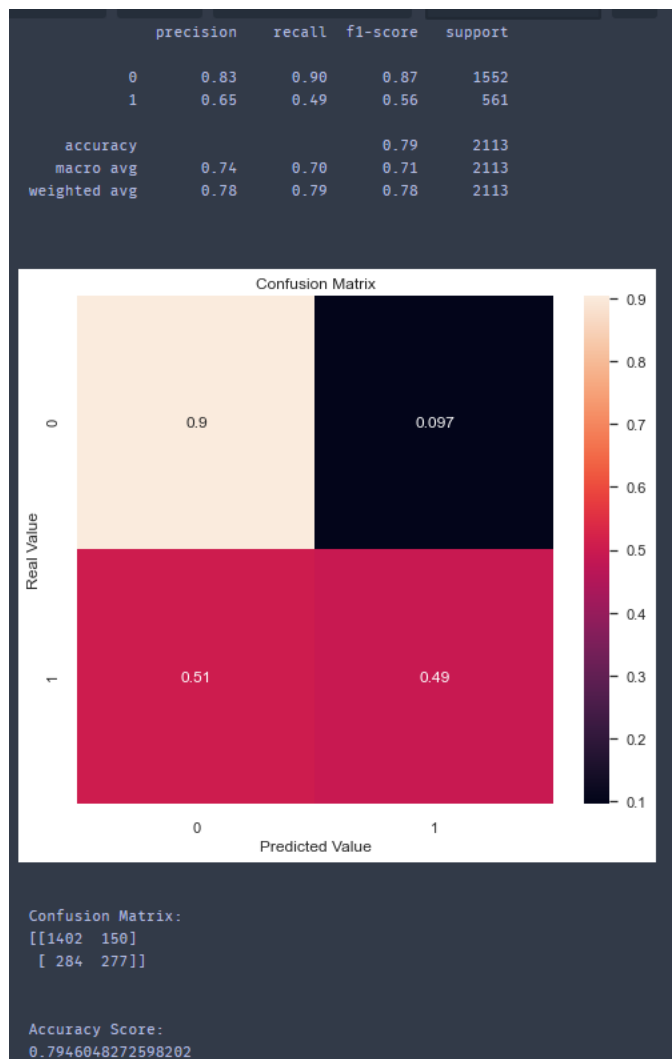
print("Confusion Matrix:")
print(metrics.confusion_matrix(y_test, y_pred_xgb_test))
print('\n')
print("Accuracy Score:")
print(metrics.accuracy_score(y_test, y_pred_xgb_test))
print("\n")

# show feature importance
importance = zip(xg_cl.feature_importances_, X)

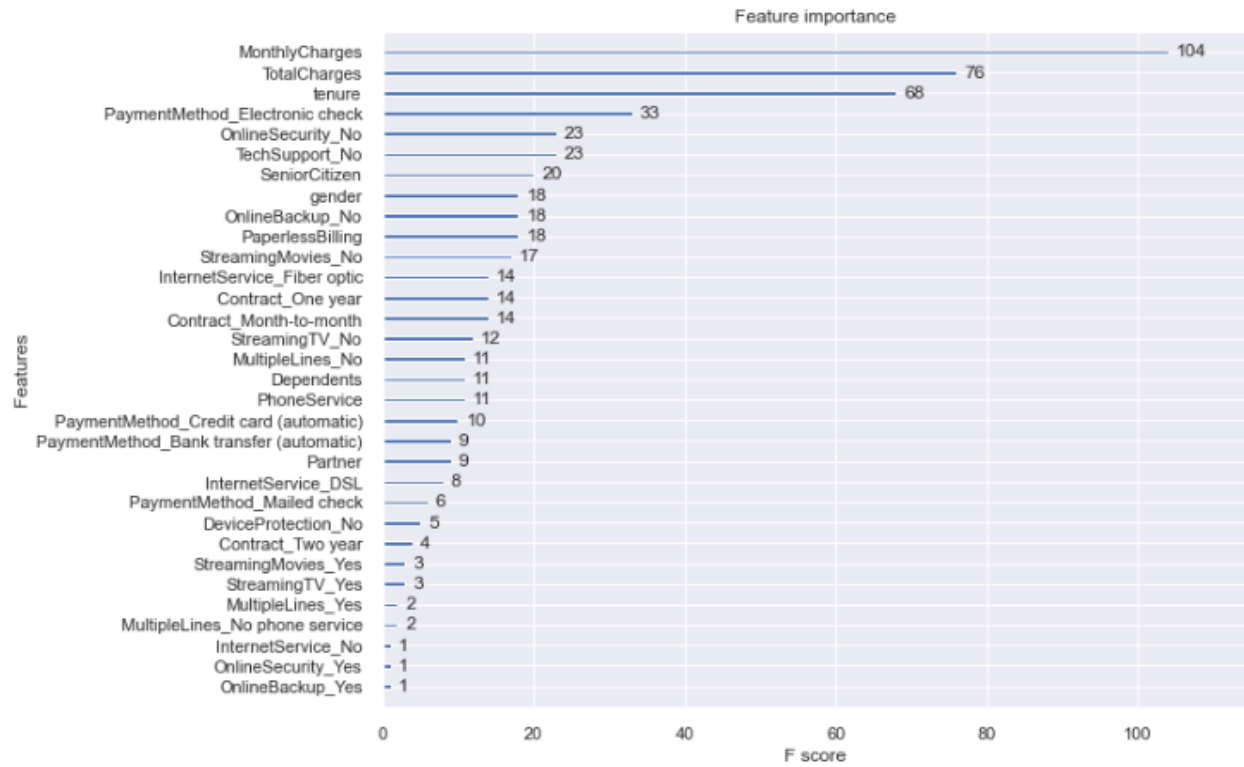
for rank in sorted(importance, key = lambda x: x[0], reverse = True):
    print(rank)

```

Evaluation metrics:



Feature importance's:



Final Model – Logistic Regression Tuned

Model:

```

#Initialize the log reg model instance
lr = LogisticRegression(solver='liblinear', C=0.0001)

# fit the training data
lr.fit(X_train, y_train)

# Measureing the model accuracy
X_test_lr = scaler.transform(X_test)
y_pred_lr = lr.predict(X_test_lr)

# Generate Classification report
print(classification_report(y_test, y_pred_lr))

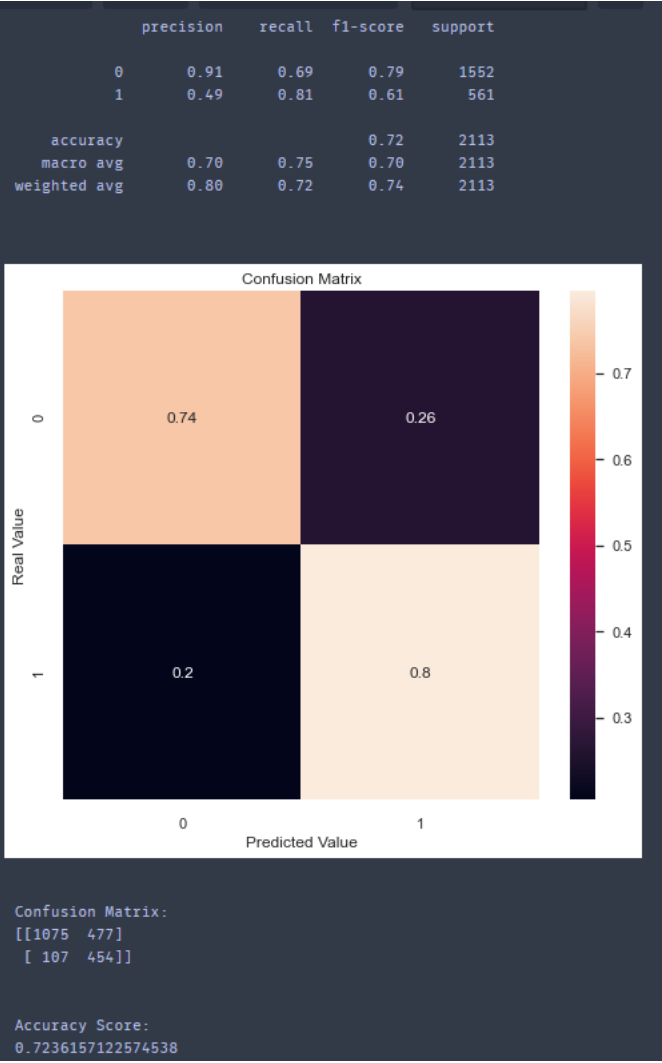
# confusion matrix
fig, ax = plt.subplots()
sns.heatmap(confusion_matrix(y_test, y_pred_test, normalize='true'), annot=True, ax=ax)
ax.set_title('Confusion Matrix')
ax.set_ylabel('Real Value')
ax.set_xlabel('Predicted Value')

plt.show()

print("Confusion Matrix:")
print(metrics.confusion_matrix(y_test, y_pred_lr))
print('\n')
print("Accuracy Score:")
print(metrics.accuracy_score(y_test, y_pred_lr))
print("\n")

```

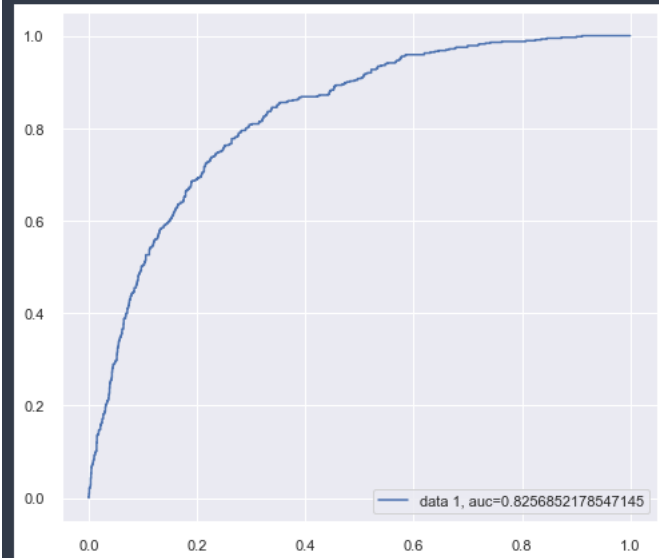
Evaluation Metrics:



AUC score:

```
# AUC curve
y_pred_prob = lr.predict_proba(X_test_lr)[:, 1]

fpr, tpr, _ = metrics.roc_curve(y_test, y_pred_prob)
auc = metrics.roc_auc_score(y_test, y_pred_prob)
plt.plot(fpr, tpr, label="data 1, auc="+str(auc))
plt.legend(loc=4)
plt.show()
```



Influence Plot – Final Logistic Regression:

```
# influence plotting
weights = pd.Series(lr.coef_[0],
                    index=X.columns.values)
weights.sort_values(ascending = True)

tenure -0.051096
Contract_Two_year -0.036978
TotalCharges -0.032872
OnlineSecurity_Yes -0.027642
Contract_One_year -0.025880
TechSupport_Yes -0.025490
InternetService_DSL -0.025116
Partner -0.020523
Dependents -0.020491
OnlineBackup_No internet service -0.019399
OnlineSecurity_No internet service -0.019399
InternetService_No -0.019399
TechSupport_No internet service -0.019399
StreamingMovies_No internet service -0.019399
DeviceProtection_No internet service -0.019399
StreamingTV_No internet service -0.019399
PaymentMethod_Bank transfer (automatic) -0.016746
PaymentMethod_Credit card (automatic) -0.015903
OnlineBackup_Yes -0.015297
DeviceProtection_Yes -0.011004
PaymentMethod_Mailed check -0.010893
MultipleLines_No phone service -0.004309
gender -0.002188
MultipleLines_No -0.001178
MultipleLines_Yes 0.003832
PhoneService 0.004309
StreamingTV_Yes 0.005903
StreamingMovies_No 0.007935
StreamingMovies_Yes 0.008350
StreamingTV_No 0.010362
MonthlyCharges 0.020533
SeniorCitizen 0.021757
PaperlessBilling 0.022696
DeviceProtection_No 0.026488
OnlineBackup_No 0.030677
PaymentMethod_Electronic check 0.038141
TechSupport_No 0.039061
InternetService_Fiber optic 0.040079
OnlineSecurity_No 0.040905
Contract_Month-to-month 0.052779
dtype: float64
```

Summary:

The Logistic Regression model seems to be showing a balanced performance than compared to the other classification models we have experimented with.

After performing the cross validation on the different classification models that were selected, Logistic Regression model has the highest accuracy with recall and the decision criterion.

Used GridSearchCV to tune the hyper parameters for the Logistic Regression model.

After the final model is run and the influence plotting is completed to understand which features effected the churn most. It can be observed that tenure, Contract_two_year and TotalCharges are the top three feature that can effect customer churn.

Next Steps:

This project helped me in getting hands on experience with applying machine learning modelling for answering business questions such as the customer churn. By studying these customer churn probabilities can have various applications in company to better serve their customers.

If time would have permitted, I wanted to dig deeper into the xGBoost models hyper parameter tuning even though the recall metric was higher when used in the model. I believe fine tuning the parameter would result in a optimized xGBoost model.

Also, I would like to try new classification algorithms.