Project: Supervised Learning Classification

Objective: The main objective of the analysis is to focus on the prediction of customer churn rates utilizing different classification models and the benefits of the analysis are to provide the idea of the customer who cancels the membership depending on the customer's behavior.

We will use a customer churn dataset from the telecom industry, which includes customer data such as long-distance usage, data usage, monthly revenue, types of offerings, and other services purchased by customers. We are using the subset of customers who have phone accounts. Since the data includes a mix of numeric, categorical, and ordinal variables, we will need to do some preprocessing.

Brief description of the data set: This dataset is related to the customers' different service that they are utilizing and payment type as below,

```
data.columns
```

```
Index(['months', 'multiple', 'gb_mon', 'security', 'backup', 'protection',
       'support', 'unlimited', 'contract', 'paperless', 'monthly',
       'satisfaction', 'churn_value', 'payment_Credit Card',
       'payment_Mailed Check', 'internet_type_DSL',
       'internet_type_Fiber Optic', 'internet_type_None', 'offer_Offer A',
       'offer_Offer B', 'offer_Offer C', 'offer_Offer D', 'offer_Offer E'],
      dtype='object')
```

From the description, we can find the details information of all features such as mean, standard deviation, different percentile and minimum and maximum value.

	months	multiple	gb_mon	security	backup	protection	support	unlimited	contract	paperless
count	7043.000000	7043.000000	7043.000000	7043.000000	7043.000000	7043.000000	7043.000000	7043.000000	7043.000000	7043.000000
mean	0.433551	0.421837	0.241358	0.286668	0.344881	0.343888	0.290217	0.673719	0.377396	0.592219
std	0.398231	0.493888	0.240223	0.452237	0.475363	0.475038	0.453895	0.468885	0.424234	0.491457
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.035294	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
50%	0.250000	0.000000	0.200000	0.000000	0.000000	0.000000	0.000000	1.000000	0.000000	1.000000
75%	0.750000	1.000000	0.317647	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

Data cleaning and feature engineering:

At the beginning, we can check whether there are any null values in the dataset, from the following analysis, we found that there is no null value.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7043 entries, 0 to 7042
Data columns (total 23 columns):
 #   Column                      Non-Null Count  Dtype  
---  ------                      --------------  -----  
 0   months                      7043 non-null   float64
 1   multiple                    7043 non-null   int64  
 2   gb_mon                      7043 non-null   float64
 3   security                    7043 non-null   int64  
 4   backup                      7043 non-null   int64  
 5   protection                  7043 non-null   int64  
 6   support                     7043 non-null   int64  
 7   unlimited                   7043 non-null   int64  
 8   contract                    7043 non-null   float64
 9   paperless                   7043 non-null   int64  
 10  monthly                     7043 non-null   float64
 11  satisfaction                7043 non-null   float64
 12  churn_value                 7043 non-null   int64  
 13  payment_Credit Card         7043 non-null   int64  
 14  payment_Mailed Check        7043 non-null   int64  
 15  internet_type_DSL           7043 non-null   int64  
 16  internet_type_Fiber Optic   7043 non-null   int64  
 17  internet_type_None          7043 non-null   int64  
 18  offer_Offer A               7043 non-null   int64  
 19  offer_Offer B               7043 non-null   int64  
 20  offer_Offer C               7043 non-null   int64  
 21  offer_Offer D               7043 non-null   int64  
 22  offer_Offer E               7043 non-null   int64  
dtypes: float64(5), int64(18)
memory usage: 1.2 MB
```

We will need to do details analysis of the dataset to identify which variables are binary, categorical and not ordinal, categorical and ordinal, and numeric. We will need to find out the unique values to identify the binary and categorical variables at the beginning. We choose contract, satisfaction and months as ordinary variables. The non-numeric features will need to be encoded using methods such as LabelEncoder. We processed data utilizing LabelBinarizer, LabelEncoder libraries. After processing data, we get the following format of data:

```
data.head()
```

	months	multiple	gb_mon	security	backup	protection	support	unlimited	contract	paperless
0	0	0	0.094118	0	0	1	0	0	0	1
1	0	1	0.200000	0	1	0	0	1	0	1
2	1	1	0.611765	0	0	0	0	1	0	1
3	1	0	0.141176	0	1	1	0	1	0	1
4	2	1	0.164706	0	0	0	0	1	0	1

Summary of training at least three different classifier models:

After completing all processing of data, at the beginning we split the train and test set keeping the testing data as 40 percent of the whole dataset.

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, accuracy_score, classification_report, f1_score
```

```
# Set up X and y variables
y = data['churn_value']
X = data.drop(columns='churn_value')
# Split the data into training and test samples
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state=42)
```

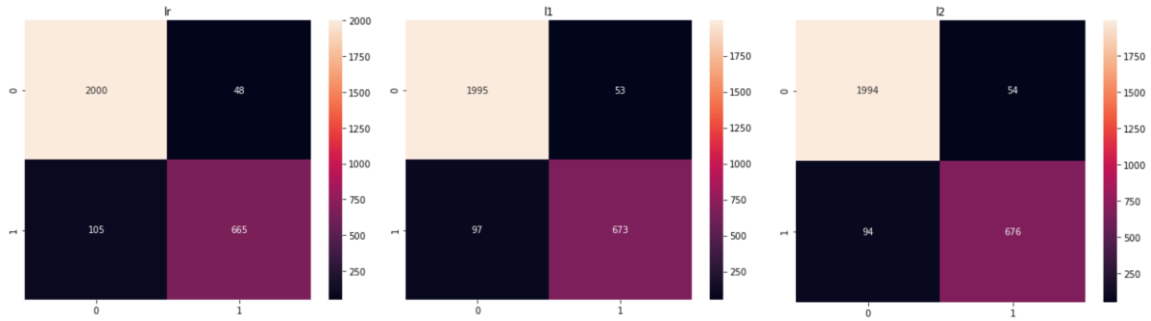
We used Logistic Regression as a baseline, then we did L1 and L2 regularization:

```
#Logistic Regression
from sklearn.linear_model import LogisticRegression

# Standard Logistic regression
lr = LogisticRegression(solver='liblinear').fit(X_train, y_train)
```

```
from sklearn.linear_model import LogisticRegressionCV

# L1 regularized logistic regression
lr_l1 = LogisticRegressionCV(Cs=10, cv=4, penalty='l1', solver='liblinear').fit(X_train, y_train)
```

```
# L2 regularized logistic regression
lr_l2 = LogisticRegressionCV(Cs=10, cv=4, penalty='l2', solver='liblinear').fit(X_train, y_train)
```

We found the confusion matrix for Logistic Regression, L1 and L2 regularization:



Then we built a K-Nearest Neighbors model and calculated the accuracy and F1 score:

```
#Using the same training and test set we evaluated K-Nearest Neighbors Model
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix, accuracy_score, classification_report, f1_score
```

```
# Estimating KNN model and report outcomes
knn = KNeighborsClassifier(n_neighbors=3)
knn = knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)
# Preciision, recall, f-score from the multi-class support function
print(classification_report(y_test, y_pred))
print('Accuracy score: ', round(accuracy_score(y_test, y_pred), 2))
print('F1 Score: ', round(f1_score(y_test, y_pred), 2))
```

Then we used Random Forest Classifier:

```
#Random Forest Classification
from sklearn.ensemble import RandomForestClassifier
```

```
M_features=X.shape[1]
max_features=round(np.sqrt(M_features))-1
max_features
```

```
4
```

```
model = RandomForestClassifier( max_features=max_features,n_estimators=20, random_state=0)
```

```
model.fit(X_train,y_train)
```

```
RandomForestClassifier(max_features=4, n_estimators=20, random_state=0)
```

Recommended Model comparing accuracy:

For Logistic Regression, L1 and L2 regularization, we found the precision, recall, f1score and accuracy as follows,

	lr	l1	l2
precision	0.945353	0.946355	0.947070
recall	0.945706	0.946771	0.947480
fscore	0.945035	0.946271	0.947034
accuracy	0.945706	0.946771	0.947480

For K-Nearest Neighbor, we found as follows:

```
              precision    recall  f1-score   support

           0       0.90      0.92      0.91      2048
           1       0.77      0.72      0.74       770

    accuracy                           0.86      2818
   macro avg       0.83      0.82      0.82      2818
weighted avg       0.86      0.86      0.86      2818

Accuracy score:  0.86
F1 Score:  0.74
```

For Random Forest Classifier, we found the following result:

```
{'test Accuracy': 0.9453513129879347, 'trian Accuracy': 0.9971597633136094}
```

By evaluating the above result, it is observed that Logistic Regression and Random Forest is better choice for the prediction of the customer churn.

Summary Key Findings and Insights:

We have evaluated three different models (Logistic Regression, K Nearest Neighbors, Random Forest) for our customer churn dataset. For each case we have calculated Accuracy of the models. In case of K Nearest Neighbors, our result is around 86 percent, however, in case of Logistic Regression and Random Forest our result is around 94 percent which is very impressive.

Suggestions for next steps:

For the better result, it would be great if we do hyper parameter tuning to get the best parameters and also evaluate other available models to compare each other to get more accurate results.