Predicting Churn For An ISP

By: Fiona Amuda GH1030807

Institution: Gisma University of Applied Sciences

Course: AI and Machine Learning M505A

Date: 26th September 2024

1. Problem Statement

Reduce the rate at which customers are unsubscribing from the internet services by predicting Churn. Predicting churn will help the company come up with better customer retention strategies and maintain sales which is a key KPI for the company.

1.1 Business Context

I have been contracted by Hello Telecommunication an internet service provider, to help them analyse their customer dataset. As a data scientist the business is expecting to see a churn predictive model that they can use to help them retain their customers. Customer attrition or Churn is when a business losses customers. Since my client offers subscription-based services churn is an important KPI for the business. I will analyse a sample of the customer subscription history to understand the trends in the dataset provided. Based in the data structure i will improve to some of the data features. I will use python programming language. Load the applicable Libraries and the CSV data file. Explore and understand the data before preprocessing it. I will Provide visualization. I will be using different classification regression models e.g logistic regression, I will compare and select best fit based on the performance matrix. Finally, i will give recommendation based on the findings.

1.2 Dataset Description

The data is made up of a ID column for unique identification of each customer. A count of the TV and movie subscribers captured in two columns. For each customer how many years they have been using the services and the billing average in separate columns. The numbers of years remaining on the contract. Download average, download limit and upload average. Then lastly the churn column.

1.3 How data was collected

Data is on customers who are using the ISP internet services which include TV and Movies subscriptions.

1.3 Modelling my project

Given that the churn label is a binary classification class, am going to use 4 models SVM, Decision tree, Random Forest and Linear regression. I will divide my data into training and test sets, features and label set. 80% allocated to training and 20% to testing. I will then test the performance of each model, cross validate the accuracy and tune the hyperparameters of the final selected model to improve its accuracy.

2. Exploration of the dataset

2.1 Import Libraries

Am going to use pandas to load the dataset on a data frame for ease of interpretation. NumPy for arrays. Seaborn, Missingno and matplotlib for visualization. Sklearn to run all my different model tasks.

```
import pandas as pd
import numpy as np
import seaborn as sns
import missingno as msno
import matplotlib.pyplot as plt
import sklearn.model_selection
from sklearn.model_selection import train_test_split
from imblearn.over_sampling import SMOTE
from sklearn.decomposition import PCA
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, classification_report, confusion_matrix
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.model_selection import cross_val_score
```

```
from sklearn.model_selection import KFold
from sklearn.model_selection import GridSearchCV
```

2.2 Reading the ISP dataset

```
isp_df = pd.read_csv('/content/churn_ISP.csv')
```

2.3 How many columns and row are present

From the below results there are 72,274 Rows and 11 Columns

```
isp_df.shape
```

```
(72274, 11)
```

2.4 What features does the dataset have

```
pd.DataFrame(isp_df.columns, columns=['Column Names'])
```

	Column Names
0	id
1	is_tv_subscriber
2	is_movie_package_subscriber
3	subscription_age
4	bill_avg
5	reamining_contract
6	service_failure_count
7	download_avg
8	upload_avg
9	download_over_limit
10	churn

2.5 Selecting 10 random rows to analyze

I can see some values are missing

```
isp_df.sample(10)
```

	id	is_tv_subscriber	is_movie_package_subscriber	subscription_age	bill_avg	reamining_contract	service_failure_count
63456	1487683	1	1	0.77	19	1.20	0
46792	1096529	0	0	0.16	60	0.00	4
45047	1055409	1	0	1.15	16	0.00	0
58903	1380690	1	0	1.12	14	0.86	0
31079	728979	1	0	4.62	24	NaN	0
33342	782977	0	0	1.41	20	NaN	0
69728	1631054	1	1	0.15	12	1.82	0
13069	303916	0	0	2.29	0	NaN	0
21720	508711	1	0	5.19	17	NaN	1
71738	1677676	1	0	0.01	0	0.94	0

2.6 What is the data type on each column

There only numerical values in the dataset

```
isp_df.dtypes
```

	0
id	int64
is_tv_subscriber	int64
is_movie_package_subscriber	int64
subscription_age	float64
bill_avg	int64
reamining_contract	float64
service_failure_count	int64
download_avg	float64
upload_avg	float64
download_over_limit	int64
churn	int64

2.7 Are there any missing values

Remaining contract has the most missing values, dowlaod avg and upload avg has some missing values.

```
#Checking percentage of missing values
data_missing = isp_df.isna().sum()
percentage_missing = (data_missing/len(isp_df))*100
show_table = pd.DataFrame({'Missing Values': data_missing, 'Percentage': percentage_missing})
show_table
```

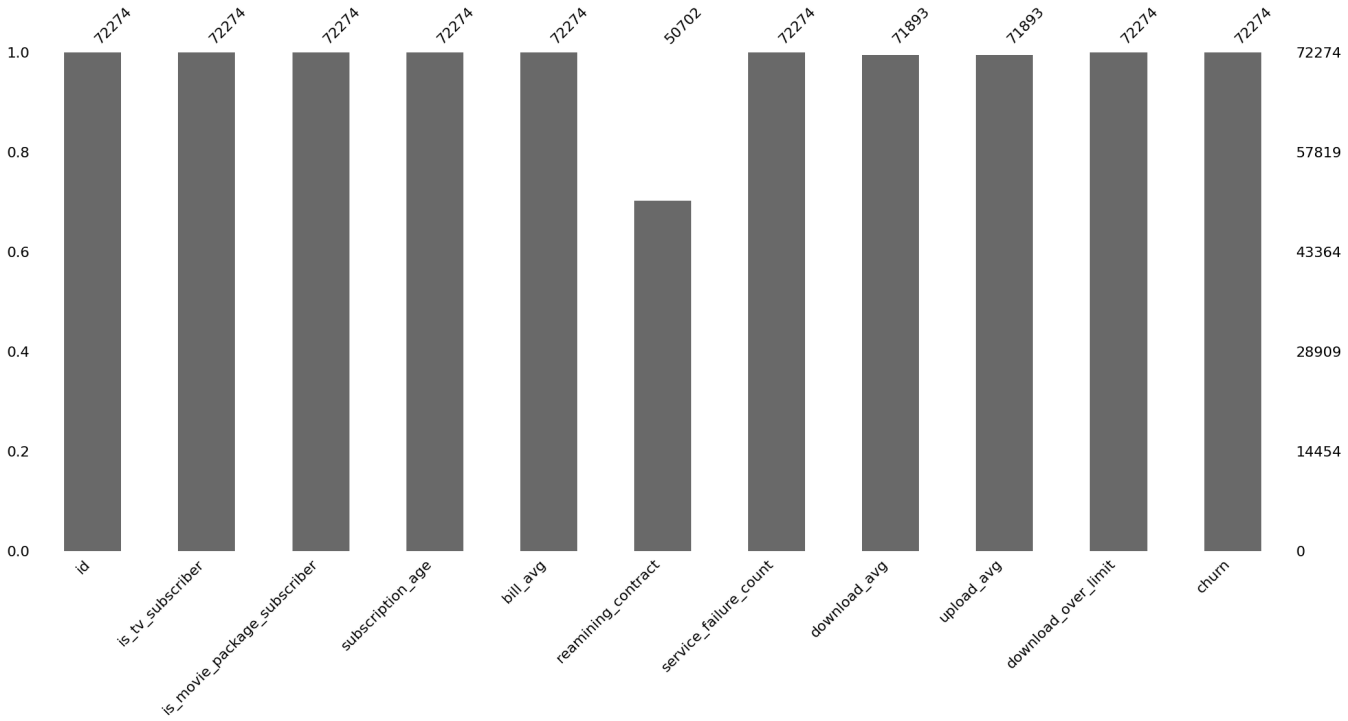
	Missing Values	Percentage
id	0	0.000000
is_tv_subscriber	0	0.000000
is_movie_package_subscriber	0	0.000000
subscription_age	0	0.000000
bill_avg	0	0.000000
reamining_contract	21572	29.847525
service_failure_count	0	0.000000
download_avg	381	0.527161
upload_avg	381	0.527161
download_over_limit	0	0.000000
churn	0	0.000000

Next steps: Generate code with show_table | View recommended plots | New interactive sheet

2.8 Visual representation of the missing values

```
#Using Missingno to present missing values on a diagram
msno.bar(isp_df)
```

```
<Axes: >
```



2.9 Findings from the initial observation (Metadata)

The dataset has an ID column that will not be required during model fitting and analysis, I will drop the ID column. ID will not contribute to the final result; it has no association with the targeted 'churn'. After dropping ID I will check for any duplicates. The ID column is a unique identifier, this was making each row entry unique. There are about 22,000 missing values, given that the dataset has about 70,000 rows i will drop the rows with missing values. The alternative option would have been to impute the missing data which might introduce some biases given the large quantity of missing data. There are no categorical values in the dataset hence no encoding is required. There will be 9 features to be used to predict the target churn. The number rows remaining is enough data to use for analysis.

3.0 Preprocessing the dataset before Explanatory analysis

3.1 Droping ID Column

```
print('*Before drop id, column id is present in the Data Frame')
isp_df.sample(2)
```

```
*Before drop id, column id is present in the Data Frame
```

	id	is_tv_subscriber	is_movie_package_subscriber	subscription_age	bill_avg	reamining_contract	service_failure_count
37167	871451	0	0	0.47	22	NaN	0
65851	1541893	1	0	0.56	21	0.43	0

```
#droping id column
isp_df.drop('id', axis=1, inplace = True)


print('*After drop id, its no longer part of the Data Frame')
isp_df.sample(2)
```

```
*After drop id, its no longer part of the Data Frame
```

	is_tv_subscriber	is_movie_package_subscriber	subscription_age	bill_avg	reamining_contract	service_failure_count	downloa
64114	1	0	0.71	14	1.26	2	
33104	1	0	1.59	29	0.00	0	

3.2 Are there any duplicates

There some duplicated values.

```
#Checking for duplicates
isp_df.duplicated().sum()
```

```
2149
```

```
#Drop the duplicates
isp_df.drop_duplicates(inplace=True)


#Checking for duplicates after drop
isp_df.duplicated().sum()
```

```
0
```

3.3 Let's Drop any rows with missing values

```
#Drop all rows with NaN
isp_df.dropna(inplace=True)


#Confirm if there are any NaN values
isp_df.isna().sum()
```

	0
is_tv_subscriber	0
is_movie_package_subscriber	0
subscription_age	0
bill_avg	0
reamining_contract	0
service_failure_count	0
download_avg	0
upload_avg	0
download_over_limit	0
churn	0

4.0 Explanatory analysis

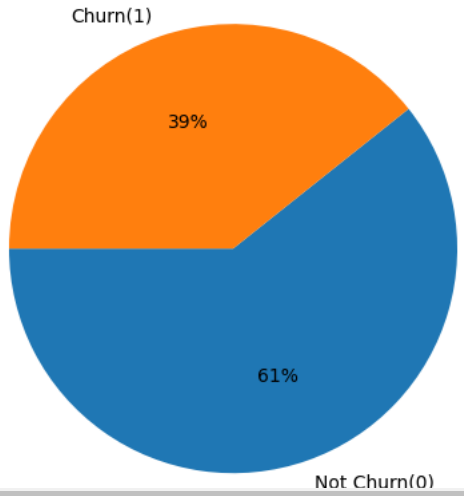
4.1 Is the target column balanced

From the below outcome, the target churn is imbalanced, 39% is churn and 61% not churn. To correct this will use SMOTE after splitting the data to train and test. This will help with the imbalance.

```
#Calculating the value count on the target
isp_df['churn'].value_counts()
```

churn	count
0	30044
1	19462

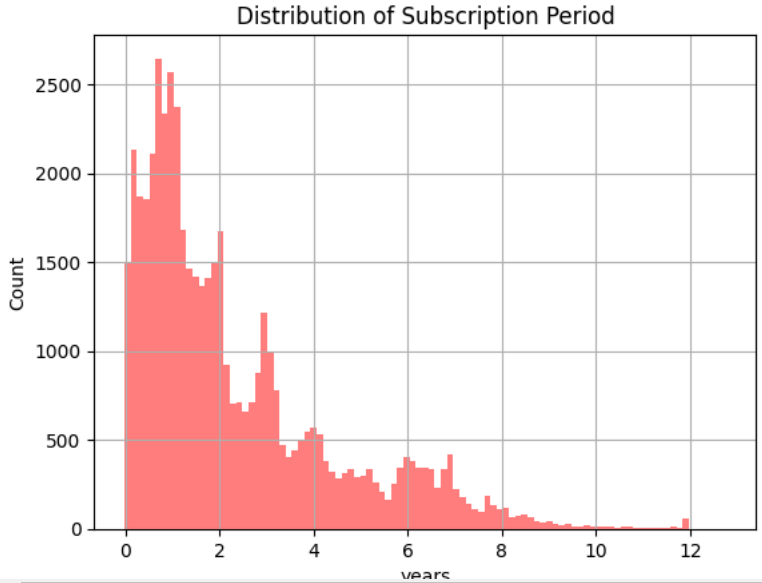
```
#Pie chart showing the current data distributin on churn
target = ['Not Churn(0)', 'Churn(1)']
sizes = [30044, 19462]
plt.pie(sizes, labels=target, autopct='%1.0f%%', startangle=180)
plt.axis('equal')
plt.show()
```



4.2 Subscription age distribution

The below histogram shows how many years have customers subscribed to the ISP services. Most customer are below 8 years subscription.

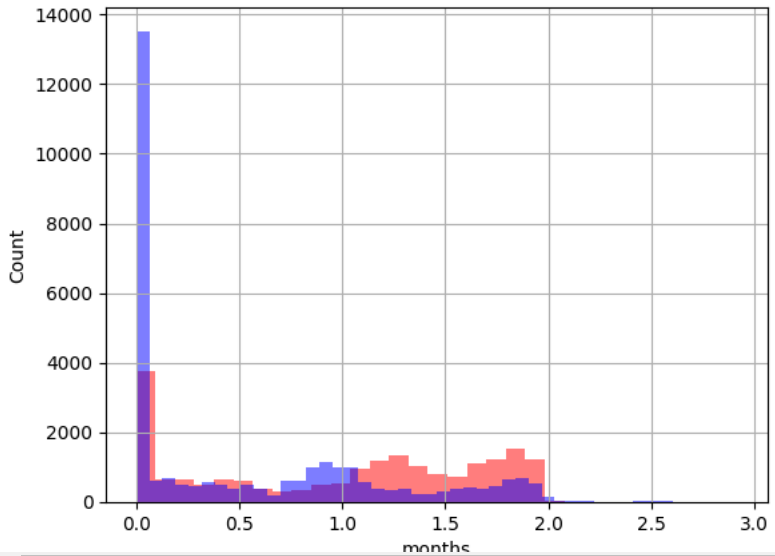
```
isp_df['subscription_age'].hist(bins='auto',color='red', alpha=.5)
plt.title("Distribution of Subscription Period")
plt.xlabel("years")
plt.ylabel("Count")
plt.show()
```



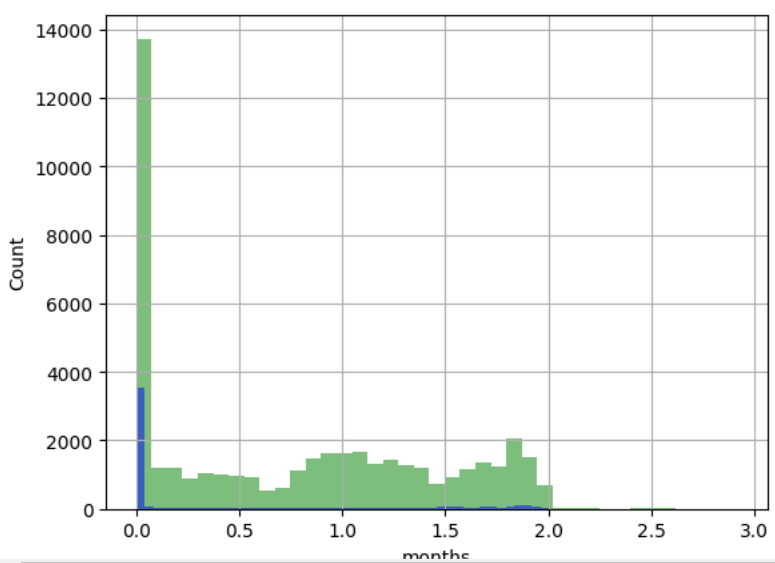
4.3 Distribution of movie and TV package aganist remaining contract

For the past 3 months, 90% of the remaining contract are subscribed to movie and tv

```
isp_df[isp_df['is_movie_package_subscriber']==True]['reamining_contract'].hist(bins='auto',color='red', alpha=.5)
isp_df[isp_df['is_movie_package_subscriber']==False]['reamining_contract'].hist(bins='auto',color='blue', alpha=.5)
plt.xlabel("months")
plt.ylabel("Count")
plt.show()
plt.show()
```




```
isp_df[isp_df['is_tv_subscriber']==True]['reamining_contract'].hist(bins='auto',color='green', alpha=.5)
isp_df[isp_df['is_tv_subscriber']==False]['reamining_contract'].hist(bins='auto',color='blue', alpha=.5)
plt.xlabel("months")
plt.ylabel("Count")
plt.show()
plt.show()
```

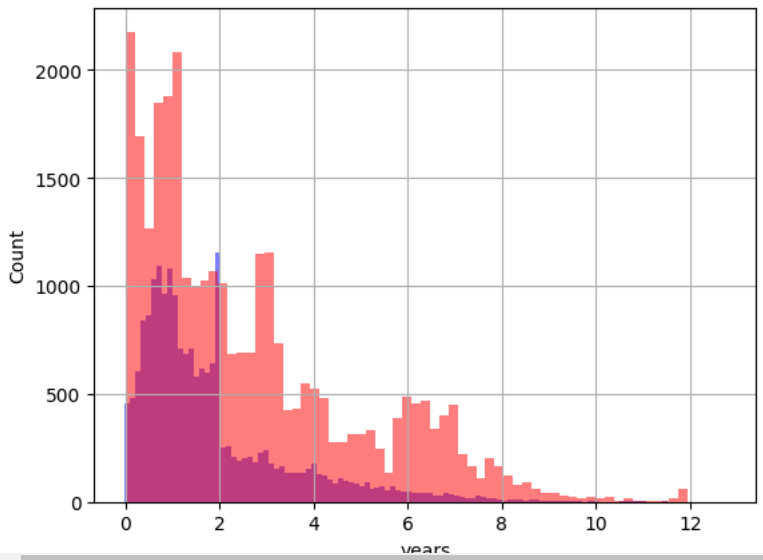


4.4 Churn over the years

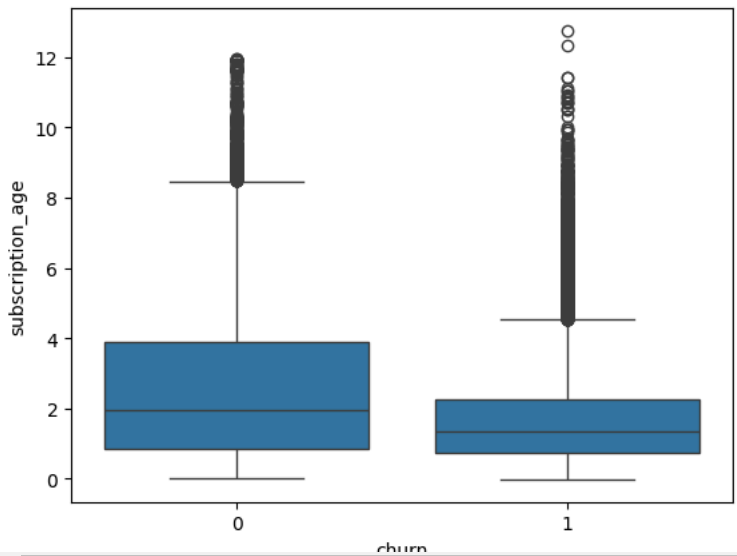
The business has retained most if the customers over the yaers

```
isp_df[isp_df['churn']==True]['subscription_age'].hist(bins='auto',color='blue', alpha=.5)
isp_df[isp_df['churn']==False]['subscription_age'].hist(bins='auto',color='red', alpha=.5)
plt.xlabel("years")
plt.ylabel("Count")
```

```
plt.show()
plt.show()
```




```
sns.boxplot(x = isp_df.churn, y = isp_df.subscription_age)
plt.show()
```



5.0 Model

On this next step will fit the models and test the performance before selecting the best fit

5.1 Split data to train set and hold out set

```
train, test = sklearn.model_selection.train_test_split(isp_df)
print("Data to use for Training:", train.shape)
print("Data to use for Testing", test.shape)
```

```
Data to use for Training: (37129, 10)
Data to use for Testing (12377, 10)
```

5.2 Defining varieables to use for feautures and target

```
#droping the target column, assigning variable to the feature set
x=isp_df.drop('churn',axis=1)
#assigning variable to the target set
y=isp_df['churn']
```

5.3 Assigning feature and target set, training and test data

```
#spliting
x_train,x_test,y_train,y_test = train_test_split(x,y,stratify=y,test_size=0.2,random_state=42)
print(x_train.shape)
print(y_train.shape)
print(x_test.shape)
print(y_test.shape)
```

```
(39604, 9)
(39604,)
(9902, 9)
(9902,)
```

5.4 SMOTE

As we had seen the is imbalance, hence introducing SMOTE to resolve the imbalance issue by adding some synthetic samples for the low class on the training dataset. We can not do the same on test as it may introduce data leakage.

```
#calling SMOTE because of the data imbalance
sm = SMOTE()
x_train, y_train = sm.fit_resample(x_train, y_train)
print(x_train.shape)
print(y_train.shape)
```

```
(48070, 9)
(48070,)
```

5.5 PCA

Here am using PCA to reduce dimensionality and pick best features to use for the model training and test.

```
#assigning the best features for the model
pca=PCA(n_components=0.8)
x_train=pca.fit_transform(x_train)
x_test=pca.transform(x_test)
```

6.0 Fitting the models used in this project

I used 4 model on the dataset. Since am working with classification algorithms, i have selected SVC, decision tree, Logistic regression and random forest.

6.1 SVM

```
model_one = sklearn.svm.SVC()
model_one_svm = model_one.fit(x_train,y_train)
```

6.2 Decision Tree

```
model_two = DecisionTreeClassifier()
model_two_dt = model_two.fit(x_train,y_train)
```

6.3 Logistic Regression

```
model_three = LogisticRegression()
model_three_lr = model_three.fit(x_train,y_train)
```

6.4 Random Forest

```
model_four = RandomForestClassifier()
model_four_rf = model_four.fit(x_train,y_train)
```

7.0 Training and computing the accuracy of the 4 models

On this next step i will use the train dataset and see how best the models can interpreted and work on the dataset. I will get the train accuracy matric for each model. From the below results Decision tree model has the highest accuracy of 99.98% ** Followed by Random Forest with accuracy of **99.94%, then SVM with 65.8%, then Logistic regression with 63.2. My next step is to use the hold out dataset on the four-model ad see how well the model is predicting the outcomes.

7.1 SVM

```
#predicting the acuracy of the train data
from sklearn.metrics import accuracy_score

svm_label_train = model_one.predict(x_train)
svm_accuracy_train = accuracy_score(y_train, svm_label_train)
print(f"Accuracy: {svm_accuracy_train * 100:.2f}%")
```

```
Accuracy: 66.11%
```

7.2 Decision Tree

```
#predicting the acuracy of the train data
dt_label_train = model_two.predict(x_train)
dt_accuracy_train = accuracy_score(y_train, dt_label_train)
print(f"Accuracy: {dt_accuracy_train * 100:.2f}%")
```

```
Accuracy: 99.97%
```

7.3 Logistic Regression

```
#predicting the acuracy of the train data
lr_label_train = model_three.predict(x_train)
lr_accuracy_train = accuracy_score(y_train,lr_label_train)
print(f"Accuracy: {lr_accuracy_train * 100:.2f}%")
```

```
Accuracy: 63.37%
```

7.4 Random Forest

```
#predicting the acuracy of the train data
rf_label_prediction = model_four.predict(x_train)
rf_accuracy_train = accuracy_score (y_train,rf_label_prediction)
print(f"Accuracy: {rf_accuracy_train * 100:.2f}%")
```

```
Accuracy: 99.94%
```

8.0 Baseline Testing and computing matric of the 4 models used

On this next step am looking to run the test data set on the 4 models. Get the baseline metrics of the models including a cross validation matric, deciding on the next steps. Am going to determine if there are any biases; given that i have the above training accuracy score i can compare this to the test accuracy score. I will then look at additional metrics like precision, recall and F1 to further help me with my analysis. Based on the below outcome

SVM model is balanced with a slightly higher score on the test set, SVM has the highest accuracy score on test data Decision Tree model has a huge accuracy score difference. The training set performed better than the test by about 40% Linear regression is balanced with the training set

performing better than the test by about 2%. The accuracy scores are lower than SVM. Radom forest is very similar to the scores in decision tree, decision tree outperformed random forest Based on this analysis on accuracy alone i will proceed with the two models SVM and decision tree. Decision tree has the very good training accuracy score of 99% and SVM test score is higher than the decision tree score.

On my next steps i will work on the decision tree model. My aim is to improve the test accuracy score before deciding with model to select between the two.

8.1 SVM

```
svm_label_prediction = model_one.predict(x_test)
svm_accuracy_test = accuracy_score (y_test,svm_label_prediction)
print(f"Accuracy_Train: {svm_accuracy_train * 100:.2f}%")
print(f"Accuracy_Test: {svm_accuracy_test * 100:.2f}%")
```

```
Accuracy_Train: 66.11%
Accuracy_Test: 68.67%
```

8.2 Decision Tree

```
dt_label_prediction = model_two.predict(x_test)
dt_accuracy_test = accuracy_score (y_test,dt_label_prediction)
print(f"Accuracy_Train: {dt_accuracy_train * 100:.2f}%")
print(f"Accuracy_Test: {dt_accuracy_test * 100:.2f}%")
```

```
Accuracy_Train: 99.97%
Accuracy_Test: 59.56%
```

8.3 Logistic Regression

```
lr_label_prediction = model_three.predict(x_test)
lr_accuracy_test = accuracy_score (y_test,dt_label_prediction)
print(f"Accuracy_Train: {lr_accuracy_train * 100:.2f}%")
print(f"Accuracy_Test: {lr_accuracy_test * 100:.2f}%")
```

```
Accuracy_Train: 63.37%
Accuracy_Test: 59.56%
```

8.4 Random Forest

```
rf_label_prediction = model_four.predict(x_test)
rf_accuracy_test = accuracy_score (y_test,rf_label_prediction)
print(f"Accuracy_Train: {rf_accuracy_train * 100:.2f}%")
print(f"Accuracy_Test: {rf_accuracy_test * 100:.2f}%")
```

```
Accuracy_Train: 99.94%
Accuracy_Test: 59.59%
```

9.0 SVM cross validation and confusion matrix

Based on the below Kfold analysis SVM accuracy is almost like the cross validated folds mean accuracy. This indicates that the performance of this SVM model is around 65% accurate. This is true as the confusion matrix shows that the model seems to be missing some amount of true positive, with a precision score of 72%. My next step is to tune the hyperparameters of the decision tree model and bring the test accuracy up before looking at the confusion matrix on the decision tree model.

9.1 kFolds on SVM

```
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
#assigning fold size
kfold = KFold(n_splits=5, shuffle=True, random_state=42)
```

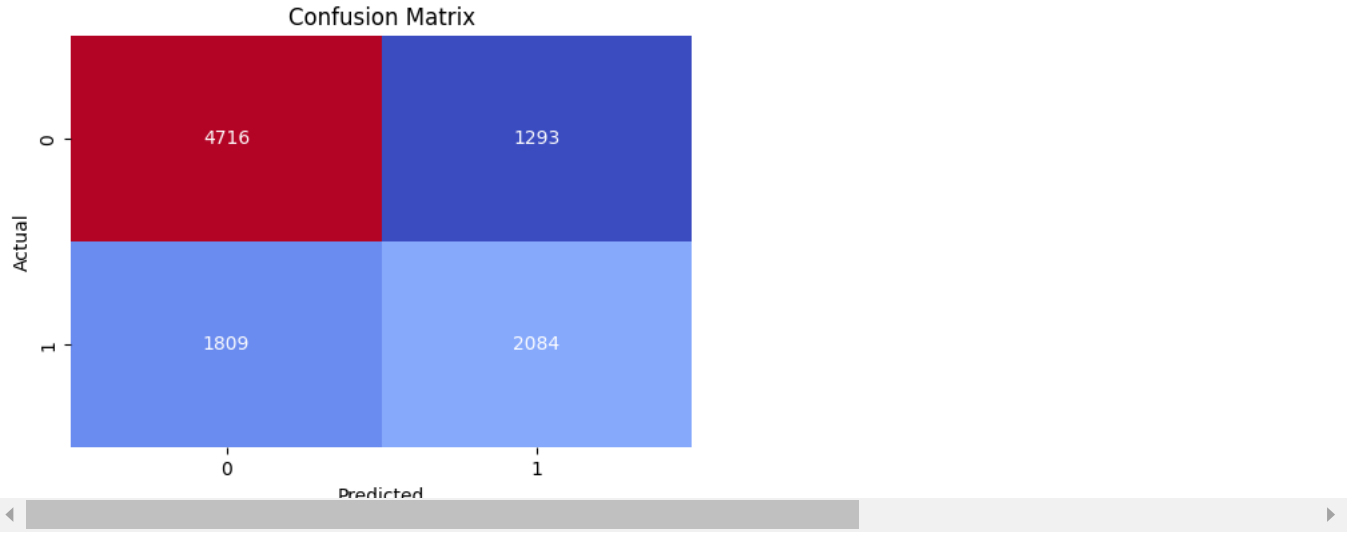
```
#computing the accuracy of the folds
scores = cross_val_score(model_one, x_train, y_train, cv=kfold, scoring='accuracy')
#accuracy of each fold
print("Accuracy scores for each fold:", scores)
#getting the matrix from the 5 folds
print(f"Mean accuracy: {scores.mean() * 100:.2f}%")
print(f"Standard deviation: {scores.std() * 100:.2f}%")
```

```
Accuracy scores for each fold: [0.66309549 0.66049511 0.66215935 0.652798   0.66652798]
Mean accuracy: 66.10%
Standard deviation: 0.46%
```

9.2 Confusion Matrix on SVM

```
#ploting the lable predictions of the test set
conf_matrix = confusion_matrix(y_test, svm_label_prediction)
print("Confusion Matrix:")
print(conf_matrix)
#setting the diagram size and axis names
plt.figure(figsize=(6, 4))
sns.heatmap(conf_matrix, annot=True, cmap="coolwarm", fmt='g', cbar=False)
plt.title("Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()
```

```
Confusion Matrix:
[[4716 1293]
 [1809 2084]]
```




```
#Looking at the other measures other than accuracy
svm_classification = classification_report(y_test, svm_label_prediction)
print(svm_classification)
```

```
              precision    recall  f1-score   support

           0       0.72      0.78      0.75      6009
           1       0.62      0.54      0.57      3893

    accuracy                           0.69      9902
   macro avg       0.67      0.66      0.66      9902
weighted avg       0.68      0.69      0.68      9902
```

10.0 Decision Tree Hyperparameter Tuning and Confusion Matrix

The next step is to improve the accuracy score of decision tree model. I will use grid search to tune the hyperparameters. Then have a look at metrics measures after implementing grid search. From the below results grid search did improve the accuracy score from 59 to 69. The model is testing much better than before.

10.1 Decisison Tree Grid Search

```
#Setting the grid
```

```
#setting the grid
param_grid = {
    'max_depth': [5, 10, 15, 20, None],
    'min_samples_split': [2, 10, 20],
    'min_samples_leaf': [1, 5, 10],
    'max_features': [None, 'sqrt', 'log2'],
    'criterion': ['gini', 'entropy']
}

dt = DecisionTreeClassifier()
grid_search = GridSearchCV(estimator=dt, param_grid=param_grid, cv=5, scoring='accuracy')
grid_search.fit(x_train, y_train)
best_params = grid_search.best_params_
best_model = grid_search.best_estimator_



#these are the hyperparameters tuned
print(f"Decison Tree Hyperparameters: {best_params}")
```

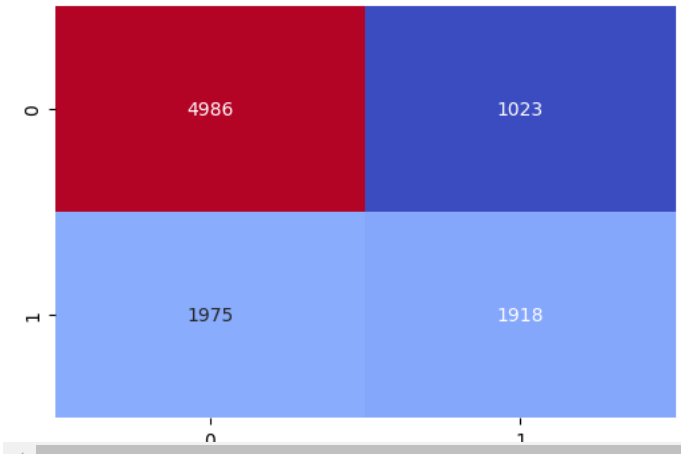
```
Decison Tree Hyperparameters: {'criterion': 'entropy', 'max_depth': 5, 'max_features': None, 'min_samples_leaf': 1, 'min_samples_spl
```

```
#implementing the changes on the test set
dt = best_model.predict(x_test)


#Confirming accuracy score after grid search
from sklearn.tree import DecisionTreeClassifier
dt_accuracy_test = accuracy_score(y_test, dt)
print(f"Test Accuracy: {dt_accuracy_test * 100:.2f}%")
```

```
Test Accuracy: 69.72%
```

```
#checking how may predicted true positive are there as this is important for the business
conf_matrix = confusion_matrix(y_test, dt)
print("Confusion Matrix:")
print(conf_matrix)

plt.figure(figsize=(6, 4))
sns.heatmap(conf_matrix, annot=True, cmap="coolwarm", fmt='g', cbar=False)
#plt.title("Confusion Matrix")
#plt.xlabel("Predicted Labels")
#plt.ylabel("True Labels")
plt.show()
```

```
Confusion Matrix:
[[4986 1023]
 [1975 1918]]
```




```
#Looking at the other metrics other than accuracy score to assess the prediction
dt_scores_CV = classification_report(y_test, dt)
print(dt_scores_CV)
```

```
              precision    recall  f1-score   support

           0       0.72      0.83      0.77      6009
           1       0.65      0.49      0.56      3893

    accuracy                           0.70      9902
   macro avg       0.68      0.66      0.67      9902
weighted avg       0.69      0.70      0.69      9902
```

11. Recommendation and Conclusion

My recommendation would be to proceed with Model two, Decision Tree. It has the highest accuracy score on training 99.9% which is very good. The steps taken to balance the training data played a big part in the outcome. The imbalance if not corrected would have introduced some biases. However, the model test accuracy score could still be better. The 69% score is higher than all the other three models. This was after tunning the hyperparameters on the decision tree model. The dataset has an imbalance hance making the test set less effeminate. The Test prediction is good but could be better than the 69% score attained. More data on the churn label need to be present to ensure there is a balance. The model accuracy results show there is a problem with overfitting. The training scores are very good but when it comes to testing the model scores are 30% less efficient. Since its important for the company to predict churn the recall and precision matrix are a key indicator in this model. The scores after grid search implementation improved but could still be better with the availability of a balanced dataset. I recommend that the company should collect more data regarding their churn customers. The company should use Decision tree for best results.

12. Refrence

Data Link

https://www.kaggle.com/datasets/mehmetsabrikunt/internet-service-churn/data

Notebook Link

https://drive.google.com/file/d/167GlUROBrWExfm6r9K1mE8DUJ-uS9Wl9/view?usp=drive_link

HTML Convertion

I tried conversting my file to htlm but had issus. saving it as PDF.

```
from google.colab import files
Assessment = files.upload
import subprocess
file0 = list(f.keys())[0]
_= subprocess.run(["pip","install","nbconvert"])
_= subprocess.run(["jupyter","nbconvert", file0,"--to", "html"])
files.download(file0[:-5]+"html")
```