Classification of Telco Customer using Machine Learning Algorithms for Churn Prediction

Abstract: This paper provides and discusses the results of applying different Supervised Machine Learning techniques to a telco customer churn data set. Churn means "leaving the organisation". In any Industry, Knowing why and when consumers are likely to leave is crucial for a company. Businesses can take steps to keep consumers from leaving by using a reliable and accurate churn prediction model. There are four steps in this project. Foremost one is Exploratory Data Analysis, second one is Data Pre-processing, third one is Implementation of machine learning algorithms such as decision trees, logistic regression, and random forests and finally to compare accuracy of these three Models. The above mentioned techniques were implemented using Python programing language utilizing machine learning repositories, prediction results and algorithm performance measures were obtained.

I. INTRODUCTION – PREDICTION OF CUSTOMER CHURN

One of the biggest sectors in developed nations is telecommunication. Without telecom industry there is no digital communication in the world. Customer churn is one of the main issues facing the telecom sector. Customer churn is nothing but loss of customers from the particular service/organisation to others due to the competitors or some other factors [1]. The upcoming progress on latest technologies and the heavy number of competitors highly impact the telecom industry. Companies surviving hard in the growing market, depending on different strategies. The primary factor in telecom service losses is customer churn, and this is a huge issue. The first and foremost secret to succeed in any service sector is to identify the factors of customer churn and then introducing new schemes/offers, where/when ever required.

This is possible by using machine learning techniques and prediction methods by using past data. In section II the detailed explanation of the data set is discussed. In section III the Machine Learning classification techniques which were applied to the data set are briefly explained. In section IV the steps required to pre-process the data are discussed. In section V the outcomes of using machine learning methods on the data set are shown. Section VI gives a brief discussions and conclusions of the project. The appendix includes detailed description of dataset and the source code for this project which can be re-run to accurately reproduce the results.

This study implements several methods, including logistic regression, decision trees and random forests, analyses accuracy, and uses the results to predict customer attrition.

II. DETAILS OF DATA SET

The selected Dataset is a real-world classification problem which was taken from an open source repository named Kaggle. It contains 7043 instances each containing 21 attributes. Each instance is one costumer's unique data. the services each customer has subscribed to (phone, multiple lines, internet, online security, online backup, device protection, tech support, and streaming TV and movies), information about each customer's account (how long they've been a customer, contract, payment method,

paperless billing, monthly charges, and total charges), and personal information about customers (gender, age range, and if they have partners and dependents) are all included in the data set.

For 7043 consumers, there are 20 attributes (independent variables) and 1 target (dependent variable). The target variable specifies whether a customer has left the business recently (i.e., churn=yes).

III. MACHINE LEARNING TECHNIQUES

A. Classification using Decision Tree

Although decision trees can be used to solve classification and regression issues, classification issues are the most frequently encountered. Decision Node and Leaf Node are the two nodes. While Leaf nodes are the results of decisions and do not have any more branches, Decision nodes are used to make decisions and have numerous branches. The fundamental problem that emerges while developing a decision tree is how to choose the best attribute for the root node and for sub-nodes. As a result, a method called ASM can be used to tackle these issues (Attribute selection measure). Information Gain (assessment of changes in entropy following the segmentation of a dataset based on an attribute) and Gini Index (measure of impurity or purity utilised while generating a decision tree in the CART (Classification and Regression Tree) algorithm) are two prominent strategies for ASM [2].

B. Logistic Regression

In order to accomplish a binary classification, logistic regression models the probability of an occurrence based on the input features. By calculating the likelihood that a particular observation belongs to a given class, classifications are carried out. Logistic regression is a type of regression that incorporates the idea of predictive modelling; as a result, it is used to classify data and belongs to the classification algorithm.

It is used to forecast how a categorical dependent variable will behave. The result should therefore be discrete or categorical, such as 0 OR 1, Yes OR No, True OR False, High OR Low. We fit an "S" shaped function instead of a regression line, which predicts two values (0 or 1). Here, the threshold value is set at 0.5. This cutoff value represents the likelihood of either winning or losing. If the value is larger than 0.5, it can be treated as 1, whereas if it is less than 0.5, it can be decreased to 0. The sigmoid curve is shown in Fig. 1 below [3].

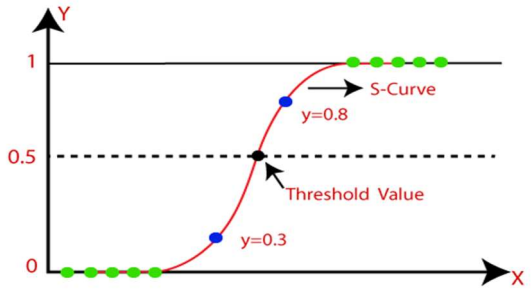

Fig 1. Logistic Regression Sigmoid curve

C. Classification using Random Forest

Random Forest is used for classification, regression, and other applications. A randomly chosen portion of the training data is used to generate a collection of decision trees. It generally consists of a number of decision trees (DT) drawn from a randomly chosen subset of the training set, and it then compiles the votes from various DTs to determine the final prediction. The model is more efficient with more trees, what results in a higher accuracy of the model. Random Forest classification algorithm performs on the same regulations as Decision Tree [4].

IV. EXPERIMENTAL SETUP

This section covers Exploratory Data Analysis, Data Pre-processing in order to

prepare the dataset for model creation. Also implemented all algorithms proposed in the above section.

Exploratory Data Analysis:

- First we need to check for missing values, if any we can delete the rows.
- Now we need to focus on target variable i.e.; churn column. Fig 2 shows the value count of churn variable.

```
#checking value count of target variables
churn_data['Churn'].value_counts()
```

```
No     5174
Yes    1869
Name: Churn, dtype: int64
```

Fig 2. Value count of churn column

- Target variable's class distribution is unbalanced. Negative class churn (churn = No) is much greater than positive class churn (churn = yes).

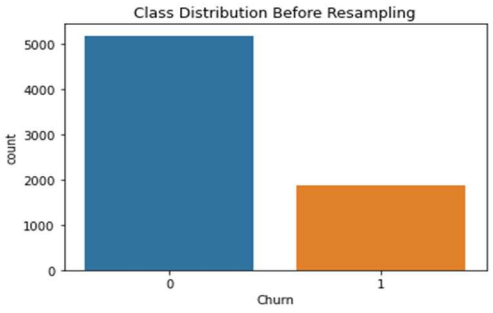

Fig3. Class distribution before resampling

Figure 3 illustrates imbalanced class distributions that affect a machine learning model's performance. So, to solve this problem, we can utilise up sampling or down sampling. Here we implemented up sampling and balanced the class distribution shown in Fig4 [5].

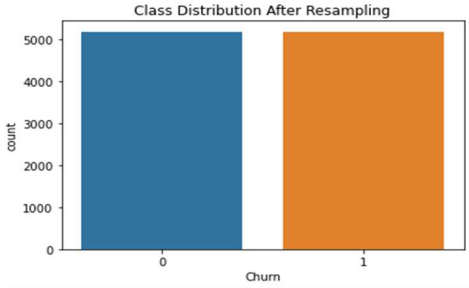

Fig4. Class distribution after resampling

- The dataset has a lot of features, but we only select the ones that are essential for improving the model's accuracy and helping us make decisions. The rest are given less weight. If the dataset contains only valuable and highly predictable variables, classification performance improves. Thus, limiting the amount of irrelevant attributes and using just significant features improves classification performance.
- Now in order to select the independent variables, we need to split all the variables into two categories.

1. Binary valued columns (only two values)

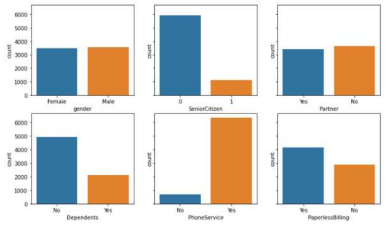

Fig5. Plots b/w binary columns & churn

By seeing Fig5, we can drop gender as it has no impact on the target variable.

2. Multi valued columns (more than two values)

(i) Customer ID is a unique identification number. This doesn't show any impact on target variable. So we can drop it.

(ii) Because the churn rate varies depending on which clients are using their services, all features related to internet services have various churn rates. As a result, the features in Fig. 6 have value for the model.

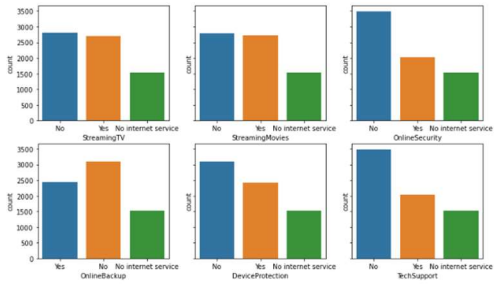

Fig6. Plots b/w service related columns & churn

(iii) According to Fig. 7, consumers with short-term contracts are more likely to leave. This indicates that maintaining a long-term relationship with clients is crucial for any business. You can remove this column.

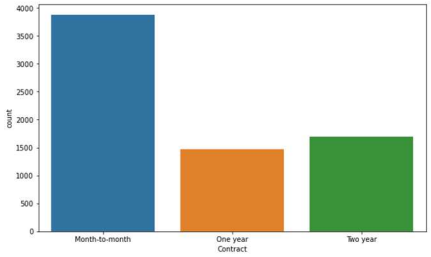

Fig7. Plots b/w contract column & churn

(iv) Fig8 shows the customers who pay with electronic check is the most common payment than any other types and are more likely to churn. So this segment may be further investigated.

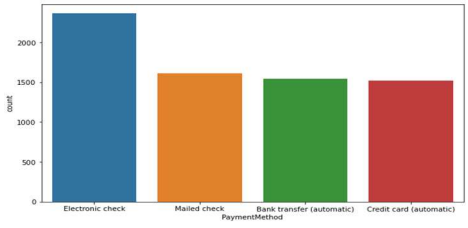

Fig8. Plot b/w payment method column & churn

(v) Tenure, monthly charges, and total charges are the continuous features. According to Fig. 9, the amount in the total charges column is proportionate to the tenure (months) X the monthly charges. Therefore, the model can ignore total charges.

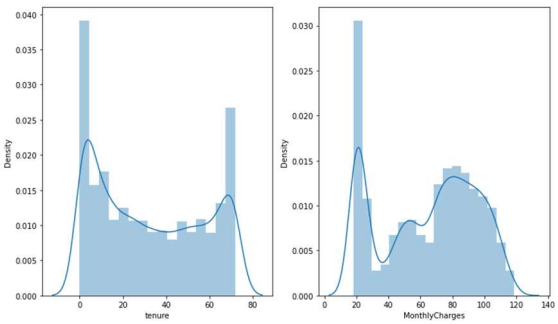

Fig9. Plot b/w payment method column & churn

(vi) As shown in Fig10 Phone Service and Multiple Lines columns are inter-related. A customer can't have multiple lines without phone service. So we can ignore phone services as it is covered in multiple lines.

```
Yes     6361
No       682
Name: PhoneService, dtype: int64


No                  3390
Yes                 2971
No phone service     682
Name: MultipleLines, dtype: int64
```

Fig10. Relation b/w phone Service & Multiple Lines

So now after the EDA [6], the columns Customer ID, Gender, Phone Service, Contract, Total Charges can be dropped from the data frame as these have no impact on the target variable.

Data Pre-processing:

- Categorical Feature Encoding: Many machine learning algorithms only work with numerical values, so categorical data need to be converted to numeric so that they can be included on machine learning models. In this paper, fifteen categorical features have been encoded using python library pandas utilizing get_dummies function to convert values to binary [7].
- Standardization: The numerical input variables need to be scaled to a standard range (0 to 1), inorder to get better performance for machine learning

algorithms. In this project, we applied MinMaxScalar transformation on tenure and monthly charges columns [8].

- Model Creation: As we selected the categorical variables, done with the encoding part and also done with resampling and balanced the dataset. Now the dataset is ready for the model creation. We must split the dataset into subgroups i.e.; train and test (80% for train, 20% for test). So that we are able to implement the models and measure the performance of our model [9]. By using sklearn library and importing appropriate model fit transform methods to implement the machine learning algorithms on train and test data. Also by importing sklearn library, the accuracy %, confusion matrix and classification report by importing the respective functions. To define the output quality, Receiver Operating Characteristic (ROC) curves are plotted, and AUC (Area under the ROC Curve) is determined [10]. In a similar manner, we used the other two techniques to determine which would yield more precise findings. Finally, for all three models, the ROC Curves are plotted and the AUC for test and train data is calculated.

V. RESULTS

Results from the model employing Gini Index and Entropy for Decision Tree Classifier are shown in Figs. 11 and 12.

```
Decision Tree with giniIndex Results
Predicted values: [1 0 1 ... 0 0 0]
Confusion Matrix:  [[713 331]
 [191 835]]
Accuracy :  74.78260869565217
Report :               precision    recall  f1-score   support

           0       0.79      0.68      0.73      1044
           1       0.72      0.81      0.76      1026

    accuracy                           0.75      2070
   macro avg       0.75      0.75      0.75      2070
weighted avg       0.75      0.75      0.75      2070
```

Fig11 Decision Tree Results using Gini Index

```
Decision Tree with entropy Results
Predicted values: [1 0 1 ... 0 0 0]
Confusion Matrix:  [[713 331]
 [191 835]]
Accuracy :  74.78260869565217
Report :               precision    recall  f1-score   support

           0       0.79      0.68      0.73      1044
           1       0.72      0.81      0.76      1026

    accuracy                           0.75      2070
   macro avg       0.75      0.75      0.75      2070
weighted avg       0.75      0.75      0.75      2070
```

Fig12 Decision Tree Results using Entropy

Figure 13 displays the outcomes of the experiment using logistic regression.

```
Logistic Regression Results
Confusion Matrix :
 [[783 261]
 [235 791]]
Accuracy :  76.03864734299518
Report :               precision    recall  f1-score   support

           0       0.77      0.75      0.76      1044
           1       0.75      0.77      0.76      1026

    accuracy                           0.76      2070
   macro avg       0.76      0.76      0.76      2070
weighted avg       0.76      0.76      0.76      2070
```

Fig13 Logistic Regression Results

Fig. 14 displays the outcomes of the experiment using the Random Forest Classifier.

```
Random Forest Classifier Results
Confusion Matrix :
 [[805 239]
 [ 94 932]]
ACCURACY OF THE MODEL:  83.91304347826087
Report :               precision    recall  f1-score   support

           0       0.90      0.77      0.83      1044
           1       0.80      0.91      0.85      1026

    accuracy                           0.84      2070
   macro avg       0.85      0.84      0.84      2070
weighted avg       0.85      0.84      0.84      2070
```

Fig14 Random Forest Results

Fig15, shows the ROC curves and AUC score for all three models using data.

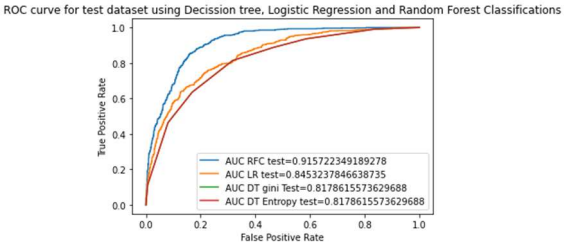

Fig15 ROC Curve and AUC Score for all three classifications

VI. DISCUSSIONS AND CONCLUSIONS

By using these classification algorithms we will get accuracy score and ROC AUC score to measure the quality of the output. Also plotted ROC curves for all three models. The curves red and green are overlapping as the accuracy values for Gini Index and Entropy for Decision tree are same. The orange curve is the results of Logistic regression and blue curve is the result of random forest classification. Here, we can obtain an accurate projection that will aid in identifying the customers moving on to other business offerings. By doing this, the telecom company can see everything clearly and offer them some exciting incentives to the customers to continue using the service. Table I demonstrate that our suggested churn model performed better and delivered better results by utilising machine learning approaches. Random Forest produced better accuracy among the various methods.

Classification Algorithms	Accuracy Score %	ROC_AUC score%
Random Forest	83.9%	91.5%
Logistic Regression	76%	84.5%
Decision Tree	74.7%	81.7%

Table I. Accuracy table

References

[1] K. Dahiya and S. Bhatia, "Customer churn analysis in telecom industry," in *IEEE*, Noida, India, 2015.

[2] "Decision Tree Classification in Python Tutorial," [Online]. Available: https://www.datacamp.com/tutorial/decision-tree-classification-python.

[3] J. Brownlee, "Logistic Regression for Machine Learning," 15 August 2020. [Online]. Available: https://machinelearningmastery.com/logistic-regression-for-machine-learning/. [Accessed 5 August 2022].

[4] N. Donges, "Random Forest Algorithm: A Complete Guide," 22 july 2021. [Online]. Available: https://builtin.com/data-science/random-forest-algorithm.

[5] S. Wang, L. L. Minku and X. Yao, "Resampling-Based Ensemble Methods for Online Class Imbalance Learning," *IEEE Transactions on Knowledge and Data Engineering ,* 05 August 2014.

[6] A. K. S. J. P. S. K. P. Kabita Sahoo, "Exploratory Data Analysis using Python," october 2019. [Online]. Available: https://www.ijitee.org/wp-content/uploads/papers/v8i12/L35911081219.pdf.

[7] "pandas.get_dummies," the pandas development team, [Online]. Available: https://pandas.pydata.org/docs/reference/api/pandas.get_dummies.html.

[8] "StandardScaler and MinMaxScalar," [Online]. Available: https://machinelearningmastery.com/standardscaler-and-minmaxscaler-transforms-in-python/.

[9] "sklearn.metrics.roc_auc_score," [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_auc_score.html#sklearn.metrics.roc_auc_score..

[10] "train-test-split," [Online]. Available: https://machinelearningmastery.com/train-test-split-for-evaluating-machine-learning-algorithms/.

APPENDIX A – DATASET

Dataset Name: Telco Customer Churn

Dataset Link: https://www.kaggle.com/datasets/blastchar/telco-customer-churn?sortBy=hotness&group=everyone&pageSize=20&datasetId=13996&language=Python

TABLE II. DESCRIPTION OF A SINGLE ENTRY IN A DATASET

Column Name	Description
CustomerID	Unique Identification number given to all customer
Gender	Whether the customer is a male or a female
SeniorCitizen	Whether the customer is old (1:Yes, 0:No)
Partner	Whether the customer has a partner (Yes,No)
Dependents	Whether the customer has dependents(Yes,No)
tenure	The number of months that the customer has stayed with the company
PhoneService	Whether the customer has phone service(Yes or No)
MultipleLines	Whether the customer has multiple lines(Yes,No,No phone service)
InternetService	Whether the customer has internet service(Yes, No, No internet service
OnlineSecurity	Whether the customer has online security(Yes, No, No internet service)
OnlineBackup	Whether the customer has online backup(Yes, No, No internet service)
DeviceProtection	Whether the customer has device protection(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 channel (Yes, No, No internet service)
StreamingMovies	Whether the customer has streaming Movies channel(Yes, No, No internet service)
Contract	Customer's contract duration(month-to-month, one year, two years)
PaperlessBiling	Whether customer has paperless invoice(Yes, No)
PaymentMethod	The customer payment method(Electronic check, mail, bank transfer(automatic),credit card(automatic)
MonthlyCharges	Monthly amount received from the customer
TotalCharges	Yearly amount received from the customer
Churn	Whether the customer churned or not (Yes or No)

The below figure shows the first 10 rows of the dataset.

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...	DeviceProtection	TechSupp
0	7590-VHVEG	Female	0	Yes	No	1	No	No phone service	DSL	No	...	No	
1	5575-GNVDE	Male	0	No	No	34	Yes	No	DSL	Yes	...	Yes	
2	3668-QPYBK	Male	0	No	No	2	Yes	No	DSL	Yes	...	No	
3	7795-CFOCW	Male	0	No	No	45	No	No phone service	DSL	Yes	...	Yes	
4	9237-HQITU	Female	0	No	No	2	Yes	No	Fiber optic	No	...	No	
5	9305-CDSKC	Female	0	No	No	8	Yes	Yes	Fiber optic	No	...	Yes	
6	1452-KIOVK	Male	0	No	Yes	22	Yes	Yes	Fiber optic	No	...	No	
7	6713-OKOMC	Female	0	No	No	10	No	No phone service	DSL	Yes	...	No	
8	7892-POOKP	Female	0	Yes	No	28	Yes	Yes	Fiber optic	No	...	Yes	
9	6388-TABGU	Male	0	No	Yes	62	Yes	No	DSL	Yes	...	No	

10 rows × 21 columns

APPENDIX B – Source code

➔ **Importing the necessary libraries, reading the dataset and performing Exploratory Data Analysis**

```
#Exploratory Data Analysis
#importing libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
#read the dataset
churn_data = pd.read_csv("WA_Fn-UseC_-Telco-Customer-Churn.csv")
churn_data.head(10)
churn_data.shape
#look for missing values
churn_data.isnull().sum()
#no missing values
#checking value count of target variables
churn_data['Churn'].value_counts()
# binary columns
columns = churn_data.columns
binary_cols = []
for col in columns:
    if churn_data[col].value_counts().shape[0] == 2:
        binary_cols.append(col)
# multiple columns
```

```
columns = churn_data.columns
multiple_cols = []
for col in columns:
    if churn_data[col].value_counts().shape[0] > 2:
        multiple_cols.append(col)
multiple_cols
fig, axes = plt.subplots(2, 3, figsize=(12, 7), sharey=True)
sns.countplot("gender", data=churn_data, ax=axes[0,0])
sns.countplot("SeniorCitizen", data=churn_data, ax=axes[0,1])
sns.countplot("Partner", data=churn_data, ax=axes[0,2])
sns.countplot("Dependents", data=churn_data, ax=axes[1,0])
sns.countplot("PhoneService", data=churn_data, ax=axes[1,1])
sns.countplot("PaperlessBilling", data=churn_data, ax=axes[1,2])
#replacing yes = 1 and no = 0 for churn column
churn_data.Churn.replace({"Yes":1, "No":0}, inplace = True)
#relation between binary cols and target variables
churn_data[['gender','Churn']].groupby(['gender']).mean()
churn_data[['SeniorCitizen','Churn']].groupby(['SeniorCitizen']).mean()
churn_data[['Partner','Churn']].groupby(['Partner']).mean()
churn_data[['Dependents','Churn']].groupby(['Dependents']).mean()
churn_data[['PhoneService','Churn']].groupby(['PhoneService']).mean()
churn_data[['PaperlessBilling','Churn']].groupby(['PaperlessBilling']).mean
()
#relation between multiple cols and target variables
sns.countplot("InternetService", data=churn_data)
churn_data[['InternetService','Churn']].groupby(['InternetService']).mean()
fig, axes = plt.subplots(2, 3, figsize=(12, 7), sharey=True)
sns.countplot("StreamingTV", data=churn_data, ax=axes[0,0])
sns.countplot("StreamingMovies", data=churn_data, ax=axes[0,1])
sns.countplot("OnlineSecurity", data=churn_data, ax=axes[0,2])
sns.countplot("OnlineBackup", data=churn_data, ax=axes[1,0])
sns.countplot("DeviceProtection", data=churn_data, ax=axes[1,1])
sns.countplot("TechSupport", data=churn_data, ax=axes[1,2])
churn_data[['StreamingTV','Churn']].groupby(['StreamingTV']).mean()
churn_data[['StreamingMovies','Churn']].groupby(['StreamingMovies']).mean()
churn_data[['OnlineSecurity','Churn']].groupby(['OnlineSecurity']).mean()
churn_data[['OnlineBackup','Churn']].groupby(['OnlineBackup']).mean()
churn_data[['DeviceProtection','Churn']].groupby(['DeviceProtection']).mean
()
churn_data[['TechSupport','Churn']].groupby(['TechSupport']).mean()
churn_data.PhoneService.value_counts()
churn_data.MultipleLines.value_counts()
churn_data[['MultipleLines','Churn']].groupby(['MultipleLines']).mean()
plt.figure(figsize=(10,6))
sns.countplot("Contract", data=churn_data)
churn_data[['Contract','Churn']].groupby(['Contract']).mean()
plt.figure(figsize=(10,6))
sns.countplot("PaymentMethod", data=churn_data)
churn_data[['PaymentMethod','Churn']].groupby(['PaymentMethod']).mean()
fig, axes = plt.subplots(1,2, figsize=(12, 7))
sns.distplot(churn_data["tenure"], ax=axes[0])
sns.distplot(churn_data["MonthlyCharges"], ax=axes[1])
churn_data[['tenure','MonthlyCharges','Churn']].groupby(['Churn']).mean()
#dropping unwanted columns
churn_data.drop(["customerID","gender","PhoneService","Contract","TotalChar
ges"], axis=1, inplace=True)
```

➔ **Data Pre-processing (Feature Encoding , Standardization and Resampling)**

```
#Categorical Feature Encoding
from sklearn.preprocessing import MinMaxScaler
cat_features = ['SeniorCitizen', 'Partner', 'Dependents',
'MultipleLines', 'InternetService','OnlineSecurity','OnlineBackup',
'DeviceProtection', 'TechSupport', 'StreamingTV',
'StreamingMovies', 'PaperlessBilling', 'PaymentMethod']
X = pd.get_dummies(churn_data, columns=cat_features, drop_first=True)
X.head()
sc = MinMaxScaler()
a = sc.fit_transform(churn_data[['tenure']])
b = sc.fit_transform(churn_data[['MonthlyCharges']])
X['tenure'] = a
X['MonthlyCharges'] = b
X.shape
sns.countplot('Churn', data=churn_data).set_title('Class Distribution
Before Resampling')
X_no = X[X.Churn == 0]
X_yes = X[X.Churn == 1]
X_yes_upsampled = X_yes.sample(n=len(X_no), replace=True, random_state=42)
print(len(X_yes_upsampled))
X_upsampled = X_no.append(X_yes_upsampled).reset_index(drop=True)
sns.countplot('Churn', data=X_upsampled).set_title('Class Distribution
After Resampling')
```

➔ **Splitting the data set into train and test**

```
#train-test split
from sklearn.model_selection import train_test_split
X = X_upsampled.drop(['Churn'], axis=1) #features (independent variables)
y = X_upsampled['Churn'] #target (dependent variable)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2,
random_state=42)
```

➔ **Decission Tree Classification using Gini Index and Entropy and calculating accuracy score, Confusion matrix, classification report , ROC curve and AUC score**

```
#Decission Tree classification
from sklearn.metrics import confusion_matrix
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
# Decision tree with giniIndex
clf_gini = DecisionTreeClassifier(criterion = "gini",random_state =
100,max_depth=3, min_samples_leaf=5)
# Performing training
clf_gini.fit(X_train, y_train)
# Predicton with giniIndex
y_pred = clf_gini.predict(X_test)
print("Decision Tree with giniIndex Results")
print("Predicted values:",y_pred)
print("Confusion Matrix: ",confusion_matrix(y_test, y_pred))
print ("Accuracy : ",accuracy_score(y_test,y_pred)*100)
print("Report : ",classification_report(y_test, y_pred))

#AUC & ROC CURVE
y_pred_proba = clf_gini.predict_proba(X_test)[::,1]
fpr_DTGTEST, tpr_DTGTEST, _ = metrics.roc_curve(y_test,  y_pred_proba)
```

```
auc_DTGTEST = metrics.roc_auc_score(y_test, y_pred_proba)

#create ROC curve
plt.plot(fpr_DTGTEST,tpr_DTGTEST,label="AUC DT ="+str(auc_DTGTEST))
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.legend(loc=4)
plt.title("ROC curve using Decission Tree with giniIndex")
plt.show()

# Decision tree with entropy
clf_entropy = DecisionTreeClassifier(criterion = "entropy", random_state =
100,max_depth = 3, min_samples_leaf = 5)
# Performing training
clf_entropy.fit(X_train, y_train)
y_pred = clf_entropy.predict(X_test)
print("")
print("Decision Tree with entropy Results")
print("Predicted values:",y_pred)
print("Confusion Matrix: ",confusion_matrix(y_test, y_pred))
print ("Accuracy : ",accuracy_score(y_test,y_pred)*100)
print("Report : ",classification_report(y_test, y_pred))

#AUC & ROC CURVE
y_pred_proba = clf_entropy.predict_proba(X_test)[::,1]
fpr_DTETEST, tpr_DTETEST, _ = metrics.roc_curve(y_test,  y_pred_proba)
auc_DTETEST = metrics.roc_auc_score(y_test, y_pred_proba)
#create ROC curve
plt.plot(fpr,tpr,label="AUC DT Entropy ="+str(auc_DTETEST))
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.legend(loc=4)
plt.title("ROC curve using Decission Tree with Entropy")
plt.show()
```

➔ **Logistic Regression Classification and calculating accuracy score, Confusion matrix, classification report , ROC curve and AUC score**

```
#Logistic Regression
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression(random_state = 0)
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
print("Logistic Regression Results")
print ("Confusion Matrix : \n", cm)
from sklearn.metrics import accuracy_score
print ("Accuracy : ", accuracy_score(y_test, y_pred)*100)
from sklearn.metrics import classification_report
print("Report : ",classification_report(y_test, y_pred))
#define metrics Logistic regression
from sklearn import metrics

y_pred_proba = classifier.predict_proba(X_test)[::,1]
fpr_lt, tpr_lt, _ = metrics.roc_curve(y_test,  y_pred_proba)
auc_lt = metrics.roc_auc_score(y_test, y_pred_proba)
```

```
#create ROC curve
plt.plot(fpr_lt,tpr_lt,label="AUC LR test="+str(auc_lt))
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.legend(loc=4)
plt.title("ROC curve using logistic regression")
plt.show()
```

➔ **Random Forest Classification and calculating accuracy score, Confusion matrix, classification report , ROC curve and AUC score**

```
#Random forest classifier
from sklearn.ensemble import RandomForestClassifier
clf_forest = RandomForestClassifier(n_estimators=100, max_depth=10)
clf_forest.fit(X_train, y_train)
pred = clf_forest.predict(X_test)
cm = confusion_matrix(y_test, pred)
print("Random Forest Classifier Results")
print ("Confusion Matrix : \n", cm)
print("ACCURACY OF THE MODEL: ", accuracy_score(y_test, pred)*100)
print("Report : ",classification_report(y_test, pred))

#define metrics
from sklearn import metrics
y_pred_proba_random = clf_forest.predict_proba(X_test)[::,1]
fpr_r, tpr_r, _ = metrics.roc_curve(y_test,  y_pred_proba_random)
auc_random_test = metrics.roc_auc_score(y_test, y_pred_proba_random)

#create ROC curve
plt.plot(fpr_r,tpr_r,label="AUC RFC ="+str(auc_random_test))
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.legend(loc=4)
plt.title("ROC curve using Random Forest Classification")
plt.show()
```

➔ **ROC curve and AUC score for Random Forest ,Decission Tree and Logistic Regression Classification 9**

```
#create ROC curve
plt.plot(fpr_r,tpr_r,label="AUC RFC="+str(auc_random_test))
plt.plot(fpr_lt,tpr_lt,label="AUC LR ="+str(auc_lt))
plt.plot(fpr,tpr,label="AUC DT Entropy ="+str(auc_DTETEST))
plt.plot(fpr_DTGTEST,tpr_DTGTEST,label="AUC DT gini="+str(auc_DTGTEST))

plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.legend(loc=4)
plt.title("ROC curve using Decission tree, Logistic Regression and Random
Forest Classifications")
plt.show()
```