CUSTOMER CHURN PREDICTION ANALYSIS

REPORT

Big Data Analytics (BITE411L)

in

B.Tech. – Information Technology

by

NISHYANTH NANDAGOPAL (21BIT0160)
ALWIN INFANT K (21BIT0683)
PREM T (21BIT0621)
VIKRAM (21BIT0235)
AKSHAYA (21BIT0416)

SCORE

School of Computer Science Engineering & Information Systems

FALL SEM 2024-25

TABLE OF CONTENTS

ABSTRACT:

Customer churn, the phenomenon where customers discontinue their engagement with a company, poses a substantial threat to revenue streams and necessitates proactive mitigation strategies. This paper explores the imperative for businesses to comprehend the underlying causes of customer attrition and to implement targeted retention measures. By identifying vulnerable customer segments and deploying tailored retention strategies, companies can staunch the outflow of clientele. Furthermore, leveraging data analytics and machine learning techniques enables predictive modelling of future churn, empowering businesses to pre-emptively intervene and preserve customer relationships. This study underscores the criticality of identifying churn predictors, particularly within the telecommunications sector, to optimize revenue generation and foster sustainable growth.

1.Model Description & Details

We aim to build a basic predictive model for anticipating customer churn using the customer_churn_dataset. Employing Python tools such as pandas for data manipulation and matplotlib for visualizations, we will explore the following classification algorithms:

K-Nearest Neighbors (KNN):

KNN is a non-parametric algorithm used for classification and regression tasks. It classifies data points based on the majority class among their k nearest neighbors in the feature space.

Logistic Regression:

Logistic Regression is a linear model widely utilized for binary classification tasks. It models the probability of a binary outcome using a logistic function, making it suitable for predicting customer churn.

Decision Tree Classifier:

Decision Tree Classifier partitions the feature space into disjoint regions and predicts the target variable by assigning the majority class in each region. It offers interpretability and handles non-linear relationships between features and target.

Random Forest Classifier:

Random Forest Classifier is an ensemble learning method that constructs multiple decision trees and combines their predictions through voting or averaging. It enhances prediction accuracy and reduces overfitting compared to individual decision trees.

Support Vector Machine (SVM):

SVM is a powerful supervised learning algorithm for classification and regression tasks. It identifies the optimal hyperplane that separates different classes in the feature space, maximizing the margin between classes.

For each algorithm, we will follow a standard workflow:

Data Preprocessing:

Handling missing values, encoding categorical variables, and scaling features if necessary.

Model Training:

Training the classification algorithm on the labeled data to learn the underlying patterns.

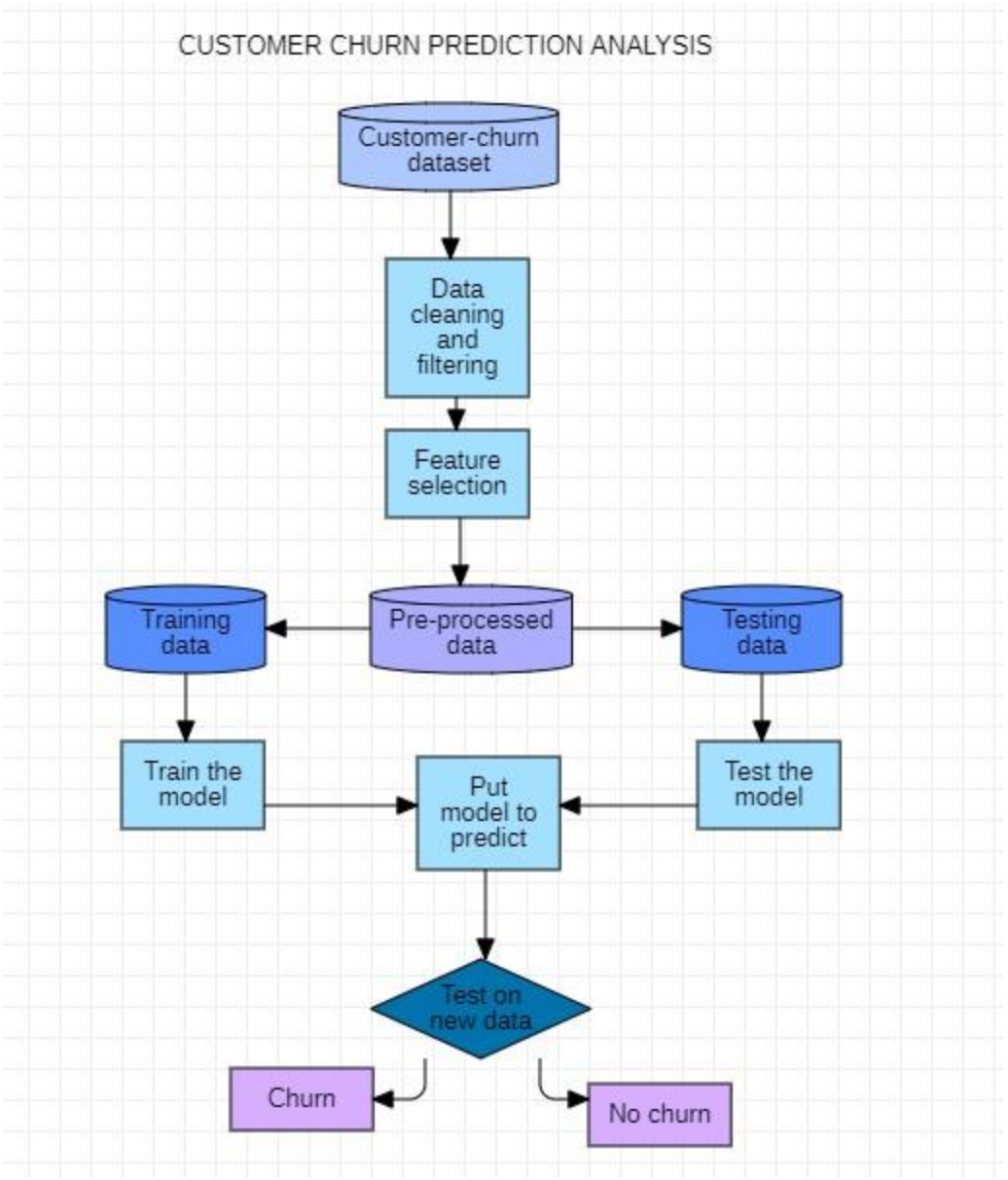
Model Evaluation:

Assessing the performance of the trained model using appropriate evaluation metrics such as accuracy, precision, recall, and F1-score.

By comparing the performance of these classification algorithms on the customer churn dataset, we aim to identify the most effective model for predicting and mitigating customer churn, thereby aiding businesses in retaining valuable clientele and maximizing revenue.

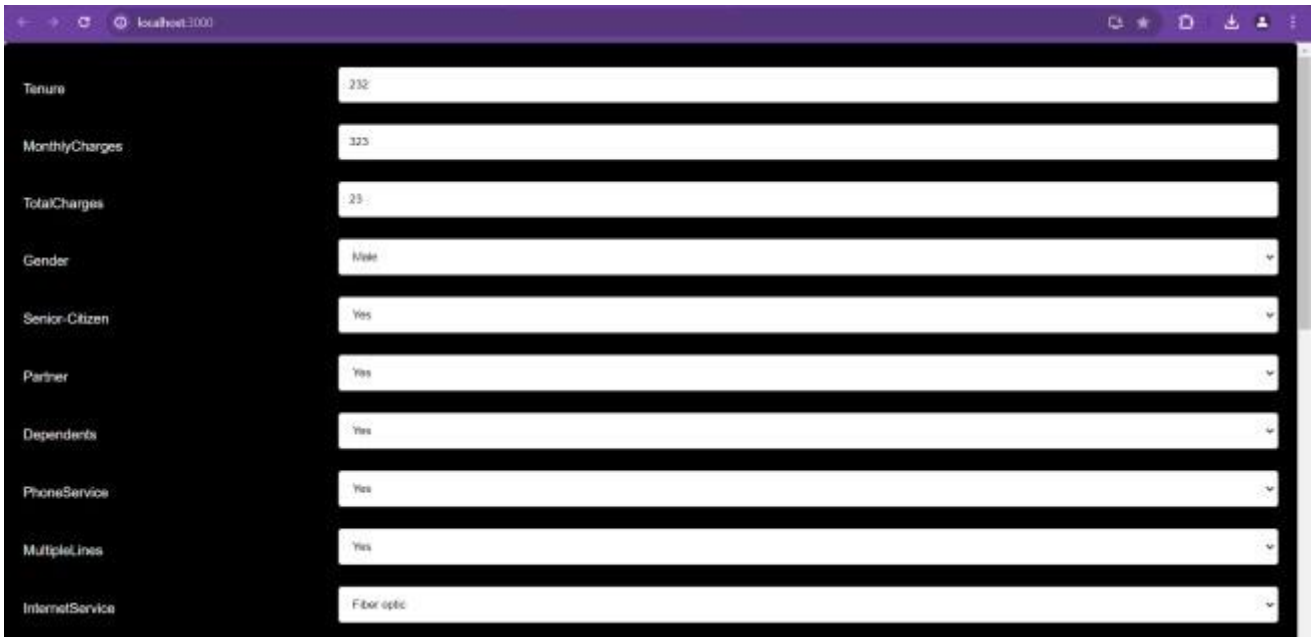
2. Model design

2.1 Flow Diagram



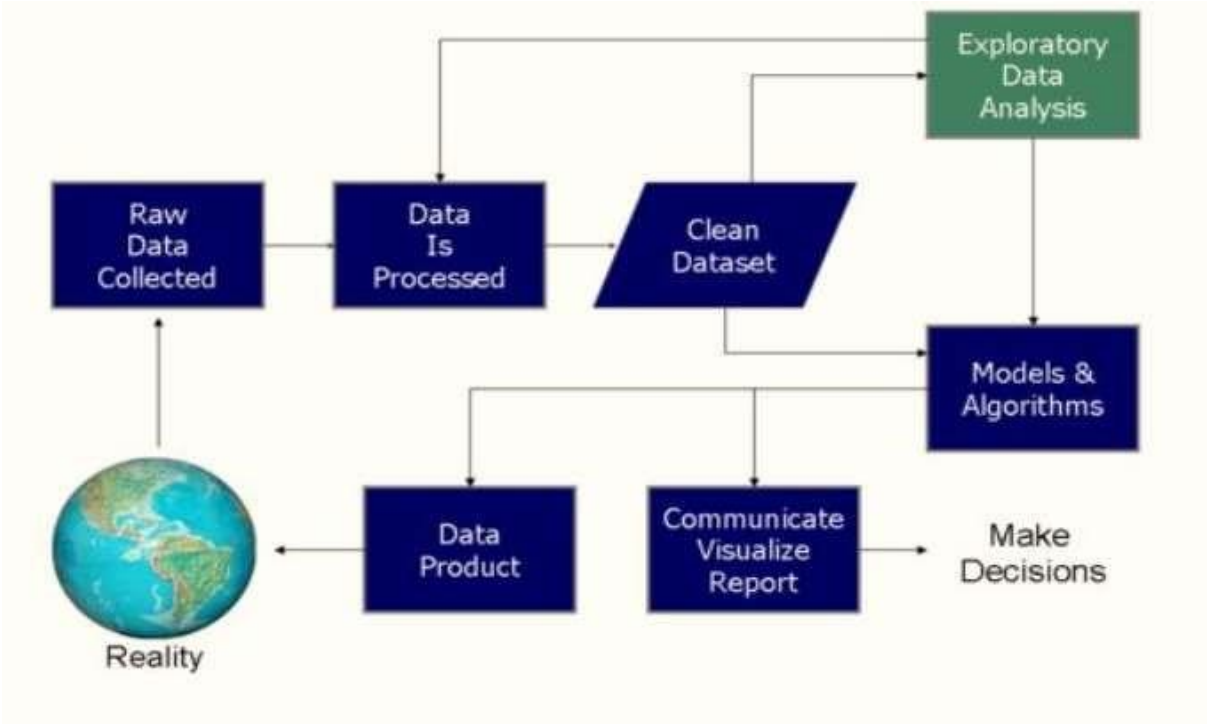
=

UI Analysis screenshots:



Before Input:

After Input:



3. Software Requirement Specifications

Software Requirement Specifications (SRS) outline the functional and non-functional requirements of the software system being developed. Here's an outline tailored to your project for predicting customer churn using classification algorithms:

4. Experimental Results & Discussion

5. 4.1.Source code:

```
import platform
import pandas as pd
import sklearn
import numpy as np
import graphviz
import seaborn as sns
import matplotlib
import matplotlib.pyplot as plt
import plotly.express as px
import plotly.graph_objects as go
import matplotlib.ticker as mtick
import math
from sklearn.preprocessing import OneHotEncoder, MinMaxScaler
from sklearn.model_selection import train_test_split, RandomizedSearchCV
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report, roc_curve, roc_auc_score
from sklearn.metrics import precision_recall_curve, auc, f1_score, ConfusionMatrixDisplay, precision_score, recall_score
from torch.utils.data import DataLoader
%matplotlib inline
```

```
# this dataset is in .csv form

df = pd.read_csv("/content/Customer-Churn.csv")
```

```
feature_le = ["Partner","Dependents","PhoneService", "Churn","PaperlessBilling"]
def label_encoding(df,features):
    for i in features:
        df[i] = df[i].map({"Yes":1, "No":0})
    return df

df = label_encoding(df,feature_le)
df["gender"] = df["gender"].map({"Female":1, "Male":0})
```

```
features_ohe = ["MultipleLines","InternetService","OnlineSecurity","OnlineBackup",
                "DeviceProtection","TechSupport","StreamingTV","StreamingMovies","Contract","PaymentMethod"]
df_ohe = pd.get_dummies(df, columns=features_ohe)
```

```
features_mms = ["tenure","MonthlyCharges","TotalCharges"]

df_mms = pd.DataFrame(df_ohe, columns=features_mms)
df_remaining = df_ohe.drop(columns=features_mms)

mms = MinMaxScaler(feature_range=(0,1))
rescaled_feature = mms.fit_transform(df_mms)

rescaled_feature_df = pd.DataFrame(rescaled_feature, columns=features_mms, index=df_remaining.index)
df = pd.concat([rescaled_feature_df,df_remaining],axis=1)
```

```
X = df.drop(columns = "Churn")
y = df.Churn

X_train,X_test, y_train, y_test = train_test_split(X,y,test_size=0.2,random_state=42,stratify=y)
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

Feature Selection

```
from sklearn.model_selection import train_test_split
X = scaled_features
Y = df1['Churn_Yes']
X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size = 0.3,random_state=44)
```

Prediction using Logistic Regression

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report,accuracy_score ,confusion_matrix

logmodel = LogisticRegression()
logmodel.fit(X_train,Y_train)
LogisticRegression()
```

```
print(classification_report(Y_test, predLR))
```

Decision Tree Classifier

```
model_dt=DecisionTreeClassifier(criterion = "gini",random_state = 100,max_depth=6, min_samples_leaf=8)
```

```
model_dt.fit(x_train,y_train)
```

```
DecisionTreeClassifier(max_depth=6, min_samples_leaf=8, random_state=100)
```

```
y_pred=model_dt.predict(x_test)
y_pred
```

```
model_dt.score(x_test,y_test)
```

```
print(classification_report(y_test, y_pred, labels=[0,1]))
```

```
sm = SMOTEENN()
X_resampled, y_resampled = sm.fit_sample(x,y)
```

```
xr_train,xr_test,yr_train,yr_test=train_test_split(X_resampled, y_resampled,test_size=0.2)
```

```
model_dt_smote=DecisionTreeClassifier(criterion = "gini",random_state = 100,max_depth=6, min_samples_leaf=8)
```

```
model_dt_smote.fit(xr_train,yr_train)
yr_predict = model_dt_smote.predict(xr_test)
model_score_r = model_dt_smote.score(xr_test, yr_test)
print(model_score_r)
print(metrics.classification_report(yr_test, yr_predict))
```

Random Forest Classifier

```
from sklearn.ensemble import RandomForestClassifier
```

```
model_rf=RandomForestClassifier(n_estimators=100, criterion='gini', random_state = 100,max_depth=6, min_samples_leaf=8)
```

```
model_rf.fit(x_train,y_train)
```

```
RandomForestClassifier(max_depth=6, min_samples_leaf=8, random_state=100)
```

```
y_pred=model_rf.predict(x_test)
```

```
model_rf.score(x_test,y_test)
```

```
print(classification_report(y_test, y_pred, labels=[0,1]))
```

```
sm = SMOTEENN()
X_resampled1, y_resampled1 = sm.fit_sample(x,y)
```

```
xr_train1,xr_test1,yr_train1,yr_test1=train_test_split(X_resampled1, y_resampled1,test_size=0.2)
```

```
model_rf_smote=RandomForestClassifier(n_estimators=100, criterion='gini', random_state = 100,max_depth=6, min_samples_leaf=8)
```

```
model_rf_smote.fit(xr_train1,yr_train1)
```

```
RandomForestClassifier(max_depth=6, min_samples_leaf=8, random_state=100)
```

```
yr_predict1 = model_rf_smote.predict(xr_test1)
```

```
model_score_r1 = model_rf_smote.score(xr_test1, yr_test1)
```

```
print(model_score_r1)
print(metrics.classification_report(yr_test1, yr_predict1))
```

Prediction using KNN Classifier

```
from sklearn.neighbors import KNeighborsClassifier

knn = KNeighborsClassifier(n_neighbors = 30)
knn.fit(X_train,Y_train)
```

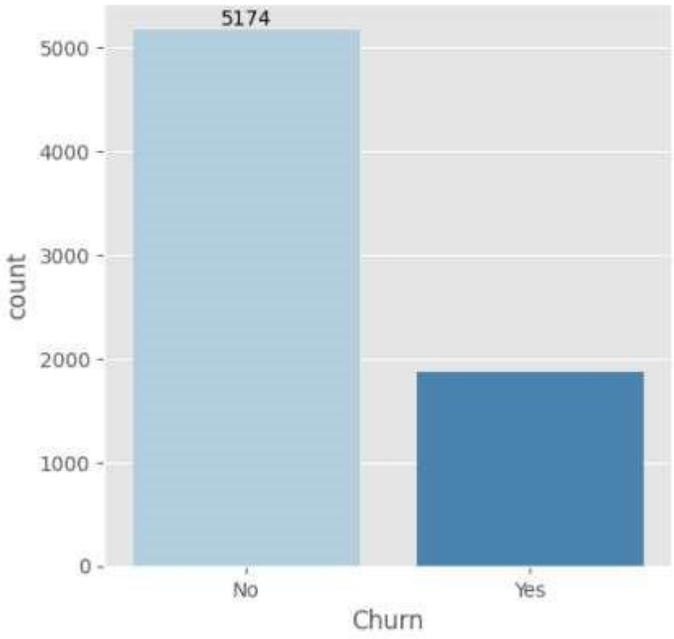
```
pred_knn = knn.predict(X_test)
```

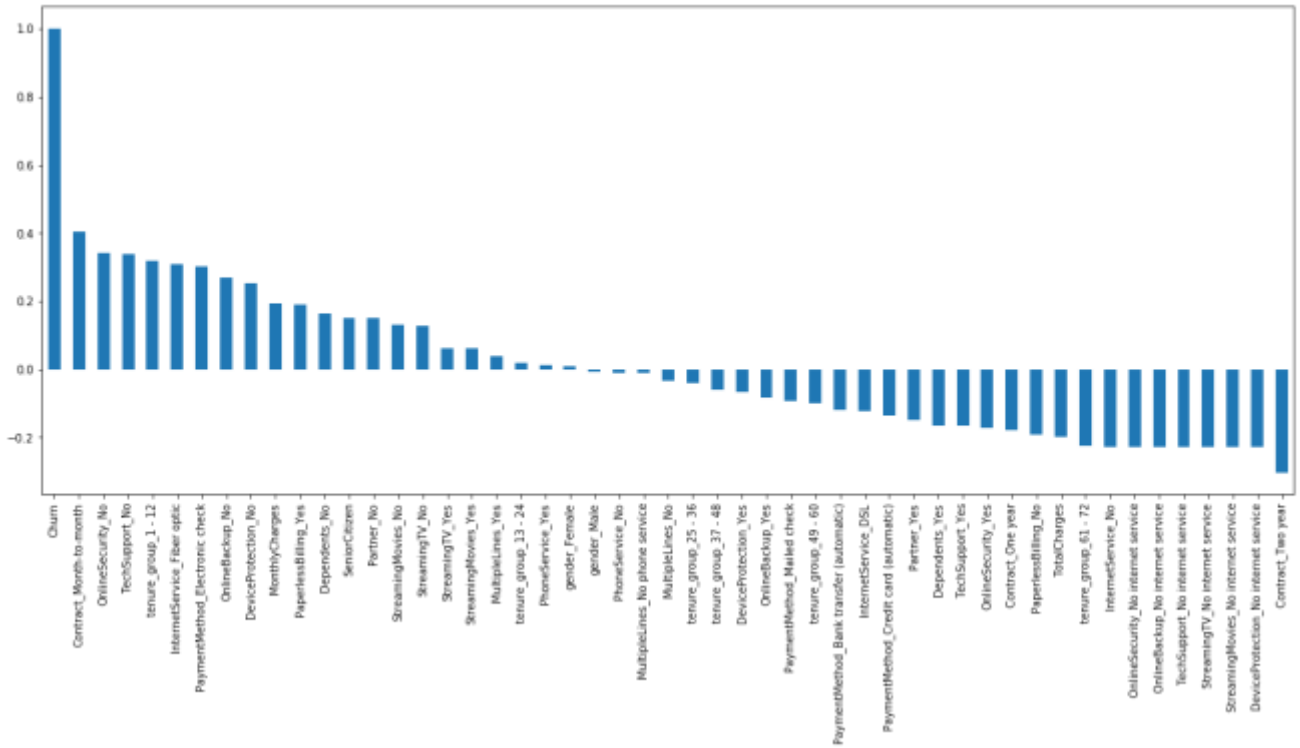
```
print(classification_report(Y_test,pred_knn))
```

```
accuracy_score(Y_test, pred_knn)
```

5.2 Screenshots with caption:

Target variable distribution:





HIGH Churn seen in case of Month-to-month contracts, No online security, No Tech support, First year of subscription and Fibre Optics Internet

LOW Churn is seen in case of long term contracts, Subscriptions without internet service and The customers engaged for 5+ years

Factors like Gender, Availability of Phone Service and # of multiple lines have almost NO impact on Churn

This is also evident from the Heatmap below:

Model comparison output:

1. Prediction using Logistic Regression

Accuracy Score: 0.8062

2. Prediction using Support Vector Classifier (SVC)

Accuracy Score: 0.8012

3. Prediction using Decision Tree Classifier

Accuracy Score: 0.77960

4. Prediction using KNN Cl

Accuracy Score: 0.7922

5. Random Forest Classifier

Accuracy Score: 0.7889

6. Artificial Neural Network

Accuracy Score: 0.776