SUPERVISED MACHINE LEARNING REINFORCEMENT PROJECT

Name	: M.Pavithra
Course	: DA/DS
Batch	: July 2025 A Cohort
Online/Offline	: Offline
Date	: 11/11/2025

Table of content

1. Introduction

Customer churn is a major challenge in the telecommunications industry, directly impacting revenue and business growth. The objective of this project is to analyze customer data from a telecom company to identify the key factors that influence customer churn. By applying machine learning techniques, the project aims to build a predictive model that can accurately classify whether a customer is likely to churn or remain with the company. This analysis will help the organization develop targeted retention strategies and improve customer satisfaction.

2.Aim

The aim of this project is to analyze telecom customer data and develop a machine learning model capable of predicting customer churn with high accuracy.
This project seeks to identify the key factors that influence customers' decisions to discontinue services and provide actionable insights to help the company reduce churn, enhance customer satisfaction, and improve overall business performance.
Additionally, the project aims to deploy the model using Streamlit, enabling real-time churn prediction for effective customer retention strategies.

3. Problem Statement

Customer churn is a persistent challenge in the telecommunications industry, directly affecting revenue growth and customer retention. Identifying customers who are likely to discontinue their services is essential for implementing effective retention strategies.

This project addresses the problem by analyzing telecom customer data to determine the key factors contributing to churn. By leveraging machine learning techniques, the project aims to develop a predictive model that can accurately classify customers as potential churners or loyal users. The insights derived from this analysis will enable the company to take proactive measures, improve customer satisfaction, and minimize churn-related losses.

4. Project Workflow

The project follows a systematic workflow to ensure accurate analysis, model development, and deployment. It begins with **data collection**, where the telecom customer dataset containing demographic, service, and account information is obtained. Next, during **data understanding**, the dataset is explored to identify its structure, data types, and feature distribution. The **data cleaning and preprocessing** phase involves handling missing values, removing unnecessary columns, encoding categorical variables, and scaling numerical features to prepare the data for analysis.

Following this, **exploratory data analysis (EDA)** is conducted using visualizations to uncover patterns, correlations, and key factors influencing customer churn. In the **model building** phase, various machine learning algorithms such as Logistic Regression, SVM, KNN, Decision Tree, and Random Forest are implemented to predict churn. The models are then assessed in the

model evaluation and selection stage using performance metrics like accuracy, precision, recall, and F1-score to identify the best-performing model.

Once the best model is chosen, **hyperparameter tuning** is applied to optimize its performance to determine which variables have the greatest impact on churn. The final model is then **deployed using Streamlit**, enabling interactive, real-time churn predictions based on user input. Finally, **result interpretation and insights** are derived to provide actionable recommendations that help the telecom company reduce customer churn and improve retention strategies.

5. Data Understanding

The dataset contains information about telecom customers, including their demographics, service details, and account information. It consists of both **categorical** and **numerical** features that help analyze customer behavior and identify churn patterns. The target variable is **"Churn"**, which shows whether a customer has left the service (Yes) or stayed (No).

Key numerical features include **tenure**, **MonthlyCharges**, and **TotalCharges**, while important categorical features include **gender**, **InternetService**, **Contract**, **OnlineSecurity**, **TechSupport**, and **PaymentMethod**. The dataset was explored to understand data types, check for missing values, and identify redundant columns. This understanding helped in selecting the most relevant features for building accurate churn prediction models.

```
data = pd.read_csv(r"C:\Users\WELCOME\Downloads\Telco_Customer_Churn.csv")
data
```

```
print("Shape of Data :", data.shape)
```

```
Shape of Data : (7043, 21)
```

```
print("\n Columns of Data :\n", data.columns)
```

```
 Columns of Data :
 Index(['customerID', 'gender', 'SeniorCitizen', 'Partner', 'Dependents',
        'tenure', 'PhoneService', 'MultipleLines', 'InternetService',
        'OnlineSecurity', 'OnlineBackup', 'DeviceProtection', 'TechSupport',
        'StreamingTV', 'StreamingMovies', 'Contract', 'PaperlessBilling',
        'PaymentMethod', 'MonthlyCharges', 'TotalCharges', 'Churn'],
       dtype='object')
```

```
print("\n Information of Data :\n")
data.info()
```

```
 Information of Data :

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7043 entries, 0 to 7042
Data columns (total 21 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   customerID        7043 non-null   object 
 1   gender            7043 non-null   object 
 2   SeniorCitizen     7043 non-null   int64  
 3   Partner           7043 non-null   object 
 4   Dependents        7043 non-null   object 
 5   tenure            7043 non-null   int64  
 6   PhoneService      7043 non-null   object 
 7   MultipleLines     7043 non-null   object 
 8   InternetService   7043 non-null   object 
 9   OnlineSecurity    7043 non-null   object 
 10  OnlineBackup      7043 non-null   object 
 11  DeviceProtection  7043 non-null   object 
 12  TechSupport       7043 non-null   object 
 13  StreamingTV       7043 non-null   object 
 14  StreamingMovies   7043 non-null   object 
 15  Contract          7043 non-null   object 
 16  PaperlessBilling  7043 non-null   object 
 17  PaymentMethod     7043 non-null   object 
 18  MonthlyCharges    7043 non-null   float64
 19  TotalCharges      7043 non-null   object 
 20  Churn             7043 non-null   object 
dtypes: float64(1), int64(2), object(18)
memory usage: 1.1+ MB
```

```
print("Data Types :\n")
print(data.dtypes)
```

```
Data Types :

customerID           object
gender               object
SeniorCitizen         int64
Partner              object
Dependents           object
tenure                int64
PhoneService         object
MultipleLines        object
InternetService      object
OnlineSecurity       object
OnlineBackup         object
DeviceProtection     object
TechSupport          object
StreamingTV          object
StreamingMovies      object
Contract             object
PaperlessBilling     object
PaymentMethod        object
MonthlyCharges      float64
TotalCharges         object
Churn                object
dtype: object
```

```
print("Missing values :\n")
print(data.isnull().sum())
```

```
Missing values :

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        0
Churn               0
dtype: int64
```

```
data.head()
```

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...	DeviceProtection	TechSup
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 rows × 21 columns

```
data.tail()
```

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...	DeviceProtection	Te
7038	6840-RESVB	Male	0	Yes	Yes	24	Yes	Yes	DSL	Yes	...	Yes	
7039	2234-XADUH	Female	0	Yes	Yes	72	Yes	Yes	Fiber optic	No	...	Yes	
7040	4801-JZAZL	Female	0	Yes	Yes	11	No	No phone service	DSL	Yes	...	No	
7041	8361-LTMKD	Male	1	Yes	No	4	Yes	Yes	Fiber optic	No	...	No	
7042	3186-AJIEK	Male	0	No	No	66	Yes	No	Fiber optic	Yes	...	Yes	

5 rows × 21 columns

6. Data Cleaning

Data cleaning was performed to prepare the dataset for accurate analysis and model building.Dropped duplicates in the dataset. Unnecessary columns such as customerID, Partner, Dependents, and other less useful features were removed to reduce noise. Missing values in the **TotalCharges** column were identified and filled with the median to maintain consistency. Duplicate records were checked and removed to prevent biased results. Categorical features were converted into numerical form using label encoding, and data types were corrected where needed. These steps ensured that the dataset was clean, consistent, and ready for preprocessing and modeling.

Checking for duplicates and there were no duplicates in the dataset.

```
# Dropping Duplicates
data.drop_duplicates()
```

Dropped unnecessary features and converted to appropriate data types.

```
# Dropping unimportant features
drop_cols = ['customerID',
             'SeniorCitizen',
             'Partner',
             'Dependents',
             'PhoneService',
             'OnlineBackup',
             'DeviceProtection',
             'StreamingTV',
             'StreamingMovies',
             'PaperlessBilling']

data.drop(drop_cols, axis=1, inplace=True)
```

```
# Changing appropriate Datatypes
data['TotalCharges'] = pd.to_numeric(data['TotalCharges'], errors='coerce')

cat_columns = ['gender',
               'MultipleLines',
               'OnlineSecurity',
               'Contract',
               'InternetService',
               'TechSupport',
               'PaymentMethod',
               'Churn']

data[cat_columns] = data[cat_columns].astype('category')
```

```
data.dtypes
```

```
gender             category
tenure                int64
MultipleLines      category
InternetService    category
OnlineSecurity     category
TechSupport        category
Contract           category
PaymentMethod      category
MonthlyCharges      float64
TotalCharges        float64
Churn              category
dtype: object
```

```
# Checking Null values
print(data.isnull().sum())
```

```
gender             0
tenure             0
MultipleLines      0
InternetService    0
OnlineSecurity     0
TechSupport        0
Contract           0
PaymentMethod      0
MonthlyCharges     0
TotalCharges      11
Churn              0
dtype: int64
```

Filling missing values in TotalCharges using fillna.

```
# Filling null values
data['TotalCharges'].fillna(data['TotalCharges'].median(), inplace=True)
```

```
# Rechecking null values
print("Missing values :",data.isnull().sum().sum())
```

```
Missing values : 0
```

Described data and the final rows and columns.

```
# Data Description
print("Data Description :")
data.describe()
```

```
Data Description :
```

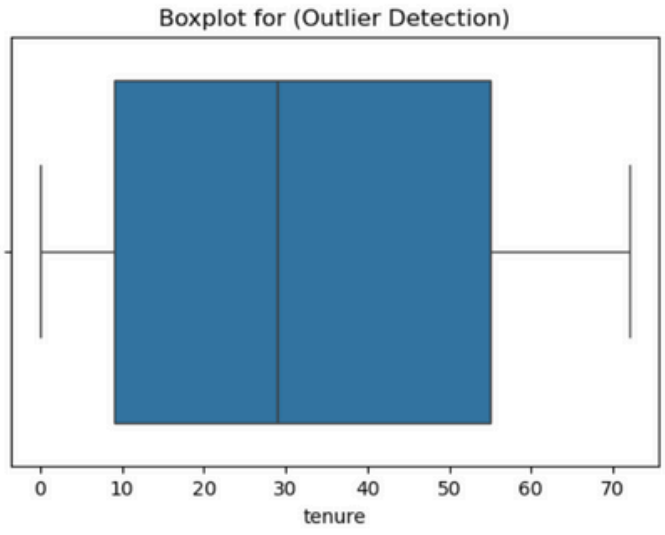
	tenure	MonthlyCharges	TotalCharges
count	7043.000000	7043.000000	7043.000000
mean	32.371149	64.761692	2281.916928
std	24.559481	30.090047	2265.270398
min	0.000000	18.250000	18.800000
25%	9.000000	35.500000	402.225000
50%	29.000000	70.350000	1397.475000
75%	55.000000	89.850000	3786.600000
max	72.000000	118.750000	8684.800000

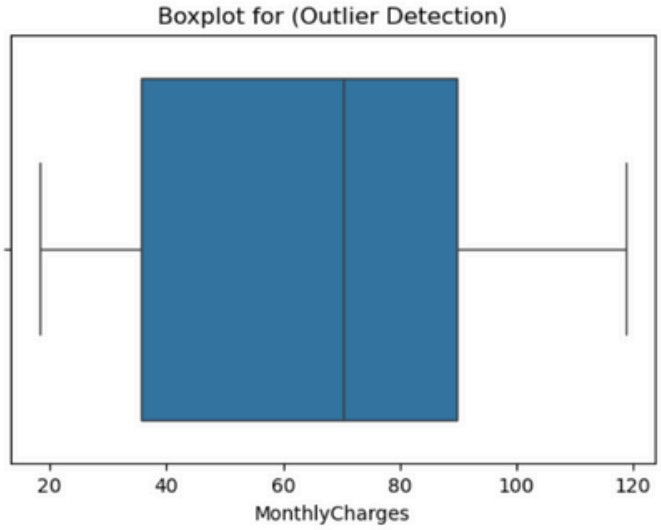
```
print("After Cleaning :", data.shape)
```

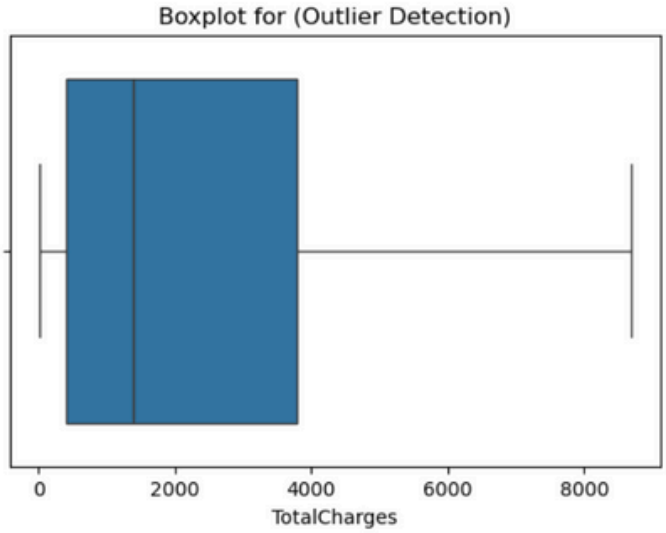
```
After Cleaning : (7043, 11)
```

Checked for outliers and there were no outliers in the dataset.

```
num_cols = ['tenure','MonthlyCharges','TotalCharges']
for col in num_cols:
    plt.figure(figsize=(6,4))
    sns.boxplot(x=data[col])
    plt.title("Boxplot for (Outlier Detection)")
    plt.show()
```







Insights

- Customers with tenure below the median (29 months) are more likely to churn.
- Those paying higher than the median monthly charge (more than 70) may churn due to high costs or lack of perceived value.
- Lower total charges usually belong to short-term or new customers, who are more likely to leave soon.
- Customers with higher tenure and total charges are loyal and less likely to churn, showing long-term satisfaction.

7. Feature Engineering

Feature engineering was applied to improve the quality of the data and enhance model performance. Categorical variables such as gender, InternetService, Contract, TechSupport, and PaymentMethod were converted into numerical values using Label Encoding, enabling machine learning models to interpret them correctly. Numerical features like tenure, MonthlyCharges, and TotalCharges were examined for missing values, with appropriate imputation performed where required.

Less meaningful or redundant columns were removed to reduce noise and simplify the dataset. Additionally, important features were selected based on their relationship with the target variable, helping the model focus on attributes that influence churn the most. Overall, feature engineering improved the dataset's structure and supported better predictive accuracy.

Label Encoding

```
from sklearn.preprocessing import LabelEncoder

label_encoder = LabelEncoder()
binary_cols = ['gender',
               'MultipleLines',
               'InternetService',
               'OnlineSecurity',
               'Contract',
               'TechSupport',
               'PaymentMethod',
               'Churn']

for col in binary_cols:
    data[col] = label_encoder.fit_transform(data[col])
print(data.head())
```

Feature Scaling

```
x = data.drop(columns='Churn', axis=1)
y = data['Churn']

from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(x)

standardized_data = scaler.transform(x)

x = standardized_data
y = data['Churn']
```

Data Splitting

```
from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42)
print(" Shape of x :", x.shape , "\n Shape of x_train :", x_train.shape ,"\n Shape of x_test :", x_test.shape)
```

```
Shape of x : (7043, 10)
Shape of x_train : (5634, 10)
Shape of x_test : (1409, 10)
```

8. Model Building

The following machine learning algorithms were applied:

- Logistic Regression – for baseline performance.
- Decision Tree Classifier
- Random Forest Classifier
- Support Vector Machine (SVM)
- K-Nearest Neighbors (KNN)
- XG Boost

1.LOGISTIC REGRESSION

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

lr = LogisticRegression(max_iter=1000)

# Model Training
lr.fit(x_train, y_train)

# Prediction
y_pred_lr = lr.predict(x_test)

# Model Evaluation
print("\n Logistic Regression Accuracy Score:", accuracy_score(y_test, y_pred_lr))
print("\n Logistic Regression Classification Report:\n", classification_report(y_test, y_pred_lr))
print("\n Logistic Regression Confusion Matrix:\n", confusion_matrix(y_test, y_pred_lr))
```

```
 Logistic Regression Accuracy Score: 0.8048261178140526

 Logistic Regression Classification Report:
               precision    recall  f1-score   support

           0       0.85      0.90      0.87      1036
           1       0.66      0.55      0.60       373

    accuracy                           0.80      1409
   macro avg       0.75      0.72      0.73      1409
weighted avg       0.80      0.80      0.80      1409


 Logistic Regression Confusion Matrix:
 [[929 107]
 [168 205]]
```

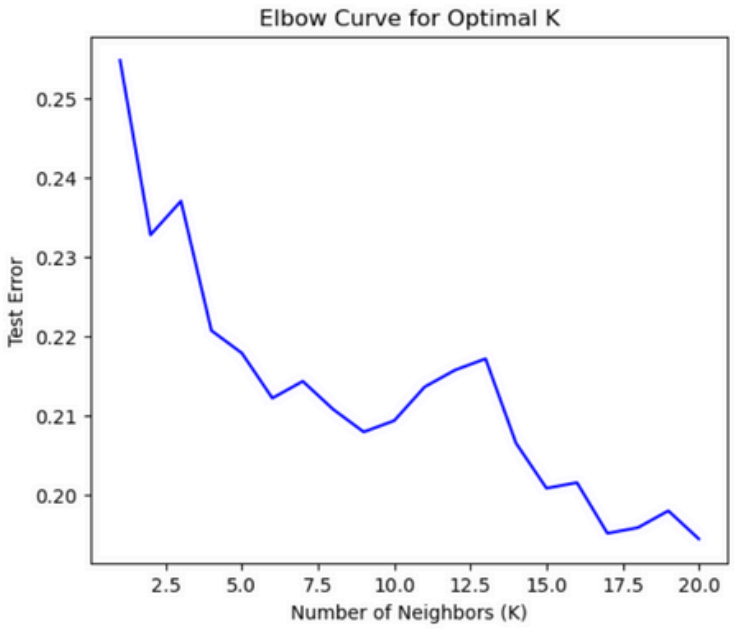
2.SUPPORT VECTOR MACHINE

```python
from sklearn.svm import SVC

svc = SVC()

# Model Training
svc.fit(x_train, y_train)

# Prediction
y_pred_svc = svc.predict(x_test)

# Model Evaluation
print("\n SVC Accuracy Score:", accuracy_score(y_test, y_pred_svc))
print("\n SVC Classification Report:\n", classification_report(y_test, y_pred_svc))
print("\n SVC Confusion Matrix:\n", confusion_matrix(y_test, y_pred_svc))
```

```
 SVC Accuracy Score: 0.8090844570617459

 SVC Classification Report:
               precision    recall  f1-score   support

           0       0.84      0.92      0.88      1036
           1       0.70      0.50      0.58       373

    accuracy                           0.81      1409
   macro avg       0.77      0.71      0.73      1409
weighted avg       0.80      0.81      0.80      1409


 SVC Confusion Matrix:
 [[955  81]
 [188 185]]
```

3.K-NEAREST NEIGHBOR

```python
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import f1_score

# Finding best K using Elbow Method
error = []

for k in range(1, 21):  # Test K = 1 to 21
    knn = KNeighborsClassifier(n_neighbors=k)
    knn.fit(x_train, y_train)
    y_pred = knn.predict(x_test)
    error.append(1 - accuracy_score(y_test, y_pred))

# Plot for Elbow Curve
plt.figure(figsize=(6,5))
plt.plot(range(1, 21), error, color='blue')
plt.title('Elbow Curve for Optimal K')
plt.xlabel('Number of Neighbors (K)')
plt.ylabel('Test Error')
plt.show()

# Best K (minimum error)
best_k = error.index(min(error)) + 1
print("\n Best K Value :", best_k)


# Model Training with Best K
knn = KNeighborsClassifier(n_neighbors=best_k)
knn.fit(x_train, y_train)

# Prediction
y_pred_knn = knn.predict(x_test)

# Model Evaluation
print("\n f1_score :", f1_score(y_test, y_pred_knn))
print("\n KNN Final Accuracy Score:", accuracy_score(y_test, y_pred_knn))
print("\n KNN Classification Report:\n", classification_report(y_test, y_pred_knn))
print("\n KNN Confusion Matrix:\n", confusion_matrix(y_test, y_pred_knn))
```



```
Best K Value : 20

f1_score : 0.573208722741433

KNN Final Accuracy Score: 0.8055358410220014

KNN Classification Report:
              precision    recall  f1-score   support

           0       0.83      0.92      0.87      1036
           1       0.68      0.49      0.57       373

    accuracy                           0.81      1409
   macro avg       0.76      0.71      0.72      1409
weighted avg       0.79      0.81      0.79      1409


KNN Confusion Matrix:
[[951  85]
[189 184]]
```

4.DECISION TREE

```python
from sklearn.tree import DecisionTreeClassifier

dt = DecisionTreeClassifier(random_state=42)

# Model Training
dt.fit(x_train, y_train)

# Predictions
y_pred_dt = dt.predict(x_test)

# Model Evaluation
print("\n DTC Accuracy Score :", accuracy_score(y_test, y_pred_dt))
print("\n DTC Classification Report :\n", classification_report(y_test, y_pred_dt))
print("\n DTC Confusion Matrix :\n", confusion_matrix(y_test, y_pred_dt))
```

```
DTC Accuracy Score : 0.7295954577714692

DTC Classification Report :
              precision    recall  f1-score   support

           0       0.82      0.81      0.81      1036
           1       0.49      0.52      0.50       373

    accuracy                           0.73      1409
   macro avg       0.66      0.66      0.66      1409
weighted avg       0.73      0.73      0.73      1409


DTC Confusion Matrix :
[[835 201]
[180 193]]
```

5.RANDOM FOREST

```python
from sklearn.ensemble import RandomForestClassifier

rfc = RandomForestClassifier(random_state=42)

# Model Training
rfc.fit(x_train, y_train)

# Prediction
y_pred_rfc = rfc.predict(x_test)

# Model Evaluation
print("\n RFC Accuracy Score :", accuracy_score(y_test, y_pred_rfc))
print("\n RFC Classification Report :\n", classification_report(y_test, y_pred_rfc))
print("\n RFC Confusion Matrix :\n", confusion_matrix(y_test, y_pred_rfc))
```

```
RFC Accuracy Score : 0.7970191625266146

RFC Classification Report :
              precision    recall  f1-score   support

           0       0.83      0.91      0.87      1036
           1       0.65      0.50      0.56       373

    accuracy                           0.80      1409
   macro avg       0.74      0.70      0.72      1409
weighted avg       0.79      0.80      0.79      1409


RFC Confusion Matrix :
[[938  98]
[188 185]]
```

6.XG BOOST

```python
from xgboost import XGBClassifier

xgb = XGBClassifier(use_label_encoder=False, eval_metric='logloss')

# Model Training
xgb.fit(x_train, y_train)

# Prediction
y_pred_xgb = xgb.predict(x_test)

# Model Evaluation
print("\n XGBoost Accuracy:", accuracy_score(y_test, y_pred_xgb))
print("\n XGBoost Classification Report :\n", classification_report(y_test, y_pred_xgb))
print("\n XGBoost Confusion Matrix :\n", confusion_matrix(y_test, y_pred_xgb))
```

```
XGBoost Accuracy: 0.7984386089425124

XGBoost Classification Report :
              precision    recall  f1-score   support

           0       0.84      0.90      0.87      1036
           1       0.65      0.51      0.57       373

    accuracy                           0.80      1409
   macro avg       0.74      0.71      0.72      1409
weighted avg       0.79      0.80      0.79      1409


XGBoost Confusion Matrix :
[[933 103]
[181 192]]
```

9. Model Evaluation

Model evaluation is a critical step to assess the performance of machine learning models and ensure they make accurate predictions. In this project, multiple evaluation metrics were considered:

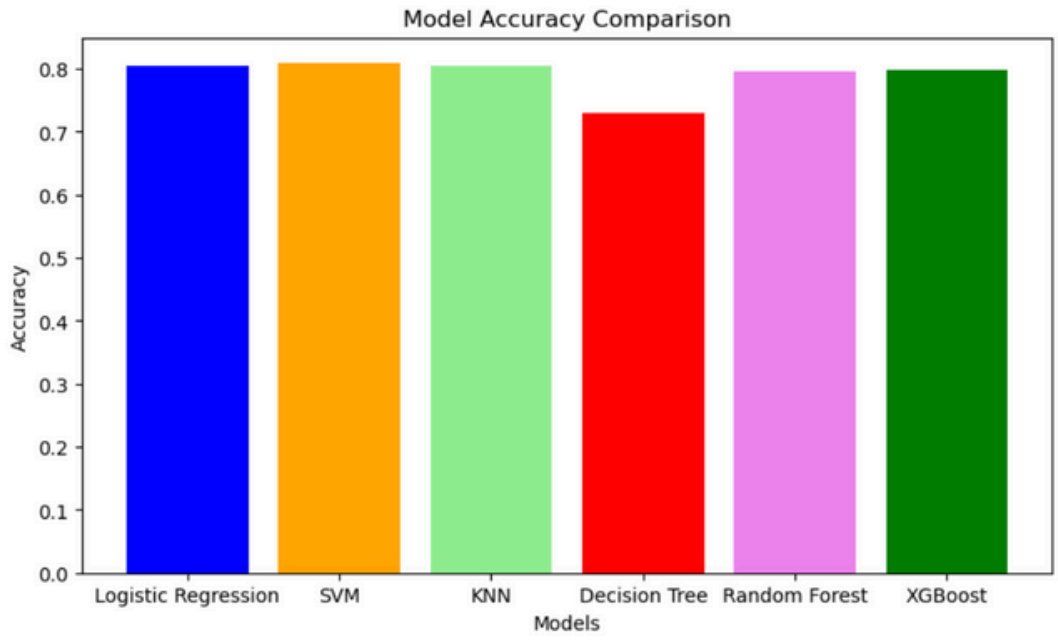
- **Accuracy**: Measures the overall correctness of the model, calculated as the ratio of correctly predicted cases to the total cases. While useful, accuracy alone can be misleading in imbalanced datasets.
- **Precision**: Represents the proportion of correctly predicted positive cases out of all predicted positives. High precision indicates few false positives.
- **Recall (Sensitivity)**: Measures the proportion of actual positive cases correctly identified. High recall is crucial in medical applications to minimize missed cases (false negatives).
- **F1-Score**: The harmonic mean of precision and recall, providing a balance between the two.

Among all models tested, **Support Vector Classifier** performed the best:

```
SVC Accuracy Score: 0.8090844570617459

SVC Classification Report:
              precision    recall  f1-score   support

           0       0.84      0.92      0.88      1036
           1       0.70      0.50      0.58       373

    accuracy                           0.81      1409
   macro avg       0.77      0.71      0.73      1409
weighted avg       0.80      0.81      0.80      1409


SVC Confusion Matrix:
[[955  81]
[188 185]]
```

COMPARISON OF ACCURACY OF ALL PERFORMED MODELS

	Model	Accuracy
0	Logistic Regression	0.804826
1	SVM	0.809084
2	KNN	0.805536
3	Decision Tree	0.729595
4	Random Forest	0.797019
5	XGBoost	0.798439



Here gives the best accuracy scored in **SUPPORT VECTOR CLASSIFIER**.

HYPERPARAMETER TUNING

Performed Hyperparameter tuning for the best performed model and there is not much difference after performing hyperparameter tuning.

GRIDSEARCHCV

```python
from sklearn.model_selection import GridSearchCV

param_grid = {
    'C': [0.1, 1, 10, 100],
    'kernel': ['linear', 'rbf'],
    'gamma': ['scale', 'auto']
}

print("\n GridSearchCV for Best Model \n")

# Perform GridSearchCV
grid_search = GridSearchCV(estimator=svc, param_grid=param_grid, cv=5,  n_jobs=-1, scoring='accuracy', verbose=4)
grid_search.fit(x_train, y_train)

# Best model and parameters
best_model = grid_search.best_estimator_
best_model.fit(x_train, y_train)

print("\n Best Parameters:", grid_search.best_params_)
print("\n Best Cross-Validation Accuracy:", grid_search.best_score_)

# Prediction
y_pred_svm = best_model.predict(x_test)

# Model Evaluation
print("\n SVM Best Test Accuracy:", accuracy_score(y_test, y_pred_svm))
print("\n SVM Classification Report:\n", classification_report(y_test, y_pred_svm))
print("\n SVM Confusion Matrix:\n", confusion_matrix(y_test, y_pred_svm))
```

```
 GridSearchCV for Best Model

Fitting 5 folds for each of 16 candidates, totalling 80 fits

 Best Parameters: {'C': 1, 'gamma': 'auto', 'kernel': 'rbf'}

 Best Cross-Validation Accuracy: 0.7880739352656654

 SVM Best Test Accuracy: 0.808374733853797

 SVM Classification Report:
              precision    recall  f1-score   support

           0       0.83      0.92      0.88      1036
           1       0.69      0.49      0.58       373

    accuracy                           0.81      1409
   macro avg       0.76      0.71      0.73      1409
weighted avg       0.80      0.81      0.80      1409


 SVM Confusion Matrix:
 [[955  81]
 [189 184]]
```

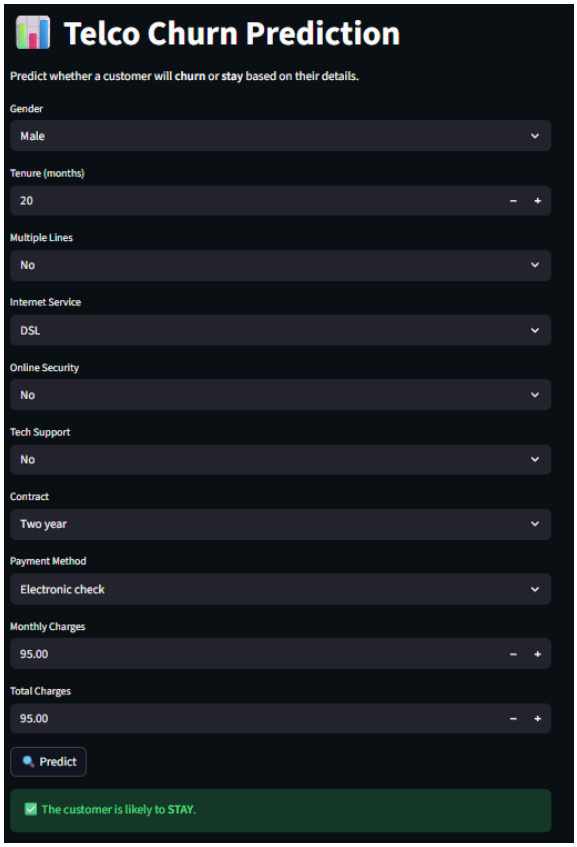
After performing all models and hyperparameter tuning , saved the best model, label encoder and scaler as pickle files using **Joblib** for model deployment.

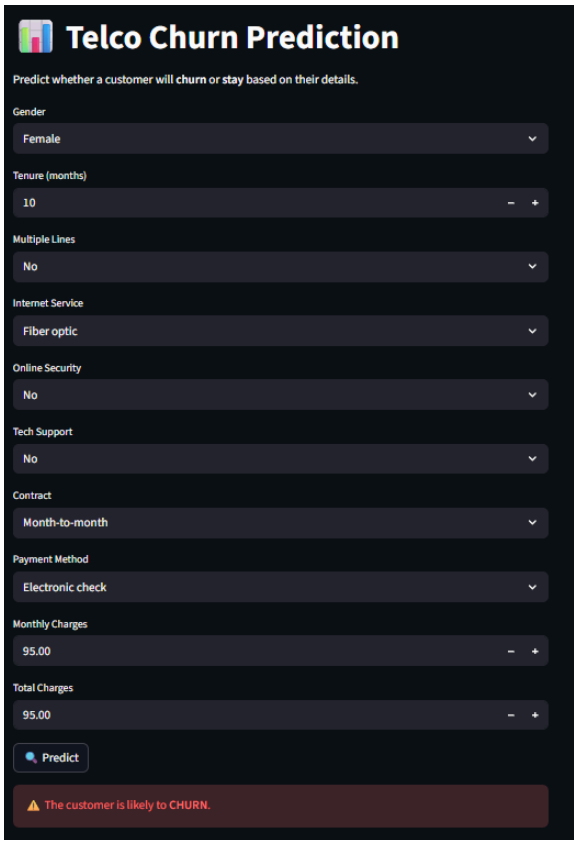
10. Streamlit Deployment:

In this project, Streamlit was used to deploy the churn prediction model, allowing real-time customer churn classification through a simple web interface.

The saved machine learning model (best_model.pkl), label encoder (label_encoders.pkl), and scaler (scaler.pkl) were loaded into the Streamlit application to ensure that the new input data is processed in the same way as the training data. The app collects user inputs such as gender, tenure, internet service type, contract type, payment method, and monthly charges through dropdown menus and numeric fields. These inputs are then encoded and scaled using the previously fitted transformations.

Once the user clicks the **Predict** button, the model analyzes the processed data and returns whether the customer is likely to **CHURN** or **STAY**, making the application practical, interactive, and suitable for business decision-making.





11. Exploratory Data Analysis (EDA)

Exploratory Data Analysis (EDA) was performed to understand the trends and patterns within the Telco Customer Churn dataset.

Univariate analysis : Individual feature lile Churn was studied to understand their distributions and common customer behaviors.

Bivariate analysis : Explored the relationship between each feature and churn, revealing that customers with short tenure and month-to-month contracts were more likely to churn.

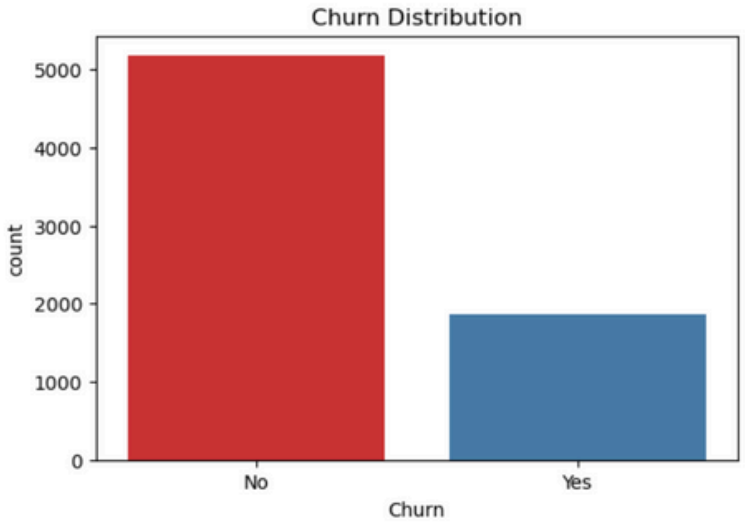
Multivariate analysis : Including correlation heatmaps and combined plots, helped identify how multiple factors together influence churn, such as the relationship between tenure, TotalCharges, and contract type.

Overall, EDA provided valuable insights that guided feature selection and improved model performance.

UNIVARIATE ANALYSIS :

Performed Univariate analysis of Churn distribution using countplot.

```
# Churn
plt.figure(figsize=(6, 4))
sns.countplot(x='Churn', data=data, palette='Set1')
plt.title('Churn Distribution')
plt.show()
```



Insights

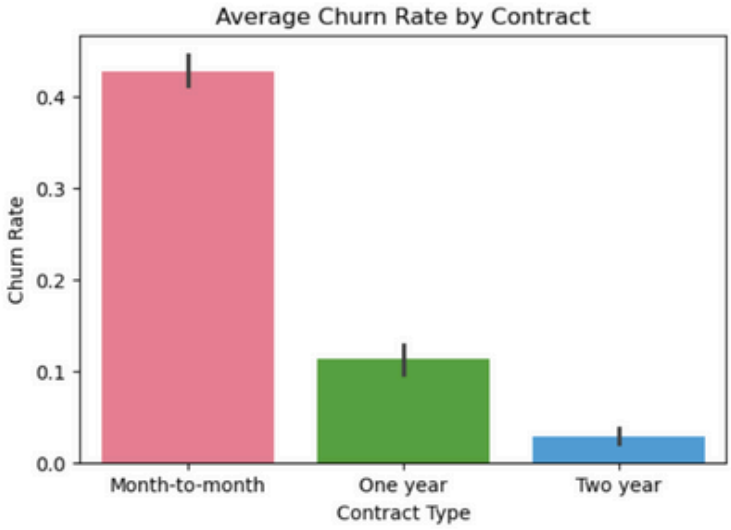
- The company has a high customer retention rate, as most customers have not churned.
- Customer churn is significantly lower
- The data is imbalanced, with more “No” churn cases than “Yes”.

BIVARIATE ANALYSIS

Analyzed from Bivariate analysis comparing two features.

Contract type vs Churn

```
# Contract Type vs Churn
data['Churn'] = data['Churn'].map({'Yes': 1, 'No': 0})
plt.figure(figsize=(6,4))
sns.barplot(x='Contract', y='Churn', data=data, palette='husl')
plt.title("Average Churn Rate by Contract")
plt.xlabel("Contract Type")
plt.ylabel("Churn Rate")
plt.gca().invert_yaxis()
plt.show()
```

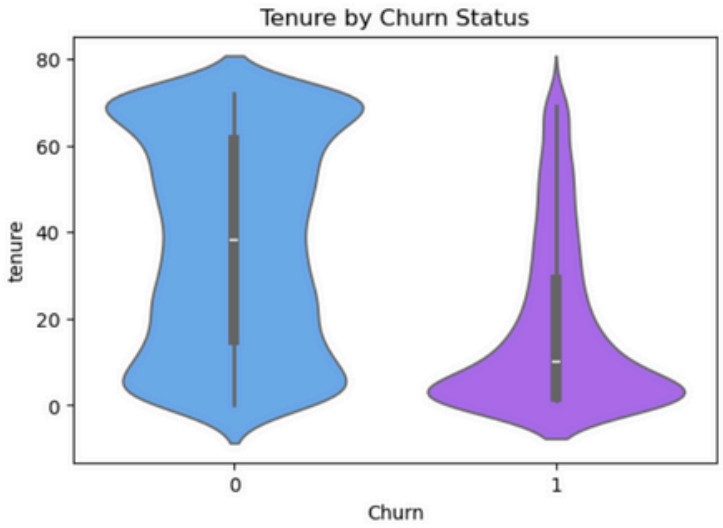


Insights

- Customers with month-to-month contracts have the highest churn rate, meaning they leave more often.
- One-year contract customers churn less, showing they are somewhat more loyal.
- Two-year contract customers have the lowest churn rate, showing strong retention.
- Longer contracts seem to reduce customer churn effectively.

Tenure vs Churn

```
# Tenure vs Churn
plt.figure(figsize=(6,4))
sns.violinplot(x='Churn', y='tenure', data=data, palette='cool')
plt.title("Tenure by Churn Status")
plt.show()
```



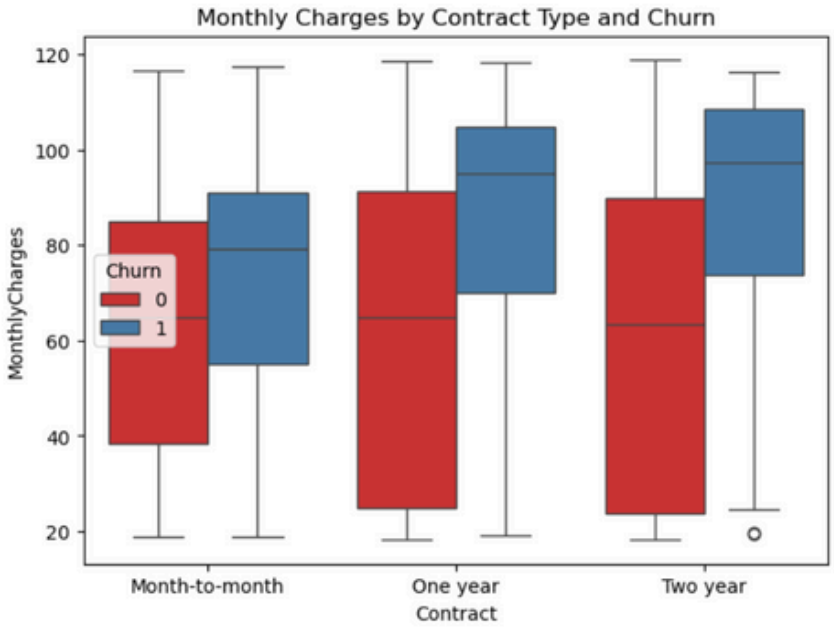
Insights

- The plot shows a high churn density at low tenure, so improving early customer experience can reduce.
- The churned customers (Yes), the median tenure is very low, meaning most churners leave early.
- The non-churn customers (No), the median tenure is higher, meaning most staying customers have been with the company longer.

MULTIVARIATE ANALYSIS

Compared Monthly Charges by Contract type and Churn.

```
# MonthlyCharges by Contract and Churn
plt.figure(figsize=(7,5))
sns.boxplot(x='Contract', y='MonthlyCharges', hue='Churn', data=data, palette='Set1')
plt.title("Monthly Charges by Contract Type and Churn")
plt.show()
```

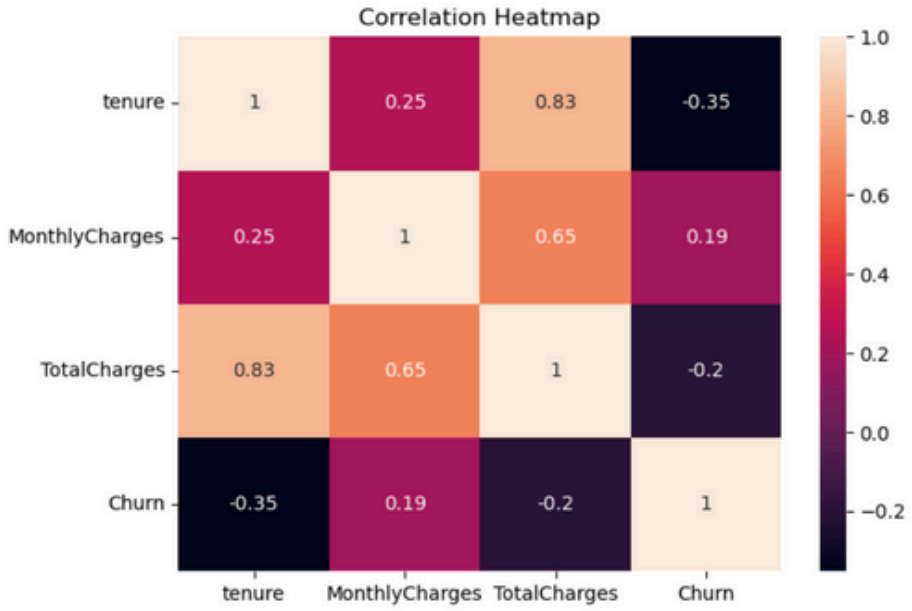


Insights

- Month-to-month customers who churn pay more every month, so high bills may make them leave.
- People with 1-year and 2-year contracts usually stay, even if their monthly charges are high.
- Long-term contracts reduce churn, because customers are more stable and less likely to leave.

CORRELATION HEATMAP

```
# Correlation Heatmap For Numerical Features - Tenure, MonthlyCharges, TotalCharges
plt.figure(figsize=(7,5))
corr_matrix = data[['tenure','MonthlyCharges','TotalCharges','Churn']].corr()
sns.heatmap(corr_matrix, annot=True)
plt.title("Correlation Heatmap")
plt.show()
```



Insights

- Tenure and Churn have a correlation of −0.35, meaning customers with shorter tenure are more likely to churn.
- MonthlyCharges and Churn have a weak positive correlation of 0.19, showing customers paying higher monthly charges churn slightly more.
- TotalCharges and Tenure have a strong positive correlation of 0.83, meaning customers who stay longer naturally accumulate higher total charges.

12. OVERALL INSIGHTS:

1. Objective

- The goal was to predict customer churn — identifying customers likely to discontinue telecom services.
- Helps the company take preventive actions and improve customer retention strategies.

2. Data Insights

- The dataset included customer demographics, contract details, payment methods, tenure, and charges.
- About 25–30% of customers were churners.

3. Key factors influencing churn

- Short tenure (newer customers).
- Month-to-month contracts.
- High monthly charges.
- Electronic check payments.
- Longer contracts and automatic payments were linked to lower churn.

4. Model Insights

- Models tested: Logistic Regression, SVM, KNN, Decision Tree, Random Forest, and XGBoost.
- Support vector Classifier performed best with 80.09% accuracy, slightly outperforming ensemble models.
- This suggests churn patterns are mostly linear, and logistic regression captured them effectively.
- Ensemble models (Random Forest, XGBoost) performed closely but didn’t exceed Support vector machine in accuracy.

5. Key Findings

- Customers with month-to-month contracts and higher monthly bills are most likely to leave.
- Loyal, long-term customers (with longer tenure) are less likely to churn.
- Payment type also influences churn — customers paying by electronic check churn more frequently.

13. Conclusion

This project successfully analyzed telecom customer data to identify the key factors that contribute to customer churn. Through detailed data cleaning, feature engineering, and exploratory data analysis, meaningful patterns were uncovered—showing that short tenure, high monthly charges, lack of security services, and month-to-month contracts are major drivers of churn. Multiple machine learning models were built and evaluated, with the best-performing model deployed using Streamlit for real-time prediction.

Overall, the project demonstrates how data-driven insights and predictive modeling can help telecom companies proactively identify high-risk customers, improve retention strategies, and make informed business decisions to reduce churn and enhance customer satisfaction.