Predicting Customer Churn in a Telecommunications Company

(Machine Learning Project Report)

1.Introduction:

In today's highly competitive telecom market, keeping customers happy and sticking with your service is crucial for a company's success. Customer churn happens when people decide to stop using a company's services and either switch to a competitor or just quit altogether. This isn't just about losing money; it's a sign that something might be wrong with the service or how customers feel about it.

The Telecom Customer Churn Analysis project is all about understanding why customers leave telecom companies. We're going to use fancy computer techniques and look at a lot of data to figure out what makes customers leave, predict when it might happen, and find ways to stop it from happening so much. It's like detective work for businesses, but instead of solving crimes, we're solving why people leave their phone plans.

2.Project Objectives:

Understanding Churn Dynamics: Explore the multifaceted landscape of customer churn, including demographic factors, service usage patterns, and contract preferences, to gain deeper insights into customer behavior.

Predictive Modeling: Develop robust machine learning models capable of accurately predicting customer churn based on historical data. This involves preprocessing data, feature engineering, model selection, and evaluation to achieve optimal predictive performance.

Insights and Recommendations: Translate model predictions and analysis findings into actionable insights and strategic recommendations for telecom companies to proactively manage churn and enhance customer retention strategies.

3.Dataset Overview:

The dataset utilized in this project comprises customer information collected by a telecom company. With over 7,044 customer records and 21 features capturing various aspects of customer interactions and account details, the dataset provides a rich foundation for exploring churn dynamics and building predictive models.

By embarking on this journey of telecom customer churn analysis, we aim to empower telecom companies with the tools and insights needed to navigate the challenges posed by churn and cultivate lasting relationships with their customer base.

Model Used

Model Selection:

1) Logistic Regression

2) Random Forest Classifier

1)Logistic Regression

- Description: Logistic regression is a linear classification model used to predict the probability of a binary outcome based on one or more predictor variables. It's a widely used algorithm due to its simplicity and interpretability.
- Suitability: Logistic regression is well-suited for binary classification tasks, making it an appropriate choice for predicting customer churn (binary outcome: churned or not churned). Additionally, its interpretability allows us to understand the impact of each feature on the likelihood of churn.
- Logistic regression estimates the probability that a given input instance belongs to a certain class (in this case, churned or not churned) by fitting a logistic function to the observed data.
- It outputs probabilities ranging from 0 to 1, with values closer to 1 indicating a higher probability of the positive class (churn) and values closer to 0 indicating a higher probability of the negative class (not churn).
- The model's coefficients provide insights into the direction and magnitude of the effect of each feature on the probability of churn.

2) Random Forest Classifier

- Description: Random forest is an ensemble learning method that constructs a multitude of decision trees during training and outputs the mode of the classes for classification tasks. It operates by building multiple decision trees and combining their predictions through voting.
- Suitability: Random forest is known for its ability to handle complex relationships and non-linear decision boundaries. With its robustness to overfitting and feature importance estimation, random forest is a suitable choice for predicting customer churn, especially when dealing with many features and potential interactions between them.
- Random forest operates by aggregating the predictions of multiple decision trees, each trained on a random subset of the training data and features.
- It is an ensemble method that combines the predictions of individual trees through voting or averaging, resulting in a robust and stable classifier.
- Random forest is effective in capturing complex relationships and interactions between features, making it suitable for tasks where the underlying data distribution is non-linear and high-dimensional.

Exploratory Data Analysis (EDA)

Data Overview:

- The dataset consists of customer demographics, account information, and service usage details.
- Key features include tenure, MonthlyCharges, TotalCharges, gender, SeniorCitizen, Partner, Dependents, PhoneService, MultipleLines, InternetService, OnlineSecurity, OnlineBackup, DeviceProtection, TechSupport, StreamingTV, StreamingMovies, Contract, PaperlessBilling, and PaymentMethod.

Initial Insights:

- Churn rate is around 26.5%.
- Features like tenure, MonthlyCharges, and TotalCharges show different distributions for churned and non-churned customers.

Visualizations:

- Distribution of Numerical Features: Histograms show the distribution of numerical features.
- Churn Distribution: Count plot showing the distribution of churned vs. non-churned customers.
- Numerical Features vs. Churn: KDE plots illustrating the relationship between numerical features and churn.
- Correlation Matrix: Heatmap displaying correlations between features.
- Gender Distribution: Bar plot showing churn distribution by gender.
- Payment Method Distribution: Bar plot showing churn distribution by payment method.

Model Evaluation

Evaluation Metrics:

For assessing the performance of the predictive models, the following evaluation metrics were utilized:

Accuracy: The proportion of correctly classified instances out of the total instances.

Precision: The proportion of true positive predictions out of all positive predictions made by the model.

Recall: The proportion of true positive predictions out of all actual positive instances in the dataset.

F1-Score: The harmonic mean of precision and recall, providing a balanced measure between the two.

Results: The performance metrics for the logistic regression and random forest classifiers are presented below:

Confusion Matrix: The confusion matrices for both models were displayed interactively in the Streamlit app. They provide a detailed breakdown of model predictions compared to the actual class labels, aiding in the interpretation of model performance across difference classes

Sno	Metric	Logistic Regression	Random Forest Accuracy
0	Accuracy	0.7861	0.7875
1	Precision	0.6189	0.6384
2	Recall	0.5080	0.4626
3	F1-Score	0.5580	0.5364

Interpretation of Results:

Logistic Regression:

- Accuracy (0.7861): Approximately 78.61% of the predictions made by the logistic regression model are correct.
- Precision (0.6189): Out of all the customers predicted to churn, around 61.89% actually did churn.
- Recall (0.5080): The model correctly identifies 50.80% of all actual churn cases.
- F1-Score (0.5580): The harmonic mean of precision and recall, which balances the trade-off between the two metrics, is 55.80%.

Random Forest:

- Accuracy (0.7875): Approximately 78.75% of the predictions made by the random forest model are correct.
- Precision (0.6384): Out of all the customers predicted to churn, around 63.84% actually did churn.
- Recall (0.4626): The model correctly identifies 46.26% of all actual churn cases.
- F1-Score (0.5364): The harmonic mean of precision and recall is 53.64%.

Confusion Matrices:

- Logistic Regression Confusion Matrix:
  - [999 112] 999 true negatives, 112 false positives
  - [209 216] 209 false negatives, 216 true positives
  - The model correctly classified 999 customers as not churned and 216 customers as churned. However, it incorrectly classified 112 customers as churned who did not churn and 209 customers as not churned who did churn.

- Random Forest Confusion Matrix:
  - [1043 68] 1043 true negatives, 68 false positives
  - [229 196] 229 false negatives, 196 true positives
  - The model correctly classified 1043 customers as not churned and 196 customers as churned. However, it incorrectly classified 68 customers as churned who did not churn and 229 customers as not churned who did churn.

Conclusion:

Both models exhibit similar accuracy, with the random forest model slightly outperforming logistic regression in precision but slightly underperforming in recall. The choice of model may depend on whether the focus is on minimizing false positives (precision) or capturing as many actual churn cases as possible (recall). Further tuning and model evaluation might be required to optimize performance for specific business needs.

Code

```
import streamlit as st
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score,
f1_score, confusion_matrix

# Load the dataset
df = pd.read_csv('WA_Fn-UseC_-Telco-Customer-Churn.csv')

# Handling missing values
df.dropna(inplace=True)  # Drop rows with missing values

# Convert TotalCharges to numeric, setting errors='coerce' will convert non-
convertible values to NaN
df['TotalCharges'] = pd.to_numeric(df['TotalCharges'], errors='coerce')

# Drop any remaining rows with NaN values
df.dropna(inplace=True)

# List of categorical columns
categorical_columns = ['gender', 'Partner', 'Dependents', 'PhoneService',
'MultipleLines', 'InternetService',
                       'OnlineSecurity', 'OnlineBackup', 'DeviceProtection',
'TechSupport', 'StreamingTV',
                       'StreamingMovies', 'Contract', 'PaperlessBilling',
'PaymentMethod']

# Manually map categorical columns to numeric values
mappings = {
    'gender': {'Female': 0, 'Male': 1},
    'Partner': {'No': 0, 'Yes': 1},
    'Dependents': {'No': 0, 'Yes': 1},
    'PhoneService': {'No': 0, 'Yes': 1},
    'MultipleLines': {'No': 0, 'Yes': 1, 'No phone service': 2},
    'InternetService': {'DSL': 0, 'Fiber optic': 1, 'No': 2},
    'OnlineSecurity': {'No': 0, 'Yes': 1, 'No internet service': 2},
    'OnlineBackup': {'No': 0, 'Yes': 1, 'No internet service': 2},
    'DeviceProtection': {'No': 0, 'Yes': 1, 'No internet service': 2},
    'TechSupport': {'No': 0, 'Yes': 1, 'No internet service': 2},
    'StreamingTV': {'No': 0, 'Yes': 1, 'No internet service': 2},
    'StreamingMovies': {'No': 0, 'Yes': 1, 'No internet service': 2},
```

```
    'Contract': {'Month-to-month': 0, 'One year': 1, 'Two year': 2},
    'PaperlessBilling': {'No': 0, 'Yes': 1},
    'PaymentMethod': {'Electronic check': 0, 'Mailed check': 1, 'Bank transfer
(automatic)': 2, 'Credit card (automatic)': 3}
}

for col in categorical_columns:
    df[col] = df[col].map(mappings[col])

# Convert the target variable 'Churn' to numeric
df['Churn'] = df['Churn'].map({'No': 0, 'Yes': 1})

# Feature Engineering: Creating a new feature 'SeniorCitizen_Partner'
df['SeniorCitizen_Partner'] = df['SeniorCitizen'] * df['Partner']

# Split the data into training and testing sets
X = df.drop(['Churn', 'customerID'], axis=1)
y = df['Churn']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Standardize the numerical features
numerical_columns = ['tenure', 'MonthlyCharges', 'TotalCharges']
scaler = StandardScaler()
X_train[numerical_columns] = scaler.fit_transform(X_train[numerical_columns])
X_test[numerical_columns] = scaler.transform(X_test[numerical_columns])

# Train a logistic regression model
logreg = LogisticRegression()
logreg.fit(X_train, y_train)
y_pred_logreg = logreg.predict(X_test)

# Train a random forest classifier
rf = RandomForestClassifier()
rf.fit(X_train, y_train)
y_pred_rf = rf.predict(X_test)

# Evaluation function
def evaluate_model(y_test, y_pred):
    accuracy = accuracy_score(y_test, y_pred)
    precision = precision_score(y_test, y_pred)
    recall = recall_score(y_test, y_pred)
    f1 = f1_score(y_test, y_pred)
    cm = confusion_matrix(y_test, y_pred)

    return accuracy, precision, recall, f1, cm
```

```
# Streamlit app
st.title("Telecom Customer Churn Analysis")

# Sidebar for navigation
st.sidebar.title("Navigation")
option = st.sidebar.radio("Select Analysis", ["Data Overview", "EDA", "Model
Evaluation", "Check Churn"])

# Data Overview
if option == "Data Overview":
    st.header("Data Overview")
    st.write("First 5 rows of the dataset:")
    st.write(df.head())
    st.write("Summary statistics of the dataset:")
    st.write(df.describe())

# Exploratory Data Analysis (EDA)
elif option == "EDA":
    st.header("Exploratory Data Analysis")
    eda_option = st.radio("Select EDA", ["Distribution of Numerical Features",
"Churn Distribution", "Numerical Features vs Churn", "Correlation Matrix",
"Gender Distribution","Payment Method Distribution"])

    if eda_option == "Distribution of Numerical Features":
        for col in numerical_columns:
            plt.figure(figsize=(10, 6))
            sns.histplot(df[col], kde=True, color='darkblue')
            plt.title(f'Distribution of {col}')
            st.pyplot(plt)
            st.write(df[col].describe())  # Summary statistics

    elif eda_option == "Churn Distribution":
        plt.figure(figsize=(10, 6))
        sns.countplot(x='Churn', data=df, color='darkblue')
        plt.title('Distribution of Churn')
        st.pyplot(plt)
        st.write(df['Churn'].value_counts())  # Summary statistics

    elif eda_option == "Numerical Features vs Churn":
        numerical_features = ['tenure', 'MonthlyCharges', 'TotalCharges']

        for feature in numerical_features:
            plt.figure(figsize=(10, 6))
            plot = sns.kdeplot(df[feature][(df['Churn'] == 0)], color="Red",
shade=True)
            plot = sns.kdeplot(df[feature][(df['Churn'] == 1)], ax=plot,
color="Blue", shade=True)
            plot.legend(["No Churn", "Churn"], loc='upper right')
```

```
            plot.set_ylabel('Density')
            plot.set_xlabel(feature)
            plot.set_title(f'{feature} by churn')
            st.pyplot(plt)
            st.write(df.groupby('Churn')[feature].describe())

    elif eda_option == "Correlation Matrix":
        plt.figure(figsize=(14, 10))
        correlation_matrix = df.drop(['customerID'], axis=1).corr()
        sns.heatmap(correlation_matrix, annot=True, fmt='.2f',
cmap='coolwarm')
        plt.title('Correlation Matrix')
        st.pyplot(plt)
        st.write(correlation_matrix)  # Summary statistics

    elif eda_option == "Gender Distribution":
        churn_by_gender = df.groupby(['gender', 'Churn']).size().unstack()
        churn_by_gender.plot(kind='bar', stacked=True, color=['lightblue',
'lightcoral'], figsize=(10, 6))
        plt.title('Churn Distribution by Gender')
        plt.xlabel('Gender')
        plt.ylabel('Count')
        plt.xticks([0, 1], ['Female', 'Male'], rotation=0)
        plt.legend(['No Churn', 'Churn'], loc='upper right')
        st.pyplot(plt)

    elif eda_option == "Payment Method Distribution":
    # Map numerical labels to payment method names
        payment_method_labels = {
        0: 'Electronic Check',
        1: 'Bank Transfer',
        2: 'Credit Card',
        3: 'Mailed Check'
    }

    # Count the churn distribution for each payment method
        churn_by_payment_method = df.groupby(['PaymentMethod',
'Churn']).size().unstack()

    # Rename the index to use payment method names
        churn_by_payment_method.index =
churn_by_payment_method.index.map(payment_method_labels)

    # Plot the churn distribution for each payment method
        churn_by_payment_method.plot(kind='bar', stacked=True, figsize=(10,
6), color=['lightblue', 'lightcoral'])
        plt.title('Churn Distribution by Payment Method')
        plt.xlabel('Payment Method')
```

```
        plt.ylabel('Count')
        plt.xticks(rotation=45, ha='right')
        plt.legend(['No Churn', 'Churn'])
        st.pyplot(plt)



# Model Evaluation
elif option == "Model Evaluation":
    st.header("Model Evaluation")

    logreg_accuracy, logreg_precision, logreg_recall, logreg_f1, logreg_cm =
evaluate_model(y_test, y_pred_logreg)
    rf_accuracy, rf_precision, rf_recall, rf_f1, rf_cm =
evaluate_model(y_test, y_pred_rf)

    st.write("### Logistic Regression Confusion Matrix")
    st.write(logreg_cm)

    st.write("### Random Forest Confusion Matrix")
    st.write(rf_cm)

    data = {
        "Metric": ["Accuracy", "Precision", "Recall", "F1-Score"],
        "Logistic Regression": [logreg_accuracy, logreg_precision,
logreg_recall, logreg_f1],
        "Random Forest": [rf_accuracy, rf_precision, rf_recall, rf_f1]
    }

    st.table(pd.DataFrame(data))

# Check Churn
# Check Churn
# Check Churn
elif option == "Check Churn":
    st.header("Predict Customer Churn")

    st.write("Please enter the following details:")

    # User input fields
    gender = st.selectbox("Gender", ['Female', 'Male'])
    senior_citizen = st.selectbox("Senior Citizen", [0, 1])
    partner = st.selectbox("Partner", ['No', 'Yes'])
    dependents = st.selectbox("Dependents", ['No', 'Yes'])
    tenure = st.number_input("Tenure (months)", min_value=0, max_value=72,
step=1)
    phone_service = st.selectbox("Phone Service", ['No', 'Yes'])
    multiple_lines = st.selectbox("Multiple Lines", ['No', 'Yes', 'No phone
service'])
```

```
    internet_service = st.selectbox("Internet Service", ['DSL', 'Fiber optic',
'No'])
    online_security = st.selectbox("Online Security", ['No', 'Yes', 'No
internet service'])
    online_backup = st.selectbox("Online Backup", ['No', 'Yes', 'No internet
service'])
    device_protection = st.selectbox("Device Protection", ['No', 'Yes', 'No
internet service'])
    tech_support = st.selectbox("Tech Support", ['No', 'Yes', 'No internet
service'])
    streaming_tv = st.selectbox("Streaming TV", ['No', 'Yes', 'No internet
service'])
    streaming_movies = st.selectbox("Streaming Movies", ['No', 'Yes', 'No
internet service'])
    contract = st.selectbox("Contract", ['Month-to-month', 'One year', 'Two
year'])
    paperless_billing = st.selectbox("Paperless Billing", ['No', 'Yes'])
    payment_method = st.selectbox("Payment Method", ['Electronic check',
'Mailed check', 'Bank transfer (automatic)', 'Credit card (automatic)'])
    monthly_charges = st.number_input("Monthly Charges", min_value=0.0)
    total_charges = st.number_input("Total Charges", min_value=0.0)

    # Map user input
    user_data = {
        'gender': mappings['gender'][gender],
        'SeniorCitizen': senior_citizen,
        'Partner': mappings['Partner'][partner],
        'Dependents': mappings['Dependents'][dependents],
        'tenure': tenure,
        'PhoneService': mappings['PhoneService'][phone_service],
        'MultipleLines': mappings['MultipleLines'][multiple_lines],
        'InternetService': mappings['InternetService'][internet_service],
        'OnlineSecurity': mappings['OnlineSecurity'][online_security],
        'OnlineBackup': mappings['OnlineBackup'][online_backup],
        'DeviceProtection': mappings['DeviceProtection'][device_protection],
        'TechSupport': mappings['TechSupport'][tech_support],
        'StreamingTV': mappings['StreamingTV'][streaming_tv],
        'StreamingMovies': mappings['StreamingMovies'][streaming_movies],
        'Contract': mappings['Contract'][contract],
        'PaperlessBilling': mappings['PaperlessBilling'][paperless_billing],
        'PaymentMethod': mappings['PaymentMethod'][payment_method],
        'MonthlyCharges': monthly_charges,
        'TotalCharges': total_charges,
        'SeniorCitizen_Partner': senior_citizen *
mappings['Partner'][partner]  # Include the derived feature
    }

    # Convert user input to dataframe
```

```
    user_df = pd.DataFrame(user_data, index=[0])

    # Standardize the numerical features
    user_df[numerical_columns] = scaler.transform(user_df[numerical_columns])

    # Predict churn
    if st.button("Check Churn"):
        churn_pred_logreg = logreg.predict(user_df)
        churn_pred_rf = rf.predict(user_df)

        # Use Random Forest prediction as final arbiter if models disagree
        if churn_pred_logreg[0] == churn_pred_rf[0]:
            final_churn = churn_pred_logreg[0]
        else:
            final_churn = churn_pred_rf[0]  # Trust Random Forest model

        st.write("### Prediction")
        st.write(f"Logistic Regression Prediction: {'Yes' if
churn_pred_logreg[0] == 1 else 'No'}")
        st.write(f"Random Forest Prediction: {'Yes' if churn_pred_rf[0] == 1
else 'No'}")
        st.write(f"Final Churn Prediction: {'Yes' if final_churn == 1 else
'No'}")
```

Summary of the code:

This Streamlit application performs a comprehensive analysis of customer churn in the telecom sector using logistic regression and random forest models. The code is structured into several main sections: data preprocessing, exploratory data analysis (EDA), model training and evaluation, and customer churn prediction.

1. Data Loading and Preprocessing:

- Data Loading: The dataset is loaded from a CSV file.
- Missing Values Handling: Rows with missing values are dropped.
- Data Type Conversion: The TotalCharges column is converted to numeric.
- Categorical Encoding: Categorical variables are manually mapped to numeric values using predefined mappings.
- Feature Engineering: A new feature, SeniorCitizen_Partner, is created by multiplying the SeniorCitizen and Partner columns.

2. Data Splitting and Scaling:

- Data Splitting: The dataset is split into training and testing sets.
- Feature Scaling: Numerical features (tenure, MonthlyCharges, TotalCharges) are standardized using StandardScaler.

3. Model Training:

- Logistic Regression Model: A logistic regression model is trained on the training data.
- Random Forest Model: A random forest classifier is also trained on the training data.

4. Model Evaluation:

- Evaluation Metrics: The accuracy, precision, recall, F1-score, and confusion matrix are computed for both models.
- Streamlit Interface: The evaluation results, including confusion matrices and performance metrics, are displayed in the Streamlit app.

5. Exploratory Data Analysis (EDA):

- Distribution Plots: The distribution of numerical features, churn distribution, and numerical features vs. churn are visualized using histograms and KDE plots.
- Correlation Matrix: The correlation matrix is displayed using a heatmap.
- Gender and Payment Method Distribution: Bar plots show the churn distribution by gender and payment method.

6. Customer Churn Prediction:

- User Input: The app collects user input for various features.
- Prediction: The user input is mapped and standardized. Both models predict churn, and if their predictions differ, the random forest model's prediction is used as the final output.
- Streamlit Interface: The app displays the predictions from both models and the final churn prediction.

Code Structure:

- Imports: Necessary libraries such as streamlit, pandas, matplotlib.pyplot, seaborn, sklearn modules, and others are imported.
- Data Loading and Preprocessing: This section includes reading the CSV file, handling missing values, encoding categorical variables, and feature engineering.
- Data Splitting and Scaling: The data is split into training and testing sets, and numerical features are scaled.
- Model Training: Logistic regression and random forest models are trained.
- Model Evaluation Function: A function to evaluate the models and return performance metrics and confusion matrices.
- Streamlit App: The app includes navigation options for data overview, EDA, model evaluation, and checking customer churn.

This structure ensures a comprehensive analysis of customer churn, providing valuable insights through EDA, model evaluation, and predictive capabilities.