# Predicting Customer Churn in Telecom Industry

By: Fahad Kabir

# Problem Understanding and Data Collection

- **Project Goals:** this project aims to predict customer churn in the telecom industry using historical customer data. By leveraging machine learning techniques, we seek to identify customers who are likely to leave the service, enabling proactive retention strategies.
- **Dataset Overview:** we'll utilize the Telco Customer Churn dataset for our analysis. This dataset provides valuable insights into customer demographics, services, and churn status.
- **Initial Data Exploration:** we conducted an initial exploration of the dataset to understand its structure and contents. This preliminary step laid the foundation for subsequent data preprocessing and analysis tasks.

# Data preprocessing

What we'll be doing in this step is Handling missing values, normalization/standardization, and encoding categorical variables.

- **Data Cleaning:**
    - Handled missing values to ensure data quality and integrity.
    - Address inconsistencies and errors in the dataset.
- **Normalization/Standardization:**
    - Scaled numerical features to ensure uniformity and improve model performance.
    - Standardized features to have a mean of 0 and a standard deviation of 1.
- **Encoding Categorical Variables:**
    - Converted categorical variables into numerical format using one-hot encoding.
    - Ensured all categorical data is properly represented for machine learning algorithms.

# Importing the dataset

```python
#Import libraries
import pandas as pd
import os

#Load the dataset
file_path = 'telcochurn.csv'
df = pd.read_csv(file_path)

#Display the first few rows of the dataset
df.head()
```

✓ 0.0s Python

| | customerID | gender | SeniorCitizen | Partner | Dependents | tenure | PhoneService | MultipleLines | InternetService | OnlineSecurity | ... | DeviceProtectio |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 7590-VHVEG | Female | 0 | Yes | No | 1 | No | No phone service | DSL | No | ... | N |
| 1 | 5575-GNVDE | Male | 0 | No | No | 34 | Yes | No | DSL | Yes | ... | Ye |
| 2 | 3668-QPYBK | Male | 0 | No | No | 2 | Yes | No | DSL | Yes | ... | N |
| 3 | 7795-CFOCW | Male | 0 | No | No | 45 | No | No phone service | DSL | Yes | ... | Ye |
| 4 | 9237-HQITU | Female | 0 | No | No | 2 | Yes | No | Fiber optic | No | ... | N |

5 rows × 21 columns

# Data Preprocessing

```
[ ] #Check for missing values
    df.isnull().sum()
    #Handle missing values
    for column in df.columns:
        if df[column].dtype == 'object':
            df[column].fillna(df[column].mode()[0], inplace=True)
        else:
            df[column].fillna(df[column].median(), inplace=True)

    #Encoding categorical variables
    df_encoded = pd.get_dummies(df, drop_first=True)

    #Normalization/Standardization
    from sklearn.preprocessing import StandardScaler
    scaler = StandardScaler()
    numerical_features = df_encoded.select_dtypes(include=['float64', 'int64']).columns
    df_encoded[numerical_features] = scaler.fit_transform(df_encoded[numerical_features])

    #Display first few rows of the dataset
    df_encoded.head()
```

| | SeniorCitizen | tenure | MonthlyCharges | customerID_0003-MKNFE | customerID_0004-TLHLJ | customerID |
|---|---|---|---|---|---|---|
| 0 | -0.439916 | -1.277445 | -1.160323 | False | False | |
| 1 | -0.439916 | 0.066327 | -0.259629 | False | False | |
| 2 | -0.439916 | -1.236724 | -0.362660 | False | False | |
| 3 | -0.439916 | 0.514251 | -0.746535 | False | False | |
| 4 | -0.439916 | -1.236724 | 0.197365 | False | False | |

5 rows × 13602 columns

# Exploratory Data Analysis (EDA)

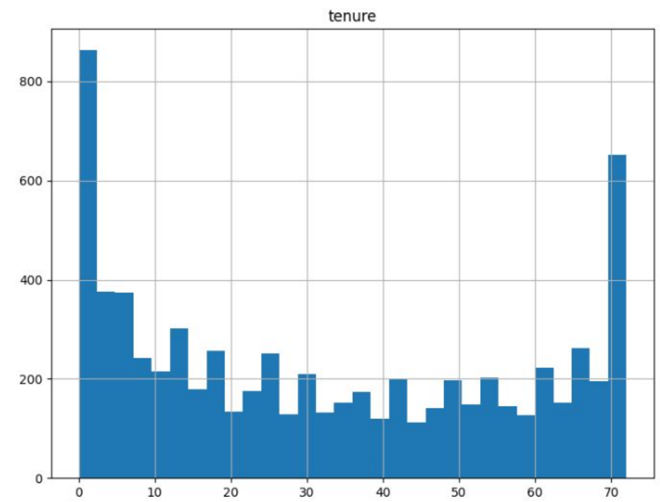
- **Conduct thorough EDA to understand data patterns and relationships.**
  - Analyzed customer demographics and service usage.
  - Examined the distribution of tenure, monthly charges, and total charges.
  - Investigated correlations between different features and churn.

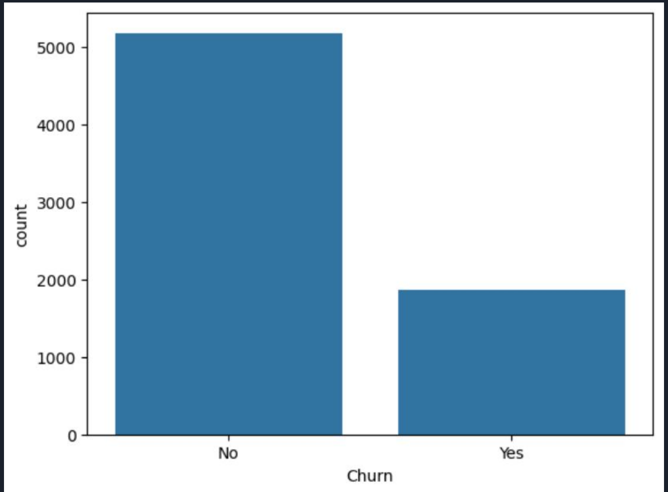
- **Visualization Tools:**
  - Utilized a few different types of graphs to visualize data.
  - Used matplotlib to code each graph.

# Code and visual graphs

```python
# Import necessary libraries for visualization
import matplotlib.pyplot as plt
import seaborn as sns

# Descriptive statistics
df.describe()

# Visualization: distribution of numerical features
df.hist(bins=30, figsize=(20, 15))
plt.show()

# Check the distribution of the target variable
sns.countplot(x='Churn', data=df)
plt.show()
```

✓ 0.5s

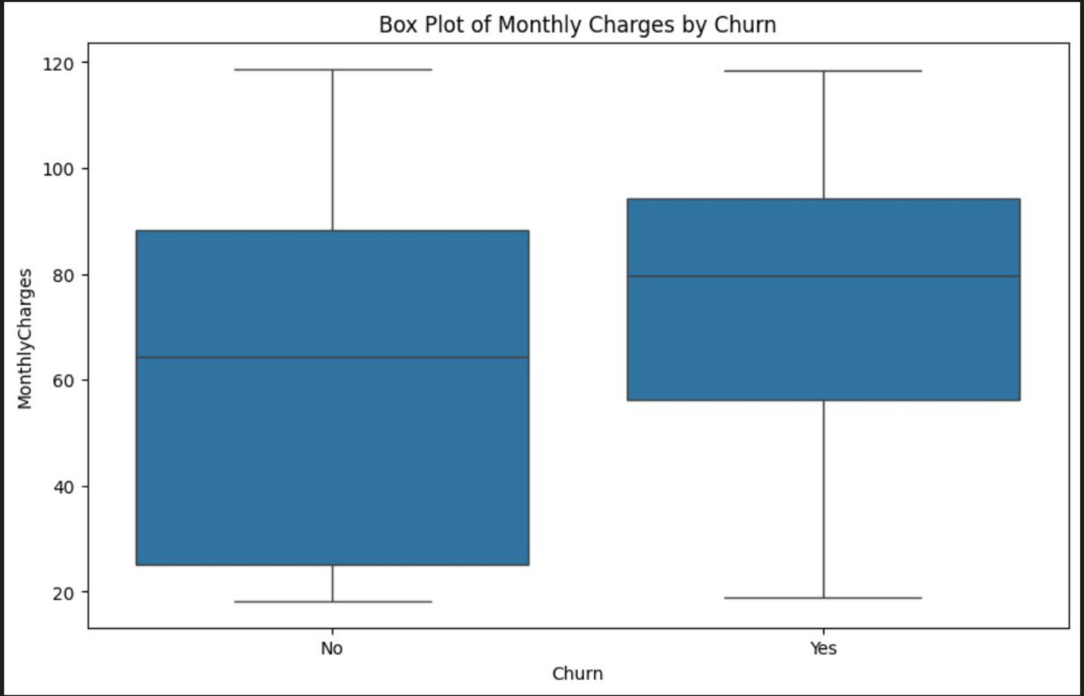


# Box Plot of Monthly Charges by Churn

```
#Box Plot of Monthly Charges by Churn
plt.figure(figsize=(10, 6))
sns.boxplot(x='Churn', y='MonthlyCharges', data=df)
plt.title('Box Plot of Monthly Charges by Churn')
plt.show()
```

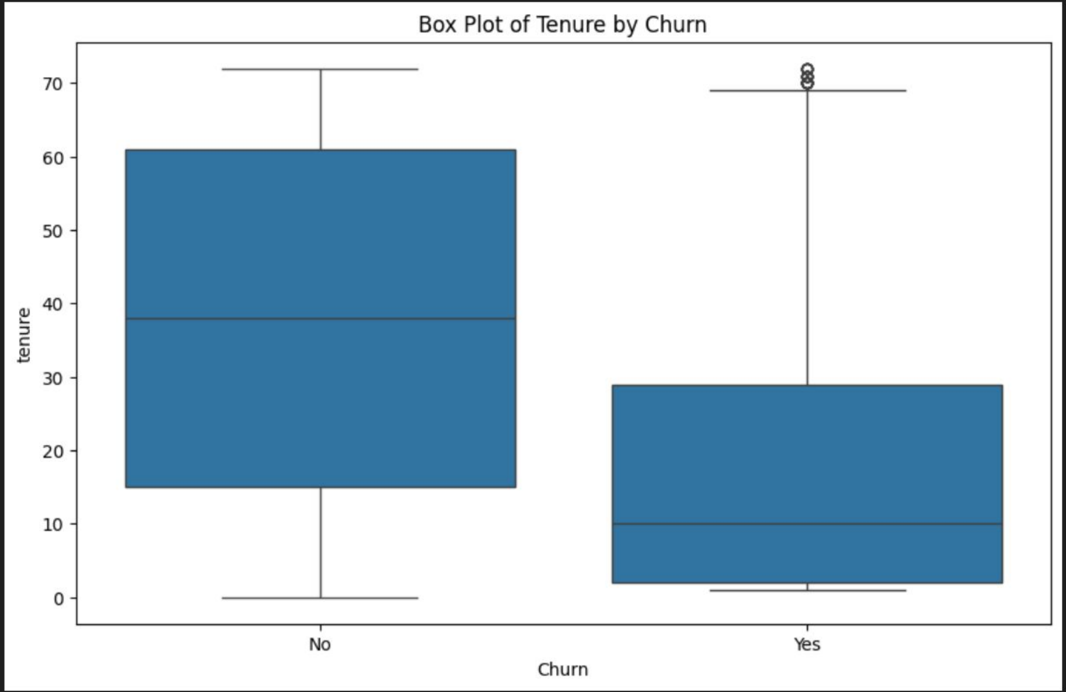
✓ 0.1s



# Box Plot of Tenure by Churn

```
#Box Plot of Tenure by Churn
plt.figure(figsize=(10, 6))
sns.boxplot(x='Churn', y='tenure', data=df)
plt.title('Box Plot of Tenure by Churn')
plt.show()
```

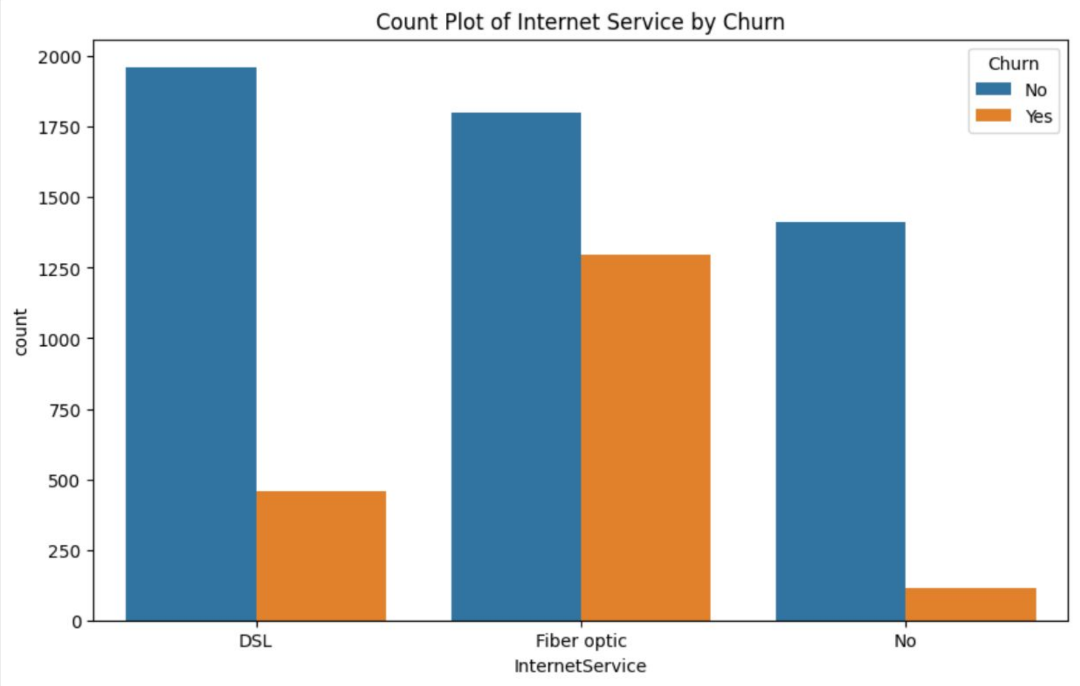
✓ 0.1s



# Count Plot of Internet Service by Churn

```python
#Count Plot of Internet Service by Churn
plt.figure(figsize=(10, 6))
sns.countplot(x='InternetService', hue='Churn', data=df)
plt.title('Count Plot of Internet Service by Churn')
plt.show()
```

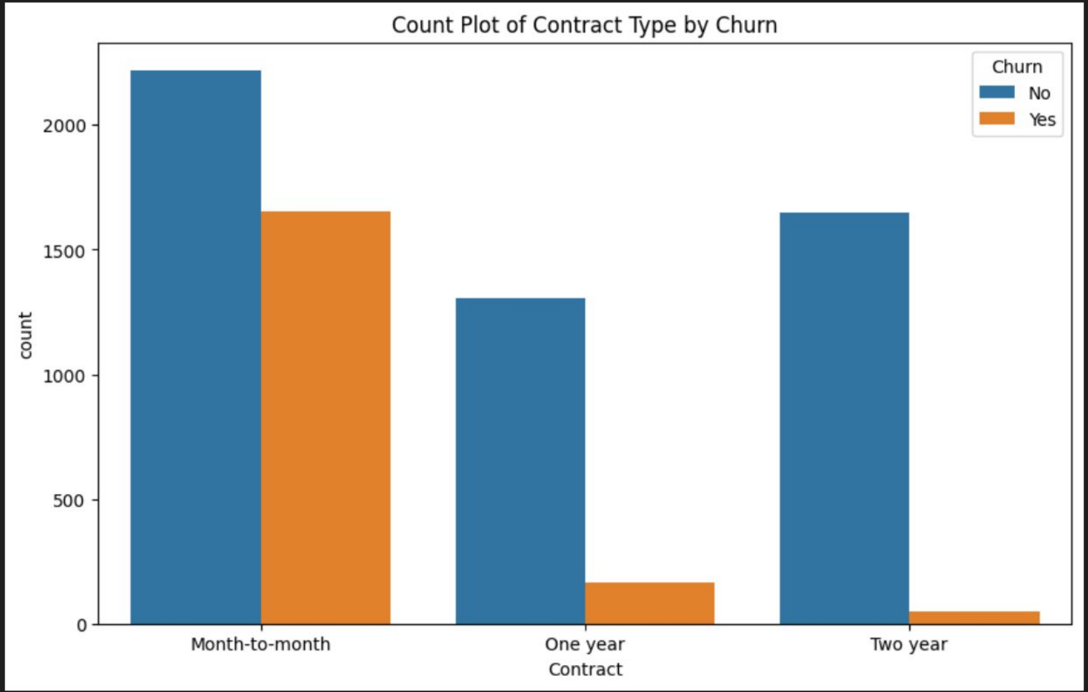
✓ 0.1s



# Count Plot of Contract Type by Churn

```
#Count Plot of Contract Type by Churn
plt.figure(figsize=(10, 6))
sns.countplot(x='Contract', hue='Churn', data=df)
plt.title('Count Plot of Contract Type by Churn')
plt.show()
```

✓ 0.1s



# Analysis

From these visual depictions of the data we can see that

- People with higher monthly charges are more likely to churn.
- People with higher tenures are less likely to churn.
- People with fiber optic internet service are more likely to churn than people with DSL
- People with month to month contracts are more likely to churn than people with one year or two year contracts

# Making the Model

By implementing sklearn, we were able to develop, evaluate, and interpret machine learning models, enabling us to derive actionable insights and make informed decisions to address the problem of customer churn prediction.

# Model Building and Visual Analysis of the Training Model

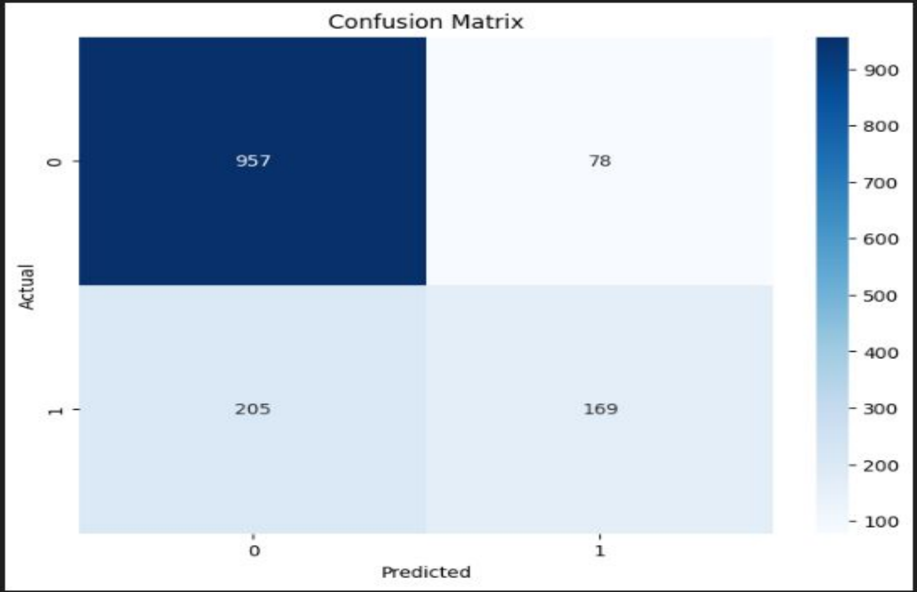
```python
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix, roc_auc_score


#Encoding  variables
df_encoded = pd.get_dummies(df, drop_first=True)

#Split the data
X = df_encoded.drop('Churn_Yes', axis=1)  # Exclude the target variable
y = df_encoded['Churn_Yes']  # Target variable

#Normalize/Standardize the features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

#Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42, stratify=y)


#Initialize models
logistic_model = LogisticRegression(max_iter=1000)
random_forest_model = RandomForestClassifier(n_estimators=100, random_state=42)

#Train models
logistic_model.fit(X_train, y_train)
random_forest_model.fit(X_train, y_train)

#Predict using the models
y_pred_logistic = logistic_model.predict(X_test)
y_pred_rf = random_forest_model.predict(X_test)

for name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    print(f"{name} Model")
    print("Confusion Matrix:")
    print(confusion_matrix(y_test, y_pred))
    print("ROC AUC Score:")
    print(roc_auc_score(y_test, y_pred))
    print("Classification Report:")
    print(classification_report(y_test, y_pred))
    print("\n")


    print("Random Forest Model")
    print("Confusion Matrix:")
    print(confusion_matrix(y_test, y_pred_rf))
    print("ROC AUC Score:")
    print(roc_auc_score(y_test, y_pred_rf))
    print("Classification Report:")
    print(classification_report(y_test, y_pred_rf))

#Calculate feature importances
importances = random_forest_model.feature_importances_
importance_df = pd.DataFrame({
    'Feature': X.columns,
    'Importance': importances
}).sort_values(by='Importance', ascending=False)
```

```
Logistic Regression Model
Confusion Matrix:
[[591 444]
 [ 52 322]]
ROC AUC Score:
0.7159885297992714
Classification Report:
              precision    recall  f1-score   support

       False       0.92      0.57      0.70      1035
        True       0.42      0.86      0.56       374

    accuracy                           0.65      1409
   macro avg       0.67      0.72      0.63      1409
weighted avg       0.79      0.65      0.67      1409



Random Forest Model
Confusion Matrix:
[[955  80]
 [201 173]]
ROC AUC Score:
0.692636079464724
Classification Report:
...
    accuracy                           0.80      1409
   macro avg       0.75      0.69      0.71      1409
weighted avg       0.79      0.80      0.79      1409
```

# Confusion Matrix

```
import seaborn as sns
import matplotlib.pyplot as plt

#Plot confusion matrix
plt.figure(figsize=(8, 6))
sns.heatmap(confusion_matrix(y_test, y_pred), annot=True, fmt='d', cmap='Blues')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```

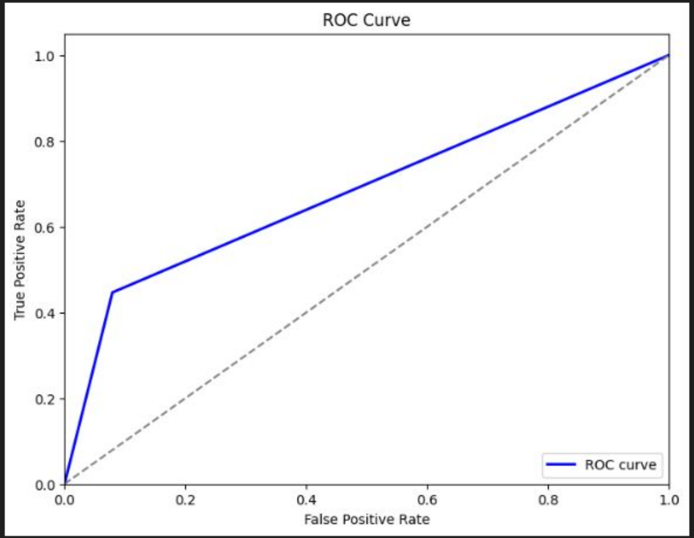
✓ 0.1s



# ROC Curve

```python
from sklearn.metrics import roc_curve

# Compute ROC curve
fpr, tpr, thresholds = roc_curve(y_test, y_pred)

# Plot ROC curve
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, color='blue', lw=2, label='ROC curve')
plt.plot([0, 1], [0, 1], color='gray', linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve')
plt.legend(loc='lower right')
plt.show()
```

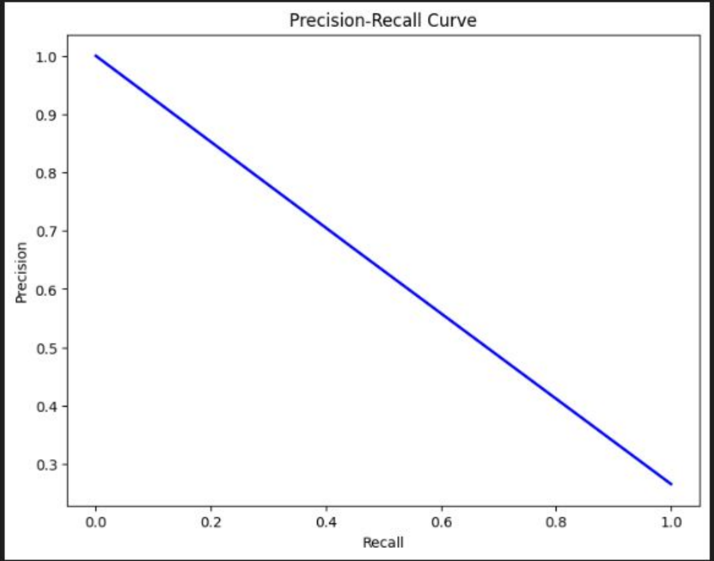
✓ 0.1s



# Precision-Recall Curve

```python
from sklearn.metrics import precision_recall_curve

# Compute precision-recall curve
precision, recall, _ = precision_recall_curve(y_test, y_pred)

# Plot precision-recall curve
plt.figure(figsize=(8, 6))
plt.plot(recall, precision, color='blue', lw=2)
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.title('Precision-Recall Curve')
plt.show()
```

✓ 0.1s



# Model Analysis

- **Evaluation metrics:**
  - Accuracy: Overall performance of the model.
  - ROC AUC Score: Measure of the model's ability to distinguish between classes.
  - Confusion Matrix: Breakdown of true positives, true negatives, false positives, and false negatives.
  - Classification Report: Detailed metrics including precision, recall, and F1-score.
- **Logistic Regression Results:**
  - **Accuracy: 67%**
  - **ROC AUC Score: 0.71**
- **Random Forest Results:**
  - **Accuracy: 75%**
  - **ROC AUC Score: 0.69**

# Conclusions:

- Both logistic regression and random forest models provide valuable insights into predicting customer churn.
- Random forest outperformed logistic regression in terms of overall accuracy and predictive power.
- Leveraging these models, the company can proactively identify customers at risk of churning and implement targeted retention strategies to mitigate churn rates.
- Continuous monitoring and refinement of the models based on evolving customer behavior and market dynamics are crucial for sustaining long-term customer loyalty and profitability.