PRESENTATION

Presented by: Kislay Kumar

INTRODUCTION

- Customer churn explained: why customers leave matters
- Importance of predicting churn for business growth
- Data has been taken from kaggle - Telco Customer Churn on Kaggle.

BACKGROUND

- Problem: Predict if a customer will leave a service (churn) based on usage data
- Collect & clean customer datasets
- Analyze features like contract type, tenure, payment method with charts
- Build classification models (Decision Trees, Random Forest, or XGBoost)
- Validate model with accuracy, confusion matrix
- Generate insights to help reduce churn
- Document clearly with code and findings

PROBLEMS

Predict if a customer will leave a service (churn) based on usage data

High churn leads to revenue loss

GOALS

Predict which customers may churn using data

- Collect & clean customer datasets

Analyze features like contract type, tenure, payment method with charts

THEORY

- What is customer churn prediction?
- Common algorithms: Random Forest, Decision Tree, XGBoost
- Key metrics: accuracy, precision, recall, confusion matrix

Customer churn prediction means using data and machine learning to guess which customers might stop using a service soon. It helps businesses keep customers by acting before they leave.

ANALYSIS

- Dataset overview (Telco Customer Churn)
- Data cleaning steps: handling missing values, encoding categories
- Visualizations like churn distribution, tenure histogram, correlation heatmap

Visualize data trends like churn rate by contract type, tenure distribution, payment methods.

RESULT

I Have attached the screenshot of the code and result on next page

I Have collected the data from kaggle –

You can find the dataset here: Telco Customer Churn on Kaggle.

1. Problem Definition:

Predict if a customer will churn (leave) based on their data.

```
[ ] import pandas as pd
    import matplotlib.pyplot as plt
    import seaborn as sns
    from sklearn.model_selection import train_test_split, GridSearchCV
    from sklearn.ensemble import RandomForestClassifier
    from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
```

2. Data Collection & Cleaning:

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
data = pd.read_csv('/content/drive/MyDrive/WA_Fn-UseC_-Telco-Customer-Churn.csv')
```

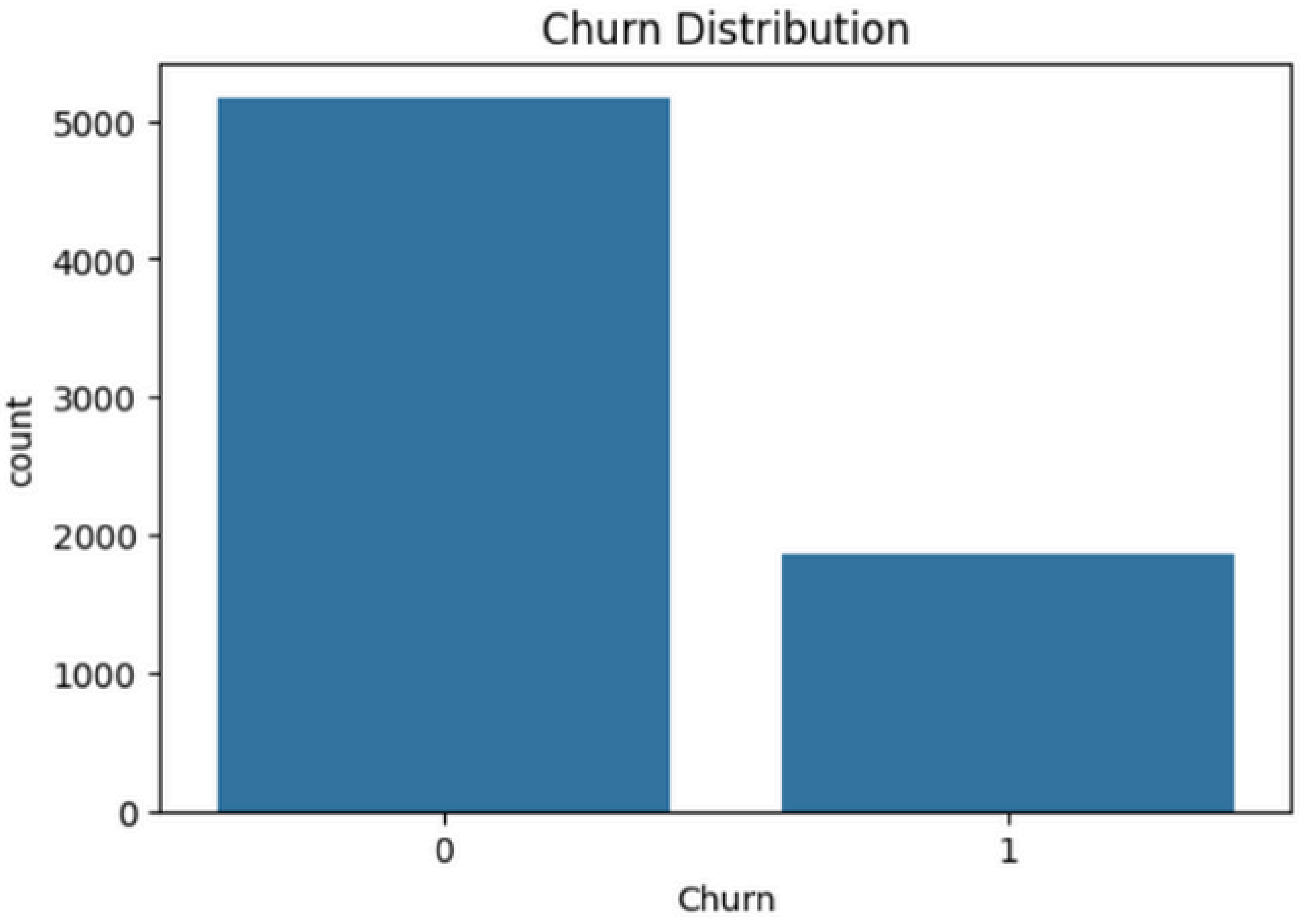
```
data = data.drop('customerID', axis=1)
data['TotalCharges'] = pd.to_numeric(data['TotalCharges'], errors='coerce')
data = data.dropna()
```

Encode categorical variables

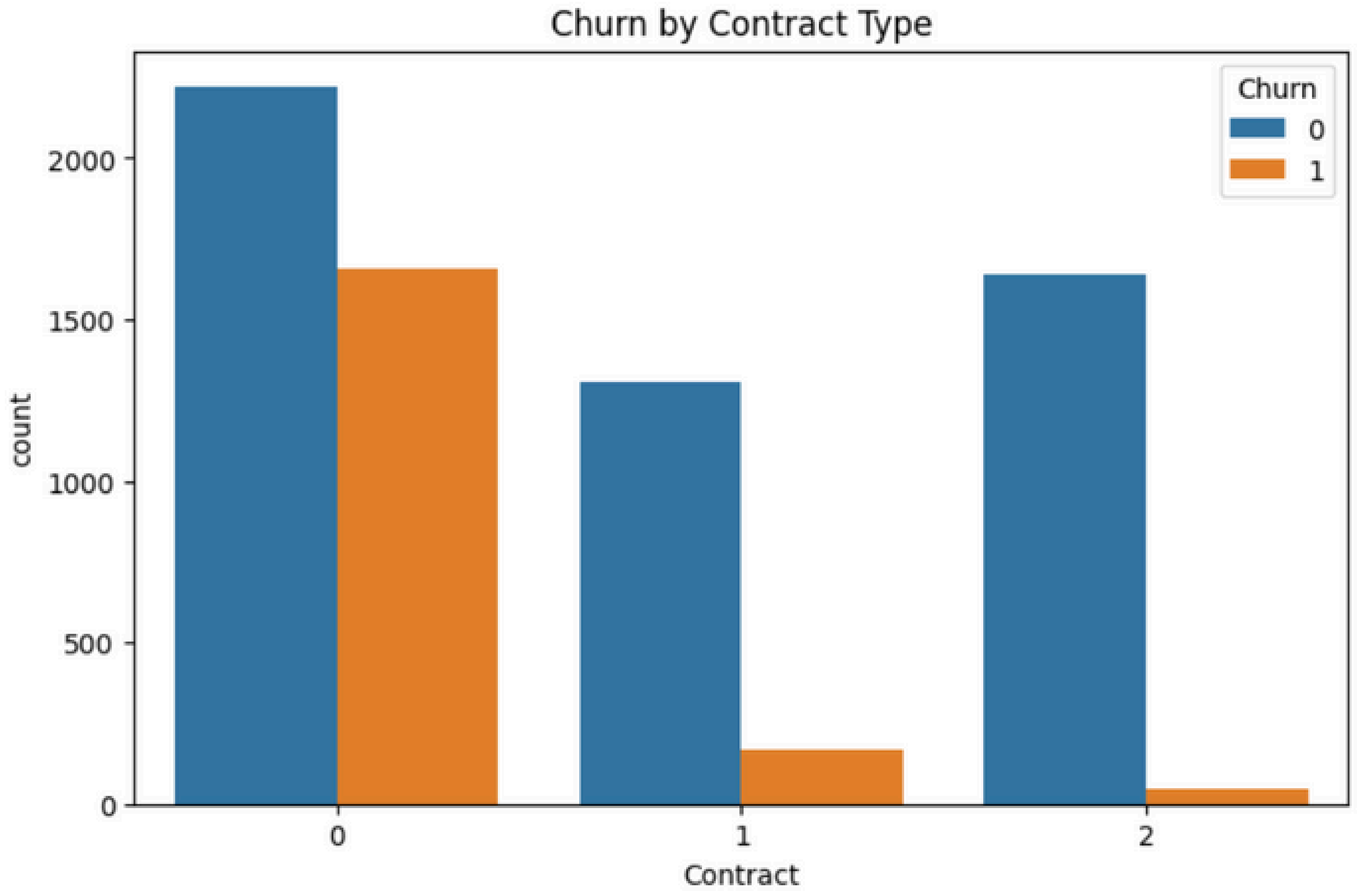
```
for col in data.select_dtypes(include='object').columns:
    data[col] = data[col].astype('category').cat.codes
```

3. Data Analysis & Charts:

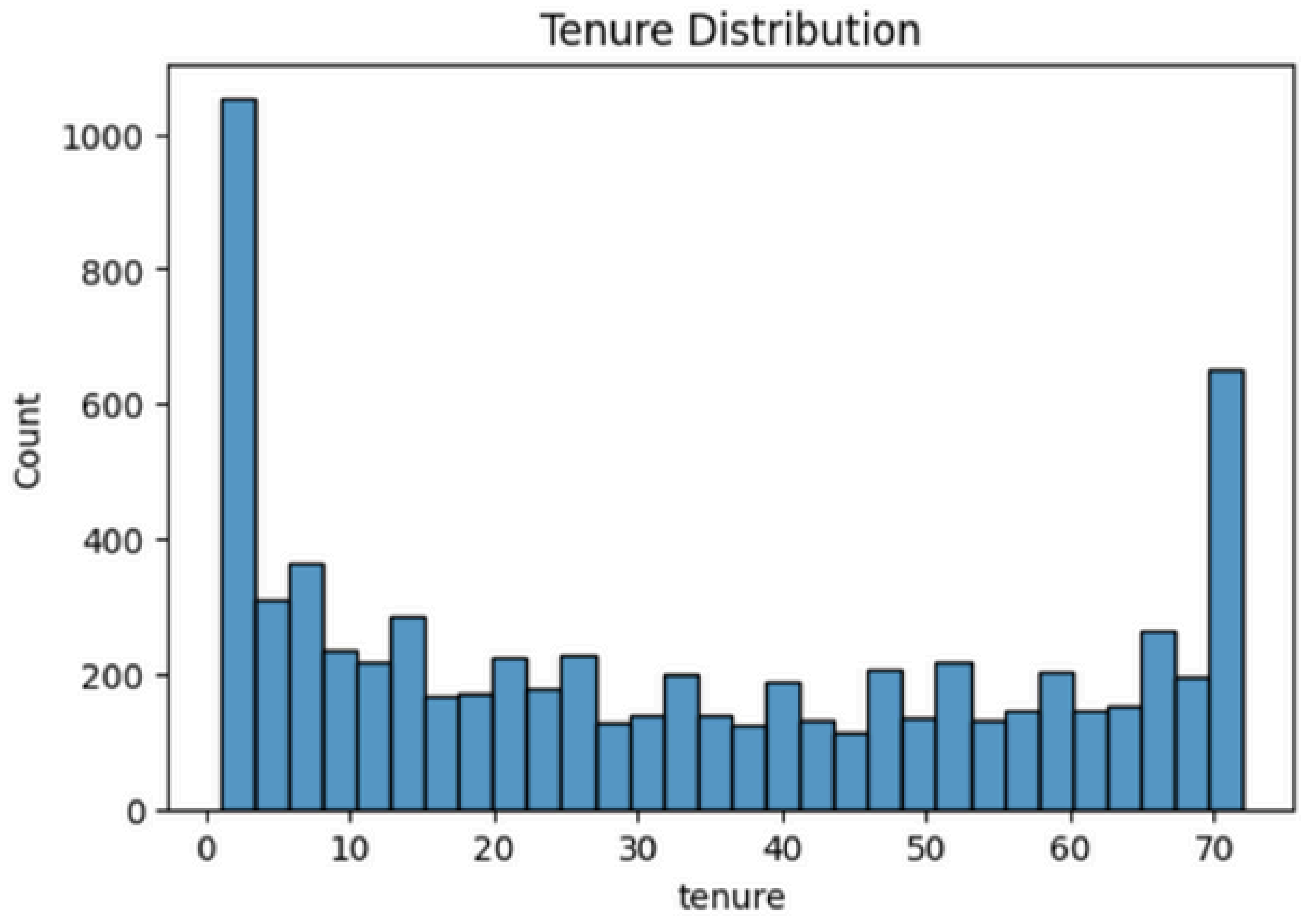
```
plt.figure(figsize=(6,4))
sns.countplot(x='Churn', data=data)
plt.title('Churn Distribution')
plt.show()
```



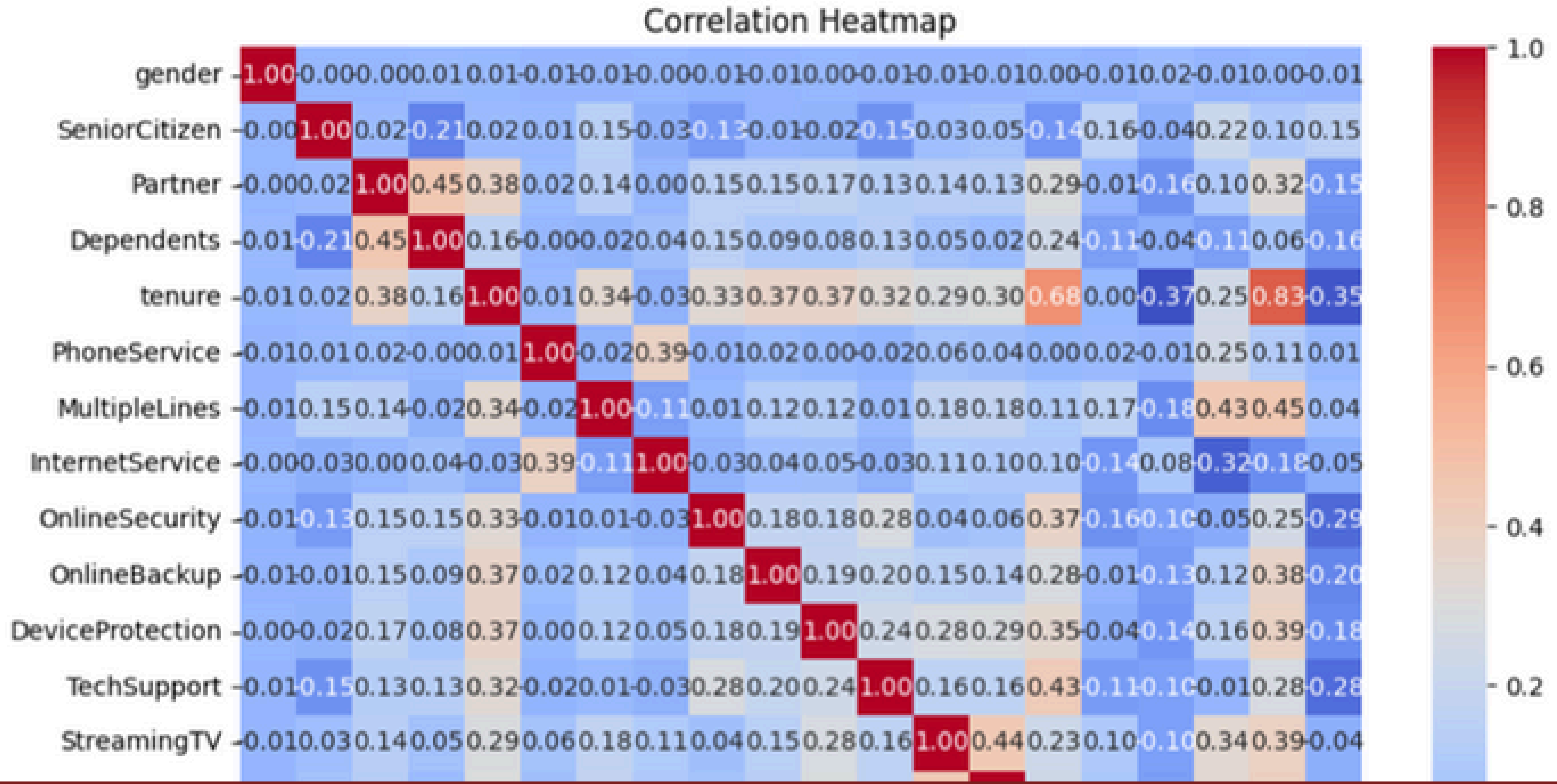
```
plt.figure(figsize=(8,5))
sns.countplot(x='Contract', hue='Churn', data=data)
plt.title('Churn by Contract Type')
plt.show()
```



```
plt.figure(figsize=(6,4))
sns.histplot(data['tenure'], bins=30)
plt.title('Tenure Distribution')
plt.show()
```



```python
plt.figure(figsize=(10,8))
sns.heatmap(data.corr(), annot=True, fmt=".2f", cmap="coolwarm")
plt.title('Correlation Heatmap')
plt.show()
```



4. Model Building:

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

```
[ ] X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

```
[ ] rf = RandomForestClassifier(random_state=42)
    rf.fit(X_train, y_train)
```

```
RandomForestClassifier
RandomForestClassifier(random_state=42)
```

5. Model Checking:

```
[ ] y_pred = rf.predict(X_test)
    print("Accuracy:", accuracy_score(y_test, y_pred))
    print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred))
    print("\nClassification Report:\n", classification_report(y_test, y_pred))
```

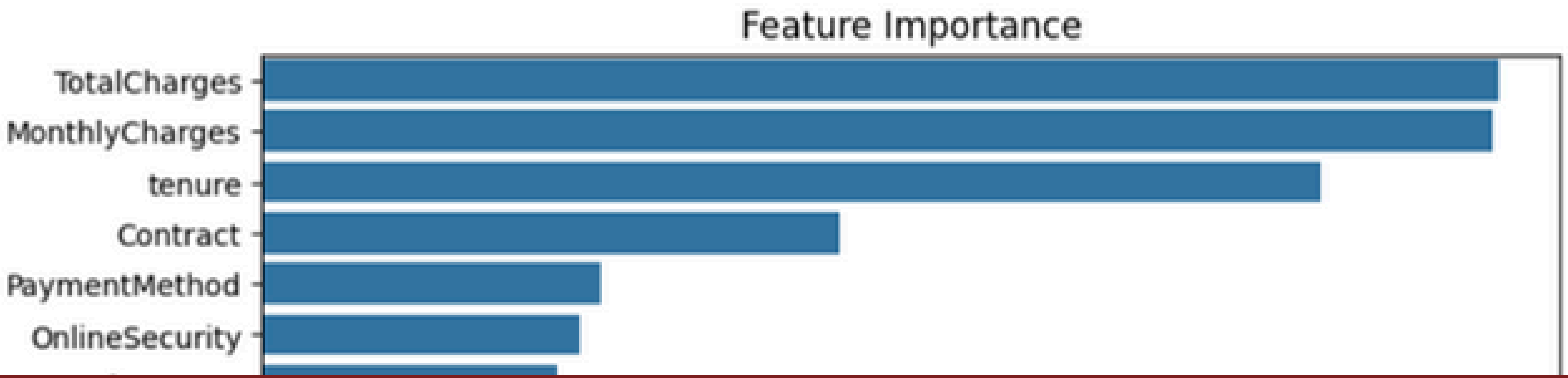
```
Accuracy: 0.7848341232227488

Confusion Matrix:
 [[1384  165]
 [ 289  272]]

Classification Report:
              precision    recall  f1-score   support

           0       0.83      0.89      0.86      1549
           1       0.62      0.48      0.55       561

    accuracy                           0.78      2110
   macro avg       0.72      0.69      0.70      2110
weighted avg       0.77      0.78      0.78      2110
```

6. Results & Insights:

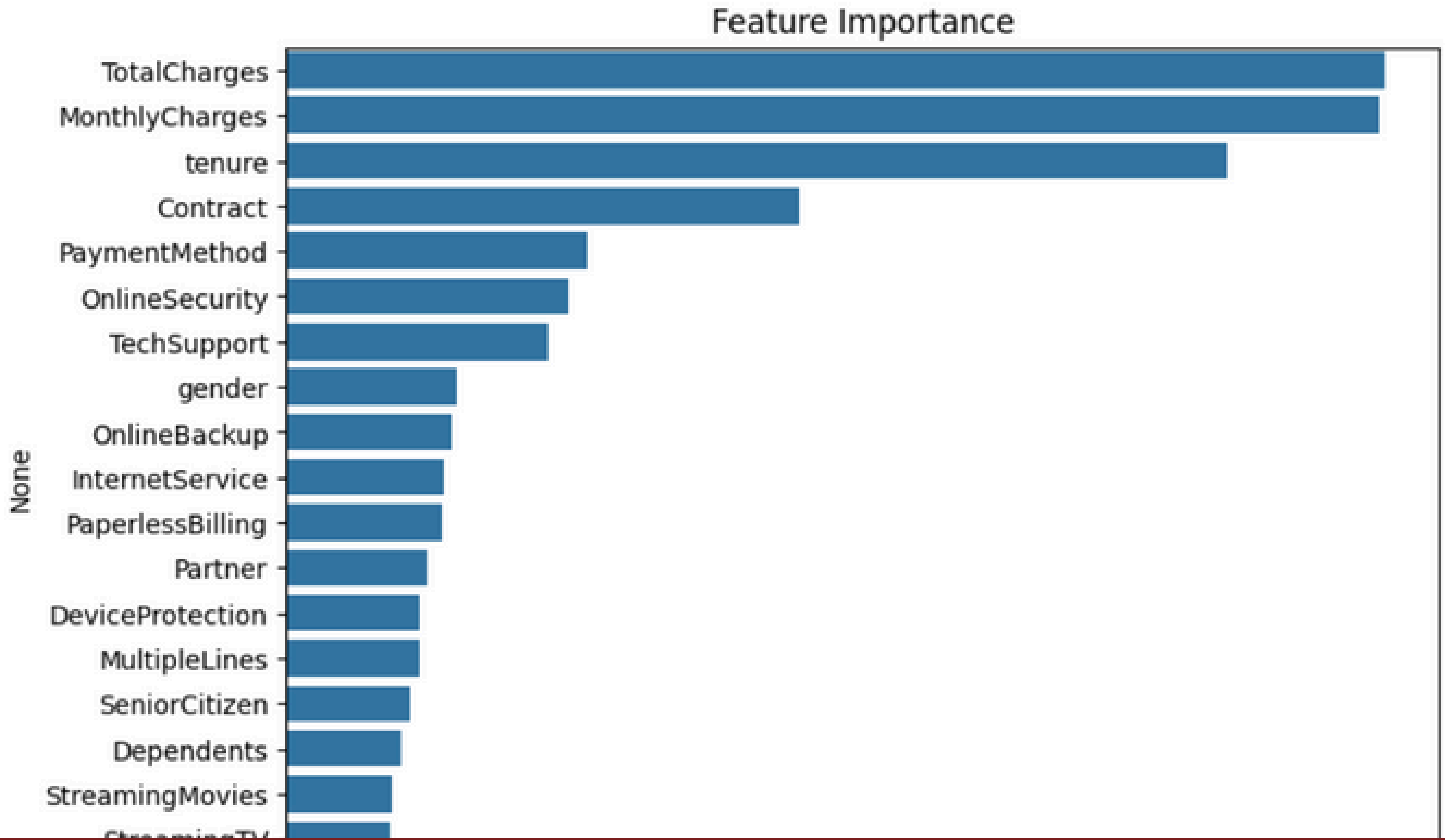
Feature importance to identify key churn drivers

```
importances = rf.feature_importances_
feat_names = X.columns
feat_imp = pd.Series(importances, index=feat_names).sort_values(ascending=False)
```

```
plt.figure(figsize=(8,6))
sns.barplot(x=feat_imp, y=feat_imp.index)
plt.title('Feature Importance')
plt.show()
```



```python
plt.figure(figsize=(8,6))
sns.barplot(x=feat_imp, y=feat_imp.index)
plt.title('Feature Importance')
plt.show()
```



7. Report & Documentation:

Clearly document each step with comments and save visualizations if needed.

Optional: Hyperparameter tuning for better model

```
[ ] param_grid = {
        'n_estimators': [100, 200],
        'max_depth': [None, 10, 20],
        'min_samples_split': [2, 5]
    }
```

```
[ ] grid_search = GridSearchCV(rf, param_grid, cv=3, scoring='accuracy')
    grid_search.fit(X_train, y_train)
    print("Best params:", grid_search.best_params_)
```

```
Best params: {'max_depth': 10, 'min_samples_split': 2, 'n_estimators': 100}
```

```
[ ] best_rf = grid_search.best_estimator_
    y_pred_best = best_rf.predict(X_test)
    print("Tuned Model Accuracy:", accuracy_score(y_test, y_pred_best))
```

```
Tuned Model Accuracy: 0.7900473933649289
```

THANK YOU