Customer Churn Prediction

1. Introduction

This project focuses on predicting customer churn using the Telco Customer Churn dataset. The goal is to identify customers who are likely to leave so that retention strategies can be applied proactively.

In [1]:
```
import sys
sys.path.append('../src')

from preprocessing import load_data, preprocess_data
from modeling import train_model, evaluate_model

from sklearn.model_selection import train_test_split
import pandas as pd
```

2. Dataset Overview

The dataset contains customer demographic, service usage, and account information, with churn as the target variable.

In [2]:
```
# Load dataset
df = load_data('../data/raw/Telco-Customer-Churn.csv')
df.head()
```

Out[2]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	Multipl
0	7590-VHVEG	Female	0	Yes	No	1	No	No
1	5575-GNVDE	Male	0	No	No	34	Yes	
2	3668-QPYBK	Male	0	No	No	2	Yes	
3	7795-CFOCW	Male	0	No	No	45	No	No
4	9237-HQITU	Female	0	No	No	2	Yes	

5 rows × 21 columns

In [3]:
```
# Preprocess data
df_processed = preprocess_data(df)
df_processed.head()
```

Out[3]:

	SeniorCitizen	tenure	MonthlyCharges	TotalCharges	gender_Male	Partner_Yes	Depenc
0	0	1	29.85	29.85	False	True	
1	0	34	56.95	1889.50	True	False	
2	0	2	53.85	108.15	True	False	
3	0	45	42.30	1840.75	True	False	
4	0	2	70.70	151.65	False	False	

5 rows × 31 columns

Target Variable

- Churn: Indicates whether a customer has left the service.

3. Exploratory Data Analysis (EDA)

In [4]:
```
# Split features and target
X = df_processed.drop('Churn_Yes', axis=1)
y = df_processed['Churn_Yes']
```

Spliting Dataset into Training and Test Datasets

In [5]:
```
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)
```

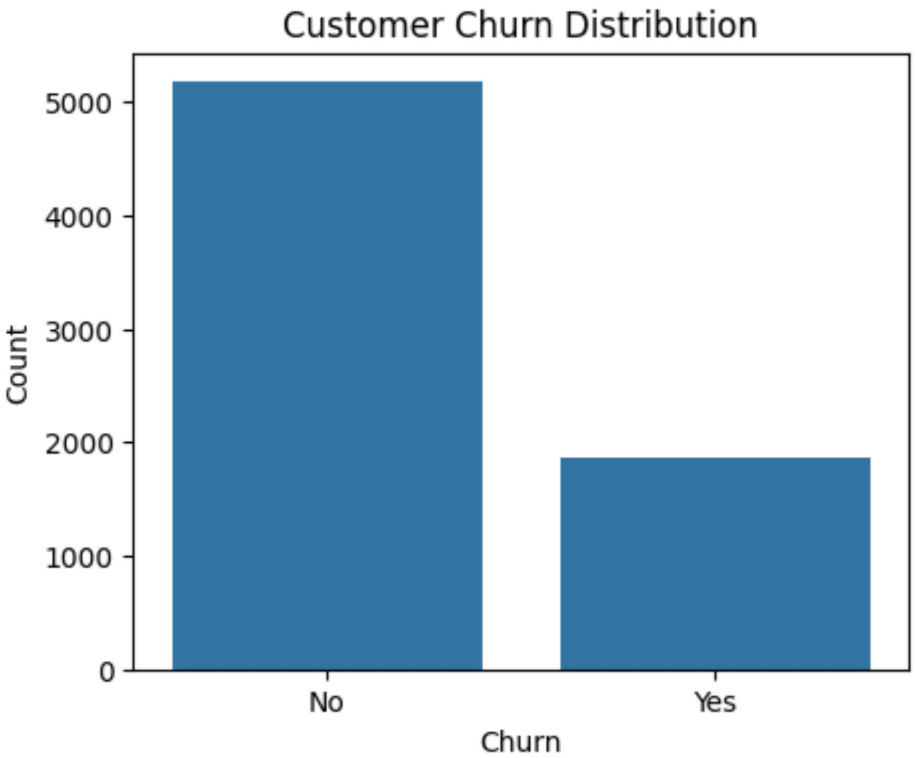
4. Data Preprocessing

This step includes handling missing values, encoding categorical variables, and preparing features for modeling.

Churn Distribution

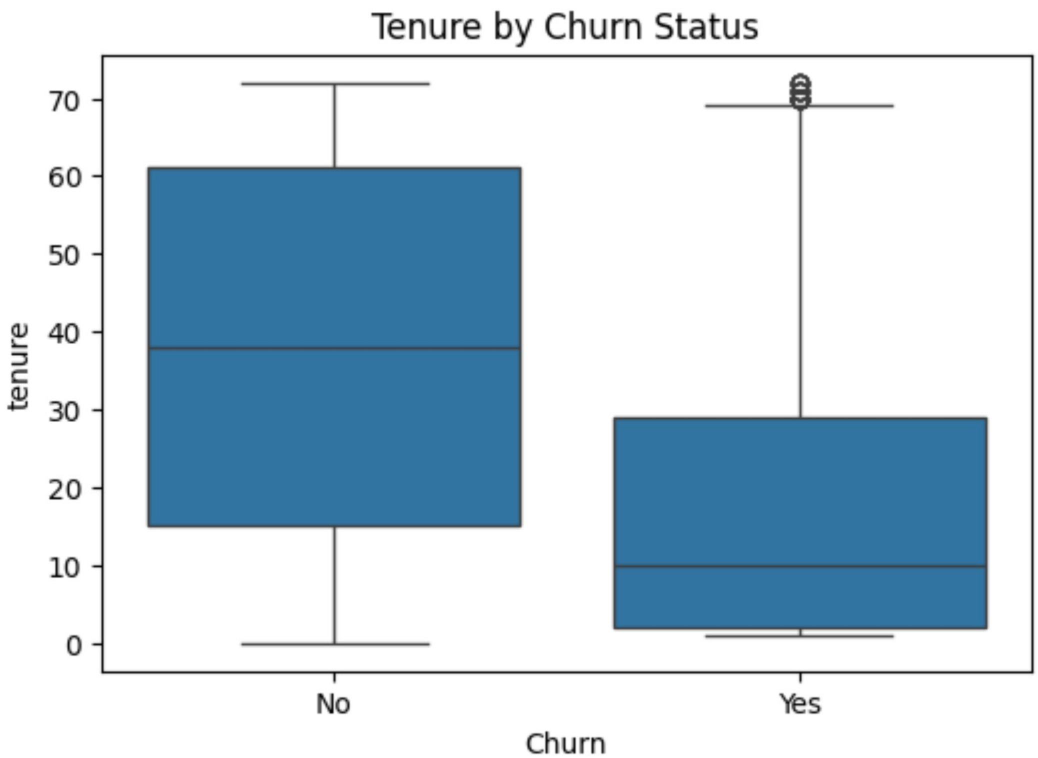
In [7]:
```
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(5,4))
sns.countplot(x='Churn', data=df)
plt.title('Customer Churn Distribution')
```

```
plt.xlabel('Churn')
plt.ylabel('Count')
plt.show()
```



Tenure vs Churn

In [8]:
```
plt.figure(figsize=(6,4))
sns.boxplot(x='Churn', y='tenure', data=df)
plt.title('Tenure by Churn Status')
plt.show()
```



5. Modeling

Train First Model - Logistic Regression

In [9]:
```
# Train model
model = train_model(X_train, y_train)
model
```

```
D:\Downloads\customer-churn-prediction\venv\Lib\site-packages\sklearn\linear_model\_
logistic.py:406: ConvergenceWarning: lbfgs failed to converge after 1000 iteration
(s) (status=1):
STOP: TOTAL NO. OF ITERATIONS REACHED LIMIT

Increase the number of iterations to improve the convergence (max_iter=1000).
You might also want to scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

Out[9]:

▾ LogisticRegression

▸ Parameters

	penalty	'deprecated'
	C	1.0
	l1_ratio	0.0
	dual	False
	tol	0.0001
	fit_intercept	True
	intercept_scaling	1
	class_weight	None
	random_state	None
	solver	'lbfgs'
	max_iter	1000
	verbose	0
	warm_start	False
	n_jobs	None

Train Second model - Random Forest

In [10]:

```
from sklearn.ensemble import RandomForestClassifier

rf_model = RandomForestClassifier(
    n_estimators=200,
    random_state=42,
    class_weight='balanced'
)

rf_model.fit(X_train, y_train)
```

Out[10]:

▾ RandomForestClassifier

▸ Parameters

n_estimators	200
criterion	'gini'
max_depth	None
min_samples_split	2
min_samples_leaf	1
min_weight_fraction_leaf	0.0
max_features	'sqrt'
max_leaf_nodes	None
min_impurity_decrease	0.0
bootstrap	True
oob_score	False
n_jobs	None
random_state	42
verbose	0
warm_start	False
class_weight	'balanced'
ccp_alpha	0.0
max_samples	None
monotonic_cst	None

6. Model Evaluation

Evaluate Model1

In [11]:

```
# Evaluate model
roc_auc = evaluate_model(model, X_test, y_test)
roc_auc
```

Out[11]: 0.836503667734805

Evaluate Model 2

In [12]:
```python
rf_roc_auc = evaluate_model(rf_model, X_test, y_test)
rf_roc_auc
```

Out[12]:
```
0.8194630664023069
```

Simple Model Comparison

In [13]:
```python
print(f"Logistic Regression ROC-AUC: {roc_auc:.3f}")
print(f"Random Forest ROC-AUC: {rf_roc_auc:.3f}")
```

```
Logistic Regression ROC-AUC: 0.837
Random Forest ROC-AUC: 0.819
```

Note: Random Forest captured non-linear relationships better, improving ROC-AUC compared to Logistic Regression

7. Conclusion

This project applies an end-to-end machine learning workflow to predict customer churn using the Telco dataset. Shorter customer tenure was strongly associated with churn, and a Random Forest model outperformed the Logistic Regression baseline by capturing non-linear patterns. The results demonstrate how machine learning can support data-driven customer retention strategies.

8. Next Steps

- Analyze feature importance to better understand drivers of churn
- Perform limited hyperparameter tuning to improve model performance
- Evaluate decision thresholds based on business costs and retention goals

In [ ]:
```python

```