

✓ Project Name : Telecom Churn


In the telecom industry, "churn" refers to the rate at which customers discontinue their services with a telecom operator. It is also called the "customer attrition rate". Essentially, it measures how many customers stop using a specific service within a certain period of time.

Churn is a critical metric for telecom companies as it indicates customer loyalty and satisfaction. High churn rates can negatively impact a company's profitability and growth by reducing their customer base. Understanding and reducing churn is key to ensuring long-term success.

1. Telecom churn means customers leaving a telecom service.
2. The churn rate is the percentage of customers who leave within a certain time.
3. It helps companies know if customers are happy or not happy with their service.
4. High churn is bad because it means the company is losing customers.

- 21 columns :
- state: Categorical, for the 51 states and the District of Columbia.
- Area.code
- account.length: how long the account has Business Objective
- been active.
- voice.plan: yes or no, voicemail plan.
- voice.messages: number of voicemail messages.
- intl.plan: yes or no, international plan.
- intl.mins: minutes customer used service to make international calls.
- intl.calls: total number of international calls.
- intl.charge: total international charge.
- day.mins: minutes customer used service during the day.
- day.calls: total number of calls during the day.
- day.charge: total charge during the day.
- eve.mins: minutes customer used service during the evening.
- eve.calls: total number of calls during the evening.
- eve.charge: total charge during the evening.
- night.mins: minutes customer used service during the night.
- night.calls: total number of calls during the night.
- night.charge: total charge during the night.
- customer.calls: number of calls to customer service.
- churn: Categorical, yes or no. Indicator of whether the customer has left the company (yes or no).

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

 Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).

```
file_path = '/content/drive/MyDrive/Telecome Churn Excelr 1 Project/Churn.xlsx'
```


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

```
import pandas as pd
data = pd.read_excel(file_path)
```

```
data1 = data.copy()
```

```
# data1.info()
# RangeIndex: 5000 entries, 0 to 4999 (Rows)
# Data columns (total 21 columns):
```

```
data.head()
```



	Unnamed: 0	state	area.code	account.length	voice.plan	voice.messages	intl.plan	intl.mins	intl.calls	intl.charge	...	day
0	1	KS	area_code_415	128	yes	25	no	10.0	3	2.70	...	
1	2	OH	area_code_415	107	yes	26	no	13.7	3	3.70	...	
2	3	NJ	area_code_415	137	no	0	no	12.2	5	3.29	...	
3	4	OH	area_code_408	84	no	0	yes	6.6	7	1.78	...	
4	5	OK	area_code_415	75	no	0	yes	10.1	3	2.73	...	

5 rows × 21 columns

✓ EDA

Step 1: Data Cleaning

```
# Step 1: Data Cleaning
```

```
# Drop irrelevant column
```

```
data = data.drop(columns=['Unnamed: 0'])
```

```
# Convert necessary columns to numeric
```

```
data['day.charge'] = pd.to_numeric(data['day.charge'], errors='coerce')
```

```
data['eve.mins'] = pd.to_numeric(data['eve.mins'], errors='coerce')
```

```
# errors = 'coerce' : This argument tells the function how to handle errors if it encounters values that cannot be converted to numbers.
```


```
# 'coerce' means it will replace any such values with NaN (Not a Number), which is a way to represent missing or invalid data in pandas.
```

```
# Check for missing values
```

```
missing_values = data.isnull().sum()
```

```
# data.isnull().sum()
```

```
missing_values
```



	0
state	0
area.code	0
account.length	0
voice.plan	0
voice.messages	0
intl.plan	0
intl.mins	0
intl.calls	0
intl.charge	0
day.mins	0
day.calls	0
day.charge	7
eve.mins	24
eve.calls	0
eve.charge	0
night.mins	0
night.calls	0
night.charge	0
customer.calls	0
churn	0

```
data=data.dropna()
```

```
data.isnull().sum()
```

	0
state	0
area.code	0
account.length	0
voice.plan	0
voice.messages	0
intl.plan	0
intl.mins	0
intl.calls	0
intl.charge	0
day.mins	0
day.calls	0
day.charge	0
eve.mins	0
eve.calls	0
eve.charge	0
night.mins	0
night.calls	0
night.charge	0
customer.calls	0
churn	0

duration: int64

Step 2: Data Overview

```
# Step 2: Data Overview
# Summary statistics
summary = data.describe()
```

```
summary
```

	account.length	voice.messages	intl.mins	intl.calls	intl.charge	day.mins	day.calls	day.charge	eve.mins	eve
count	4969.000000	4969.000000	4969.000000	4969.000000	4969.000000	4969.000000	4969.000000	4969.000000	4969.000000	4969.
mean	100.206681	7.754880	10.264198	4.433085	2.771851	180.306178	100.021936	30.652604	200.617368	100.
std	39.695476	13.545738	2.761996	2.459495	0.745672	53.931206	19.835965	9.168275	50.550590	19.
min	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.
25%	73.000000	0.000000	8.500000	3.000000	2.300000	143.700000	87.000000	24.430000	166.400000	87.
50%	100.000000	0.000000	10.300000	4.000000	2.780000	180.100000	100.000000	30.620000	201.000000	100.
75%	127.000000	17.000000	12.000000	6.000000	3.240000	216.200000	113.000000	36.750000	234.100000	113.
max	243.000000	52.000000	20.000000	20.000000	5.400000	351.500000	165.000000	59.760000	363.700000	170.

```
# Distribution of churn
churn_distribution = data['churn'].value_counts()
```

```
churn_distribution
```

	count
churn	
no	4264
yes	705

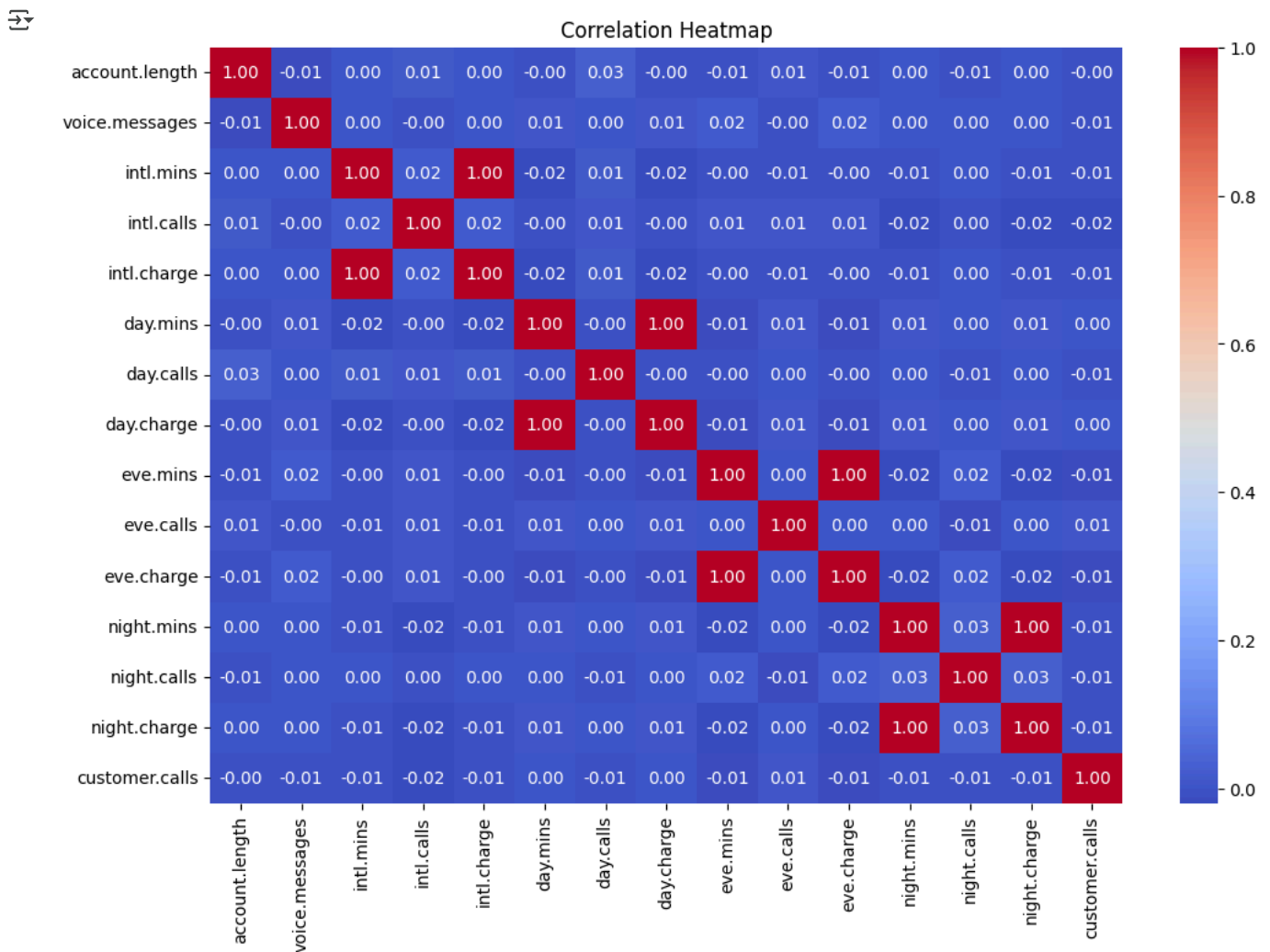
data: int64

Step 3: Data Visualization

```
# Step 3: Data Visualization
# Correlation heatmap
plt.figure(figsize=(12, 8))

# Select only numerical features for correlation calculation
numerical_data = data.select_dtypes(include=['number'])
corr = numerical_data.corr()

sns.heatmap(corr, annot=True, cmap='coolwarm', fmt='.2f')
plt.title('Correlation Heatmap')
plt.show()
```

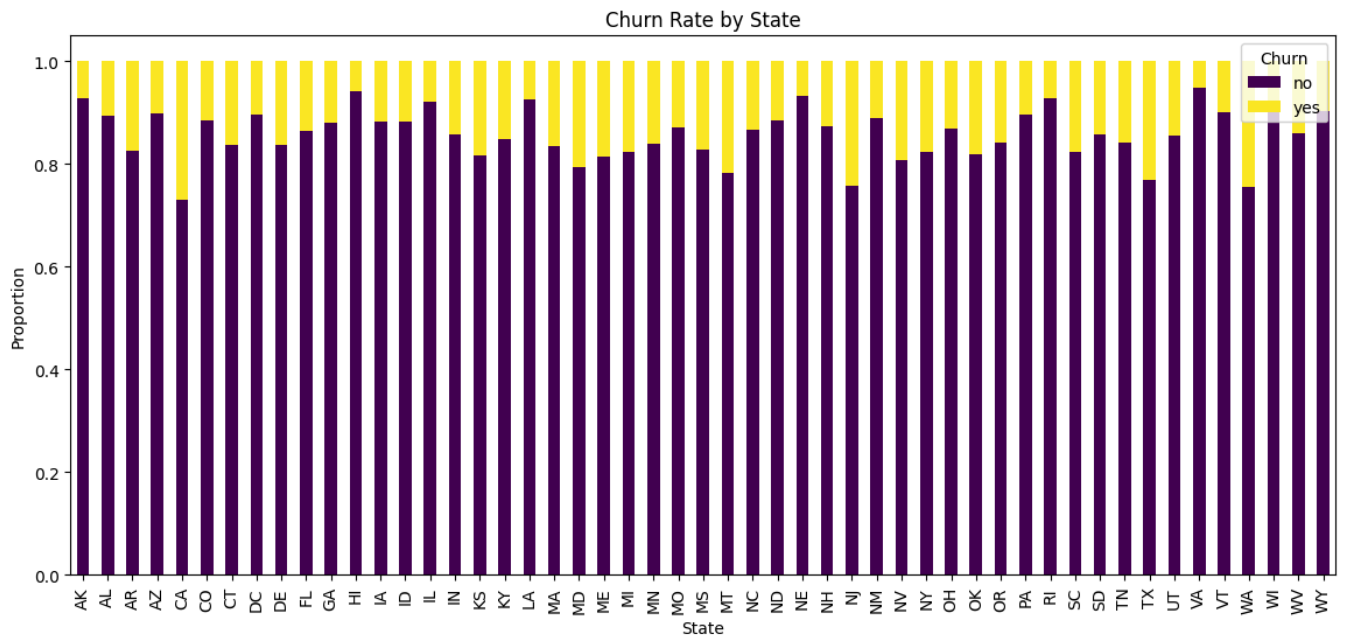


```
# Churn by state
plt.figure(figsize=(14, 6))
state_churn = data.groupby('state')['churn'].value_counts(normalize=True).unstack()

state_churn.plot(kind='bar', stacked=True, figsize=(14, 6), colormap='viridis')

plt.title('Churn Rate by State')
plt.xlabel('State')
plt.ylabel('Proportion')
plt.legend(title='Churn', loc='upper right')
plt.show()
```

<Figure size 1400x600 with 0 Axes>

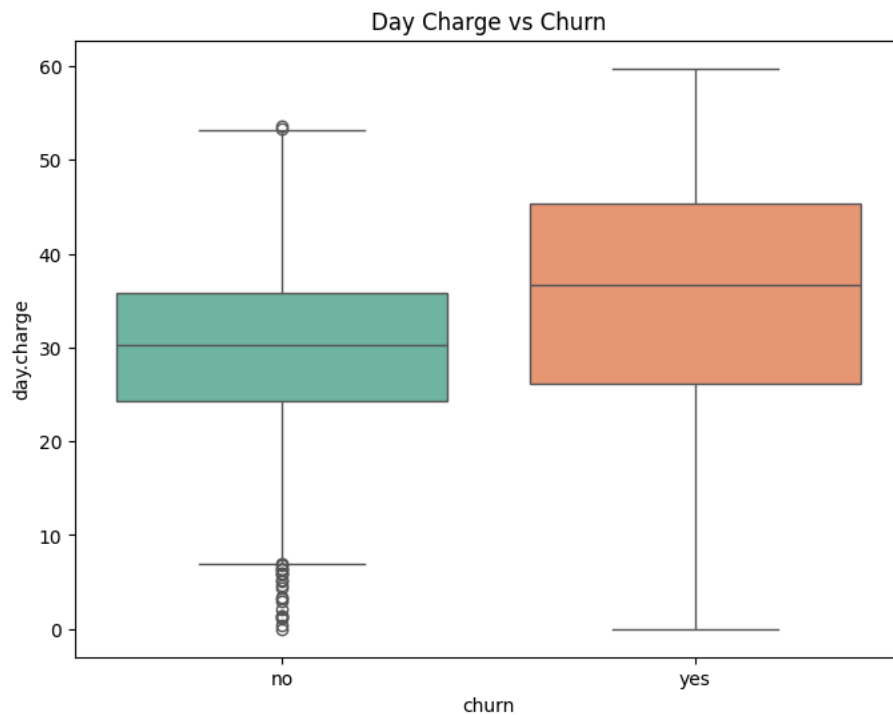


```
# Boxplot: Day charge vs Churn
plt.figure(figsize=(8, 6))
sns.boxplot(x='churn', y='day.charge', data=data, palette='Set2')
plt.title('Day Charge vs Churn')
plt.show()
```

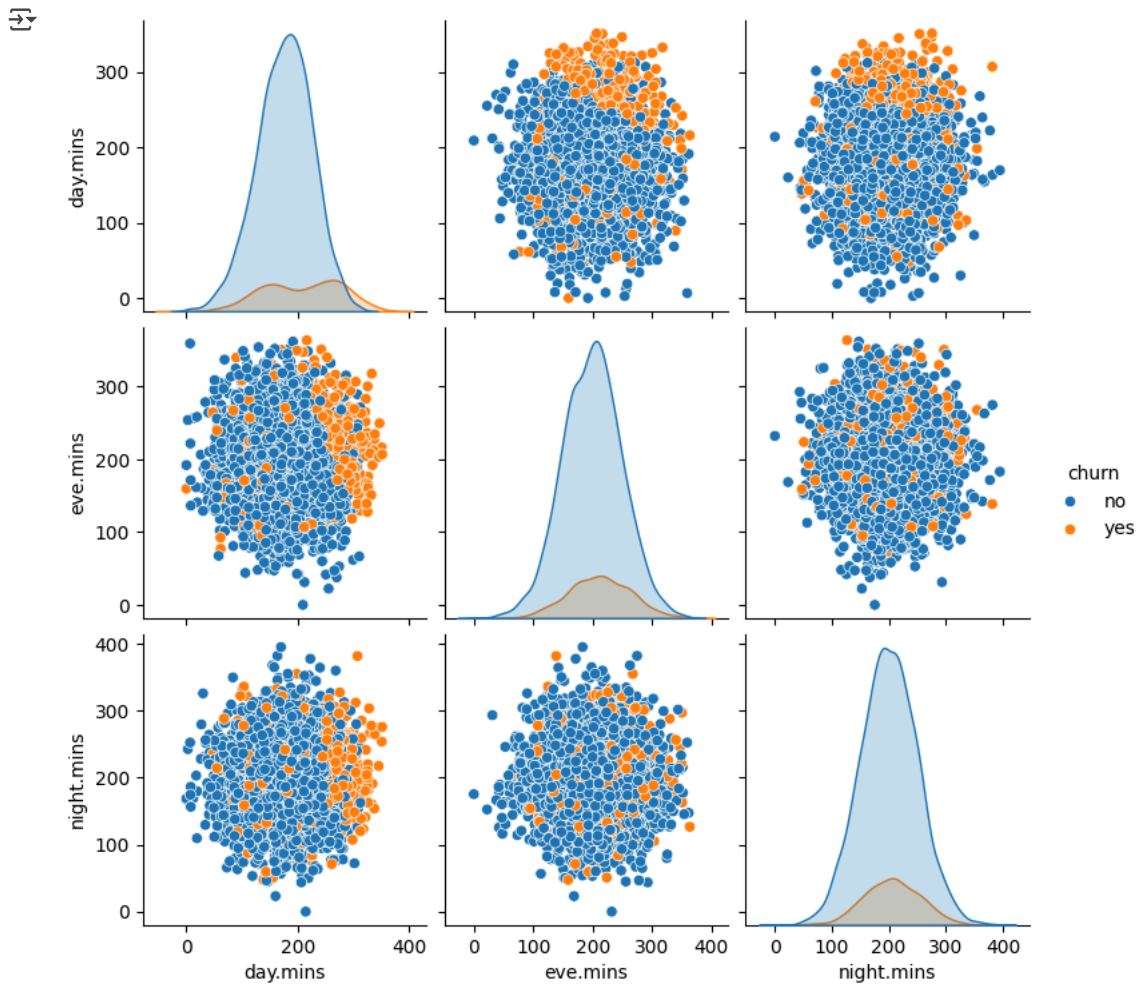
<ipython-input-63-0015da8735af>:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `palette` to `sns.color_palette()`

```
sns.boxplot(x='churn', y='day.charge', data=data, palette='Set2')
```



```
# Pair Plot
sns.pairplot(data[['day.mins', 'eve.mins', 'night.mins', 'churn']], hue='churn')
plt.show()
```



```
# Save cleaned data for future use
import os

# Define the directory and file path
cleaned_file_path = '/mnt/data/Cleaned_Churn.csv'
directory = os.path.dirname(cleaned_file_path) # Extract the directory path

# Create the directory if it doesn't exist
if not os.path.exists(directory):
    os.makedirs(directory)

# Now save the data to the CSV file
data.to_csv(cleaned_file_path, index=False)

print("Data cleaning and EDA completed. Cleaned data saved to:", cleaned_file_path)
```

📄 Data cleaning and EDA completed. Cleaned data saved to: /mnt/data/Cleaned_Churn.csv

Start coding or [generate](#) with AI.

✓ Feature Engineering Section

```
# Feature Engineering Section
# Step 4: Feature Engineering
# Create new features
data['total.mins'] = data['day.mins'] + data['eve.mins'] + data['night.mins'] + data['intl.mins']
data['total.calls'] = data['day.calls'] + data['eve.calls'] + data['night.calls'] + data['intl.calls']
data['avg.charge.per.call'] = (data['day.charge'] + data['eve.charge'] + data['night.charge'] + data['intl.charge']) / data['total.calls']
```

data

	state	area.code	account.length	voice.plan	voice.messages	intl.plan	intl.mins	intl.calls	intl.charge	day.mins	...
0	KS	area_code_415	128	yes	25	no	10.0	3	2.70	265.1	...
1	OH	area_code_415	107	yes	26	no	13.7	3	3.70	161.6	...
2	NJ	area_code_415	137	no	0	no	12.2	5	3.29	243.4	...
3	OH	area_code_408	84	no	0	yes	6.6	7	1.78	299.4	...
4	OK	area_code_415	75	no	0	yes	10.1	3	2.73	166.7	...
...
4995	HI	area_code_408	50	yes	40	no	9.9	5	2.67	235.7	...
4996	WV	area_code_415	152	no	0	no	14.7	2	3.97	184.2	...
4997	DC	area_code_415	61	no	0	no	13.6	4	3.67	140.6	...
4998	DC	area_code_510	109	no	0	no	8.5	6	2.30	188.8	...
4999	VT	area_code_415	86	yes	34	no	9.3	16	2.51	129.4	...

4969 rows × 23 columns

▼ Encode categorical variables

```
# Encode categorical variables
data['voice.plan'] = data['voice.plan'].map({'yes': 1, 'no': 0})
data['intl.plan'] = data['intl.plan'].map({'yes': 1, 'no': 0})
data['churn'] = data['churn'].map({'yes': 1, 'no': 0})
```

data											
	state	area.code	account.length	voice.plan	voice.messages	intl.plan	intl.mins	intl.calls	intl.charge	day.mins	...
0	KS	area_code_415	128	1	25	0	10.0	3	2.70	265.1	...
1	OH	area_code_415	107	1	26	0	13.7	3	3.70	161.6	...
2	NJ	area_code_415	137	0	0	0	12.2	5	3.29	243.4	...
3	OH	area_code_408	84	0	0	1	6.6	7	1.78	299.4	...
4	OK	area_code_415	75	0	0	1	10.1	3	2.73	166.7	...
...
4995	HI	area_code_408	50	1	40	0	9.9	5	2.67	235.7	...
4996	WV	area_code_415	152	0	0	0	14.7	2	3.97	184.2	...
4997	DC	area_code_415	61	0	0	0	13.6	4	3.67	140.6	...
4998	DC	area_code_510	109	0	0	0	8.5	6	2.30	188.8	...
4999	VT	area_code_415	86	1	34	0	9.3	16	2.51	129.4	...

4969 rows × 23 columns

```
# Save cleaned and engineered data for future use
engineered_file_path = '/mnt/data/Engineered_Churn.csv'
data.to_csv(engineered_file_path, index=False)

print("EDA completed. Feature Engineering completed. Data saved to:", engineered_file_path)
```

EDA completed. Feature Engineering completed. Data saved to: /mnt/data/Engineered_Churn.csv

Start coding or [generate](#) with AI.

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
import pickle
```

```
# Select only numerical features for scaling
numerical_features = data.select_dtypes(include=['number']).drop(columns=['churn'])
```

```
# Define features and target
X = numerical_features
y = data['churn']
```

Split

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

```
# Standardize numerical features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

Model Building

```
# Model Building
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train_scaled, y_train)
```

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

Model Evaluation

```
# Model Evaluation
y_pred = model.predict(X_test_scaled)
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
class_report = classification_report(y_test, y_pred)
```

```
print("Model Accuracy:", accuracy)
print("Confusion Matrix:\n", conf_matrix)
print("Classification Report:\n", class_report)
```

```
Model Accuracy: 0.954728370221328
Confusion Matrix:
[[829  4]
 [ 41 120]]
Classification Report:
              precision    recall  f1-score   support

      0       0.95      1.00      0.97       833
      1       0.97      0.75      0.84       161

   accuracy      0.95      0.95      0.95       994
  macro avg      0.96      0.87      0.91       994
weighted avg      0.96      0.95      0.95       994
```

Feedback based on model performance

```
# Feedback based on model performance
if accuracy > 0.85:
    print("Great model performance! Consider fine-tuning hyperparameters for further improvement.")
elif accuracy > 0.75:
    print("Good performance, but there is room for improvement. Try feature selection or different algorithms.")
else:
    print("The model needs improvement. Consider engineering new features or using more complex models.")
```

```
Great model performance! Consider fine-tuning hyperparameters for further improvement.
```