```
import pandas as pd
import numpy as np
from sklearn.preprocessing import LabelEncoder
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score , classification_report , confusion_matrix
from sklearn.linear_model import LogisticRegression
import joblib
from flask import Flask, request, jsonify
```

```
np.random.seed(42)

customers = 1000

data = {

        "CustomerID" : range(1001 , 1001 + customers),
        "Gender" : np.random.choice(["Female" , "Male"] , customers),
        "Age" : np.random.randint(15 , 70 , customers),
        "ServiceLength (months)" : np.random.randint(1 , 70 , customers),
        "ContractType" : np.random.choice(["Month-to-Month", "One-Year", "Two-Year"] , customers),
        "MonthlyCharges (USD)" : np.random.uniform(50 , 1000 , customers),
        "TotalCharges (USD)" : np.random.uniform(50 , 150000 , customers),
        "Churn"  : np.random.choice(["Yes" , "No"] , customers , p=[0.3 ,0.7])
}
df = pd.DataFrame(data)
```

```
df.isna().sum()
```

```
CustomerID                0
Gender                    0
Age                       0
ServiceLength (months)    0
ContractType              0
MonthlyCharges (USD)      0
TotalCharges (USD)        0
Churn                     0
dtype: int64
```

```
df.drop_duplicates(inplace = True)
```

```
encoder = LabelEncoder()
df["Gender"] = encoder.fit_transform(data["Gender"])
df["ContractType"] = encoder.fit_transform(data["ContractType"])
```

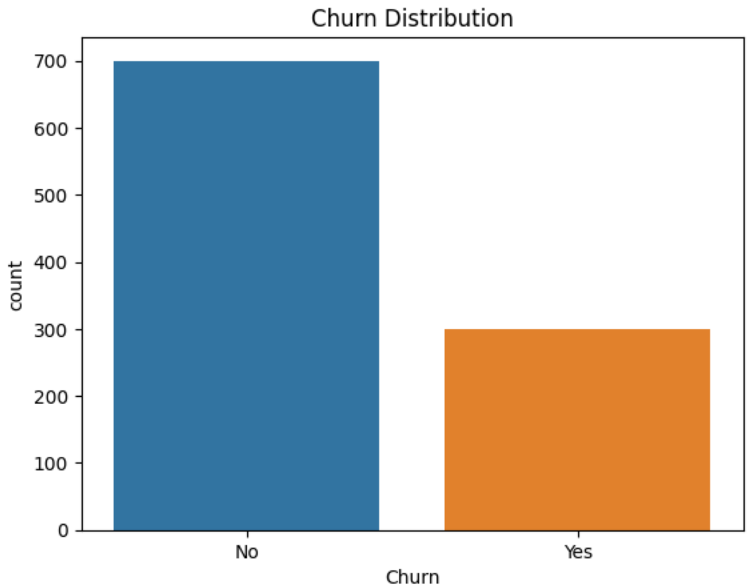
```
df.head(5)
```

	CustomerID	Gender	Age	ServiceLength (months)	ContractType	MonthlyCharges (USD)	TotalCharges (USD)	Churn
0	1001	0	68	15	0	254.953645	7583.939323	No
1	1002	1	31	49	1	693.922540	5284.987321	No
2	1003	0	23	68	0	627.364516	82696.436289	No
3	1004	0	47	12	1	330.536404	65754.883057	No
4	1005	0	67	59	1	179.770858	125885.100227	Yes

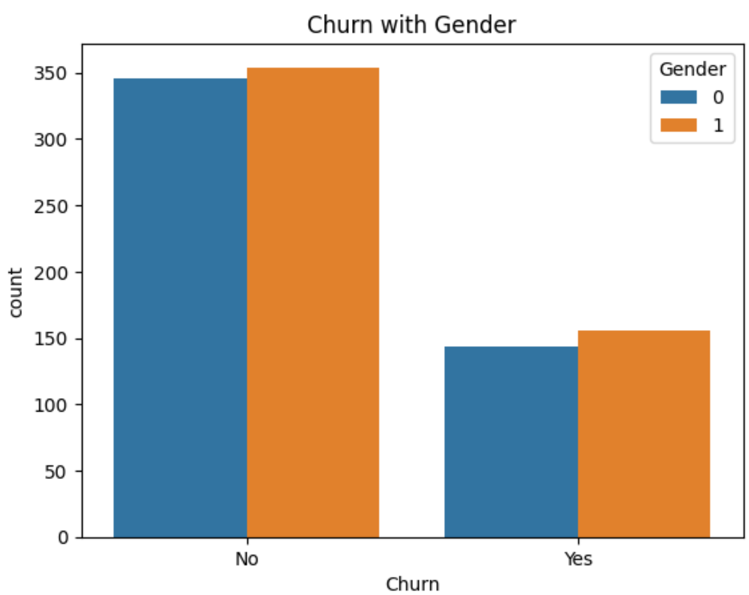
```
df.describe()
```

	CustomerID	Gender	Age	ServiceLength (months)	ContractType	MonthlyCharges (USD)	TotalCharges (USD)
count	1000.000000	1000.00000	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000
mean	1500.500000	0.51000	42.503000	33.627000	0.989000	522.379235	73802.187427
std	288.819436	0.50015	16.010444	19.876341	0.829196	273.501131	42887.306291

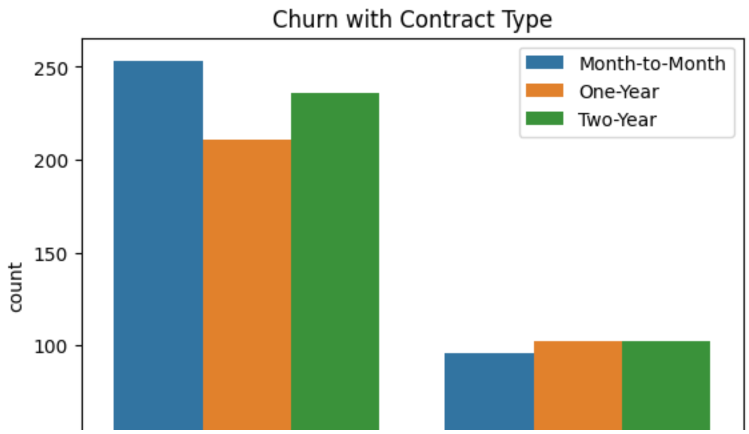
```
sns.countplot(x = "Churn" , data = df)
plt.title("Churn Distribution")
plt.show()
```



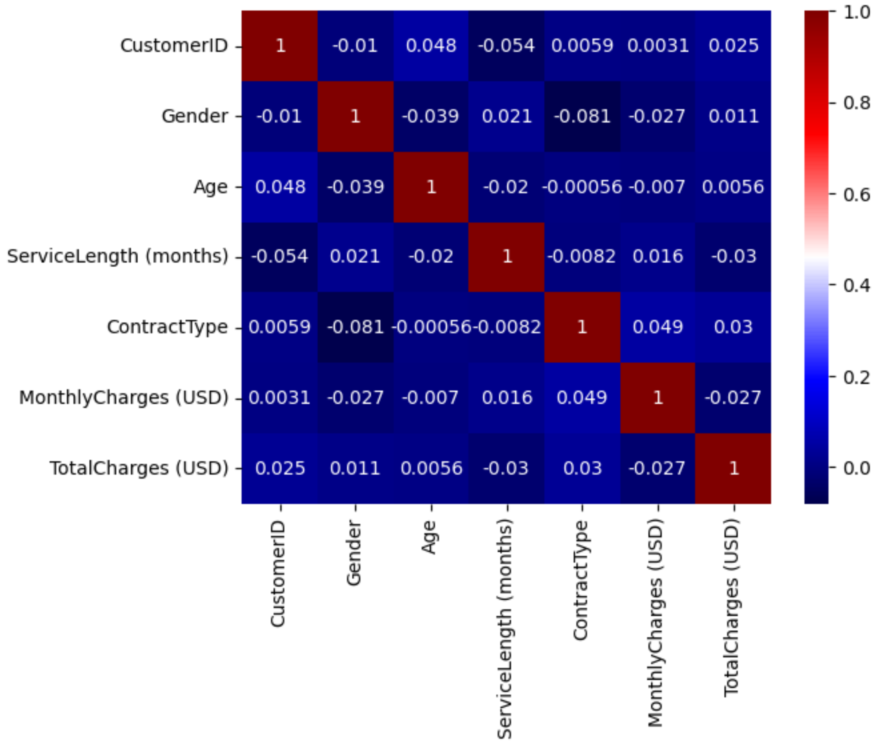

```
sns.countplot(x = "Churn" , hue = "Gender" , data = df)
plt.title("Churn with Gender")
plt.show()
```




```
sns.countplot(x="Churn", hue="ContractType", data=data)
plt.title("Churn with Contract Type")
plt.show()
```




```
correlation = df.corr()
sns.heatmap(correlation , annot = True , cmap = "seismic")
plt.show()
```




```
#data splitting
x = df.drop("Churn" , 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)
```

```
#with random forest
rnmodel = RandomForestClassifier(n_estimators=100, random_state=42)
rnmodel.fit(x_train , y_train)
```

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

```
prediction = rnmodel.predict(x_test)
```

```
print("Accuracy:", accuracy_score(y_test, prediction))
print("Report:\n", classification_report(y_test, prediction))
print("Confusion matrix:\n", confusion_matrix(y_test, prediction))
```

```
Accuracy: 0.67
Report:
               precision    recall  f1-score   support

          No       0.69      0.93      0.79       137
         Yes       0.41      0.11      0.18        63

    accuracy                           0.67       200
   macro avg       0.55      0.52      0.48       200
weighted avg       0.61      0.67      0.60       200

Confusion matrix:
 [[127  10]
 [ 56   7]]
```

```
#with logistic regression
model = LogisticRegression(random_state = 42)
model.fit(x_train , y_train)
```

```
▾        LogisticRegression
LogisticRegression(random_state=42)
```

```
prediction = model.predict(x_test)
```

```
print("Accuracy:", accuracy_score(y_test, prediction))
print("Report:\n", classification_report(y_test, prediction))
print("Confusion matrix:\n", confusion_matrix(y_test, prediction))
```

```
Accuracy: 0.685
Report:
               precision    recall  f1-score   support

          No       0.69      1.00      0.81       137
         Yes       0.00      0.00      0.00        63

    accuracy                           0.69       200
   macro avg       0.34      0.50      0.41       200
weighted avg       0.47      0.69      0.56       200

Confusion matrix:
 [[137   0]
 [ 63   0]]
```

Random Forest model is better in terms of overall accuracy and its ability to correctly identify the "No" class with higher precision and recall. However, Random Forest model still struggles to correctly predict the "Yes" class as indicated by its low precision, recall, and F1-score.

```
app = Flask(__name__)

# Load the serialized model
model_filename = 'random_forest_churn_model.pkl'
model = joblib.load(model_filename)

@app.route('/predict', methods=['POST'])
def predict():
    try:
        data = request.get_json()
        features = data['features']  # Access the features from the JSON data

        # Make predictions using the loaded model
        prediction = model.predict([features])

        return jsonify({'prediction': prediction.tolist()})
    except Exception as e:
        return jsonify({'error': str(e)})

if __name__ == '__main__':
    app.run(debug=True)
```