```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.metrics import classification_report, accuracy_score, confusion_matrix

# Step 1: Create a sample dataset and save it as customer_churn.csv
data = {
    'gender': ['Male', 'Female', 'Male', 'Female', 'Female', 'Male'],
    'tenure': [1, 24, 12, 5, 36, 8],
    'MonthlyCharges': [29.85, 56.95, 53.85, 42.30, 70.10, 62.25],
    'Contract': ['Month-to-month', 'One year', 'Month-to-month', 'Two year', 'Month-to-month', 'One year'],
    'Churn': ['Yes', 'No', 'No', 'Yes', 'Yes', 'No']
}

df_sample = pd.DataFrame(data)
df_sample.to_csv("customer_churn.csv", index=False)  # Save to file

# Step 2: Read the CSV file (no error now)
df = pd.read_csv("customer_churn.csv")

# Step 3: Handle missing values
df.fillna(method='ffill', inplace=True)

# Step 4: Encode categorical variables
le = LabelEncoder()
for col in df.select_dtypes(include='object').columns:
    if col != 'Churn':
        df[col] = le.fit_transform(df[col])

# Encode the target variable 'Churn'
df['Churn'] = df['Churn'].map({'Yes': 1, 'No': 0})

# Step 5: Split features and target
X = df.drop('Churn', axis=1)
y = df['Churn']

# Step 6: Train/test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Step 7: Standardize features
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Step 8: Train Random Forest model
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

# Step 9: Predict and evaluate
y_pred = model.predict(X_test)

print("\nClassification Report:\n", classification_report(y_test, y_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("Accuracy Score:", accuracy_score(y_test, y_pred))

# Step 10: Visualize confusion matrix
sns.heatmap(confusion_matrix(y_test, y_pred), annot=True, fmt='d', cmap='YlGnBu')
plt.title("Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()
```

```
<ipython-input-3-0e62707e9634>:26: FutureWarning: DataFrame.fillna with 'method' is deprecated and will raise in a future version. U
  df.fillna(method='ffill', inplace=True)
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined ar
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined ar
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined ar
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))

Classification Report:
              precision    recall  f1-score   support

           0       0.50      1.00      0.67         1
           1       0.00      0.00      0.00         1

    accuracy                           0.50         2
   macro avg       0.25      0.50      0.33         2
weighted avg       0.25      0.50      0.33         2

Confusion Matrix:
 [[1 0]
 [1 0]]
Accuracy Score: 0.5
```