```
from google.colab import files
upload=files.upload()
```

Choose Files Heart_Attac…Dataset.csv

- **Heart_Attack_Risk_Levels_Dataset.csv**(text/csv) - 97910 bytes, last modified: 4/16/2025 - 100% done

```
Saving Heart_Attack_Risk_Levels_Dataset.csv to Heart_Attack_Risk_Levels_Dataset.csv
```

```
import pandas as pd
df=pd.read_csv("Heart_Attack_Risk_Levels_Dataset.csv")
df
```

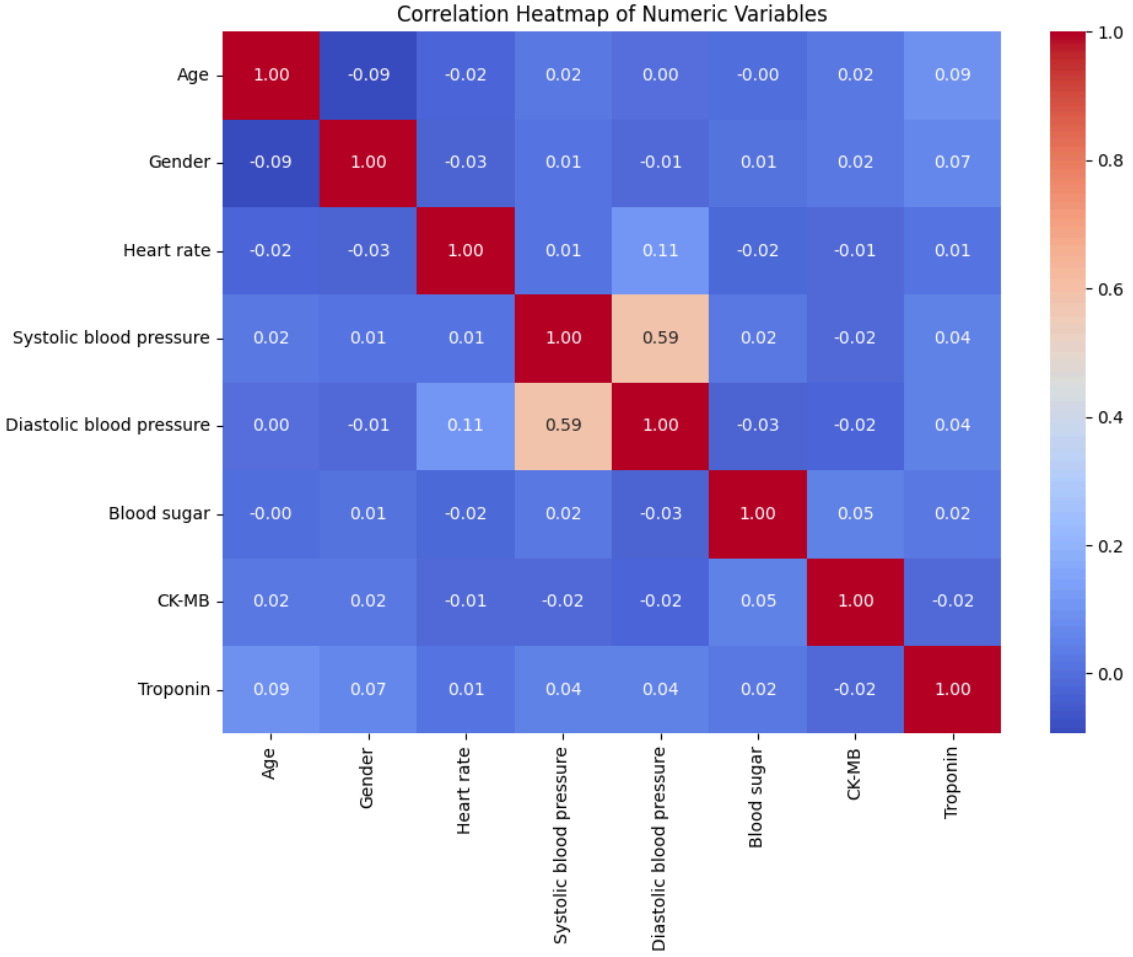
	Age	Gender	Heart rate	Systolic blood pressure	Diastolic blood pressure	Blood sugar	CK-MB	Troponin	Result	Risk_Level	Recommendation
0	63	1	66	160	83	160.0	1.80	0.012	negative	Moderate	Monitor closely and consult doctor
1	20	1	94	98	46	296.0	6.75	1.060	positive	High	Immediate medical attention
2	56	1	64	160	77	270.0	1.99	0.003	negative	Moderate	Monitor closely and consult doctor
3	66	1	70	120	55	270.0	13.87	0.122	positive	High	Immediate medical attention
4	54	1	64	112	65	300.0	1.08	0.003	negative	Moderate	Monitor closely and consult doctor
...	...	...	...	...	...	...	...	...	...	...	...
1314	44	1	94	122	67	204.0	1.63	0.006	negative	Moderate	Monitor closely and consult doctor
1315	66	1	84	125	55	149.0	1.33	0.172	positive	High	Immediate medical attention

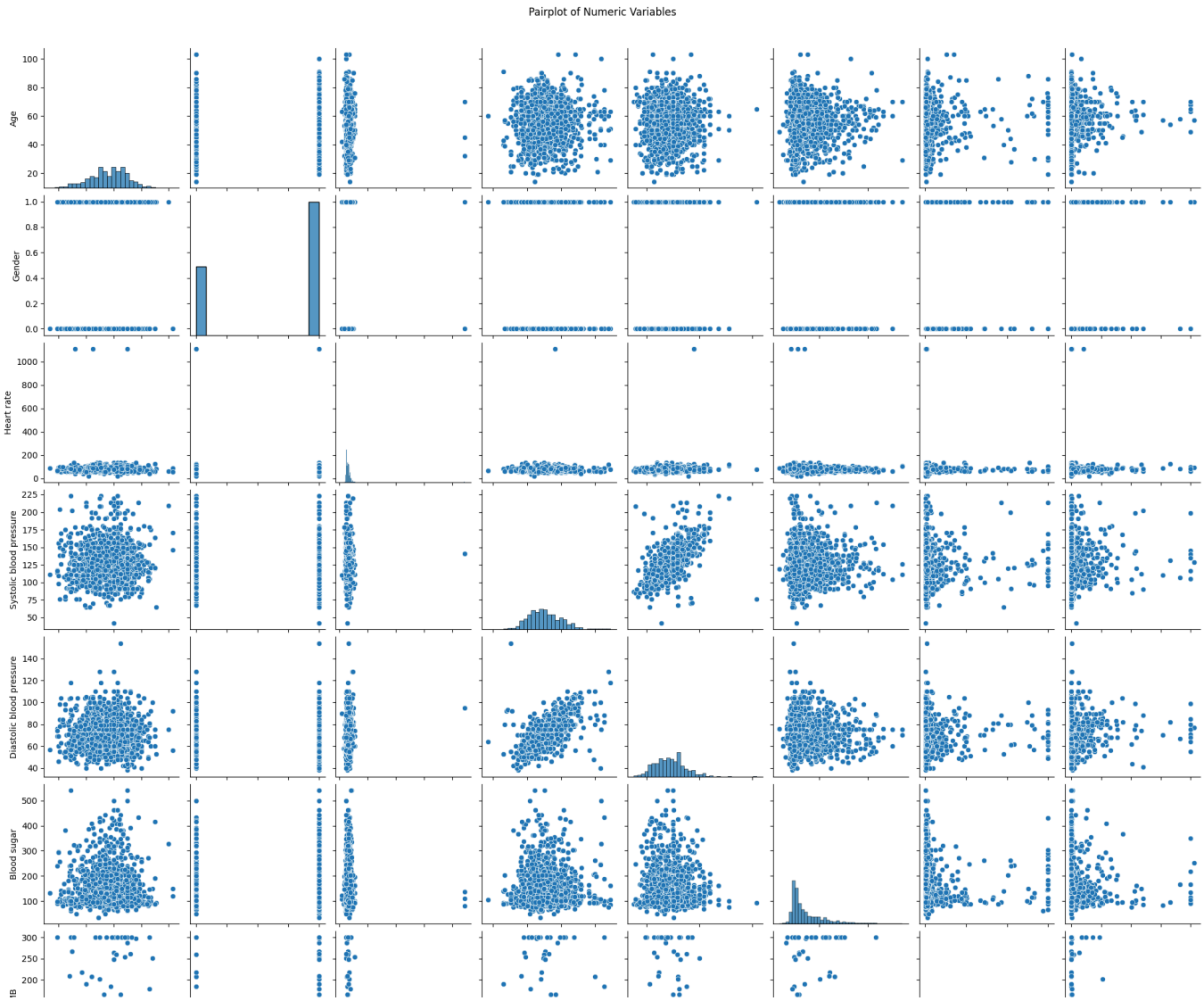
Next steps: Generate code with df | View recommended plots | New interactive sheet

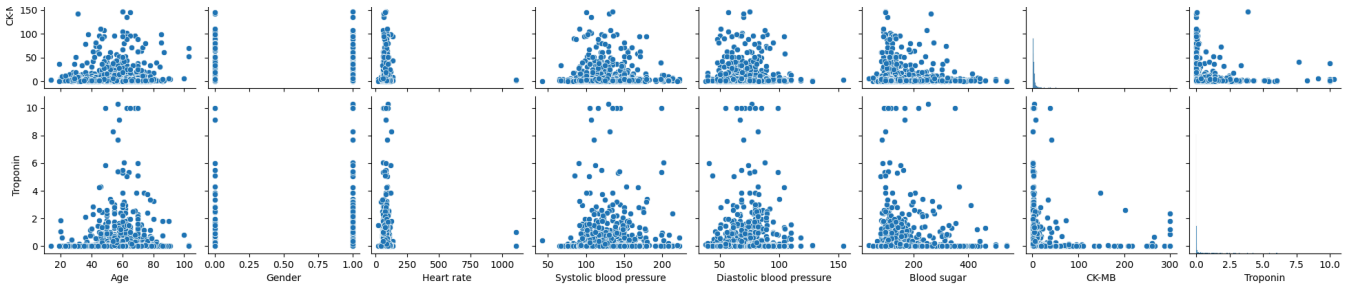
```
print('Dataframe Info:')
df.info()

# Check for missing values
print('\nMissing values in each column:')
print(df.isnull().sum())

# Renaming columns (if necessary) to remove extra spaces
df.columns = [col.strip() for col in df.columns]

# Verify changes
print('\nUpdated column names:')
print(list(df.columns))
```

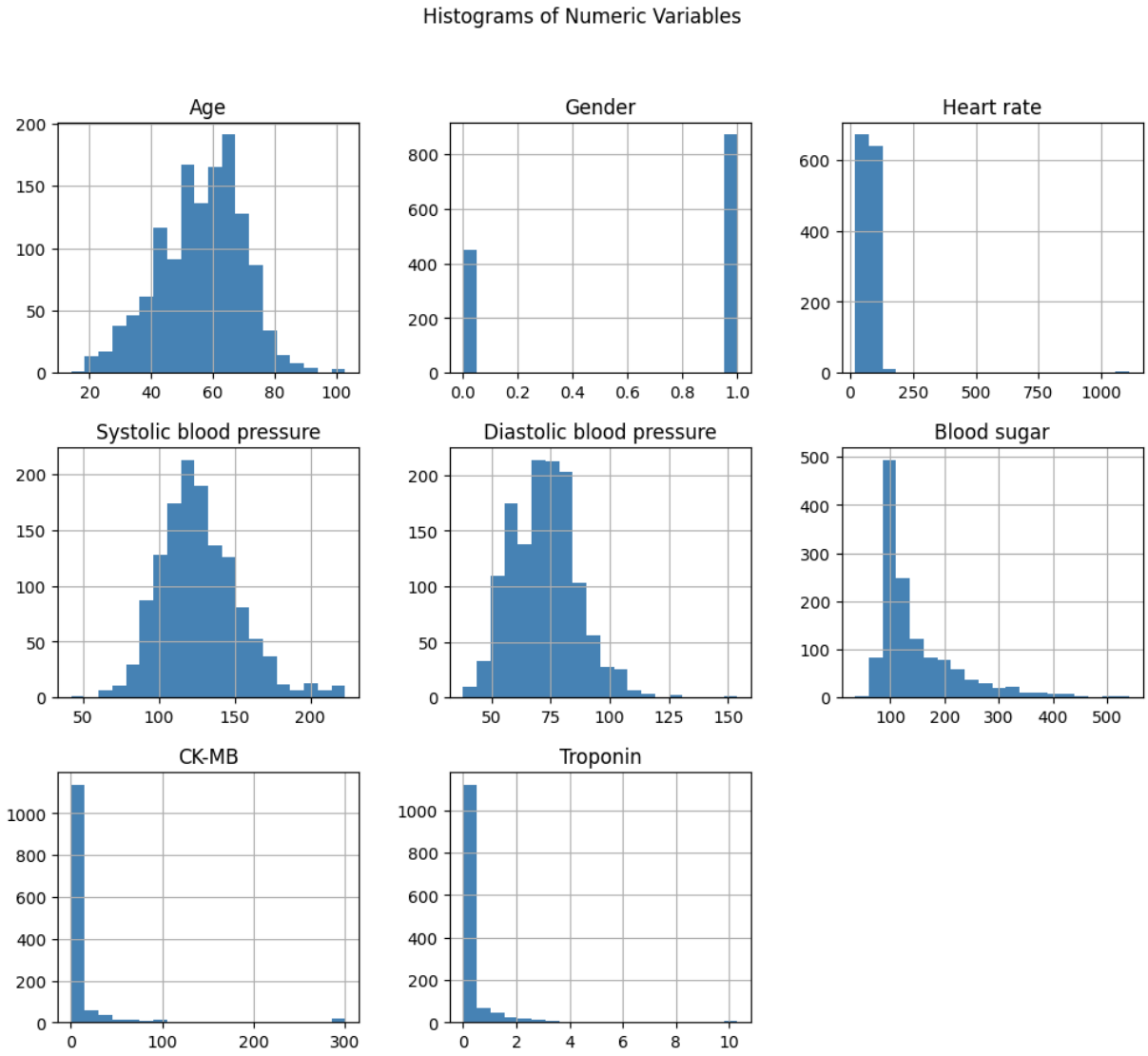
```
Dataframe Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1319 entries, 0 to 1318
Data columns (total 11 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Age                       1319 non-null   int64
 1   Gender                    1319 non-null   int64
 2   Heart rate                1319 non-null   int64
 3   Systolic blood pressure   1319 non-null   int64
 4   Diastolic blood pressure  1319 non-null   int64
 5   Blood sugar               1319 non-null   float64
 6   CK-MB                     1319 non-null   float64
 7   Troponin                  1319 non-null   float64
 8   Result                    1319 non-null   object
 9   Risk_Level                1319 non-null   object
 10  Recommendation            1319 non-null   object
dtypes: float64(3), int64(5), object(3)
memory usage: 113.5+ KB

Missing values in each column:
Age                         0
Gender                      0
Heart rate                  0
Systolic blood pressure     0
Diastolic blood pressure    0
Blood sugar                 0
CK-MB                       0
Troponin                    0
Result                      0
Risk_Level                  0
Recommendation              0
dtype: int64

Updated column names:
['Age', 'Gender', 'Heart rate', 'Systolic blood pressure', 'Diastolic blood pressure', 'Blood sugar', 'CK-MB', 'Troponin', 'Result', 'Risk_Level']
```

```
import pandas as pd
import numpy as np  # Import numpy and assign it the alias 'np'
import matplotlib.pyplot as plt
import seaborn as sns

# Set up the numeric dataframe for correlation analysis
numeric_df = df.select_dtypes(include=[np.number])

# If there are four or more numeric columns, plot the correlation heatmap
if numeric_df.shape[1] >= 4:
    plt.figure(figsize=(10, 8))
    corr = numeric_df.corr()
    sns.heatmap(corr, annot=True, fmt='.2f', cmap='coolwarm')
    plt.title('Correlation Heatmap of Numeric Variables')
    plt.tight_layout()
    plt.show()

# Pairplot for numeric features
sns.pairplot(numeric_df)
plt.suptitle('Pairplot of Numeric Variables', y=1.02)
plt.show()

# Histograms of numeric variables
numeric_df.hist(figsize=(12,10), bins=20, color='steelblue')
plt.suptitle('Histograms of Numeric Variables')
plt.show()

# Count plot for Risk_Level (categorical variable)
plt.figure(figsize=(8, 5))
sns.countplot(x='Risk_Level', data=df, palette='Set2')
plt.title('Distribution of Risk Levels')
plt.tight_layout()
plt.show()
# Box plot to inspect the distribution of Heart rate by Risk_Level
plt.figure(figsize=(8,5))
sns.boxplot(x='Risk_Level', y='Heart rate', data=df, palette='Set3')
plt.title('Heart Rate Distribution by Risk Level')
plt.tight_layout()
plt.show()
```

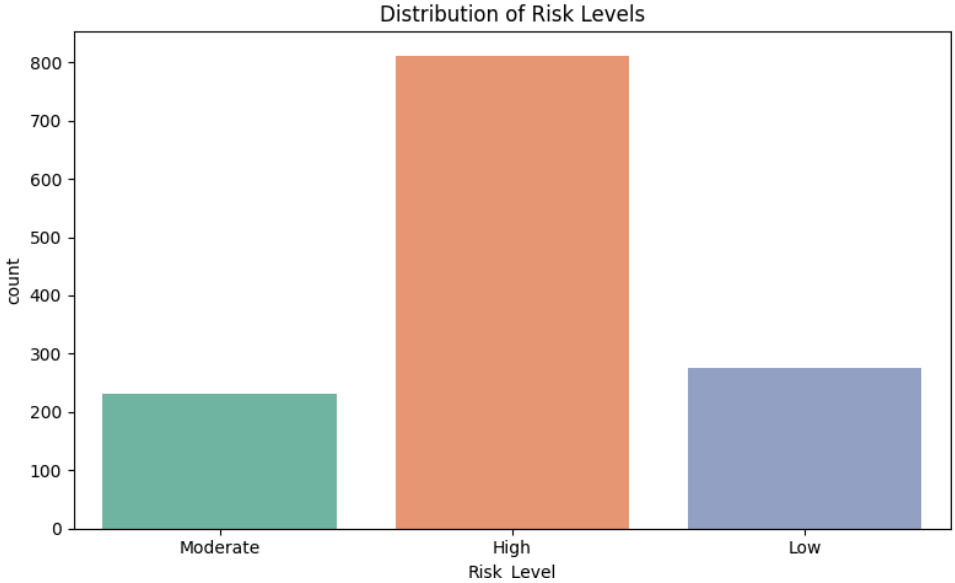

Correlation Heatmap of Numeric Variables
Age 1.00 -0.09 -0.02 0.02 0.00 -0.00 0.02 0.09
Gender -0.09 1.00 -0.03 0.01 -0.01 0.01 0.02 0.07
Heart rate -0.02 -0.03 1.00 0.01 0.11 -0.02 -0.01 0.01
Systolic blood pressure 0.02 0.01 0.01 1.00 0.59 0.02 -0.02 0.04
Diastolic blood pressure 0.00 -0.01 0.11 0.59 1.00 -0.03 -0.02 0.04
Blood sugar -0.00 0.01 -0.02 0.02 -0.03 1.00 0.05 0.02
CK-MB 0.02 0.02 -0.01 -0.02 -0.02 0.05 1.00 -0.02
Troponin 0.09 0.07 0.01 0.04 0.04 0.02 -0.02 1.00
Age
Gender
Heart rate
Systolic blood pressure
Diastolic blood pressure
Blood sugar
CK-MB
Troponin
1.0
0.8
0.6
0.4
0.2
0.0


Pairplot of Numeric Variables
Age
Gender
Heart rate
Systolic blood pressure
Diastolic blood pressure
Blood sugar

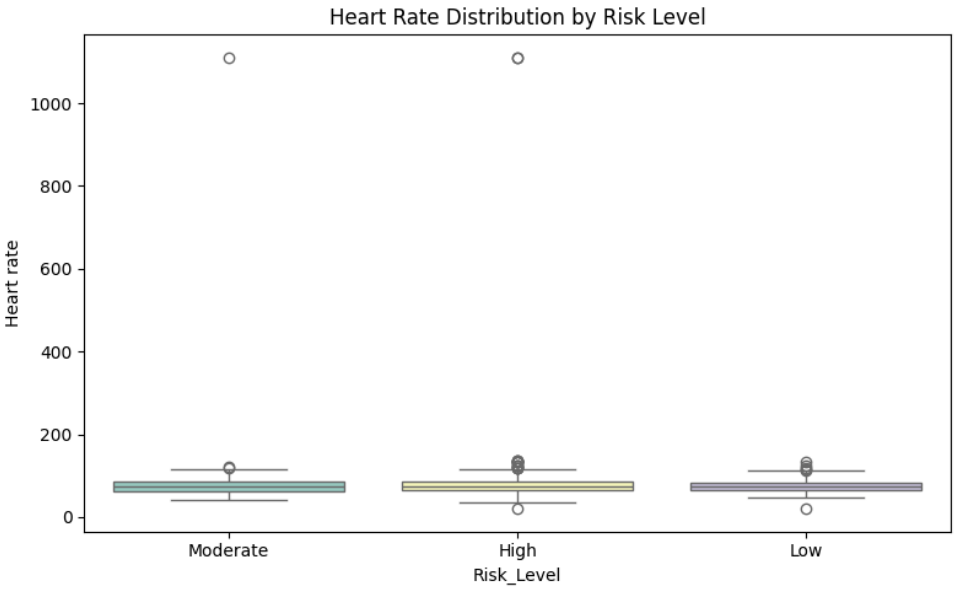





```
<ipython-input-5-5a21a94d4462>:30: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False`

  sns.countplot(x='Risk_Level', data=df, palette='Set2')
```



```
<ipython-input-5-5a21a94d4462>:36: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False`

  sns.boxplot(x='Risk_Level', y='Heart rate', data=df, palette='Set3')
```



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

# Define predictor variables (only numeric features) and target variable
X = df[['Age', 'Gender', 'Heart rate', 'Systolic blood pressure', 'Diastolic blood pressure', 'Blood sugar', 'CK-MB', 'Troponin']]
y = df['Risk_Level']

# Encode the target variable to numeric labels
le = LabelEncoder()
y_encoded = le.fit_transform(y)

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y_encoded, test_size=0.3, random_state=42)

# Initialize and train the Random Forest model
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)

# Make predictions on the test set
y_pred = rf_model.predict(X_test)

# Evaluate the predictor accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f'Prediction Accuracy: {accuracy:.2f}')
# Generate and plot the Confusion Matrix
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.title('Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.tight_layout()
plt.show()

# Plot Feature Importances
feature_importances = rf_model.feature_importances_
feat_imp_series = pd.Series(feature_importances, index=X.columns).sort_values(ascending=True)
plt.figure(figsize=(8, 6))
feat_imp_series.plot(kind='barh', color='teal')
plt.title('Feature Importances')
plt.xlabel('Importance')
plt.tight_layout()
plt.show()

# Print classification report
print('Classification Report:')
print(classification_report(y_test, y_pred, target_names=le.classes_))
```

```
Prediction Accuracy: 0.98
```