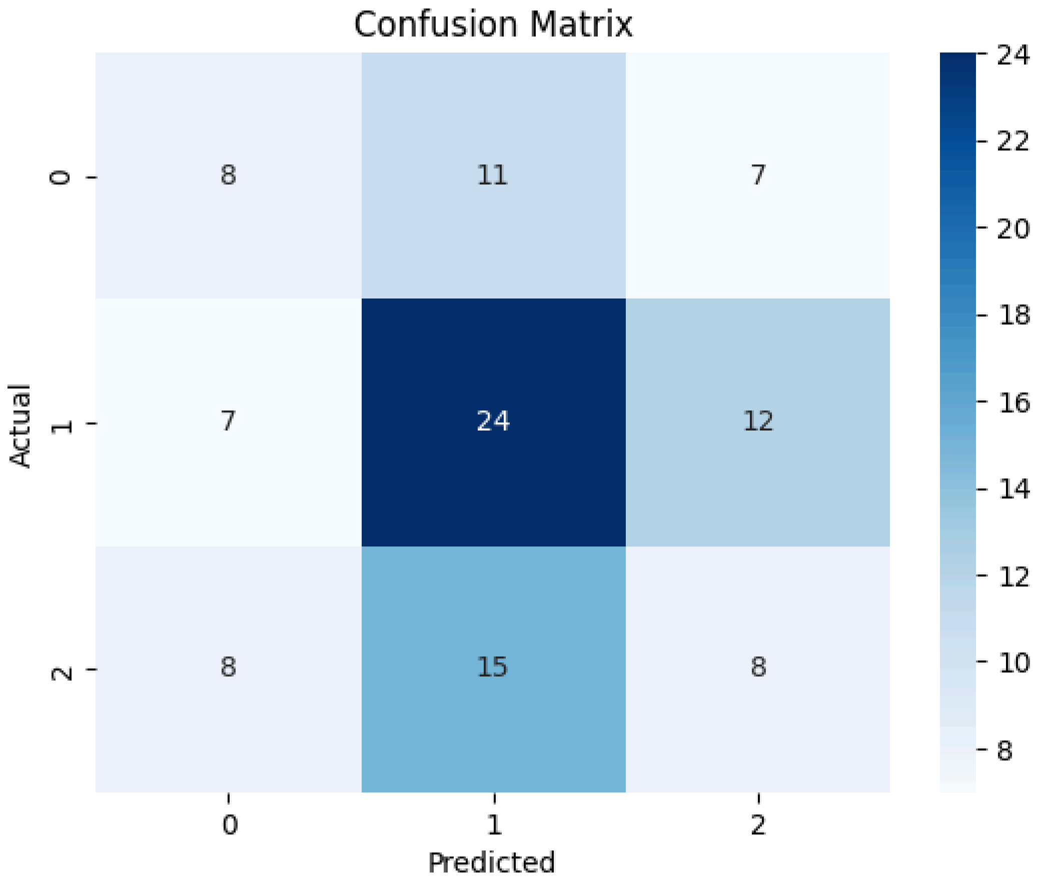
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix
import seaborn as sns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

# Create mock data
data = {
    'Weather_Conditions': np.random.choice(['Fine', 'Rain', 'Snow'], 500),
    'Light_Conditions': np.random.choice(['Daylight', 'Darkness'], 500),
    'Road_Surface_Conditions': np.random.choice(['Dry', 'Wet'], 500),
    'Vehicle_Type': np.random.choice(['Car', 'Bike', 'Truck'], 500),
    'Accident_Severity': np.random.choice(['Slight', 'Serious', 'Fatal'], 500)
}

df = pd.DataFrame(data)

# Encode categorical data
encoders = {}
for col in df.columns:
    le = LabelEncoder()
    df[col] = le.fit_transform(df[col])
    encoders[col] = le

X = df[['Weather_Conditions', 'Light_Conditions', 'Road_Surface_Conditions', 'Vehicle_Type'
y = df['Accident_Severity']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

y_pred = model.predict(X_test)

print(classification_report(y_test, y_pred))

cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix")
plt.show()

importances = pd.Series(model.feature_importances_, index=X.columns)
importances.sort_values().plot(kind='barh', color='orange')
plt.title("Feature Importance")
plt.show()
```

```
              precision    recall  f1-score   support

           0       0.35      0.31      0.33        26
           1       0.48      0.56      0.52        43
           2       0.30      0.26      0.28        31

    accuracy                           0.40       100
   macro avg       0.37      0.37      0.37       100
weighted avg       0.39      0.40      0.39       100
```



Feature Importance