```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder,
StandardScaler

from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report,
accuracy_score, confusion_matrix
# Load data
df = pd.read_excel("/kaggle/input/road-accidents-and-
conditions/Road Accidents.xlsx")
# Remove Duplicate Rows
df.drop_duplicates(inplace=True)
# Standardize Categorical Features (Lowercase & Strip)
df = df.apply(lambda x: x.str.lower().str.strip() if x.dtype ==
"object" else x)
# Handle Outliers using IQR Method
numeric_columns = df.select_dtypes(include=['int64',
'float64']).columns
for col in numeric_columns:
    Q1 = df[col].quantile(0.25)
    Q3 = df[col].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    df[col] = df[col].clip(lower=lower_bound, upper=upper_bound)
# Feature Engineering: Time-Based Features

df['Accident Date'] = pd.to_datetime(df['Accident Date'])
df['Month'] = df['Accident Date'].dt.month
df['DayOfWeek'] = df['Accident Date'].dt.dayofweek
df['Hour'] = df['Time (24hr)'] // 100
# Remove Unnecessary Columns
df.drop(columns=['Reference Number', 'Easting', 'Northing', 'Accident Date', 'Time (24hr)'], inplace=True)
# Identify categorical columns (excluding target)
categorical_columns = df.select_dtypes(include=['object']).columns.tolist()
categorical_columns.remove("Casualty Severity")  # Exclude target variable
# Apply Label Encoding ONLY to 'Casualty Severity'
label_encoder = LabelEncoder()
df['Casualty Severity'] = label_encoder.fit_transform(df['Casualty Severity'])
# Convert categorical features to numerical using Label Encoding
```

```
for col in categorical_columns:
    df[col] = LabelEncoder().fit_transform(df[col])
# Define Features (X) and Target (y)
X = df.drop(columns=['Casualty Severity'])
y = df['Casualty Severity']


# Train-Test Split (without SMOTE)
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42, stratify=y)
# Normalize Numerical Features (Avoid Data Leakage)
numeric_features = ['Number of Vehicles', 'Hour', 'Month']
scaler = StandardScaler()
X_train[numeric_features] =
scaler.fit_transform(X_train[numeric_features])
X_test[numeric_features] =
scaler.transform(X_test[numeric_features])
# Model Training: Random Forest Classifier
rf_model = RandomForestClassifier(n_estimators=100,
random_state=42)
rf_model.fit(X_train, y_train)
# Model Evaluation
y_pred = rf_model.predict(X_test)
print("Model Accuracy:", accuracy_score(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))

# Confusion Matrix Plot (Fixed)

cm = confusion_matrix(y_test, y_pred)

plt.figure(figsize=(6,4))




sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", xticklabels=label_encoder.classes_, yticklabels=label_encoder.classes_)

plt.xlabel("Predicted")

plt.ylabel("Actual")

plt.title("Confusion Matrix")

plt.show()
```