UPLOAD AN LOAD DATA

```
from google.colab import files
uploaded = files.upload()
import pandas as pd
df = pd.read_excel("Road Accident Data (1).xlsx")
```

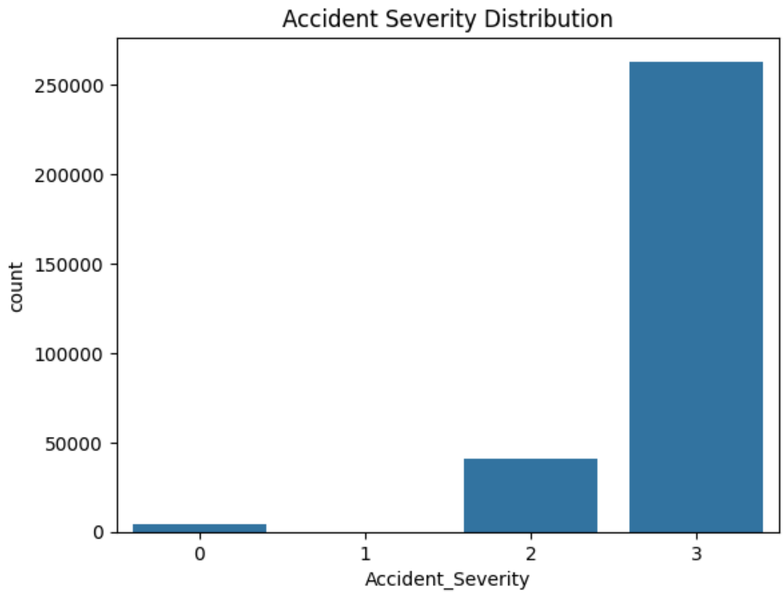
Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

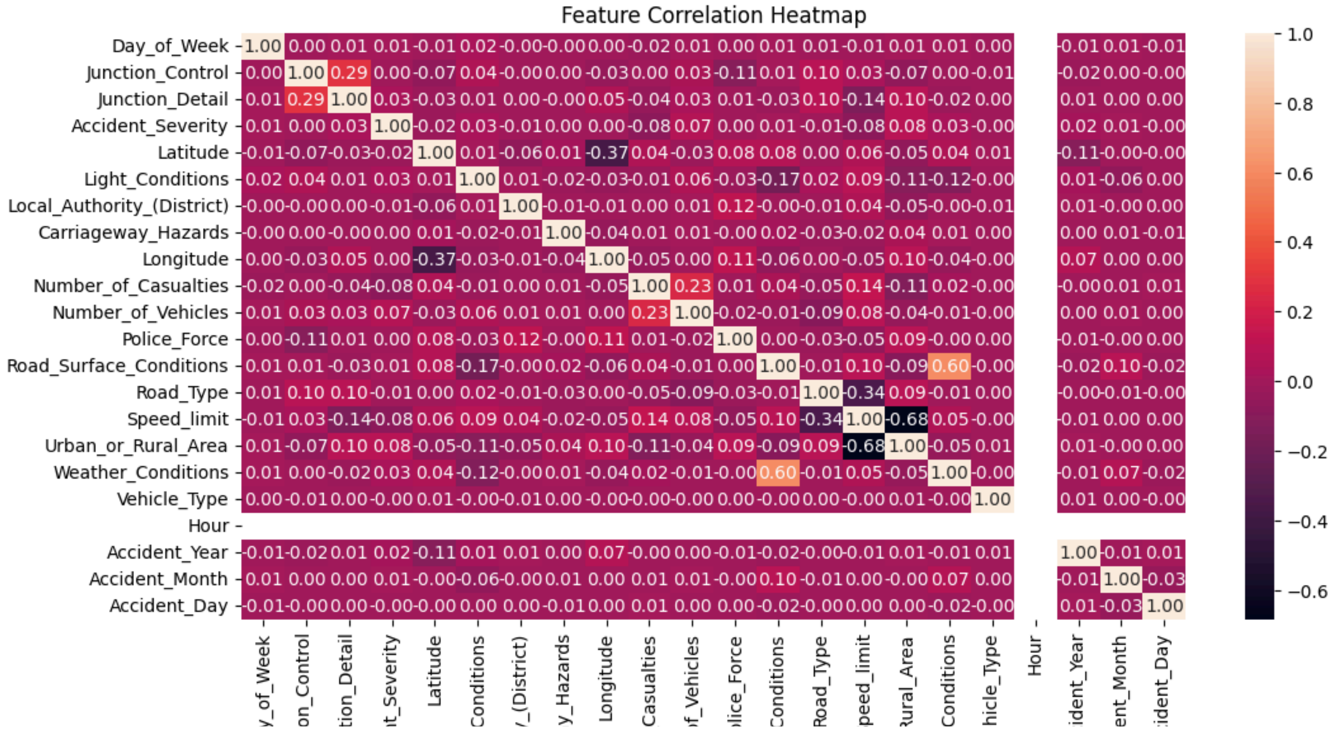
```
Saving Road Accident Data (1).xlsx to Road Accident Data (1) (1).xlsx
```

```
# STEP 2: Preprocessing
df.dropna(thresh=int(0.7 * df.shape[1]), inplace=True)
df.fillna(method='ffill', inplace=True)

if 'Accident_Index' in df.columns:
    df.drop(['Accident_Index'], axis=1, inplace=True)

if 'Time' in df.columns:
    df['Hour'] = pd.to_datetime(df['Time'], errors='coerce').dt.hour
    df.drop('Time', axis=1, inplace=True)

# Encode categorical columns
from sklearn.preprocessing import LabelEncoder
categorical_cols = df.select_dtypes(include='object').columns
le = LabelEncoder()
for col in categorical_cols:
    try:
        df[col] = le.fit_transform(df[col].astype(str))
    except:
        print(f"Could not encode {col}")
```

```
<ipython-input-17-c8844b0d048a>:3: FutureWarning: DataFrame.fillna with 'method' is deprecated and will raise in a future version. U
  df.fillna(method='ffill', inplace=True)
```

```
# STEP 3: EDA (Example Plots)
import seaborn as sns
import matplotlib.pyplot as plt

sns.countplot(x='Accident_Severity', data=df)
plt.title('Accident Severity Distribution')
plt.show()

plt.figure(figsize=(12, 6))
sns.heatmap(df.corr(), annot=True, fmt='.2f')
plt.title('Feature Correlation Heatmap')
plt.show()
```





```
#STEP 4: Feature Engineering
from sklearn.preprocessing import StandardScaler
import pandas as pd
from sklearn.impute import SimpleImputer # Import SimpleImputer

# Convert 'Accident Date' to numerical features
# Extract year, month, and day
if 'Accident Date' in df.columns:
    df['Accident_Year'] = pd.to_datetime(df['Accident Date']).dt.year
    df['Accident_Month'] = pd.to_datetime(df['Accident Date']).dt.month
    df['Accident_Day'] = pd.to_datetime(df['Accident Date']).dt.day
    # Drop the original 'Accident Date' column
    df.drop('Accident Date', axis=1, inplace=True)


X = df.drop('Accident_Severity', axis=1)
y = df['Accident_Severity']

# Select only numerical features for scaling
numerical_features = X.select_dtypes(include=['number']).columns
X_numerical = X[numerical_features]

# Impute missing values before scaling # New lines to impute missing values
imputer = SimpleImputer(strategy='mean') # or 'median', 'most_frequent'
X_numerical = imputer.fit_transform(X_numerical)

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X_numerical)
```

```
/usr/local/lib/python3.11/dist-packages/sklearn/impute/_base.py:635: UserWarning: Skipping features without any observed values: ['
  warnings.warn(
```

```
# STEP 5: Model Training
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier

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

# Logistic Regression
lr = LogisticRegression()
lr.fit(X_train, y_train)
y_pred_lr = lr.predict(X_test)

# Random Forest
rf = RandomForestClassifier(n_estimators=100)
rf.fit(X_train, y_train)
y_pred_rf = rf.predict(X_test)
```

```
# STEP 6: Evaluation
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)

# Initialize and train Logistic Regression model
lr = LogisticRegression(max_iter=1000, random_state=42) # Increased max_iter
lr.fit(X_train, y_train)

# Make predictions using Logistic Regression
y_pred_lr = lr.predict(X_test)

# Initialize and train Random Forest model
rf = RandomForestClassifier(random_state=42)
rf.fit(X_train, y_train)

# Make predictions using Random Forest
y_pred_rf = rf.predict(X_test)


print("Logistic Regression:\n", classification_report(y_test, y_pred_lr))
print("Random Forest:\n", classification_report(y_test, y_pred_rf))

# Confusion Matrix
plt.figure(figsize=(6,4))
sns.heatmap(confusion_matrix(y_test, y_pred_rf), annot=True, fmt='d', cmap='Blues')
plt.title("Confusion Matrix - Random Forest")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()

# Feature Importance
import numpy as np
features = X.columns
importances = rf.feature_importances_
indices = np.argsort(importances)[::-1]

plt.figure(figsize=(10,6))
sns.barplot(x=importances[indices], y=features[indices])
plt.title("Feature Importance (Random Forest)")
plt.show()
```

```
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined
  _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
  _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
  _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
  _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
  _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
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
Logistic Regression:
               precision    recall  f1-score   support

           0       0.50      0.00      0.01       823
           1       0.00      0.00      0.00         8
           2       0.27      0.00      0.00      8101
           3       0.86      1.00      0.92     52663

    accuracy                           0.85     61595
   macro avg       0.41      0.25      0.23     61595
weighted avg       0.77      0.85      0.79     61595

Random Forest:
               precision    recall  f1-score   support

           0       0.62      0.01      0.01       823
           1       0.00      0.00      0.00         8
           2       0.30      0.01      0.02      8101
           3       0.86      1.00      0.92     52663

    accuracy                           0.85     61595
   macro avg       0.44      0.25      0.24     61595
weighted avg       0.78      0.85      0.79     61595
```