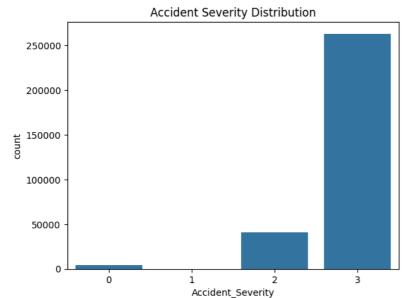
## UPLOAD AN LOAD DATA

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





## Feature Correlation Heatmap 1.0 Day\_of\_Week -1.00 0.00 0.01 0.01-0.01 0.02-0.00-0.00 0.00 -0.02 0.01 0.00 0.01 0.01-0.01 0.01 0.01 0.00 0.010.01-0.01 Junction\_Control -0.00 1.00 0.29 0.00-0.07 0.04-0.000.00 -0.030.00 0.03-0.11 0.01 0.10 0.03-0.07 0.00-0.01 0.020.00-0.00 Junction\_Detail -0.01 0.29 1.00 0.03-0.03 0.01 0.00-0.00 0.05-0.04 0.03 0.01-0.03 0.10-0.14 0.10-0.02 0.00 0.01 0.00 0.00 - 0.8 Accident\_Severity -0.01 0.00 0.03 1.00-0.02 0.03-0.01 0.00 0.00-0.08 0.07 0.00 0.01-0.01-0.08 0.08 0.03-0.00 0.02 0.01-0.00 Latitude --0.01-0.07-0.03-0.02 1.00 0.01-0.060.01 -0.37 0.04-0.03 0.08 0.08 0.00 0.06-0.05 0.04 0.01 0.11-0.00-0.00 Light\_Conditions - 0.02 0.04 0.01 0.03 0.01 1.00 0.01-0.02-0.03-0.01 0.06-0.03-0.17 0.02 0.09-0.11-0.12-0.00 - 0.6 0.01-0.06 0.00 Local\_Authority\_(District) -0.00-0.000.00-0.01-0.060.01 1.00 -0.01-0.01 0.00 0.01 0.12-0.00-0.010.04-0.05-0.00-0.01 0.01-0.00 0.00 Carriageway\_Hazards -0.00 0.00-0.000.00 0.01-0.02-0.01 1.00 0.04 0.01 0.01-0.00 0.02-0.03-0.02 0.04 0.01 0.00 0.00 0.01-0.01 - 0.4 Longitude -0.00-0.030.05 0.00-0.37-0.03-0.01-0.04 1.00-0.05 0.00 0.11-0.060.00-0.05 0.10-0.04-0.00 0.07 0.00 0.00 Number\_of\_Casualties --0.02 0.00 -0.04-0.08 0.04-0.01 0.00 0.01 -0.05 1.00 0.23 0.01 0.04-0.05 0.14-0.11 0.02-0.00 -0.000.01 0.01 Number of Vehicles -0.01 0.03 0.03 0.03 0.07-0.03 0.06 0.01 0.01 0.00 0.23 1.00-0.02-0.01-0.09 0.08-0.04-0.01-0.00 0.00 0.01 0.00 0.2 Police\_Force - 0.00-0.110.01 0.00 0.08-0.03 0.12-0.00 0.11 0.01-0.02 1.00 0.00-0.03-0.05 0.09-0.00 0.00 -0.01-0.00 0.00 0.02<mark>0.10</mark>-0.02 0.0 Road\_Type -0.01 0.10 0.10-0.01 0.00 0.02-0.01-0.03 0.00-0.05-0.09-0.03-0.01 1.00 0.34 0.09-0.01 0.00 0.00-0.01-0.00 Speed limit -0.010.03-0.14-0.080.060.090.04-0.02-0.050.140.08-0.050.10-0.341.00-0.680.05-0.00 -0.01 0.00 0.00 Urban\_or\_Rural\_Area -0.01-0.070.10 0.08-0.05-0.11-0.050.04 0.10-0.11-0.04 0.09-0.090.09-0.68 1.00-0.05 0.01 0.01-0.00 0.00 -0.2 Weather\_Conditions - 0.01 0.00-0.020.03 0.04-0.12-0.000.01 -0.04 0.02-0.01-0.00 0.60 -0.01 0.05 -0.05 1.00 -0.00 -0.01 <mark>0.07</mark> -0.02 0.01 0.00-0.00 -0.4 Hour Accident Year -0.01-0.02 0.01 0.02 -0.11 0.01 0.01 0.00 0.07 -0.00 0.00 -0.01-0.02-0.00-0.01 0.01 -0.01 0.01 1.00-0.01 0.01 Accident\_Month - 0.01 0.00 0.00 0.01-0.00-0.06-0.000.01 0.00 0.01 0.01-0.00 0.10 -0.010.00-0.00 0.07 0.00 -0.01<mark>1.00</mark>-0.03 0.01-0.03<mark>1.00</mark> Accident\_Year Accident\_Month Day of Week unction\_Control Junction\_Detail Latitude Carriageway\_Hazards Police\_Force Speed limit Weather\_Conditions ocal\_Authority\_(District) Longitude Number\_of\_Casualties Number\_of\_Vehicles Road Surface Conditions Road\_Type Urban\_or\_Rural\_Area Vehicle\_Type Accident\_Severity Light\_Conditions Accident Day

```
#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)
warnings.warn(
# STEP 5: Model Training
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from \ sklearn.ensemble \ import \ Random Forest Classifier
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 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))

/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))

Logistic Regre	ssion:			
	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

## Confusion Matrix - Random Forest 50000 0 13 805 0 -40000 0 8 30000 Actual 0 71 8029 20000 10000 0 52507 2 154 - 0 3 0 1 2 Predicted

