1. Upload the Dataset

```
from google.colab import files

uploaded = files.upload()
```

2. Load the Dataset

```
import pandas as pd

df = pd.read_csv("accident.csv")

df.head()
```

3. Data Exploration

```
df.info()

df.describe()

df.columns
```

4. Check for Missing Values and Duplicates

```
# Missing values

df.isnull().sum()


# Duplicates

df.duplicated().sum()

df = df.drop_duplicates()
```

5. Visualize a Few Features

```
import seaborn as sns
import matplotlib.pyplot as plt

# Histogram of Age
sns.histplot(df["Age"], kde=True)
plt.title("Distribution of Age")
plt.show()

# Countplot of Gender
sns.countplot(x="Gender", data=df)
plt.title("Gender Distribution")
plt.show()

# Boxplot for Speed of Impact
sns.boxplot(x=df["Speed_of_Impact"])
plt.title("Speed of Impact")
plt.show()
```

6. Identify Target and Features

```
# Features and target
X = df.drop("Survived", axis=1)
y = df["Survived"]
```

7. Convert Categorical Columns to Numerical

```
# View categorical columns
X.select_dtypes(include=['object']).columns
```

8. One-Hot Encoding

```
X = pd.get_dummies(X, drop_first=True)
```

9. Feature Scaling

```
from sklearn.preprocessing import StandardScaler

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

10. Train-Test Split

```
from sklearn.model_selection import train_test_split

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

11. Model Building

```
from sklearn.ensemble import RandomForestClassifier

model = RandomForestClassifier()
model.fit(X_train, y_train)
```

12. Evaluation

```
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

y_pred = model.predict(X_test)
print("Accuracy:", accuracy_score(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

13. Make Predictions from New Input

```
new_data = [[25, 70, 1, 1, 0]]  # Age, Speed, Helmet_Used, Seatbelt_Used, Gender_Male
new_data_scaled = scaler.transform(new_data)
model.predict(new_data_scaled)
```

14. Convert to DataFrame and Encode

```
# Define expected columns as in training
expected_columns = X.columns

# Sample input
input_dict = {
    'Age': 25,
    'Speed_of_Impact': 70,
    'Helmet_Used': 'Yes',
    'Seatbelt_Used': 'Yes',
    'Gender': 'Male'
}

# Convert to DataFrame
input_df = pd.DataFrame([input_dict])

# Map Yes/No to 1/0
input_df['Helmet_Used'] = input_df['Helmet_Used'].map({'Yes': 1, 'No': 0})
input_df['Seatbelt_Used'] = input_df['Seatbelt_Used'].map({'Yes': 1, 'No': 0})

# One-hot encode 'Gender'
input_df = pd.get_dummies(input_df)

# Add any missing columns and ensure order matches training
```

```
for col in expected_columns:
    if col not in input_df.columns:
        input_df[col] = 0
input_df = input_df[expected_columns]


# Scale and predict
input_scaled = scaler.transform(input_df)
model.predict(input_scaled)
```

15. Predict the Final Grade (Survival)

```
model.predict(input_scaled)
```

16. Deployment - Building an Interactive App

```
!pip install gradio
import gradio as gr
```

17. Create a Prediction Function

```
def predict_survival(age, speed, helmet, seatbelt, gender):
    helmet = 1 if helmet == "Yes" else 0
    seatbelt = 1 if seatbelt == "Yes" else 0
    gender_male = 1 if gender == "Male" else 0
    input_data = [[age, speed, helmet, seatbelt, gender_male]]

    input_scaled = scaler.transform(input_data)

    prediction = model.predict(input_scaled)

    return "Survived" if prediction[0] == 1 else "Did not survive"
```

18. Create the Gradio Interface

```
interface = gr.Interface(

    fn=predict_survival,

    inputs=[

        gr.Number(label="Age"),

        gr.Number(label="Speed of Impact"),

        gr.Radio(["Yes", "No"], label="Helmet Used"),

        gr.Radio(["Yes", "No"], label="Seatbelt Used"),

        gr.Radio(["Male", "Female"], label="Gender")

    ],

    outputs="text",

    title="Traffic Accident Survival Predictor"

)


interface.launch()
```