```
from google.colab import files
uploaded = files.upload()
```

Choose files accident.csv

- **accident.csv**(text/csv) - 4542 bytes, last modified: 07/05/2025 - 100% done

```
Saving accident.csv to accident.csv
```

```
import pandas as pd

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

	Age	Gender	Speed_of_Impact	Helmet_Used	Seatbelt_Used	Survived
0	56	Female	27.0	No	No	1
1	69	Female	46.0	No	Yes	1
2	46	Male	46.0	Yes	Yes	0
3	32	Male	117.0	No	Yes	0
4	60	Female	40.0	Yes	Yes	0

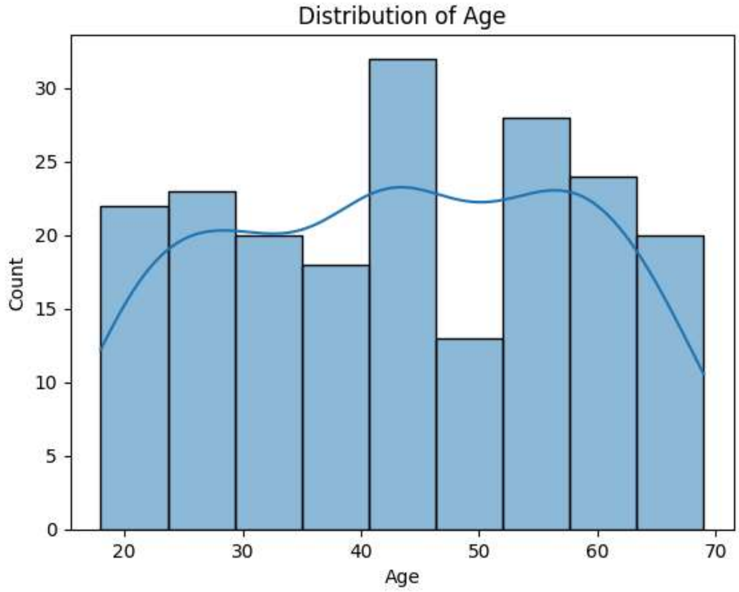
Next steps: Generate code with df View recommended plots New interactive sheet

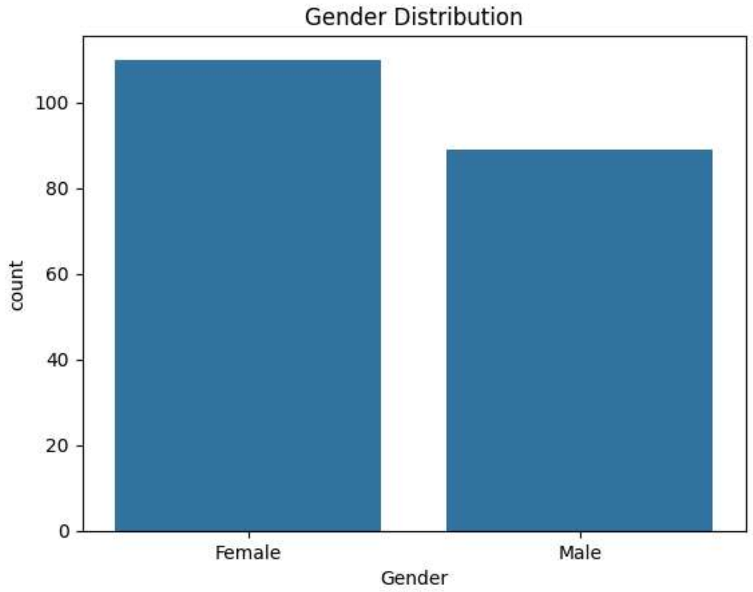
```
df.info()
df.describe()
df.columns
```

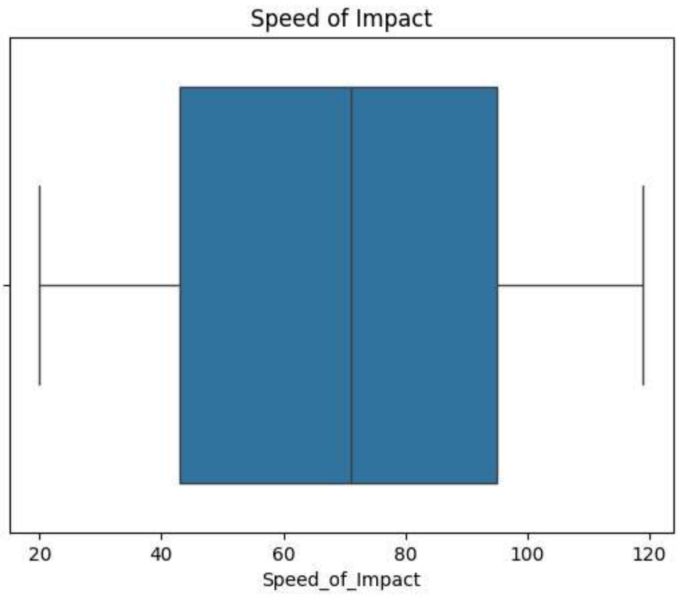
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 6 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   Age              200 non-null    int64  
 1   Gender           199 non-null    object 
 2   Speed_of_Impact  197 non-null    float64
 3   Helmet_Used      200 non-null    object 
 4   Seatbelt_Used    200 non-null    object 
 5   Survived         200 non-null    int64  
dtypes: float64(1), int64(2), object(3)
memory usage: 9.5+ KB
Index(['Age', 'Gender', 'Speed_of_Impact', 'Helmet_Used', 'Seatbelt_Used',
       'Survived'],
      dtype='object')
```

```
# Missing values
df.isnull().sum()

# Duplicates
df.duplicated().sum()
df = df.drop_duplicates()
```

```
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()
```







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

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

```
Index(['Gender', 'Helmet_Used', 'Seatbelt_Used'], dtype='object')
```

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

```
from sklearn.preprocessing import StandardScaler

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

```
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)
```

```
from sklearn.ensemble import RandomForestClassifier

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

```
▾ RandomForestClassifier ⓘ ?
RandomForestClassifier()
```

```
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))
```

```
Accuracy: 0.525
[[12 10]
 [ 9  9]]
              precision    recall  f1-score   support

           0       0.57      0.55      0.56        22
           1       0.47      0.50      0.49        18

    accuracy                           0.53        40
   macro avg       0.52      0.52      0.52        40
weighted avg       0.53      0.53      0.53        40
```

```
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)
```

```
/usr/local/lib/python3.11/dist-packages/sklearn/utils/validation.py:2739: UserWarning: X does not have valid feature names, but Star
  warnings.warn(
array([1])
```

```
# 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)
```

```
array([1])
```

```
model.predict(input_df_scaled)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-17-17a0d2f90d31> in <cell line: 0>()
----> 1 model.predict(input_df_scaled)

NameError: name 'input_df_scaled' is not defined
```

Next steps: Explain error

```
# Make the prediction (Topic 15)
model.predict(input_scaled)
```

```
array([1])
```

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

```
array([1])
```

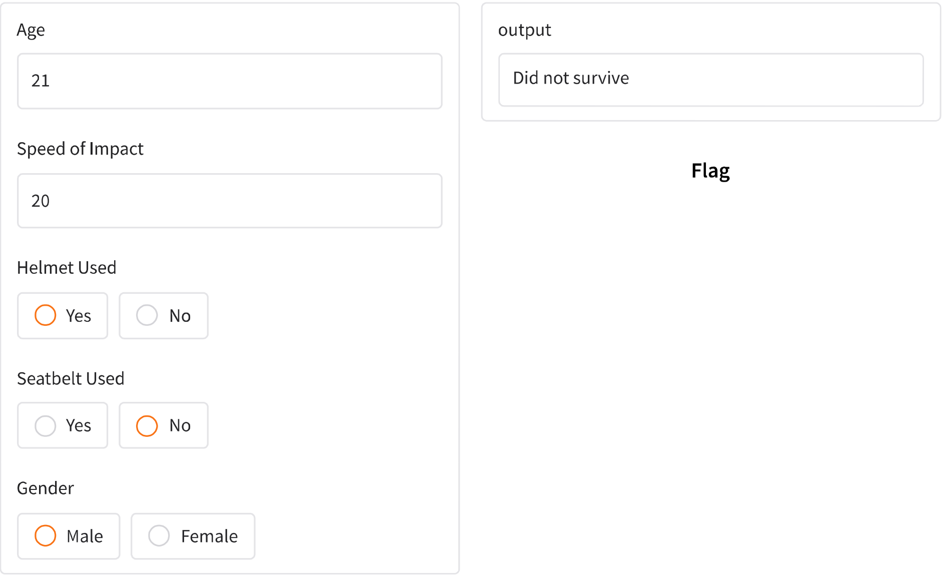
```
!pip install gradio
import gradio as gr
```

```
Collecting semantic-version~=2.0 (from gradio)
  Downloading semantic_version-2.10.0-py2.py3-none-any.whl.metadata (9.7 kB)
Collecting starlette<1.0,>=0.40.0 (from gradio)
  Downloading starlette-0.46.2-py3-none-any.whl.metadata (6.2 kB)
Collecting tomlkit<0.14.0,>=0.12.0 (from gradio)
  Downloading tomlkit-0.13.2-py3-none-any.whl.metadata (2.7 kB)
```

```
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 322.9/322.9 kB 25.0 MB/s eta 0:00:00
Downloading aiofiles-24.1.0-py3-none-any.whl (15 kB)
Downloading fastapi-0.115.12-py3-none-any.whl (95 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 95.2/95.2 kB 7.5 MB/s eta 0:00:00
Downloading groovy-0.1.2-py3-none-any.whl (14 kB)
Downloading python_multipart-0.0.20-py3-none-any.whl (24 kB)
Downloading ruff-0.11.8-py3-none-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (11.5 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 11.5/11.5 MB 111.1 MB/s eta 0:00:00
Downloading safehttpx-0.1.6-py3-none-any.whl (8.7 kB)
Downloading semantic_version-2.10.0-py2.py3-none-any.whl (15 kB)
Downloading starlette-0.46.2-py3-none-any.whl (72 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 72.0/72.0 kB 6.3 MB/s eta 0:00:00
Downloading tomlkit-0.13.2-py3-none-any.whl (37 kB)
Downloading uvicorn-0.34.2-py3-none-any.whl (62 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 62.5/62.5 kB 5.2 MB/s eta 0:00:00
Downloading ffmpy-0.5.0-py3-none-any.whl (6.0 kB)
Downloading pydub-0.25.1-py2.py3-none-any.whl (32 kB)
Installing collected packages: pydub, uvicorn, tomlkit, semantic-version, ruff, python-multipart, groovy, ffmpy, aiofiles, starlet
Successfully installed aiofiles-24.1.0 fastapi-0.115.12 ffmpy-0.5.0 gradio-5.29.0 gradio-client-1.10.0 groovy-0.1.2 pydub-0.25.1
```

```python
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"



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()
```

```
It looks like you are running Gradio on a hosted a Jupyter notebook. For the Gradio app to work, sharing must be enabled. Automatica

Colab notebook detected. To show errors in colab notebook, set debug=True in launch()
* Running on public URL: https://df9036f30e7f1369d0.gradio.live

This share link expires in 1 week. For free permanent hosting and GPU upgrades, run `gradio deploy` from the terminal in the workin
```



Age

21

Speed of Impact

20

Helmet Used

● Yes ○ No

Seatbelt Used

○ Yes ● No

Gender

● Male ○ Female

output

Did not survive