```
from google.colab import files
import pandas as pd

# Upload tested.csv manually
uploaded = files.upload()

# Load the uploaded file into dataframe
df = pd.read_csv("tested.csv")
print("Shape:", df.shape)
df.head()
```

Choose files tested.csv

- **tested.csv**(text/csv) - 29474 bytes, last modified: 25/08/2025 - 100% done

```
Saving tested.csv to tested.csv
Shape: (418, 12)
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	892	0	3	Kelly, Mr. James	male	34.5	0	0	330911	7.8292	NaN	Q
1	893	1	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	1	0	363272	7.0000	NaN	S
2	894	0	2	Myles, Mr. Thomas Francis	male	62.0	0	0	240276	9.6875	NaN	Q
3	895	0	3	Wirz, Mr. Albert	male	27.0	0	0	315154	8.6625	NaN	S
4	896	1	3	Hirvonen, Mrs. Alexander (Helga E	female	22.0	1	1	3101298	12.2875	NaN	S

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

```
print(df.columns.tolist())
```

```
['PassengerId', 'Survived', 'Pclass', 'Name', 'Sex', 'Age', 'SibSp', 'Parch', 'Ticket', 'Fare', 'Cabin', 'Embarked']
```

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder

data = df.copy()

# Drop unnecessary columns if present
for col in ['PassengerId','Ticket','Cabin','Name']:
    if col in data.columns:
        data.drop(col, axis=1, inplace=True)

# Fill missing values
for col in data.select_dtypes(include=['float64','int64']).columns:
    data[col].fillna(data[col].median(), inplace=True)

for col in data.select_dtypes(include=['object']).columns:
    data[col].fillna(data[col].mode()[0], inplace=True)

# Encode categorical columns
label_enc = LabelEncoder()
for col in data.select_dtypes(include=['object']).columns:
    data[col] = label_enc.fit_transform(data[col])

# Split into features & target
X = data.drop("Survived", axis=1)
y = data["Survived"]

X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)

print("✅ Data ready for modeling")
```

```
✅ Data ready for modeling
/tmp/ipython-input-3824266200.py:13: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained a
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col


  data[col].fillna(data[col].median(), inplace=True)
/tmp/ipython-input-3824266200.py:16: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained a
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col


  data[col].fillna(data[col].mode()[0], inplace=True)
```

```
from sklearn.ensemble import RandomForestClassifier

model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

print("✅ Model trained")
```

```
✅ Model trained
```

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

y_pred = model.predict(X_val)

print("Accuracy:", accuracy_score(y_val, y_pred))
print("\nClassification Report:\n", classification_report(y_val, y_pred))
print("\nConfusion Matrix:\n", confusion_matrix(y_val, y_pred))
```

```
Accuracy: 1.0

Classification Report:
               precision    recall  f1-score   support

           0       1.00      1.00      1.00        53
           1       1.00      1.00      1.00        31

    accuracy                           1.00        84
   macro avg       1.00      1.00      1.00        84
weighted avg       1.00      1.00      1.00        84


Confusion Matrix:
 [[53  0]
 [ 0 31]]
```