```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sb
from google.colab import files
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from xgboost import XGBClassifier
from sklearn import metrics

import warnings
warnings.filterwarnings('ignore')
```

```
uploaded= files.upload()
```

Choose Files titanic.csv

titanic.csv(text/csv) - 29474 bytes, last modified: 9/23/2025 - 100% done

Saving titanic.csv to titanic (2).csv

```
df=pd.read_csv('titanic.csv')
```

```
df.shape
```

```
(418, 12)
```

```
df.isnull().sum()
```

	0
PassengerId	0
Survived	0
Pclass	0
Name	0
Sex	0
Age	86
SibSp	0
Parch	0
Ticket	0
Fare	1
Cabin	327
Embarked	0

dtype: int64

```
df.head()
```

	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 New interactive sheet

```
mode = df['Age'].mode().values[0]
df['Age'] = df['Age'].replace(np.nan,mode)
df.isnull().sum()
```

	0
PassengerId	0
Survived	0
Pclass	0
Name	0
Sex	0
Age	0
SibSp	0
Parch	0
Ticket	0
Fare	1
Cabin	327
Embarked	0

dtype: int64

```
mode = df['Cabin'].mode().values[0]
df['Cabin'] = df['Cabin'].replace(np.nan,mode)
df.isnull().sum()
```

	0
PassengerId	0
Survived	0
Pclass	0
Name	0
Sex	0
Age	0
SibSp	0
Parch	0
Ticket	0
Fare	1
Cabin	0
Embarked	0

dtype: int64

```
df = df.dropna(axis=0,how='any')
print(df.isnull().sum())
df.shape
```

```
PassengerId    0
Survived       0
Pclass         0
Name           0
Sex            0
Age            0
SibSp          0
Parch          0
Ticket         0
Fare           0
Cabin          0
Embarked       0
dtype: int64
(417, 12)
```

```
df = df.rename(columns = {"PassengerId":"Passenger ID","Pclass":"Passenger Class","Sex":"Gender"})
```

```
df.head()
```

	Passenger ID	Survived	Passenger Class	Name	Gender	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	892	0	3	Kelly, Mr. James	male	34.5	0	0	330911	7.8292	B57 B59 B63 B66	Q
1	893	1	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	1	0	363272	7.0000	B57 B59 B63 B66	S
2	894	0	2	Myles, Mr. Thomas Francis	male	62.0	0	0	240276	9.6875	B57 B59 B63 B66	Q
3	895	0	3	Wirz, Mr. Albert	male	27.0	0	0	315154	8.6625	B57 B59 B63 B66	S
4	896	1	3	Hirvonen, Mrs. Alexander (Helga E	female	22.0	1	1	3101298	12.2875	B57 B59 B63 B66	S

Next steps: Generate code with df | New interactive sheet

```
df = df.drop(columns = ["SibSp","Parch"])
```

```
df.head()
```

	Passenger ID	Survived	Passenger Class	Name	Gender	Age	Ticket	Fare	Cabin	Embarked
0	892	0	3	Kelly, Mr. James	male	34.5	330911	7.8292	B57 B59 B63 B66	Q
1	893	1	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	363272	7.0000	B57 B59 B63 B66	S
2	894	0	2	Myles, Mr. Thomas Francis	male	62.0	240276	9.6875	B57 B59 B63 B66	Q
3	895	0	3	Wirz, Mr. Albert	male	27.0	315154	8.6625	B57 B59 B63 B66	S
4	896	1	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	22.0	3101298	12.2875	B57 B59 B63 B66	S

Next steps: Generate code with df | New interactive sheet

```
df['Embarked'] = df['Embarked'].replace({'C': 'Cherbourg','Q': 'Queenstown','S': 'Southampton'})
```

```
df.head()
```

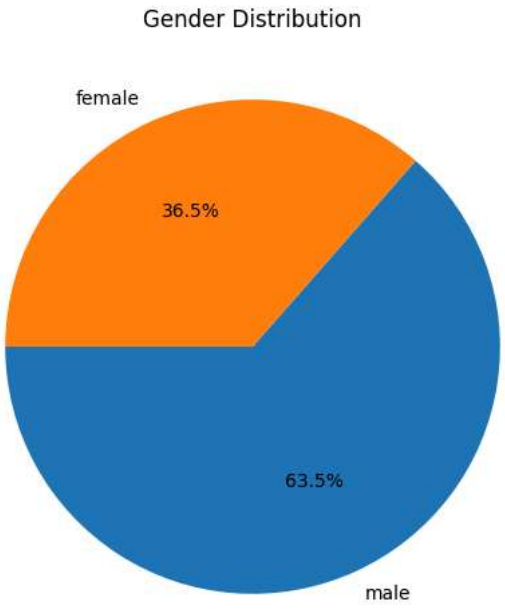
	Passenger ID	Survived	Passenger Class	Name	Gender	Age	Ticket	Fare	Cabin	Embarked
0	892	0	3	Kelly, Mr. James	male	34.5	330911	7.8292	B57 B59 B63 B66	Queenstown
1	893	1	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	363272	7.0000	B57 B59 B63 B66	Southampton
2	894	0	2	Myles, Mr. Thomas Francis	male	62.0	240276	9.6875	B57 B59 B63 B66	Queenstown
3	895	0	3	Wirz, Mr. Albert	male	27.0	315154	8.6625	B57 B59 B63 B66	Southampton
4	896	1	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	22.0	3101298	12.2875	B57 B59 B63 B66	Southampton

Next steps: Generate code with df | New interactive sheet

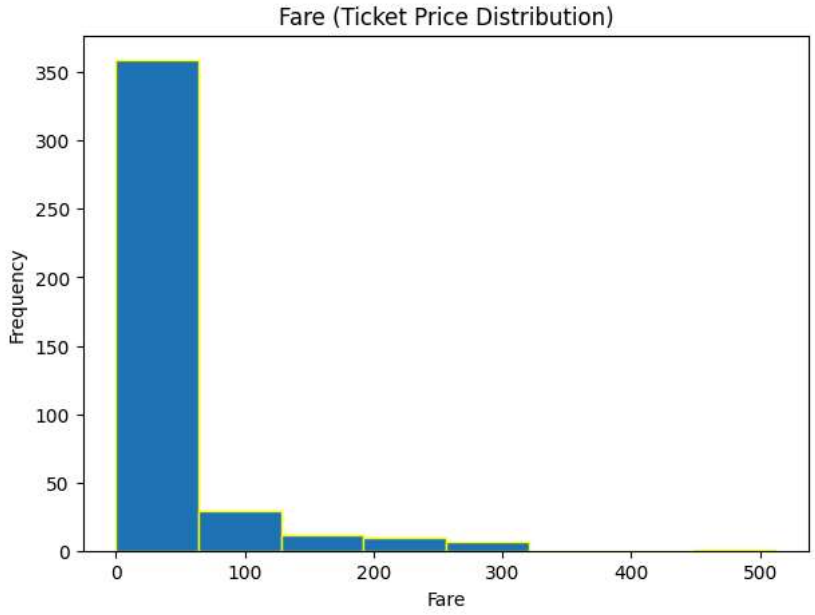
```
df.to_csv("clean_titanic.csv", index=False)
```

```
from google.colab import files
files.download("clean_titanic.csv")
```

```
plt.figure(figsize=(6,6))
df["Gender"].value_counts().plot(kind="pie",autopct="%1.1f%%",startangle=180)
plt.title("Gender Distribution")
plt.ylabel("")
plt.show()
```




```
plt.figure(figsize=(7,5))
df["Fare"].plot(kind="hist",bins=8,edgecolor="yellow")
plt.title("Fare (Ticket Price Distribution)")
plt.xlabel("Fare")
plt.show()
```




```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix, classification_report
```

```
import ipywidgets as widgets
from IPython.display import display
```

```
y=df["Survived"].astype(bool)
X=df[["Passenger ID","Passenger Class","Name","Gender","Age","Ticket","Fare","Cabin","Embarked"]]
```

```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.3,random_state=42,stratify=y)
```

```
X_train_enc = pd.get_dummies(X_train)
X_test_enc  = pd.get_dummies(X_test).reindex(columns=X_train_enc.columns, fill_value=0)
clf = DecisionTreeClassifier(max_depth=4, random_state=42)
clf.fit(X_train_enc, y_train)
```

```
DecisionTreeClassifier
DecisionTreeClassifier(max_depth=4, random_state=42)
```

```
y_pred = clf.predict(X_test_enc)
print("Accuracy :", accuracy_score(y_test, y_pred))
print("Precision:", precision_score(y_test, y_pred, zero_division=0))
print("Recall   :", recall_score(y_test, y_pred, zero_division=0))
print("F1-score :", f1_score(y_test, y_pred, zero_division=0))
print("\nClassification Report:\n", classification_report(y_test, y_pred, zero_division=0))
```

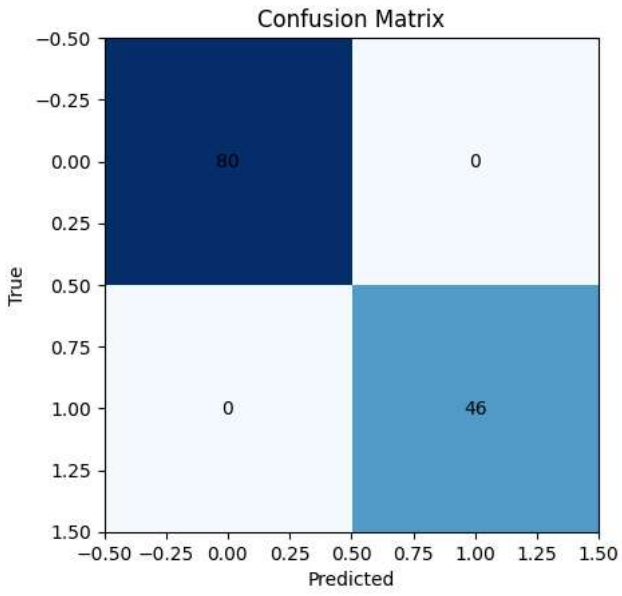
```
Accuracy : 1.0
Precision: 1.0
Recall   : 1.0
F1-score : 1.0

Classification Report:
              precision    recall  f1-score   support

       False       1.00      1.00      1.00        80
        True       1.00      1.00      1.00        46

    accuracy                           1.00       126
   macro avg       1.00      1.00      1.00       126
weighted avg       1.00      1.00      1.00       126
```

```
cm = confusion_matrix(y_test, y_pred)
plt.imshow(cm, cmap="Blues")
plt.title("Confusion Matrix")
plt.xlabel("Predicted"); plt.ylabel("True")
for i in range(cm.shape[0]):
    for j in range(cm.shape[1]):
        plt.text(j, i, cm[i, j], ha="center", va="center")
plt.show()
```




```
import warnings
```

```
import warnings
warnings.filterwarnings("ignore")

import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline
from sklearn.metrics import accuracy_score, classification_report
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression

# Optional XGBoost
try:
    import xgboost as xgb
    HAS_XGB = True
except Exception:
    HAS_XGB = False

import ipywidgets as widgets
from IPython.display import display, clear_output

# --- Load your uploaded file ---
file_path = "/content/clean_titanic.csv"
df = pd.read_csv(file_path)

# --- Pre-processing ---
df['Age'].fillna(df['Age'].median(), inplace=True)
df['Embarked'].fillna(df['Embarked'].mode()[0], inplace=True)

# Encode Gender and Embarked
df['Gender_male'] = (df['Gender'].str.lower() == 'male').astype(int)

feature_cols = ['Passenger Class', 'Age', 'Fare', 'Gender_male'] + \
               [c for c in df.columns if c.startswith('Embarked_')]

X = df[feature_cols]
y = df['Survived']

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

# --- Models ---
models = {
    "Decision Tree": Pipeline([("clf", DecisionTreeClassifier(random_state=42))]),
    "Random Forest": Pipeline([("clf", RandomForestClassifier(n_estimators=200, random_state=42))]),
    "Gradient Boosting": Pipeline([("clf", GradientBoostingClassifier(random_state=42))]),
    "SVM (RBF)": Pipeline([("scaler", StandardScaler()), ("clf", SVC(kernel="rbf", probability=True, random_state=42))]),
    "Logistic Regression": Pipeline([("scaler", StandardScaler()), ("clf", LogisticRegression(max_iter=1000, random_state=42))]),
}
if HAS_XGB:
    models["XGBoost"] = Pipeline([("clf", xgb.XGBClassifier(n_estimators=200, random_state=42, eval_metric="logloss"))])

# --- Widgets ---
model_dd  = widgets.Dropdown(options=list(models.keys()), description="Model:")
train_btn = widgets.Button(description="Train", button_style="primary")

passengerclass_in = widgets.Dropdown(options=[1,2,3], description="Class:", value=3)
age_in    = widgets.FloatText(description="Age:", value=30)
fare_in   = widgets.FloatText(description="Fare:", value=30.0)
gender_in    = widgets.Dropdown(options=["male","female"], description="Gender:")
embarked_in = widgets.Dropdown(options=sorted(df['Embarked'].unique()), description="Embarked:")
pred_btn  = widgets.Button(description="Predict")

out = widgets.Output(layout={'border': '1px solid #ddd'})
fitted = {"pipe": None}

def on_train_clicked(_):
    with out:
        clear_output(wait=True)
        name = model_dd.value
        pipe = models[name]
        pipe.fit(X_train, y_train)
        y_pred = pipe.predict(X_test)
        acc = accuracy_score(y_test, y_pred)
        rep = classification_report(y_test, y_pred, digits=3)
```

```
        fitted["pipe"] = pipe
        print(f"✅ Trained: {name}")
        print(f"Test Accuracy: {acc:.3f}\n")
        print(rep)

def on_predict_clicked(_):
    with out:
        if fitted["pipe"] is None:
            print("Please click Train first.")
            return
        row = pd.DataFrame([{
            'Passenger Class': passengerclass_in.value,
            'Age': age_in.value,
            'Fare': fare_in.value,
            'Gender_male': 1 if gender_in.value == "male" else 0,
            **{f"Embarked_{e}": 1 if embark_in.value == e else 0
               for e in df.columns if e.startswith("Embarked_")}
        }])
        pred = fitted["pipe"].predict(row)[0]
        try:
            prob = fitted["pipe"].predict_proba(row)[0][1]
        except Exception:
            prob = None
        print("Passenger:", dict(row.iloc[0]))
        print("Prediction:", "Survived ✅" if pred==1 else "Did NOT Survive ❌")
        if prob is not None:
            print(f"Survival probability: {prob:.2%}")

train_btn.on_click(on_train_clicked)
pred_btn.on_click(on_predict_clicked)

ui = widgets.VBox([
    widgets.HBox([model_dd, train_btn]),
    widgets.HTML("<b>Enter Passenger Details</b>"),
    widgets.HBox([passengerclass_in, age_in, fare_in, gender_in, embarked_in]),
    pred_btn,
    out])
display(ui)
```

Model: Decision Tree Train

Enter Passenger Details

Class: 3 Age: 47 Fare: 7 Gender: male Embarked: Southamptor

Predict

```
✅ Trained: Decision Tree
Test Accuracy: 1.000

              precision    recall  f1-score   support

           0      1.000     1.000     1.000        50
           1      1.000     1.000     1.000        34

    accuracy                          1.000        84
   macro avg      1.000     1.000     1.000        84
weighted avg      1.000     1.000     1.000        84

Passenger: {'Passenger Class': np.float64(3.0), 'Age': np.float64(47.0), 'Fare': np.float64(7.0), 'Gender_male': np.float64(0.0)}
Prediction: Survived ✅
Survival probability: 100.00%
Passenger: {'Passenger Class': np.float64(3.0), 'Age': np.float64(47.0), 'Fare': np.float64(7.0), 'Gender_male': np.float64(1.0)}
Prediction: Did NOT Survive ❌
Survival probability: 0.00%
```

```
df.groupby('Gender')['Survived'].mean()
```

Gender	Survived
female	1.0
male	0.0

dtype: float64