```
# Install required packages
!pip install pandas scikit-learn seaborn matplotlib

# Import libraries
import pandas as pd
from seaborn import load_dataset
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score

# Load Titanic dataset
df = load_dataset('titanic')
df = df[['survived', 'pclass', 'sex', 'age']].dropna()
df['sex'] = df['sex'].map({'male': 0, 'female': 1})

# Prepare features and labels
X = df[['pclass', 'sex', 'age']]
y = df['survived']

# Split data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, ran

# Train model
model = DecisionTreeClassifier()
model.fit(X_train, y_train)

# Predict and evaluate
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"Prediction Accuracy: {accuracy:.2f}")
```

```
Requirement already satisfied: pandas in /usr/local/lib/python3.11/dist
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.1
Requirement already satisfied: seaborn in /usr/local/lib/python3.11/dis
Requirement already satisfied: matplotlib in /usr/local/lib/python3.11/
Requirement already satisfied: numpy>=1.23.2 in /usr/local/lib/python3.
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.1
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3
Requirement already satisfied: scipy>=1.6.0 in /usr/local/lib/python3.1
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/p
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/pytho
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.1
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/pyth
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/pyth
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python
Requirement already satisfied: pillow>=8 in /usr/local/lib/python3.11/d
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/pytho
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/di
Prediction Accuracy: 0.78
```

Titanic Survival Prediction using Data Mining

This project uses real Titanic passenger data to predict which passengers were likely to survive. It’s a beginner-friendly data mining project using Python, Pandas, and Scikit-learn. Use machine learning to find patterns in the Titanic dataset and build a simple model that can predict survival based on features like age, gender, and passenger class. We use the Titanic dataset from the Seaborn library. It includes passenger data such as:

- Survival status
- Age
- Gender
- Passenger class

We'll use these to train a decision tree model.

We clean the dataset by:

- Removing missing values
- Converting categorical data (like "sex") to numeric

- Selecting only the most useful features for prediction

We use a Decision Tree Classifier from Scikit-learn. This algorithm splits the data into branches based on the most informative questions to predict survival. After training the model and testing it, we achieved an accuracy of about 77%. This means our model predicts survival correctly about 3 out of 4 times.

This project shows how simple data mining and machine learning techniques can be applied to real-world problems. With more features and deeper models, we could improve accuracy further.

Install required packages

```
!pip install pandas scikit-learn seaborn matplotlib
```

Import libraries

```
import pandas as pd from seaborn import load_dataset from sklearn.model_selection import train_test_split from sklearn.tree import DecisionTreeClassifier from sklearn.metrics import accuracy_score
```

Load Titanic dataset

```
df = load_dataset('titanic') df = df[['survived', 'pclass', 'sex', 'age']].dropna() df['sex'] = df['sex'].map({'male': 0, 'female': 1})
```

Prepare features and labels

```
X = df[['pclass', 'sex', 'age']] y = df['survived']
```

Split data

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

Train model

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

Predict and evaluate

```
y_pred = model.predict(X_test) accuracy = accuracy_score(y_test, y_pred)
print(f"Prediction Accuracy: {accuracy:.2f}")
```