

'''

Let's create a classification model using the Titanic dataset as an example.
We will use the following libraries: pandas for data manipulation,
seaborn and matplotlib for visualization, and scikit-learn (sklearn)
for building and evaluating the machine learning model.

Here's a step-by-step guide:

Load the dataset
Explore the data
Preprocess the data
Build and evaluate the model

Step 1: Load the Dataset

We'll use the Titanic dataset from Seaborn's built-in datasets for simplicity.

'''

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
```

```
# Load the Titanic dataset
titanic = sns.load_dataset('titanic')
```

```
# Display the first few rows of the dataset
print(titanic.head())
```

```
'''Step 2: Explore the Data
We'll explore the dataset to understand '
its structure and the relationships between features.'''
```

```
# Display summary statistics
print(titanic.describe(include='all'))
```

```
# Check for missing values
print(titanic.isnull().sum())
```

```
# Visualize the distribution of survival
sns.countplot(x='survived', data=titanic)
plt.title('Survival Distribution')
plt.show()
```

```
# Visualize the relationship between features and survival
sns.catplot(x='sex', hue='survived', kind='count', data=titanic)
sns.catplot(x='class', hue='survived', kind='count', data=titanic)
plt.show()
```

'''Step 3: Preprocess the Data

We'll handle missing values, encode categorical variables, and ' split the data into training and testing sets.'''

Handle missing values

titanic['age'].fillna(titanic['age'].median(), inplace=True)

titanic['embarked'].fillna(titanic['embarked'].mode()[0], inplace=True)

Drop columns that won't be used in the model

titanic.drop(columns=['deck', 'embark_town', 'alive'], inplace=True)

Convert categorical variables to numerical using one-hot encoding

titanic = pd.get_dummies(titanic, columns=['sex', 'embarked', 'class', 'who', 'adult_male', 'alone'], drop_first=True)

Define features (X) and target (y)

X = titanic.drop(columns='survived')

y = titanic['survived']

Split the data into training and testing sets

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

Scale the features

scaler = StandardScaler()

X_train_scaled = scaler.fit_transform(X_train)

X_test_scaled = scaler.transform(X_test)

''' Step 4: Build and Evaluate the Model

We'll use a Logistic Regression model for this classification task.'''

Create and train the Logistic Regression model

model = LogisticRegression()

model.fit(X_train_scaled, y_train)

Make predictions

y_pred = model.predict(X_test_scaled)

Evaluate the model

accuracy = accuracy_score(y_test, y_pred)

conf_matrix = confusion_matrix(y_test, y_pred)

class_report = classification_report(y_test, y_pred)

print(f'Accuracy: {accuracy}')

print(f'Confusion Matrix:\n{conf_matrix}')

print(f'Classification Report:\n{class_report}')

Plot the confusion matrix


sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues')

plt.title('Confusion Matrix')

plt.xlabel('Predicted')

plt.ylabel('Actual')

```
plt.show()
```

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	\
0	0	3	male	22.0	1	0	7.2500	S	Third	
1	1	1	female	38.0	1	0	71.2833	C	First	
2	1	3	female	26.0	0	0	7.9250	S	Third	
3	1	1	female	35.0	1	0	53.1000	S	First	
4	0	3	male	35.0	0	0	8.0500	S	Third	

	who	adult_male	deck	embark_town	alive	alone
0	man	True	NaN	Southampton	no	False
1	woman	False	C	Cherbourg	yes	False
2	woman	False	NaN	Southampton	yes	True
3	woman	False	C	Southampton	yes	False
4	man	True	NaN	Southampton	no	True

	survived	pclass	sex	age	sibsp	parch	\
count	891.000000	891.000000	891	714.000000	891.000000	891.000000	
unique	NaN	NaN	2	NaN	NaN	NaN	
top	NaN	NaN	male	NaN	NaN	NaN	
freq	NaN	NaN	577	NaN	NaN	NaN	
mean	0.383838	2.308642	NaN	29.699118	0.523008	0.381594	
std	0.486592	0.836071	NaN	14.526497	1.102743	0.806057	
min	0.000000	1.000000	NaN	0.420000	0.000000	0.000000	
25%	0.000000	2.000000	NaN	20.125000	0.000000	0.000000	
50%	0.000000	3.000000	NaN	28.000000	0.000000	0.000000	
75%	1.000000	3.000000	NaN	38.000000	1.000000	0.000000	
max	1.000000	3.000000	NaN	80.000000	8.000000	6.000000	

	fare	embarked	class	who	adult_male	deck	embark_town	alive	\
count	891.000000	889	891	891	891	203	889	891	
unique	NaN	3	3	3	2	7	3	2	
top	NaN	S	Third	man	True	C	Southampton	no	
freq	NaN	644	491	537	537	59	644	549	
mean	32.204208	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
std	49.693429	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
min	0.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
25%	7.910400	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
50%	14.454200	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
75%	31.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
max	512.329200	NaN	NaN	NaN	NaN	NaN	NaN	NaN	

	alone
count	891
unique	2
top	True
freq	537
mean	NaN
std	NaN
min	NaN
25%	NaN
50%	NaN
75%	NaN
max	NaN
survived	0
pclass	0
sex	0
age	177
sibsp	0
parch	0
fare	0
embarked	2
class	0
who	0
adult_male	0
deck	688
embark_town	2
alive	0
alone	0
dtype:	int64

