Exercise: 1

Dataset:

Consider a dataset with the following information:

Bedrooms	House Price (in thousands of dollars)
1	100
2	150
3	200
4	250
5	300

Objective:

Build a simple linear regression model to predict house prices based on the number of bedrooms.

Steps:

Load the data.

Visualize the data using a scatter plot.

Split the data into training and testing sets.

Train a linear regression model.

Make predictions on the test set.

Evaluate the model's performance.

HOMEWORK QUESTIONS:

1. Load the data into a Pandas DataFrame. Provide the code to accomplish this.

2. Plot the data using a scatter plot. What can you infer from the plot?

3. Split the data into training and testing sets. Provide the code to achieve this.

4. Train a linear regression model using the training set. Provide the code for model training.

5. Make predictions on the test set. Provide the code for making predictions.

6. Evaluate the model's performance. Calculate and print the mean squared error.

1. Load the data into a Pandas DataFrame. Provide the code to accomplish this.

Answer:

python

```
import pandas as pd

data = {'Bedrooms': [1, 2, 3, 4, 5],

        'House_Price': [100, 150, 200, 250, 300]}

df = pd.DataFrame(data)
```

2. Plot the data using a scatter plot. What can you infer from the plot?

Answer:

python

```
import matplotlib.pyplot as plt

plt.scatter(df['Bedrooms'], df['House_Price'])

plt.title('House Price vs. Bedrooms')

plt.xlabel('Number of Bedrooms')

plt.ylabel('House Price (in thousands)')

plt.show()
```

From the plot, we can observe a positive correlation between the number of bedrooms and house prices.

3. Split the data into training and testing sets. Provide the code to achieve this.

Answer:

python

```
from sklearn.model_selection import train_test_split

X = df[['Bedrooms']]

y = df['House_Price']

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

4. Train a linear regression model using the training set. Provide the code for model training.

Answer:

python

```
from sklearn.linear_model import LinearRegression

model = LinearRegression()

model.fit(X_train, y_train)
```

5. Make predictions on the test set. Provide the code for making predictions.

Answer:

python

```
y_pred = model.predict(X_test)
```

6. Evaluate the model's performance. Calculate and print the mean squared error.

Answer:

python

```
from sklearn.metrics import mean_squared_error

mse = mean_squared_error(y_test, y_pred)

print(f'Mean Squared Error: {mse}')
```

These questions and exercises cover the fundamental steps in a supervised learning task using a simple linear regression model

If you want to work with a dataset for the given exercise, you can create a CSV file or use the provided data directly. Here's how you can use a CSV file:

Step 1: Create a CSV File

Save the following data in a CSV file named "house_prices.csv":

csv

```
Bedrooms,House_Price

1,100
```

2,150

3,200

4,250

5,300

Step 2: Load the Dataset in Python

```
python

import pandas as pd

# Load the dataset from the CSV file

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

Now, we can proceed with the rest of the steps in the exercise using this loaded dataset.

Exercise: 2

Dataset:

Consider a dataset with the following information:

Hours Studied	Result
2	Fail
3	Fail
4	Pass
5	Pass
6	Pass
7	Pass
8	Pass

Objective:

Build a simple logistic regression model to predict the outcome (Pass/Fail) based on the number of hours studied.

Steps:

Load the data.

Visualize the data using a scatter plot or other appropriate visualization.

Split the data into training and testing sets.

Train a logistic regression model.

Make predictions on the test set.

Evaluate the model's performance.

HOMEWORK QUESTIONS:

1. Load the data into a Pandas DataFrame. Provide the code to accomplish this.

2. Plot the data using a scatter plot or other appropriate visualization. What insights can you draw from the plot?

3. Split the data into training and testing sets. Provide the code for this step.

4. Train a logistic regression model using the training set. Provide the code for model training.

5. Make predictions on the test set. Provide the code for making predictions.

6. Evaluate the model's performance. Calculate and print accuracy, precision, recall, and F1-score.

1. Load the data into a Pandas DataFrame. Provide the code to accomplish this.

Answer:

python

```
import pandas as pd

data = {'Hours_Studied': [2, 3, 4, 5, 6, 7, 8],

        'Result': ['Fail', 'Fail', 'Pass', 'Pass', 'Pass', 'Pass', 'Pass']}

df = pd.DataFrame(data)
```

2. Plot the data using a scatter plot or other appropriate visualization. What insights can you draw from the plot?

Answer:

python

```
import matplotlib.pyplot as plt

import seaborn as sns

sns.countplot(x='Result', data=df)

plt.title('Number of Students Passing and Failing')

plt.xlabel('Result')

plt.ylabel('Count')

plt.show()
```

The plot shows the distribution of students passing and failing based on the hours studied.

3. Split the data into training and testing sets. Provide the code for this step.

Answer:

python

```
from sklearn.model_selection import train_test_split

X = df[['Hours_Studied']]

y = df['Result']

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

4. Train a logistic regression model using the training set. Provide the code for model training.

Answer:

python

```
from sklearn.linear_model import LogisticRegression

model = LogisticRegression()

model.fit(X_train, y_train)
```

5. Make predictions on the test set. Provide the code for making predictions.

Answer:

python

```
y_pred = model.predict(X_test)
```

6. Evaluate the model's performance. Calculate and print accuracy, precision, recall, and F1-score.

Answer:

python

```
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

accuracy = accuracy_score(y_test, y_pred)

precision = precision_score(y_test, y_pred, pos_label='Pass')

recall = recall_score(y_test, y_pred, pos_label='Pass')

f1 = f1_score(y_test, y_pred, pos_label='Pass')

print(f'Accuracy: {accuracy:.2f}')

print(f'Precision: {precision:.2f}')

print(f'Recall: {recall:.2f}')

print(f'F1 Score: {f1:.2f}')
```

These questions and exercises cover the basics of a supervised learning classification task using a logistic regression model. They allow students to practice loading data, visualizing it, and evaluating the performance of a machine learning model.

Here's a simple CSV dataset for the mentioned supervised learning exercise:

Dataset: "student_results.csv"

csv

```
Hours_Studied,Result

2,Fail

3,Fail

4,Pass

5,Pass

6,Pass

7,Pass

8,Pass
```

We can use this dataset by saving it as a CSV file and then loading it into your Python environment using a library like Pandas. For example:

```python
import pandas as pd

# Load the dataset

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

Let's consider another supervised learning exercise involving a classification task. This time, we'll use the famous Iris dataset and build a model to predict the species of iris flowers based on their features.

Exercise: 3

Dataset: Iris Dataset

The Iris dataset is a well-known dataset that contains the sepal length, sepal width, petal length, and petal width of three species of iris flowers: setosa, versicolor, and virginica.

Objective:

Build a simple classification model to predict the species of iris flowers based on their features.

Steps:

Load the Iris dataset.

Explore and visualize the data.

Split the data into training and testing sets.

Train a classification model (e.g., logistic regression, decision tree, or k-nearest neighbors).

Make predictions on the test set.

Evaluate the model's performance.

HOMEWORK QUESTIONS:

1. Load the Iris dataset and display the first few rows. Provide the code for this step.

2. Visualize the relationship between petal length and petal width for each species. Provide the code for this step.

3. Split the data into training and testing sets. Provide the code for this step.

4. Train a classification model (e.g., logistic regression). Provide the code for this step.

5. Make predictions on the test set. Provide the code for this step.

6. Evaluate the model's performance. Calculate and print accuracy, confusion matrix, and classification report.

1. Load the Iris dataset and display the first few rows. Provide the code for this step.

Answer:

python

```
from sklearn.datasets import load_iris

import pandas as pd

# Load the Iris dataset

iris = load_iris()

df = pd.DataFrame(data=iris.data, columns=iris.feature_names)

df['Species'] = iris.target_names[iris.target]

# Display the first few rows

print(df.head())
```

2. Visualize the relationship between petal length and petal width for each species. Provide the code for this step.

Answer:

python

```
import seaborn as sns

import matplotlib.pyplot as plt
```

```
sns.scatterplot(x='petal length (cm)', y='petal width (cm)', hue='Species', data=df)

plt.title('Petal Length vs. Petal Width by Species')

plt.show()
```

3. Split the data into training and testing sets. Provide the code for this step.

Answer:

python

```
from sklearn.model_selection import train_test_split

X = df.drop('Species', axis=1)

y = df['Species']

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

4. Train a classification model (e.g., logistic regression). Provide the code for this step.

Answer:

python

```
from sklearn.linear_model import LogisticRegression

model = LogisticRegression()

model.fit(X_train, y_train)
```

5. Make predictions on the test set. Provide the code for this step.

Answer:

python

```
y_pred = model.predict(X_test)
```

6. Evaluate the model's performance. Calculate and print accuracy, confusion matrix, and classification report.

Answer:

python

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

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:.2f}')

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

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

These questions and exercises cover the basics of a supervised learning classification task using the Iris dataset. They allow students to practice loading data, exploring it, and evaluating the performance of a classification model.

DATASET OF THE ABOVE

The Iris dataset is commonly available in machine learning libraries. Here's how we can load it using the scikit-learn library in Python:

```
python

from sklearn.datasets import load_iris

import pandas as pd

# Load the Iris dataset

iris = load_iris()

df = pd.DataFrame(data=iris.data, columns=iris.feature_names)

df['Species'] = iris.target_names[iris.target]

# Display the first few rows

print(df.head())
```

The dataset is now stored in the DataFrame 'df'. If you want to save it as a CSV file for reference, you can use the following code:

```
python

# Save the dataset as a CSV file

df.to_csv('iris_dataset.csv', index=False)
```

You can then use this CSV file for your supervised learning exercise.