

```

1 # Import the necessary libraries
2 import pandas as pd
3 import numpy as np
4 import matplotlib.pyplot as plt
5 from sklearn.preprocessing import StandardScaler
6 from sklearn.model_selection import train_test_split
7 from sklearn.metrics import mean_squared_error, r2_score
8 from sklearn.linear_model import LinearRegression
9
10 # Load the dataset
11 data = pd.read_csv('data.csv')
12
13 # Display the first few rows of the dataset
14 print(data.head())
15
16 # Check the data types of the columns
17 print(data.dtypes)
18
19 # Split the data into training and testing sets
20 X_train, X_test, y_train, y_test = train_test_split(
21     data[['feature1', 'feature2', 'feature3']],
22     data['target'],
23     test_size=0.2,
24     random_state=42
25 )
26
27 # Standardize the features
28 scaler = StandardScaler()
29 X_train = scaler.fit_transform(X_train)
30 X_test = scaler.transform(X_test)
31
32 # Create a linear regression model
33 model = LinearRegression()
34
35 # Train the model on the training data
36 model.fit(X_train, y_train)
37
38 # Predict the target values for the test data
39 y_pred = model.predict(X_test)
40
41 # Calculate the Mean Squared Error (MSE) and R-squared (R^2) score
42 mse = mean_squared_error(y_test, y_pred)
43 r2 = r2_score(y_test, y_pred)
44
45 # Print the MSE and R^2 score
46 print('MSE: ', mse)
47 print('R^2: ', r2)
48
49 # Plot the predicted values against the actual values
50 plt.scatter(y_test, y_pred)
51 plt.xlabel('Actual Values')
52 plt.ylabel('Predicted Values')
53 plt.title('Scatter Plot of Actual vs Predicted Values')
54 plt.show()
55
56 # Save the model to a file
57 joblib.dump(model, 'model.pkl')
58
59 # Load the saved model
60 loaded_model = joblib.load('model.pkl')
61
62 # Predict the target values for the test data using the loaded model
63 y_pred_loaded = loaded_model.predict(X_test)
64
65 # Calculate the Mean Squared Error (MSE) and R-squared (R^2) score for the loaded model
66 mse_loaded = mean_squared_error(y_test, y_pred_loaded)
67 r2_loaded = r2_score(y_test, y_pred_loaded)
68
69 # Print the MSE and R^2 score for the loaded model
70 print('MSE (loaded): ', mse_loaded)
71 print('R^2 (loaded): ', r2_loaded)
72
73 # Plot the predicted values against the actual values for the loaded model
74 plt.scatter(y_test, y_pred_loaded)
75 plt.xlabel('Actual Values')
76 plt.ylabel('Predicted Values (loaded model)')
77 plt.title('Scatter Plot of Actual vs Predicted Values (loaded model)')
78 plt.show()
79
80 # End of the script

```

```

1 # Import the necessary libraries
2 import pandas as pd
3 import numpy as np
4 import matplotlib.pyplot as plt
5 from sklearn.preprocessing import StandardScaler
6 from sklearn.model_selection import train_test_split
7 from sklearn.metrics import mean_squared_error, r2_score
8 from sklearn.linear_model import LinearRegression
9
10 # Load the dataset
11 data = pd.read_csv('data.csv')
12
13 # Display the first few rows of the dataset
14 print(data.head())
15
16 # Check the data types of the columns
17 print(data.dtypes)
18
19 # Split the data into training and testing sets
20 X_train, X_test, y_train, y_test = train_test_split(
21     data[['feature1', 'feature2', 'feature3']],
22     data['target'],
23     test_size=0.2,
24     random_state=42
25 )
26
27 # Standardize the features
28 scaler = StandardScaler()
29 X_train = scaler.fit_transform(X_train)
30 X_test = scaler.transform(X_test)
31
32 # Create a linear regression model
33 model = LinearRegression()
34
35 # Train the model on the training data
36 model.fit(X_train, y_train)
37
38 # Predict the target values for the test data
39 y_pred = model.predict(X_test)
40
41 # Calculate the Mean Squared Error (MSE) and R-squared (R^2) score
42 mse = mean_squared_error(y_test, y_pred)
43 r2 = r2_score(y_test, y_pred)
44
45 # Print the MSE and R^2 score
46 print('MSE: ', mse)
47 print('R^2: ', r2)
48
49 # Plot the predicted values against the actual values
50 plt.scatter(y_test, y_pred)
51 plt.xlabel('Actual Values')
52 plt.ylabel('Predicted Values')
53 plt.title('Scatter Plot of Actual vs Predicted Values')
54 plt.show()
55
56 # Save the model to a file
57 joblib.dump(model, 'model.pkl')
58
59 # Load the saved model
60 loaded_model = joblib.load('model.pkl')
61
62 # Predict the target values for the test data using the loaded model
63 y_pred_loaded = loaded_model.predict(X_test)
64
65 # Calculate the Mean Squared Error (MSE) and R-squared (R^2) score for the loaded model
66 mse_loaded = mean_squared_error(y_test, y_pred_loaded)
67 r2_loaded = r2_score(y_test, y_pred_loaded)
68
69 # Print the MSE and R^2 score for the loaded model
70 print('MSE (loaded): ', mse_loaded)
71 print('R^2 (loaded): ', r2_loaded)
72
73 # Plot the predicted values against the actual values for the loaded model
74 plt.scatter(y_test, y_pred_loaded)
75 plt.xlabel('Actual Values')
76 plt.ylabel('Predicted Values (loaded model)')
77 plt.title('Scatter Plot of Actual vs Predicted Values (loaded model)')
78 plt.show()
79
80 # End of the script

```

```

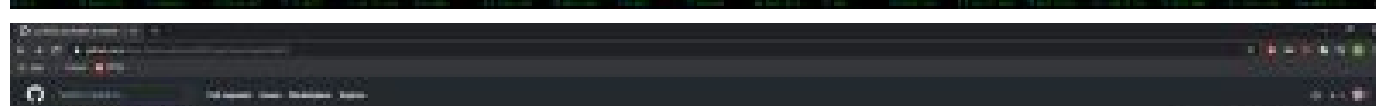
1 // Import the React library
2 import React from 'react';
3
4 // Define the App component
5 function App() {
6   // Define the state
7   const [count, setCount] = useState(0);
8
9   // Define the increment function
10  const increment = () => {
11    setCount(count + 1);
12  };
13
14  // Define the decrement function
15  const decrement = () => {
16    setCount(count - 1);
17  };
18
19  // Define the reset function
20  const reset = () => {
21    setCount(0);
22  };
23
24  // Return the JSX
25  return (
26    <div>
27      <h1>Counter</h1>
28      <p>Count: {count}</p>
29      <p>Increment: {increment}</p>
30      <p>Decrement: {decrement}</p>
31      <p>Reset: {reset}</p>
32    </div>
33  );
34}
35
36 // Export the App component
37 export default App;

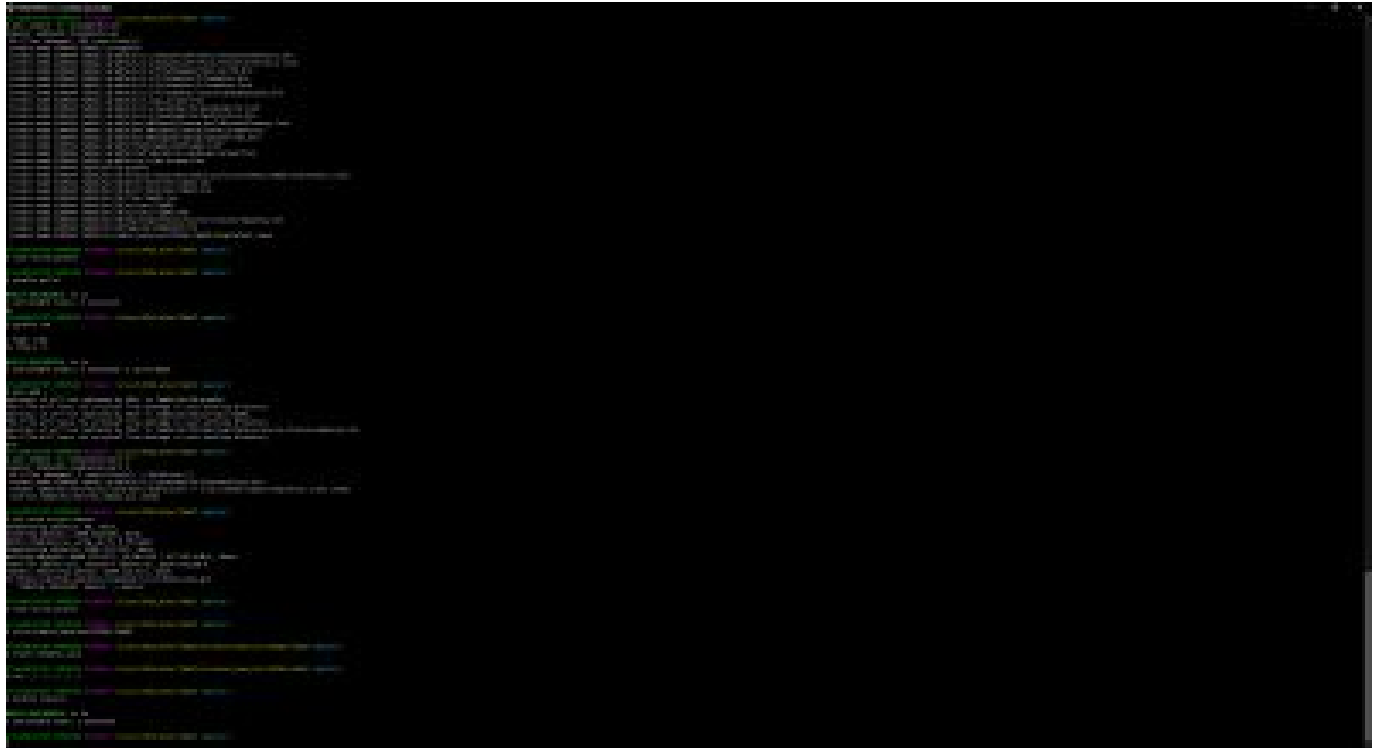
```

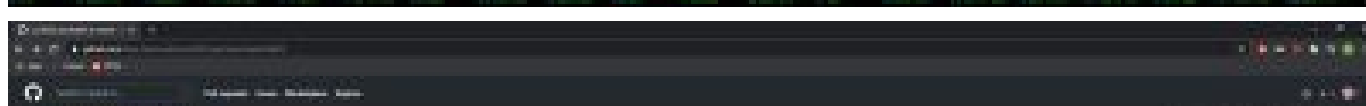
```

1 // Import the React library
2 import React from 'react';
3
4 // Define the App component
5 function App() {
6   // Define the state
7   const [count, setCount] = useState(0);
8
9   // Define the increment function
10  const increment = () => {
11    setCount(count + 1);
12  };
13
14  // Define the decrement function
15  const decrement = () => {
16    setCount(count - 1);
17  };
18
19  // Define the reset function
20  const reset = () => {
21    setCount(0);
22  };
23
24  // Return the JSX
25  return (
26    <div>
27      <h1>Counter</h1>
28      <p>Count: {count}</p>
29      <p>Increment: {increment}</p>
30      <p>Decrement: {decrement}</p>
31      <p>Reset: {reset}</p>
32    </div>
33  );
34}
35
36 // Export the App component
37 export default App;

```


[illegible]



[illegible]