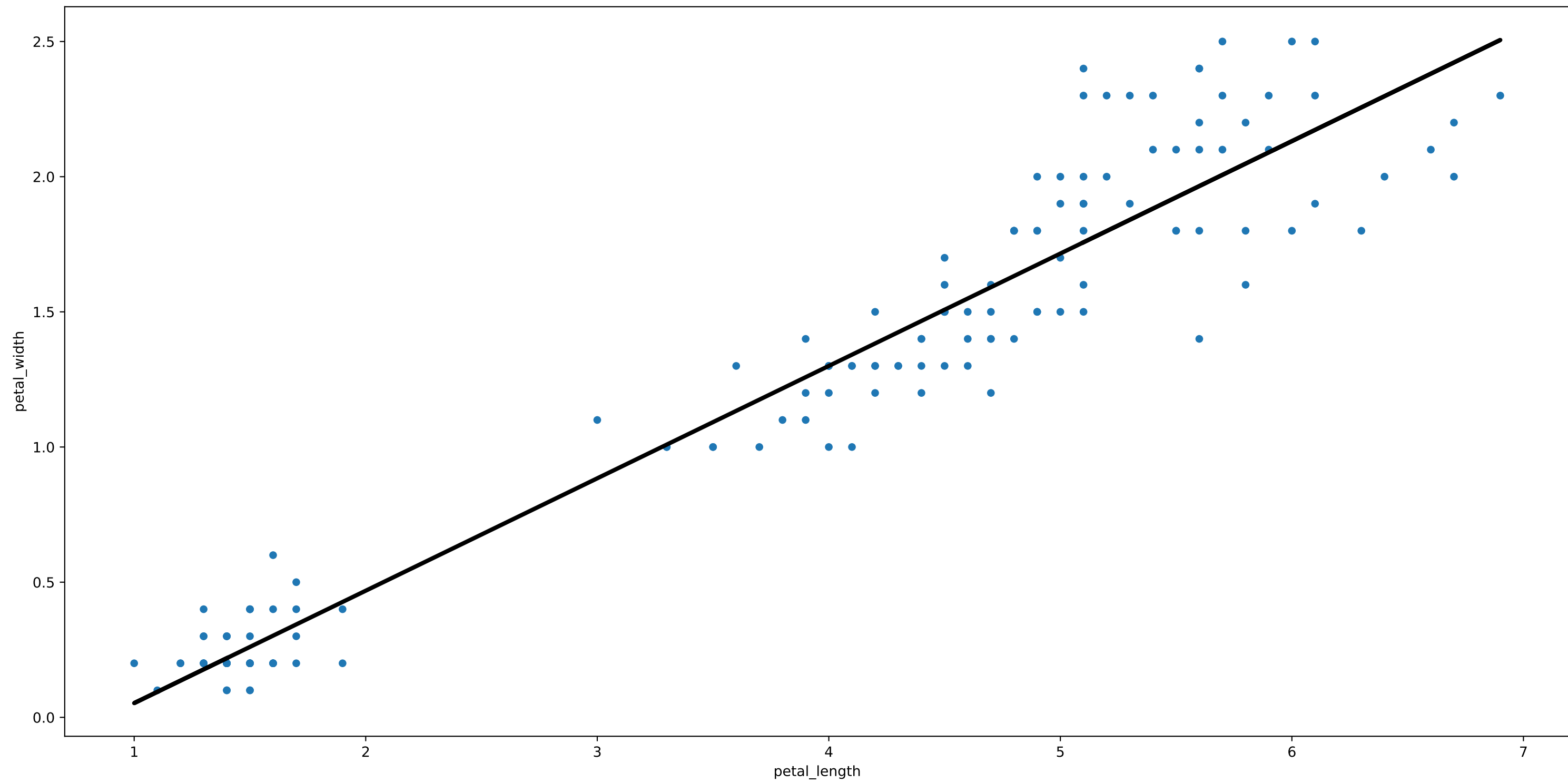
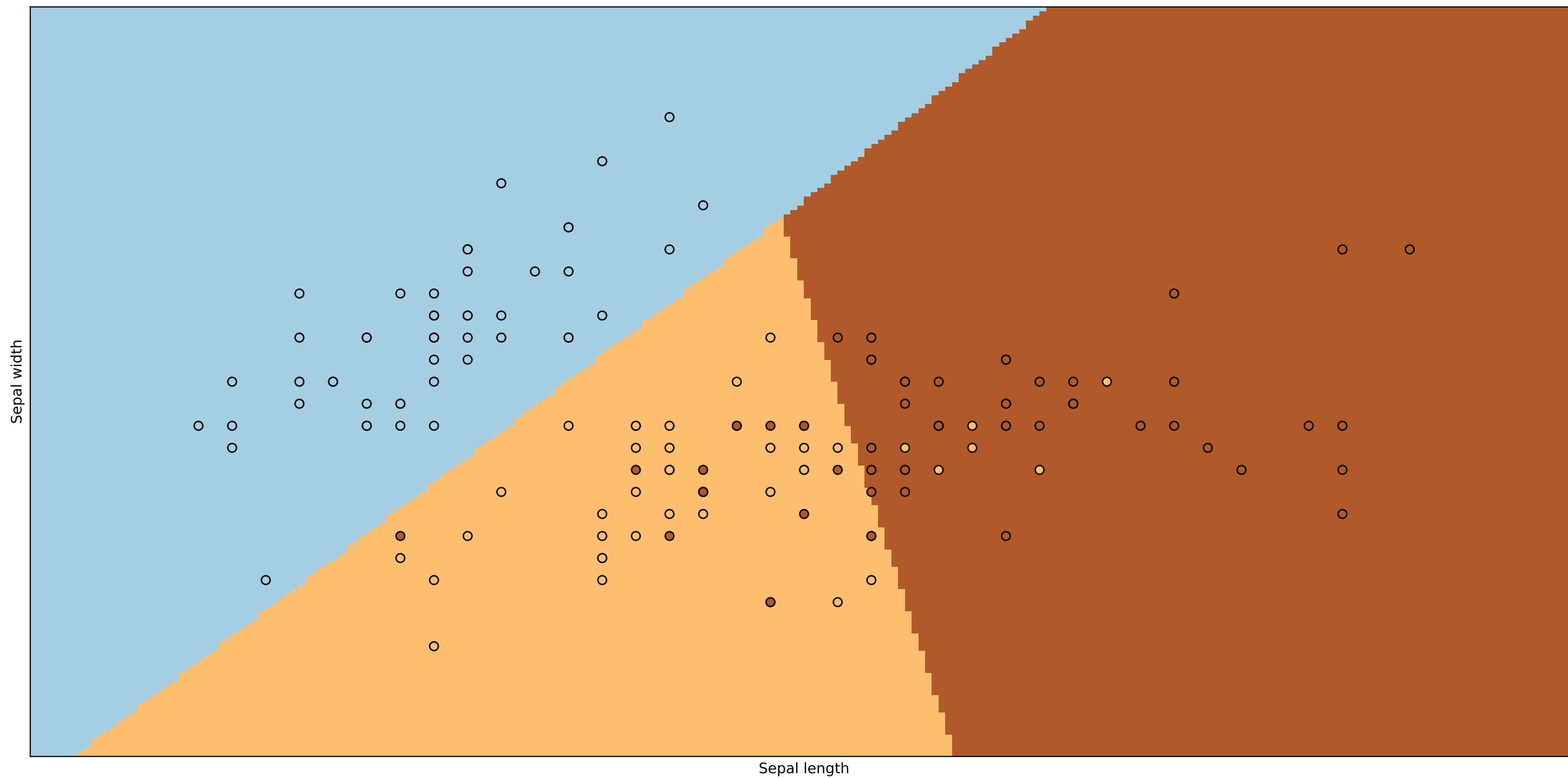


```
1 # -*- coding: utf-8 -*-
2 """
3 Created on Tue Aug  4 12:47:37 2020
4
5 @author: HETSHAH
6 """
7
8 from sklearn import datasets, linear_model
9 import pandas as pd
10 import matplotlib.pyplot as plt
11 import seaborn.apionly as sns
12
13 iris = sns.load_dataset('iris')
14 fit_data = iris[["petal_length", "petal_width"]].values
15 x_data = fit_data[:,0].reshape(-1,1)
16 y_data = fit_data[:,1].reshape(-1,1)
17
18 # Create linear regression object
19 regr = linear_model.LinearRegression()
20 # once the data is reshaped, running the fit is simple
21 regr.fit(x_data, y_data)
22
23 # we can then plot the data and out fit
24 axes = iris.plot(x="petal_length", y="petal_width", kind="
    scatter")
25 plt.plot(x_data, regr.predict(x_data), color='black',
    linewidth=3)
26 plt.show()
```



```
1  # -*- coding: utf-8 -*-
2  """
3  Created on Tue Aug  4 12:43:16 2020
4
5  @author: HETSHAH
6  """
7
8  import numpy as np
9  import matplotlib.pyplot as plt
10 from sklearn.linear_model import LogisticRegression
11 from sklearn import datasets
12
13 iris = datasets.load_iris()
14 X = iris.data[:, :2]  # we only take the first two
    features.
15 Y = iris.target
16
17 logreg = LogisticRegression(C=1e5)
18
19 # Create an instance of Logistic Regression Classifier and
    fit the data.
20 logreg.fit(X, Y)
21
22 # Plot the decision boundary. For that, we will assign a
    color to each
23 # point in the mesh [x_min, x_max]x[y_min, y_max].
24 x_min, x_max = X[:, 0].min() - .5, X[:, 0].max() + .5
25 y_min, y_max = X[:, 1].min() - .5, X[:, 1].max() + .5
26 h = .02  # step size in the mesh
27 xx, yy = np.meshgrid(np.arange(x_min, x_max, h), np.arange
    (y_min, y_max, h))
28 Z = logreg.predict(np.c_[xx.ravel(), yy.ravel()])
29
30 # Put the result into a color plot
31 Z = Z.reshape(xx.shape)
32 plt.figure(1, figsize=(4, 3))
33 plt.pcolormesh(xx, yy, Z, cmap=plt.cm.Paired)
34
35 # Plot also the training points
36 plt.scatter(X[:, 0], X[:, 1], c=Y, edgecolors='k', cmap=
    plt.cm.Paired)
37 plt.xlabel('Sepal length')
38 plt.ylabel('Sepal width')
39
40 plt.xlim(xx.min(), xx.max())
```

```
41 plt.ylim(yy.min(), yy.max())
42 plt.xticks(())
43 plt.yticks(())
44
45 plt.show()
```



```
1  # -*- coding: utf-8 -*-
2  """
3  Created on Tue Aug  4 10:18:40 2020
4
5  @author: HETSHAH
6  """
7
8
9  import pandas as pd;
10 import seaborn as sns
11 import numpy as np
12 from sklearn.model_selection import train_test_split
13 from sklearn.datasets import load_iris
14 from sklearn.linear_model import LinearRegression;
15 from sklearn.metrics import mean_absolute_error
16 from sklearn.metrics import mean_squared_error
17
18
19 print("start")
20 iris = load_iris()
21
22 iris_df = pd.DataFrame(data=iris.data, columns=iris.
    feature_names)
23 target_df= pd.DataFrame(data=iris.target, columns=['
    species'])
24
25 def converter(specie):
26     if(specie == 0):
27         return 'setosa'
28     elif specie == 1:
29         return 'versicolor'
30     else:
31         return 'virginica'
32
33 target_df['species']=target_df['species'].apply(converter)
34
35 iris_df = pd.concat([iris_df,target_df],axis=1)
36
37 iris_df.info();
38
39 sns.pairplot(iris_df, hue= 'species')
40 iris_df.drop('species', axis= 1, inplace= True)
41 target_df = pd.DataFrame(columns= ['species'], data= iris.
    target)
42 iris_df = pd.concat([iris_df, target_df], axis= 1)
```

```
43
44
45 X= iris_df.drop(labels= 'sepal length (cm)', axis= 1)
46 y= iris_df['sepal length (cm)']
47
48
49 X_train, X_test, y_train, y_test = train_test_split(X, y,
    test_size= 0.33, random_state= 101)
50
51 lr = LinearRegression()
52
53 #train
54 lr.fit(X_train, y_train)
55
56 #predict
57 lr.predict(X_test)
58 pred = lr.predict(X_test)
59
60
61 print('Mean Absolute Error:', mean_absolute_error(y_test,
    pred))
62 print('Mean Squared Error:', mean_squared_error(y_test,
    pred))
63 print('Mean Root Squared Error:', np.sqrt(
    mean_squared_error(y_test, pred)))
64 print("stop")
65
66
67
68
69
70 import matplotlib.pyplot as plt
71 import seaborn as sns
72 from sklearn.linear_model import LogisticRegression
73 from sklearn.metrics import classification_report
74 from sklearn.metrics import accuracy_score
75 from sklearn.model_selection import train_test_split
76 import pandas.util.testing as tm
77
78
79 data = sns.load_dataset("iris")
80 data.head()
81 X = data.iloc[:, :-1]
82 y = data.iloc[:, -1]
83 plt.xlabel('Features')
```

```
84 plt.ylabel('Species')
85
86 pltX = data.loc[:, 'sepal_length']
87 pltY = data.loc[:, 'species']
88 plt.scatter(pltX, pltY, color='blue', label='sepal_length
    ')
89
90 pltX = data.loc[:, 'sepal_width']
91 pltY = data.loc[:, 'species']
92 plt.scatter(pltX, pltY, color='green', label='sepal_width
    ')
93
94 pltX = data.loc[:, 'petal_length']
95 pltY = data.loc[:, 'species']
96 plt.scatter(pltX, pltY, color='red', label='petal_length'
    )
97
98 pltX = data.loc[:, 'petal_width']
99 pltY = data.loc[:, 'species']
100 plt.scatter(pltX, pltY, color='black', label='petal_width
    ')
101
102 plt.legend(loc=4, prop={'size':8})
103 plt.show()
104
105 #Split the data into 80% training and 20% testing
106 x_train, x_test, y_train, y_test = train_test_split(X, y,
    test_size=0.2, random_state=42)
107
108 #Train the model
109 model = LogisticRegression()
110 model.fit(x_train, y_train) #Training the model
111
112 #Test the model
113 predictions = model.predict(x_test)
114 print(predictions) # printing predictions
115
116 print() # Printing new line
117
118 #Check precision, recall, f1-score
119 print( classification_report(y_test, predictions) )
120
121 print( accuracy_score(y_test, predictions))
```



