

Spyder (Python 3.8)

File Edit Search Source Run Debug Consoles Projects Tools View Help

C:\Users\Muratcan\Desktop\MachineLearningCourse.py

```
1 import pandas as pd
2 import quandl
3
4 df = quandl.get('WIKI/GOOGL')
5
6 print(df.head())
7
```

Usage

Here you can get help of any object by pressing **Ctrl+I** in front of it, either on the Editor or the Console.

Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in *Preferences > Help*.

New to Spyder? Read our [tutorial](#)

Help Variable explorer Plots Files

Console 1/A

```
<bound method NDFrame.head of
Low ... Adj. Low Adj. Close Adj. Volume Open High
Date ...
2004-08-19 100.01 104.06 95.96 ... 48.128568
50.322842 44659000.0
2004-08-20 101.01 109.08 100.50 ... 50.405597
54.322689 22834300.0
2004-08-23 110.76 113.48 109.05 ... 54.693835
54.869377 18256100.0
2004-08-24 111.24 111.60 103.57 ... 51.945350
52.597363 15247300.0
2004-08-25 104.76 108.00 103.88 ... 52.100830
53.164113 9188600.0
...
...
2018-03-21 1092.57 1108.70 1087.21 ... 1087.210000
1094.000000 1990515.0
2018-03-22 1080.01 1083.92 1049.64 ... 1049.640000
1053.150000 3418154.0
2018-03-23 1051.37 1066.78 1024.87 ... 1024.870000
1026.550000 2413517.0
2018-03-26 1050.60 1059.27 1010.58 ... 1010.580000
1054.090000 3272409.0
2018-03-27 1063.90 1064.54 997.62 ... 997.620000
1006.940000 2940957.0

[3424 rows x 12 columns]>

In [24]:
```

IPython console History

↳ JSPython: ready Kite: ready conda: base (Python 3.8.8) Line 7, Col 1 UTF-8 CRLF RW Mem 47%

Spyder (Python 3.8)

File Edit Search Source Run Debug Consoles Projects Tools View Help

temp.py x untitled0.py* x MachineLearningCourse.py x

```
1 import pandas as pd
2 import quandl
3
4 df = quandl.get('WIKI/GOOGL')
5
6 print(df.head)
7
8 df = df[['Adj. Open', 'Adj. High', 'Adj. Low', 'Adj. CL
9 df['HL_PCT'] = (df['Adj. High'] - df['Adj. Close'])
10 df['PCT_change'] = (df['Adj. Close'] - df['Adj. Open
11 df = df[['Adj. Close', 'HL_PCT', 'PCT_change', 'Adj. Vo
12
13 print(df.head)
```

Usage

Here you can get help of any object by pressing **Ctrl+I** in front of it, either on the Editor or the Console.

Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in **Preferences > Help**.

New to Spyder? Read our [tutorial](#)

Help Variable explorer Plots Files

Console 1/A x

```
1094.000000 1990515.0
2018-03-22 1080.01 1083.92 1049.64 ... 1049.640000
1053.150000 3418154.0
2018-03-23 1051.37 1066.78 1024.87 ... 1024.870000
1026.550000 2413517.0
2018-03-26 1050.60 1059.27 1010.58 ... 1010.580000
1054.090000 3272409.0
2018-03-27 1063.90 1064.54 997.62 ... 997.620000
1006.940000 2940957.0

[3424 rows x 12 columns]>
<bound method NDFrame.head of                                Adj. Close  HL_PCT
PCT_change Adj. Volume
Date
2004-08-19    50.322842  3.712563    0.324968  44659000.0
2004-08-20    54.322689  0.710922    7.227007  22834300.0
2004-08-23    54.869377  3.729433   -1.227880  18256100.0
2004-08-24    52.597363  6.417469   -5.726357  15247300.0
2004-08-25    53.164113  1.886792    1.183658   9188600.0
...
2018-03-21  1094.000000  1.343693    0.130884  1990515.0
2018-03-22  1053.150000  2.921711   -2.487014  3418154.0
2018-03-23  1026.550000  3.918952   -2.360729  2413517.0
2018-03-26  1054.090000  0.491419    0.332191  3272409.0
2018-03-27  1006.940000  5.720301   -5.353887  2940957.0

[3424 rows x 4 columns]>

In [34]:
```

C:\Users\Muratcan\Desktop\MachineLearningCourse.py

```
temp.py × untitled0.py* × MachineLearningCourse.py* ×  
1 import pandas as pd  
2 import quandl  
3 import math  
4  
5 df = quandl.get('WIKI/GOOGL')  
6  
7 print(df.head)  
8  
9 df = df[['Adj. Open', 'Adj. High', 'Adj. Low', 'Adj. Close',  
10 df['HL_PCT'] = (df['Adj. High'] - df['Adj. Close'])  
11 df['PCT_change'] = (df['Adj. Close'] - df['Adj. Open'])  
12 df = df[['Adj. Close', 'HL_PCT', 'PCT_change', 'Adj. Volume']]  
13  
14 forecast_col = 'Adj. Close'  
15 df.fillna('-99999', inplace=True)  
16  
17 forecast_out = int(math.ceil(0.1*len(df)))  
18  
19 df['Label'] = df[forecast_col].shift(-forecast_out)  
20 df.dropna(inplace=True)  
21 print(df.head())  
22  
23
```

```
s\Muratcan\Desktop\MachineLearningCourse.py
np.py × untitled0.py* × MachineLearningCourse.py ×

import pandas as pd
import quandl, math
import numpy as np
from sklearn import preprocessing, cross_validation, svm
from sklearn.linear_model import LinearRegression
df = quandl.get('WIKI/GOOGL')

print(df.head)

df = df[['Adj. Open', 'Adj. High', 'Adj. Low', 'Adj. Close', 'Adj. Volume']]
df['HL_PCT'] = (df['Adj. High'] - df['Adj. Close']) / df['Adj. Close']
df['PCT_change'] = (df['Adj. Close'] - df['Adj. Open']) / df['Adj. Open']
df = df[['Adj. Close', 'HL_PCT', 'PCT_change', 'Adj. Volume']]

forecast_col = 'Adj. Close'
df.fillna('-99999', inplace=True)

forecast_out = int(math.ceil(0.1*len(df)))

df['label'] = df[forecast_col].shift(-forecast_out)
df.dropna(inplace=True)
print(df.head())

x = np.array(df.drop(['label'], 1))
y = np.array(df['label'])
x = preprocessing.scale(x)
y = np.array(df['label'])

x_train, x_test, y_train, y_test = cross_validation.train_test_split(x, y, test_size=0.3)

clf = LinearRegression()
clf.fit(x_train, y_train)
accuracy = clf.score(x_test, y_test)

print(accuracy)
```

LSP Python: ready Kite: ready conda: base (Python 3.8.8)

```
df = quandl.get('WIKI/GOOGL')

print(df.head)

df = df[['Adj. Open', 'Adj. High', 'Adj. Low', 'Adj. Close', 'Adj. Volume']]
df['HL_PCT'] = (df['Adj. High'] - df['Adj. Close']) / df['Adj. Close']
df['PCT_change'] = (df['Adj. Close'] - df['Adj. Open']) / df['Adj. Close']
df = df[['Adj. Close', 'HL_PCT', 'PCT_change', 'Adj. Volume']]

forecast_col = 'Adj. Close'
df.fillna(-99999, inplace=True)

forecast_out = int(math.ceil(0.1*len(df)))

df['Label'] = df[forecast_col].shift(-forecast_out)

print(df.head())

x = np.array(df.drop(['Label'], 1))
x = preprocessing.scale(x)
x = x[:-forecast_out]
x_lately = x[-forecast_out:]

df.dropna(inplace=True)
y = np.array(df['Label'])
y = np.array(df['Label'])

x_train, x_test, y_train, y_test = cross_validation.train_test_split(x, y)

clf = LinearRegression(n_jobs=-1)
clf.fit(x_train, y_train)
accuracy = clf.score(x_test, y_test)

forecast_set = clf.predict(x_lately)

print(forecast_set, accuracy, forecast_out)

df['Forecast'] = np.nan
last_date = df.iloc[-1].name
last_unix = last_date.timestamp
one_day = 86400
next_unix = last_unix + one_day

for i in forecast_set:
    next_date = datetime.datetime.fromtimestamp(next_unix)
    next_unix += one_day
    df.loc[next_date] = [np.nan for _ in range(len(df.columns)-1)]

df['Adj. Close'].plot()
df['Forecast'].plot()
plt.legend(loc=4)
plt.xlabel('Date')
plt.ylabel('Price')
plt.show()
```

```

file Edit Search Source Run Debug Consoles Projects Tools View Help

:Users\Muratcan\Desktop\MachineLearningCourse.py
temp.py x untitle0.py* x MachineLearningCourse.py x

16 df = df[['Adj. Open', 'Adj. High', 'Adj. Low', 'Adj. Close', 'Adj. Vol
17 df['HL_PCT'] = (df['Adj. High'] - df['Adj. Close']) / df['Adj. Clo
18 df['PCT_change'] = (df['Adj. Close'] - df['Adj. Open']) / df['Adj.
19 df = df[['Adj. Close', 'HL_PCT', 'PCT_change', 'Adj. Volume']]
20
21 forecast_col = 'Adj. Close'
22 df.fillna('-99999', inplace=True)
23
24 forecast_out = int(math.ceil(0.1*len(df)))
25
26 df['label'] = df[forecast_col].shift(-forecast_out)
27
28 print(df.head())
29
30 x = np.array(df.drop(['label'],1))
31 x = preprocessing.scale(x)
32 x = x[:-forecast_out]
33 x_lately = x[-forecast_out:]
34
35 df.dropna(inplace=True)
36 y = np.array(df['label'])
37
38 x_train, x_test, y_train, y_test = cross_validation.train_test_spl
39
40 clf = LinearRegression(n_jobs=-1)
41 clf.fit(x_train, y_train)
42
43 with open('linearregression.pickle','wb') as f:
44     pickle.dump(clf, f)
45
46 pickle_in = open('linearregression.pickle','rb')
47 clf = pickle.load(pickle_in)
48
49
50 accuracy = clf.score(x_test, y_test)
51 forecast_set = clf.predict(x_lately)
52 print(forecast_set, accuracy, forecast_out)
53
54 df['Forecast'] = np.nan
55 last_date = df.iloc[-1].name
56 last_unix = last_date.timestamp
57 one_day = 86400
58 next_unix = last_unix + one_day
59
60 for i in forecast_set:
61     next_date = datetime.datetime.fromtimestamp(next_unix)
62     next_unix += one_day
63     df.loc[next_date] = [np.nan for _ in range(len(df.columns)-1)]
64
65 df['Adj. Close'].plot()
66 df['Forecast'].plot()
67 plt.legend(loc=4)
68 plt.xlabel('Date')
69 plt.ylabel('Price')
70 plt.show()
71
72
73
74
75
76

```

```

5 from statistics import mean
6 import numpy as np
7 import matplotlib.pyplot as plt
8 from matplotlib import style
9 style.use('fivethirtyeight')
10
11 xs = np.array([1,2,3,4,5,6], dtype = np.float64)
12 ys = np.array([5,4,6,5,6,7], dtype = np.float64)
13
14     def best_fit_slope_and_intercept(xs,ys):
15         m = ((mean(xs) * mean(ys)) - mean(xs*ys)) / ((mean(x
16         b = mean(ys) - m*mean(xs)
17         return m, b
18     m,b = best_fit_slope_and_intercept(xs,ys)
19
20     regression_line = [(m*x)+b for x in xs]
21
22 plt.scatter(xs,ys)
23 plt.plot(xs, regression)

```

LSP Python: ready

Kite: ready

conda: base (Python 3.8

```

Edit Search Source Run Debug Consoles Projects Tools View Help

ers\Muratcan\Desktop\MachineLearningCourse.py

temp.py x unttitled0.py* x MachineLearningCourse.py* x

63 # next_date = datetime.datetime.fromtimestamp(next_unix)
64 # next_unix += one_day
65 # df.loc[next_date] = [np.nan for _ in range(len(df.columns)-
66 #
67 # df['Adj. Close'].plot()
68 # df['Forecast'].plot()
69 # plt.legend(loc=4)
70 # plt.xlabel('Date')
71 # plt.ylabel('Price')
72 # plt.show()
73 # =====
74
75
76 from statistics import mean
77 import numpy as np
78 import matplotlib.pyplot as plt
79 from matplotlib import style
80 style.use('fivethirtyeight')
81
82 xs = np.array([1,2,3,4,5,6], dtype = np.float64)
83 ys = np.array([5,4,6,5,6,7], dtype = np.float64)
84
85 def best_fit_slope_and_intercept(xs,ys):
86     m = ((mean(xs) * mean(ys)) - mean(xs*ys)) / ((mean(xs)
87     b = mean(ys) - m*mean(xs)
88     return m, b
89
90
91 def squared_error(ys_orig, ys_line):
92     return sum((ys_line-ys_orig**2))
93
94 def coefficient_of_determination(ys_orig,ys_line):
95     y_mean_line = [mean(ys_orig) for y in ys_orig]
96     squared_error_regr = squared_error(ys_orig,ys_line)
97     squared_error_y_mean = squared_error(ys_orig,y_mean_line)
98     return 1 - (squared_error_regr / squared_error_y_mean)
99
100 m,b = best_fit_slope_and_intercept(xs,ys)
101
102 regression_line = [(m*x)+b for x in xs]
103
104 predict_x = 8
105 predict_y = (m*predict_x) + b
106
107 r_squared = coefficient_of_determination(ys,regression_line)
108 print(r_squared)
109
110
111 plt.scatter(xs,ys)
112 plt.scatter(predict_x,predict_y,color = 'g')
113 plt.plot(xs, regression_line)
114 plt.show()
115
116
117
118
119
120
121
122
123

```



```

Edit Search Source Run Debug Consoles Projects Tools View Help

sers\Muratcan\Desktop\MachineLearningCourse.py
temp.py x untitle0.py* x MachineLearningCourse.py x

66 #
67 # df['Adj. Close'].plot()
68 # df['Forecast'].plot()
69 # plt.legend(loc=4)
70 # plt.xlabel('Date')
71 # plt.ylabel('Price')
72 # plt.show()
73 # =====
74
75
76 from statistics import mean
77 import numpy as np
78 import matplotlib.pyplot as plt
79 from matplotlib import style
80 style.use('fivethirtyeight')
81
82 xs = np.array([1,2,3,4,5,6], dtype = np.float64)
83 ys = np.array([5,4,6,5,6,7], dtype = np.float64)
84
85 def best_fit_slope_and_intercept(xs,ys):
86     m = ( ((mean(xs) * mean(ys)) - mean(xs*ys)) / ((mean(xs)
87     b = mean(ys) - m*mean(xs)
88     return m, b
89
90
91 def squared_error(ys_orig, ys_line):
92     return sum((ys_line-ys_orig**2))
93
94 def coefficient_of_determination(ys_orig,ys_line):
95     y_mean_line = [mean(ys_orig) for y in ys_orig]
96     squared_error_regr = squared_error(ys_orig,ys_line)
97     squared_error_y_mean = squared_error(ys_orig,y_mean_line)
98     return 1 - (squared_error_regr / squared_error_y_mean)
99
100 m,b = best_fit_slope_and_intercept(xs,ys)
101
102 regression_line = [(m*x)+b for x in xs]
103
104 predict_x = 8
105 predict_y = (m*predict_x) + b
106
107 r_squared = coefficient_of_determination(ys,regression_line)
108 print(r_squared)
109
110
111 plt.scatter(xs,ys)
112 plt.scatter(predict_x,predict_y,color = 'g')
113 plt.plot(xs, regression_line)
114 plt.show()
115
116
117
118
119
120
121
122
123
124
125

```

```
Edit Search Source Run Debug Consoles Projects Tools View Help

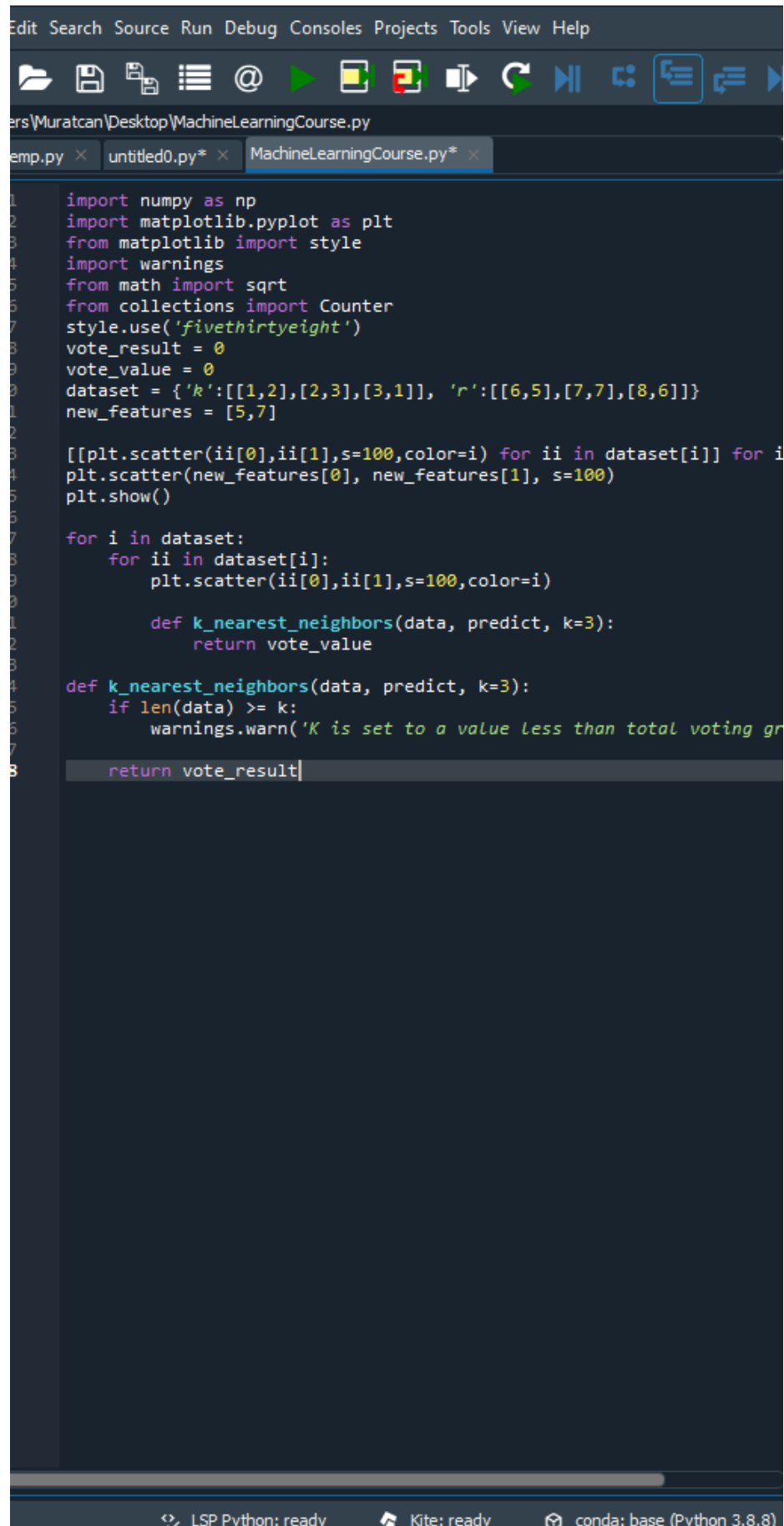
Users\Muratcan\Desktop\MachineLearningCourse.py

temp.py x untitle0.py* x MachineLearningCourse.py x

1  import numpy as np
2  from sklearn import preprocessing, cross_validation, neighbors
3  import pandas as pd
4
5  df = pd.read_csv('data.txt')
6  df.replace('?', -99999, inplace=True)
7  df.drop(['id'], 1, inplace=True)
8
9
10 X = np.array(df.drop(['class'], 1))
11 y = np.array(df['class'])
12
13 X_train, X_test, y_train, y_test = cross_validation.train_test_split(X, y, test_size=0.3)
14 clf = neighbors.KNeighborsClassifier()
15 clf.fit(X_train, y_train)
16 accuracy = clf.score(X_test, y_test)
17 print(accuracy)
18
19
20 example_measures = np.array([4,2,1,1,1,2,3,2,1])
21 prediction = clf.predict(example_measures)
22 print(prediction)
23
24
25
26 example_measures = np.array([4,2,1,1,1,2,3,2,1])
27 example_measures = example_measures.reshape(1, -1)
28 prediction = clf.predict(example_measures)
29 print(prediction)
30 example_measures = np.array([[4,2,1,1,1,2,3,2,1],[4,2,1,1,1,2,3,2,1]])
31 example_measures = example_measures.reshape(2, -1)
32 prediction = clf.predict(example_measures)
33 print(prediction)
34
35 example_measures = np.array([[4,2,1,1,1,2,3,2,1],[4,2,1,1,1,2,3,2,1]])
36 example_measures = example_measures.reshape(len(example_measures), -1)
37 prediction = clf.predict(example_measures)
38 print(prediction)
```

LSP Python: ready Kite: ready conda: base (Python 3.8.8)

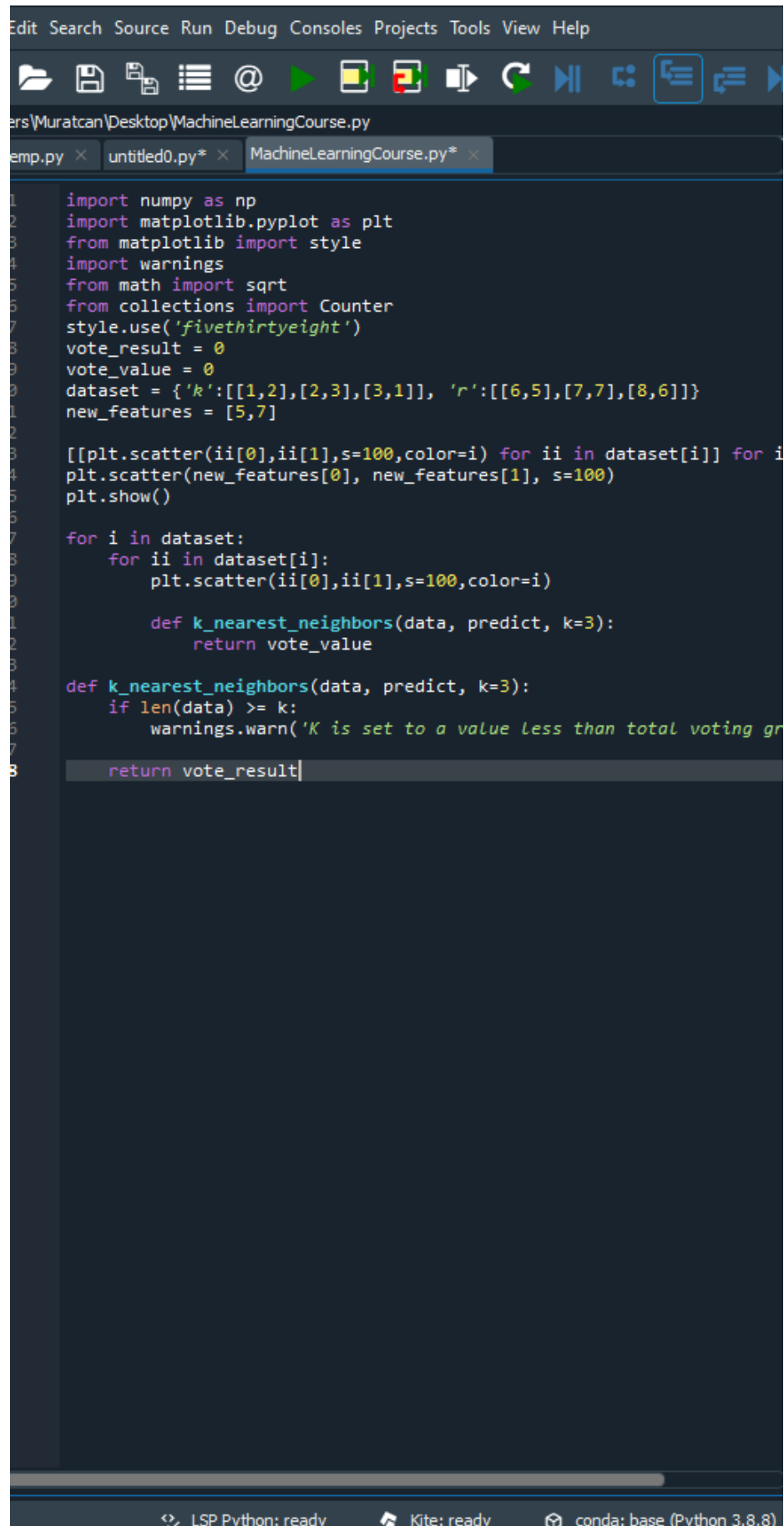
pyder (Python 3.8)



```
1 import numpy as np
2 import matplotlib.pyplot as plt
3 from matplotlib import style
4 import warnings
5 from math import sqrt
6 from collections import Counter
7 style.use('fivethirtyeight')
8 vote_result = 0
9 vote_value = 0
10 dataset = {'k':[[1,2],[2,3],[3,1]], 'r':[[6,5],[7,7],[8,6]]}
11 new_features = [5,7]
12
13 [[plt.scatter(ii[0],ii[1],s=100,color=i) for ii in dataset[i]] for i in dataset]
14 plt.scatter(new_features[0], new_features[1], s=100)
15 plt.show()
16
17 for i in dataset:
18     for ii in dataset[i]:
19         plt.scatter(ii[0],ii[1],s=100,color=i)
20
21         def k_nearest_neighbors(data, predict, k=3):
22             return vote_value
23
24 def k_nearest_neighbors(data, predict, k=3):
25     if len(data) >= k:
26         warnings.warn('K is set to a value less than total voting gr
27
28     return vote_result
```

LSP Python: ready Kite: ready conda: base (Python 3.8.8)

pyder (Python 3.8)



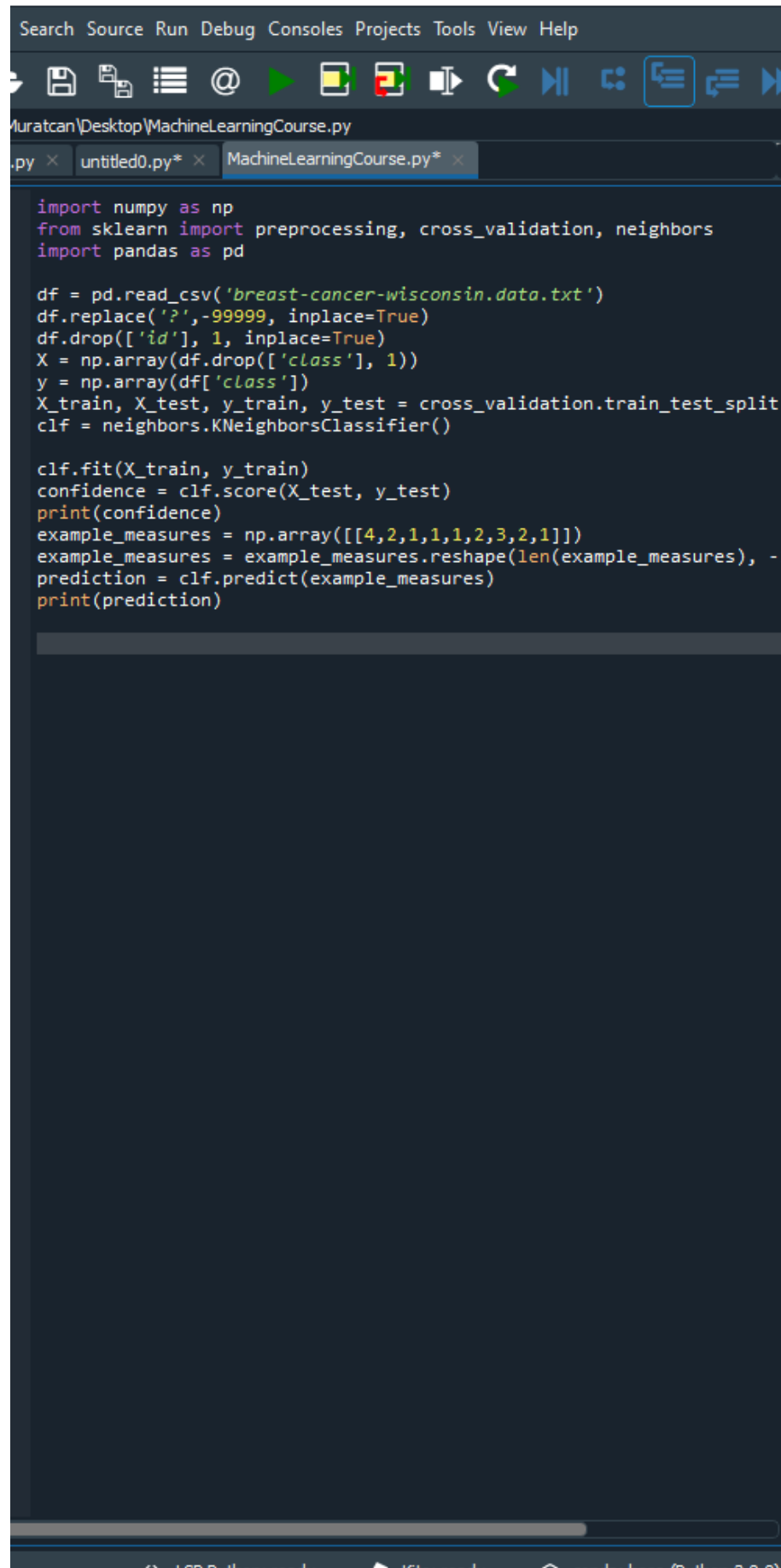
```
1 import numpy as np
2 import matplotlib.pyplot as plt
3 from matplotlib import style
4 import warnings
5 from math import sqrt
6 from collections import Counter
7 style.use('fivethirtyeight')
8 vote_result = 0
9 vote_value = 0
10 dataset = {'k':[[1,2],[2,3],[3,1]], 'r':[[6,5],[7,7],[8,6]]}
11 new_features = [5,7]
12
13 [[plt.scatter(ii[0],ii[1],s=100,color=i) for ii in dataset[i]] for i in dataset]
14 plt.scatter(new_features[0], new_features[1], s=100)
15 plt.show()
16
17 for i in dataset:
18     for ii in dataset[i]:
19         plt.scatter(ii[0],ii[1],s=100,color=i)
20
21         def k_nearest_neighbors(data, predict, k=3):
22             return vote_value
23
24 def k_nearest_neighbors(data, predict, k=3):
25     if len(data) >= k:
26         warnings.warn('K is set to a value less than total voting gr
27
28     return vote_result
```

LSP Python: ready Kite: ready conda: base (Python 3.8.8)

```
temp.py ×  untitled0.py* ×  MachineLearningCourse.py* ×  
1  import numpy as np  
2  from math import sqrt  
3  import warnings  
4  from collections import Counter  
5  import pandas as pd  
6  import random  
7  
8  def k_nearest_neighbors(data, predict, k=3):  
9      if len(data) >= k:  
10         warnings.warn('K is set to a value less than total voting gr  
11         distances = []  
12         for group in data:  
13             for features in data[group]:  
14                 euclidean_distance = np.linalg.norm(np.array(features)-n  
15                 distances.append([euclidean_distance, group])  
16  
17         votes = [i[1] for i in sorted(distances)[:k]]  
18         vote_result = Counter(votes).most_common(1)[0][0]  
19         confidence = Counter(votes).most_common(1)[0][1] / k  
20  
21         return vote_result, confidence  
22 df = pd.read_csv("breast-cancer-wisconsin.data.txt")  
23 df.replace('?', -99999, inplace=True)  
24 df.drop(['id'], 1, inplace=True)  
25 full_data = df.astype(float).values.tolist()  
26 random.shuffle(full_data)  
27 test_size = 0.4  
28 train_set = {2:[], 4:[]}  
29 test_set = {2:[], 4:[]}  
30 train_data = full_data[:int(test_size*len(full_data))]  
31 test_data = full_data[int(test_size*len(full_data)):]  
32  
33 for i in train_data:  
34     train_set[i[-1]].append(i[:-1])  
35  
36 for i in test_data:  
37     test_set[i[-1]].append(i[:-1])  
38  
39 correct = 0  
40 total = 0  
41 for group in test_set:  
42     for data in test_set[group]:  
43         vote, confidence = k_nearest_neighbors(train_set, data, k=5)  
44         if group == vote:  
45             correct += 1  
46         total += 1  
47 print('Accuracy:', correct/total)
```

LSP Python: ready Kite: ready conda: base (Python 3.8.8)

er (Python 3.8)



```
Search Source Run Debug Consoles Projects Tools View Help

Muratcan\Desktop\MachineLearningCourse.py

.py x untitled0.py* x MachineLearningCourse.py* x

import numpy as np
from sklearn import preprocessing, cross_validation, neighbors
import pandas as pd

df = pd.read_csv('breast-cancer-wisconsin.data.txt')
df.replace('?', -99999, inplace=True)
df.drop(['id'], 1, inplace=True)
X = np.array(df.drop(['class'], 1))
y = np.array(df['class'])
X_train, X_test, y_train, y_test = cross_validation.train_test_split
clf = neighbors.KNeighborsClassifier()

clf.fit(X_train, y_train)
confidence = clf.score(X_test, y_test)
print(confidence)
example_measures = np.array([[4,2,1,1,1,2,3,2,1]])
example_measures = example_measures.reshape(len(example_measures), -
prediction = clf.predict(example_measures)
print(prediction)
```

1 CPU Python ready 1 Kiter ready 1 pandas base (Python 3.8.8)

File Edit Search Source Run Debug Consoles Projects Tools View Help



C:\Users\Muratcan\Desktop\MachineLearningCourse.py

temp.py x untitle0.py* x MachineLearningCourse.py* x

```
141     a0 = -4; a1 = f(a0, clf.w, clf.b)
142     b0 = 4; b1 = f(b0, clf.w, clf.b)
143     pl.plot([a0,b0], [a1,b1], "k")
144     a0 = -4; a1 = f(a0, clf.w, clf.b, 1)
145     b0 = 4; b1 = f(b0, clf.w, clf.b, 1)
146     pl.plot([a0,b0], [a1,b1], "k--")
147     a0 = -4; a1 = f(a0, clf.w, clf.b, -1)
148     b0 = 4; b1 = f(b0, clf.w, clf.b, -1)
149     pl.plot([a0,b0], [a1,b1], "k--")
150     pl.axis("tight")
151     pl.show()
152
153     def plot_contour(X1_train, X2_train, clf):
154         pl.plot(X1_train[:,0], X1_train[:,1], "ro")
155         pl.plot(X2_train[:,0], X2_train[:,1], "bo")
156         pl.scatter(clf.sv[:,0], clf.sv[:,1], s=100, c="g")
157         X1, X2 = np.meshgrid(np.linspace(-6,6,50), np.linspace(-6
158         X = np.array([[x1, x2] for x1, x2 in zip(np.ravel(X1), np
159         Z = clf.project(X).reshape(X1.shape)
160         pl.contour(X1, X2, Z, [0.0], colors='k', linewidths=1, or
161         pl.contour(X1, X2, Z + 1, [0.0], colors='grey', linewidth
162         pl.contour(X1, X2, Z - 1, [0.0], colors='grey', linewidth
163         pl.axis("tight")
164         pl.show()
165
166     def test_linear():
167         X1, y1, X2, y2 = gen_lin_separable_data()
168         X_train, y_train = split_train(X1, y1, X2, y2)
169         X_test, y_test = split_test(X1, y1, X2, y2)
170         clf = SVM()
171         clf.fit(X_train, y_train)
172         y_predict = clf.predict(X_test)
173         correct = np.sum(y_predict == y_test)
174         print("%d out of %d predictions correct" % (correct, len(
175         plot_margin(X_train[y_train==1], X_train[y_train==-1], cl
176
177     def test_non_linear():
178         X1, y1, X2, y2 = gen_non_lin_separable_data()
179         X_train, y_train = split_train(X1, y1, X2, y2)
180         X_test, y_test = split_test(X1, y1, X2, y2)
181         clf = SVM(polynomial_kernel)
182         clf.fit(X_train, y_train)
183         y_predict = clf.predict(X_test)
184         correct = np.sum(y_predict == y_test)
185         print("%d out of %d predictions correct" % (correct, len(
186         plot_contour(X_train[y_train==1], X_train[y_train==-1], c
187
188     def test_soft():
189         X1, y1, X2, y2 = gen_lin_separable_overlap_data()
190         X_train, y_train = split_train(X1, y1, X2, y2)
191         X_test, y_test = split_test(X1, y1, X2, y2)
192         clf = SVM(C=1000.1)
193         clf.fit(X_train, y_train)
194         y_predict = clf.predict(X_test)
195         correct = np.sum(y_predict == y_test)
196         print("%d out of %d predictions correct" % (correct, len(
197         plot_contour(X_train[y_train==1], X_train[y_train==-1], c
198
199     test_soft()
```



