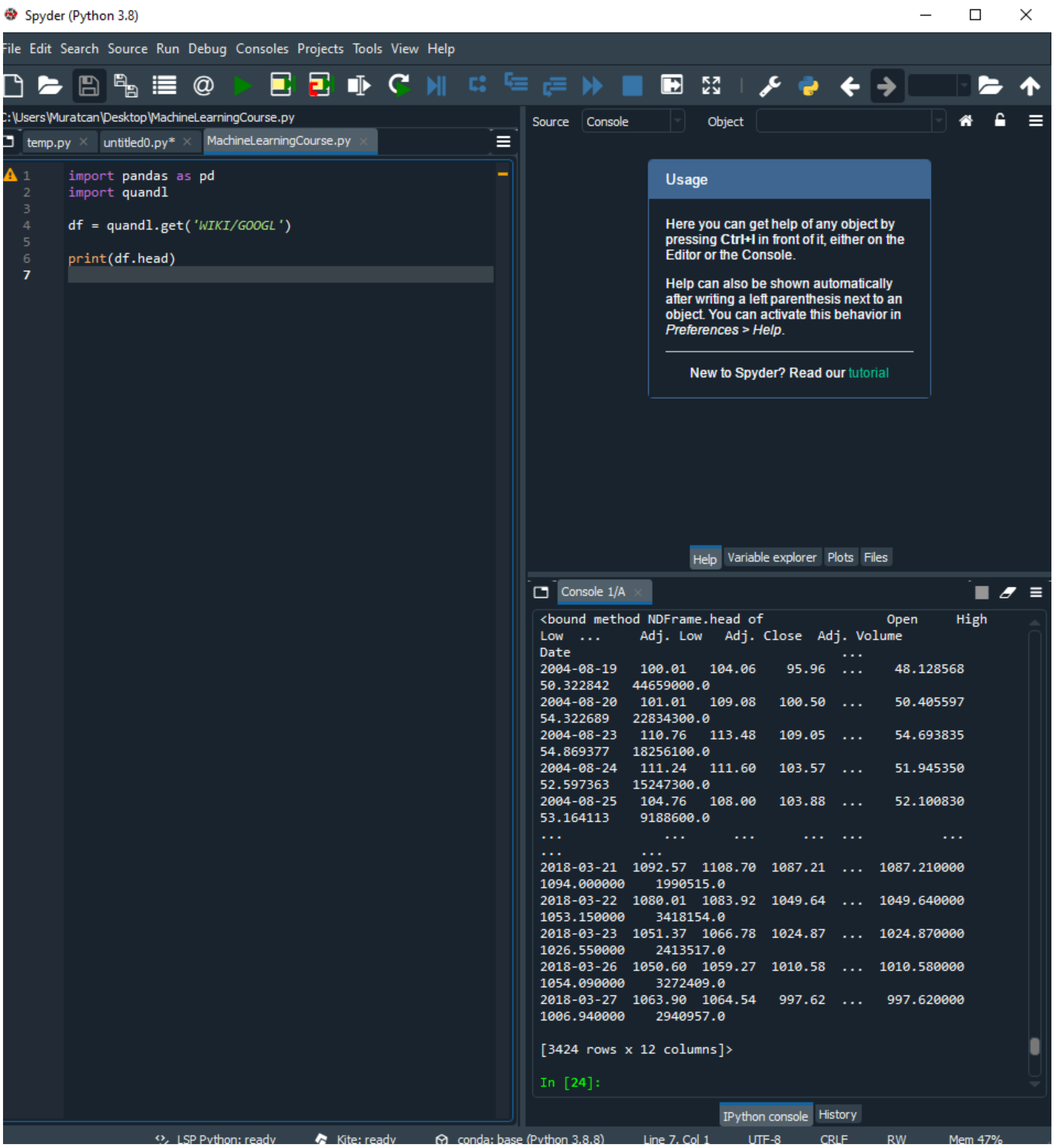
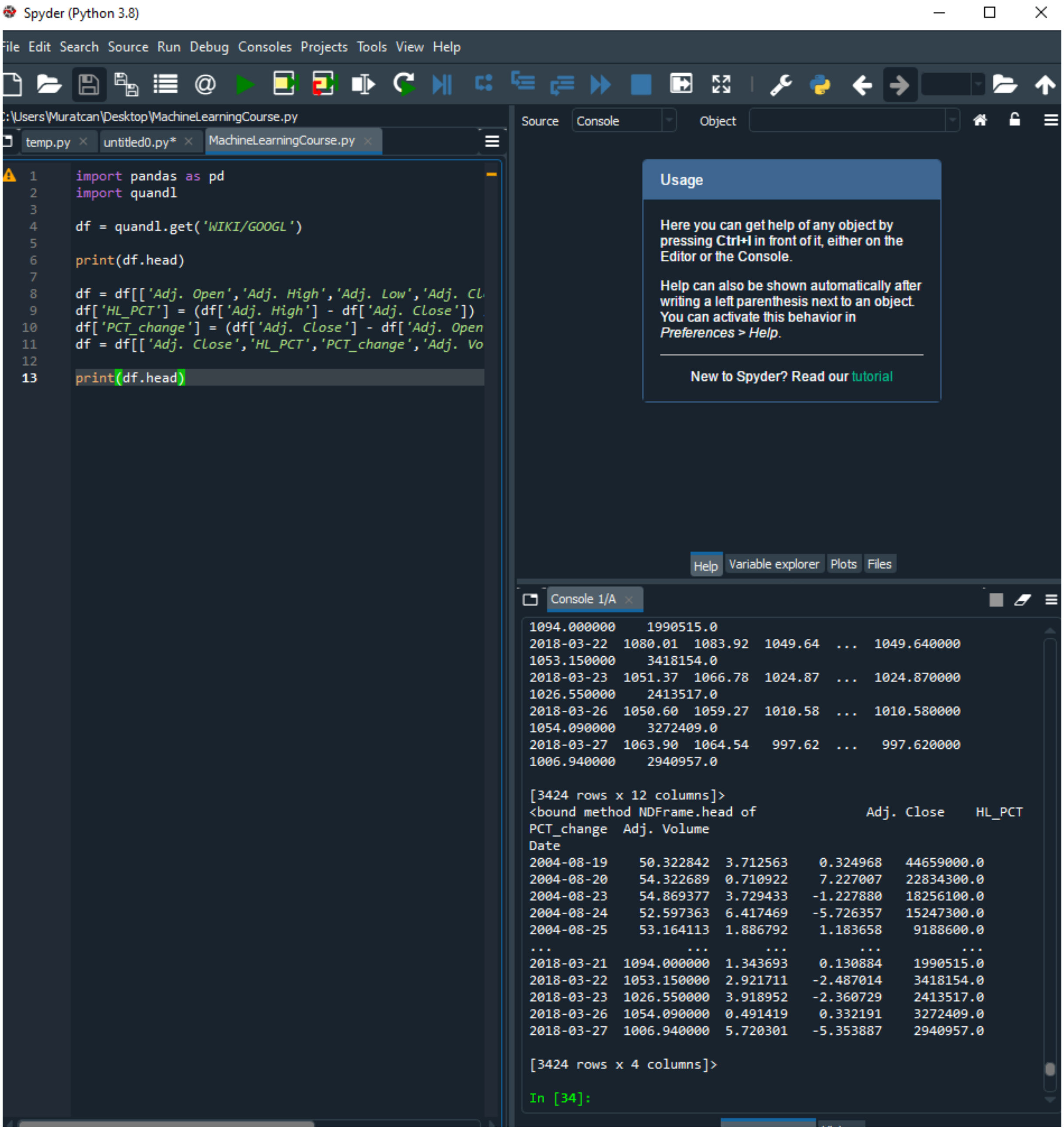


Spyder (Python 3.8)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\Muratcan\Desktop\MachineLearningCourse.py
temp.py
untitled0.py*
MachineLearningCourse.py
import pandas as pd
import quandl
df = quandl.get('WIKI/GOOGL')
print(df.head)
Source
Console
Object
Usage
Here you 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 Open High
Low ... Adj. Low Adj. Close Adj. Volume
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
LSP Python: 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
C:\Users\Muratcan\Desktop\MachineLearningCourse.py
temp.py
untitled0.py*
MachineLearningCourse.py
import pandas as pd
import quandl
df = quandl.get('WIKI/GOOGL')
print(df.head)
df = df[['Adj. Open','Adj. High','Adj. Low','Adj. Cl
df['HL_PCT'] = (df['Adj. High'] - df['Adj. Close'])
df['PCT_change'] = (df['Adj. Close'] - df['Adj. Open
df = df[['Adj. Close','HL_PCT','PCT_change','Adj. Vo
print(df.head)
Source
Console
Object
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
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* ×

```python
import pandas as pd
import quandl
import math

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

print(df.head)

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

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())
```

s\Muratcan\Desktop\MachineLearningCourse.py

np.py × untitled0.py* × MachineLearningCourse.py ×

```python
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. Volum
df['HL_PCT'] = (df['Adj. High'] - df['Adj. Close']) / df['Adj. Close
df['PCT_change'] = (df['Adj. Close'] - df['Adj. Open']) / df['Adj. O
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

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)

s\Muratcan\Desktop\MachineLearningCourse.py

mp.py | untitled0.py* | MachineLearningCourse.py

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

print(df.head)

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

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()
```

Spyder (Python 3.8)

ile Edit Search Source Run Debug Consoles Projects Tools View Help

:\Users\Muratcan\Desktop\MachineLearningCourse.py

temp.py × untitled0.py* × MachineLearningCourse.py ×

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

x_train, x_test, y_train, y_test = cross_validation.train_test_spl

clf = LinearRegression(n_jobs=-1)
clf.fit(x_train, y_train)

with open('linearregression.pickle','wb') as f:
    pickle.dump(clf, f)

pickle_in = open('linearregression.pickle','rb')
clf = pickle.load(pickle_in)


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()
```

```python
from statistics import mean
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import style
style.use('fivethirtyeight')

xs = np.array([1,2,3,4,5,6], dtype = np.float64)
ys = np.array([5,4,6,5,6,7], dtype = np.float64)

    def best_fit_slope_and_intercept(xs,ys):
        m = ( ((mean(xs) * mean(ys))  -  mean(xs*ys)) / ((mean(x
        b = mean(ys) - m*mean(xs)
        return m, b
m,b = best_fit_slope_and_intercept(xs,ys)

regression_line = [(m*x)+b for x in xs]

plt.scatter(xs,ys)
plt.plot(xs, regression)
```

LSP Python: ready   Kite: ready   conda: base (Python 3.8

›pyder (Python 3.8)

Edit Search Source Run Debug Consoles Projects Tools View Help

ers\Muratcan\Desktop\MachineLearningCourse.py

temp.py × untitled0.py* × MachineLearningCourse.py* ×

```python
#     next_date = datetime.datetime.fromtimestamp(next_unix)
#     next_unix += one_day
#     df.loc[next_date] = [np.nan for _ in range(len(df.columns)-
#
# df['Adj. Close'].plot()
# df['Forecast'].plot()
# plt.legend(loc=4)
# plt.xlabel('Date')
# plt.ylabel('Price')
# plt.show()
# ===============================================================


from statistics import mean
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import style
style.use('fivethirtyeight')

xs = np.array([1,2,3,4,5,6], dtype = np.float64)
ys = np.array([5,4,6,5,6,7], dtype = np.float64)

    def best_fit_slope_and_intercept(xs,ys):
        m = ( ((mean(xs) * mean(ys))  -  mean(xs*ys)) / ((mean(xs
        b = mean(ys) - m*mean(xs)
        return m, b


def squared_error(ys_orig, ys_line):
    return sum((ys_line-ys_orig**2))

def coefficient_of_determination(ys_orig,ys_line):
    y_mean_line = [mean(ys_orig) for y in ys_orig]
    squared_error_regr = squared_error(ys_orig,ys_line)
    squared_error_y_mean = squared_error(ys_orig,y_mean_line)
    return 1 - (squared_error_regr / squared_error_y_mean)

m,b = best_fit_slope_and_intercept(xs,ys)

regression_line = [(m*x)+b for x in xs]

predict_x = 8
predict_y = (m*predict_x) + b

r_squared = coefficient_of_determination(ys,regression_line)
print(r_squared)


plt.scatter(xs,ys)
plt.scatter(predict_x,predict_y,color = 'g')
plt.plot(xs, regression_line)
plt.show()
```

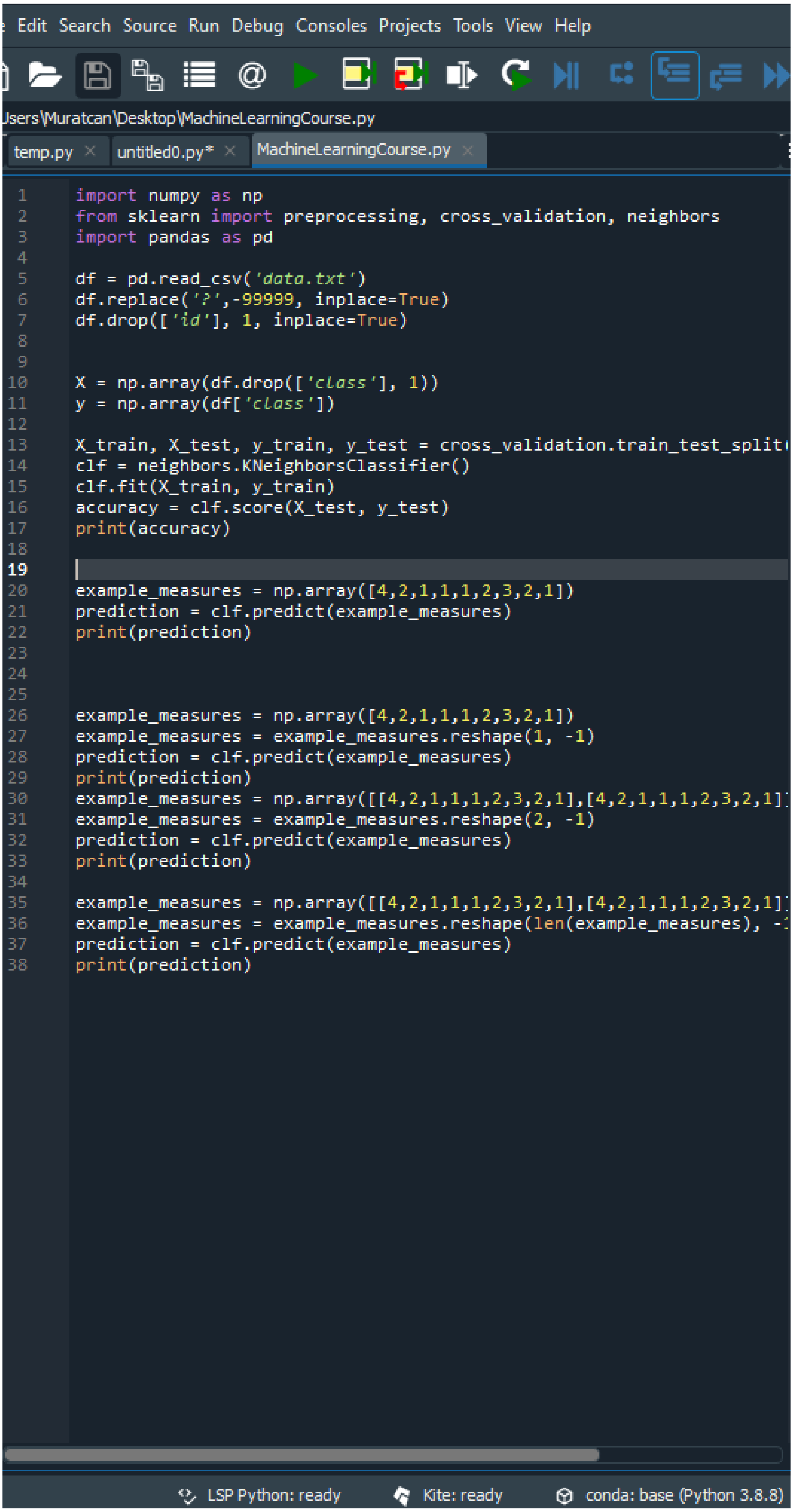
Spyder (Python 3.8)

Edit Search Source Run Debug Consoles Projects Tools View Help

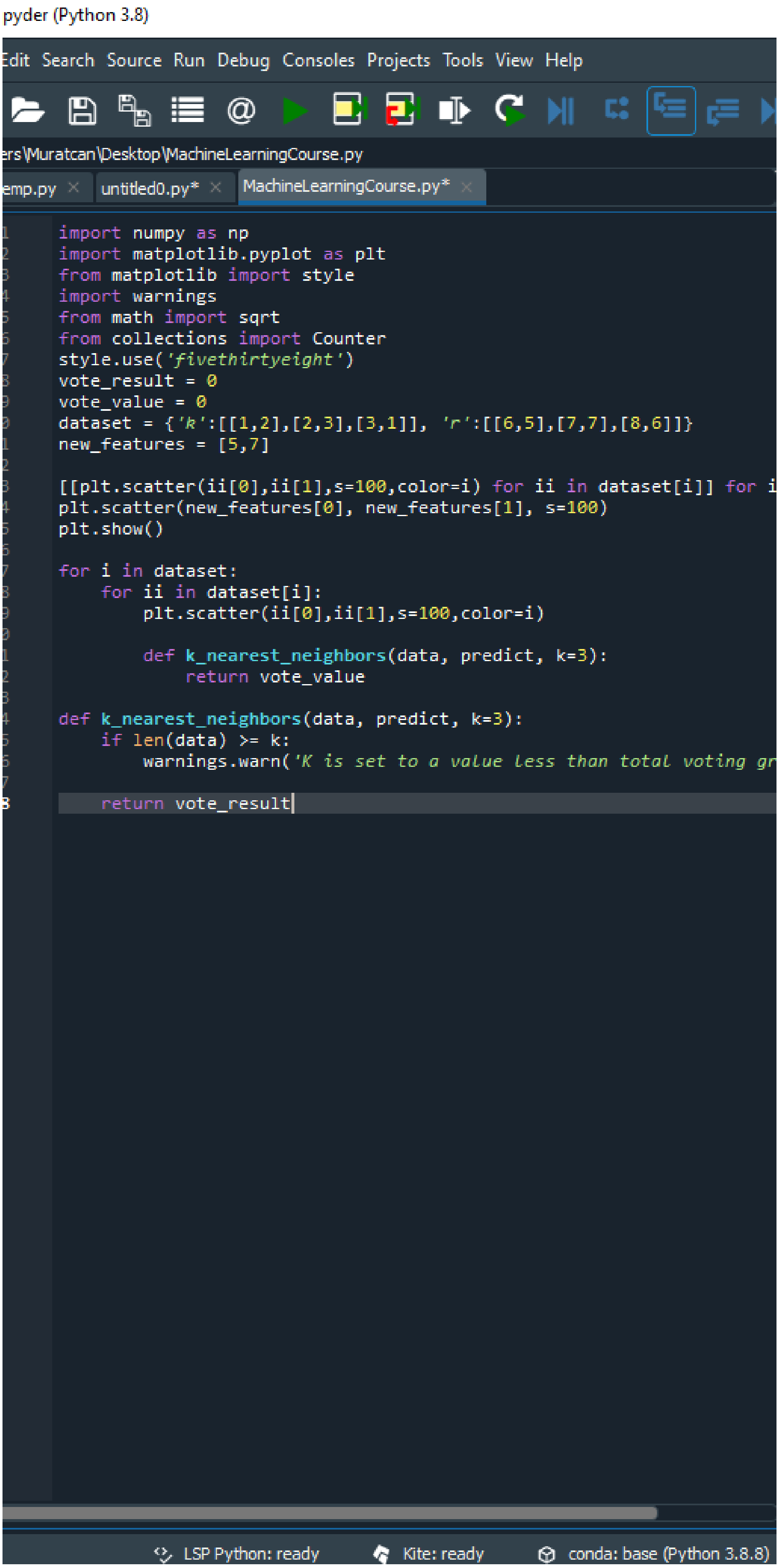
sers\Muratcan\Desktop\MachineLearningCourse.py

temp.py × untitled0.py* × MachineLearningCourse.py ×

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


from statistics import mean
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import style
style.use('fivethirtyeight')

xs = np.array([1,2,3,4,5,6], dtype = np.float64)
ys = np.array([5,4,6,5,6,7], dtype = np.float64)

    def best_fit_slope_and_intercept(xs,ys):
        m = ( ((mean(xs) * mean(ys))  -  mean(xs*ys)) / ((mean(xs
        b = mean(ys) - m*mean(xs)
        return m, b


def squared_error(ys_orig, ys_line):
    return sum((ys_line-ys_orig**2))

def coefficient_of_determination(ys_orig,ys_line):
    y_mean_line = [mean(ys_orig) for y in ys_orig]
    squared_error_regr = squared_error(ys_orig,ys_line)
    squared_error_y_mean = squared_error(ys_orig,y_mean_line)
    return 1 - (squared_error_regr / squared_error_y_mean)

m,b = best_fit_slope_and_intercept(xs,ys)

regression_line = [(m*x)+b for x in xs]

predict_x = 8
predict_y = (m*predict_x) + b

r_squared = coefficient_of_determination(ys,regression_line)
print(r_squared)


plt.scatter(xs,ys)
plt.scatter(predict_x,predict_y,color = 'g')
plt.plot(xs, regression_line)
plt.show()
```




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

df = pd.read_csv('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)
accuracy = clf.score(X_test, y_test)
print(accuracy)


example_measures = np.array([4,2,1,1,1,2,3,2,1])
prediction = clf.predict(example_measures)
print(prediction)



example_measures = np.array([4,2,1,1,1,2,3,2,1])
example_measures = example_measures.reshape(1, -1)
prediction = clf.predict(example_measures)
print(prediction)
example_measures = np.array([[4,2,1,1,1,2,3,2,1],[4,2,1,1,1,2,3,2,1]]
example_measures = example_measures.reshape(2, -1)
prediction = clf.predict(example_measures)
print(prediction)

example_measures = np.array([[4,2,1,1,1,2,3,2,1],[4,2,1,1,1,2,3,2,1]]
example_measures = example_measures.reshape(len(example_measures), -1
prediction = clf.predict(example_measures)
print(prediction)
```




```python
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import style
import warnings
from math import sqrt
from collections import Counter
style.use('fivethirtyeight')
vote_result = 0
vote_value = 0
dataset = {'k':[[1,2],[2,3],[3,1]], 'r':[[6,5],[7,7],[8,6]]}
new_features = [5,7]

[[plt.scatter(ii[0],ii[1],s=100,color=i) for ii in dataset[i]] for i
plt.scatter(new_features[0], new_features[1], s=100)
plt.show()

for i in dataset:
    for ii in dataset[i]:
        plt.scatter(ii[0],ii[1],s=100,color=i)

        def k_nearest_neighbors(data, predict, k=3):
            return vote_value

def k_nearest_neighbors(data, predict, k=3):
    if len(data) >= k:
        warnings.warn('K is set to a value less than total voting gr

    return vote_result
```

pyder (Python 3.8)

Edit Search Source Run Debug Consoles Projects Tools View Help

ers\Muratcan\Desktop\MachineLearningCourse.py

emp.py × untitled0.py* × MachineLearningCourse.py* ×

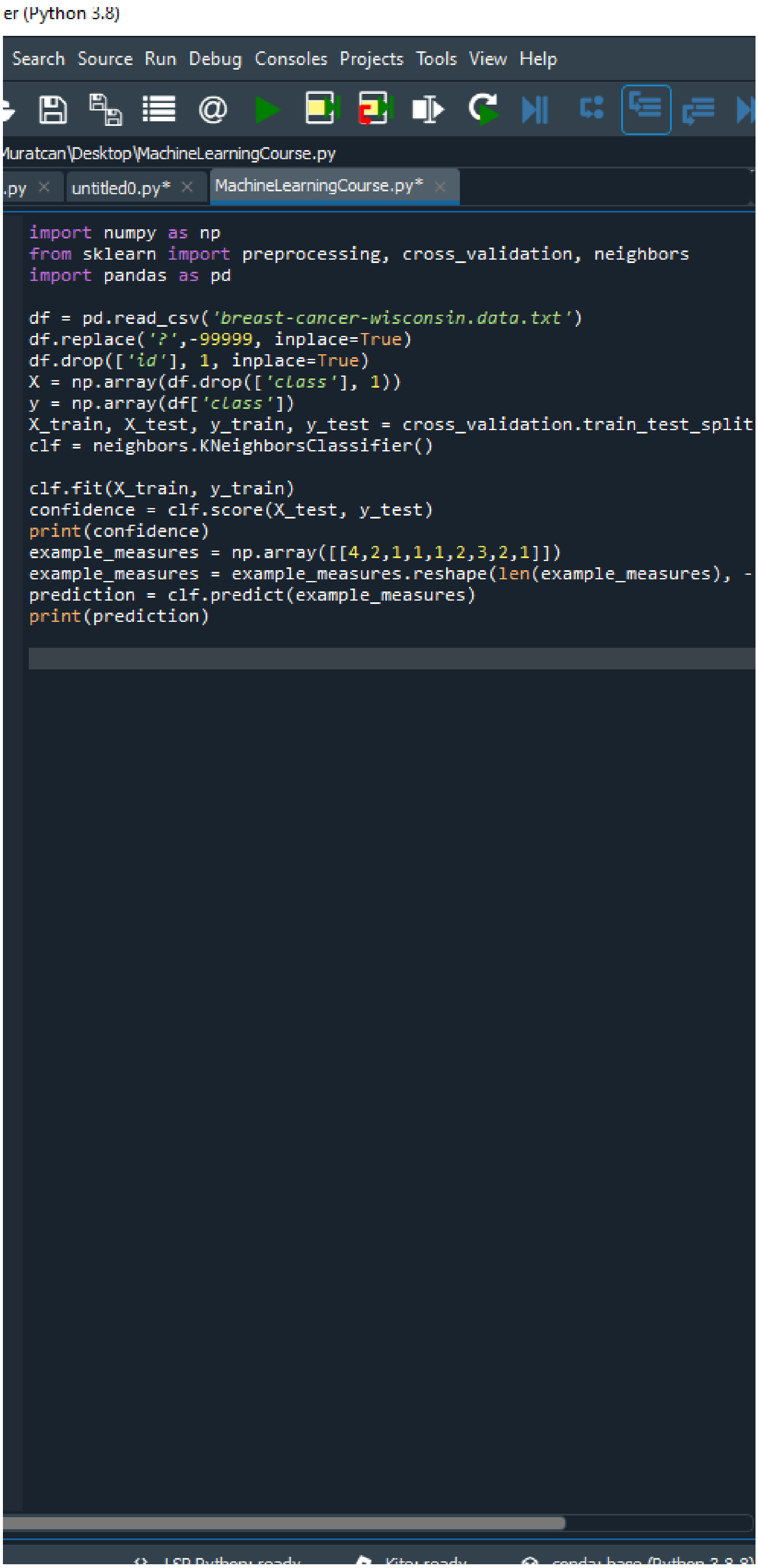
```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import style
import warnings
from math import sqrt
from collections import Counter
style.use('fivethirtyeight')
vote_result = 0
vote_value = 0
dataset = {'k':[[1,2],[2,3],[3,1]], 'r':[[6,5],[7,7],[8,6]]}
new_features = [5,7]

[[plt.scatter(ii[0],ii[1],s=100,color=i) for ii in dataset[i]] for i
plt.scatter(new_features[0], new_features[1], s=100)
plt.show()

for i in dataset:
    for ii in dataset[i]:
        plt.scatter(ii[0],ii[1],s=100,color=i)

        def k_nearest_neighbors(data, predict, k=3):
            return vote_value

def k_nearest_neighbors(data, predict, k=3):
    if len(data) >= k:
        warnings.warn('K is set to a value less than total voting gr

    return vote_result
```

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

temp.py × untitled0.py* × MachineLearningCourse.py* ×

```python
import numpy as np
from math import sqrt
import warnings
from collections import Counter
import pandas as pd
import random

def k_nearest_neighbors(data, predict, k=3):
    if len(data) >= k:
        warnings.warn('K is set to a value less than total voting gr
    distances = []
    for group in data:
        for features in data[group]:
            euclidean_distance = np.linalg.norm(np.array(features)-n
            distances.append([euclidean_distance, group])

    votes = [i[1] for i in sorted(distances)[:k]]
    vote_result = Counter(votes).most_common(1)[0][0]
    confidence = Counter(votes).most_common(1)[0][1] / k

    return vote_result, confidence
df = pd.read_csv("breast-cancer-wisconsin.data.txt")
df.replace('?',-99999, inplace=True)
df.drop(['id'], 1, inplace=True)
full_data = df.astype(float).values.tolist()
random.shuffle(full_data)
test_size = 0.4
train_set = {2:[], 4:[]}
test_set = {2:[], 4:[]}
train_data = full_data[:-int(test_size*len(full_data))]
test_data = full_data[-int(test_size*len(full_data)):]

for i in train_data:
    train_set[i[-1]].append(i[:-1])

for i in test_data:
    test_set[i[-1]].append(i[:-1])

correct = 0
total = 0
for group in test_set:
    for data in test_set[group]:
        vote,confidence = k_nearest_neighbors(train_set, data, k=5)
        if group == vote:
            correct += 1
        total += 1
print('Accuracy:', correct/total)
```

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

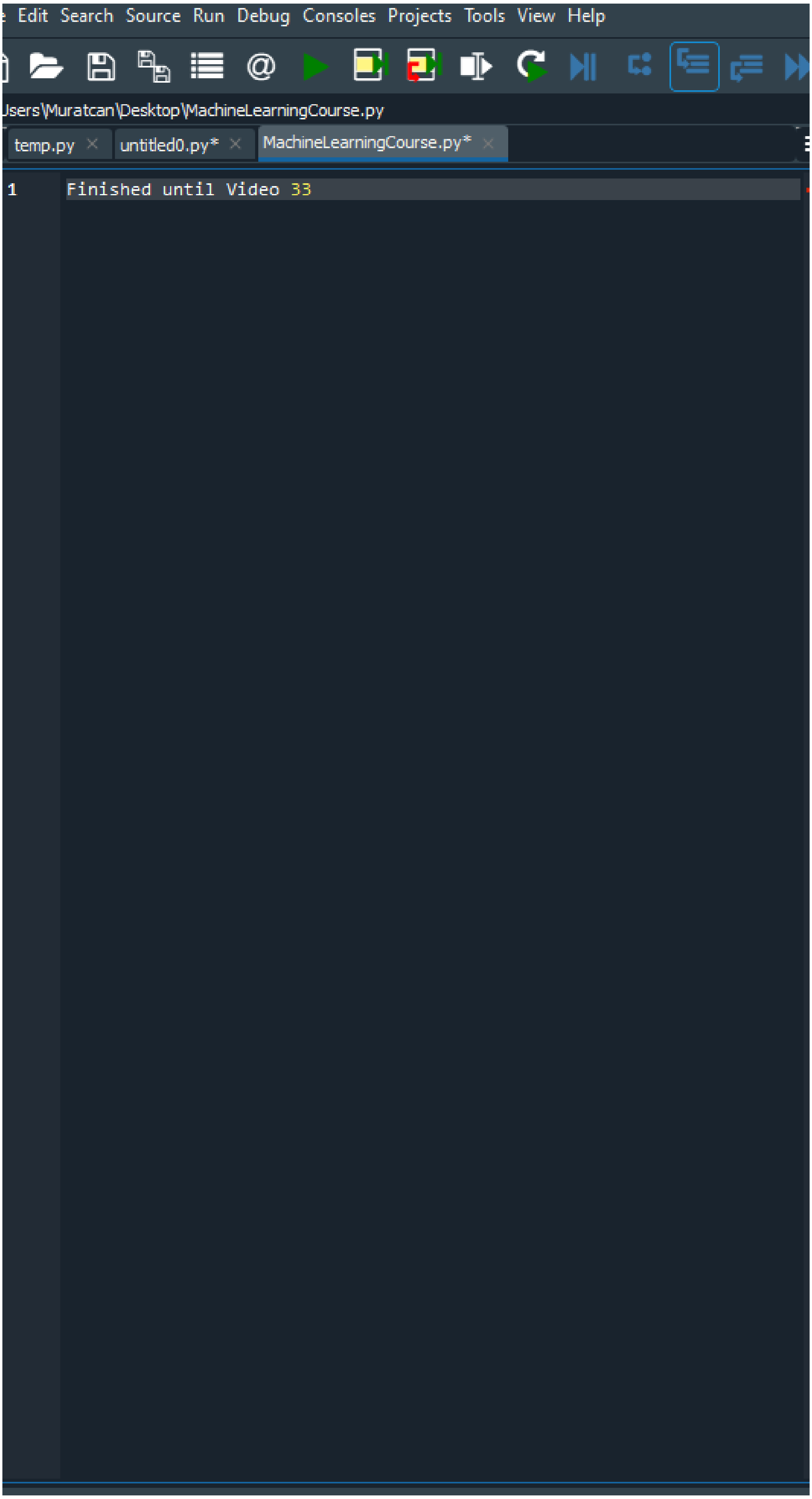

er (Python 3.8)
Search Source Run Debug Consoles Projects Tools View Help
uratcan\Desktop\MachineLearningCourse.py
.py
untitled0.py*
MachineLearningCourse.py*
import numpy as np
from sklearn import preprocessing, cross_validation, neighbors
import pandas as pd
df = pd.read_csv('breast-cancer-wisconsin.data.txt')
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)

File Edit Search Source Run Debug Consoles Projects Tools View Help

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

temp.py × untitled0.py* × MachineLearningCourse.py* ×

```
        a0 = -4; a1 = f(a0, clf.w, clf.b)
        b0 = 4; b1 = f(b0, clf.w, clf.b)
        pl.plot([a0,b0], [a1,b1], "k")
        a0 = -4; a1 = f(a0, clf.w, clf.b, 1)
        b0 = 4; b1 = f(b0, clf.w, clf.b, 1)
        pl.plot([a0,b0], [a1,b1], "k--")
        a0 = -4; a1 = f(a0, clf.w, clf.b, -1)
        b0 = 4; b1 = f(b0, clf.w, clf.b, -1)
        pl.plot([a0,b0], [a1,b1], "k--")
        pl.axis("tight")
        pl.show()

    def plot_contour(X1_train, X2_train, clf):
        pl.plot(X1_train[:,0], X1_train[:,1], "ro")
        pl.plot(X2_train[:,0], X2_train[:,1], "bo")
        pl.scatter(clf.sv[:,0], clf.sv[:,1], s=100, c="g")
        X1, X2 = np.meshgrid(np.linspace(-6,6,50), np.linspace(-6
        X = np.array([[x1, x2] for x1, x2 in zip(np.ravel(X1), np
        Z = clf.project(X).reshape(X1.shape)
        pl.contour(X1, X2, Z, [0.0], colors='k', linewidths=1, or
        pl.contour(X1, X2, Z + 1, [0.0], colors='grey', linewidth
        pl.contour(X1, X2, Z - 1, [0.0], colors='grey', linewidth
        pl.axis("tight")
        pl.show()

    def test_linear():
        X1, y1, X2, y2 = gen_lin_separable_data()
        X_train, y_train = split_train(X1, y1, X2, y2)
        X_test, y_test = split_test(X1, y1, X2, y2)
        clf = SVM()
        clf.fit(X_train, y_train)
        y_predict = clf.predict(X_test)
        correct = np.sum(y_predict == y_test)
        print("%d out of %d predictions correct" % (correct, len(
        plot_margin(X_train[y_train==1], X_train[y_train==-1], cl

    def test_non_linear():
        X1, y1, X2, y2 = gen_non_lin_separable_data()
        X_train, y_train = split_train(X1, y1, X2, y2)
        X_test, y_test = split_test(X1, y1, X2, y2)
        clf = SVM(polynomial_kernel)
        clf.fit(X_train, y_train)
        y_predict = clf.predict(X_test)
        correct = np.sum(y_predict == y_test)
        print("%d out of %d predictions correct" % (correct, len(
        plot_contour(X_train[y_train==1], X_train[y_train==-1], c

    def test_soft():
        X1, y1, X2, y2 = gen_lin_separable_overlap_data()
        X_train, y_train = split_train(X1, y1, X2, y2)
        X_test, y_test = split_test(X1, y1, X2, y2)
        clf = SVM(C=1000.1)
        clf.fit(X_train, y_train)
        y_predict = clf.predict(X_test)
        correct = np.sum(y_predict == y_test)
        print("%d out of %d predictions correct" % (correct, len(
        plot_contour(X_train[y_train==1], X_train[y_train==-1], c

test_soft()
```


Edit Search Source Run Debug Consoles Projects Tools View Help
Users\Muratcan\Desktop\MachineLearningCourse.py
temp.py
untitled0.py*
MachineLearningCourse.py*
1 Finished until Video 33