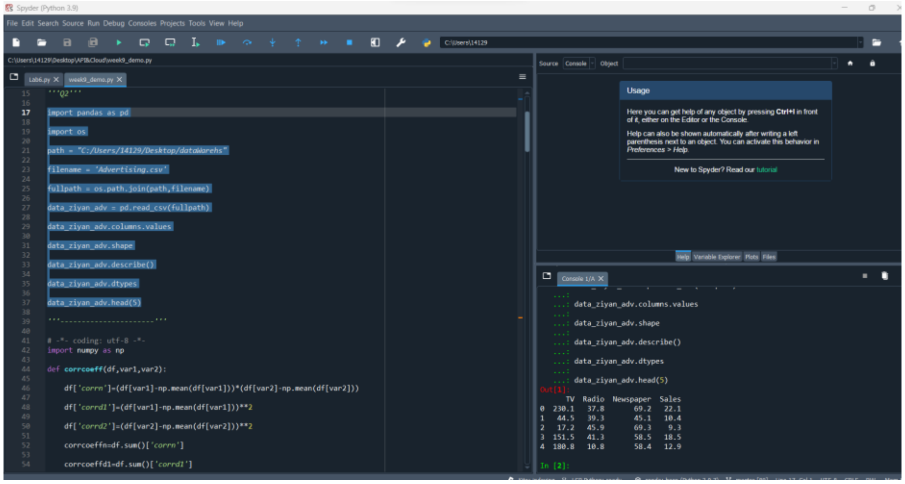
# Summary:

---



```python
'''Q2'''

import pandas as pd

import os

path = "C:/Users/14129/Desktop/dataWarehs"

filename = 'Advertising.csv'

fullpath = os.path.join(path,filename)

data_ziyan_adv = pd.read_csv(fullpath)

data_ziyan_adv.columns.values

data_ziyan_adv.shape

data_ziyan_adv.describe()

data_ziyan_adv.dtypes

data_ziyan_adv.head(5)

'''---------------------'''

# -*- coding: utf-8 -*-
import numpy as np

def corrcoeff(df,var1,var2):

    df['corrn']=(df[var1]-np.mean(df[var1]))*(df[var2]-np.mean(df[var2]))

    df['corrd1']=(df[var1]-np.mean(df[var1]))**2

    df['corrd2']=(df[var2]-np.mean(df[var2]))**2

    corrcoeffn=df.sum()['corrn']

    corrcoeffd1=df.sum()['corrd1']
```

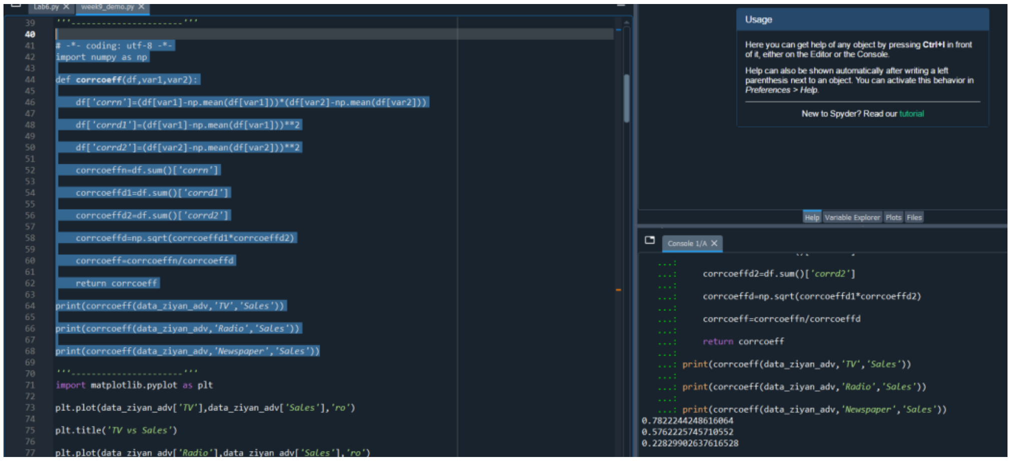
Usage

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

```
   ...:
   ...: data_ziyan_adv.columns.values
   ...:
   ...: data_ziyan_adv.shape
   ...:
   ...: data_ziyan_adv.describe()
   ...:
   ...: data_ziyan_adv.dtypes
   ...:
   ...: data_ziyan_adv.head(5)
Out[1]:
      TV  Radio  Newspaper  Sales
0  230.1   37.8       69.2   22.1
1   44.5   39.3       45.1   10.4
2   17.2   45.9       69.3    9.3
3  151.5   41.3       58.5   18.5
4  180.8   10.8       58.4   12.9

In [2]:
```



```python
'''---------------------'''

# -*- coding: utf-8 -*-
import numpy as np

def corrcoeff(df,var1,var2):

    df['corrn']=(df[var1]-np.mean(df[var1]))*(df[var2]-np.mean(df[var2]))

    df['corrd1']=(df[var1]-np.mean(df[var1]))**2

    df['corrd2']=(df[var2]-np.mean(df[var2]))**2

    corrcoeffn=df.sum()['corrn']

    corrcoeffd1=df.sum()['corrd1']

    corrcoeffd2=df.sum()['corrd2']

    corrcoeffd=np.sqrt(corrcoeffd1*corrcoeffd2)

    corrcoeff=corrcoeffn/corrcoeffd

    return corrcoeff

print(corrcoeff(data_ziyan_adv,'TV','Sales'))

print(corrcoeff(data_ziyan_adv,'Radio','Sales'))

print(corrcoeff(data_ziyan_adv,'Newspaper','Sales'))

'''---------------------'''
import matplotlib.pyplot as plt

plt.plot(data_ziyan_adv['TV'],data_ziyan_adv['Sales'],'ro')

plt.title('TV vs Sales')

plt.plot(data_ziyan_adv['Radio'],data_ziyan_adv['Sales'],'ro')
```

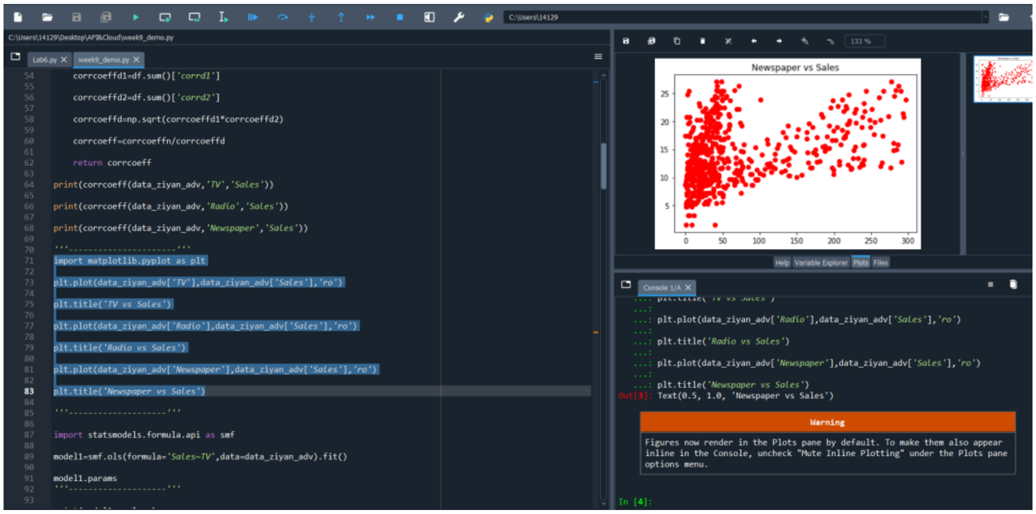
Usage

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*.

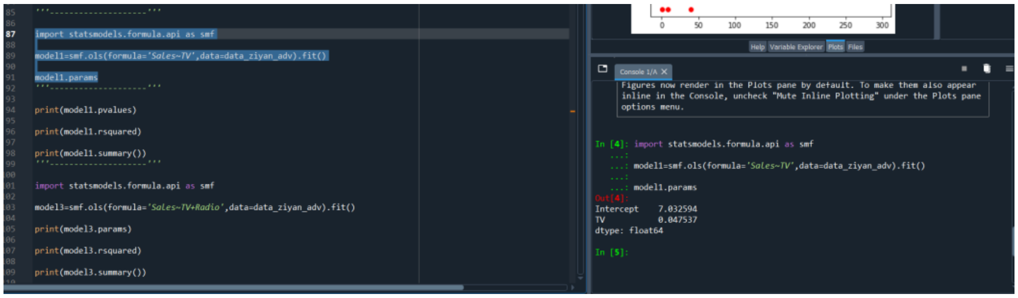
New to Spyder? Read our tutorial

```
   ...:
   ...:     corrcoeffd2=df.sum()['corrd2']
   ...:
   ...:     corrcoeffd=np.sqrt(corrcoeffd1*corrcoeffd2)
   ...:
   ...:     corrcoeff=corrcoeffn/corrcoeffd
   ...:
   ...:     return corrcoeff
   ...:
   ...: print(corrcoeff(data_ziyan_adv,'TV','Sales'))
   ...:
   ...: print(corrcoeff(data_ziyan_adv,'Radio','Sales'))
   ...:
   ...: print(corrcoeff(data_ziyan_adv,'Newspaper','Sales'))
0.7822244248616064
0.5762225745710552
0.22829902637616528
```



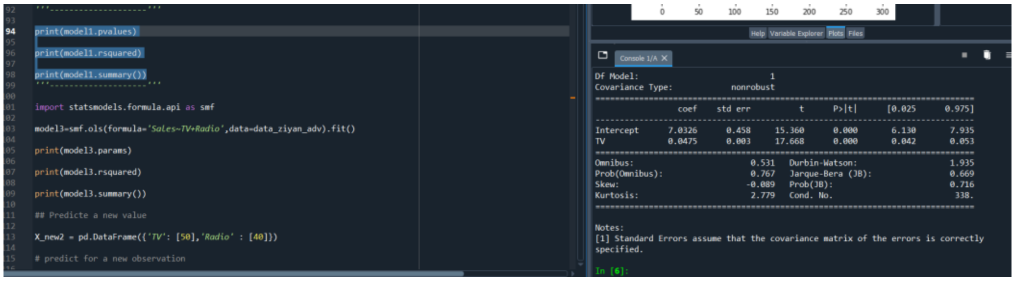
```python
    corrcoeffd1=df.sum()['corrd1']

    corrcoeffd2=df.sum()['corrd2']

    corrcoeffd=np.sqrt(corrcoeffd1*corrcoeffd2)

    corrcoeff=corrcoeffn/corrcoeffd

    return corrcoeff

print(corrcoeff(data_ziyan_adv,'TV','Sales'))

print(corrcoeff(data_ziyan_adv,'Radio','Sales'))

print(corrcoeff(data_ziyan_adv,'Newspaper','Sales'))
'''---------------------'''
import matplotlib.pyplot as plt

plt.plot(data_ziyan_adv['TV'],data_ziyan_adv['Sales'],'ro')

plt.title('TV vs Sales')

plt.plot(data_ziyan_adv['Radio'],data_ziyan_adv['Sales'],'ro')

plt.title('Radio vs Sales')

plt.plot(data_ziyan_adv['Newspaper'],data_ziyan_adv['Sales'],'ro')

plt.title('Newspaper vs Sales')

'''------------------'''

import statsmodels.formula.api as smf

model1=smf.ols(formula='Sales~TV',data=data_ziyan_adv).fit()

model1.params
'''------------------'''
```

```
   ...:
   ...: plt.plot(data_ziyan_adv['Radio'],data_ziyan_adv['Sales'],'ro')
   ...:
   ...: plt.title('Radio vs Sales')
   ...:
   ...: plt.plot(data_ziyan_adv['Newspaper'],data_ziyan_adv['Sales'],'ro')
   ...:
   ...: plt.title('Newspaper vs Sales')
Out[3]: Text(0.5, 1.0, 'Newspaper vs Sales')

                              Warning
 Figures now render in the Plots pane by default. To make them also appear
 inline in the Console, uncheck "Mute Inline Plotting" under the Plots pane
 options menu.

In [4]:
```



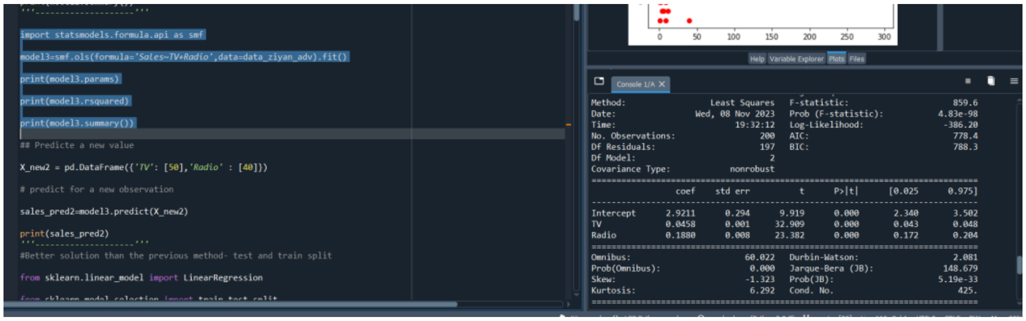
```python
'''--------------------'''

import statsmodels.formula.api as smf

model1=smf.ols(formula='Sales~TV',data=data_ziyan_adv).fit()

model1.params
'''--------------------'''

print(model1.pvalues)

print(model1.rsquared)

print(model1.summary())
'''--------------------'''

import statsmodels.formula.api as smf

model3=smf.ols(formula='Sales~TV+Radio',data=data_ziyan_adv).fit()

print(model3.params)

print(model3.rsquared)

print(model3.summary())
```

```
    Figures now render in the Plots pane by default. To make them also appear
    inline in the Console, uncheck "Mute Inline Plotting" under the Plots pane
    options menu.


In [4]: import statsmodels.formula.api as smf
   ...:
   ...: model1=smf.ols(formula='Sales~TV',data=data_ziyan_adv).fit()
   ...:
   ...: model1.params
Out[4]:
Intercept    7.032594
TV           0.047537
dtype: float64

In [5]:
```



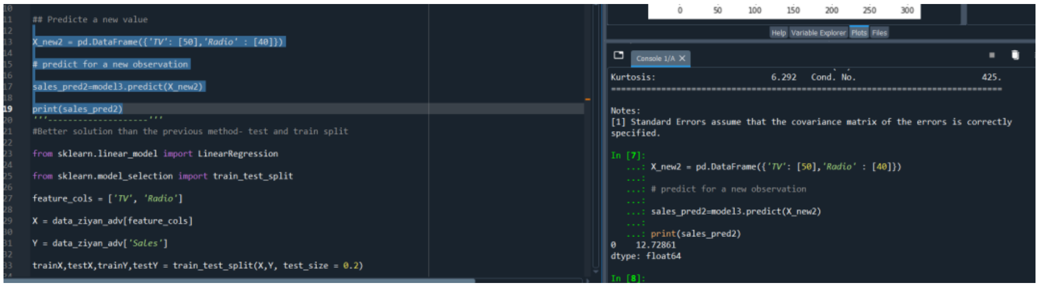
```python
'''--------------------'''

print(model1.pvalues)

print(model1.rsquared)

print(model1.summary())
'''--------------------'''

import statsmodels.formula.api as smf

model3=smf.ols(formula='Sales~TV+Radio',data=data_ziyan_adv).fit()

print(model3.params)

print(model3.rsquared)

print(model3.summary())

## Predicte a new value

X_new2 = pd.DataFrame({'TV': [50],'Radio' : [40]})

# predict for a new observation
```

```
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept      7.0326      0.458     15.360      0.000       6.130       7.935
TV             0.0475      0.003     17.668      0.000       0.042       0.053
==============================================================================
Omnibus:                        0.531   Durbin-Watson:                   1.935
Prob(Omnibus):                  0.767   Jarque-Bera (JB):                0.669
Skew:                          -0.089   Prob(JB):                        0.716
Kurtosis:                       2.779   Cond. No.                         338.
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.

In [6]:
```



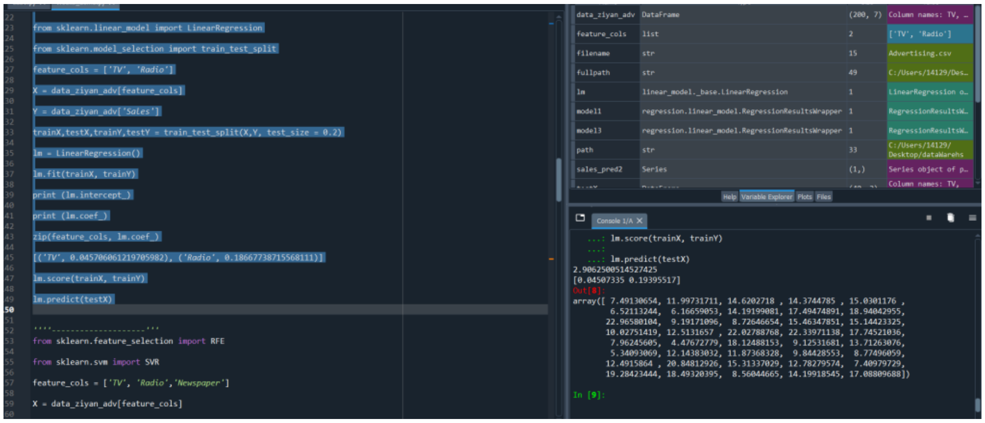
```python
'''--------------------'''

import statsmodels.formula.api as smf

model3=smf.ols(formula='Sales~TV+Radio',data=data_ziyan_adv).fit()

print(model3.params)

print(model3.rsquared)

print(model3.summary())

## Predicte a new value

X_new2 = pd.DataFrame({'TV': [50],'Radio' : [40]})

# predict for a new observation

sales_pred2=model3.predict(X_new2)

print(sales_pred2)
'''--------------------'''
#Better solution than the previous method- test and train split

from sklearn.linear_model import LinearRegression
```

```
Method:                 Least Squares   F-statistic:                     859.6
Date:                Wed, 08 Nov 2023   Prob (F-statistic):           4.83e-98
Time:                        19:32:12   Log-Likelihood:                -386.20
No. Observations:                 200   AIC:                             778.4
Df Residuals:                     197   BIC:                             788.3
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept      2.9211      0.294      9.919      0.000       2.340       3.502
TV             0.0458      0.001     32.909      0.000       0.043       0.048
Radio          0.1880      0.008     23.382      0.000       0.172       0.204
==============================================================================
Omnibus:                       60.022   Durbin-Watson:                   2.081
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              148.679
Skew:                          -1.323   Prob(JB):                     5.19e-33
Kurtosis:                       6.292   Cond. No.                         425.
==============================================================================
```



```python
## Predicte a new value

X_new2 = pd.DataFrame({'TV': [50],'Radio' : [40]})

# predict for a new observation

sales_pred2=model3.predict(X_new2)

print(sales_pred2)
'''--------------------'''
#Better solution than the previous method- test and train split

from sklearn.linear_model import LinearRegression

from sklearn.model_selection import train_test_split

feature_cols = ['TV', 'Radio']

X = data_ziyan_adv[feature_cols]

Y = data_ziyan_adv['Sales']

trainX,testX,trainY,testY = train_test_split(X,Y, test_size = 0.2)
```

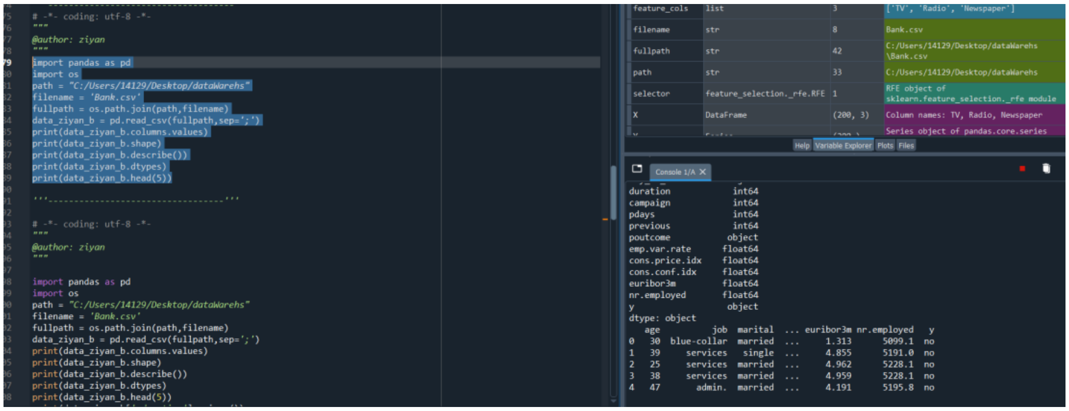
```
Kurtosis:                       6.292   Cond. No.                         425.
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.

In [7]:
   ...: X_new2 = pd.DataFrame({'TV': [50],'Radio' : [40]})
   ...:
   ...: # predict for a new observation
   ...:
   ...: sales_pred2=model3.predict(X_new2)
   ...:
   ...: print(sales_pred2)
0    12.72861
dtype: float64

In [8]:
```



```python

from sklearn.linear_model import LinearRegression

from sklearn.model_selection import train_test_split

feature_cols = ['TV', 'Radio']

X = data_ziyan_adv[feature_cols]

Y = data_ziyan_adv['Sales']

trainX,testX,trainY,testY = train_test_split(X,Y, test_size = 0.2)

lm = LinearRegression()

lm.fit(trainX, trainY)

print (lm.intercept_)

print (lm.coef_)

zip(feature_cols, lm.coef_)

[('TV', 0.045706061219705982), ('Radio', 0.18667738715568111)]

lm.score(trainX, trainY)

lm.predict(testX)


''''--------------------'''
from sklearn.feature_selection import RFE

from sklearn.svm import SVR

feature_cols = ['TV', 'Radio','Newspaper']

X = data_ziyan_adv[feature_cols]
```

| | | | |
|---|---|---|---|
| data_ziyan_adv | DataFrame | (200, 7) | Column names: TV, … |
| feature_cols | list | 2 | ['TV', 'Radio'] |
| filename | str | 15 | Advertising.csv |
| fullpath | str | 49 | C:/Users/14129/Des… |
| lm | linear_model._base.LinearRegression | 1 | LinearRegression o… |
| model1 | regression.linear_model.RegressionResultsWrapper | 1 | RegressionResultsW… |
| model3 | regression.linear_model.RegressionResultsWrapper | 1 | RegressionResultsW… |
| path | str | 33 | C:/Users/14129/<br>Desktop/dataWarehs |
| sales_pred2 | Series | (1,) | Series object of p… |

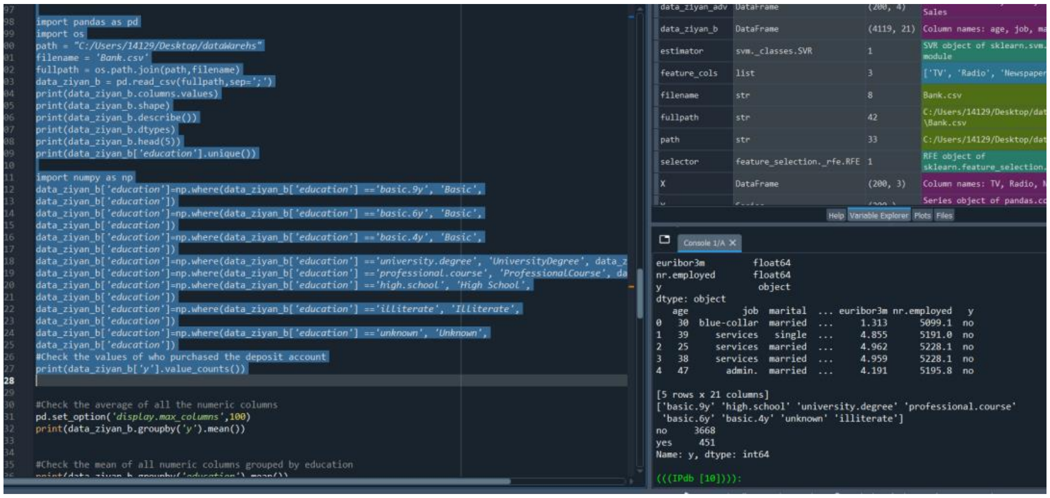
```
   ...: lm.score(trainX, trainY)
   ...:
   ...: lm.predict(testX)
2.9062500514527425
[0.04507335 0.19395517]
Out[8]:
array([ 7.49130654, 11.99731711, 14.6202718 , 14.3744785 , 15.0301176 ,
        6.52113244,  6.16659053, 14.19199081, 17.49474891, 18.94042955,
       22.96580104,  9.19171096,  8.72646654, 15.46347851, 15.14423325,
       10.02751419, 12.5131657 , 22.02788768, 22.33971138, 17.74521036,
        7.96245605,  4.47672779, 18.12488153,  9.12531681, 13.71263076,
        5.34093069, 12.14383032, 11.87368328,  9.84428553,  8.77496059,
       12.4915864 , 20.84812926, 15.31337029, 12.78279574,  7.40979729,
       19.28423444, 18.49320395,  8.56044665, 14.19918545, 17.08809688])

In [9]:
```



```
# -*- coding: utf-8 -*-
"""
@author: ziyan
"""
import pandas as pd
import os
path = "C:/Users/14129/Desktop/dataWarehs"
filename = 'Bank.csv'
fullpath = os.path.join(path,filename)
data_ziyan_b = pd.read_csv(fullpath,sep=';')
print(data_ziyan_b.columns.values)
print(data_ziyan_b.shape)
print(data_ziyan_b.describe())
print(data_ziyan_b.dtypes)
print(data_ziyan_b.head(5))

'''--------------------------------'''

# -*- coding: utf-8 -*-
"""
@author: ziyan
"""

import pandas as pd
import os
path = "C:/Users/14129/Desktop/dataWarehs"
filename = 'Bank.csv'
fullpath = os.path.join(path,filename)
data_ziyan_b = pd.read_csv(fullpath,sep=';')
print(data_ziyan_b.columns.values)
print(data_ziyan_b.shape)
print(data_ziyan_b.describe())
print(data_ziyan_b.dtypes)
print(data_ziyan_b.head(5))
```

| | | | |
|---|---|---|---|
| feature_cols | list | 3 | ['TV', 'Radio', 'Newspaper'] |
| filename | str | 8 | Bank.csv |
| fullpath | str | 42 | C:/Users/14129/Desktop/dataWarehs<br>\Bank.csv |
| path | str | 33 | C:/Users/14129/Desktop/dataWarehs |
| selector | feature_selection._rfe.RFE | 1 | RFE object of sklearn.feature_selection._rfe module |
| X | DataFrame | (200, 3) | Column names: TV, Radio, Newspaper |
| | | | Series object of pandas.core.series |

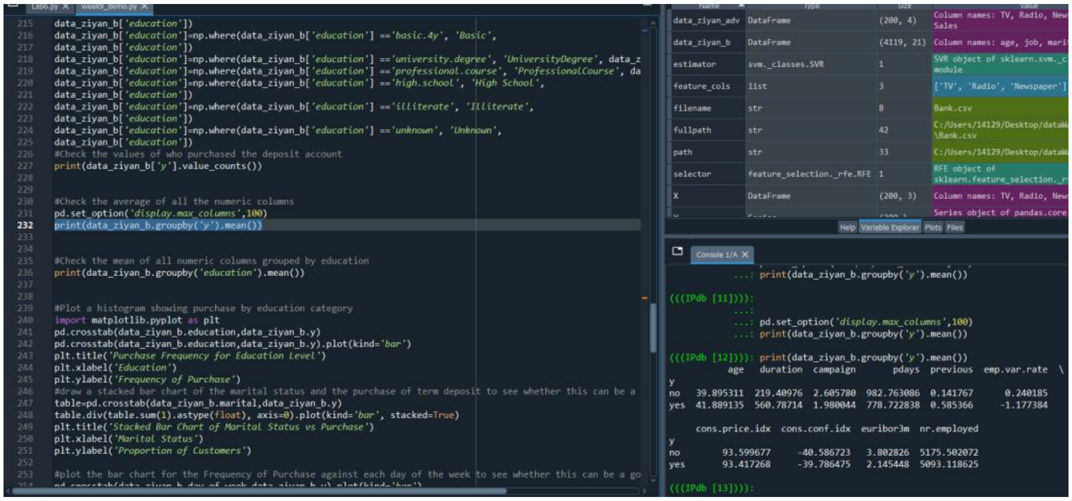
```
duration            int64
campaign            int64
pdays               int64
previous            int64
poutcome           object
emp.var.rate      float64
cons.price.idx    float64
cons.conf.idx     float64
euribor3m         float64
nr.employed       float64
y                  object
dtype: object
   age          job  marital  ... euribor3m nr.employed   y
0   30  blue-collar  married  ...     1.313      5099.1  no
1   39     services   single  ...     4.855      5191.0  no
2   25     services  married  ...     4.962      5228.1  no
3   38     services  married  ...     4.959      5228.1  no
4   47       admin.  married  ...     4.191      5195.8  no
```



```
import pandas as pd
import os
path = "C:/Users/14129/Desktop/dataWarehs"
filename = 'Bank.csv'
fullpath = os.path.join(path,filename)
data_ziyan_b = pd.read_csv(fullpath,sep=';')
print(data_ziyan_b.columns.values)
print(data_ziyan_b.shape)
print(data_ziyan_b.describe())
print(data_ziyan_b.dtypes)
print(data_ziyan_b.head(5))
print(data_ziyan_b['education'].unique())

import numpy as np
data_ziyan_b['education']=np.where(data_ziyan_b['education'] =='basic.9y', 'Basic',
data_ziyan_b['education'])
data_ziyan_b['education']=np.where(data_ziyan_b['education'] =='basic.6y', 'Basic',
data_ziyan_b['education'])
data_ziyan_b['education']=np.where(data_ziyan_b['education'] =='basic.4y', 'Basic',
data_ziyan_b['education'])
data_ziyan_b['education']=np.where(data_ziyan_b['education'] =='university.degree', 'UniversityDegree', data_z
data_ziyan_b['education']=np.where(data_ziyan_b['education'] =='professional.course', 'ProfessionalCourse', da
data_ziyan_b['education']=np.where(data_ziyan_b['education'] =='high.school', 'High School',
data_ziyan_b['education'])
data_ziyan_b['education']=np.where(data_ziyan_b['education'] =='illiterate', 'Illiterate',
data_ziyan_b['education'])
data_ziyan_b['education']=np.where(data_ziyan_b['education'] =='unknown', 'Unknown',
data_ziyan_b['education'])
#Check the values of who purchased the deposit account
print(data_ziyan_b['y'].value_counts())


#Check the average of all the numeric columns
pd.set_option('display.max_columns',100)
print(data_ziyan_b.groupby('y').mean())


#Check the mean of all numeric columns grouped by education
```

| | | | |
|---|---|---|---|
| data_ziyan_adv | DataFrame | (200, 4) | Sales |
| data_ziyan_b | DataFrame | (4119, 21) | Column names: age, job, ma |
| estimator | svm._classes.SVR | 1 | SVR object of sklearn.svm.<br>module |
| feature_cols | list | 3 | ['TV', 'Radio', 'Newspaper |
| filename | str | 8 | Bank.csv |
| fullpath | str | 42 | C:/Users/14129/Desktop/dat<br>\Bank.csv |
| path | str | 33 | C:/Users/14129/Desktop/dat |
| selector | feature_selection._rfe.RFE | 1 | RFE object of<br>sklearn.feature_selection. |
| X | DataFrame | (200, 3) | Column names: TV, Radio, N |
| | | | Series object of pandas.co |

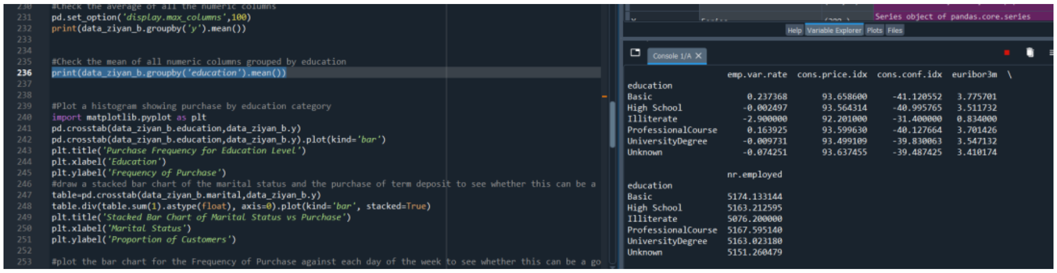
```
euribor3m         float64
nr.employed       float64
y                  object
dtype: object
   age          job  marital  ... euribor3m nr.employed   y
0   30  blue-collar  married  ...     1.313      5099.1  no
1   39     services   single  ...     4.855      5191.0  no
2   25     services  married  ...     4.962      5228.1  no
3   38     services  married  ...     4.959      5228.1  no
4   47       admin.  married  ...     4.191      5195.8  no

[5 rows x 21 columns]
['basic.9y' 'high.school' 'university.degree' 'professional.course'
 'basic.6y' 'basic.4y' 'unknown' 'illiterate']
no     3668
yes     451
Name: y, dtype: int64

(((IPdb [10]))):
```



```
data_ziyan_b['education'])
data_ziyan_b['education']=np.where(data_ziyan_b['education'] =='basic.4y', 'Basic',
data_ziyan_b['education'])
data_ziyan_b['education']=np.where(data_ziyan_b['education'] =='university.degree', 'UniversityDegree', data_z
data_ziyan_b['education']=np.where(data_ziyan_b['education'] =='professional.course', 'ProfessionalCourse', da
data_ziyan_b['education']=np.where(data_ziyan_b['education'] =='high.school', 'High School',
data_ziyan_b['education'])
data_ziyan_b['education']=np.where(data_ziyan_b['education'] =='illiterate', 'Illiterate',
data_ziyan_b['education'])
data_ziyan_b['education']=np.where(data_ziyan_b['education'] =='unknown', 'Unknown',
data_ziyan_b['education'])
#Check the values of who purchased the deposit account
print(data_ziyan_b['y'].value_counts())


#Check the average of all the numeric columns
pd.set_option('display.max_columns',100)
print(data_ziyan_b.groupby('y').mean())


#Check the mean of all numeric columns grouped by education
print(data_ziyan_b.groupby('education').mean())


#Plot a histogram showing purchase by education category
import matplotlib.pyplot as plt
pd.crosstab(data_ziyan_b.education,data_ziyan_b.y)
pd.crosstab(data_ziyan_b.education,data_ziyan_b.y).plot(kind='bar')
plt.title('Purchase Frequency for Education Level')
plt.xlabel('Education')
plt.ylabel('Frequency of Purchase')
#draw a stacked bar chart of the marital status and the purchase of term deposit to see whether this can be a
table=pd.crosstab(data_ziyan_b.marital,data_ziyan_b.y)
table.div(table.sum(1).astype(float), axis=0).plot(kind='bar', stacked=True)
plt.title('Stacked Bar Chart of Marital Status vs Purchase')
plt.xlabel('Marital Status')
plt.ylabel('Proportion of Customers')

#plot the bar chart for the Frequency of Purchase against each day of the week to see whether this can be a go
```

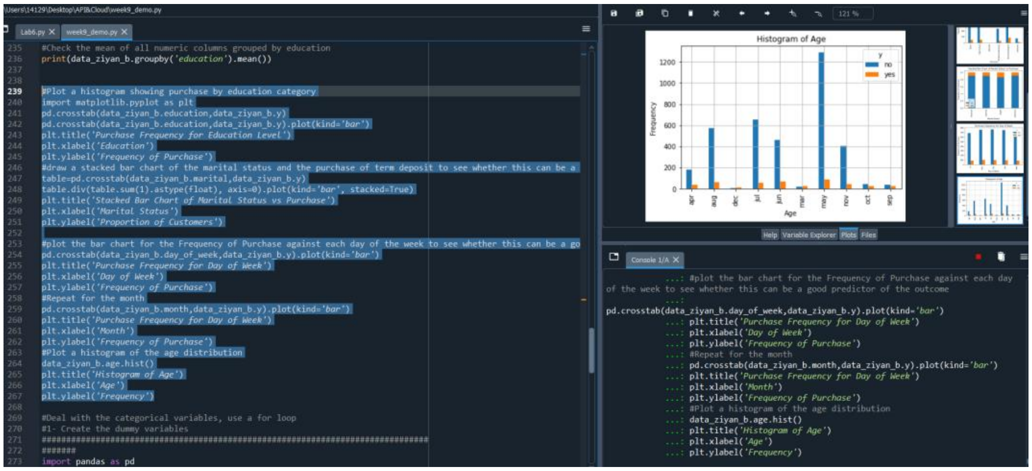
| Name | Type | Size | Value |
|---|---|---|---|
| data_ziyan_adv | DataFrame | (200, 4) | Column names: TV, Radio, New<br>Sales |
| data_ziyan_b | DataFrame | (4119, 21) | Column names: age, job, mari |
| estimator | svm._classes.SVR | 1 | SVR object of sklearn.svm._c<br>module |
| feature_cols | list | 3 | ['TV', 'Radio', 'Newspaper'] |
| filename | str | 8 | Bank.csv |
| fullpath | str | 42 | C:/Users/14129/Desktop/dataW<br>\Bank.csv |
| path | str | 33 | C:/Users/14129/Desktop/dataW |
| selector | feature_selection._rfe.RFE | 1 | RFE object of<br>sklearn.feature_selection._r |
| X | DataFrame | (200, 3) | Column names: TV, Radio, New |
| | | | Series object of pandas.core |

```
          ...: print(data_ziyan_b.groupby('y').mean())

(((IPdb [11]))):
          ...:
          ...: pd.set_option('display.max_columns',100)
          ...: print(data_ziyan_b.groupby('y').mean())

(((IPdb [12]))): print(data_ziyan_b.groupby('y').mean())
           age   duration  campaign       pdays  previous  emp.var.rate  \
y
no   39.895311  219.40976  2.605780  982.763086  0.141767      0.240185
yes  41.889135  560.78714  1.980044  778.722838  0.585366     -1.177384

     cons.price.idx  cons.conf.idx  euribor3m  nr.employed
y
no        93.599677     -40.586723   3.802826  5175.502072
yes       93.417268     -39.786475   2.145448  5093.118625

(((IPdb [13]))):
```



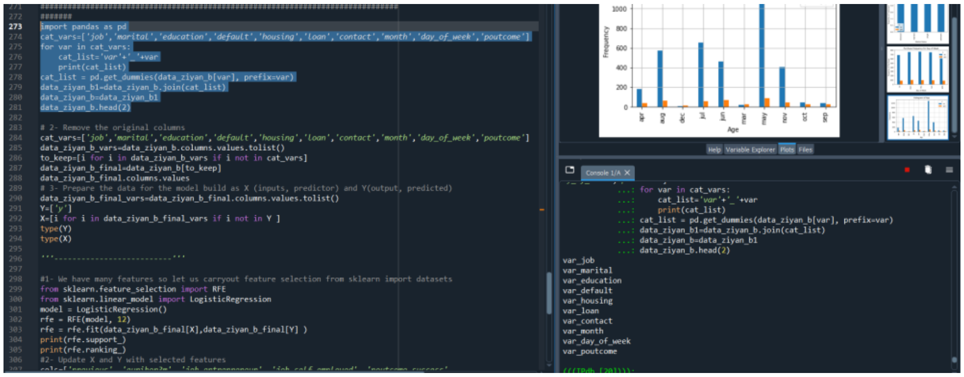
```python
#Check the average of all the numeric columns
pd.set_option('display.max_columns',100)
print(data_ziyan_b.groupby('y').mean())


#Check the mean of all numeric columns grouped by education
print(data_ziyan_b.groupby('education').mean())


#Plot a histogram showing purchase by education category
import matplotlib.pyplot as plt
pd.crosstab(data_ziyan_b.education,data_ziyan_b.y)
pd.crosstab(data_ziyan_b.education,data_ziyan_b.y).plot(kind='bar')
plt.title('Purchase Frequency for Education Level')
plt.xlabel('Education')
plt.ylabel('Frequency of Purchase')
#draw a stacked bar chart of the marital status and the purchase of term deposit to see whether this can be a
table=pd.crosstab(data_ziyan_b.marital,data_ziyan_b.y)
table.div(table.sum(1).astype(float), axis=0).plot(kind='bar', stacked=True)
plt.title('Stacked Bar Chart of Marital Status vs Purchase')
plt.xlabel('Marital Status')
plt.ylabel('Proportion of Customers')

#plot the bar chart for the Frequency of Purchase against each day of the week to see whether this can be a go
```

```
                   emp.var.rate  cons.price.idx  cons.conf.idx  euribor3m  \
education
Basic                  0.237368       93.658600     -41.120552   3.775701
High School           -0.002497       93.564314     -40.995765   3.511732
Illiterate            -2.900000       92.201000     -31.400000   0.834000
ProfessionalCourse     0.163925       93.599630     -40.127664   3.701426
UniversityDegree      -0.009731       93.499109     -39.830063   3.547132
Unknown               -0.074251       93.637455     -39.487425   3.410174

                   nr.employed
education
Basic              5174.133144
High School        5163.212595
Illiterate         5076.200000
ProfessionalCourse 5167.595140
UniversityDegree   5163.023180
Unknown            5151.260479
```



```python
#Check the mean of all numeric columns grouped by education
print(data_ziyan_b.groupby('education').mean())


#Plot a histogram showing purchase by education category
import matplotlib.pyplot as plt
pd.crosstab(data_ziyan_b.education,data_ziyan_b.y)
pd.crosstab(data_ziyan_b.education,data_ziyan_b.y).plot(kind='bar')
plt.title('Purchase Frequency for Education Level')
plt.xlabel('Education')
plt.ylabel('Frequency of Purchase')
#draw a stacked bar chart of the marital status and the purchase of term deposit to see whether this can be a
table=pd.crosstab(data_ziyan_b.marital,data_ziyan_b.y)
table.div(table.sum(1).astype(float), axis=0).plot(kind='bar', stacked=True)
plt.title('Stacked Bar Chart of Marital Status vs Purchase')
plt.xlabel('Marital Status')
plt.ylabel('Proportion of Customers')

#plot the bar chart for the Frequency of Purchase against each day of the week to see whether this can be a go
pd.crosstab(data_ziyan_b.day_of_week,data_ziyan_b.y).plot(kind='bar')
plt.title('Purchase Frequency for Day of Week')
plt.xlabel('Day of Week')
plt.ylabel('Frequency of Purchase')
#Repeat for the month
pd.crosstab(data_ziyan_b.month,data_ziyan_b.y).plot(kind='bar')
plt.title('Purchase Frequency for Day of Week')
plt.xlabel('Month')
plt.ylabel('Frequency of Purchase')
#Plot a histogram of the age distribution
data_ziyan_b.age.hist()
plt.title('Histogram of Age')
plt.xlabel('Age')
plt.ylabel('Frequency')

#Deal with the categorical variables, use a for loop
#1- Create the dummy variables
##############################################################################
#######
import pandas as pd
```

```
           ...: #plot the bar chart for the Frequency of Purchase against each day
of the week to see whether this can be a good predictor of the outcome
           ...:
pd.crosstab(data_ziyan_b.day_of_week,data_ziyan_b.y).plot(kind='bar')
           ...: plt.title('Purchase Frequency for Day of Week')
           ...: plt.xlabel('Day of Week')
           ...: plt.ylabel('Frequency of Purchase')
           ...: #Repeat for the month
           ...: pd.crosstab(data_ziyan_b.month,data_ziyan_b.y).plot(kind='bar')
           ...: plt.title('Purchase Frequency for Day of Week')
           ...: plt.xlabel('Month')
           ...: plt.ylabel('Frequency of Purchase')
           ...: #Plot a histogram of the age distribution
           ...: data_ziyan_b.age.hist()
           ...: plt.title('Histogram of Age')
           ...: plt.xlabel('Age')
           ...: plt.ylabel('Frequency')
```



```python
##############################################################################
#######
import pandas as pd
cat_vars=['job','marital','education','default','housing','loan','contact','month','day_of_week','poutcome']
for var in cat_vars:
    cat_list='var'+'_'+var
    print(cat_list)
cat_list = pd.get_dummies(data_ziyan_b[var], prefix=var)
data_ziyan_b1=data_ziyan_b.join(cat_list)
data_ziyan_b=data_ziyan_b1
data_ziyan_b.head(2)

# 2- Remove the original columns
cat_vars=['job','marital','education','default','housing','loan','contact','month','day_of_week','poutcome']
data_ziyan_b_vars=data_ziyan_b.columns.values.tolist()
to_keep=[i for i in data_ziyan_b_vars if i not in cat_vars]
data_ziyan_b_final=data_ziyan_b[to_keep]
data_ziyan_b_final.columns.values
# 3- Prepare the data for the model build as X (inputs, predictor) and Y(output, predicted)
data_ziyan_b_final_vars=data_ziyan_b_final.columns.values.tolist()
Y=['y']
X=[i for i in data_ziyan_b_final_vars if i not in Y ]
type(Y)
type(X)

'''..........................'''

#1- We have many features so let us carryout feature selection from sklearn import datasets
from sklearn.feature_selection import RFE
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
rfe = RFE(model, 12)
rfe = rfe.fit(data_ziyan_b_final[X],data_ziyan_b_final[Y] )
print(rfe.support_)
print(rfe.ranking_)
#2- Update X and Y with selected features
```

```
           ...: for var in cat_vars:
           ...:     cat_list='var'+'_'+var
           ...:     print(cat_list)
           ...: cat_list = pd.get_dummies(data_ziyan_b[var], prefix=var)
           ...: data_ziyan_b1=data_ziyan_b.join(cat_list)
           ...: data_ziyan_b=data_ziyan_b1
           ...: data_ziyan_b.head(2)
var_job
var_marital
var_education
var_default
var_housing
var_loan
var_contact
var_month
var_day_of_week
var_poutcome

(((IPdb [20]))):
```