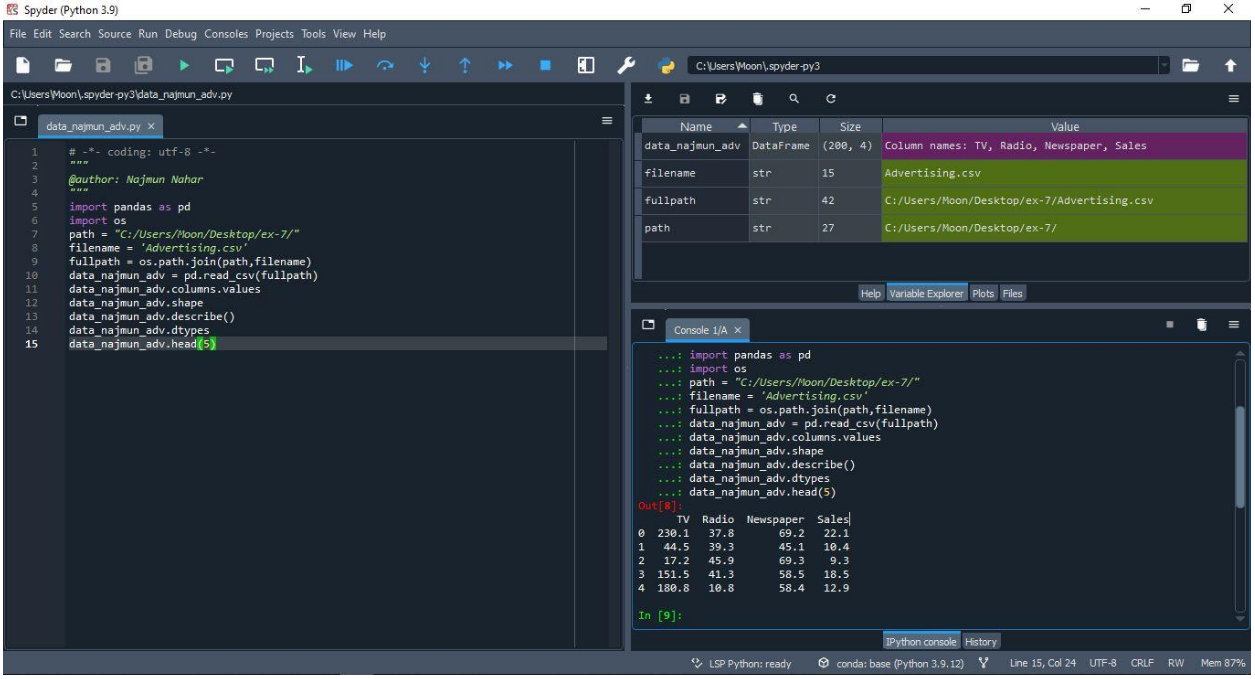
Name: Najmun Nahar

ID: 301160081

Exercise-7A (Linear Regression)

1.

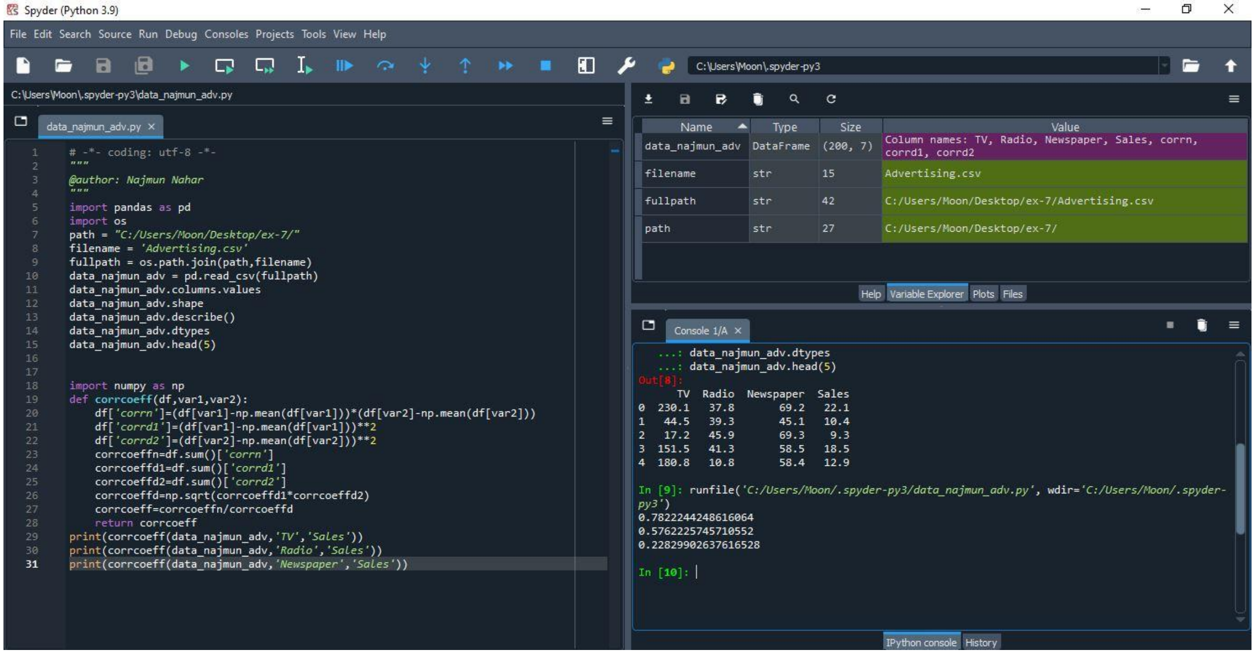

Spyder (Python 3.9)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\Moon\.spyder-py3
C:\Users\Moon\.spyder-py3\data_najmun_adv.py
data_najmun_adv.py
Help Variable Explorer Plots Files
Console 1/A
IPython console History
LSP Python: ready
conda: base (Python 3.9.12)
Line 15, Col 24
UTF-8
CRLF
RW
Mem 87%

```python
# -*- coding: utf-8 -*-
"""
@author: Najmun Nahar
"""
import pandas as pd
import os
path = "C:/Users/Moon/Desktop/ex-7/"
filename = 'Advertising.csv'
fullpath = os.path.join(path,filename)
data_najmun_adv = pd.read_csv(fullpath)
data_najmun_adv.columns.values
data_najmun_adv.shape
data_najmun_adv.describe()
data_najmun_adv.dtypes
data_najmun_adv.head(5)
```

Name	Type	Size	Value
data_najmun_adv	DataFrame	(200, 4)	Column names: TV, Radio, Newspaper, Sales
filename	str	15	Advertising.csv
fullpath	str	42	C:/Users/Moon/Desktop/ex-7/Advertising.csv
path	str	27	C:/Users/Moon/Desktop/ex-7/

```
   ...: import pandas as pd
   ...: import os
   ...: path = "C:/Users/Moon/Desktop/ex-7/"
   ...: filename = 'Advertising.csv'
   ...: fullpath = os.path.join(path,filename)
   ...: data_najmun_adv = pd.read_csv(fullpath)
   ...: data_najmun_adv.columns.values
   ...: data_najmun_adv.shape
   ...: data_najmun_adv.describe()
   ...: data_najmun_adv.dtypes
   ...: data_najmun_adv.head(5)
Out[8]:
      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 [9]:
```

2.

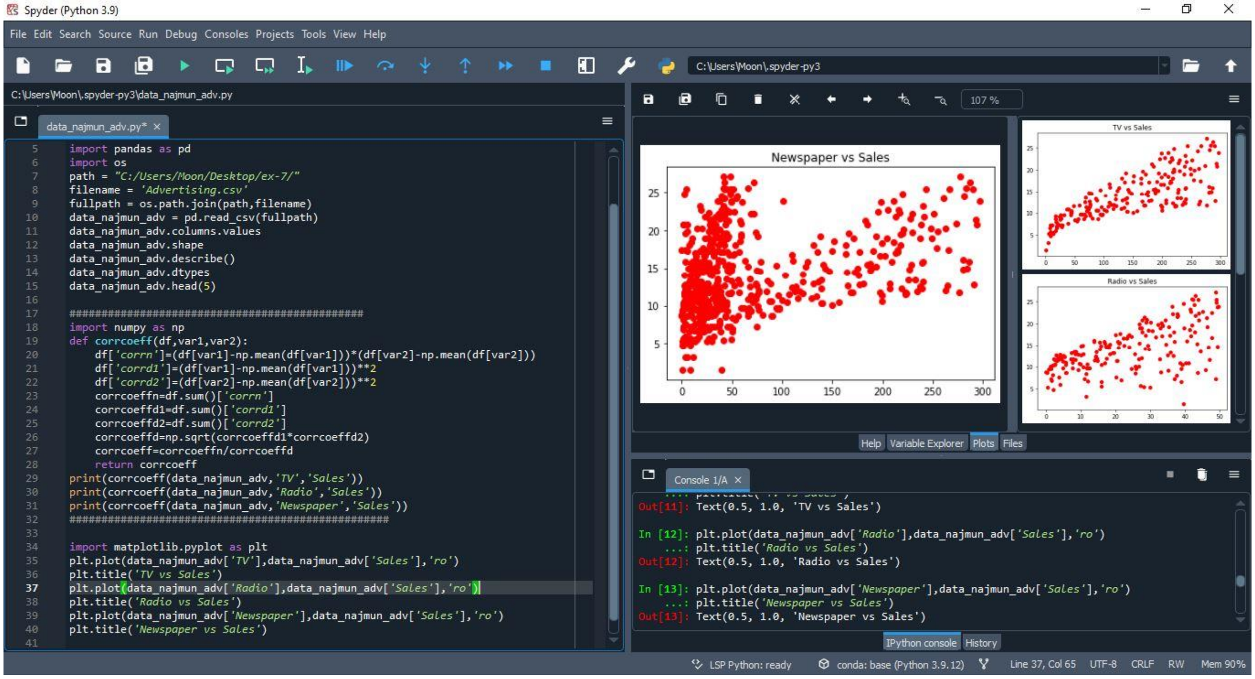

Spyder (Python 3.9)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\Moon\.spyder-py3
C:\Users\Moon\.spyder-py3\data_najmun_adv.py
data_najmun_adv.py
Help Variable Explorer Plots Files
Console 1/A
IPython console History

```python
# -*- coding: utf-8 -*-
"""
@author: Najmun Nahar
"""
import pandas as pd
import os
path = "C:/Users/Moon/Desktop/ex-7/"
filename = 'Advertising.csv'
fullpath = os.path.join(path,filename)
data_najmun_adv = pd.read_csv(fullpath)
data_najmun_adv.columns.values
data_najmun_adv.shape
data_najmun_adv.describe()
data_najmun_adv.dtypes
data_najmun_adv.head(5)


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_najmun_adv,'TV','Sales'))
print(corrcoeff(data_najmun_adv,'Radio','Sales'))
print(corrcoeff(data_najmun_adv,'Newspaper','Sales'))
```

Name	Type	Size	Value
data_najmun_adv	DataFrame	(200, 7)	Column names: TV, Radio, Newspaper, Sales, corrn, corrd1, corrd2
filename	str	15	Advertising.csv
fullpath	str	42	C:/Users/Moon/Desktop/ex-7/Advertising.csv
path	str	27	C:/Users/Moon/Desktop/ex-7/

```
   ...: data_najmun_adv.dtypes
   ...: data_najmun_adv.head(5)
Out[8]:
      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 [9]: runfile('C:/Users/Moon/.spyder-py3/data_najmun_adv.py', wdir='C:/Users/Moon/.spyder-
py3')
0.7822244248616064
0.5762225745710552
0.22829902637616528

In [10]:
```

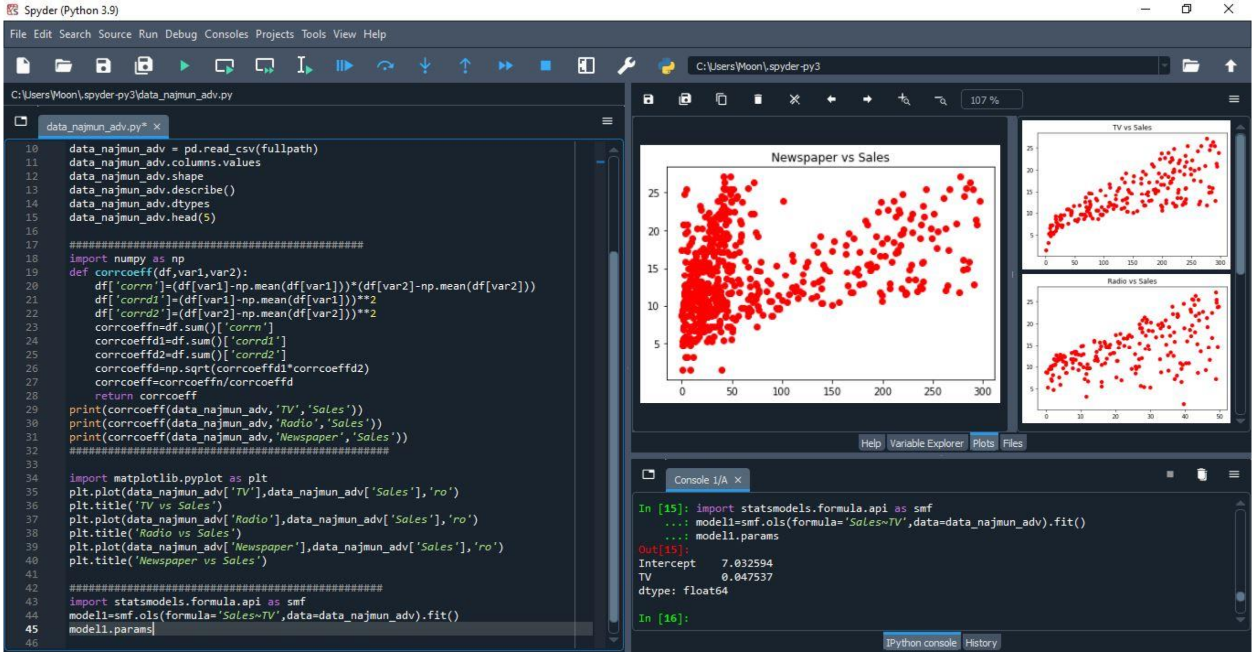
3.



```python
import pandas as pd
import os
path = "C:/Users/Moon/Desktop/ex-7/"
filename = 'Advertising.csv'
fullpath = os.path.join(path,filename)
data_najmun_adv = pd.read_csv(fullpath)
data_najmun_adv.columns.values
data_najmun_adv.shape
data_najmun_adv.describe()
data_najmun_adv.dtypes
data_najmun_adv.head(5)

#############################################
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_najmun_adv,'TV','Sales'))
print(corrcoeff(data_najmun_adv,'Radio','Sales'))
print(corrcoeff(data_najmun_adv,'Newspaper','Sales'))
##################################################

import matplotlib.pyplot as plt
plt.plot(data_najmun_adv['TV'],data_najmun_adv['Sales'],'ro')
plt.title('TV vs Sales')
plt.plot(data_najmun_adv['Radio'],data_najmun_adv['Sales'],'ro')
plt.title('Radio vs Sales')
plt.plot(data_najmun_adv['Newspaper'],data_najmun_adv['Sales'],'ro')
plt.title('Newspaper vs Sales')
```

```
Out[11]: Text(0.5, 1.0, 'TV vs Sales')

In [12]: plt.plot(data_najmun_adv['Radio'],data_najmun_adv['Sales'],'ro')
    ...: plt.title('Radio vs Sales')
Out[12]: Text(0.5, 1.0, 'Radio vs Sales')

In [13]: plt.plot(data_najmun_adv['Newspaper'],data_najmun_adv['Sales'],'ro')
    ...: plt.title('Newspaper vs Sales')
Out[13]: Text(0.5, 1.0, 'Newspaper vs Sales')
```

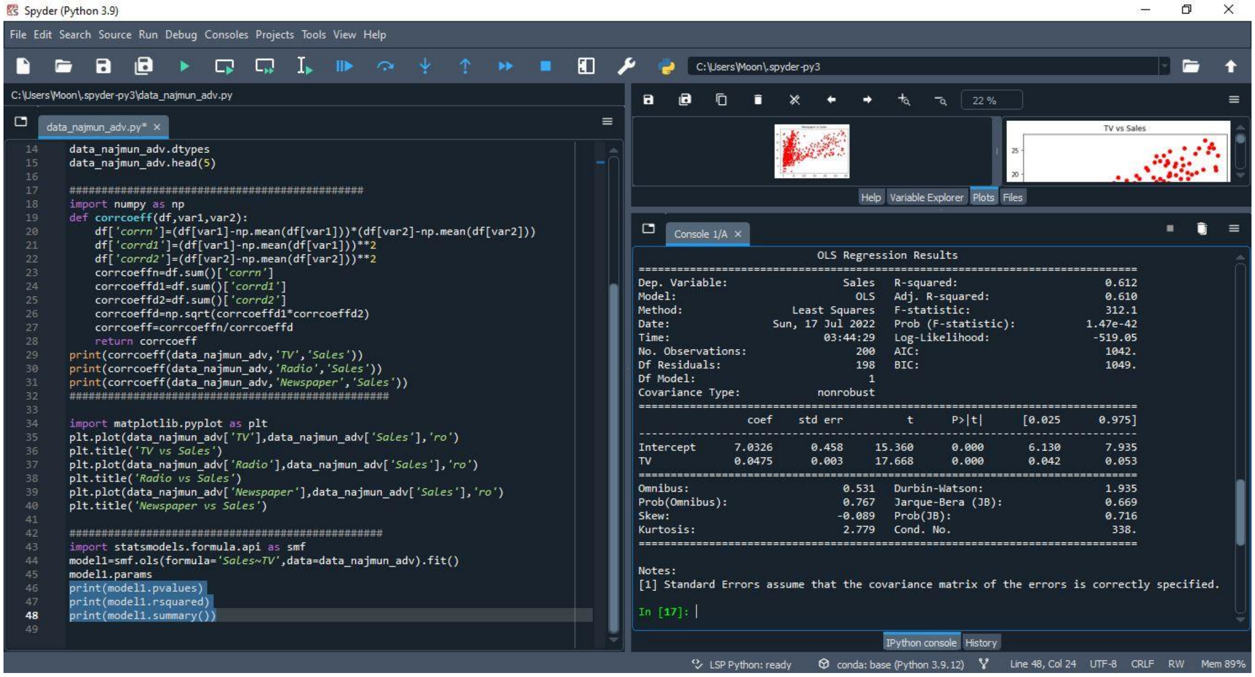
4.



```python
data_najmun_adv = pd.read_csv(fullpath)
data_najmun_adv.columns.values
data_najmun_adv.shape
data_najmun_adv.describe()
data_najmun_adv.dtypes
data_najmun_adv.head(5)

#############################################
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_najmun_adv,'TV','Sales'))
print(corrcoeff(data_najmun_adv,'Radio','Sales'))
print(corrcoeff(data_najmun_adv,'Newspaper','Sales'))
##################################################

import matplotlib.pyplot as plt
plt.plot(data_najmun_adv['TV'],data_najmun_adv['Sales'],'ro')
plt.title('TV vs Sales')
plt.plot(data_najmun_adv['Radio'],data_najmun_adv['Sales'],'ro')
plt.title('Radio vs Sales')
plt.plot(data_najmun_adv['Newspaper'],data_najmun_adv['Sales'],'ro')
plt.title('Newspaper vs Sales')

##################################################
import statsmodels.formula.api as smf
model1=smf.ols(formula='Sales~TV',data=data_najmun_adv).fit()
model1.params
```

```
In [15]: import statsmodels.formula.api as smf
    ...: model1=smf.ols(formula='Sales~TV',data=data_najmun_adv).fit()
    ...: model1.params
Out[15]:
Intercept    7.032594
TV           0.047537
dtype: float64

In [16]:
```

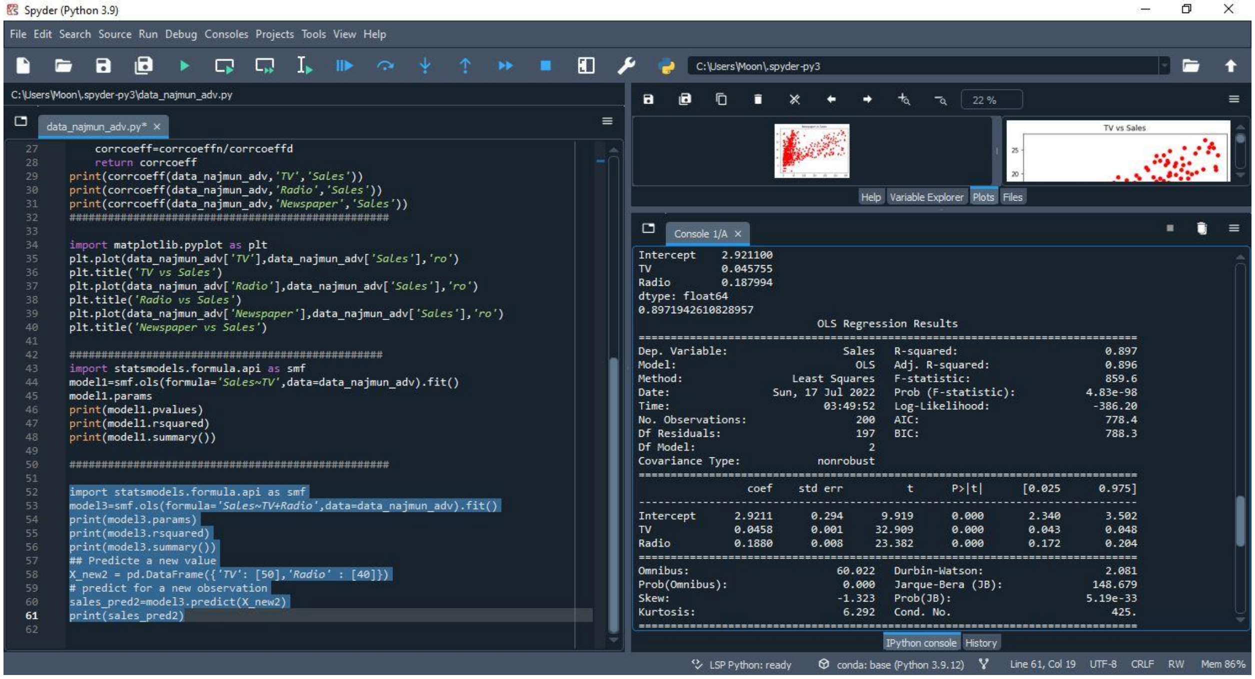
5.


Spyder (Python 3.9)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\Moon\.spyder-py3
C:\Users\Moon\.spyder-py3\data_najmun_adv.py
data_najmun_adv.py*
22 %
TV vs Sales
25
20
Help Variable Explorer Plots Files
Console 1/A
IPython console History
LSP Python: ready
conda: base (Python 3.9.12)
Line 48, Col 24
UTF-8
CRLF
RW
Mem 89%

```python
data_najmun_adv.dtypes
data_najmun_adv.head(5)

##############################################
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_najmun_adv,'TV','Sales'))
print(corrcoeff(data_najmun_adv,'Radio','Sales'))
print(corrcoeff(data_najmun_adv,'Newspaper','Sales'))
##################################################

import matplotlib.pyplot as plt
plt.plot(data_najmun_adv['TV'],data_najmun_adv['Sales'],'ro')
plt.title('TV vs Sales')
plt.plot(data_najmun_adv['Radio'],data_najmun_adv['Sales'],'ro')
plt.title('Radio vs Sales')
plt.plot(data_najmun_adv['Newspaper'],data_najmun_adv['Sales'],'ro')
plt.title('Newspaper vs Sales')

#################################################
import statsmodels.formula.api as smf
model1=smf.ols(formula='Sales~TV',data=data_najmun_adv).fit()
model1.params
print(model1.pvalues)
print(model1.rsquared)
print(model1.summary())

```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  Sales   R-squared:                       0.612
Model:                            OLS   Adj. R-squared:                  0.610
Method:                 Least Squares   F-statistic:                     312.1
Date:                Sun, 17 Jul 2022   Prob (F-statistic):           1.47e-42
Time:                        03:44:29   Log-Likelihood:                -519.05
No. Observations:                 200   AIC:                             1042.
Df Residuals:                     198   BIC:                             1049.
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 [17]: 
```

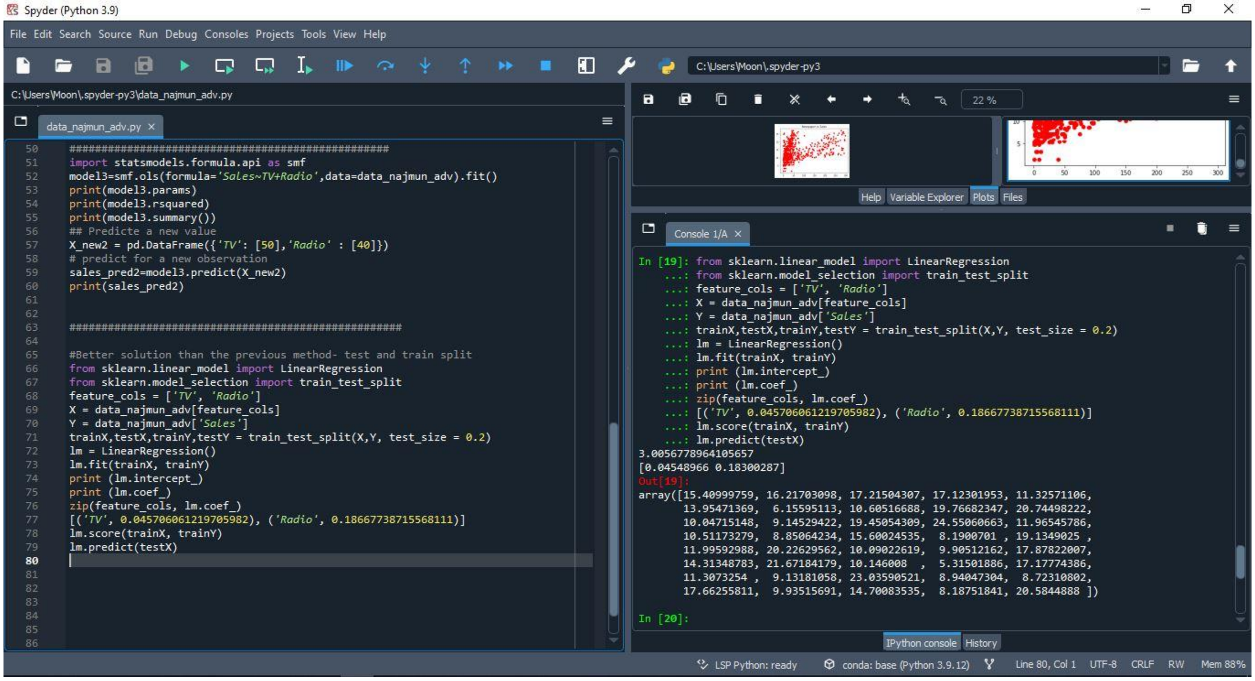
6.


Spyder (Python 3.9)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\Moon\.spyder-py3
C:\Users\Moon\.spyder-py3\data_najmun_adv.py
data_najmun_adv.py*
22 %
TV vs Sales
25
20
Help Variable Explorer Plots Files
Console 1/A
IPython console History
LSP Python: ready
conda: base (Python 3.9.12)
Line 61, Col 19
UTF-8
CRLF
RW
Mem 86%

```python
    corrcoeff=corrcoeffn/corrcoeffd
    return corrcoeff
print(corrcoeff(data_najmun_adv,'TV','Sales'))
print(corrcoeff(data_najmun_adv,'Radio','Sales'))
print(corrcoeff(data_najmun_adv,'Newspaper','Sales'))
##################################################

import matplotlib.pyplot as plt
plt.plot(data_najmun_adv['TV'],data_najmun_adv['Sales'],'ro')
plt.title('TV vs Sales')
plt.plot(data_najmun_adv['Radio'],data_najmun_adv['Sales'],'ro')
plt.title('Radio vs Sales')
plt.plot(data_najmun_adv['Newspaper'],data_najmun_adv['Sales'],'ro')
plt.title('Newspaper vs Sales')

#################################################
import statsmodels.formula.api as smf
model1=smf.ols(formula='Sales~TV',data=data_najmun_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_najmun_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)

```

```
Intercept    2.921100
TV           0.045755
Radio        0.187994
dtype: float64
0.8971942610828957
                            OLS Regression Results
==============================================================================
Dep. Variable:                  Sales   R-squared:                       0.897
Model:                            OLS   Adj. R-squared:                  0.896
Method:                 Least Squares   F-statistic:                     859.6
Date:                Sun, 17 Jul 2022   Prob (F-statistic):           4.83e-98
Time:                        03:49:52   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.
==============================================================================
```

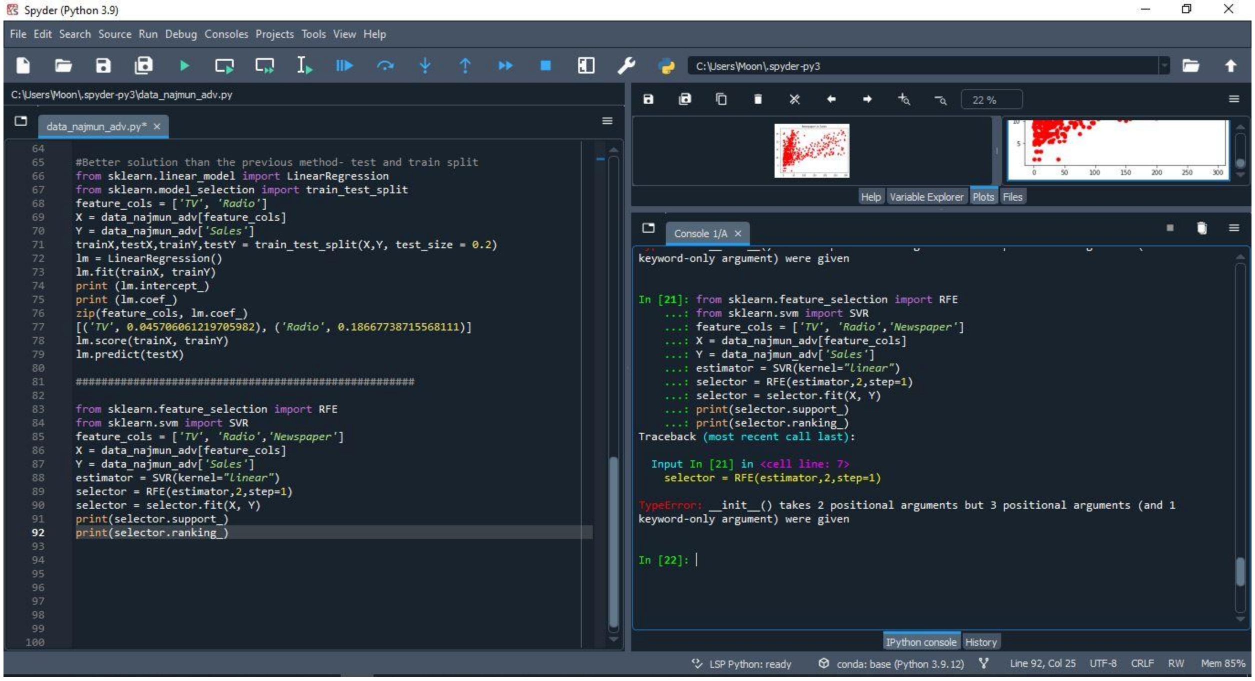
7.


Spyder (Python 3.9)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\Moon\.spyder-py3
C:\Users\Moon\.spyder-py3\data_najmun_adv.py
data_najmun_adv.py
22 %
Help Variable Explorer Plots Files
Console 1/A
IPython console History
LSP Python: ready
conda: base (Python 3.9.12)
Line 80, Col 1 UTF-8 CRLF RW Mem 88%

```
##################################################
import statsmodels.formula.api as smf
model3=smf.ols(formula='Sales~TV+Radio',data=data_najmun_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
from sklearn.model_selection import train_test_split
feature_cols = ['TV', 'Radio']
X = data_najmun_adv[feature_cols]
Y = data_najmun_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)
```

```
In [19]: from sklearn.linear_model import LinearRegression
    ...: from sklearn.model_selection import train_test_split
    ...: feature_cols = ['TV', 'Radio']
    ...: X = data_najmun_adv[feature_cols]
    ...: Y = data_najmun_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)
3.0056778964105657
[0.04548966 0.18300287]
Out[19]: 
array([15.40999759, 16.21703098, 17.21504307, 17.12301953, 11.32571106,
       13.95471369,  6.15595113, 10.60516688, 19.76682347, 20.74498222,
       10.04715148,  9.14529422, 19.45054309, 24.55060663, 11.96545786,
       10.51173279,  8.85064234, 15.60024535,  8.1900701 , 19.1349025 ,
       11.99592988, 20.22629562, 10.09022619,  9.90512162, 17.87822007,
       14.31348783, 21.67184179, 10.146008  ,  5.31501886, 17.17774386,
       11.3073254 ,  9.13181058, 23.03590521,  8.94047304,  8.72310802,
       17.66255811,  9.93515691, 14.70083535,  8.18751841, 20.5844888 ])

In [20]:
```

8.

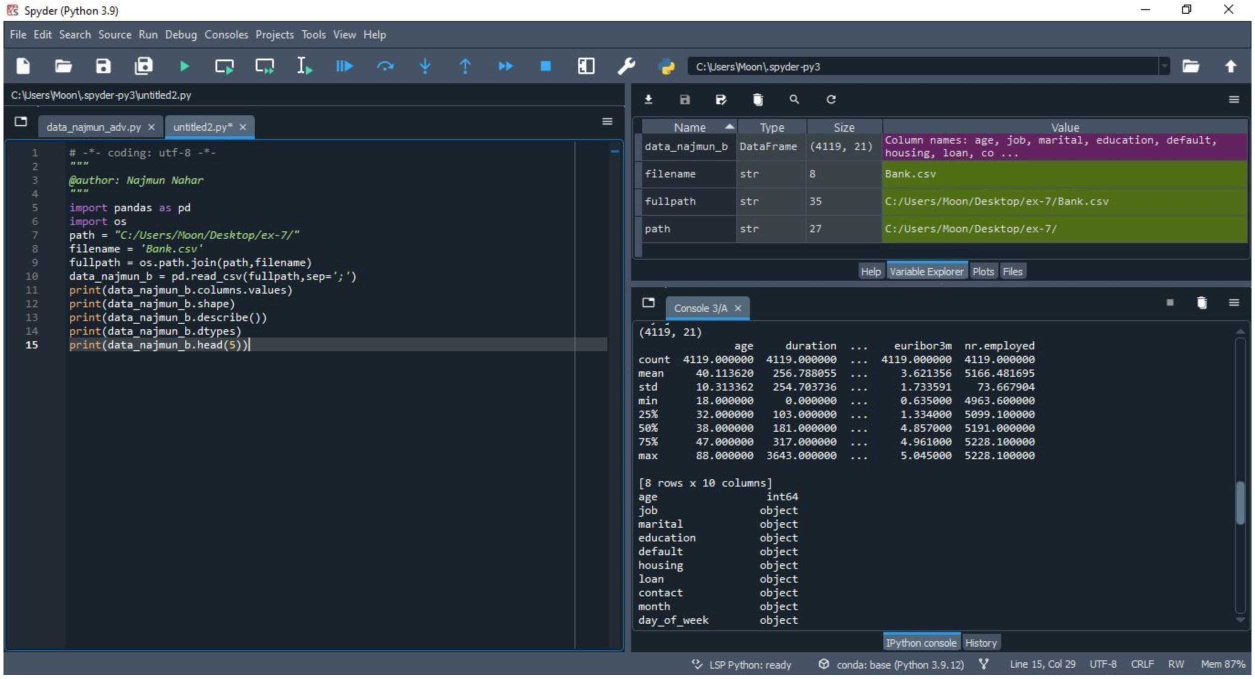

Spyder (Python 3.9)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\Moon\.spyder-py3
C:\Users\Moon\.spyder-py3\data_najmun_adv.py
data_najmun_adv.py*
22 %
Help Variable Explorer Plots Files
Console 1/A
IPython console History
LSP Python: ready
conda: base (Python 3.9.12)
Line 92, Col 25 UTF-8 CRLF RW Mem 85%

```

#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_najmun_adv[feature_cols]
Y = data_najmun_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_najmun_adv[feature_cols]
Y = data_najmun_adv['Sales']
estimator = SVR(kernel="linear")
selector = RFE(estimator,2,step=1)
selector = selector.fit(X, Y)
print(selector.support_)
print(selector.ranking_)
```

```
keyword-only argument) were given


In [21]: from sklearn.feature_selection import RFE
    ...: from sklearn.svm import SVR
    ...: feature_cols = ['TV', 'Radio','Newspaper']
    ...: X = data_najmun_adv[feature_cols]
    ...: Y = data_najmun_adv['Sales']
    ...: estimator = SVR(kernel="linear")
    ...: selector = RFE(estimator,2,step=1)
    ...: selector = selector.fit(X, Y)
    ...: print(selector.support_)
    ...: print(selector.ranking_)
Traceback (most recent call last):

  Input In [21] in <cell line: 7>
    selector = RFE(estimator,2,step=1)

TypeError: __init__() takes 2 positional arguments but 3 positional arguments (and 1
keyword-only argument) were given


In [22]: 
```

Exercise 7B (Logistic Regression)

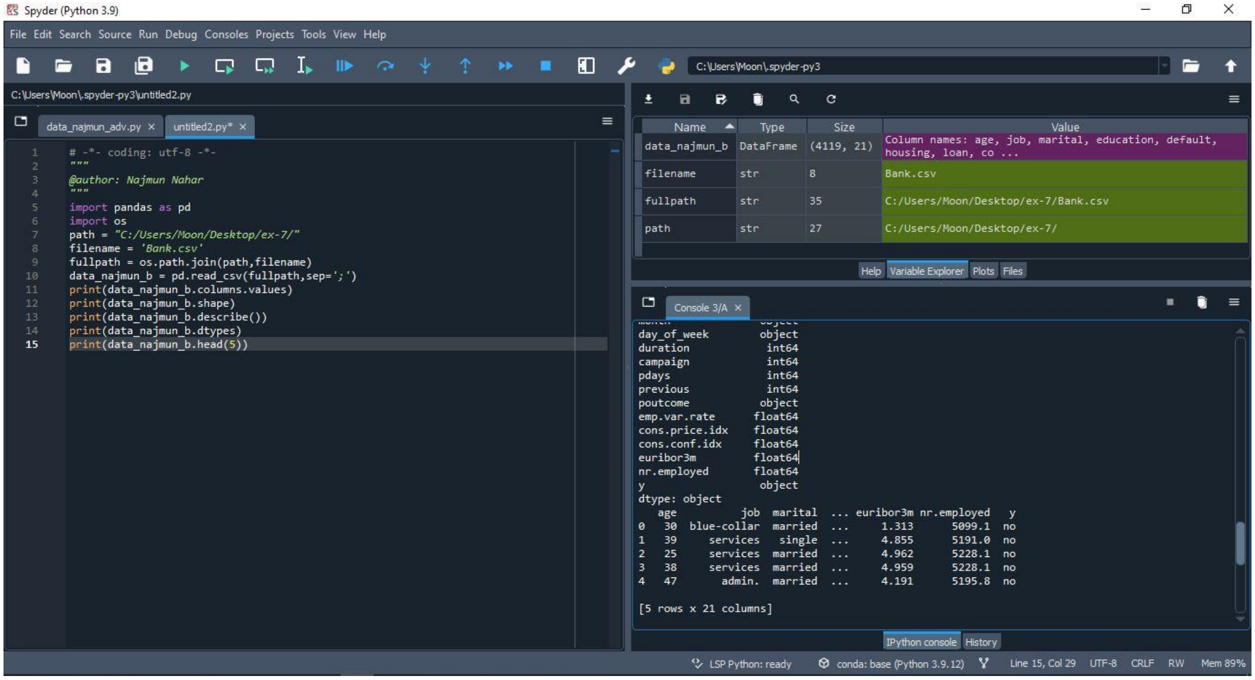
1.


Spyder (Python 3.9)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\Moon\.spyder-py3
C:\Users\Moon\.spyder-py3\untitled2.py
data_najmun_adv.py
untitled2.py*
Help Variable Explorer Plots Files
Console 3/A
IPython console History
LSP Python: ready conda: base (Python 3.9.12) Line 15, Col 29 UTF-8 CRLF RW Mem 87%

```python
# -*- coding: utf-8 -*-
"""
@author: Najmun Nahar
"""
import pandas as pd
import os
path = "C:/Users/Moon/Desktop/ex-7/"
filename = 'Bank.csv'
fullpath = os.path.join(path,filename)
data_najmun_b = pd.read_csv(fullpath,sep=';')
print(data_najmun_b.columns.values)
print(data_najmun_b.shape)
print(data_najmun_b.describe())
print(data_najmun_b.dtypes)
print(data_najmun_b.head(5))
```

Name	Type	Size	Value
data_najmun_b	DataFrame	(4119, 21)	Column names: age, job, marital, education, default, housing, loan, co ...
filename	str	8	Bank.csv
fullpath	str	35	C:/Users/Moon/Desktop/ex-7/Bank.csv
path	str	27	C:/Users/Moon/Desktop/ex-7/

```
(4119, 21)
               age     duration  ...    euribor3m  nr.employed
count  4119.000000  4119.000000  ...  4119.000000  4119.000000
mean     40.113620   256.788055  ...     3.621356  5166.481695
std      10.313362   254.703736  ...     1.733591    73.667904
min      18.000000     0.000000  ...     0.635000  4963.600000
25%      32.000000   103.000000  ...     1.334000  5099.100000
50%      38.000000   181.000000  ...     4.857000  5191.000000
75%      47.000000   317.000000  ...     4.961000  5228.100000
max      88.000000  3643.000000  ...     5.045000  5228.100000

[8 rows x 10 columns]
age                int64
job               object
marital           object
education         object
default           object
housing           object
loan              object
contact           object
month             object
day_of_week       object
```

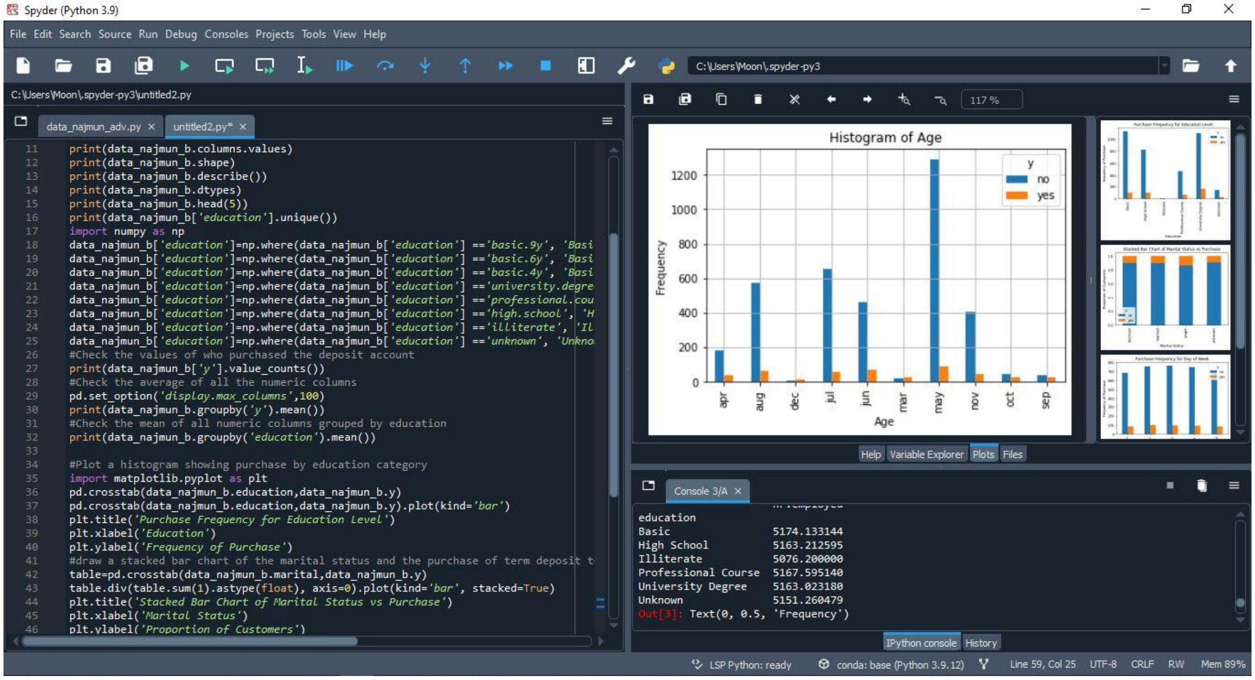

Spyder (Python 3.9)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\Moon\.spyder-py3
C:\Users\Moon\.spyder-py3\untitled2.py
data_najmun_adv.py
untitled2.py*
Help Variable Explorer Plots Files
Console 3/A
IPython console History
LSP Python: ready conda: base (Python 3.9.12) Line 15, Col 29 UTF-8 CRLF RW Mem 89%

```python
# -*- coding: utf-8 -*-
"""
@author: Najmun Nahar
"""
import pandas as pd
import os
path = "C:/Users/Moon/Desktop/ex-7/"
filename = 'Bank.csv'
fullpath = os.path.join(path,filename)
data_najmun_b = pd.read_csv(fullpath,sep=';')
print(data_najmun_b.columns.values)
print(data_najmun_b.shape)
print(data_najmun_b.describe())
print(data_najmun_b.dtypes)
print(data_najmun_b.head(5))
```

Name	Type	Size	Value
data_najmun_b	DataFrame	(4119, 21)	Column names: age, job, marital, education, default, housing, loan, co ...
filename	str	8	Bank.csv
fullpath	str	35	C:/Users/Moon/Desktop/ex-7/Bank.csv
path	str	27	C:/Users/Moon/Desktop/ex-7/

```
day_of_week       object
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

[5 rows x 21 columns]
```

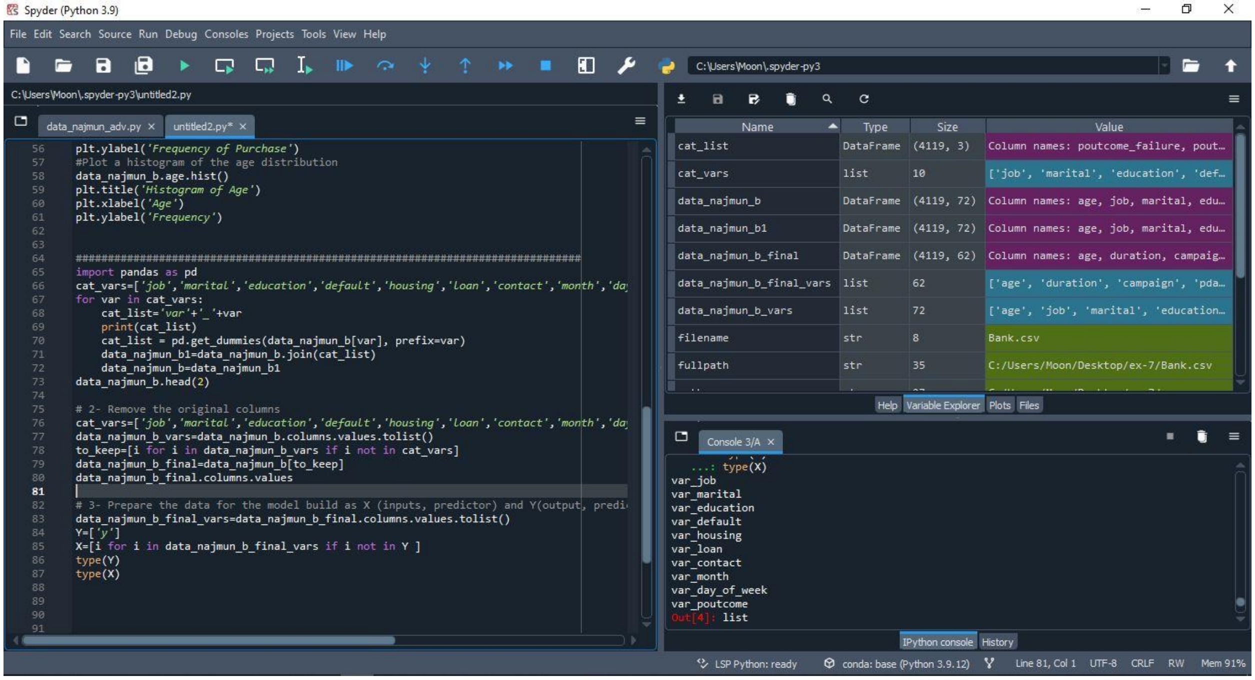
2.


Spyder (Python 3.9)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\Moon\.spyder-py3
C:\Users\Moon\.spyder-py3\untitled2.py
data_najmun_adv.py
untitled2.py*
117 %
Histogram of Age
Frequency
Age
y
no
yes
0
200
400
600
800
1000
1200
apr
aug
dec
jul
jun
mar
may
nov
oct
sep
Purchase Frequency for Education Level
Stacked Bar Chart of Marital Status vs Purchase
Purchase Frequency for Day of Week
Help
Variable Explorer
Plots
Files
Console 3/A
IPython console
History
LSP Python: ready
conda: base (Python 3.9.12)
Line 59, Col 25
UTF-8
CRLF
RW
Mem 89%

```
print(data_najmun_b.columns.values)
print(data_najmun_b.shape)
print(data_najmun_b.describe())
print(data_najmun_b.dtypes)
print(data_najmun_b.head(5))
print(data_najmun_b['education'].unique())
import numpy as np
data_najmun_b['education']=np.where(data_najmun_b['education'] =='basic.9y', 'Basi
data_najmun_b['education']=np.where(data_najmun_b['education'] =='basic.6y', 'Basi
data_najmun_b['education']=np.where(data_najmun_b['education'] =='basic.4y', 'Basi
data_najmun_b['education']=np.where(data_najmun_b['education'] =='university.degre
data_najmun_b['education']=np.where(data_najmun_b['education'] =='professional.cou
data_najmun_b['education']=np.where(data_najmun_b['education'] =='high.school', 'H
data_najmun_b['education']=np.where(data_najmun_b['education'] =='illiterate', 'Il
data_najmun_b['education']=np.where(data_najmun_b['education'] =='unknown', 'Unkno
#Check the values of who purchased the deposit account
print(data_najmun_b['y'].value_counts())
#Check the average of all the numeric columns
pd.set_option('display.max_columns',100)
print(data_najmun_b.groupby('y').mean())
#Check the mean of all numeric columns grouped by education
print(data_najmun_b.groupby('education').mean())

#Plot a histogram showing purchase by education category
import matplotlib.pyplot as plt
pd.crosstab(data_najmun_b.education,data_najmun_b.y)
pd.crosstab(data_najmun_b.education,data_najmun_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 t
table=pd.crosstab(data_najmun_b.marital,data_najmun_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')
```

```
education
Basic                  5174.133144
High School            5163.212595
Illiterate             5076.200000
Professional Course    5167.595140
University Degree      5163.023180
Unknown                5151.260479
Out[3]: Text(0, 0.5, 'Frequency')
```

3.

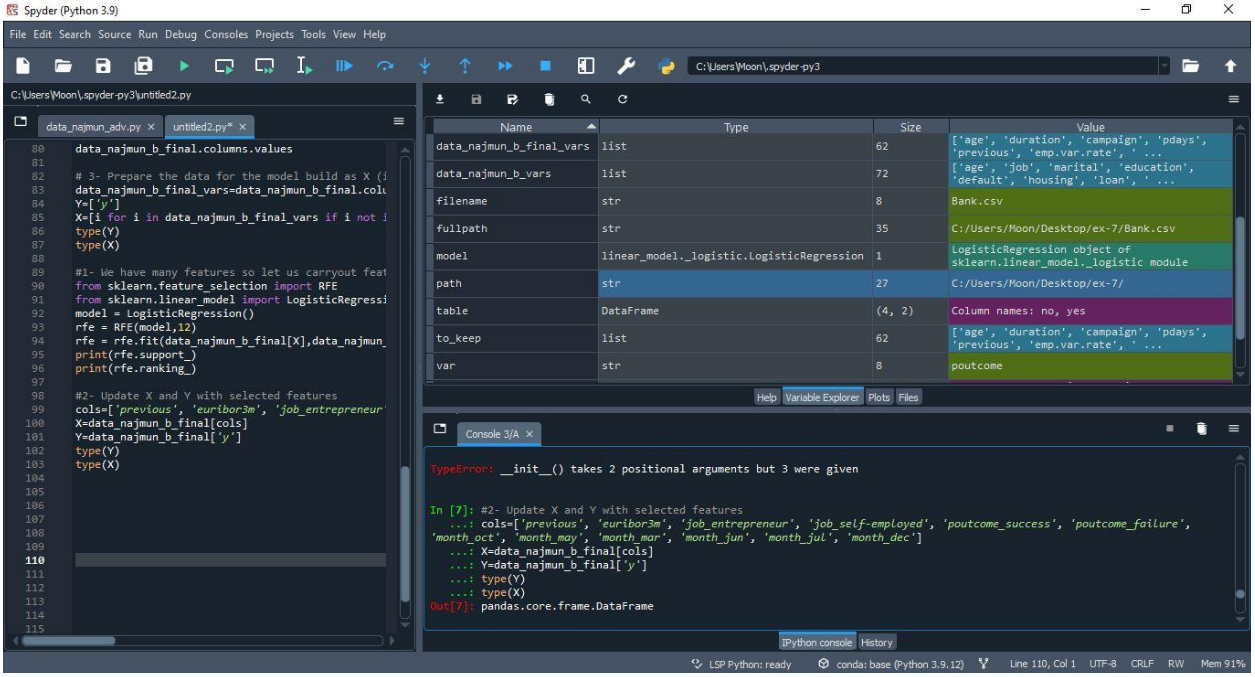


```
plt.ylabel('Frequency of Purchase')
#Plot a histogram of the age distribution
data_najmun_b.age.hist()
plt.title('Histogram of Age')
plt.xlabel('Age')
plt.ylabel('Frequency')


################################################################################
import pandas as pd
cat_vars=['job','marital','education','default','housing','loan','contact','month','da
for var in cat_vars:
    cat_list='var'+'_'+var
    print(cat_list)
    cat_list = pd.get_dummies(data_najmun_b[var], prefix=var)
    data_najmun_b1=data_najmun_b.join(cat_list)
    data_najmun_b=data_najmun_b1
data_najmun_b.head(2)

# 2- Remove the original columns
cat_vars=['job','marital','education','default','housing','loan','contact','month','da
data_najmun_b_vars=data_najmun_b.columns.values.tolist()
to_keep=[i for i in data_najmun_b_vars if i not in cat_vars]
data_najmun_b_final=data_najmun_b[to_keep]
data_najmun_b_final.columns.values

# 3- Prepare the data for the model build as X (inputs, predictor) and Y(output, predi
data_najmun_b_final_vars=data_najmun_b_final.columns.values.tolist()
Y=['y']
X=[i for i in data_najmun_b_final_vars if i not in Y ]
type(Y)
type(X)
```

Name	Type	Size	Value
cat_list	DataFrame	(4119, 3)	Column names: poutcome_failure, pout…
cat_vars	list	10	['job', 'marital', 'education', 'def…
data_najmun_b	DataFrame	(4119, 72)	Column names: age, job, marital, edu…
data_najmun_b1	DataFrame	(4119, 72)	Column names: age, job, marital, edu…
data_najmun_b_final	DataFrame	(4119, 62)	Column names: age, duration, campaig…
data_najmun_b_final_vars	list	62	['age', 'duration', 'campaign', 'pda…
data_najmun_b_vars	list	72	['age', 'job', 'marital', 'education…
filename	str	8	Bank.csv
fullpath	str	35	C:/Users/Moon/Desktop/ex-7/Bank.csv

```
   ...: type(X)
var_job
var_marital
var_education
var_default
var_housing
var_loan
var_contact
var_month
var_day_of_week
var_poutcome
Out[4]: list
```

4.

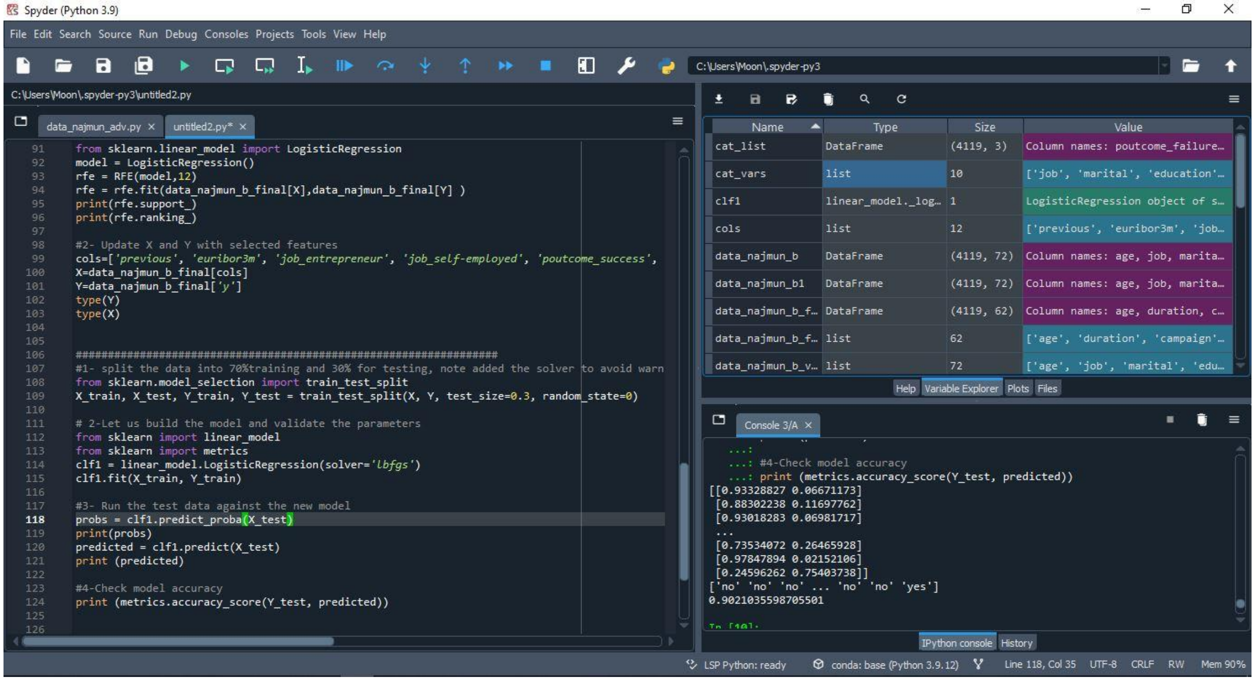


```python
data_najmun_b_final.columns.values

# 3- Prepare the data for the model build as X (i
data_najmun_b_final_vars=data_najmun_b_final.colu
Y=['y']
X=[i for i in data_najmun_b_final_vars if i not i
type(Y)
type(X)

#1- We have many features so let us carryout feat
from sklearn.feature_selection import RFE
from sklearn.linear_model import LogisticRegressi
model = LogisticRegression()
rfe = RFE(model,12)
rfe = rfe.fit(data_najmun_b_final[X],data_najmun_
print(rfe.support_)
print(rfe.ranking_)

#2- Update X and Y with selected features
cols=['previous', 'euribor3m', 'job_entrepreneur'
X=data_najmun_b_final[cols]
Y=data_najmun_b_final['y']
type(Y)
type(X)
```

Name	Type	Size	Value
data_najmun_b_final_vars	list	62	['age', 'duration', 'campaign', 'pdays', 'previous', 'emp.var.rate', ' ...
data_najmun_b_vars	list	72	['age', 'job', 'marital', 'education', 'default', 'housing', 'loan', ' ...
filename	str	8	Bank.csv
fullpath	str	35	C:/Users/Moon/Desktop/ex-7/Bank.csv
model	linear_model._logistic.LogisticRegression	1	LogisticRegression object of sklearn.linear_model._logistic module
path	str	27	C:/Users/Moon/Desktop/ex-7/
table	DataFrame	(4, 2)	Column names: no, yes
to_keep	list	62	['age', 'duration', 'campaign', 'pdays', 'previous', 'emp.var.rate', ' ...
var	str	8	poutcome

```
TypeError: __init__() takes 2 positional arguments but 3 were given


In [7]: #2- Update X and Y with selected features
   ...: cols=['previous', 'euribor3m', 'job_entrepreneur', 'job_self-employed', 'poutcome_success', 'poutcome_failure',
'month_oct', 'month_may', 'month_mar', 'month_jun', 'month_jul', 'month_dec']
   ...: X=data_najmun_b_final[cols]
   ...: Y=data_najmun_b_final['y']
   ...: type(Y)
   ...: type(X)
Out[7]: pandas.core.frame.DataFrame
```

5.

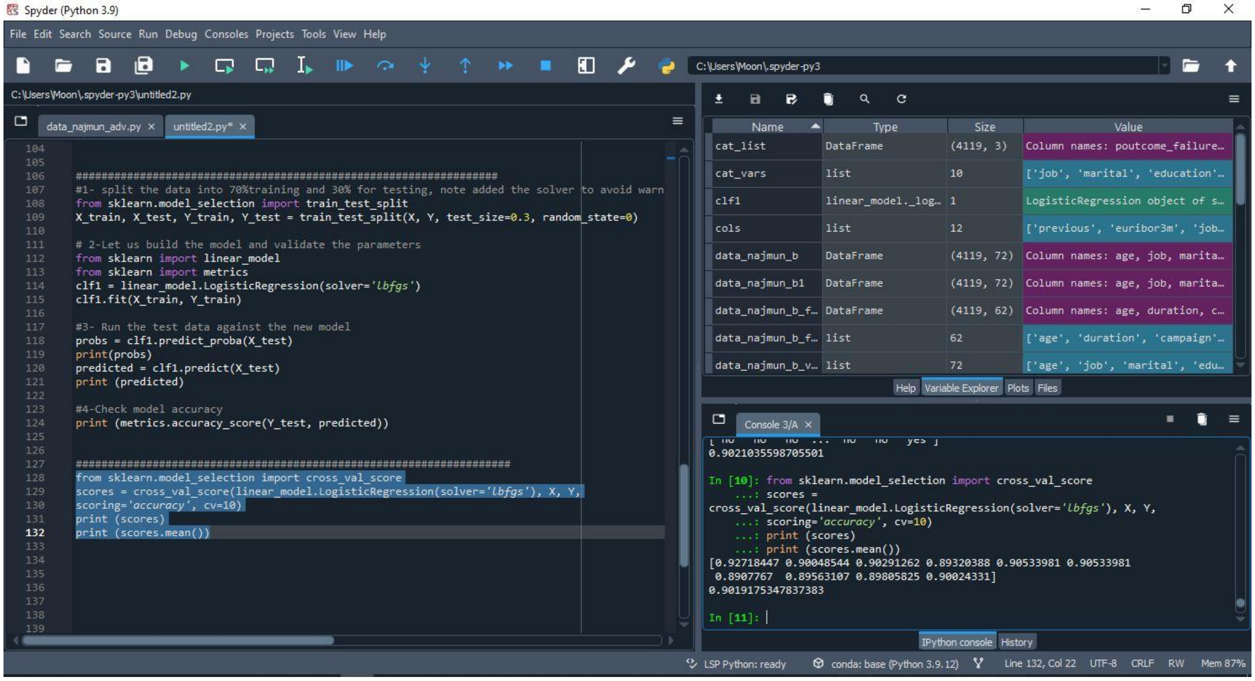


```python
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
rfe = RFE(model,12)
rfe = rfe.fit(data_najmun_b_final[X],data_najmun_b_final[Y] )
print(rfe.support_)
print(rfe.ranking_)

#2- Update X and Y with selected features
cols=['previous', 'euribor3m', 'job_entrepreneur', 'job_self-employed', 'poutcome_success',
X=data_najmun_b_final[cols]
Y=data_najmun_b_final['y']
type(Y)
type(X)


###################################################################
#1- split the data into 70%training and 30% for testing, note added the solver to avoid warn
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.3, random_state=0)

# 2-Let us build the model and validate the parameters
from sklearn import linear_model
from sklearn import metrics
clf1 = linear_model.LogisticRegression(solver='lbfgs')
clf1.fit(X_train, Y_train)

#3- Run the test data against the new model
probs = clf1.predict_proba(X_test)
print(probs)
predicted = clf1.predict(X_test)
print (predicted)

#4-Check model accuracy
print (metrics.accuracy_score(Y_test, predicted))
```

Name	Type	Size	Value
cat_list	DataFrame	(4119, 3)	Column names: poutcome_failure…
cat_vars	list	10	['job', 'marital', 'education'…
clf1	linear_model._log…	1	LogisticRegression object of s…
cols	list	12	['previous', 'euribor3m', 'job…
data_najmun_b	DataFrame	(4119, 72)	Column names: age, job, marita…
data_najmun_b1	DataFrame	(4119, 72)	Column names: age, job, marita…
data_najmun_b_f…	DataFrame	(4119, 62)	Column names: age, duration, c…
data_najmun_b_f…	list	62	['age', 'duration', 'campaign'…
data_najmun_b_v…	list	72	['age', 'job', 'marital', 'edu…

```
   ...:
   ...: #4-Check model accuracy
   ...: print (metrics.accuracy_score(Y_test, predicted))
[[0.93328827 0.06671173]
 [0.88302238 0.11697762]
 [0.93018283 0.06981717]
 ...
 [0.73534072 0.26465928]
 [0.97847894 0.02152106]
 [0.24596262 0.75403738]]
['no' 'no' 'no' ... 'no' 'no' 'yes']
0.9021035598705501

In [10]:
```

6.

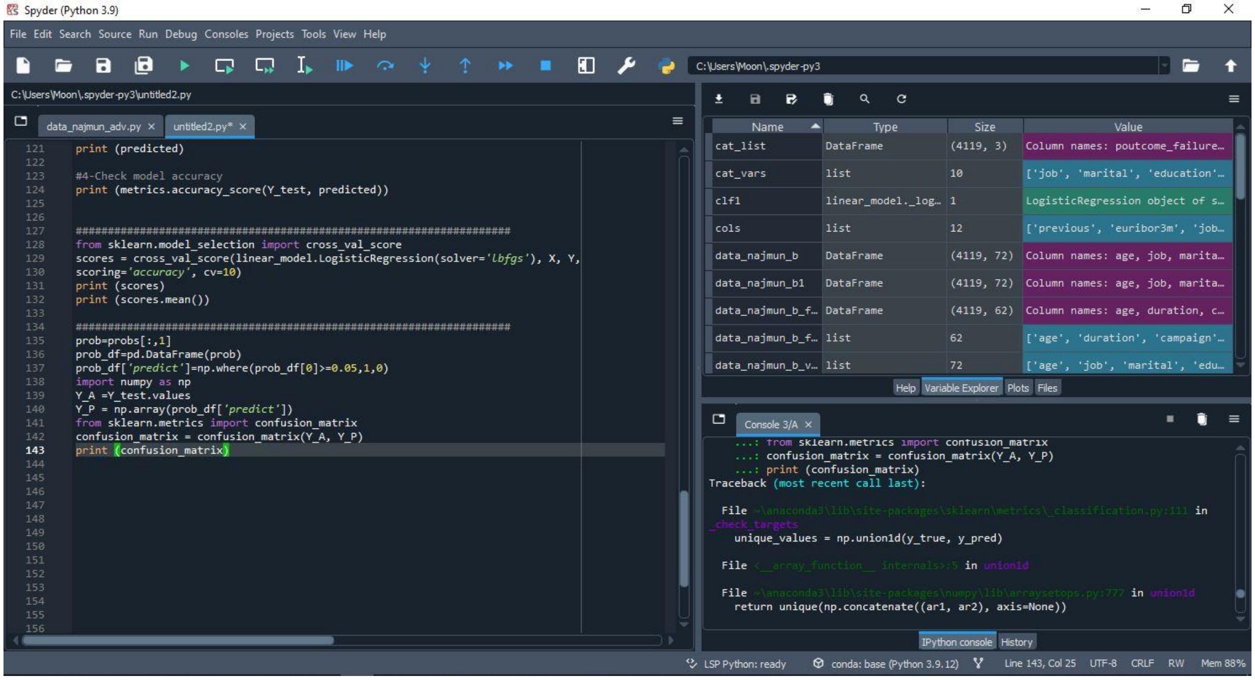

Spyder (Python 3.9)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\Moon\.spyder-py3
C:\Users\Moon\.spyder-py3\untitled2.py
data_najmun_adv.py
untitled2.py*
Help Variable Explorer Plots Files
Console 3/A
IPython console History
LSP Python: ready
conda: base (Python 3.9.12)
Line 132, Col 22
UTF-8
CRLF
RW
Mem 87%

```python


##################################################################
#1- split the data into 70%training and 30% for testing, note added the solver to avoid warn
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.3, random_state=0)

# 2-Let us build the model and validate the parameters
from sklearn import linear_model
from sklearn import metrics
clf1 = linear_model.LogisticRegression(solver='lbfgs')
clf1.fit(X_train, Y_train)

#3- Run the test data against the new model
probs = clf1.predict_proba(X_test)
print(probs)
predicted = clf1.predict(X_test)
print (predicted)

#4-Check model accuracy
print (metrics.accuracy_score(Y_test, predicted))


####################################################################
from sklearn.model_selection import cross_val_score
scores = cross_val_score(linear_model.LogisticRegression(solver='lbfgs'), X, Y,
scoring='accuracy', cv=10)
print (scores)
print (scores.mean())
```

Name	Type	Size	Value
cat_list	DataFrame	(4119, 3)	Column names: poutcome_failure…
cat_vars	list	10	['job', 'marital', 'education'…
clf1	linear_model._log…	1	LogisticRegression object of s…
cols	list	12	['previous', 'euribor3m', 'job…
data_najmun_b	DataFrame	(4119, 72)	Column names: age, job, marita…
data_najmun_b1	DataFrame	(4119, 72)	Column names: age, job, marita…
data_najmun_b_f…	DataFrame	(4119, 62)	Column names: age, duration, c…
data_najmun_b_f…	list	62	['age', 'duration', 'campaign'…
data_najmun_b_v…	list	72	['age', 'job', 'marital', 'edu…

```
[ no   no   no  ...  no   no   yes ]
0.9021035598705501

In [10]: from sklearn.model_selection import cross_val_score
    ...: scores =
cross_val_score(linear_model.LogisticRegression(solver='lbfgs'), X, Y,
    ...: scoring='accuracy', cv=10)
    ...: print (scores)
    ...: print (scores.mean())
[0.92718447 0.90048544 0.90291262 0.89320388 0.90533981 0.90533981
 0.8907767  0.89563107 0.89805825 0.90024331]
0.9019175347837383

In [11]: 
```

7.


Spyder (Python 3.9)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\Moon\.spyder-py3
C:\Users\Moon\.spyder-py3\untitled2.py
data_najmun_adv.py
untitled2.py*
Help Variable Explorer Plots Files
Console 3/A
IPython console History
LSP Python: ready
conda: base (Python 3.9.12)
Line 143, Col 25
UTF-8
CRLF
RW
Mem 88%

```python
print (predicted)

#4-Check model accuracy
print (metrics.accuracy_score(Y_test, predicted))


####################################################################
from sklearn.model_selection import cross_val_score
scores = cross_val_score(linear_model.LogisticRegression(solver='lbfgs'), X, Y,
scoring='accuracy', cv=10)
print (scores)
print (scores.mean())

####################################################################
prob=probs[:,1]
prob_df=pd.DataFrame(prob)
prob_df['predict']=np.where(prob_df[0]>=0.05,1,0)
import numpy as np
Y_A =Y_test.values
Y_P = np.array(prob_df['predict'])
from sklearn.metrics import confusion_matrix
confusion_matrix = confusion_matrix(Y_A, Y_P)
print (confusion_matrix)
```

Name	Type	Size	Value
cat_list	DataFrame	(4119, 3)	Column names: poutcome_failure…
cat_vars	list	10	['job', 'marital', 'education'…
clf1	linear_model._log…	1	LogisticRegression object of s…
cols	list	12	['previous', 'euribor3m', 'job…
data_najmun_b	DataFrame	(4119, 72)	Column names: age, job, marita…
data_najmun_b1	DataFrame	(4119, 72)	Column names: age, job, marita…
data_najmun_b_f…	DataFrame	(4119, 62)	Column names: age, duration, c…
data_najmun_b_f…	list	62	['age', 'duration', 'campaign'…
data_najmun_b_v…	list	72	['age', 'job', 'marital', 'edu…

```
    ...: from sklearn.metrics import confusion_matrix
    ...: confusion_matrix = confusion_matrix(Y_A, Y_P)
    ...: print (confusion_matrix)
Traceback (most recent call last):

  File ~\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:111 in
_check_targets
    unique_values = np.union1d(y_true, y_pred)

  File <__array_function__ internals>:5 in union1d

  File ~\anaconda3\lib\site-packages\numpy\lib\arraysetops.py:777 in union1d
    return unique(np.concatenate((ar1, ar2), axis=None))
```