BIRZEIT UNIVERSITY

ENCS3320 COMP4388: Machine Learning
Fall 2021/2022

Project#1

Prepared by:

Salah AlDin Dar Aldeek 1192404

Notes :

- **Name of file:** Algerian_forest_fires_dataset_UPDATE.csv
- **The file contains 2 sets, we set it in the same data set using pandas:**
  1. Bejaia Region Dataset
  2. Sidi-Bel Abbes Region Dataset
- **Attributes description:**

  1. Date: (DD/MM/YYYY) Day, month ('June' to 'September'), year (2012) Weather data observations
  2. Temp: temperature noon (temperature max) in Celsius degrees: 22 to 42
  3. RH: Relative Humidity in %: 21 to 90
  4. Ws: Wind speed in km/h: 6 to 29
  5. Rain: total day in mm: 0 to 16.8

  FWI Components

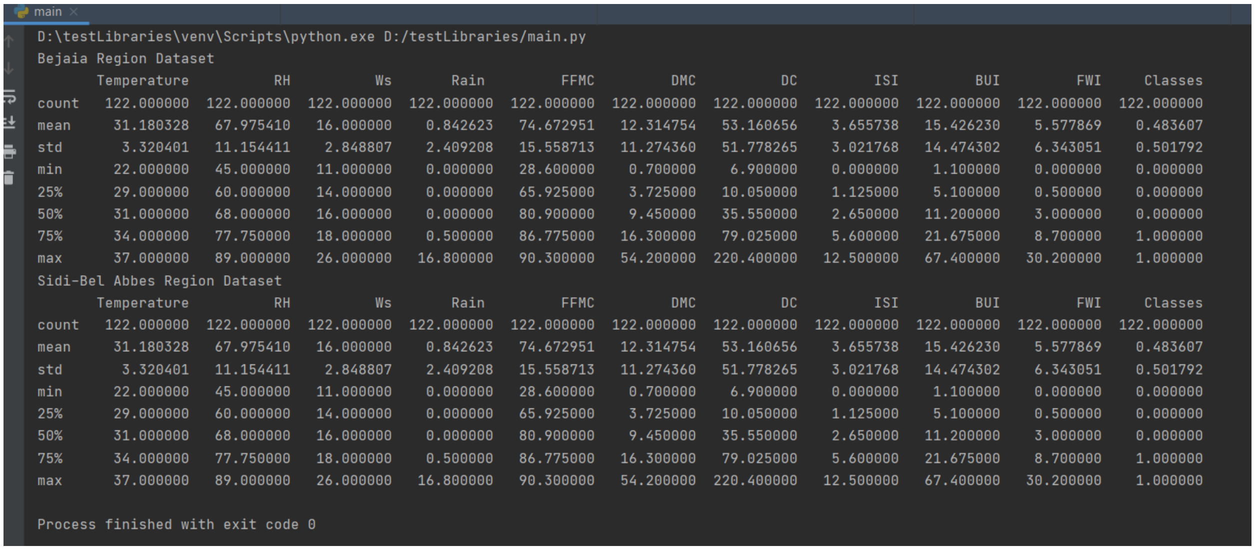
  6. Fine Fuel Moisture Code (FFMC) index from the FWI system: 28.6 to 92.5
  7. Duff Moisture Code (DMC) index from the FWI system: 1.1 to 65.9
  8. Drought Code (DC) index from the FWI system: 7 to 220.4
  9. Initial Spread Index (ISI) index from the FWI system: 0 to 18.5
  10. Buildup Index (BUI) index from the FWI system: 1.1 to 68
  11. Fire Weather Index (FWI) Index: 0 to 31.1
  12. Classes: two classes, namely a fire and not fire

Task1:

First I read the file and set it in two data sets for each region and I get the columns of data sets to use it to determine the columns I need, then I replace the fire to 1 and not fire to 1 in the Classes feature, to calculate the mean and standard deviation (max, min,1st 2nd 3rd Quartile are nut useful in the feature Classes), and to use it in the post tasks (e.g. correlation), and then I use the method describe (), to get the mean, standard deviation , 1st 2nd 3rd Quartile of columns 3-14(the summary of first 3 columns <day,month,year> are not useful , because the variables in day,month,years is static , and its not helpful to get the summary of its) , and then I print the summary of this 2 data sets.

```
import pandas as pd
#read the Bejaia Region Dataset(from row 1 read 122 line)
dataSet1=pd.read_csv("Algerian_forest_fires_dataset_UPDATE.csv",skiprows=1,nrows=122)
#read the Bejaia Region Dataset(from row 126 read 122 line)
dataSet2=pd.read_csv("Algerian_forest_fires_dataset_UPDATE.csv",skiprows=126,nrows=122)
#Get the attributes of data set (day, month, year, Temp, HR, Ws,....  )
l=dataSet1.columns
'''replace the fire to 1 and not fire to 0 in the last column(Classes)
to get the mean and Standard Deviation of fire and not fire(mean,
max,1st&2nd&3rd Quartile are not important in this feature)'''
dataSet1[l[13]].replace(['not fire','fire'],[0,1],inplace=True)
dataSet2[l[13]].replace(['not fire','fire'],[0,1],inplace=True)
'''Print the Summary of columns from 3 to 12 of 2 data sets ,because the
first 3 columns is day,month,year and the summary of this features doesn’t
useful , the method describe() used to get the mean,
std(Standard Deviation), min, max, 1st 2nd 3rd quartile of data set '''
print("Bejaia Region Dataset")
print(dataSet1[l[3:14]].describe().to_string());
print("Sidi-Bel Abbes Region Dataset")
print(dataSet1[l[3:14]].describe().to_string());
```

Result:

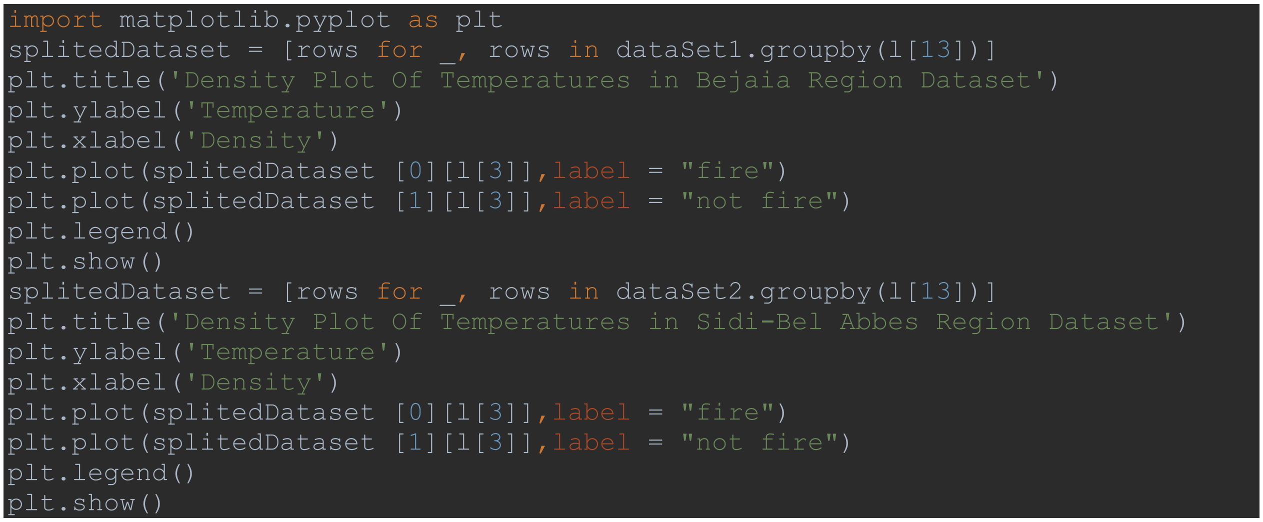


```
D:\testLibraries\venv\Scripts\python.exe D:/testLibraries/main.py
Bejaia Region Dataset
       Temperature          RH          Ws        Rain        FFMC         DMC          DC         ISI         BUI         FWI     Classes
count   122.000000  122.000000  122.000000  122.000000  122.000000  122.000000  122.000000  122.000000  122.000000  122.000000  122.000000
mean     31.180328   67.975410   16.000000    0.842623   74.672951   12.314754   53.160656    3.655738   15.426230    5.577869    0.483607
std       3.320401   11.154411    2.848807    2.409208   15.558713   11.274360   51.778265    3.021768   14.474302    6.343051    0.501792
min      22.000000   45.000000   11.000000    0.000000   28.600000    0.700000    6.900000    0.000000    1.100000    0.000000    0.000000
25%      29.000000   60.000000   14.000000    0.000000   65.925000    3.725000   10.050000    1.125000    5.100000    0.500000    0.000000
50%      31.000000   68.000000   16.000000    0.000000   80.900000    9.450000   35.550000    2.650000   11.200000    3.000000    0.000000
75%      34.000000   77.750000   18.000000    0.500000   86.775000   16.300000   79.025000    5.600000   21.675000    8.700000    1.000000
max      37.000000   89.000000   26.000000   16.800000   90.300000   54.200000  220.400000   12.500000   67.400000   30.200000    1.000000
Sidi-Bel Abbes Region Dataset
       Temperature          RH          Ws        Rain        FFMC         DMC          DC         ISI         BUI         FWI     Classes
count   122.000000  122.000000  122.000000  122.000000  122.000000  122.000000  122.000000  122.000000  122.000000  122.000000  122.000000
mean     31.180328   67.975410   16.000000    0.842623   74.672951   12.314754   53.160656    3.655738   15.426230    5.577869    0.483607
std       3.320401   11.154411    2.848807    2.409208   15.558713   11.274360   51.778265    3.021768   14.474302    6.343051    0.501792
min      22.000000   45.000000   11.000000    0.000000   28.600000    0.700000    6.900000    0.000000    1.100000    0.000000    0.000000
25%      29.000000   60.000000   14.000000    0.000000   65.925000    3.725000   10.050000    1.125000    5.100000    0.500000    0.000000
50%      31.000000   68.000000   16.000000    0.000000   80.900000    9.450000   35.550000    2.650000   11.200000    3.000000    0.000000
75%      34.000000   77.750000   18.000000    0.500000   86.775000   16.300000   79.025000    5.600000   21.675000    8.700000    1.000000
max      37.000000   89.000000   26.000000   16.800000   90.300000   54.200000  220.400000   12.500000   67.400000   30.200000    1.000000

Process finished with exit code 0
```

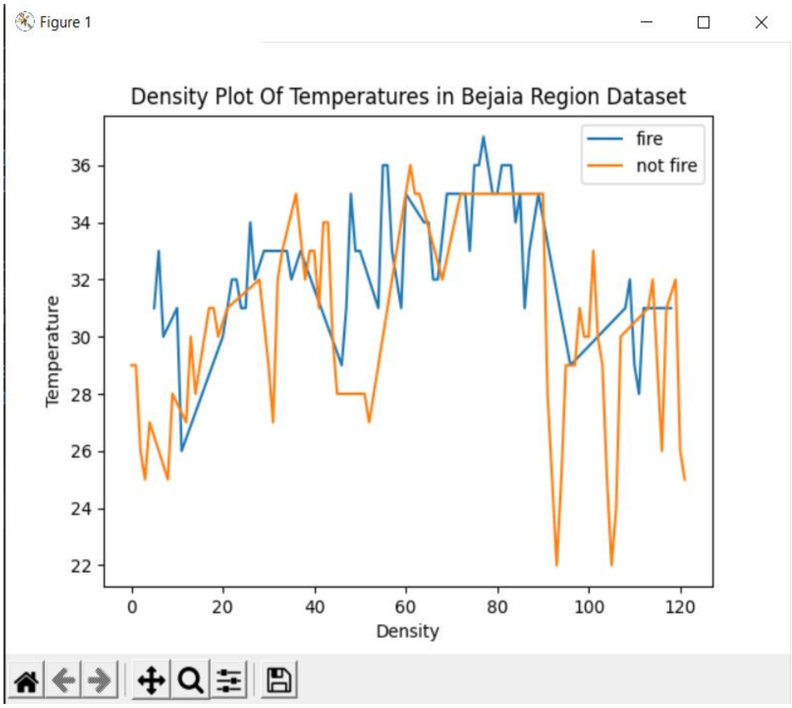
Task2:

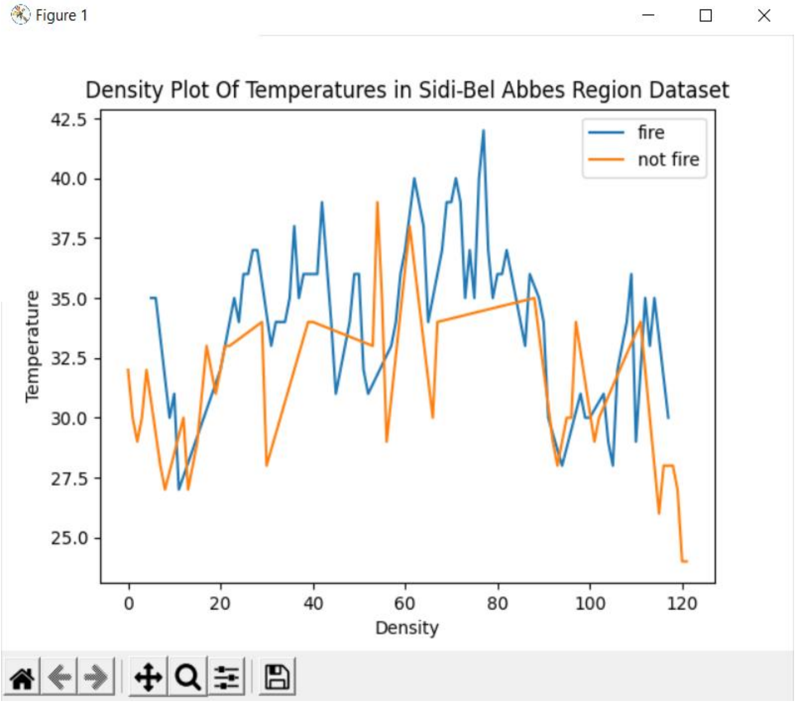
I split the data in two curves by the classes (fire, not fire) of Bejaia region dataset, then I set the title and x, y-axis label ,and use plot function to draw the density plot of temperature in the two data sets(fire and not fire data sets) ,then show the plt and legend(describe every line).

I use same steps to draw the density plot of sidi-Bel Abbes region dataset.



```
import matplotlib.pyplot as plt
splitedDataset = [rows for _, rows in dataSet1.groupby(l[13])]
plt.title('Density Plot Of Temperatures in Bejaia Region Dataset')
plt.ylabel('Temperature')
plt.xlabel('Density')
plt.plot(splitedDataset [0][l[3]],label = "fire")
plt.plot(splitedDataset [1][l[3]],label = "not fire")
plt.legend()
plt.show()
splitedDataset = [rows for _, rows in dataSet2.groupby(l[13])]
plt.title('Density Plot Of Temperatures in Sidi-Bel Abbes Region Dataset')
plt.ylabel('Temperature')
plt.xlabel('Density')
plt.plot(splitedDataset [0][l[3]],label = "fire")
plt.plot(splitedDataset [1][l[3]],label = "not fire")
plt.legend()
plt.show()
```

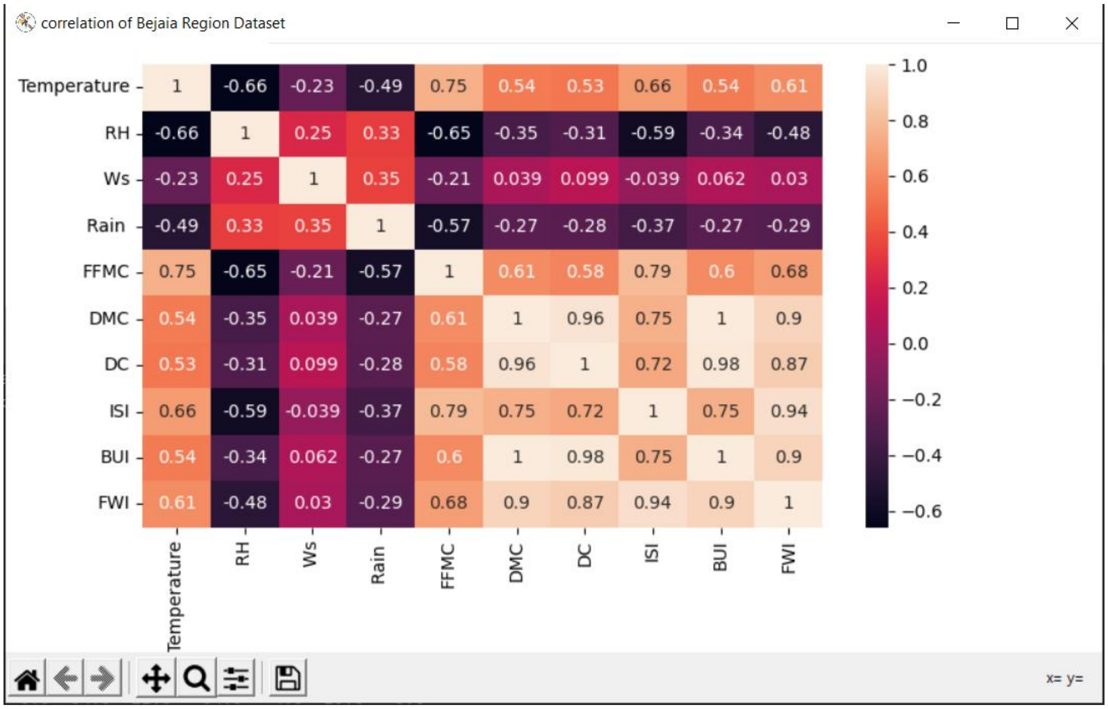




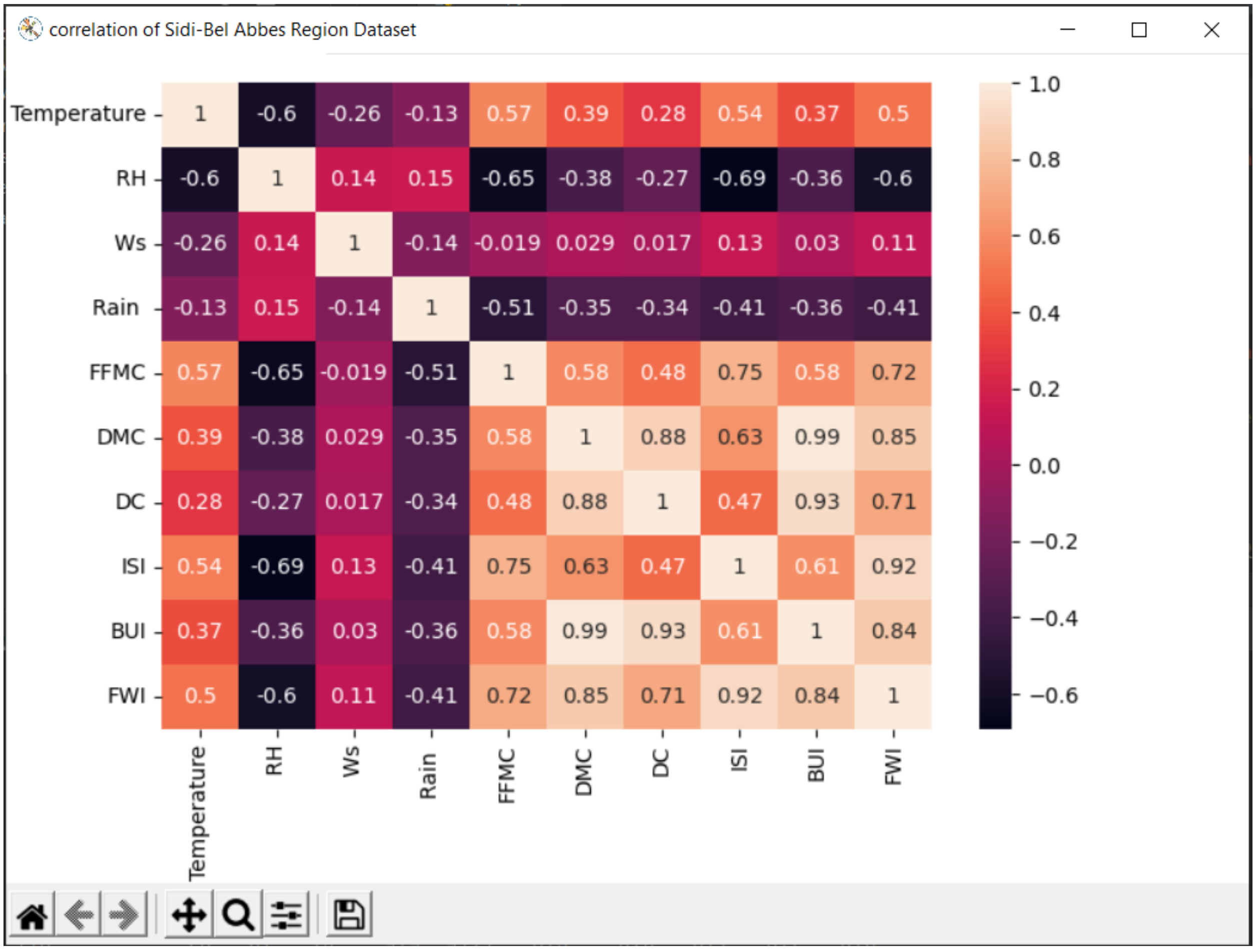
Task3:

First I use the function .corr() to get the correlation coefficient between all the features with out date and classes ,because the data will not be affected by the date and the classes is dependent feature, then I use heatmap to set labels and values of colors to figure to make it clear .

```
##task3
Var_Corr = dataSet1[l[3:13]].corr()
sns.heatmap(Var_Corr, xticklabels=Var_Corr.columns,
yticklabels=Var_Corr.columns, annot=True)
fig = pylab.gcf()
fig.canvas.manager.set_window_title('correlation of Bejaia Region Dataset')
plt.show()
# i use this line to check where the error (becouse that doesnt get the
feature DC and FWI)
# and i find that missing ',' between this 2 values (14.6 9) and i edit it
dataSet2[l[9]]=dataSet2[l[9]].astype(float)

Var_Corr = dataSet2[l[3:13]].corr()
sns.heatmap(Var_Corr, xticklabels=Var_Corr.columns,
yticklabels=Var_Corr.columns, annot=True)
fig = pylab.gcf()
fig.canvas.manager.set_window_title('correlation of Sidi-Bel Abbes Region
Dataset')
plt.show()
```


correlation of Bejaia Region Dataset

	Temperature	RH	Ws	Rain	FFMC	DMC	DC	ISI	BUI	FWI
Temperature	1	-0.66	-0.23	-0.49	0.75	0.54	0.53	0.66	0.54	0.61
RH	-0.66	1	0.25	0.33	-0.65	-0.35	-0.31	-0.59	-0.34	-0.48
Ws	-0.23	0.25	1	0.35	-0.21	0.039	0.099	-0.039	0.062	0.03
Rain	-0.49	0.33	0.35	1	-0.57	-0.27	-0.28	-0.37	-0.27	-0.29
FFMC	0.75	-0.65	-0.21	-0.57	1	0.61	0.58	0.79	0.6	0.68
DMC	0.54	-0.35	0.039	-0.27	0.61	1	0.96	0.75	1	0.9
DC	0.53	-0.31	0.099	-0.28	0.58	0.96	1	0.72	0.98	0.87
ISI	0.66	-0.59	-0.039	-0.37	0.79	0.75	0.72	1	0.75	0.94
BUI	0.54	-0.34	0.062	-0.27	0.6	1	0.98	0.75	1	0.9
FWI	0.61	-0.48	0.03	-0.29	0.68	0.9	0.87	0.94	0.9	1



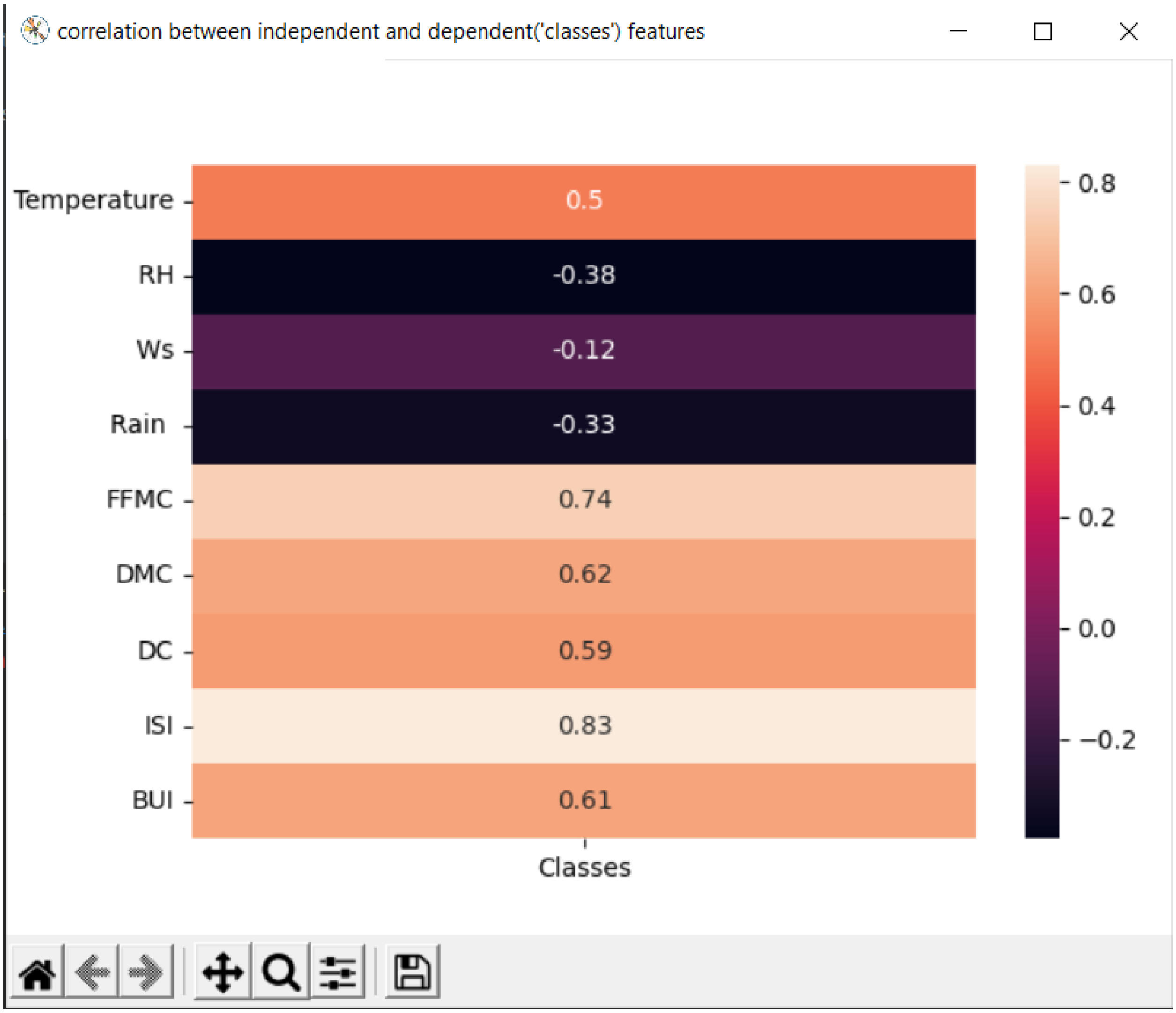
correlation of Sidi-Bel Abbes Region Dataset

	Temperature	RH	Ws	Rain	FFMC	DMC	DC	ISI	BUI	FWI
Temperature	1	-0.6	-0.26	-0.13	0.57	0.39	0.28	0.54	0.37	0.5
RH	-0.6	1	0.14	0.15	-0.65	-0.38	-0.27	-0.69	-0.36	-0.6
Ws	-0.26	0.14	1	-0.14	-0.019	0.029	0.017	0.13	0.03	0.11
Rain	-0.13	0.15	-0.14	1	-0.51	-0.35	-0.34	-0.41	-0.36	-0.41
FFMC	0.57	-0.65	-0.019	-0.51	1	0.58	0.48	0.75	0.58	0.72
DMC	0.39	-0.38	0.029	-0.35	0.58	1	0.88	0.63	0.99	0.85
DC	0.28	-0.27	0.017	-0.34	0.48	0.88	1	0.47	0.93	0.71
ISI	0.54	-0.69	0.13	-0.41	0.75	0.63	0.47	1	0.61	0.92
BUI	0.37	-0.36	0.03	-0.36	0.58	0.99	0.93	0.61	1	0.84
FWI	0.5	-0.6	0.11	-0.41	0.72	0.85	0.71	0.92	0.84	1

Task4:

First I use the function .corr() to get the correlation coefficient between all the independent features with classes(dependent feature), then I use heatmap to set labels and values of colors to figure to make it clear .

```
#task4
dependentFeature =dataSet1[l[13]]
dependentFeature.columns=['Classes']
Var_Corr = dataSet1[l[3:12]].corrwith(dependentFeature)
sns.heatmap(Var_Corr[:,np.newaxis], xticklabels=dependentFeature.columns,
yticklabels=dataSet1[l[3:12]].columns, annot=True)
fig = pylab.gcf()
fig.canvas.manager.set_window_title("correlation between independent and
dependent('classes') features")
plt.show()
```



Task5:

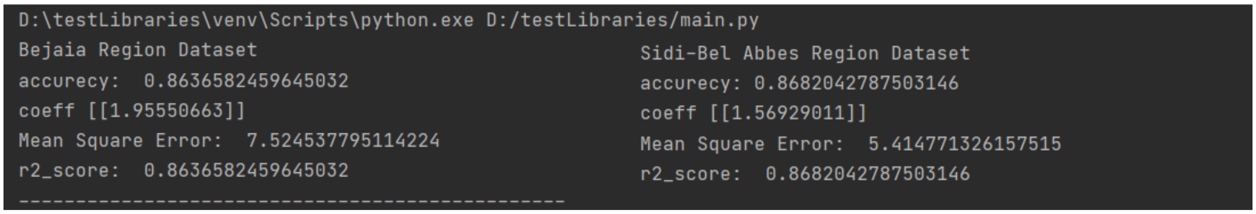
5.a

from the correlation coefficient that resulted from the task 3, I chose the ISI (Initial Spread Index) to decide FWI (Fire Weather Index) because it has the highest correlation, and that’s mean the FWI have the largest connection with the feature ISI.

5.b

First I get the feature have the highest correlation with FWI from task 3 and split the data of this feature(x) and FWI(y) 20:80 test:train and I make the model by the function linear.fit(), and get the predicted results of test data from model and Compare the expected answers with the real results to get the accuracy measured of this machine learning.

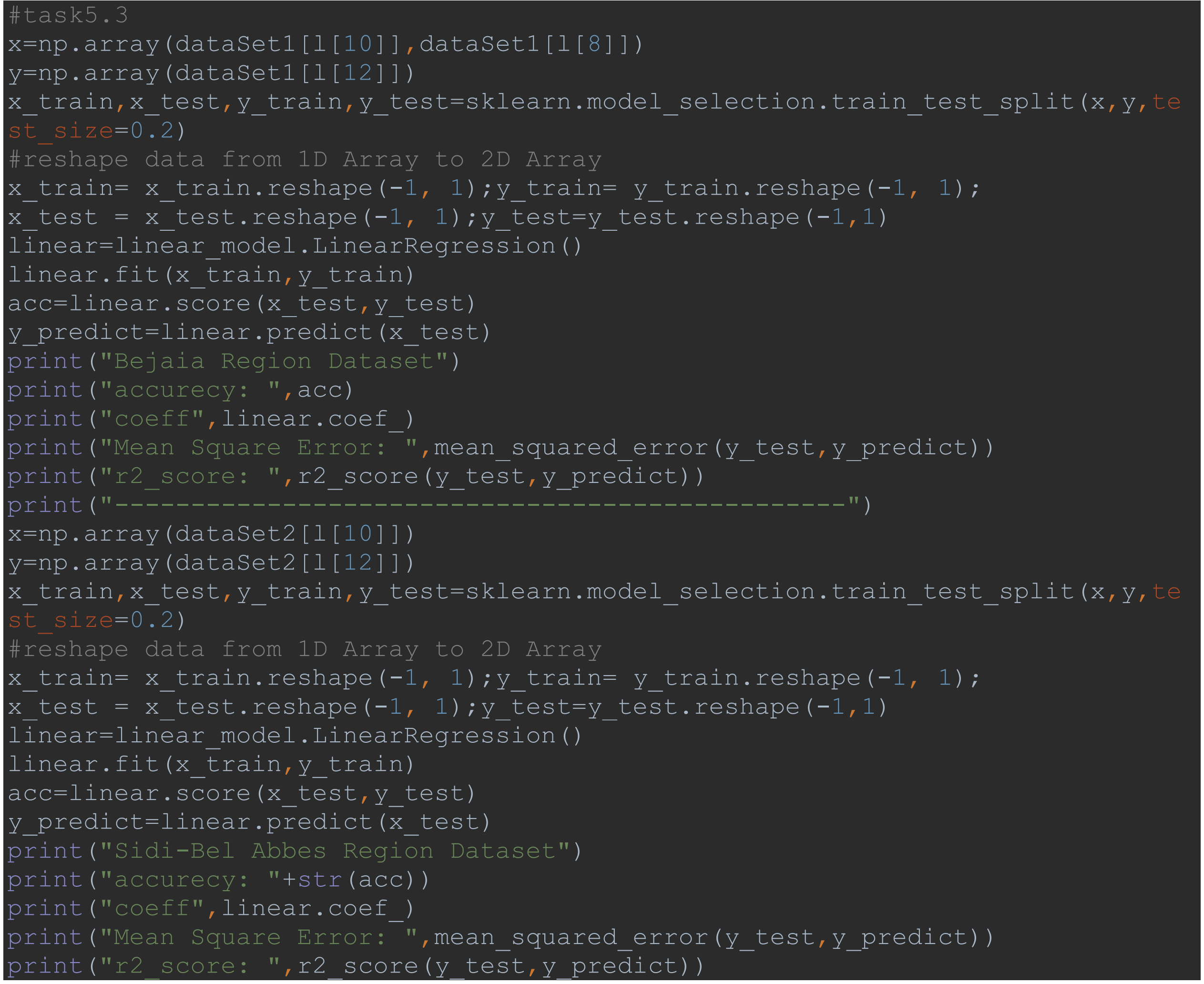
```
# task5.2
x=np.array(dataSet1[l[10]])
y=np.array(dataSet1[l[12]])
x_train,x_test,y_train,y_test=sklearn.model_selection.train_test_split(x,y,test_size=0.2)
#reshape data from 1D Array to 2D Array
x_train= x_train.reshape(-1, 1);y_train= y_train.reshape(-1, 1);
x_test = x_test.reshape(-1, 1);y_test=y_test.reshape(-1,1)
linear=linear_model.LinearRegression()
linear.fit(x_train,y_train)
acc=linear.score(x_test,y_test)
y_predict=linear.predict(x_test)
print("Bejaia Region Dataset")
print("accurecy: ",acc)
print("coeff",linear.coef_)
print("Mean Square Error: ",mean_squared_error(y_test,y_predict))
print("r2_score: ",r2_score(y_test,y_predict))
print("-------------------------------------------------")
x=np.array(dataSet2[l[10]])
y=np.array(dataSet2[l[12]])
x_train,x_test,y_train,y_test=sklearn.model_selection.train_test_split(x,y,test_size=0.2)
#reshape data from 1D Array to 2D Array
x_train= x_train.reshape(-1, 1);y_train= y_train.reshape(-1, 1);
x_test = x_test.reshape(-1, 1);y_test=y_test.reshape(-1,1)
linear=linear_model.LinearRegression()
linear.fit(x_train,y_train)
acc=linear.score(x_test,y_test)
y_predict=linear.predict(x_test)
print("Sidi-Bel Abbes Region Dataset")
print("accurecy: "+str(acc))
print("coeff",linear.coef_)
print("Mean Square Error: ",mean_squared_error(y_test,y_predict))
print("r2_score: ",r2_score(y_test,y_predict))
```



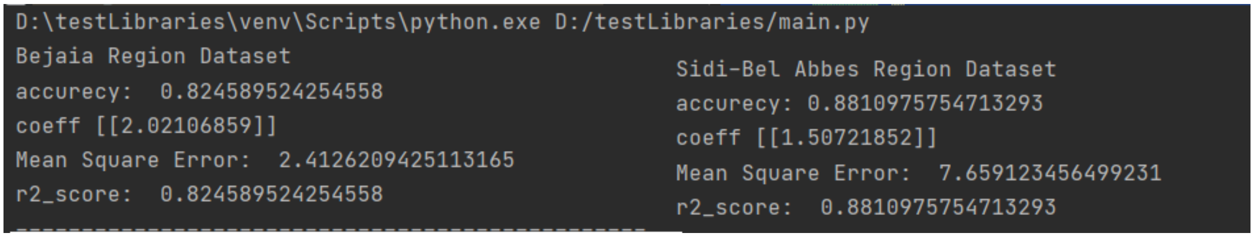
```
D:\testLibraries\venv\Scripts\python.exe D:/testLibraries/main.py
Bejaia Region Dataset                                  Sidi-Bel Abbes Region Dataset
accurecy:  0.8636582459645032                          accurecy: 0.8682042787503146
coeff [[1.95550663]]                                   coeff [[1.56929011]]
Mean Square Error:  7.524537795114224                  Mean Square Error:  5.414771326157515
r2_score:  0.8636582459645032                          r2_score:  0.8682042787503146
------------------------------------------------
```

5.c

First I get the 2 feature have the highest correlation with FWI from task 3 and split the data of this feature(x) and FWI(y) 20:80 test:train and I make the model by the function linear.fit(), and get the predicted results of test data from model and Compare the expected answers with the real results to get the accuracy measured of this machine learning.



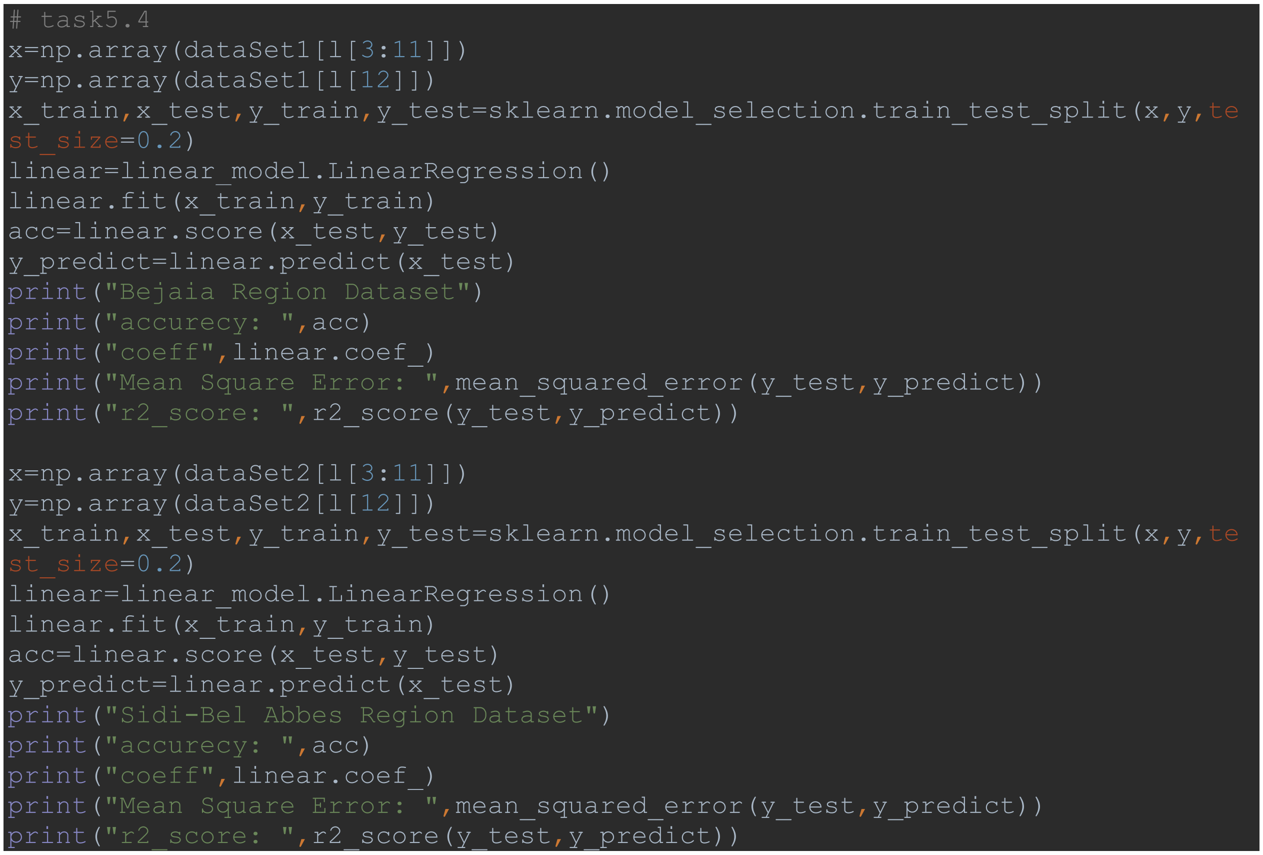
```
#task5.3
x=np.array(dataSet1[l[10]],dataSet1[l[8]])
y=np.array(dataSet1[l[12]])
x_train,x_test,y_train,y_test=sklearn.model_selection.train_test_split(x,y,test_size=0.2)
#reshape data from 1D Array to 2D Array
x_train= x_train.reshape(-1, 1);y_train= y_train.reshape(-1, 1);
x_test = x_test.reshape(-1, 1);y_test=y_test.reshape(-1,1)
linear=linear_model.LinearRegression()
linear.fit(x_train,y_train)
acc=linear.score(x_test,y_test)
y_predict=linear.predict(x_test)
print("Bejaia Region Dataset")
print("accurecy: ",acc)
print("coeff",linear.coef_)
print("Mean Square Error: ",mean_squared_error(y_test,y_predict))
print("r2_score: ",r2_score(y_test,y_predict))
print("------------------------------------------------")
x=np.array(dataSet2[l[10]])
y=np.array(dataSet2[l[12]])
x_train,x_test,y_train,y_test=sklearn.model_selection.train_test_split(x,y,test_size=0.2)
#reshape data from 1D Array to 2D Array
x_train= x_train.reshape(-1, 1);y_train= y_train.reshape(-1, 1);
x_test = x_test.reshape(-1, 1);y_test=y_test.reshape(-1,1)
linear=linear_model.LinearRegression()
linear.fit(x_train,y_train)
acc=linear.score(x_test,y_test)
y_predict=linear.predict(x_test)
print("Sidi-Bel Abbes Region Dataset")
print("accurecy: "+str(acc))
print("coeff",linear.coef_)
print("Mean Square Error: ",mean_squared_error(y_test,y_predict))
print("r2_score: ",r2_score(y_test,y_predict))
```



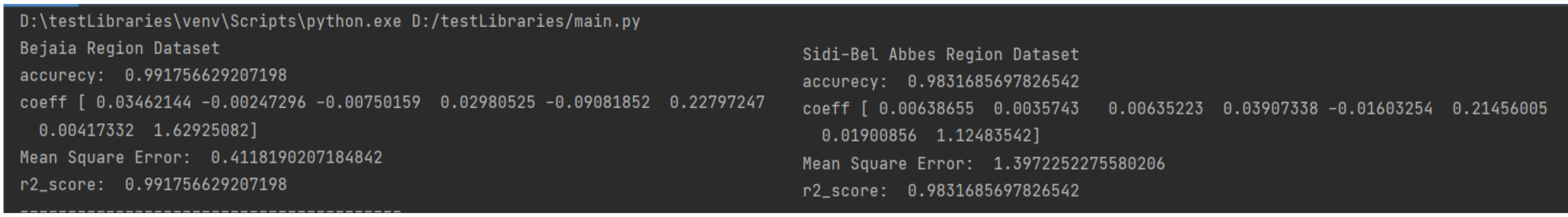
```
D:\testLibraries\venv\Scripts\python.exe D:/testLibraries/main.py
Bejaia Region Dataset
accurecy:  0.824589524254558
coeff [[2.02106859]]
Mean Square Error:  2.4126209425113165
r2_score:  0.824589524254558
-------------------------------------------

Sidi-Bel Abbes Region Dataset
accurecy: 0.8810975754713293
coeff [[1.50721852]]
Mean Square Error:  7.659123456499231
r2_score:  0.8810975754713293
```

5.d

First Isplit the data of the all features(x) and FWI(y) 20:80 test:train and I make the model by the function linear.fit(), and get the predicted results of test data from model and Compare the expected answers with the real results to get the accuracy measured of this machine learning.



```python
# task5.4
x=np.array(dataSet1[l[3:11]])
y=np.array(dataSet1[l[12]])
x_train,x_test,y_train,y_test=sklearn.model_selection.train_test_split(x,y,test_size=0.2)
linear=linear_model.LinearRegression()
linear.fit(x_train,y_train)
acc=linear.score(x_test,y_test)
y_predict=linear.predict(x_test)
print("Bejaia Region Dataset")
print("accurecy: ",acc)
print("coeff",linear.coef_)
print("Mean Square Error: ",mean_squared_error(y_test,y_predict))
print("r2_score: ",r2_score(y_test,y_predict))

x=np.array(dataSet2[l[3:11]])
y=np.array(dataSet2[l[12]])
x_train,x_test,y_train,y_test=sklearn.model_selection.train_test_split(x,y,test_size=0.2)
linear=linear_model.LinearRegression()
linear.fit(x_train,y_train)
acc=linear.score(x_test,y_test)
y_predict=linear.predict(x_test)
print("Sidi-Bel Abbes Region Dataset")
print("accurecy: ",acc)
print("coeff",linear.coef_)
print("Mean Square Error: ",mean_squared_error(y_test,y_predict))
print("r2_score: ",r2_score(y_test,y_predict))
```



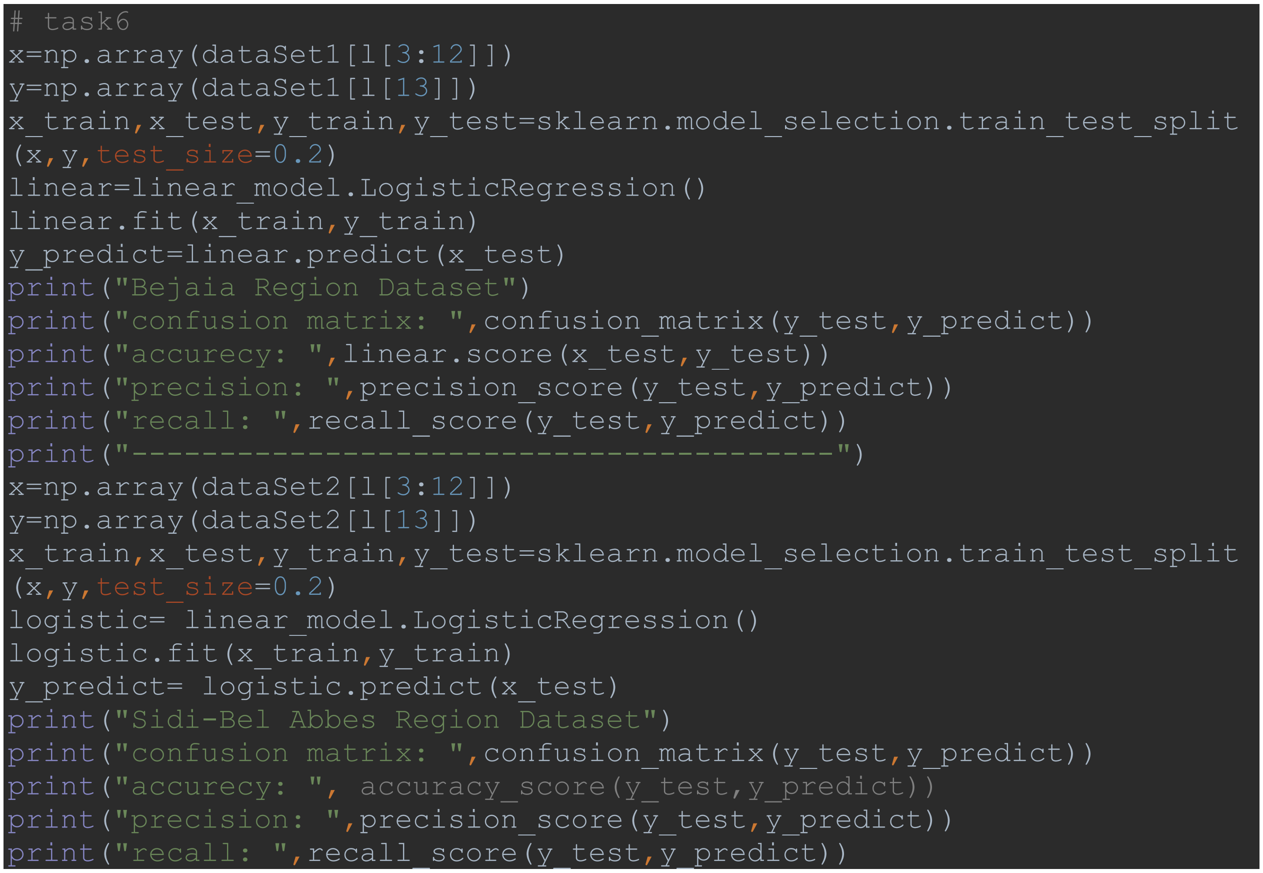
```
D:\testLibraries\venv\Scripts\python.exe D:/testLibraries/main.py
Bejaia Region Dataset
accurecy:  0.991756629207198
coeff [ 0.03462144 -0.00247296 -0.00750159  0.02980525 -0.09081852  0.22797247
  0.00417332  1.62925082]
Mean Square Error:  0.4118190207184842
r2_score:  0.991756629207198
----------------------------------------

Sidi-Bel Abbes Region Dataset
accurecy:  0.9831685697826542
coeff [ 0.00638655  0.0035743   0.00635223  0.03907338 -0.01603254  0.21456005
  0.01900856  1.12483542]
Mean Square Error:  1.3972252275580206
r2_score:  0.9831685697826542
```

5.e

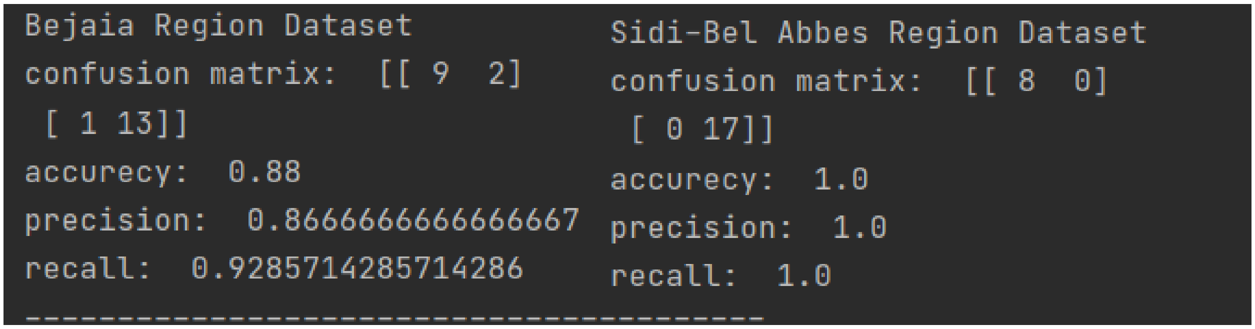
The accuracy results of FWI are increasing when we add more features to make the linear regression model, that’s mean the predicted results of model will be more correct when we add more features have the relation with FWI.

Task6:

I split the data 20:80, test:train I made the module by logistic.fit() and get the predicted results of test data from model and Compare the expected answers with the real results to get the accuracy measured of this machine learning.



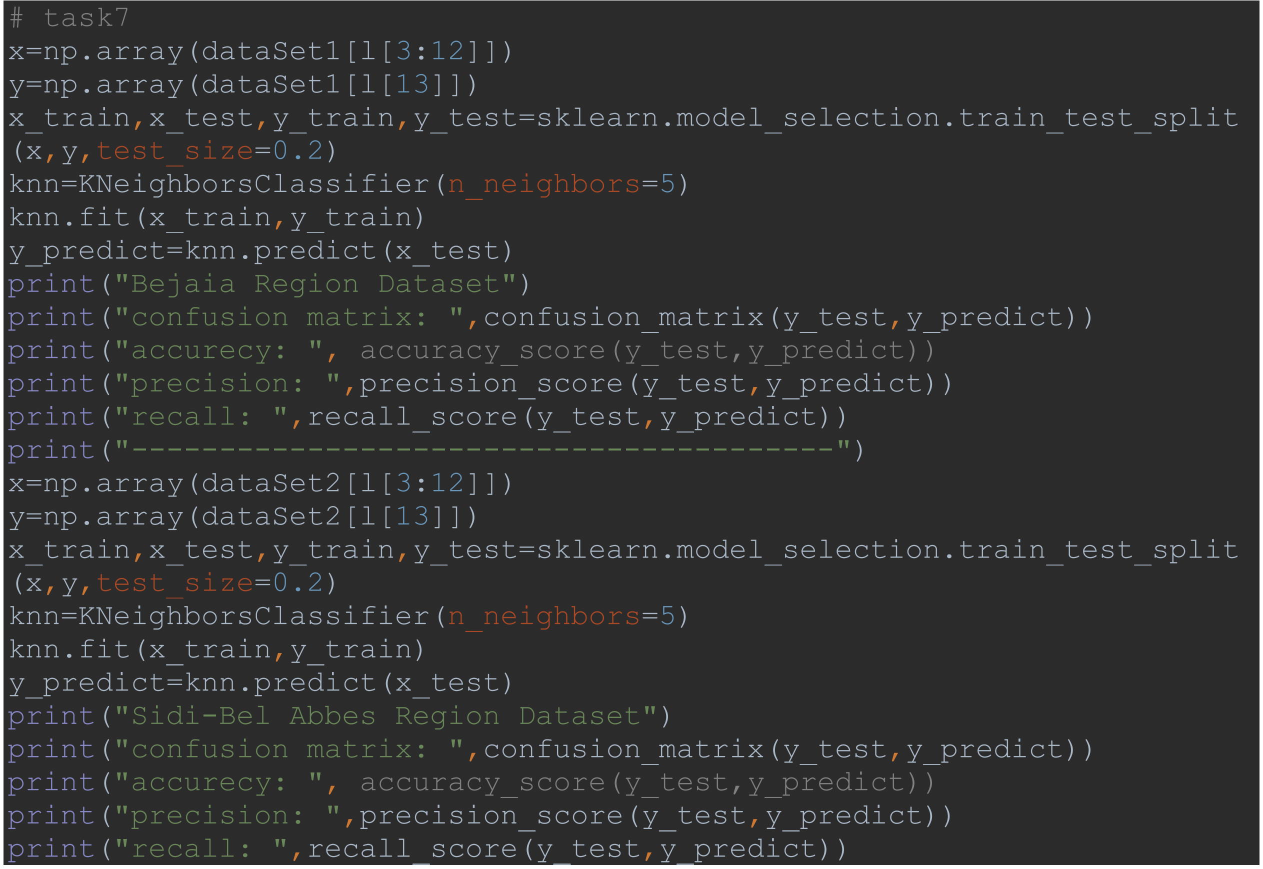
```
# task6
x=np.array(dataSet1[l[3:12]])
y=np.array(dataSet1[l[13]])
x_train,x_test,y_train,y_test=sklearn.model_selection.train_test_split
(x,y,test_size=0.2)
linear=linear_model.LogisticRegression()
linear.fit(x_train,y_train)
y_predict=linear.predict(x_test)
print("Bejaia Region Dataset")
print("confusion matrix: ",confusion_matrix(y_test,y_predict))
print("accurecy: ",linear.score(x_test,y_test))
print("precision: ",precision_score(y_test,y_predict))
print("recall: ",recall_score(y_test,y_predict))
print("----------------------------------------")
x=np.array(dataSet2[l[3:12]])
y=np.array(dataSet2[l[13]])
x_train,x_test,y_train,y_test=sklearn.model_selection.train_test_split
(x,y,test_size=0.2)
logistic= linear_model.LogisticRegression()
logistic.fit(x_train,y_train)
y_predict= logistic.predict(x_test)
print("Sidi-Bel Abbes Region Dataset")
print("confusion matrix: ",confusion_matrix(y_test,y_predict))
print("accurecy: ", accuracy_score(y_test,y_predict))
print("precision: ",precision_score(y_test,y_predict))
print("recall: ",recall_score(y_test,y_predict))
```



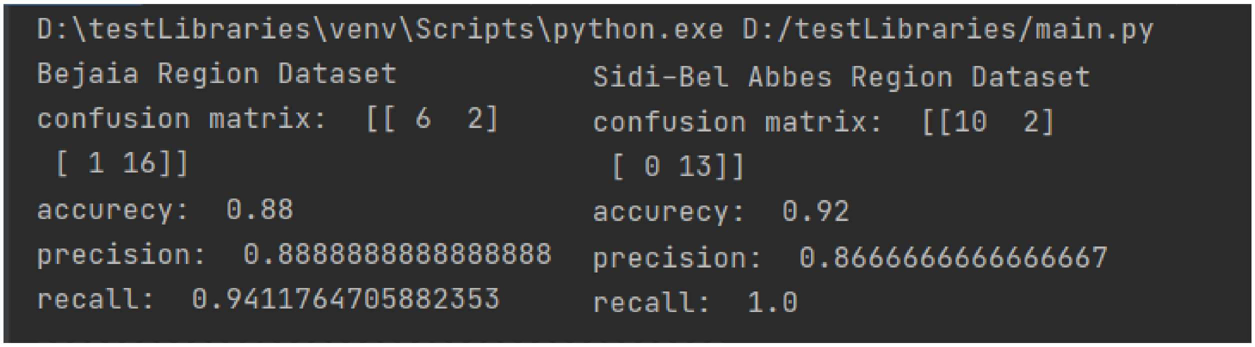
```
Bejaia Region Dataset                        Sidi-Bel Abbes Region Dataset
confusion matrix:  [[ 9  2]                  confusion matrix:  [[ 8  0]
 [ 1 13]]                                     [ 0 17]]
accurecy:  0.88                              accurecy:  1.0
precision:  0.8666666666666667               precision:  1.0
recall:  0.9285714285714286                  recall:  1.0
----------------------------------------
```

Task7:

I split the data 20:80, test:train I made the module by knn.fit() and get the predicted results of test data from model and Compare the expected answers with the real results to get the accuracy measured of this machine learning.



```
# task7
x=np.array(dataSet1[l[3:12]])
y=np.array(dataSet1[l[13]])
x_train,x_test,y_train,y_test=sklearn.model_selection.train_test_split
(x,y,test_size=0.2)
knn=KNeighborsClassifier(n_neighbors=5)
knn.fit(x_train,y_train)
y_predict=knn.predict(x_test)
print("Bejaia Region Dataset")
print("confusion matrix: ",confusion_matrix(y_test,y_predict))
print("accurecy: ", accuracy_score(y_test,y_predict))
print("precision: ",precision_score(y_test,y_predict))
print("recall: ",recall_score(y_test,y_predict))
print("----------------------------------------")
x=np.array(dataSet2[l[3:12]])
y=np.array(dataSet2[l[13]])
x_train,x_test,y_train,y_test=sklearn.model_selection.train_test_split
(x,y,test_size=0.2)
knn=KNeighborsClassifier(n_neighbors=5)
knn.fit(x_train,y_train)
y_predict=knn.predict(x_test)
print("Sidi-Bel Abbes Region Dataset")
print("confusion matrix: ",confusion_matrix(y_test,y_predict))
print("accurecy: ", accuracy_score(y_test,y_predict))
print("precision: ",precision_score(y_test,y_predict))
print("recall: ",recall_score(y_test,y_predict))
```



```
D:\testLibraries\venv\Scripts\python.exe D:/testLibraries/main.py
Bejaia Region Dataset               Sidi-Bel Abbes Region Dataset
confusion matrix:  [[ 6  2]         confusion matrix:  [[10  2]
 [ 1 16]]                            [ 0 13]]
accurecy:  0.88                     accurecy:  0.92
precision:  0.8888888888888888      precision:  0.8666666666666667
recall:  0.9411764705882353         recall:  1.0
```

Task8:

First, I need to explain the difference between this 3-accuracy measured

- Accuracy: calculate the sum of true positive (TP) and true Negative (TN) and divide the sum by all the results (False positive (FP), False Negative (FN), True Positive (TP), True Negative (TN)), that’s mean divide the results that the algorithm expects by all the results (which it expected and did not expect by algorithm), and this accuracy measure will be not fair if the most data are TP or TN.
- Precession: divide true positive (TP) by the sum of (False positive (FP), True Positive (TP)), that’s mean divide the results that’s true and the algorithm expect it by all the true results (which it expected and did not expect by algorithm).
- Recall: divide true positive (TP) by the sum of (True positive (TP), False Negative (FN)), that’s mean divide the results that’s true and the algorithm expect it by all the results that algorithm expects it.

The accuracy of results from two algorithms (logistic regression and kNN) are very similar, and when we change the training and test data the results will be different and very close to each other, and in the results I put it in the above they show the Logistic regression are best in this 3 accuracy measures .