Name-Surname:Kaan GÖKÇEK
Student ID:20150601020

CE 475 – Fundamentals and Applications of Machine Learning

Project Report

İZMİR UNIVERSITY OF ECONOMICS

1. Introduction

In CE 475 Fall course 2019, we learn many supervised learning techniques. In this project, we suppose to predict y values according to 6 x values in the given csv list. In the csv, we have 120 values. First 100 x values have include y values itself, last 20 x value's y needs to be predicted by supervised techniques. Furthermore, cross validation and many regression methods will be used in this project.

2. Methodology

Mainly, I import sklearn and other tools to use in future.

```python
import csv
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_squared_error, r2_score
```

Next, I read csv and separate first 100 and 20 values because last 20 data does not contain y values.

```python
x1 = np.array([])
x2 = np.array([])
x3 = np.array([])
x4 = np.array([])
x5 = np.array([])
x6 = np.array([])
y = np.array([])
x1a = np.array([])
x2a = np.array([])
x3a = np.array([])
x4a = np.array([])
x5a = np.array([])
x6a = np.array([])
```

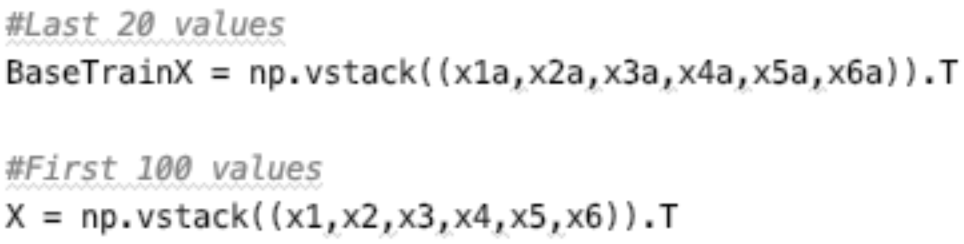
```python
with open("CE 475 Fall 2019 Project Data - Data.csv", encoding="Latin-1") as f:
    csv_list = list(csv.reader(f))

firstHundred = csv_list[:101]
lastTwenty = csv_list[100:]

#First 100 calues
for row in firstHundred:
    if row != firstHundred[0]:
        x1 = np.append(x1,int(row[1]))
        x2 = np.append(x2,int(row[2]))
        x3 = np.append(x3,int(row[3]))
        x4 = np.append(x4,int(row[4]))
        x5 = np.append(x5,int(row[5]))
        x6 = np.append(x6,int(row[6]))
        y = np.append(y, int(row[7]))

#Last 20 x values
for row in lastTwenty:
    if row != lastTwenty[0]:
        x1a = np.append(x1a,int(row[1]))
        x2a = np.append(x2a,int(row[2]))
        x3a = np.append(x3a,int(row[3]))
        x4a = np.append(x4a,int(row[4]))
        x5a = np.append(x5a,int(row[5]))
        x6a = np.append(x6a,int(row[6]))
```

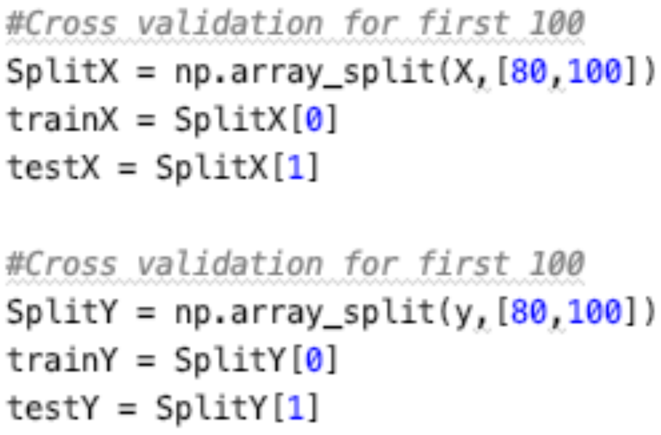
Then, I create the test and train values of x.



```
#Last 20 values
BaseTrainX = np.vstack((x1a,x2a,x3a,x4a,x5a,x6a)).T

#First 100 values
X = np.vstack((x1,x2,x3,x4,x5,x6)).T
```

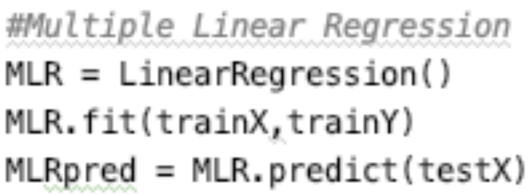
For testing which supervised learning is good for our data, I split the data with %20 for test and train.



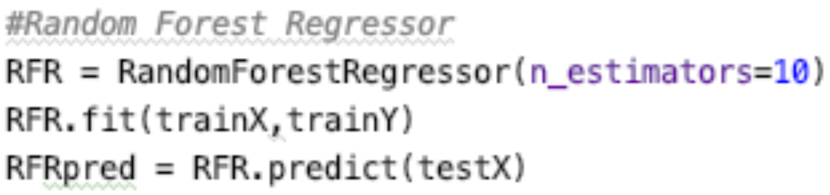
```
#Cross validation for first 100
SplitX = np.array_split(X,[80,100])
trainX = SplitX[0]
testX = SplitX[1]

#Cross validation for first 100
SplitY = np.array_split(y,[80,100])
trainY = SplitY[0]
testY = SplitY[1]
```

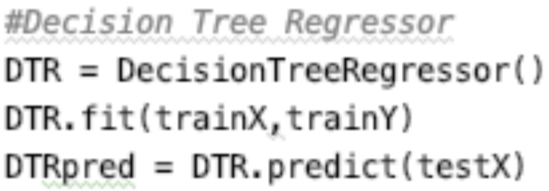
After that operation, I try to analyze data with regression methods.



```
#Multiple Linear Regression
MLR = LinearRegression()
MLR.fit(trainX,trainY)
MLRpred = MLR.predict(testX)
```

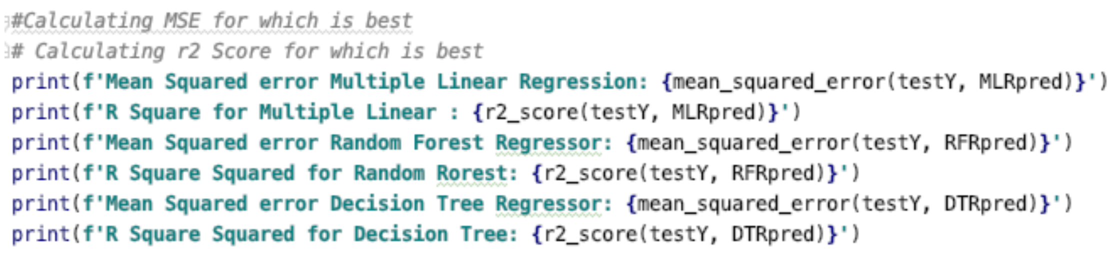


```
#Random Forest Regressor
RFR = RandomForestRegressor(n_estimators=10)
RFR.fit(trainX,trainY)
RFRpred = RFR.predict(testX)
```

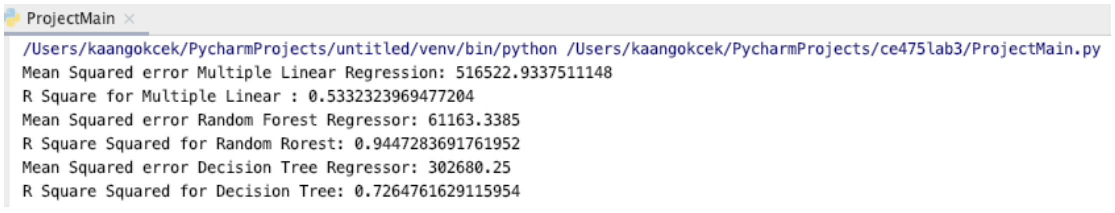


```
#Decision Tree Regressor
DTR = DecisionTreeRegressor()
DTR.fit(trainX,trainY)
DTRpred = DTR.predict(testX)
```

The results I get showed me that I should use Random Forest Regressor. Because I see that MSE is lower that other supervised learning techniques and r^2 is close to 1 in that technique.

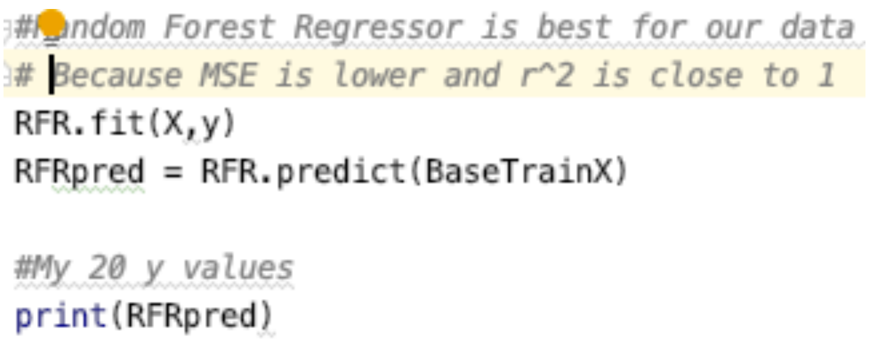


```
#Calculating MSE for which is best
# Calculating r2 Score for which is best
print(f'Mean Squared error Multiple Linear Regression: {mean_squared_error(testY, MLRpred)}')
print(f'R Square for Multiple Linear : {r2_score(testY, MLRpred)}')
print(f'Mean Squared error Random Forest Regressor: {mean_squared_error(testY, RFRpred)}')
print(f'R Square Squared for Random Rorest: {r2_score(testY, RFRpred)}')
print(f'Mean Squared error Decision Tree Regressor: {mean_squared_error(testY, DTRpred)}')
print(f'R Square Squared for Decision Tree: {r2_score(testY, DTRpred)}')
```



```
/Users/kaangokcek/PycharmProjects/untitled/venv/bin/python /Users/kaangokcek/PycharmProjects/ce475lab3/ProjectMain.py
Mean Squared error Multiple Linear Regression: 516522.9337511148
R Square for Multiple Linear : 0.5332323969477204
Mean Squared error Random Forest Regressor: 61163.3385
R Square Squared for Random Rorest: 0.9447283691761952
Mean Squared error Decision Tree Regressor: 302680.25
R Square Squared for Decision Tree: 0.7264761629115954
```

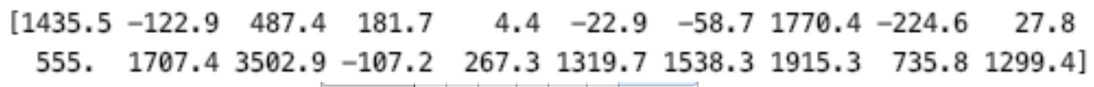
Lastly, I train first 100 x values with 100 y values and then fit and also predict last 20 values according to last 20 x(x1,x2,x3,x4,x5,x6) values.



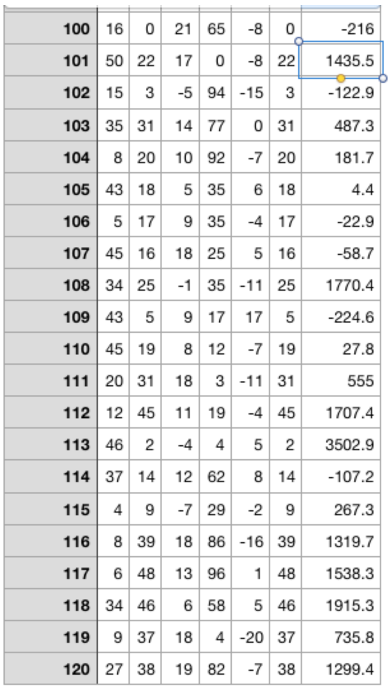
```
#Random Forest Regressor is best for our data
# Because MSE is lower and r^2 is close to 1
RFR.fit(X,y)
RFRpred = RFR.predict(BaseTrainX)

#My 20 y values
print(RFRpred)
```

The Results of last 20 y values are :



```
[1435.5 -122.9  487.4  181.7    4.4  -22.9  -58.7 1770.4 -224.6   27.8
  555.  1707.4 3502.9 -107.2  267.3 1319.7 1538.3 1915.3  735.8 1299.4]
```



100	16	0	21	65	-8	0	-216
101	50	22	17	0	-8	22	1435.5
102	15	3	-5	94	-15	3	-122.9
103	35	31	14	77	0	31	487.3
104	8	20	10	92	-7	20	181.7
105	43	18	5	35	6	18	4.4
106	5	17	9	35	-4	17	-22.9
107	45	16	18	25	5	16	-58.7
108	34	25	-1	35	-11	25	1770.4
109	43	5	9	17	17	5	-224.6
110	45	19	8	12	-7	19	27.8
111	20	31	18	3	-11	31	555
112	12	45	11	19	-4	45	1707.4
113	46	2	-4	4	5	2	3502.9
114	37	14	12	62	8	14	-107.2
115	4	9	-7	29	-2	9	267.3
116	8	39	18	86	-16	39	1319.7
117	6	48	13	96	1	48	1538.3
118	34	46	6	58	5	46	1915.3
119	9	37	18	4	-20	37	735.8
120	27	38	19	82	-7	38	1299.4

3. Conclusion

To sum up, essentially I experienced with above supervised learning techniques. With the Random Forest Regressor, I get lower MSE error and closer to 1 R^2 score in this data prediction. Since we don't know the last 20 y values, I apply this Regressor to first 100 values with separated %20 test data. And then, I implemented for last 20 y values to predict them.

As we seen above, the data may not be perform good results on every regressor. It can be overfit or predict data values far from it can be.

In this course, I learned to use python libraries, advanced python methods and machine learning basics.