CE 475 Term Project Report

Oktay Özkan [1]

[1] *Computer Engineering, Engineering Faculty, Izmir University of Economics*
Teleferik Mahallesi, Sakarya Cd. No:156, 35330 Balçova/İzmir, Republic of Turkey

---

1.Identification and Significance of Problem

In my term project, my problem is that; I have a dataset which contains 100 train data (every x and y values are known) (Fig.1). But I do not know the first 20 y values after hundredth data.(Fig.2) I should predict the y values as best as I can.

SampleNo	x1	x2	x3	x4	x5	Y
1	32	37	10	41	6	68
2	72	12	13	69	-6	1266
3	41	31	12	43	-10	3094
4	22	45	-1	95	-8	89
5	90	17	11	72	6	1519
6	44	35	-6	82	-7	71
7	21	28	16	41	0	3265
8	49	30	4	93	-13	1532
.	.	.	.	.	.	.
.	.	.	.	.	.	.
92	23	26	4	4	6	862
93	82	0	-10	95	-2	12
94	26	26	17	20	5	2989
95	83	50	-10	34	-5	107
96	88	28	-10	36	-7	-1638
97	57	32	12	52	-13	68
98	12	30	9	39	10	2104
99	29	35	0	16	-15	76
100	24	22	-5	44	-12	-268

Fig.1- Train dataset

101	55	12	4	36	2	
102	33	48	14	67	8	
103	7	50	18	84	18	
104	32	38	-3	81	-6	
105	71	17	16	73	6	
106	49	47	-9	92	-10	
107	26	48	6	92	4	
108	12	35	13	17	-17	
109	27	1	-10	83	-12	
110	26	29	18	15	7	
111	65	5	17	44	8	
112	63	36	-10	57	-10	
113	88	22	-10	47	1	
114	75	15	1	82	-4	
115	18	28	-3	59	2	
116	93	39	18	21	-3	
117	16	47	10	32	-14	
118	52	24	24	27	0	
119	94	17	-6	42	-16	
120	47	39	2	94	0	

Fig.2- Unknown y values

2.Methodology and Implementation

At the beginning , I have used multiple linear regression approach. I have implemented my code according to this formula: y=a*x1+ b*x2+c*x3+d*x4+e*x5 . I have chosen this approach because I could not found how ı will predict y values using 5 different x values. In my first implementation, first I have founded a, b ,c, d, e values according to my train dataset.(Fig.1) And then I have predicted y values and calculated the variance. My variance is 0.4134904260249598. It means that, my error difference between train set and test set is much big.(Fig.3)

```
a,b,c,d,e:
[[  4.80069386]
 [-16.64459272]
 [ 85.94046075]
 [  7.14395891]
 [ 13.72313054]]
*******************************
     SampleNo  x1  x2  x3  x4  x5            Y
100       101  55  12   4  36   2   692.693674
101       102  33  48  14  67   8  1151.079188
102       103   7  50  18  84  18  1595.412412
103       104  32  38  -3  81  -6  -240.371814
104       105  71  17  16  73   6  2036.786343
105       106  49  47  -9  92 -10  -800.513091
106       107  26  48   6  92   4   553.657096
107       108  12  35  13  17 -17   480.427653
108       109  27   1 -10  83 -12  -318.159443
109       110  26  29  18  15   7  1392.274442
110       111  65   5  17  44   8  2113.929206
111       112  63  36 -10  57 -10  -886.191880
112       113  88  22 -10  47   1  -453.635388
113       114  75  15   1  82  -4   727.235718
114       115  18  28  -3  59   2  -188.517652
115       116  93  39  18  21  -3  1453.107452
116       117  16  47  10  32 -14   190.402709
117       118  52  24  24  27   0  2105.623804
118       119  94  17  -6  42 -16  -266.859432
119       120  47  39   2  94   0   419.906554
Variance:  0.4134904260249598
```

Fig.3-(a, b, c, d, e) values, predicted y values and variance

Because of big error difference between train set and test set, I decided to modify my code. My professor said that some x features may not be required. Due to this hint in order to find unnecessary x features I have chosen backward elimination technique. Using backward elimination technique I have eliminated x1 and x4 features according to their p-values.(Fig.4) (p-values of x1 and x4 are greater than my significance level(SL)). After this elimination, I have predicted y values according to ('x2','x3','x5') features.(Fig.5)

```python
#Backward Elimination to eliminate parameter which p value>sl

import statsmodels.formula.api as sm

x2.insert(0, "x0", 1)

x2_optimal_model = x2[['x0', 'x1', 'x2', 'x3', 'x4', 'x5']]
Ordinary_Least_Squares = sm.OLS(endog = y2, exog = x2_optimal_model).fit()
print(Ordinary_Least_Squares.summary())
#x1 was eliminated p=0,967
x2_optimal_model = x2[['x0', 'x2', 'x3', 'x4', 'x5']]
Ordinary_Least_Squares= sm.OLS(endog = y2, exog = x2_optimal_model).fit()
print(Ordinary_Least_Squares.summary())
#x4 was eliminated p=0,872
x2_optimal_model = x2[['x0', 'x2', 'x3', 'x5']]
Ordinary_Least_Squares = sm.OLS(endog = y2, exog = x2_optimal_model).fit()
print(Ordinary_Least_Squares.summary())
```

Fig.4-Backward Elimination

```python
#Predict y
x2_train=df1[['x2','x3','x5']]
y2_train=df1.Y
x2_test=df2[['x2','x3','x5']]
linearRegression.fit(x2_train, y2_train)
df2['Y'] = linearRegression.predict(x2_test)
print(df2)
```

Fig.5-Predict y

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      Y   R-squared:                       0.467
Model:                            OLS   Adj. R-squared:                  0.439
Method:                 Least Squares   F-statistic:                     16.47
Date:                Tue, 04 Dec 2018   Prob (F-statistic):           1.20e-11
Time:                        23:35:10   Log-Likelihood:                -825.21
No. Observations:                 100   AIC:                             1662.
Df Residuals:                      94   BIC:                             1678.
Df Model:                           5
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
x0          1113.1040    349.862      3.182      0.002     418.446    1807.762
x1            -0.1481      3.556     -0.042      0.967      -7.208       6.911
x2           -32.1007      7.468     -4.298      0.000     -46.929     -17.272
x3            82.6744     10.004      8.264      0.000      62.810     102.538
x4             0.5974      3.631      0.165      0.870      -6.612       7.807
x5            12.2140      8.437      1.448      0.151      -4.538      28.966
==============================================================================
Omnibus:                        6.029   Durbin-Watson:                   1.890
Prob(Omnibus):                  0.049   Jarque-Bera (JB):                5.501
Skew:                           0.555   Prob(JB):                       0.0639
Kurtosis:                       3.298   Cond. No.                         296.
==============================================================================
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      Y   R-squared:                       0.467
Model:                            OLS   Adj. R-squared:                  0.444
Method:                 Least Squares   F-statistic:                     20.80
Date:                Tue, 04 Dec 2018   Prob (F-statistic):           2.44e-12
Time:                        23:35:10   Log-Likelihood:                -825.21
No. Observations:                 100   AIC:                             1660.
Df Residuals:                      95   BIC:                             1673.
Df Model:                           4
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
x0          1106.7271    312.948      3.536      0.001     485.447    1728.008
x2           -32.0779      7.409     -4.330      0.000     -46.787     -17.369
x3            82.6731      9.952      8.307      0.000      62.916     102.430
x4             0.5788      3.585      0.161      0.872      -6.538       7.696
x5            12.2223      8.390      1.457      0.148      -4.434      28.879
==============================================================================
Omnibus:                        5.964   Durbin-Watson:                   1.889
Prob(Omnibus):                  0.051   Jarque-Bera (JB):                5.432
Skew:                           0.552   Prob(JB):                       0.0661
Kurtosis:                       3.295   Cond. No.                         217.
```

2.1.Libraries

```
@author: oktay
"""
#Libraries
import numpy as np
import pandas as pd
import warnings
warnings.filterwarnings('ignore')
from pandas.core.common import SettingWithCopyWarning
warnings.filterwarnings(action = 'ignore',category = SettingWithCopyWarning)
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
import statsmodels.formula.api as sm
from sklearn.model_selection import cross_val_score
```

3.Result

After I predict y values, in order to compare approach 1 ['x1','x2','x3','x4','x5'] and approach 2 ['x2','x3','x5'] I calculated root mean squared error.(Fig.6) RMSE of approach 1 is greater than RMSE of approach 2. (Fig.7) That is to say , approach 2 ['x2','x3','x5'] is better than the other.

```
from sklearn.model_selection import cross_val_score

print("Mean of rmse [ 'x2','x3','x5' ]: ", np.sqrt(-cross_val_score(linearRegression, x3, y, cv=10, scoring='neg_mean_squared_error')).mean())
print("Mean of rmse [ 'x1','x2','x3','x4','x5' ]: ", np.sqrt(-cross_val_score(linearRegression, x2, y, cv=10, scoring='neg_mean_squared_error')).mean())
```

Fig.6-Code of RMSE

```
     SampleNo  x1  x2  x3  x4  x5            Y
100       101  55  12   4  36   2  1110.362913
101       102  33  48  14  67   8   850.545366
102       103   7  50  18  84  18  1238.375785
103       104  32  38  -3  81  -6  -403.082481
104       105  71  17  16  73   6  1990.107544
105       106  49  47  -9  92 -10 -1237.671312
106       107  26  48   6  92   4   140.480205
107       108  12  35  13  17 -17   882.785903
108       109  27   1 -10  83 -12   136.991003
109       110  26  29  18  15   7  1781.106779
110       111  65   5  17  44   8  2483.610998
111       112  63  36 -10  57 -10  -966.046072
112       113  88  22 -10  47   1  -381.435654
113       114  75  15   1  82  -4   692.774456
114       115  18  28  -3  59   2    16.232413
115       116  93  39  18  21  -3  1337.487548
116       117  16  47  10  32 -14   284.679171
117       118  52  24  24  27   0  2353.182738
118       119  94  17  -6  42 -16   -96.245501
119       120  47  39   2  94   0    51.031079
Mean of rmse [ 'x2','x3','x5' ]:  962.5889085840488
Mean of rmse [ 'x1','x2','x3','x4','x5' ]:  1000.0719831858909
```

Fig.7-Result

4.Conclusion

I have learned fundamentals of machine learning and statistical math concept. I can compare single linear and multiple linear regression and I know which one is more effective in which cases. Furthermore I have learned Python language and using its libraries. Now still I am learning nonlinear models. There is no limit of learning.