**IZMIR UNIVERSITY OF ECONOMICS**

**SOFTWARE ENGINEERING**

**CE 475 – FUNDAMENTALS AND APPLICATIONS OF MACHINE LEARNING**

**TERM PROJECT REPORT**

**MODEL SELECTION AND ESTIMATION IN REGRESSION WITH MULTIPLE VARIABLES**

**ISMAIL SEMIH PEHLIVAN**

**20130601066**

**CONTENT**

1. **METHODOLOGY. . . . . . . . . . . . . . . . . . . . . . . . . . 3**
   1. **Multiple Linear Regression**
   2. **Polynomial Regression**
   3. **Decision Tree**
   4. **Random Forest**
2. **IMPLEMENTATION . . . . . . . . . . . . . . . . . . . . . . . . 5**
   1. **Multiple Linear Regression**
   2. **Polynomial Regression**
   3. **Decision Tree**
   4. **Random Forest**
3. **RESULTS . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 13**
4. **CONCLUSION . . . . . . . . . . . . . . . . . . . . . . . . . . . . 14**
5. **REFERENCES . . . . . . . . . . . . . . . . . . . . . . . . . . . . 14**

**1.METHODOLOGY**

I made four regression to predict the Y values. One of them Multiple linear regression, others nonlinear regressions. They also include Polynomial Regression, Decision Tree and Random Forest.

* 1. **Multiple Linear Regression**

Multiple linear regression (MLR), also known simply as multiple regression, is a statistical technique that uses several explanatory variables to predict the outcome of a response variable. The goal of multiple linear regression (MLR) is to model the linear relationship between the explanatory (independent) variables and response (dependent) variable.

**Advantages of Multiple Linear Regression**

The main advantage of multiple regression is that it allows multiple independent/predictor variable to be the part of the regression model. With this flexibility you can include as many variable as you want but keeping in mind that adding certain independent variable doesn’t increase the quality of the model but decrease it.

**Disadvantages of Multiple Linear Regression**

Any disadvantage of using a multiple regression model usually comes down to the data being used. Two examples of this are using incomplete data and falsely concluding that a correlation is a causation.

* 1. **Polynomial Regression**

In statistics, polynomial regression is a form of regression analysis in which the relationship between the independent variable x and the dependent variable y is modelled as an nth degree polynomial in x. Polynomial regression fits a nonlinear relationship between the value of x and the corresponding conditional mean of y, denoted E(y |x) Although polynomial regression fits a nonlinear model to the data, as a statistical estimation problem it is linear, in the sense that the regression function E(y | x) is linear in the unknown parameters that are estimated from the data. For this reason, polynomial regression is considered to be a special case of multiple linear regression.

**Advantages of Polynomial Regression**

Biggest advantage of nonlinear regression over many other techniques is the board range of functions that can be fit. Polynomial fit wide rage od curvature.

Polynomial provide a good approximation of the relationship

**Disadvantages of Polynomial Regression**

Disadvanges include a strong sensitive to outliers.

* 1. **Decision Tree**

Decision tree builds regression or classification models in the form of a tree structure. It breaks down a dataset into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The final result is a tree with decision nodes and leaf nodes.

**Advantages of Decision Tree**

Decision trees implicitly perform variable screening or feature selection.

Decision trees require relatively little effort from users for data preparation

**Disadvantages of Decision Tree**

Complexity, Unwieldy, Costs, Too Much Information, Analysis Limitations

* 1. **Random Forest**

Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random decision forests correct for decision trees' habit of overfitting to their training set.

**Advantages of Random Forest**

It is one of the most accurate learning algorithms available. For many data sets, it produces a highly accurate classifier.

It runs efficiently on large databases.

It can handle thousands of input variables without variable deletion.

It gives estimates of what variables are important in the classification.

**Disadvantages of Random Forest**

Random forests have been observed to overfit for some datasets with noisy classification/regression tasks.

**2.IMPLEMENTATION**

**2.1. Multiple Linear Regression**

**The libraries :**

import pandas as pd

import numpy as np

from sklearn.linear\_model import LinearRegression

**The implementation :**

from sklearn.linear\_model import LinearRegression

regressor = LinearRegression()

regressor.fit(x\_0\_100,y\_0\_100)

y\_prediction = regressor.predict(x\_100\_120)

**#BACKWARD ELIMINATION**

import statsmodels.formula.api as sm

a = np.append(arr=np.ones((100,1)).astype(int), values = x\_0\_100, axis=1)

a\_l = x\_0\_100[:,[0,1,2,3,4,5]]

r\_ols = sm.OLS(endog = y\_0\_100, exog = a\_l)

r = r\_ols.fit()

print(r.summary())

a\_l = x\_0\_100[:,[1,2,3,4,5]]

r\_ols = sm.OLS(endog = y\_0\_100, exog = a\_l)

r = r\_ols.fit()

print(r.summary())

a\_l = x\_0\_100[:,[1,2,3,5]]

r\_ols = sm.OLS(endog = y\_0\_100, exog = a\_l)

r = r\_ols.fit()

print(r.summary())

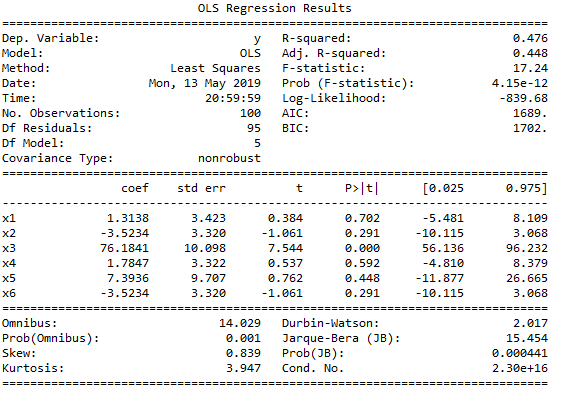
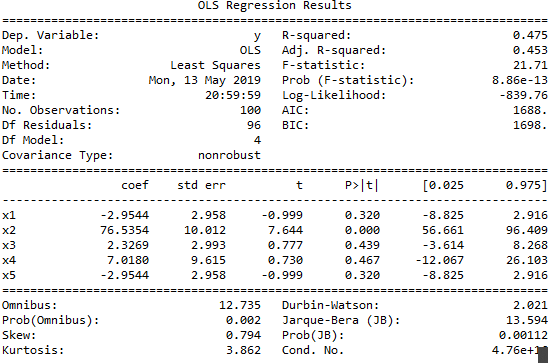
a\_l = x\_0\_100[:,[1,2,5]]

r\_ols = sm.OLS(endog = y\_0\_100, exog = a\_l)

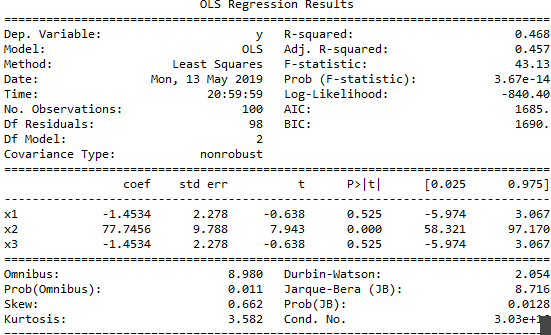
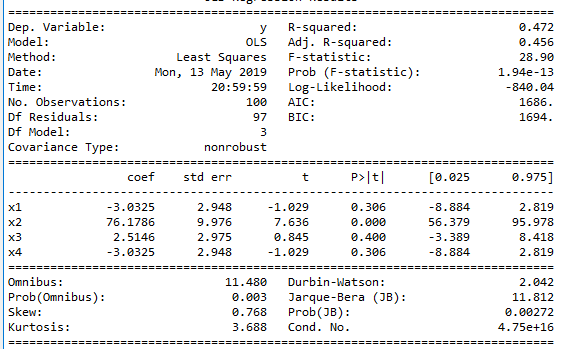
r = r\_ols.fit()

print(r.summary())

**The Outputs :**

I used for backward elimination to understand which x is most valuable. We need to find lowest ****

P.



**2.2. Polynomial Regression**

**The libraries :**

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.preprocessing import PolynomialFeatures

**The implementation :**

#2th degree polynom

from sklearn.preprocessing import PolynomialFeatures

poly\_reg = PolynomialFeatures(degree = 2)

x\_poly = poly\_reg.fit\_transform(first\_100\_x3)

lin\_reg2 = LinearRegression()

lin\_reg2.fit(x\_poly,first\_100\_y\_nonvalue)

# 4th degree polynom

poly\_reg3 = PolynomialFeatures(degree = 4)

x\_poly3 = poly\_reg3.fit\_transform(first\_100\_x3)

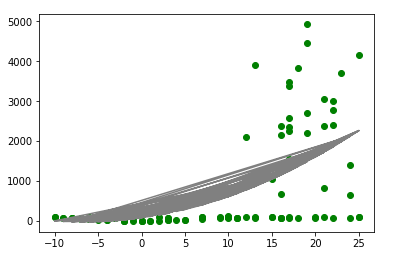
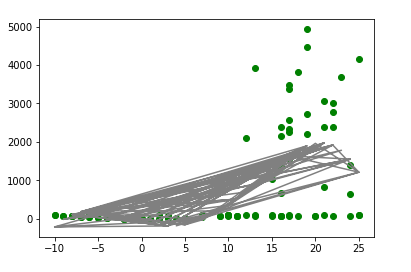
lin\_reg3 = LinearRegression()

lin\_reg3.fit(x\_poly3,first\_100\_y\_nonvalue)

x\_poly3\_1 = poly\_reg3.fit\_transform(last\_20\_x3)

y\_prediction\_100\_120= lin\_reg3.predict(x\_poly3\_1)

**The Outputs :**



**2.3. Decision Tree**

**The libraries :**

import pandas as pd

from sklearn.tree import DecisionTreeRegressor

from sklearn.model\_selection import cross\_val\_score

**The implementation :**

# Decision Tree Regression with dataset

from sklearn.tree import DecisionTreeRegressor

regressor = DecisionTreeRegressor(random\_state = 0)

regressor.fit(x\_elimination, first\_100\_y)

# Predicting a resul

prediction = regressor.predict(x\_elimination\_100\_120)

print(prediction)

#check the model

from sklearn.model\_selection import cross\_val\_score

accuracies = cross\_val\_score(estimator = regressor, X = x\_elimination, y = first\_100\_y , cv= 4)

print(accuracies.mean())

**The Outputs :**

[1.700e+01 9.000e+01 1.040e+02 7.900e+01 2.343e+03 9.200e+01 9.700e+01

6.600e+01 7.800e+01 3.822e+03 8.140e+02 8.100e+01 9.900e+01 1.000e+00

1.500e+01 7.900e+01 9.100e+01 4.467e+03 3.500e+01 7.800e+01]

Success Rate : **0.8479647215288466**

**2.4. Random Forest**

**The libraries :**

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.ensemble import RandomForestRegressor

from sklearn.metrics import r2\_score

**The implementation :**

from sklearn.model\_selection import train\_test\_split

x\_train, x\_test, y\_train, y\_test = train\_test\_split(first\_100\_x, first\_100\_y, test\_size=0.33,random\_state=0)

from sklearn.ensemble import RandomForestRegressor

regressor = RandomForestRegressor(n\_estimators=27,random\_state=0)

regressor.fit(x\_train, y\_train)

r=regressor.score(x\_test,y\_test)

y\_prediction\_100\_120 = regressor.predict(last\_20\_x)

print(y\_prediction\_100\_120)

print(r)

from sklearn.metrics import r2\_score

r\_score=r2\_score(y\_test,regressor.predict(x\_test))

print(r\_score)

**The Outputs :**

[1.56190476e+01 4.47904762e+02 8.78095238e+01 7.93809524e+01

1.90842857e+03 9.22857143e+01 9.61904762e+01 6.19714286e+02

1.02761905e+02 4.18152381e+03 1.18776190e+03 7.73333333e+01

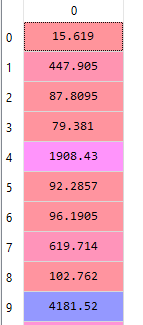
7.00952381e+01 2.04761905e+00 1.69047619e+01 7.30952381e+01

9.42380952e+01 3.89323810e+03 3.03809524e+01 7.90952381e+01]

0.9652334429019868

0.9652334429019868

**3.RESULTS**



**4.CONCLUSION**

I chose the random forest among the regressions I used to make the most reliable estimate because it gave the highest success rate. I gave all the y values in the result.

**5.REFERENCES**

<https://www.quora.com/What-are-some-advantages-of-using-a-random-forest-over-a-decision-tree-given-that-a-decision-tree-is-simpler>

<https://medium.com/@sagar.rawale3/understanding-decision-tree-algorithm-drawbacks-and-advantages-4486efa6b8c3>

<https://turi.com/learn/userguide/supervised-learning/random_forest_regression.html>