Multiple Regression(with Polynomial Regression)

Muhammad Moaz Amin
dept. Electronic Engineering
Hocschule Hamm Lippstadt
Hamm, Germany
muhammad-moaz.amin@stud.hshl.de

***Abstract*—Estimating the relationship among different variable which have reason and result relation holds great importance, to estimate that Regression analysis is a statistical technique that is adapted. Main focus of this uni-variate regression is to analyse the relationship between a non linear and a linear variable and to formulate a linear equation between the two. a regression model which contain one linear and multiple non-linear independent variables is most often called multi linear regression. [1] This paper is concentrated on the polynomial regression model, which is useful when there is reason to believe that relationship between two variables is curvilinear. The polynomial regression model has been applied using the characterisation of the connection between strains and drilling depth. Parameters of the model were estimated employing a least square method. After fitting, the model was evaluated using a number of the common indicators wont to evaluate accuracy of regression model.**

I. Introduction

Regression analysis involves identifying the link between a variable quantity and one or more independent variables. It's one amongst the foremost important statistical tools which is extensively utilized in the majority sciences. It's specially used in business and economics to check the link between two or more variables that are related causally. A model of the relationship is hypothesized, and estimates of the parameter values are accustomed develop an estimated equation.

Common questions which are generally asked in this research of Multiple regression analysis generally revolve around " are there any relations between dependent and independent variable?", and "if there are some relations that exist, what is total power of the relation?"," is there any possibility to predict about orientation regarding the dependent variable?', and " if certain conditions are controlled, what influences does a special variable or a group of variables have over another variable or variables?". [1]

Uni-variate analysis is the regression which uses a single independent variable while multivariate regression analysis uses more than one independent variable . [15], [18] The relation between dependent variable and an independent variable is analysed through uni-variate regression analysis, and the equation which represents the linear relationship between the both is formulated.

In Multivariate regression analysis, an attempt is made to account for the variation between the dependent variable and the independent variable synchronously. [11] Formulation of Multivariate model is represented in Figure 1.

$$y = \beta_0 + \beta_1 x_1 + ... + \beta_n x_n + \epsilon$$

y = *dependent variable*
X_i = *independent variable*
β_i = *parameter*
ϵ = *error*

Fig. 1. Multivariate Model [12]

The assumptions of multivariate regression analysis are normal distribution, linearity, freedom from extreme values and having no multiple ties between independent variable. [15]

II. Multiple Regression with Python

Linear Regression generally is a supervised method for machine learning rooted in statistics. From this method numeric values are forecasted using a combination of predictors which can be both numeric or binary variables. The condition for getting the variable depends on a certain relation at hand (a linear, measurable by a correlation) with the target variable. [13]

However, in reality there are a number of factors which alter the results of the working predictive model. There are usually more than variables that work together to achieve better and reliable results from a prediction. This causes more complexity in our model and hence representing it on a two- dimensional plot is not easy. All the predictors will constitute their own unique dimension and we would have to assume that our predictors apart from being related to the response are also related among themselves and this characteristic of data is called multicollinearity. [13]

A. Multiple Regression Formulation

The basic Multiple Regression is described in Figure 1 and in depth detail is shown in Figure 2 where dependent(response) variable Y on a set of k independent(predictor) variables X_1,X_2,.....X_k can be expressed as

where

- y_i= value of dependent variable, Y is for ith case.
- x_{ij} = value of jth independent variable,X_j for ith case.
- β_0 is the Y-intercept of the regression surface.

$$\begin{cases} y_1 &= \beta_0 + \beta_1 x_{11} + \ldots + \beta_k x_{1k} + e_1 \\ y_1 &= \beta_0 + \beta_1 x_{21} + \ldots + \beta_k x_{2k} + e_2 \\ & . \\ & . \\ & . \\ y_n &= \beta_0 + \beta_1 x_{n1} + \ldots + \beta_k x_{nk} + e_n \end{cases}$$

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_k x_i k + e_i \text{ , for } i = 1, 2, \ldots, n$$

Fig. 2. In depth Multiple Regression Model showing relation of independent(k) and dependent variable(Y)

- each β_j, j=1,2,....k, is the slope of the regression surface w.r.t. variable X_j and e_i is the random error component for the ith case.

In the first equation we have n observations and k predictors ($n > k$+1) The assumptions of the multiple regression model are similar to those for the simple linear regression model. Model assumptions [13]:

B. Observations

- errors e_i are normally distributed with their mean as zero and their standard deviation σ are independent of the error terms associated with all other observations. Errors are not related to each other.
- variables X_j in the context of regression analysis are considered as fixed quantities, whereas they are random variables in the context of correlation analysis. But in both the cases X_j are totally independent of the error term. If X_j are assumed as fixed quantities, then we are assuming that we have realizations of k variables X_j and the only randomness in Y is coming from the error term.

In matrix form, we can rewrite the regression model as described in Figure. 3.

$$\mathbf{Y} = \mathbf{X}\boldsymbol{\beta} + \mathbf{e}$$

Fig. 3. Matrix notation of Model

- response vector **Y** and error vector **e** are the column vectors of length n.
- vector of parameters $beta$ is the column vector of length k+1 and its design matrix (where all elements in the first column are equal to 1, and second column's values are filled by the observed values of X_1,etc).

Here, values of β and e are unknown and assumed.

III. Polynomial Regression

A. History

The first design of an experiment for polynomial Regression appeared in a paper in 1815 by Gergonne. [17] In the twentieth century, polynomial regression played an important role in the development of regression analysis with greater emphasis on issues of design and interface. [16]. Recently, polynomial models have been complemented by other methods, with no-polynomial models being more advantageous for certain classes of problems.

B. Definition

Polynomial Regression is a technique of Multiple Regression which provides an automatic means of creating both interactions and non-linear power transformations of the original variables systematically. The number of bends that fit the curve depend on the degree of power. Higher the power higher will be the bends.

For Example in simple linear regression form:

$$\boldsymbol{y} = \beta_0 + \beta_1 \boldsymbol{x}$$

With a second degree transformation of the simple form, **quadratic** form is achieved:

$$\boldsymbol{y} = \beta_0 + \beta_1 \boldsymbol{x} + \beta_2 x^2$$

Likewise, with a third degree transformation the equation turns into **cubic**:

$$\boldsymbol{y} = \beta_0 + \beta_1 \boldsymbol{x} + \beta_2 x^2 + \beta_3 x^3$$

In case of regression being a multiple one, the expansion will create more terms as the power is increasing resulting in more features being derived. A regression which is a result of two predictors (x_1 and x_2 which is expanded using quadratic transformation will result in the following equation:

$$\boldsymbol{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_1 x_2 + \beta_4 x_1^2 + \beta_5 x_2^2$$

With such an expansion two important aspects are observed:

- Polynomial expansion increases number of predictors rapidly.
- Power of predictors is directly proportional to degree of polynomials, which results in deteriorating numeric stability, thus it requires standardized numeric values which are too large or suitable numeric formats.

Matrix Form of Polynomial Regression is described in Figure 4. [12]

C. Why do we need Polynomial Regression?

In the case of Simple Linear Regression the best fit formed line is a straight line where as actual values form a curve which results in formed line not cutting the mean of the points.

In such cases, Polynomial Regression is beneficial, as it predicts the best fit line that follows the pattern of the plotted points resulting in forming a curve. This is described in the Figure 5.

One of the few differences between Polynomial Regression and Linear Regression is that Polynomial Regression does not require the relationship between the independent and dependent variables to be linear in the data set. Moreover,

$$\begin{bmatrix} y_1\\ y_2\\ y_3\\ .\\ .\\ .\\ y_n\end{bmatrix} = \begin{bmatrix}1 & x_1 & x_1^2 & \dots & x_1^m\\ 1 & x_2 & x_2^2 & \dots & x_2^m\\ 1 & x_3 & x_3^2 & \dots & x_3^m\\ . & . & . & . & .\\ . & . & . & . & .\\ . & . & . & . & .\\ 1 & x_n & x_n^2 & \dots & x_n^m\end{bmatrix} \begin{bmatrix}\beta_0\\ \beta_1\\ \beta_2\\ .\\ .\\ .\\ \beta_m\end{bmatrix} + \begin{bmatrix}\epsilon_1\\ \epsilon_2\\ \epsilon_3\\ .\\ .\\ .\\ \epsilon_n\end{bmatrix}$$

Fig. 4. Matrix Model of Polynomial Regression [2]

Polynomial Regression Model is used when the Linear Regression fails to describe the best result and and the points are not captured by it.

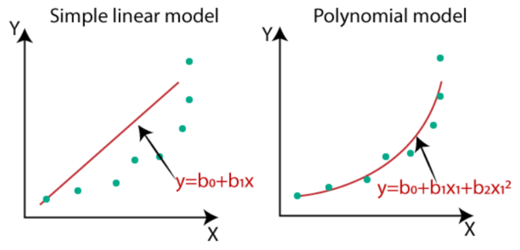

Fig. 5. Linear and Polynomial Regression plotted [2]

IV. Why should we Choose Polynomial Regression?

Linear Regression requires the relation between the dependent and the independent variable to be linear. But in the case of distribution data being more complex? Is it possible to fit non-linear data into linear models? How can a curve be produced that captures the data as in Figure 6? To understand

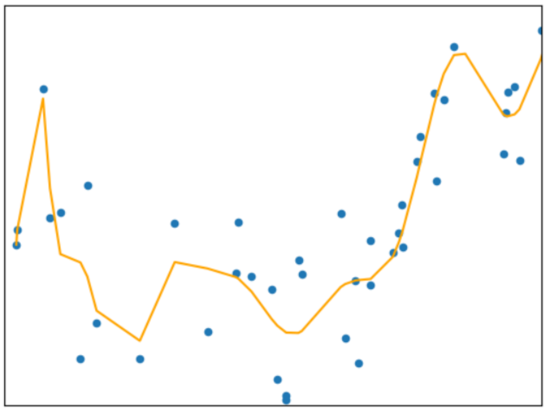

Fig. 6. Complex data Set [3]

something's importance, it's comparison with its competitor is very important. Just like that to understand the need of Polynomial Regression we need to first evaluate set of data with linear Regression and then produce it using Polynomial Regression. at First we need to generate some random data with Python language:

```python
import numpy as np
import matplotlib.pyplot as plt

np.random.seed(0)
x = 2 - 3 * np.random.normal(0, 1, 20)
y = x - 2 * (x ** 2) + 0.5 * (x ** 3) +
np.random.normal(-3, 3, 20)

plt.scatter(x,y, s=10)
plt.show()

```

Listing 1. Random data generation [5]

The data generated would look like Figure 7.

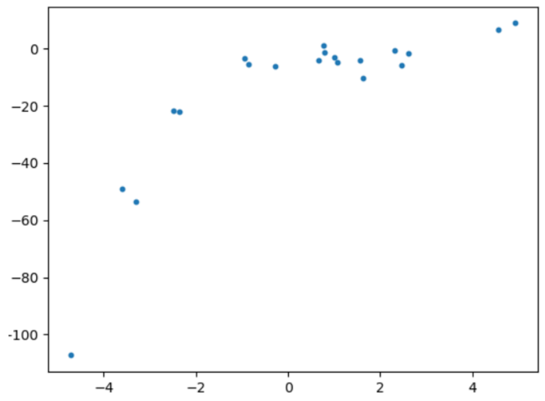

Fig. 7. [3]

Now, applying Linear regression Model to the same dataset.

```python
import numpy as np
import matplotlib.pyplot as plt

from sklearn.linear_model import LinearRegression

np.random.seed(0)
x = 2 - 3 * np.random.normal(0, 1, 20)
y = x - 2 * (x ** 2) + 0.5 * (x ** 3) + np.random.
    normal(-3, 3, 20)

# transforming the data to include another axis
x = x[:, np.newaxis]
y = y[:, np.newaxis]

model = LinearRegression()
model.fit(x, y)
y_pred = model.predict(x)

plt.scatter(x, y, s=10)
plt.plot(x, y_pred, color='r')
plt.show()

```

Listing 2. Linear regression model [6]

When Linear Regression is applied to the data the result shown in Figure 8 is seen. Here we can see that straight line is still unable to capture the patterns.This brings us to a concept called **under-fitting**.

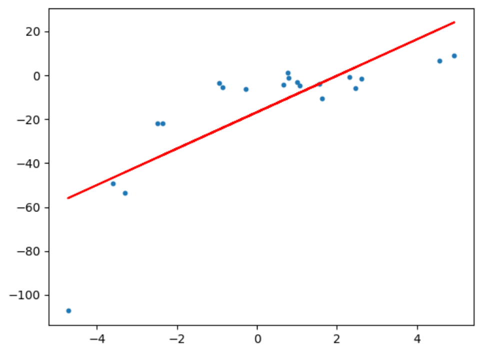

Fig. 8. Plot of best fit line [3]

Under-Fitting

It refers to a model that can neither model the training data nor generalize to new data. In turn, it will have poor performance on the training data. It is generally more easy to detect given a good performance metric. One way to fix it is try alternate machine learning algorithms. Another cause of under-fitting is using too few parameters for the predictions. In Figure 8 it is clearly visible the line doesn't match the curve bends and some of its predictions are misleading. [13] [9]

```
RMSE of Linear Regression is 15.90824.
R2 score of Linear Regression is 0.63868
```

To overcome under-fitting one solution is to increase the complexity of the whole model which in turn means generating a higher order equation which will transform Linear Regression form to quadratic form:

$$\boldsymbol{y} = \beta_0 + \beta_1 \boldsymbol{x} + \beta_2 x^2$$

To convert the original feature into higher order terms we will use PolynomialFeatures class provided by scikit-learn.

```
import operator

import numpy as np
import matplotlib.pyplot as plt

from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error,
    r2_score
from sklearn.preprocessing import PolynomialFeatures

np.random.seed(0)
x = 2 - 3 * np.random.normal(0, 1, 20)
y = x - 2 * (x ** 2) + 0.5 * (x ** 3) + np.random.
    normal(-3, 3, 20)

# transforming the data to include another axis
x = x[:, np.newaxis]
y = y[:, np.newaxis]

polynomial_features= PolynomialFeatures(degree=2)
x_poly = polynomial_features.fit_transform(x)

model = LinearRegression()
model.fit(x_poly, y)
y_poly_pred = model.predict(x_poly)

rmse = np.sqrt(mean_squared_error(y,y_poly_pred))
r2 = r2_score(y,y_poly_pred)
print(rmse)
print(r2)

plt.scatter(x, y, s=10)
# sort the values of x before line plot
sort_axis = operator.itemgetter(0)
sorted_zip = sorted(zip(x,y_poly_pred), key=
    sort_axis)
x, y_poly_pred = zip(*sorted_zip)
plt.plot(x, y_poly_pred, color='m')
plt.show()
```

Listing 3. Linear regression model with higher degree [8]

If we fit a linear regression model on the transformed features it gives the below plot.

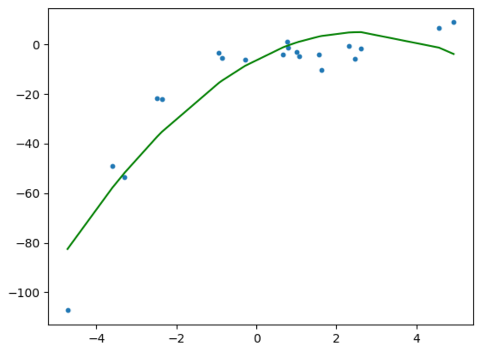

Fig. 9. Plot of best fit line on quadratic data

```
RMSE of polynomial regression is 10.1204
R2 of polynomial regression is 0.8537
```

It is visible RMSE has decreased and R^2-score has increased as compared to linear line.

A cubic curve of degree 3 will show us that more data points are passed than quadratic and linear plots.

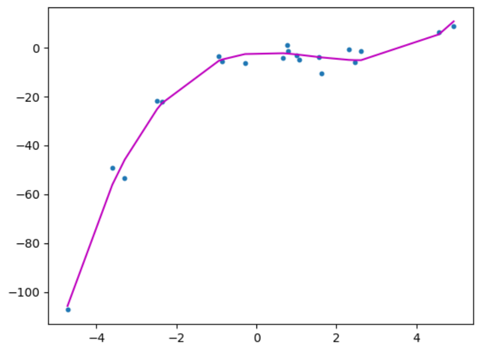

Fig. 10. Plot of best fit line on cubic data [3]

A comparison of linear, quadratic and cubic curve is shown in Figure 11.

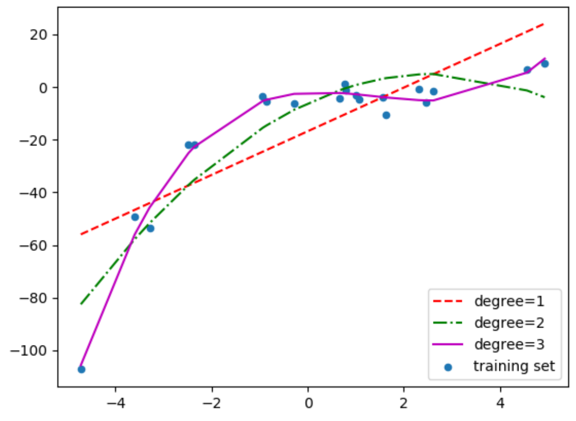

Fig. 11. [3]

By increasing the degree even further to 20, the curve passes through more data points. The curve of degree 20 is also capturing noise in the data. This brings us to the concept called **over-fitting**.

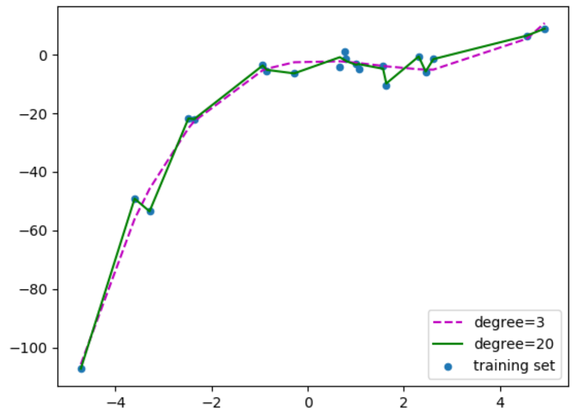

Fig. 12. Comparison of curve of degree 3 and 20 [3]

Over-Fitting

It refers to a model that models the data too well which in turn means that the result is just a mere memorization. This happens when a model learn the details in the training data to the extent that it even interprets the noise which impacts negatively to the performance of the model on new data. [13] Over-fitting is more common with non parametric and non linear models which are more flexible when learning a target function. AS an example of over-fitting decision tress which are non parametric machine learning algorithm which is very flexible can be used. Over-fitting can be addressed by pruning a tree after it has learned in order to remove some detail it has picked up. [9]

Choosing optimal model according to the situation is always vital and for understanding the logic behind making the right decision understanding the **bias vs variance trade off is important**.

Bias vs Variance trade-off

Bias refers to the simplifying assumptions made by the model to make the target function easier to approximate.A higher bias means that model s unable to capture the patterns and this results in under-fitting. [4], [10]

Variance refers to the amount that the estimate of the target function will change given different training data. Higher variance will mean that the model passes through most of the data points and it will result in over-fitting of the data. [4], [10]

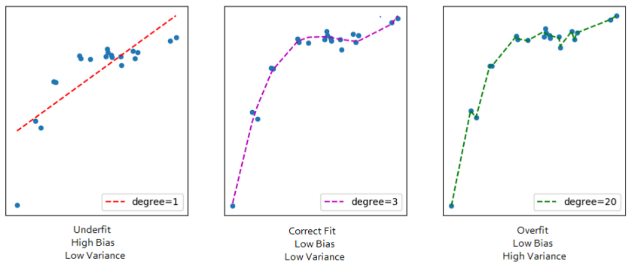

Fig. 13. Bias, Variance trade off from degree 1 till 20 [3]

Figure 14 below is showing as the model complexity will increase, bias will decrease and variance will subsequently increase. In ideal situation a machine learning model should have low variance and low bias. It is not possible to achieve both. To achieve a good model which performs well in both aspects a **trade-off** is made.

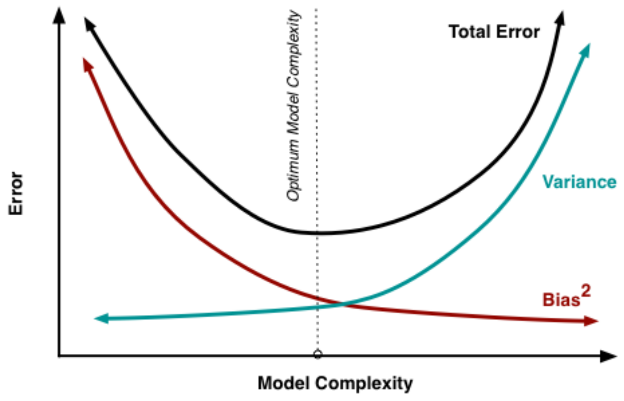

Fig. 14. Visualisation of optimum model complexity [10]

Using a more practical approach we will transform some training data into higher degree polynomials. In Figure 15 its is quite visible that LSTAT has a slightly non linear variation with target variable MEDV.

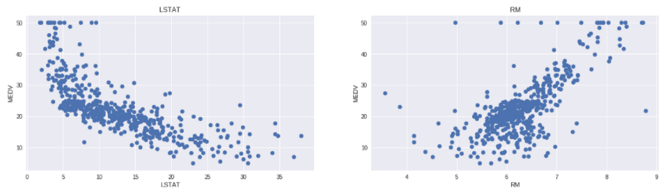

Fig. 15. [3]

A function for transforming the original features into polynomial features and then applying linear Regression is as follows.

```python
from sklearn.preprocessing import PolynomialFeatures

def create_polynomial_regression_model(degree):
  "Creates a polynomial regression model for the given degree"

  poly_features = PolynomialFeatures(degree=degree)

  # transforms the existing features to higher degree features.
  X_train_poly = poly_features.fit_transform(X_train)

  # fit the transformed features to Linear Regression
  poly_model = LinearRegression()
  poly_model.fit(X_train_poly, Y_train)

  # predicting on training data-set
  y_train_predicted = poly_model.predict(X_train_poly)

  # predicting on test data-set
  y_test_predict = poly_model.predict(poly_features.fit_transform(X_test))

  # evaluating the model on training dataset
  rmse_train = np.sqrt(mean_squared_error(Y_train, y_train_predicted))
  r2_train = r2_score(Y_train, y_train_predicted)

  # evaluating the model on test dataset
  rmse_test = np.sqrt(mean_squared_error(Y_test, y_test_predict))
  r2_test = r2_score(Y_test, y_test_predict)

  print("The model performance for the training set")
  print("-------------------------------------------")
  print("RMSE of training set is {}".format(rmse_train))
  print("R2 score of training set is {}".format(r2_train))

  print("\n")

  print("The model performance for the test set")
  print("-------------------------------------------")
  print("RMSE of test set is {}".format(rmse_test))
  print("R2 score of test set is {}".format(r2_test))
```

Listing 4. [7]

Using scikit-learns's LinearRegresssion to to train the model following is achieved.

```
The model performance for training set
--------------------------------------
RMSE is 5.6371293350711955
R2 score is 0.6300745149331701

The model performance for testing set
--------------------------------------
RMSE is 5.137400784702911
R2 score is 0.6628996975186952
```

For calling the function with the degree as 2 we use

```python
create_polynomial_regression_model(2)
```

Model's performance using Polynomial Regression:

```
The model performance for the training set
-------------------------------------------
RMSE of training set is 4.703071027847756
R2 score of training set is 0.7425094297364765

The model performance for the test set
-------------------------------------------
RMSE of test set is 3.784819884545044
R2 score of test set is 0.8170372495892174
```

The result with Polynomial Regression is better than using Linear Regression Model.

V. Conclusion

In this paper, Ploynomial Regression as a whole and its comparison with Linear Regression is described. In general, polynomial provides best approximation, broader range of function that can fit under it and a broader curvature but on the other hand it has its down sides as well. Presence of outliers can serriously effect the overall result of the analysis as polynomial regression is very sensitive to them. Apart of this there are few validation tools to detect the outliers in no linear regression compared to the linear one. [14]

References

[1] Pandora - uygulamalı Çok değişkenli İstatistiksel yöntemlere giriş 1 - reha alpar - kitap - isbn 9789755914312.
[2] Abhigyan. Understanding polynomial regression!!!, Aug 2020.
[3] Animesh Agarwal. Polynomial regression, Oct 2018.
[4] Imran Ahmad. *40 Algorithms Every Programmer Should Know*. Packt Publishing, 2020.
[5] Animesh-Agarwal. data-set.py.
[6] Animesh-Agarwal. Linear regression on non linear data.py.
[7] Animesh-Agarwal. polynomial regression on boston housing data set.py.
[8] Animesh-Agarwal. Polynomial regression.py.
[9] Jason Brownlee. Overfitting and underfitting with machine learning algorithms, 2021.
[10] Scott Fortmann-Roe. Understanding the bias-variance tradeoff.
[11] Ünver. Ö. Gamgam. Uygulamali İstatİstİk i - İstanbul.
[12] Gülden Kaya Uyanık and Neşe Güler. A study on multiple linear regression analysis. *Procedia - Social and Behavioral Sciences*, 106:234–240, 12 2013.
[13] Luca Massaron. Regression analysis with python: Nook book, Feb 2016.
[14] Ayush Pant. Introduction to linear regression and polynomial regression, Jan 2019.
[15] Pegem.Net. Sosyal bilimler için veri analizi el kitabı İstatistik, araştırma deseni spss uygulamaları ve yorum - Şener büyüköztürk&nbsp;: Kpss, Öabt, ales, dgs, yks, lgs, yds, gys kitapları: Pegem.net İnternetteki kitapçınız.
[16] Kirstine Smith. On the Standard Deviations of Adjusted and Interpolated Values of an Observed Polynomial Function and its Constants and the Guidance they give Towards a Proper Choice of the Distribution of Observations, November 1918.

[17] Stephen M Stigler. Gergonne's 1815 paper on the design and analysis of polynomial regression experiments. *Historia Mathematica*, 1(4):431–439, 1974.

[18] Barbara G. Tabachnick and Linda S. Fidell. *Using Multivariate Statistics (5th Edition)*. Allyn & Bacon, Inc., USA, 2006.