Data Mining Lab 1 Report

Lavinia Pulcinella
Student ID: i6233926

September 2020

Assignment 1

a

III. For a fixed value of IQ and GPA, males earn more on average than females provided that the GPA is high enough.
In general, the least square line is given by

$$\hat{y} = 50 + 20GPA + 0.07IQ + 35GENDER + 0.01GPA \times IQ - 10GPA \times GENDER \tag{1}$$

Intuitively, since the β_5 coefficient is negative when it is "on" (i.e. for females) it should have a negative effect on the outcome. However since the coefficient affects the salary though the interaction between the GENDER and GPA variables, the role of the latter should be further investigated.
For females i.e. when $X_3 = GENDER = 1$, we have

$$\hat{y} = 85 + 10GPA + 0.07IQ + 0.01GPA \times IQ$$

while in the case of males, we get

$$\hat{y} = 50 + 20GPA + 0.07IQ + 0.01GPA \times IQ$$

The comparison of the least square lines for females and males leads to

$$50 + 20GPA \geq 85 + 10GPA$$

Hence, the starting salary for males is higher (wrt females) - on average - if and only if $GPA \geq 3.5$

b.

By plugging in the given values in the least square line for females we easily get

$$\hat{y} = 50 + 20(4.0) + 0.07(110) + 35(1) + 0.01(4.0) \times (110) - 10(4.0) \times (1) = 137.1 \tag{2}$$

ASSIGNMENT 2

[123]:
```
import numpy as np
import pandas as pd
np.random.seed(1234) #set seed to ensure consistent results
```

a

Create vector of 100 observations distributed according to a standard normal distribution which represents a feature X.

[124]:
```
mu, sigma = 0, 1 # mean and standard deviation
x = np.random.normal(mu, sigma, 100)
print(mu, sigma)
```

b

Create vector eps (i.e. ϵ) normally distributed with 0 mean and variance of 0.25.

[125]:
```
mu1 = 0
sigma1 = np.sqrt(0.25) # 0.5
eps = np.random.normal(mu1, sigma1, 100)
print(mu1, sigma1)
```

c

Generate vector y according to:

$$Y = -1 + 0.5X + \epsilon \tag{3}$$

which can be generalized in

$$Y = \beta_0 + \beta_1 X + \epsilon, \quad s.t. \quad \beta_0 = -1 \quad and \quad \beta_1 = 0.5 \tag{4}$$

[126]:
```
b0 = -1
b1 = 0.5
y = b0 + b1*x + eps
print('Lenght of vector y:', len(y))
# as expected since the regression is made over 100 observations.
```

```
Lenght of vector y: 100
```

d

The scatterplot displays the relationship between x and y over 100 observations. Each point corresponds to an observation with coordinates (x_i, y_i) for the i-th point.
In general, the observations appear to follow an *uphill* pattern which suggests a positive relationship between the feature vector X and the dependent variable y (see Figure 1).

e

[128]:
```
from sklearn.linear_model import LinearRegression

lm=LinearRegression()
lm.fit(x.reshape(-1,1),y)
y_pred=lm.predict(x.reshape(-1,1))

print('Predicted b0:', lm.intercept_, 'compared to', b0)
print('Predicted b1:', lm.coef_, 'compared to', b1)
```

After fitting a least square model using the LinearRegression() function from the modeule linear_model we obtain the following estimators $\hat{\beta_0}$ and $\hat{\beta_1}$ for the coefficients β_0 and β_1 :

```
Predicted b0: [-1.02650639] compared to -1
Predicted b1: [[0.54070724]] compared to 0.5
```

In Figure 1 the least square line is plotted on the scatterplot, which formally minimizes the distances between all observations on the plane. The linear and positive relationship between x and y is indeed shown.

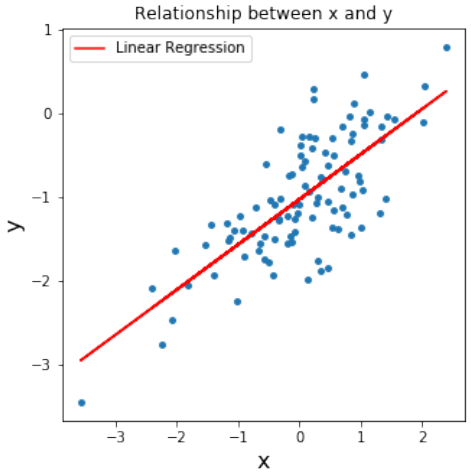

Figure 1: Linear Regression

The R^2 coefficient which estimates the proportion of variance explained by the model has been computed and results to be

[128]:
```
from sklearn.metrics import r2_score
r_sq = r2_score(y, y_pred) #Compute the R2 statistic
print('Coefficient of determination for Linear Regression:', r_sq)
```

```
Coefficient of determination for Linear Regression:
   >>> 0.5493701391253949
```

That is, the model is able to explain approximately 55% of the variance.

f

By fitting a 2-degree Polynomial Regression we are essentially fitting a nonlinear relationship between the x and y values.
In python, the implementation of polynomial regression is made in two-steps:

1. Transform data into a polynomial using the PolynomialFeatures function from sklearn
2. Use linear regression to fit the parameters

This can be automated using pipelines:

[130]:
```
# importing libraries for polynomial transform
from sklearn.preprocessing import PolynomialFeatures
from sklearn.pipeline import Pipeline

# creating pipeline and fitting it on data
Input=[('polynomial',PolynomialFeatures(degree=2)),('modal',LinearRegression())]
pipe=Pipeline(Input)
pipe.fit(x.reshape(-1,1),y.reshape(-1,1))
poly_pred=pipe.predict(x.reshape(-1,1))
```

The R^2 coefficient of determination is again computed and results in

```
Coefficient of determination (Polynomial Regression):
   >>> -0.7182974965188058
```

Thus, when the R^2 is negative, it suggests that the chosen model doesn't fit the trend of the data.

In figure 2 the Scatteplot created in point d) is improved by adding the polynomial regression curve.

By considering the R^2 score and the curve on the scatterplot we can see that the addition of the quadratic term doesn't seem to improve the model fit.

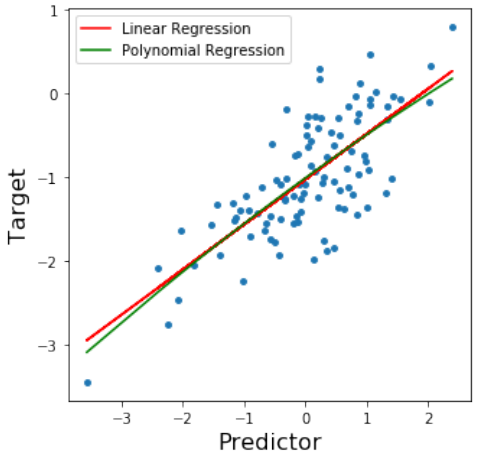

Figure 2: Polynomial and Linear Regression

g

By repeating the above analysis based on the following model

$$Y = -1 + 0.5X + X^2 + \epsilon \tag{5}$$

The role of Linear and Polynomial regression is reversed with respect to the previous analysis. The scatterplot show a clear "parabolic" behaviour of the observations hinting at a quadratic curve to best descibe the data.

The plot in 3 clearly suggests the above informal analysis. The 2nd degree polynomial degree regression describes best the data.

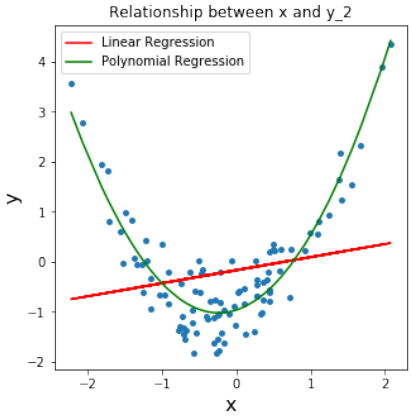

Figure 3: Polynomial and Linear Regression

in addition, the R^2 scores support the claim showing how the polynomial regression accounts for more than the 90% of the variance while linear regression only the 20%

```
Coefficient of determination (Linear Regression):
   >>> 0.20247784832070836

Coefficient of determination (Polynomial Regression):
   >>>  0.9116997189096664
```