Assignment-3

Pooja Umathe

10/1/2018

Polynomial_regression.R

What is Polynomial Regression?

• Polynomial regression extends the linear model by adding extra predictors, obtained by raising each of the original predictors to a power. The approach that we describe for extending the linear model to accommodate non-linear relationships is known as polynomial regression.

$$y_i = \beta_0 + \beta_1 x_i + \epsilon_i$$

with a polynomial function

$$y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \beta_3 x_i^3 + \ldots + \beta_d x_i^d + \epsilon_i,$$

This approach is known as polynomial regression.

let's have a look at the dataset, we'll be using

	A	B	C
1	Position	Level	Salary
2	Business Analyst	1	45000
3	Junior Consultant	2	50000
4	Senior Consultant	3	60000
5	Manager	4	80000
6	Country Manager	5	110000
7	Region Manager	6	150000
8	Partner	7	200000
9	Senior Partner	8	300000
10	C-level	9	500000
11	CEO	10	1000000

So, we have a data set, of a company and its employee positions and their annual salaries. Human Resources team is about to hire a new employee in the company. So, the HR team looking to hire a person. So, a level of let's say 6.5. Now, the company wants to make an offer to the person, and he asks for, more than 160k a year, because he says that, his salary was 160k in the previous company. So, the company asks us to build bluff predictor, which can predict if the person is lying. So, we tend to get the dataset (mentioned above), which gives the

salary and of the level of the position of the employee, where our potential employee, is working currently. So, now we need to build a lie detector using polynomial regression in R.

First, we have to import the Dataset,

```
# Import the position_salary data
dataset <- read.csv("C:/Users/Pooja/Downloads/Position_Salaries.csv")

dataset=dataset[2:3]
```

Now, we are fitting Linear Regression to the Dataset, that is to fit the data set, to the X and y variable, because we don't have the training and testing data. So, we import the linear Regression class, instantiate it, and fit it to the X and y variable. Fitting just means that, it is the linear Regression class will learn the co-relations between salaries and positions.

```
# Fitting Linear Regression to the dataset

lin_reg=lm(formula=Salary ~. , data=dataset)
summary(lin_reg)

##
## Call:
## lm(formula = Salary ~ ., data = dataset)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -170818 -129720  -40379   65856  386545
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -195333     124790  -1.565  0.15615
## Level          80879      20112   4.021  0.00383 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 182700 on 8 degrees of freedom
## Multiple R-squared:  0.669, Adjusted R-squared:  0.6277
## F-statistic: 16.17 on 1 and 8 DF, p-value: 0.003833
```

Now, we are importing Polynomial Features class and then we instantiate it, with a parameter. A degree of the equation is the highest power of the x variable.

Then we fit the class with the X_poly and y dataset, after transforming the data, and then again fit the data with Linear Regression class. So, we can see that in the code where we import Polynomial Features class, which just allows us to include some polynomial terms. Then, we make a new variable poly_reg, which is just the instantiation of the class. This allows us to add, the variables of x till the degree 4. What that means, just like in multiple linear regression, we had, multiple independent variables (x1, x2, x3), similarly, this instantiation of the class will

create the x^2, x^3 and x^4 for the equation. And, why only till degree 4, is because we gave it the degree as the parameter. Default is 2.

```
# Fitting Polynomial Regression to the Salary dataset
#(In order to generate the polynomial term, we need to build a a new variable level3, level4)

# polynomial level3

dataset$level2=dataset$Level^2
dataset$level3=dataset$Level^3
poly_reg=lm(formula=Salary ~. , data=dataset)
summary(poly_reg)

##
## Call:
## lm(formula = Salary ~ ., data = dataset)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -75695 -28148   7091  29256  49538
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -121333.3    97544.8  -1.244  0.25994
## Level        180664.3    73114.5   2.471  0.04839 *
## level2       -48549.0    15081.0  -3.219  0.01816 *
## level3         4120.0      904.3   4.556  0.00387 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 50260 on 6 degrees of freedom
## Multiple R-squared:  0.9812, Adjusted R-squared:  0.9718
## F-statistic: 104.4 on 3 and 6 DF,  p-value: 1.441e-05
```

So, we can see that the multiple R-squared is 0.9812, and adjusted R-squared is 0.9718. As we know that Multiple R squared is simply a measure of R-squared for models that have multiple predictor variables. Therefore, it measures the amount of variation in the response variable that can be explained by the predictor variables.

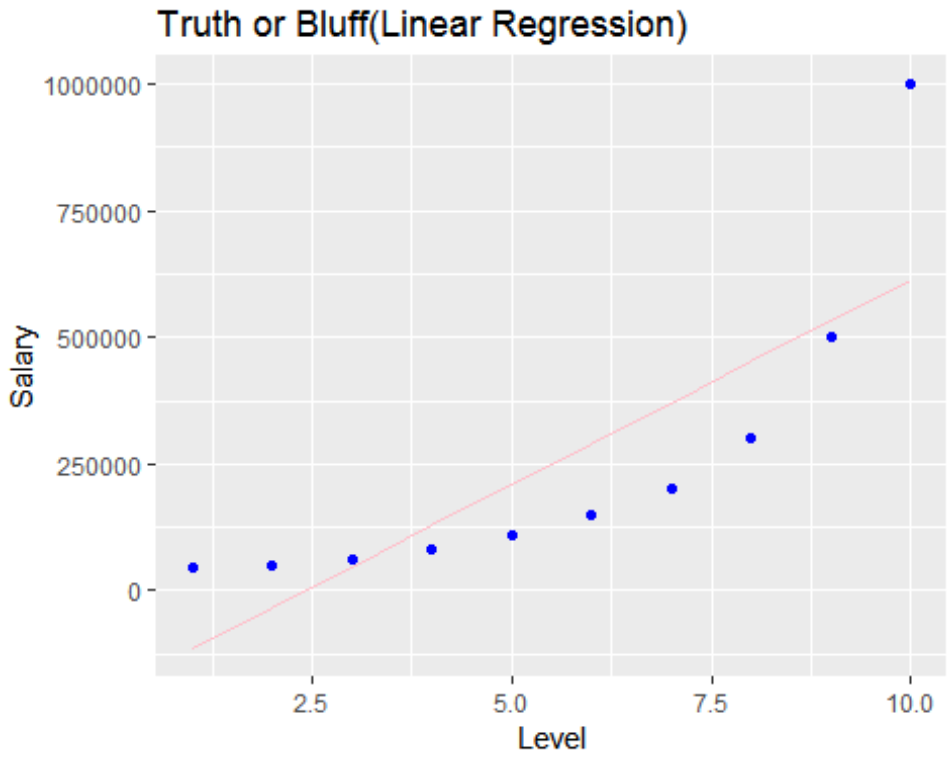
Adjusted R-squared controls against this increase, and adds penalties for the number of predictors in the model. Therefore, it shows a balance between the most parsimonious model, and the best fitting model.

Now, let's visualize the data. Here we are using ggplot2 libraries and code to visualize the data.

```
# Visualizing the Linear Regression results

library(ggplot2)

g1=ggplot()+
  geom_point(aes(x=dataset$Level , y=dataset$Salary),
             colour="blue")+
  geom_line(aes(x=dataset$Level, y=predict(lin_reg, newdata= dataset)),
            colour="pink")+
  ggtitle("Truth or Bluff(Linear Regression)")+
  xlab('Level')+
  ylab("Salary")
g1
```



Here, we can see the graph of linear regression named as Truth or bluff using salary and level of the dataset. we can easily see, that this will not work at all, in predicting a lie. So, we need a better curve that fits this data. So, we will plot a polynomial linear regression data. Now we are predicting the salary with linear regression prediction using level 6.5 and visualize the poly regression with level 3.

```
# Predict the salary with Linear Regression prediction
```

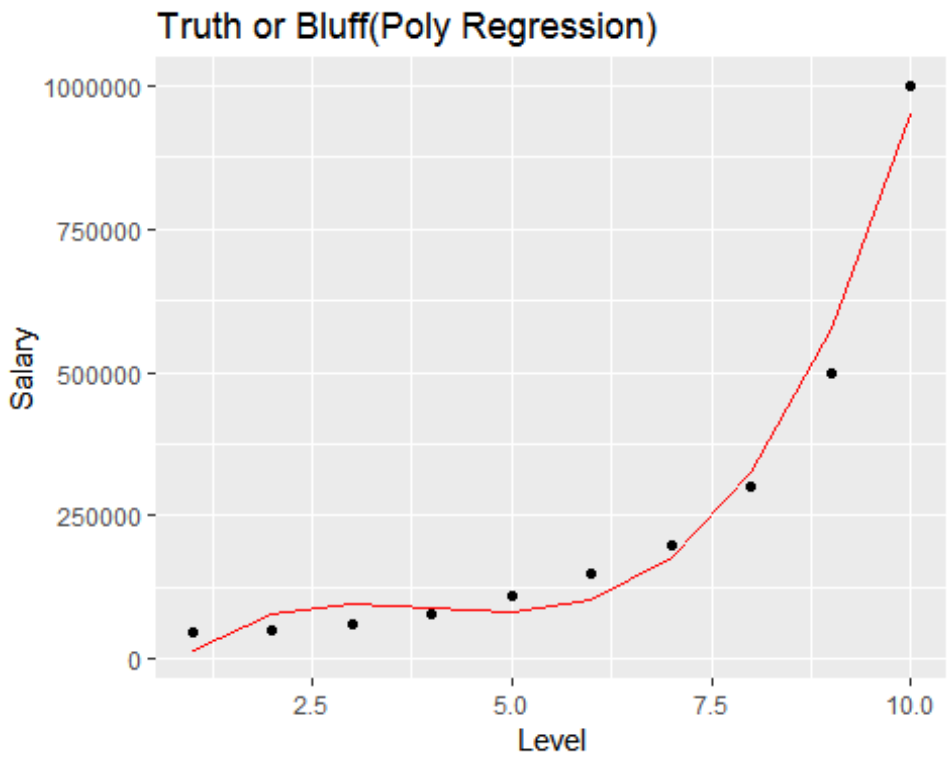
```
y_linear_pred=predict(lin_reg, data.frame(Level=6.5))
y_linear_pred
```

```
##        1
## 330378.8
```

```
# Visualizing the Poly Regression

# results with level 3

g2=ggplot()+
  geom_point(aes(x=dataset$Level , y=dataset$Salary),
             colour="black")+
  geom_line(aes(x=dataset$Level, y=predict(poly_reg, newdata= dataset)),
            colour="red")+
  ggtitle("Truth or Bluff(Poly Regression)")+
  xlab('Level')+
  ylab("Salary")
g2
```



So, From the poly regression graph we can see that, this curve fits the data much better, and also predicts the salary of our potential employee, (level = 6.5) pretty accurately. But there is a problem, in this graph, and that is, if we look closely, we can see straight lines, between the data points. So, let's make the data more continuous.

```
y_poly3_pred=predict(poly_reg, data.frame(Level=6.5, level2=6.5^2, level3=6.5^3))
y_poly3_pred
```

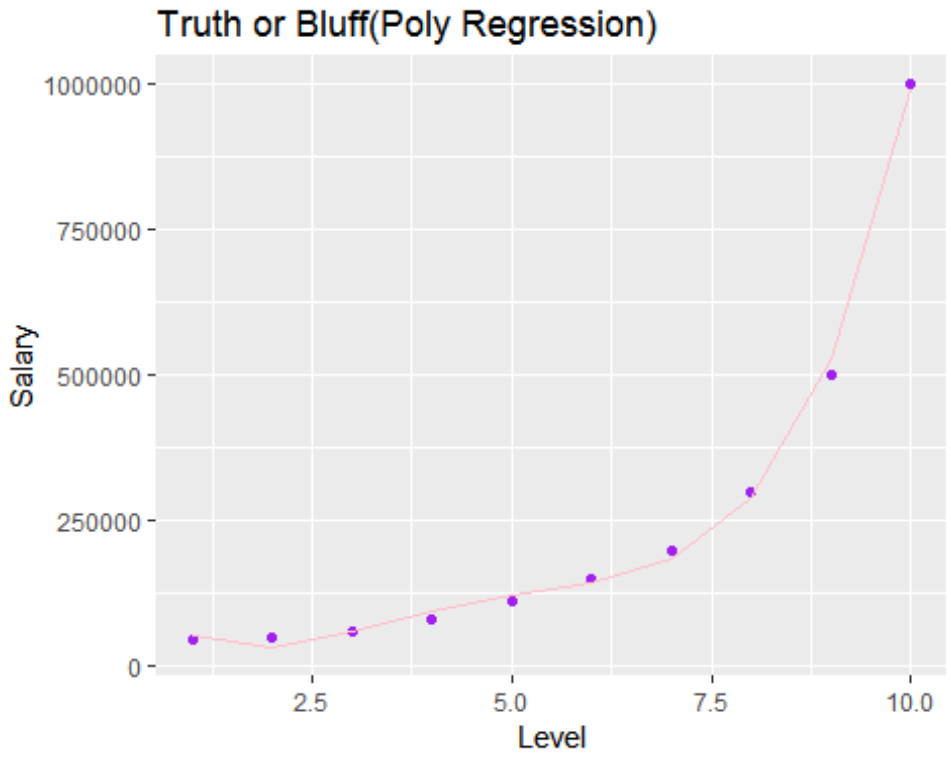
```
##        1
## 133259.5
```

```
#results with level 4

dataset1=dataset
dataset1$level4=dataset1$Level^4
poly_reg1=lm(formula=Salary ~. , data=dataset1)
summary(poly_reg1)
```

```
##
## Call:
## lm(formula = Salary ~ ., data = dataset1)
##
## Residuals:
##     1     2     3     4     5     6     7     8     9    10
##  -8357  18240   1358 -14633 -11725   6725  15997  10006 -28695  11084
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  184166.7    67768.0   2.718  0.04189 *
## Level       -211002.3    76382.2  -2.762  0.03972 *
## level2        94765.4    26454.2   3.582  0.01584 *
## level3       -15463.3     3535.0  -4.374  0.00719 **
## level4         890.2      159.8   5.570  0.00257 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 20510 on 5 degrees of freedom
## Multiple R-squared:  0.9974, Adjusted R-squared:  0.9953
## F-statistic: 478.1 on 4 and 5 DF,  p-value: 1.213e-06
```

```
g3=ggplot()+
  geom_point(aes(x=dataset$Level , y=dataset$Salary),
             colour="purple")+
  geom_line(aes(x=dataset$Level, y=predict(poly_reg1, newdata= dataset1)),
            colour="pink")+
  ggtitle("Truth or Bluff(Poly Regression)")+
  xlab('Level')+
  ylab("Salary")
g3
```



So, here we can see that, the data curve is much more smooth. And thus, the prediction is pretty accurate. And thus, we have build our lie detector with pretty good accuracy.

```
y_poly4_pred=predict(poly_reg1, data.frame(Level=6.5, level2=6.5^2, level3=6.5^3,
level4=6.5^4))
y_poly4_pred
```

```
##        1
## 158862.5
```

Conclusion:

As we can see the polynominal regession is better than the linear regression, in addition, 4th polynomial regression is better than the 3rd polynomial regression. We can make the final conclusion that the new employee told us the truth, because the predicted value is 158862.5 dollars.