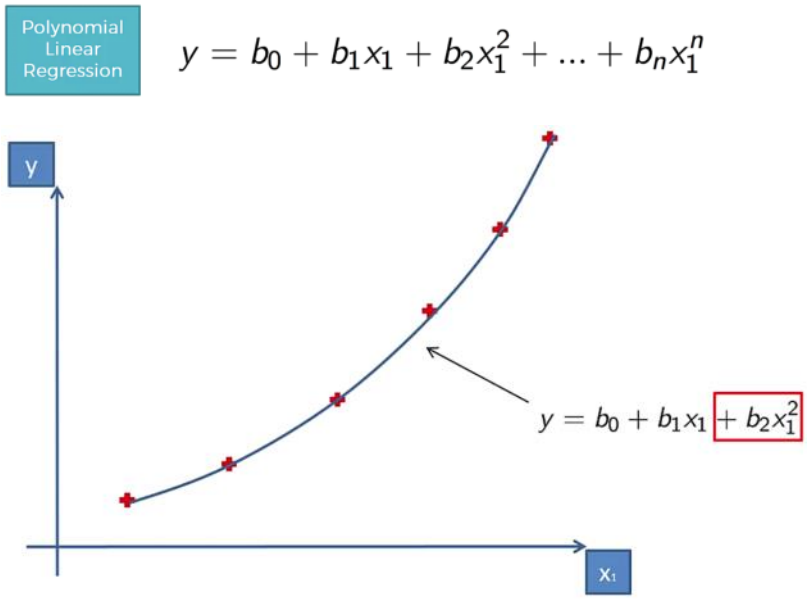
Polynomial Regression

Monday, January 16, 2017 10:27 AM

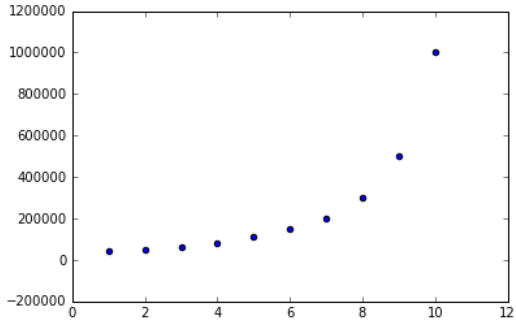
Here, we are using one variable but different powers of it.



It is called Polynomial Linear regression because, here the variables are b0, b1, etc. These are linear. Hence it is called so.

Data

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



Python

Data preprocessing steps are the same as normal. Here, we do not divide the data into test and train sets because the amount of information we have is limited and hence we are fitting the entire data.

```
# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Importing the dataset
dataset = pd.read_csv('Position_Salaries.csv')
X = dataset.iloc[:, 1:2].values
y = dataset.iloc[:, 2].values
```

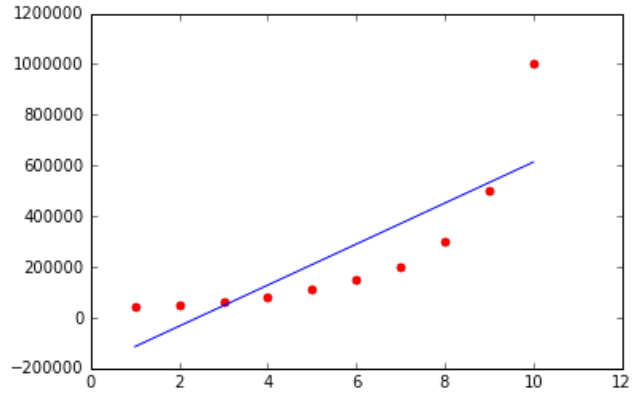
Now, we use a scatter plot to get to understand the variable correlations

```
plt.scatter(x=X, y=y)
```

We build a linear model first inorder to determine whether the variables can be linked to each other in a linear fashion. Over here, even though it's evident that such a thing is not possible, we do it for getting a contrast between a linear and polynomial model.

```
# Fitting Linear Regression to the Dataset
from sklearn.linear_model import LinearRegression
lin_reg = LinearRegression()
lin_reg.fit(X, y)
```

```
# Visualizing the Linear Regression results
y_pred_lin = lin_reg.predict(X)
plt.scatter(X, y, color='red')
plt.plot(X, y_pred_lin)
plt.show()
```



Now, we fit a polynomial model of degree 2. This is done as follows.

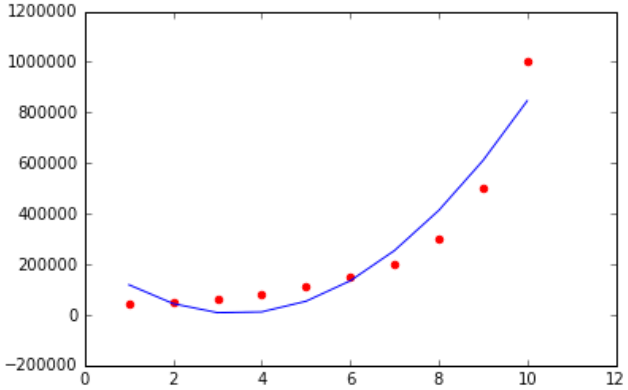
```
# Fitting Polynomial Regression to the Dataset
from sklearn.preprocessing import PolynomialFeatures
poly_reg = PolynomialFeatures(degree=2)
X_poly = poly_reg.fit_transform(X)
lin_reg_poly = LinearRegression()
lin_reg_poly.fit(X_poly, y)
```

PolynomialFeatures enumerates the powers of a particular feature.

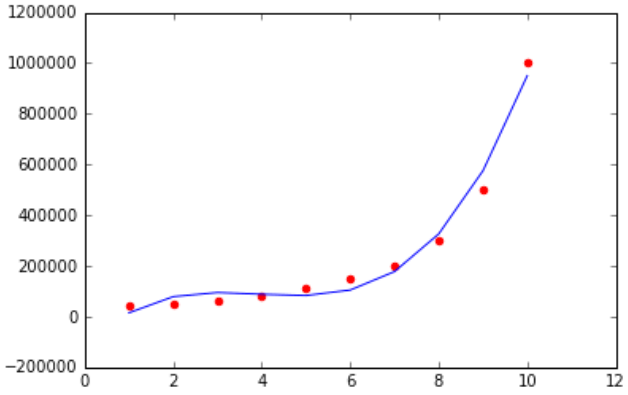
	0	1	2
0	1.000	1.000	1.000
1	1.000	2.000	4.000
2	1.000	3.000	9.000
3	1.000	4.000	16.000
4	1.000	5.000	25.000
5	1.000	6.000	36.000
6	1.000	7.000	49.000
7	1.000	8.000	64.000
8	1.000	9.000	81.000
9	1.000	10.000	100.000

We use a Linear model with this set of features to try to predict the Salary.
We plot this to get an understanding of the prediction now after using polynomial features.

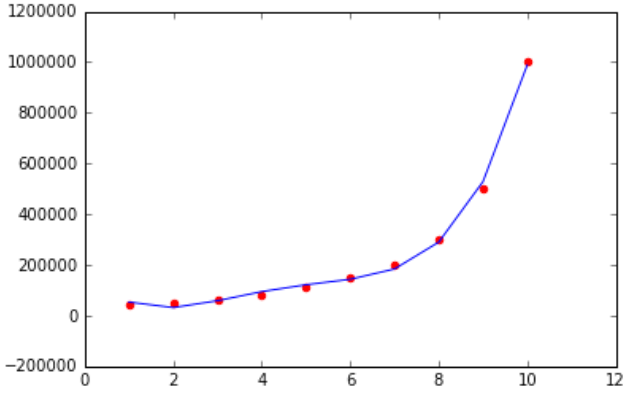
```
# Visualizing the Polynomial Regression results
y_pred_poly = lin_reg_poly.predict(X_poly)
plt.scatter(X, y, color='red')
plt.plot(X, y_pred_poly)
plt.show()
```



This is clearly a much better representation that a line and will provide a more accurate model for predicting the Price (y)
With degree = 3, we get



With degree = 4, we get



This is about as accurate as we can get and hence we will use this model to predict the salary given level.'
We can't use this model to predict salary based on Level.

```
# Predicting a new result with lin_reg
res = lin_reg.predict(6.5)
330378.78787879
```

```
res_poly = lin_reg_poly.predict(P)
print(res_poly)
158862.45265153
```

Thus the prediction by polynomial regression model seems more accurate and we can use this as reference.

R

Importing and preprocessing the dataset is similar to what has been done before. Here, we don't divide the data into train and test sets. Instead we work on the entire data.

```
# Polynomial Regression
# Importing the dataset
dataset = read.csv('Position_Salaries.csv')
dataset = dataset[2:3]
```

We utilize only the Level and Salary columns. Hence the range 2:3

We later build a linear model using R to act as a contrasting model to the polynomial model. We do this as follows.

```
# Fitting Linear Regression to the Dataset
lin_reg = lm(formula = Salary ~ ., data = dataset)
dataset_lin = dataset
summary(lin_reg)
```

Summary command is used to get statistical information about the model build like P-value corresponding to the variables involved.
Now to train a polynomial regression model, we create new features which are the raised to the power values of feature Level. This is done as follows

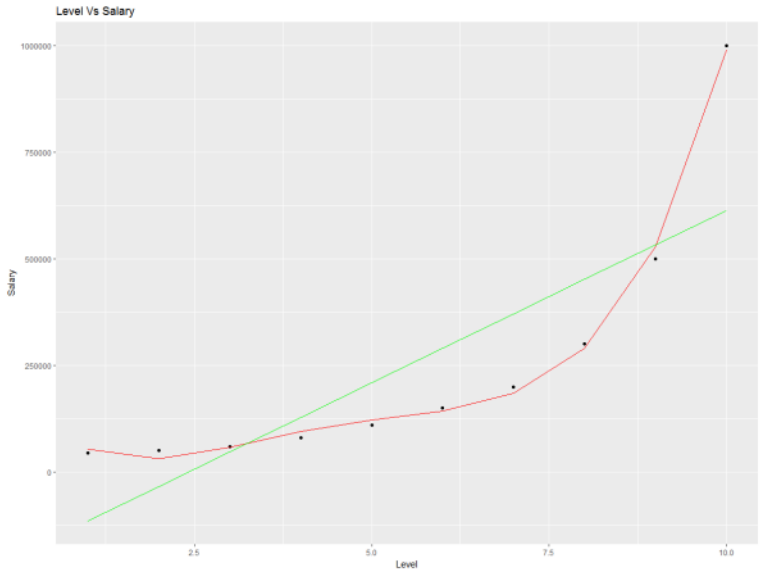
```
dataset$level_sqr = dataset$Level^2
dataset$level_cube = dataset$Level^3
dataset$level_quad = dataset$Level^4
```

Finally we build a polynomial regressor using this as follows.

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

We then Visualize the model predictions and contrast it with the Linear regression model.

```
ggplot() +
  geom_point(aes(dataset$Level, dataset$Salary)) +
  geom_line(aes(dataset$Level, predict(poly_reg, data = dataset)), color = 'red') +
  geom_line(aes(dataset_lin$Level, predict(lin_reg, data = dataset_lin)), color = 'green')
```



In order to predict the value of dependent variable given new data, we use

```
# Predicting Salary given a Level - Linear Model
y_pred_lin = predict(lin_reg, data.frame(Level = 6.5))
```

Result - 330379

```
# Predicting Salary givem a Level - Polynomial Model
y_pred_poly = predict(poly_reg, data.frame(Level = 6.5,
                                           level_sqr = 6.5^2,
                                           level_cube = 6.5^3,
                                           level_quad = 6.5^4))
```

Result - 158862