Regression

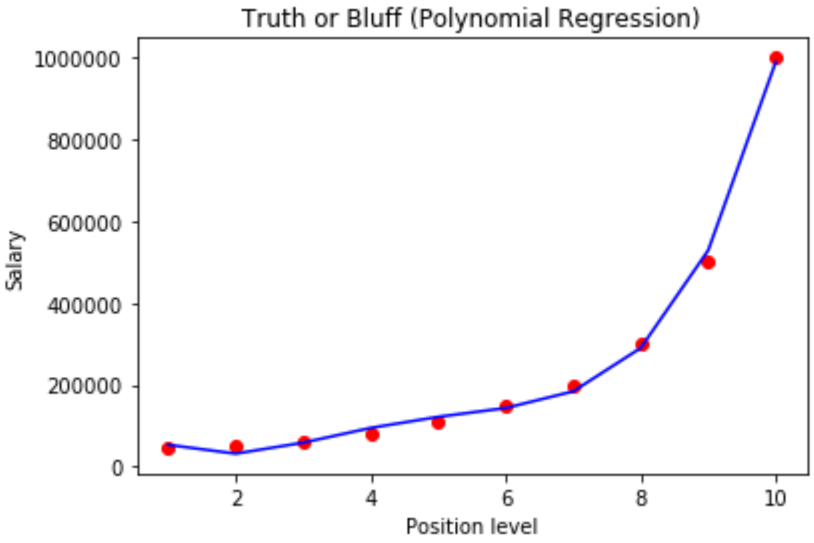
Polynomial Regression

posted Jan 24, 2018, 12:22 AM by Atul Rana **[updated 5 minutes ago]**

Introduction

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.

Although this model allows for a nonlinear relationship between Y and X, *polynomial regression* is still considered linear *regression* since it is linear in the *regression* coefficients,



Import pandas, numpy, matplotlib,(You'll import sklearn as you need it.)

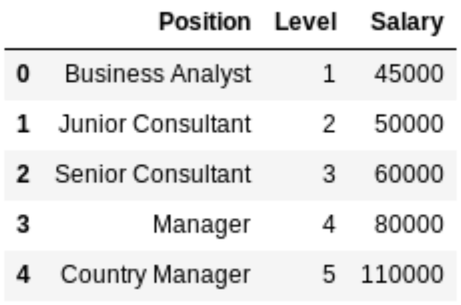
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

Getting the Data

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

printing the imported data file.

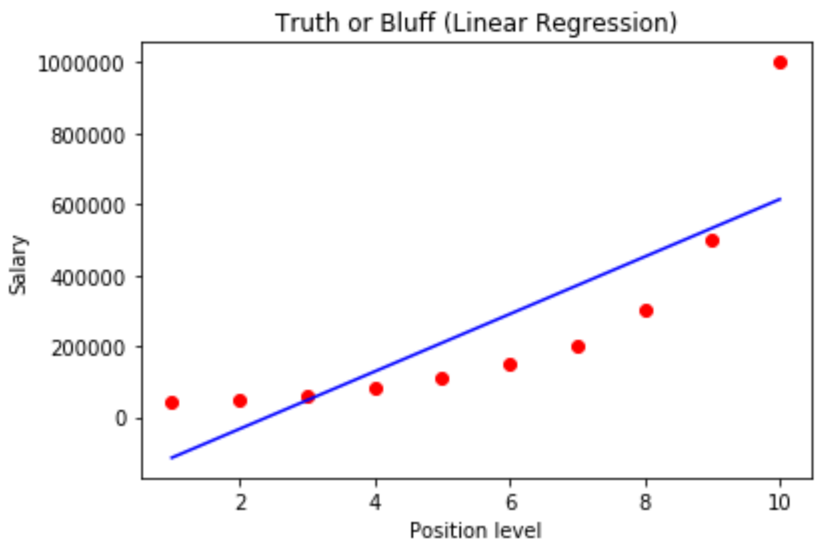
```
dataset.head()
```



	Position	Level	Salary
0	Business Analyst	1	45000
1	Junior Consultant	2	50000
2	Senior Consultant	3	60000
3	Manager	4	80000
4	Country Manager	5	110000

Training and Testing Data

```
from sklearn.linear_model import LinearRegression       #importing model
lin_reg = LinearRegression()                            #defining model object
lin_reg.fit(X, y)                                       #feedin data to model
```

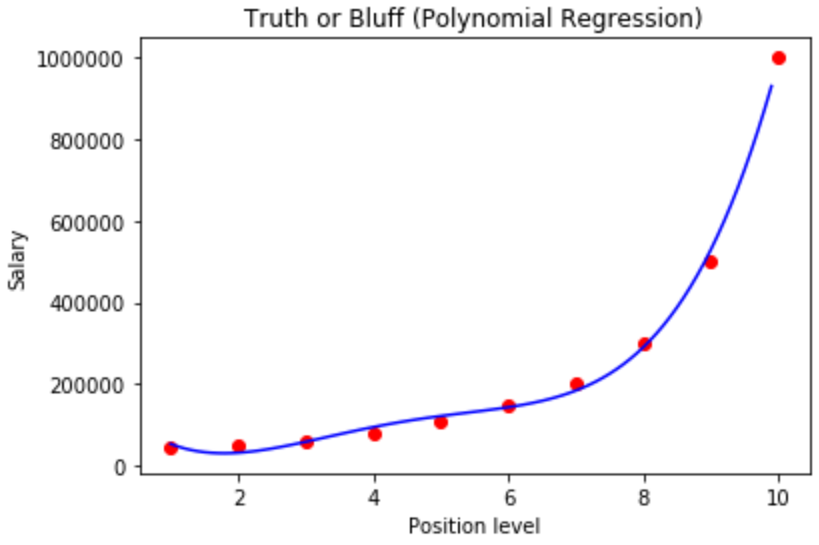


Defining the polynomial variable to the dataset.

```
from sklearn.preprocessing import PolynomialFeatures
poly_reg = PolynomialFeatures(degree = 4)
X_poly = poly_reg.fit_transform(X)
poly_reg.fit(X_poly, y)
lin_reg_2 = LinearRegression()
lin_reg_2.fit(X_poly, y)
```

Visualization of polynomial regression fitting on the dataset

```
X_grid = np.arange(min(X), max(X), 0.1)
X_grid = X_grid.reshape((len(X_grid), 1))
plt.scatter(X, y, color = 'red')
plt.plot(X_grid, lin_reg_2.predict(poly_reg.fit_transform(X_grid)), color = 'blue')
plt.title('Truth or Bluff (Polynomial Regression)')
plt.xlabel('Position level')
plt.ylabel('Salary')
plt.show()
```



```
lin_reg.score(X,y)
```

```
0.66904123319298947
```

```
lin_reg_2.score(X_poly,y)
```

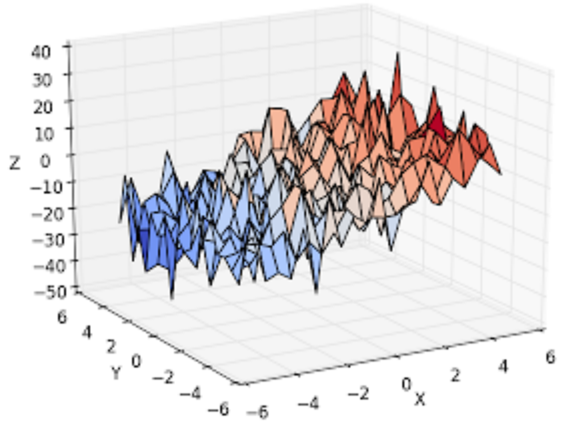
```
0.99739228917066147
```

Multiple Regression

posted Jan 23, 2018, 8:25 PM by Atul Rana **[updated 16 minutes ago]**

Introduction

Multiple regression is an extension of simple linear *regression*. It is used when we want to predict the value of a variable based on the value of two or more other variables. The variable we want to predict is called the dependent variable (or sometimes, the outcome, target or criterion variable).



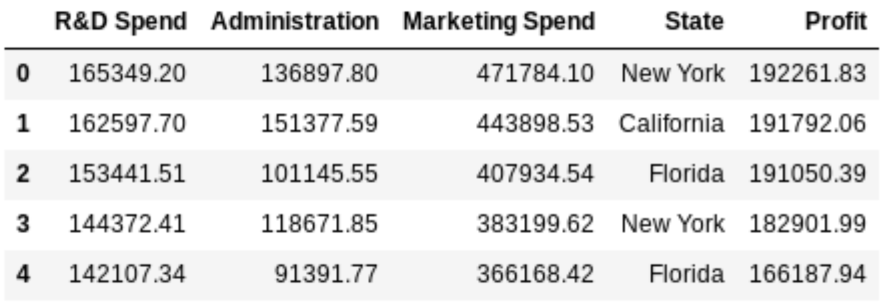
```
import numpy as np
import pandas as pd
```

Getting the Data

```
dataset = pd.read_csv('50_Startups.csv')
X = dataset[['R&D Spend','Administration','Marketing Spend']]
y = dataset["Profit"]
```

Have a look on the imported data.

```
dataset.head()
```



	R&D Spend	Administration	Marketing Spend	State	Profit
0	165349.20	136897.80	471784.10	New York	192261.83
1	162597.70	151377.59	443898.53	California	191792.06
2	153441.51	101145.55	407934.54	Florida	191050.39
3	144372.41	118671.85	383199.62	New York	182901.99
4	142107.34	91391.77	366168.42	Florida	166187.94

Use model_selection.train_test_split from sklearn to split the data into training and testing sets. Set test_size=0.3 and random_state=101

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state =
0)
```

Training the Model

Now its time to train our model on our training data! Import LinearRegression from sklearn.linear_model

```
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train, y_train)
```

Evalution and Predicting the Test Data

```
y_pred = regressor.predict(X_test)
```

```
regressor.coef_
```

```
array([ 0.77884104,  0.0293919 ,  0.03471025])
```

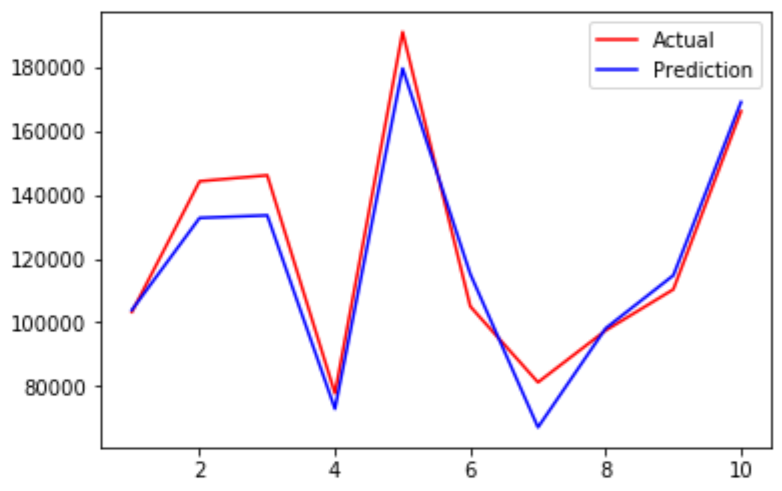
```
regressor.score(X_test,y_test)
```

```
0.93939559178205712
```

```
y_pred.shape
```

```
(10,)
```

```
plt.plot(range(1,11),y_test,c='red')
plt.plot(range(1,11),y_pred,c='blue')
plt.legend(['Actual','Prediction'],loc = 0)
plt.show()
```

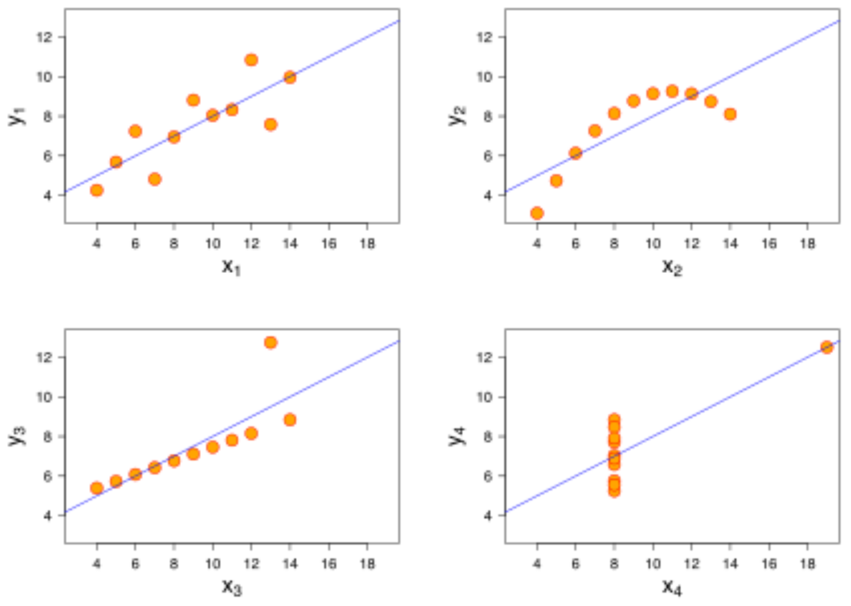


Linear Regression

posted Jan 23, 2018, 6:35 PM by Atul Rana **[updated 26 minutes ago]**

Introduction

Simple *linear regression* is a statistical method that allows us to summarize and study relationships between two continuous (quantitative) variables.

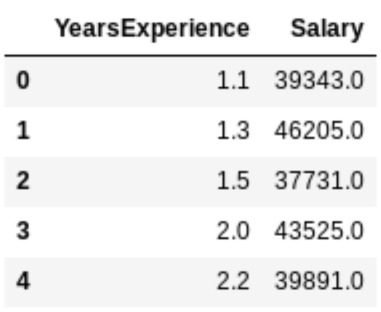


Import pandas, numpy, matplotlib (You'll import sklearn as you need it.)

```
import numpy as np
import pandas as pd
```

Getting the Data

```
dataset = pd.read_csv('Salary_Data.csv')
dataset.head()
```



	YearsExperience	Salary
0	1.1	39343.0
1	1.3	46205.0
2	1.5	37731.0
3	2.0	43525.0
4	2.2	39891.0

```
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, 1].values
```

Use model_selection.train_test_split from sklearn to split the data into training and testing sets. Set test_size=0.3 and random_state=101

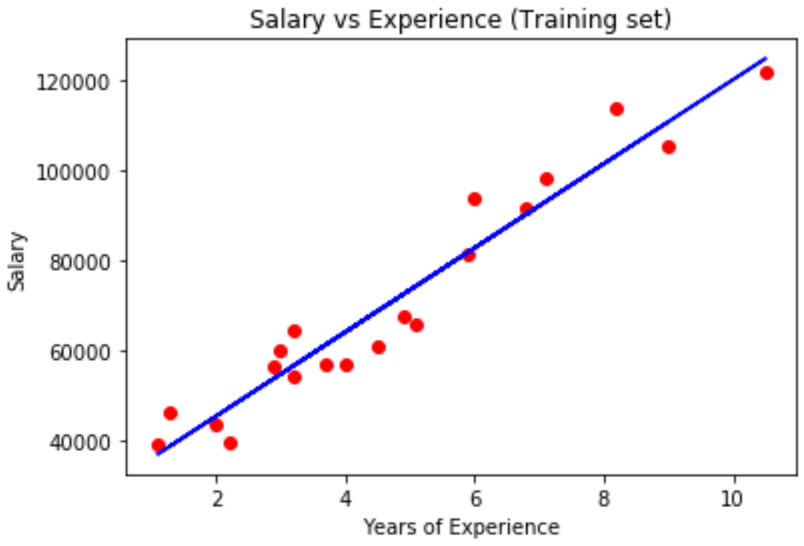
```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X, y, test_size = 1/3, random_state = 0)
```

Training the Model

Now its time to train our model on our training data! Import LinearRegression from sklearn.linear_model.

```
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
```

```
regressor.fit(X_train, y_train)
```

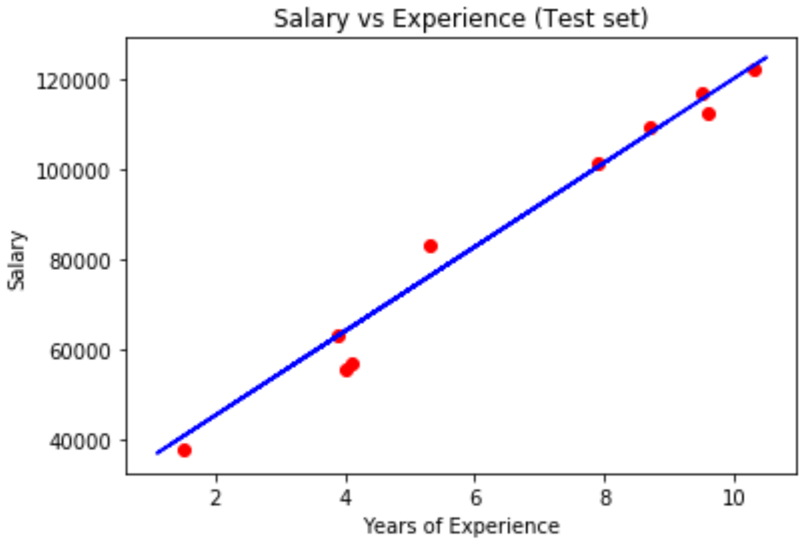


Predicting Test Data

Now that we have fit our model, let's evaluate its performance by predicting off the test values! Use regressor.predict() to predict off the X_test set of the data.

```
predictions = regressor.predict(X_test)
```

```
# Visualising the Test set results
plt.scatter(X_test, y_test, color = 'red')
plt.plot(X_train, regressor.predict(X_train), color = 'blue')
plt.title('Salary vs Experience (Test set)')
plt.xlabel('Years of Experience')
plt.ylabel('Salary')
plt.show()
```



1-3 of 3 ‹ ›