Multiple Linear Regression

Machine Learning

Dr. Adnan Abid

Regressions

Simple Linear Regression

$$y = b_0 + b_1 * x_1$$

Multiple Linear Regression

Dependent variable (DV) Independent variables (IVs)

$$y = b_0 + b_1 * x_1 + b_2 * x_2 + \ldots + b_n * x_n$$

Activate Windows

Dummy Variables

Profit	R&D Spend	Admin	Marketing	State
192,261.83	165,349.20	136,897.80	471,784.10	New York
191,792.06	162,597.70	151,377.59	443,898.53	California
191,050.39	153,441.51	101,145.55	407,934.54	California
182,901.99	144,372.41	118,671.85	383,199.62	New York
166,187.94	142,107.34	91,391.77	366,168.42	California

$$y = b_0 + b_1{}^*x_1 + b_2{}^*x_2 + b_3{}^*x_3 + \;???$$

Activate Windows
Go to Settings to activate Windows.

Dummy Variables

Profit	R&D Spend	Admin	Marketing	State
192,261.83	165,349.20	136,897.80	471,784.10	New York
191,792.06	162,597.70	151,377.59	443,898.53	California
191,050.39	153,441.51	101,145.55	407,934.54	California
182,901.99	144,372.41	118,671.85	383,199.62	New York
166,187.94	142,107.34	91,391.77	366,168.42	California

New York	California

$$y = b_0 + b_1{}^*x_1 + b_2{}^*x_2 + b_3{}^*x_3 + \ ???$$

Activate Windows
Go to Settings to activate Windows.

Dummy Variables

Profit	R&D Spend	Admin	Marketing	State	New York	California
192,261.83	165,349.20	136,897.80	471,784.10	New York	1	0
191,792.06	162,597.70	151,377.59	443,898.53	California	0	1
191,050.39	153,441.51	101,145.55	407,934.54	California	0	1
182,901.99	144,372.41	118,671.85	383,199.62	New York	1	0
166,187.94	142,107.34	91,391.77	366,168.42	California	0	1

$$y = b_0 + b_1{}^*x_1 + b_2{}^*x_2 + b_3{}^*x_3 + \quad ???$$

Activate Windows
Go to Settings to activate Windows.

Dummy Variables

Profit	R&D Spend	Admin	Marketing	State
192,261.83	165,349.20	136,897.80	471,784.10	~~New York~~
191,792.06	162,597.70	151,377.59	443,898.53	~~California~~
191,050.39	153,441.51	101,145.55	407,934.54	~~California~~
182,901.99	144,372.41	118,671.85	383,199.62	~~New York~~
166,187.94	142,107.34	91,391.77	366,168.42	~~California~~

Dummy Variables

New York	California
1	0
0	1
0	1
1	0
0	1

$$y = b_0 + b_1{*}x_1 + b_2{*}x_2 + b_3{*}x_3 + b_4{*}D_1$$

1. Add dummy variables for all possible values in the attribute with categorical data, which is STATE in this example.
2. Populate them with One Hot Encoding method
3. Replace the column with Dummy variables

Activate Windows
Go to Settings to activate Windows.

Dummy Variables

Profit	R&D Spend	Admin	Marketing	State
192,261.83	165,349.20	136,897.80	471,784.10	~~New York~~
191,792.06	162,597.70	151,377.59	443,898.53	~~California~~
191,050.39	153,441.51	101,145.55	407,934.54	~~California~~
182,901.99	144,372.41	118,671.85	383,199.62	~~New York~~
166,187.94	142,107.34	91,391.77	366,168.42	~~California~~

Dummy Variables

New York	California
1	~~0~~
0	~~1~~
0	~~1~~
1	~~0~~
0	~~1~~

$$y = b_0 + b_1{}^*x_1 + b_2{}^*x_2 + b_3{}^*x_3 \qquad + b_4{}^*D_1$$

1. Add dummy variables for all possible values in the attribute with categorical data, which is STATE in this example.
2. Populate them with One Hot Encoding method
3. Replace the column with Dummy variables

1. D1 works like a switch, when it is 1 then it means New York and in case of 0 it is California
2. Thus we may not include the last one, as the information will be complete even when we eliminate it.-

Dummy Variables

Profit	R&D Spend	Admin	Marketing	State
192,261.83	165,349.20	136,897.80	471,784.10	New York
191,792.06	162,597.70	151,377.59	443,898.53	California
191,050.39	153,441.51	101,145.55	407,934.54	California
182,901.99	144,372.41	118,671.85	383,199.62	New York
166,187.94	142,107.34	91,391.77	366,168.42	California

Dummy Variables

New York	California
1	0
0	1
0	1
1	0
0	1

$$y = b_0 + b_1{}^*x_1 + b_2{}^*x_2 + b_3{}^*x_3 + b_4{}^*D_1$$

1. D1 works like a switch, when it is 1 then it means New York and in case of 0 it is California
2. Thus we may not include the last one, as the information will be complete even when we eliminate it.

1. Theoretically, we SHOULD NOT include all the dummy variables in the equation.
2. Intuition is that when D1 is 0, then b4*D1 is also 0, and we may think that there is no coefficient for California.
3. Well, the coefficient for California is going to be included in b0, i.e. when D1 is 0 the equation becomes an expression for California.
4. When D1 is 1 then the expression becomes the difference between New York and California

Dummy Variable Trap

Profit	R&D Spend	Admin	Marketing	State
192,261.83	165,349.20	136,897.80	471,784.10	New York
191,792.06	162,597.70	151,377.59	443,898.53	California
191,050.39	153,441.51	101,145.55	407,934.54	California
182,901.99	144,372.41	118,671.85	383,199.62	New York
166,187.94	142,107.34	91,391.77	366,168.42	California

Dummy Variables

New York	California
1	0
0	1
0	1
1	0
0	1

$$y = b_0 + b_1{*}x_1 + b_2{*}x_2 + b_3{*}x_3 \qquad + b_4{*}D_1 + \underline{b_5{*}D_2}$$

Activate Windows
Go to Settings to activate Windows.

1. The fact that one or more independent variables in a linear regression predict another is called multi-collinearity.
2. If so, the model cannot distinguish the facts of D1 from the facts of D2.
3. This is called Dummy Variable Trap.
4. Thus, you cannot have b0, D1, and D2 in the model at the same time.

Dummy Variable Trap

					Dummy Variables	
Profit	R&D Spend	Admin	Marketing	State	New York	California
192,261.83	165,349.20	136,897.80	471,784.10	New York	1	0
191,792.06	162,597.70	151,377.59	443,898.53	California	0	1
191,050.39	153,441.51	101,145.55	407,934.54	California	0	1
182,901.99	144,372.41	118,671.85	383,199.62	New York	1	0
166,187.94	142,107.34	91,391.77	366,168.42	California	0	1

$$y = b_0 + b_1{*}x_1 + b_2{*}x_2 + b_3{*}x_3 \qquad + b_4{*}D_1 + \underline{b_5{*}D_2}$$

1. Never include all the dummy variables.
2. Always ignore 1 dummy variable. If you have 10 dummy variables, then include 9 and leave 1; if you have 100 dummy variables include 99 and leave 1.

1. Never include all the dummy variables.
2. If you have more than one categorical data attributes, then you have to do the same for the other variable too, i.e. include n-1 dummy variables and leave one out.
3. For instance, if you have another variable that tell about the target sector of the organization, then we shall follow the same process all over for this variable too.

Activate Windows
Go to Settings to activate Windows.

Dummy Variable Trap

Profit	R&D Spend	Admin	Marketing	State
192,261.83	165,349.20	136,897.80	471,784.10	New York
191,792.06	162,597.70	151,377.59	443,898.53	California
191,050.39	153,441.51	101,145.55	407,934.54	California
182,901.99	144,372.41	118,671.85	383,199.62	New York
166,187.94	142,107.34	91,391.77	366,168.42	California

Dummy Variables

New York	California
1	0
0	1
0	1
1	0
0	1

$$y = b_0 + b_1{*}x_1 + b_2{*}x_2 + b_3{*}x_3 \qquad + b_4{*}D_1 + \cancel{b_5{*}D_2}$$

Always omit one dummy variable

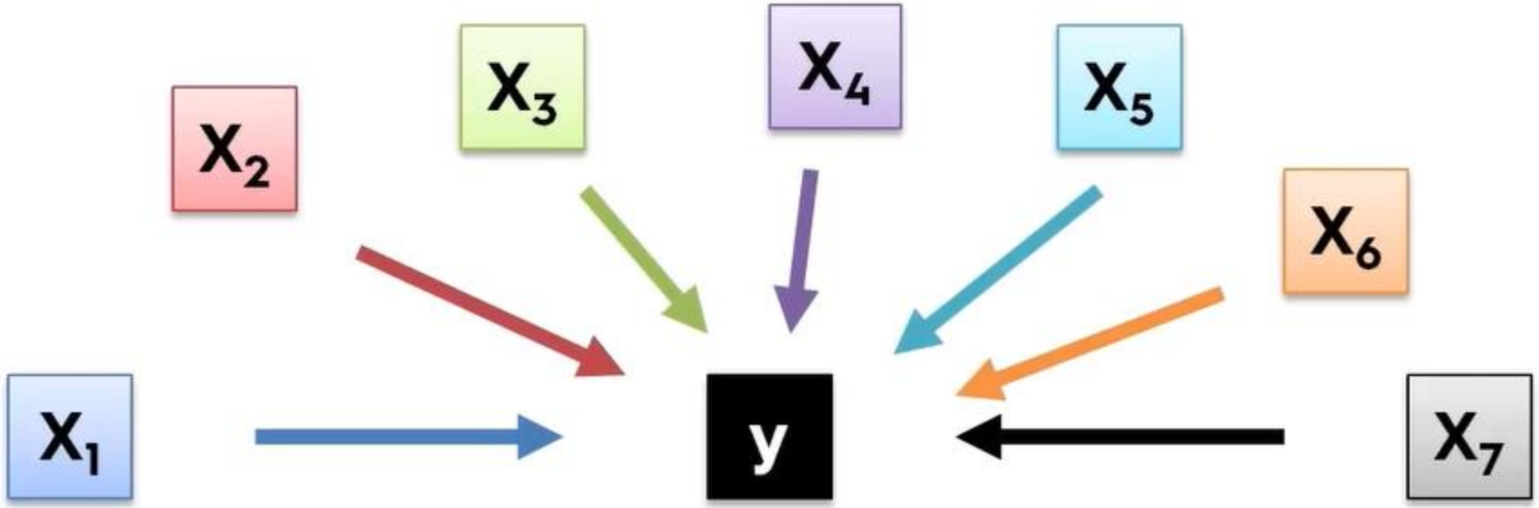
Activate Windows
Go to Settings to activate Windows.

Building A Model

(Step-By-Step)

Activate Windows
Go to Settings to activate Windows.

Building A Model



1. There are a lot of variables
2. Should we include all the independent variables?
3. In fact, we need to include the ones which are helpful in prediction, and drop the rest.
4. There are many methods for doing it…

Activate Windows

Building A Model

5 methods of building models:

1. All-in
2. Backward Elimination
3. Forward Selection
4. Bidirectional Elimination
5. Score Comparison

Activate Windows
Go to Settings to activate Windows.

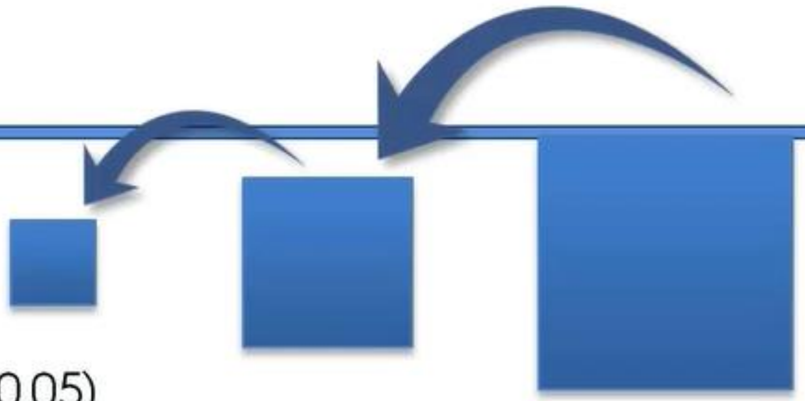
Building A Model

"All-in" – cases:

- Prior knowledge; OR
- You have to; OR
- Preparing for Backward Elimination

Activate Windows
Go to Settings to activate Windows

Building A Model



Backward Elimination

STEP 1: Select a significance level to stay in the model (e.g. SL = 0.05)

STEP 2: Fit the full model with all possible predictors

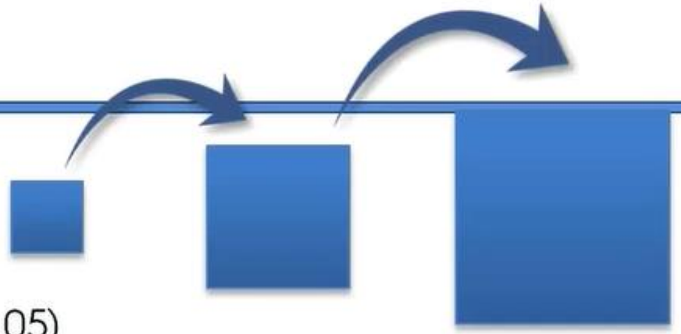
STEP 3: Consider the predictor with the highest P-value. If P > SL, go to STEP 4, otherwise go to FIN

STEP 4: Remove the predictor

STEP 5: Fit model without this variable*

1. Remove the attribute with highest P-value, if the P-value is greater than the threshold Significance Level.
2. Then go to Step 3 and rebuild the model
3. If there is not such variable with P-value higher than SL, the go to FINISH (FIN) state, which means the model is ready and all the remain attributes are statistically significant for prediction.

Building A Model



Forward Selection

STEP 1: Select a significance level to enter the model (e.g. SL = 0.05)

STEP 2: Fit all simple regression models $\mathbf{y} \sim \mathbf{x_n}$ Select the one with the lowest P-value

STEP 3: Keep this variable and fit all possible models with one extra predictor added to the one(s) you already have

STEP 4: Consider the predictor with the lowest P-value. If P < SL, go to STEP 3, otherwise go to FIN

1. Add one variable at a time.
2. Looks reverse of backward elimination, but is much more tedious and complex.
3. Start with a single variable (Simple Regression) like manner.
4. Choose the variable which has lowest SL.
5. Then add each of the remaining variables one by one; and choose the one with lowest SL value.
6. Keep doing it till you find variable with P values < SL.
7. If you are unable to find any variable which, if added, to the model does not offer PL < SL.

Building A Model

Forward Selection

STEP 1: Select a significance level to enter the model (e.g. SL = 0.05)

STEP 2: Fit all simple regression models **$y \sim x_n$** Select the one with the lowest P-value

STEP 3: Keep this variable and fit all possible models with one extra predictor added to the one(s) you already have

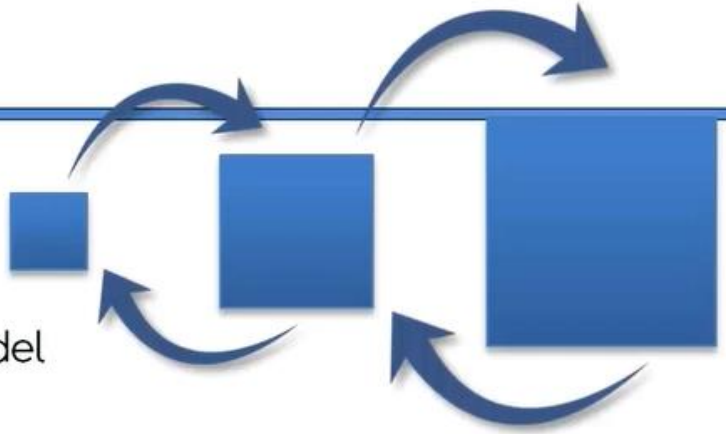
STEP 4: Consider the predictor with the lowest P-value. If ~~P < SL~~, go to STEP 3, otherwise go to FIN

1. Stop when you are unable to find a variable where P < SL
2. IMPORTANT: Use the previous model, as this attribute does not satisfy the SL criteria. Thus this model should not be included in the model.

FIN: Keep the previous model

Activate Windows
Go to Settings to activate Windows.

Building A Model



Bidirectional Elimination

STEP 1: Select a significance level to enter and to stay in the model
e.g.: SLENTER = 0.05, SLSTAY = 0.05

STEP 2: Perform the next step of Forward Selection (new variables must have: P < SLENTER to enter)

STEP 3: Perform ALL steps of Backward Elimination (old variables must have P < SLSTAY to stay)

STEP 4: No new variables can enter and no old variables can exit

FIN: Your Model Is Ready

Activate Windows
Go to Settings to activate Windows.

Building A Model

All Possible Models

STEP 1: Select a criterion of goodness of fit (e.g. Akaike criterion)

STEP 2: Construct All Possible Regression Models: 2^N-1 total combinations

STEP 3: Select the one with the best criterion

Activate Windows
Go to Settings to activate Windows

Building A Model

All Possible Models

STEP 1: Select a criterion of goodness of fit (e.g. Akaike criterion)

STEP 2: Construct All Possible Regression Models: 2^N-1 total combinations

STEP 3: Select the one with the best criterion

FIN: Your Model Is Ready

Example:
10 columns means
1,023 models

Activate Windows
Go to Settings to activate Windows.

Building A Model

5 methods of building models:

1. All-in
2. Backward Elimination
3. Forward Selection
4. Bidirectional Elimination
5. Score Comparison

Activate Windows
Go to Settings to activate Windows.

```
Console 1/A
Date:                  Sun, 25 Sep 2016   Prob (F-statistic):           1.34e-27
Time:                          16:13:59   Log-Likelihood:                -525.38
No. Observations:                    50   AIC:                             1063.
Df Residuals:                        44   BIC:                             1074.
Df Model:                             5
Covariance Type:              nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [95.0% Conf. Int.]
------------------------------------------------------------------------------
const       5.013e+04   6884.820      7.281      0.000      3.62e+04   6.4e+04
x1           198.7888   3371.007      0.059      0.953     -6595.030  6992.607
x2           -41.8870   3256.039     -0.013      0.990     -6604.003  6520.229
x3             0.8060      0.046     17.369      0.000         0.712     0.900
x4            -0.0270      0.052     -0.517      0.608        -0.132     0.078
x5             0.0270      0.017      1.574      0.123        -0.008     0.062
==============================================================================
Omnibus:                       14.782   Durbin-Watson:                   1.283
Prob(Omnibus):                  0.001   Jarque-Bera (JB):               21.266
Permissions: RW   End-of-lines: LF   Encoding: UTF-8-GUESSED   Line: 49   Column: 24   Memory: 55 %
```

```
IPython console
Console 1/A
==============================================================================
                 coef    std err          t      P>|t|      [95.0% Conf. Int.]
------------------------------------------------------------------------------
const       5.011e+04   6647.870      7.537      0.000      3.67e+04  6.35e+04
x1           220.1585   2900.536      0.076      0.940     -5621.821  6062.138
x2             0.8060      0.046     17.606      0.000         0.714     0.898
x3            -0.0270      0.052     -0.523      0.604        -0.131     0.077
x4             0.0270      0.017      1.592      0.118        -0.007     0.061
==============================================================================
Omnibus:                       14.758   Durbin-Watson:                   1.282
Prob(Omnibus):                  0.001   Jarque-Bera (JB):               21.172
Skew:                          -0.948   Prob(JB):                     2.53e-05
Kurtosis:                       5.563   Cond. No.                     1.40e+06
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is
correctly specified.
Permissions: RW   End-of-lines: LF   Encoding: UTF-8-GUESSED   Line: 50   Column: 17   Memory: 56 %
```

```python
# Building the optimal model using Backward Elimination
import statsmodels.formula.api as sm
X = np.append(arr = np.ones((50, 1)).astype(int), values = X, axis =
X_opt = X[:, [0, 1, 2, 3, 4, 5]]
regressor_OLS = sm.OLS(endog = y, exog = X_opt).fit()
regressor_OLS.summary()
X_opt = X[:, [0, 1, 3, 4, 5]]
regressor_OLS = sm.OLS(endog = y, exog = X_opt).fit()
regressor_OLS.summary()
X_opt = X[:, [0, 3, 4, 5]]
regressor_OLS = sm.OLS(endog = y, exog = X_opt).fit()
regressor_OLS.summary()
```

Activate Windows
Go to Settings to activate Windows.

Spyder (Python 3.5)

/Users/Hadelin/Desktop/Machine Learning A-Z/Part 2 - Regression/Section 5 - Multiple Linear Regression

Editor - /Users/Hadelin/Desktop/Machine Learning A-Z/Part 2 - Regression/Section 5 - Multiple Linear Regression/multiple_linear_regression.py

data_preprocessing_template.py | multiple_linear_regression.py*

```python
X = onehotencoder.fit_transform(X).toarray()

# Avoiding the Dummy Variable Trap
X = X[:, 1:]

# Splitting the dataset into the Training set and Test set
from sklearn.cross_validation import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_s

# Feature Scaling
"""from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()
X_train = sc_X.fit_transform(X_train)
X_test = sc_X.transform(X_test)"""

# Fitting Multiple Linear Regression to the Training set
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train, y_train)

# Predicting the Test set results
y_pred = regressor.predict(X_test)

# Building the optimal model using Backward Elimination
import statsmodels.formula.api as sm
X = np.append(arr = np.ones((50, 1)).astype(int), values = X, axis = 1)
X_opt = X[:, [0, 1, 2, 3, 4, 5]]
regressor_OLS = sm.OLS(endog = y, exog = X_opt).fit()
regressor_OLS.summary()
X_opt = X[:, [0, 1, 3, 4, 5]]
regressor_OLS = sm.OLS(endog = y, exog = X_opt).fit()
regressor_OLS.summary()
X_opt = X[:, [0, 3, 4, 5]]
regressor_OLS = sm.OLS(endog = y, exog = X_opt).fit()
regressor_OLS.summary()
X_opt = X[:, [0, 3, 5]]
regressor_OLS = sm.OLS(endog = y, exog = X_opt).fit()
regressor_OLS.summary()
X_opt = X[:, [0, 3]]
regressor_OLS = sm.OLS(endog = y, exog = X_opt).fit()
regressor_OLS.summary()
```

Variable explorer

Name	Type	Size	Value
X	float64	(50, 6)	array([[1.00000000e+00, 0.00000000e+00, 1.00000000e+00, 1.65349200e+05, 1.36897800e+05, 4.71784100e+05],
X_opt	float64	(50, 2)	array([[1.00000000e+00, 1.65349200e+05], [1.00000000e+00, 1.62597700e+05],
X_test	float64	(10, 5)	array([[1.00000000e+00, 0.00000000e+00, 6.60515200e+04, 1.82645560e+05, 1.18148200e+05],
X_train	float64	(40, 5)	array([[1.00000000e+00, 0.00000000e+00, 5.54939500e+04, 1.03057490e+05, 2.14634810e+05],
dataset	DataFrame	(50, 5)	Column names: R&D Spend, Administration, Marketing Spend, State, Profit
y	float64	(50,)	array([192261.83, 191792.06, 191050.39, 182901.99, 166187.94, 156991.12, 156122.51, 155752.6 , 152211.77, 149759.96,
y_pred	float64	(10,)	array([103015.20159796, 132582.27760815, 132447.73845175, 71976.09851258, 178537.48221056, 116161.24230166,
y_test	float64	(10,)	array([103282.38, 144259.4 , 146121.95, 77798.83, 191050.39, 105008.31, 81229.06, 97483.56, 110352.25, 166187.94])
y_train	float64	(40,)	array([96778.92, 96479.51, 105733.54, 96712.8 , 124266.9 , 155752.6 , 132602.65, 64926.08, 35673.41, 101004.64,

Object inspector | Variable explorer | File explorer

IPython console

Console 1/A

```
==============================================================================
Dep. Variable:                      y   R-squared:                       0.947
Model:                            OLS   Adj. R-squared:                  0.945
Method:                 Least Squares   F-statistic:                     849.8
Date:                Sun, 25 Sep 2016   Prob (F-statistic):           3.50e-32
Time:                        19:17:55   Log-Likelihood:                -527.44
No. Observations:                  50   AIC:                             1059.
Df Residuals:                      48   BIC:                             1063.
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [95.0% Conf. Int.]
------------------------------------------------------------------------------
const       4.903e+04   2537.897     19.320      0.000      4.39e+04  5.41e+04
x1             0.8543      0.029     29.151      0.000         0.795     0.913
==============================================================================
Omnibus:                       13.727   Durbin-Watson:                   1.116
Prob(Omnibus):                  0.001   Jarque-Bera (JB):               18.536
```

Activate Windows
Go to Settings to activate Windows.

Permissions: RW End-of-lines: LF Encoding: UTF-8-GUESSED Line: 58 Column: 24 Memory: 56 %

Polynomial Linear Regression

Regressions

Simple Linear Regression

$$y = b_0 + b_1 x_1$$

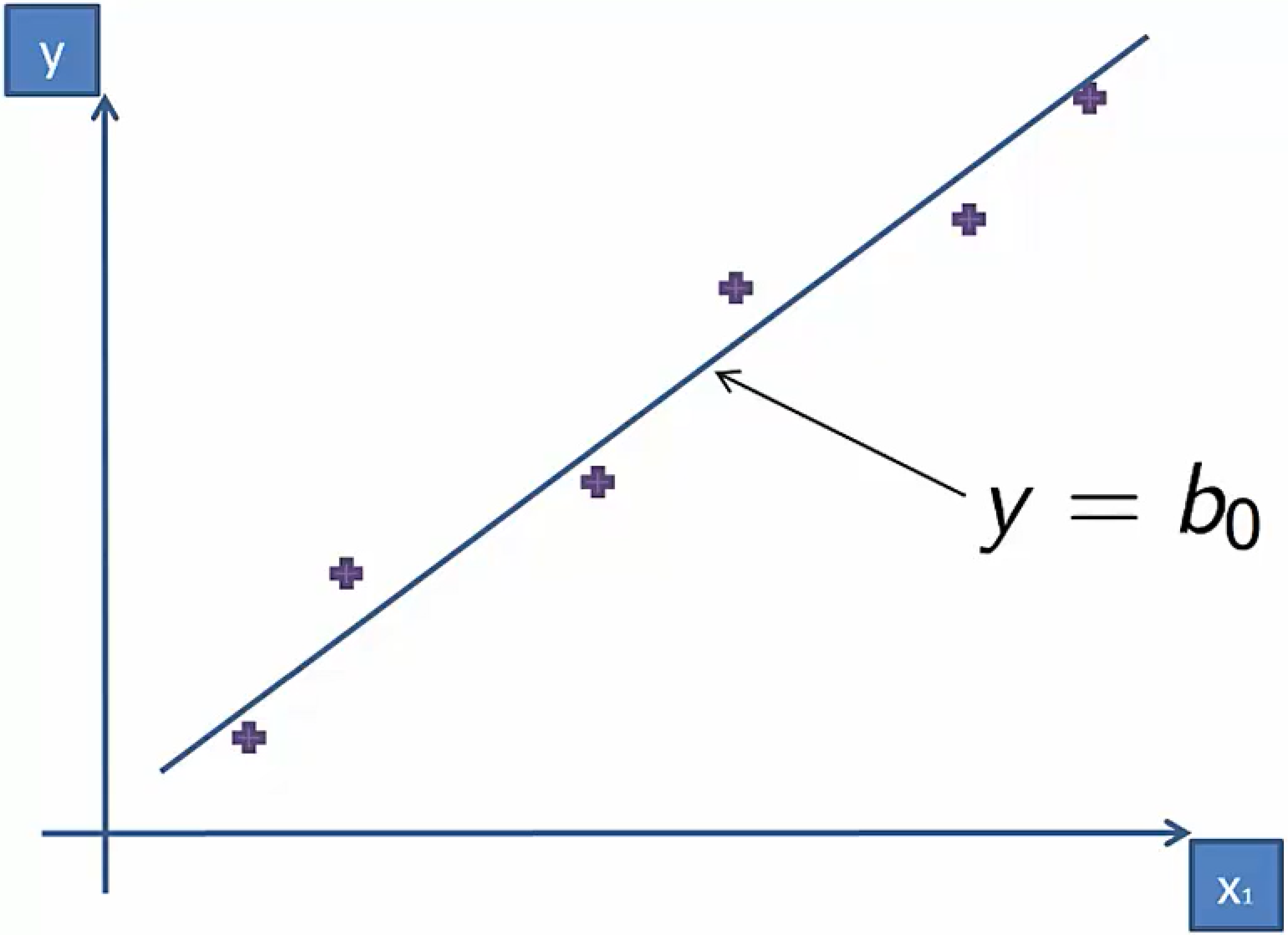
Multiple Linear Regression

$$y = b_0 + b_1 x_1 + b_2 x_2 + \dots + b_n x_n$$

Polynomial Linear Regression

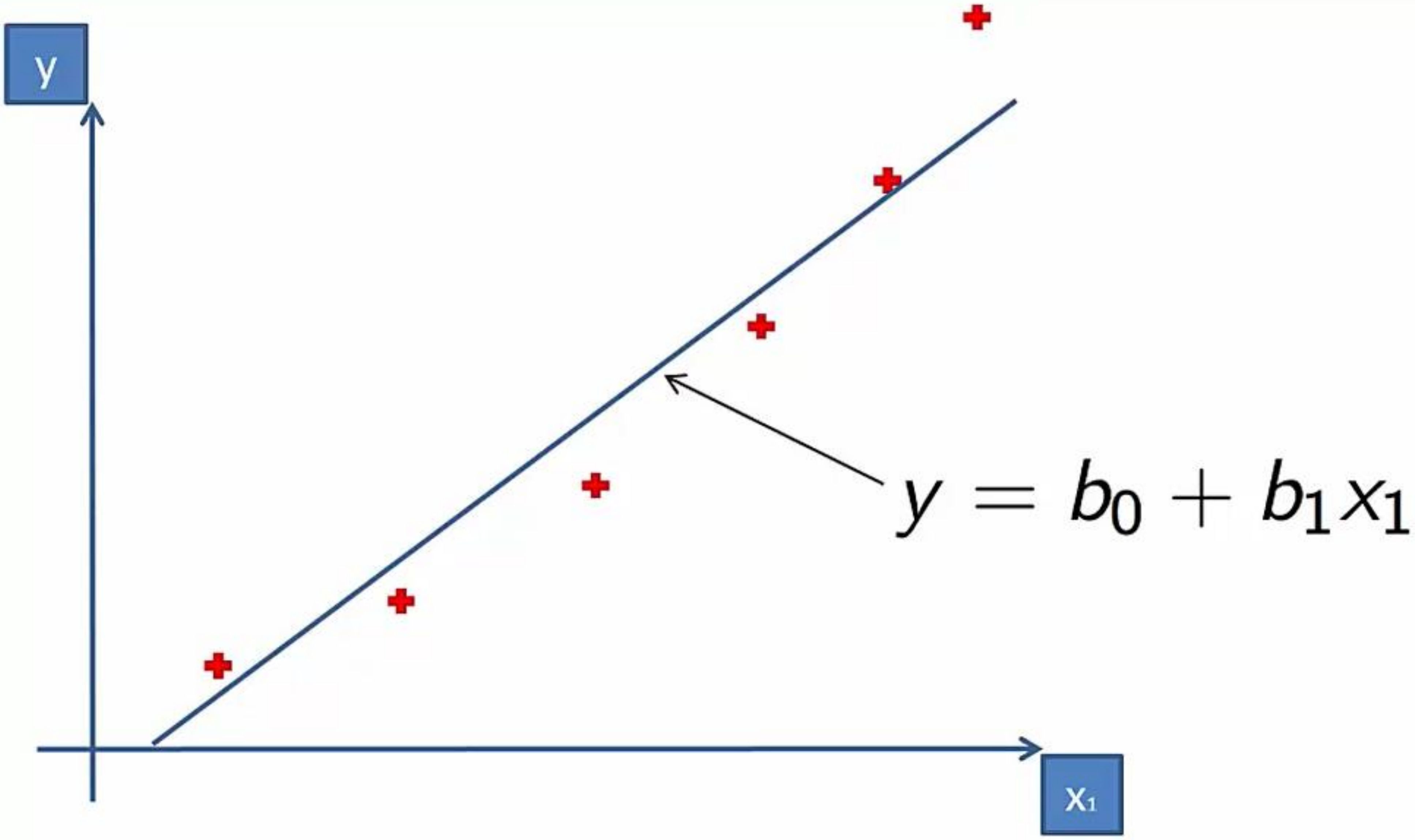
$$y = b_0 + b_1 x_1 + b_2 x_1^2 + \dots + b_n x_1^n$$

Simple Linear Regression

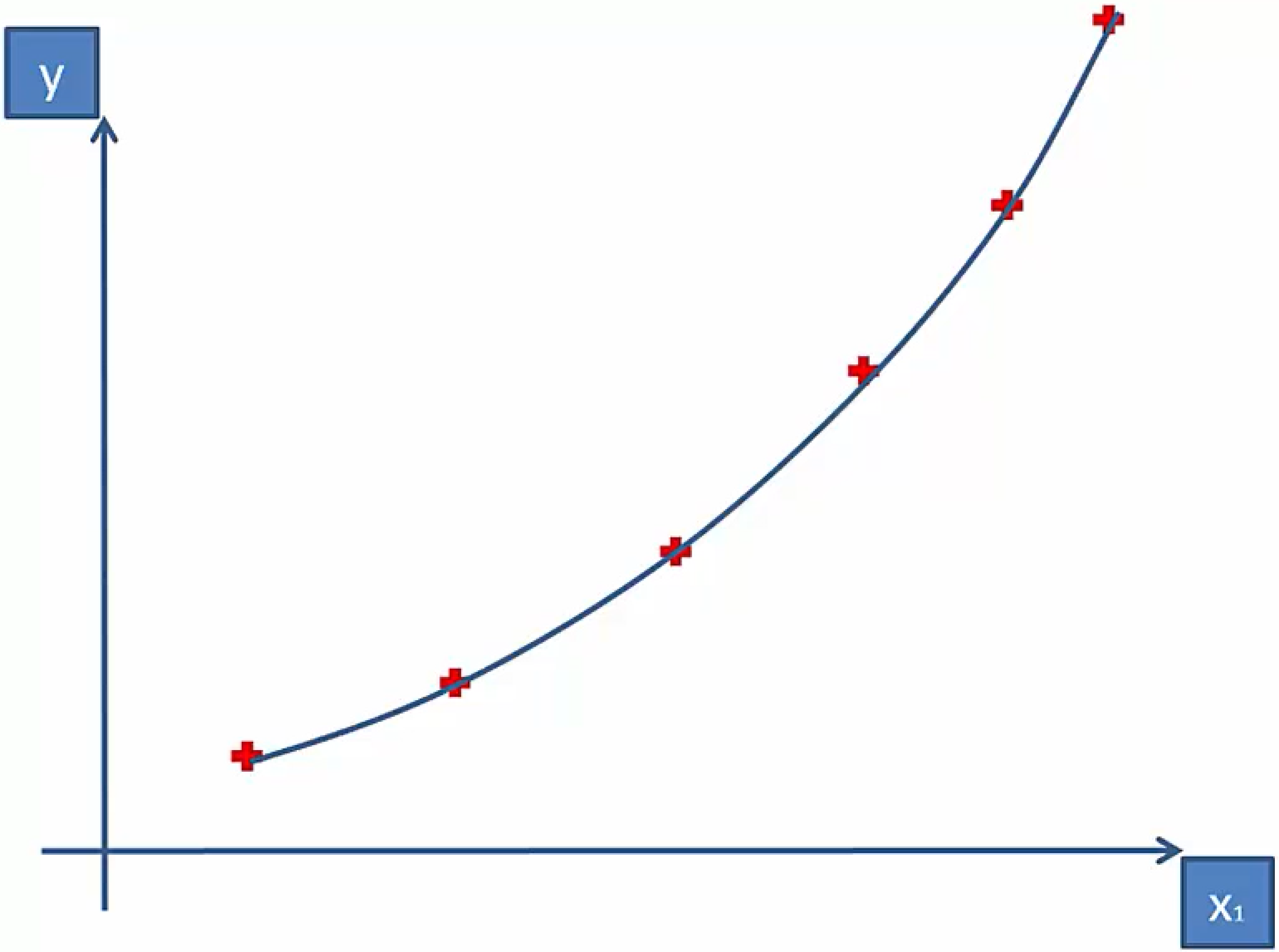


$$y = b_0 + b_1 x_1$$

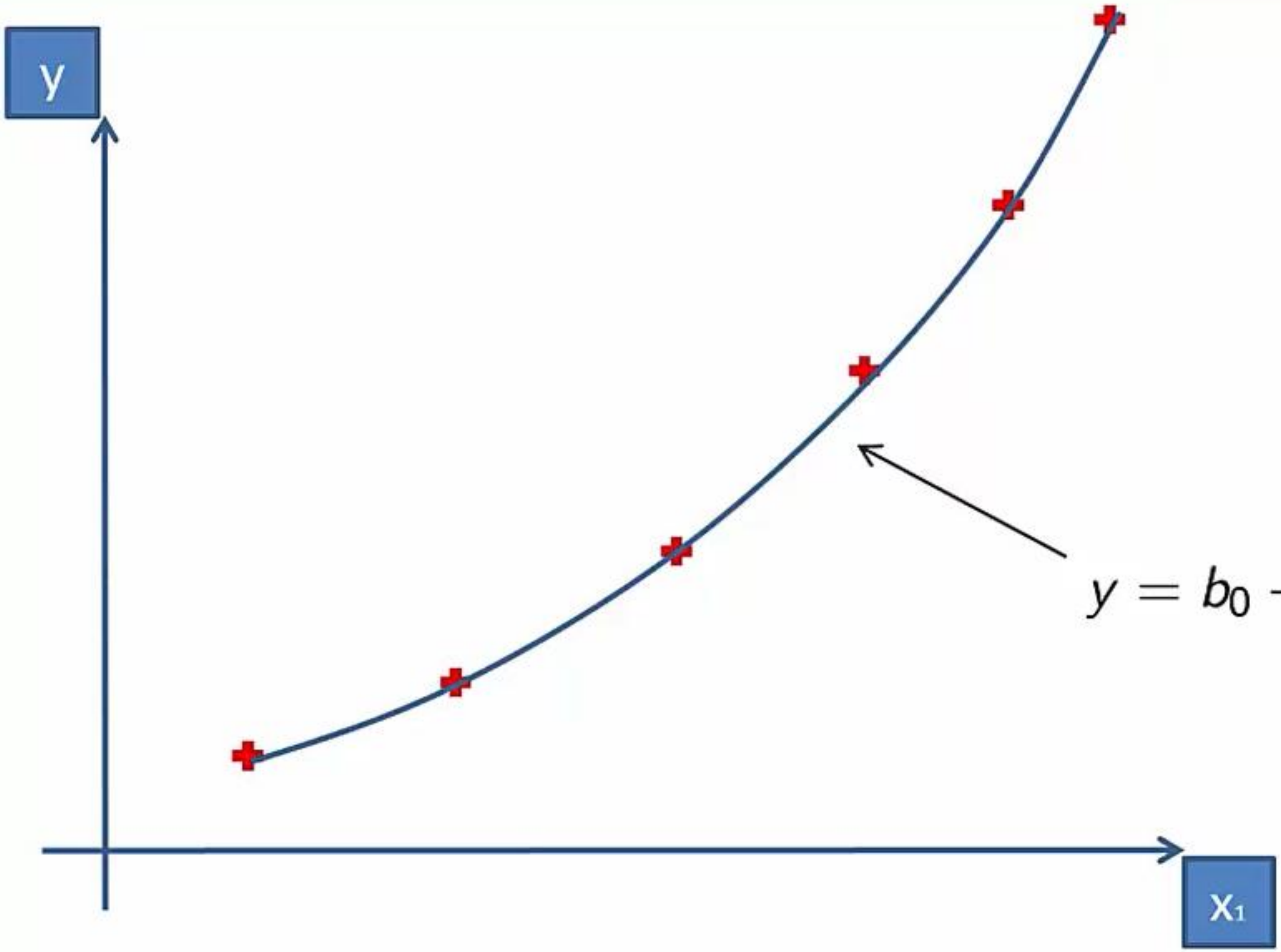
Simple Linear Regression



Polinomial Regression



Polinomial Regression



$$y = b_0 + b_1 x_1 \boxed{+ b_2 x_1^2}$$

Polinomial Regression

Polynomial Linear Regression

$$y = b_0 + b_1 x_1 + b_2 x_1^2 + \ldots + b_n x_1^n$$

data_preprocessing_template.py | polynomial_regression.py*

```python
# Feature Scaling
"""from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()
X_train = sc_X.fit_transform(X_train)
X_test = sc_X.transform(X_test)"""

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

# Fitting Polynomial Regression 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)

# Visualising the Linear Regression results
plt.scatter(X, y, color = 'red')
plt.plot(X, lin_reg.predict(X), color = 'blue')
plt.title('Truth or Bluff (Linear Regression)')
plt.xlabel('Position level')
plt.ylabel('Salary')
plt.show()

# Visualising the Polynomial Regression results
plt.scatter(X, y, color = 'red')
plt.plot(X, lin_reg_2.predict(poly_reg.fit_transform(X)), color = 'blue')
plt.title('Truth or Bluff (Polynomial Regression)')
plt.xlabel('Position level')
plt.ylabel('Salary')
plt.show()

# Predicting a new result with Linear Regression
lin_reg.predict(6.5)

# Predicting a new result with Polynomial Regression
lin_reg_2.predict(poly_reg.fit_transform(6.5))
```

Simple Linear Regression Model

Polynomial Regression Model

Simple Linear Regression Model Prediction

Polynomial Regression Model Prediction

Name	Type	Size	Value
X	int64	(10, 1)	array([[1], [2],
X_poly	float64	(10, 5)	array([[1.00000000e+00, 1.00000000e+00, 1.00000000e+00, 1.00000000e+00, 1.00000000e+00],
dataset	DataFrame	(10, 3)	Column names: Position, Level, Salary
y	int64	(10,)	array([45000, 50000, 60000, 80000, 110000, 150000, 200000, 300000, 500000, 1000000])

Object inspector | Variable explorer | File explorer

IPython console

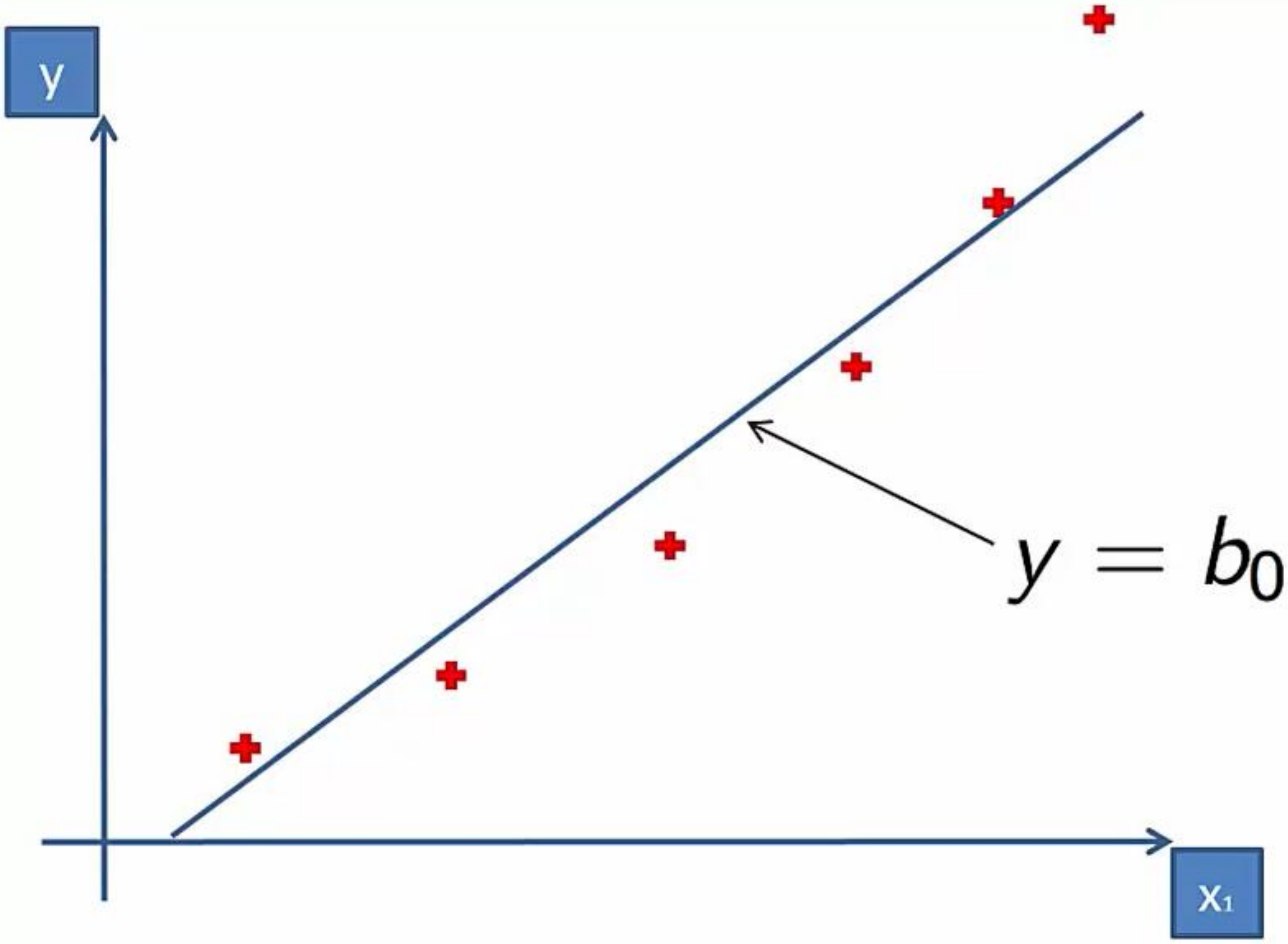
Console 1/A

```
   ...: plt.ylabel('Salary')
   ...: plt.show()

In [14]: lin_reg.predict(6.5)
Out[14]: array([ 330378.78787879])

In [15]: plt.scatter(X, y, color = 'red')
   ...: plt.plot(X, lin_reg.predict(X), color = 'blue')
   ...: plt.title('Truth or Bluff (Linear Regression)')
   ...: plt.xlabel('Position level')
   ...: plt.ylabel('Salary')
   ...: plt.show()

In [16]: lin_reg_2.predict(poly_reg.fit_transform(6.5))
Out[16]: array([ 158862.45265153])

In [17]:
```

Activate Windows
Go to Settings to activate Windows.

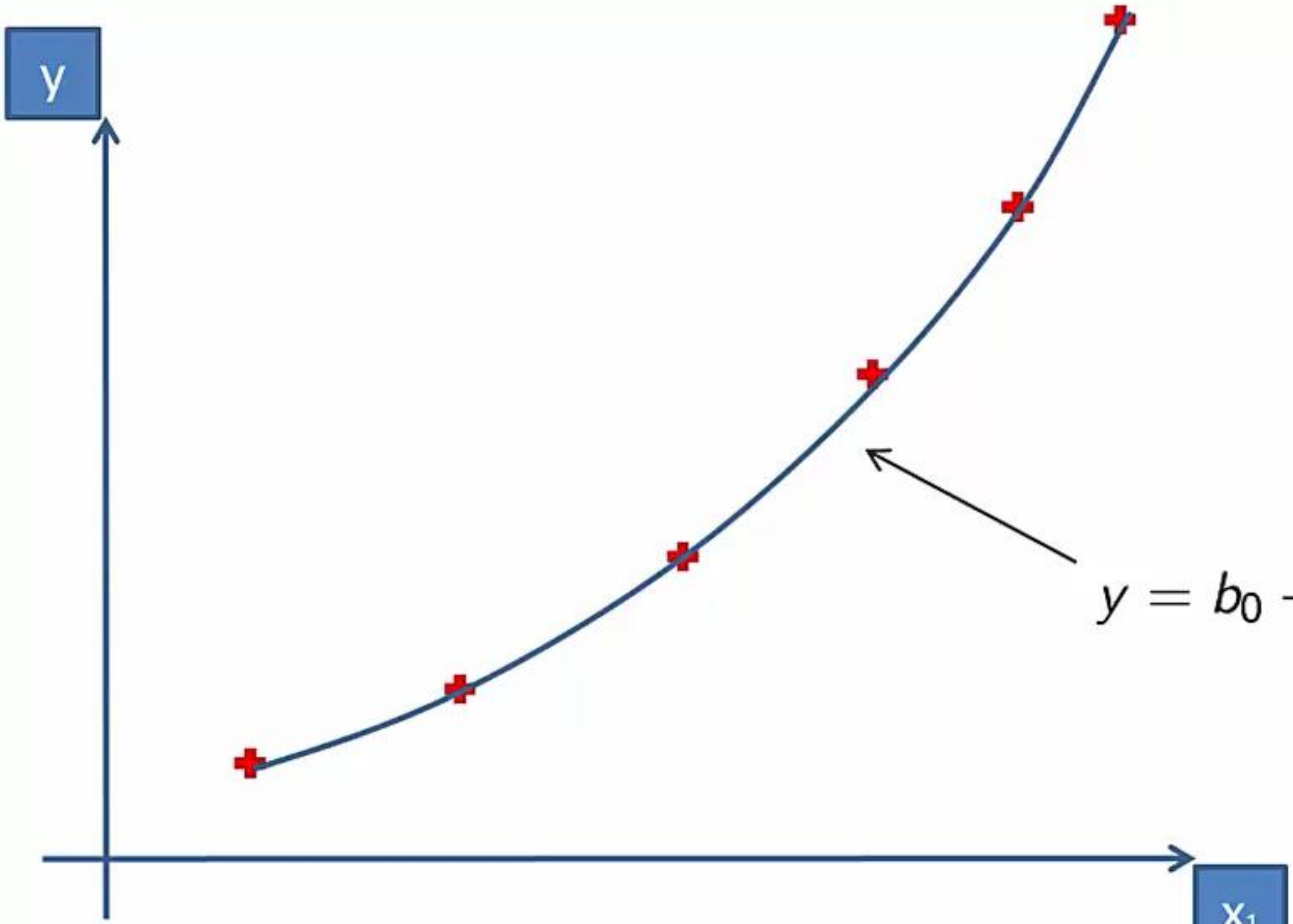
Permissions: RW End-of-lines: LF Encoding: UTF-8-GUESSED Line: 56 Column: 47 Memory: 38 %

Simple Linear Regression



$$y = b_0 + b_1 x_1$$

Polinomial Regression



$$y = b_0 + b_1 x_1 \boxed{+ b_2 x_1^2}$$