

Non Linear Regression Models

Machine Learning

Dr. Adnan Abid

Linear Regression Models Summary

Regressions

Simple
Linear
Regression

$$y = b_0 + b_1 x_1$$

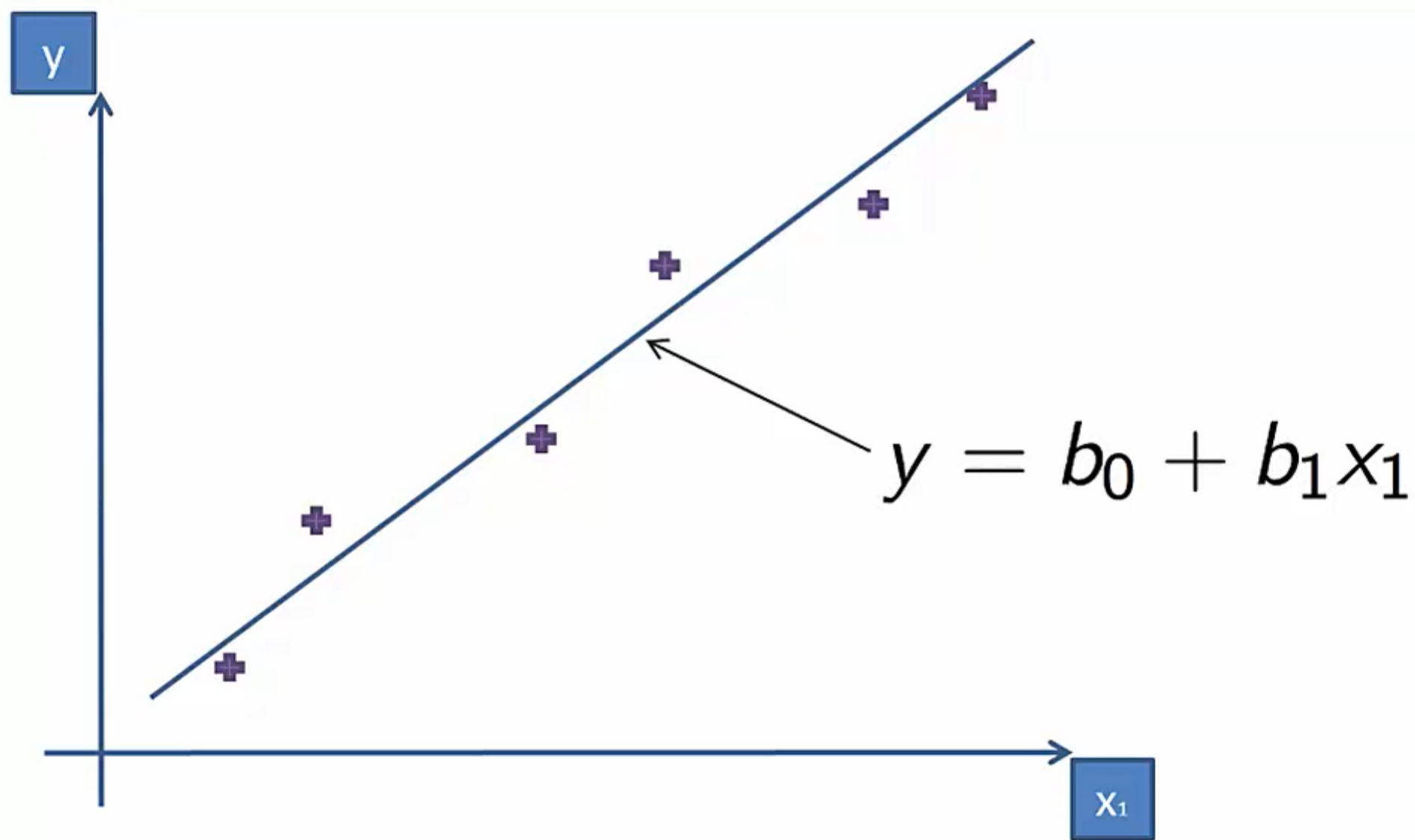
Multiple
Linear
Regression

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

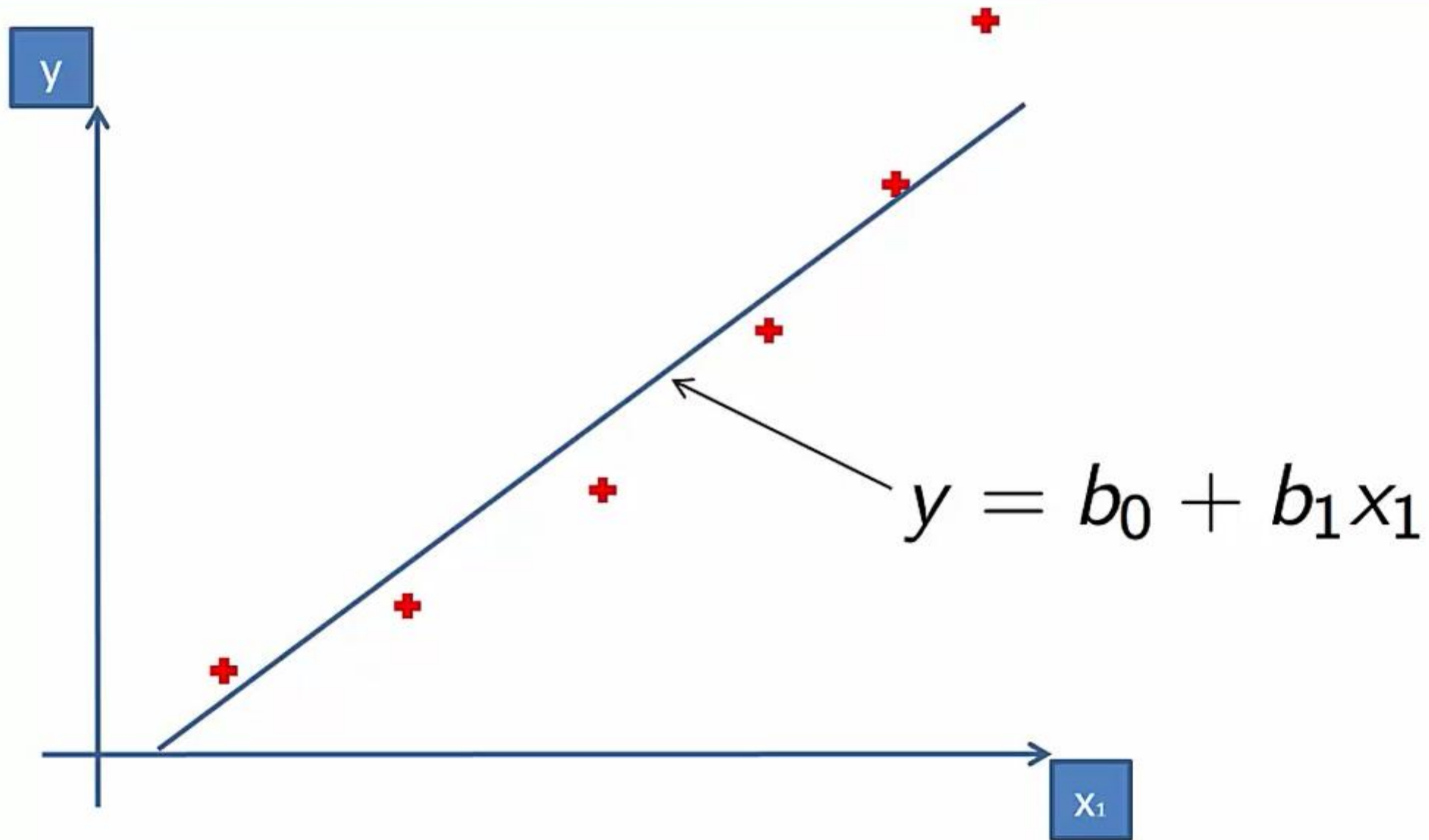
Polynomial
Linear
Regression

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

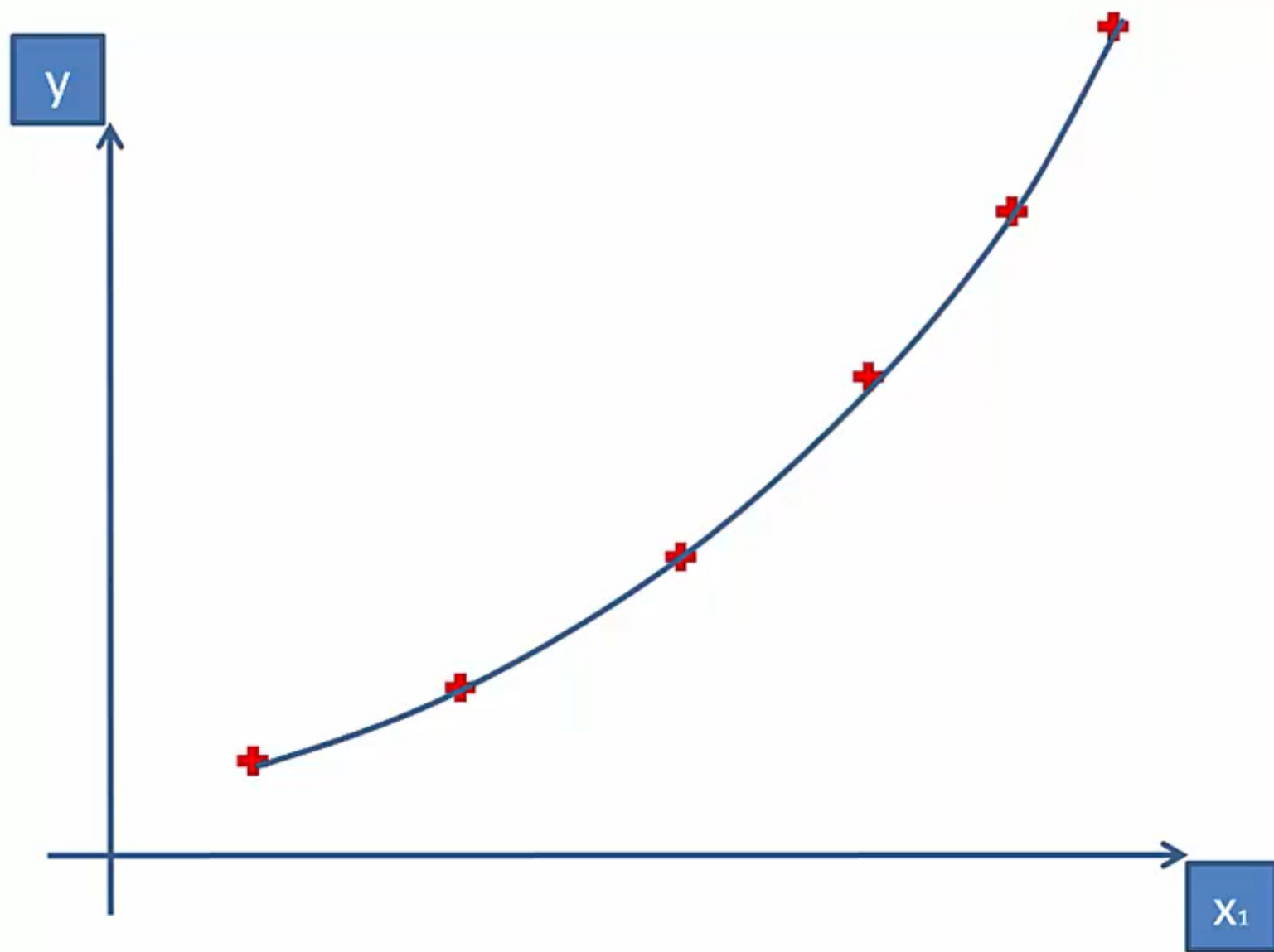
Simple Linear Regression



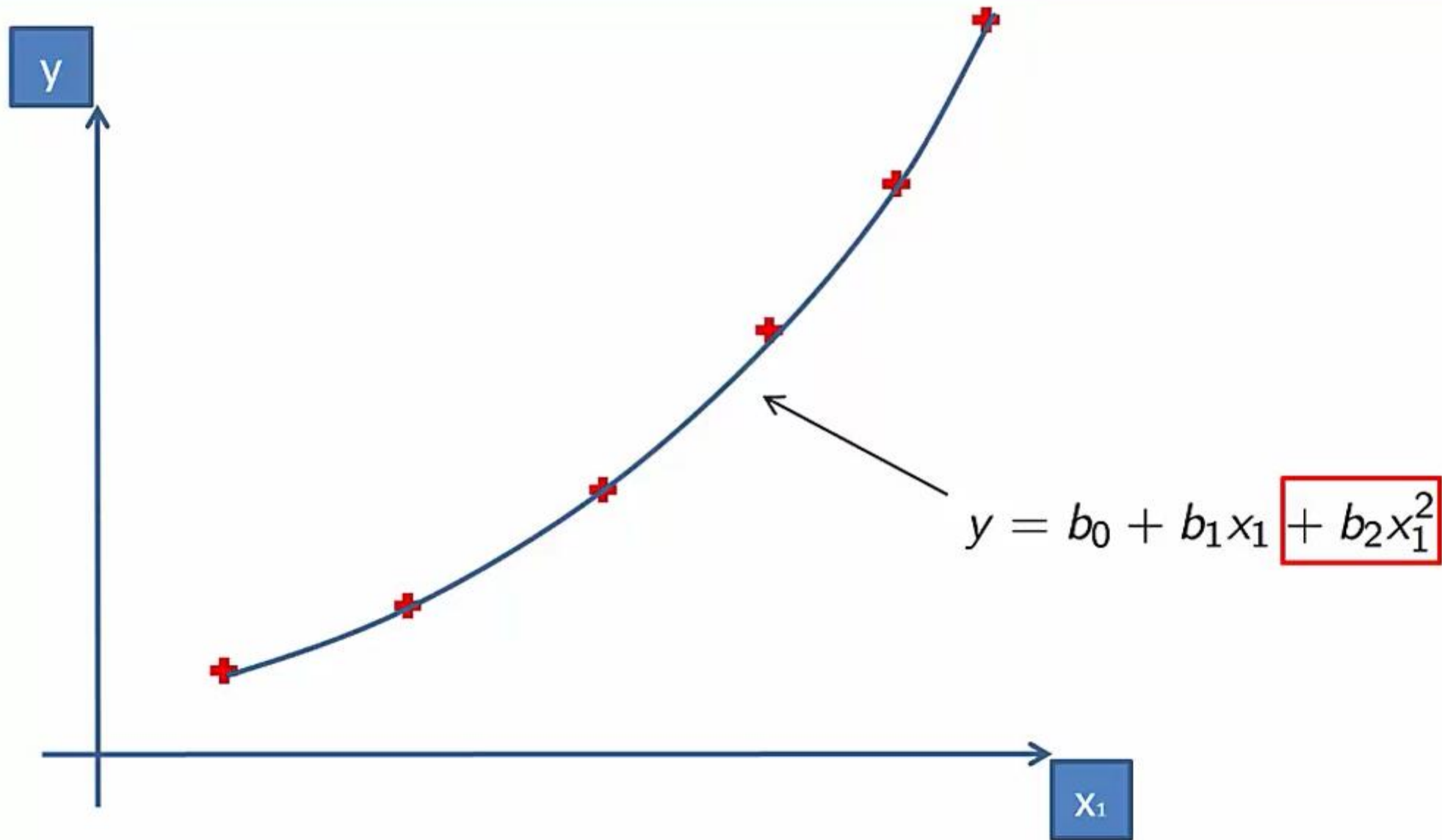
Simple Linear Regression



Polynomial Regression



Polynomial Regression



Polynomial Regression

Polynomial
Linear
Regression

$$y = b_0 + b_1x_1 + b_2x_1^2 + \dots + b_nx_1^n$$

Non Linear Regression Models

Support Vector Regression

Decision Tree Regression

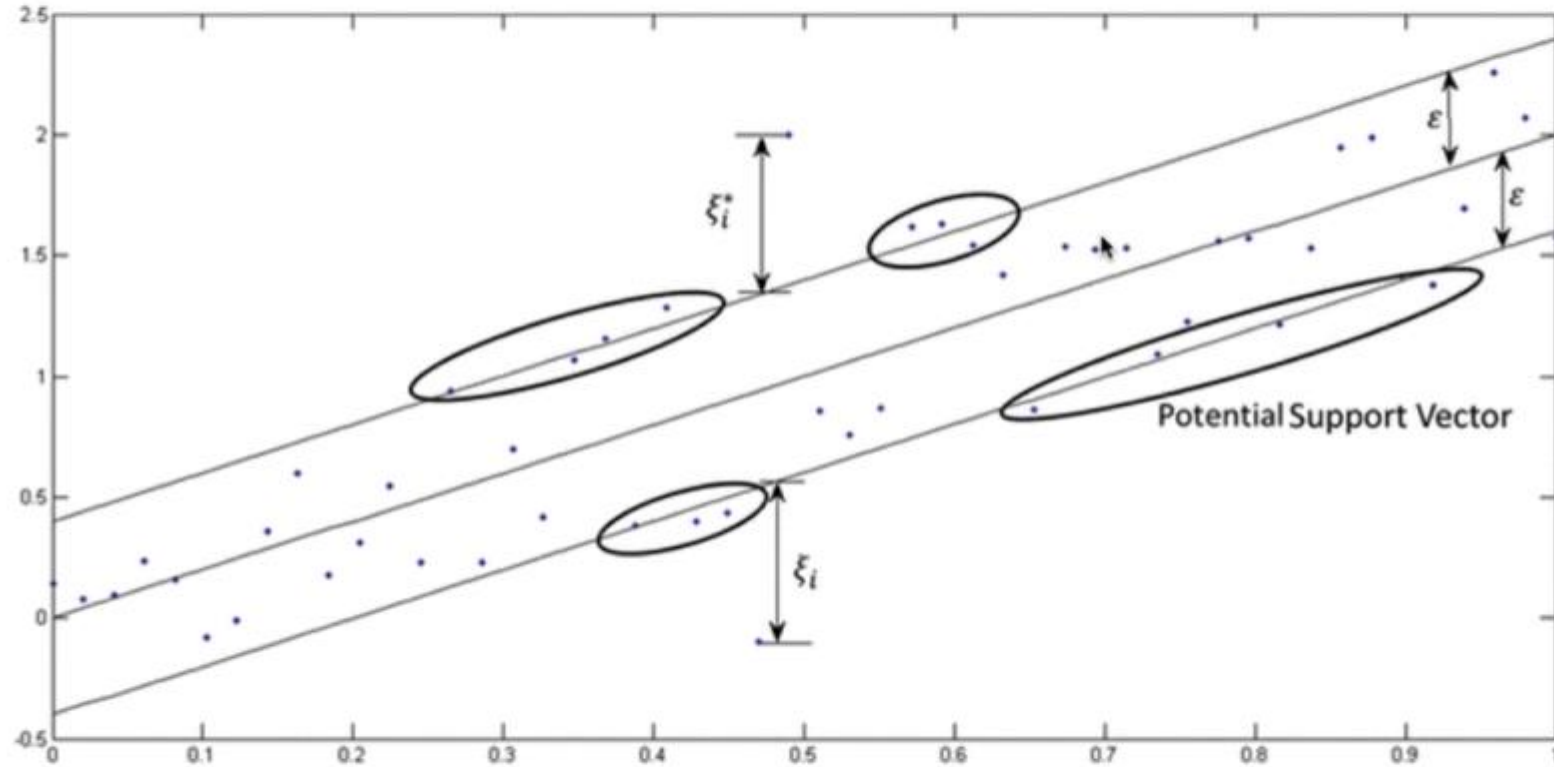
Random Forest Regression

Support Vector Regression

Support Vector Regression - SVR

- Support Vector Machines support linear and nonlinear regression that we can refer to as SVR
- Instead of trying to fit the largest possible street between two classes while limiting margin violations, SVR tries to fit as many instances as possible on the street while limiting margin violations.
- The width of the street is controlled by a hyper parameter Epsilon.

Support Vector Regression - SVR



https://link.springer.com/chapter/10.1007/978-1-4302-5990-9_4

```

1# SVR
2
3# Importing the libraries
4import numpy as np
5import matplotlib.pyplot as plt
6import pandas as pd
7
8# Importing the dataset
9dataset = pd.read_csv('Position_Salaries.csv')
10X = dataset.iloc[:, 1:2].values
11y = dataset.iloc[:, 2].values
12
13# Splitting the dataset into the Training set and Test set
14"""from sklearn.cross_validation import train_test_split
15X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 0)"""
16
17# Feature Scaling
18from sklearn.preprocessing import StandardScaler
19sc_X = StandardScaler()
20sc_y = StandardScaler()
21X = sc_X.fit_transform(X)
22y = sc_y.fit_transform(y)
23
24# Fitting SVR to the dataset
25from sklearn.svm import SVR
26regressor = SVR(kernel = 'rbf')
27regressor.fit(X, y)
28
29# Predicting a new result
30y_pred = sc_y.inverse_transform(regressor.predict(sc_X.transform(np.array([[6.5]]))))
31
32# Visualising the SVR results
33plt.scatter(X, y, color = 'red')
34plt.plot(X, regressor.predict(X), color = 'blue')
35plt.title('Truth or Bluff (SVR)')
36plt.xlabel('Position level')
37plt.ylabel('Salary')
38plt.show()

```

SVM library does not including feature scaling, so we have to explicitly scale features

After explicit scaling, we need to make sure that we scale the input value (6.5); and inverse scale the prediction to get actual/meaningful predicted value

For SVM Regression the predicted value is 170370

Name	Type	Size	Value
X	float64	(10, 1)	array([[-1.5666989], [-1.21854359]])
dataset	DataFrame	(10, 3)	Column names: Position, Level, Salary
y	float64	(10,)	array([-0.72004253, -0.70243757, -0.66722767, -0.59680786, -0.49117815, -0.35033854, -0.17428902, 0.17781001, 0.88200808, 2.64250325...])
y_pred	float64	(1,)	array([170370.0204065])

```

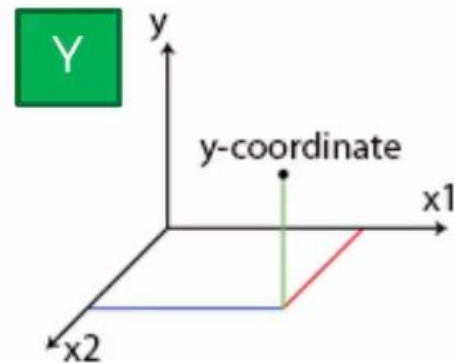
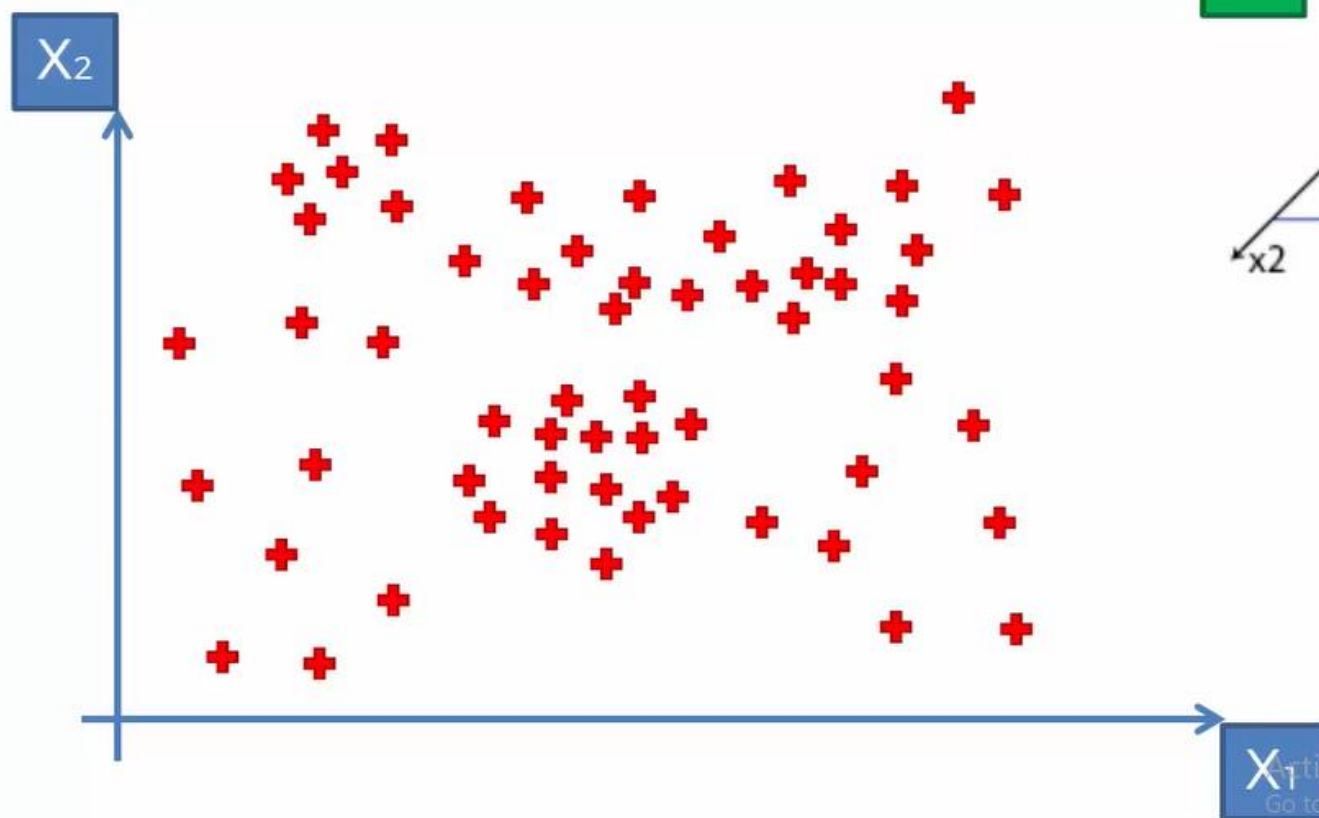
Out[4]:
SVR(C=1.0, cache_size=200, coef0=0.0, degree=3, epsilon=0.1,
gamma='auto',
kernel='rbf', max_iter=-1, shrinking=True, tol=0.001,
verbose=False)

In [5]: plt.scatter(X, y, color = 'red')
...: plt.plot(X, regressor.predict(X), color = 'blue')
...: plt.title('Truth or Bluff (SVR)')
...: plt.xlabel('Position level')
...: plt.ylabel('Salary')
...: plt.show()

```

Decision Tree Regression

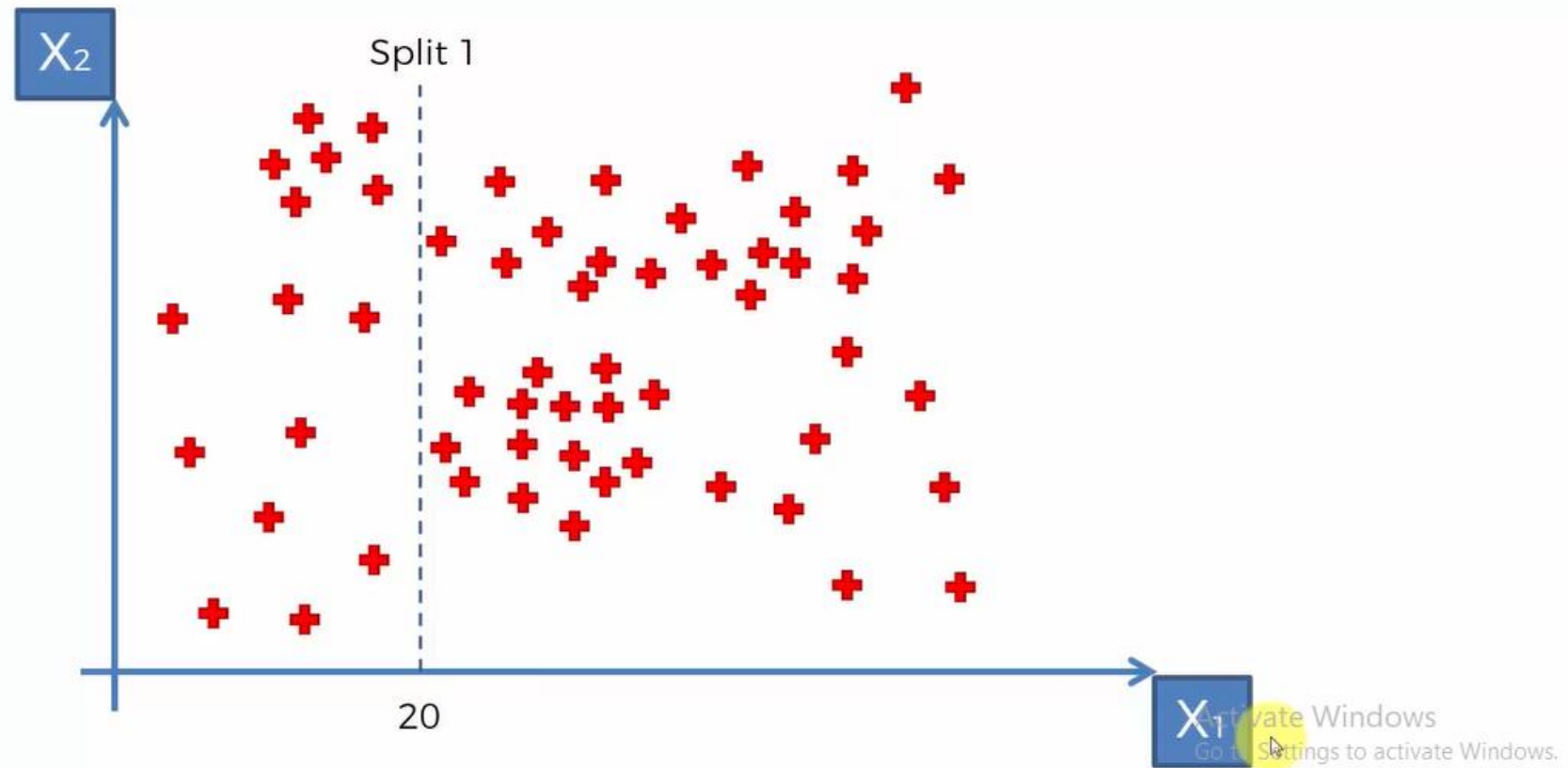
Decision Tree Intuition



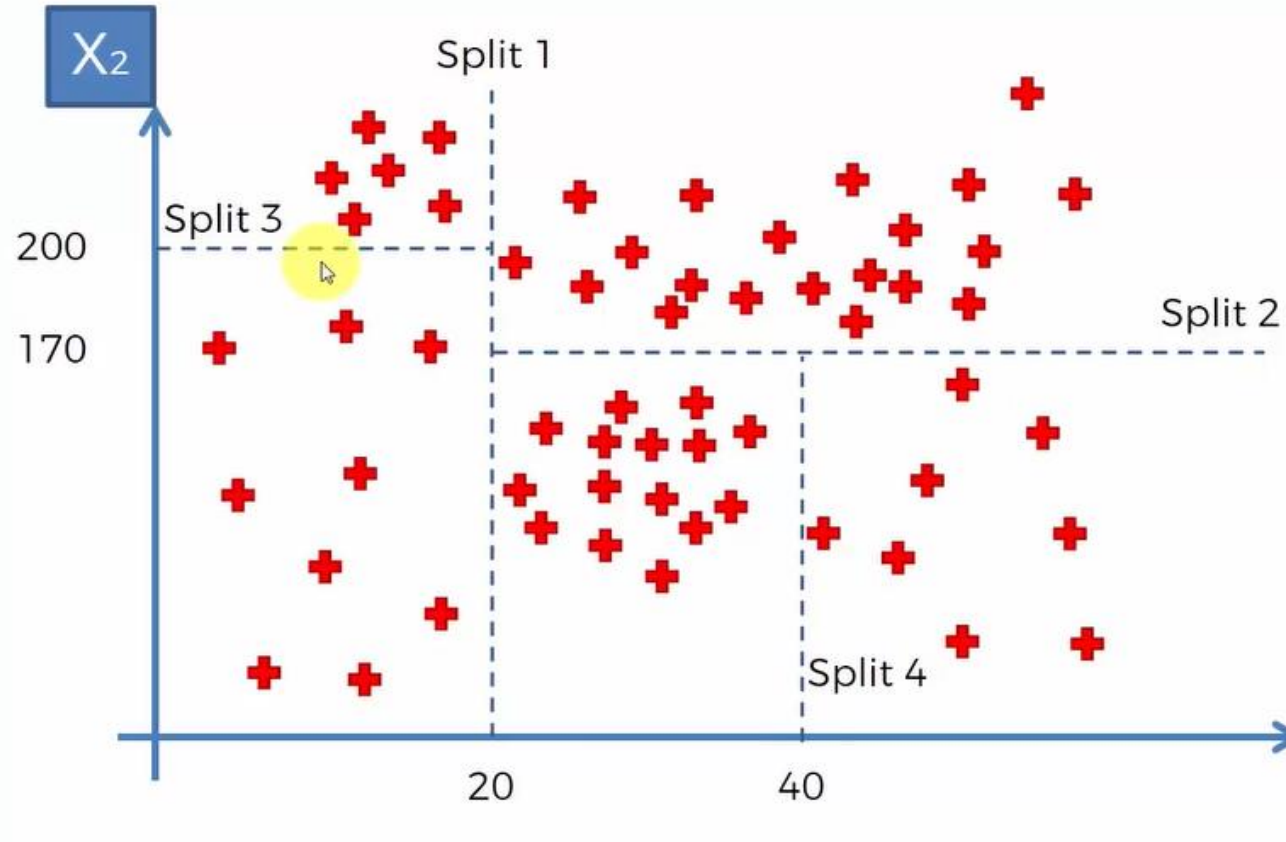
X_1

Activate Windows
Go to Settings to activate Windows.

Decision Tree Intuition



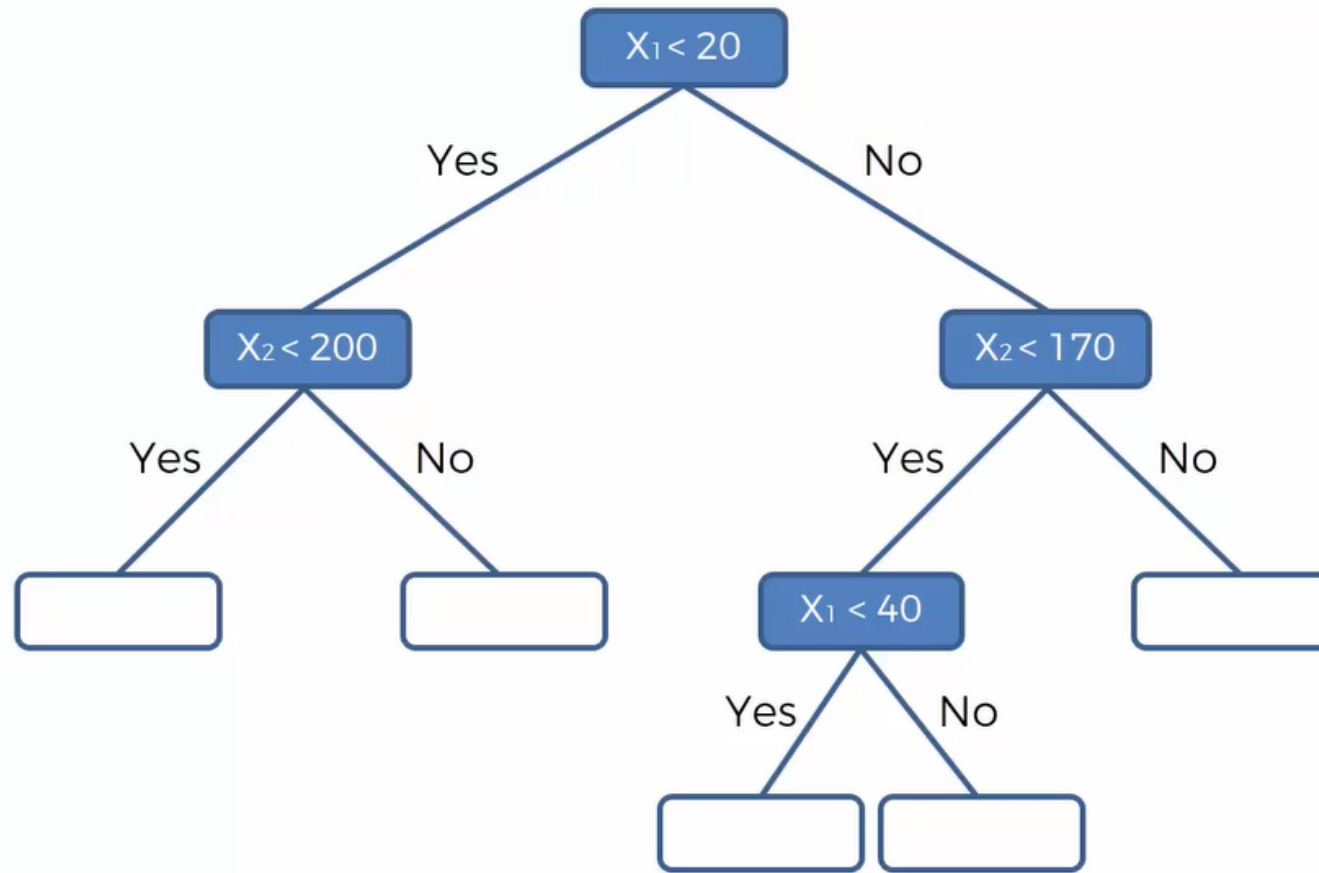
Decision Tree Intuition



X_1

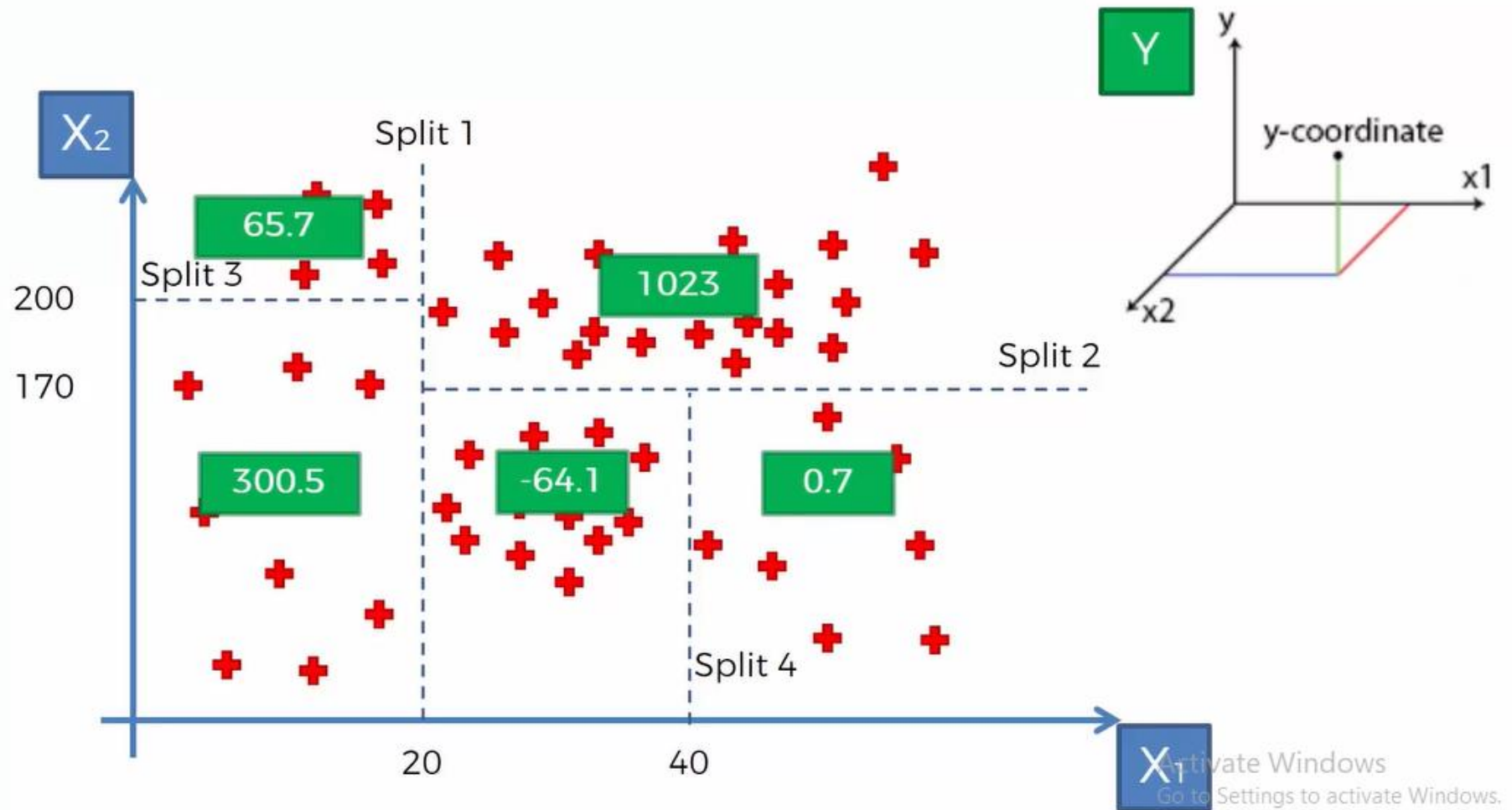
Activate Windows
Go to Settings to activate Windows.

Decision Tree Intuition

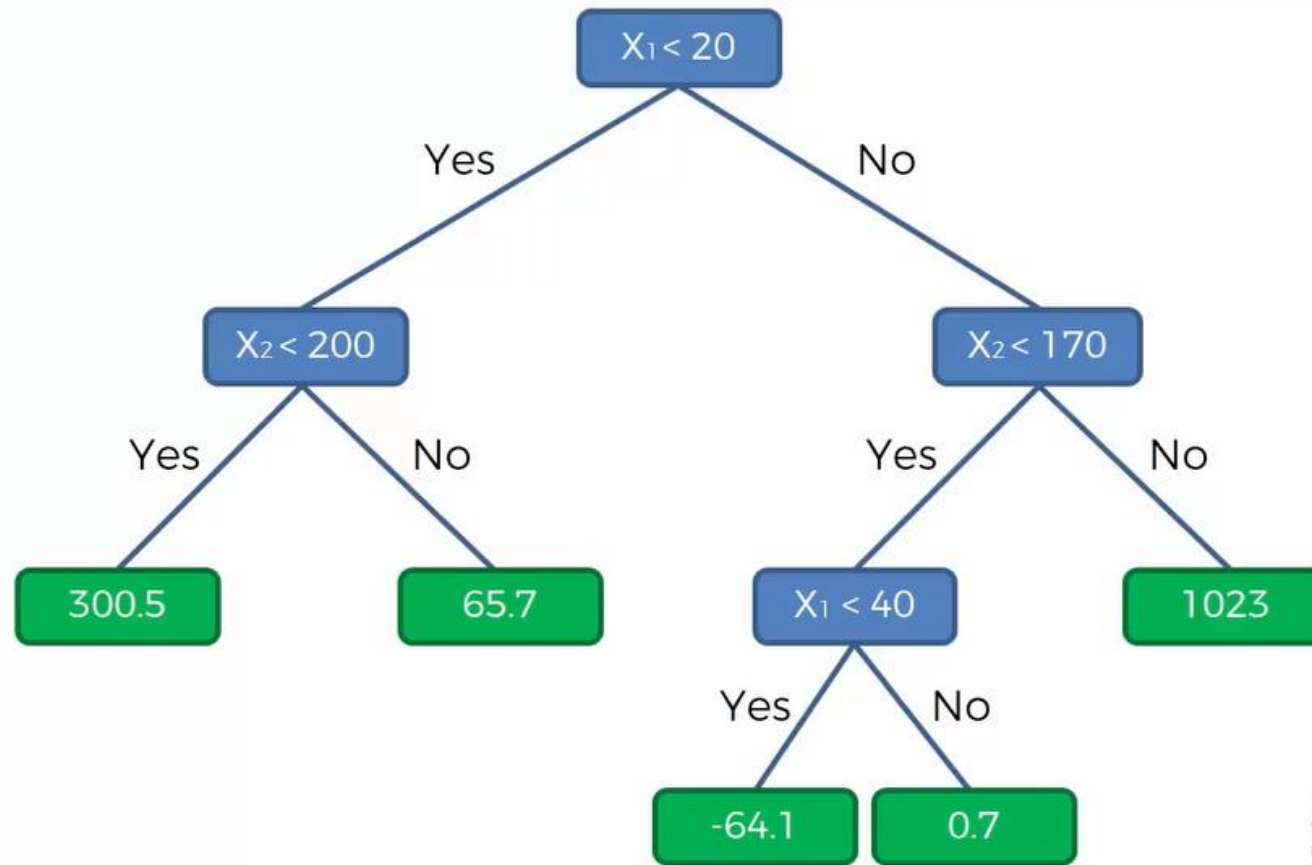


Activate Windows
Go to Settings to activate Windows.

Decision Tree Intuition



Decision Tree Intuition



Activate Windows
Go to Settings to activate Windows.

```

1 # Decision Tree Regression
2
3 # Importing the libraries
4 import numpy as np
5 import matplotlib.pyplot as plt
6 import pandas as pd
7
8 # Importing the dataset
9 dataset = pd.read_csv('Position_Salaries.csv')
10 X = dataset.iloc[:, 1:2].values
11 y = dataset.iloc[:, 2].values
12
13 # Splitting the dataset into the Training set and Test set
14 """from sklearn.cross_validation import train_test_split
15 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 0)"""
16
17 # Feature Scaling
18 """from sklearn.preprocessing import StandardScaler
19 sc_X = StandardScaler()
20 X_train = sc_X.fit_transform(X_train)
21 X_test = sc_X.transform(X_test)
22 sc_y = StandardScaler()
23 y_train = sc_y.fit_transform(y_train)"""
24
25 # Fitting the Decision Tree Regression to the dataset
26 from sklearn.tree import DecisionTreeRegressor
27 regressor = DecisionTreeRegressor(random_state = 0)
28 regressor.fit(X, y)
29
30 # Predicting a new result
31 y_pred = regressor.predict(6.5)
32
33 # Visualising the Decision Tree Regression results (for higher resolution and smoother curve)
34 X_grid = np.arange(min(X), max(X), 0.01)
35 X_grid = X_grid.reshape((len(X_grid), 1))
36 plt.scatter(X, y, color = 'red')
37 plt.plot(X_grid, regressor.predict(X_grid), color = 'blue')
38 plt.title('Truth or Bluff (Decision Tree Regression)')

```

X	int64	(10, 1)	array([[2],
X_grid	float64	(900, 1)	array([[1.],
dataset	DataFrame	(10, 3)	Column names: Position, Level, Sala
y	int64	(10,)	array([45000, 50000, 60000,
y_pred	float64	(1,)	array([150000.])

For Decision Tree the predicted v

Object inspector Variable explorer

IPython console

Console 1/A

```

...: plt.title('Truth or Bluff (Decision
...: plt.xlabel('Position level')
...: plt.ylabel('Salary')
...: plt.show()

In [8]: X_grid = np.arange(min(X), max(X),
...: X_grid = X_grid.reshape((len(X_grid)
...: plt.scatter(X, y, color = 'red')
...: plt.plot(X_grid, regressor.predict(
...: plt.title('Truth or Bluff (Decision
...: plt.xlabel('Position level')
...: plt.ylabel('Salary')
...: plt.show()

```

Random Forest Regression

Random Forest Intuition

Ensemble Learning

In ensemble learning we build many models (different or same) and then make a collective decision in the end.

For example we use Simple Linear Regression, SVR, Random Forest Regressor and take the average of all predictions to build an ensemble learning model.

Activate Windows
Go to Settings to activate Windows.

Alternatively, we may use same model, e.g. many decision tree models built on different subsets of data points, and take the average of all models.

Random Forest Intuition

STEP 1: Pick at random K data points from the Training set.



STEP 2: Build the Decision Tree associated to these K data points.



STEP 3: Choose the number N_{tree} of trees you want to build and repeat STEPS 1 & 2



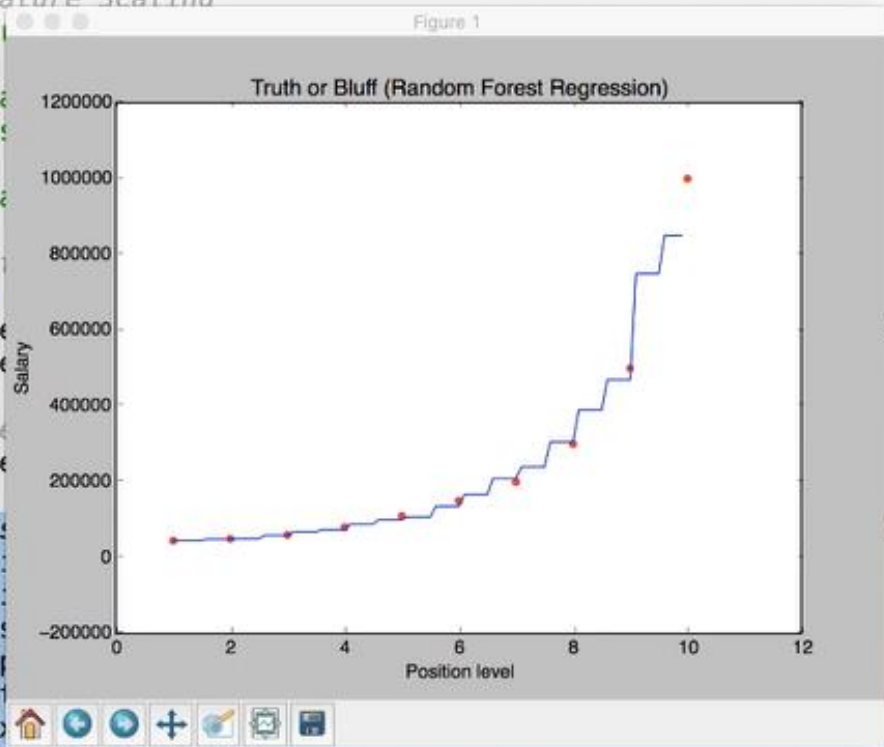
STEP 4: For a new data point, make each one of your N_{tree} trees predict the value of Y to for the data point in question, and assign the new data point the average across all of the predicted Y values.


```
1# Random Forest Regression
2
3# Importing the libraries
4import numpy as np
5import matplotlib.pyplot as plt
6import pandas as pd
7
8# Importing the dataset
9dataset = pd.read_csv('Position_Salaries.csv')
10X = dataset.iloc[:, 1:2].values
11y = dataset.iloc[:, 2].values
12
13# Splitting the dataset into the Training set and Test set
14"""from sklearn.cross_validation import train_test_split
15X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 0)"""
16
17# Feature Scaling
18"""from sklearn.preprocessing import StandardScaler
19sc_X = StandardScaler()
20X_train = sc_X.fit_transform(X_train)
21X_test = sc_X.transform(X_test)
22sc_y = StandardScaler()
23y_train = sc_y.fit_transform(y_train)"""
24
25# Fitting Random Forest Regression to the dataset
26from sklearn.ensemble import RandomForestRegressor
27regressor = RandomForestRegressor(n_estimators = 10, random_state = 0)
28regressor.fit(X, y)
29
30# Predicting a new result
31y_pred = regressor.predict(6.5)
32
33# Visualising the Regression results (for higher resolution and smoother curve)
34X_grid = np.arange(min(X), max(X), 0.1)
35X_grid = X_grid.reshape((len(X_grid), 1))
36plt.scatter(X, y, color = 'red')
37plt.plot(X_grid, regressor.predict(X_grid), color = 'blue')
38plt.title('Truth or Bluff (Random Forest Regression)')
39plt.xlabel('Position level')
40plt.ylabel('Salary')
41plt.show()
```

```

1# Random Forest Regression
2
3# Importing the libraries
4import numpy as np
5import matplotlib.pyplot as plt
6import pandas as pd
7
8# Importing the dataset
9dataset = pd.read_csv('Position_Salaries.csv')
10X = dataset.iloc[:, 1:2].values
11y = dataset.iloc[:, 2].values
12
13# Splitting the dataset into the Training set and Test set
14"""from sklearn.cross_validation import train_test_split
15X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 0)"""
16
17# Feature Scaling
18"""from sklearn.preprocessing import StandardScaler
19sc_X = StandardScaler()
20X_train = sc_X.fit_transform(X_train)
21X_test = sc_X.transform(X_test)
22sc_y = StandardScaler()
23y_train = sc_y.fit_transform(y_train)
24
25# Fitting the Random Forest Regression model to the dataset
26from sklearn.ensemble import RandomForestRegressor
27regressor = RandomForestRegressor(n_estimators = 10, random_state = 0)
28regressor.fit(X_train, y_train)
29
30# Predicting the salary of a candidate with 6 years of experience
31y_pred = regressor.predict(X_test)
32
33# Visualising the Random Forest Regression results
34X_grid = np.arange(min(X_train), max(X_train), 0.01)
35X_grid = X_grid.reshape((len(X_grid), 1))
36plt.scatter(X_test, y_test)
37plt.plot(X_grid, regressor.predict(X_grid))
38plt.xlabel('Position level')
39plt.ylabel('Salary')
40plt.show()

```



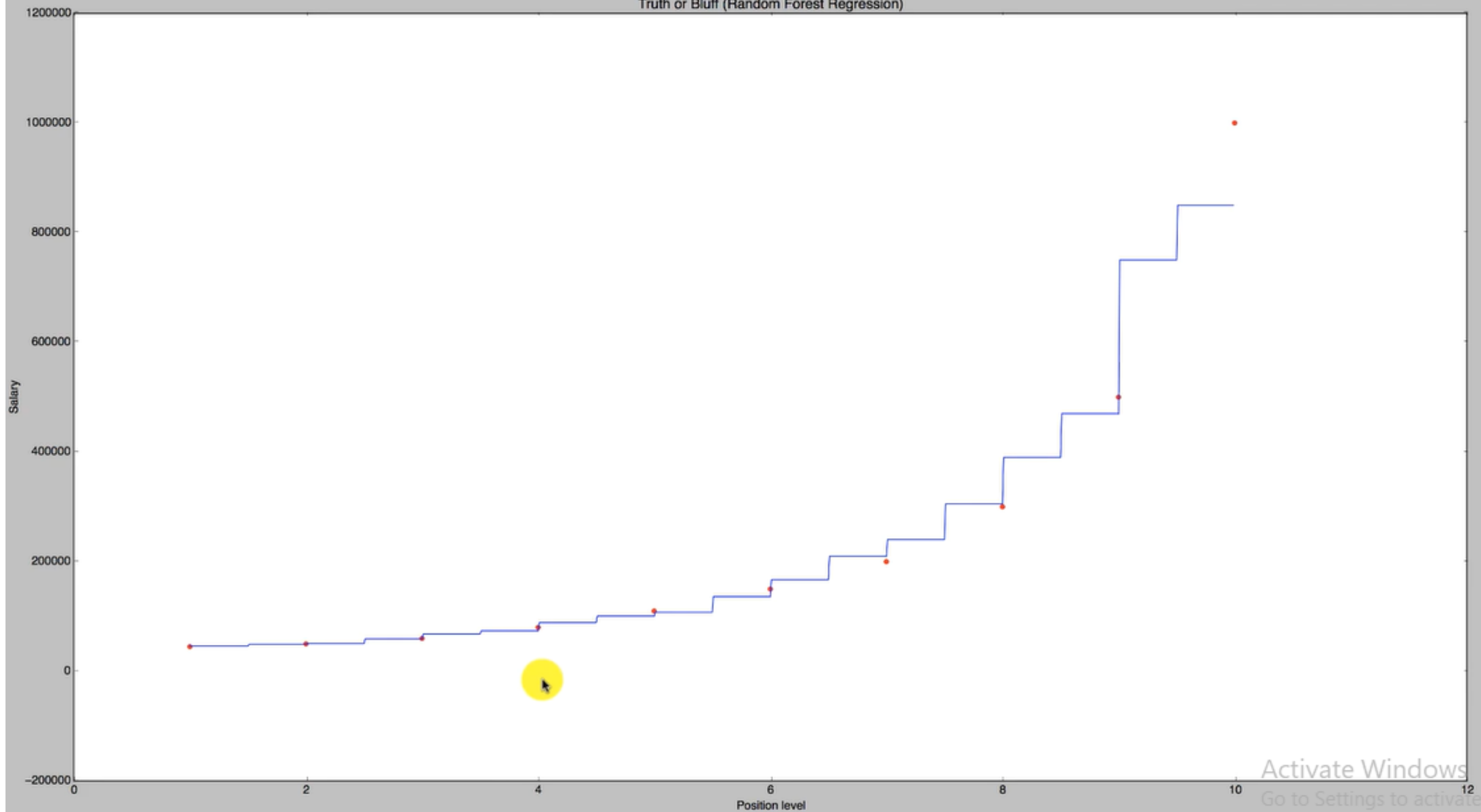
random_state = 0)

I

on and smoother curve)

)

Truth or Bluff (Random Forest Regression)



```

1 # Decision Tree Regression
2
3 # Importing the libraries
4 import numpy as np
5 import matplotlib.pyplot as plt
6 import pandas as pd
7
8 # Importing the dataset
9 dataset = pd.read_csv('Position_Salaries.csv')
10 X = dataset.iloc[:, 1:2].values
11 y = dataset.iloc[:, 2].values
12
13 # Splitting the dataset into the Training set and Test set
14 """from sklearn.cross_validation import train_test_split
15 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 0)"""
16
17 # Feature Scaling
18 """from sklearn.preprocessing import StandardScaler
19 sc_X = StandardScaler()
20 X_train = sc_X.fit_transform(X_train)
21 X_test = sc_X.transform(X_test)
22 sc_y = StandardScaler()
23 y_train = sc_y.fit_transform(y_train)"""
24
25 # Fitting the Decision Tree Regression to the dataset
26 from sklearn.tree import DecisionTreeRegressor
27 regressor = DecisionTreeRegressor(random_state = 0)
28 regressor.fit(X, y)
29
30 # Predicting a new result
31 y_pred = regressor.predict(6.5)
32
33 # Visualising the Decision Tree Regression results (for higher resolution and smoother curve)
34 X_grid = np.arange(min(X), max(X), 0.01)
35 X_grid = X_grid.reshape((len(X_grid), 1))
36 plt.scatter(X, y, color = 'red')
37 plt.plot(X_grid, regressor.predict(X_grid), color = 'blue')
38 plt.title('Truth or Bluff (Decision Tree Regression)')

```

X	int64	(10, 1)	array([[2],
X_grid	float64	(900, 1)	array([[1.],
dataset	DataFrame	(10, 3)	Column names: Position, Level, Salary
y	int64	(10,)	array([45000, 50000, 60000, 80000, 110000, 150000, 200000,
y_pred	float64	(1,)	array([150000.])

For Decision Tree i.e. Random Forest with 1 tree the predicted value is 150000

Object inspector Variable explorer File explorer

IPython console

Console 1/A

```

...: plt.title('Truth or Bluff (Decision Tree Regression)')
...: plt.xlabel('Position level')
...: plt.ylabel('Salary')
...: plt.show()

In [8]: X_grid = np.arange(min(X), max(X), 0.01)
...: X_grid = X_grid.reshape((len(X_grid), 1))
...: plt.scatter(X, y, color = 'red')
...: plt.plot(X_grid, regressor.predict(X_grid), color = 'blue')
...: plt.title('Truth or Bluff (Decision Tree Regression)')
...: plt.xlabel('Position level')
...: plt.ylabel('Salary')
...: plt.show()

```



```
regression_template.py random_forest_regression.py*
1# Random Forest Regression
2
3# Importing the libraries
4import numpy as np
5import matplotlib.pyplot as plt
6import pandas as pd
7
8# Importing the dataset
9dataset = pd.read_csv('Position_Salaries.csv')
10X = dataset.iloc[:, 1:2].values
11y = dataset.iloc[:, 2].values
12
13# Splitting the dataset into the Training set and Test set
14"""from sklearn.cross_validation import train_test_split
15X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 0)"""
16
17# Feature Scaling
18"""from sklearn.preprocessing import StandardScaler
19sc_X = StandardScaler()
20X_train = sc_X.fit_transform(X_train)
21X_test = sc_X.transform(X_test)
22sc_y = StandardScaler()
23y_train = sc_y.fit_transform(y_train)"""
24
25# Fitting Random Forest Regression to the dataset
26from sklearn.ensemble import RandomForestRegressor
27regressor = RandomForestRegressor(n_estimators = 10, random_state = 0)
28regressor.fit(X, y)
29
30# Predicting a new result
31y_pred = regressor.predict(6.5)
32
33# Visualising the Regression results (for higher resolution and smoother curve)
34X_grid = np.arange(min(X), max(X), 0.01)
35X_grid = X_grid.reshape((len(X_grid), 1))
36plt.scatter(X, y, color = 'red')
37plt.plot(X_grid, regressor.predict(X_grid), color = 'blue')
38plt.title('Truth or Bluff (Random Forest Regression)')
39plt.xlabel('Position level')
40plt.ylabel('Salary')
41plt.show()
```

Name	Type	Size	Value
X	int64	(10, 1)	array([[1], [2],
X_grid	float64	(900, 1)	array([[1.], [1.01],
dataset	DataFrame	(10, 3)	Column names: Position, Level, Salary
y	int64	(10,)	array([45000, 50000, 60000, 80000, 110000, 150000, 200000, 300000, 500000, 1000000])
y_pred	float64	(1,)	array([167000.])

For 10 trees the predicted value is 167000

```
Object inspector Variable explorer File explorer
IPython console
Console 1/A
...: plt.title('Truth or Bluff (Random Forest Regression)')
...: plt.xlabel('Position level')
...: plt.ylabel('Salary')
...: plt.show()

In [5]: X_grid = np.arange(min(X), max(X), 0.01)
...: X_grid = X_grid.reshape((len(X_grid), 1))
...: plt.scatter(X, y, color = 'red')
...: plt.plot(X_grid, regressor.predict(X_grid), color = 'blue')
...: plt.title('Truth or Bluff (Random Forest Regression)')
...: plt.xlabel('Position level')
...: plt.ylabel('Salary')
...: plt.show()

In [6]: y_pred = regressor.predict(6.5)

In [7]:
```

```

1 # Random Forest Regression
2
3 # Importing the libraries
4 import numpy as np
5 import matplotlib.pyplot as plt
6 import pandas as pd
7
8 # Importing the dataset
9 dataset = pd.read_csv('Position_Salaries.csv')
10 X = dataset.iloc[:, 1:2].values
11 y = dataset.iloc[:, 2].values
12
13 # Splitting the dataset into the Training set and Test set
14 """from sklearn.cross_validation import train_test_split
15 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 0)"""
16
17 # Feature Scaling
18 """from sklearn.preprocessing import StandardScaler
19 sc_X = StandardScaler()
20 X_train = sc_X.fit_transform(X_train)
21 X_test = sc_X.transform(X_test)
22 sc_y = StandardScaler()
23 y_train = sc_y.fit_transform(y_train)"""
24
25 # Fitting Random Forest Regression to the dataset
26 from sklearn.ensemble import RandomForestRegressor
27 regressor = RandomForestRegressor(n_estimators = 100, random_state = 0)
28 regressor.fit(X, y)
29
30 # Predicting a new result
31 y_pred = regressor.predict(6.5)
32
33 # Visualising the Regression results (for higher resolution and smoother curve)
34 X_grid = np.arange(min(X), max(X), 0.01)
35 X_grid = X_grid.reshape((len(X_grid), 1))
36 plt.scatter(X, y, color = 'red')
37 plt.plot(X_grid, regressor.predict(X_grid), color = 'blue')
38 plt.title('Truth or Bluff (Random Forest Regression)')
39 plt.xlabel('Position level')
40 plt.ylabel('Salary')
41 plt.show()

```

Name	Type	Size	Value
X	int64	(10, 1)	array([[1], [2], [3], [4], [5], [6], [7], [8], [9], [10]])
X_grid	float64	(900, 1)	array([[1.], [1.01], [1.02], [1.03], [1.04], [1.05], [1.06], [1.07], [1.08], [1.09], [1.1], [1.11], [1.12], [1.13], [1.14], [1.15], [1.16], [1.17], [1.18], [1.19], [1.2], [1.21], [1.22], [1.23], [1.24], [1.25], [1.26], [1.27], [1.28], [1.29], [1.3], [1.31], [1.32], [1.33], [1.34], [1.35], [1.36], [1.37], [1.38], [1.39], [1.4], [1.41], [1.42], [1.43], [1.44], [1.45], [1.46], [1.47], [1.48], [1.49], [1.5], [1.51], [1.52], [1.53], [1.54], [1.55], [1.56], [1.57], [1.58], [1.59], [1.6], [1.61], [1.62], [1.63], [1.64], [1.65], [1.66], [1.67], [1.68], [1.69], [1.7], [1.71], [1.72], [1.73], [1.74], [1.75], [1.76], [1.77], [1.78], [1.79], [1.8], [1.81], [1.82], [1.83], [1.84], [1.85], [1.86], [1.87], [1.88], [1.89], [1.9], [1.91], [1.92], [1.93], [1.94], [1.95], [1.96], [1.97], [1.98], [1.99], [2.0], [2.01], [2.02], [2.03], [2.04], [2.05], [2.06], [2.07], [2.08], [2.09], [2.1], [2.11], [2.12], [2.13], [2.14], [2.15], [2.16], [2.17], [2.18], [2.19], [2.2], [2.21], [2.22], [2.23], [2.24], [2.25], [2.26], [2.27], [2.28], [2.29], [2.3], [2.31], [2.32], [2.33], [2.34], [2.35], [2.36], [2.37], [2.38], [2.39], [2.4], [2.41], [2.42], [2.43], [2.44], [2.45], [2.46], [2.47], [2.48], [2.49], [2.5], [2.51], [2.52], [2.53], [2.54], [2.55], [2.56], [2.57], [2.58], [2.59], [2.6], [2.61], [2.62], [2.63], [2.64], [2.65], [2.66], [2.67], [2.68], [2.69], [2.7], [2.71], [2.72], [2.73], [2.74], [2.75], [2.76], [2.77], [2.78], [2.79], [2.8], [2.81], [2.82], [2.83], [2.84], [2.85], [2.86], [2.87], [2.88], [2.89], [2.9], [2.91], [2.92], [2.93], [2.94], [2.95], [2.96], [2.97], [2.98], [2.99], [3.0], [3.01], [3.02], [3.03], [3.04], [3.05], [3.06], [3.07], [3.08], [3.09], [3.1], [3.11], [3.12], [3.13], [3.14], [3.15], [3.16], [3.17], [3.18], [3.19], [3.2], [3.21], [3.22], [3.23], [3.24], [3.25], [3.26], [3.27], [3.28], [3.29], [3.3], [3.31], [3.32], [3.33], [3.34], [3.35], [3.36], [3.37], [3.38], [3.39], [3.4], [3.41], [3.42], [3.43], [3.44], [3.45], [3.46], [3.47], [3.48], [3.49], [3.5], [3.51], [3.52], [3.53], [3.54], [3.55], [3.56], [3.57], [3.58], [3.59], [3.6], [3.61], [3.62], [3.63], [3.64], [3.65], [3.66], [3.67], [3.68], [3.69], [3.7], [3.71], [3.72], [3.73], [3.74], [3.75], [3.76], [3.77], [3.78], [3.79], [3.8], [3.81], [3.82], [3.83], [3.84], [3.85], [3.86], [3.87], [3.88], [3.89], [3.9], [3.91], [3.92], [3.93], [3.94], [3.95], [3.96], [3.97], [3.98], [3.99], [4.0], [4.01], [4.02], [4.03], [4.04], [4.05], [4.06], [4.07], [4.08], [4.09], [4.1], [4.11], [4.12], [4.13], [4.14], [4.15], [4.16], [4.17], [4.18], [4.19], [4.2], [4.21], [4.22], [4.23], [4.24], [4.25], [4.26], [4.27], [4.28], [4.29], [4.3], [4.31], [4.32], [4.33], [4.34], [4.35], [4.36], [4.37], [4.38], [4.39], [4.4], [4.41], [4.42], [4.43], [4.44], [4.45], [4.46], [4.47], [4.48], [4.49], [4.5], [4.51], [4.52], [4.53], [4.54], [4.55], [4.56], [4.57], [4.58], [4.59], [4.6], [4.61], [4.62], [4.63], [4.64], [4.65], [4.66], [4.67], [4.68], [4.69], [4.7], [4.71], [4.72], [4.73], [4.74], [4.75], [4.76], [4.77], [4.78], [4.79], [4.8], [4.81], [4.82], [4.83], [4.84], [4.85], [4.86], [4.87], [4.88], [4.89], [4.9], [4.91], [4.92], [4.93], [4.94], [4.95], [4.96], [4.97], [4.98], [4.99], [5.0], [5.01], [5.02], [5.03], [5.04], [5.05], [5.06], [5.07], [5.08], [5.09], [5.1], [5.11], [5.12], [5.13], [5.14], [5.15], [5.16], [5.17], [5.18], [5.19], [5.2], [5.21], [5.22], [5.23], [5.24], [5.25], [5.26], [5.27], [5.28], [5.29], [5.3], [5.31], [5.32], [5.33], [5.34], [5.35], [5.36], [5.37], [5.38], [5.39], [5.4], [5.41], [5.42], [5.43], [5.44], [5.45], [5.46], [5.47], [5.48], [5.49], [5.5], [5.51], [5.52], [5.53], [5.54], [5.55], [5.56], [5.57], [5.58], [5.59], [5.6], [5.61], [5.62], [5.63], [5.64], [5.65], [5.66], [5.67], [5.68], [5.69], [5.7], [5.71], [5.72], [5.73], [5.74], [5.75], [5.76], [5.77], [5.78], [5.79], [5.8], [5.81], [5.82], [5.83], [5.84], [5.85], [5.86], [5.87], [5.88], [5.89], [5.9], [5.91], [5.92], [5.93], [5.94], [5.95], [5.96], [5.97], [5.98], [5.99], [6.0], [6.01], [6.02], [6.03], [6.04], [6.05], [6.06], [6.07], [6.08], [6.09], [6.1], [6.11], [6.12], [6.13], [6.14], [6.15], [6.16], [6.17], [6.18], [6.19], [6.2], [6.21], [6.22], [6.23], [6.24], [6.25], [6.26], [6.27], [6.28], [6.29], [6.3], [6.31], [6.32], [6.33], [6.34], [6.35], [6.36], [6.37], [6.38], [6.39], [6.4], [6.41], [6.42], [6.43], [6.44], [6.45], [6.46], [6.47], [6.48], [6.49], [6.5], [6.51], [6.52], [6.53], [6.54], [6.55], [6.56], [6.57], [6.58], [6.59], [6.6], [6.61], [6.62], [6.63], [6.64], [6.65], [6.66], [6.67], [6.68], [6.69], [6.7], [6.71], [6.72], [6.73], [6.74], [6.75], [6.76], [6.77], [6.78], [6.79], [6.8], [6.81], [6.82], [6.83], [6.84], [6.85], [6.86], [6.87], [6.88], [6.89], [6.9], [6.91], [6.92], [6.93], [6.94], [6.95], [6.96], [6.97], [6.98], [6.99], [7.0], [7.01], [7.02], [7.03], [7.04], [7.05], [7.06], [7.07], [7.08], [7.09], [7.1], [7.11], [7.12], [7.13], [7.14], [7.15], [7.16], [7.17], [7.18], [7.19], [7.2], [7.21], [7.22], [7.23], [7.24], [7.25], [7.26], [7.27], [7.28], [7.29], [7.3], [7.31], [7.32], [7.33], [7.34], [7.35], [7.36], [7.37], [7.38], [7.39], [7.4], [7.41], [7.42], [7.43], [7.44], [7.45], [7.46], [7.47], [7.48], [7.49], [7.5], [7.51], [7.52], [7.53], [7.54], [7.55], [7.56], [7.57], [7.58], [7.59], [7.6], [7.61], [7.62], [7.63], [7.64], [7.65], [7.66], [7.67], [7.68], [7.69], [7.7], [7.71], [7.72], [7.73], [7.74], [7.75], [7.76], [7.77], [7.78], [7.79], [7.8], [7.81], [7.82], [7.83], [7.84], [7.85], [7.86], [7.87], [7.88], [7.89], [7.9], [7.91], [7.92], [7.93], [7.94], [7.95], [7.96], [7.97], [7.98], [7.99], [8.0], [8.01], [8.02], [8.03], [8.04], [8.05], [8.06], [8.07], [8.08], [8.09], [8.1], [8.11], [8.12], [8.13], [8.14], [8.15], [8.16], [8.17], [8.18], [8.19], [8.2], [8.21], [8.22], [8.23], [8.24], [8.25], [8.26], [8.27], [8.28], [8.29], [8.3], [8.31], [8.32], [8.33], [8.34], [8.35], [8.36], [8.37], [8.38], [8.39], [8.4], [8.41], [8.42], [8.43], [8.44], [8.45], [8.46], [8.47], [8.48], [8.49], [8.5], [8.51], [8.52], [8.53], [8.54], [8.55], [8.56], [8.57], [8.58], [8.59], [8.6], [8.61], [8.62], [8.63], [8.64], [8.65], [8.66], [8.67], [8.68], [8.69], [8.7], [8.71], [8.72], [8.73], [8.74], [8.75], [8.76], [8.77], [8.78], [8.79], [8.8], [8.81], [8.82], [8.83], [8.84], [8.85], [8.86], [8.87], [8.88], [8.89], [8.9], [8.91], [8.92], [8.93], [8.94], [8.95], [8.96], [8.97], [8.98], [8.99], [9.0], [9.01], [9.02], [9.03], [9.04], [9.05], [9.06], [9.07], [9.08], [9.09], [9.1], [9.11], [9.12], [9.13], [9.14], [9.15], [9.16], [9.17], [9.18], [9.19], [9.2], [9.21], [9.22], [9.23], [9.24], [9.25], [9.26], [9.27], [9.28], [9.29], [9.3], [9.31], [9.32], [9.33], [9.34], [9.35], [9.36], [9.37], [9.38], [9.39], [9.4], [9.41], [9.42], [9.43], [9.44], [9.45], [9.46], [9.47], [9.48], [9.49], [9.5], [9.51], [9.52], [9.53], [9.54], [9.55], [9.56], [9.57], [9.58], [9.59], [9.6], [9.61], [9.62], [9.63], [9.64], [9.65], [9.66], [9.67], [9.68], [9.69], [9.7], [9.71], [9.72], [9.73], [9.74], [9.75], [9.76], [9.77], [9.78], [9.79], [9.8], [9.81], [9.82], [9.83], [9.84], [9.85], [9.86], [9.87], [9.88], [9.89], [9.9], [9.91], [9.92], [9.93], [9.94], [9.95], [9.96], [9.97], [9.98], [9.99], [10.0], [10.01], [10.02], [10.03], [10.04], [10.05], [10.06], [10.07], [10.08], [10.09], [10.1], [10.11], [10.12], [10.13], [10.14], [10.15], [10.16], [10.17], [10.18], [10.19], [10.2], [10.21], [10.22], [10.23], [10.24], [10.25], [10.26], [10.27], [10.28], [10.29], [10.3], [10.31], [10.32], [10.33], [10.34], [10.35], [10.36], [10.37], [10.38], [10.39], [10.4], [10.41], [10.42], [10.43], [10.44], [10.45], [10.46], [10.47], [10.48], [10.49], [10.5], [10.51], [10.52], [10.53], [10.54], [10.55], [10.56], [10.57], [10.58], [10.59], [10.6], [10.61], [10.62], [10.63], [10.64], [10.65], [10.66], [10.67], [10.68], [10.69], [10.7], [10.71], [10.72], [10.73], [10.74], [10.75], [10.76], [10.77], [10.78], [10.79], [10.8], [10.81], [10.82], [10.83], [10.84], [10.85], [10.86], [10.87], [10.88], [10.89], [10.9], [10.91], [10.92], [10.93], [10.94], [10.95], [10.96], [10.97], [10.98], [10.99], [11.0], [11.01], [11.02], [11.03], [11.04], [11.05], [11.06], [11.07], [11.08], [11.09], [11.1], [11.11], [11.12], [11.13], [11.14], [11.15], [11.16], [11.17], [11.18], [11.19], [11.2], [11.21], [11.22], [11.23], [11.24], [11.25], [11.26], [11.27], [11.28], [11.29], [11.3], [11.31], [11.32], [11.33], [11.34], [11.35], [11.36], [11.37], [11.38], [11.39], [11.4], [11.41], [11.42], [11.43], [11.44], [11.45], [11.46], [11.47], [11.48], [11.49], [11.5], [11.51], [11.52], [11.53], [11.54], [11.55], [11.56], [11.57], [11.58], [11.59], [11.6], [11.61], [11.62], [11.63], [11.64], [11.65], [11.66], [11.67], [11.68], [11.69], [11.7], [11.71], [11.72], [11.73], [11.74], [11.75], [11.76], [11.77], [11.78], [11.79], [11.8], [11.81], [11.82], [11.83], [11.84], [11.85], [11.86], [11.87], [11.88], [11.89], [11.9], [11.91], [11.92], [11.93], [11.94], [11.95], [11.96], [11.97], [11.98], [11.99], [12.0], [12.01], [12.02], [12.03], [12.04], [12.05], [12.06], [12.07], [12.08], [12.09], [12.1], [12.11], [12.12], [12.13], [12.14], [12.15], [12.16], [12.17], [12.18], [12.19], [12.2], [12.21], [12.22], [12.23], [12.24], [12.25], [12.26], [12.27], [12.28], [12.29], [12.3], [12.31], [12.32], [12.33], [12.34], [12.35], [12.36], [12.37], [12.38], [12.39], [12.4], [12.41], [12.42], [12.43], [12.44], [12.45], [12.46], [12.47], [12.48], [12.49], [12.5], [12.51], [12.52], [12.53], [12.54], [12.55], [12.56], [12.57], [12.58], [12.59], [12.6], [12.61], [12.62], [12.63], [12.64], [12.65], [12.66], [12.67], [12.68], [12.69], [12.7], [12.71], [12.72], [12.73], [12.74], [12.75], [12.76], [12.77], [12.78], [12.79], [12.8], [12.81], [12.82], [12.83],


```

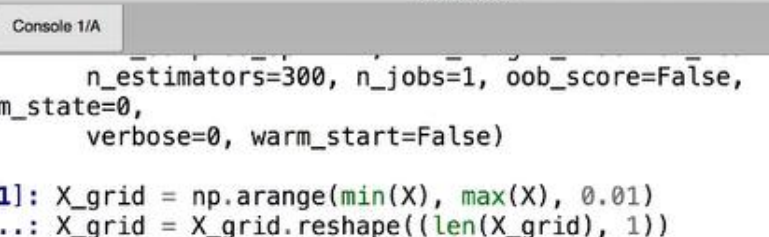
1# Random Forest Regression
2
3# Importing the libraries
4import numpy as np
5import matplotlib.pyplot as plt
6import pandas as pd
7
8# Importing the dataset
9dataset = pd.read_csv('Position_Salaries.csv')
10X = dataset.iloc[:, 1:2].values
11y = dataset.iloc[:, 2].values
12
13# Splitting the dataset into the Training set and Test set
14"""from sklearn.cross_validation import train_test_split
15X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 0)"""
16
17# Feature Scaling
18"""from sklearn.preprocessing import StandardScaler
19sc_X = StandardScaler()
20X_train = sc_X.fit_transform(X_train)
21X_test = sc_X.transform(X_test)
22sc_y = StandardScaler()
23y_train = sc_y.fit_transform(y_train)"""
24
25# Fitting Random Forest Regression to the dataset
26from sklearn.ensemble import RandomForestRegressor
27regressor = RandomForestRegressor(n_estimators = 300, random_state = 0)
28regressor.fit(X, y)
29
30# Predicting a new result
31y_pred = regressor.predict(6.5)
32
33# Visualising the Regression results (for higher resolution and smoother curve)
34X_grid = np.arange(min(X), max(X), 0.01)
35X_grid = X_grid.reshape((len(X_grid), 1))
36plt.scatter(X, y, color = 'red')
37plt.plot(X_grid, regressor.predict(X_grid), color = 'blue')
38plt.title('Truth or Bluff (Random Forest Regression)')
39plt.xlabel('Position level')
40plt.ylabel('Salary')
41plt.show()

```

Name	Type	Size	Value
X	int64	(10, 1)	array([[1], [2], [3], [4], [5], [6], [7], [8], [9], [10]])
X_grid	float64	(900, 1)	array([[1.], [1.01], [1.02], [1.03], [1.04], [1.05], [1.06], [1.07], [1.08], [1.09], [1.1], [1.11], [1.12], [1.13], [1.14], [1.15], [1.16], [1.17], [1.18], [1.19], [1.2], [1.21], [1.22], [1.23], [1.24], [1.25], [1.26], [1.27], [1.28], [1.29], [1.3], [1.31], [1.32], [1.33], [1.34], [1.35], [1.36], [1.37], [1.38], [1.39], [1.4], [1.41], [1.42], [1.43], [1.44], [1.45], [1.46], [1.47], [1.48], [1.49], [1.5], [1.51], [1.52], [1.53], [1.54], [1.55], [1.56], [1.57], [1.58], [1.59], [1.6], [1.61], [1.62], [1.63], [1.64], [1.65], [1.66], [1.67], [1.68], [1.69], [1.7], [1.71], [1.72], [1.73], [1.74], [1.75], [1.76], [1.77], [1.78], [1.79], [1.8], [1.81], [1.82], [1.83], [1.84], [1.85], [1.86], [1.87], [1.88], [1.89], [1.9], [1.91], [1.92], [1.93], [1.94], [1.95], [1.96], [1.97], [1.98], [1.99], [2.0], [2.01], [2.02], [2.03], [2.04], [2.05], [2.06], [2.07], [2.08], [2.09], [2.1], [2.11], [2.12], [2.13], [2.14], [2.15], [2.16], [2.17], [2.18], [2.19], [2.2], [2.21], [2.22], [2.23], [2.24], [2.25], [2.26], [2.27], [2.28], [2.29], [2.3], [2.31], [2.32], [2.33], [2.34], [2.35], [2.36], [2.37], [2.38], [2.39], [2.4], [2.41], [2.42], [2.43], [2.44], [2.45], [2.46], [2.47], [2.48], [2.49], [2.5], [2.51], [2.52], [2.53], [2.54], [2.55], [2.56], [2.57], [2.58], [2.59], [2.6], [2.61], [2.62], [2.63], [2.64], [2.65], [2.66], [2.67], [2.68], [2.69], [2.7], [2.71], [2.72], [2.73], [2.74], [2.75], [2.76], [2.77], [2.78], [2.79], [2.8], [2.81], [2.82], [2.83], [2.84], [2.85], [2.86], [2.87], [2.88], [2.89], [2.9], [2.91], [2.92], [2.93], [2.94], [2.95], [2.96], [2.97], [2.98], [2.99], [3.0], [3.01], [3.02], [3.03], [3.04], [3.05], [3.06], [3.07], [3.08], [3.09], [3.1], [3.11], [3.12], [3.13], [3.14], [3.15], [3.16], [3.17], [3.18], [3.19], [3.2], [3.21], [3.22], [3.23], [3.24], [3.25], [3.26], [3.27], [3.28], [3.29], [3.3], [3.31], [3.32], [3.33], [3.34], [3.35], [3.36], [3.37], [3.38], [3.39], [3.4], [3.41], [3.42], [3.43], [3.44], [3.45], [3.46], [3.47], [3.48], [3.49], [3.5], [3.51], [3.52], [3.53], [3.54], [3.55], [3.56], [3.57], [3.58], [3.59], [3.6], [3.61], [3.62], [3.63], [3.64], [3.65], [3.66], [3.67], [3.68], [3.69], [3.7], [3.71], [3.72], [3.73], [3.74], [3.75], [3.76], [3.77], [3.78], [3.79], [3.8], [3.81], [3.82], [3.83], [3.84], [3.85], [3.86], [3.87], [3.88], [3.89], [3.9], [3.91], [3.92], [3.93], [3.94], [3.95], [3.96], [3.97], [3.98], [3.99], [4.0], [4.01], [4.02], [4.03], [4.04], [4.05], [4.06], [4.07], [4.08], [4.09], [4.1], [4.11], [4.12], [4.13], [4.14], [4.15], [4.16], [4.17], [4.18], [4.19], [4.2], [4.21], [4.22], [4.23], [4.24], [4.25], [4.26], [4.27], [4.28], [4.29], [4.3], [4.31], [4.32], [4.33], [4.34], [4.35], [4.36], [4.37], [4.38], [4.39], [4.4], [4.41], [4.42], [4.43], [4.44], [4.45], [4.46], [4.47], [4.48], [4.49], [4.5], [4.51], [4.52], [4.53], [4.54], [4.55], [4.56], [4.57], [4.58], [4.59], [4.6], [4.61], [4.62], [4.63], [4.64], [4.65], [4.66], [4.67], [4.68], [4.69], [4.7], [4.71], [4.72], [4.73], [4.74], [4.75], [4.76], [4.77], [4.78], [4.79], [4.8], [4.81], [4.82], [4.83], [4.84], [4.85], [4.86], [4.87], [4.88], [4.89], [4.9], [4.91], [4.92], [4.93], [4.94],

For 300 trees the predicted value is 160333

Now, by using 300 trees, RF has beaten Polynomial regression as well



```
Object inspector  Variable explorer  File explorer  IPython console

Console 1/A

n_estimators=300, n_jobs=1, oob_score=False,
random_state=0,
verbose=0, warm_start=False)

In [11]: X_grid = np.arange(min(X), max(X), 0.01)
....: X_grid = X_grid.reshape((len(X_grid), 1))
....: plt.scatter(X, y, color = 'red')
....: plt.plot(X_grid, regressor.predict(X_grid), color =
'blue')
....: plt.title('Truth or Bluff (Random Forest Regression)')
....: plt.xlabel('Position level')
....: plt.ylabel('Salary')
....: plt.show()

In [12]: y_pred = regressor.predict(6.5)

In [13]:
```