Non Linear Regression Models

Machine Learning

Dr. Adnan Abid

Linear Regression Models Summary

Regressions

Simple Linear Regression

$$y = b_0 + b_1 x_1$$

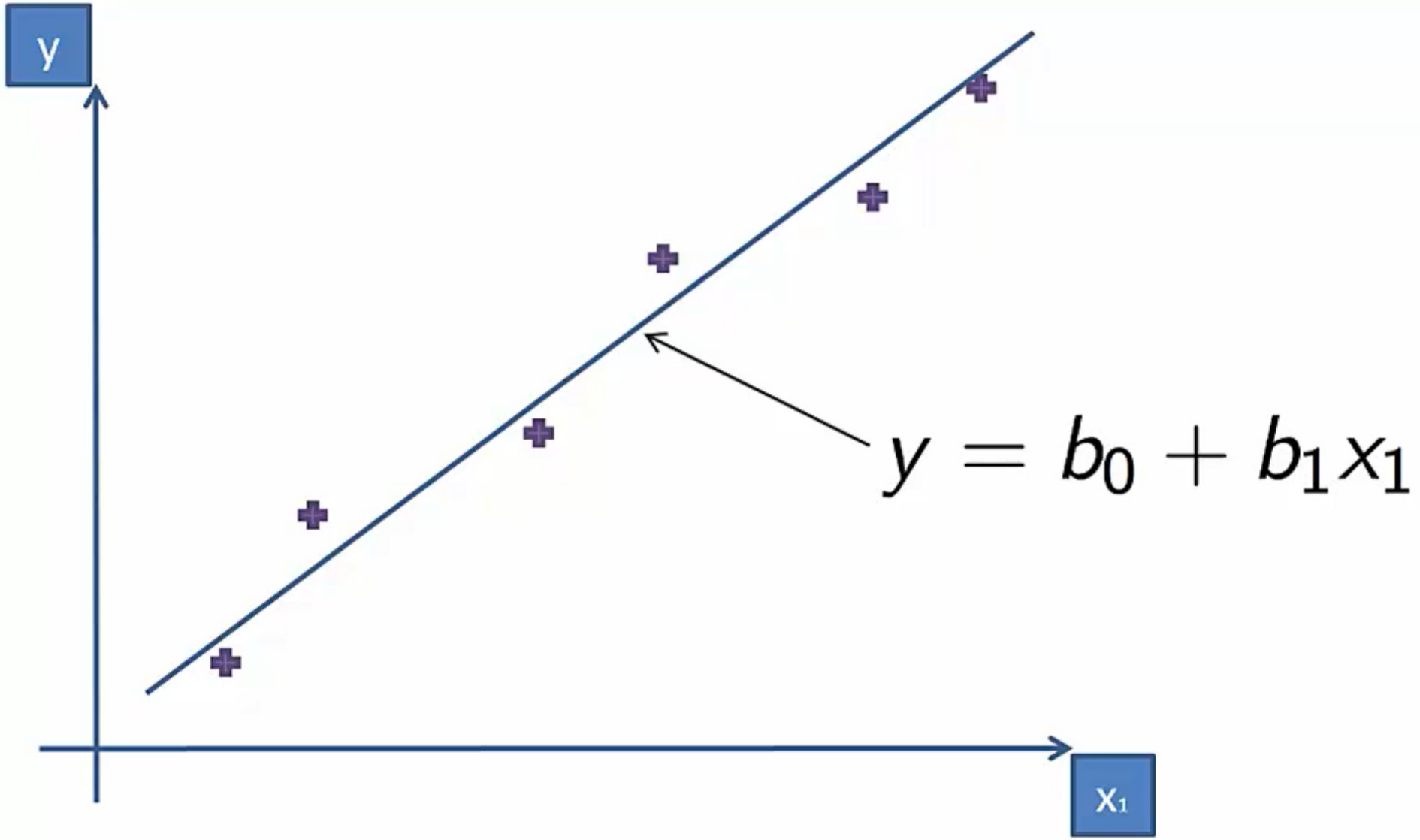
Multiple Linear Regression

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

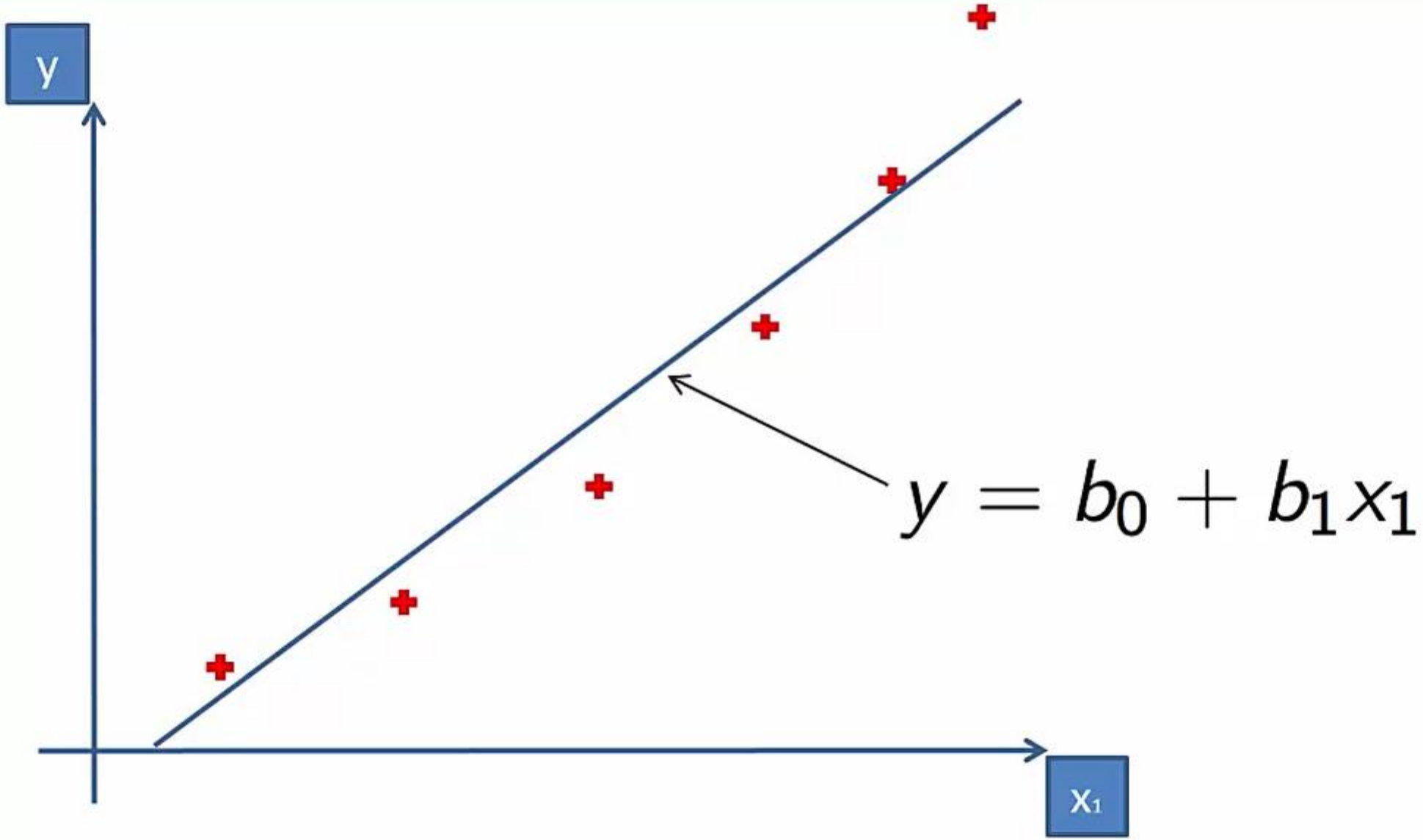
Polynomial Linear Regression

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

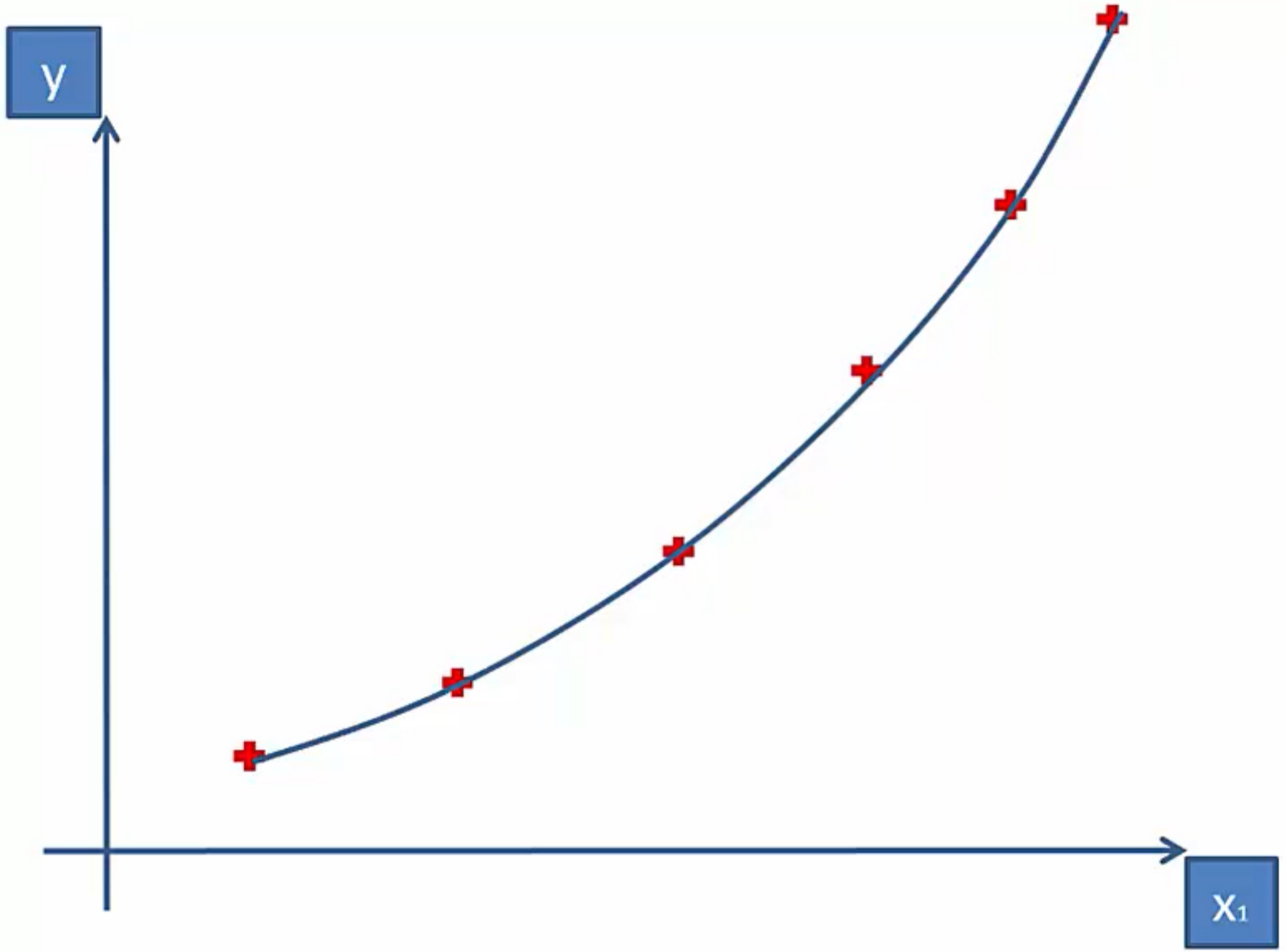
Simple Linear Regression



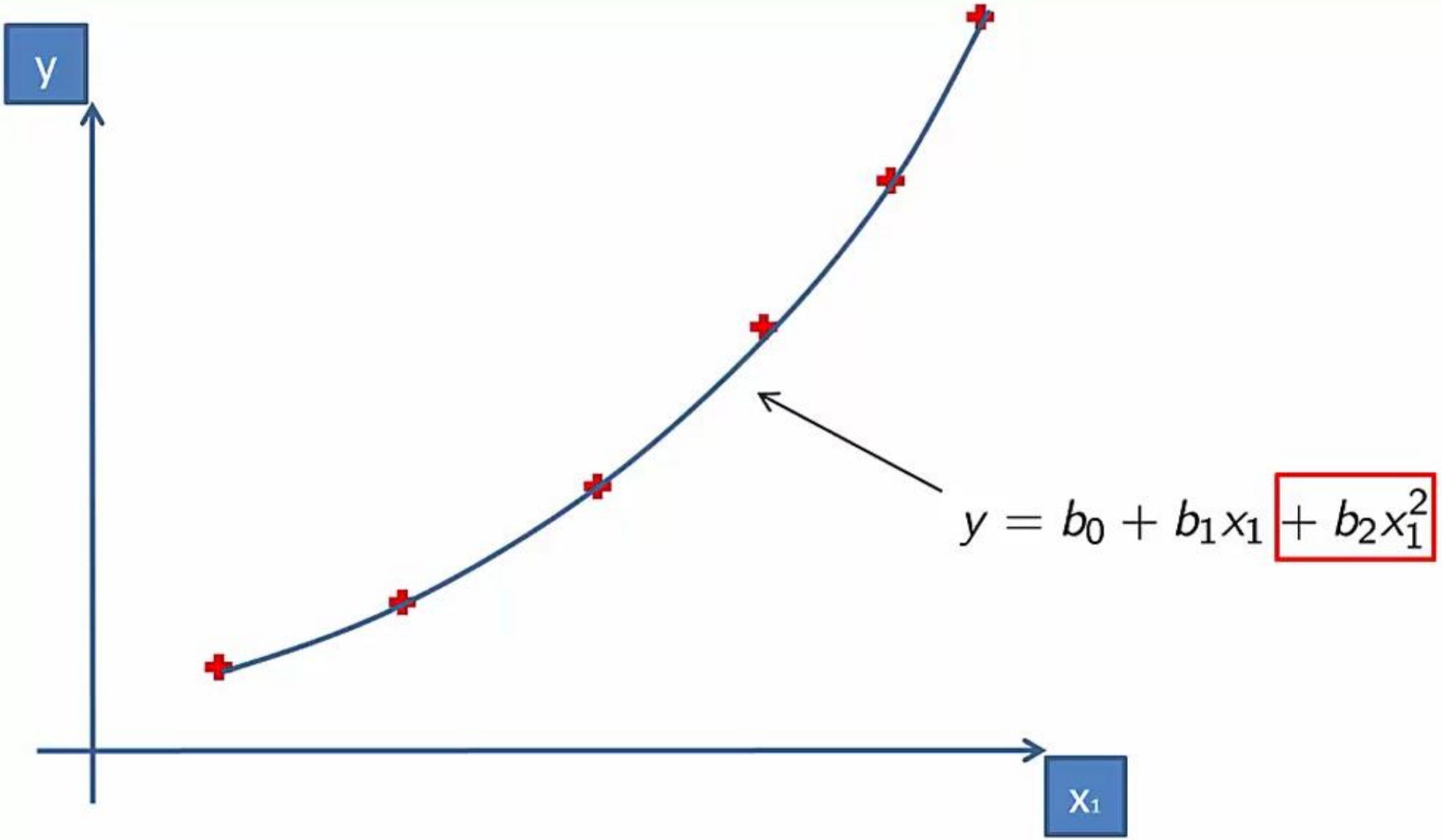
Simple Linear Regression



Polinomial Regression



Polinomial Regression



Polinomial Regression

Polynomial Linear Regression

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

Non Linear Regression Models

Support Vector Regression

Decision Tree Regression

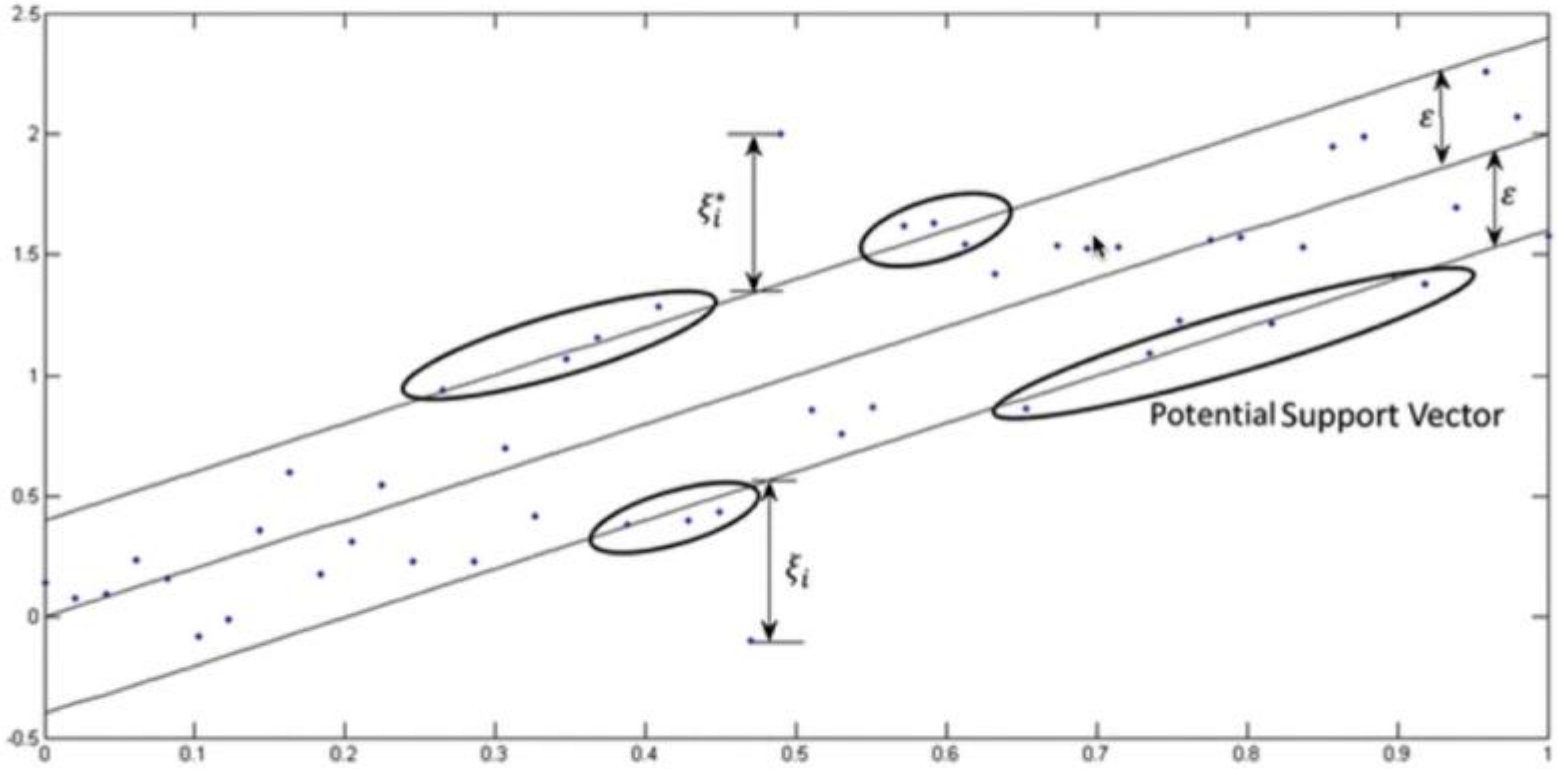
Random Forest Regression

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

Editor - /Users/Hadelin/Desktop/Machine Learning A-Z/Part 2 - Regression/Section 7 - Support Vector Regression (SVR)/svr.py

regression_template.py | svr.py*

```
# SVR

# 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

# 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_state = 0)"""

# Feature Scaling
from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()
sc_y = StandardScaler()
X = sc_X.fit_transform(X)
y = sc_y.fit_transform(y)

# Fitting SVR to the dataset
from sklearn.svm import SVR
regressor = SVR(kernel = 'rbf')
regressor.fit(X, y)

# Predicting a new result
y_pred = sc_y.inverse_transform(regressor.predict(sc_X.transform(np.array([[6.5]]))))

# Visualising the SVR results
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()
```

Variable explorer

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])

For SVM Regression the predicted value is 170370

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

Object inspector | Variable explorer | File explorer

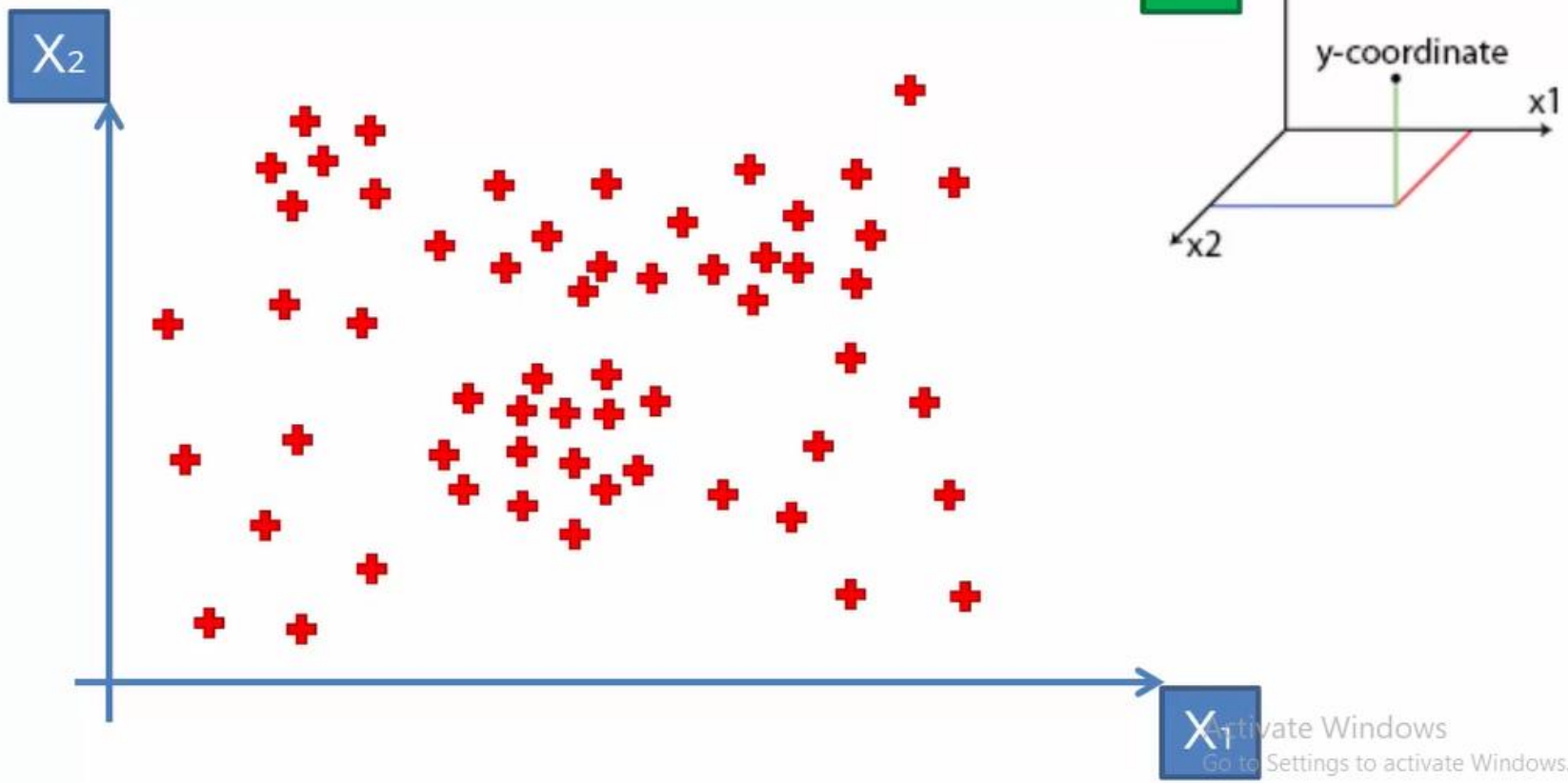
IPython console

```
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()

ansform(regressor.predict(sc_X.transform(np.array(
```

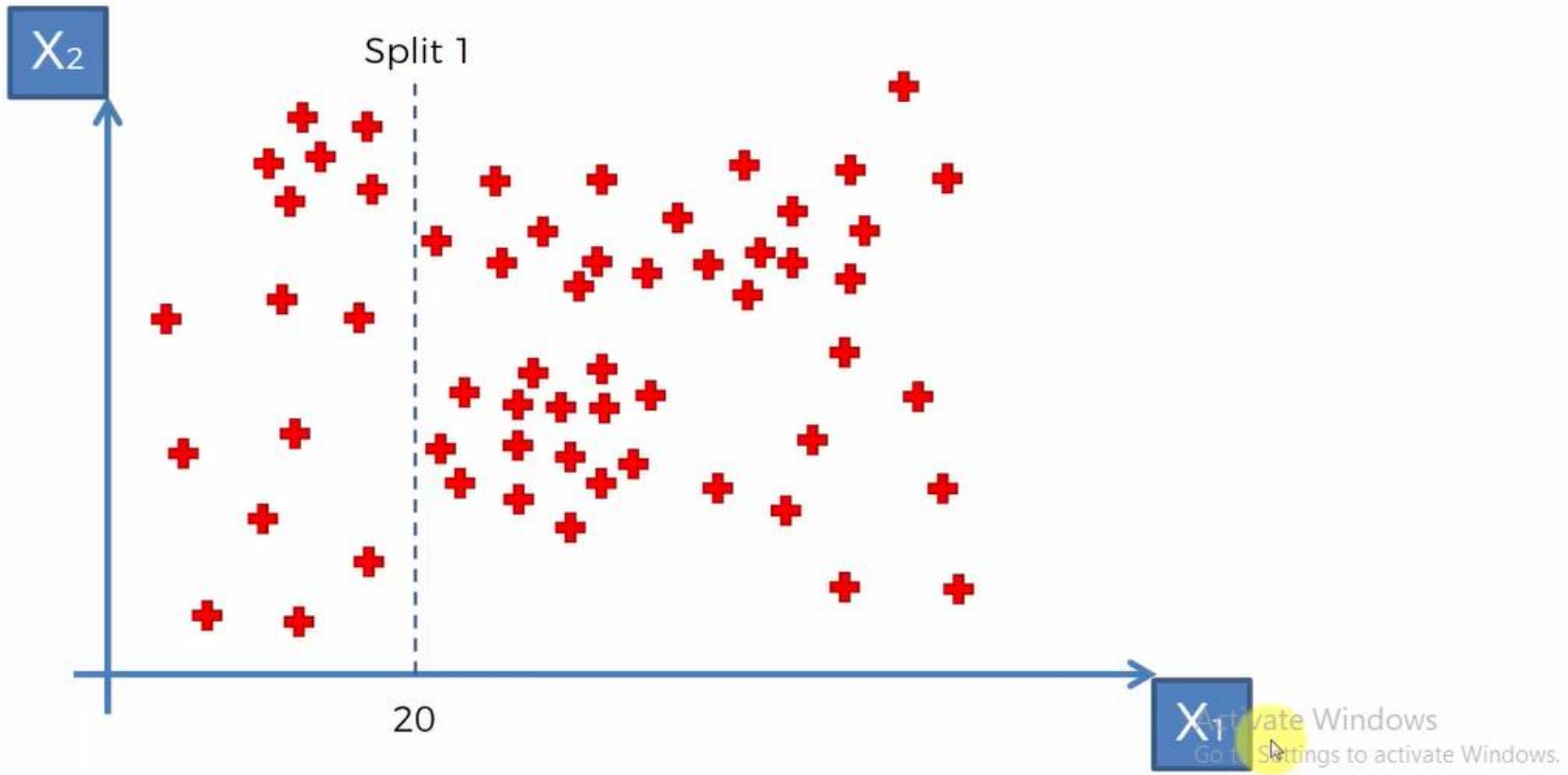

After explicit scaling, we need to make sure that we scale the input value (6.5); and inverse scale the prediction to get actual/meaningful predicted value

Decision Tree Regression

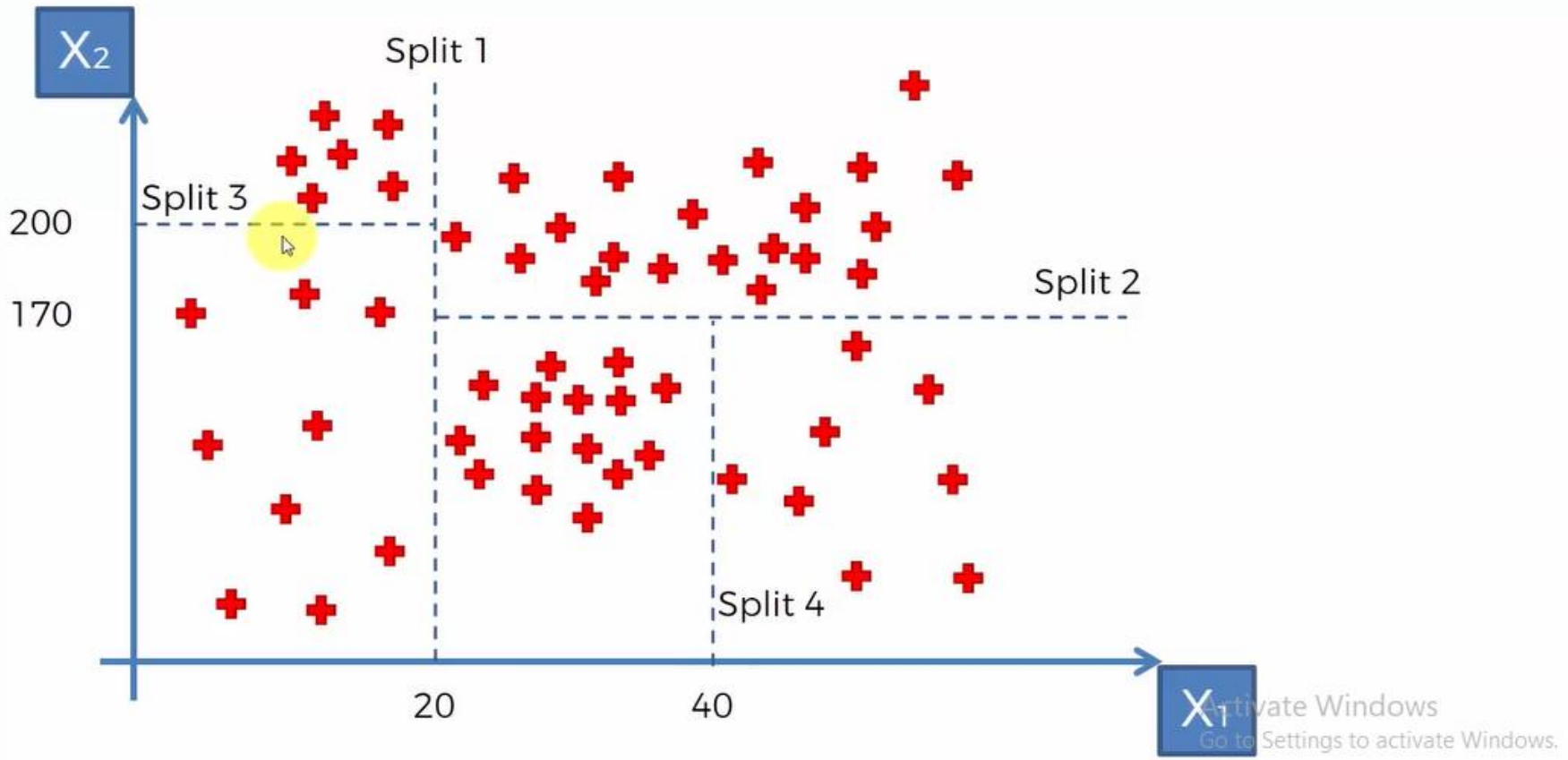
Decision Tree Intuition



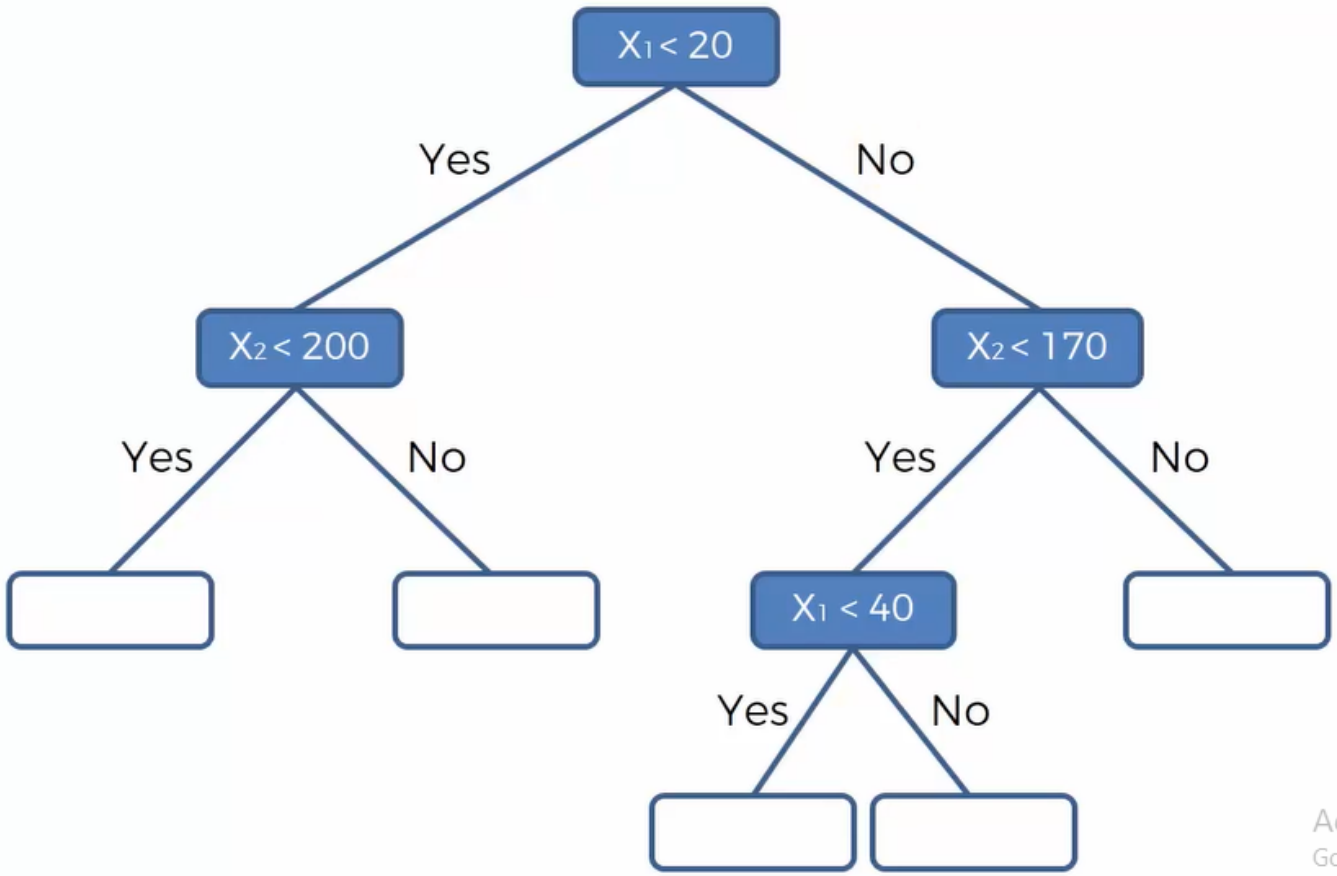
Decision Tree Intuition



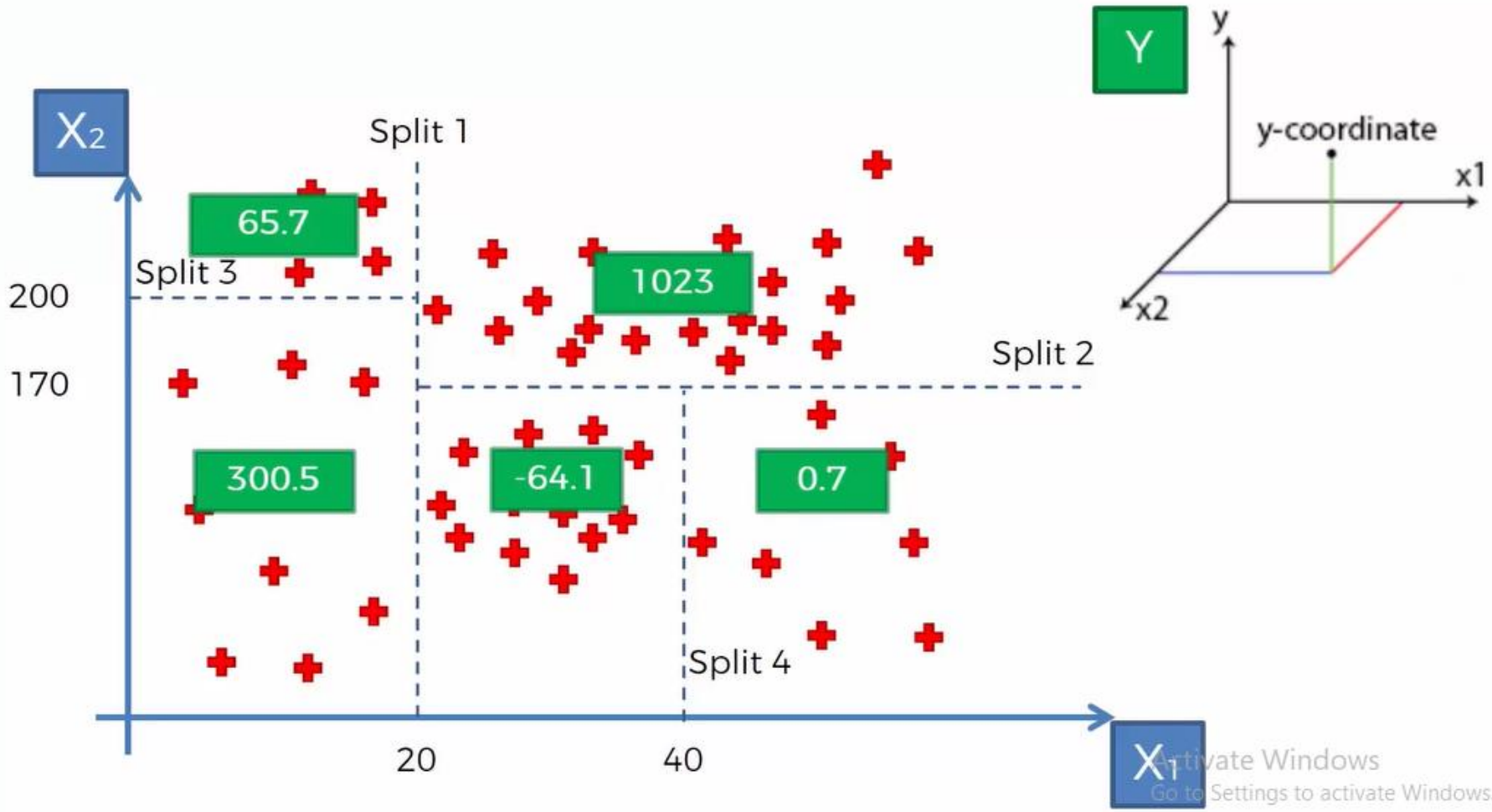
Decision Tree Intuition



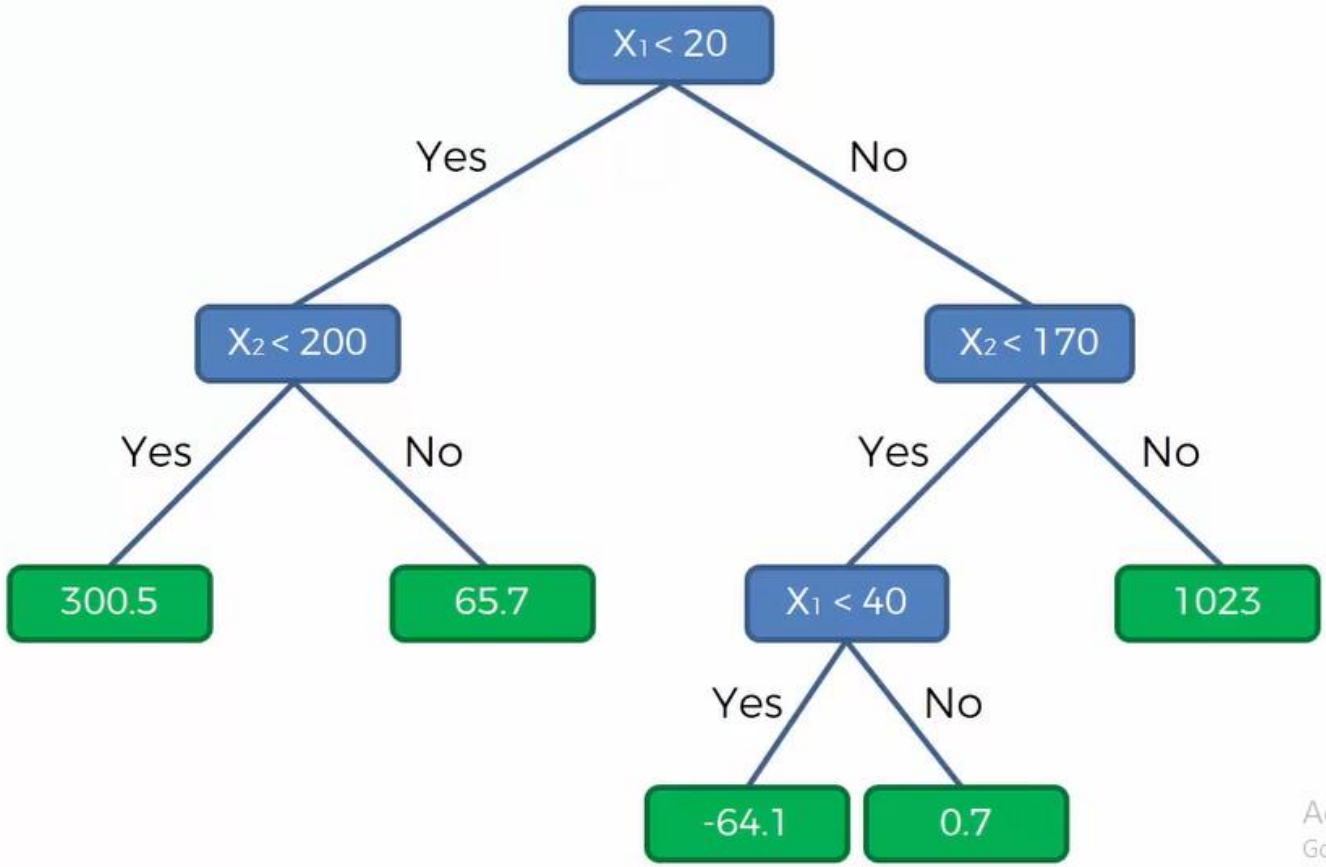
Decision Tree Intuition



Decision Tree Intuition



Decision Tree Intuition



```python
# Decision Tree Regression

# 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

# 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_state = 0)"""

# 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)
sc_y = StandardScaler()
y_train = sc_y.fit_transform(y_train)"""

# Fitting the Decision Tree Regression to the dataset
from sklearn.tree import DecisionTreeRegressor
regressor = DecisionTreeRegressor(random_state = 0)
regressor.fit(X, y)

# Predicting a new result
y_pred = regressor.predict(6.5)

# Visualising the Decision Tree Regression results (for higher resolution and smoother curve)
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)')
```

X_grid	float64	(900, 1)	array([[1.], [1.01],
dataset	DataFrame	(10, 3)	Column names: Position, Level, Sala
y	int64	(10,)	array([45000, 50000, 60000, 300000, 500000, 1000000])
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.

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.

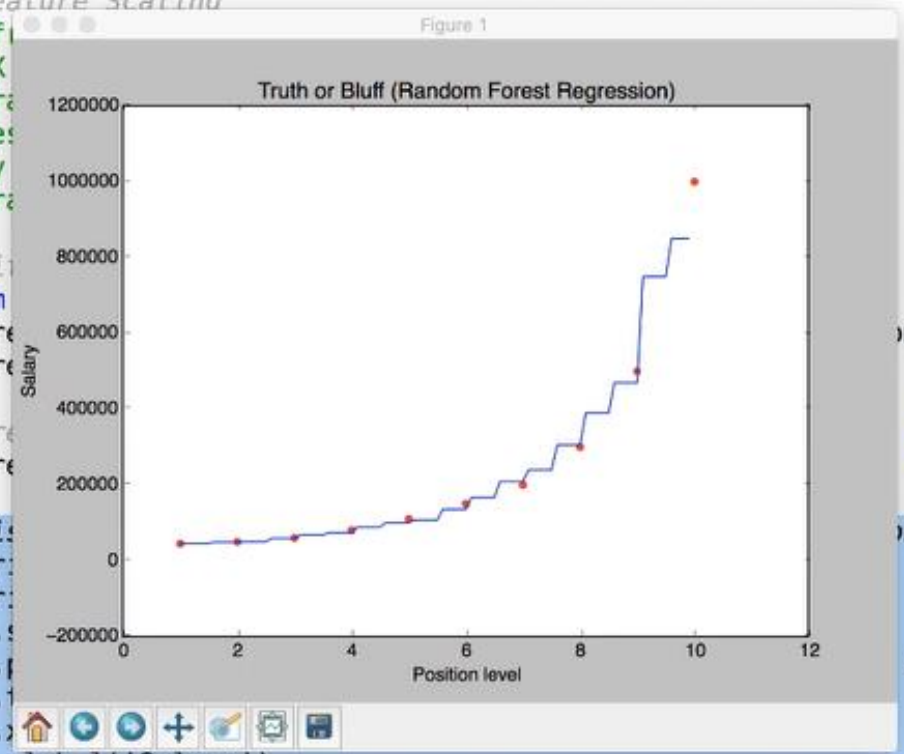
STEP 3: Choose the number Ntree of trees you want to build and repeat STEPS 1 & 2

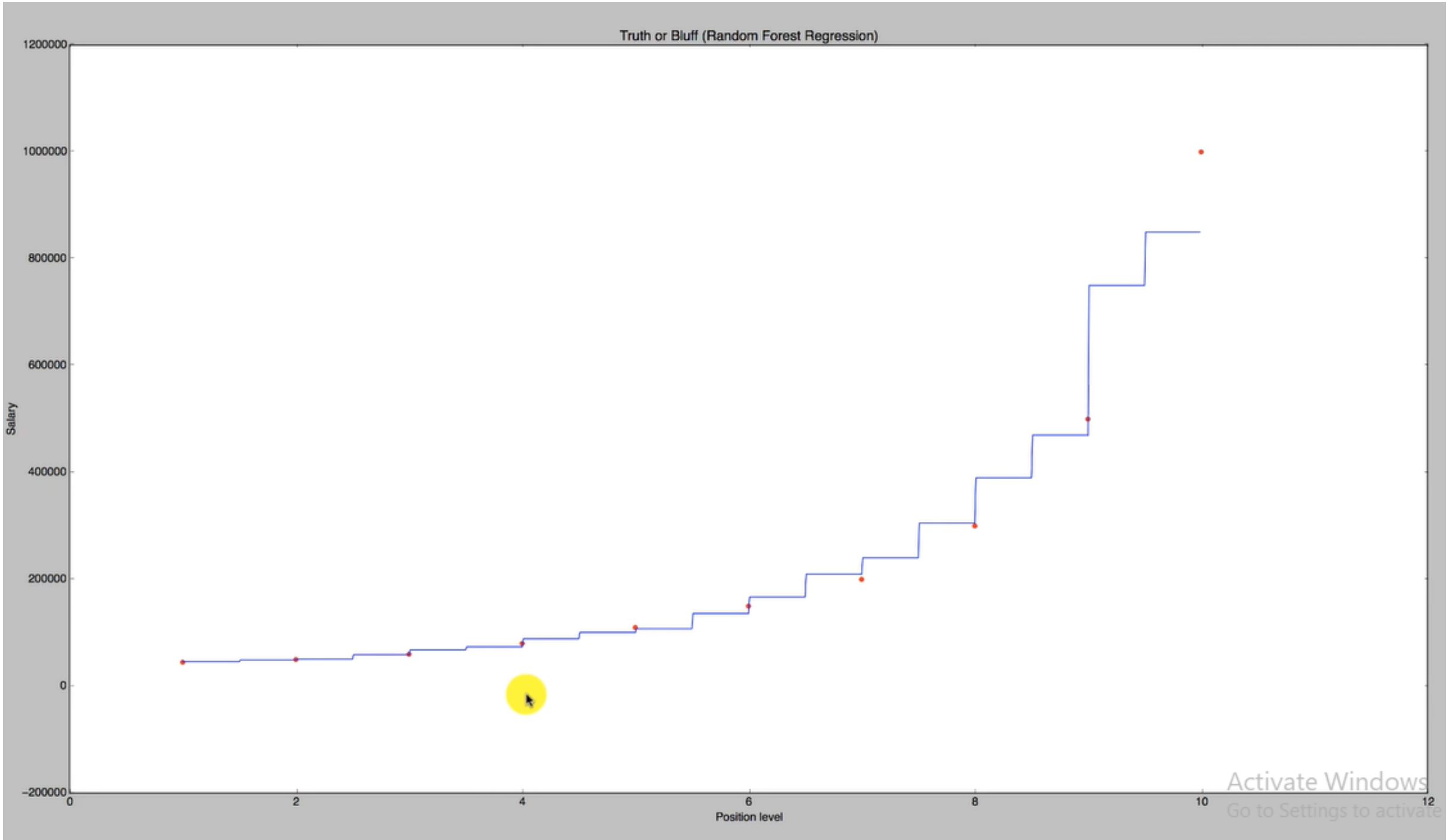
STEP 4: For a new data point, make each one of your Ntree trees predict the value of Y to for the data point in question, and assign the new data point the average across all of the predicted Y values.

```python
# Random Forest Regression

# 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

# 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_state = 0)"""

# 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)
sc_y = StandardScaler()
y_train = sc_y.fit_transform(y_train)"""

# Fitting Random Forest Regression to the dataset
from sklearn.ensemble import RandomForestRegressor
regressor = RandomForestRegressor(n_estimators = 10, random_state = 0)
regressor.fit(X, y)

# Predicting a new result
y_pred = regressor.predict(6.5)

# Visualising the Regression results (for higher resolution and smoother curve)
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, regressor.predict(X_grid), color = 'blue')
plt.title('Truth or Bluff (Random Forest Regression)')
plt.xlabel('Position level')
plt.ylabel('Salary')
plt.show()
```

```
# Random Forest Regression

# 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

# 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_state = 0)"""

# Feature Scaling
"""fi
sc_X
X_tra
X_tes
sc_y
y_tra

# Fi
from
regre
regre

# Pre
y_pre

# Vis
X_gri
X_gri
plt.s
plt.p
plt.t
plt.x
plt.ylabel('Salary')
plt.show()
```

```
om_state = 0)

n and smoother curve)

)
```





```python
# Decision Tree Regression

# 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

# 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_state = 0)"""

# 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)
sc_y = StandardScaler()
y_train = sc_y.fit_transform(y_train)"""

# Fitting the Decision Tree Regression to the dataset
from sklearn.tree import DecisionTreeRegressor
regressor = DecisionTreeRegressor(random_state = 0)
regressor.fit(X, y)

# Predicting a new result
y_pred = regressor.predict(6.5)

# Visualising the Decision Tree Regression results (for higher resolution and smoother curve)
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)')
```

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([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*

```python
# Random Forest Regression

# 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

# 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_state = 0)"""

# 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)
sc_y = StandardScaler()
y_train = sc_y.fit_transform(y_train)"""

# Fitting Random Forest Regression to the dataset
from sklearn.ensemble import RandomForestRegressor
regressor = RandomForestRegressor(n_estimators = 10, random_state = 0)
regressor.fit(X, y)

# Predicting a new result
y_pred = regressor.predict(6.5)

# Visualising the Regression results (for higher resolution and smoother curve)
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()
```

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 = 'blu
   ...: 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]:
```


Permissions: RW | End-of-lines: LF | Encoding: UTF-8-GUESSED | Line: 31 | Column: 32 | Memory: 54

regression_template.py | random_forest_regression.py*

```python
# Random Forest Regression

# 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

# 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_state = 0)"""

# 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)
sc_y = StandardScaler()
y_train = sc_y.fit_transform(y_train)"""

# Fitting Random Forest Regression to the dataset
from sklearn.ensemble import RandomForestRegressor
regressor = RandomForestRegressor(n_estimators = 100, random_state = 0)
regressor.fit(X, y)

# Predicting a new result
y_pred = regressor.predict(6.5)

# Visualising the Regression results (for higher resolution and smoother curve)
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()
```

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([158300.])

For 100 trees the predicted value is 158300

Object inspector | Variable explorer | File explorer

IPython console

Console 1/A

```
           min_samples_split=2, min_weight_fraction_leaf=0.0,
           n_estimators=100, n_jobs=1, oob_score=False,
random_state=0,
           verbose=0, warm_start=False)

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 (Random Forest Regression)')
   ...: plt.xlabel('Position level')
   ...: plt.ylabel('Salary')
   ...: plt.show()

In [9]: y_pred = regressor.predict(6.5)

In [10]:
```


Permissions: RW | End-of-lines: LF | Encoding: UTF-8-GUESSED | Line: 31 | Column: 32 | Memory: 54 %

regression_template.py | random_forest_regression.py*

```python
# Random Forest Regression

# 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

# 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_state = 0)"""

# 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)
sc_y = StandardScaler()
y_train = sc_y.fit_transform(y_train)"""

# Fitting Random Forest Regression to the dataset
from sklearn.ensemble import RandomForestRegressor
regressor = RandomForestRegressor(n_estimators = 300, random_state = 0)
regressor.fit(X, y)

# Predicting a new result
y_pred = regressor.predict(6.5)

# Visualising the Regression results (for higher resolution and smoother curve)
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()
```

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([160333.33333333])

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]:
```