Paper 060010907: Machine Learning

Unit-2 Regression

1

Regression Analysis

- It is a statistical modeling technique.
- It is the process of investigating relationships between dependent and independent variables.
- Aims to predict outputs on a continuous scale rather than categorical class labels.

2

Regression Analysis

- The dependent or target variable is estimated as a function of independent or predictor variables.
- The estimation function is called the *regression function.*

3

Regression Analysis

- According to Wikipedia,
  - In statistical modeling, regression analysis is a set of statistical processes for estimating the **relationships between a dependent variable** (often called the 'outcome variable') **and one or more independent variables** (often called 'predictors', 'covariates', or 'features').
- https://en.wikipedia.org/wiki/Regression_analysis

4

Regression Analysis

- It includes,
  - The independent variable X
  - The dependent or target variable Y
  - Unknown parameter(s), denoted as β
- $Y = f(X, \beta)$
- The function f() needs to learn from the training dataset.

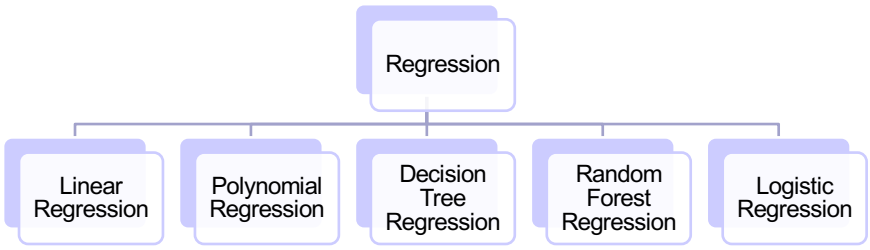
5

Why Regression Analysis??

- Regression estimates the relationship between the target and the independent variable.
- It is used to find the trends in data.
- It helps to predict real/continuous values.
- It is used to determine the most/least important factor.

6

Types of Regression



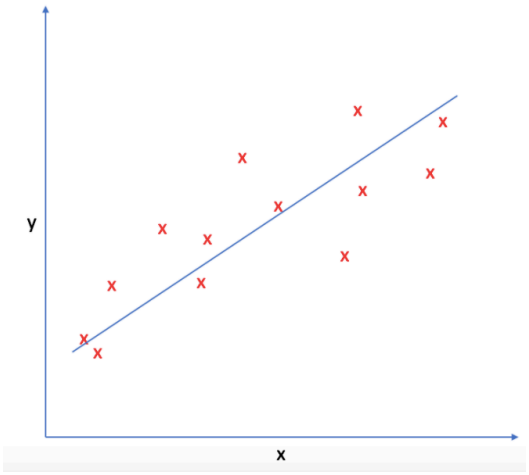
7

Linear Regression

- Linear regression is the analysis of relationship between the dependent and independent variables.
- Linear regression assumes linear relationship between the two variables.

8

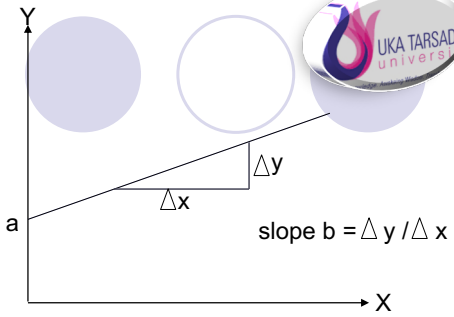
Linear Regression



9

Linear Regression

- Y= a + bX



- Where,
  - Y is the dependent variable (that's the variable that goes on the Y axis)
  - X is the independent variable (i.e. it is plotted on the X axis)
  - b is the slope of the line
  - a is the y-intercept

10

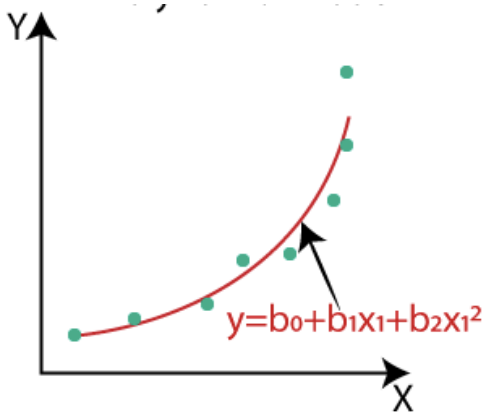
Linear Regression

- According to Wikipedia,
  - In statistics, linear regression is a linear approach to modeling the relationship between a scalar response (or dependent variable) and one or more explanatory variables (or independent variables).
- https://en.wikipedia.org/wiki/Linear_regression

Polynomial Regression

- Dependent variable is related polynomially to independent variable
- Datapoints are best fitted using a polynomial line

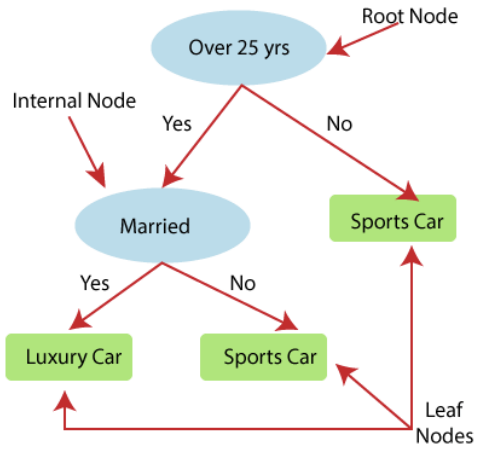
Polynomial Regression



Decision Tree Regression

- Builds a tree-like structure in which each internal node represents the "test" for an attribute, each branch represent the result of the test, and each leaf node represents the final decision or result.

Decision Tree Regression



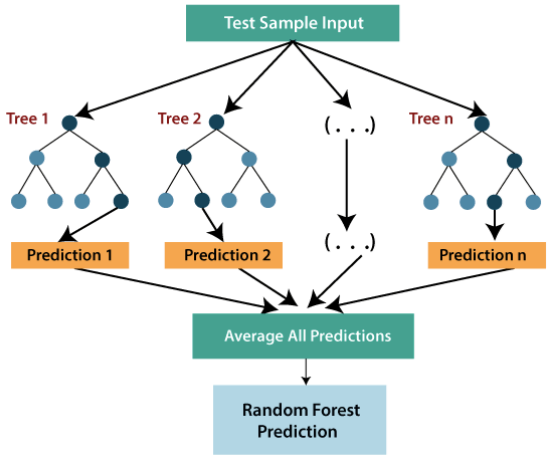
15

Random Forest Regression

- Combines multiple decision trees and predicts the final output based on the average of each tree output.

16

Random Forest Regression



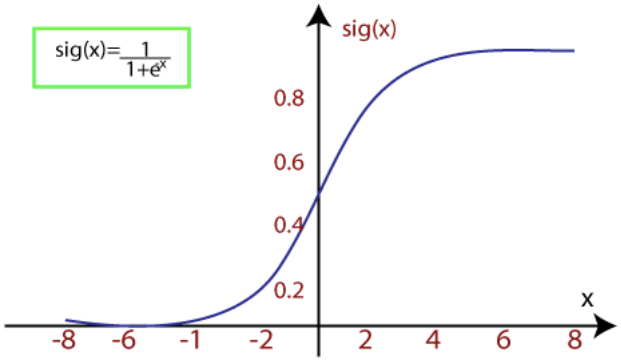
17

Logistic Regression

- Used to solve the classification problems
- Works with the categorical variable such as 0 or 1, Yes or No, True or False, Spam or not spam, etc.

18

Logistic Regression



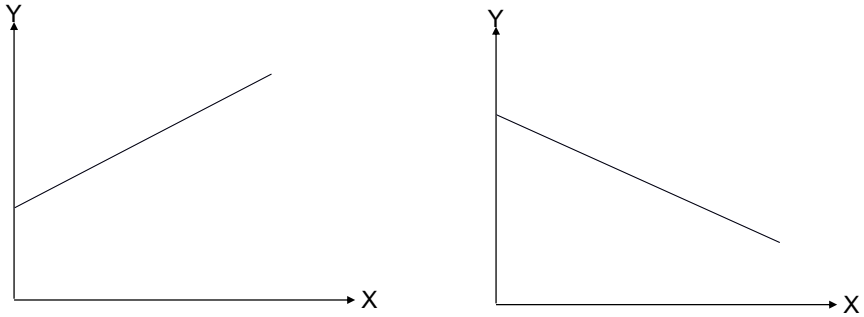
19

LINEAR REGRESSION

20

Linear Regression

- A linear line showing the relationship between the dependent and independent variables.



Positive Line

Negative Line

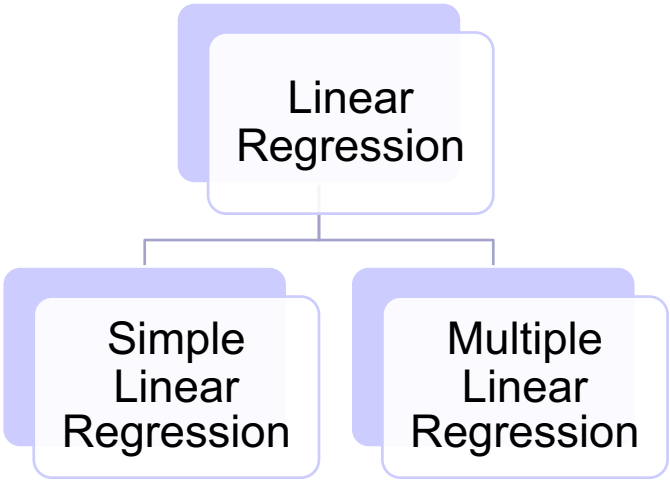
21

Linear Regression

- Objective is to find the best fit line.
- Error between predicted values and actual values should be minimized.

22

Linear Regression



23

Simple Linear Regression

- When we have a single input attribute (x) and we want to use linear regression, this is called simple linear regression.

24

Simple Linear Regression

- According to Wikipedia,
  - In statistics, simple linear regression is a linear regression model with a single explanatory variable.
- https://en.wikipedia.org/wiki/Simple_linear_regression

25

Simple Linear Regression

- It models the relationship between a dependent variable and a single independent variable.
- Dependent variable must be continuous/numerical value.

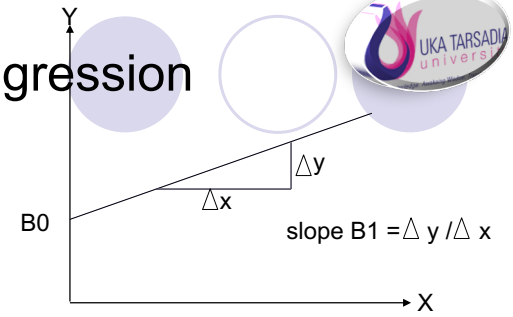
26

Simple Linear Regression

- Examples,
  - Increase in website traffic vs SEO efforts
  - Product sell vs marketing expenditure
  - Reducing air from tyre vs time

27

Simple Linear Regression



- $Y = B_0 + B_1X$
- Where,
  - Y is the dependent variable (output variable we want to predict)
  - X is the independent variable (we know/have it)
  - B_1 is the slope of the line
  - B_0 is the y-intercept

28

Simple Linear Regression

- $Y = B_0 + B_1X$
- x is the input variable we know and B_0 and B_1 are coefficients that we need to estimate
- The objective is to find the best estimates for the coefficients B_0 and B_1 to minimize the errors in predicting y from x.

Simple Linear Regression

- Estimate the slope B1
  - B1 = sum((xi-mean(x)) * (yi-mean(y))) / sum((xi – mean(x))^2)
- Calculate B0
  - B0 = mean(y) – B1 * mean(x)

Simple Linear Regression

- Sample calculation

x	y
1	1
2	3
4	3
3	2
5	5

- mean(x) = 3
- mean(y) = 2.8

31

Simple Linear Regression

- Calculate the error of each variable from the mean

x	mean(x)	x - mean(x)
1	3	-2
2	3	-1
4	3	1
3	3	0
5	3	2

y	mean(y)	y - mean(y)
1	2.8	-1.8
3	2.8	0.2
3	2.8	0.2
2	2.8	-0.8
5	2.8	2.2

32

Simple Linear Regression

- Now, multiply the error for each x with the error for each y

x - mean(x)	y - mean(y)	multiplication
-2	-1.8	3.6
-1	0.2	-0.2
1	0.2	0.2
0	-0.8	0
2	2.2	4.4

- calculate the sum of these multiplications
  - Sum of these multiplications is 8

Simple Linear Regression

- B1 = sum((xi-mean(x)) * (yi-mean(y))) / sum((xi – mean(x))^2)
- Calculate denominator (sum of the squared differences of each x value from the mean)
- Sum is 10

x – mean(x)	squared
-2	4
-1	1
1	1
0	0
2	4

Simple Linear Regression

- Now calculate slope
- B1 = 8 / 10
- B1 = 0.8

35

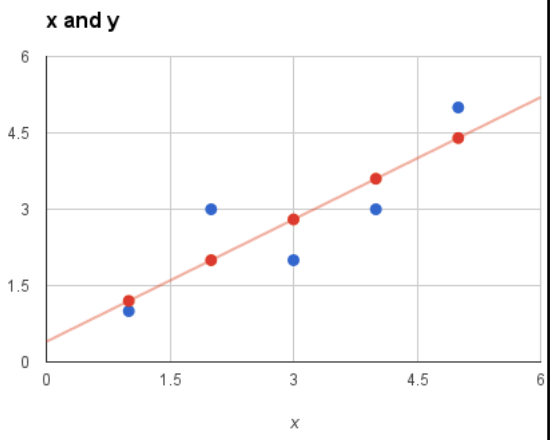
Simple Linear Regression

- Calculate interceptor B0
- B0 = mean(y) – B1 * mean(x)
- B0 = 2.8 – 0.8 * 3
- B0 = 0.4

36

Simple Linear Regression

- Now, Predict
- y = B0 + B1 * x
- y = 0.4 + 0.8 * x



Simple Linear Regression

- Calculate error in prediction
- Root Mean Squared Error = sqrt(sum((pi – yi)^2)/n)

predicted y	y	error
1.2	1	0.2
2	3	-1
3.6	3	0.6
2.8	2	0.8
4.4	5	-0.6

Simple Linear Regression

- calculate the square of each of these error values

error	squared error
0.2	0.04
-1	1
0.6	0.36
0.8	0.64
-0.6	0.36

- The sum of these errors is 2.4

39

Simple Linear Regression

- Dividing by n and taking the square root
  - Root Mean Square Error = sqrt(2.4 / 5)
  - Root Mean Square Error = 0.692
- Each prediction is on average wrong by about 0.692

40

Multiple Linear Regression

- It is the extension of simple linear regression that predicts a response using two or more features.
- It models the linear relationship between a single dependent continuous variable and more than one independent variables.

41

Multiple Linear Regression

- The dependent or target variable(Y) must be the continuous/real, but the predictor or independent variable may be of continuous or categorical form.

42

Multiple Linear Regression

- Each feature variable must model the linear relationship with the dependent variable.
- It tries to fit a regression line through a multidimensional space of data-points.

43

Multiple Linear Regression

- $y = b0 + b1x1 + b2x2 + \cdots + bnxn$
- Where,
  - y= Output/Response variable
  - b_0, b_1, b_2, b_3 , b_n....= Coefficients of the model.
  - x_1, x_2, x_3, x_4,...= Various Independent/feature variable

44

Multiple Linear Regression

- Assumptions
  - A linear relationship should exist between the Target and predictor variables.
  - The regression residuals must be normally distributed.
  - No multicollinearity (correlation between the independent variable) in data exists.

45

Multiple Linear Regression

- Example,
  - Company is spending money on R&D, Administration, Marketing
  - Companies operate in different states
  - Profit for a financial year is available

46

Multiple Linear Regression

- Importing Libraries

```
11  # Importing the libraries
12  import numpy as np
13  import matplotlib.pyplot as plt
14  import pandas as pd
```


47

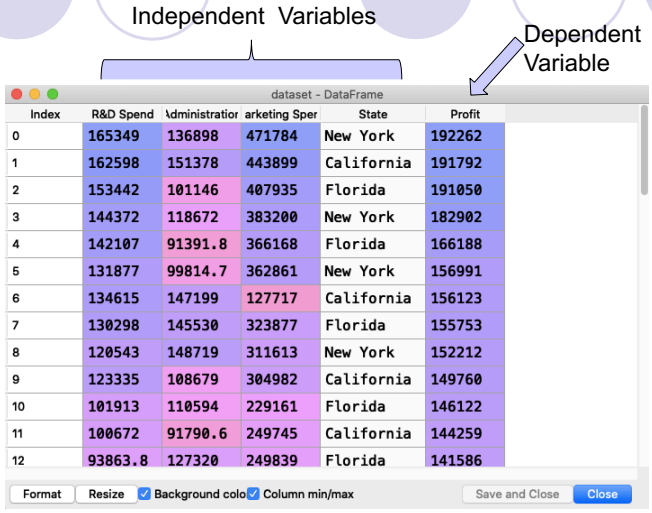
Multiple Linear Regression

- Importing Dataset

- # Importing the dataset
- dataset = pd.read_csv('Invest2Profit.csv')

48

Multiple Linear Regression



Multiple Linear Regression

- #Extracting Independent and dependent Variable
- X= dataset.iloc[:, :-1].values
- y= dataset.iloc[:, 4].values
- print(X)

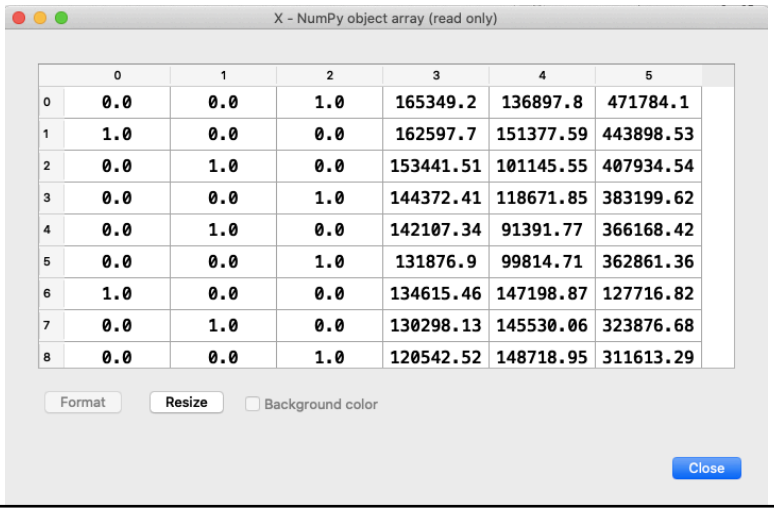
Multiple Linear Regression

- **Encoding Dummy Variables**
  - # Encoding categorical data
  - from sklearn.compose import ColumnTransformer
  - from sklearn.preprocessing import OneHotEncoder
  - ct = ColumnTransformer(
  - transformers=[
  - ('encoder', OneHotEncoder(), [3])], remainder='passthrough')
  - X = np.array(ct.fit_transform(X))
  - print(X)

51

Multiple Linear Regression

- X - array



	0	1	2	3	4	5
0	0.0	0.0	1.0	165349.2	136897.8	471784.1
1	1.0	0.0	0.0	162597.7	151377.59	443898.53
2	0.0	1.0	0.0	153441.51	101145.55	407934.54
3	0.0	0.0	1.0	144372.41	118671.85	383199.62
4	0.0	1.0	0.0	142107.34	91391.77	366168.42
5	0.0	0.0	1.0	131876.9	99814.71	362861.36
6	1.0	0.0	0.0	134615.46	147198.87	127716.82
7	0.0	1.0	0.0	130298.13	145530.06	323876.68
8	0.0	0.0	1.0	120542.52	148718.95	311613.29

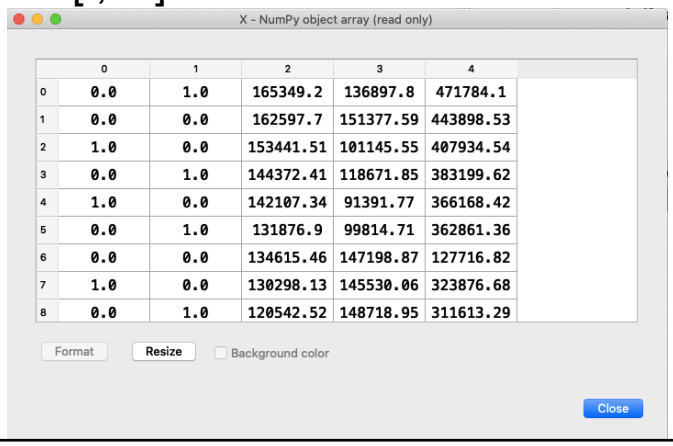
52

Multiple Linear Regression

- We **should not** use all the dummy variables at the same time
- So it must be 1 less than the total number of dummy variables, else it will create a dummy variable trap.

Multiple Linear Regression

- #avoiding the dummy variable trap:
- X = X[:, 1:]



	0	1	2	3	4
0	0.0	1.0	165349.2	136897.8	471784.1
1	0.0	0.0	162597.7	151377.59	443898.53
2	1.0	0.0	153441.51	101145.55	407934.54
3	0.0	1.0	144372.41	118671.85	383199.62
4	1.0	0.0	142107.34	91391.77	366168.42
5	0.0	1.0	131876.9	99814.71	362861.36
6	0.0	0.0	134615.46	147198.87	127716.82
7	1.0	0.0	130298.13	145530.06	323876.68
8	0.0	1.0	120542.52	148718.95	311613.29

Multiple Linear Regression

- # Splitting the dataset into training and test set.
- 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)

55

Multiple Linear Regression

- #Fitting the MLR model to the training set:
- from sklearn.linear_model **import** LinearR egression
- regressor= LinearRegression()
- regressor.fit(X_train, y_train)

56

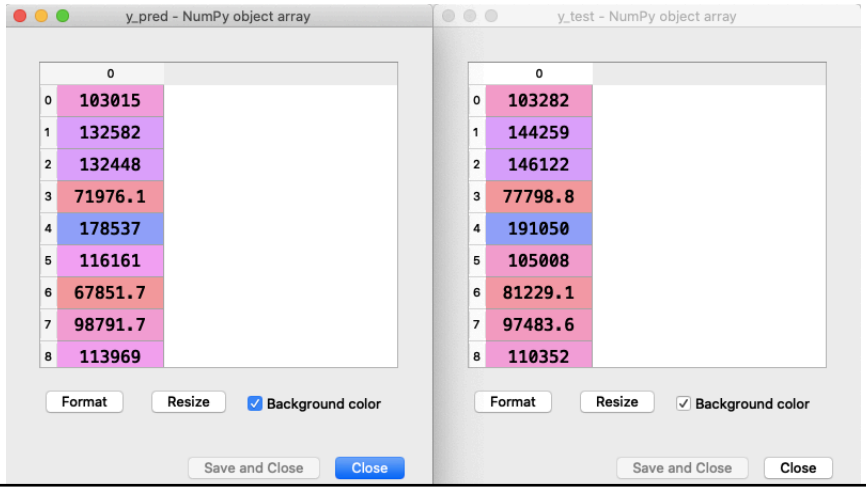
Multiple Linear Regression

- #Predicting the Test set result;
- y_pred= regressor.predict(X_test)

57

Multiple Linear Regression

- Now, compare y_pred and y_test



58

Multiple Linear Regression

- print("Train Score: ", regressor.score(X_train, y_train))
- print("Test Score: ", regressor.score(X_test, y_test))
- **Output:**

```
Train Score:  0.9501847627493607
Test Score:  0.9347068473282446
```

- Model is 95% accurate with the training dataset and 93% accurate with the test dataset.

Multiple Linear Regression – Backward Elimination

- Feature selection is used to remove features that do not have a significant effect on the dependent variable or prediction of output.

Multiple Linear Regression – Backward Elimination

- Feature selection algorithms:
  - All-in
  - Backward Elimination
  - Forward Selection
  - Bidirectional Elimination
  - Score Comparison

61

Multiple Linear Regression – Backward Elimination

Which is Most/least significant variable?

Independent Variables

Dependent Variable

dataset - DataFrame

	R&D Spend	\dministratior	arketing Sper	State	Profit
	165349	136898	471784	New York	192262
	162598	151378	443899	California	191792
2	153442	101146	407935	Florida	191050
3	144372	118672	383200	New York	182902
4	142107	91391.8	366168	Florida	166188
5	131877	99814.7	362861	New York	156991
6	134615	147199	127717	California	156123
7	130298	145530	323877	Florida	155753
8	120543	148719	311613	New York	152212
9	123335	108679	304982	California	149760
10	101913	110594	229161	Florida	146122
11	100672	91790.6	249745	California	144259
12	93863.8	127320	249839	Florida	141586

Format Resize Background colo Column min/max Save and Close Close

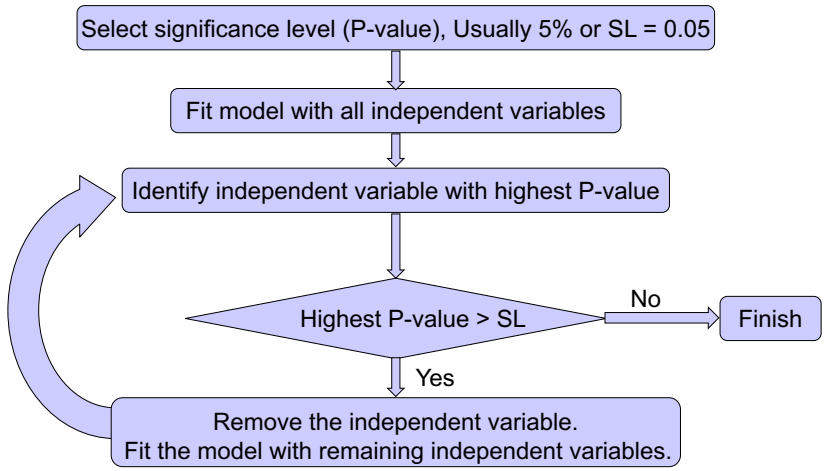
62

Multiple Linear Regression – Backward Elimination

- Objective:
  - Use only statistically significant independent variable(s)
  - Find optimal matrix of features
- Each of the selected independent variable is having great impact on the dependent variable.

63

Multiple Linear Regression – Backward Elimination



64

Multiple Linear Regression – Backward Elimination

- # Backward Elimination
- import statsmodels.api as sm

65

Multiple Linear Regression – Backward Elimination

- $y = b0 + b1x1 + b2x2 + \cdots + bnxn$

$y = b0x0 + b1x1 + b2x2 + \cdots + bnxn$

- Where,
  - y= Output/Response variable
  - b_0, b_1, b_2, b_3 , b_n....= Coefficients of the model.
  - x_1, x_2, x_3, x_4,...= Various Independent/feature variable
  - x0 = 1

66

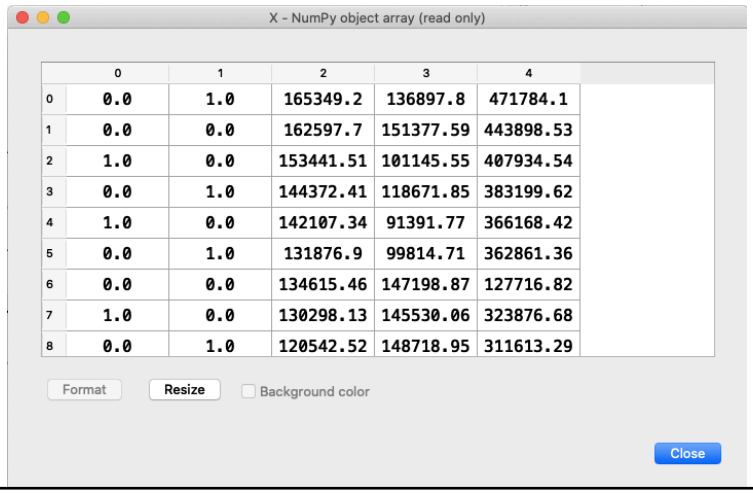
Multiple Linear Regression – Backward Elimination

- Add column of 1s at position 0
- X = np.append(arr = np.onces((50, 1)).astype(int), values = X, axis = 1)

67

Multiple Linear Regression – Backward Elimination

- Original X

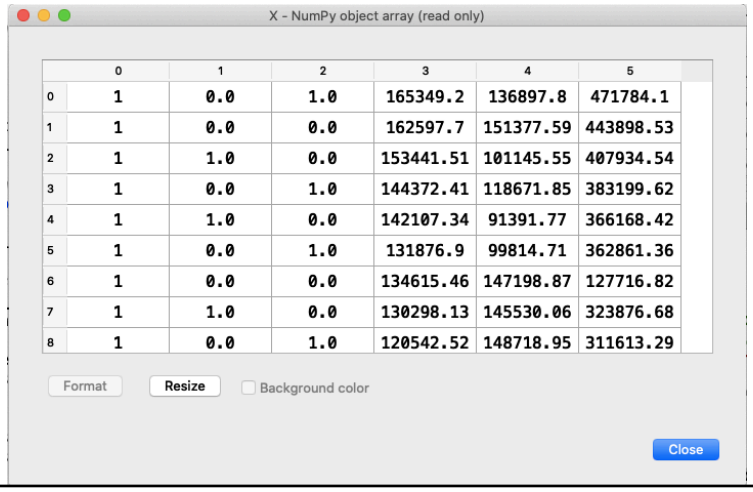


	0	1	2	3	4
0	0.0	1.0	165349.2	136897.8	471784.1
1	0.0	0.0	162597.7	151377.59	443898.53
2	1.0	0.0	153441.51	101145.55	407934.54
3	0.0	1.0	144372.41	118671.85	383199.62
4	1.0	0.0	142107.34	91391.77	366168.42
5	0.0	1.0	131876.9	99814.71	362861.36
6	0.0	0.0	134615.46	147198.87	127716.82
7	1.0	0.0	130298.13	145530.06	323876.68
8	0.0	1.0	120542.52	148718.95	311613.29

68

Multiple Linear Regression – Backward Elimination

- After adding ones



	0	1	2	3	4	5
0	1	0.0	1.0	165349.2	136897.8	471784.1
1	1	0.0	0.0	162597.7	151377.59	443898.53
2	1	1.0	0.0	153441.51	101145.55	407934.54
3	1	0.0	1.0	144372.41	118671.85	383199.62
4	1	1.0	0.0	142107.34	91391.77	366168.42
5	1	0.0	1.0	131876.9	99814.71	362861.36
6	1	0.0	0.0	134615.46	147198.87	127716.82
7	1	1.0	0.0	130298.13	145530.06	323876.68
8	1	0.0	1.0	120542.52	148718.95	311613.29

69

69

Multiple Linear Regression – Backward Elimination

- Use all independent variables

- X_opt = X[:, [0, 1, 2, 3, 4, 5]]
- X_opt = X_opt.astype(np.float64)
- regressor_OLS = sm.OLS(endog = y, exog = X_opt).fit()
- regressor_OLS.summary()

70

Multiple Linear Regression – Backward Elimination

- X_opt = X[:, [0, 1, 2, 3, 4, 5]]

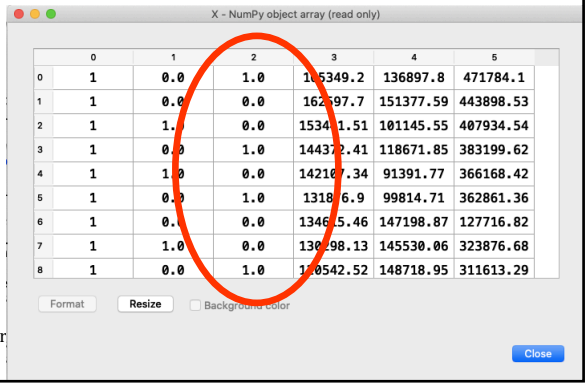
	coef	std err	t	P>|t|	[0.025	0.975]
const	5.013e+04	6884.820	7.281	0.000	3.62e+04	6.4e+04
x1	198.7888	3371.007	0.059	0.953	-6595.030	6992.607
x2	-41.8870	3256.039	-0.013	0.990	-6604.003	6520.229
x3	0.8060	0.046	17.369	0.000	0.712	0.900
x4	-0.0270	0.052	-0.517	0.608	-0.132	0.078
x5	0.0270	0.017	1.574	0.123	-0.008	0.062

Omnibus:	14.782	Durbin-Watson:	1.283
Prob(Omnibus):	0.001	Jarque-Bera (JB):	21.266
Skew:	-0.948	Prob(JB):	2.41e-05
Kurtosis:	5.572	Cond. No.	1.45e+06

71

Multiple Linear Regression – Backward Elimination

- Highest P-value is 0.990 for independent variable 2
- So remove 2nd independent variable from the X



72

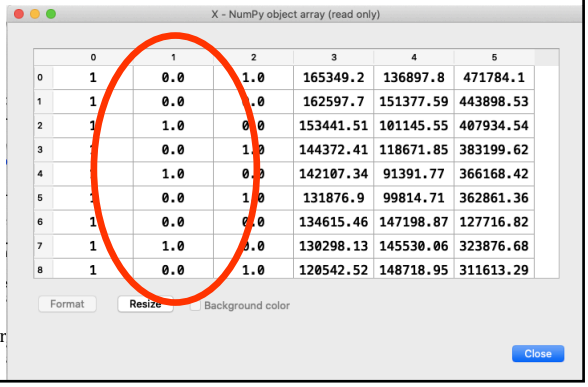
Multiple Linear Regression – Backward Elimination

- X_opt = X[:, [0, 1, 3, 4, 5]]

	coef	std err	t	P>|t|	[0.025	0.975]
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	-6621.821	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

Multiple Linear Regression – Backward Elimination

- Highest P-value is 0.940 for independent variable 1
- So remove 1st independent variable from the X



	0	1	2	3	4	5
0	1	0.0	1.0	165349.2	136897.8	471784.1
1	1	0.0	1.0	162597.7	151377.59	443898.53
2	1	1.0	0.0	153441.51	101145.55	407934.54
3	1	0.0	1.0	144372.41	118671.85	383199.62
4	1	1.0	0.0	142107.34	91391.77	366168.42
5	1	0.0	1.0	131876.9	99814.71	362861.36
6	1	0.0	0.0	134615.46	147198.87	127716.82
7	1	1.0	0.0	130298.13	145530.06	323876.68
8	1	0.0	1.0	120542.52	148718.95	311613.29

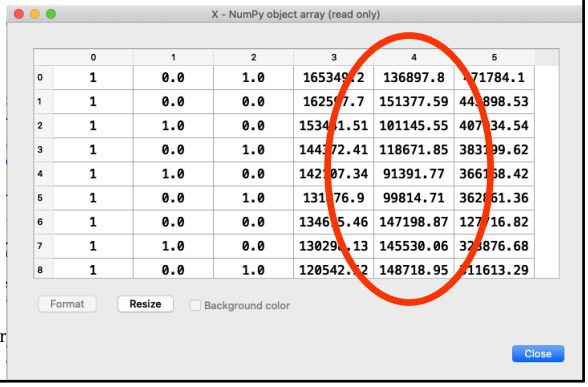
Multiple Linear Regression – Backward Elimination

- X_opt = X[:, [0, 3, 4, 5]]

	coef	std err	t	P>|t|	[0.025	0.975]
const	5.012e+04	6572.353	7.626	0.000	3.69e+04	6.34e+04
x1	0.8057	0.045	17.846	0.000	0.715	0.897
x2	−0.0268	0.051	−0.526	0.602	−0.130	0.076
x3	0.0272	0.016	1.655	0.105	−0.006	0.060

Multiple Linear Regression – Backward Elimination

- Highest P-value is 0.602 for independent variable 4
- So remove 4th independent variable from the X



Multiple Linear Regression – Backward Elimination

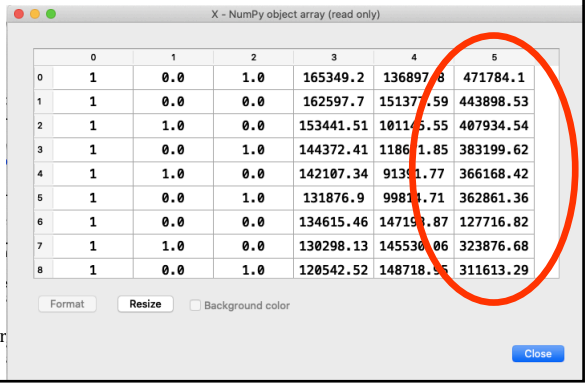
- X_opt = X[:, [0, 3, 5]]

	coef	std err	t	P>|t|	[0.025	0.975]
const	4.698e+04	2689.933	17.464	0.000	4.16e+04	5.24e+04
x1	0.7966	0.041	19.266	0.000	0.713	0.880
x2	0.0299	0.016	1.927	0.060	-0.001	0.061

77

Multiple Linear Regression – Backward Elimination

- Highest P-value is 0.060 for independent variable 5
- So remove 5th independent variable from the X



78

Multiple Linear Regression – Backward Elimination

- X_opt = X[:, [0, 3]]

	coef	std err	t	P>|t|	[0.025	0.975]
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

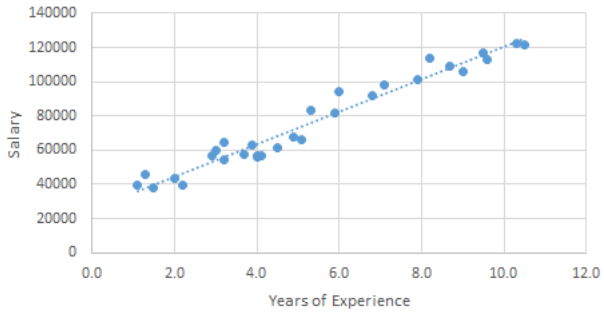
- Highest P-value is 0.000 which is < significance level 0.05 so our model is optimal and ready

Polynomial Regression

- Simple Linear Regression is good if linear relationship is present between dependent variable and independent variable.
- What if the relationship is non-linear or curvilinear??

Polynomial Regression

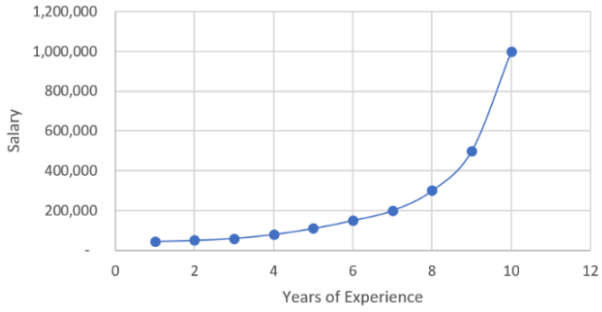
- Simple Linear Regression



81

Polynomial Regression

- Polynomial Regression



	A	B	C
	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

82

Polynomial Regression

- Polynomial regression is a special case of linear regression where we fit a polynomial equation on the data with a curvilinear relationship between the target variable and the independent variables.

83

Polynomial Regression

- 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 Regression

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

84

Polynomial Regression

- Polynomial regression, like linear regression, uses the relationship between the variables x and y to find the best way to draw a line through the data points.

85

Polynomial Regression

- Position_Salaries.csv

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

86

Polynomial Regression

- Importing Libraries

```
 9  # Polynomial Regression
10
11  # Importing the libraries
12  import numpy as np
13  import matplotlib.pyplot as plt
14  import pandas as pd
```


87

Polynomial Regression

- Importing Dataset

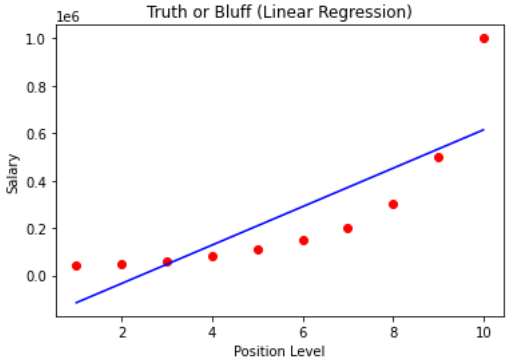
```
16  # Importing the dataset
17  dataset = pd.read_csv('Position_Salaries.csv')
18  X = dataset.iloc[:, 1:-1].values
19  y = dataset.iloc[:, -1].values
```


88

Polynomial Regression

- Training Linear Model

```
21  # Training the Linear Regression model on the whole dataset
22  from sklearn.linear_model import LinearRegression
23  lin_reg = LinearRegression()
24  lin_reg.fit(X, y)
```



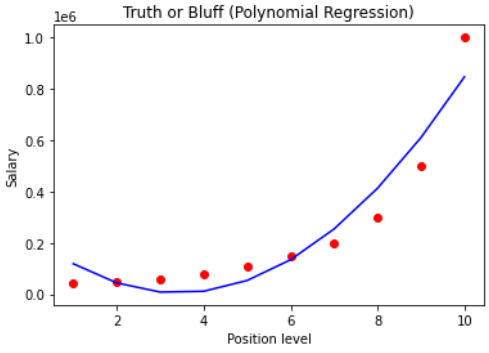
89

89

Polynomial Regression

- Training Polynomial Model

```
26  # Training the Polynomial Regression model on the whole dataset
27  from sklearn.preprocessing import PolynomialFeatures
28  poly_reg = PolynomialFeatures(degree = 2)
29  X_poly = poly_reg.fit_transform(X)
30  lin_reg_2 = LinearRegression()
31  lin_reg_2.fit(X_poly, y)
```



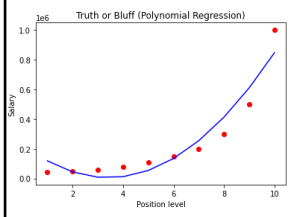
Prof.(Dr.) Dh

90

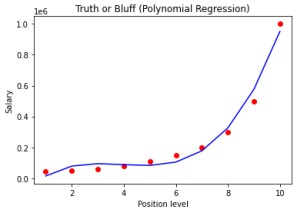
Polynomial Regression

- Training Polynomial Model

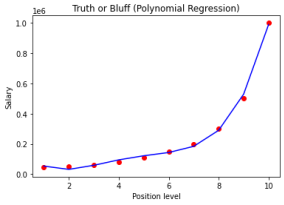
Degree = 2



Degree = 3



Degree = 4



91

Polynomial Regression

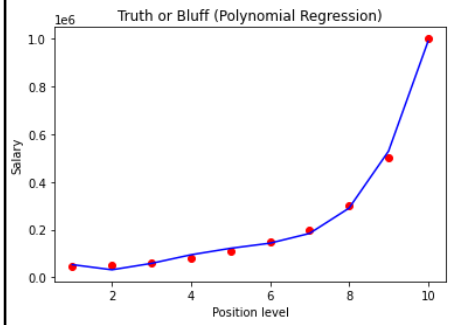
- # Visualizing the Polynomial Regression results (for higher resolution and smoother curve)

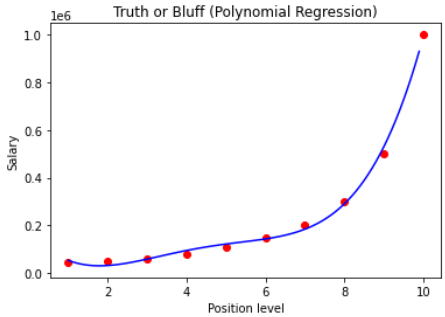
```
49  # Visualising the Polynomial Regression results
50  # (for higher resolution and smoother curve)
51  X_grid = np.arange(min(X), max(X), 0.1)
52  X_grid = X_grid.reshape((len(X_grid), 1))
53  plt.scatter(X, y, color = 'red')
54  plt.plot(X_grid, lin_reg_2.predict(
55      poly_reg.fit_transform(X_grid)), color = 'blue')
56  plt.title('Truth or Bluff (Polynomial Regression)')
57  plt.xlabel('Position level')
58  plt.ylabel('Salary')
59  plt.show()
```

92

Polynomial Regression

- # Visualizing the Polynomial Regression results (for higher resolution and smoother curve)





93

Polynomial Regression

- Prediction – Linear vs Polynomial

```
61  # Predicting a new result with Linear Regression
62  lin_reg.predict([[6.5]])
```

```
In [119]: lin_reg.predict([[6.5]])
Out[119]: array([330378.79])
```

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

```
In [120]:
lin_reg_2.predict(poly_reg.fit_transform([[6.5]]))
Out[120]: array([158862.45])
```


94

Decision Tree Regression

95

Decision Tree Regression

- Decision tree learning is one of the predictive modelling approaches used in machine learning.

96

Decision Tree Regression

- Decision trees are predictive models that use a set of binary rules to calculate a target value.
- Each individual tree has branches, nodes and leaves.

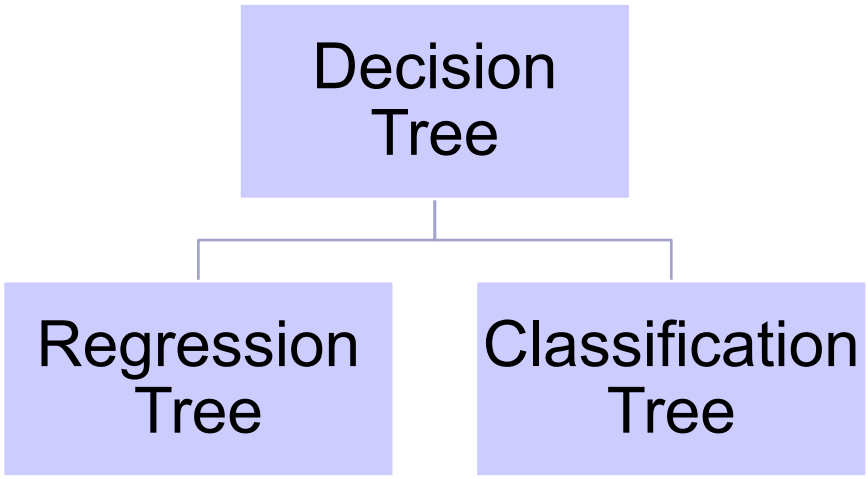
97

Decision Tree Regression

- Tree models where the target variable can take a discrete set of values are called **classification trees** while if the target variable can take continuous values (typically real numbers) are called **regression trees**.

98

Decision Tree Regression



99

Decision Tree Regression

- Basic terminology in decision tree
  - **Root Node** represents entire population or dataset.
  - **Splitting** is a process of dividing a node into two or more sub-nodes.
  - When a sub-node splits into further sub-nodes, then it is called **decision node**.
  - Nodes, which do not split, is called **Leaf/Terminal node**.

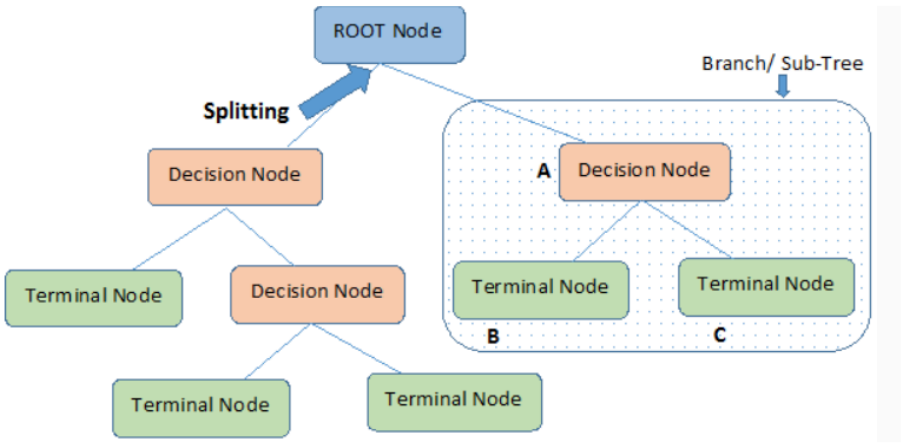
100

Decision Tree Regression

- Basic terminology in decision tree
  - When we remove sub-nodes of a decision node, this process is called **pruning**.
  - A sub section of entire tree is called **branch/subtree**.
  - A node, which is divided into sub-nodes is called **parent node** of sub-nodes whereas sub-nodes are the **child nodes** of parent node.

101

Decision Tree Regression



102

Decision Tree Regression

- A decision tree estimates by asking a series of questions, each question narrowing possible values until the model reaches to the final prediction.

Decision Tree Regression

- The order of the question as well as their content are being determined by the model.
- The questions asked are all of True/False type.

Decision Tree Regression

- During training, the model learns relationships between the data and the target variable.
- After the training phase, the decision tree produces a tree, calculating the best questions as well as their order to ask in order to make the most accurate estimates possible.

105

Decision Tree Regression

- Decision tree regression normally use mean squared error (MSE) to decide to split a node in two or more sub-nodes.

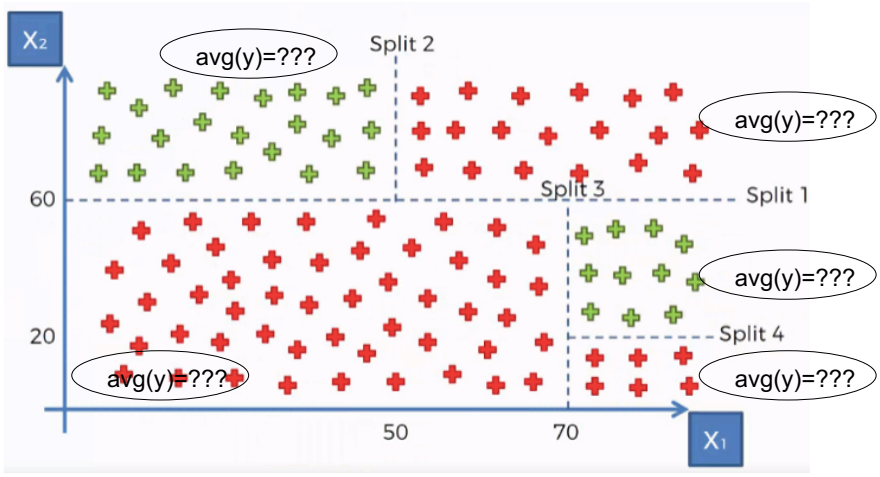
106

Decision Tree Regression

- How to split?
  - Select a variable and the value to split such that the two groups are different from each other.
  - For each variable, for each possible value of that variable see whether it is better (using mean square error).

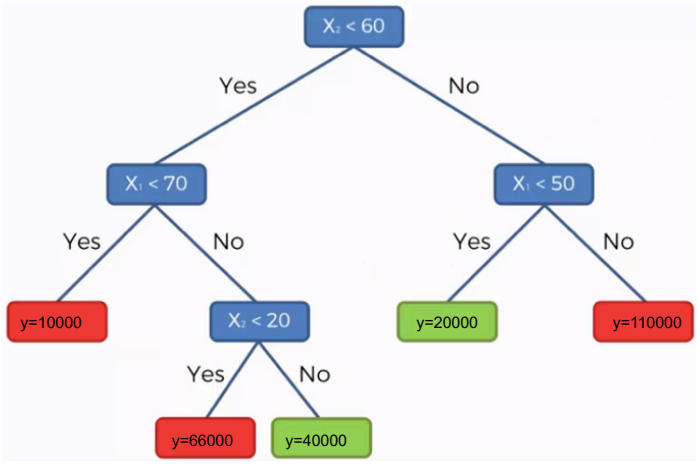
107

Decision Tree Regression



108

Decision Tree Regression



109

Decision Tree Regression

- Position_Salaries.csv

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

110

Decision Tree Regression

- Importing Libraries

```
 9  # Decision Tree Regression
10
11  # Importing the libraries
12  import numpy as np
13  import matplotlib.pyplot as plt
14  import pandas as pd
```


111

Decision Tree Regression

- Importing Dataset

```
16  # Importing the dataset
17  dataset = pd.read_csv('Position_Salaries.csv')
18  X = dataset.iloc[:, 1:-1].values
19  y = dataset.iloc[:, -1].values
```


112

Decision Tree Regression

- Training the Decision Tree Regression model

```
21  # Training the Decision Tree Regression model on the whole dataset
22  from sklearn.tree import DecisionTreeRegressor
23  regressor = DecisionTreeRegressor(random_state = 0)
24  regressor.fit(X, y)
```


Decision Tree Regression

- Predicting a new result

```
26  # Predicting a new result
27  regressor.predict([[6.5]])
```

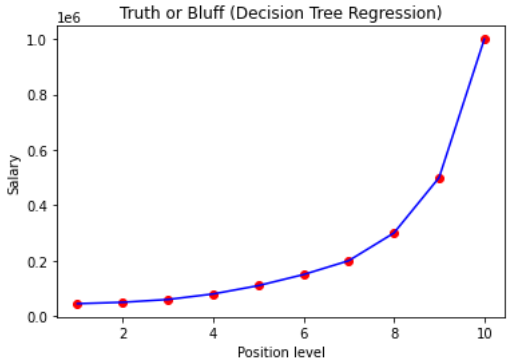
Decision Tree Regression

- Visualizing the Decision Tree Regression results

```
29  # Visualising the Decision Tree Regression results
30  plt.scatter(X, y, color = 'red')
31  plt.plot(X, regressor.predict(X), color = 'blue')
32  plt.title('Truth or Bluff (Decision Tree Regression)')
33  plt.xlabel('Position level')
34  plt.ylabel('Salary')
35  plt.show()
```


Decision Tree Regression

- Visualizing the Decision Tree Regression results



Decision Tree Regression

- BUT,
  - Decision Tree Regression is non-linear and non-continuous

117

Decision Tree Regression

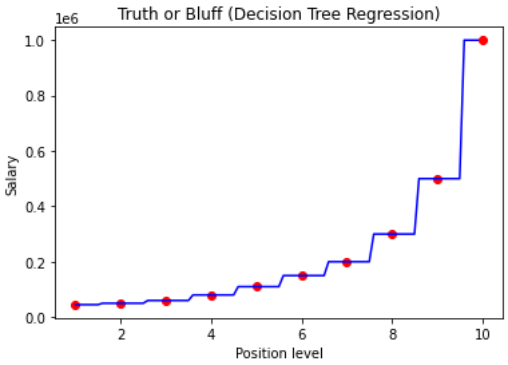
- Visualizing the Decision Tree Regression results (high resolution)

```
37  # Visualising the Decision Tree Regression results (high resolution)
38  X_grid = np.arange(min(X), max(X), 0.1)
39  X_grid = X_grid.reshape((len(X_grid), 1))
40  plt.scatter(X, y, color = 'red')
41  plt.plot(X_grid, regressor.predict(X_grid), color = 'blue')
42  plt.title('Truth or Bluff (Decision Tree Regression)')
43  plt.xlabel('Position level')
44  plt.ylabel('Salary')
45  plt.show()
```


118

Decision Tree Regression

- Visualizing the Decision Tree Regression results (high resolution)



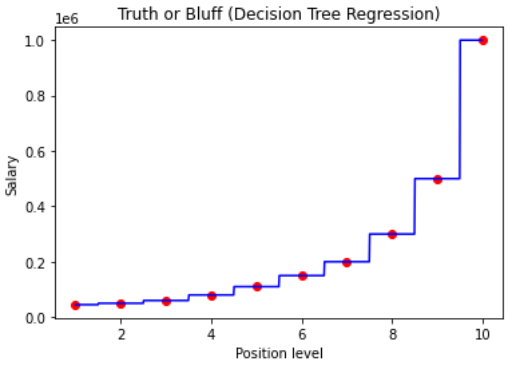
Decision Tree Regression

- Visualizing the Decision Tree Regression results (higher resolution)

```
47  # Visualising the Decision Tree Regression results (higher resolution)
48  X_grid = np.arange(min(X), max(X), 0.01)
49  X_grid = X_grid.reshape((len(X_grid), 1))
50  plt.scatter(X, y, color = 'red')
51  plt.plot(X_grid, regressor.predict(X_grid), color = 'blue')
52  plt.title('Truth or Bluff (Decision Tree Regression)')
53  plt.xlabel('Position level')
54  plt.ylabel('Salary')
55  plt.show()
```

Decision Tree Regression

- Visualizing the Decision Tree Regression results (higher resolution)



Random Forest Regression

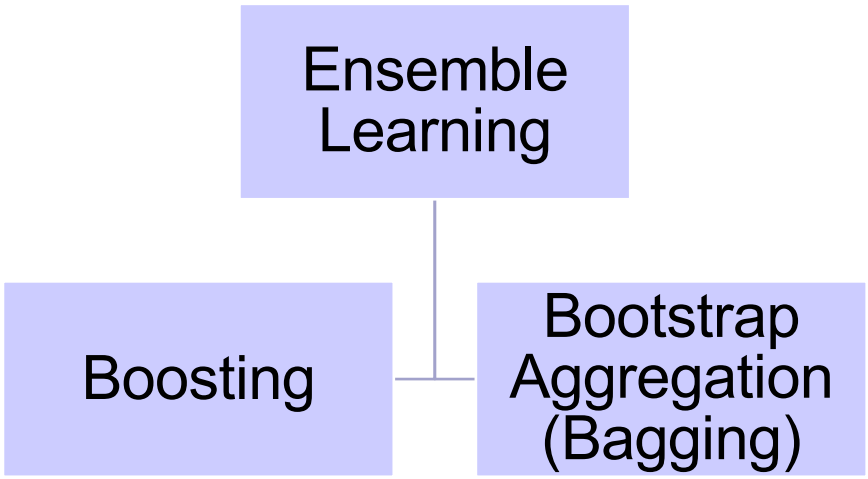
- Random Forest is an **ensemble technique** capable of performing both regression and classification tasks with the use of multiple decision trees.

Random Forest Regression

- An **ensemble** method is a technique that combines the predictions from multiple machine learning algorithms together to make more accurate predictions than any individual model.

123

Random Forest Regression



124

Random Forest Regression

- **Boosting** refers to a set of algorithms that utilize weighted averages to make weak learners into stronger learners.
- Boosting focuses on “teamwork”.
- Each model that runs, dictates what features the next model will focus on.

125

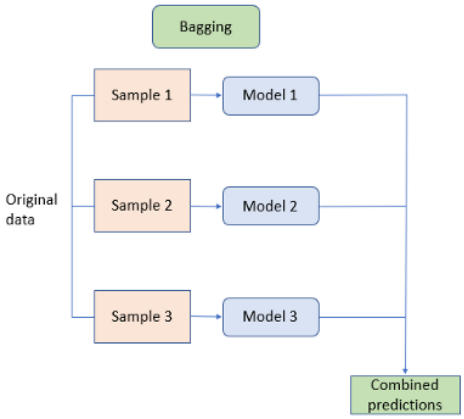
Random Forest Regression

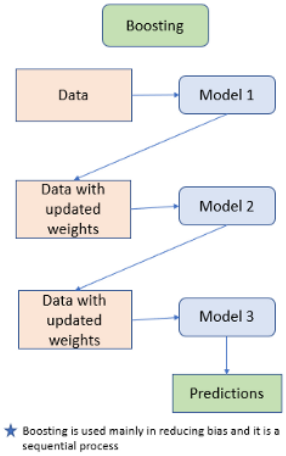
- **Bootstrap Aggregation (Bagging)** refers to random sampling with replacement.
- Bootstrap involves random sampling of small subset of data from the dataset.
- Each model run independently and then aggregates the outputs.

126

Random Forest Regression

- **Bagging Vs Boosting**





127

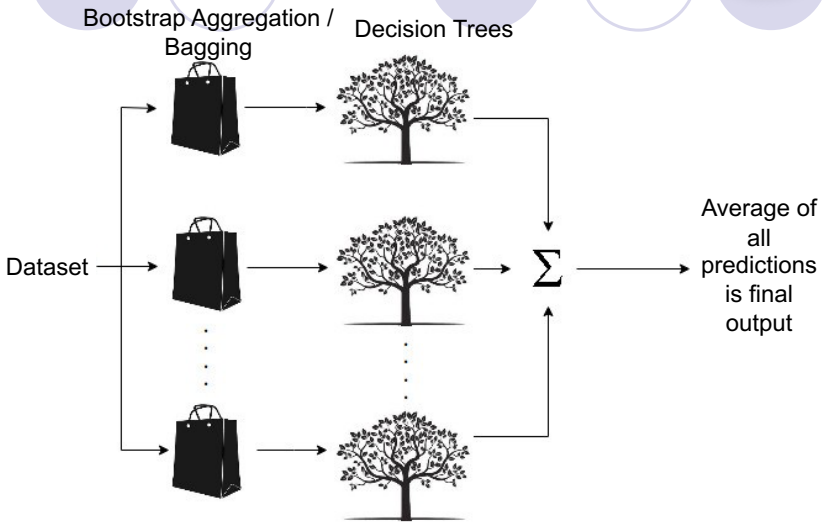
127

Random Forest Regression

- Random forest is a bagging technique and not a boosting technique.

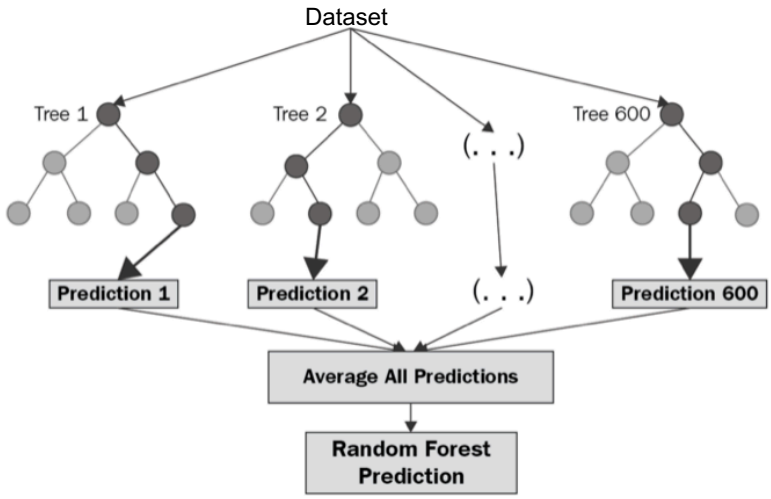
128

Random Forest Regression



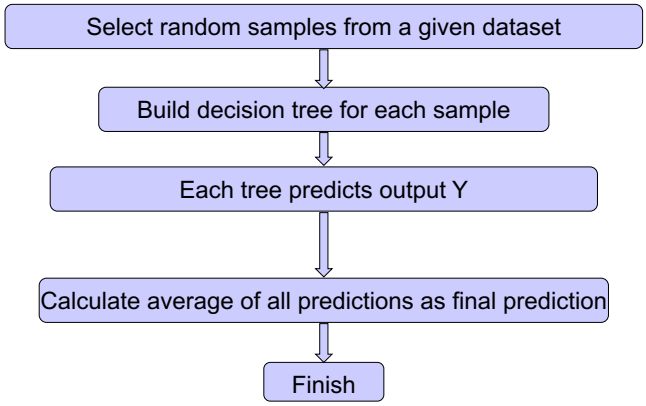
129

Random Forest Regression



130

Random Forest Regression



131

Random Forest Regression

- Position_Salaries.csv

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

132

Random Forest Regression

- Importing Libraries

```
 9  # Random Forest Regression
10
11  # Importing the libraries
12  import numpy as np
13  import matplotlib.pyplot as plt
14  import pandas as pd
```


Random Forest Regression

- Importing Dataset

```
16  # Importing the dataset
17  dataset = pd.read_csv('Position_Salaries.csv')
18  X = dataset.iloc[:, 1:-1].values
19  y = dataset.iloc[:, -1].values
```

Random Forest Regression

- Training the Random Forest Regression model with 10, 100, 300 trees and predict

```
# Training the Random Forest Regression model on the whole dataset
from sklearn.ensemble import RandomForestRegressor

# Generate 10 trees for training
regressor = RandomForestRegressor(n_estimators = 10, random_state = 0)
regressor.fit(X, y)

# Predicting a new result with 10 trees
regressor.predict([[6.5]])

# Generate 100 trees for training
regressor = RandomForestRegressor(n_estimators = 100, random_state = 0)
regressor.fit(X, y)

# Predicting a new result with 100 trees
regressor.predict([[6.5]])

# Generate 300 trees for training
regressor = RandomForestRegressor(n_estimators = 300, random_state = 0)
regressor.fit(X, y)

# Predicting a new result with 300 trees
regressor.predict([[6.5]])
```

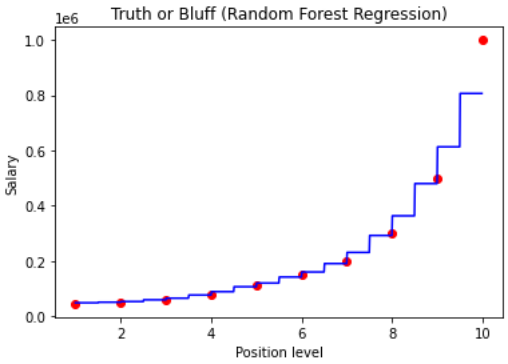

Random Forest Regression

- Visualizing the Random Forest Regression results

```
# Visualising the Random Forest Regression results (higher resolution)
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()
```

Random Forest Regression

- Visualizing the Random Forest Regression results



137

Questions ???

138