**MATPLOTLIB**

**#import libraries**

**import matplotlib.pyplot as plt**

**x = [1,2,3,4,5]**

**y = [2,3,4,5,6]**

**plt.plot(x,y) (or) plt.plot(x,y,’o’,color=’red’)**

**plt.title(“the sample project”)**

**plt.xlabel(“time”)**

**plt.ylabel(“distance”)**

**plt.show()**

**#import libraries**

**import matplotlib.pyplot as plt**

**x = [1,2,3,4,5]**

**y = [2,3,4,5,6]**

**plt.scatter(x,y) (or) plt.scatter(x,y,color=’blue’)**

**plt.title(“the sample project”)**

**plt.xlabel(“time”)**

**plt.ylabel(“distance”)**

**plt.show()**

**-----------------------------------------------------------------------------------------------**

**EXPLORATORY DATA ANALYSIS**

**Data analysis:**

* **Understanding the shape of data -> dataset.shape[]**
* **Median , percentile, no of data points -> dataset.describe()**
* **Data reading using head and tail function -> dataset.head() / df.tail()**

**#import libraries**

**import pandas as pd**

**import numpy as np**

**import matplotlib.pyplot as plt**

**import seaborn as sns**

**#import dataset**

**dataset = pd.read\_csv("50\_startups\_EDA.csv")**

**dataset.shape**

**dataset.shape[0]**

**dataset.shape[1]**

**dataset.describe()**

**dataset.head(10)**

**dataset.tail(10)**

**pd.options.display.max\_rows**

**pd.options.display.max\_columns**

**pd.set\_option(‘display.max\_rows’,500)**

**pd.set\_option(‘display.max\_columns’,500)**

**pd.set\_option(‘display.width’,1000)**

**dataset.info()**

**dataset.dtypes**

**#identify categorical data**

**dataset.nunique() (or) dataset.value\_counts()**

**#check categorical o/p data**

**dataset.iloc[ : , 4].values\_counts()**

**#visualize data**

**Sns.countplot(x=’state’ , data = dataset)**

**#identify total null values**

**dataset.isnull().sum()**

**----------------------------------------------------------------------------------------------**

**Mean -> dataset.fillna(dataset.mean(), inplace=True)**

**Mode -> dataset.fillna(dataset.mode(), inplace=True)**

**Specific data -> dataset[‘variable name’].fillna((dataset[‘variable name’].mean()), inplace=True)**

**-----------------------------------------------------------------------------------------------**

**#filling the missing value using**

**dataset.mean()**

**dataset.mode()**

**dataset[‘R&D spend’].fillna((dataset[‘R&D spend’].mean()), inplace=true)**

**dataset.fillna(dataset.mean(),inplace=True)**

**-----------------------------------------------------------------------------------------------**

**DATA PREPROCESSING**

1. **Missing values -> 1. Regression variable 2. Categorical variable.**
2. **Categorical data (string to numerical data).**
3. **Data split data for train and test.**
4. **Feature scaling (scaling of data).**

**1. Missing values:**

**Strategy:**

* **Mean = average (sum/no of value)**

**Ex: sum - > 5+5+5+5+5 = 25 mean = 25 / 5 = 5**

* **Median = middle value (make it ascending or descending order and find median)**

**Ex: 1, 2, 3, 4, 5, 6, 7, 8, 9 median = 5**

* **Mode = most frequent (find repeated number)**

**Ex: 1,2,3,3,4,5,3,1 mode = 3**

* **Advanced method - > algorithm**

**Null values filled with mean sample code:**

**#import libraries**

**import pandas as pd**

**import numpy as np**

**import matplotlib.pyplot as plt**

**#import dataset**

**dataset = pd.read\_csv("Data\_preprocessing.csv")**

**#finds null values**

**dataset.isnull()**

**#find null values total each variable**

**dataset.isnull().sum()**

**#independent and dependent variable**

**x= dataset.iloc[:,0:3].values**

**y= dataset.iloc[:,3:4].values**

**#import imputer and fill the null values**

**from sklearn.preprocessing import Imputer**

**imputer = Imputer(missing\_values="NaN", strategy = "mean", axis = 0)**

**imputer = imputer.fit(x[:,1:3])**

**x[:,1:3] = imputer.transform(x[:,1:3])**

**2. CATEGORICAL DATA**

**(Nothing but text data into numerical data)**

**Ex:**

**Text data label encoding one hot encoding**

**Apple/0 banana/1 grape/2 orange/3**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Apple** | **0** | **1** | **0** | **0** | **0** |
| **orange** | **3** | **0** | **0** | **0** | **1** |
| **banana** | **1** | **0** | **1** | **0** | **0** |
| **grape** | **2** | **0** | **0** | **1** | **0** |
| **apple** | **0** | **1** | **0** | **0** | **0** |
| **grape** | **2** | **0** | **0** | **1** | **0** |
| **Banana** | **1** | **0** | **1** | **0** | **0** |
| **orange** | **3** | **0** | **0** | **0** | **1** |

**#import libraries**

**import pandas as pd**

**import numpy as np**

**import matplotlib.pyplot as plt**

**#import dataset**

**dataset = pd.read\_csv("Data\_preprocessing.csv")**

**dataset.isnull()**

**dataset.isnull().sum()**

**#independent and dependent variable**

**x= dataset.iloc[:,0:3].values**

**y= dataset.iloc[:,3:4].values**

**#import imputer and fill the null values**

**from sklearn.preprocessing import Imputer**

**imputer = Imputer(missing\_values="NaN", strategy = "mean", axis = 0)**

**imputer = imputer.fit(x[:,1:3])**

**x[:,1:3] = imputer.transform(x[:,1:3])**

**#import label encoder and one hot encoder for categorical data**

**from sklearn.preprocessing import LabelEncoder , OneHotEncoder**

**Label\_x = LabelEncoder ()**

**x[:,0] = Label\_x.fit\_transform(x[:,0])**

**onehotencoder\_x = OneHotEncoder (categorical\_features = [0])**

**x = onehotencoder\_x.fit\_transform(x).toarray ()**

**x = x[:,1:]**

**Label\_y = LabelEncoder ()**

**Y = Label\_y.fit\_transform(y)**

**3. TRAIN AND TEST DATA SPLIT**

**Issues:**

* **More and less data trained model – performance issue**
* **Testing the model**

**Solution:**

**So split data into train & test based on availability**

**(Ex) (80 % & 20 %) or (70 % & 30%) of train and test data**

**#import libraries**

**import pandas as pd**

**import numpy as np**

**import matplotlib.pyplot as plt**

**#import dataset**

**dataset = pd.read\_csv("Data\_preprocessing.csv")**

**dataset.isnull()**

**dataset.isnull().sum()**

**#independent and dependent variable**

**x= dataset.iloc[:,0:3].values**

**y= dataset.iloc[:,3:4].values**

**#import imputer and fill the null values**

**from sklearn.preprocessing import Imputer**

**imputer = Imputer(missing\_values="NaN", strategy = "mean", axis = 0)**

**imputer = imputer.fit(x[:,1:3])**

**x[:,1:3] = imputer.transform(x[:,1:3])**

**#import label encoder and one hot encoder for categorical data**

**from sklearn.preprocessing import LabelEncoder , OneHotEncoder**

**Label\_x = LabelEncoder ()**

**x[:,0] = Label\_x.fit\_transform(x[:,0])**

**onehotencoder\_x = OneHotEncoder (categorical\_features = [0])**

**x = onehotencoder\_x.fit\_transform(x).toarray ()**

**x = x[:,1:]**

**Label\_y = LabelEncoder ()**

**Y = Label\_y.fit\_transform(y)**

**#import sklearn for train and test split data**

**from sklearn.model\_selection import train\_test\_split**

**x\_train, x\_test, y\_train, y\_text = train\_test\_split(x, y, test\_size=.20, random\_state=0)**

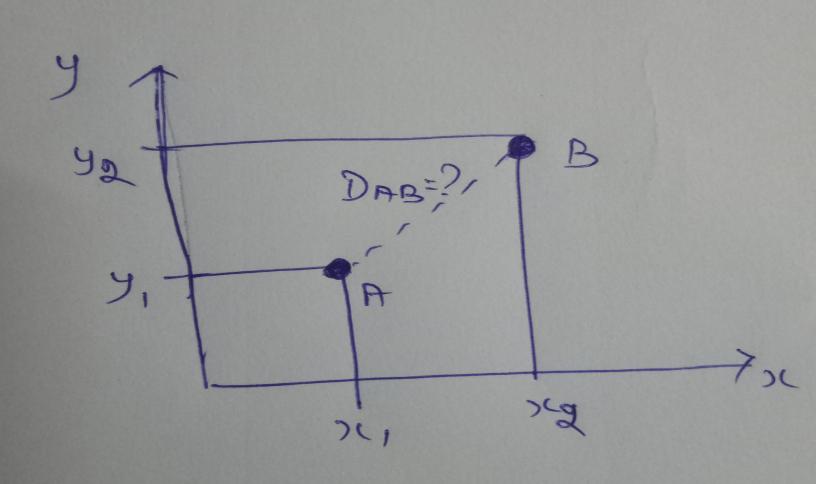
1. **FEATURE SCALING**

**Why?**

* **Dataset is one scale to minimize the distance, less domination**
* **Test computation – less memory (ex) Decision tree**

|  |  |
| --- | --- |
| **AGE** | **SALARY** |
| **62** | **52000** |
| **72** | **100000** |
| **82** | **40000** |

**Ex:**

****

**Euclidean distance**

**DAB = root of((x2 – x1)^2 + (y2 – y1)^2) 🡪 2 Dimension**

**DAB = root of((x2 – x1)^2 + (y2 – y1)^2 + (z2 – z1)^2) 🡪 3 Dimension**

**Ex:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Country** | **Age** | **Salary** | **Purchased** |
| **Australia** | **27** | **52000** | **1** |
| **Russia** | **25** | **49000** | **1** |
| **India** | **30** | **56000** | **0** |
| **Russia** | **29** | **54000** | **0** |
| **India** | **35** | **58000** | **1** |
| **Russia** | **36** | **60000** | **1** |
| **India** | **37** | **61000** | **1** |
| **Australia** | **41** | **67000** | **1** |
| **Australia** | **38** | **65000** | **0** |
| **Russia** | **39** | **66000** | **0** |

**X1 = 25, x2 = 41 -> x2 – x1 = 41 – 25 = 20 -> (x2 – x1)^2 = (20)^2 = 400**

**y1 = 49000, y2 = 67000 -> y2 – y1 = 67000 – 49000 = 18000 -> (y2 – y1)^2 = (18000)^2 = 324000000**

**Age is dominated by salary so we need scaling**

**Solution:**

1. **Standardization (preferred)**

**X std = x – mean(x) / standard deviation (x) => (+ve and –ve values)**

1. **Normalization**

**X norm = x – min(x) / max(x) – min(x) => (+ve values)**

**#import libraries**

**import pandas as pd**

**import numpy as np**

**import matplotlib.pyplot as plt**

**#import dataet**

**dataset = pd.read\_csv("Data\_preprocessing.csv")**

**dataset.isnull()**

**dataset.isnull().sum()**

**#independent and dependent variable**

**x= dataset.iloc[:,0:3].values**

**y= dataset.iloc[:,3:4].values**

**#import imputer and fill the null values**

**from sklearn.preprocessing import Imputer**

**imputer = Imputer(missing\_values="NaN", strategy = "mean", axis = 0)**

**imputer = imputer.fit(x[:,1:3])**

**x[:,1:3] = imputer.transform(x[:,1:3])**

**#import label encoder and one hot encoder for categorical data**

**from sklearn.preprocessing import LabelEncoder , OneHotEncoder**

**Label\_x = LabelEncoder ()**

**x[:,0] = Label\_x.fit\_transform(x[:,0])**

**onehotencoder\_x = OneHotEncoder (categorical\_features = [0])**

**x = onehotencoder\_x.fit\_transform(x).toarray ()**

**x = x[:,1:]**

**Label\_y = LabelEncoder ()**

**Y = Label\_y.fit\_transform(y)**

**#import sklearn for train and test split data**

**from sklearn.model\_selection import train\_test\_split**

**x\_train, x\_test, y\_train, y\_text = train\_test\_split(x, y, test\_size=.20, random\_state=0)**

**#feature scaling**

**from sklearn.preprocessing import StandardScalar**

**sc\_x = StandardScalar()**

**x\_train = sc\_x.fit\_transform(x\_train)**

**x\_test = sc\_x.transform(x\_test)**

**SUPERVISED LEARNING**

1. **Regression:**

**Predict future based on relationship (Ex) time**

**Types:**

* **Simple Linear Regression**
* **Multiple Linear Regression**
* **Polynomial Linear Regression**
* **Decision tree**
* **Random forest**
* **Support vector Regression**

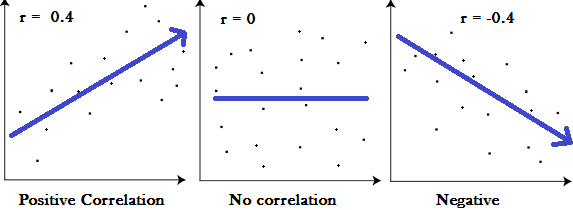
1. **Classification:**

**Grouping or classify data (Ex) grading system grade A,B,C etc**

**Types:**

* **Logistic Regression**
* **Support vector machine(SVM)**
* **K-Nearest neighbors (K-NN)**
* **Naïve Bayes**
* **Kernel SVM**
* **Decision tree classification**
* **Random forest classification**

**SIMPLE LINEAR REGRESSION FUNDAMENTAL**



**Formula:**

**Y = C + M X**

**X -> predictor (present in data) -> Independent variable**

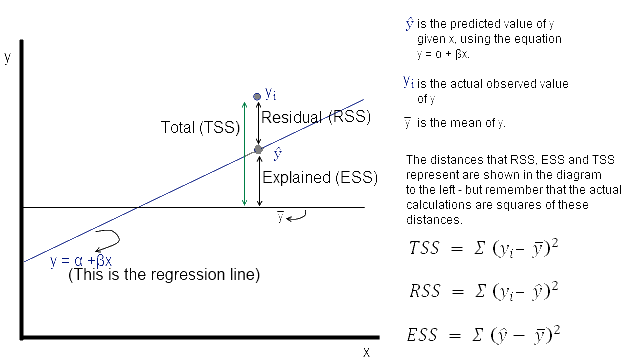
**M -> co-efficient (estimated by regression)**

**C -> intercept (estimated by regression)**

**Y -> predicted value (calculated from C, M, X) -> dependent variable.**

**Best fitted slope:**

**Find sum of squares**



**Error = Actual value – predicted value = y1 –y`**

**#import libraries**

**import pandas as pd**

**import numpy as np**

**import matplotlib.pyplot as plt**

**#import dataset**

**dataset = pd.read\_csv(“salary\_data.csv")**

**#independent and dependent variable**

**x= dataset.iloc[:,0:-1].values**

**y= dataset.iloc[:,1].values**

**#import sklearn for train and test split data**

**from sklearn.model\_selection import train\_test\_split**

**x\_train, x\_test, y\_train, y\_text = train\_test\_split(x, y, test\_size=.30, random\_state=0)**

**#building model**

**from sklearn.linear\_model import LinearRegression**

**reg = LinearRegression()**

**reg.fit(x\_train,y\_train)**

**#prediction**

**Y\_pred = reg.predict(x\_test)**

**#visualize output SLR**

**plt.title(“simple linear regression”)**

**plt.xlabel(“Experience”)**

**plt.ylabel(“salary”)**

**plt.scatter(x\_train,y\_train,color=’red’)**

**plt.plot(x\_train,reg.predict(x\_train),color=’blue’)**

**plt.show()**

**#visualize output SLR with test data**

**plt.title(“simple linear regression with test data”)**

**plt.xlabel(“Experience”)**

**plt.ylabel(“salary”)**

**plt.scatter(x\_test,y\_test,color=’red’)**

**plt.plot(x\_test,reg.predict(x\_test),color=’blue’)**

**plt.show()**

**#visualize output SLR with test data and train slope**

**plt.title(“simple linear regression with test data and train slope”)**

**plt.xlabel(“Experience”)**

**plt.ylabel(“salary”)**

**plt.scatter(x\_test,y\_test,color=’red’)**

**plt.plot(x\_train,reg.predict(x\_train),color=’blue’)**

**plt.show()**

**HANDS ON**

**Using height\_weight.csv dataset**

**#import libraries**

**import pandas as pd**

**import numpy as np**

**import matplotlib.pyplot as plt**

**#import dataset**

**dataset = pd.read\_csv(“height\_weight.csv")**

**#missing value detection**

**dataset.isnull().sum()**

**#independent and dependent variable**

**x= dataset.iloc[:,0:-1].values**

**y= dataset.iloc[:,1].values**

**#missing value detection in an array**

**np.isnan(x).sum()**

**np.isnan(y).sum()**

**#fixing missing values**

**from sklearn.preprocessing import Imputer**

**Imputer = Imputer(missing\_values=”nan”, strategy=”mean”, axis=0)**

**x = imputer.fit\_transform(x)**

**#check if the issue is fixed**

**np.isnan(x).sum()**

**#updated fixing missing values**

**from sklearn.impute import SimpleImputer**

**imp = simpleImputer(missing\_values=np.nan, strategy=”mean”)**

**x = imp.fit\_transform(x)**

**#check if the issue is fixed**

**np.isnan(x).sum()**

**#import sklearn for train and test split data**

**from sklearn.model\_selection import train\_test\_split**

**x\_train, x\_test, y\_train, y\_text = train\_test\_split(x, y, test\_size=.30, random\_state=0)**

**#building model**

**from sklearn.linear\_model import LinearRegression**

**reg = LinearRegression()**

**reg.fit(x\_train,y\_train)**

**#prediction**

**Y\_pred = reg.predict(x\_test)**

**#visualize output SLR**

**plt.title(“Height & weight correlation on train data”)**

**plt.xlabel(“Height”)**

**plt.ylabel(“Weight”)**

**plt.scatter(x\_train,y\_train,color=’red’)**

**plt.plot(x\_train,reg.predict(x\_train),color=’blue’)**

**plt.show()**

**#visualize output SLR with test data**

**plt.title(“Height & weight correlation on test data”)**

**plt.xlabel(“Height”)**

**plt.ylabel(“Weight”)**

**plt.scatter(x\_test,y\_test,color=’red’)**

**plt.plot(x\_test,reg.predict(x\_test),color=’blue’)**

**plt.show()**

**#prediction form user data and test your model**

**my\_ht = [[185]]**

**my\_wt = reg.predict(my\_ht)**

**MULTI LINEAR REGRESSION FUNDAMENTAL**

1. **SLR Vs MLR:**

**SLR -> Y = C + M X**

|  |  |
| --- | --- |
| **Height** | **Weight** |
| **185** | **90** |
| **160** | **62** |
| **173** | **70** |
| **166** | **65** |

**MLR -> y = C + M1X1 + M2X2 + M3X3 + M4X4**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Height** | **Age** | **Sex** | **Location** | **Weight** |
| **185** | **40** | **M** | **India** | **90** |
| **160** | **65** | **M** | **UK** | **62** |
| **173** | **43** | **F** | **Russia** | **70** |
| **166** | **55** | **F** | **Japan** | **65** |

1. **Dummy variable Trap:**

**MLR -> y = C + M1X1 + M2X2 + M3X3 + M4X4**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Height** | **Age** | **Sex** | **Location** | **Weight** |
| **185** | **40** | **M** | **India** | **90** |
| **160** | **65** | **M** | **UK** | **62** |
| **173** | **43** | **F** | **Russia** | **70** |
| **166** | **55** | **F** | **Japan** | **65** |

**Sex and location are categorical data so we need to convert into numbers**

**Dummy variable for sex data:**

|  |  |
| --- | --- |
| **Male** | **Female** |
| **1** | **0** |
| **1** | **0** |
| **0** | **1** |
| **0** | **1** |

**1 -> Male**

**2 -> Female**

**Solution:**

**Removes first column and remaining columns needed**

1. **MLR Model Assumption:**

* **Linearity**
* **Number of lack multicollinearity**
* **Residual**
* **Homoscedasticity**
* **Normally distributed**

1. **Variable selection:**

**Independent Dependent**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Height** | **Age** | **Sex** | **Father** | **Mother** | **Location** | **Weight** |
| **185** | **40** | **M** | **Raj** | **Divya** | **India** | **90** |
| **160** | **65** | **M** | **Krish** | **Priya** | **UK** | **62** |
| **173** | **43** | **F** | **kwan** | **suna** | **Russia** | **70** |
| **166** | **55** | **F** | **kan** | **fema** | **Japan** | **65** |

**Correlation No correlation**

**If you used No correlation data below issues are done**

1. **Garbage in 🡪 Garbage out**
2. **No linearity**
3. **More processing time & resource utilization**

**5.P-value:**

**Probability value**

**Result occurred by chance alone**

**Rule:**

**If P-value is less than or equal to the significance level (SL), reject the null hypothesis. (i.e) P < SL = Reject H0 (or) Accept Ha**

**Inference:**

**Generally SL = 0.05(5%) or 0.01(1%)**

**P > SL = H0 (MLR: no relation)**

**P < SL = Ha (MLR: relation)**

**Null Hypothesis (H0):**

**H0 of a multiple regression is that there is no relationship between the x-variable & y-variable**

**Alternate Hypothesis (Ha):**

**H1 (or) Ha of a multiple regression is that there is relationship between the x-variable & y-variable**

**BUILDING MODEL (Tuning and Variable Selection)**

**1. What is building model?**

**Same algorithm & same data is used by two colleagues, but one person has good performance & another one has bad performance in the model.**

**2. Do we input all dependent variable?**

**3. How we are going fine tune model?**

* **All in**
* **Stepwise regression**

1. **Backward Elimination**
2. **Forward Elimination**
3. **Bi-directional Elimination**

* **Score comparison**

**Backward Elimination:**

**Step 1:**

**Select significance level (Ex) SL=0.05**

**Step 2:**

**Build the model with all variables**

**Step 3:**

**Check the highest P-values variable: if P > SL , move to step 4 & if P < SL move to step 5.**

**Step 4:**

**Remove the highest P-value variable & go to step 3**

**Step 5:**

**Fit the model with the finalized values. (Note: In this scenario all the variable P-value will be less than SL)**

**Iteration 1**

**Independent Dependent**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Height** | **Age** | **Sex** | **Father** | **Mother** | **Weight** |
| **185** | **40** | **M**  **(1)** | **Raj**  **(0.343)** | **Divya**  **(0.635)** | **90** |
| **160** | **65** | **M**  **(1)** | **Krish**  **(0.473)** | **Priya**  **(0.894)** | **62** |
| **173** | **43** | **F**  **(0)** | **Kwan**  **(0.473)** | **Suna**  **(0.736)** | **70** |
| **166** | **55** | **F**  **(0)** | **Kan**  **(0.983)** | **Fema**  **(0.943)** | **65** |

**P-value: 0.032 0.036 0.042 0.68 0.847**

**Iteration 2**

|  |  |  |  |
| --- | --- | --- | --- |
| **Height** | **Age** | **Sex** | **Father** |
| **185** | **40** | **M**  **(1)** | **Raj**  **(0.343)** |
| **160** | **65** | **M**  **(1)** | **Krish**  **(0.473)** |
| **173** | **43** | **F**  **(0)** | **Kwan**  **(0.473)** |
| **166** | **55** | **F**  **(0)** | **Kan**  **(0.983)** |

**P-value: 0.032 0.036 0.042 0.68**

**Iteration 3**

|  |  |  |
| --- | --- | --- |
| **Height** | **Age** | **Sex** |
| **185** | **40** | **M**  **(1)** |
| **160** | **65** | **M**  **(1)** |
| **173** | **43** | **F**  **(0)** |
| **166** | **55** | **F**  **(0)** |

**P-value: 0.032 0.036 0.042**

**Now less than all variables P-values under 0.05**

**Forward Elimination:**

**Step 1:**

**Select significance level (Ex) SL=0.05**

**Step 2:**

**Fit all variables as simple Regression(i.e one independent variables to one dependent variables). Select the variable with Lowest P-value.**

**Step 3:**

**Fit into the model with selected variable and combinations of additional variable from the dataset.**

**Step 4:**

**Select the variable with the lowest P-value. if P < SL go to step 3.(if you have finished all iteration with all combinations) or if P > SL go to step 5.**

**Step 5:**

**Remove the variables that was added in the step 3 and fit into model.**

**Iteration 1**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Height** | **Weight** |  | **Age** | **Weight** |  | **Father** | **Weight** |
| **185** | **90** |  | **40** | **90** |  | **Raj**  **(0.343)** | **90** |
| **160** | **62** |  | **65** | **62** |  | **Krish**  **(0.473)** | **62** |
| **173** | **70** |  | **43** | **70** |  | **Kwan**  **(0.473)** | **70** |
| **166** | **65** |  | **55** | **65** |  | **Kan**  **(0.983)** | **65** |

**P-value: 0.034 P-value: 0.045 P-value: 0.734**

**Iteration 2**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Height** | **Age** | **Weight** |  | **Height** | **Father** | **Weight** |
| **185** | **40** | **90** |  | **185** | **Raj**  **(0.343)** | **90** |
| **160** | **65** | **62** |  | **160** | **Krish**  **(0.473)** | **62** |
| **173** | **43** | **70** |  | **173** | **Kwan**  **(0.473)** | **70** |
| **166** | **55** | **65** |  | **166** | **Kan**  **(0.983)** | **65** |

**P-value: 0.045 P-value: 0.734**

**Iteration 3**

|  |  |  |  |
| --- | --- | --- | --- |
| **Height** | **Age** | **Father** | **Weight** |
| **185** | **40** | **Raj**  **(0.343)** | **90** |
| **160** | **65** | **Krish**  **(0.473)** | **62** |
| **173** | **43** | **Kwan**  **(0.473)** | **70** |
| **166** | **55** | **Kan**  **(0.983)** | **65** |

**P-value: 0.693 (its greater than 0.05 so choose before iteration )**

|  |  |  |
| --- | --- | --- |
| **Height** | **Age** | **Weight** |
| **185** | **40** | **90** |
| **160** | **65** | **62** |
| **173** | **43** | **70** |
| **166** | **55** | **65** |

**P-value: 0.045**

**Bi-Directional Elimination:**

**Step 1:**

**Select significance level for forward selection (SLenter) and backward elimination (SLstay); (Ex) SLenter (FS) = 0.05 and SLstay (BE) = 0.05**

**Step 2:**

**Perform the forward selection process of step 2 i.e fitting all the variables**

**Step 3:**

**Perform the forward selection process(Note: always the outcome variables must have P < SLenter to enter)**

**Step 4:**

**Perform the backward selection process(Note: always the outcome variables must have P < SLstay to stay)**

**Step 5:**

**Perform until No new variables can be added or NO old variables removed**

**Hands on**

**Multiple linear regression (Data processing & Building model)**

**#import libraries**

**import pandas as pd**

**import numpy as np**

**import matplotlib.pyplot as ptd**

**#import dataset**

**dataset = pd.read\_csv(“50\_startups.csv")**

**#missing and null value detection**

**dataset.isnull().sum()**

**#independent and dependent variable**

**x= dataset.iloc[:,:-1].values**

**y= dataset.iloc[:,4].values**

**#import label encoder and one hot encoder for categorical data**

**from sklearn.preprocessing import LabelEncoder , OneHotEncoder**

**le\_x = LabelEncoder ()**

**x[:,3] = le\_x.fit\_transform(x[:,3])**

**one\_x = OneHotEncoder (categorical\_features = [3])**

**x = one\_x.fit\_transform(x).toarray () # dummy variable issue**

**#dummy variable trap**

**x = x[:,1:]**

**#import sklearn for train and test split data**

**from sklearn.model\_selection import train\_test\_split**

**x\_train, x\_test, y\_train, y\_text = train\_test\_split(x, y, test\_size=.30, random\_state=0)**

**#building model**

**from sklearn.linear\_model import LinearRegression**

**regress = LinearRegression()**

**regress.fit(x\_train,y\_train)**

**#prediction**

**Y\_pred = regress.predict(x\_test)**

**#model performance – best fitted line line error rate**

**regress.score(x\_train,y\_train) # check scores its measured from 0 to 1 if its ‘0’ means worst fit or else ‘1’ means accuracy.**

**regress.score(x\_test,y\_test)**

**MLR WITH BACKWARD ELIMINATION & P-VALUE**

**Above code continued…**

**#lets build the model using backward elimination**

**#now we will use linear regression from statsmodel library**

**import statsmodels.formula.api as sm**

**x = np.append(arr = x, values = np.ones(shape = (50,1), dtype = int) , axis = 1) # we need constant but here country there so using numpy and created**

**#iteration1**

**x\_ov = x[:,[0,1,2,3,4,5]] #x\_ov = x optimum values # step 1**

**regress\_ols = sm.OLS(endog = y , exog = x\_ov).fit() #step 2**

**regress\_ols.summary()**

**#iteration2**

**x\_ov = x[:,[0,1,3,4,5]] #x\_ov = x optimum values # step 1**

**regress\_ols = sm.OLS(endog = y , exog = x\_ov).fit() #step 2**

**regress\_ols.summary()**

**#iteration3**

**x\_ov = x[:,[0,3,4,5]] #x\_ov = x optimum values # step 1**

**regress\_ols = sm.OLS(endog = y , exog = x\_ov).fit() #step 2**

**regress\_ols.summary()**

**#iteration4**

**x\_ov = x[:,[0,3,5]] #x\_ov = x optimum values # step 1**

**regress\_ols = sm.OLS(endog = y , exog = x\_ov).fit() #step 2**

**regress\_ols.summary()**

**#iteration5**

**x\_ov = x[:,[0,3]] #x\_ov = x optimum values # step 1**

**regress\_ols = sm.OLS(endog = y , exog = x\_ov).fit() #step 2**

**regress\_ols.summary()**

**Hands On**

**Variable impact on model performance**

**Above code continued…**

**#MLR using sklearn**

**X\_ovc = x\_ov[:,1:]**

**X\_ov\_train, x\_ov\_test, y\_ov\_train , y\_ov\_test = train\_test\_split(x\_ovc , y, test\_size = .30 , random\_state = 0)**

**regress\_ov1 = LinearRegression()**

**regress\_ov1.fit(x\_ov\_train, y\_ov\_train)**

**regress\_ov1.score(x\_ov\_train, y\_ov\_train)**

**regress\_ov1.score(x\_ov\_test, y\_ov\_test)**

**#iteration2**

**x\_ov = x[:,[0,1,3,4,5]] #x\_ov = x optimum values # step 1**

**regress\_ols = sm.OLS(endog = y , exog = x\_ov).fit() #step 2**

**regress\_ols.summary()**

**#MLR using sklearn**

**X\_ovc = x\_ov[:,1:]**

**X\_ov\_train, x\_ov\_test, y\_ov\_train , y\_ov\_test = train\_test\_split(x\_ovc , y, test\_size = .30 , random\_state = 0)**

**regress\_ov2 = LinearRegression()**

**regress\_ov2.fit(x\_ov\_train, y\_ov\_train)**

**regress\_ov2.score(x\_ov\_train, y\_ov\_train)**

**regress\_ov2.score(x\_ov\_test, y\_ov\_test)**

**#iteration3**

**x\_ov = x[:,[0,3,4,5]] #x\_ov = x optimum values # step 1**

**regress\_ols = sm.OLS(endog = y , exog = x\_ov).fit() #step 2**

**regress\_ols.summary()**

**#MLR using sklearn**

**X\_ovc = x\_ov[:,1:]**

**X\_ov\_train, x\_ov\_test, y\_ov\_train , y\_ov\_test = train\_test\_split(x\_ovc , y, test\_size = .30 , random\_state = 0)**

**regress\_ov3 = LinearRegression()**

**regress\_ov3.fit(x\_ov\_train, y\_ov\_train)**

**regress\_ov3.score(x\_ov\_train, y\_ov\_train)**

**regress\_ov3.score(x\_ov\_test, y\_ov\_test)**

**#iteration4**

**x\_ov = x[:,[0,3,5]] #x\_ov = x optimum values # step 1**

**regress\_ols = sm.OLS(endog = y , exog = x\_ov).fit() #step 2**

**regress\_ols.summary()**

**#MLR using sklearn**

**X\_ovc = x\_ov[:,1:]**

**X\_ov\_train, x\_ov\_test, y\_ov\_train , y\_ov\_test = train\_test\_split(x\_ovc , y, test\_size = .30 , random\_state = 0)**

**regress\_ov4 = LinearRegression()**

**regress\_ov4.fit(x\_ov\_train, y\_ov\_train)**

**regress\_ov4.score(x\_ov\_train, y\_ov\_train)**

**regress\_ov4.score(x\_ov\_test, y\_ov\_test)**

**#iteration5**

**x\_ov = x[:,[0,3]] #x\_ov = x optimum values # step 1**

**regress\_ols = sm.OLS(endog = y , exog = x\_ov).fit() #step 2**

**regress\_ols.summary()**

**#MLR using sklearn**

**X\_ovc = x\_ov[:,1:]**

**X\_ov\_train, x\_ov\_test, y\_ov\_train , y\_ov\_test = train\_test\_split(x\_ovc , y, test\_size = .30 , random\_state = 0)**

**regress\_ov5 = LinearRegression()**

**regress\_ov5.fit(x\_ov\_train, y\_ov\_train)**

**regress\_ov5.score(x\_ov\_train, y\_ov\_train)**

**regress\_ov5.score(x\_ov\_test, y\_ov\_test)**

**Hands On**

**Bit.ly/MLMLRPR**