**Multiple linear Regression**

Multiple linear Regression means you have more than one feature column in regression problem.

**Guideline for Multiple linear Regression:**

1. Check for Missing values using
2. Check for Outliers - Only on Feature Columns if performing Inferential Stats else perform for all columns

(Never check outliers for Label Column if performing Supervised Learning)

1. Inferential Stats

- Ensure your data is Complete

- Ensure your data is Strictly Numeric

1. Separate data as features and label and store the same in the form on Numpy Array(Inferential Stats)
2. Split our data as Training Set and Testing Set where Training Set is used for training/convergence purpose(Inferential Stats)

* (whereas Testing set is used for Quality Check Purpose)
* Decide the Split Ratio (65%:35%)(80:20)
* Implement the same using Sci-kit Learn package

1. Build the model:

* Regression Algo ------> Linear Regression
* y = mx + b(Slope Intercept Formula)
* profit = b0(California) + b1(Florida) + b2(New York)+ b3(R&D Spend) + b4(AdminSpend) + b5(MarketingSpend) + intercept
* Goal of this algo is to derive values of b0,b1,b2,b3,b4,b5,intercept based on the historical data !

1. Train the model | Converging Training Set to the Algorithm
2. fit(features,label)
3. **Quality Check (Guideline)**
4. Ensure your model that is converged is a Generalized Model

- **Generalized model is a model that performs well with Known and Unknown data**

-Technique: **Accuracy(Test Data) > Accuracy(Train Data)** ------> Model is **Generalized Model**

1. Ensure the Model's accuracy score must be greater than or equal to CL score (\*CL = 1 - SL)

- Technique: **Accuracy(Test Data) >= CL**

1. Deployment Test

**Use-case:** To create a model that can **predict the profit of the company** based on **company's location** and **company's spending pattern**.

|  |  |  |
| --- | --- | --- |
|  |  | data = pd.read\_csv('50\_Startups.csv') |
|  |  | data.info()  <class 'pandas.core.frame.DataFrame'>  RangeIndex: 50 entries, 0 to 49  Data columns (total 5 columns):  # Column Non-Null Count Dtype  --- ------ -------------- -----  0 R&D Spend 50 non-null float64  1 Administration 50 non-null float64  2 Marketing Spend 50 non-null float64  3 State 50 non-null object  4 Profit 50 non-null float64  dtypes: float64(4), object(1)  memory usage: 2.1+ KB |
|  |  | data.State.unique()  array(['New York', 'California', 'Florida'], dtype=object) |
|  | Check for Missing values using | data.isna().sum()  R&D Spend 0  Administration 0  Marketing Spend 0  State 0  Profit 0  dtype: int64 |
|  | Check for Outliers - Only on Feature Columns if performing Inferential Stats else perform for all columns  Never check outliers for Label Column if performing Supervised Learning | data.describe() |
|  | Inferential Stats  1. Ensure your data is Complete  2. Ensure your data is Strictly Numeric | data.head(5) |
|  |  | finalData = pd.concat([ pd.get\_dummies(data['State']) , data.iloc[:,[0,1,2,4]]] , axis = 1)  finalData.head()   |  | **California** | **Florida** | **New York** | **R&D Spend** | **Administration** | **Marketing Spend** | **Profit** | | --- | --- | --- | --- | --- | --- | --- | --- | | 0 | 0 | 0 | 1 | 165349.20 | 136897.80 | 471784.10 | 192261.83 | | 1 | 1 | 0 | 0 | 162597.70 | 151377.59 | 443898.53 | 191792.06 | | 2 | 0 | 1 | 0 | 153441.51 | 101145.55 | 407934.54 | 191050.39 | | 3 | 0 | 0 | 1 | 144372.41 | 118671.85 | 383199.62 | 182901.99 | | 4 | 0 | 1 | 0 | 142107.34 | 91391.77 | 366168.42 | 166187.94 | |
|  | Inferential Stats  1.Seperate data as features and label and store the same in the form on Numpy Array  2. Split our data as Training Set and Testing Set where Training Set is used for training/convergence purpose  whereas Testing set is used for Quality Check Purpose. | 1. Seperate data as features and label and store the same in the form on Numpy Array  features = finalData.iloc[:,0:6].values  label = finalData.iloc[:,[6]].values |
|  | # 2. Split our data as Training Set and Testing Set where Training Set  Step1: Decide the Split Ratio (65%:35%)(80:20)  Step2: Implement the same using Sci-kit Learn package  X\_train --> Training Features  y\_train --> Training Label  X\_test --> Testing Features  y\_test --> Testing Label  For training ----------> X\_train,y\_train  For QA ----------------> X\_test,y\_test | from sklearn.model\_selection import train\_test\_split  X\_train,X\_test,y\_train,y\_test = train\_test\_split(features,  label,  test\_size=0.2,  random\_state=10) |
|  | Build the model  Regression Algo ------> Linear Regression  y = mx + b(Slope Intercept Formula)  profit = b0(California) + b1(Florida) + b2(New York)+ b3(R&D Spend) + b4(AdminSpend) + b5(MarketingSpend) + intercept  Goal of this algo is to derive values of b0,b1,b2,b3,b4,b5,intercept based on the historical data ! | from sklearn.linear\_model import LinearRegression  model = LinearRegression() |
|  | Train the model | Converging Training Set to the Algorithm  # fit(features,label) | model.fit(X\_train,y\_train) |
|  |  | model.intercept\_  array([50001.73604086]) |
|  |  | model.coef\_  array([[ 8.41023126e+01, 6.95447747e+02, -7.79550060e+02,  8.05859453e-01, -1.79706621e-02, 2.28153524e-02]]) |
|  | Profit = 8.41023126e+01(California) + 6.95447747e+02(Florida) - 7.79550060e+02(New York) + 8.05859453e-01 (RDSpend) - 1.79706621e-02(Admin) + 2.28153524e-02(MarkSpend) + 50001.73604086 |  |
|  | **Quality Check (Guideline)**  1-Ensure your model that is converged is a Generalized Model  - **Generalized model is a model that performs well with Known and Unknown data**  -Technique: **Accuracy(Test Data) > Accuracy(Train Data)** ------> Model is **Generalized Model**  2-Ensure the Model's accuracy score must be greater than or equal to CL score (\*CL = 1 - SL)  -Technique: **Accuracy(Test Data) >= CL** |  |
|  | 1. Ensure your model that is converged is a Generalized Model  #In scikit learn you can get the accuracy score using score function | testAccuracy = model.score(X\_test,y\_test)  trainAccuracy = model.score(X\_train,y\_train)  print("Test Score is {} and train Score is {}".format(testAccuracy,trainAccuracy))  **Test Score** is 0.9901105113397691 and **train Score** is 0.9385918220043519 |
|  |  | As observed above, testScore > trainScore , thus the model is generalized !!! |
|  | SL = 0.05  CL = 0.95  2. Ensure the Model's accuracy score must be greater than or equal to CL score (\*CL = 1 - SL)  Technique : Accuracy(Test Data) >= CL  Since my testScore is greater than CL, model passed the Quality Check ! |  |
|  | **Deployment Test – method one** | rdSpend = float(input("Enter R&D Spend: "))  admSpend = float(input("Enter Administration Spend: "))  markSpend = float(input("Enter Marketing Spend: "))  state = input("Enter State: ")  refState = ['California', 'Florida','New York']  if state in refState:  if state == "California":  stateDummy = np.array([[1,0,0]])  elif state == "Florida":  stateDummy = np.array([[0,1,0]])  else:  stateDummy = np.array([[0,0,1]])  finalFeatures = np.concatenate((stateDummy, np.array([[rdSpend,admSpend,markSpend]])) , axis=1)  profit = model.predict(finalFeatures)[0][0]  print("Predicted profit is $ {}".format(profit))  else:  print("Model can't predict profit for the given {} state".format(state))  **Enter R&D Spend: 234567**  **Enter Administration Spend: 45678**  **Enter Marketing Spend: 76543**  **Enter State: Florida**  **Predicted profit is $ 240650.70978691286** |
|  | For Deployment Test, method 2 we need to do these before train/test split:  -we handle Categorical Data with:  Sci-kit Package/ OneHotEncoder.  - do FeatureScaling with StandardScaler.   1. Categorical Data Handling   Using Sci-kit Package | from sklearn.preprocessing import OneHotEncoder  ohe = OneHotEncoder(sparse=False)  fState = ohe.fit\_transform( np.array(data['State']).reshape(-1,1) )  features = np.concatenate( [fState , np.array(data.iloc[:,[0,1,2]])] , axis = 1)  features |
|  | #FeatureScaling | from sklearn.preprocessing import StandardScaler  sc = StandardScaler()  features = sc.fit\_transform(features) |
|  | Deployment Test / method 2 | rdSpend = float(input("Enter R&D Spend: "))  admSpend = float(input("Enter Administration Spend: "))  markSpend = float(input("Enter Marketing Spend: "))  state = input("Enter State: ")  refState = ['California', 'Florida','New York']  if state in refState:  **stateDummy = ohe.transform(np.array([[state]]))**    finalFeatures = np.concatenate((stateDummy, np.array([[rdSpend,admSpend,markSpend]])) , axis=1)  **stdScaleFeatures = sc.transform(finalFeatures)**  profit = model.predict(**stdScaleFeatures**)[0][0]  print("Predicted profit is $ {}".format(profit))  else:  print("Model can't predict profit for the given {} state".format(state)) |
|  |  | data['State'].unique()  array(['New York', 'California', 'Florida'], dtype=object) |
|  |  | [0.0000000e+00, 0.0000000e+00, 1.0000000e+00, 1.6534920e+05,1.3689780e+05, 4.7178410e+05]  California Florida New York rdSpend admSpend markSpend |