**Feature Engineering**

Identifying the most relevant features that has important information which helps the algorithm to understand the data.

Feature Engineering is applicable for Supervised Learning.

Primary Goal 🡺 Select relevant features based on:

1. Label
2. Algorithm
3. Feature Columns

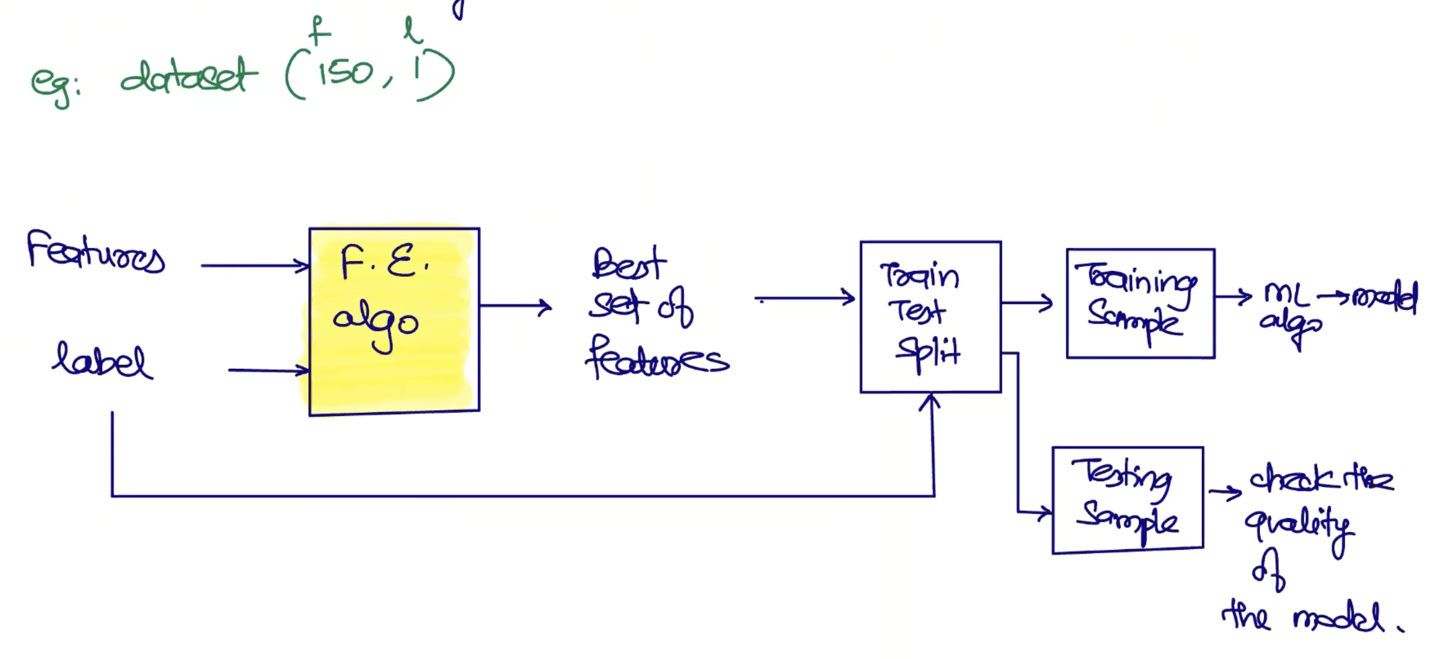
Such that you can achieve **Optimal Accuracy**.

We have 4 types of feature Engineering:

1. Feature extraction
2. Feature Selection (Feature Elimination)
3. Feature Scaling 🡺 1)StandardScaler 2)MinMaxScaler
4. Feature Encoding 🡺 1)Label Encoding 2)OHE

**Feature Selection (Feature Elimination):**

It’s all about selecting relevant features from the total set of features.



Some of Feature Selection algorithms:

1. Correlation analysis
2. Backward Elimination Technique Using OLS 🡺 Only applicable for Regression Use-Case
3. RFE [Recursive Feature Elimination]
4. SBM [ Select By Model]
5. ANOVA [ Analysis Of Variance]

**Example**

Use-case: An investor company has assigned you a project. The goal of the project is to create a model that can predict the profit of the company based on Company's Spending Pattern and Company's Location.

|  |  |  |
| --- | --- | --- |
|  |  |  |
| 1 | Upload dataset | 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 |
| 2 | Make the data compatible  Make data numerical | 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 | |
| 3 | Let’s figure out the features are relevant to the use-case or not? ( in terms of label or ML algorithms or other features columns) 🡺 **so we need feature selection algorithms** | features = finalData.iloc[:,:-1].values  label = finalData.iloc[:,[-1]].values |
| 4 |  | **Feature Engineering Track** |
| 5 | **Feature Engineering Track**  **Method 1. Correlation Analysis** | finalData.corr() |
|  |  | **Rule:**  Select those features who have corr val greater than 50% (0.5)  Features Selected: R&d, Mkg  Features Eliminated: California, Florida, NewYork, Administration |

**Method 2. Backward Elimination Technique using OLS(Ordinary Least Square)**

* Only applicable for Regression Problems

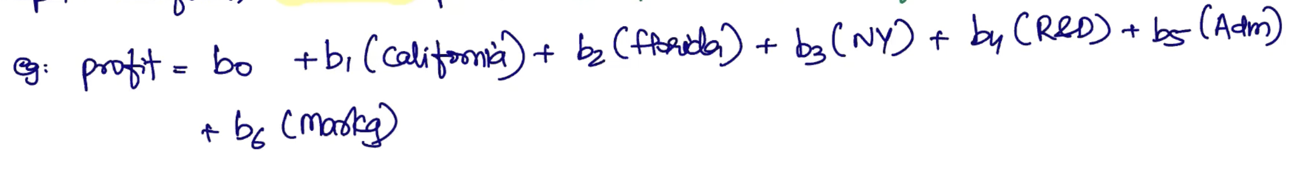
**SL (Significant Level):** it’s a value defined by Data Scientist during elicitation Phase to decide error tolerance (model acceptance criteria).

**CL (Confidence Level): 🡺 1 – SL:** This is the minimum threshold of the metric (like accuracy, …) for a given model to be deployed at staging level.

**p-value**: It is the SL value that is calculated by your Hypothesis Testing/Statistical Testing.

**Backward Elimination Algorithm:**

**Step 1:** perform All-IN operation(Input all features of coefficient and Intercept)

****

Coefficient is: all b in the profit formula( except b zero)

Intercept is: b zero

Features: California, Florida, NY, R&D, Adm and Markg.

**Step 2:** Decide the SL of Your model.

**Step 3:** Perform OLS operation and get the p-values of all features.

**Step 4:**  Select the feature column that has the HIGHEST P-value.

**Step 5:** check the following condition:

**If P-value > SL:**

**Eliminate that feature.**

**Else:**

**Go to Step 7**

**Step 6:** Repeat Step3 with newfeature col.

**Step 7:** Consider the featureSet as final set to propose the model

|  |  |  |
| --- | --- | --- |
|  | **Method 2. Backward Elimination Technique**  **using OLS** |  |
|  | Step1: Perform All in.  In [1, True, False, False, 162597.7, 151377.59, 443898.53]:  Intercept is 1,  California is True ,  Florida is False,  NY is False,  R&D is 162597.7,  Adm is 151377.59  Markg is 443898.53 | featuresAllIn = np.append(np.ones((50,1)).astype(int) , features, axis = 1 )  featuresAllIn[1,:]  array([1, True, False, False, 162597.7, 151377.59, 443898.53],  dtype=object) |
|  | Step2: Decide the SL  SL = 0.05 |  |
|  | Step3: Perform OLS  endog is label column --- label numpy array  exog is feature columns(here is featuresAllIn)  Iteration1: All In feature column, Iteration X: New Feature Column  # (OLS(endog,exog))  In model summary result, we are interested in P-value column: **P>|t|**  Const is b zero  Const --- Intercept Feature (b zero)  x1 --- California(b one)  x2. --- Florida(b two)  x3 --- NY(b three)  x4 --- R&D(b four)  x5. --- Adm(b fifth)  x6. --- Markg(b six) | import statsmodels.regression.linear\_model as stat  #Iteration 1:  model = stat.OLS(endog=label, exog=featuresAllIn).fit()  model.summary()   |  |  |  |  | | --- | --- | --- | --- | | OLS Regression Results | | | | | Dep. Variable: | y | R-squared: | 0.951 | | Model: | OLS | Adj. R-squared: | 0.945 | | Method: | Least Squares | F-statistic: | 169.9 | | Date: | Sat, 30 May 2020 | Prob (F-statistic): | 1.34e-27 | | Time: | 16:03:41 | Log-Likelihood: | -525.38 | | No. Observations: | 50 | AIC: | 1063. | | Df Residuals: | 44 | BIC: | 1074. | | Df Model: | 5 |  |  | | Covariance Type: | nonrobust |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | |  | coef | std err | t | **P>|t|** | [0.025 | 0.975] | | const | 3.763e+04 | 5073.636 | 7.417 | 0.000 | 2.74e+04 | 4.79e+04 | | x1 | 1.249e+04 | 2449.797 | 5.099 | 0.000 | 7554.868 | 1.74e+04 | | x2 | 1.269e+04 | 2726.700 | 4.654 | 0.000 | 7195.596 | 1.82e+04 | | x3 | 1.245e+04 | 2486.364 | 5.007 | 0.000 | 7439.285 | 1.75e+04 | | x4 | 0.8060 | 0.046 | 17.369 | 0.000 | 0.712 | 0.900 | | x5 | -0.0270 | 0.052 | -0.517 | **0.608** | -0.132 | 0.078 | | x6 | 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. | 2.69e+17 | |
|  | **Step4:**  Select the feature column that has the HIGHEST P-value.  Const --- Intercept Feature  x1 --- California  x2. --- Florida  x3 --- NY  x4 --- R&D  x5. --- Adm  x6. --- Markg  We select Admin = 0.608 | We select x5: ( It has the highest p-value)   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | x5 | -0.0270 | 0.052 | -0.517 | **0.608** | -0.132 | 0.078 | |
|  | **Step5:** Since Admin Pvalue > SL(0.05)  Therefore, eliminate Admin  In newFeatureCol1 we don’t have X5 | newFeatureCol1 = featuresAllIn[:,[0,1,2,3,4]] |
|  | **Step 6:** Repeat Step3 with newfeature col. | model = stat.OLS(endog=label, exog=newFeatureCol1).fit()  model.summary()   |  |  |  |  | | --- | --- | --- | --- | | OLS Regression Results | | | | | Dep. Variable: | y | R-squared: | 0.950 | | Model: | OLS | Adj. R-squared: | 0.946 | | Method: | Least Squares | F-statistic: | 215.8 | | Date: | Sat, 30 May 2020 | Prob (F-statistic): | 9.72e-29 | | Time: | 16:08:34 | Log-Likelihood: | -525.53 | | No. Observations: | 50 | AIC: | 1061. | | Df Residuals: | 45 | BIC: | 1071. | | Df Model: | 4 |  |  | | Covariance Type: | nonrobust |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | |  | coef | std err | t | **P>|t|** | [0.025 | 0.975] | | const | 3.525e+04 | 2100.376 | 16.782 | 0.000 | 3.1e+04 | 3.95e+04 | | x1 | 1.171e+04 | 1910.312 | 6.130 | 0.000 | 7861.854 | 1.56e+04 | | x2 | 1.185e+04 | 2170.903 | 5.459 | 0.000 | 7477.785 | 1.62e+04 | | x3 | 1.169e+04 | 1988.428 | 5.879 | 0.000 | 7684.996 | 1.57e+04 | | x4 | 0.7967 | 0.042 | 18.771 | 0.000 | 0.711 | 0.882 | | x5 | 0.0298 | 0.016 | 1.842 | **0.072** | -0.003 | 0.062 |  |  |  |  |  | | --- | --- | --- | --- | | Omnibus: | 14.640 | Durbin-Watson: | 1.257 | | Prob(Omnibus): | 0.001 | Jarque-Bera (JB): | 21.037 | | Skew: | -0.938 | Prob(JB): | 2.70e-05 | | Kurtosis: | 5.565 | Cond. No. | 6.69e+17 | |
|  | **Repeat Step3 and 4:**  Select the feature column that has the HIGHEST P-value.  Const --- Intercept Feature  x1 --- California  x2. --- Florida  x3 --- NY  x4 --- R&D  x5. --- Markg  We select Markg = 0.072 | Select x5 with p-value of **0.072** |
|  | **Step 5:**  check the following condition:  **If P-value > SL:**  **Eliminate that feature.**  **Else:**  **Go to Step 7**    Step5: Since Markg Pvalue > SL(0.05)  Therefore, eliminate Markg | newFeatureCol1 = featuresAllIn[:,[0,1,2,3,4]] |
|  | **Step 6:** Repeat Step3 with newfeature col. | #Iteration 3:  model = stat.OLS(endog=label, exog=newFeatureCol1).fit()  model.summary()   |  |  |  |  | | --- | --- | --- | --- | | OLS Regression Results | | | | | Dep. Variable: | y | R-squared: | 0.947 | | Model: | OLS | Adj. R-squared: | 0.943 | | Method: | Least Squares | F-statistic: | 272.4 | | Date: | Sat, 30 May 2020 | Prob (F-statistic): | 2.76e-29 | | Time: | 16:10:00 | Log-Likelihood: | -527.35 | | No. Observations: | 50 | AIC: | 1063. | | Df Residuals: | 46 | BIC: | 1070. | | Df Model: | 3 |  |  | | Covariance Type: | nonrobust |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | |  | coef | std err | t | **P>|t|** | [0.025 | 0.975] | | Const | 3.686e+04 | 1959.786 | 18.806 | 0.000 | 3.29e+04 | 4.08e+04 | | x1 | 1.189e+04 | 1956.677 | 6.079 | 0.000 | 7955.697 | 1.58e+04 | | x2 | 1.306e+04 | 2122.665 | 6.152 | 0.000 | 8785.448 | 1.73e+04 | | x3 | 1.19e+04 | 2036.022 | 5.847 | 0.000 | 7805.580 | 1.6e+04 | | x4 | 0.8530 | 0.030 | 28.226 | 0.000 | 0.792 | 0.914 |  |  |  |  |  | | --- | --- | --- | --- | | Omnibus: | 13.418 | Durbin-Watson: | 1.122 | | Prob(Omnibus): | 0.001 | Jarque-Bera (JB): | 17.605 | | Skew: | -0.907 | Prob(JB): | 0.000150 | | Kurtosis: | 5.271 | Cond. No. | 3.70e+17 | |
|  | **Repeat Step3 and 4:**  Select the feature column that has the HIGHEST P-value. | All p-values are zero   |  | | --- | | **P>|t|** | | 0.000 | | 0.000 | | 0.000 | | 0.000 | | 0.000 | |
|  | **Step 5:**  check the following condition:  **If P-value > SL:**  **Eliminate that feature.**  **Else:**  **Go to Step 7** | Go To STEP 7 |
|  | **Step 7**  (newFeatureCol1 = featuresAllIn[:,[0,1,2,3,4]])  x1 --- California  x2. --- Florida  x3 --- NY  x4 --- R&D  **Based on OLS strategy, the state column and R&D has the highest significance with profit** | featuresFinal = newFeatureCol1 |
|  |  | **Based on correlation analysis :**  Features Selected: R&d, Mkg  But  **Based on OLS strategy:**  Features Selected: **the state column and R&D** |
|  | The above will become input to your modelling algo!!! |  |

**Extra note:**

|  |  |
| --- | --- |
| In the linear Regression, we had this code to find the best sample.  All test s (test\_score) are R squared.  RMSE | from sklearn.linear\_model import LinearRegression  from sklearn.model\_selection import train\_test\_split  for i in range(1,100):  X\_train,X\_test,y\_train,y\_test = train\_test\_split(features,label,test\_size=0.2,random\_state=i)  model = LinearRegression()  model.fit(X\_train,y\_train)  train\_score = model.score(X\_train,y\_train)  test\_score = model.score(X\_test,y\_test)    if test\_score > train\_score:  print("Test S {}, Train Score {}, RandomSeed {}".format(test\_score,train\_score,i))  **Test S 0.9695039421049821**, Train Score 0.9545249190394052, RandomSeed 3  **Test S 0.9631182154839475**, Train Score 0.9528197369259258, RandomSeed 8  **Test S 0.9816423482070255**, Train Score 0.9494673013344644, RandomSeed 10  Test S 0.9606215790278543, Train Score 0.9527636176933665, RandomSeed 14  Test S 0.9835849730044817, Train Score 0.9460054870434312, RandomSeed 26  Test S 0.9636425773684422, Train Score 0.9527636606684406, RandomSeed 27  Test S 0.9944092048209744, Train Score 0.9400496694274888, RandomSeed 30  Test S 0.9778242092591887, Train Score 0.9486350116716654, RandomSeed 37  Test S 0.9724794487377619, Train Score 0.9473317052697812, RandomSeed 38  Test S 0.9928344802911049, Train Score 0.9492886917497556, RandomSeed 39  Test S 0.9802519469633169, Train Score 0.9491742100347064, RandomSeed 41  Test S 0.9789129767378081, Train Score 0.948821675263085, RandomSeed 46  Test S 0.98399193890564, Train Score 0.9486450781125914, RandomSeed 47  Test S 0.980277279178695, Train Score 0.9500780390200971, RandomSeed 48  Test S 0.9608624689052039, Train Score 0.9541375225175409, RandomSeed 51  Test S 0.9743646706957547, Train Score 0.952756273050018, RandomSeed 52  Test S 0.9804067424885895, Train Score 0.9504872715098402, RandomSeed 56  Test S 0.9719509793938971, Train Score 0.9473987125707488, RandomSeed 62  Test S 0.95820089851047, Train Score 0.9505483928196958, RandomSeed 63  Test S 0.9588832495320915, Train Score 0.9562672856609079, RandomSeed 67  Test S 0.9791787060652751, Train Score 0.937932068950384, RandomSeed 68  Test S 0.9694792167947474, Train Score 0.9504137960985714, RandomSeed 71  Test S 0.9562771755752736, Train Score 0.9562030951258303, RandomSeed 72  Test S 0.981214310330871, Train Score 0.9453900863447221, RandomSeed 73  Test S 0.9618591691900452, Train Score 0.9553251075019685, RandomSeed 74  ...  Test S 0.9676701872390631, Train Score 0.9529778812782739, RandomSeed 90  Test S 0.9793995823406391, Train Score 0.9469346629378338, RandomSeed 92  Test S 0.9682219576297961, Train Score 0.9534166513146052, RandomSeed 93  Test S 0.9676991009836634, Train Score 0.9514417860805683, RandomSeed 94 |

**Method 3: RFE (Recursive Feature Elimination ) Technique**

In this method, your Machine Learning algorithm let you know which features has highest importance!

RFE can be applied only on some algorithms

Regression: (Algorithm must have coeff variable output)

1. LinearRegression

2. SupportVectorRegression

3. DecisionTreeRegressor

4. RandomForestRegressor

Classification: (Algorithm must have feature importance variable)

1. DecisionTreeClassifier

2. RandomForestClassifier

Steps to apply RFE

1. Initialize the model algorithm

2. Apply RFE to model (ALL FEATURES AND LABEL)

3. Get Features with High Ranking (1,2,3,4,...) (Get features that has Rank 1. Sometimes Rank 2 is considered)

|  |  |  |
| --- | --- | --- |
|  | 1.Initialize the model algorithm  2. Apply RFE to model (ALL FEATURES AND LABEL)  # Fit the data with RFE  3. Get Features with High Ranking (1,2,3,4,...) (Get features that has Rank 1. Sometimes Rank 2 is considered)  # California,Florida,NY,R&D,Adm,Markg  In RFE you have an option to decide in a single Iteration, how much features you want to eliminate. For this we use Step.  RFE(estimator=modelLR,step=1)  One elimination at a time. | #1.  from sklearn.linear\_model import LinearRegression  modelLR = LinearRegression()  # 2.  from sklearn.feature\_selection import RFE  selectFeaturesFromRFE = RFE(estimator=modelLR,  step=1)  # Fit the data with RFE  selectFeaturesFromRFE.fit(features,label)  # 3.  print(selectFeaturesFromRFE.ranking\_)  [1 1 1 2 3 4] |
|  | Observation: State and R&D is shortlisted for Model Building (Based on [**1 1 1 2** 3 4]) |  |

**Method 4: Select By Model**

Steps to apply SBM (All Model algo will work)

1. Initialize the model algorithm

2. Apply RFE to model (ALL FEATURES AND LABEL)

3. Get Features with High Support (True/False)

|  |  |  |
| --- | --- | --- |
|  | 1. Initialize the model algorithm  2. Apply SBM to model (ALL FEATURES AND LABEL)  3. Get Features with True i.e.  ( based on [ True True True False False False] )  That has positive support (1,2,3,4,...)  # California,Florida,NY,R&D,Adm,Markg | # 1.  from sklearn.linear\_model import LinearRegression  modelLR = LinearRegression()  # 2.  from sklearn.feature\_selection import SelectFromModel  selectFeaturesFromSBM = SelectFromModel(modelLR)  # Fit the data with SBM  selectFeaturesFromSBM.fit(features,label)  # 3.  print(selectFeaturesFromSBM.get\_support())  [ True True True False False False] |
|  | Observation:  Get Features with True i.e.  ( based on [ True True True False False False] )  Select California,Floriada and NY as finalFeatures |  |
|  |  |  |

**Apply Outputs of Feature Engineering**

|  |  |  |
| --- | --- | --- |
|  | We have all selected features from all feature engineering methods we have done. | featureCorrAnalysis = features[:,[3,5]]  featureBackwardEliminationOLS = features[:,[0,1,2,3]]  featureRFERank1n2= features[:,[0,1,2,3]]  featureRFERank1Only = features[:,[0,1,2]]  featureSBMTrueOnly = features[:,[0,1,2]] |
|  |  | from sklearn.linear\_model import LinearRegression  from sklearn.model\_selection import train\_test\_split  for i in range(1,51):  X\_train,X\_test,y\_train,y\_test = train\_test\_split(featureCorrAnalysis, label, test\_size=0.2, random\_state = i)  model1 = LinearRegression()  model1.fit(X\_train,y\_train)    train\_score = model1.score(X\_train,y\_train)  test\_score = model1.score(X\_test,y\_test)    if test\_score > train\_score and test\_score >= CL:  print("Test: {} , Train: {} , RS : {}".format(test\_score,train\_score,i))  Test: 0.1507109998333278 , Train: -0.06593244561790113 , RS : 6  **Test: 0.12089159244307222 , Train: -0.01969795421804088 , RS : 10**  Test: 0.015080875800281057 , Train: 0.0017731552271403883 , RS : 32  Test: 0.025328630963713517 , Train: 0.013479770737339636 , RS : 49 |
|  | Output: 99% / 0.99 achieved using Corr Analysis using LinearRegresion, RS=10  Output: 98% / 0.98 achieved using Backward Elimination using OLS, RS=10  Output: 98% achieved using RFE (Rank 1 and 2) , RS=10  Output: 15% achieved using RFE(Rank1) using LinearRegression algo, RS=10  Output: 15% achieved using SBM using LinearRegression algo, RS=10 |  |
|  |  |  |