**DATA SCIENCE**

**PART 1 : PREPARING DATASET**

**Step 1 :** Read CSV

* Read CSV : ds = pd.read\_csv( “filename”)
* Divide CSV : ds.iloc [: , :].values

Col : Col

Row : Row

First row , first Col - 0

Last row , last Col - -1

All row, all Col - :

**Step 2 :** Handling Missing Data

* Import : from sklearn. preprocessing import Impter
* Create Imputer : Imputer (missing\_values = ‘NaN’ , strategy = ‘mean’ , axis = 0)
* Fit Imputer in DS : imputer.fit (dataset\_var[: , :])
* Transform in DS : imputer.transform(dataset\_var [: , :])

**Step 3 :** Encoding Independent Variables

* Import : from sklearn. preprocessing import LableEncoder
* Cerate LableEncoder : LableEncoder( )
* Fit and Transform : lableEncoder.fit\_transform(dataset\_var[: , :])

|  |  |
| --- | --- |
| Country | LableEncoder |
| India | 0 |
| Germany | 1 |
| America | 2 |

**Step 4 :** Encoding Categorical Data

* Import : from sklearn. preprocessing import OneHotEncoder
* Create OneHotEncoder : OneHotEncoder(categorical\_features = [col])
* Convert DS to Arr : oneHotEncoder.fit\_transform(dataset\_var).to\_array()

|  |  |  |  |
| --- | --- | --- | --- |
| Country | India | Germany | America |
| India | 1 | 0 | 0 |
| America | 0 | 0 | 1 |
| Germany | 0 | 1 | 0 |
| India | 1 | 0 | 0 |
| America | 0 | 0 | 1 |
| Germany | 0 | 1 | 0 |

**Step 5:** Split Train and Test Set

* Import : from sklearn.cross\_validation import train\_test\_split
* Create Train &Test Set :

x\_train, x\_test, y\_train, y\_test = train\_testsplit( x , y , test\_size = 0.2 , random\_state = 1)

x – Independent Variable

y – Dependent Variable

test \_size = 0.2 - 20 % of test data

random\_state = random sampling

**Step 6 :** Feature Scaling

* Formula :

Xnorm =

Xstand =

* Import : from sklearn. preprocessing import StandardScaler
* Create StandardScaler : StandardSacler()
* Fit and Transform : standardScaler.fit\_transform( dataset\_var)

Why Feature Scaling :

Euclidean Distance =

Reducing the Euclidean distance between the values.

**PART 2 : REGRESSION**

**Regression Formulas: y = a +bx**

**Model 1 :** Simple Linear Regression

Step 1 : Formula

**y = P0 + P1 \* x**

y - Dependent Variable

x - Independent Variable

P0 - Constant ( 2 is constant )

P1 - Independent Co- efficient (Slope of G)

Step 2 : How to find best fit line

* Formula :

Least Square = SUM ( yi - yi^ ) 2

yi – Actual (x , y)

yi^ - Predicted (x , y)

Which Slope has minimum Least Square value that is the best fit line.

Step 3 : Date Preprocessing

* Read the CSV & Assign x and y variables
* Split the train and test set
* No need of Feature Scaling in SLR

Step 4 : Fitting SLR

* Import : from sklearn.linear\_model import LinearRegression
* Create LR : LinearRegression ()
* Fit the LR to training set : regressor.fit ( x\_train , y\_train )

Step 5 : Predict the Test set

* Find predicted result : pred = regressor.predict( x\_test )

**Model 2 :** Multiple Linear Regression

Step 1 : Formula

**y = P0 + P1 \* x1+P2\*x2+P3\*x3+….+Pn\*xn**

y - Dependent Variable

x1 …. xn - Independent Variable

P0 - Constant ( 2 is constant )

P1….Pn - Independent Co- efficient (Slope of G)

Step 3 : Date Preprocessing & Eliminate dummy variable

* Read the CSV & Assign x and y variables
* Remove the unwanted columns through step wise regression
* Split the train and test set

Step 4 : Five models

* All – in - Randomly take all the models . Predict best one.
* Forward Selection & Backward Elimination

SL = 0.05

Best model - P < SL

* Bidirectional Elimination

Combination of both Forward and Backward

* Score Comparison

n col - 2^n – 1 models

Ex :

10 col – 2^10 – 1 = 1023 models

Best MLR data build model – **Backward elimination**

Step 5: Backward elimination

* Add constant 1 in front of the dataset

Ex :

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| X0 | X1 | X2 | X3 | X4 | X5 |
| 1 | Data | data | data | data | data |
| 1 | Data | data | data | data | data |

* np.append( arr = np.ones((data.size, 1)).astype(int), values = x, axis = 1)
* Import : import statsmodels.formula.api as sm
* Call the method OLS - ols = sm.OLS(endog = y , exog = x).fit()
* Model details - ols.summary()
* Find the best data model

Step 6:

* Import : from sklearn.linear\_model import LinearRegression
* Create LR : LinearRegression ()
* Fit the LR to training set : regressor.fit ( x\_train , y\_train )

Step 7 : Predict the Test set

* Find predicted result : pred = regressor.predict( x\_test )

**Model 3 :** Polynomial Linear Regression

Step 1 : Formula

**y = P0 + P1 \* x1+P2\*x1^2+P3\*x1^3+….+Pn\*x1^n**

y - Dependent Variable

x1 - Independent Variable

P0 - Constant ( 2 is constant )

P1….Pn - Independent Co- efficient (Slope of G)

Step 2 : Date Preprocessing

* Read the CSV & Assign x and y variables

Step 3 : Fitting SLR

* Import : from sklearn.linear\_model import LinearRegression
* Create LR : LinearRegression ()
* Fit the LR to training set : regressor.fit ( x , y )

Step 4 : Fitting PLR

* Import : from sklearn.preprocessing import PolynomialFeatures
* Create PR : PolynomialFeatures ( degree = 4)
* Fit & Transform PR : poly\_regressor.fit\_transform ( x )
* Create LR : LinearRegression ()
* Fit the LR to training set : regressor.fit ( x, y)

Step 5 : Predict the LR Test set

* Find predicted result : pred = regressor.predict( 6 )

Step 6 : Predict the PR Test set

* Find predicted result : pred = regressor.predict(poly\_regressor.fit\_transform ( 6 ))

**Association Rule Learning**

**Apriori ARL :**

Support :

Support = (Transaction involving item1) / (Total no of transaction)

Ex :

Support = 200/ 2000 = 10%

Confidence:

Confidence = (Transaction involving item1 vs item2 ) / (Transaction involving item1)

Ex:

Confidence = 100 / 200 = 50%

Lift :

Lift = Confidence (item1 vs item2) / Support(item1)

Ex:

Lift = 50 / 10 = 5

Result :

Customer buying item1 and item2 together five times more than the chance of purchasing item1 alone.

Ex :

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Transaction | Onion | Potato | Burger | Milk | Beer |
| T1 | 1 | 1 | 1 | 0 | 0 |
| T2 | 0 | 1 | 1 | 1 | 0 |
| T3 | 0 | 0 | 0 | 1 | 1 |
| T4 | 1 | 1 | 0 | 1 | 0 |
| T5 | 1 | 1 | 1 | 0 | 1 |
| T6 | 1 | 1 | 1 | 1 | 1 |

|  |  |
| --- | --- |
| Items | Frequency |
| Onion | 4 |
| Potato | 5 |
| Burger | 4 |
| Milk | 4 |

|  |  |
| --- | --- |
| Item Set | Frequency |
| OP | 4` |
| OB | 3 |
| OM | 2 |
| PB | 4 |
| PM | 3 |
| BM | 2 |

|  |  |
| --- | --- |
| Item Set | Frequency |
| OPB | 4` |
| PBM | 3 |

|  |  |  |  |
| --- | --- | --- | --- |
| Transaction | Confidence |  |  |
| O^P^B - O^P | 4/4 | 1 | 100% |
| O^P^B - O^B | 4/3 | 1.3 | 130% |
| O^P^B - P^B | 4/4 | 1 | 100% |
| O^P^B - O | 4/4 | 1 | 100% |
| O^P^B - P | 4/5 | .8 | 80% |
| O^P^B - M | 4/4 | 1 | 100% |

|  |  |  |  |
| --- | --- | --- | --- |
| Transaction | Confidence |  |  |
| P^B^M – P^B | 3/4 | .75 | 75% |
| P^B^M - P^M | 3/3 | 1 | 100% |
| P^B^M - B^M | 3/2 | 1.5 | 150% |
| P^B^M - P | 3/5 | .6 | 60% |
| P^B^M - B | 3/4 | .75 | 75% |
| P^B^M - M | 3/4 | .75 | 75% |