Name : Sumon Singh

Roll No. 16

Paper : Statistical Method

Import Libraries

In [1]:
```python
import pandas as pd
from sklearn.cluster import KMeans
from sklearn import metrics
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
import numpy as np
from sklearn.decomposition import PCA
from pandas_profiling import ProfileReport
```

Modeling On Unigram Dataset

Read the unigram dataset

In [2]:
```python
df_uni = pd.read_excel('/home/sumon/Documents/Datasets/DS-sem1.xlsx',sheet_name='uni_euclid_mat_intersection')
```

In [3]:
```python
df_uni
```

Out[3]:

	Sr #	raga	closest	farthest	Unnamed: 4	euclidean distance	AbhaireeTodi	Abhogi	Adana	Adi Bharivi	...	V
0	1	AbhaireeTodi	YogJyoti	Plasi	NaN	AbhaireeTodi	0.000000	152.272782	257.699437	171.452617	...	32
1	2	Abhogi	DevMukhari	Bhageshari	NaN	Abhogi	152.272782	0.000000	250.868491	67.461100	...	37
2	3	Adana	KaliyanBasant	Bangal Bhairav	NaN	Adana	257.699437	250.868491	0.000000	229.780330	...	11
3	4	Adi Bharivi	Rudher_Gandhar	Jaywanti	NaN	Adi Bharivi	171.452617	67.461100	229.780330	0.000000	...	41
4	5	Aheer Bhariv	Hamir	Bhageshari	NaN	Aheer Bhariv	230.822443	204.151904	104.584894	249.242853	...	31
...	...	...	...	...	...	...	...	...	...	...	...	...
324	325	Yamani Bilawal	Vegwahini	Bhageshari	NaN	Yamani Bilawal	526.480769	214.923242	606.164994	238.470124	...	44
325	326	YamanKaliyan	Pahadi	Bangal Bhairav	NaN	YamanKaliyan	538.697503	268.233108	658.758681	406.606690	...	50
326	327	YashPriya	Rsik_priya	SerswatiSarang	NaN	YashPriya	124.519075	135.173962	393.010178	278.888867	...	44
327	328	YashRanjani	AnandLila	Jaywanti	NaN	YashRanjani	426.909827	322.225077	221.621299	559.581093	...	32
328	329	YogJyoti	ChanderRekha	NagSwarabali	NaN	YogJyoti	227.525823	155.730537	385.112970	285.115766	...	34

329 rows × 335 columns

Data cleaning of unigram dataset

In [4]:
```python
df_uni.columns
```

Out[4]:
```
Index(['Sr #', 'raga', 'closest', 'farthest', 'Unnamed: 4',
       'euclidean distance', 'AbhaireeTodi', 'Abhogi', 'Adana', 'Adi Bharivi',
       ...
       'VidyaPati', 'Vijay', 'VijayKokila', 'Vinayak', 'Yaman',
       'Yamani Bilawal', 'YamanKaliyan', 'YashPriya', 'YashRanjani',
       'YogJyoti'],
      dtype='object', length=335)
```

In [5]:
```python
df_uni.shape
```

Out[5]:
```
(329, 335)
```

Drop the unnecessary columns

In [6]:
```python
df_uni.drop(df_uni.columns[0:5],axis=1,inplace=True)
```

In [7]:
```python
df_uni.dtypes
```

Out[7]:
```
euclidean distance     object
AbhaireeTodi          float64
Abhogi                float64
Adana                 float64
Adi Bharivi           float64
                       ...
Yamani Bilawal        float64
YamanKaliyan          float64
YashPriya             float64
YashRanjani           float64
YogJyoti              float64
Length: 330, dtype: object
```

Clustering on unigram dataset

Select only numeric data for clustering

In [8]:
```python
X=df_uni.iloc[:,1:]
```

K-means cluster model

In [9]:
```python
model_uni=KMeans(n_clusters=10,random_state=0)
```

In [10]:
```python
model_uni.fit(X)
```

Out[10]:
```
KMeans(n_clusters=10, random_state=0)
```

In [11]:
```python
p_uni=model_uni.predict(X)
```

In [12]:
```python
df_uni["target"]=p_uni
```

Quality of cluster

In [13]:
```python
metrics.silhouette_score(X,model_uni.labels_)
```

Out[13]:
```
0.1118143413849179
```

In [14]:
```python
result_pred_uni=pd.DataFrame({'X':df_uni.iloc[:,0],'Y':p_uni})
```

In [15]:
```python
result_pred_uni.sort_values(['Y'])
```

Out[15]:

	X	Y
72	Devgiri	0
21	Barba	0
225	Noopur	0
142	KaliyanKaisri	0
145	KamalManoharee	0
...	...	...
5	Aheerlalit	9
175	LachariTodi	9
50	Chanderika Bharivi	9
283	SawaniKaliyan	9
113	Hembanti	9

329 rows × 2 columns

In [16]:
```python
model_uni.cluster_centers_
```

Out[16]:
```
array([[333.46860676, 275.15591411, 231.07517818, ..., 359.37952949,
        325.30166697, 341.43512777],
       [187.90721354, 116.97828216, 293.546454  , ..., 197.48976042,
        394.77231346, 212.72894893],
       [293.34517601, 216.98786772, 428.43982998, ..., 228.7696526 ,
        450.0602799 , 301.6617876 ],
       ...,
       [420.56294706, 285.94816016, 550.23524809, ..., 327.162874  ,
        490.82478074, 442.67824261],
       [192.86360464, 215.97607984,  89.65798258, ..., 267.23686472,
        388.16171074, 244.45505262],
       [302.25787368, 299.83171746, 134.46267056, ..., 360.26714741,
        346.50031984, 385.22786156]])
```

Regression on unigram dataset

Extract top 20 best correlated features with the cluster

In [17]:
```python
df_uni.corr().iloc[:,-1].abs().sort_values(ascending=False).iloc[0:20]
```

Out[17]:
```
target            1.000000
Girija            0.456178
KamalRanjani      0.361103
SerswatRanjani    0.359402
NagSwarabali      0.340339
varati            0.330283
ChandarNandan     0.321111
Devkipuriya       0.315458
Dhaney Dhaiwat    0.306547
JaunaPuri         0.297688
Neetmati          0.285086
chhanika          0.275447
Adana             0.274022
Manvi             0.271846
KalaBati          0.271613
MangalGujari      0.271463
Abhogi            0.267413
Sohini            0.262853
KaliyanBasant     0.260045
SgunRanjani       0.259410
Name: target, dtype: float64
```

In [18]:
```python
fea_col_uni=df_uni.corr().iloc[:,-1].abs().sort_values(ascending=False).iloc[1:21].index
```

In [19]:
```python
fea_col_uni
```

Out[19]:
```
Index(['Girija', 'KamalRanjani', 'SerswatRanjani', 'NagSwarabali', 'varati',
       'ChandarNandan', 'Devkipuriya', 'Dhaney Dhaiwat', 'JaunaPuri',
       'Neetmati', 'chhanika', 'Adana', 'Manvi', 'KalaBati', 'MangalGujari',
       'Abhogi', 'Sohini', 'KaliyanBasant', 'SgunRanjani', 'Rsik_priya'],
      dtype='object')
```

Create new dataset of top 20 correlated features with the target

In [20]:
```python
df_uni_new=df_uni[fea_col_uni]
```

In [21]:
```python
df_uni_new
```

Out[21]:

	Girija	KamalRanjani	SerswatRanjani	NagSwarabali	varati	ChandarNandan	Devkipuriya	Dhaney Dhaiwat	JaunaPuri	Neetmat
0	545.947800	386.139871	162.769776	486.569625	424.123803	352.681159	250.701017	273.857627	186.316934	340.35863
1	345.421482	230.282435	93.749667	381.241393	285.220967	305.638676	100.970293	151.383619	197.909070	249.09436
2	533.702164	425.779286	341.764539	647.787774	534.815856	425.806294	349.455290	329.913625	161.830158	470.05957
3	410.369346	138.032605	244.965304	406.135445	422.407386	286.052443	192.296646	194.486503	158.154987	281.52442
4	397.307186	213.475994	262.994297	610.286818	417.435025	426.075111	307.896086	155.521703	184.691635	331.56145
...	...	...	...	...	...	...	...	...	...	...
324	516.111422	249.819935	372.476845	438.421031	353.980225	490.703577	429.633565	299.604740	531.968044	474.98631
325	573.869323	284.416947	390.932219	607.080720	395.845930	614.627529	356.164288	278.154633	619.099346	432.94687
326	456.404426	310.665415	163.719272	287.416075	336.740256	348.162318	211.926874	235.539806	343.250637	279.08959
327	334.233451	315.412428	500.236944	337.615166	435.873835	333.151617	509.374126	291.982876	297.889241	446.68221
328	599.976666	323.677308	131.045794	603.688661	394.124346	480.960497	270.992620	209.475536	351.239235	235.79228

329 rows × 20 columns

Spilting dataset in training and testing

In [22]:
```python
X_train,X_test,Y_train,Y_test=train_test_split(df_uni_new,p_uni,test_size=0.3,random_state=0)
```

Dimension reduction using PCA

In [23]:
```python
pca=PCA(n_components=2)
```

In [24]:
```python
X_train=pca.fit_transform(X_train)
```

In [25]:
```python
X_test=pca.transform(X_test)
```

Regression Model

In [26]:
```python
lm_model_uni = LinearRegression().fit(X_train,Y_train)
```

In [27]:
```python
lm_model_pred_uni=lm_model_uni.predict(X_test)
```

In [28]:
```python
lm_model_pred_uni=np.round(lm_model_pred_uni)
```

Accuaracy of regression model

In [29]:
```python
metrics.mean_absolute_error(Y_test,lm_model_pred_uni)
```

Out[29]:
```
1.6666666666666667
```

In [30]:
```python
metrics.mean_squared_error(Y_test,lm_model_pred_uni)
```

Out[30]:
```
4.717171717171717
```

In [31]:
```python
metrics.r2_score(Y_test,lm_model_pred_uni)
```

Out[31]:
```
0.2665154207386724
```

Modeling On bigram Dataset

Read the bigram dataset

In [32]:
```python
df_bi = pd.read_excel('/home/sumon/Documents/Datasets/DS-sem1.xlsx',sheet_name='bi_euclid_mat_intersection')
```

In [33]:
```python
df_bi
```

Out[33]:

	raga	closest	farthest	Unnamed: 3	euclidean distance	AbhaireeTodi	Abhogi	Adana	Adi Bharivi	Aheer Bhariv
0	AbhaireeTodi	YogJyoti	Plasi	NaN	AbhaireeTodi	0.000000	100.364336	123.511133	122.073748	70.590368
1	Abhogi	DevMukhari	Bhageshari	NaN	Abhogi	100.364336	0.000000	99.030298	43.817805	41.182521
2	Adana	KaliyanBasant	Bangal Bhairav	NaN	Adana	123.511133	99.030298	0.000000	138.949631	2.828427
3	Adi Bharivi	Rudher_Gandhar	Jaywanti	NaN	Adi Bharivi	122.073748	43.817805	138.949631	0.000000	24.041631
4	Aheer Bhariv	Hamir	Bhageshari	NaN	Aheer Bhariv	70.590368	41.182521	2.828427	24.041631	0.000000
...	...	...	...	...	...	...	...	...	...	...
324	Yamani Bilawal	Vegwahini	Bhageshari	NaN	Yamani Bilawal	242.950612	83.264638	142.720006	80.199751	148.243044
325	YamanKaliyan	Pahadi	Bangal Bhairav	NaN	YamanKaliyan	210.853029	113.013274	132.174128	133.592664	86.029065
326	YashPriya	Rsik_priya	SerswatiSarang	NaN	YashPriya	123.656783	78.752778	154.369686	96.010416	56.187187
327	YashRanjani	AnandLila	Jaywanti	NaN	YashRanjani	159.132021	13.000000	37.589892	157.330862	177.910090
328	YogJyoti	ChanderRekha	NagSwarabali	NaN	YogJyoti	117.025638	53.507009	130.610107	68.942005	51.458721

329 rows × 334 columns

Data cleaning of bigram dataset

In [34]:
```python
df_bi.columns
```

Out[34]:
```
Index(['raga', 'closest', 'farthest', 'Unnamed: 3', 'euclidean distance',
       'AbhaireeTodi', 'Abhogi', 'Adana', 'Adi Bharivi', 'Aheer Bhariv',
       ...
       'VidyaPati', 'Vijay', 'VijayKokila', 'Vinayak', 'Yaman',
       'Yamani Bilawal', 'YamanKaliyan', 'YashPriya', 'YashRanjani',
       'YogJyoti'],
      dtype='object', length=334)
```

Drop the unnecessary columns

In [35]:
```python
df_bi.drop(df_bi.columns[:4],axis=1,inplace=True)
```

In [36]:
```python
df_bi.dtypes
```

Out[36]:
```
euclidean distance     object
AbhaireeTodi          float64
Abhogi                float64
Adana                 float64
Adi Bharivi           float64
                       ...
Yamani Bilawal        float64
YamanKaliyan          float64
YashPriya             float64
YashRanjani           float64
YogJyoti              float64
Length: 330, dtype: object
```

Clustering on bigram dataset

Select only numeric data for clustering

In [37]:
```python
X_bi=df_bi.iloc[:,1:]
```

K-means cluster model

In [38]:
```python
model_bi=KMeans(n_clusters=10,random_state=0)
```

In [39]:
```python
model_bi.fit(X_bi)
```

Out[39]:
```
KMeans(n_clusters=10, random_state=0)
```

In [40]:
```python
p_bi=model_bi.predict(X_bi)
```

In [41]:
```python
df_bi["target"]=p_bi
```

Quality of cluster

In [42]:
```python
metrics.silhouette_score(X_bi,model_bi.labels_)
```

Out[42]:
```
0.0998404078314635
```

In [43]:
```python
result_pred_bi=pd.DataFrame({'X':df_bi.iloc[:,0],'Y':p_bi})
```

In [44]:
```python
result_pred_bi.sort_values(['Y'])
```

Out[44]:

	X	Y
217	Nand	0
25	Bawani	0
52	ChanderRekha	0
127	Jayant Malhar	0
125	Jay-Jay Banti	0
...	...	...
193	Maligaura	9
178	LalitMangal	9
194	Malini	9
314	Triveni	9
269	Roopwati	9

329 rows × 2 columns

In [45]:
```python
model_bi.cluster_centers_
```

Out[45]:
```
array([[114.40996745,  80.58193664,  98.97286906, ..., 109.48096032,
         41.38217213,  91.69671444],
       [121.05025502,  61.60361311, 122.73233308, ..., 111.61396375,
        128.8036765 , 115.41845009],
       [ 69.44961244,  74.89100855,  35.73531741, ..., 120.1340847 ,
        127.40401326, 106.20485999],
       ...,
       [ 90.84311703,  63.2117992 ,  63.69537505, ..., 191.00729052,
        122.62059652, 192.28599531],
       [ 89.04394764,  33.9780384 ,  83.9214368 , ...,  74.81660191,
         95.34084075,  73.0497058 ],
       [ 67.68872821,  24.18580552,  30.35961384, ...,  74.41977266,
        171.23181579,  82.95947827]])
```

Regression on bigram dataset

Extract top 20 best correlated features with the cluster

In [46]:
```python
df_bi.corr().iloc[:,-1].abs().sort_values(ascending=False).iloc[0:20]
```

Out[46]:
```
target            1.000000
Gope Kamboji      0.480897
Pahadi Kamodi     0.477454
RaketHans         0.468059
Samairee          0.464335
Karankwrali       0.455289
Hans Manjari      0.444496
Kolhas            0.432878
Kamaiyee          0.423506
GujariTodi        0.412382
kukubh            0.412265
KanakBasant       0.407818
RasRanjani        0.407316
SamaireeMalhar    0.407002
ManRang           0.398956
Chandrika         0.386864
Jayant            0.380992
Khoker            0.378607
ChhayaTodi        0.376847
Sur malhar        0.370452
Name: target, dtype: float64
```

In [47]:
```python
fea_col_bi=df_bi.corr().iloc[:,-1].abs().sort_values(ascending=False).iloc[1:21].index
```

In [48]:
```python
fea_col_bi
```

Out[48]:
```
Index(['Gope Kamboji', 'Pahadi Kamodi', 'RaketHans', 'Samairee', 'Karankwrali',
       'Hans Manjari', 'Kolhas', 'Kamaiyee', 'GujariTodi', 'kukubh',
       'KanakBasant', 'RasRanjani', 'SamaireeMalhar', 'ManRang', 'Chandrika',
       'Jayant', 'Khoker', 'ChhayaTodi', 'Sur malhar', 'RaagRanjani'],
      dtype='object')
```

Create new dataset of top 20 correlated features with the target

In [49]:
```python
df_bi_new=df_bi[fea_col_bi]
```

In [50]:
```python
df_bi_new
```

Out[50]:

	Gope Kamboji	Pahadi Kamodi	RaketHans	Samairee	Karankwrali	Hans Manjari	Kolhas	Kamaiyee	GujariTodi	kukubh	KanakBasar
0	146.400137	200.544259	167.017963	147.193750	133.555232	172.365890	164.170643	168.433963	61.530480	183.806964	138.95323
1	105.806427	113.225439	132.238421	138.747973	100.184829	132.657454	104.661359	91.809586	3.000000	103.898989	104.35995
2	175.982954	163.367683	167.071841	178.275068	145.048268	187.045449	198.924609	169.826382	49.819675	154.673850	133.11649
3	98.954535	128.035151	119.653667	151.043040	107.536040	143.289916	128.358872	118.941162	90.890043	98.752215	121.43722
4	62.072538	100.054985	69.318107	43.174066	60.415230	83.078276	67.305275	135.738720	88.475985	123.328829	75.21967
...	...	...	...	...	...	...	...	...	...	...	...
324	161.068308	163.765076	194.383641	151.115850	236.292192	212.903734	130.418557	200.117466	25.455844	203.803827	157.80367
325	192.919672	157.200509	194.797331	163.205392	182.477396	240.299813	195.662976	206.891276	31.304952	229.595296	145.25494
326	108.415866	109.539947	129.410201	69.763888	94.116949	150.519102	171.746907	56.797887	53.544374	160.993789	108.21737
327	44.508426	73.579889	67.941151	74.390860	63.521650	121.954910	92.892411	64.288413	233.942301	90.360390	68.11020
328	93.429118	140.591607	135.547040	85.545310	104.527508	143.551385	113.313724	94.942088	60.695964	162.729837	98.62555

329 rows × 20 columns

Spilting dataset in training and testing

In [51]:
```python
X_train,X_test,Y_train,Y_test=train_test_split(df_bi_new,p_bi,test_size=0.3,random_state=0)
```

In [52]:
```python
pca=PCA(n_components=2)
```

In [53]:
```python
X_train=pca.fit_transform(X_train)
```

In [54]:
```python
X_test=pca.transform(X_test)
```

Regression Model

In [55]:
```python
lm_model_bi = LinearRegression().fit(X_train,Y_train)
```

In [56]:
```python
lm_model_pred_bi=lm_model_bi.predict(X_test)
```

In [57]:
```python
lm_model_pred_bi=np.round(lm_model_pred_bi)
```

Accuaracy of regression model

In [58]:
```python
metrics.mean_absolute_error(Y_test,lm_model_pred_bi)
```

Out[58]:
```
2.212121212121212
```

In [59]:
```python
metrics.mean_squared_error(Y_test,lm_model_pred_bi)
```

Out[59]:
```
6.555555555555555
```

In [60]:
```python
metrics.r2_score(Y_test,lm_model_pred_bi)
```

Out[60]:
```
0.20624861018456764
```

Modeling On trigram Dataset

Read the trigram dataset

In [61]:
```python
df_tri = pd.read_excel('/home/sumon/Documents/Datasets/DS-sem1.xlsx',sheet_name='tri_euclid_mat_intersection')
```

In [62]:
```python
df_tri
```

Out[62]:

	raga	closest	farthest	Unnamed: 3	euclidean distance	AbhaireeTodi	Abhogi	Adana
0	AbhaireeTodi	vaijayanti	NagSwarabali	NaN	AbhaireeTodi	0.000000	34.088121	43.33589
1	Abhogi	Aheerlalit,Bilaskhani,Dhoulamberi,Gaumati,Guna...	Haivitri	NaN	Abhogi	34.088121	0.000000	32.93933
2	Adana	Aheerlalit,BhatiYaar,Bilaskhani,Dhoulamberi	Bangal Bhairav	NaN	Adana	43.335897	32.939338	0.00000
3	Adi Bharivi	Anand Bhariv,BhatiYaar,Malini,Sohini	Jayant	NaN	Adi Bharivi	48.363209	41.085277	64.84597
4	Aheer Bhariv	Chandrika	Bhageshari	NaN	Aheer Bhariv	17.860571	0.000000	0.00000
...	...	...	...	...	...	...	...	...
324	Yamani Bilawal	GunaKali	Chhaya	NaN	Yamani Bilawal	78.835271	17.748239	34.71311
325	YamanKaliyan	Bilaskhani	NagSwarabali	NaN	YamanKaliyan	51.078371	28.460499	39.50949
326	YashPriya	Lalitagauri	ShriKaliyan	NaN	YashPriya	47.812132	31.685959	51.07837
327	YashRanjani	Bawani,Shud Sharang,Tilung	PushapRanjani	NaN	YashRanjani	38.340579	5.000000	7.34846
328	YogJyoti	Malkauns,Rudra_Panchaum	ShriKaliyan	NaN	YogJyoti	35.256205	21.656408	30.26549

329 rows × 334 columns

Data cleaning of trigram dataset

In [63]:
```python
df_tri.columns
```

Out[63]:
```
Index(['raga', 'closest', 'farthest', 'Unnamed: 3', 'euclidean distance',
       'AbhaireeTodi', 'Abhogi', 'Adana', 'Adi Bharivi', 'Aheer Bhariv',
       ...
       'VidyaPati', 'Vijay', 'VijayKokila', 'Vinayak', 'Yaman',
       'Yamani Bilawal', 'YamanKaliyan', 'YashPriya', 'YashRanjani',
       'YogJyoti'],
      dtype='object', length=334)
```

Drop the unnecessary columns

In [64]:
```python
df_tri.drop(df_tri.columns[:4],axis=1,inplace=True)
```

In [65]:
```python
df_tri
```

Out[65]:

	euclidean distance	AbhaireeTodi	Abhogi	Adana	Adi Bharivi	Aheer Bhariv	Aheerlalit	AhireeTodi	Alhiya Bilawal	Anand Bhariv	...	VidyaPati	
0	AbhaireeTodi	0.000000	34.088121	43.335897	48.363209	17.860571	31.874755	47.116876	32.726136	28.160256	...	9.591663	20
1	Abhogi	34.088121	0.000000	32.939338	41.085277	0.000000	1.000000	12.767145	19.899749	2.000000	...	26.702060	28
2	Adana	43.335897	32.939338	0.000000	64.845971	0.000000	1.000000	27.730849	33.481338	4.000000	...	12.369317	32
3	Adi Bharivi	48.363209	41.085277	64.845971	0.000000	4.000000	0.000000	34.278273	16.522712	1.000000	...	23.537205	44
4	Aheer Bhariv	17.860571	0.000000	0.000000	4.000000	0.000000	65.703881	25.238859	8.000000	26.324893	...	36.097091	23
...	...	...	...	...	...	...	...	...	...	...	...	...	...
324	Yamani Bilawal	78.835271	17.748239	34.713110	22.978251	55.713553	75.756188	34.351128	98.147848	19.157244	...	56.409219	40
325	YamanKaliyan	51.078371	28.460499	39.509493	18.761663	13.341664	17.888544	20.024984	68.563839	12.083046	...	33.689761	30
326	YashPriya	47.812132	31.685959	51.078371	38.832976	13.379088	16.217275	24.000000	36.304270	14.491377	...	62.377881	67
327	YashRanjani	38.340579	5.000000	7.348469	49.213819	58.830264	35.355339	13.228757	0.000000	52.325902	...	23.727621	45
328	YogJyoti	35.256205	21.656408	30.265492	20.639767	5.099020	14.594520	15.297059	44.922155	19.052559	...	53.870214	50

329 rows × 330 columns

In [66]:
```python
df_tri.dtypes
```

Out[66]:
```
euclidean distance     object
AbhaireeTodi          float64
Abhogi                float64
Adana                 float64
Adi Bharivi           float64
                       ...
Yamani Bilawal        float64
YamanKaliyan          float64
YashPriya             float64
YashRanjani           float64
YogJyoti              float64
Length: 330, dtype: object
```

Clustering on trigram dataset

Select only numeric data for clustering

In [67]:
```python
X_tri=df_tri.iloc[:,1:]
```

K-means cluster model

In [68]:
```python
model_tri=KMeans(n_clusters=10,random_state=0)
```

In [69]:
```python
model_tri.fit(X_tri)
```

Out[69]:
```
KMeans(n_clusters=10, random_state=0)
```

In [70]:
```python
p_tri=model_tri.predict(X_tri)
```

In [71]:
```python
df_tri["target"]=p_tri
```

Quality of cluster

In [72]:
```python
metrics.silhouette_score(X_tri,model_tri.labels_)
```

Out[72]:
```
0.10735921805954907
```

In [73]:
```python
result_pred_tri=pd.DataFrame({'X':df_tri.iloc[:,0],'Y':p_tri})
```

In [74]:
```python
result_pred_tri.sort_values(['Y'])
```

Out[74]:

	X	Y
106	Hamir	0
142	KaliyanKaisri	0
26	Behag	0
297	ShriKaliyan	0
32	Bhankhar	0
...	...	...
298	Shud Kaliyan	9
291	ShaymSharang	9
81	Dharmbati	9
264	RatanDeep	9
328	YogJyoti	9

329 rows × 2 columns

In [75]:
```python
model_tri.cluster_centers_
```

Out[75]:
```
array([[14.82302794, 18.1333827 , 14.66954771, ..., 54.3606451 ,
        35.75294648, 56.16961147],
       [38.96977299, 23.3329206 , 31.67890306, ..., 42.9998839 ,
         9.07801315, 33.2063983 ],
       [29.90931228,  6.4132776 , 21.43869122, ..., 29.21748687,
        28.34598269, 26.79043205],
       ...,
       [43.4443412 , 19.68005223, 37.62947567, ..., 30.78168892,
        28.07094535, 20.74844774],
       [19.38233902, 16.85852634, 12.11501722, ..., 27.42108552,
        31.59491648, 20.09797163],
       [37.70996475, 19.11336703, 36.55794675, ..., 51.92547067,
        21.29365919, 54.3090882 ]])
```

Regression on trigram dataset

Extract top 20 best correlated features with the cluster

In [76]:
```python
df_tri.corr().iloc[:,-1].abs().sort_values(ascending=False).iloc[0:20]
```

Out[76]:
```
target                1.000000
Bilaskhani            0.321819
Sawairee              0.263021
Chanderika Bharivi    0.261804
Komal Ashawari        0.260151
SerswatiSarang        0.257546
Yamani Bilawal        0.255419
Dhoulamberi           0.252811
Homeshikha            0.252425
Kokillpancham         0.249330
Mohan kaliyan         0.245981
Komal Desi            0.245120
BhadureeTodi          0.244037
Dev Gandhar           0.243408
YashPriya             0.232789
HemBihag              0.232398
ShaymSharang          0.218722
SerswatRanjani        0.217177
ChampaKali            0.216949
Pahadi Kamodi         0.216506
Name: target, dtype: float64
```

In [77]:
```python
fea_col_tri=df_tri.corr().iloc[:,-1].abs().sort_values(ascending=False).iloc[1:21].index
```

In [78]:
```python
fea_col_tri
```

Out[78]:
```
Index(['Bilaskhani', 'Sawairee', 'Chanderika Bharivi', 'Komal Ashawari',
       'SerswatiSarang', 'Yamani Bilawal', 'Dhoulamberi', 'Homeshikha',
       'Kokillpancham', 'Mohan kaliyan', 'Komal Desi', 'BhadureeTodi',
       'Dev Gandhar', 'YashPriya', 'HemBihag', 'ShaymSharang',
       'SerswatRanjani', 'ChampaKali', 'Pahadi Kamodi', 'Durga'],
      dtype='object')
```

Create new dataset of top 20 correlated features with the target

In [79]:
```python
df_tri_new=df_tri[fea_col_tri]
```

In [80]:
```python
df_tri_new
```

Out[80]:

	Bilaskhani	Sawairee	Chanderika Bharivi	Komal Ashawari	SerswatiSarang	Yamani Bilawal	Dhoulamberi	Homeshikha	Kokillpancham	Mohan kaliyan	Ko D
0	38.678159	25.903668	22.538855	39.623226	64.683847	78.835271	32.741411	51.293274	41.737274	54.863467	27.964
1	1.000000	3.000000	29.051678	3.000000	19.364917	17.748239	1.000000	19.052559	15.652476	23.494680	16.309
2	1.000000	6.164414	8.246211	17.860571	55.398556	34.713110	1.000000	65.230361	21.494185	33.630343	5.099
3	34.871192	39.661064	26.153394	58.642988	24.576411	22.978251	35.227830	22.803509	55.812185	25.592968	24.020
4	34.496377	63.702433	39.433488	56.044625	22.248595	55.713553	50.764161	18.547237	23.494680	7.141428	23.430
...	...	...	...	...	...	...	...	...	...	...	
324	0.000000	60.753601	9.380832	4.582576	70.866071	0.000000	28.195744	92.455395	11.269428	89.833179	4.472
325	1.000000	16.822604	13.564660	5.000000	80.796040	90.432295	20.736441	96.710909	11.661904	76.648549	5.099
326	11.180340	7.141428	45.354162	14.317821	73.878278	54.249424	45.166359	71.672868	5.744563	53.422842	19.570
327	71.239034	50.000000	3.316625	54.470175	22.022716	29.949958	40.853396	16.248077	34.899857	35.986108	56.621
328	4.000000	7.681146	33.600595	0.000000	60.199668	78.663842	44.418465	68.760454	7.071068	69.036222	7.483

329 rows × 20 columns

Spilting dataset in training and testing

In [81]:
```python
X_train,X_test,Y_train,Y_test=train_test_split(df_tri_new,p_tri,test_size=0.3,random_state=0)
```

Dimension reduction using PCA

In [82]:
```python
pca=PCA(n_components=2)
```

In [83]:
```python
X_train=pca.fit_transform(X_train)
```

In [84]:
```python
X_test=pca.transform(X_test)
```

Regression Model

In [85]:
```python
lm_model_tri = LinearRegression().fit(X_train,Y_train)
```

In [86]:
```python
lm_model_pred_tri=lm_model_tri.predict(X_test)
```

In [87]:
```python
lm_model_pred_tri=np.round(lm_model_pred_tri)
```

Accuaracy of regression model

In [88]:
```python
metrics.mean_absolute_error(Y_test,lm_model_pred_tri)
```

Out[88]:
```
2.3636363636363638
```

In [89]:
```python
metrics.mean_squared_error(Y_test,lm_model_pred_tri)
```

Out[89]:
```
7.838383838383838
```

In [90]:
```python
metrics.r2_score(Y_test,lm_model_pred_tri)
```

Out[90]:
```
0.13000543576734913
```