In [1]:
```
import pandas as pd
import numpy as np
from sklearn import linear_model
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from matplotlib import pyplot as plt
from sklearn.preprocessing import OneHotEncoder
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import OrdinalEncoder
f1 = pd.read_csv('2016.csv')
f2 = pd.read_csv('2017.csv')
f3 = pd.read_csv('2018.csv')
f4 = pd.read_csv('2019.csv')
f5 = pd.read_csv('2020.csv')
f6 = pd.read_csv('2021.csv')
test = pd.read_csv('2022.csv')
data = pd.concat([f1,f2,f3,f4,f5,f6,test])
data.head()
```

Out[1]:

	Institute	Academic Program Name	Quota	Seat Type	Gender	Opening Rank	Closing Rank	Year	Round
0	Indian Institute of Technology Bhubaneswar	Civil Engineering (4 Years, Bachelor of Techno...	AI	OPEN	NaN	3533.0	5947.0	2016	1
1	Indian Institute of Technology Bhubaneswar	Civil Engineering (4 Years, Bachelor of Techno...	AI	OBC-NCL	NaN	1829.0	2213.0	2016	1
2	Indian Institute of Technology Bhubaneswar	Civil Engineering (4 Years, Bachelor of Techno...	AI	SC	NaN	663.0	1023.0	2016	1
3	Indian Institute of Technology Bhubaneswar	Civil Engineering (4 Years, Bachelor of Techno...	AI	ST	NaN	331.0	357.0	2016	1
4	Indian Institute of Technology Bhubaneswar	Civil Engineering and M. Tech. in Structural E...	AI	OPEN	NaN	5408.0	6561.0	2016	1

In [2]:
```
data.dtypes['Opening Rank']
```

Out[2]:
```
dtype('O')
```

In [3]:
```
data.isnull().sum()
```

Out[3]:
```
Institute                    0
Academic Program Name        1
Quota                        1
Seat Type                    2
Gender                   55781
Opening Rank                 4
Closing Rank                 4
Year                         0
Round                        0
dtype: int64
```

In [4]:
```
import re
def fun(x):
    x = str(x)
    if len(x)<1:
        return 0
    num = re.findall(r'\b\d+\b', x)#inp_str.split()
    return num
data["Opening Rank"]=data["Opening Rank"].apply(fun).str[0]
data["Closing Rank"] = data["Closing Rank"].apply(fun).str[0]
data = data.dropna(subset = ['Quota','Academic Program Name','Seat Type','Opening
data['Gender']=data[['Gender']].fillna('Neutral')
data.head()
```

Out[4]:

	Institute	Academic Program Name	Quota	Seat Type	Gender	Opening Rank	Closing Rank	Year	Round
0	Indian Institute of Technology Bhubaneswar	Civil Engineering (4 Years, Bachelor of Techno...	AI	OPEN	Neutral	3533	5947	2016	1
1	Indian Institute of Technology Bhubaneswar	Civil Engineering (4 Years, Bachelor of Techno...	AI	OBC-NCL	Neutral	1829	2213	2016	1
2	Indian Institute of Technology Bhubaneswar	Civil Engineering (4 Years, Bachelor of Techno...	AI	SC	Neutral	663	1023	2016	1
3	Indian Institute of Technology Bhubaneswar	Civil Engineering (4 Years, Bachelor of Techno...	AI	ST	Neutral	331	357	2016	1
4	Indian Institute of Technology Bhubaneswar	Civil Engineering and M. Tech. in Structural E...	AI	OPEN	Neutral	5408	6561	2016	1

In [5]:
```
data["Opening Rank"] = pd.to_numeric(data["Opening Rank"],errors='coerce')
data["Closing Rank"] = pd.to_numeric(data["Closing Rank"],errors='coerce')
data.isnull().sum()
```

Out[5]:
```
Institute                0
Academic Program Name    0
Quota                    0
Seat Type                0
Gender                   0
Opening Rank             0
Closing Rank             0
Year                     0
Round                    0
dtype: int64
```

In [6]:
```
data.head(2)
```

Out[6]:

	Institute	Academic Program Name	Quota	Seat Type	Gender	Opening Rank	Closing Rank	Year	Round
0	Indian Institute of Technology Bhubaneswar	Civil Engineering (4 Years, Bachelor of Techno...	AI	OPEN	Neutral	3533	5947	2016	1
1	Indian Institute of Technology Bhubaneswar	Civil Engineering (4 Years, Bachelor of Techno...	AI	OBC-NCL	Neutral	1829	2213	2016	1

In [7]:
```
#for lebling Output column
arr1 =data['Institute'].unique()
arr2=data['Academic Program Name'].unique()
res1 = {k: v for v, k in enumerate(arr1)}
res2 = {k: v for v, k in enumerate(arr2)}
def label1(x):
    return res1[x]+1
def label2(x):
    return res2[x]+1
data['Institute'] = data['Institute'].apply(label1)
data["Academic Program Name"] = data['Academic Program Name'].apply(label2)
data.head(2)
```

Out[7]:

	Institute	Academic Program Name	Quota	Seat Type	Gender	Opening Rank	Closing Rank	Year	Round
0	1	1	AI	OPEN	Neutral	3533	5947	2016	1
1	1	1	AI	OBC-NCL	Neutral	1829	2213	2016	1

In [8]:
```
df = data.copy()
df.head(2)
```

Out[8]:

	Institute	Academic Program Name	Quota	Seat Type	Gender	Opening Rank	Closing Rank	Year	Round
0	1	1	AI	OPEN	Neutral	3533	5947	2016	1
1	1	1	AI	OBC-NCL	Neutral	1829	2213	2016	1

Using Ordinal Encoding

In [9]:
```
Gender = df["Gender"].unique()
Seat = df["Seat Type"].unique()
Quota = df["Quota"].unique()
gender = {k: v for v, k in enumerate(Gender)}
seat = {k: v for v, k in enumerate(Seat)}
quota = {k: v for v, k in enumerate(Quota)}
#giving labels to gender, quota and Seat Type.
def l1(x):
    return quota[x]+1
def l2(x):
    return seat[x]+1
def l3(x):
    return gender[x]+1
df['Quota'] = df['Quota'].apply(l1)
df['Seat Type'] = df['Seat Type'].apply(l2)
df['Gender'] = df['Gender'].apply(l3)
#laveling.
# df['Opening Rank'] = df['Opening Rank']/1000
# df['Closing Rank'] = df['Closing Rank']/1000
# df['Year'] = df['Year']/2000
df.head(2)
```

Out[9]:

	Institute	Academic Program Name	Quota	Seat Type	Gender	Opening Rank	Closing Rank	Year	Round
0	1	1	1	1	1	3533	5947	2016	1
1	1	1	1	2	1	1829	2213	2016	1

Now pre prosessing has been completed. 2 Methods we can use

1. On Quota, Gender, Seat Type -> we can apply ordinal encoding. 2.nominal encoding(before applying nominal we have to split test & train data.)

Using Linear Regression

In [13]:
```
from sklearn.preprocessing import StandardScaler
x = df.iloc[:,2:]
y = df.iloc[:,1:2]
#y = y.to_frame()
scaler = StandardScaler()
reg = linear_model.LinearRegression()
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size= 0.2, shuffle=
scaler.fit(X_train)
X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)
reg = reg.fit(X_train, y_train)
reg.score(X_test, y_test)
```

Out[13]:
```
0.02676330433380103
```

In [16]:
```
# x = dff.iloc[:,2:]
# y = dff.iloc[:,0:1]
# # x = x.to_frame()
# reg = linear_model.LinearRegression()
# X_train, X_test, y_train, y_test = train_test_split(x, y, test_size= 0.2, shuffle
# y_train.head(2)
```

In [17]:
```
# from sklearn.compose import ColumnTransformer
# transformer = ColumnTransformer(transformers=[
#     ('tnf1',OneHotEncoder(sparse=False,drop='first'),[ 'Quota', 'Seat Type', 'Ge
# ],remainder='passthrough')
# X_train=transformer.fit_transform(X_train)
# X_test=transformer.transform(X_test)
# reg = linear_model.LinearRegression()
# reg = reg.fit(X_train, y_train)
# reg.score(X_test, y_test)
```

Using DecisionTreeClassifier

In [18]:
```
# Load libraries
import pandas as pd
from sklearn.tree import DecisionTreeClassifier # Import Decision Tree Classifier
from sklearn.model_selection import train_test_split # Import train_test_split fun
from sklearn import metrics #Import scikit-learn metrics module for accuracy calcu

clf = DecisionTreeClassifier()

# Train Decision Tree Classifer
clf = clf.fit(X_train,y_train)
y_pred = clf.predict(X_test)
#Predict the response for test dataset
clf.score(X_test, y_test)
```

Out[18]:
```
0.7242510460251046
```

In [19]:
```
y_pred=clf.predict(X_test)
y_pred
```

Out[19]:
```
array([ 4, 52,  4, ..., 10,  5,  6], dtype=int64)
```

In [27]:
```
y_test.head(2)
```

Out[27]:

	Academic Program Name
7228	119
48870	93

Y-column value to array

In [25]:
```
y_test.values.ravel()
```

Out[25]:
```
array([119,  93,   6, ...,  10,  17,   6], dtype=int64)
```

Performance Report

In [23]:
```
from sklearn.metrics import classification_report
print(classification_report(y_test.values.ravel(), y_pred))
```

	precision	recall	f1-score	support
1	0.73	0.71	0.72	5839
2	0.78	0.80	0.79	50
3	0.73	0.79	0.76	61
4	0.82	0.82	0.82	8250
5	0.75	0.75	0.75	4281
6	0.73	0.72	0.73	6105
7	0.71	0.72	0.72	6175
8	0.85	0.82	0.83	55
9	0.73	0.81	0.77	54
10	0.65	0.66	0.65	1795
11	0.81	0.79	0.80	467
12	0.70	0.71	0.71	3534
13	0.82	0.82	0.82	145
14	0.92	1.00	0.96	22
15	1.00	1.00	1.00	30
16	0.88	0.94	0.91	65
17	0.79	0.80	0.79	730
18	1.00	1.00	1.00	4
19	0.91	0.89	0.90	70
20	0.87	0.80	0.83	134
21	0.85	0.96	0.90	23
22	0.75	0.94	0.83	16
23	0.92	0.92	0.92	25
24	0.75	0.75	0.75	198
25	0.85	0.81	0.83	258
26	0.87	0.89	0.88	494
27	0.92	0.90	0.91	68
28	0.87	0.86	0.87	347
29	0.92	0.86	0.89	118
30	0.62	0.61	0.61	545
31	0.54	0.54	0.54	171
32	0.90	0.91	0.90	114
33	0.81	0.76	0.79	58
34	0.76	0.73	0.75	75
35	0.82	0.69	0.75	48
36	0.85	0.82	0.84	1555
37	0.67	0.72	0.70	43
38	0.55	0.54	0.55	395
39	0.74	0.74	0.74	70
40	0.89	0.87	0.88	38
41	0.86	0.97	0.91	64
42	0.94	0.94	0.94	88
43	0.86	0.89	0.88	85
44	0.84	0.87	0.85	30
45	0.88	0.89	0.89	65
46	0.78	0.84	0.81	58
47	0.76	0.78	0.77	100
48	0.84	0.87	0.86	79
49	0.88	0.84	0.86	62
50	0.79	0.82	0.81	68
51	0.92	0.88	0.90	83
52	0.68	0.71	0.70	167
53	0.60	0.62	0.61	692
54	0.70	0.72	0.71	176
55	0.78	0.78	0.78	54
56	0.83	0.76	0.79	68
57	0.71	0.71	0.71	62
58	0.56	0.59	0.57	320
59	0.89	0.86	0.87	36
60	0.97	0.93	0.95	72
61	0.57	0.64	0.60	194
62	0.82	0.75	0.78	67

63	0.66	0.72	0.69	61
64	0.87	0.90	0.89	165
65	0.62	0.65	0.63	300
66	0.85	0.88	0.87	60
67	0.81	0.81	0.81	75
68	0.84	0.84	0.84	58
69	0.75	0.81	0.78	67
70	0.74	0.82	0.78	68
71	0.88	0.93	0.91	92
72	0.88	0.85	0.86	75
73	0.78	0.81	0.80	130
74	0.79	0.78	0.78	58
75	0.77	0.81	0.79	21
76	0.65	0.72	0.68	129
77	0.98	0.91	0.95	47
78	0.61	0.60	0.61	1191
79	0.72	0.83	0.77	65
80	0.89	0.90	0.89	69
81	0.70	0.70	0.70	37
82	0.72	0.67	0.69	57
83	0.68	0.79	0.73	61
84	0.60	0.65	0.62	416
85	0.78	0.71	0.74	59
86	0.80	0.82	0.81	73
87	0.78	0.66	0.71	32
88	0.64	0.56	0.60	52
89	0.79	0.80	0.79	70
90	0.63	0.59	0.61	46
91	0.65	0.62	0.64	58
92	0.61	0.63	0.62	169
93	0.53	0.49	0.51	49
94	0.83	1.00	0.91	5
95	0.90	0.83	0.86	72
96	0.86	0.88	0.87	80
97	0.50	0.50	0.50	2
98	0.60	0.60	0.60	5
99	0.71	0.54	0.62	59
100	0.65	0.63	0.64	117
101	0.64	0.69	0.66	51
102	0.40	1.00	0.57	2
103	0.75	0.60	0.67	5
104	0.73	0.70	0.71	57
105	0.93	0.84	0.88	81
106	0.44	0.40	0.42	168
107	0.65	0.64	0.65	335
108	0.57	0.53	0.55	257
109	0.66	0.67	0.66	1545
110	0.62	0.52	0.57	67
111	0.59	0.60	0.59	77
112	0.69	0.71	0.70	112
113	0.77	0.74	0.76	23
114	0.60	0.57	0.59	411
115	0.69	0.70	0.70	1879
116	0.76	0.70	0.73	158
117	0.71	0.70	0.70	355
118	0.50	0.14	0.22	7
119	0.57	0.52	0.54	246
120	0.48	0.49	0.48	260
121	0.64	0.70	0.67	204
122	0.56	0.47	0.51	64
123	0.42	0.62	0.50	8
124	0.88	0.88	0.88	8
125	0.50	0.50	0.50	4
126	0.33	0.33	0.33	6

127	0.71	0.91	0.80	11
128	0.85	0.80	0.82	55
129	0.60	0.43	0.50	7
130	0.67	0.67	0.67	9
131	0.49	0.53	0.51	120
132	0.60	0.64	0.62	110
133	0.39	0.38	0.39	87
134	0.90	0.90	0.90	10
135	0.68	0.66	0.67	64
136	0.64	0.70	0.67	10
137	0.89	0.73	0.80	11
138	0.67	0.29	0.40	7
139	0.90	0.64	0.75	14
140	0.50	0.40	0.44	10
141	0.80	0.57	0.67	7
142	1.00	0.50	0.67	10
143	0.77	0.83	0.80	12
144	0.80	0.67	0.73	6
145	1.00	0.55	0.71	11
146	0.62	0.56	0.59	9
147	0.62	0.62	0.62	13
148	0.73	0.80	0.76	10
149	0.56	0.71	0.63	7
150	0.62	0.89	0.73	9
151	0.88	0.64	0.74	11
152	0.29	0.40	0.33	15
153	0.54	0.78	0.64	9
154	0.33	1.00	0.50	1
155	0.50	0.43	0.46	7
156	0.67	0.50	0.57	8
157	0.29	1.00	0.44	2
158	0.40	1.00	0.57	4
159	0.46	0.50	0.48	12
160	0.35	0.26	0.30	117
161	0.28	0.28	0.28	40
162	0.28	0.20	0.24	44
163	0.49	0.43	0.46	51
164	0.33	0.23	0.27	26
165	0.50	0.53	0.51	157
166	0.23	0.26	0.24	77
167	0.38	0.28	0.32	18
168	0.14	0.20	0.17	15
169	0.33	0.29	0.31	52
170	0.82	0.83	0.82	48
171	0.76	0.82	0.79	45
172	0.69	0.92	0.79	12
173	0.69	0.67	0.68	46
174	0.85	0.68	0.75	65
175	0.57	0.60	0.58	47
176	0.63	0.57	0.60	46
177	0.69	0.80	0.74	41
178	0.63	0.69	0.66	48
179	0.57	0.51	0.54	57
180	0.67	0.50	0.57	8
181	0.67	1.00	0.80	8
182	0.48	0.50	0.49	62
183	0.33	0.57	0.42	7
184	0.27	1.00	0.43	3
185	0.34	0.35	0.34	52
186	0.76	0.76	0.76	34
187	0.93	0.91	0.92	43
188	0.89	0.85	0.87	60
189	0.44	0.57	0.50	7
190	0.35	0.86	0.50	7

191	0.70	0.67	0.68	21
192	0.67	0.50	0.57	12
193	0.47	0.58	0.52	12
194	0.75	0.55	0.63	11
195	0.36	0.41	0.38	61
196	0.15	0.14	0.15	28
197	0.50	0.55	0.52	11
198	0.89	0.89	0.89	35
199	0.65	0.80	0.71	25
200	0.89	0.92	0.90	62
201	0.84	0.90	0.87	77
202	0.84	0.70	0.76	67
203	0.66	0.67	0.67	46
204	0.74	0.76	0.75	92
205	0.37	0.27	0.31	26
206	0.84	0.86	0.85	59
207	0.82	0.76	0.78	82
208	0.59	0.67	0.63	33
209	0.70	0.76	0.73	84
210	0.65	0.68	0.67	38
211	0.76	0.82	0.78	38
212	0.69	0.76	0.72	50
213	0.81	0.83	0.82	36
214	0.65	0.58	0.61	38
215	0.64	0.75	0.69	24
216	0.50	0.57	0.53	23
217	0.77	0.80	0.79	41
218	0.67	0.45	0.54	22
219	0.70	0.82	0.76	17
220	0.85	0.78	0.82	37
221	1.00	0.80	0.89	20
222	0.71	1.00	0.83	10
223	0.73	0.89	0.80	9
224	0.92	0.92	0.92	25
225	0.69	0.79	0.73	14
226	0.86	0.86	0.86	22
227	0.85	0.76	0.80	29
228	0.58	0.47	0.52	15
229	0.77	0.74	0.76	23
230	0.67	0.78	0.72	23
231	0.60	0.60	0.60	15
232	0.58	0.70	0.64	20
233	0.29	0.45	0.35	20
234	0.40	0.43	0.41	14
235	0.44	0.40	0.42	10
236	0.20	0.20	0.20	15
237	0.35	0.60	0.44	10
238	0.22	0.20	0.21	10
239	0.38	0.42	0.40	12
240	0.42	0.77	0.54	13
241	0.46	0.33	0.39	18
242	0.92	0.75	0.83	16
243	0.33	0.43	0.38	7
244	1.00	0.33	0.50	6
245	0.50	0.50	0.50	8
246	0.73	0.89	0.80	9
247	0.69	0.69	0.69	13
248	0.20	0.29	0.24	7
249	0.89	0.75	0.81	32
250	0.82	0.90	0.86	20
251	0.60	0.71	0.65	21
252	0.67	0.89	0.76	18
253	0.25	0.33	0.29	6
254	0.71	0.62	0.67	8

```
              255       0.86      0.78      0.82        23
              256       0.79      0.85      0.81        13
              257       0.80      0.86      0.83        14
              258       0.45      0.62      0.53        16
              259       0.43      0.38      0.40         8
              260       0.73      0.79      0.76        14
              261       0.71      0.56      0.63         9
              262       0.43      0.38      0.40         8
              263       0.60      0.46      0.52        13
              264       0.75      0.75      0.75        12
              265       0.33      0.25      0.29         8

    accuracy                           0.72     59750
   macro avg       0.67      0.69      0.67     59750
weighted avg       0.73      0.72      0.72     59750
```

Confusion Matrix

In [24]:
```
from sklearn.metrics import confusion_matrix
confusion_matrix = confusion_matrix(y_test.values.ravel(), y_pred)
print(confusion_matrix)
```

```
[[4151    0    2 ...    0    0    0]
 [   2   40    0 ...    0    0    0]
 [   0    0   48 ...    0    0    0]
 ...
 [   1    0    0 ...    6    0    0]
 [   0    0    0 ...    0    9    0]
 [   0    0    0 ...    0    0    2]]
```

Conclusion

Using DecisionTreeClassifier, the model's accuracy was72%; precision, recall, and f1-score were73%,72%, and 72% after the model had been trained.

In [ ]: