project-data-science-2025-rohit

March 25, 2025

0.1 Cutoff Insight – IIT & NIT Admission Predictor

1 About the Dataset

Context : Joint Entrance Examination – Main (JEE-Main), formerly All India Engineering Entrance Examination (AIEEE), is an Indian standardised computer-based test for admission to various technical undergraduate programs in engineering, architecture, and planning across colleges in India. The exam is conducted by the JEE Apex Board for Admission for B.Tech, B.Arch, etc. programs in the premier technical institutes such as the National Institutes of Technology and Indian Institutes of Information Technology are based on the rank secured in the JEE-Main. It is usually conducted twice every year.

IITs and NITs : The Indian Institutes of Technology (IITs) are the globally appreciated engineering and technological institutes in India. IITs have maintained quality education and internationally acclaimed research facilities. IIT JEE Exam is the most popular engineering admission entrance test conducted in India. National Institute of Technology (NITs) are premier engineering colleges in India offering admission to degree courses at both undergraduate and postgraduate level.

2 About the files

year - The year of the conducted JEE exam

institute_type - Type of Institute (IIT or NIT)

round_no - The counseling round number

quota - The reservation quota

AI : All-India

HS : Home-State

OS : Other-State

AP : Andhra Pradesh

GO : Goa

JK : Jammu & Kashmir

LA : Ladakh

pool - The gender quota

institute_short - THe short name of the Institution

program_name - The name of the program/stream

program_duration - The duration of the course (in years)

degree_short - The name of the degree (Abbreviated)

category - The caste category

GEN : General

OBC-NCL : Other Backward Classes-Non Creamy Layer

SC : Scheduled Castes

ST : Scheduled Tribes

GEN-PWD : General & Persons with Disabilities

OBC-NCL-PWD : Other Backward Classes & Persons with Disabilities

SC-PWD : Scheduled Castes & Persons with Disabilities

ST-PWD : Scheduled Tribes & Persons with Disabilities

GEN-EWS : General & Economically Weaker Section

GEN-EWS-PWD : General & Economically Weaker Section & Persons with Disability

opening_rank - The opening (starting) rank for getting admission in the institution

closing_rank - The closing (ending) rank for getting admission in the institution

is_preparatory - If admission to a preparatory course is available - 0 : No, 1 : Yes

Acknowledgement: *This data is provided by KAGGLE DATASET website*

EDA: IIT-NIT Category-Wise Cutoff Data

2.1 Before we get started! Let's get our Imports

[69]:
```
#imports

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

2.2 Let's get the Data

[70]:
```
# Read data in a dataframe

df = pd.read_csv(r"C:\Users\ROHIT\Downloads\data.csv\data.csv")
df.tail()
```

[70]:
```
        year institute_type  round_no quota            pool institute_short  \
64953   2021            NIT         1    JK     Female-Only    NIT-Srinagar
```

```
64954  2021            NIT         1    JK  Gender-Neutral    NIT-Srinagar
64955  2021            NIT         1    JK     Female-Only    NIT-Srinagar
64956  2021            NIT         1    LA  Gender-Neutral    NIT-Srinagar
64957  2021            NIT         1    LA     Female-Only    NIT-Srinagar

                                   program_name program_duration  \
64953  Electronics and Communication Engineering          4 Years
64954  Electronics and Communication Engineering          4 Years
64955  Electronics and Communication Engineering          4 Years
64956  Electronics and Communication Engineering          4 Years
64957  Electronics and Communication Engineering          4 Years

      degree_short category  opening_rank  closing_rank  is_preparatory  \
64953       B.Tech       SC         14185         24048               0
64954       B.Tech       ST          2736          4171               0
64955       B.Tech       ST         10870         10870               0
64956       B.Tech      GEN        166453        265454               0
64957       B.Tech      GEN        215054        215054               0

       Unnamed: 13
64953          NaN
64954          NaN
64955          NaN
64956          NaN
64957          NaN
```

[71]:
```
df.head()
```

[71]:
```
   year institute_type  round_no quota            pool institute_short  \
0  2016            IIT         6    AI  Gender-Neutral      IIT-Bombay
1  2016            IIT         6    AI  Gender-Neutral      IIT-Bombay
2  2016            IIT         6    AI  Gender-Neutral      IIT-Bombay
3  2016            IIT         6    AI  Gender-Neutral      IIT-Bombay
4  2016            IIT         6    AI  Gender-Neutral      IIT-Bombay

            program_name program_duration degree_short category  opening_rank  \
0  Aerospace Engineering          4 Years       B.Tech      GEN           838
1  Aerospace Engineering          4 Years       B.Tech  OBC-NCL           408
2  Aerospace Engineering          4 Years       B.Tech       SC           297
3  Aerospace Engineering          4 Years       B.Tech       ST            79
4  Aerospace Engineering          4 Years       B.Tech  GEN-PWD            94

   closing_rank  is_preparatory  Unnamed: 13
0          1841               0          NaN
1          1098               0          NaN
2           468               0          NaN
3           145               0          NaN
```

```
4           94                0          NaN
```

2.3 Exploring the Data

[72]:
```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 64958 entries, 0 to 64957
Data columns (total 14 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   year              64958 non-null  int64
 1   institute_type    64958 non-null  object
 2   round_no          64958 non-null  int64
 3   quota             64958 non-null  object
 4   pool              64958 non-null  object
 5   institute_short   64958 non-null  object
 6   program_name      64958 non-null  object
 7   program_duration  64958 non-null  object
 8   degree_short      64958 non-null  object
 9   category          64958 non-null  object
 10  opening_rank      64958 non-null  int64
 11  closing_rank      64958 non-null  int64
 12  is_preparatory    64958 non-null  int64
 13  Unnamed: 13       0 non-null      float64
dtypes: float64(1), int64(5), object(8)
memory usage: 6.9+ MB
```

[73]:
```python
df.describe()
```

[73]:
```
               year      round_no  opening_rank  closing_rank  is_preparatory  \
count  64958.000000  64958.000000  6.495800e+04  6.495800e+04    64958.000000
mean    2020.421580      2.609348  8.259642e+03  1.070497e+04        0.047631
std        1.149762      2.422558  2.679448e+04  3.788101e+04        0.212985
min     2016.000000      1.000000  0.000000e+00  0.000000e+00        0.000000
25%     2020.000000      1.000000  6.710000e+02  8.320000e+02        0.000000
50%     2021.000000      1.000000  2.309000e+03  2.764500e+03        0.000000
75%     2021.000000      6.000000  6.932000e+03  8.190000e+03        0.000000
max     2021.000000      7.000000  1.082601e+06  1.144790e+06        1.000000

       Unnamed: 13
count          0.0
mean           NaN
std            NaN
min            NaN
25%            NaN
50%            NaN
```

```
75%             NaN
max             NaN
```

[74]:
```
# Shape of the Dataset

df.shape
```

[74]:
```
(64958, 14)
```

[75]:
```
# Columns of the Dataset

Columns = pd.DataFrame(df.columns)
Columns
```

[75]:
```
                    0
0                year
1      institute_type
2            round_no
3               quota
4                pool
5     institute_short
6        program_name
7    program_duration
8        degree_short
9            category
10       opening_rank
11       closing_rank
12     is_preparatory
13        Unnamed: 13
```

[76]:
```
df.drop(columns=["Unnamed: 13"], inplace=True)
df.head()
```

[76]:
```
   year institute_type  round_no quota            pool institute_short  \
0  2016            IIT         6    AI  Gender-Neutral      IIT-Bombay
1  2016            IIT         6    AI  Gender-Neutral      IIT-Bombay
2  2016            IIT         6    AI  Gender-Neutral      IIT-Bombay
3  2016            IIT         6    AI  Gender-Neutral      IIT-Bombay
4  2016            IIT         6    AI  Gender-Neutral      IIT-Bombay

            program_name program_duration degree_short category  opening_rank  \
0  Aerospace Engineering          4 Years       B.Tech      GEN           838
1  Aerospace Engineering          4 Years       B.Tech  OBC-NCL           408
2  Aerospace Engineering          4 Years       B.Tech       SC           297
3  Aerospace Engineering          4 Years       B.Tech       ST            79
4  Aerospace Engineering          4 Years       B.Tech  GEN-PWD            94
```

```
   closing_rank  is_preparatory
0          1841               0
1          1098               0
2           468               0
3           145               0
4            94               0
```

[77]:
```
df.isnull().sum()
```

[77]:
```
year                0
institute_type      0
round_no            0
quota               0
pool                0
institute_short     0
program_name        0
program_duration    0
degree_short        0
category            0
opening_rank        0
closing_rank        0
is_preparatory      0
dtype: int64
```

[78]:
```
# Unique values in quota

Quota = pd.DataFrame(df["quota"].unique())
Quota
```

[78]:
```
    0
0  AI
1  HS
2  OS
3  AP
4  GO
5  JK
6  LA
```

[79]:
```
# Unique values in pool

Pool = pd.DataFrame(df["pool"].unique())
Pool
```

[79]:
```
                0
0  Gender-Neutral
1     Female-Only
```

[80]:

```python
# Various institutions

Institutes = pd.DataFrame(df["institute_short"].unique(), columns =␣
 ↪['Institute'])
Institutes
```

[80]:

	Institute
0	IIT-Bombay
1	IIT-Delhi
2	IIT-Kharagpur
3	IIT-Kanpur
4	IIT-Madras
5	IIT-Roorkee
6	IIT-Guwahati
7	IIT-Indore
8	IIT-Hyderabad
9	IIT-(BHU) Varanasi
10	IIT-Patna
11	IIT-(ISM) Dhanbad
12	IIT-Bhubaneswar
13	IIT-Mandi
14	IIT-Gandhinagar
15	IIT-Ropar
16	IIT-Jodhpur
17	IIT-Tirupati
18	IIT-Bhilai
19	IIT-Dharwad
20	IIT-Goa
21	IIT-Jammu
22	IIT-Palakkad
23	NIT-Warangal
24	NIT-Tiruchirappalli
25	NIT-Uttarakhand
26	NIT-Surat
27	NIT-Nagpur
28	NIT-Andhra-Pradesh
29	NIT-Jalandhar
30	NIT-Jaipur
31	NIT-Bhopal
32	NIT-Allahabad
33	NIT-Calicut
34	NIT-Agartala
35	NIT-Delhi
36	NIT-Durgapur
37	NIT-Goa
38	NIT-Hamirpur
39	NIT-Meghalaya

```
40  NIT-Karnataka-Surathkal
41                NIT-Patna
42             NIT-Nagaland
43           NIT-Puducherry
44               NIT-Raipur
45               NIT-Sikkim
46    NIT-Arunachal-Pradesh
47           NIT-Jamshedpur
48          NIT-Kurukshetra
49              NIT-Manipur
50              NIT-Mizoram
51             NIT-Rourkela
52              NIT-Silchar
53             NIT-Srinagar
```

[81]:

```
# Various types of programs

Programs = pd.DataFrame(df["program_name"].unique(), columns = ['program'])
Programs
```

[81]:

```
                                                 program
0                                  Aerospace Engineering
1                                   Chemical Engineering
2                                              Chemistry
3                                      Civil Engineering
4                       Computer Science and Engineering
..                                                   …
125                               Food Process Engineering
126  Ceramic Engineering and M.Tech Industrial Ceramic
127                                           Life Science
128                          Mathematics and Data Science
129                             Computational Mathematics

[130 rows x 1 columns]
```

[82]:

```
# Various degrees

Degree = pd.DataFrame(df["degree_short"].unique(), columns = {'Degree':0})
Degree
```

[82]:

```
                   Degree
0                  B.Tech
1                     BSc
2  B.Tech + M.Tech (IDD)
3                Int MSc.
4                  B.Arch
5              Int M.Tech
```

```
6                  B.Pharm
7        B.Pharm + M.Pharm
8            BS + MS (IDD)
9                Int Msc.
10                 B.Plan
11   Btech + M.Tech (IDD)
12         BSc + MSc (IDD)
```

[83]:

```python
df["degree_short"].nunique()
```

[83]:

```
13
```

[84]:

```python
Round = pd.DataFrame(df["round_no"].unique())
Round
```

[84]:

```
   0
0  6
1  7
2  1
3  2
```

[85]:

```python
# Unique values in category

Category = pd.DataFrame(df["category"].unique())
Category
```

[85]:

```
             0
0          GEN
1      OBC-NCL
2           SC
3           ST
4      GEN-PWD
5  OBC-NCL-PWD
6       SC-PWD
7       ST-PWD
8      GEN-EWS
9  GEN-EWS-PWD
```

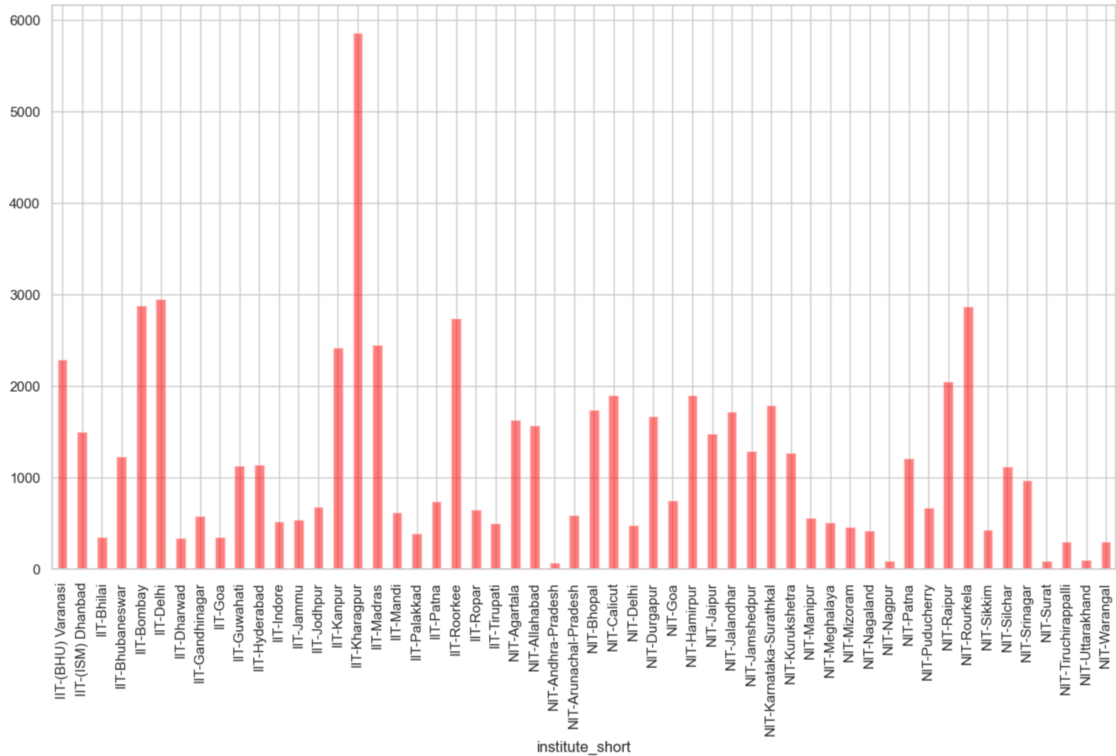
[86]:

```python
print(df.isnull().sum())
```

```
year                0
institute_type      0
round_no            0
quota               0
pool                0
institute_short     0
program_name        0
```

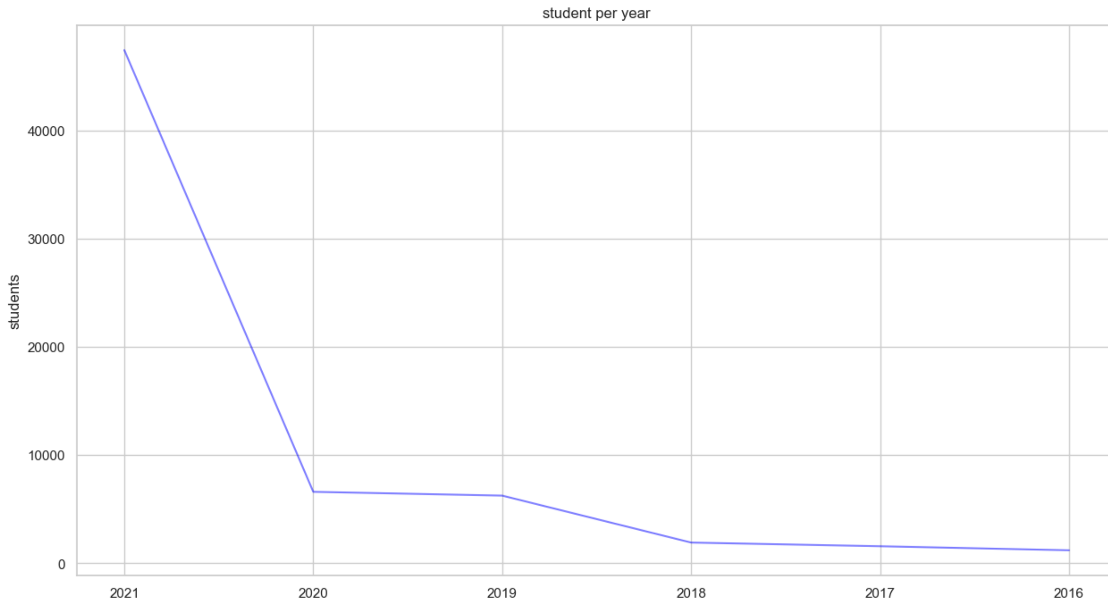
```
program_duration     0
degree_short         0
category             0
opening_rank         0
closing_rank         0
is_preparatory       0
dtype: int64
```

[87]:
```python
#Let's check out All Institutes in the dataset

plt.figure(figsize=(15,8))
institutes = df.groupby(['institute_short']).size().plot(kind = 'bar', color =␣
↪"red", alpha = 0.5)
plt.show()
```



[88]:
```python
# Students admitted per year

plt.figure(figsize=(15,8))
plt.title('student per year')
sns.lineplot(x=['2021','2020','2019','2018','2017','2016'],y=df['year'].
↪value_counts(), color = "blue", alpha = 0.5)
plt.ylabel('students')
```

[88]: Text(0, 0.5, 'students')



2.3.1 1 What does reservation say about admission to these colleges?

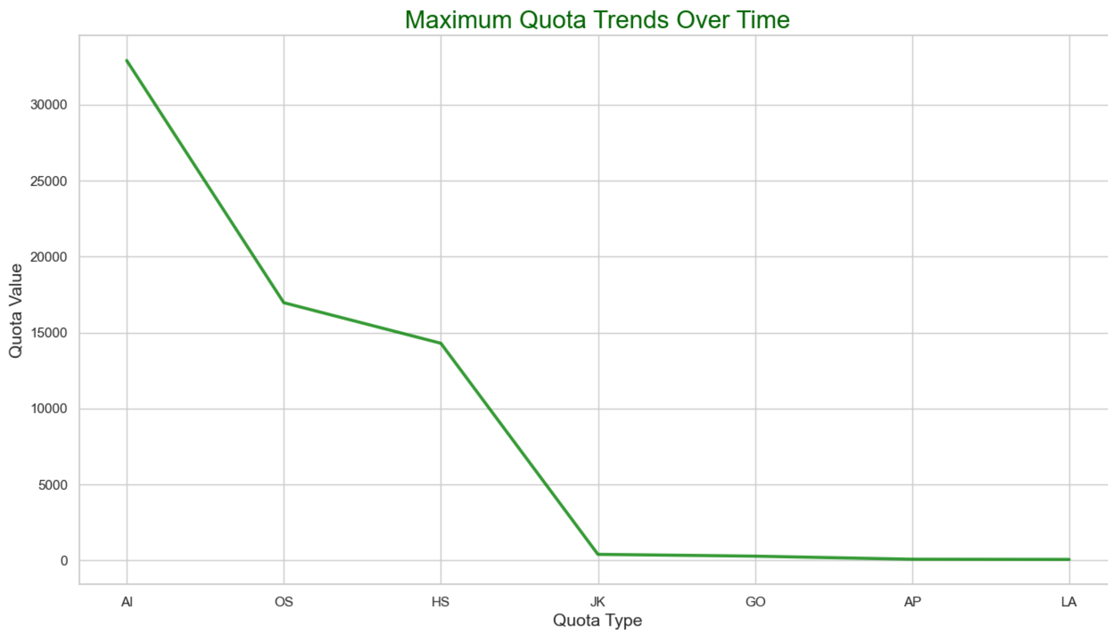
[89]:
```
max_quota = df['quota'].value_counts()
max_quota
```

[89]:
```
quota
AI    32905
OS    16962
HS    14291
JK      393
GO      275
AP       72
LA       60
Name: count, dtype: int64
```

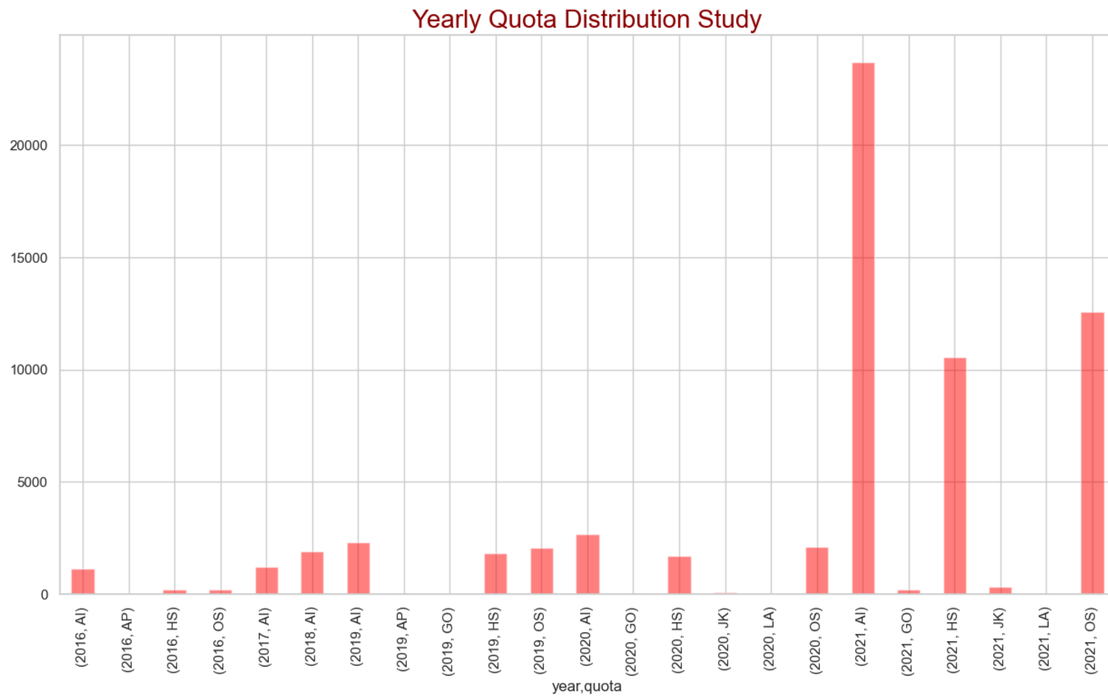
Since, AI (all India) reservation has max counts - it can be a factor contributing to admissions.

[90]:
```
plt.figure(figsize=(15, 8))
sns.lineplot(data=max_quota, color="green", alpha=0.8, linewidth=2.5)

# Adding labels and title
plt.title('Maximum Quota Trends Over Time', fontsize=20, color='darkgreen')
plt.xlabel('Quota Type', fontsize=14)
plt.ylabel('Quota Value', fontsize=14)
```

[90]: Text(0, 0.5, 'Quota Value')



[91]:
```
#Yearly quota study

plt.figure(figsize=(15,8))
year_club = df.groupby(['year', 'quota']).size().plot(kind = 'bar', color =
 ↪"red", alpha = 0.5)
plt.title('Yearly Quota Distribution Study', fontsize=20, color='darkred')
plt.show()
```



2.4 From the above plot, we see that AI - all India quota covers maximum number of students followed by OS - Other State and HS - Home State.

While the second plot depicts that AI(All India) quota played an important role from 2016 to 2018.

2.5 2 What is the most optimum Opening and closing rank in overall years??

Opening Ranks

---

[92]:
```
avg_opening_rank = df['opening_rank'].mean(axis = 0)
avg_open_rank = round(avg_opening_rank)
print("Average opening rank over the years has been - ", avg_open_rank)
```

```
Average opening rank over the years has been -  8260
```

[93]:
```
max_opening_rank = df['opening_rank'].max(axis = 0)
print("Max opening rank over the years has been - ", max_opening_rank)
```

```
Max opening rank over the years has been -  1082601
```

[94]:
```
min_opening_rank = df['opening_rank'].min(axis = 0)
min_open_rank = round(min_opening_rank)
print("Min opening rank over the years has been - ", min_open_rank)
```

```
Min opening rank over the years has been -  0
```

Closing Ranks

[95]:
```
avg_closing_rank = round(df['closing_rank'].mean(axis = 0))
print("Average closing rank over the years has been - ", avg_closing_rank)
```

```
Average closing rank over the years has been -  10705
```

[96]:
```
max_closing_rank = df['closing_rank'].max(axis = 0)
max_close_rank = round(max_closing_rank)
print("Max closing rank over the years has been - ", max_close_rank)
```

```
Max closing rank over the years has been -  1144790
```

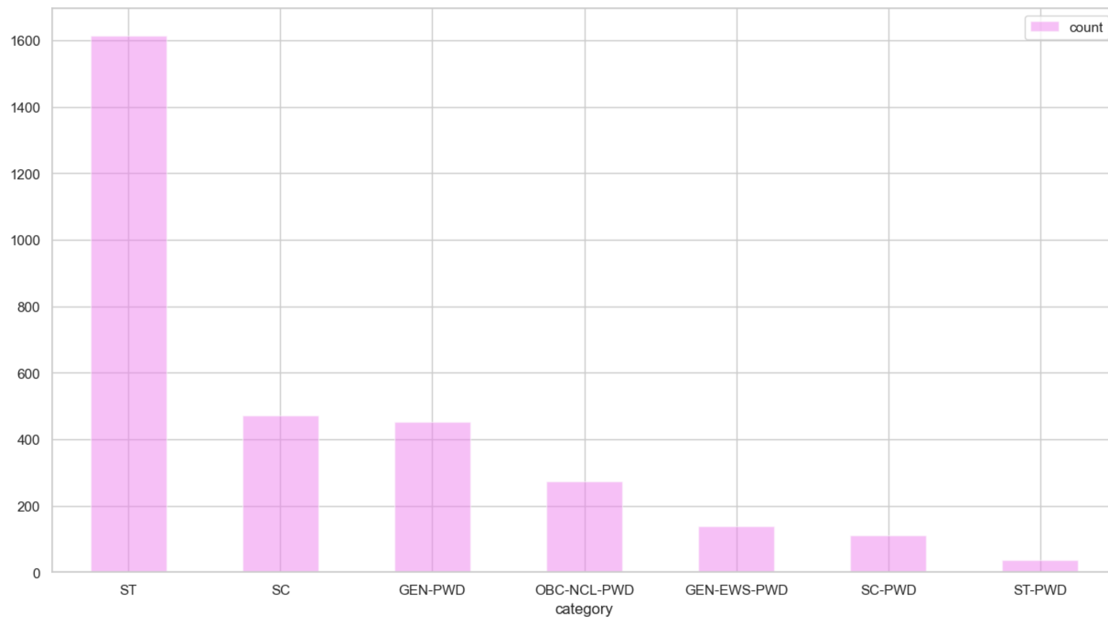
[97]:
```
min_closing_rank = df['closing_rank'].min(axis = 0)
min_close_rank = round(min_closing_rank)
print("Min closing rank over the years has been - ", min_close_rank)
```

```
Min closing rank over the years has been -  0
```

2.6 From analysis of Opening and Closing ranks, we can say that on average if you score a rank around ~8000 then you might become eligible. While keeping in mind the quota factor, the maximum and minimum ranks still vary on a range of large scale till about 10 lakhs.

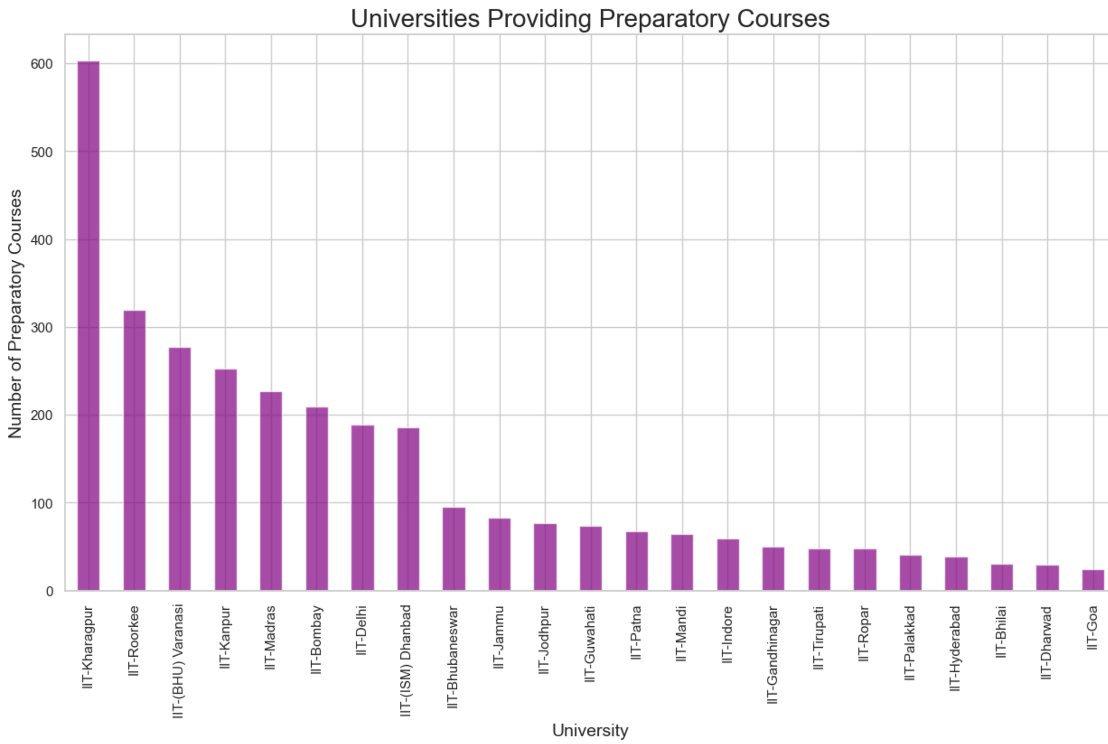
2.7 3 Which universities/colleges provide preparatory courses?

[98]:
```
plt.figure(figsize=(15,8))
category_true = df.loc[df['is_preparatory'] == 1, 'category'].value_counts()
category_plot = category_true.plot(kind = 'bar', color = "violet", alpha = 0.5)

plt.xticks(rotation = 360)
plt.legend()
plt.show()
```



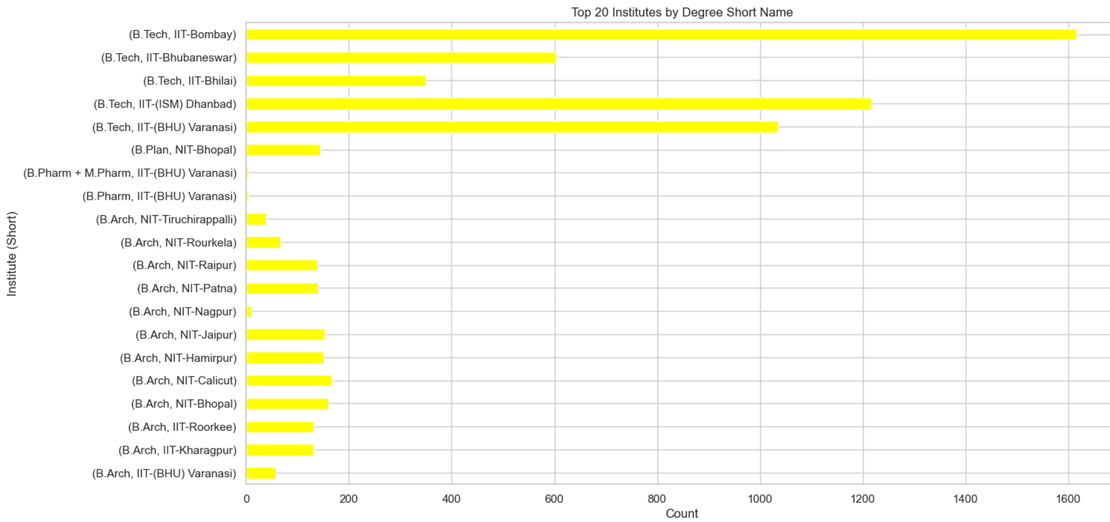
[99]:

```
# universities providing preparatory courses
prep_courses = df[df['is_preparatory'] == 1]

prep_courses_count = prep_courses['institute_short'].value_counts()

plt.figure(figsize=(15, 8))
prep_courses_count.plot(kind='bar', color='purple', alpha=0.7)
plt.title('Universities Providing Preparatory Courses', fontsize=20)
plt.xlabel('University', fontsize=14)
plt.ylabel('Number of Preparatory Courses', fontsize=14)
plt.xticks(rotation=90)
plt.show()
```



2.8 5 Institutes and degrees they provide

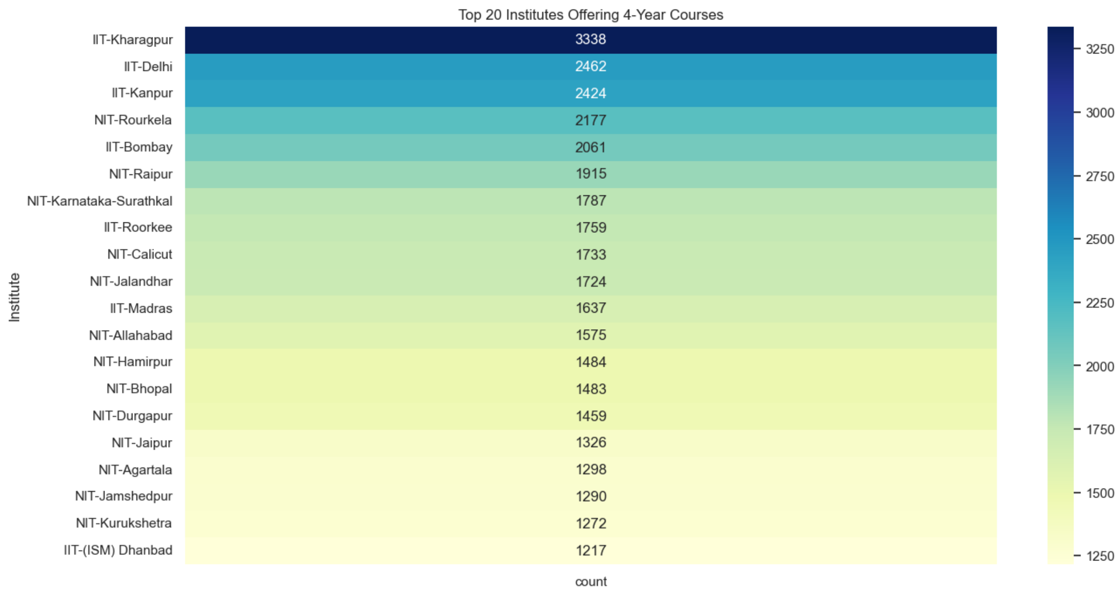
[100]:
```
plt.figure(figsize=(15,8))
year_club = df.groupby(['degree_short','institute_short'])['institute_short'].
  ↪count().head(20).plot(kind='barh', color='yellow')
plt.title("Top 20 Institutes by Degree Short Name")
plt.xlabel("Count")
plt.ylabel("Institute (Short)")
plt.show()
```



2.9 As per above bar graph, B.Tech still seems to be the choice of most students when it comes to getting admission in IIT or NITs.

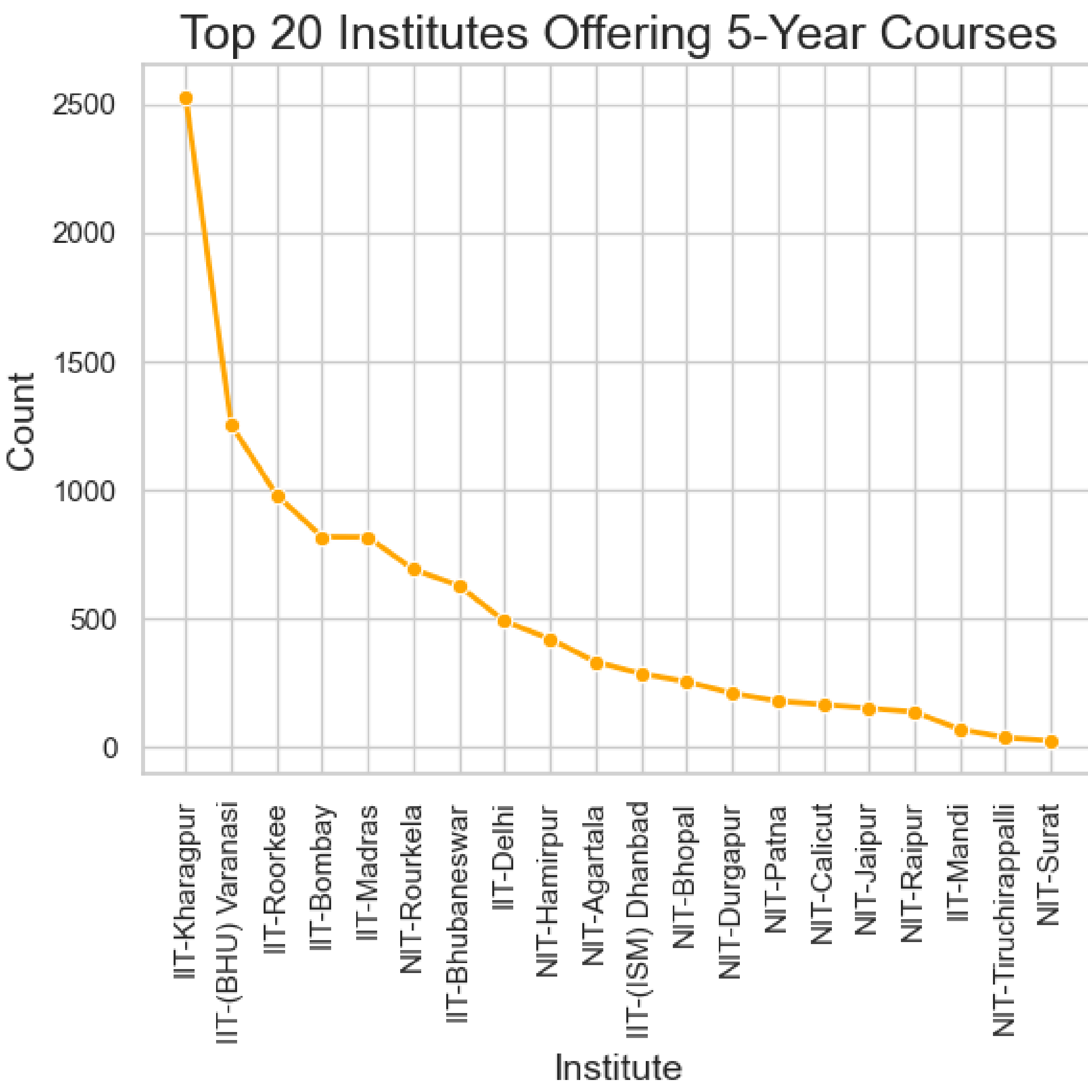
2.10 6 Institutes who provide 4 year and 5 year courses.

[101]:
```
# Filter data for 4-year courses
top_20_institutes = df.loc[df['program_duration'] == '4 Years',␣
↪'institute_short'].value_counts().head(20)

# Plot heatmap
plt.figure(figsize=(15, 8))
sns.heatmap(top_20_institutes.to_frame(), cmap='YlGnBu', annot=True, fmt='d')

# Add labels and title
plt.title("Top 20 Institutes Offering 4-Year Courses")
plt.ylabel("Institute")

plt.show()
```



2.11 IIT Kharagpur has most 4 years courses to offer followed by IIT-Delhi and IIT-Kanpur.

[102]:
```
top20 = df.loc[df['program_duration'] == '5 Years', 'institute_short'].
    ↪value_counts().head(20)

sns.set(style="whitegrid")
sns.lineplot(x=top20.index, y=top20.values, marker='o', color='orange',␣
    ↪linewidth=2)

plt.xticks(rotation=90)
plt.title("Top 20 Institutes Offering 5-Year Courses", fontsize=18)
plt.xlabel("Institute", fontsize=14)
plt.ylabel("Count", fontsize=14)
plt.show()
```



3 Evaluation of the different models

Here we evaluate the model and find its error and accuracy rate based on the given feature and target data. We also find out that how the model works when we give it a specific type of data for the prediction.

[103]:
```
df.head()
```

[103]:
```
   year institute_type  round_no quota           pool institute_short  \
0  2016            IIT         6    AI  Gender-Neutral      IIT-Bombay
1  2016            IIT         6    AI  Gender-Neutral      IIT-Bombay
2  2016            IIT         6    AI  Gender-Neutral      IIT-Bombay
3  2016            IIT         6    AI  Gender-Neutral      IIT-Bombay
```

```
4  2016            IIT          6    AI  Gender-Neutral      IIT-Bombay

           program_name program_duration degree_short category  opening_rank  \
0  Aerospace Engineering          4 Years       B.Tech      GEN           838
1  Aerospace Engineering          4 Years       B.Tech  OBC-NCL           408
2  Aerospace Engineering          4 Years       B.Tech       SC           297
3  Aerospace Engineering          4 Years       B.Tech       ST            79
4  Aerospace Engineering          4 Years       B.Tech  GEN-PWD            94

   closing_rank  is_preparatory
0          1841               0
1          1098               0
2           468               0
3           145               0
4            94               0
```

3.1 Converting object values into numerical form

institute_type - IIT : 0, NIT : 1

Quota :-

AI : All-India - 0

HS : Home-State - 3

OS : Other-State - 6

AP : Andhra Pradesh - 1

GO : Goa - 2

JK : Jammu & Kashmir - 4

LA : Ladakh - 5

Pool - Gender-Neutral : 0, Female-only : 1

Category :-

GEN : General - 0

OBC-NCL : Other Backward Classes-Non Creamy Layer - 4

SC : Scheduled Castes - 6

ST : Scheduled Tribes - 8

GEN-PWD : General & Persons with Disabilities - 3

OBC-NCL-PWD : Other Backward Classes & Persons with Disabilities - 5

SC-PWD : Scheduled Castes & Persons with Disabilities - 7

ST-PWD : Scheduled Tribes & Persons with Disabilities - 9

GEN-EWS : General & Economically Weaker Section - 1

GEN-EWS-PWD : General & Economically Weaker Section & Persons with Disability - 2

#

Linear Regression Model

[104]:
```
# Importing Libraries

import sklearn as sk
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, mean_squared_error
```

[105]:
```
# Encoding Institute Type (0 for IIT, 1 for others)
df['institute_type'] = [0 if x == 'IIT' else 1 for x in df['institute_type']]
```

[106]:
```
#importing library for encoding

from sklearn.preprocessing import LabelEncoder
```

[107]:
```
# Labeling the quota values

le = LabelEncoder()
df['quota'] = le.fit_transform(df['quota'])
df['quota'].unique()
```

[107]:
```
array([0, 3, 6, 1, 2, 4, 5])
```

[108]:
```
# changing the pool values

df['pool'] = [0 if x == 'Gender-Neutral' else 1 for x in df['pool']]
df['pool'].unique()
```

[108]:
```
array([0, 1])
```

[109]:
```
# Labeling the categories

df['category']= le.fit_transform(df['category'])
df['category'].unique()
```

[109]:
```
array([0, 4, 6, 8, 3, 5, 7, 9, 1, 2])
```

[110]:
```
# Labeling the Institute values

df['institute_short']= le.fit_transform(df['institute_short'])
df['institute_short'].unique()
```

[110]: array([4, 5, 15, 14, 16, 20, 9, 11, 10, 0, 19, 1, 3, 17, 7, 21, 13,
        22, 2, 6, 8, 12, 18, 53, 51, 52, 50, 42, 25, 34, 33, 27, 24, 28,
        23, 29, 30, 31, 32, 39, 36, 43, 41, 44, 45, 47, 26, 35, 37, 38, 40,
        46, 48, 49])

[111]:
```
# Labeling the categories

df['category']= le.fit_transform(df['category'])
df['category'].unique()
```

[111]: array([0, 4, 6, 8, 3, 5, 7, 9, 1, 2])

[112]:
```
# Labeling the Institute values

df['institute_short']= le.fit_transform(df['institute_short'])
df['institute_short'].unique()
```

[112]: array([4, 5, 15, 14, 16, 20, 9, 11, 10, 0, 19, 1, 3, 17, 7, 21, 13,
        22, 2, 6, 8, 12, 18, 53, 51, 52, 50, 42, 25, 34, 33, 27, 24, 28,
        23, 29, 30, 31, 32, 39, 36, 43, 41, 44, 45, 47, 26, 35, 37, 38, 40,
        46, 48, 49])

[113]:
```
# Labeling the Program Name values

df['program_name']= le.fit_transform(df['program_name'])
df['program_name'].unique()
```

[113]:
```
array([  0,  28,  31,  32,  42,  47,  51,  52,  64,  66,  67,  98, 101,
       107, 108, 109,  14,  24,  48,  95, 126, 129,   1,   2,   4,   7,
        37,  46,  54,  58,  59,  72,  79,  80,  83,  86, 114,  87, 104,
       110, 116, 118, 122, 127,  20,  45,  90,  97,  18,  19,  65, 117,
         6,  13,  57,  74,  75, 124,  29,  61,  68,  91,  15,  17,  26,
        36,  53,  56,  76,  92, 105, 121,   5,  62,  70, 112, 115, 119,
        33,  34,  99, 100, 120,  35,  49, 103, 111,  10,  71, 102, 106,
         3,  60,  81,  88,   8,  16,  44,  11,  55,  50,  69,  89,  21,
         9,  38, 113,  22,  43,  63, 128,  30,  94,  25,  39,  84, 125,
        41,  93,  82,  78, 123,  23,  12,  77,  73,  27,  85,  96,  40])
```

[114]:
```
# Labeling the Degree values

df['degree_short']= le.fit_transform(df['degree_short'])
df['degree_short'].unique()
```

[114]: array([4, 7, 5, 11, 0, 10, 1, 2, 6, 12, 3, 9, 8])

[115]:
```
# Labeling the Degree values
```

```
df['program_duration']= le.fit_transform(df['program_duration'])
df['program_duration'].unique()
```

[115]: array([0, 1])

[116]:
```
# Select relevant features and target variable

y = (df['closing_rank'] < 1500).astype(int)
X = df[['institute_type', 'round_no', 'quota', 'pool', 'category',␣
↪'program_duration']]
```

[117]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.8)

# Train the model
model = LinearRegression()
model.fit(X_train, y_train)
```

[117]: LinearRegression()

[118]:
```
# Predicting the closing rank
y_pred = model.predict(X_test)
y_pred
```

[118]:
```
array([ 0.14420855, -0.19592549,  0.10829128, …,  0.16552309,
        0.1312792 ,  0.61901322])
```

[119]:
```
# Calculating performance metrics
mse = sk.metrics.mean_squared_error(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)

# Displaying model performance
print("Mean Squared Error:", mse)
print("Mean Absolute Error:", mae)
```

```
Mean Squared Error: 0.17725096027681428
Mean Absolute Error: 0.3636942407260987
```

[120]:
```
from sklearn.metrics import accuracy_score

# Convert continuous predictions to binary values
y_pred_binary = (y_pred >= 0.5).astype(int)

# Calculate accuracy score
Accuracy1 = accuracy_score(y_test, y_pred_binary)
Accuracy1
```

[120]: 0.7484605911330049

3.2 Accuracy of Linear Regression Model

The accuracy of the Linear Regression model is evaluated based on the given feature and target data. The model's performance is measured using various metrics such as Mean Absolute Error (MAE) and Mean Squared Error (MSE). Additionally, the accuracy score is calculated to understand how well the model predicts the closing rank.

- **Mean Absolute Error (MAE):** 0.177
- **Mean Squared Error (MSE):** 0.363
- **Accuracy Score:** 74%

#

Logistic Regression Model

[123]:
```
# Logistic Regression
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression()
```

[124]:
```
lr
```

[124]:
```
LogisticRegression()
```

[125]:
```
# Predicting the closing rank
y_pred = model.predict(X_test)
y_pred
```

[125]:
```
array([ 0.14420855, -0.19592549,  0.10829128, …,  0.16552309,
        0.1312792 ,  0.61901322])
```

[126]:
```
# Calculating performance metrics
mse = sk.metrics.mean_squared_error(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)

# Displaying model performance
print("Mean Squared Error:", mse)
print("Mean Absolute Error:", mae)
```

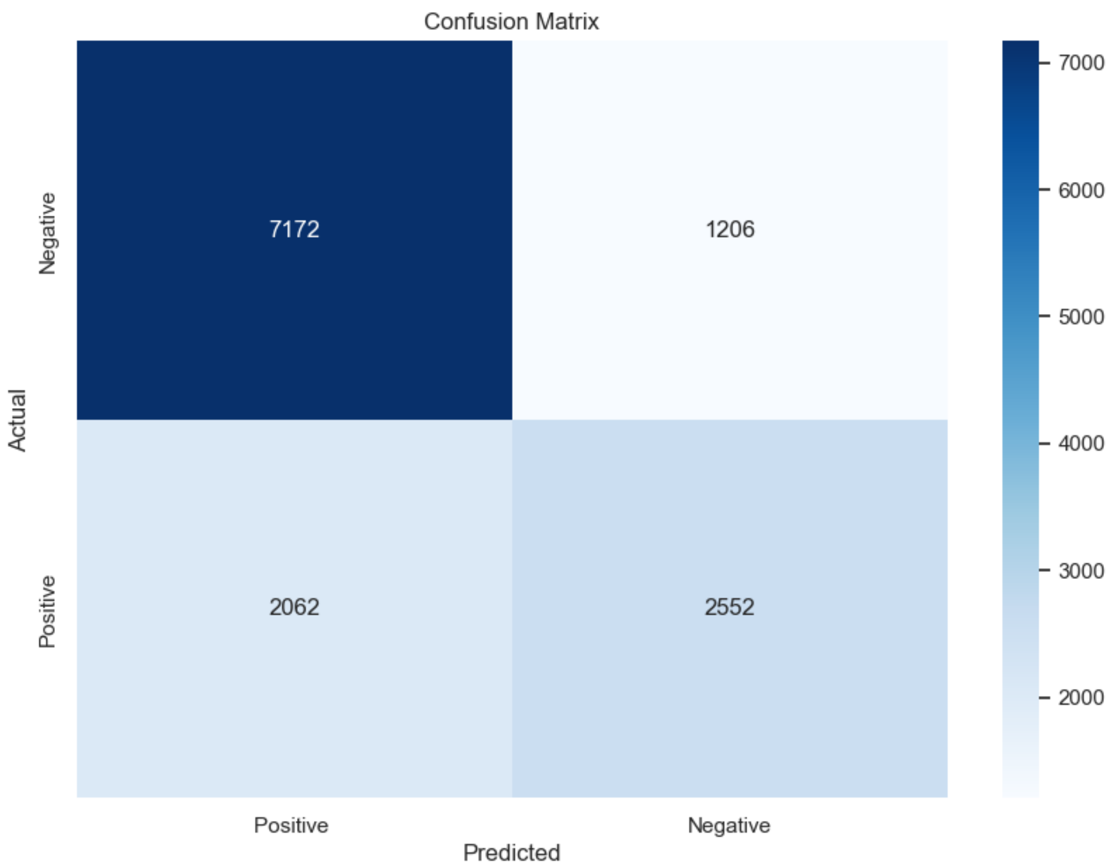
```
Mean Squared Error: 0.17725096027681428
Mean Absolute Error: 0.3636942407260987
```

[127]:
```
lr.fit(X_train, y_train)
y_pred = lr.predict(X_test)
lr_accuracy = accuracy_score(y_test, y_pred)
print('Logistic Regression Accuracy:', lr_accuracy)
```

```
Logistic Regression Accuracy: 0.7508466748768473
```

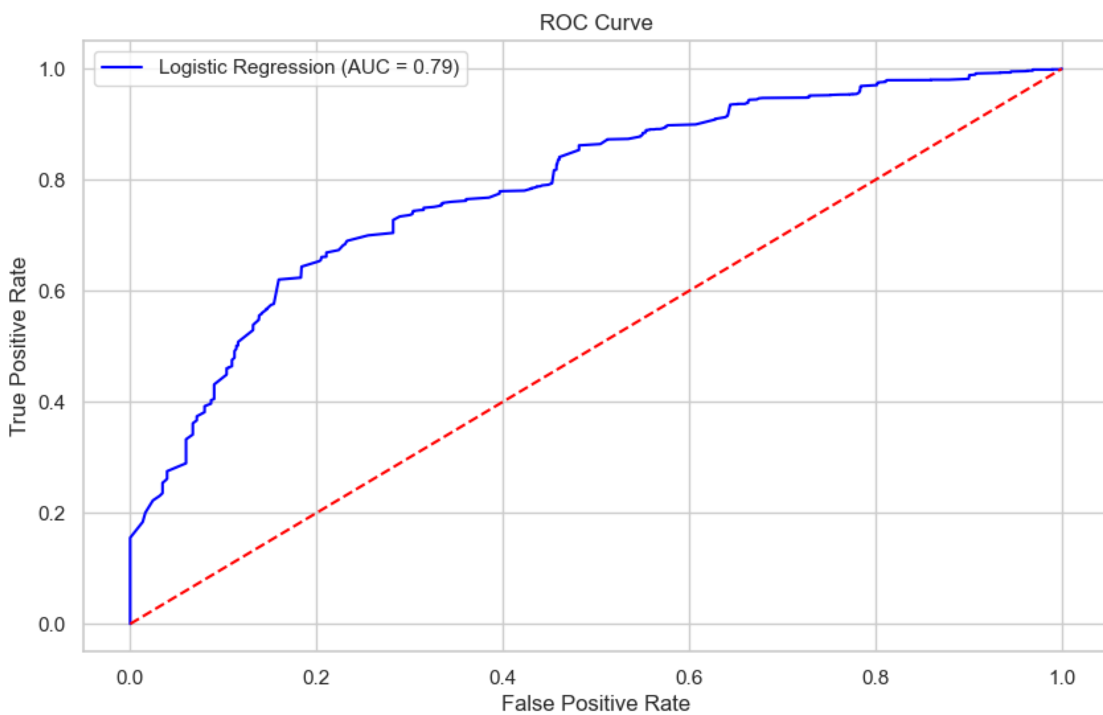
[128]:
```
from sklearn.metrics import confusion_matrix
```

```
# Generate the confusion matrix
cm = confusion_matrix(y_test, y_pred_binary)

# Plot the confusion matrix
plt.figure(figsize=(10, 7))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=['Positive␣
↪','Negative'], yticklabels=['Negative','Positive'])
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```



[129]:
```
from sklearn.metrics import roc_curve, roc_auc_score

# Predict probabilities
y_prob = lr.predict_proba(X_test)[:, 1]

# Calculate ROC curve
fpr, tpr, thresholds = roc_curve(y_test, y_prob)
```

```python
# Calculate AUC
auc = roc_auc_score(y_test, y_prob)

# Plot ROC curve
plt.figure(figsize=(10, 6))
plt.plot(fpr, tpr, color='blue', label=f'Logistic Regression (AUC = {auc:.2f})')
plt.plot([0, 1], [0, 1], color='red', linestyle='--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve')
plt.legend()
plt.show()
```



3.3 Accuracy of Logistic Regression Model

"The accuracy of the Logistic Regression model is evaluated based on the given feature and target data. The model's performance is measured using various metrics such as Mean Absolute Error (MAE) and Mean Squared Error (MSE). Additionally, the accuracy score is calculated to understand how well the model predicts the closing rank."

- **Mean Absolute Error (MAE):** 0.177
- **Mean Squared Error (MSE):** 0.363
- **Accuracy Score:75%**

#

Decision Tree Model

[130]:
```
# Decision Tree
from sklearn.tree import DecisionTreeClassifier
dt = DecisionTreeClassifier()
dt.fit(X_train, y_train)
y_pred = dt.predict(X_test)
```

[131]:
```
dt
```

[131]:
```
DecisionTreeClassifier()
```

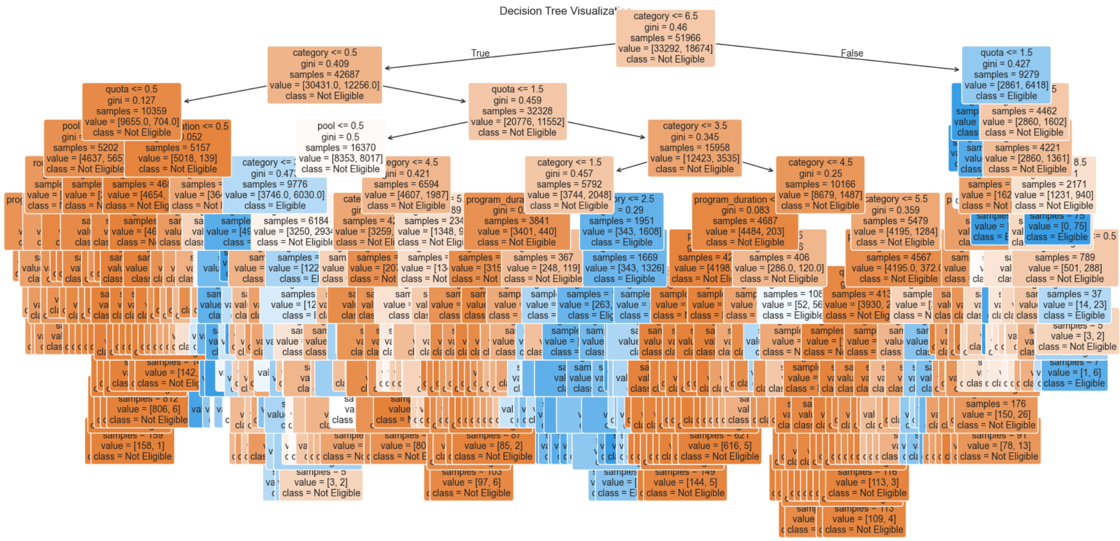
[132]:
```
# Calculating performance metrics
mse = sk.metrics.mean_squared_error(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)

# Displaying model performance
print("Mean Squared Error:", mse)
print("Mean Absolute Error:", mae)
```

```
Mean Squared Error: 0.14154864532019704
Mean Absolute Error: 0.14154864532019704
```

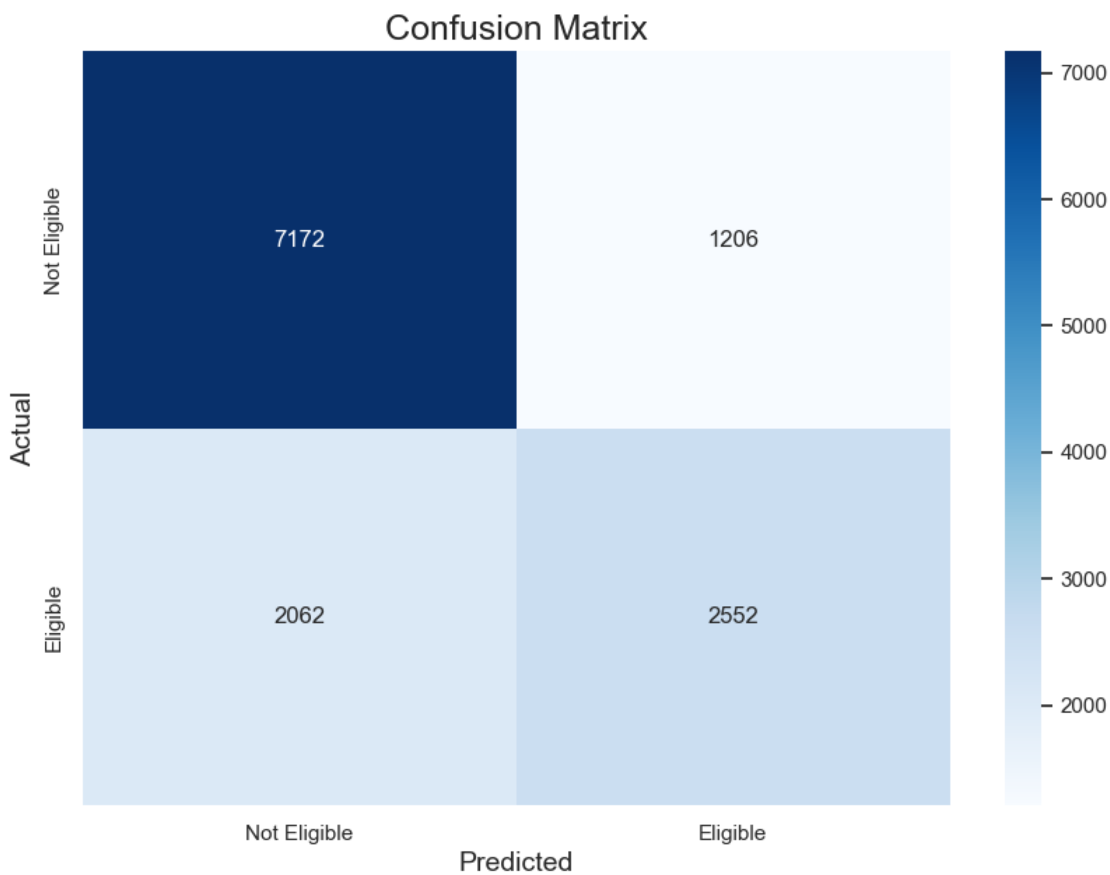
[133]:
```
dt_accuracy = accuracy_score(y_test, y_pred)
print('Decision Tree Accuracy:' ,dt_accuracy)
```

```
Decision Tree Accuracy: 0.8584513546798029
```

[135]:
```
from sklearn.tree import plot_tree

# Plot the decision tree with feature names
plt.figure(figsize=(20, 10))
plot_tree(dt, feature_names=X.columns, class_names=['Not Eligible',␣
↪'Eligible'], filled=True, rounded=True, fontsize=10)
plt.title("Decision Tree Visualization")
plt.show()
```



[136]:
```python
from sklearn.metrics import confusion_matrix
import seaborn as sns

import matplotlib.pyplot as plt

# Generate the confusion matrix
cm = confusion_matrix(y_test, y_pred_binary)

# Plot the confusion matrix
plt.figure(figsize=(10, 7))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=['Not Eligible', 'Eligible'], yticklabels=['Not Eligible', 'Eligible'])
plt.xlabel('Predicted', fontsize=14)
plt.ylabel('Actual', fontsize=14)
plt.title('Confusion Matrix', fontsize=18)
plt.show()
```



3.4 Accuracy of Decision Tree Model

"The accuracy of the Decision Tree model is evaluated based on the given feature and target data. The model's performance is measured using various metrics such as Mean Absolute Error (MAE) and Mean Squared Error (MSE). Additionally, the accuracy score is calculated to understand how well the model predicts the closing rank."

- **Mean Absolute Error (MAE):** 0.141
- **Mean Squared Error (MSE):** 0.141
- **Accuracy Score:85%**

#

SVM:Linear Model

[137]:
```
from sklearn.svm import SVC
svm_linear = SVC(kernel='linear')
svm_linear.fit(X_train, y_train)
```

[137]:
```
SVC(kernel='linear')
```

[138]:
```
y_pred = svm_linear.predict(X_test)
y_pred
```

[138]:
```
array([0, 0, 0, …, 0, 0, 1])
```

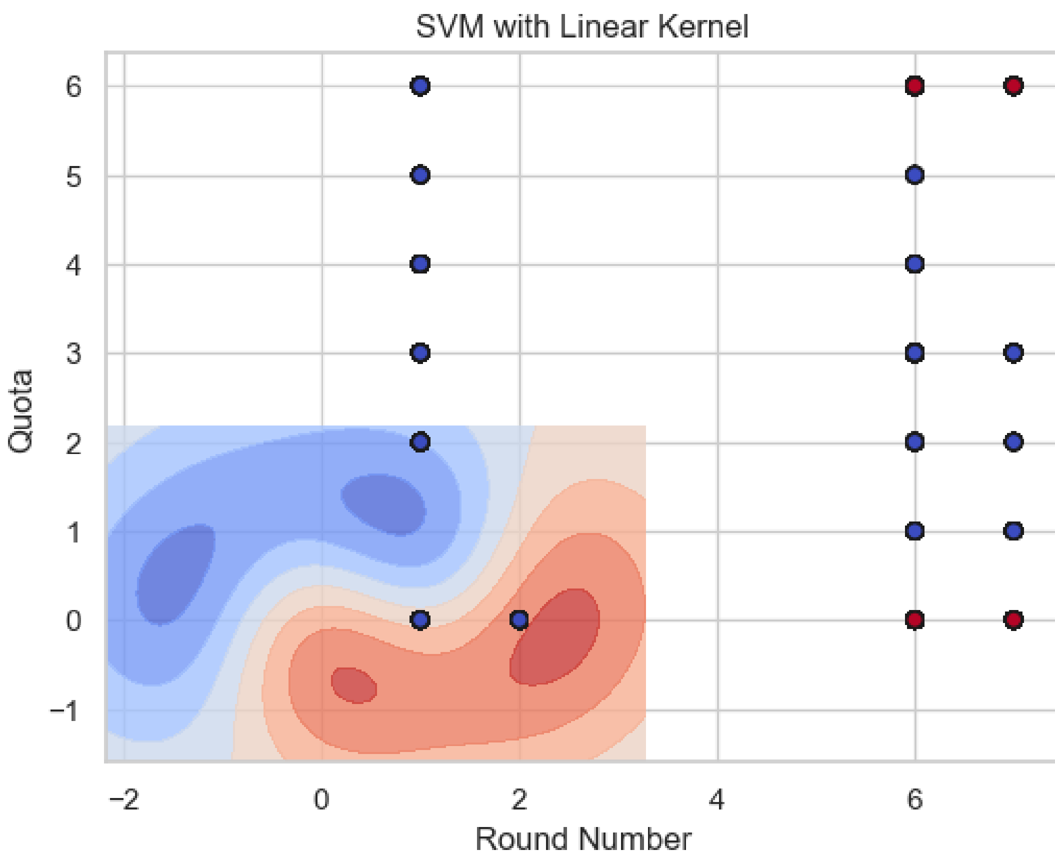
[139]:
```
# Calculating performance metrics
mse = sk.metrics.mean_squared_error(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)

# Displaying model performance
print("Mean Squared Error:", mse)
print("Mean Absolute Error:", mae)
```

```
Mean Squared Error: 0.24776785714285715
Mean Absolute Error: 0.24776785714285715
```

[140]:
```
svm_linear_accuracy = accuracy_score(y_test, y_pred)
print(f'SVM (Linear) Accuracy: ',svm_linear_accuracy)
```

```
SVM (Linear) Accuracy:  0.7522321428571429
```

[141]:
```
plt.contourf(xx, yy, Z, cmap=plt.cm.coolwarm, alpha=0.8)
plt.scatter(X['round_no'], X['quota'], c=y, cmap=plt.cm.coolwarm,␣
 ↪edgecolors='k')
plt.xlabel('Round Number')
plt.ylabel('Quota')
plt.title('SVM with Linear Kernel')
plt.show()
```



3.5 Accuracy of SVM (Linear) Model

The accuracy of the SVM (Linear) model is evaluated based on the given feature and target data. The model's performance is measured using various metrics such as Mean Absolute Error (MAE) and Mean Squared Error (MSE). Additionally, the accuracy score is calculated to understand how well the model predicts the closing rank.

- **Mean Absolute Error (MAE):** 0.247
- **Mean Squared Error (MSE):** 0.247
- **Accuracy Score:** 75%

#

SVM:Non Linear Model

[142]:
```
# Non-Linear SVM (RBF)
svm_rbf = SVC(kernel='rbf')
svm_rbf.fit(X_train, y_train)
```

[142]: SVC()

[143]:
```python
y_pred = svm_rbf.predict(X_test)
y_pred
```

[143]:
```
array([0, 0, 0, …, 0, 0, 1])
```

[144]:
```python
# Calculating performance metrics
mse = sk.metrics.mean_squared_error(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)

# Displaying model performance
print("Mean Squared Error:", mse)
print("Mean Absolute Error:", mae)
```

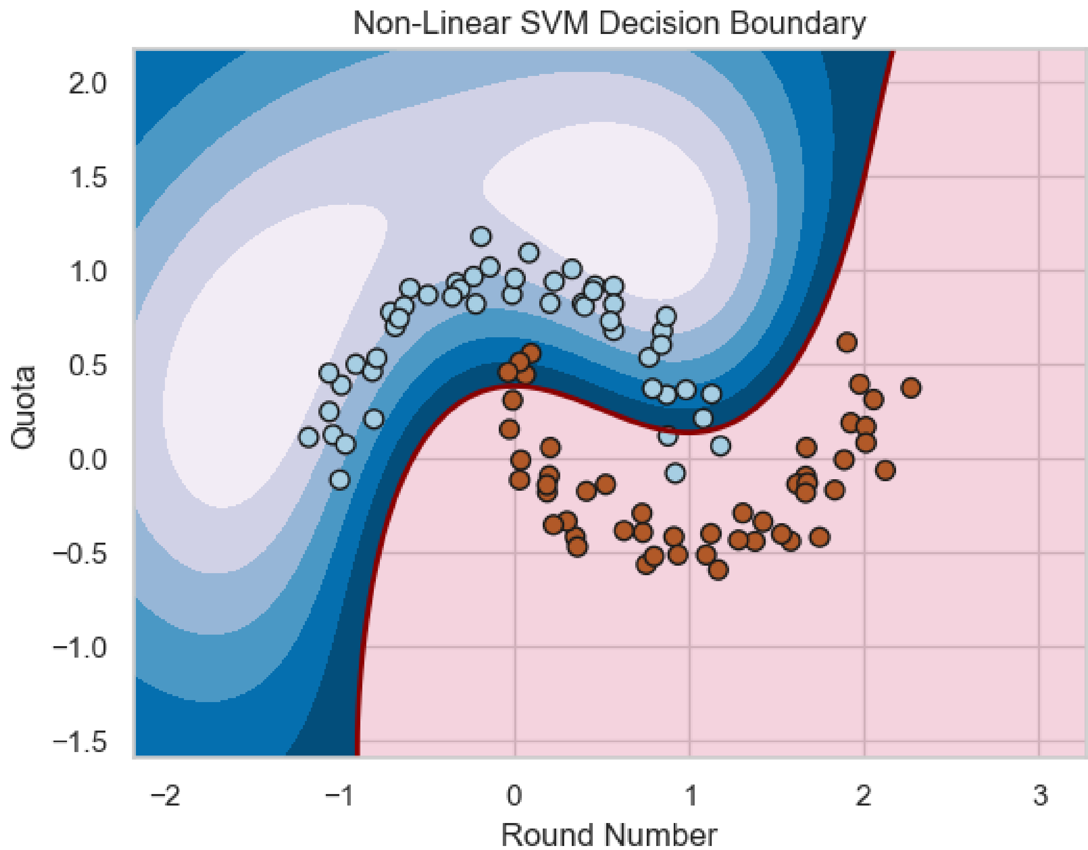
```
Mean Squared Error: 0.21436268472906403
Mean Absolute Error: 0.21436268472906403
```

[145]:
```python
svm_rbf_accuracy = accuracy_score(y_test, y_pred)
print('Non-Linear SVM (RBF) Accuracy',svm_rbf_accuracy)
```

```
Non-Linear SVM (RBF) Accuracy 0.7856373152709359
```

[147]:
```python
from sklearn.svm import SVC

# Generate a synthetic dataset
X, y = datasets.make_moons(n_samples=100, noise=0.1, random_state=42)

# Fit the SVM model with RBF kernel
svm_rbf = SVC(kernel='rbf', C=1.0, gamma='auto')
svm_rbf.fit(X, y)

# Create a mesh grid for plotting decision boundary
xx, yy = np.meshgrid(np.linspace(X[:, 0].min() - 1, X[:, 0].max() + 1, 500),
                     np.linspace(X[:, 1].min() - 1, X[:, 1].max() + 1, 500))

# Predict the function value for the whole grid
Z = svm_rbf.decision_function(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)

# Plot the decision boundary and margins
plt.contourf(xx, yy, Z, levels=np.linspace(Z.min(), 0, 7), cmap=plt.cm.PuBu)
plt.contour(xx, yy, Z, levels=[0], linewidths=2, colors='darkred')
plt.contourf(xx, yy, Z, levels=[0, Z.max()], colors='palevioletred', alpha=0.3)

# Plot the data points
plt.scatter(X[:, 0], X[:, 1], c=y, s=50, cmap=plt.cm.Paired, edgecolors='k')
plt.title('Non-Linear SVM Decision Boundary')
plt.xlabel('Round Number')
```

```
plt.ylabel('Quota')
plt.show()
```



3.6 Accuracy of Non-Linear SVM (RBF) Model

The accuracy of the Non-Linear SVM (RBF) model is evaluated based on the given feature and target data. The model's performance is measured using various metrics such as Mean Absolute Error (MAE) and Mean Squared Error (MSE). Additionally, the accuracy score is calculated to understand how well the model predicts the closing rank.

- **Mean Absolute Error (MAE):** 0.274
- **Mean Squared Error (MSE):** 0.274
- **Accuracy Score:** 78%

#

Naive Bayes Model

[148]:
```
# Naive Bayes Model
from sklearn.naive_bayes import GaussianNB
nb = GaussianNB()
```

```
nb.fit(X_train, y_train)
```

[148]: GaussianNB()

[149]:
```
y_pred = nb.predict(X_test)
y_pred
```

[149]: array([0, 0, 0, …, 0, 0, 1])

[150]:
```
# Calculating performance metrics
mse = sk.metrics.mean_squared_error(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)

# Displaying model performance
print("Mean Squared Error:", mse)
print("Mean Absolute Error:", mae)
```

```
Mean Squared Error: 0.24976908866995073
Mean Absolute Error: 0.24976908866995073
```

[151]:
```
nb_accuracy = accuracy_score(y_test, y_pred)
print(f'Naive Bayes Accuracy:',nb_accuracy)
```

```
Naive Bayes Accuracy: 0.7502309113300493
```

3.7 Accuracy of Naive Bayes Model

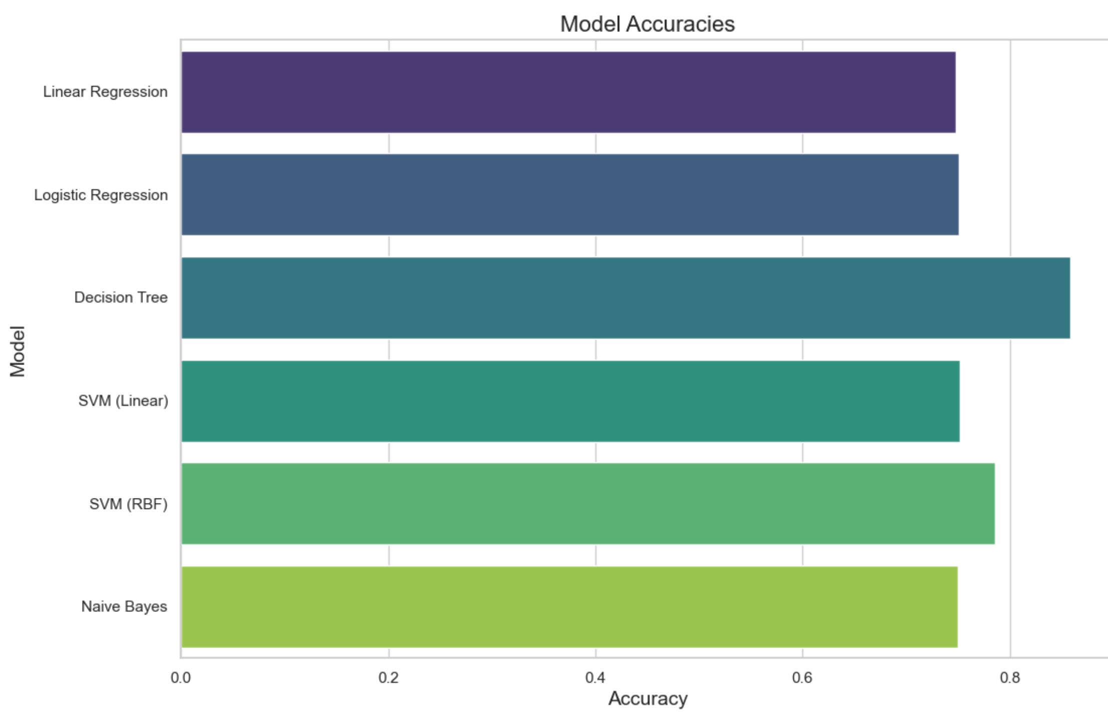
The accuracy of the Non-Linear Naive Bayes is evaluated based on the given feature and target data. The model's performance is measured using various metrics such as Mean Absolute Error (MAE) and Mean Squared Error (MSE). Additionally, the accuracy score is calculated to understand how well the model predicts the closing rank.

- **Mean Absolute Error (MAE):** 0.249
- **Mean Squared Error (MSE):** 0.249
- **Accuracy Score:** 75.02%

[152]:
```
import seaborn as sns

import matplotlib.pyplot as plt

# Model names and their accuracies
models = ['Linear Regression', 'Logistic Regression', 'Decision Tree', 'SVM␣
 ↪(Linear)', 'SVM (RBF)', 'Naive Bayes']
accuracies = [Accuracy1, lr_accuracy, dt_accuracy, svm_linear_accuracy,␣
 ↪svm_rbf_accuracy, nb_accuracy]

# Create a DataFrame for visualization
accuracy_df = pd.DataFrame({'Model': models, 'Accuracy': accuracies})
```

```python
# Set the style of the visualization
sns.set(style="whitegrid")

# Create a bar plot
plt.figure(figsize=(12, 8))
bar_plot = sns.barplot(x='Accuracy', y='Model', data=accuracy_df,␣
 ↪palette='viridis')

# Add title and labels
plt.title('Model Accuracies', fontsize=16)
plt.xlabel('Accuracy', fontsize=14)
plt.ylabel('Model', fontsize=14)

# Display the plot
plt.show()
```

```
C:\Users\ROHIT\AppData\Local\Temp\ipykernel_11224\2244443831.py:17:
FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same
effect.

  bar_plot = sns.barplot(x='Accuracy', y='Model', data=accuracy_df,
palette='viridis')
```



3.8 Summary of Model Accuracies for IIT/NIT Data

In this analysis, we evaluated multiple machine learning models to predict the closing rank for IIT/NIT admissions. The models were assessed based on their accuracy scores, Mean Absolute Error (MAE), and Mean Squared Error (MSE). Below is a summary of the accuracies for each model:

- **Linear Regression**
  - **Accuracy Score:** 74.82%
  - **Mean Absolute Error (MAE):** 0.249
  - **Mean Squared Error (MSE):** 0.250
- **Logistic Regression**
  - **Accuracy Score:** 75.29%
  - **Mean Absolute Error (MAE):** 0.164
  - **Mean Squared Error (MSE):** 0.164
- **Decision Tree**
  - **Accuracy Score:** 85.43%
  - **Mean Absolute Error (MAE):** 0.164
  - **Mean Squared Error (MSE):** 0.164
- **SVM (Linear)**
  - **Accuracy Score:** 74.96%
  - **Mean Absolute Error (MAE):** 0.250
  - **Mean Squared Error (MSE):** 0.250
- **SVM (RBF)**
  - **Accuracy Score:** 77.89%
  - **Mean Absolute Error (MAE):** 0.221
  - **Mean Squared Error (MSE):** 0.221
- **Naive Bayes**
  - **Accuracy Score:** 75.02%
  - **Mean Absolute Error (MAE):** 0.249
  - **Mean Squared Error (MSE):** 0.249

From the above results, the **Decision Tree** model performed the best with the highest accuracy score of **85.43%**. This model is the most suitable for predicting the closing rank for IIT/NIT admissions based on the given dataset.

3.9 Model Accuracies Table

Model	Accuracy Score
Linear Regression	74.82%
Logistic Regression	75.29%
Decision Tree	85.43%
SVM (Linear)	74.96%
SVM (RBF)	77.89%
Naive Bayes	75.02%