Dataset : Mumbai university result

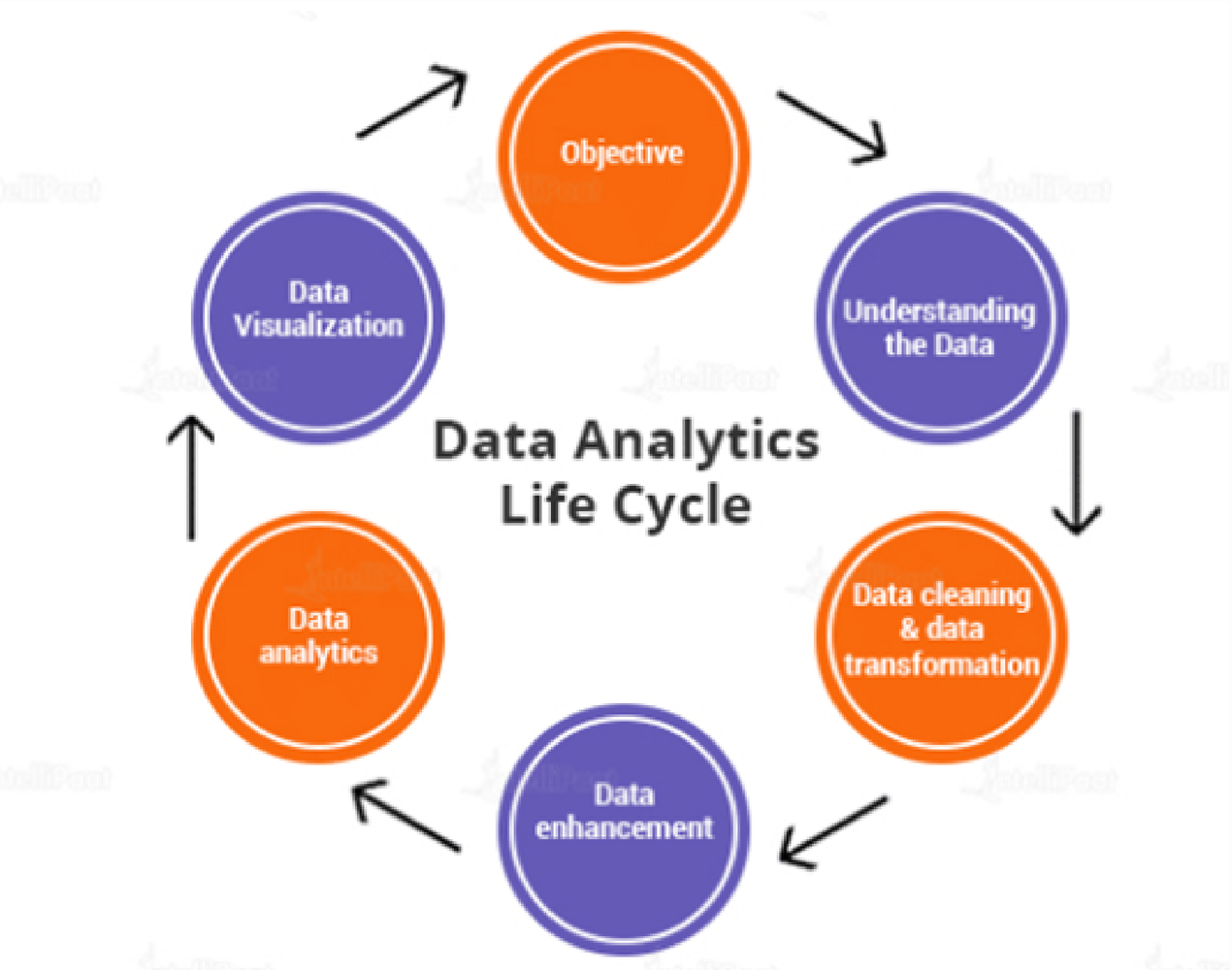
Guided by : Shubhangi Kale

SAMRUDDHI BHUJBAL(20220107O011)
VIDYA BINGI (20220104O019)
VAIBHAV DINGALWAR(20220107O017)

Introduction

- Python is one of the most popular programming languages in the world. Python ranked first as the most wanted technology among developers.
- The mumbai university student dataset can be easily managed by using python
- The various application in python like data manipulation , data visualization,Predictive techniques like KNN,Kmeans , clustering are easy to undestand

Data Analytics Life Cycle



MOTIVATION

- motivation plays a key role in education student gets motivated by the various kind of challenges and the new projects
- This data help us to undestand the student activity and intrest in studies

DETAILS OF DATASET

Dataset : Mumbai university result

Features:13

Number of records:5212

Data manipulation

- Filtering values on the basis of given condition
- Apply a certain function to create either a new variable or perform related operations
- Using a pivot function to aggregate across the desired column
- Sorting a data
- Merge of 2 column

```
[3] import pandas as pd
    df=pd.read_csv("/content/sample_data/ass4_dataset.csv")
```

```
[4] #Printing the dataset
    print(df)
```

```
    #sum of sgpi of all student
    print(df['sgpi'].sum())
```

```
33290.36
```

```
[ ] #display the count of successful status
    b=df.groupby('status').count()
    print(b)
```

```
            seat_no   prn  centre  total_gradepoints  sgpi  \
status
ABS               1     1       1                  1     1
RR              497   497     476                497   497
Successful     4227  4227    4227               4227  4227
```


✓ 0s completed at 23:49

```python
#the dimension of data frame
#average of sgpi
b=df.shape
print(b)
a=df['sgpi'].sum()
avg=a/5211
print(avg)
```

```
(5211, 13)
6.388478219151795
```

```python
#COUNTING MAX GRADEPOINTS
a=df['total_gradepoints'].max()
print(a)
```

```
220
```

```python
#converting the satus to lower case
print(df['status'].str.lower())
```

```
0       successful
1       successful
```

```
#Missing value in sgpi
c=df['sgpi']
print(c.isnull())
```

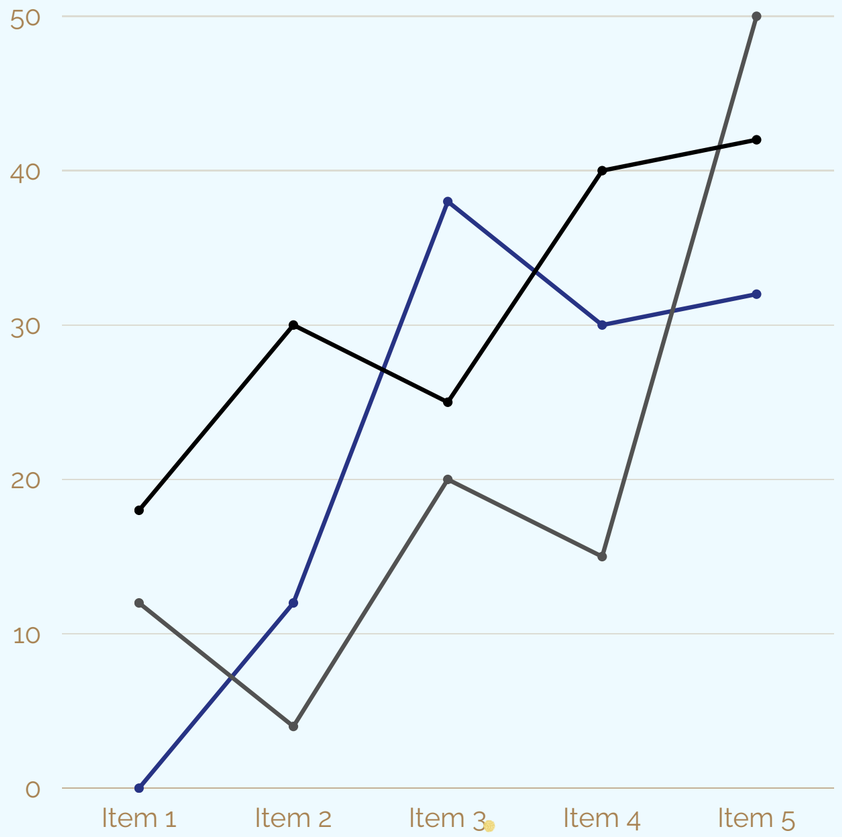
```
0       False
1       False
2       False
3       False
4       False
        ...
5206    False
5207    False
5208    False
5209    False
5210    False
Name: sgpi, Length: 5211, dtype: bool
```

```
#display the max of college id
a=df['clg_id'].max()
print(a)
```

Data visualization

- Data visualization is a branch of data analysis that focuses on visualizing data
- It plots data graphically and is a good way to communicate data inferences.
- Python libraries include various features that allow users to create highly customized, classy, and interactive plots.

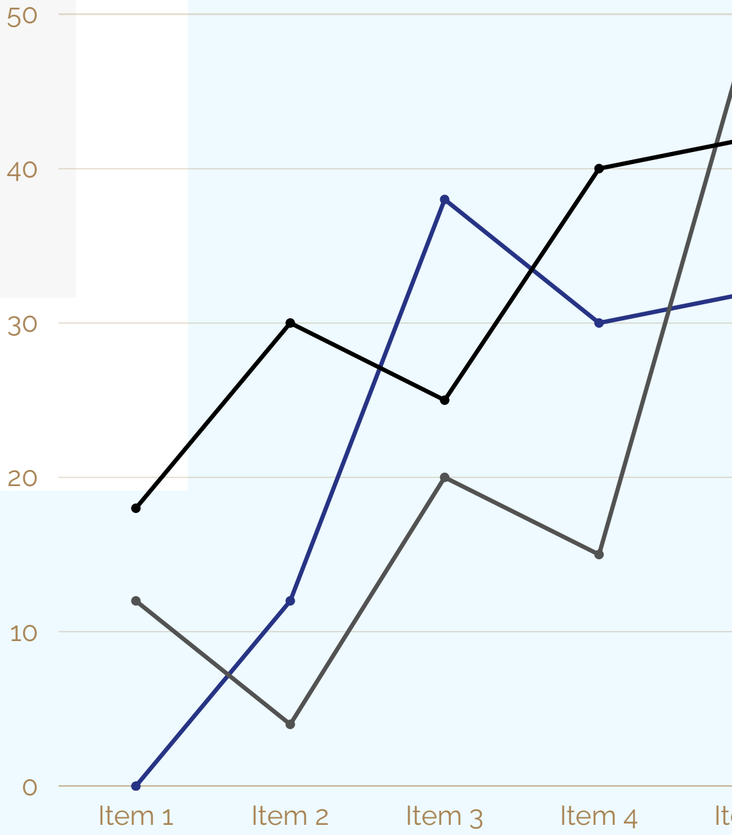
Graph

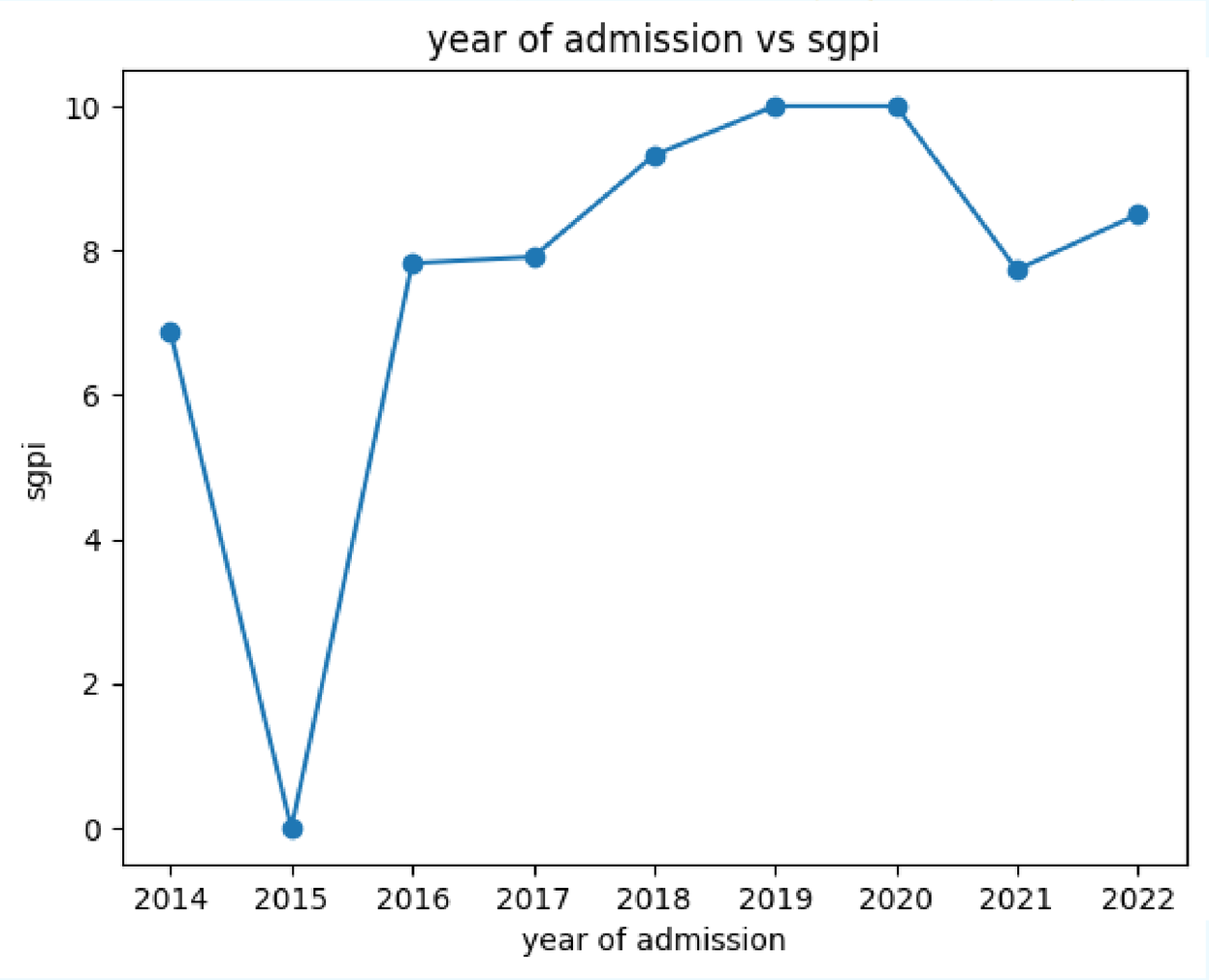


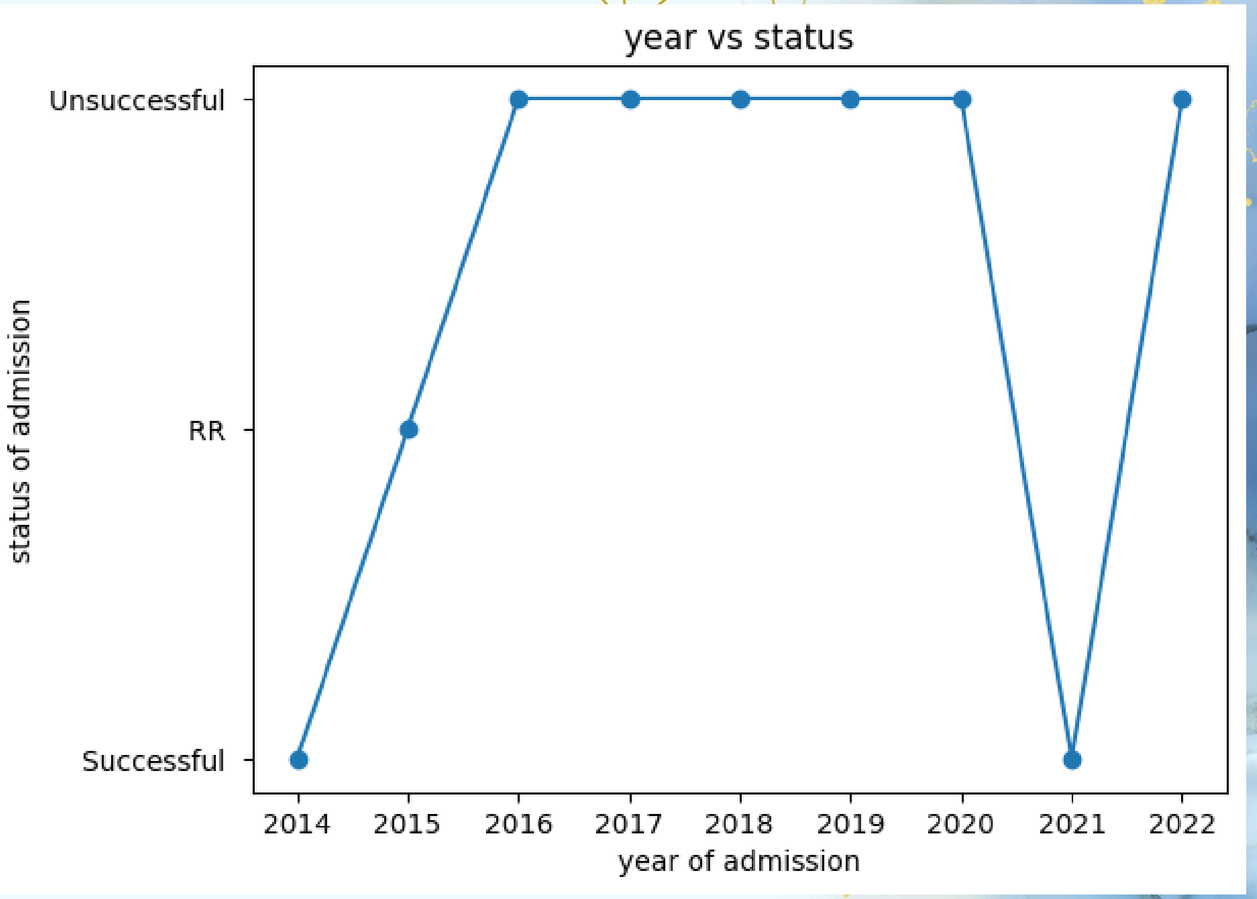
Code for plotting line graph

```
import pandas as pd
import matplotlib.pyplot as plt
df=pd.read csv("/content/sample data/ass4 dataset.csv")
df1 = df.groupby('year_of_admission').max()
plt.plot(df1.index, df1['sgpi'], marker='o')
# Customize the chart
plt.title("year of admission vs sgpi")
plt.xlabel("year of admission")
plt.ylabel("sgpi")
# Display the chart
plt.show()
```

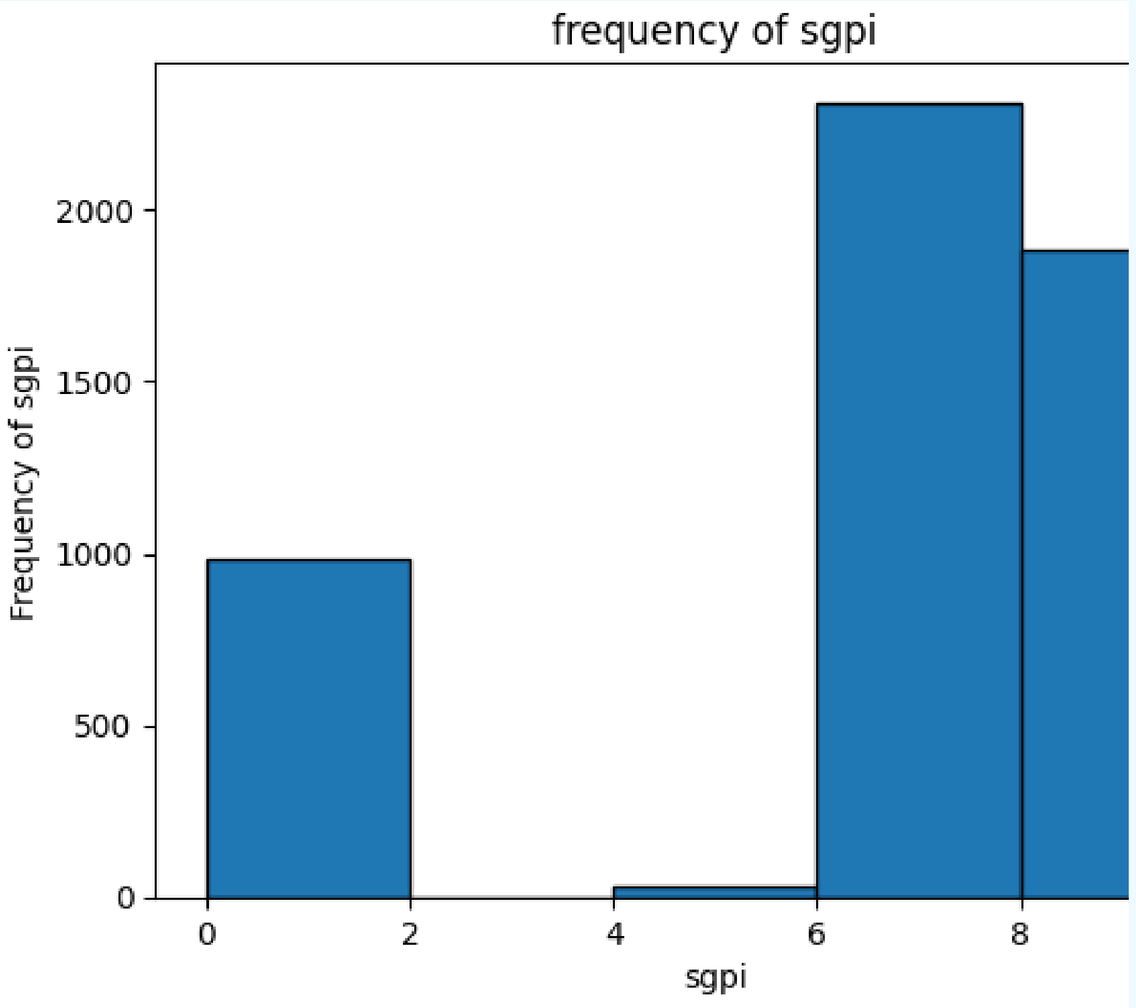
```
df1 = df.groupby('year of admission').max()
plt.plot(df1.index, df1['status'], marker='o')
# Customize the chart
plt.title("year vs status")
plt.xlabel("year of admission")
plt.ylabel("status of admission")
# Display the chart
plt.show()
```

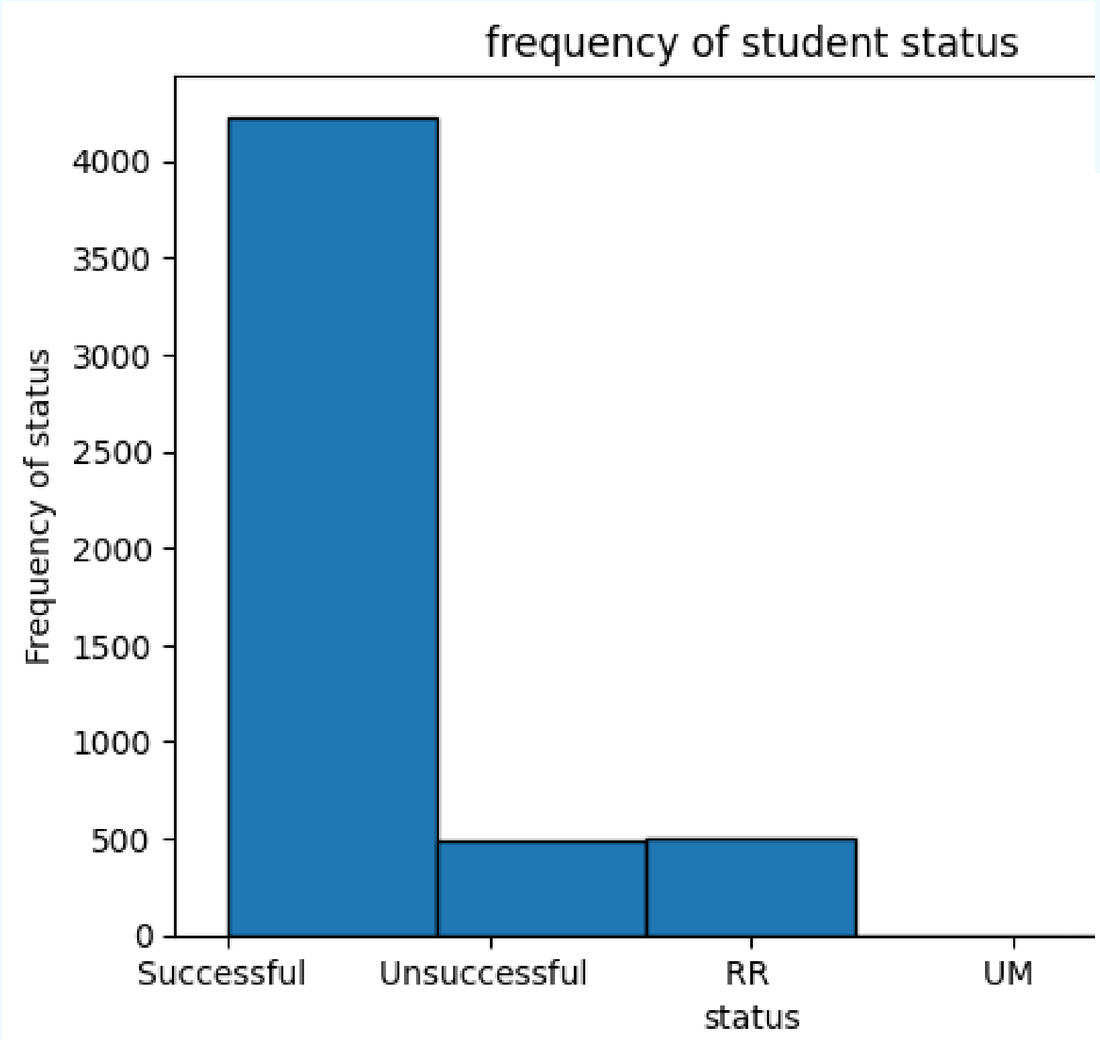



year of admission vs sgpi
sgpi
0
2
4
6
8
10
2014
2015
2016
2017
2018
2019
2020
2021
2022
year of admission


year vs status
status of admission
Unsuccessful
RR
Successful
2014
2015
2016
2017
2018
2019
2020
2021
2022
year of admission

```
b=df["sgpi"]
plt.hist(b, bins=5, edgecolor='black')
# Adding labels and title
plt.xlabel('sgpi')
plt.ylabel('Frequency of sgpi')
plt.title('frequency of sgpi')
# Displaying the histogram
plt.show()
```

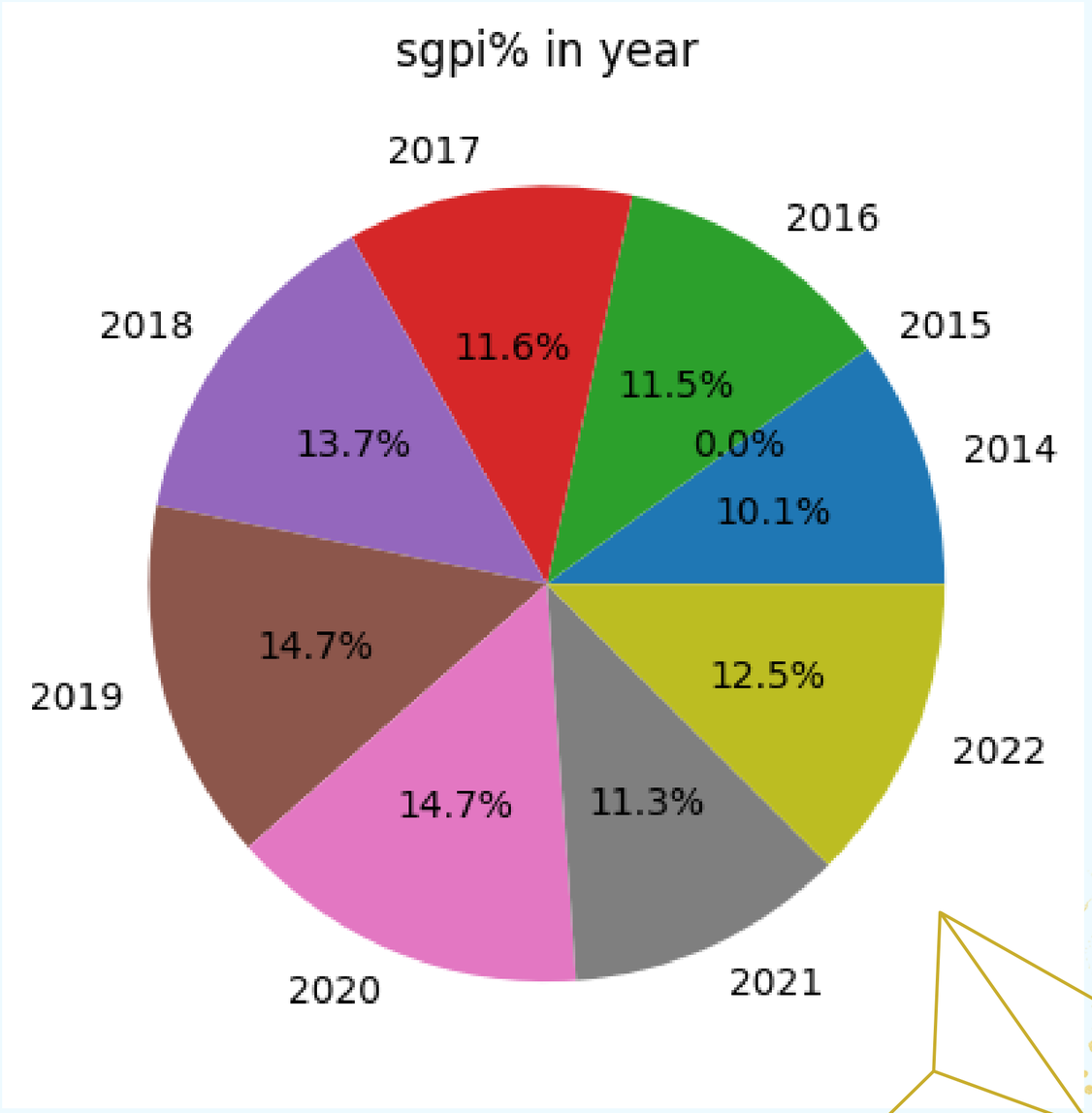




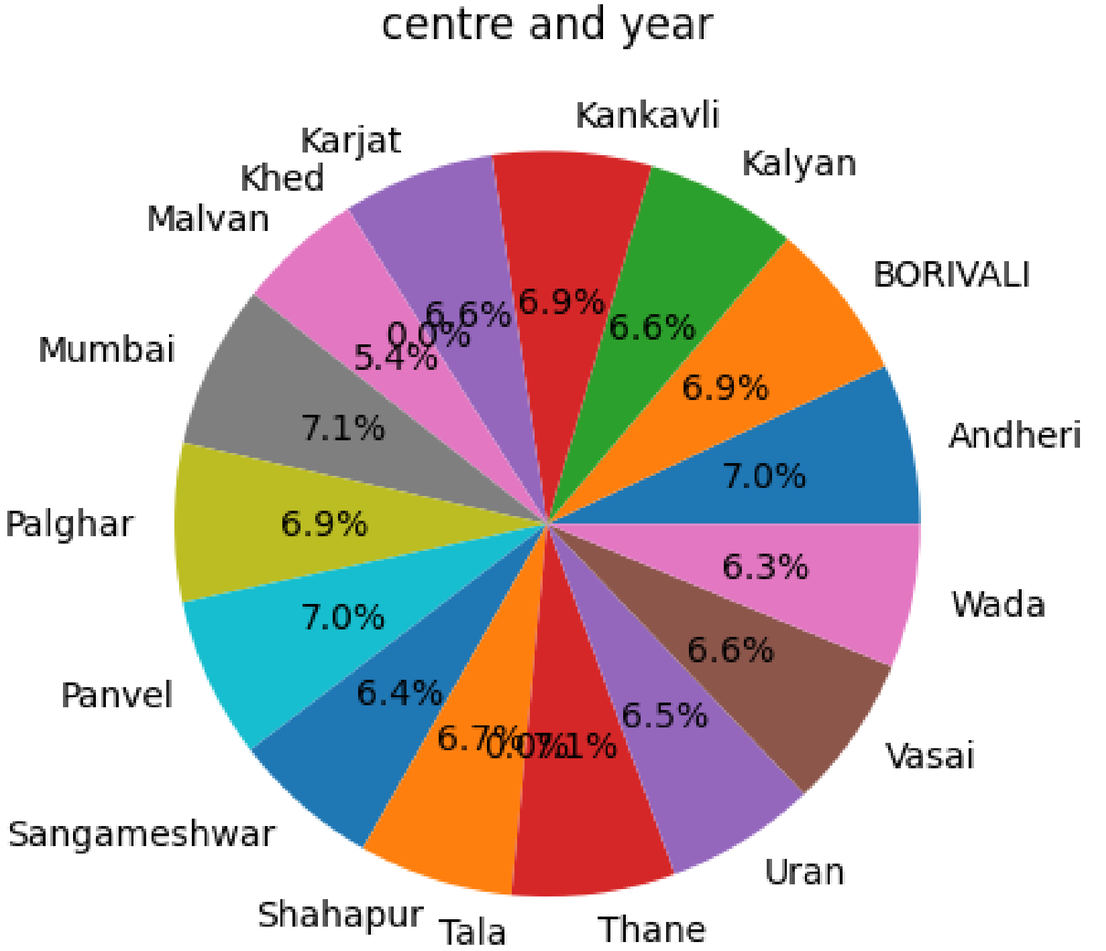
status

```
b=df["gender"]
plt.hist(b, bins=5, edgecolor='black')
# Adding labels and title
plt.xlabel('gender')
plt.ylabel('Frequency of gender')
plt.title('Frequency of gender')
# Displaying the histogram
plt.show()
```

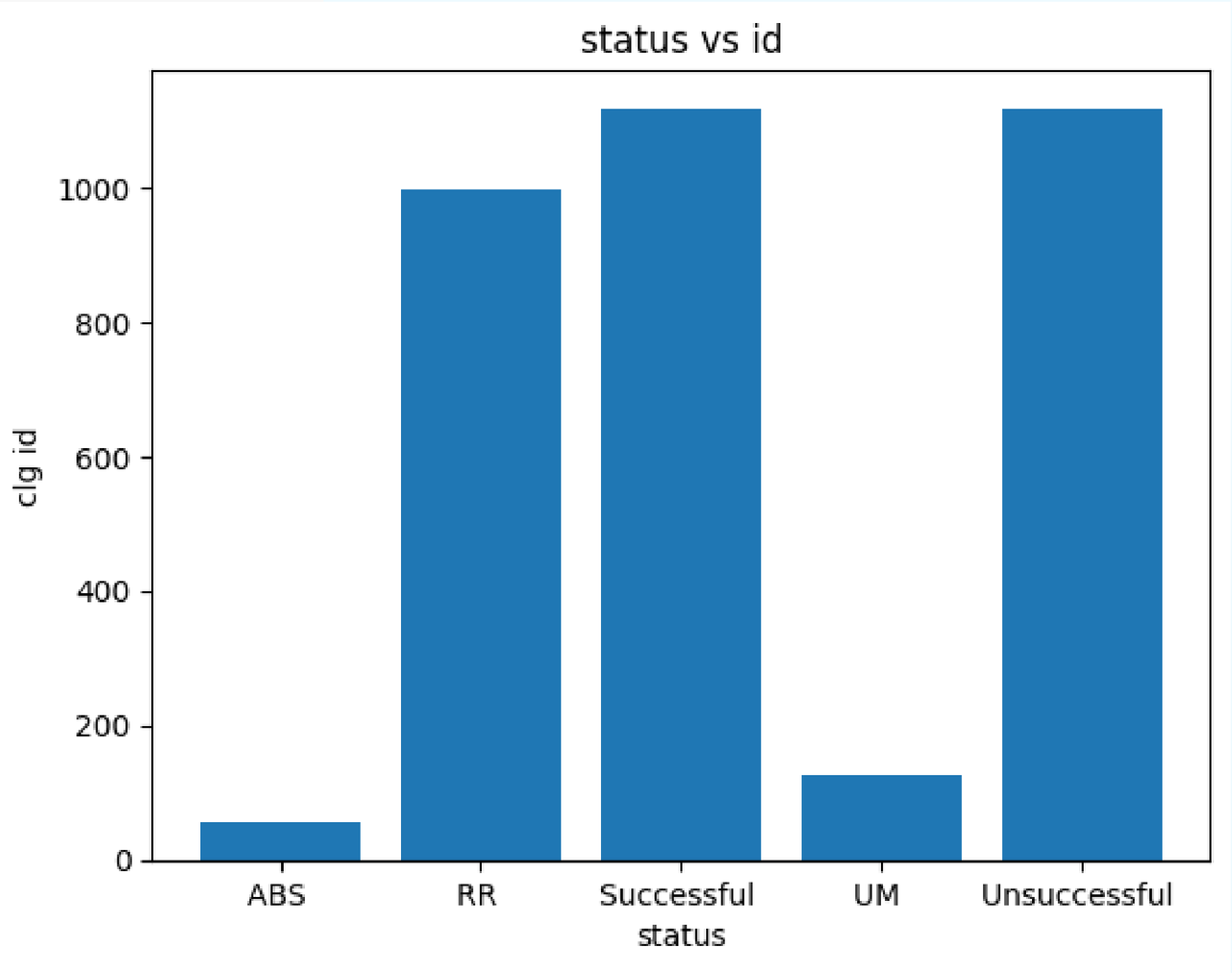
```
import matplotlib.pyplot as plt
# Example data
df1 = df.groupby('year of admission').max()
# Plotting the pie chart
plt.pie(df1['sgpi'],labels=df1.index,
autopct='%1.1f%%')
# Adding a title
plt.title('sgpi% in year')
# Displaying the pie chart
plt.show()
```



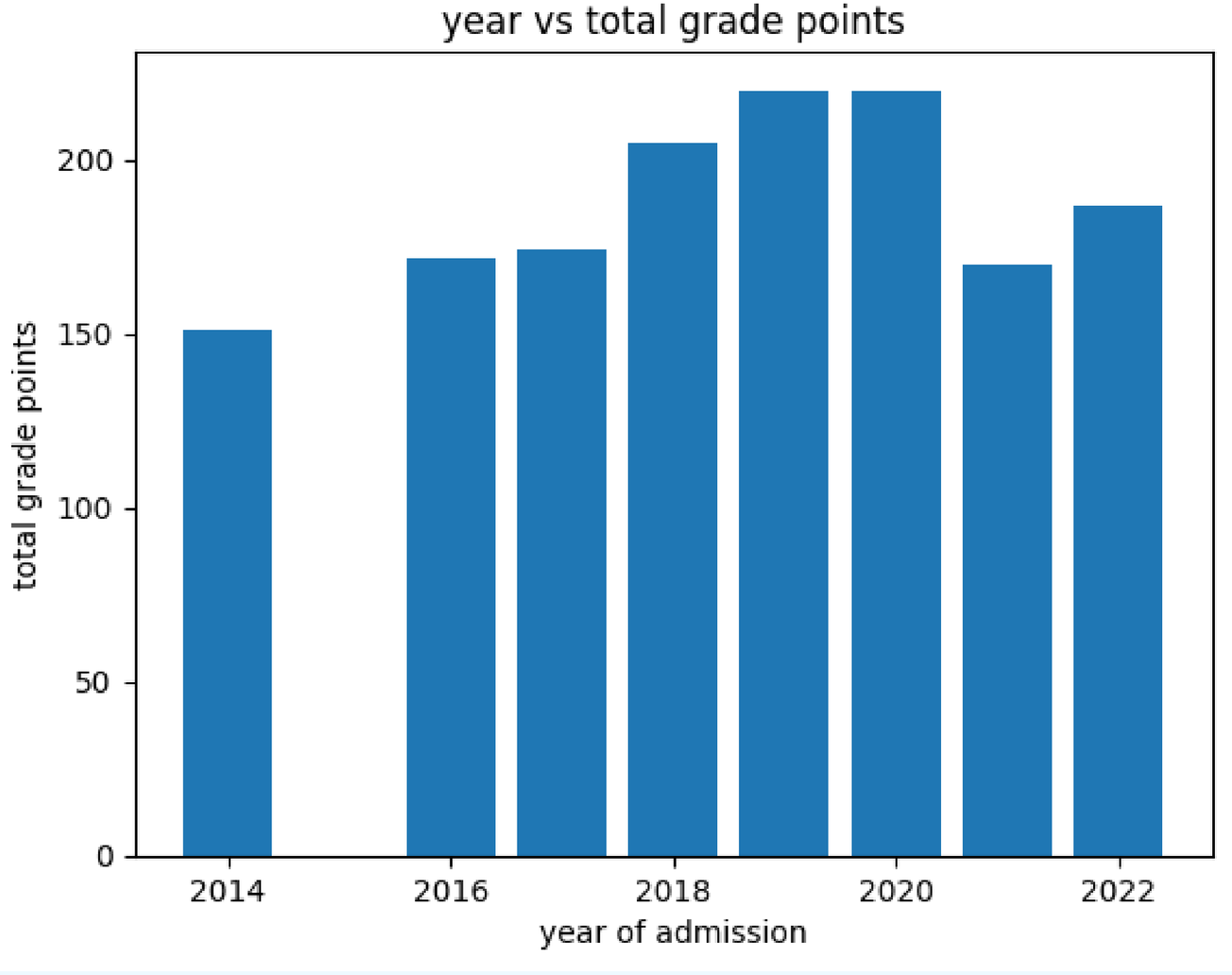
```
# Example data
df1 = df.groupby('centre').max()
# Plotting the pie chart
plt.pie(df1['sgpi'],labels=df1.index,
autopct='%1.1f%%')
# Adding a title
plt.title('centre and year')
# Displaying the pie chart
plt.show()
```



```
df1 = df.groupby('status').max()
plt.bar(df1.index, df1['clg id'])
# Customize the chart
plt.title("status vs id")
plt.xlabel("status")
plt.ylabel("clg id")
# Display the chart
plt.show()
```



```
df1 = df.groupby('year of admission').max()
plt.bar(df1.index, df1['total gradepoints'])
# Customize the chart
plt.title("year vs total grade points")
plt.xlabel("year of admission")
plt.ylabel("total grade points")
# Display the chart
plt.show()
```



Predictive Technique (LR/KNN/KMeans)

- Helps forecast behavior of people and markets
- Answers the question “What could happen?”

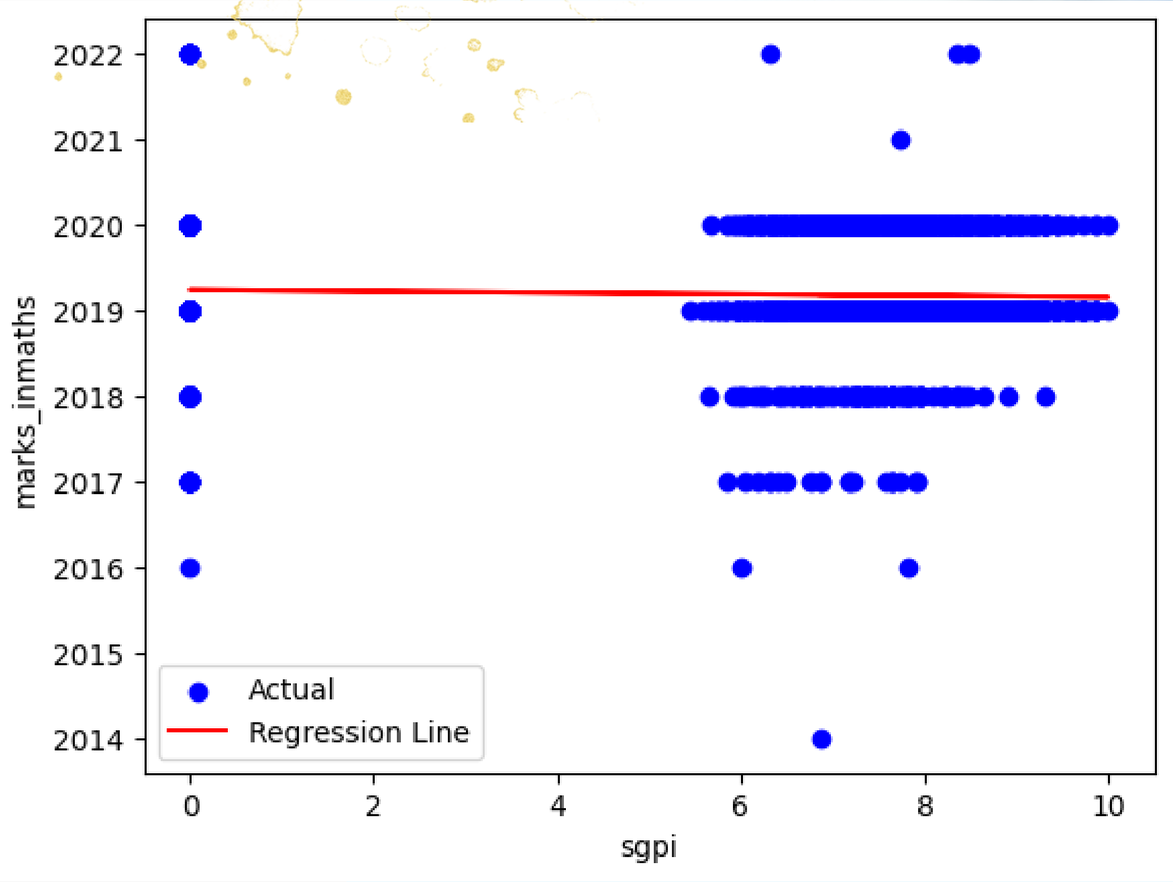
- **Some mistake predictive analysis to have exclusive relevance to predicting future events.**

- **Several of the models that can be used for predictive analysis are:**

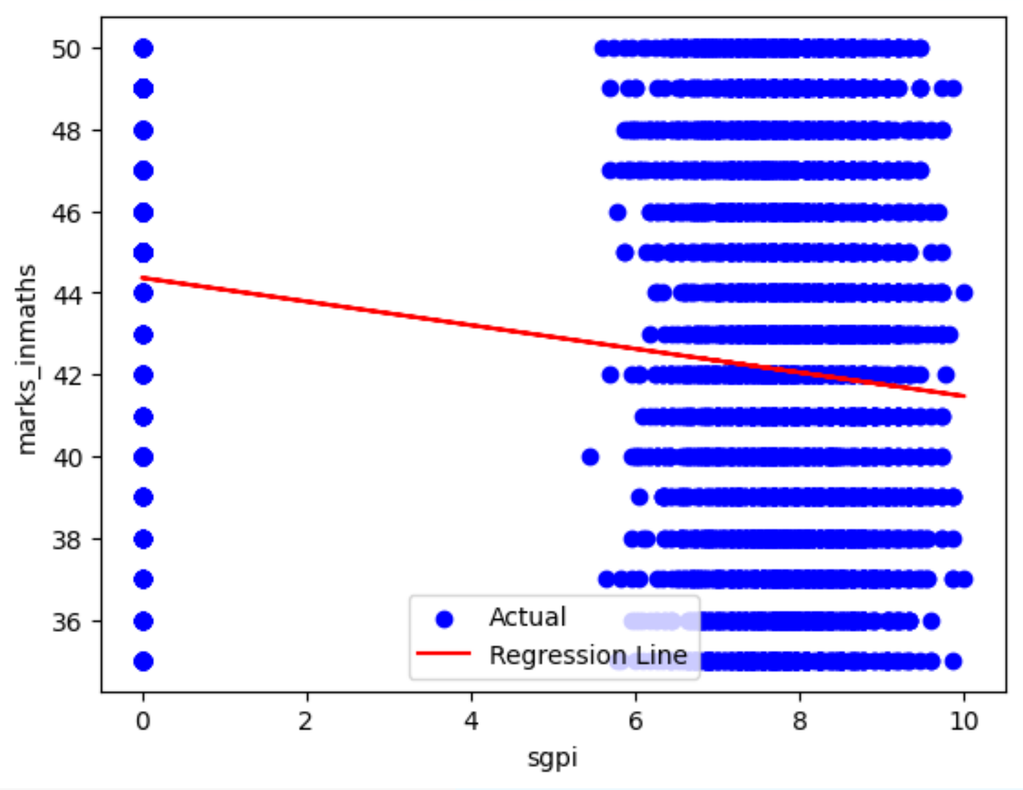
  1)Forecasting
  2)Simulation
  3)Regression
  4)Classification
  5)Clustering

Linear regression

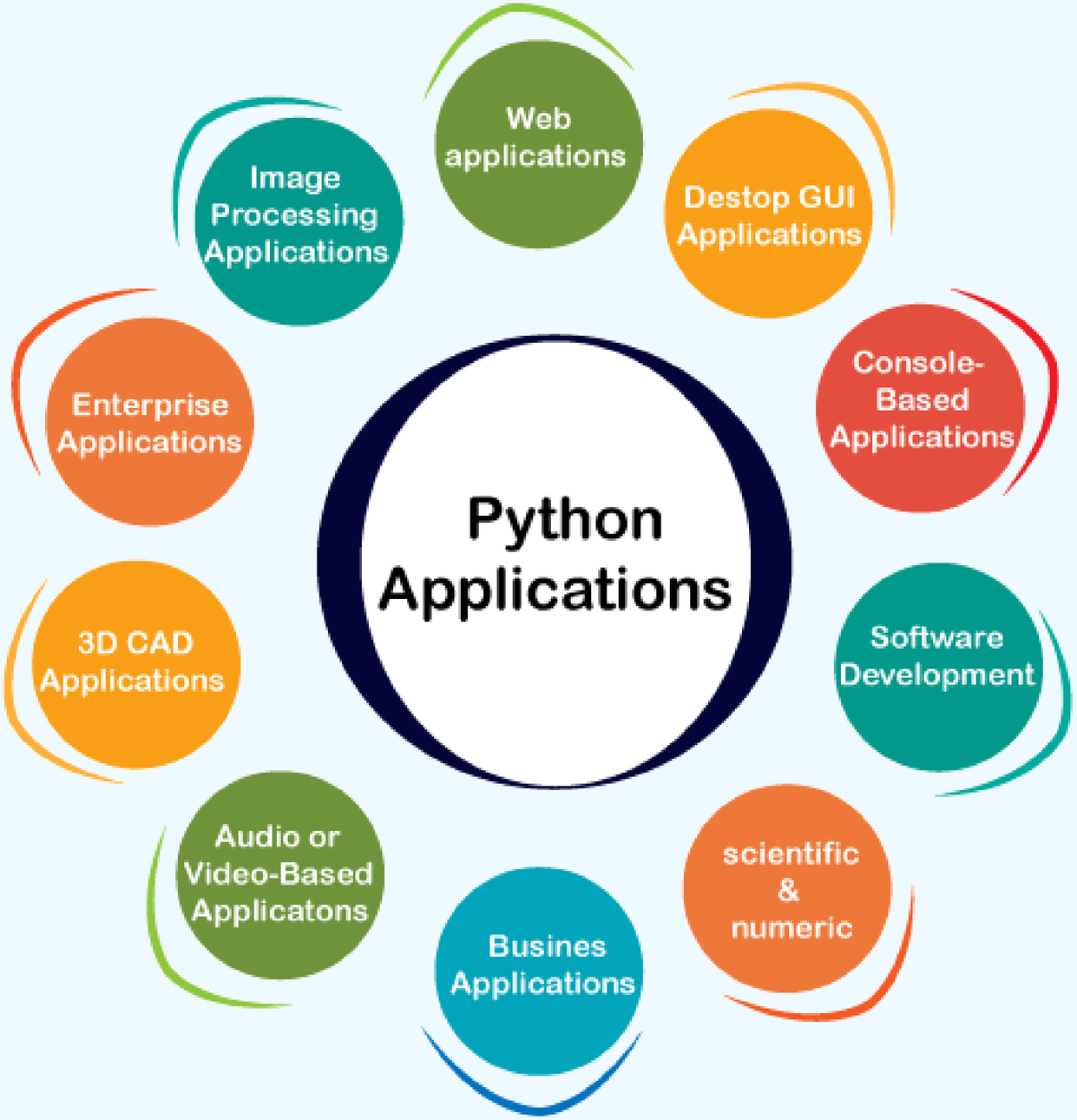
```
#Linear regration
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import seaborn as sns
from sklearn.linear model import LinearRegression
df1=pd.read_csv("/content/sample_data/ass4_dataset.csv")
data = df1.dropna()
print(data)
# Extract the columns for linear regression
X = data['sgpi'].values.reshape(-1, 1)  # Input feature
y = data['year of admission'].values  # Target variable
# Create and fit the linear regression model
model = LinearRegression()
model.fit(X, y)

# Predict the target variable
y pred = model.predict(X)

# Plot the data points and the regression line
plt.scatter(X, y, color='blue', label='Actual')
plt.plot(X, y pred, color='red', label='Regression Line')
plt.xlabel('sgpi')
plt.ylabel('marks inmaths')
plt.legend()
```



```
data = df1.dropna()
print(data)
# Extract the columns for linear regression
X = data['sgpi'].values.reshape(-1, 1)  # Input feature
y = data['marks inmaths'].values  # Target variable
X.sort()
y.sort()
# Create and fit the linear regression model
model = LinearRegression()
model.fit(X, y)

# Predict the target variable
y pred = model.predict(X)

# Plot the data points and the regression line
plt.scatter(X, y, color='blue', label='Actual')
plt.plot(X, y pred, color='red', label='Regression Line')
plt.xlabel('sgpi')
plt.ylabel('marks_inmaths')
plt.legend()
plt.show()
```



Application



Application of Data manipulation

- Data manipulation is used in various industries including accounting, finance, computer programming, banking, sales, marketing and real estate.
- Wrangling any data you need to get the total overview of it, which includes statistical conclusions like standard deviation(std), mean and it’s quartile distributions
- Creates an object with rows and columns called a data frame.

Application of Data Visualization

- Using data visualization, we can get a visual summary of our data
- Features for creating informative, customized, and appealing plots to present data in the most simple and effective way
- Important to understand the challenges and advantages of the different libraries and how to use them to their full potential

Application of Data Predictive Technique (LR/KNN/KMeans)

- Market analysis
- Financial analysis
- Sports analysis
- Visualize the Dataset.
- Splitting Data into Training and Testing Datasets
- KNN Classifier Implementation
- banking to recommendation engines
- cyber security
- document clustering to image segmentation

Conclusion

- Python language is easy to use we can analyse the given data in simple manner
- Python supports both function-oriented and structure-oriented programming.
- It has features of dynamic memory management which can make use of computational resources efficiently.
- It is also compatible with all popular operating systems and platforms

THANK YOU