Department of Information Science and Engineering

6th Semester

Subject:- Data Mining

Submitted by

Somya Srivastava (1NT18IS155)

Sowmya Sri P (1NT18IS158)

Question:

Develop a program to implement Aggregation.

Aggregation:

Data aggregation is the process where data is collected and presented in a summarized format for statistical analysis and to effectively achieve business objectives. Data aggregation is vital to **data warehousing** as it helps to make decisions based on vast amounts of raw data. Data aggregation provides the ability to forecast future trends and aids in **predictive modeling**. Effective data aggregation techniques help to minimize performance problems.

Aggregation provides more information based on related clusters of data such as an individual's income or profession. Queries with aggregation (with mathematical functions) provide faster results. This is because the aggregation is applied on the former query and only the aggregate value is displayed, while the latter query brings up individual records. Faster queries imply the better performance of the system.

Types of aggregation with mathematical functions:

- Sum - Adds together all the specified data to get a total.
- Average - Computes the average value of the specific data.
- Max - Displays the highest value for each category.
- Min - Displays the lowest value for each category.
- Count - Counts the total number of data entries for each category.

Data can also be aggregated by date, allowing trends to be shown over a period of years, quarters, months, etc. These aggregations could be placed in a hierarchy, where you can view the data trends over a period of years, then see the data trends over months for each individual year.

We have also **visualised the data** concluded by aggregation functions.

Furthermore, we have implemented a **Linear Regression model for Predictive modelling**.

Linear Regression

Regression is a form of a supervised machine learning technique that tries to predict any continuous valued attribute. **Linear Regression** is a statistical method that allows us to summarise and study relationships between two continuous (quantitative) variables. It analyses the relationship between a target variable (dependent) and its predictor variable (independent). Regression is an important tool for data analysis that can be used for time series modelling, forecasting, and others.

Regression involves the process of fitting a curve or a straight line on various data points. It is done in such a way that the distances between the curve and the data points come out to be the minimum.

Linear regression performs the task to **predict a dependent variable value** (y) based on a given independent variable (x). So, this regression technique finds out a linear relationship between x (input) and y(output). Hence, the name is Linear Regression.

Hypothesis function for Linear Regression :

$$y = \theta_1 + \theta_2.x$$

While training the model we are given :

x: input training data (univariate – one input variable(parameter))

y: labels to data (supervised learning)

When training the model – it fits the best line to predict the value of y for a given value of x. The model gets the best regression fit line by finding the best θ1 and θ2 values.

θ1: intercept

θ2: coefficient of x

Once we find the best θ1 and θ2 values, we get the best fit line. So when we are finally using our model for prediction, it will predict the value of y for the input value of x.

Dataset used:

Link to dataset: [Predict Test Scores of Students]

It contains information about a test written by some students. It includes features such as:

- school - Name of the school student is enrolled in.
- school_setting - The location of the school
- school_type - Either public or non-public.
- classroom - The type of classroom
- teaching_method - Either experimental or Standard
- n_student - Number of students in the class
- student_id - A unique ID for each student
- gender - Male or Female
- lunch - Whether a student qualifies for free/subsidized lunch
- pretest - The pretest score of the students out of 100
- posttest - Target value

The dimensions of the dataset are **2133 x 11**. It means that we have 2133 rows and 11 columns in the dataset.

Python code:

[Link to google colab notebook]

```
#importing necessary libraries for preprocessing
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
df=pd.read_csv('/content/test_scores.csv')
df.head()
df.describe()
df.isnull().sum()
df.dtypes
df.groupby(['school' ] ).count()['school_setting']
maximum = (df.groupby(['school' ] ).count()['school_setting']).max()
minimum= (df.groupby(['school' ] ).count()['school_setting']).min()
a=dict(df.groupby(['school' ] ).count()['school_setting'])
a
list(a.keys())
for i in list(a.keys()):
   if df.groupby(['school' ] ).count()['school_setting'][i]==maximum:
      print("The maximum no of students are in ",i, "school having
:",maximum,"students")
      break
for i in list(a.keys()):
   if df.groupby(['school' ] ).count()['school_setting'][i]==minimum:
      print("The minimum no of students are in ",i,"school having
:",minimum,"students")
      break
z=dict(df.groupby(['school' ] ).count()['school_setting'])
z1=list(z.keys())
z2=list(z.values())
plt.plot(z1, z2, marker='o')
plt.xticks(rotation="vertical")
plt.xlabel("School Names")
plt.ylabel("No of Students")
plt.title("Student Statistics")
```

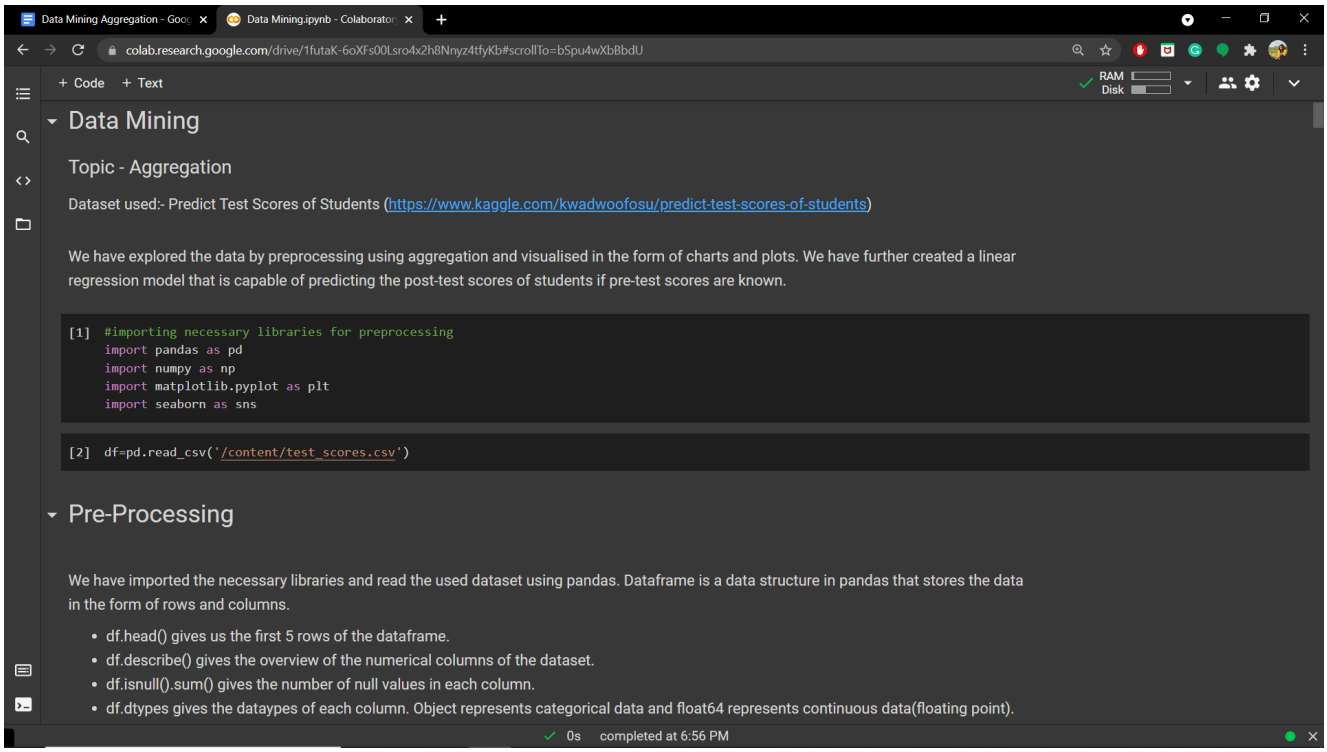
```
plt.tight_layout()
df.groupby(['school' ] )['pretest'].mean()
w=dict(df.groupby(['school' ] )['pretest'].mean())
w1=list(w.keys())
w2=list(w.values())
plt.plot(w1, w2, marker='o')
plt.xticks(rotation="vertical")
plt.xlabel("School Names")
plt.ylabel("Avg Score in each school")
plt.title("Student Statistics")
plt.tight_layout()
df.head()
df.groupby(['school_setting' ] ).count()
print("The no of students studying in different locations are: ")
df.groupby(['school_setting' ] ).count()['school']
x1=list(dict(df.groupby(['school_setting' ] ).count()['school']).keys())
y1=list(dict(df.groupby(['school_setting' ] ).count()['school']).values())
plt.pie(y1,labels=x1,startangle=90, shadow=False,
wedgeprops={"edgecolor":"1",'linewidth': 3, 'antialiased':
True},autopct='%1.2f%%')
plt.title("School Location Vs Number Of Students")
plt.axis('equal')
plt.tight_layout()
plt.show()
#relation b/w  school  and the pretest score
print("The average scores of different students in different locations are :")
df.groupby(['school_setting' ] )['pretest'].mean()
g=list(dict(df.groupby(['school_setting' ] )['pretest'].mean()).keys())
g1=list(dict(df.groupby(['school_setting' ] )['pretest'].mean()).values())
plt.bar(g, g1, color='b',width=0.5)
plt.plot()
plt.xlabel("School Setting")
plt.ylabel("Avg Score Of Students")
plt.title("School Location Vs Avg Score Of Students ")
plt.tight_layout()
plt.show()
#relation b/w  school types and the pretest score
df.groupby(['school_type' ] ).count()['school']
print("The average scores of students in public and non-public schools are:")
df.groupby(['school_type' ] )['pretest'].mean()
```

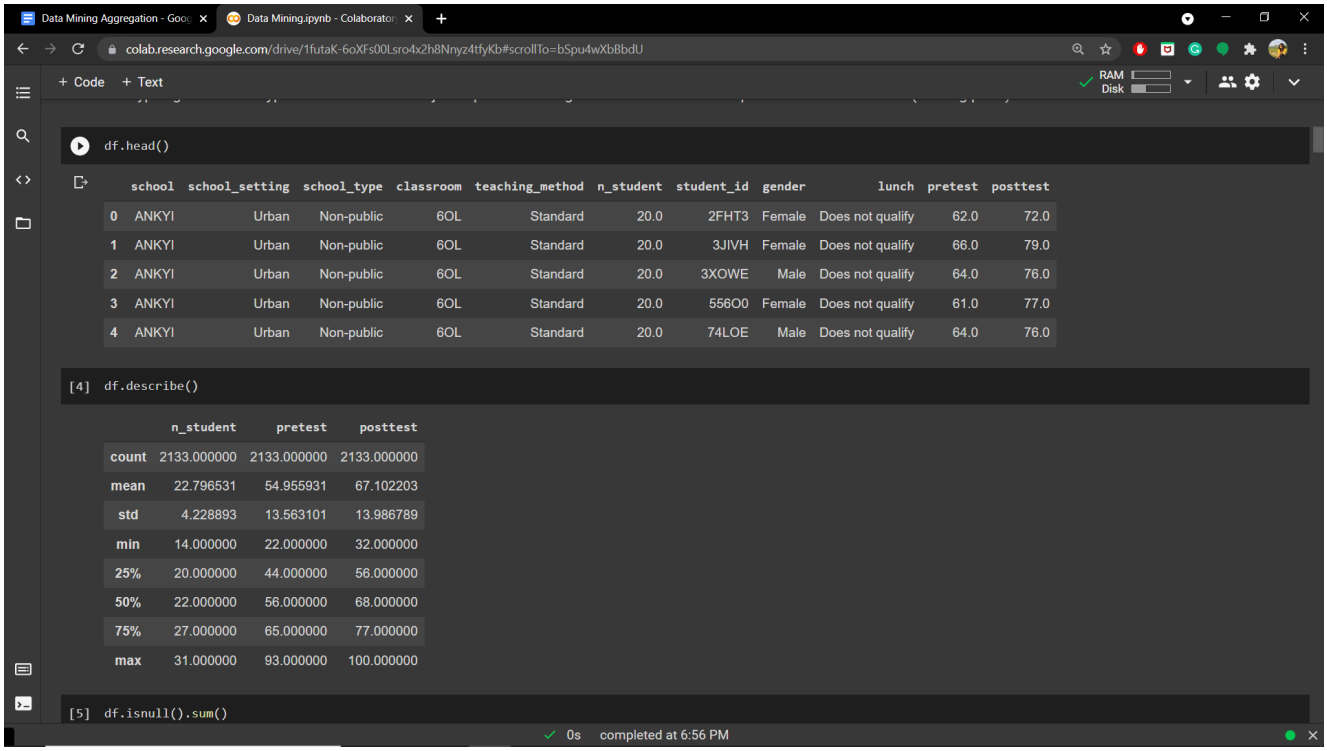
```
z3=list(dict(df.groupby(['school_type' ] )['pretest'].mean()).keys())
z4=list(dict(df.groupby(['school_type' ] )['pretest'].mean()).values())
plt.pie(z4,labels=z3,startangle=90, shadow=False,
wedgeprops={"edgecolor":"1",'linewidth': 3, 'antialiased': True})
plt.title("Avg Marks Of Students in Different School Types")
plt.axis('equal')
plt.tight_layout()
plt.show()
df.head()
df.groupby(['teaching_method' ] )['pretest'].count()
print("We can see the average scores of students for different teachniques of
teaching ")
df.groupby(['teaching_method' ] )['pretest'].mean()
plt.pie([57.055263,53.793882],labels=['Experimental','Standard'],startangle=90
, shadow=False, wedgeprops={"edgecolor":"1",'linewidth': 3, 'antialiased':
True},autopct='%1.2f%%')
plt.title("Avg Marks Of Students in Different School Types")
plt.axis('equal')
plt.tight_layout()
plt.show()
df.head()
df.groupby(['classroom' ] )['n_student'].count()
df.groupby(['classroom' ] )['n_student'].count().min()
df.groupby(['classroom' ] )['n_student'].count().mean()
df.groupby(['classroom' ] )['pretest'].mean()
df.head()
df.groupby(['gender' ] )['school'].count()
df.groupby(['gender' ] )['pretest'].mean()
data = pd.read_csv("/content/test_scores.csv", index_col ="gender")
f=data.loc["Female"]
m=data.loc["Male"]
fem=list(dict(f.groupby(['school'])['school_setting'].count()).values())
mal=list(dict(m.groupby(['school'])['school_setting'].count()).values())
# plot bars in stack manner
plt.bar(z1, fem,bottom=mal,label='Female',color='lightsalmon')
plt.bar(z1, mal,label='male',color='darkturquoise')
plt.xticks(rotation='vertical')
plt.legend()
plt.show()
plt.figure(figsize=(10,7))
```

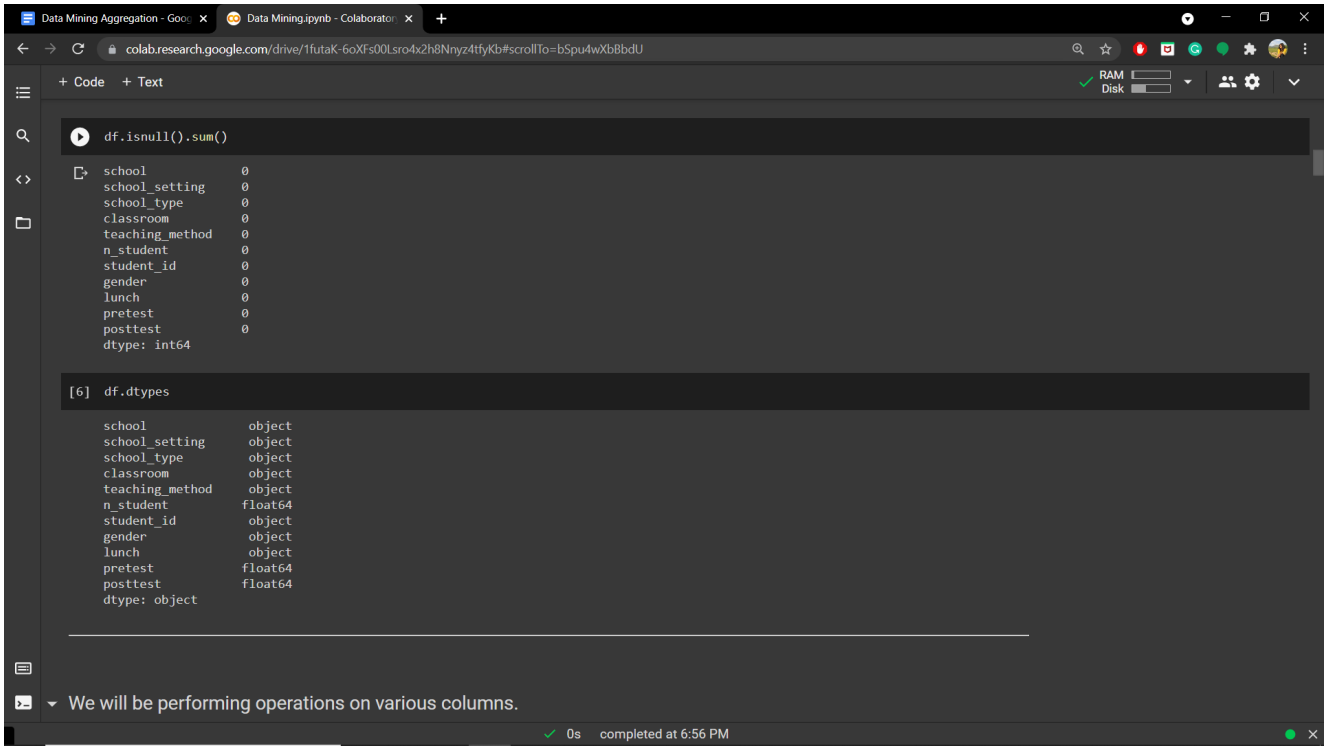
```
plt.title('Plot of Post Test Scores vs Pretest Scores')
sns.scatterplot(data=df, x='pretest', y='posttest')
plt.figure(figsize=(10,5))
sns.kdeplot(data=df['pretest'], shade=True, label='Pre-test')
sns.kdeplot(data=df['posttest'], shade=True, label='Post-test')
plt.title('Distribution of Pre Test and Post Test')
plt.legend()
plt.show()
print( "variance of pretest: ",df['pretest'].var())
print("variance of posttest: ",df['posttest'].var())
print("standard deviation of pretest scores: ",df['pretest'].std())
print("standard deviation of posttest scores: ",df['posttest'].std())
print("mean of pretest: ",df['pretest'].mean())
print("mean of posttest: ",df['posttest'].mean())
fig , ax= plt.subplots(1, 2, figsize=(10, 5))
sns.boxplot( y='pretest', data=df, ax=ax[0])
sns.boxplot( y='posttest', data=df, ax=ax[1])


from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_error
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
# Prepare for model building

# Utility function for evaluating model performance
def eval(y, y_hat):
    MAE = mean_absolute_error(y, y_hat)
    MSE = mean_squared_error(y, y_hat)
    r2 = r2_score(y, y_hat)
    print(f'Mean Abs Error: {MAE:.2f}\nMean Square Error: {MSE:.2f}\nR^2
Score: {r2:.2f}')
    return (MAE, MSE, r2)

x = df[['pretest', 'n_student', 'school_setting', 'school_type',
'teaching_method', 'lunch']]
y = df[['posttest']]

# Encode categorical data
```

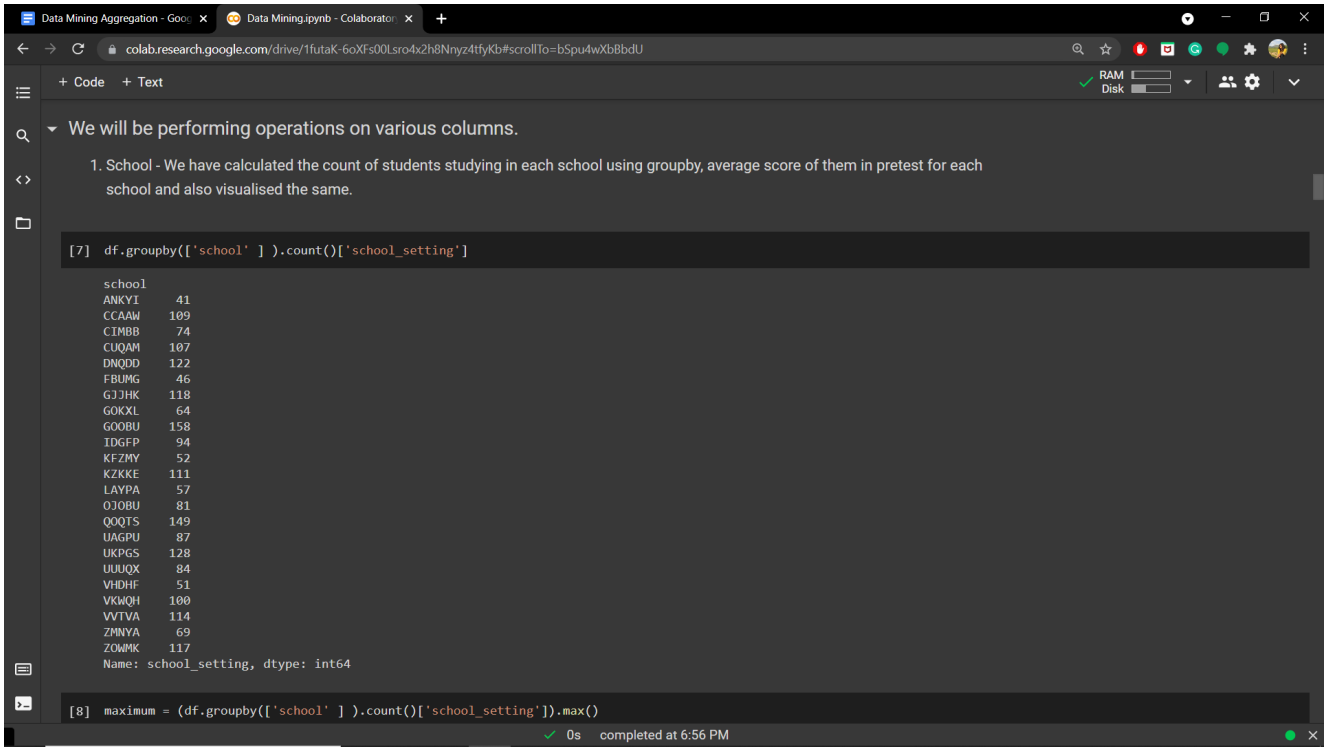
```
x = pd.get_dummies(x)
x
x_simple = x[['pretest']]

x_train, x_test, y_train, y_test = train_test_split(x_simple, y, test_size =
0.4, random_state = 0)
model_linear = LinearRegression()

model_linear.fit(x_train, y_train)
y_predict = model_linear.predict(x_test)

eval(y_test, y_predict)

model_linear.predict([[60]])
```

Snapshots of output:






Data Mining Aggregation - Goog
Data Mining.ipynb - Colaborator
colab.research.google.com/drive/1futaK-6oXFs00Lsro4x2h8Nnyz4tfyKb#scrollTo=bSpu4wXbBbdU
+ Code + Text
RAM
Disk
df.isnull().sum()
school 0
school_setting 0
school_type 0
classroom 0
teaching_method 0
n_student 0
student_id 0
gender 0
lunch 0
pretest 0
posttest 0
dtype: int64
[6] df.dtypes
school object
school_setting object
school_type object
classroom object
teaching_method object
n_student float64
student_id object
gender object
lunch object
pretest float64
posttest float64
dtype: object
We will be performing operations on various columns.
0s completed at 6:56 PM


Data Mining Aggregation - Goog
Data Mining.ipynb - Colaborator
colab.research.google.com/drive/1futaK-6oXFs00Lsro4x2h8Nnyz4tfyKb#scrollTo=bSpu4wXbBbdU
+ Code + Text
RAM
Disk
We will be performing operations on various columns.
1. School - We have calculated the count of students studying in each school using groupby, average score of them in pretest for each school and also visualised the same.
[7] df.groupby(['school']).count()['school_setting']
school
ANKYI 41
CCAAW 109
CIMBB 74
CUQAM 107
DNQDD 122
FBUMG 46
GJJHK 118
GOKXL 64
GOOBU 158
IDGFP 94
KFZMY 52
KZKKE 111
LAYPA 57
OJOBU 81
QOQTS 149
UAGPU 87
UKPGS 128
UUUQX 84
VHDHF 51
VKWQH 100
VVTVA 114
ZMNYA 69
ZOWMK 117
Name: school_setting, dtype: int64
[8] maximum = (df.groupby(['school']).count()['school_setting']).max()
0s completed at 6:56 PM

Data Mining Aggregation - Goog | Data Mining.ipynb - Colaborator

colab.research.google.com/drive/1futaK-6oXFs00Lsro4x2h8Nnyz4tfyKb#scrollTo=bSpu4wXbBbdU

+ Code + Text

RAM Disk

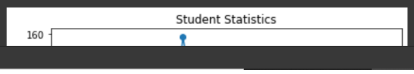
```
[8]  maximum = (df.groupby(['school' ] ).count()['school_setting']).max()
     minimum= (df.groupby(['school' ] ).count()['school_setting']).min()
```

```
[9]  a=dict(df.groupby(['school' ] ).count()['school_setting'])
     a

     list(a.keys())
     for i in list(a.keys()):
        if df.groupby(['school' ] ).count()['school_setting'][i]==maximum:
           print("The maximum no of students are in ",i, "school having :",maximum,"students")
           break
     for i in list(a.keys()):
        if df.groupby(['school' ] ).count()['school_setting'][i]==minimum:
           print("The minimum no of students are in ",i,"school having :",minimum,"students")
           break
```

```
The maximum no of students are in  GOOBU school having : 158 students
The minimum no of students are in  ANKYI school having : 41 students
```

```
[10] z=dict(df.groupby(['school' ] ).count()['school_setting'])
     z1=list(z.keys())
     z2=list(z.values())
```

```
[11] plt.plot(z1, z2, marker='o')
     plt.xticks(rotation="vertical")
     plt.xlabel("School Names")
     plt.ylabel("No of Students")
     plt.title("Student Statistics")
     plt.tight_layout()
```



0s completed at 6:56 PM

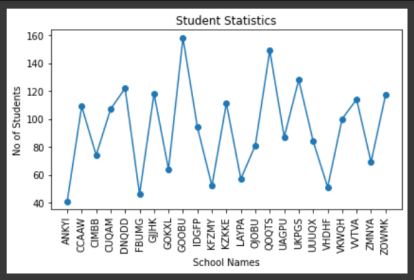
Data Mining Aggregation - Goog | Data Mining.ipynb - Colaborator

colab.research.google.com/drive/1futaK-6oXFs00Lsro4x2h8Nnyz4tfyKb#scrollTo=bSpu4wXbBbdU

+ Code + Text

RAM Disk

[11]



The plot supports our inference of the schools having maximum and minimum count of students. It can be clearly seen that "GOOBU" has the greatest number of students, and "ANKYI" has the least number of students.

```
[12] df.groupby(['school' ] )['pretest'].mean()
```

```
school
ANKYI    61.341463
CCAAW    64.623853
CIMBB    65.067568
CUQAM    53.925234
DNQDD    54.327869
FBUMG    62.891304
GJJHK    53.194915
GOKXL    50.796875
GOOBU    38.196203
IDGFP    75.202128
KFZMY    41.865385
KZKKE    37.261261
```

0s completed at 6:56 PM

Data Mining Aggregation - Goog | Data Mining.ipynb - Colaborator

colab.research.google.com/drive/1futaK-6oXFs00Lsro4x2h8Nnyz4tfyKb#scrollTo=bSpu4wXbBbdU

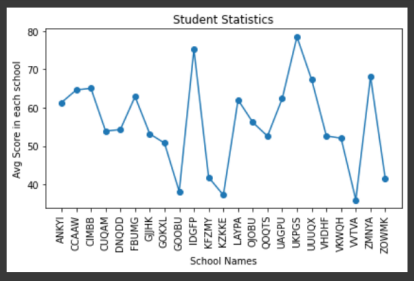
+ Code + Text

RAM Disk

```
Name: pretest, dtype: float64
```

```
[13] w=dict(df.groupby(['school' ] )['pretest'].mean())
     w1=list(w.keys())
     w2=list(w.values())
```

```
plt.plot(w1, w2, marker='o')
plt.xticks(rotation="vertical")
plt.xlabel("School Names")
plt.ylabel("Avg Score in each school")
plt.title("Student Statistics")
plt.tight_layout()
```



The graph shows average scores of students in each school. We can infer that the average score of students in pretest is maximum in "UKPGS" and minimum in "WTVA".

0s completed at 6:56 PM

Data Mining Aggregation - Goog | Data Mining.ipynb - Colaborator

colab.research.google.com/drive/1futaK-6oXFs00Lsro4x2h8Nnyz4tfyKb#scrollTo=bSpu4wXbBbdU

+ Code + Text

RAM Disk

2. **School Setting** - We have calculated the count of students studying in different school settings using groupby, average score of them in pretest for each school and also visualised the same.

```
[15] df.head()
```

	school	school_setting	school_type	classroom	teaching_method	n_student	student_id	gender	lunch	pretest	posttest
0	ANKYI	Urban	Non-public	6OL	Standard	20.0	2FHT3	Female	Does not qualify	62.0	72.0
1	ANKYI	Urban	Non-public	6OL	Standard	20.0	3JIVH	Female	Does not qualify	66.0	79.0
2	ANKYI	Urban	Non-public	6OL	Standard	20.0	3XOWE	Male	Does not qualify	64.0	76.0
3	ANKYI	Urban	Non-public	6OL	Standard	20.0	556O0	Female	Does not qualify	61.0	77.0
4	ANKYI	Urban	Non-public	6OL	Standard	20.0	74LOE	Male	Does not qualify	64.0	76.0

```
[16] df.groupby(['school_setting' ] ).count()
```

school_setting	school	school_type	classroom	teaching_method	n_student	student_id	gender	lunch	pretest	posttest
Rural	510	510	510	510	510	510	510	510	510	510
Suburban	717	717	717	717	717	717	717	717	717	717
Urban	906	906	906	906	906	906	906	906	906	906

```
[17] print("The no of students studying in different locations are: ")
     df.groupby(['school_setting' ] ).count()['school']

The no of students studying in different locations are:
```

0s completed at 6:56 PM

Data Mining Aggregation - Goog | Data Mining.ipynb - Colaborator

colab.research.google.com/drive/1futaK-6oXFs00Lsro4x2h8Nnyz4tfyKb#scrollTo=bSpu4wXbBbdU

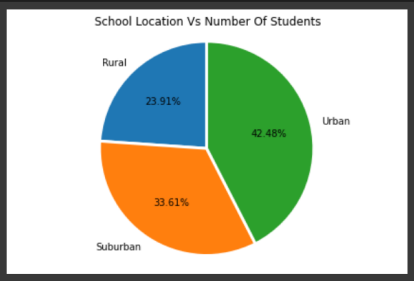
+ Code + Text

RAM Disk

```
[17] The no of students studying in different locations are:
     school_setting
     Rural       510
     Suburban    717
     Urban       906
     Name: school, dtype: int64
```

```
[18] x1=list(dict(df.groupby(['school_setting' ] ).count()['school']).keys())
     y1=list(dict(df.groupby(['school_setting' ] ).count()['school']).values())
```

```
[19] plt.pie(y1,labels=x1,startangle=90, shadow=False, wedgeprops={"edgecolor":"1",'linewidth': 3, 'antialiased': True},autopct='%1.2f%%')
     plt.title("School Location Vs Number Of Students")
     plt.axis('equal')
     plt.tight_layout()
     plt.show()
```



We can infer that the largest section of students (42.8%) study in schools in urban areas.

0s completed at 6:56 PM

Data Mining Aggregation - Goog | Data Mining.ipynb - Colaborator

colab.research.google.com/drive/1futaK-6oXFs00Lsro4x2h8Nnyz4tfyKb#scrollTo=bSpu4wXbBbdU

+ Code + Text

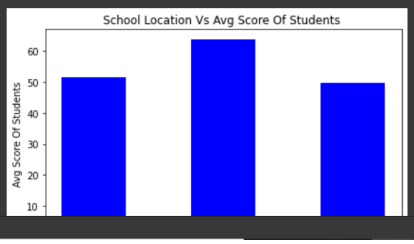
RAM Disk

```
[20] #relation b/w  school  and the pretest score
     print("The average scores of different students in different locations are :")
     df.groupby(['school_setting' ] )['pretest'].mean()
```

```
     The average scores of different students in different locations are :
     school_setting
     Rural       51.490196
     Suburban    63.841004
     Urban       49.875276
     Name: pretest, dtype: float64
```

```
[21] g=list(dict(df.groupby(['school_setting' ] )['pretest'].mean()).keys())
     g1=list(dict(df.groupby(['school_setting' ] )['pretest'].mean()).values())
```

```
[22] plt.bar(g, g1, color='b',width=0.5)
     plt.plot()
     plt.xlabel("School Setting")
     plt.ylabel("Avg Score Of Students")
     plt.title("School Location Vs Avg Score Of Students ")
     plt.tight_layout()
     plt.show()
```



0s completed at 6:56 PM

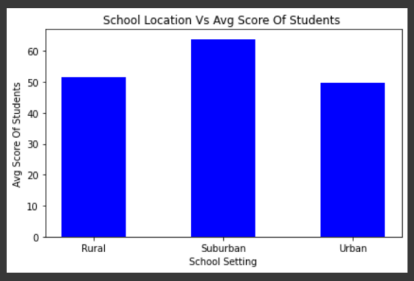
Data Mining Aggregation - Goog | Data Mining.ipynb - Colaborator

colab.research.google.com/drive/1futaK-6oXFs00Lsro4x2h8Nnyz4tfyKb#scrollTo=bSpu4wXbBbdU

+ Code + Text

RAM Disk

[22]



The chart represents that students studying in sub-urban areas are scoring the best, though maximum number of students study in Urban areas.

3. **School Type** - We have calculated the count of students studying in different school types using groupby, average score of them in pretest for each school and also visualised the same.

```
[23] #relation b/w  school types and the pretest score
     df.groupby(['school_type' ] ).count()['school']
```

```
school_type
Non-public     551
Public        1582
Name: school, dtype: int64
```

0s completed at 6:56 PM

Data Mining Aggregation - Goog | Data Mining.ipynb - Colaborator

colab.research.google.com/drive/1futaK-6oXFs00Lsro4x2h8Nnyz4tfyKb#scrollTo=bSpu4wXbBbdU

+ Code + Text

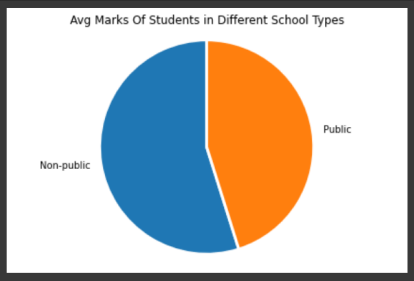
RAM Disk

```
[24] print("The average scores of students in public and non-public schools are:")
     df.groupby(['school_type' ] )['pretest'].mean()
```

```
The average scores of students in public and non-public schools are:
school_type
Non-public    63.254083
Public        52.065740
Name: pretest, dtype: float64
```

```
z3=list(dict(df.groupby(['school_type' ] )['pretest'].mean()).keys())
z4=list(dict(df.groupby(['school_type' ] )['pretest'].mean()).values())
```

```
[26] plt.pie(z4,labels=z3,startangle=90, shadow=False, wedgeprops={"edgecolor":"1",'linewidth': 3, 'antialiased': True})
     plt.title("Avg Marks Of Students in Different School Types")
     plt.axis('equal')
     plt.tight_layout()
     plt.show()
```



0s completed at 6:56 PM

Data Mining Aggregation - Goog | Data Mining.ipynb - Colaborator

colab.research.google.com/drive/1futaK-6oXFs00Lsro4x2h8Nnyz4tfyKb#scrollTo=bSpu4wXbBbdU

+ Code + Text

RAM Disk

On visualising the average scores of students studying in different types of schools, we can infer that students studying in Non-public sector score better in pretests than the ones in Public sectors.

4. **Teaching method** - We have calculated the count of students undergoing different teaching methods and analysed its impact on students' peformance.

```
[27] df.head()
```

	school	school_setting	school_type	classroom	teaching_method	n_student	student_id	gender	lunch	pretest	posttest
0	ANKYI	Urban	Non-public	6OL	Standard	20.0	2FHT3	Female	Does not qualify	62.0	72.0
1	ANKYI	Urban	Non-public	6OL	Standard	20.0	3JIVH	Female	Does not qualify	66.0	79.0
2	ANKYI	Urban	Non-public	6OL	Standard	20.0	3XOWE	Male	Does not qualify	64.0	76.0
3	ANKYI	Urban	Non-public	6OL	Standard	20.0	556O0	Female	Does not qualify	61.0	77.0
4	ANKYI	Urban	Non-public	6OL	Standard	20.0	74LOE	Male	Does not qualify	64.0	76.0

```
[28] df.groupby(['teaching_method' ] )['pretest'].count()
```

```
teaching_method
Experimental     760
Standard        1373
Name: pretest, dtype: int64
```

```
[29] print("We can see the average scores of students for different teachniques of teaching ")
     df.groupby(['teaching_method' ] )['pretest'].mean()
```

```
We can see the average scores of students for different teachniques of teaching
teaching_method
```

0s completed at 6:56 PM

Data Mining Aggregation - Goog | Data Mining.ipynb - Colaborator

colab.research.google.com/drive/1futaK-6oXFs00Lsro4x2h8Nnyz4tfyKb#scrollTo=bSpu4wXbBbdU

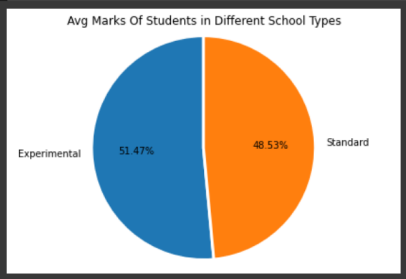
+ Code + Text

RAM Disk

```
[29] df.groupby(['teaching_method' ] )['pretest'].mean()
```

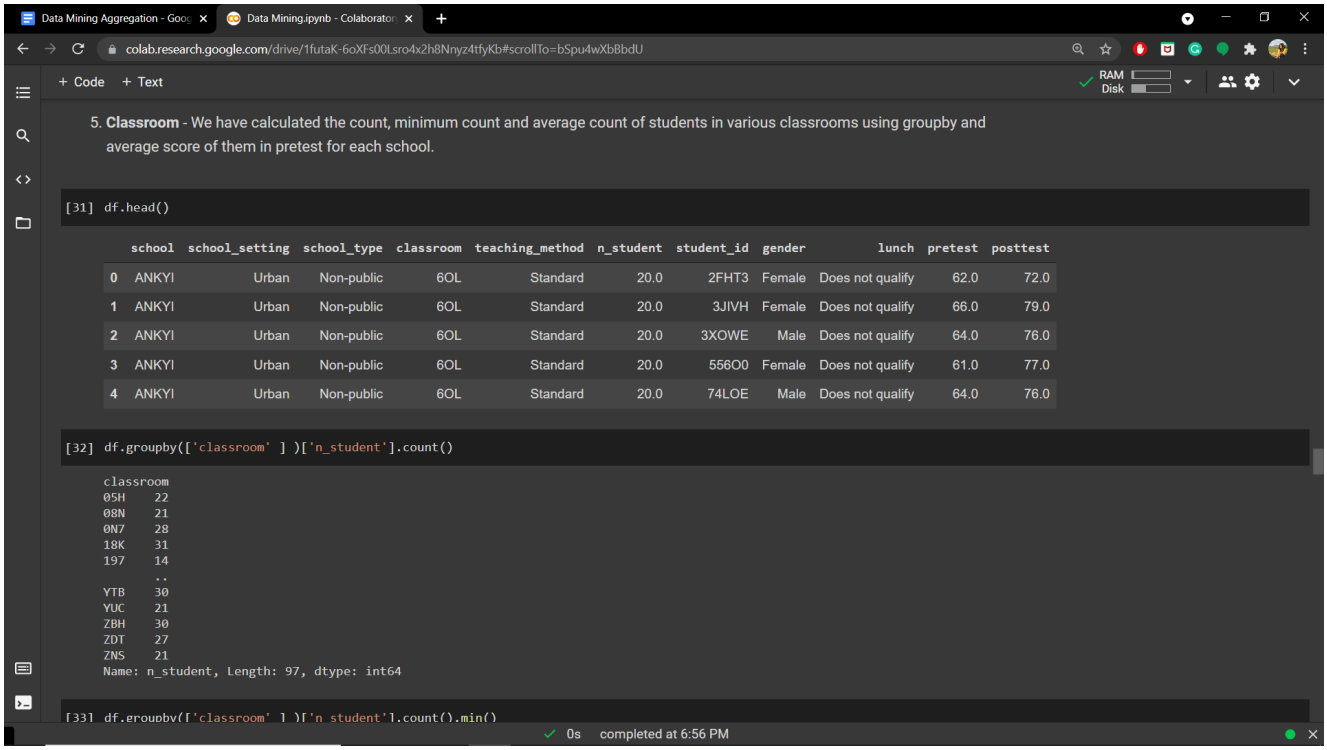
```
We can see the average scores of students for different teachniques of teaching
teaching_method
Experimental    57.055263
Standard        53.793882
Name: pretest, dtype: float64
```

```
plt.pie([57.055263,53.793882],labels=['Experimental','Standard'],startangle=90, shadow=False, wedgeprops={"edgecolor":"1",'linewidth': 3, 'antialiased': True},autopc
plt.title("Avg Marks Of Students in Different School Types")
plt.axis('equal')
plt.tight_layout()
plt.show()
```



Conventional(Standard) methods of teaching have proved to be slightly less impactful on students when compared to the Experimental methods of teaching.

0s completed at 6:56 PM



5. **Classroom** - We have calculated the count, minimum count and average count of students in various classrooms using groupby and average score of them in pretest for each school.

```
[31] df.head()
```

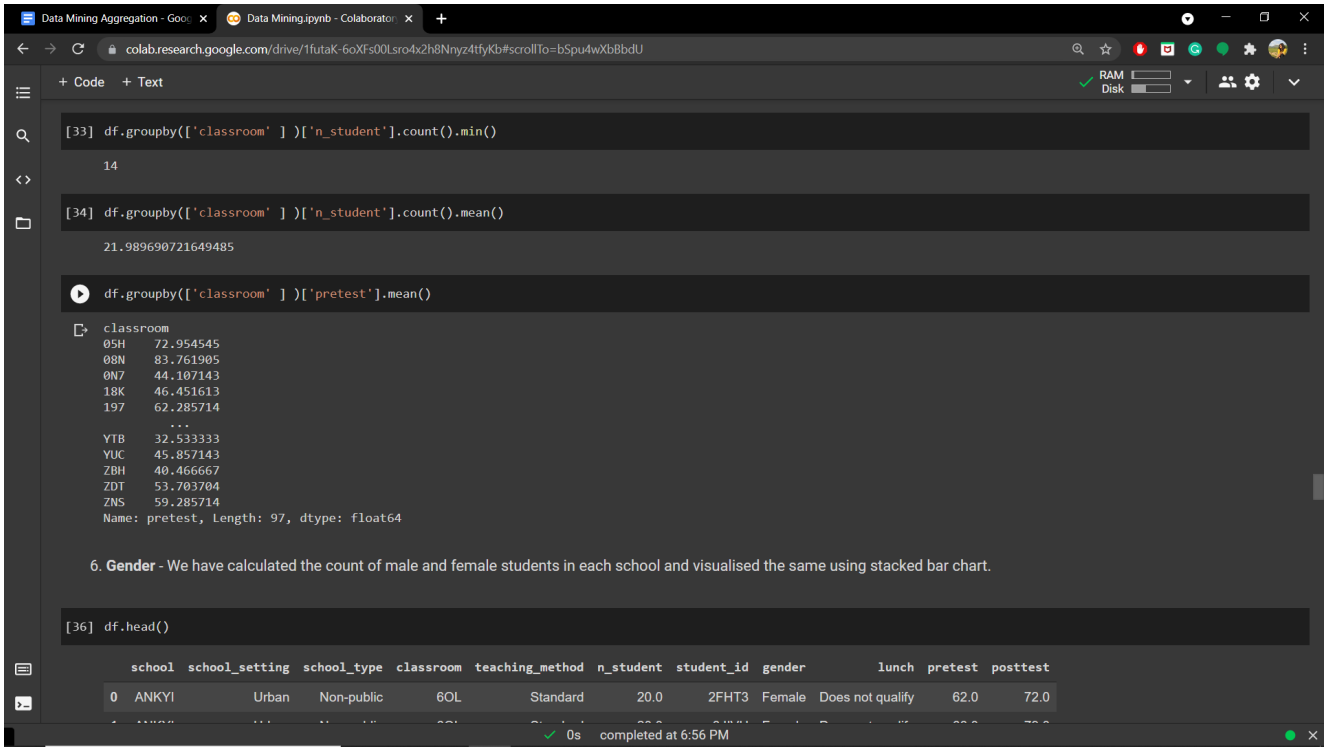
	school	school_setting	school_type	classroom	teaching_method	n_student	student_id	gender	lunch	pretest	posttest
0	ANKYI	Urban	Non-public	6OL	Standard	20.0	2FHT3	Female	Does not qualify	62.0	72.0
1	ANKYI	Urban	Non-public	6OL	Standard	20.0	3JIVH	Female	Does not qualify	66.0	79.0
2	ANKYI	Urban	Non-public	6OL	Standard	20.0	3XOWE	Male	Does not qualify	64.0	76.0
3	ANKYI	Urban	Non-public	6OL	Standard	20.0	556O0	Female	Does not qualify	61.0	77.0
4	ANKYI	Urban	Non-public	6OL	Standard	20.0	74LOE	Male	Does not qualify	64.0	76.0

```
[32] df.groupby(['classroom' ] )['n_student'].count()
```

```
classroom
05H    22
08N    21
0N7    28
18K    31
197    14
       ..
YTB    30
YUC    21
ZBH    30
ZDT    27
ZNS    21
Name: n_student, Length: 97, dtype: int64
```

```
[33] df.groupby(['classroom' ] )['n_student'].count().min()
```

0s completed at 6:56 PM




```
[33] df.groupby(['classroom' ] )['n_student'].count().min()
```

```
14
```

```
[34] df.groupby(['classroom' ] )['n_student'].count().mean()
```

```
21.989690721649485
```

```
df.groupby(['classroom' ] )['pretest'].mean()
```

```
classroom
05H    72.954545
08N    83.761905
0N7    44.107143
18K    46.451613
197    62.285714
          ...
YTB    32.533333
YUC    45.857143
ZBH    40.466667
ZDT    53.703704
ZNS    59.285714
Name: pretest, Length: 97, dtype: float64
```

6. **Gender** - We have calculated the count of male and female students in each school and visualised the same using stacked bar chart.

```
[36] df.head()
```

	school	school_setting	school_type	classroom	teaching_method	n_student	student_id	gender	lunch	pretest	posttest
0	ANKYI	Urban	Non-public	6OL	Standard	20.0	2FHT3	Female	Does not qualify	62.0	72.0

0s completed at 6:56 PM

Data Mining Aggregation - Goog | Data Mining.ipynb - Colaborator

colab.research.google.com/drive/1futaK-6oXFs00Lsro4x2h8Nnyz4tfyKb#scrollTo=bSpu4wXbBbdU

+ Code + Text

RAM Disk

```
[36] df.head()
```

	school	school_setting	school_type	classroom	teaching_method	n_student	student_id	gender	lunch	pretest	posttest
0	ANKYI	Urban	Non-public	6OL	Standard	20.0	2FHT3	Female	Does not qualify	62.0	72.0
1	ANKYI	Urban	Non-public	6OL	Standard	20.0	3JIVH	Female	Does not qualify	66.0	79.0
2	ANKYI	Urban	Non-public	6OL	Standard	20.0	3XOWE	Male	Does not qualify	64.0	76.0
3	ANKYI	Urban	Non-public	6OL	Standard	20.0	556O0	Female	Does not qualify	61.0	77.0
4	ANKYI	Urban	Non-public	6OL	Standard	20.0	74LOE	Male	Does not qualify	64.0	76.0

```
[37] df.groupby(['gender' ] )['school'].count()
```

```
gender
Female    1056
Male      1077
Name: school, dtype: int64
```

```
[38] df.groupby(['gender' ] )['pretest'].mean()
```

```
gender
Female    54.873106
Male      55.037140
Name: pretest, dtype: float64
```

```
[39] data = pd.read_csv("/content/test_scores.csv", index_col ="gender")
     f=data.loc["Female"]
     m=data.loc["Male"]
```

0s completed at 6:56 PM

Data Mining Aggregation - Goog | Data Mining.ipynb - Colaborator

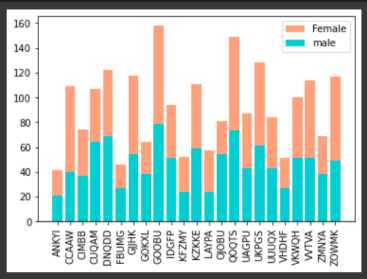
colab.research.google.com/drive/1futaK-6oXFs00Lsro4x2h8Nnyz4tfyKb#scrollTo=bSpu4wXbBbdU

+ Code + Text

RAM Disk

```
[40] fem=list(dict(f.groupby(['school'])['school_setting'].count()).values())
     mal=list(dict(m.groupby(['school'])['school_setting'].count()).values())
```

```
[41] # plot bars in stack manner
     plt.bar(z1, fem,bottom=mal,label='Female',color='lightsalmon')
     plt.bar(z1, mal,label='male',color='darkturquoise')
     plt.xticks(rotation='vertical')
     plt.legend()
     plt.show()
```



The graph represents the distribution of girls and boys in every school.

```
[42] plt.figure(figsize=(10,7))
     plt.title('Plot of Post Test Scores vs Pretest Scores')
     sns.scatterplot(data=df, x='pretest', y='posttest')
```

0s completed at 6:56 PM

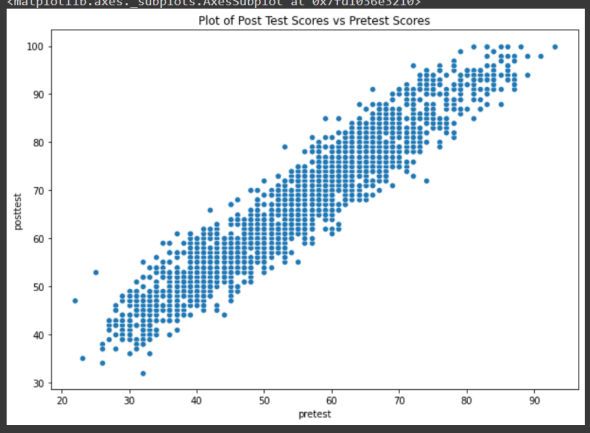
Data Mining Aggregation - Goog | Data Mining.ipynb - Colaborator

colab.research.google.com/drive/1futaK-6oXFs00Lsro4x2h8Nnyz4tfyKb#scrollTo=bSpu4wXbBbdU

+ Code + Text

RAM Disk

[42]
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd1036e3210>
```



The scatterplot shows the relationship between the pretest and posttest scores, we can see that they have a positive correlation .

[43]
```
plt.figure(figsize=(10,5))
sns.kdeplot(data=df['pretest'], shade=True, label='Pre-test')
sns.kdeplot(data=df['posttest'], shade=True, label='Post-test')
plt.title('Distribution of Pre Test and Post Test')
plt.legend()
plt.show()
```

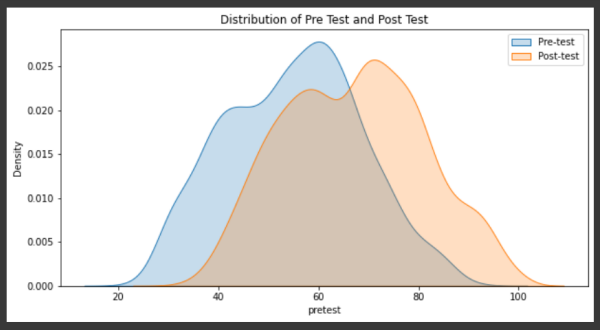
0s completed at 6:56 PM

Data Mining Aggregation - Goog | Data Mining.ipynb - Colaborator

colab.research.google.com/drive/1futaK-6oXFs00Lsro4x2h8Nnyz4tfyKb#scrollTo=bSpu4wXbBbdU

+ Code + Text

RAM Disk



The pretest and the posttest scores seem to be similar and positively correlated, as the density graph shifts to the right. The scores of students tend to increase in the posttest.

[44]
```
print( "variance of pretest: ",df['pretest'].var())
print("variance of posttest: ",df['posttest'].var())
print("standard deviation of pretest scores: ",df['pretest'].std())
print("standard deviation of posttest scores: ",df['posttest'].std())
print("mean of pretest: ",df['pretest'].mean())
print("mean of posttest: ",df['posttest'].mean())
```

```
variance of pretest:  183.9577192672281
variance of posttest:  195.6302624970427
standard deviation of pretest scores:  13.563101388223421
standard deviation of posttest scores:  13.986788858911 4
```

0s completed at 6:56 PM

Data Mining Aggregation - Goog | Data Mining.ipynb - Colaborator

colab.research.google.com/drive/1futaK-6oXFs00Lsro4x2h8Nnyz4tfyKb#scrollTo=bSpu4wXbBbdU

+ Code + Text

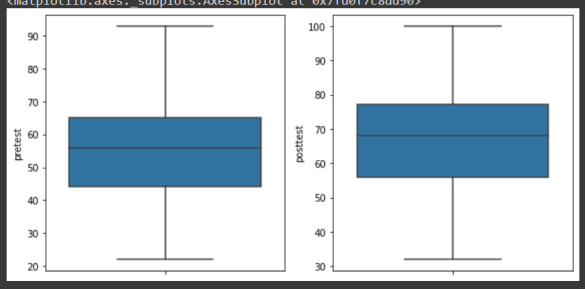
RAM Disk

```
print("mean of pretest: ",df['pretest'].mean())
print("mean of posttest: ",df['posttest'].mean())
```

```
variance of pretest:  183.9577192672281
variance of posttest:  195.6302624970427
standard deviation of pretest scores:  13.563101388223421
standard deviation of posttest scores:  13.98678885581114
mean of pretest:  54.95593061415846
mean of posttest:  67.10220346929208
```

```
fig , ax= plt.subplots(1, 2, figsize=(10, 5))
sns.boxplot( y='pretest', data=df, ax=ax[0])
sns.boxplot( y='posttest', data=df, ax=ax[1])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd0f7c8dd90>
```



We can see that the scores of the posttest are comparitively higher than the pretest.

0s completed at 6:56 PM

Data Mining Aggregation - Goog | Data Mining.ipynb - Colaborator

colab.research.google.com/drive/1futaK-6oXFs00Lsro4x2h8Nnyz4tfyKb#scrollTo=bSpu4wXbBbdU

+ Code + Text

RAM Disk

Model

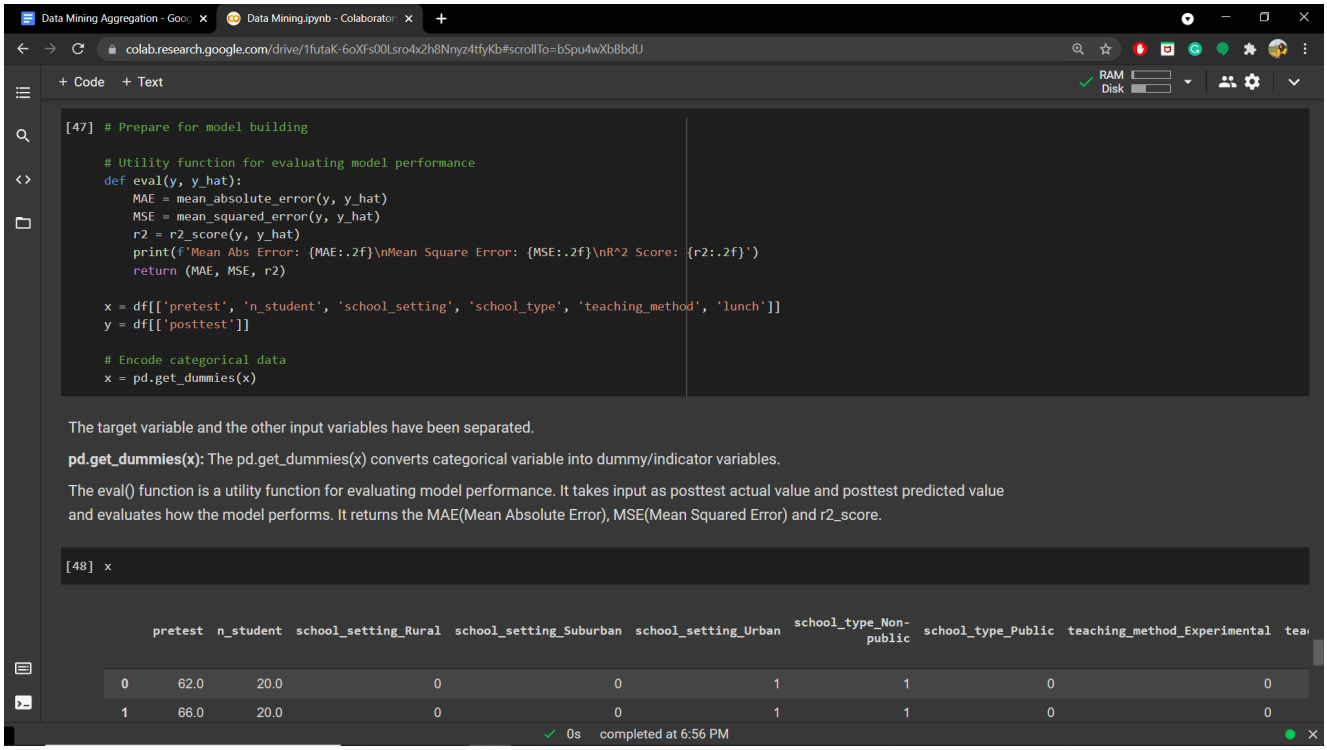
We have used linear regression model to predict the posttest scores of students using the pretest scores. We use **scikitlearn** package for building and training the model.

- **from sklearn.model_selection import train_test_split**: train_test_split is used to split the data for training and testing by partitioning the dataset in required proportions.
- **mean_absolute_error , mean_sqaured_Error**: mean_absolute_error and mean_squared_error are used to evaluate the performance of our model. The lesser the error the better our model performs .
- **from sklearn.linear_model import LinearRegression**: We have imported the linear regression model from sklearn to predict the posttest scores using the pretest scores.
- **from sklearn.metrics import r2_score**: r2_score (coefficient of determination) is a regression score function. The performance of the model is the best if the r2_Score is 1. The closer it is to 1, the better it performs. The value can be negative, if the model performs arbitrarily worse.

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_error
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
```

```
# Prepare for model building

# Utility function for evaluating model performance
def eval(y, y_hat):
    MAE = mean_absolute_error(y, y_hat)
    MSE = mean_squared_error(y, y_hat)
```

0s completed at 6:56 PM



```
[47] # Prepare for model building

     # Utility function for evaluating model performance
     def eval(y, y_hat):
         MAE = mean_absolute_error(y, y_hat)
         MSE = mean_squared_error(y, y_hat)
         r2 = r2_score(y, y_hat)
         print(f'Mean Abs Error: {MAE:.2f}\nMean Square Error: {MSE:.2f}\nR^2 Score: {r2:.2f}')
         return (MAE, MSE, r2)

     x = df[['pretest', 'n_student', 'school_setting', 'school_type', 'teaching_method', 'lunch']]
     y = df[['posttest']]

     # Encode categorical data
     x = pd.get_dummies(x)
```

The target variable and the other input variables have been separated.

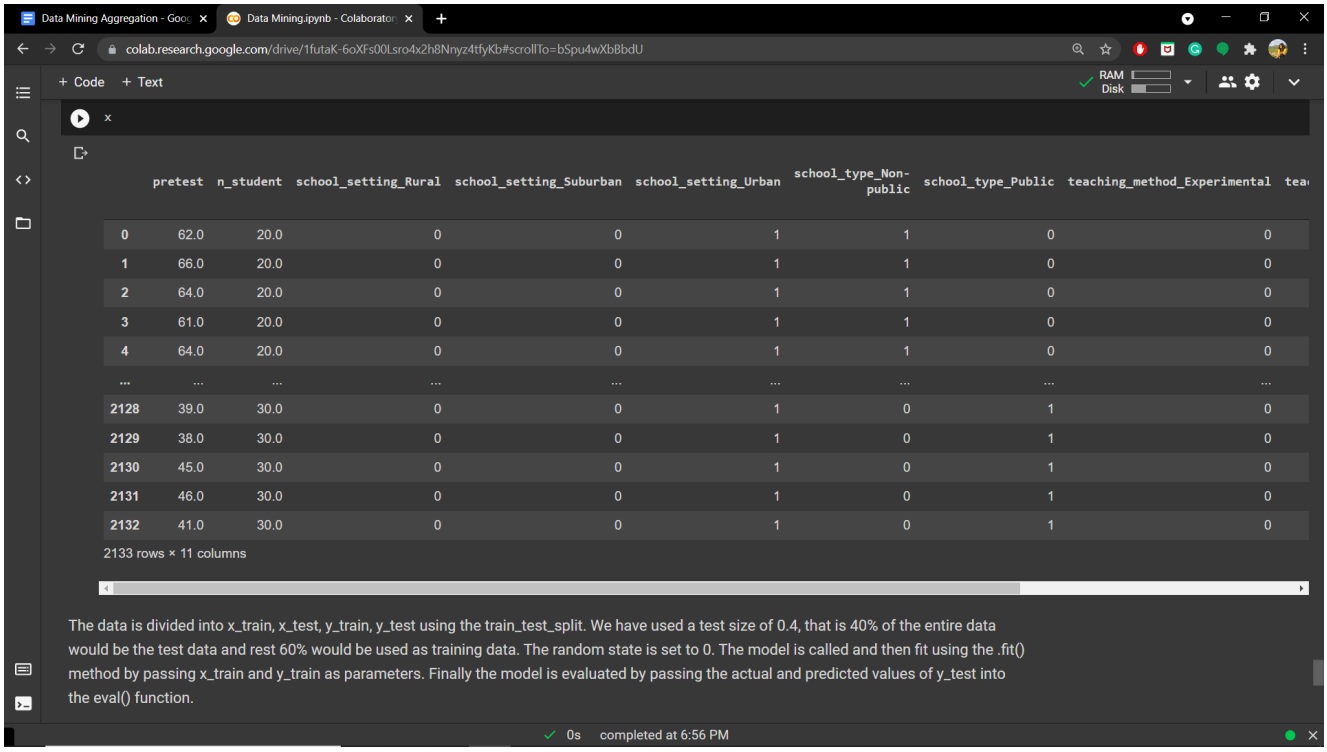
pd.get_dummies(x): The pd.get_dummies(x) converts categorical variable into dummy/indicator variables.

The eval() function is a utility function for evaluating model performance. It takes input as posttest actual value and posttest predicted value and evaluates how the model performs. It returns the MAE(Mean Absolute Error), MSE(Mean Squared Error) and r2_score.

```
[48] x
```

	pretest	n_student	school_setting_Rural	school_setting_Suburban	school_setting_Urban	school_type_Non-public	school_type_Public	teaching_method_Experimental	tea
0	62.0	20.0	0	0	1	1	0	0	
1	66.0	20.0	0	0	1	1	0	0	

0s completed at 6:56 PM

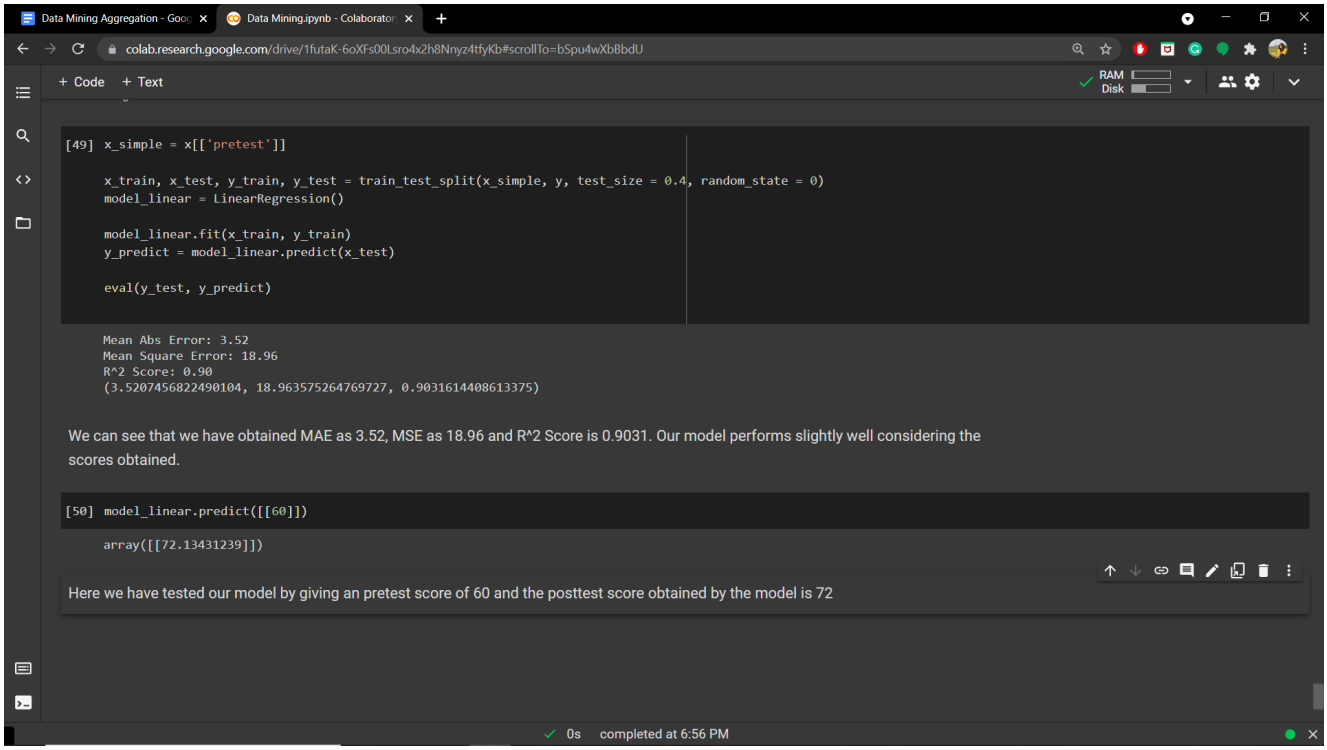


	pretest	n_student	school_setting_Rural	school_setting_Suburban	school_setting_Urban	school_type_Non-public	school_type_Public	teaching_method_Experimental	tea
0	62.0	20.0	0	0	1	1	0	0	
1	66.0	20.0	0	0	1	1	0	0	
2	64.0	20.0	0	0	1	1	0	0	
3	61.0	20.0	0	0	1	1	0	0	
4	64.0	20.0	0	0	1	1	0	0	
...	...	...	...	...	...	...	...	...	
2128	39.0	30.0	0	0	1	0	1	0	
2129	38.0	30.0	0	0	1	0	1	0	
2130	45.0	30.0	0	0	1	0	1	0	
2131	46.0	30.0	0	0	1	0	1	0	
2132	41.0	30.0	0	0	1	0	1	0	

2133 rows × 11 columns

The data is divided into x_train, x_test, y_train, y_test using the train_test_split. We have used a test size of 0.4, that is 40% of the entire data would be the test data and rest 60% would be used as training data. The random state is set to 0. The model is called and then fit using the .fit() method by passing x_train and y_train as parameters. Finally the model is evaluated by passing the actual and predicted values of y_test into the eval() function.

0s completed at 6:56 PM


Data Mining Aggregation - Goog
Data Mining.ipynb - Colaborator
colab.research.google.com/drive/1futaK-6oXFs00Lsro4x2h8Nnyz4tfyKb#scrollTo=bSpu4wXbBbdU
+ Code
+ Text
RAM
Disk
[49] x_simple = x[['pretest']]
x_train, x_test, y_train, y_test = train_test_split(x_simple, y, test_size = 0.4, random_state = 0)
model_linear = LinearRegression()
model_linear.fit(x_train, y_train)
y_predict = model_linear.predict(x_test)
eval(y_test, y_predict)
Mean Abs Error: 3.52
Mean Square Error: 18.96
R^2 Score: 0.90
(3.5207456822490104, 18.963575264769727, 0.9031614408613375)
We can see that we have obtained MAE as 3.52, MSE as 18.96 and R^2 Score is 0.9031. Our model performs slightly well considering the scores obtained.
[50] model_linear.predict([[60]])
array([[72.13431239]])
Here we have tested our model by giving an pretest score of 60 and the posttest score obtained by the model is 72
0s
completed at 6:56 PM