Introduction to Data Science (S1-22_DSECLZG532)-ASSIGNMENT

Group No - 30

Group Member Names:

1. Vinoth N - 2022da04450 - 100%
2. Sarthak Negi -2022da04128 - 100%
3. Chindarkar Gouri - 2022da04091 - 50%
4. Swapnil Madanlal - 2022da04132 - 50%

1. Business Understanding

Students are expected to identify a classification problem of your choice. You have to detail the Business Understanding part of your problem under this heading which basically addresses the following questions.

1. What is the business problem that you are trying to solve?

1. What data do you need to answer the above problem?

1. What are the different sources of data?

1. What kind of analytics task are you performing?

Score: 1 Mark in total (0.25 mark each)

----------Type the answers below this line----------

Business Understanding

What is the business problem that you are trying to solve?

Senior secondary school students from rural areas, inspite of having good learning capabilities fail to crack top competetive exams. Rural students clearing competitve exams and getting into premier institutes or government services is essential for the development of the country. A focussed coaching based on the student's competence and social background can help them succeed in competitive exams.

What data do you need to answer the above problem?

We need to get the data of rural student's Math and english comprehensive scores. Also we need a list of social elements of these students that can influence then in their competence

What are the different sources of data?
The math and english score can be obtained from the education department. Social parameters of those students must be gathered through organization which work in rural areas. For the specific problem we can get a sample of above data from different schools to formulate a coaching plan.

What kind of analytics task are you performing? All the students come from rural areas can have varied social background. We need to find cluster of students based on the social and competency parameters to focus specific training for each group of student. We will use **Partitioning Clustering** algorithms to solve this problem. Before proceeding to Machine Learning models we will prepare the data by cleaning it from noise, then perform Data wrangling. Once we have good data we will do EDA to visuvalize each feature, their spread and relationship of these features. Finally wee will find a suitable ML algorithm and implement.

2. Data Acquisition

For the problem identified , find an appropriate data set (Your data set must be unique) from any public data source.

2.1 Download the data directly

Data will be downloaded from the URL provided direcyly

In [81]:
```
##---------Type the code below this line-----------------##
import pandas as pd
import requests
```

```
# Set the URL of the Excel file on GitHub
url = 'https://github.com/vinothngit/datascience/raw/main/SecondaryEduCompetence.xlsx'

# Make a GET request to download the file
response = requests.get(url)

# Create a bytes object from the response content
file_bytes = response.content
```

2.2 Code for converting the above downloaded data into a dataframe

In [82]:
```
# Load the bytes object into a pandas dataframe
data = pd.read_excel(file_bytes)
```

2.3 Confirm the data has been correctly by displaying the first 5 and last 5 records.

In [83]:
```
##---------Type the code below this line-----------------##
data.head() ##Top 5 rows
```

Out[83]:

	Identifier	Gender	Ethinicity	Parent Education	lunch	Coaching	Numerical	Vocabulary	Comprehension
0	1001	female	group B	bachelor's degree	standard	none	72	72	74
1	1002	female	group C	some college	standard	Yes	69	90	88
2	1003	female	group B	master's degree	standard	none	90	95	93
3	1004	male	group A	associate's degree	free/reduced	none	47	57	44
4	1005	male	group C	some college	standard	none	76	78	75

In [84]:
```
data.tail() ##bottom 5 rows
```

Out[84]:

	Identifier	Gender	Ethinicity	Parent Education	lunch	Coaching	Numerical	Vocabulary	Comprehension
1005	2006	female	group E	master's degree	standard	Yes	88	99	95
1006	2007	male	group C	high school	free/reduced	none	62	55	55
1007	2008	female	group C	high school	free/reduced	Yes	59	71	65
1008	2009	female	group D	some college	standard	Yes	68	78	77
1009	2010	female	group D	some college	free/reduced	none	77	86	86

2.4 Display the column headings, statistical information, description and statistical summary of the data.

In [85]:

```
##---------Type the code below this line-----------------##
print(f"\nShape of the data set - {data.shape}") ## size of the dataset

cols = data.columns # All columns of the dataframe

num_cols = data._get_numeric_data().columns

print(f"\nNumerical Columns - {num_cols}")

print(f"\nCategorical Columns - {list(set(cols) - set(num_cols))}")

print(f"\nALL the Columns \n - {data.columns}") # All columns of the data frame

print(f"\nSTATS OF ALL NUMBERICAL COLUMNS \n\n{data.describe()}") ##Statistical Information of the numerical columns
```

```
Shape of the data set - (1010, 9)

Numerical Columns - Index(['Identifier', 'Numerical', 'Vocabulary', 'Comprehension'], dtype='object')

Categorical Columns - ['Parent Education', 'Ethinicity', 'Coaching', 'Gender', 'lunch']

ALL the Columns
 - Index(['Identifier', 'Gender', 'Ethinicity', 'Parent Education', 'lunch',
       'Coaching', 'Numerical', 'Vocabulary', 'Comprehension'],
      dtype='object')

STATS OF ALL NUMBERICAL COLUMNS

        Identifier    Numerical   Vocabulary  Comprehension
count  1010.000000  1010.000000  1010.000000    1010.000000
mean   1505.500000    65.997030    69.042574      67.953465
std     291.706188    15.171206    14.665587      15.258552
min    1001.000000     0.000000    17.000000      10.000000
25%    1253.250000    56.000000    59.000000      57.000000
50%    1505.500000    66.000000    70.000000      69.000000
75%    1757.750000    77.000000    79.000000      79.000000
max    2010.000000   100.000000   100.000000     100.000000
```

2.5 Write your observations from the above.

1. Size of the dataset The dataset has record 1010 students with 9 features.
2. What type of data attributes are there? We have categorical and numerical data
3. Is there any null data that has to be cleaned? Yes

Score: 2 Marks in total (0.25 marks for 2.1, 0.25 marks for 2.2, 0.5 marks for 2.3, 0.25 marks for 2.4, 0.75 marks for 2.5)

----------Type the answers below this line----------

1. Size of the dataset 9 columns (features) and 1010 rows

2. What type of data attributes are there? - **Categorical** (Disrete) : Identifier, Gender, Ethinicity, Lunch, Coaching, Parent Education
   Numerical : Vocabulary, Comprehension, Numberical competency scores

3 Is there any null data that has to be cleaned? **Yes**. There are NULL data in the data set which must be cleaned

Description of features

1) Identifier : Unique ID represeenting a single student

2) Gender : Specifies gender of the student (male or female)

3) Ethinicity : Specifies ethinic background of the student (group A,group B,group C. etc)

4) Parent Education : Specifies highest educational qualification of the parent of each student

5) lunch : The amount of lunch consumption or availability for a student as perr social record (standard or reduced)

6) Coaching : Availability and Accesibilityy to Coaching for that student

7) Numerical : Score denoting the Mathematical competency of a student (Out ot 100)

8) Vocabulary : Score denoting the Reading competency of a student (Out ot 100)

9) Comprehension : Score denoting the Writing competency of a student (Out ot 100)

3. Data Preparation

If input data is numerical or categorical, do 3.1, 3.2 and 3.4 If input data is text, do 3.3 and 3.4

3.1 Check for

- duplicate data
- missing data
- data inconsistencies

In [86]:
```
##---------Type the code below this line-----------------##
duplicate = data[data.duplicated()]

print("Total Duplicate Rows :")
print(duplicate)

print("\nNull values in each column :")
```

```
print(data.isnull().sum()) ## Number of Null values in each column

## Print only unique columns - eeventually check ffor any duplicate columns
print("\nUnique columns :")
data.columns.unique()
```

```
Total Duplicate Rows :
Empty DataFrame
Columns: [Identifier, Gender, Ethinicity, Parent Education, lunch, Coaching, Numerical, Vocabulary, Comprehension]
Index: []

Null values in each column :
Identifier          0
Gender              0
Ethinicity          5
Parent Education    9
lunch               5
Coaching            5
Numerical           0
Vocabulary          0
Comprehension       0
dtype: int64

Unique columns :
```

Out[86]:

```
Index(['Identifier', 'Gender', 'Ethinicity', 'Parent Education', 'lunch',
       'Coaching', 'Numerical', 'Vocabulary', 'Comprehension'],
      dtype='object')
```

3.2 Apply techiniques

- to remove duplicate data
- to impute or remove missing data
- to remove data inconsistencies

Removing duplicate data and dropping null data

- There are NO DUPLICATE data
- NULL values are found in various columns which are handled below

In [87]:

```
##---------Type the code below this line-----------------##
## Duplicate Rows
# There are no duplicate rows as each row belong to a unique student. The data does not have any typo erros
```

```
# removing the rows with NULL values. The missing values are specific to a particular studennt
# Also the missing fields are less than 1% off total rows. So we can safeely remove then from dataset

data = data.dropna()
print("\n DataFrame after removing null values...\n",data)
print("\n (Updated) Number of rows and column in our DataFrame = ",data.shape)


datacopy = data.copy()
# Preparing specifc data for computation
# Preapre total percentage and Grades off students
```

```
 DataFrame after removing null values...
      Identifier  Gender Ethinicity    Parent Education         lunch  \
0           1001  female    group B   bachelor's degree      standard
1           1002  female    group C        some college      standard
2           1003  female    group B     master's degree      standard
3           1004    male    group A  associate's degree  free/reduced
4           1005    male    group C        some college      standard
...          ...     ...        ...                 ...           ...
1005        2006  female    group E     master's degree      standard
1006        2007    male    group C         high school  free/reduced
1007        2008  female    group C         high school  free/reduced
1008        2009  female    group D        some college      standard
1009        2010  female    group D        some college  free/reduced

     Coaching  Numerical  Vocabulary  Comprehension
0        none         72          72             74
1         Yes         69          90             88
2        none         90          95             93
3        none         47          57             44
4        none         76          78             75
...       ...        ...         ...            ...
1005      Yes         88          99             95
1006     none         62          55             55
1007      Yes         59          71             65
1008      Yes         68          78             77
1009     none         77          86             86

[1000 rows x 9 columns]

 (Updated) Number of rows and column in our DataFrame =  (1000, 9)
```

Shape after removing NULL data (1000, 9) Now, Check for inconsistent values in each column/feature

In [88]:
```
#Let us check for unique values for each feature. Replace or correct any feature value
data['Parent Education'].unique()
```

Out[88]:
```
array(["bachelor's degree", 'some college', "master's degree",
       "associate's degree", 'high school', 'some high school'],
      dtype=object)
```

In [89]:
```
data['Parent Education']=data['Parent Education'].replace(['some high school'],'high school')
data['Parent Education'].unique()
```

Out[89]:
```
array(["bachelor's degree", 'some college', "master's degree",
       "associate's degree", 'high school'], dtype=object)
```

In [90]:
```
data['lunch']=data['lunch'].replace(['free/reduced'],'reduced')
data['lunch'].unique()
```

Out[90]:
```
array(['standard', 'reduced'], dtype=object)
```

In [91]:
```
data['Ethinicity'].unique()
```

Out[91]:
```
array(['group B', 'group C', 'group A', 'group D', 'group E'],
      dtype=object)
```

In [92]:
```
data['Coaching'].unique()
```

Out[92]:
```
array(['none', 'Yes'], dtype=object)
```

Clean data after removing NULL and inconsistencies

In [94]:
```
data.head()
```

Out[94]:

	Identifier	Gender	Ethinicity	Parent Education	lunch	Coaching	Numerical	Vocabulary	Comprehension
0	1001	female	group B	bachelor's degree	standard	none	72	72	74
1	1002	female	group C	some college	standard	Yes	69	90	88
2	1003	female	group B	master's degree	standard	none	90	95	93
3	1004	male	group A	associate's degree	reduced	none	47	57	44
4	1005	male	group C	some college	standard	none	76	78	75

3.3 Encode categorical data

- **One hot encoding**
- Many features have binary values so instead of using One hot encoding and increasing number of columns,
- let us **replace those values with numbers**
- One hot encoding increases the number of features.
- For Gender, Coaching, Lunch can have binary value instead of multiple columns

In [96]:
```
# Encode categorical data

newdata = data.copy()
numeric_var = {'Gender': {'male':0, 'female':1}}
newdata = newdata.replace(numeric_var)
numeric_var = {'Coaching': {'none':0, 'Yes':1}}
newdata = newdata.replace(numeric_var)
numeric_var = {'lunch': {'reduced':0, 'standard':1}}
newdata = newdata.replace(numeric_var)

newdata.head()
```

Out[96]:

	Identifier	Gender	Ethinicity	Parent Education	lunch	Coaching	Numerical	Vocabulary	Comprehension
0	1001	1	group B	bachelor's degree	1	0	72	72	74
1	1002	1	group C	some college	1	1	69	90	88
2	1003	1	group B	master's degree	1	0	90	95	93
3	1004	0	group A	associate's degree	0	0	47	57	44
4	1005	0	group C	some college	1	0	76	78	75

In [97]:
```
one_hot_encoded_data = pd.get_dummies(newdata, columns = ['Ethinicity', 'Parent Education'])
#print(one_hot_encoded_data)
one_hot_encoded_data.head()
print(one_hot_encoded_data.shape)
```

```
(1000, 17)
```

Encoded Data

In [98]:
```
##---------Type the code below this line-----------------##
one_hot_encoded_data.head()
```

Out[98]:

	Identifier	Gender	lunch	Coaching	Numerical	Vocabulary	Comprehension	Ethinicity_group A	Ethinicity_group B	Ethinicity_group C	Ethinicity
0	1001	1	1	0	72	72	74	0	1	0	
1	1002	1	1	1	69	90	88	0	0	1	
2	1003	1	1	0	90	95	93	0	1	0	
3	1004	0	0	0	47	57	44	1	0	0	
4	1005	0	1	0	76	78	75	0	0	1	

Original Data

In [100…
```
# Comparing with initial data
data.head()
```

Out[100]:

	Identifier	Gender	Ethinicity	Parent Education	lunch	Coaching	Numerical	Vocabulary	Comprehension
0	1001	female	group B	bachelor's degree	standard	none	72	72	74
1	1002	female	group C	some college	standard	Yes	69	90	88
2	1003	female	group B	master's degree	standard	none	90	95	93
3	1004	male	group A	associate's degree	reduced	none	47	57	44
4	1005	male	group C	some college	standard	none	76	78	75

3.4 Text data

1. Remove special characters
2. Change the case (up-casing and down-casing).
3. Tokenization — process of discretizing words within a document.
4. Filter Stop Words.

---------Type the code below this line----------

There are NO text data in this data set

In [482…

```
##---------Type the code below this line-----------------##
```

3.4 Report

Mention and justify the method adopted

- to remove duplicate data, if present
- to impute or remove missing data, if present
- to remove data inconsistencies, if present

OR for textdata

- How many tokens after step 3?
- how may tokens after stop words filtering?

If the any of the above are not present, then also add in the report below.

Score: 2 Marks (based on the dataset you have, the data prepreation you had to do and report typed, marks will be distributed between 3.1, 3.2, 3.3 and 3.4)

---------Type the code below this line----------

REPORT

Duplicate data: The data consists of unique student details. So there are no duplicate rows in the data set. There are no typographical errors too

Missing values: There were 10 rows which has missing data like Parent education, lunch OR Ethinicity. Theese are categorical data which cannot be imputed. Also as the missing values are less than 1% of total data we can safely removeethe rows which has missing data.

Inconsistent values: Inconsistent values representing the same class are merged or replaced togeether Parent Education - has "Some high school" and "high school" as values. "Some high school" is replaced with "high school" to brring more consisteency lunch - "free/reduced" class is removed and is replaced with "reduced".

Encoding: The Categorical features were encoded. One hot encoding used for "Ethinicity" and "Parent Education" For features with only two values the values were replaced woth 0 or 1 instead of one hot encoding. This was done so that the **dimension does not increase further** Features Gender, lunch and Coaching were replaced with binary values

In [483…

```
##---------Type the code below this line-----------------##
```

3.5 Identify the target variables.

- Separate the data from the target such that the dataset is in the form of (X,y) or (Features, Label)
- Discretize / Encode the target variable or perform one-hot encoding on the target or any other as and if required.
- Report the observations

Score: 1 Mark

---------Type the code below this line----------

Target variables

Clustering: We will use complete data for clustering and remove features which are not seelected as a part of feature selection activity. There are **no specific Y label as our objective is to cluster** the students based on all these parameters Of course after clusterring we can have one of the dropped features as Y to compare with cluster Y labels to create metrics such a F1-score, Precision and Recall. But unsupervised algorithms have Sillotte score which can be useed for comparision of performance **There is NO TARGET VARIABLE in the data set**

4. Data Exploration using various plots

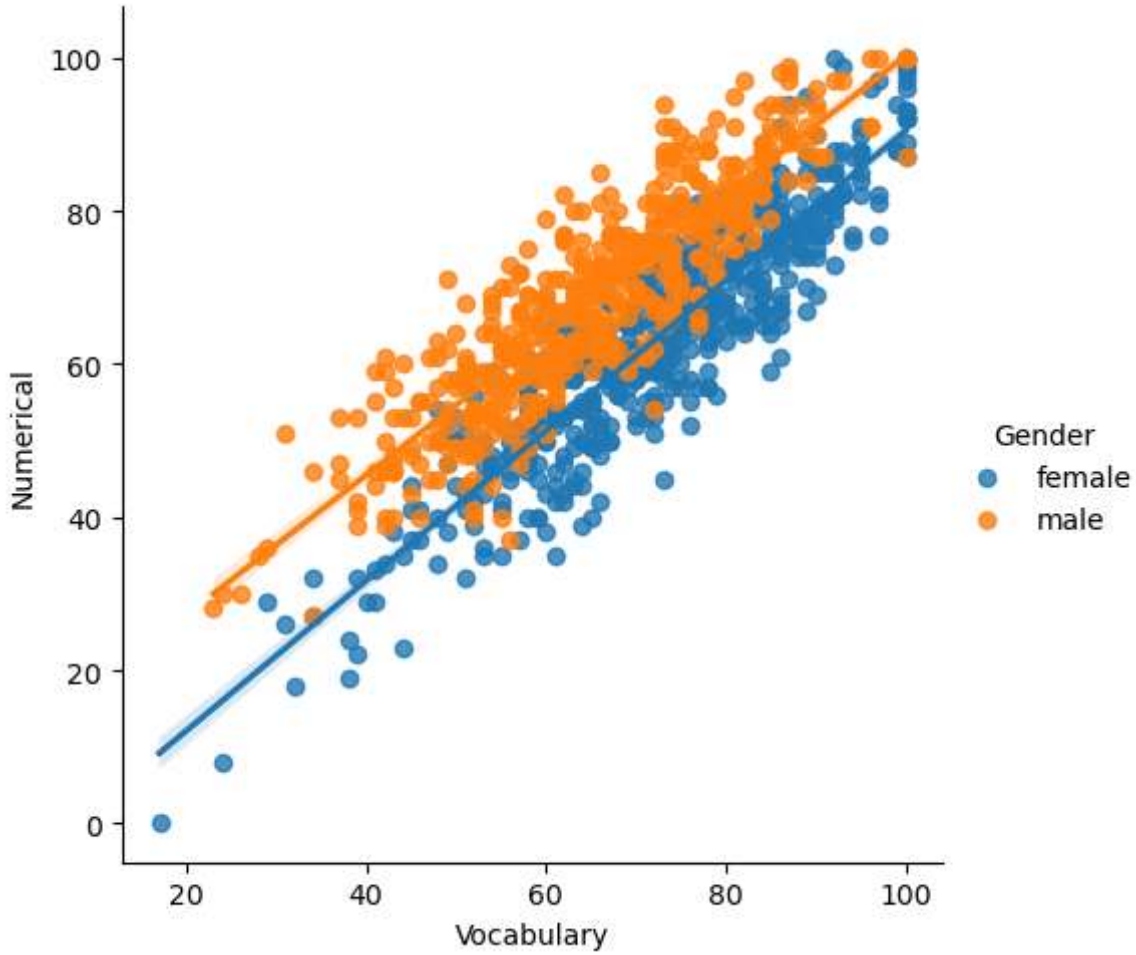
4.1 Scatter plot of each quantitative attribute with the target.

Score: 1 Mark

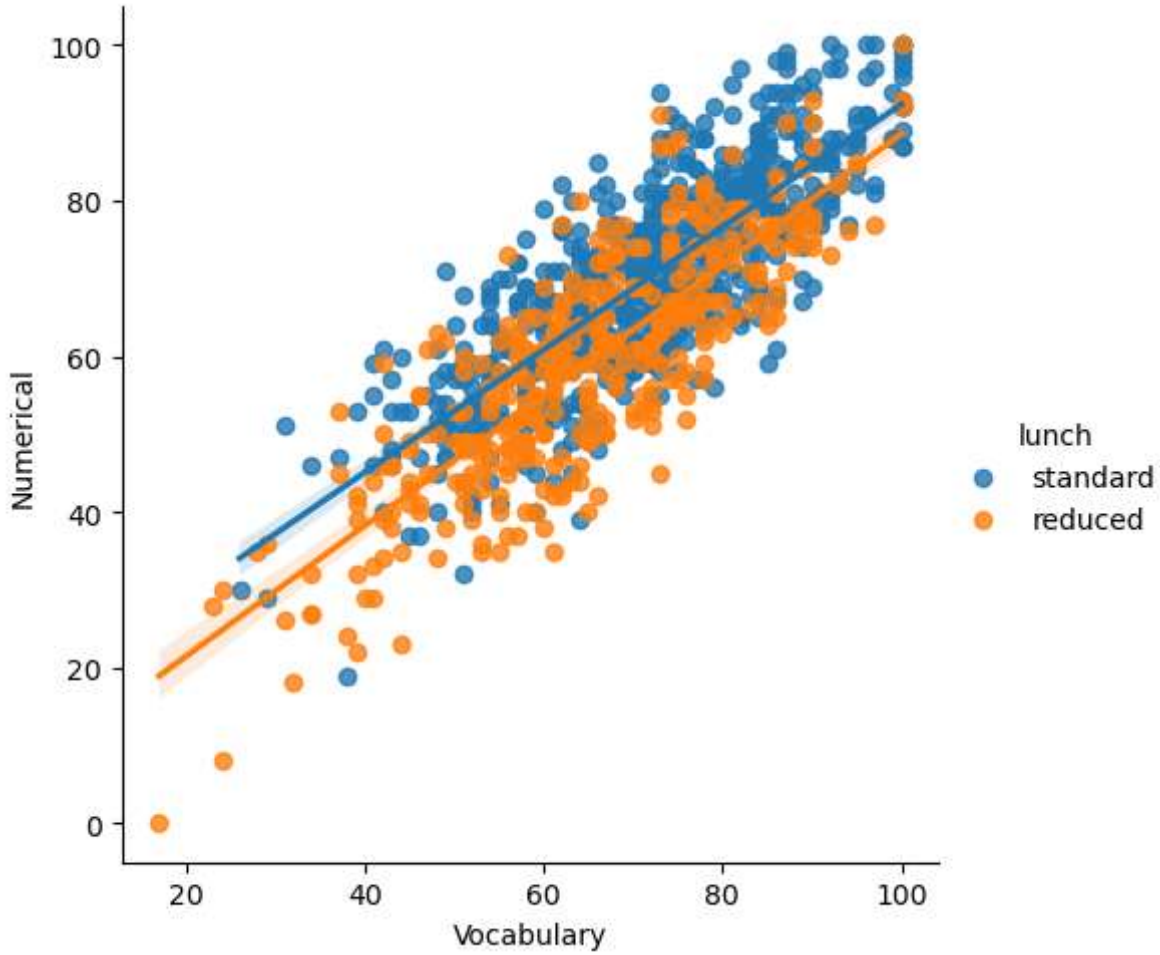
In [101…

```
##---------Type the code below this line-----------------##

import matplotlib.pyplot as plt
import seaborn as sns

hue_list=['Gender','lunch','Coaching', 'Parent Education', 'Ethinicity']
for hue in hue_list:
    plt.figure(figsize=(12,5))
    sns.lmplot(x='Vocabulary',y='Numerical',data=data,hue=hue,fit_reg=True)
#plt.show()
```

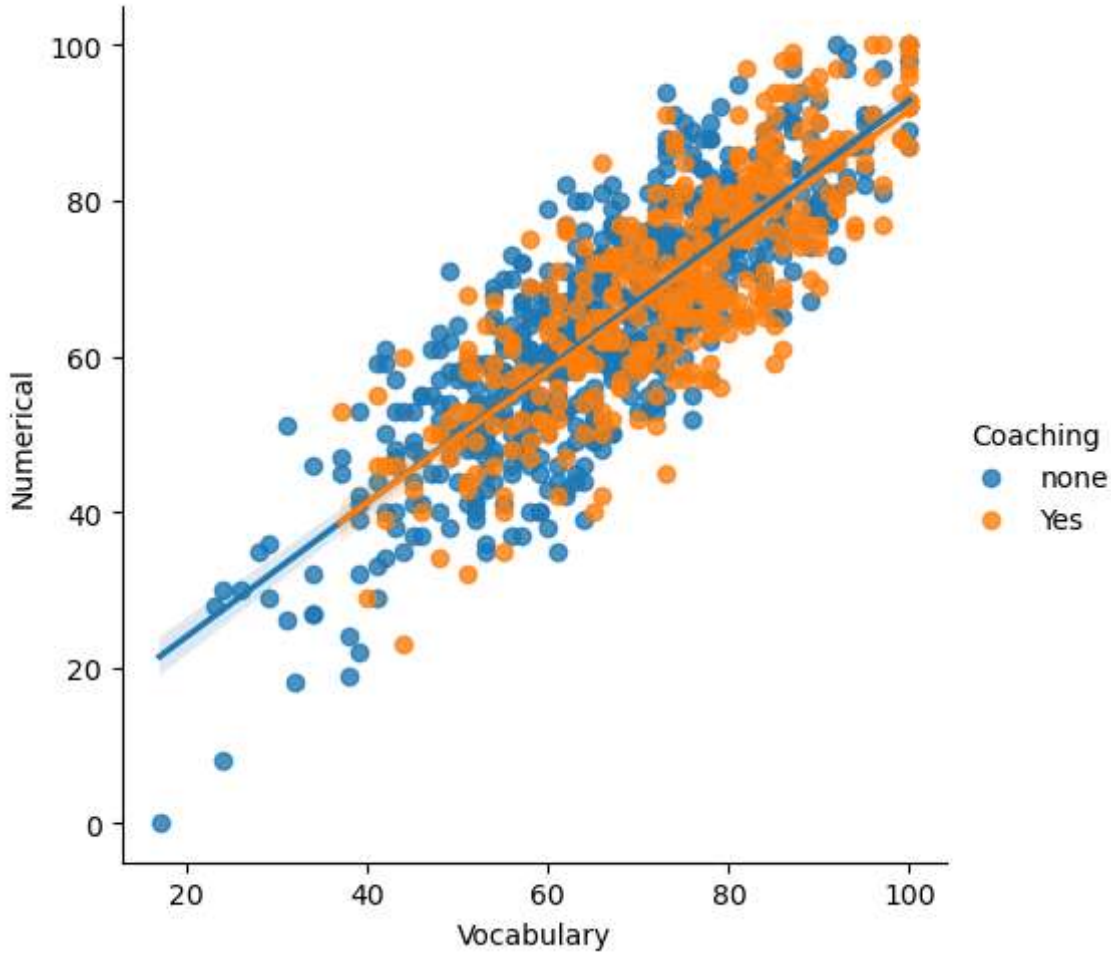
```
<Figure size 1200x500 with 0 Axes>
```



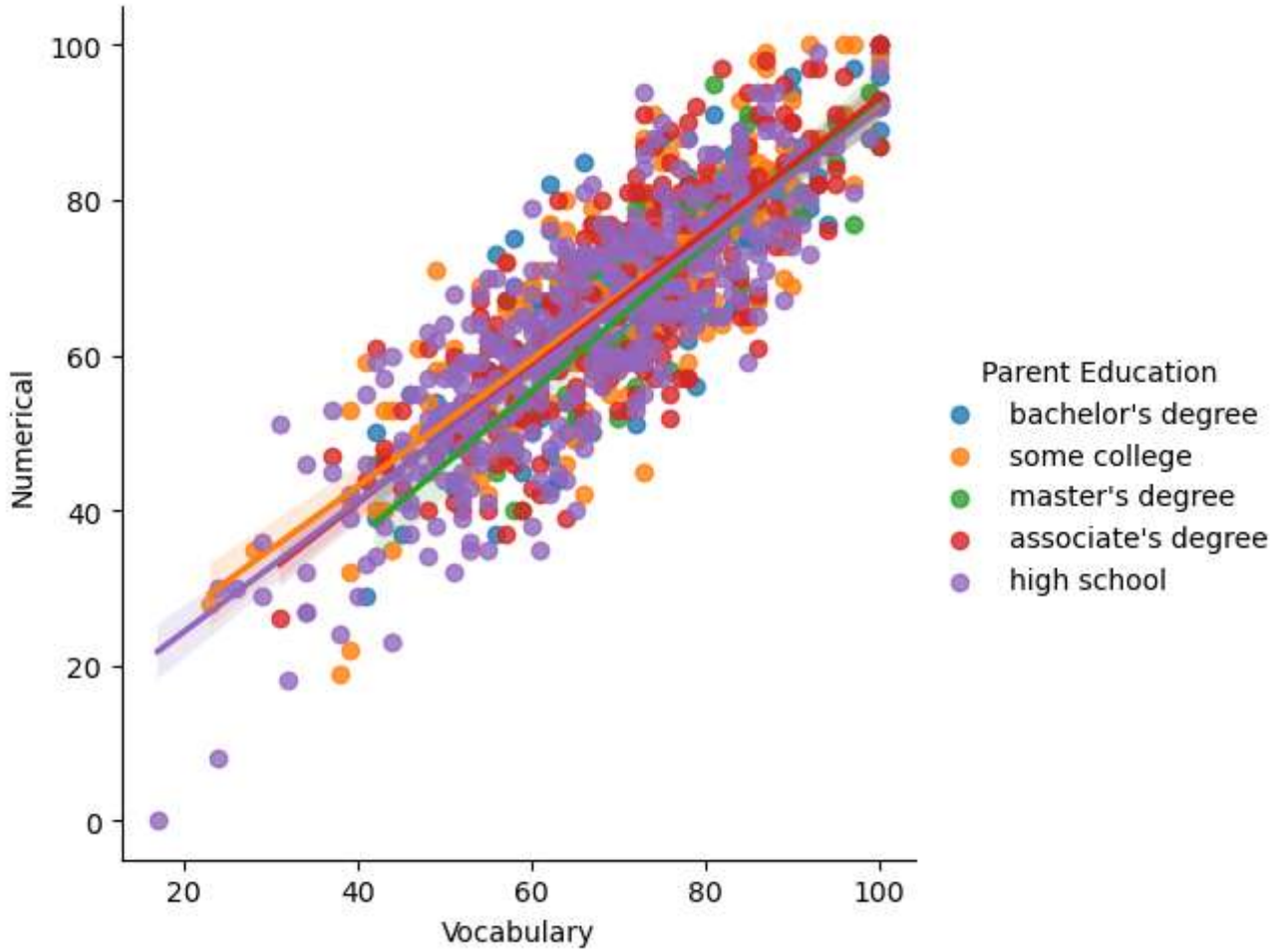
<Figure size 1200x500 with 0 Axes>



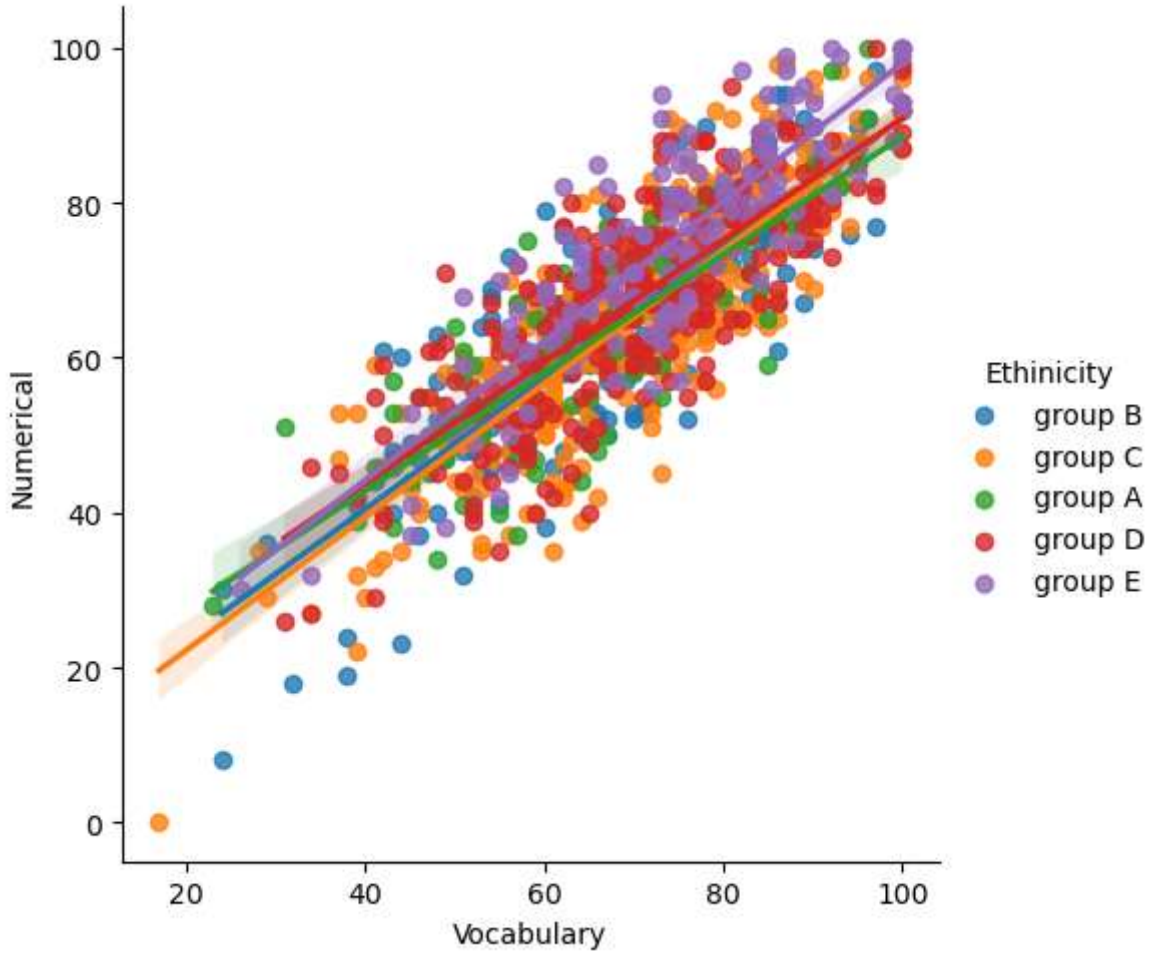
<Figure size 1200x500 with 0 Axes>



<Figure size 1200x500 with 0 Axes>



<Figure size 1200x500 with 0 Axes>



Observation on Scatter plots

The scatter plots on Numerical, Vocabulaty and Comprehensive scores suggest that all three features have a positive co-relation.
Gender: Male students seem to score better in Numerical and Vocabulary than Female students

Lunch: Students who have access to a standard lunch perform well. But it is not very signifficant. There are students who have access to only reduced lunch has also has scored well. But the higher marks are mostly in the standard lunch group.

Coaching : No Accesibility to coaching class does not stop the students from performing well. But a majority off the low scoring student do not have coaching access

Parent Education: High number of students have parent who have only high school education. While these student are in the lower most scoring category they are also spread across the scoring range. Student's who's parents have master degree are also close to the passing range of 40 which shows parent education is not playing major factor.

Ethinicity : Group E seems to be performing much better than all the other groups. Other ethinic groups are pretty much doing similarly other than a few outliers. Incidentally most off these low performing outliers are from Ethinic group B

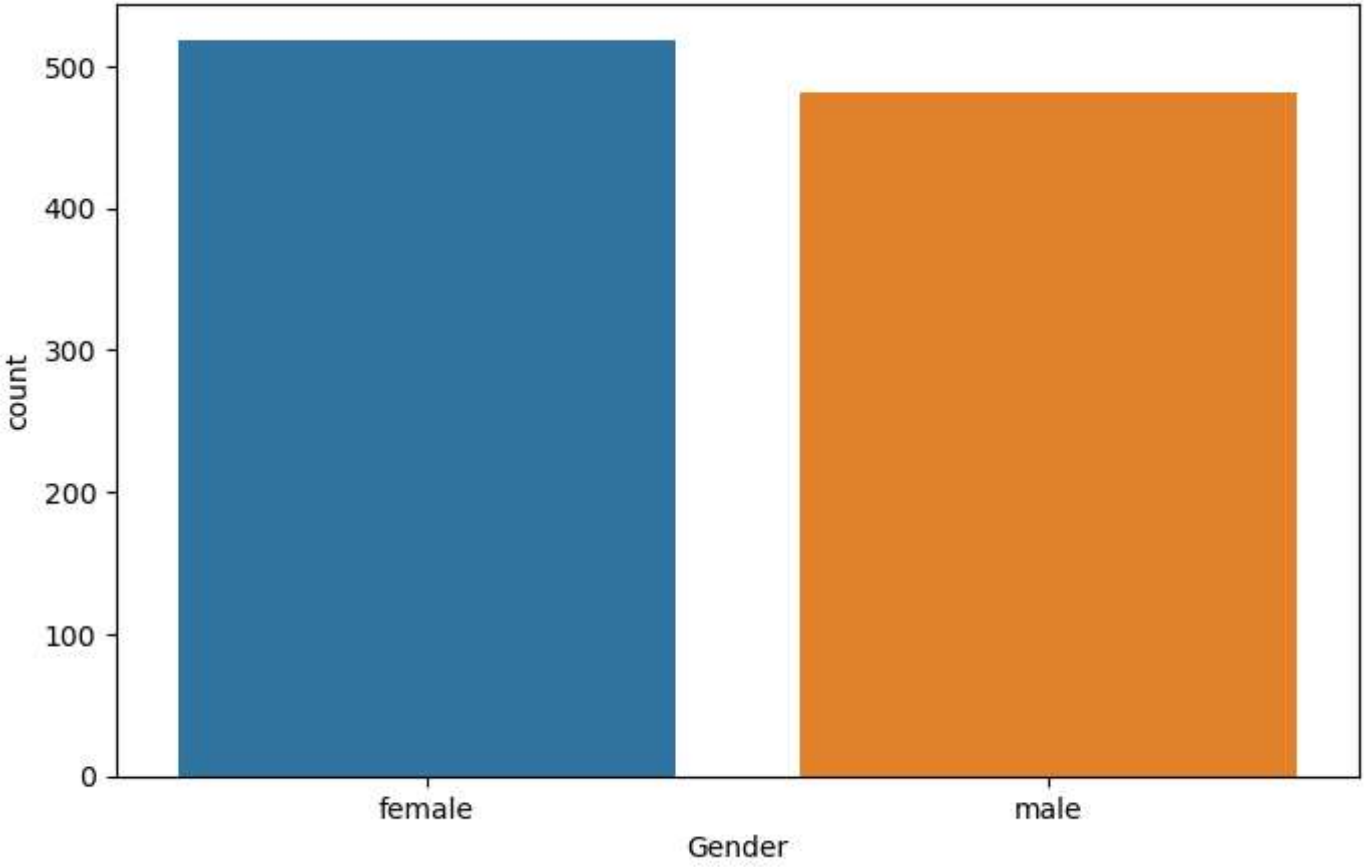
4.2 EDA using visuals

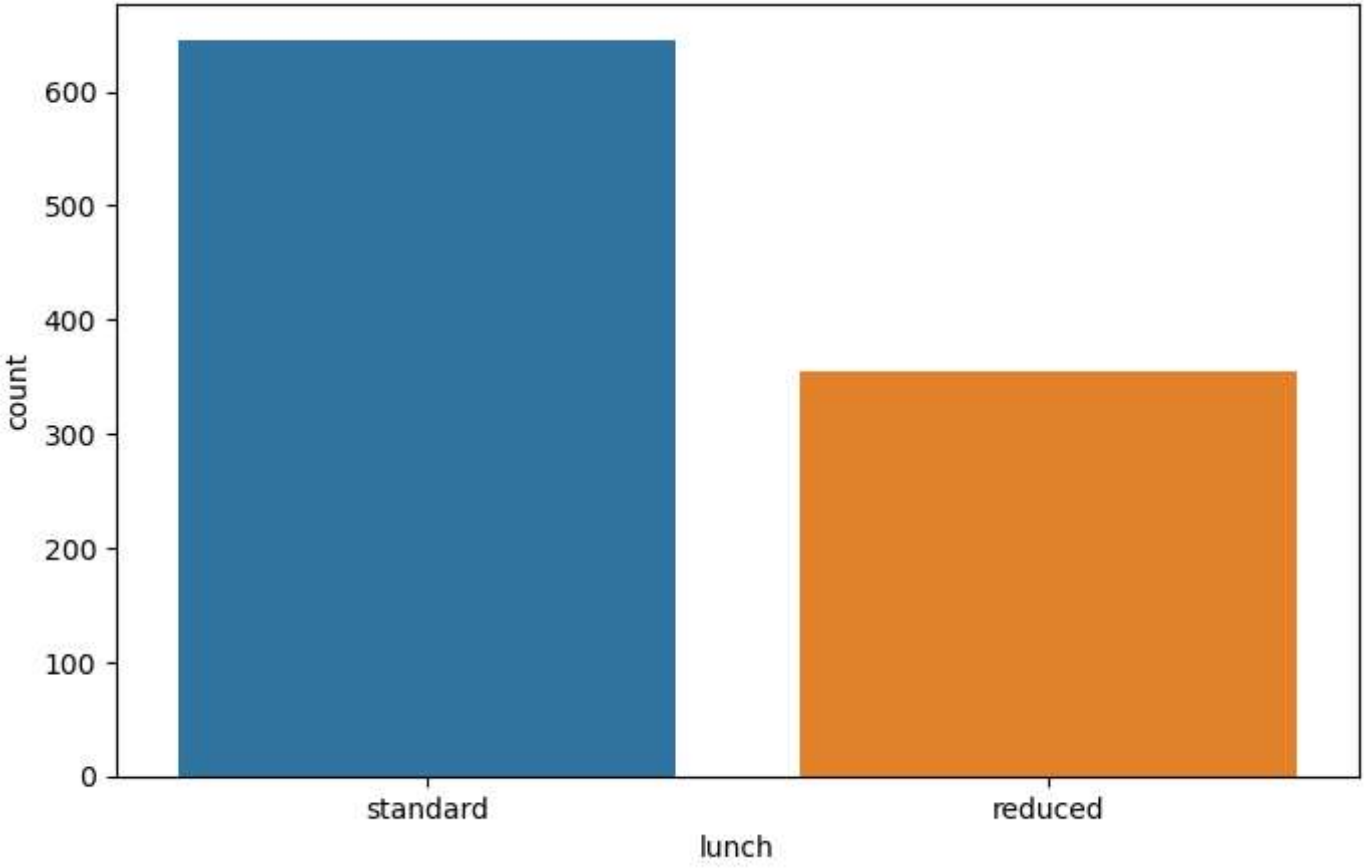
- Use (minimum) 2 plots (pair plot, heat map, correlation plot, regression plot...) to identify the optimal set of attributes that can be used for classification.
- Name them, explain why you think they can be helpful in the task and perform the plot as well. Unless proper justification for the choice of plots given, no credit will be awarded.

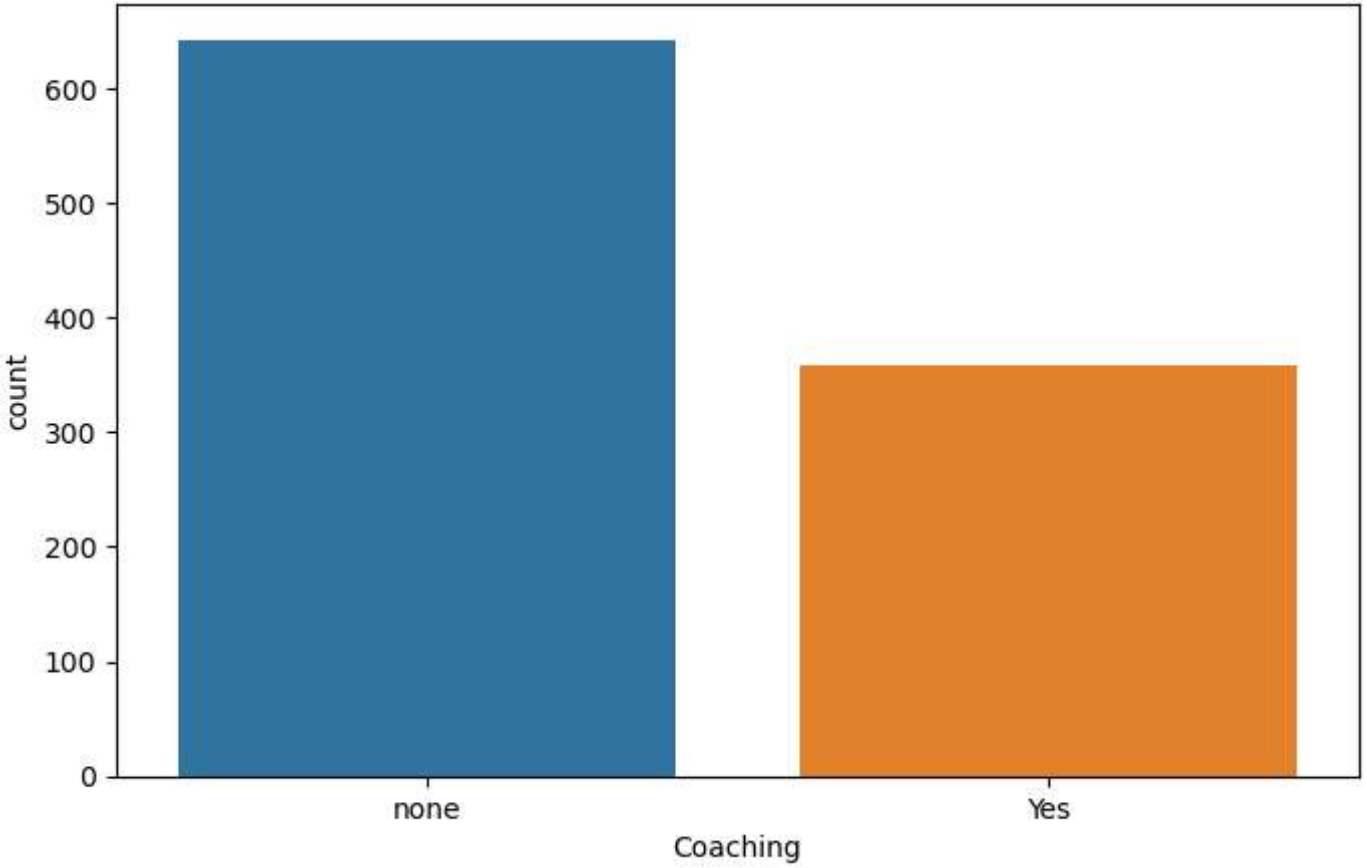
Score: 2 Marks

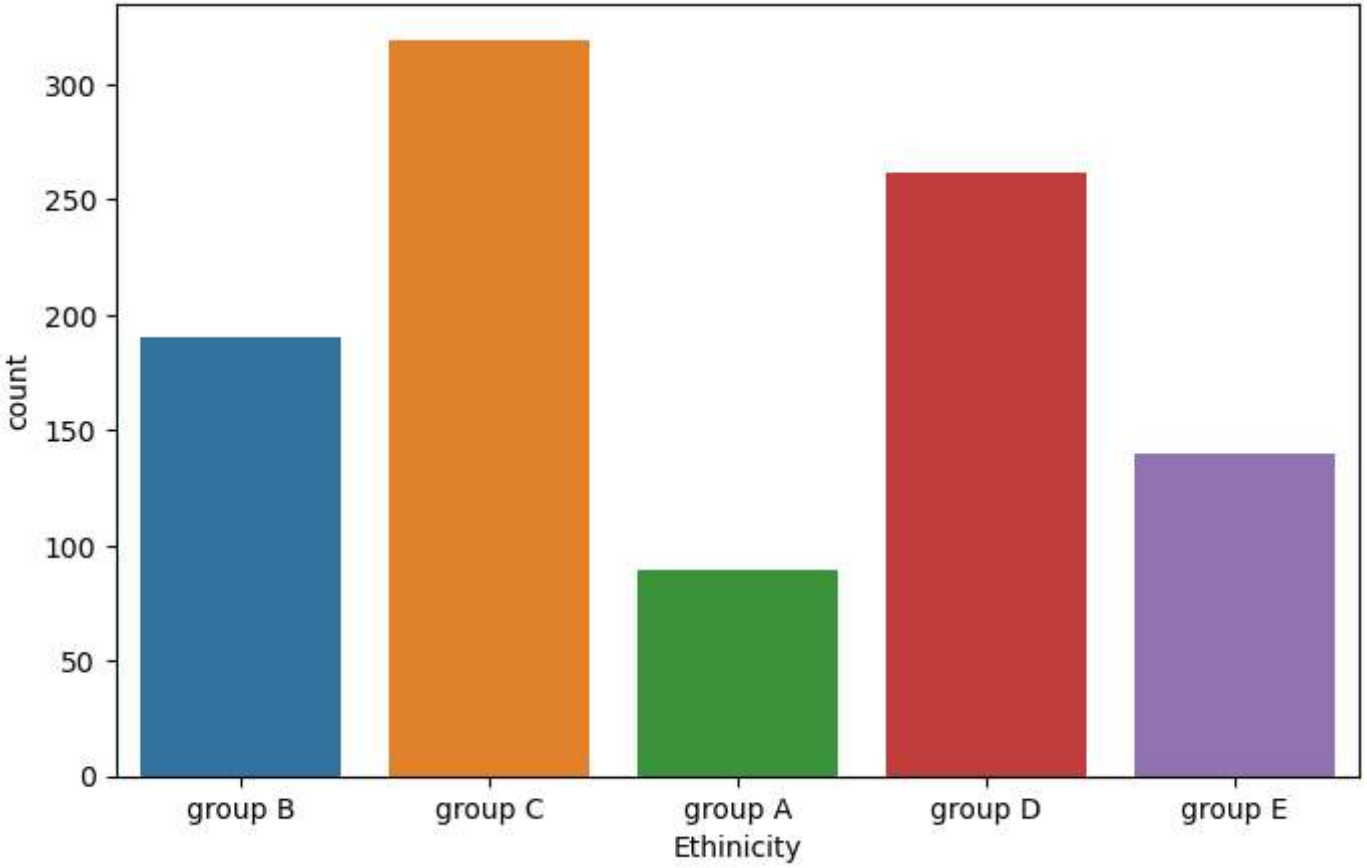
In [102…

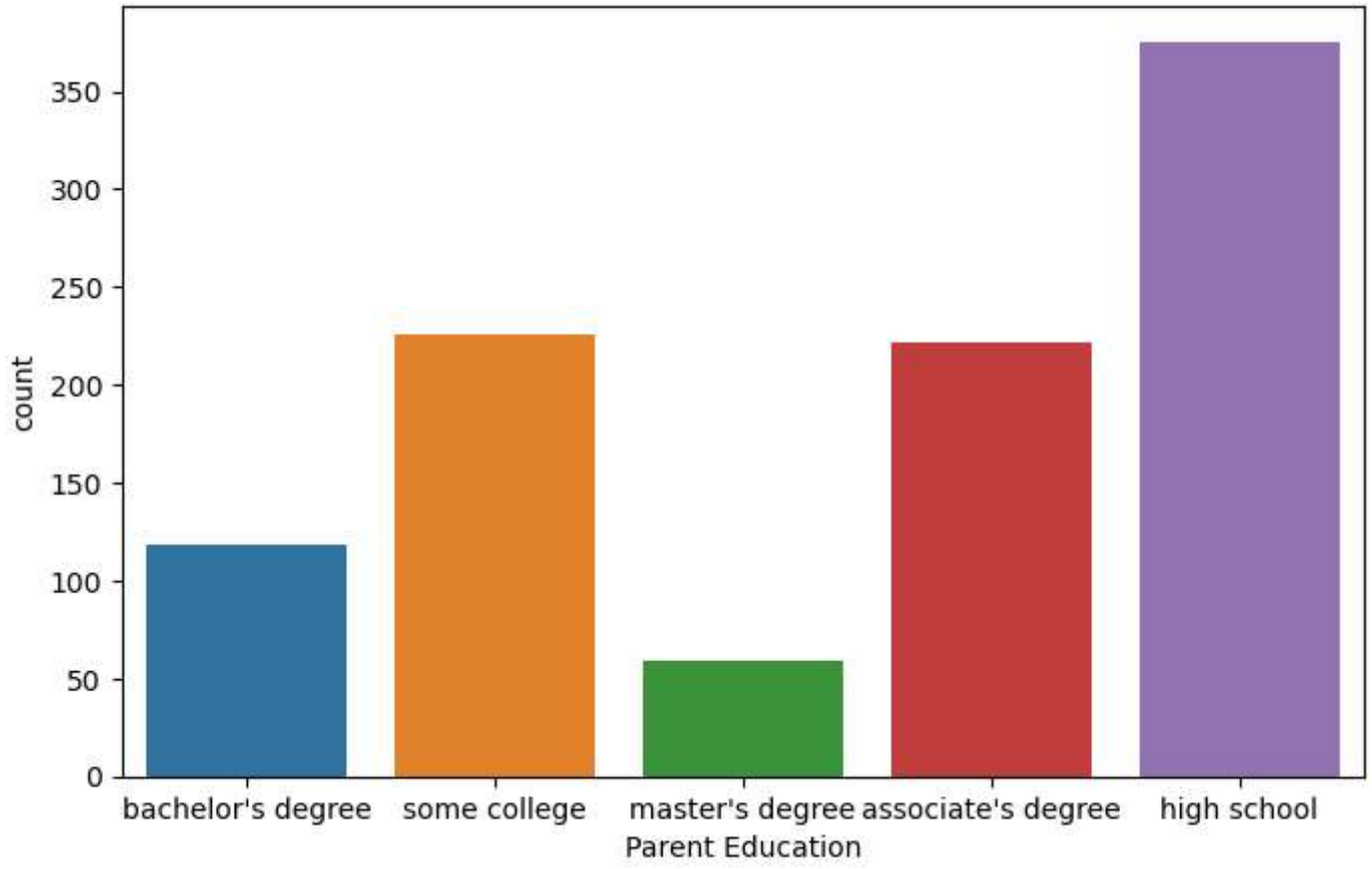
```
import matplotlib.pyplot as plt
import seaborn as sns

# Lets do bar graph for all categorical data to see the count
Col_list=['Gender','lunch','Coaching', 'Ethinicity', 'Parent Education']
for col in Col_list:
    plt.figure(figsize=(8,5))
    sns.countplot(x=col,data=data)
```


500
400
300
200
100
0
count
female
male
Gender


600
500
400
300
200
100
0
count
standard
reduced
lunch


600
500
400
300
200
100
0
count
none
Yes
Coaching


300
250
200
150
100
50
0
count
group B
group C
group A
group D
group E
Ethinicity



Observation on Bar graphs

The bar graphs are very self explanatory. EDA is performed on all the social categories to understand the count (numberr of students) in each category. Gender: Female students are higher than the male students in this sample. The data set must have been derived after Stratified sampling to keep the Gender population propotional to the actuals

Lunch: More than 50% of the students get only reduced lunch.

Coaching: Only half of the student population have access to coaching

Ethinicity: Group C are higher in the population. Group A are loweer in the population. While Group E are the second lowest in terms of population, we inferred from scatter plaots that Ethinic group E tops in the scoring.

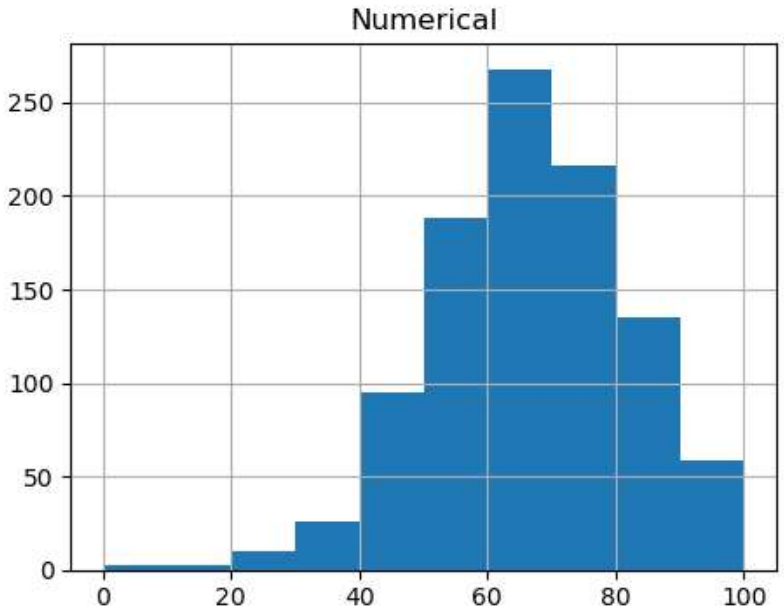
Parent Education: Almost 33% of the student's parent has only high school education. We confirmed this in the scatterr plots as well.

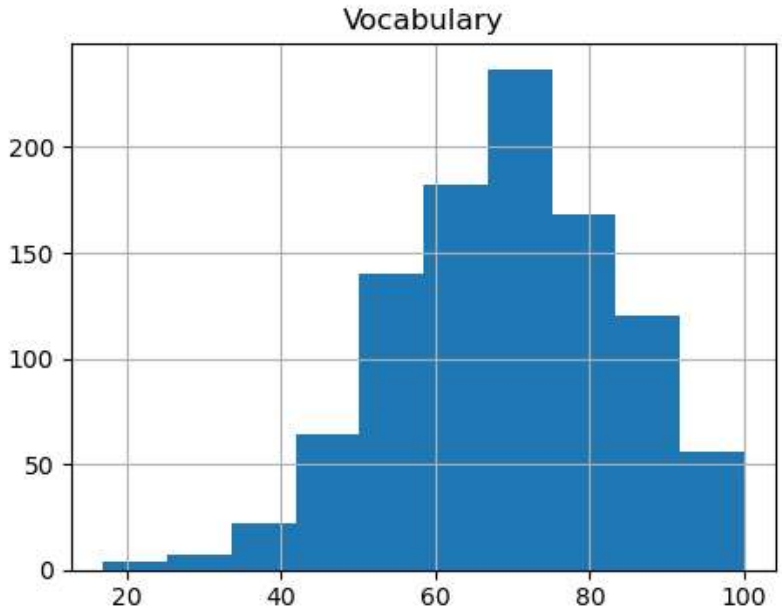
In [103…

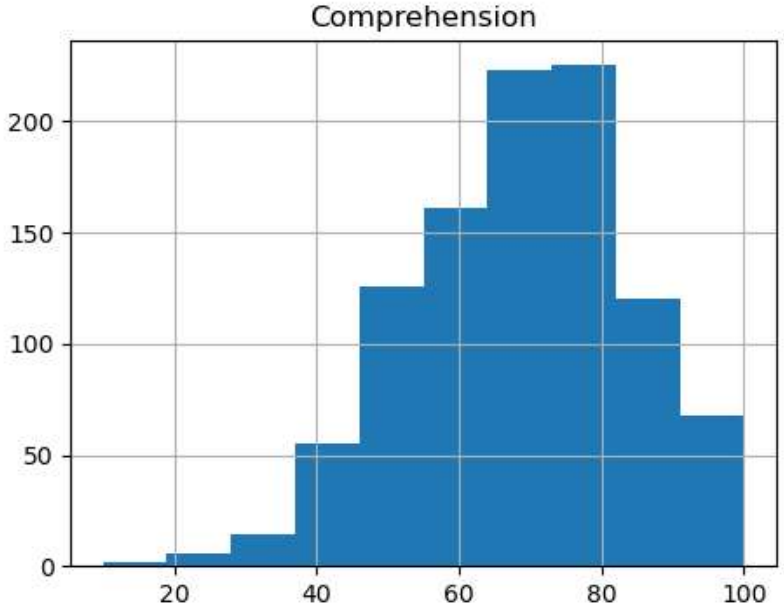
```
## Distrubution of the scores
## Let us plot one score to observe the distribution.

data[['Numerical','Vocabulary','Comprehension']].hist(figsize=(12,9))
```

Out[103]:

```
array([[<AxesSubplot:title={'center':'Numerical'}>,
        <AxesSubplot:title={'center':'Vocabulary'}>],
       [<AxesSubplot:title={'center':'Comprehension'}>, <AxesSubplot:>]],
      dtype=object)
```


Numerical
250
200
150
100
50
0
0
20
40
60
80
100


Vocabulary
200
150
100
50
0
20
40
60
80
100


Comprehension
200
150
100
50
0
20
40
60
80
100

In [ ]:

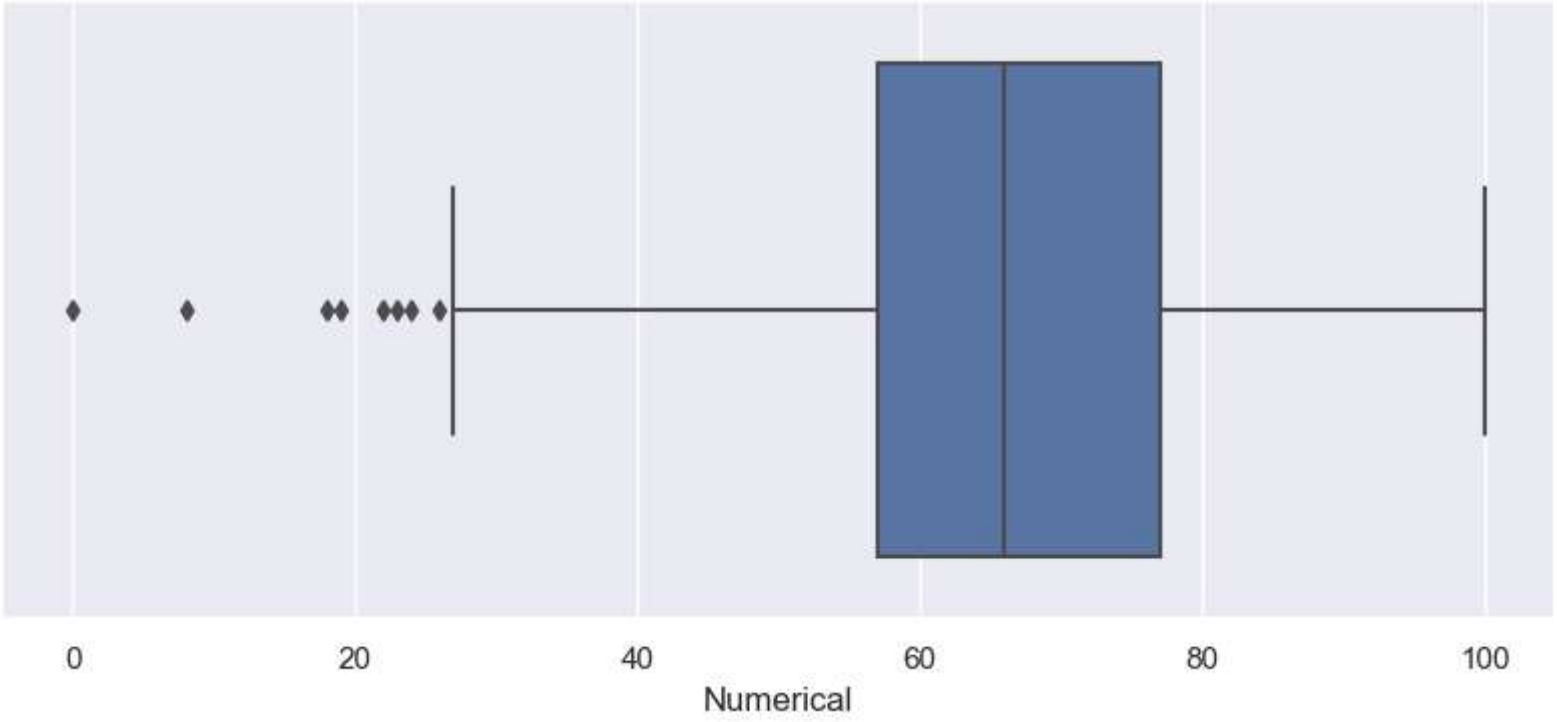
Observation on distrubution of scores

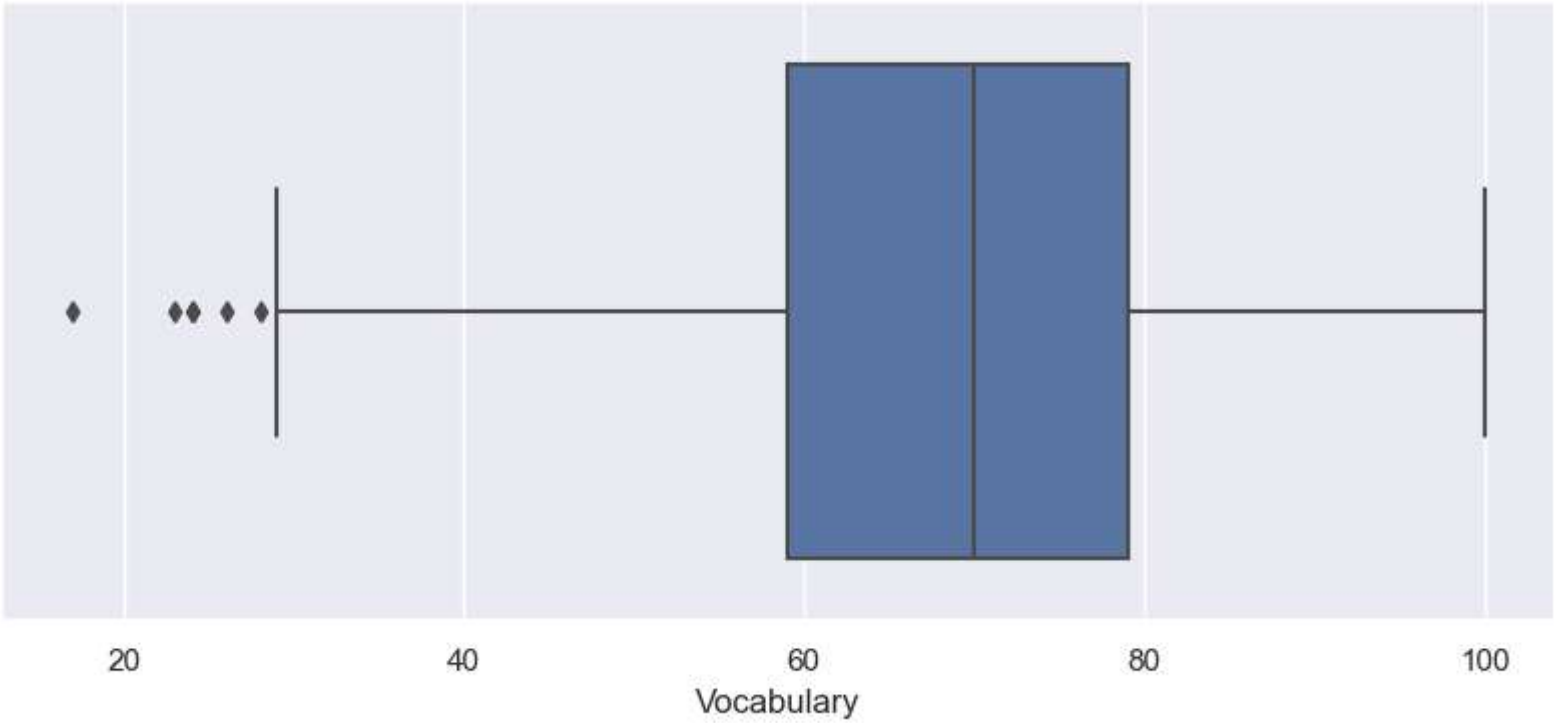
All the distrubution are left-Skewed. Inferring from the scatter plots the observation of one score must be same for all scores. Especially Vocabulary and Comprehension.

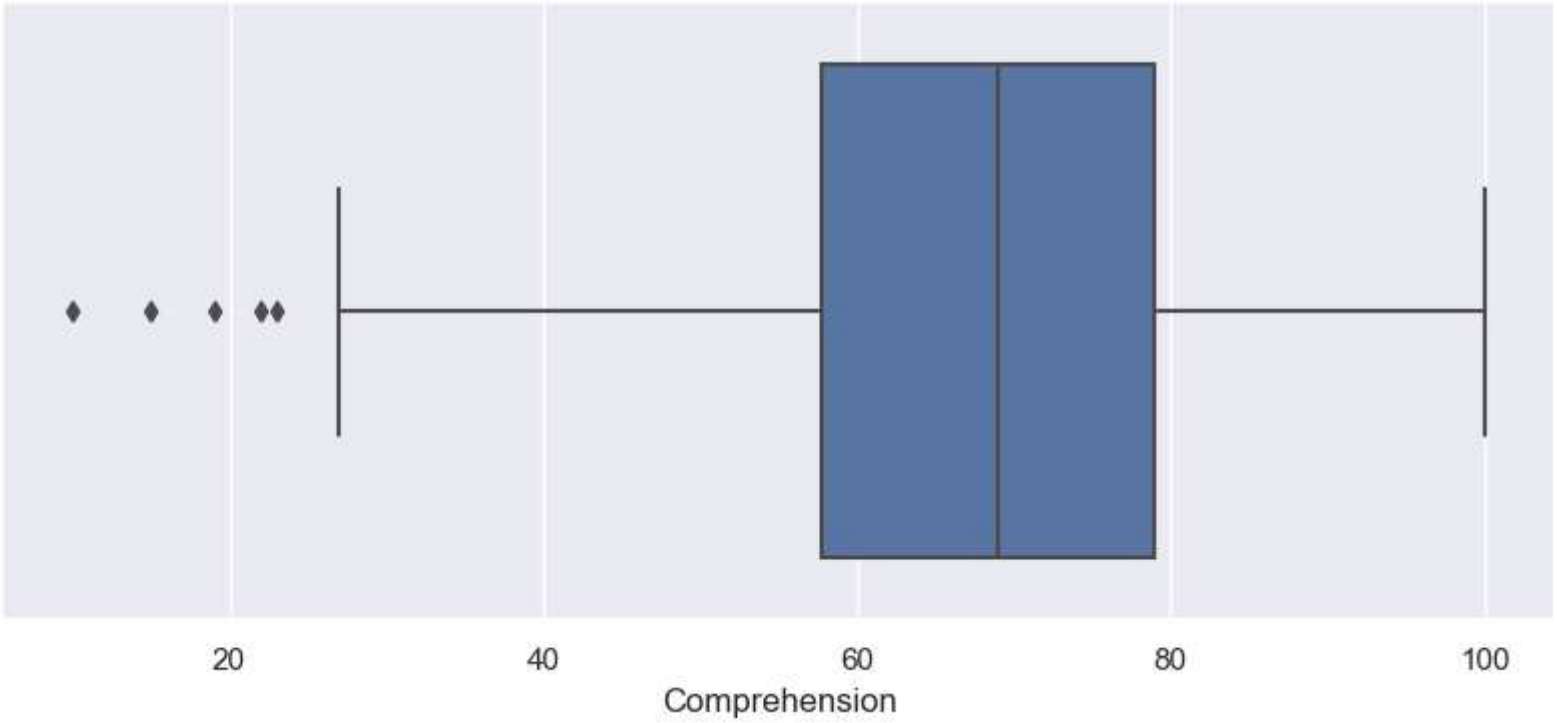
Vocabulary seems to have more students with less than pass score of 40. 60 - 70 seems to be the mean score which was conffirmed by our statistical output as well

In [418…

```
#function to create a boxplot
def boxplot(column,dataf):
    plt.figure(figsize=(10,4))
    sns.boxplot(x=column,data=dataf)

nums=['Numerical','Vocabulary','Comprehension']
for i in nums:
    boxplot(i,data)
```




20
40
60
80
100
Vocabulary



In [ ]:

Observation on box plot

The above box plot clearly shows the distribution of scores. The box is inline with the statistics which we figured on the data set Outliers: Out of the 1000 data points only 5-7 outliers exists in all plots Distribution: The box is not centerered between the whiskers. The distribution is skewed. For Vocabulary and Comprehension the distribution, mean and the 50% (box) all seems be same. The range of Nummerical is smaller than Vocabulary or Comprehension, but the 50% is almost close

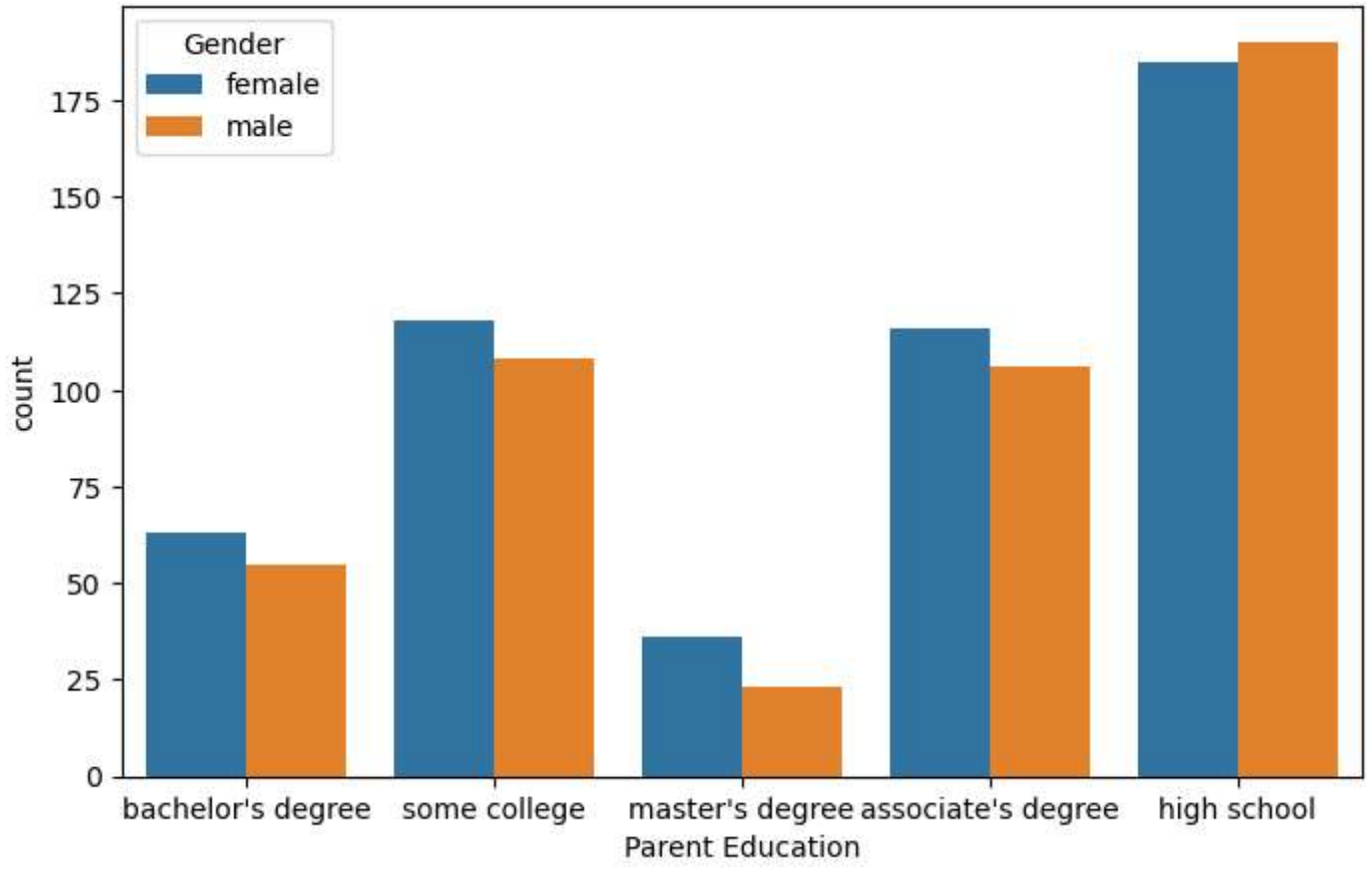
Multivariate analysis

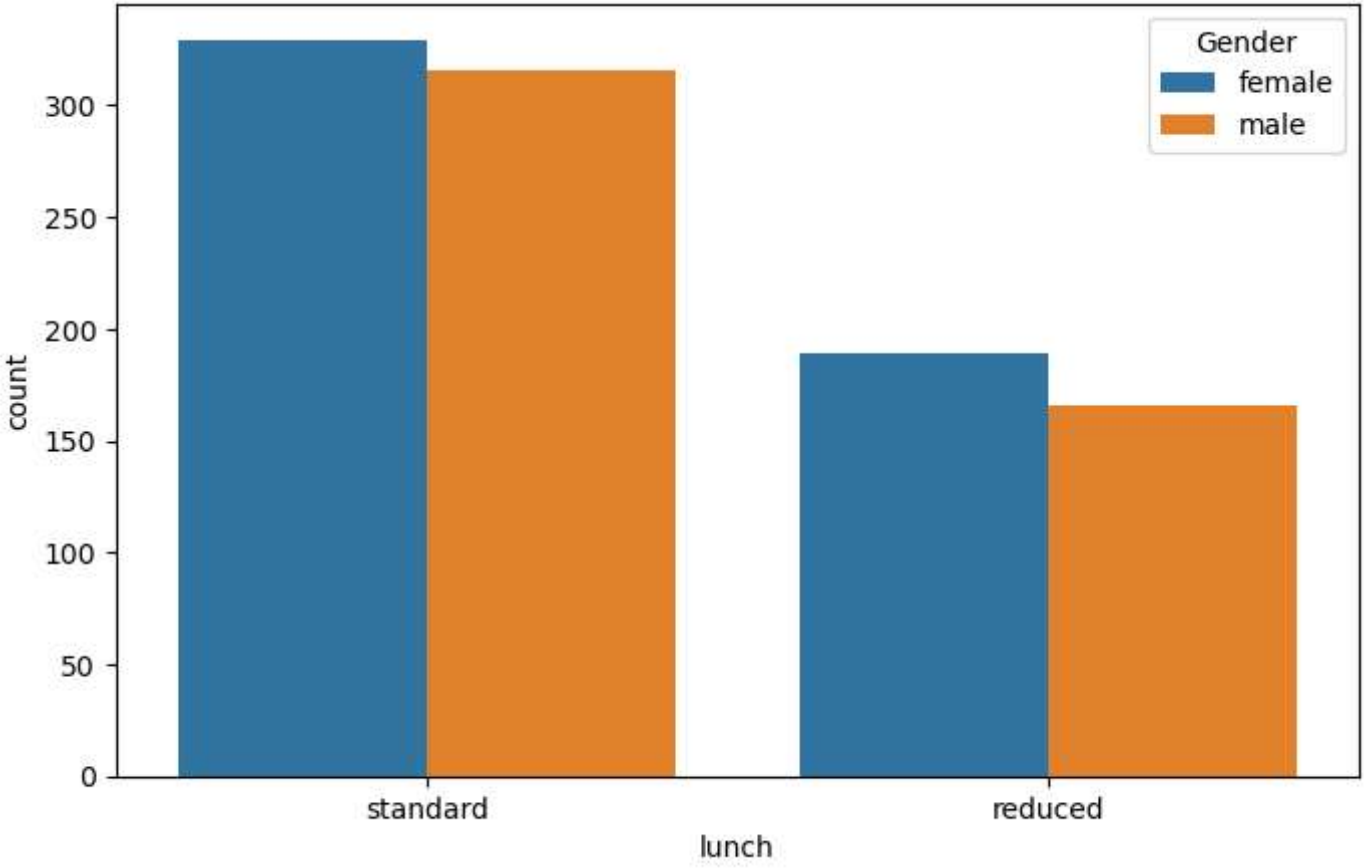
We want to figure out how much impact Gender has on all the other data. Let us do a multivariate analysis

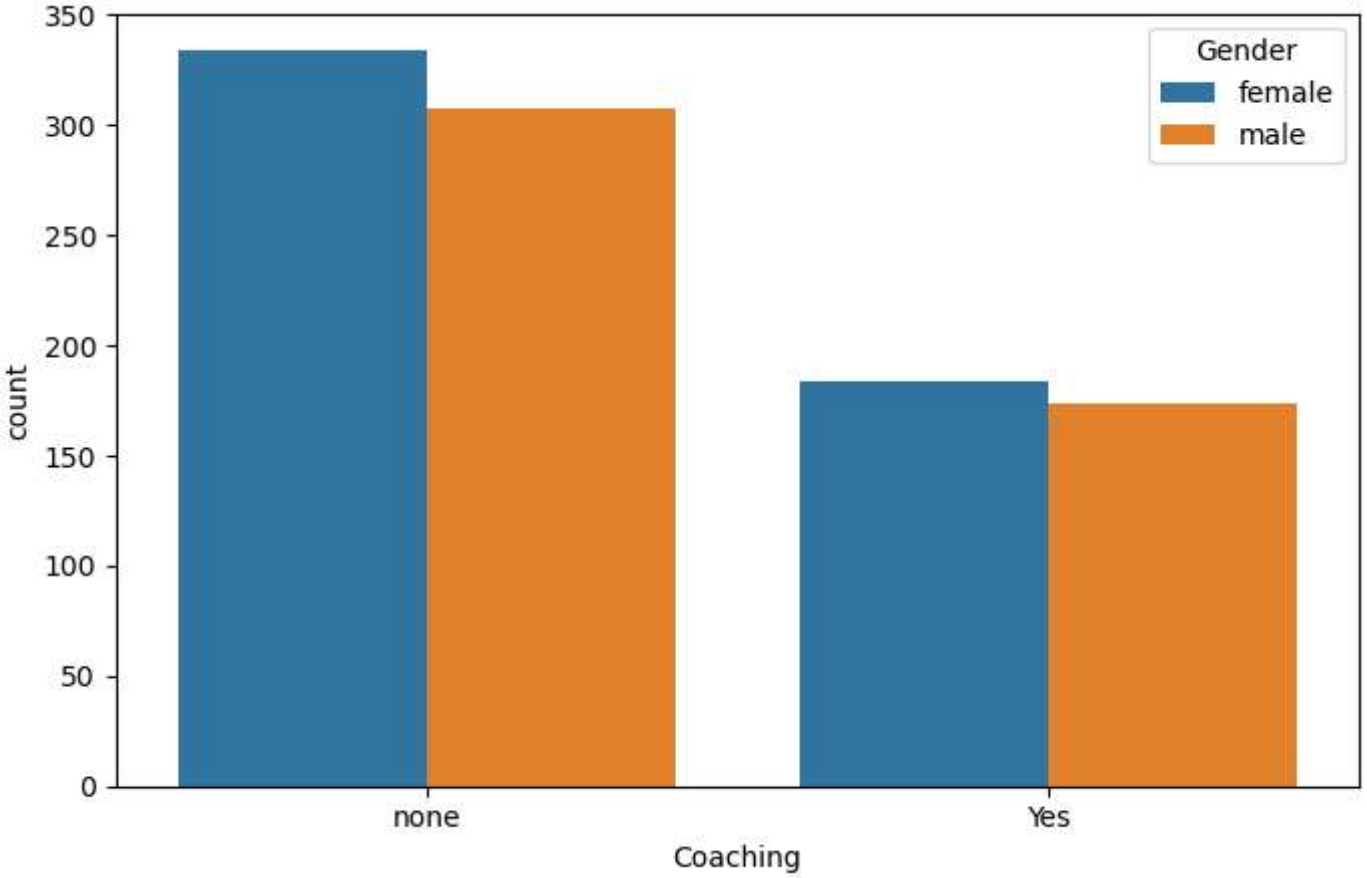
In [104…

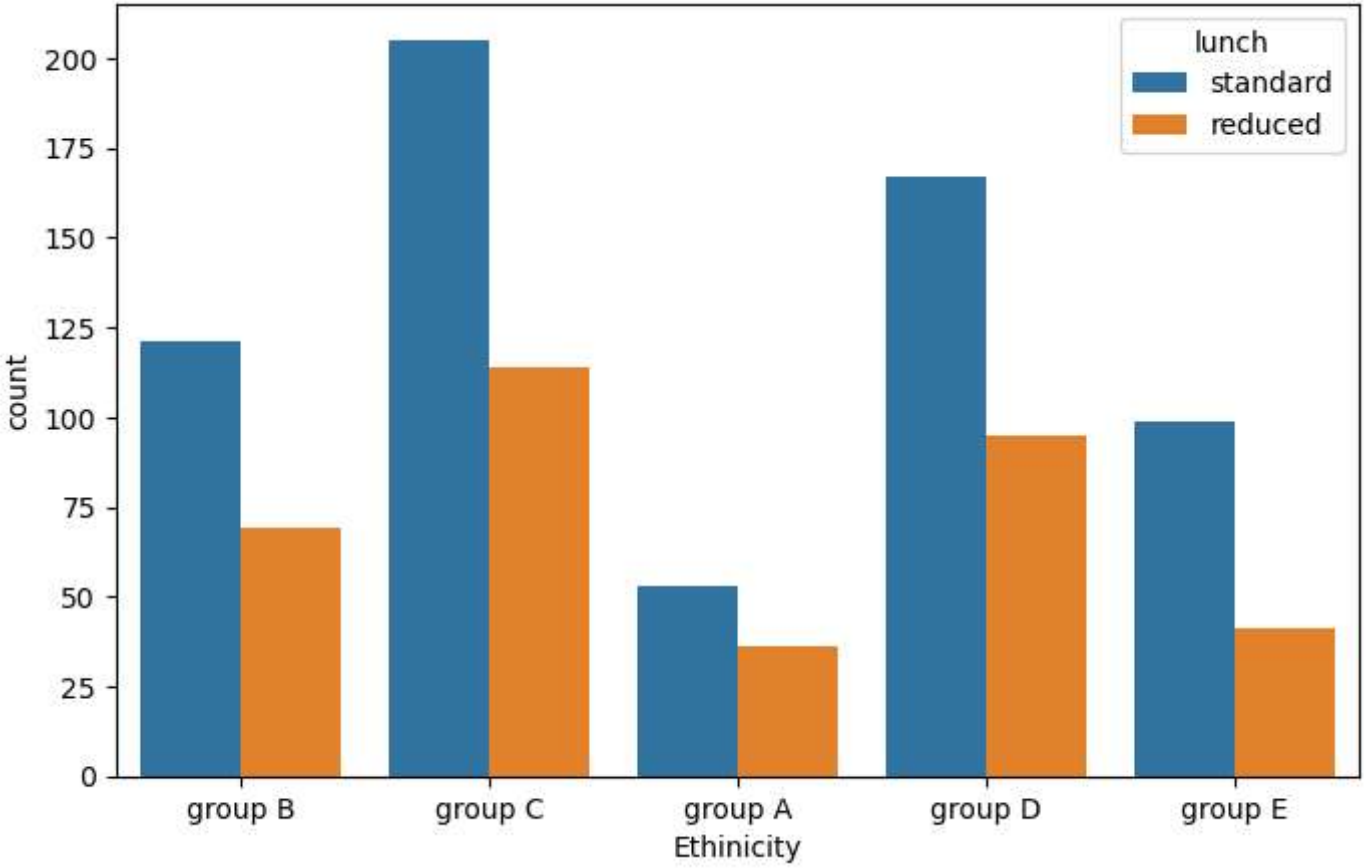
```
def Multivariate(a,b):
    plt.figure(figsize=(8,5))
```

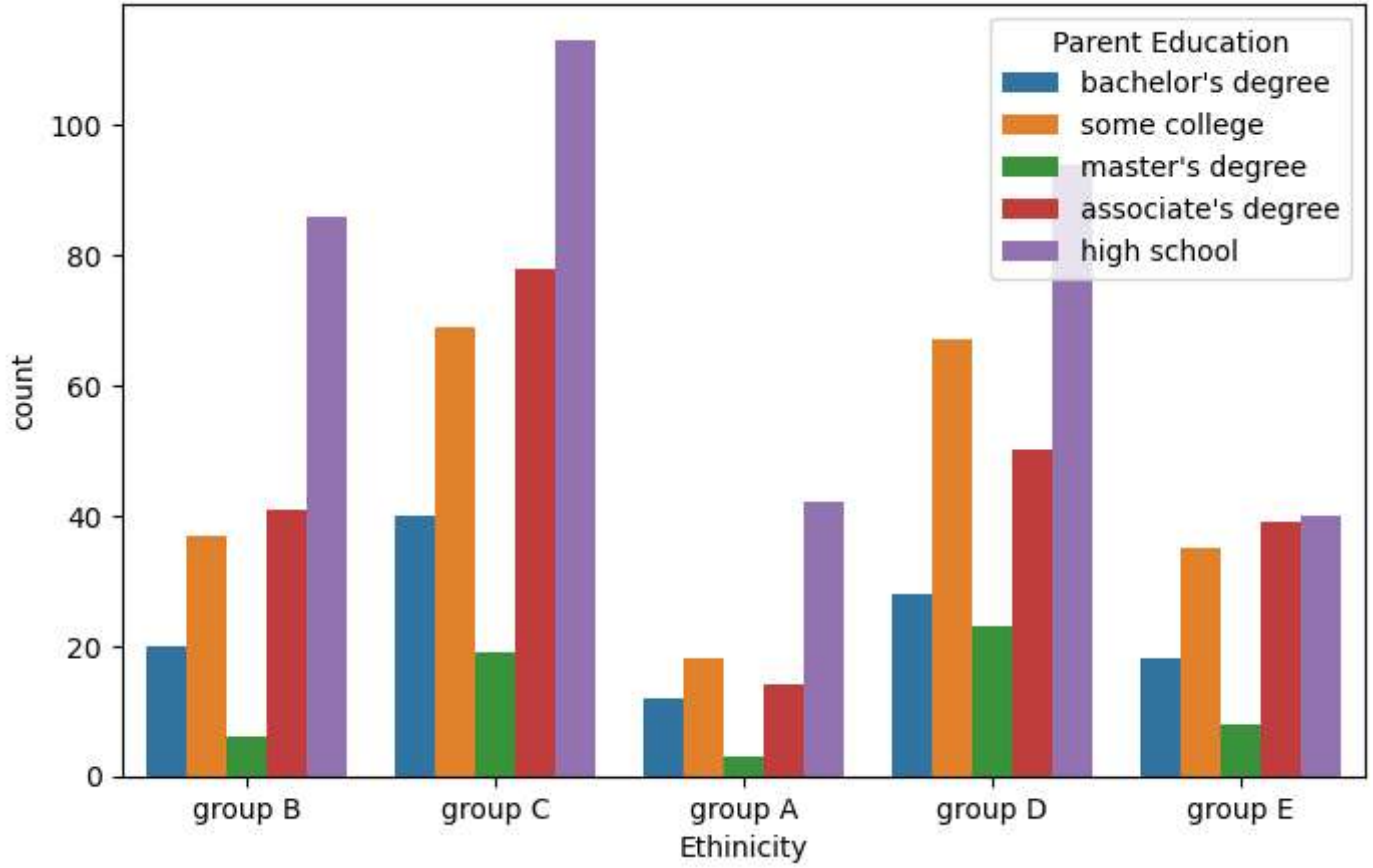
```
    sns.countplot(x=a,data=data,hue=b)

Multivariate('Parent Education','Gender')
Multivariate('lunch','Gender')
Multivariate('Coaching','Gender')
Multivariate('Ethinicity','lunch')
Multivariate('Ethinicity','Parent Education')
Multivariate('Parent Education','Ethinicity')
```

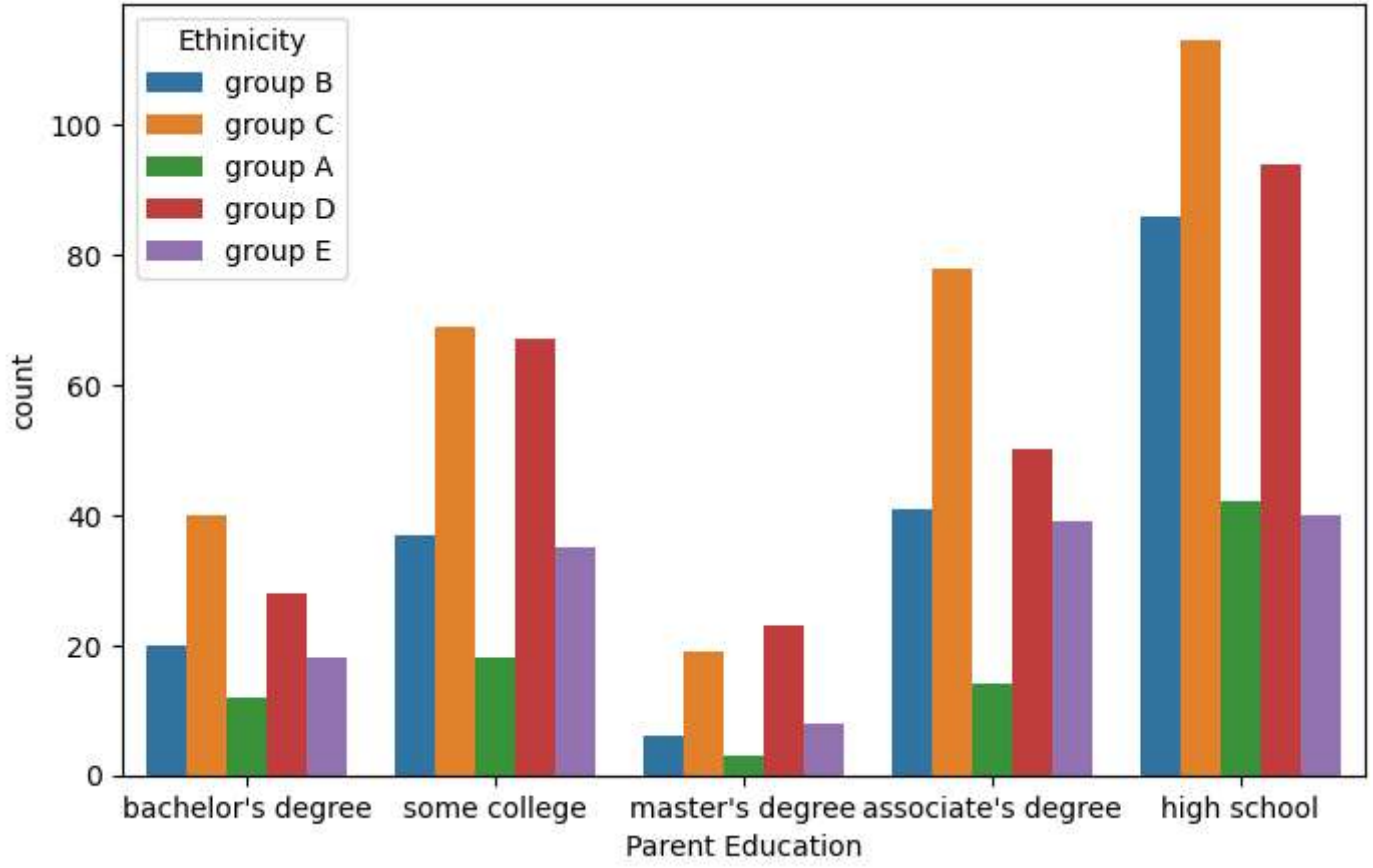



Gender
female
male
300
250
200
150
100
50
0
count
standard
reduced
lunch


350
300
250
200
150
100
50
0
count
Gender
female
male
none
Yes
Coaching


lunch
standard
reduced
count
200
175
150
125
100
75
50
25
0
group B
group C
group A
group D
group E
Ethinicity


Parent Education
bachelor's degree
some college
master's degree
associate's degree
high school
count
0
20
40
60
80
100
group B
group C
group A
group D
group E
Ethinicity



Observtion on multivariate

When it comes to rural areas Gender based discrimination might have an impact in education. Out first objectivee is to find out how Gender impacts the other features.

Gender: When compared against Lunch, Parent Education and coaching there does not seen to be an influence off Gender in any of the social parameters bassed on this data. Both Male and Female have equal distributions appropriate to their population count in the aspects of lunch availability, Parent education and coaching accesibility. But from scatter plots we observed male student performing better in Numeerical and Vocabulary

Ethinicity doesn't seem to have impact on lunch, parent education or coaching. But again as wee saw in scatter plots certain Ethinicty perrform better Numerical and Vocabulary.

5. Data Wrangling

5.1 Univariate Filters

Numerical and Categorical Data

- Identify top 5 significant features by evaluating each feature independently with respect to the target variable by exploring

1. Mutual Information (Information Gain)
2. Gini index
3. Gain Ratio
4. Chi-Squared test
5. Fisher Score (From the above 5 you are required to use only any **two**)

For Text data

1. Stemming / Lemmatization.
2. Forming n-grams and storing them in the document vector.
3. TF-IDF (From the above 2 you are required to use only any **two**)

Score: 3 Marks

Chi-Squared test

In [75]:
```
##---------Type the code below this line-----------------##
#creating a function to execute chisq test for independence
from scipy.stats import chi2_contingency

def chisq(col1,col2):
    #create a contingency table
    table=pd.crosstab(newdata[col1],newdata[col2])
    #get chi_Sq statistics,p-value,degrees of freedom and expected frequencies.
    stat, p, dof, expected = chi2_contingency(table)
    #set significance level
    alpha=0.05
```

```
    if p<=0.05:
        print('Features {0} and {1} are ASSOCIATED'.format(col1, col2))
    else:
        print('Features {0} and {1} are NOT associated'.format(col1, col2))
```

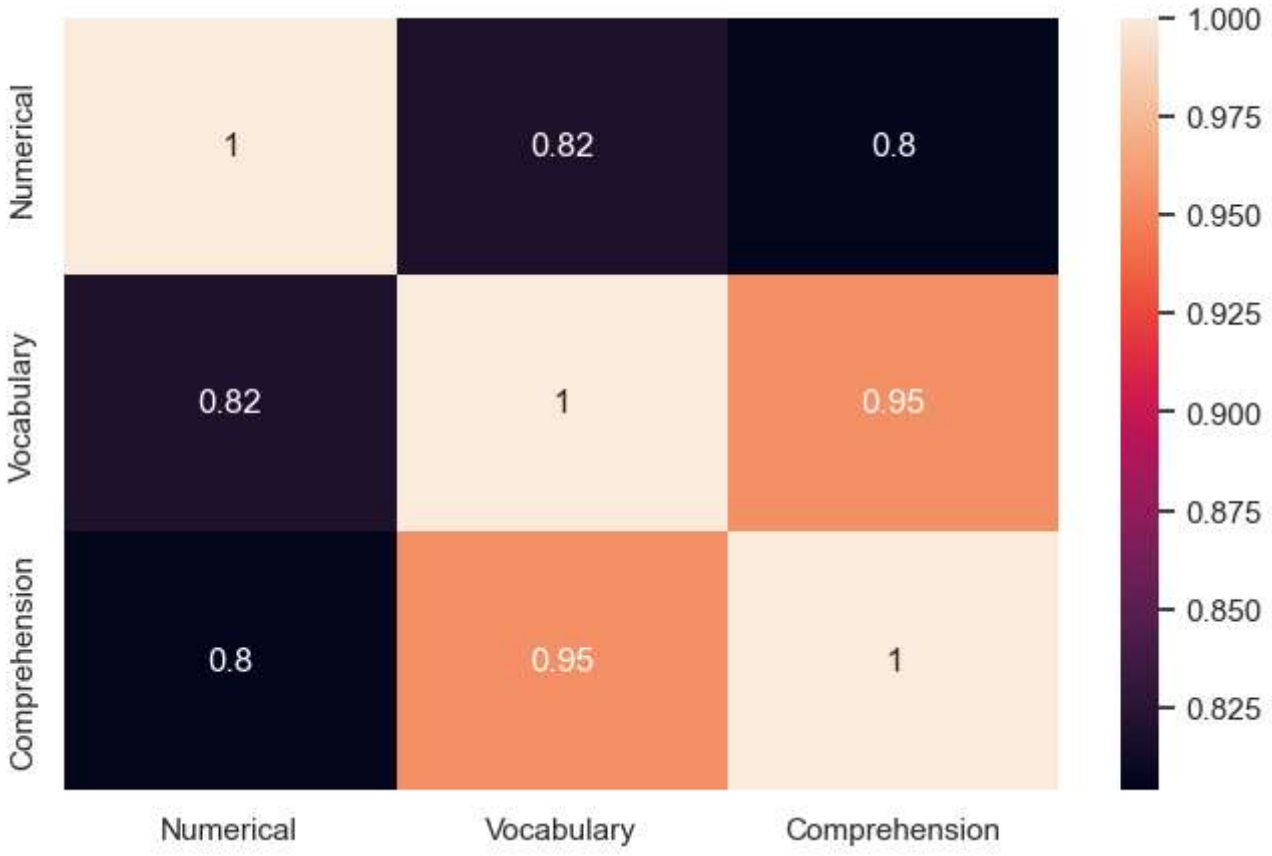
In [76]:
```
chisq('Gender','lunch')
chisq('Gender','Parent Education')
chisq('Gender','Ethinicity')
chisq('Gender','Coaching')
chisq('lunch','Coaching')
chisq('lunch','Parent Education')
chisq('lunch','Ethinicity')
chisq('Parent Education','Ethinicity')
chisq('Parent Education','Coaching')
chisq('Ethinicity','Coaching')
chisq('Numerical','Vocabulary')
chisq('Numerical','Comprehension')
chisq('Comprehension','Vocabulary')
```

```
Features Gender and lunch are NOT associated
Features Gender and Parent Education are NOT associated
Features Gender and Ethinicity are NOT associated
Features Gender and Coaching are NOT associated
Features lunch and Coaching are NOT associated
Features lunch and Parent Education are NOT associated
Features lunch and Ethinicity are NOT associated
Features Parent Education and Ethinicity are NOT associated
Features Parent Education and Coaching are NOT associated
Features Ethinicity and Coaching are NOT associated
Features Numerical and Vocabulary are ASSOCIATED
Features Numerical and Comprehension are ASSOCIATED
Features Comprehension and Vocabulary are ASSOCIATED
```

Pearson Coefficient

In [522…
```
nums=['Numerical','Vocabulary','Comprehension']
corr_matrix=data[nums].corr(method='pearson')
plt.figure(figsize=(8,5))
#fig, ax = plt.subplots(figsize=(10,10))
sns.heatmap(corr_matrix,annot=True)
```

Out[522]:
```
<AxesSubplot:>
```



In [ ]:

Fisher Score

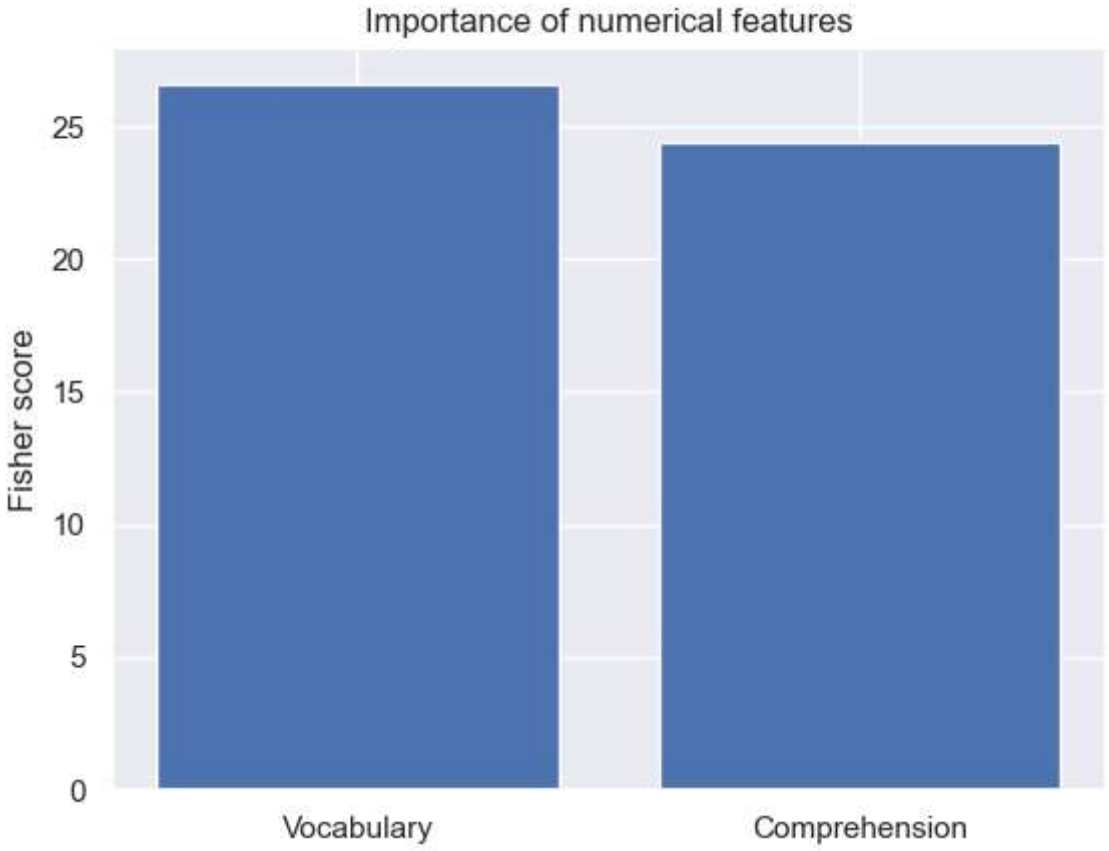
In [538…

```
# calculate the Fisher score for the numerical variables
from sklearn.feature_selection import f_classif
import numpy as np

X = data[['Vocabulary', 'Comprehension']]
y = data['Numerical']

fisher_scores, _ = f_classif(X, y)
print(fisher_scores)
```

```
[26.6145158  24.41772967]
```

In [540…

```
# plot the Fisher scores as a bar plot
plt.bar(range(len(fisher_scores)), fisher_scores)
plt.xticks(range(len(fisher_scores)), ['Vocabulary', 'Comprehension'])
plt.ylabel('Fisher score')
plt.title('Importance of numerical features')
plt.show()
```



5.2 Report observations

Write your observations from the results of each method. Clearly justify your choice of the method.

Score 1 mark

---------Type the code below this line----------

Reporting observation and selecting significant features

Feature selection : Lets us use filter methods to identify or eliminate features which must be considered for clustering

Chi-Square: The major feature which influence the clustering are of categorical type. So we used Chi-square test to determine the independence of the features and find their association. From the test it is clear the none of the categorical variables are associated. This means Gender, Ethinicity, lunch, Parent Education and Coaching are independent feature which can contribute to clustering.

Corelation: The corelation matrix of all scores clearly indicate Vocabulary and Comprehension go hand in hand. Numerical has less corelation to Vocabulary. Even though the score is on the higher side this is due to the distrubution of marks being same. From a score point of view Numerical and Vocabulary represent different competencies. So they must be considered as independent features

Fisher score: Fishore score is to identify the most informative feature in this use case. From the corelation matrix wee understand Vocabulary and Comprehension are highly corelated. We definitely need to include Numerical as competency score is critical for classification of students.
Fisher score is useed to identiffy iff we can put Vocabulary or Comprehension. Vocabulary has the high score so we will pick Vocabulary over Comprehension.

Conclusion: Comprehension will be dropped from clusterring. Gender seems to have NO influence on any parameters based on all the EDA analysis. Gender will be droppeed from clusering ***Features selected : Ethinicity, Parent Education, lunch, Coaching, Numerical and Vocabulary***

6. Implement Machine Learning Techniques

Use any 2 ML algorithms

1. Classification -- Decision Tree classifier
2. Clustering -- kmeans
3. Association Analysis

4. Anomaly detection

5. Textual data -- Naive Bayes classifier (not taught in this course)

A clear justification have to be given for why a certain algorithm was chosen to address your problem.

Score: 4 Marks (2 marks each for each algorithm)

6.1 ML technique 1 + Justification

ML Algorithm 1 - K-means clustering

Why K-means Our objective was to group students into clusters so there can be a focus training for each cluster. Our Analytics taks will be an unsuperrvised one as there is no Y label Being the most powerful and versatile clustering algorithm we have choseen K-means to perform clustering K-means tries to find groups of data points that are similar to each other based on their distances in the feature space. The algorithm tries to minimize within-cluster variance and maximizes between-cluster variance. This is exactly what we require to find out distint group of student for a focussed training

Program flow

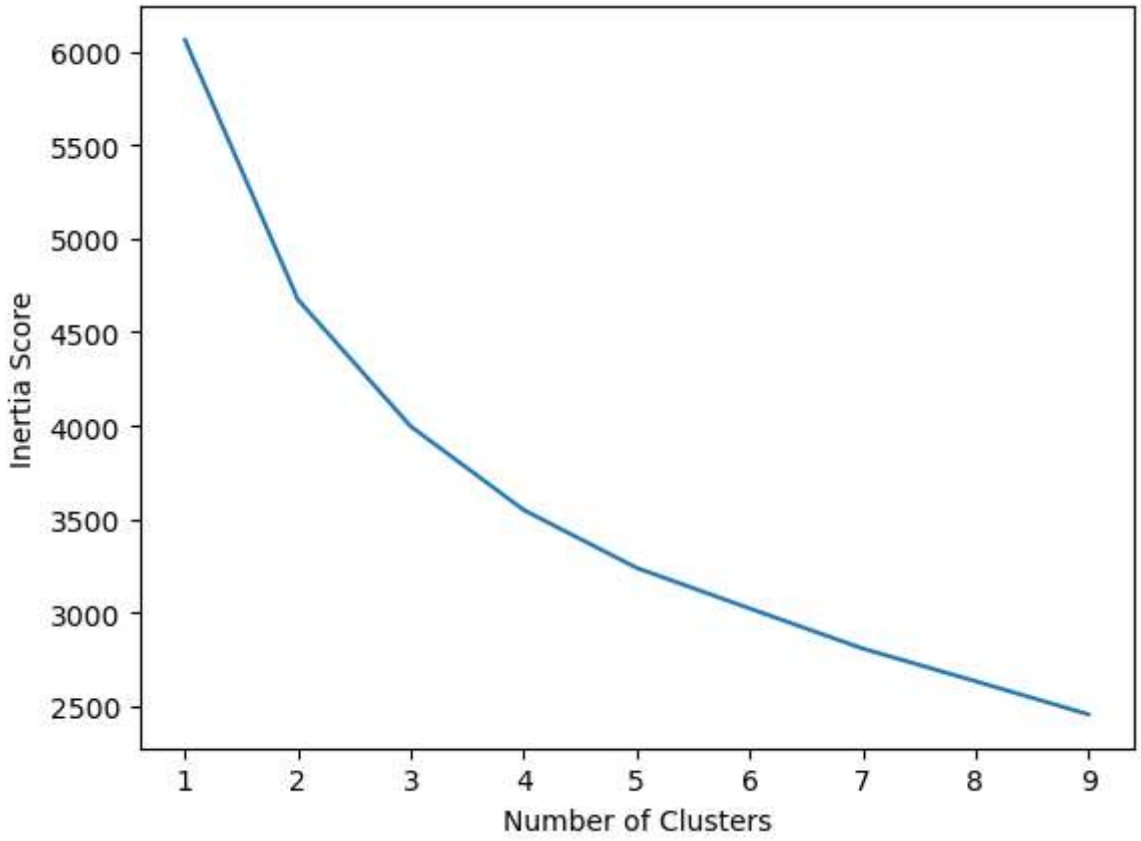
1. Perform encoding of categorrical variables and normalize data for consumption of K-means algorithm
2. Drop the featuers that are not required for clusterring from the data set bassed ont he feature seelection strategies
3. Calculate inertia using **Elbow Method** Find out the optinum **number of clusters**
4. Plot the inerrtia values and verify the clusterr value
5. Perform **K-means clustering** and get the predicted labels
6. Reduce the dimensions to 2D using PCA And plot the clusters to visuvalize
7. Plot the clusterr graph in 2D
8. Measure the performance of clustering using **Silhouette** score.
9. While not relevant to unsuperrvised learing, we also try to compare the k-means pred label to a label which didnt participate in clustering to find **Accuracy, Precision, Recall, F1-score, AUC-ROC**

In [17]:
```
##---------Type the code below this line-----------------##
# Lets us do clusterng using K-means.

from sklearn.cluster import KMeans
```

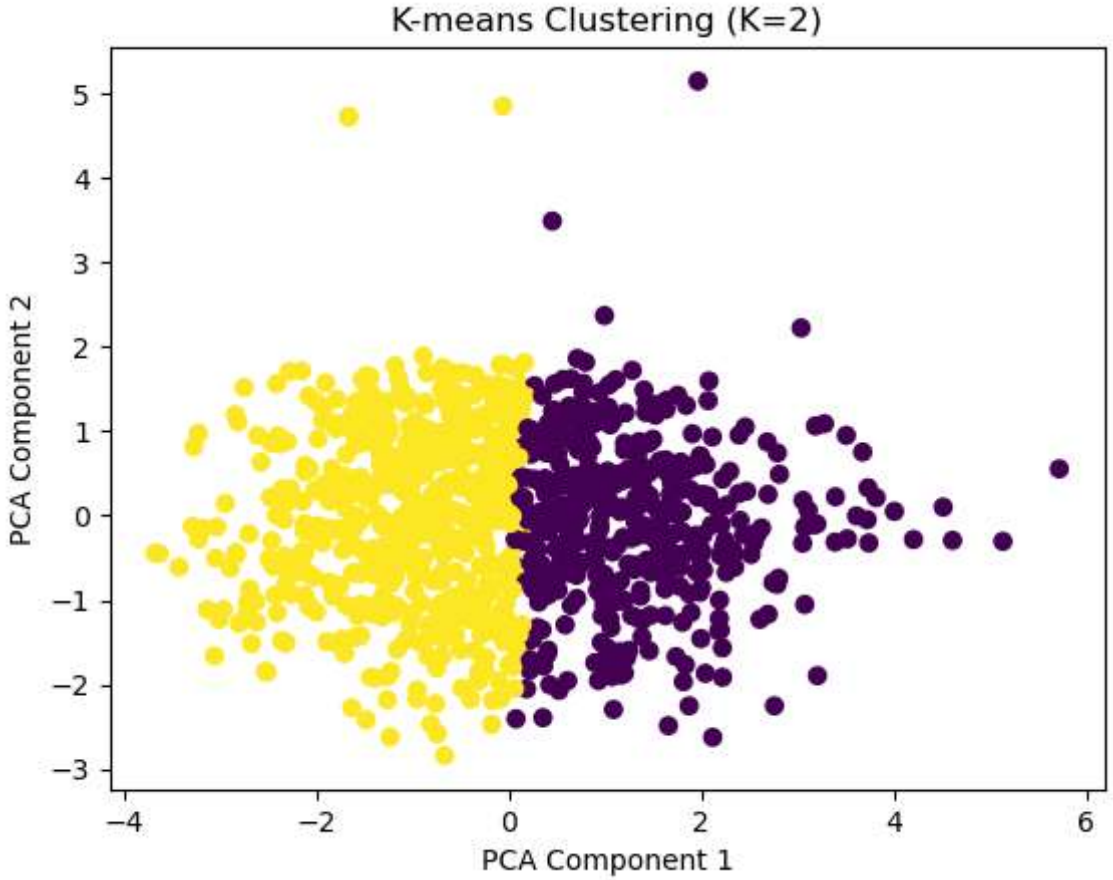
```
from sklearn.metrics import confusion_matrix, accuracy_score, precision_score, recall_score, f1_score, roc_auc_score
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt
from sklearn.metrics import silhouette_score
import numpy as np

import random

# Encode categorical variables
encoder = LabelEncoder()
for col in ['Gender', 'Ethinicity', 'Parent Education', 'lunch', 'Coaching']:
    data[col] = encoder.fit_transform(data[col])
# Normalize the data
scaler = StandardScaler()

# Drop Identifier and few other Columns as they are not useful for clustering

#newdata['target'] = random.randint(0,1)
X = data.drop(columns=['Identifier', 'Comprehension', 'Gender'])
X_norm = scaler.fit_transform(X)

# Reduce dimensionality by transforming the array
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_norm)

# Use the elbow method to determine the ideal number of clusters
inertias = []
cluster_range = range(1,10)

for i in cluster_range:
    kmeans = KMeans(n_clusters=i, random_state=79)
    kmeans.fit(X_norm)
    inertias.append(kmeans.inertia_)

# Calculate the elbow point
diff = np.diff(inertias)
diff2 = np.diff(diff)
elbow_point = cluster_range[np.argmax(diff2) + 1]
print('Elbow point is ', elbow_point)
print('Cluster size is ', elbow_point)

# Plot the inertias scores - Elbow method
plt.plot(range(1, 10), inertias)
plt.xlabel('Number of Clusters')
```

```
plt.ylabel('Inertia Score')
plt.show()


# K-means clustering
kmeans = KMeans(n_clusters=elbow_point, random_state=79)
kmeans.fit(X_norm)
# Calculate evaluation metrics
y_true = data['Gender']
y_pred = kmeans.labels_
print(y_true)
print(y_pred)

# Calculate the silhouette score for the clustering
silhouette_avg = silhouette_score(X_norm, kmeans.labels_)
print("The average silhouette_score is :", silhouette_avg)


# Visualize the clustering using PCA
plt.scatter(X_pca[:, 0], X_pca[:, 1], c=kmeans.labels_)
plt.xlabel('PCA Component 1')
plt.ylabel('PCA Component 2')
plt.title('K-means Clustering (K={})'.format(elbow_point))
plt.show()

acc = accuracy_score(y_true, y_pred)

prec = precision_score(y_true, y_pred)


rec = recall_score(y_true, y_pred)

print('Accuracy:', acc)
print('Precision:', prec)
print('Recall:', rec)
f1 = f1_score(y_true, y_pred)
auc_roc = roc_auc_score(y_true, y_pred)
cm = confusion_matrix(y_true, y_pred)

# Print the evaluation metrics and confusion matrix
print('F1 score:', f1)
print('AUC-ROC score:', auc_roc)
```

```
C:\Users\vinot\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1036: UserWarning: KMeans is known to have a memo
ry leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environ
ment variable OMP_NUM_THREADS=4.
  warnings.warn(
Elbow point is  2
Cluster size is  2
```



```
0       0
1       0
2       0
3       1
4       1
       ..
1005    0
1006    1
1007    0
1008    0
1009    0
Name: Gender, Length: 1010, dtype: int64
[1 1 1 ... 0 1 1]
The average silhouette_score is : 0.2039301562542388
```



```
Accuracy: 0.47029702970297027
Precision: 0.4567901234567901
Recall: 0.5329218106995884
F1 score: 0.49192782526115864
AUC-ROC score: 0.4725677755788018
```

In [ ]:

6.2 ML technique 2 + Justification

ML Algorithm 2 - Mini-batch K-means Clustering

Why Mini-Batch K-means Clustering The objective of using a second clustering algorithm is it compare the cluster and silhouette score with the first algorithm. We chose Agglomerative and mini-bath clustering for the 2nd algorithm. Since mini-batch K-means also picks a centroid, assigns the data point closest to centroid and update the cluster using means. **The difference is mini-batch does it on randomly sampled subsets of data** This way we are using the same methodology for picking centroid and updating clusterr but on multiple iterations in small batches Note: We tried Agglomerative clustering which produced same result even thought it is used in Hierarchial dataset.

Program flow

1. Perform encoding of categorical variables and normalize data for consumption of Mini-batch K-means algorithm
2. Drop the features that are not required for clusterring from the data set bassed ont he feature seelection strategies
3. USe the calculated Cluster valus using **Elbow Method**
4. Perform **Mini Batch K-means clustering** and get the predicted labels
5. Reduce the dimensions to 2D using PCA And plot the clusters to visuvalize
6. Plot the clusterr graph in 2D
7. Measure the performance of clustering using **Silhouette** score.
8. While not relevant to unsuperrvised learing, we also try to compare the k-means pred label to a label which didnt participate in clustering to find **Accuracy, Precision, Recall, F1-score, AUC-ROC**

Please ignore the Mini-Batch memory leak Warning

1. We have set OMP_NUM_THREADS to 1 C:\Users\vinot\anaconda3\Lib\site-packages\sklearn\cluster>echo %OMP_NUM_THREADS% 1
2. Cannot have batch size as >2048 as our data size is 1000 and we need batches in 100 **Error is due to MKL lib in Windows**

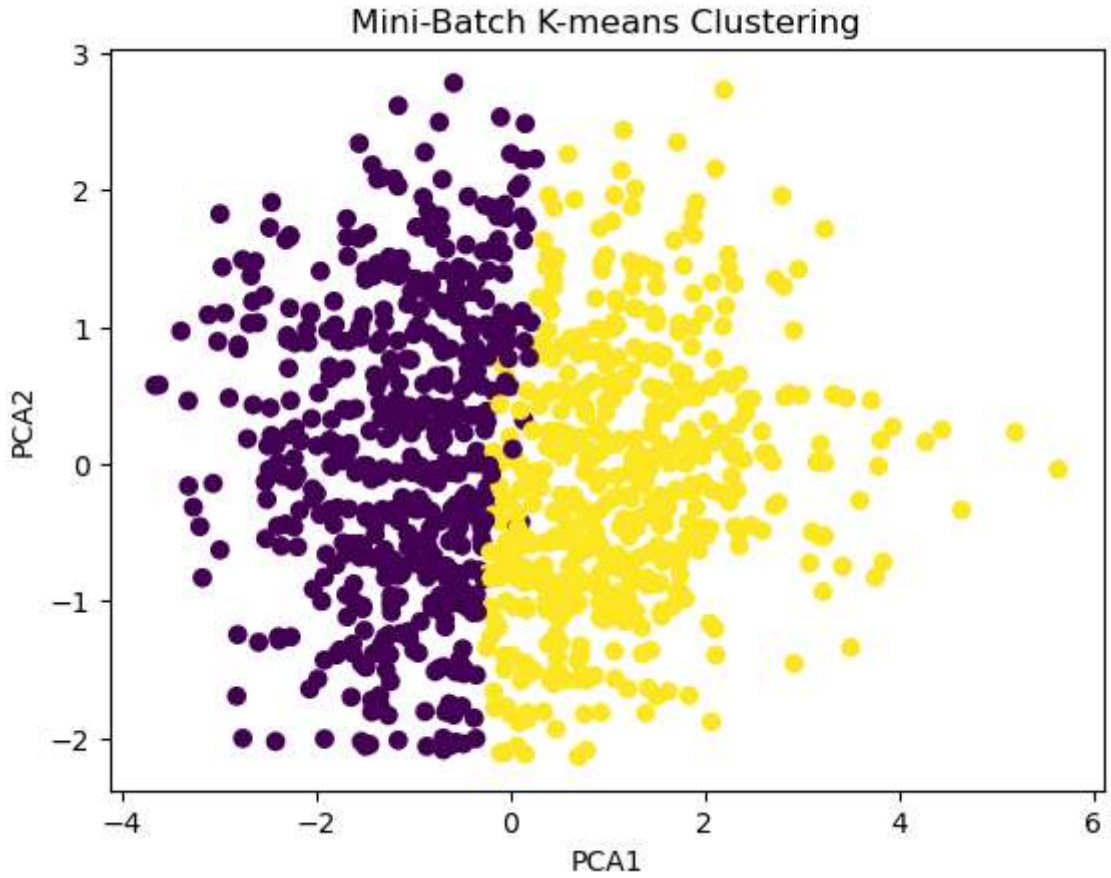
In [68]:
```
## Mini-batch clustering

import pandas as pd
from sklearn.cluster import MiniBatchKMeans
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import silhouette_score

# Encode categorical variables
encoder = LabelEncoder()
for col in ['Gender', 'Ethinicity', 'Parent Education', 'lunch', 'Coaching']:
    data[col] = encoder.fit_transform(data[col])

# Load the dataset and drop the unnecessary columns
data_scaled = data.drop(columns=['Identifier', 'Gender', 'Comprehension'])

# Standardize the data
scaler = StandardScaler()
data_scaled = scaler.fit_transform(data_scaled)

# Determine the optimal number of clusters using silhouette score
scores = []
for n_clusters in range(2, 11):
    kmeans = MiniBatchKMeans(n_clusters=n_clusters, batch_size=100, max_iter=100)
    kmeans.fit(data_scaled)
    minilabels = kmeans.labels_
    score = silhouette_score(data_scaled, minilabels)
    scores.append(score)
    print(f"Silhouette score for n_clusters={n_clusters}: {score}")

# Determine the optimal number of clusters using elbow method
distortions = []
for n_clusters in range(2, 11):
    kmeans = MiniBatchKMeans(n_clusters=n_clusters, batch_size=100, max_iter=100)
    kmeans.fit(data_scaled)
    distortions.append(kmeans.inertia_)
#plt.plot(range(2, 11), distortions, marker='o')
#plt.xlabel('Number of clusters')
#plt.ylabel('Distortion')
```

```
#plt.show()

# Fit the mini-batch k-means model using the optimal number of clusters
kmeans = MiniBatchKMeans(n_clusters=2, batch_size=100, max_iter=100)
kmeans.fit(data_scaled)
minilabels = kmeans.labels_

# Apply PCA for dimensionality reduction and visualization
pca = PCA(n_components=2)
data_reduced = pca.fit_transform(data_scaled)

# Plot the clusters
plt.scatter(data_reduced[:, 0], data_reduced[:, 1], c=minilabels)
plt.xlabel('PCA1')
plt.ylabel('PCA2')
plt.title('Mini-Batch K-means Clustering')
plt.show()

# Calculate the Silhouette score
score = silhouette_score(data_scaled, minilabels)
print(f"Mini Batch K-means Silhouette score: {scores[0]}")

y_true = data['Gender']
y_pred = kmeans.labels_
#print(y_true)
#print(y_pred)

acc = accuracy_score(y_true, y_pred)
prec = precision_score(y_true, y_pred)
rec = recall_score(y_true, y_pred)

print('Accuracy:', acc)
print('Precision:', prec)
print('Recall:', rec)

f1 = f1_score(y_true, y_pred)
auc_roc = roc_auc_score(y_true, y_pred)
cm = confusion_matrix(y_true, y_pred)

# Print the evaluation metrics and confusion matrix
print('F1 score:', f1)
print('AUC-ROC score:', auc_roc)
```

```
C:\Users\vinot\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1043: UserWarning: MiniBatchKMeans is known to ha
ve a memory leak on Windows with MKL, when there are less chunks than available threads. You can prevent it by setting
batch_size >= 2048 or by setting the environment variable OMP_NUM_THREADS=1
  warnings.warn(
Silhouette score for n_clusters=2: 0.20210354266041375
Silhouette score for n_clusters=3: 0.16967112069343715
C:\Users\vinot\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1043: UserWarning: MiniBatchKMeans is known to ha
ve a memory leak on Windows with MKL, when there are less chunks than available threads. You can prevent it by setting
batch_size >= 2048 or by setting the environment variable OMP_NUM_THREADS=1
  warnings.warn(
C:\Users\vinot\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1043: UserWarning: MiniBatchKMeans is known to ha
ve a memory leak on Windows with MKL, when there are less chunks than available threads. You can prevent it by setting
batch_size >= 2048 or by setting the environment variable OMP_NUM_THREADS=1
  warnings.warn(
Silhouette score for n_clusters=4: 0.19603582673107786
C:\Users\vinot\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1043: UserWarning: MiniBatchKMeans is known to ha
ve a memory leak on Windows with MKL, when there are less chunks than available threads. You can prevent it by setting
batch_size >= 2048 or by setting the environment variable OMP_NUM_THREADS=1
  warnings.warn(
Silhouette score for n_clusters=5: 0.19291444881753733
C:\Users\vinot\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1043: UserWarning: MiniBatchKMeans is known to ha
ve a memory leak on Windows with MKL, when there are less chunks than available threads. You can prevent it by setting
batch_size >= 2048 or by setting the environment variable OMP_NUM_THREADS=1
  warnings.warn(
Silhouette score for n_clusters=6: 0.1946554460696431
C:\Users\vinot\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1043: UserWarning: MiniBatchKMeans is known to ha
ve a memory leak on Windows with MKL, when there are less chunks than available threads. You can prevent it by setting
batch_size >= 2048 or by setting the environment variable OMP_NUM_THREADS=1
  warnings.warn(
Silhouette score for n_clusters=7: 0.17158885343033242
C:\Users\vinot\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1043: UserWarning: MiniBatchKMeans is known to ha
ve a memory leak on Windows with MKL, when there are less chunks than available threads. You can prevent it by setting
batch_size >= 2048 or by setting the environment variable OMP_NUM_THREADS=1
  warnings.warn(
Silhouette score for n_clusters=8: 0.1865603079619539
C:\Users\vinot\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1043: UserWarning: MiniBatchKMeans is known to ha
ve a memory leak on Windows with MKL, when there are less chunks than available threads. You can prevent it by setting
batch_size >= 2048 or by setting the environment variable OMP_NUM_THREADS=1
  warnings.warn(
Silhouette score for n_clusters=9: 0.2030486482489119
```

```
C:\Users\vinot\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1043: UserWarning: MiniBatchKMeans is known to ha
ve a memory leak on Windows with MKL, when there are less chunks than available threads. You can prevent it by setting
batch_size >= 2048 or by setting the environment variable OMP_NUM_THREADS=1
  warnings.warn(
Silhouette score for n_clusters=10: 0.1711303659048081
```

```
C:\Users\vinot\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1043: UserWarning: MiniBatchKMeans is known to ha
ve a memory leak on Windows with MKL, when there are less chunks than available threads. You can prevent it by setting
batch_size >= 2048 or by setting the environment variable OMP_NUM_THREADS=1
  warnings.warn(
C:\Users\vinot\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1043: UserWarning: MiniBatchKMeans is known to ha
ve a memory leak on Windows with MKL, when there are less chunks than available threads. You can prevent it by setting
batch_size >= 2048 or by setting the environment variable OMP_NUM_THREADS=1
  warnings.warn(
C:\Users\vinot\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1043: UserWarning: MiniBatchKMeans is known to ha
ve a memory leak on Windows with MKL, when there are less chunks than available threads. You can prevent it by setting
batch_size >= 2048 or by setting the environment variable OMP_NUM_THREADS=1
  warnings.warn(
C:\Users\vinot\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1043: UserWarning: MiniBatchKMeans is known to ha
ve a memory leak on Windows with MKL, when there are less chunks than available threads. You can prevent it by setting
batch_size >= 2048 or by setting the environment variable OMP_NUM_THREADS=1
  warnings.warn(
C:\Users\vinot\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1043: UserWarning: MiniBatchKMeans is known to ha
ve a memory leak on Windows with MKL, when there are less chunks than available threads. You can prevent it by setting
batch_size >= 2048 or by setting the environment variable OMP_NUM_THREADS=1
  warnings.warn(
C:\Users\vinot\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1043: UserWarning: MiniBatchKMeans is known to ha
ve a memory leak on Windows with MKL, when there are less chunks than available threads. You can prevent it by setting
batch_size >= 2048 or by setting the environment variable OMP_NUM_THREADS=1
  warnings.warn(
C:\Users\vinot\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1043: UserWarning: MiniBatchKMeans is known to ha
ve a memory leak on Windows with MKL, when there are less chunks than available threads. You can prevent it by setting
batch_size >= 2048 or by setting the environment variable OMP_NUM_THREADS=1
  warnings.warn(
C:\Users\vinot\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1043: UserWarning: MiniBatchKMeans is known to ha
ve a memory leak on Windows with MKL, when there are less chunks than available threads. You can prevent it by setting
batch_size >= 2048 or by setting the environment variable OMP_NUM_THREADS=1
  warnings.warn(
C:\Users\vinot\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1043: UserWarning: MiniBatchKMeans is known to ha
ve a memory leak on Windows with MKL, when there are less chunks than available threads. You can prevent it by setting
batch_size >= 2048 or by setting the environment variable OMP_NUM_THREADS=1
  warnings.warn(
C:\Users\vinot\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1043: UserWarning: MiniBatchKMeans is known to ha
ve a memory leak on Windows with MKL, when there are less chunks than available threads. You can prevent it by setting
batch_size >= 2048 or by setting the environment variable OMP_NUM_THREADS=1
  warnings.warn(
```




```
Mini Batch K-means Silhouette score: 0.20210354266041375
Accuracy: 0.524
Precision: 0.5060240963855421
Recall: 0.5228215767634855
F1 score: 0.5142857142857142
AUC-ROC score: 0.5239590509300053
```

7. Conclusion

Compare the performance of the ML techniques used.

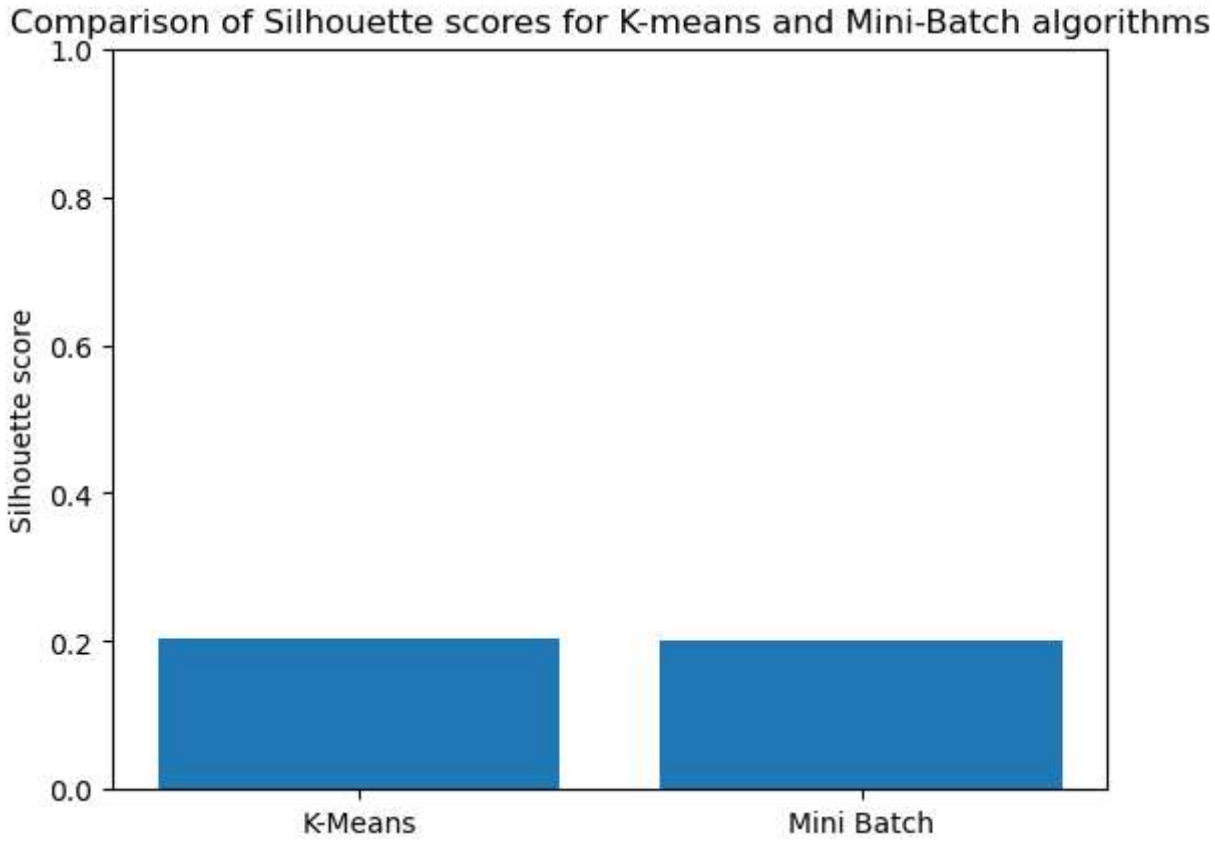
Derive values for preformance study metrics like accuracy, precision, recall, F1 Score, AUC-ROC etc to compare the ML algos and plot them. A proper comparision based on different metrics should be done and not just accuracy alone, only then the comparision

becomes authentic. You may use Confusion matrix, classification report, Word cloud etc as per the requirement of your application/problem.

Score 1 Mark

Comparing Silhoutee score of K-Means and Mini-batch K-Means

In [71]:
```
import matplotlib.pyplot as plt

# Define the Silhouette scores for two algorithms
score1 = 0.75
score2 = 0.85

# Create a bar chart
fig, ax = plt.subplots()
ax.bar(["K-Means", "Mini Batch"], [silhouette_avg, score])
ax.set_ylabel("Silhouette score")
ax.set_ylim([0, 1])
ax.set_title("Comparison of Silhouette scores for K-means and Mini-Batch algorithms")

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



---------Type the code below this line----------

Comparison Parameters other than Silhouette score - Considering Gender as Y Label

K-means Algorithm

Accuracy: 0.47029702970297027

Precision: 0.4567901234567901

Recall: 0.5329218106995884

F1 score: 0.49192782526115864

AUC-ROC score: 0.4725677755788018

Mini Batch K-means

Accuracy: 0.524

Precision: 0.5060240963855421

Recall: 0.5228215767634855

F1 score: 0.5142857142857142

AUC-ROC score: 0.5239590509300053

Conclusion Both the algorithms are performing equal and there is no betetr algorithm for this data. Also both suggest two clusterrs which are NOT actually very far apart. The Inter cluster distance also has some points with big distance.

In [ ]:

8. Solution

What is the solution that is proposed to solve the business problem discussed in Section 1. Also share your learnings while working through solving the problem in terms of challenges, observations, decisions made etc.

Score 2 Marks

----------Type the answers below this line----------

---------Type the answer below this line----------

Solution to the bussiness problem The problem we had to was to group students for a focussed training. Based on the clusterrs wee have here these are the proposed solutions,

1. Training one: for both clusterrs together as many doesn't seem to be far apart
2. Female students lack in Numerical and Vocabulary copmpetency. Training Two for Female student to improve these.
3. Group E Ethinic student do well with higher scores. Buddy them for group studies with Group B students

4. We also observed students with 'standard' lunch perform better. So take initiates such that all students receive same nutritious food

Learnings and Observations

1. Data set and underrstand the data is very key. Especially when we perrforrm Data wranggling tasks. We reffered the data set multipele times and had to go thru so many study material during Wrangling and Feature selection
2. EDA helps in data underrstanding. Initially we did only small set of EDA. The more we underrstand data and keep moving towards ML implemntation we got so many questions on the data. Eventually visualizing each feature really helped in underrstanding data.
3. ML Implementation: The decision of what algorithm to implement and its paraments kept us in loop. We literally changes parameters and even algorithms couple of times. Overrall learnings is there is a lot of thought and Deep thinkigng required in every staage of data science. Not as easy as writing code :)

Challenges Frankly, We felt lost. The more we studied the more questions we got. Truele even now I don't think all our questions are answered. ML algorithms and decision on Feature selection approach wass difficult

Decisions made We changed from Agglomerative to mini-bathc at the last minute.

This assignment is an eye opener on what to learn and how to approach a ML implementation

NOTE

All Late Submissions will incur a penalty of -2 marks. Do ensure on time submission to avoid penalty.

Good Luck!!!

In [ ]:

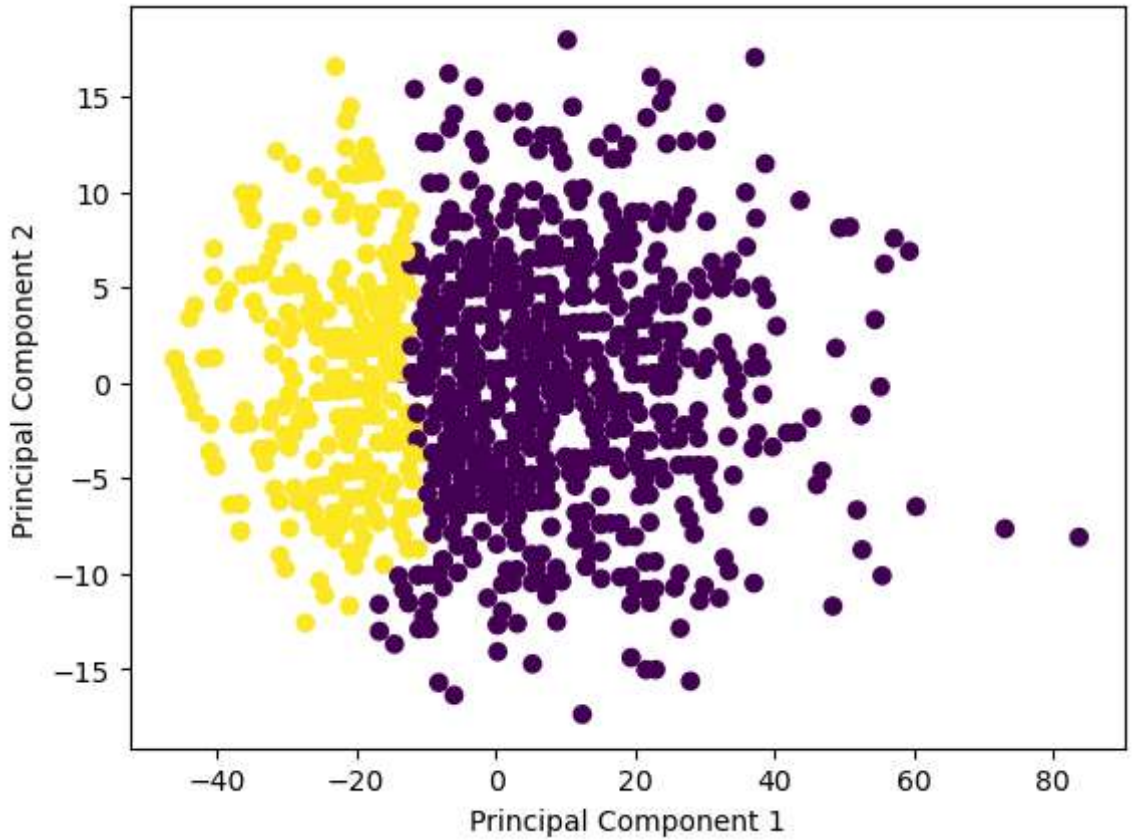
Backup Cells

In [16]:
```
##---------Type the code below this line-----------------##
# Lets us now try Agglomerative Clustering

from sklearn.cluster import AgglomerativeClustering
```

```
from scipy.cluster.hierarchy import dendrogram, linkage
from sklearn.metrics import silhouette_score
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt

# Encode categorical variables
encoder = LabelEncoder()
for col in ['Gender', 'Ethinicity', 'Parent Education', 'lunch', 'Coaching']:
    data[col] = encoder.fit_transform(data[col])

dataagg = data.drop(['Gender', 'Identifier', 'Comprehension'], axis=1)

# Initialize the Agglomerative clustering model with a range of n_clusters
scores = []
for n_clusters in range(2, 10):
    agg_clustering = AgglomerativeClustering(n_clusters=n_clusters, linkage='ward')
    agg_clustering.fit(dataagg)
    labels = agg_clustering.labels_
    score = silhouette_score(dataagg, labels, metric='euclidean')
    scores.append(score)
    print(f"Silhouette score for n_clusters={n_clusters}: {score}")

# Plot the silhouette scores
#plt.plot(range(2, 10), scores)
#plt.xlabel('Number of Clusters')
#plt.ylabel('Silhouette Score')
#plt.show()

agg_clustering = AgglomerativeClustering(n_clusters=2,linkage='ward')
agg_clustering.fit(dataagg)
labels = agg_clustering.labels_
score = silhouette_score(dataagg, labels, metric='euclidean')
print(f"Silhouette score for 2 clusters: {score}")

# Visualize the clusters using PCA
pca = PCA(n_components=2)
pca_data = pca.fit_transform(dataagg)
plt.scatter(pca_data[:,0], pca_data[:,1], c=labels)
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.show()

y_true = data['Gender']
acc = accuracy_score(y_true, labels)
print('Accuracy:', acc)
```

```
prec = precision_score(y_true, labels)
print('Precision:', prec)

rec = recall_score(y_true, labels)
print('Recall:', rec)

f1 = f1_score(y_true, labels)
auc_roc = roc_auc_score(y_true, labels)
cm = confusion_matrix(y_true, labels)

# Print the evaluation metrics and confusion matrix
print('F1 score:', f1)
print('AUC-ROC score:', auc_roc)
```

```
Silhouette score for n_clusters=2: 0.41977319055009615
Silhouette score for n_clusters=3: 0.3855875245491068
Silhouette score for n_clusters=4: 0.30695498132585075
Silhouette score for n_clusters=5: 0.3019675712509879
Silhouette score for n_clusters=6: 0.2820973654945771
Silhouette score for n_clusters=7: 0.2802042286754983
Silhouette score for n_clusters=8: 0.263158553932542
Silhouette score for n_clusters=9: 0.24902790363677355
Silhouette score for 2 clusters: 0.41977319055009615
```




```
Accuracy: 0.5108910891089109
Precision: 0.4861111111111111
Recall: 0.2880658436213992
F1 score: 0.3617571059431524
AUC-ROC score: 0.502811547764898
```