Lab Assignment 1

Member 1

Name : Sireesha Keesara

Class Id : 14

Member 2

Name : Kavin Kumar Arumugam

Class Id : 3

Objectives

- Take a list of tuples as follows: [(‘John’, (‘Physics’, 80)) , (‘ Daniel’, (‘Science’, 90)), (‘John’, (‘Science’, 95)), (‘Mark’,(‘Maths’, 100)), (‘Daniel’, (’History’, 75)), (‘Mark’, (‘Social’, 95))] Create a dictionary with keys as names and values as list of (subjects, marks)in sorted order.
- Given a string, find the longest substrings without repeating characters along with the length as a tuple
  Input: "pwwkew"
  Output: (wke,3), (kew,3)
- Airline Booking Reservation System (e.g. classes Flight, Person, Employee, Passenger etc.) Inheritance at least once.Should have one super call. Use at least one private data member in your code.Create instances of all classes and show the relationship between them.
- Create Multiple Regression by choosing a dataset of your choice (again before evaluating, clean the data set with the EDA learned in the class). Evaluate the model using RMSE and R2 and also report if you saw any improvement before and after the EDA.

- Pick any dataset from the dataset sheet in the class sheet or online which includes both numeric and non-numeric features
  a. Perform exploratory data analysis on the data set (like Handling null values, removing the features not correlated to the target class, encoding the categorical features, ...)
  b. Apply the three classification algorithms Naïve Baye's, SVM and KNN on the chosen data set and report which classifier gives a better result.
- Apply K-means on the dataset(College.csv) and visualize the clusters using matplotlib or seaborn. Report which K is the best using the elbow method. Evaluate with silhouette score or other scores relevant for unsupervised approaches (before
  applying clustering clean the data set with the EDA learned in the class)

To convert List of tuples into sorted dictionary.

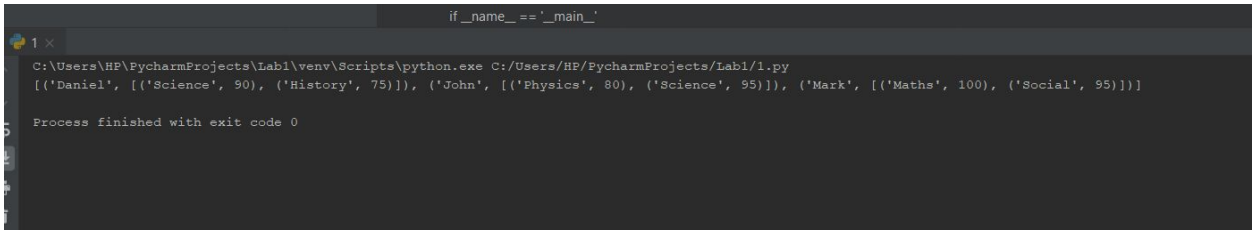
Source Code:

- Take list of tuples as input.
- Initialize a defaultdict which accepts list as a function.
- loop though each item in the list.
- If the key value is already present in the dictionary append the current value to that key.
- If not then add the new key value pair to the dictionary.

- Sort the dictionary and print it.

```
from collections import defaultdict
import operator

def tuple_convert():
    list1 = [('John', ('Physics', 80)), ('Daniel', ('Science', 90)),
    ('John', ('Science', 95)), ('Mark', ('Maths', 100)),
    ('Daniel', ('History', 75)), ('Mark', ('Social', 95))]
    # intialize the dict
    d = defaultdict(list)
    # Iterate through the values and assign it to the same key
    for k, v in list1:
        d[k].append(v)
    # Sort the dictionary on basis of key
    sorted_dict = sorted(d.items(), key=operator.itemgetter(0))
    print(sorted_dict)



if __name__ == '__main__':
    tuple_convert()
```

Output:



Find longest substrings

Algorithm

- get the input string from the user and store it in a variable.
- for a character in the input string

- add character to the map if there is no existing key as a character in map or else increase the value.
- find the highest value in the hashmap.
- add the highest value to answer list.

Source Code

```
input_string = input("enter string : ")
temp_string = ""
dict = {}
for charecter in range(len(input_string)):
    for i in range(charecter, len(input_string)):
        if not (input_string[i] in temp_string):
            temp_string += input_string[i]
        else:
            dict[temp_string] = len(temp_string)
            temp_string = ''
            break
highest_val = max(dict.values())

answer = []
for key, val in dict.items():
    if highest_val == val:
        answer.append((key, val))
print(answer)
```

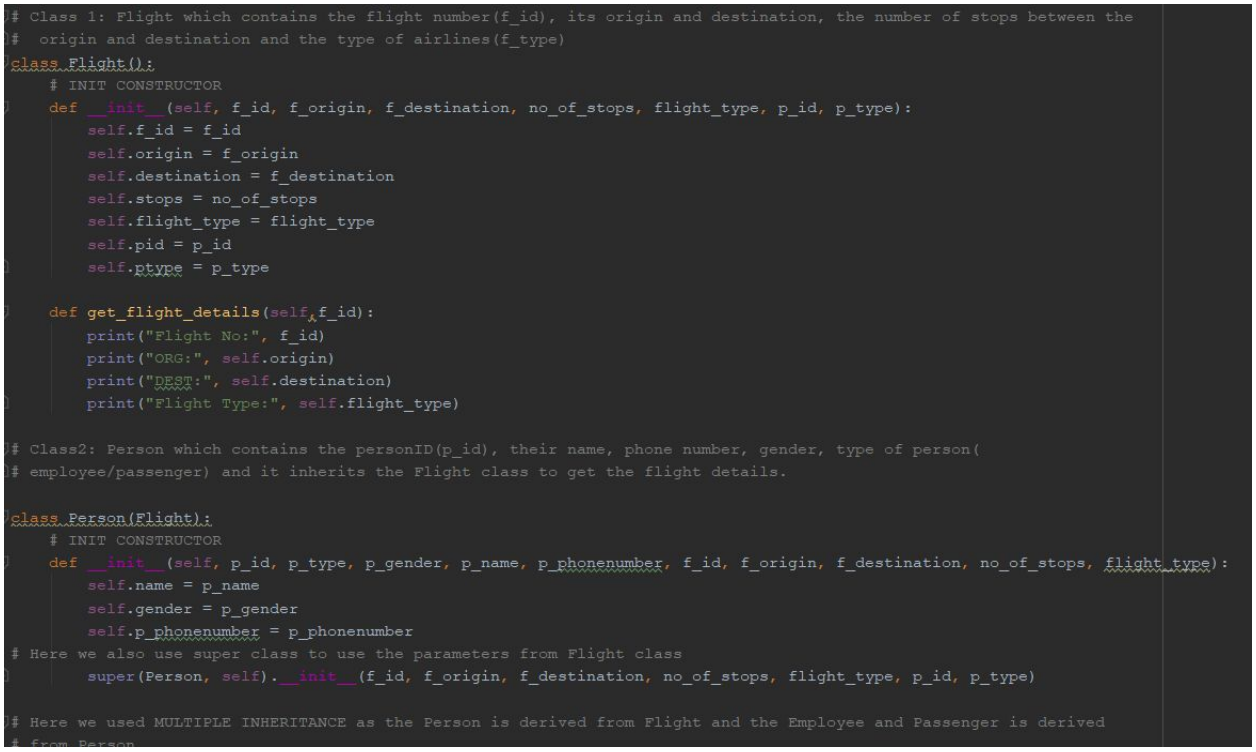
Output

```
enter string : pwwkew
[('wke', 3), ('kew', 3)]
```

Inheritance

- Created Five Classes Flight, Person, Passenger, Employee and Ticket.
- Person Class is inherited from Flight and Passenger class and Employee Classes are inherited from Person.
- Created instances for Flight , Passenger, Employee and Ticket.

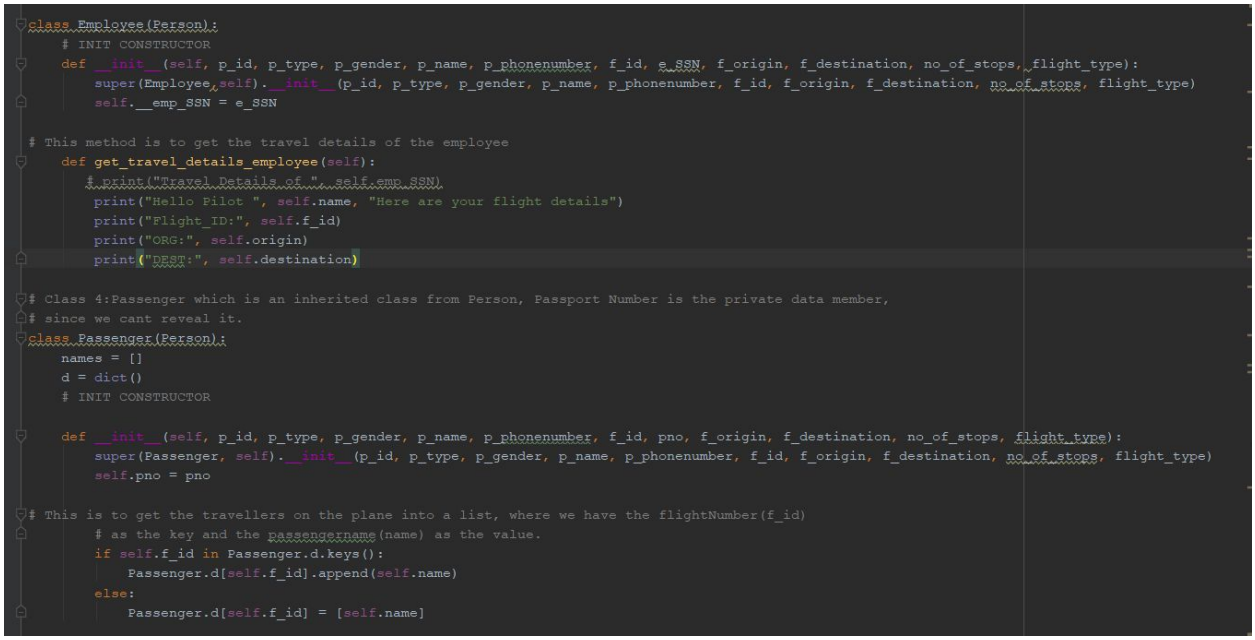
Source Code:



```
# Class 1: Flight which contains the flight number(f_id), its origin and destination, the number of stops between the
#  origin and destination and the type of airlines(f_type)
class Flight():
    # INIT CONSTRUCTOR
    def __init__(self, f_id, f_origin, f_destination, no_of_stops, flight_type, p_id, p_type):
        self.f_id = f_id
        self.origin = f_origin
        self.destination = f_destination
        self.stops = no_of_stops
        self.flight_type = flight_type
        self.pid = p_id
        self.ptype = p_type

    def get_flight_details(self,f_id):
        print("Flight No:", f_id)
        print("ORG:", self.origin)
        print("DEST:", self.destination)
        print("Flight Type:", self.flight_type)

# Class2: Person which contains the personID(p_id), their name, phone number, gender, type of person(
# employee/passenger) and it inherits the Flight class to get the flight details.

class Person(Flight):
    # INIT CONSTRUCTOR
    def __init__(self, p_id, p_type, p_gender, p_name, p_phonenumber, f_id, f_origin, f_destination, no_of_stops, flight_type):
        self.name = p_name
        self.gender = p_gender
        self.p_phonenumber = p_phonenumber
# Here we also use super class to use the parameters from Flight class
        super(Person, self).__init__(f_id, f_origin, f_destination, no_of_stops, flight_type, p_id, p_type)

# Here we used MULTIPLE INHERITANCE as the Person is derived from Flight and the Employee and Passenger is derived
# from Person
```

- Function get_tavel_details_employee inherited from Person Class to display the travel details of the Employee in Employee Class
- Passenger Class inherited from Person(Inherited from Flight class - Multiple Inheritance).
- In the Passenger class function get_travel_details_passenger used to display the details of the passenger.

- get_travelling_details to display the passengers in a particular Flight.



```
class Employee(Person):
    # INIT CONSTRUCTOR
    def __init__(self, p_id, p_type, p_gender, p_name, p_phonenumber, f_id, e_SSN, f_origin, f_destination, no_of_stops, flight_type):
        super(Employee,self).__init__(p_id, p_type, p_gender, p_name, p_phonenumber, f_id, f_origin, f_destination, no_of_stops, flight_type)
        self.__emp_SSN = e_SSN

# This method is to get the travel details of the employee
    def get_travel_details_employee(self):
        # print("Travel Details of ", self.emp_SSN)
        print("Hello Pilot ", self.name, "Here are your flight details")
        print("Flight_ID:", self.f_id)
        print("ORG:", self.origin)
        print("DEST:", self.destination)

# Class 4:Passenger which is an inherited class from Person, Passport Number is the private data member,
# since we cant reveal it.
class Passenger(Person):
    names = []
    d = dict()
    # INIT CONSTRUCTOR

    def __init__(self, p_id, p_type, p_gender, p_name, p_phonenumber, f_id, pno, f_origin, f_destination, no_of_stops, flight_type):
        super(Passenger, self).__init__(p_id, p_type, p_gender, p_name, p_phonenumber, f_id, f_origin, f_destination, no_of_stops, flight_type)
        self.pno = pno

# This is to get the travellers on the plane into a list, where we have the flightNumber(f_id)
        # as the key and the passengername(name) as the value.
        if self.f_id in Passenger.d.keys():
            Passenger.d[self.f_id].append(self.name)
        else:
            Passenger.d[self.f_id] = [self.name]
```

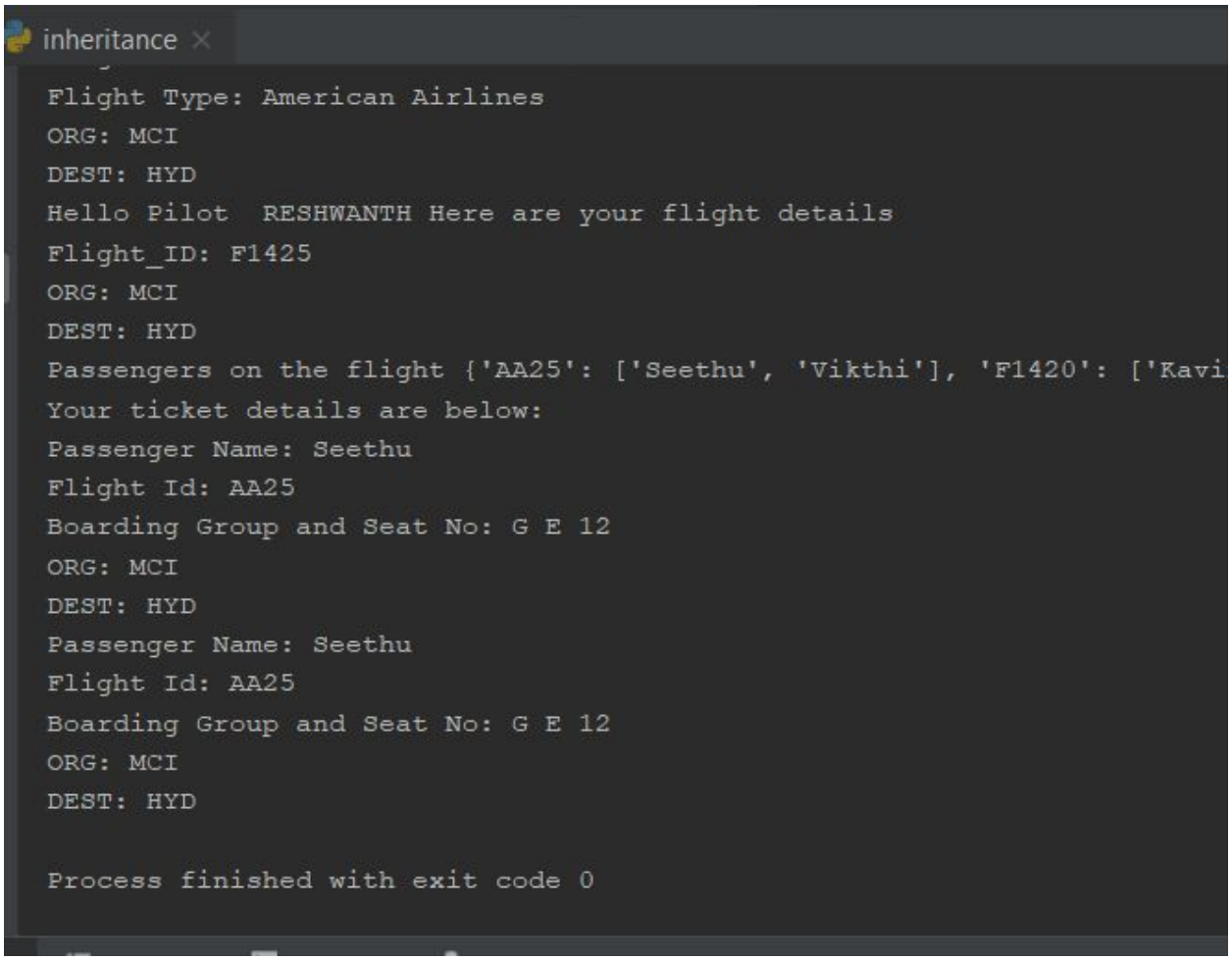
- Function get_boarding_pass is inherited from passenger class to display the ticket details of the passenger.

```
class Ticket(Passenger):
    def __init__(self, p_id, p_type, p_gender, p_name, p_phonenumber, f_id, pno, f_origin, f_destination, no_of_stops,
                 flight_type, boarding_group_no, row, seat_no):
        super(Ticket, self).__init__(p_id, p_type, p_gender, p_name, p_phonenumber, f_id, pno, f_origin, f_destination,
                                     no_of_stops, flight_type)
        self.boarding_group_no = boarding_group_no
        self.row = row
        self.seat_no = seat_no
        print("Your ticket details are below: ")

    def get_boarding_pass(self, p_name):
        for k, v in Passenger.d.items():
            names = v
            for i in names:
                if i == p_name:
                    print("Passenger Name:", p_name)
                    print("Flight Id:", k)
                    print("Boarding Group and Seat No:", self.boarding_group_no, self.row, self.seat_no)
                    print("ORG:", self.origin)
                    print("DEST:", self.destination)
```

airline_booking.py | K-means.py | inheritance.py

```
from airline_booking import Passenger, Employee, Ticket, Flight

# p_id,p_type,p_gender,p_name,p_phonenumber,f_id,pno,f_origin,f_destination,no_of_stops,flight_type
#f_id, f_origin, f_destination, no_of_stops, flight_type, p_id, p_type
f1 = Flight("AA25", "HYD", "MCI", 1, "AMERICAN AiIRLINES", "P1", "P")
f1.get_flight_details("AA25")

#Instances of Passenger class
p1 = Passenger("P1", "P", "F", "Seethu", 8167390291, "AA25", "12345", "MCI", "HYD", 3, "American Airlines")
p2 = Passenger("P2", "P", "M", "Vikthi", 816564738, "AA25", "98123", "MCI", "HYD", 3, "American Airlines")
p3 = Passenger("P3", "P", "F", "Kavin", 7385672324, "F1420", "A123", "MCI", "BLR", 3, "QATAR")

#Instances of Employee class
e1 = Employee("E1", "E", "M", "RESHWANTH", 142587961558, "F1425", "A123", "MCI", "HYD", 3, "American Airlines")
e2 = Employee("E2", "E", "M", "VINAY", 422587961558, "F1425", "A123", "MCI", "HYD", 3, "American Airlines")
e3 = Employee("E2", "E", "M", "ASHISH", 424587961558, "F1420", "A123", "MCI", "BLR", 3, "QATAR")

#This method prints the travel details for the passenger
p1.get_travel_details_passanger()
#This method prints the travel details for the employee
e1.get_travel_details_employee()

#This method prints the travelling passengers on that flight
p1.get_travelling_passengers()

#Prints the boarding pass
T1 = Ticket("P1", "P", "F", "Seethu", 8167390291, "AA25", "12345", "MCI", "HYD", 3, "American Airlines", "G", "E", 12)
T1.get_boarding_pass("Seethu")
```

Output:


inheritance
Flight Type: American Airlines
ORG: MCI
DEST: HYD
Hello Pilot RESHWANTH Here are your flight details
Flight_ID: F1425
ORG: MCI
DEST: HYD
Passengers on the flight {'AA25': ['Seethu', 'Vikthi'], 'F1420': ['Kavi
Your ticket details are below:
Passenger Name: Seethu
Flight Id: AA25
Boarding Group and Seat No: G E 12
ORG: MCI
DEST: HYD
Passenger Name: Seethu
Flight Id: AA25
Boarding Group and Seat No: G E 12
ORG: MCI
DEST: HYD
Process finished with exit code 0

Multiple Regression

Output and Algorithm

- load dataset using read_csv function from pandas library
- print dataset head

	Unnamed: 0	Private	Apps	Accept	Enroll	Top10perc	Top25perc
0	Abilene Christian University	Yes	1660	1232	721	23	52
1	Adelphi University	Yes	2186	1924	512	16	29
2	Adrian College	Yes	1428	1097	336	22	50
3	Agnes Scott College	Yes	417	349	137	60	89
4	Alaska Pacific University	Yes	193	146	55	16	44

- to see shape of the college dataset
  (777, 19)

- to see datatypes of the columns in the dataset

```
Unnamed: 0      object
Private         object
Apps             int64
Accept           int64
Enroll           int64
Top10perc        int64
Top25perc        int64
F.Undergrad      int64
P.Undergrad      int64
Outstate         int64
Room.Board       int64
Books            int64
Personal         int64
PhD              int64
Terminal         int64
S.F.Ratio      float64
perc.alumni      int64
Expend           int64
Grad.Rate        int64
dtype: object
```

- drop columns having datatypes as object and target
- print dataset head

	Apps	Accept	Enroll	Top10perc	Top25perc	F.Undergrad	P.Und
0	1660	1232	721	23	52	2885	
1	2186	1924	512	16	29	2683	
2	1428	1097	336	22	50	1036	
3	417	349	137	60	89	510	
4	193	146	55	16	44	249	

- to see shape of the college dataset
  (777, 16)

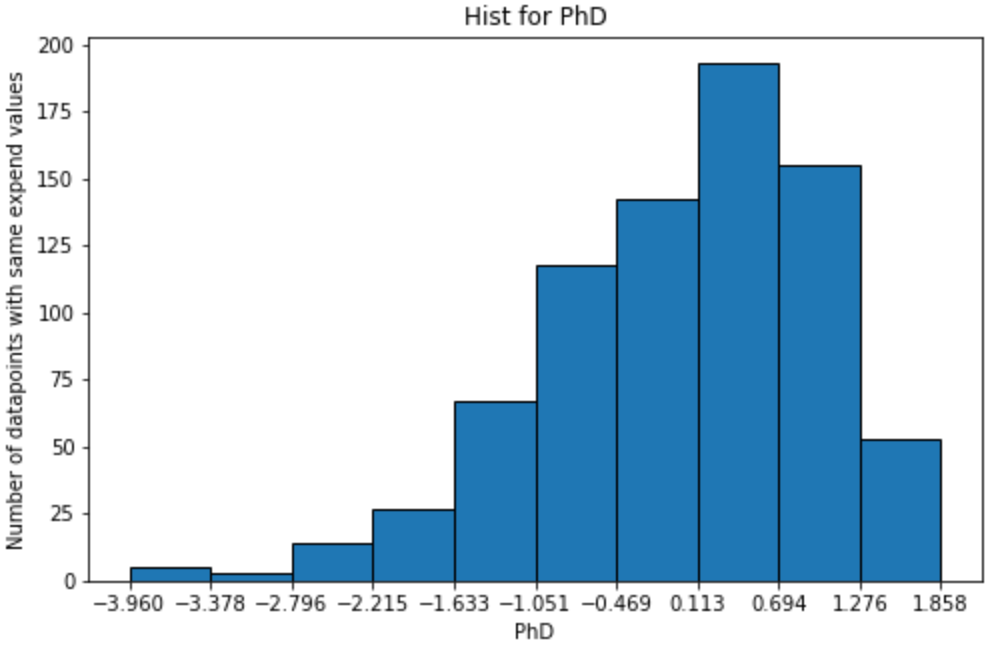
- to see datatypes of the columns in the dataset

```
Apps             int64
Accept           int64
Enroll           int64
Top10perc        int64
Top25perc        int64
F.Undergrad      int64
P.Undergrad      int64
Outstate         int64
Room.Board       int64
Books            int64
Personal         int64
PhD              int64
Terminal         int64
S.F.Ratio      float64
perc.alumni      int64
Expend           int64
dtype: object
```

- count number of null in the dataset
- change the values in the columns with respect to mean and standard deviation
- print dataset head

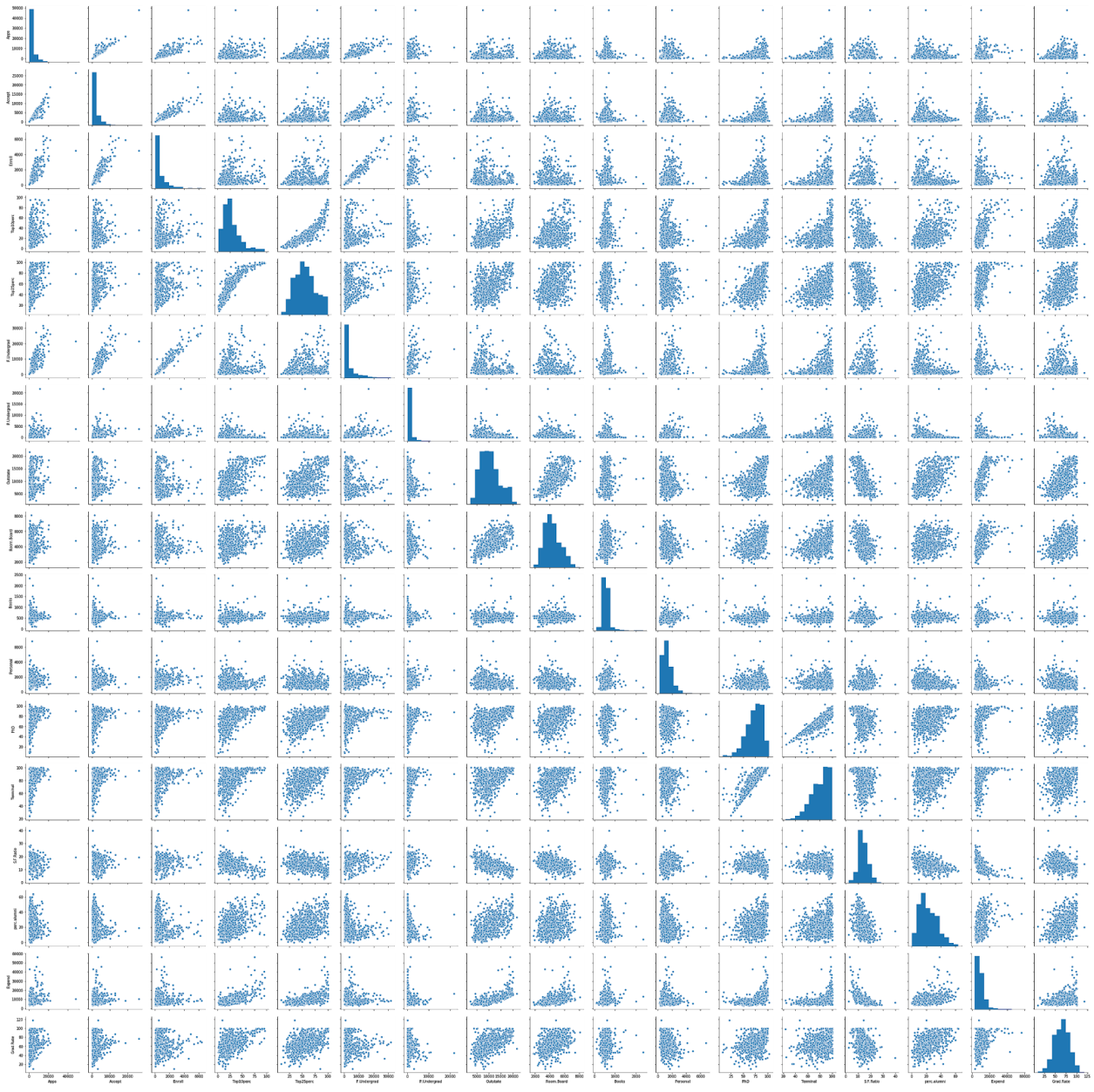
ɔoks	Personal	PhD	Terminal	S.F.Ratio	perc.alumni	Expend
450	2200	-0.162923	78	18.1	12	-0.501587
750	1500	-2.673923	30	12.2	16	0.166003
400	1165	-1.204069	66	12.9	30	-0.177176
450	875	1.184443	97	7.7	37	1.791697
800	1500	0.204540	72	11.9	2	0.241648

- plot the histogram



- find correlation between all the features

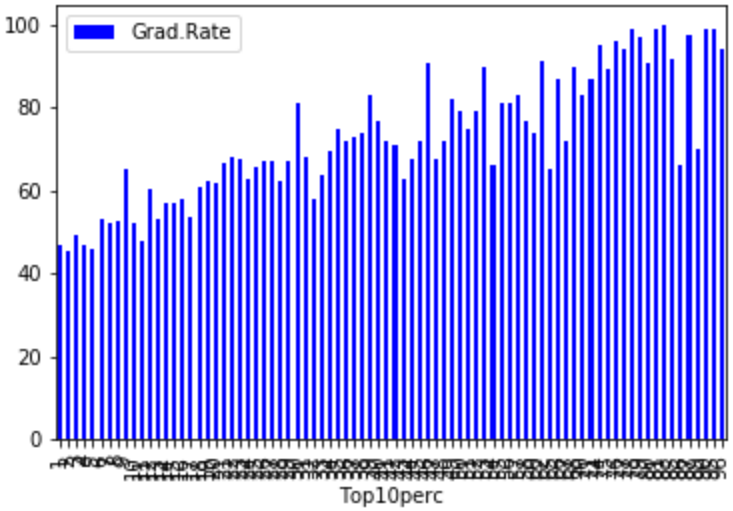
- do pairplot on the features



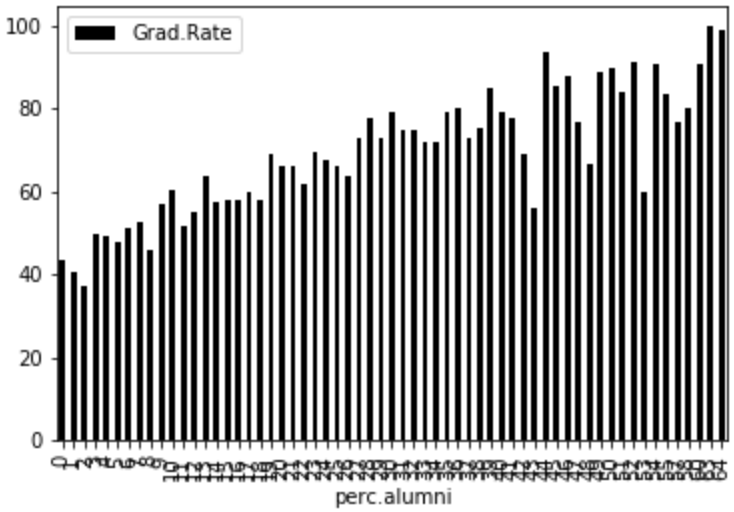
- find correlation

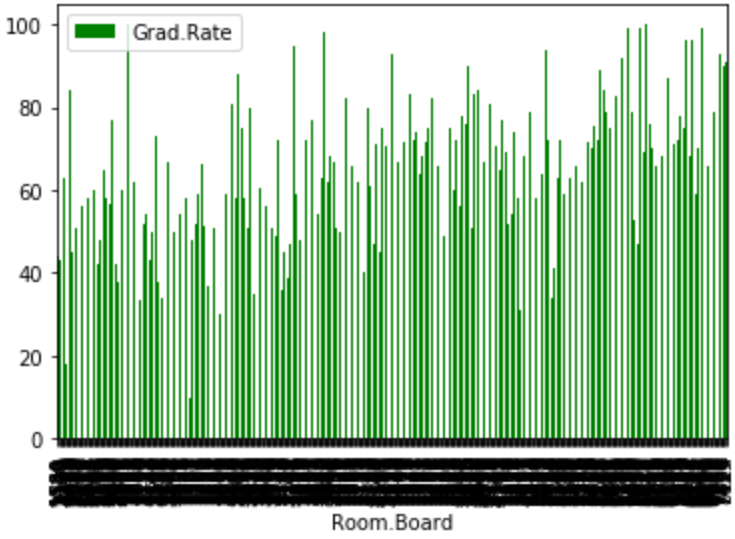
```
Grad.Rate      1.000000
Outstate       0.571290
Top10perc      0.494989
perc.alumni    0.490898
Top25perc      0.477281
Room.Board     0.424942
Expend         0.390343
PhD            0.305038
Name: Grad.Rate, dtype: flo

Apps           0.146755
Accept         0.067313
Books          0.001061
Enroll        -0.022341
F.Undergrad   -0.078773
```

- remove columns with negative correlation



- visualize the features





- create traget dataframe
- split dataframe to train and test
- train model
- model accuracy

```
R2 is:  0.3483968805083466
RMSE:  0.056778297586559 84
```

Classification Algorithms

Output and Algorithm

- import libraries
- read csv using pandas

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure
0	7590-VHVEG	Female	0	Yes	No	1
1	5575-GNVDE	Male	0	No	No	34
2	3668-QPYBK	Male	0	No	No	2
3	7795-CFOCW	Male	0	No	No	45
4	9237-HQITU	Female	0	No	No	2

- dataframe shape
  (7043, 21)

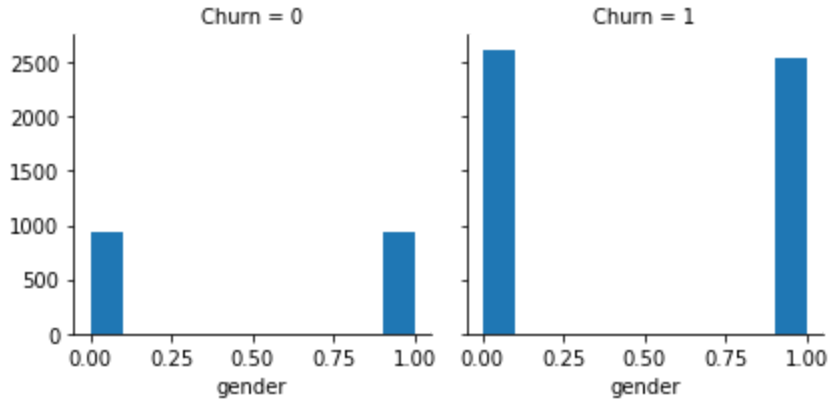
- dataset datatypes

```
customerID          object
gender              object
SeniorCitizen        int64
Partner             object
Dependents          object
tenure               int64
PhoneService        object
MultipleLines       object
InternetService     object
OnlineSecurity      object
OnlineBackup        object
DeviceProtection    object
TechSupport         object
StreamingTV         object
StreamingMovies     object
Contract            object
PaperlessBilling    object
PaymentMethod       object
MonthlyCharges     float64
TotalCharges         object
Churn                object
```

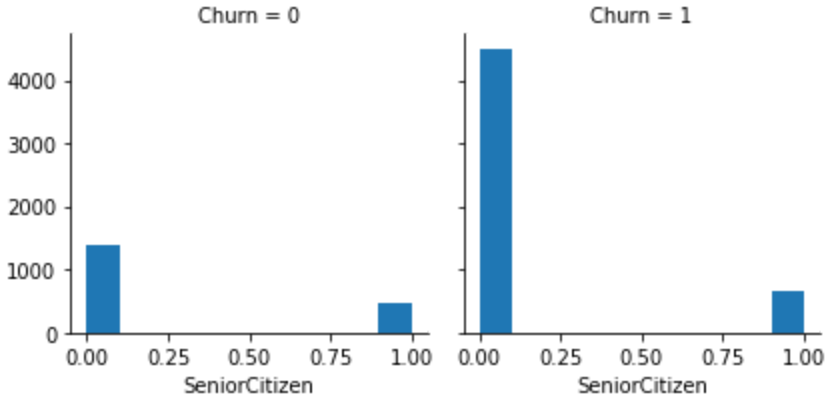
- select only object data type
- find whether any feature has null values

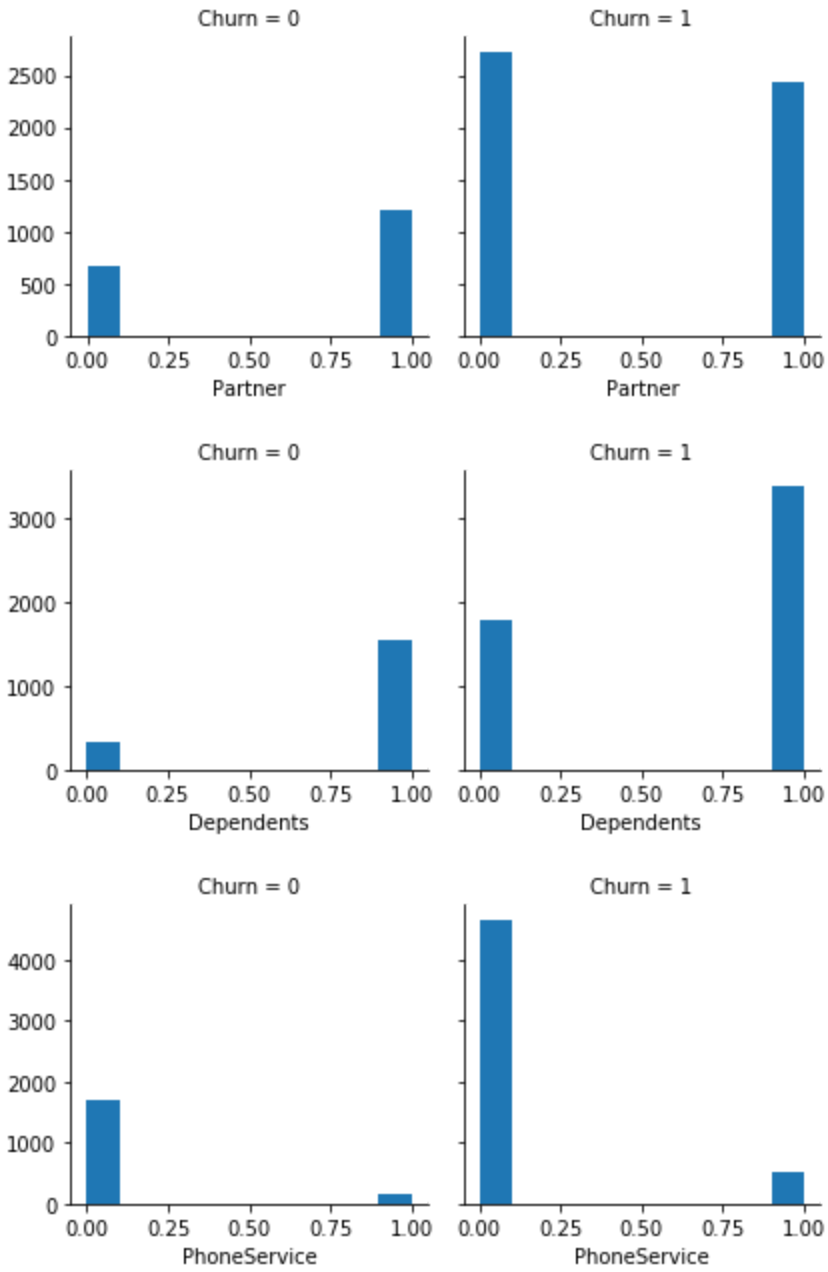
- encode the features which is having multivariate variables

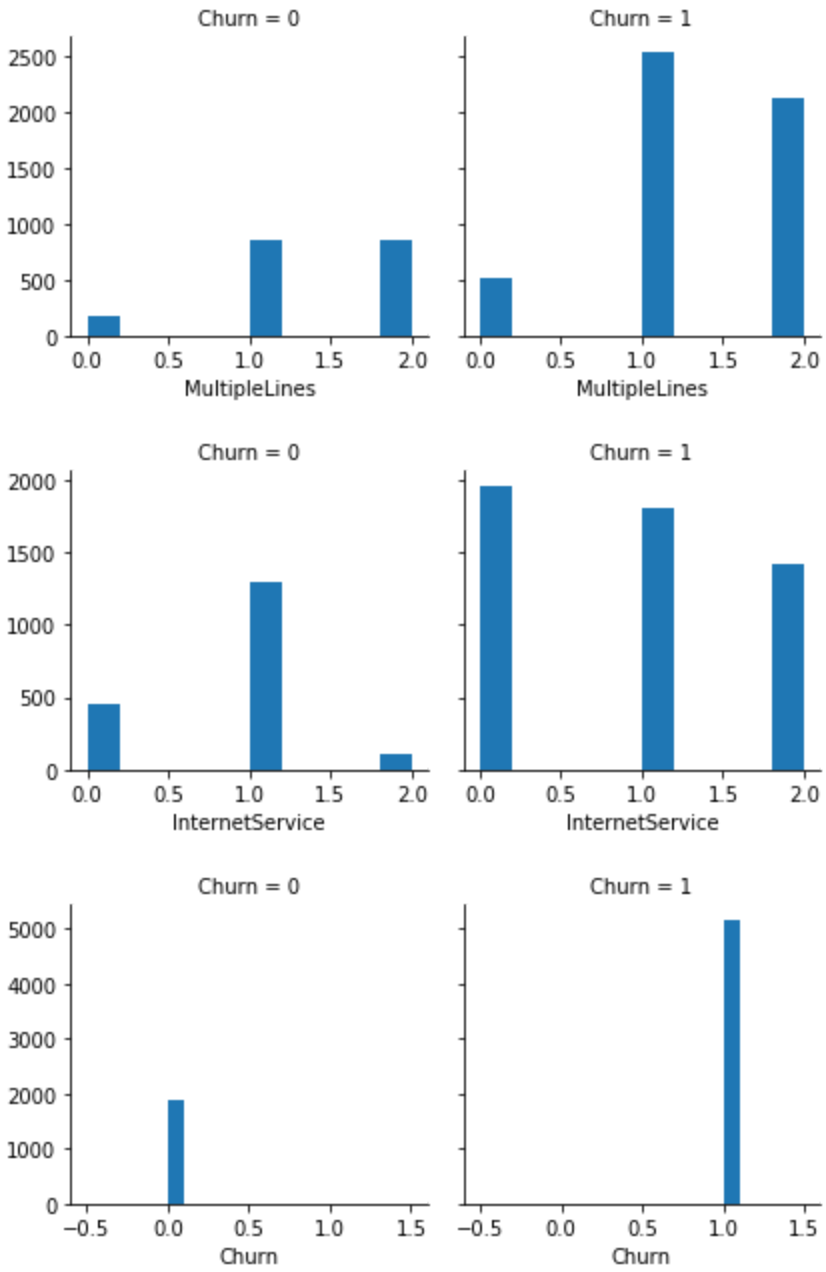
	customerID	gender	SeniorCitizen	Partner	Dependents	tenure
0	7590-VHVEG	1	0	0	1	1
1	5575-GNVDE	0	0	1	1	34
2	3668-QPYBK	0	0	1	1	2
3	7795-CFOCW	0	0	1	1	45
4	9237-HQITU	1	0	1	1	2

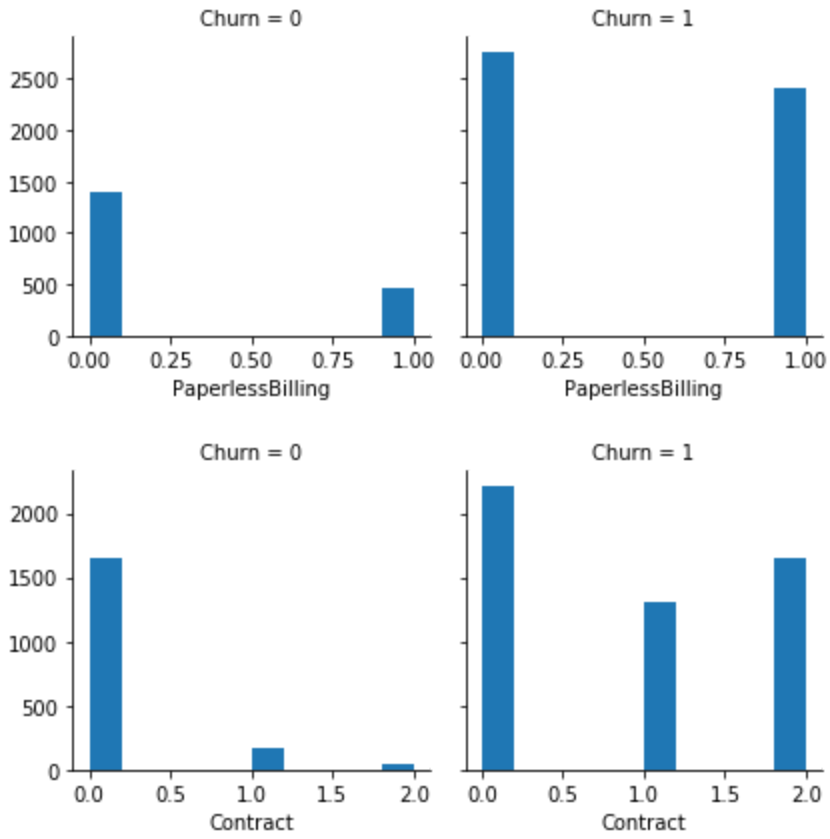


- visualize data




Churn = 0
Churn = 1
2500
2000
1500
1000
500
0
0.00
0.25
0.50
0.75
1.00
Partner
Churn = 0
Churn = 1
3000
2000
1000
0
0.00
0.25
0.50
0.75
1.00
Dependents
Churn = 0
Churn = 1
4000
3000
2000
1000
0
0.00
0.25
0.50
0.75
1.00
PhoneService


Churn = 0
Churn = 1
MultipleLines
MultipleLines
Churn = 0
Churn = 1
InternetService
InternetService
Churn = 0
Churn = 1
Churn
Churn



- KNN

```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=2)
knn.fit(X_train, y_train)
```

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, met
ric='minkowski',
           metric_params=None, n_jobs=None, n_neighbors=
2, p=2,
           weights='uniform')
```

```
print('Accuracy on training set: {:.2f}'.format(knn.score
print('Accuracy on test set: {:.2f}'.format(knn.score(X_te
```

```
Accuracy on training set: 0.79
Accuracy on test set: 0.67
```

- Support Vector Machine

```
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma='auto_d
eprecated',
  kernel='rbf', max_iter=-1, probability=False, random_s
tate=None,
  shrinking=True, tol=0.001, verbose=False)
```

```
print('Accuracy on training set: {:.2f}'.format(svm.score
# test data set acc
print('Accuracy on test set: {:.2f}'.format(svm.score(X_te
```

```
Accuracy on training set: 0.78
Accuracy on test set: 0.78
```

- GaussianNB

```
from sklearn.naive_bayes import GaussianNB
GNB = GaussianNB()
GNB.fit(X_train, y_train)
```

```
GaussianNB(priors=None, var_smoothing=1e-09)
```

```
print('Accuracy on training set: {:.2f}'.format(GNB.score
# test data set acc
print('Accuracy on test set: {:.2f}'.format(GNB.score(X_te
```

```
Accuracy on training set: 0.70
Accuracy on test set: 0.70
```

K-Means

Source Code:

```
import pandas as pd
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
import seaborn as sns
from pathlib import Path
sns.set(style="white", color_codes=True)
import warnings
warnings.filterwarnings("ignore")

# Import the data into a dataframe
df = pd.read_csv(Path('./College.csv'))
# To check the number of values in the private column
print(df["Private"].value_counts())
# check for the null values if they are present are not
nulls = pd.DataFrame(df.isnull().sum().sort_values(ascending=False)[:25])
nulls.columns= ['Null Count']
nulls.index.name= 'Feature'
print(nulls)
```

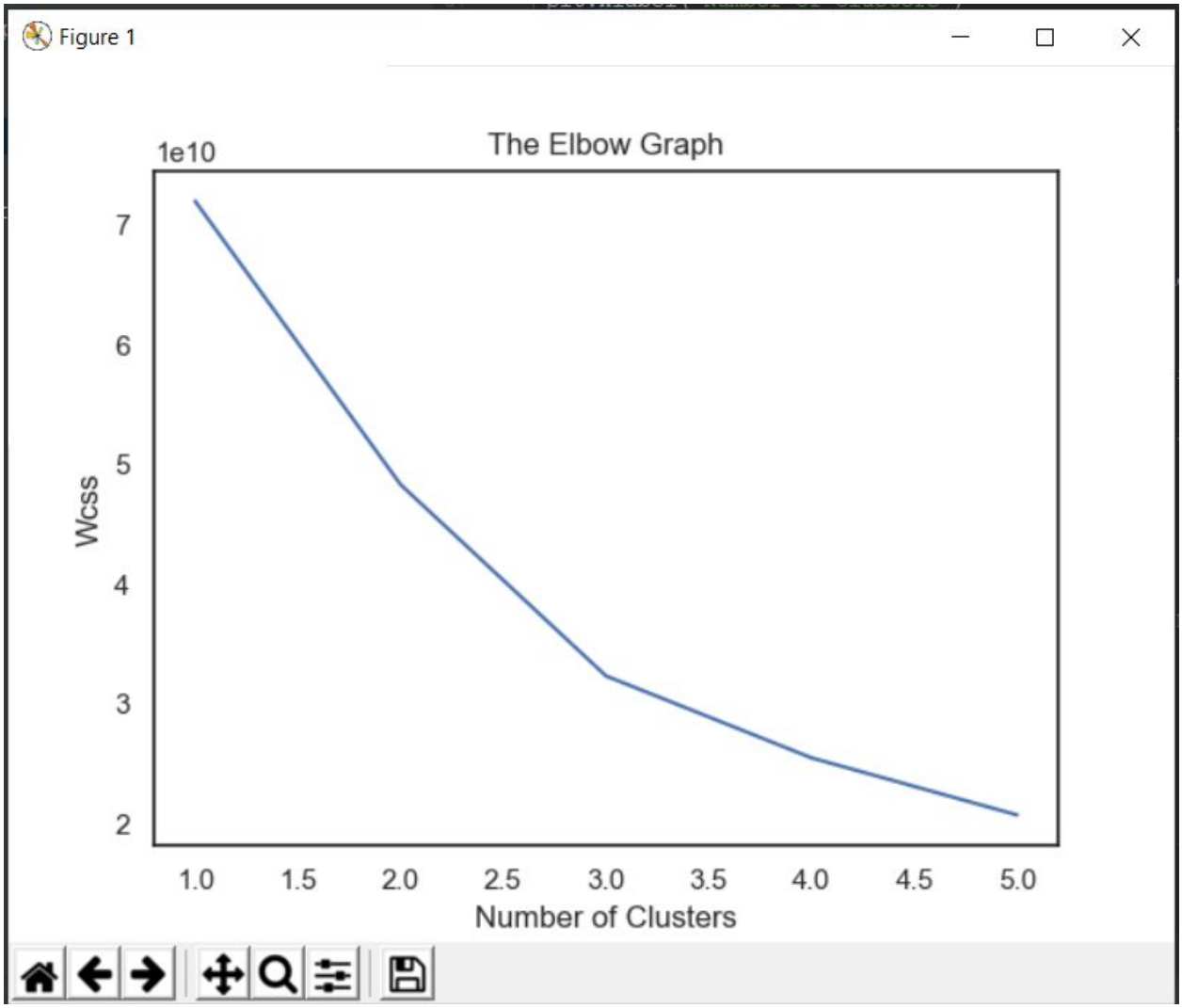
```
# Divide the dependent and independent data
y = df.iloc[:, 1:2]
x = df.iloc[:, 2:]
print(x.shape, y.shape)

# Elbow method is used for finding the optimal nnumber of clusters
wcss = []
for i in range(1, 6):
    km = KMeans(n_clusters=i, init='k-means++', max_iter=300, n_init=10, random_state=0)
    km.fit(x)
    wcss.append(km.inertia_)

plt.plot(range(1,6),wcss)
plt.title('The Elbow Graph')
plt.xlabel('Number of Clusters')
plt.ylabel('Wcss')
plt.show()
```

```
# Apply K mean clustering with no of clusters = 2
km = KMeans(n_clusters=2)
km.fit(x)
y_cluster_kmeans = km.predict(x)
from sklearn import metrics
score = metrics.silhouette_score(x, y_cluster_kmeans)
print("The silhouette_score is: ", score)

plt.scatter(x.iloc[:,10],x.iloc[:,5], c=y_cluster_kmeans[:,], s=30, cmap='viridis')

# Standardiazation is normalized and the accuracy score is falling down.
# standardization
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
# Fit on training set only.
sc.fit(x)
# Apply transform to both the training set and the test set.
x_scaler = sc.transform(x)

# Apply PCA on the data with dimension reduction to 2 axis
pca = PCA(2)
x_pca = pca.fit_transform(x)
df2 = pd.DataFrame(data=x_pca)
finaldf = pd.concat([df2, df[['Private']]], axis=1)
print(finaldf)

# KMeans after PC
km = KMeans(n_clusters=3)
km.fit(x_pca)
y_cluster_kmeans= km.predict(x_pca)
from sklearn import metrics
score = metrics.silhouette_score(x_pca, y_cluster_kmeans)
print("The silhouette_score after PCA is",score)

# Plotting the clusters
plt.scatter(x.iloc[:,10],x.iloc[:,5], c=y_cluster_kmeans[:,], s=50, cmap='viridis')
plt.show()
```

PEP 8: missing whitespace after ','

Output:

Elbow Graph



Clustering(K=2)

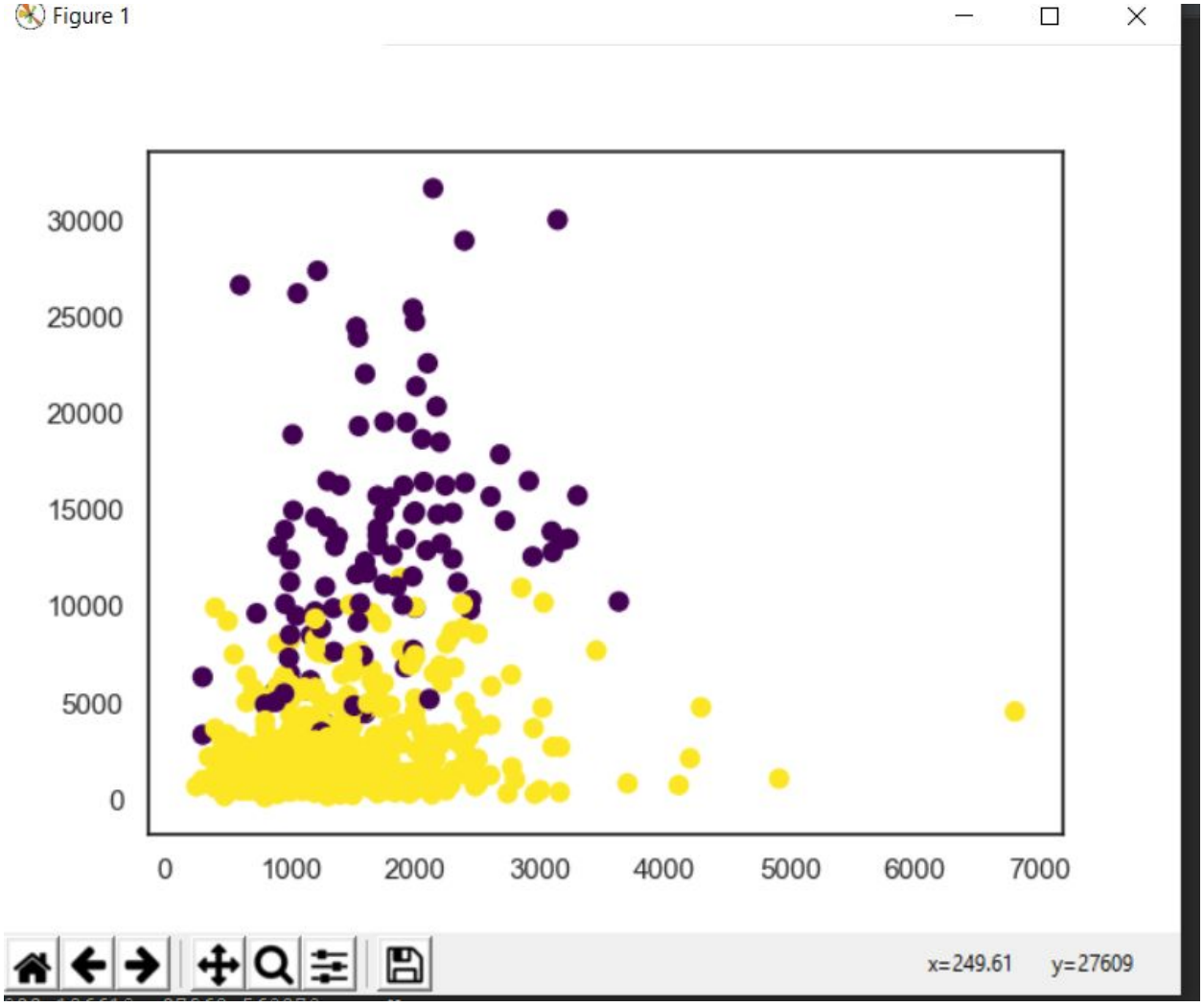


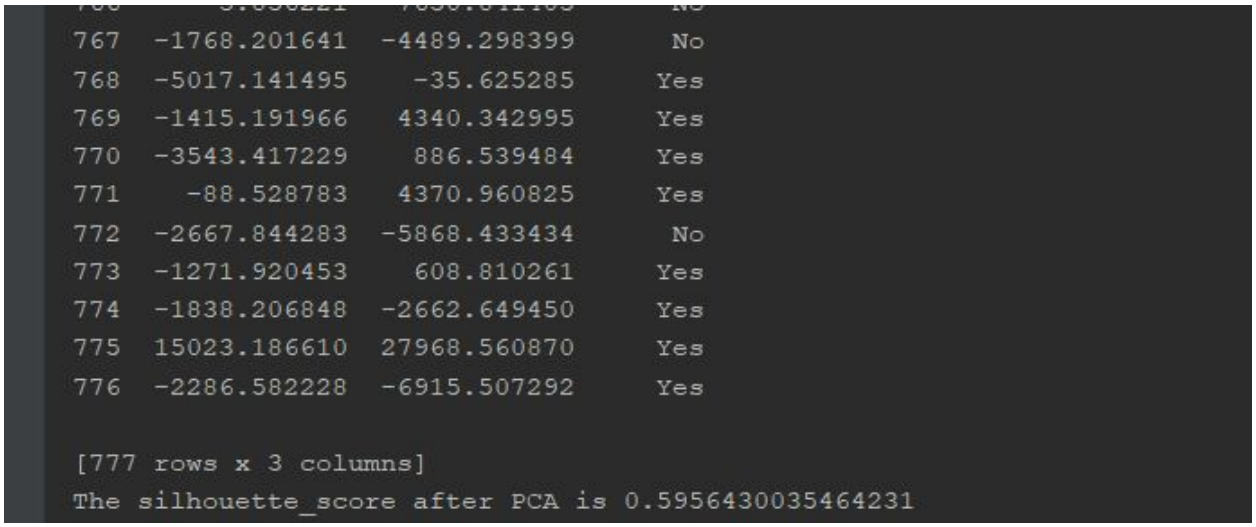
Figure 1
30000
25000
20000
15000
10000
5000
0
0
1000
2000
3000
4000
5000
6000
7000
x=249.61
y=27609

```
C:\Users\HP\PycharmProjects\Lab1\venv\Scripts\python.exe C:/Users/HP/Pyc
Yes    565
No     212
Name: Private, dtype: int64
             Null Count
Feature
Grad.Rate             0
P.Undergrad           0
Private               0
Apps                  0
Accept                0
Enroll                0
Top10perc             0
Top25perc             0
F.Undergrad           0
Outstate              0
Expend                0
Room.Board            0
Books                 0
Personal              0
PhD                   0
Terminal              0
S.F.Ratio             0
perc.alumni           0
Unnamed: 0            0
(777, 17) (777, 1)
```

Score before PCA

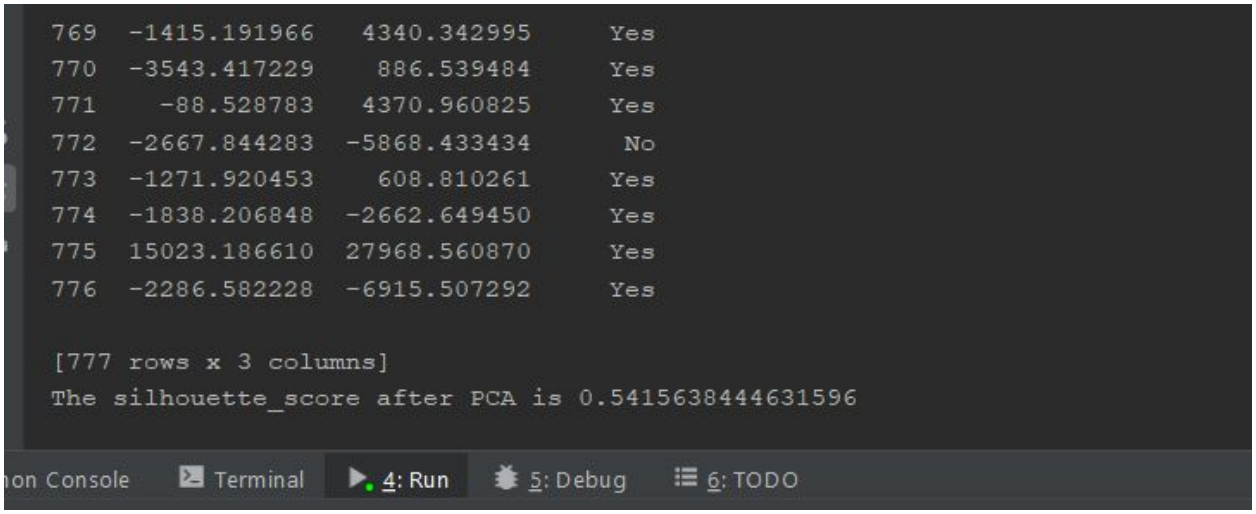
```
S.F.Ratio             0
perc.alumni           0
Unnamed: 0            0
(777, 17) (777, 1)
The silhouette_score is:  0.5599267817640777
                  0             1 Private
0     -2551.837861  -3445.947204     Yes
1      -743.729533   2227.363556     Yes
```

Score after PCA(K = 2)



```
767  -1768.201641  -4489.298399      No
768  -5017.141495    -35.625285     Yes
769  -1415.191966   4340.342995     Yes
770  -3543.417229    886.539484     Yes
771    -88.528783   4370.960825     Yes
772  -2667.844283  -5868.433434      No
773  -1271.920453    608.810261     Yes
774  -1838.206848  -2662.649450     Yes
775  15023.186610  27968.560870     Yes
776  -2286.582228  -6915.507292     Yes

[777 rows x 3 columns]
The silhouette_score after PCA is 0.5956430035464231
```

Score after PCA(K = 3)



```
769  -1415.191966   4340.342995     Yes
770  -3543.417229    886.539484     Yes
771    -88.528783   4370.960825     Yes
772  -2667.844283  -5868.433434      No
773  -1271.920453    608.810261     Yes
774  -1838.206848  -2662.649450     Yes
775  15023.186610  27968.560870     Yes
776  -2286.582228  -6915.507292     Yes

[777 rows x 3 columns]
The silhouette_score after PCA is 0.5415638444631596
```

on Console Terminal 4: Run 5: Debug 6: TODO

As the score is decreased with k = 3 from 55 to 54 % and when k =2 its increased fro 55 to 59 %. So K= 2 is the best using elbow method