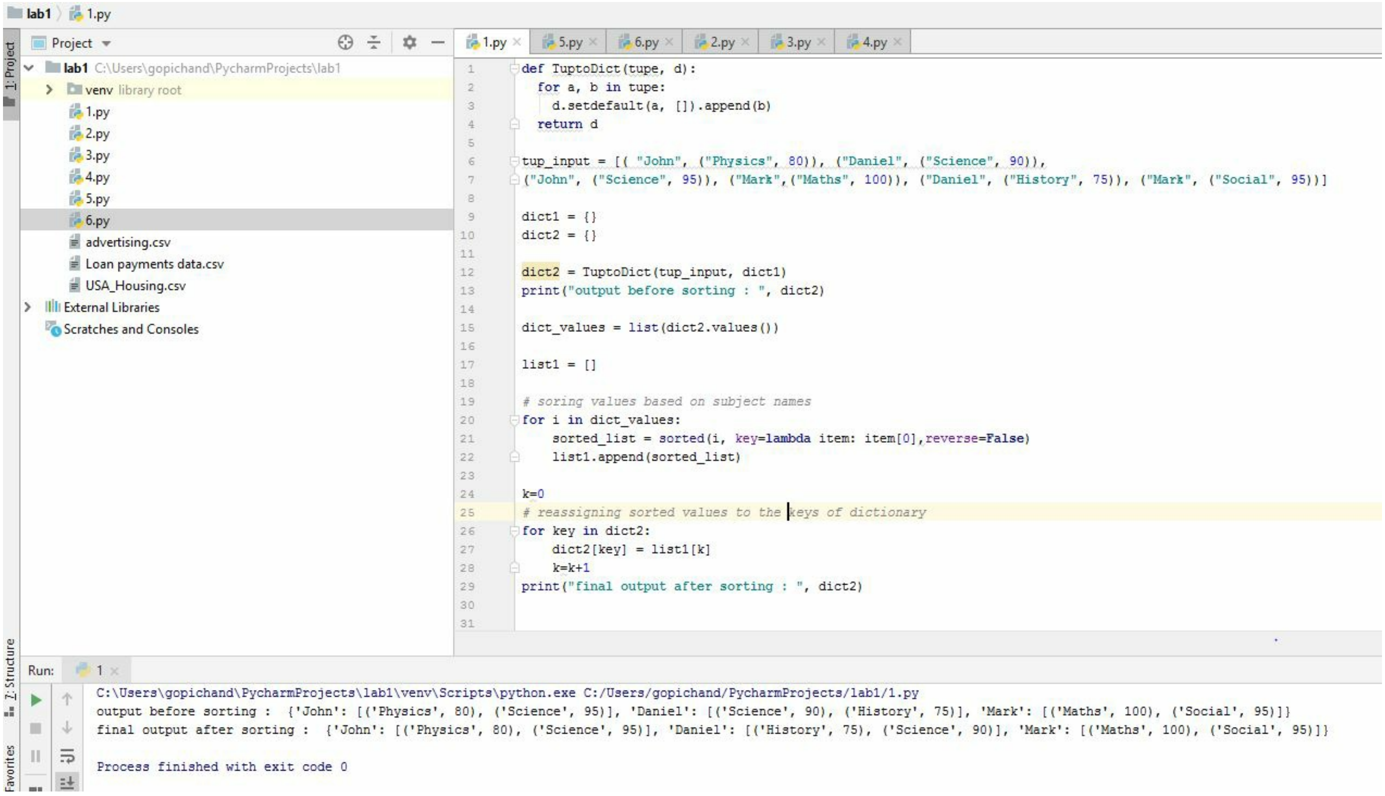
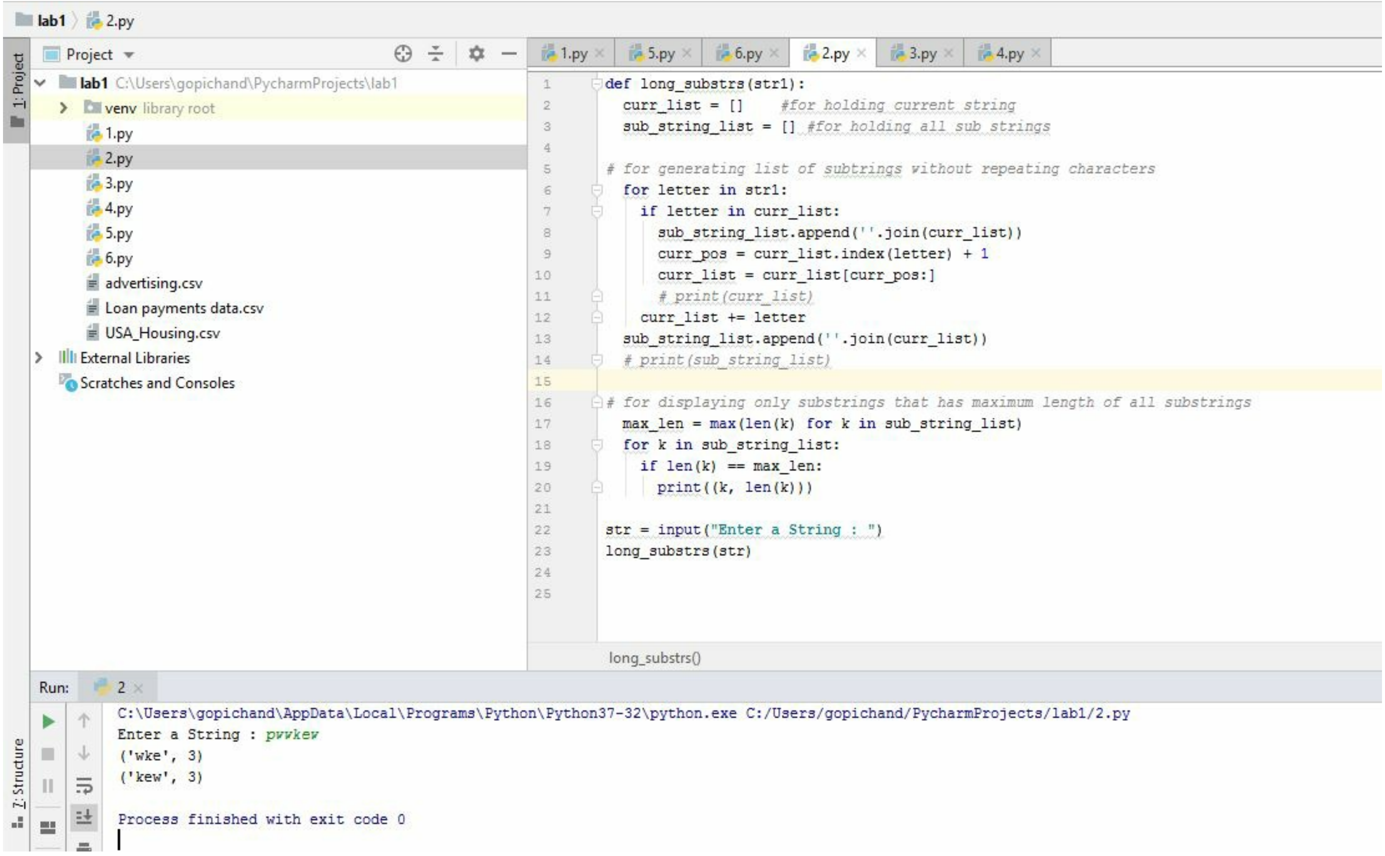
Welcome to the Python_Lab1 wiki team2!

1. In the first program we have taken list of tuples a input. A function was created and we have used setdefault inbuilt property to convert tuples to dictionary. After that e have sorted only the values and displayed the dictionary.



2. Input string is entered from Console. We have written a for loop which iterates over each letter in the string and finally creates a list of substrings. Finally we have displayed only the substrings with maximum length



3. We have taken classes Flight, Employee, Passenger, Luggage, Fare and Ticket. Fare is inheriting from Luggage class and Ticket is inheriting from Passenger and Fare classes. There is super call in Fare class. There are also two privates variables in Employee and Passenger classes respectively. It will take passenger details as input and give total fare of ticket based on type of purchase and number of check in bags

lab1 〉 3.py

Project ▾

- lab1 C:\Users\gopichand\PycharmProjects\lab1
  - venv library root
  - 1.py
  - 2.py
  - 3.py
  - 4.py
  - 5.py
  - 6.py
  - advertising.csv
  - Loan payments data.csv
  - USA_Housing.csv
- External Libraries
- Scratches and Consoles

1.py | 5.py | 6.py | 2.py | 3.py | 4.py

```python
class Flight:
    def __init__(self, f_name, f_num):
        self.f_name = f_name
        self.f_num = f_num

    def flight_display(self):
        print('flight_name: ', self.f_name)
        print('flight_number: ', self.f_num)

class Employee:
    def __init__(self, e_id, e_name, e_age, e_gender):
        self.e_name = e_name
        self.e_age = e_age
        self.__e_id = e_id #private variable
        self.e_gender =  e_gender
    def emp_display(self):
        print("employee_name: ",self.e_name)
        print('employee_id: ', self.__e_id)
        print('employee_age: ',self.e_age)
        print('employee_gender: ', self.e_gender)

class Passenger:
    def __init__(self):
        Passenger.__passport_number = input("enter the passport number of passenger: ") #private variable
        Passenger.name = input('enter name of passenger: ')
        Passenger.age = int(input('enter age of passenger: '))
        Passenger.gender = input('enter the gender: ')
        Passenger.class_type = input('select b for business or e for economy class: ')

class Luggage():
    bag_fare = 0
    def __init__(self, checkin_bags):
        self.checkin_bags = checkin_bags
        if checkin_bags > 2 :
            self.bag_fare = (checkin_bags-2)*40
        else:
            pass

class Fare(Luggage): #inheritance
```

Employee 〉 __init__()

lab1 〉 3.py

Project ▾

- lab1 C:\Users\gopichand\PycharmProjects\lab1
  - venv library root
  - 1.py
  - 2.py
  - 3.py
  - 4.py
  - 5.py
  - 6.py
  - advertising.csv
  - Loan payments data.csv
  - USA_Housing.csv
- External Libraries
- Scratches and Consoles

1.py | 5.py | 6.py | 2.py | 3.py | 4.py

```python
class Fare(Luggage): #inheritance
    offline = 250
    online = 300
    total_fare=0
    def __init__(self):
        super().__init__(int(input('enter number of check-in bags carrying other than cabin bag : '))) #super call
        x = input('buy ticket through online or offline: ')
        if x == 'online':
            Fare.total_fare = self.online + self.bag_fare
        elif x == 'offline':
            Fare.total_fare = self.offline + self.bag_fare
        else:
            pass

class Ticket(Passenger, Fare): #inheritance
    def __init__(self):
        print("Passenger name:",Passenger.name)
        if Passenger.class_type == "b":
            str = "business"
            Fare.total_fare+=80
        else:
            str = "economy"
            pass

        print("Passenger class type:",str)
        print("Total fare:",Fare.total_fare)


f1=Flight('AA','AA786J')
f1.flight_display()

emp1 = Employee('e1','Gopi Chand',24,'M')
emp1.emp_display()

p1 = Passenger()

fare1=Fare()

t= Ticket()
```

4. We have taken Advertisements dataset where ‘Clicked on AD’ column is the target column(number of hits on AD) depends on other features data like Area Income,Daily Internet Usage,Ad Topic Line,City,Country etc. We have done EDA by converting non-numerical features to numerical features and removed null values. We applied Multiple regression model to the data by taking all features once and taking only top correlated features once. By looking at r2 and rmse score, we observed that the model under performed for correlated data

4.py | 6.py | 5.py | 6tth.py

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

advv = pd.read_csv('advertising.csv')

print(advv.columns)

#converting non-numeric features to numeric features
advv['Country'] =advv['Country'].astype('category').cat.codes
advv['Ad Topic Line'] =advv['Ad Topic Line'].astype('category').cat.codes
advv['City'] =advv['City'].astype('category').cat.codes

#checking for nulls in the data
nulls = pd.DataFrame(advv.isnull().sum().sort_values(ascending=False))
nulls.columns = ['Null Count']
nulls.index.name  = 'Feature'
# print(nulls)

#removing nulls in data
adv = advv.select_dtypes(include=[np.number]).interpolate().dropna()
print(sum(adv.isnull().sum()  != 0))


print(adv.head())

df = pd.DataFrame(adv)

# checking correlation of all the columns against target column
print(' correlation of all columns are :\n' + str(df[df.columns[:]].corr()['Clicked on Ad'].sort_values(ascending=False)))

# Taking all columns for analysis
X = adv.drop('Clicked on Ad',axis=1)
y = adv['Clicked on Ad']

# taking top correlated columns for analysis
A = adv[['Age','Ad Topic Line','Country']]
b = adv['Clicked on Ad']
```

4.py 6.py 5.py 6tth.py

```python

# splitting data into train data for training model and test data for testing model
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=42, test_size=.2)
A_train, A_test, b_train, b_test = train_test_split(A, b, random_state=42, test_size=.2)

# fitting model to our train data
from sklearn.linear_model import LinearRegression
lm = LinearRegression()
lm.fit(X_train,y_train)

lm2 = LinearRegression()
lm2.fit(A_train,b_train)

# print(lm.intercept_)

coeff = pd.DataFrame(lm.coef_,X.columns,columns=['Coefficient'])
# print(coeff)

# testing the model comparing with test data
predictions = lm.predict(X_test)
predictions1 = lm2.predict(A_test)
plt.scatter(y_test,predictions)
# plt.show()

# metrics
from sklearn.metrics import r2_score
print('total r2 score is ',r2_score(y_test,predictions))
print('correlated r2 score is ',r2_score(b_test,predictions1))
# print('r2 score is ',lm.score(X_test,y_test))
# print('r2 score is ',lm.score(y_test,predictions))
from sklearn.metrics import mean_squared_error
print('total rmse',mean_squared_error(y_test,predictions))
print('correlated rmse',mean_squared_error(b_test,predictions1))
```

Run: 4

```
4                    68.37   35     73889.99  ...     0       96              0

[5 rows x 9 columns]
 correlation of all columns are :
Clicked on Ad               1.000000
Age                         0.492531
Ad Topic Line               0.022787
Country                     0.011415
City                       -0.007554
Male                       -0.038027
Area Income                -0.476255
Daily Time Spent on Site   -0.748117
Daily Internet Usage       -0.786539
Name: Clicked on Ad, dtype: float64
total r2 score is  0.746225058354061
correlated r2 score is  0.25516818767147975
total rmse 0.06267606621300575
correlated rmse 0.18395483684983632

Process finished with exit code 0
```

5.We have taken Loan Payments data dataset where ‘Paid Status’ column is the target column(number of hits on AD) depends on other features data like Principal,terms,past_due_days,age,education,Gender etc. We have done EDA by converting non-numerical features to numerical features, removed null values, taken top correlated columns for analysis. We applied 3 models namely Naive Bayes, SVM, KNN to the data and observed Naive Bayes is best model and followed by KNN and SVM

```python
from sklearn.preprocessing import LabelEncoder
import pandas as pd
import numpy as np
df = pd.read_csv('Loan payments data.csv')
print(df.info())

#converting non-numeric features to numeric features using label encoder
le=LabelEncoder()
df['loan_status']=le.fit_transform(df['loan_status'])
df['Gender']=le.fit_transform(df['Gender'])
df['education']=le.fit_transform(df['education'])
df['past_due_days']=le.fit_transform(df['past_due_days'])

print(df.head())
#checking for nulls in the data
nulls = pd.DataFrame(df.isnull().sum().sort_values(ascending=False))
nulls.columns = ['Null Count']
nulls.index.name  = 'Feature'
# print(nulls)

#removing nulls in data
df1 = df.select_dtypes(include=[np.number]).interpolate().dropna()
print(sum(df1.isnull().sum()  != 0))

# checking correlation of all the columns against target column
print(' correlation of all columns are :\n' + str(df1[df1.columns[:]].corr()['loan_status'].sort_values(ascending=False)))

# taking top correlated columns for analysis
A = df1[['past_due_days','age','education']]
# taking target column
b = df1['loan_status']

# splitting data into train data for training model and test data for testing model
from sklearn.model_selection import train_test_split
A_train, A_test, b_train, b_test = train_test_split(A, b, random_state=42, test_size=.1)

# using naive bayes model
from sklearn.naive_bayes import GaussianNB
gnb = GaussianNB()
```

```python
#using svm
from sklearn.svm import SVC
model = SVC()

#using knn
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier()


# fitting model to our train data
gnb.fit(A_train,b_train)
model.fit(A_train,b_train)
knn.fit(A_train,b_train)

# testing the model comparing with test data
predictions_naive = gnb.predict(A_test)
predictions_svm = model.predict(A_test)
predictions_knn = knn.predict(A_test)


# calculating root mean sqaure value of the errors
from sklearn.metrics import mean_squared_error
print ('RMSE for naive bayes is: \n', mean_squared_error(b_test, predictions_naive))
print ('RMSE for svm is: \n', mean_squared_error(b_test, predictions_svm))
print ('RMSE for knn is: \n', mean_squared_error(b_test, predictions_knn))


from sklearn import metrics
print("Accuracy for naive bayes is : ",round(metrics.accuracy_score(b_test, predictions_naive) * 100, 2))
# print("classification_report\n",metrics.classification_report(y_test,predictions_naive))
# print("confusion matrix\n",metrics.confusion_matrix(y_test,predictions_naive))
print("Accuracy for svm is : ",round(metrics.accuracy_score(b_test, predictions_svm) * 100, 2))
# print("classification_report\n",metrics.classification_report(y_test,predictions_svm))
# print("confusion matrix\n",metrics.confusion_matrix(y_test,predictions_svm))

print("Accuracy for knn is : ",round(metrics.accuracy_score(b_test, predictions_knn) * 100, 2))
# print("classification_report\n",metrics.classification_report(y_test,predictions_knn))
# print("confusion matrix\n",metrics.confusion_matrix(y_test,predictions_knn))
```

```
[5 rows x 11 columns]
0
 correlation of all columns are :
loan_status      1.000000
past_due_days    0.681331
age              0.032408
education        0.019917
Principal       -0.076873
Gender          -0.084500
terms           -0.098473
Name: loan_status, dtype: float64
C:\Users\gopichand\AppData\Local\Programs\Python\Python37-32\lib\site-packages\sklearn\svm\base.py:193: FutureWarning: The default value of gamma will change
  "avoid this warning.", FutureWarning)
RMSE for naive bayes is:
 0.02
RMSE for svm is:
 0.3
RMSE for knn is:
 0.04
Accuracy for naive bayes is :  98.0
Accuracy for svm is :  82.0
Accuracy for knn is :  96.0

Process finished with exit code 0
```

6.We have taken USA_Housing dataset where ‘Price’ column is the target column(number of hits on AD) depends on other features data like Avg. Area Income,Avg. Area House Age,Avg. Area Number of

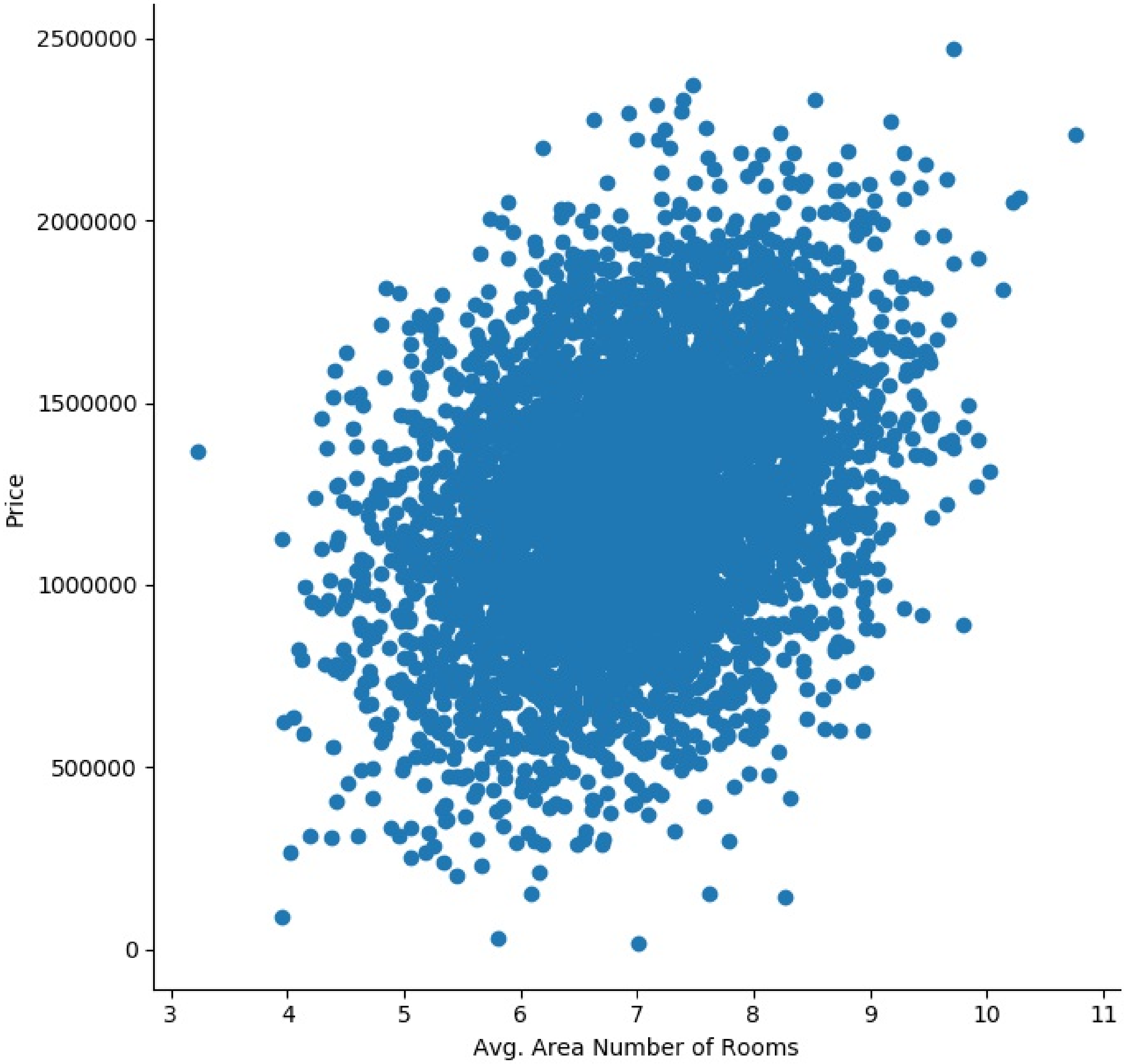
Rooms,Avg. Area Number of Bedrooms,Area Population etc. We have done EDA by converting non-numerical features to numerical features, removed null values. After plotting the graphs of each column against Price, we observed Avg. Area Number of Rooms plot is best suited for this analysis. By seeing elbow graph, we can found k=3 is best. We have done visualizations and calculated silhouette score by taking different number of clusters from 2 to 9

4.py 6.py 5.py 6tth.py

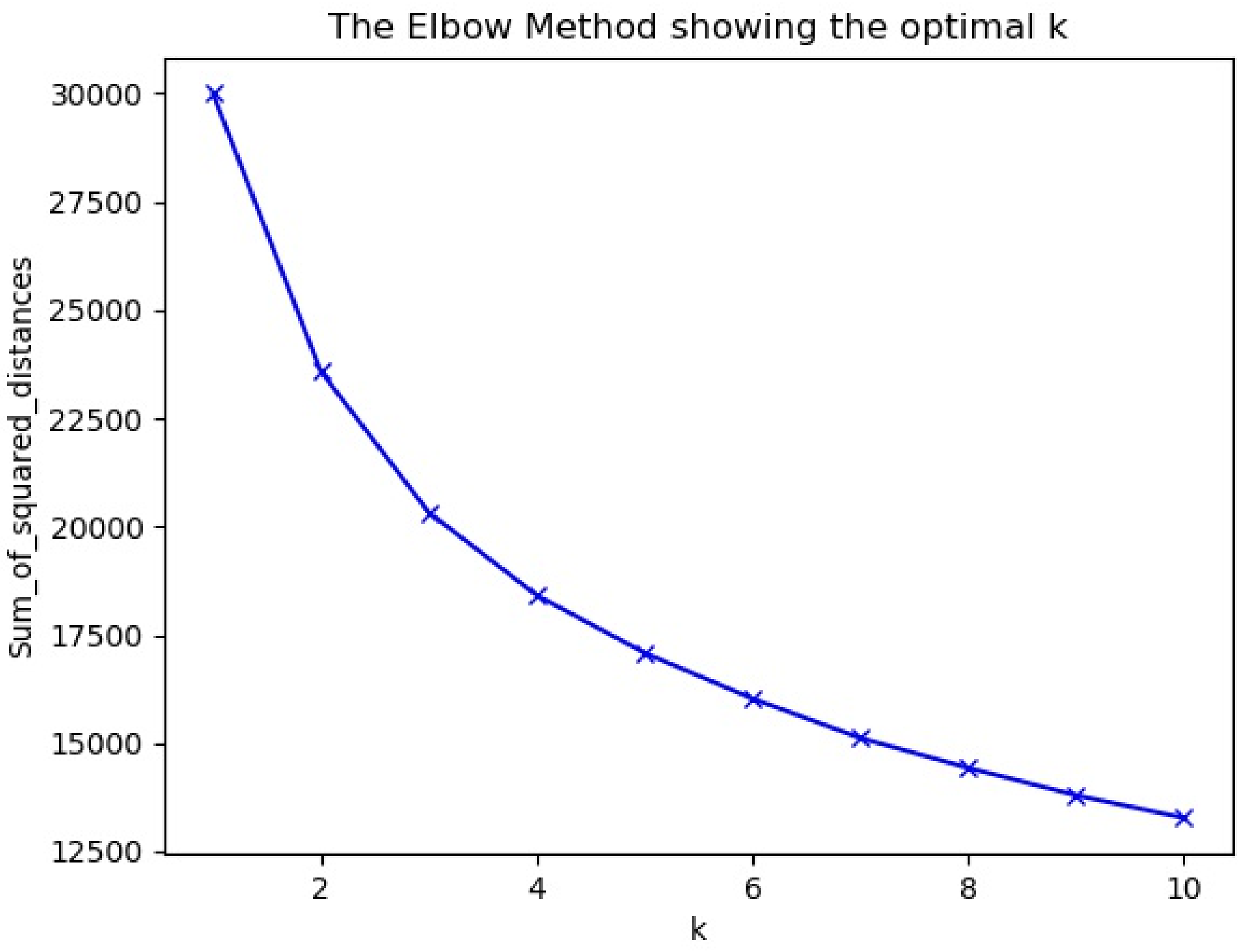
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
import seaborn as sns


data = pd.read_csv('USA_Housing.csv')
nulls = pd.DataFrame(data.isnull().sum().sort_values(ascending=False))
nulls.columns = ['Null Count']
nulls.index.name  = 'Feature'
print(nulls)

print(data.columns)

# Index(['Avg. Area Income', 'Avg. Area House Age', 'Avg. Area Number of Rooms',
#        'Avg. Area Number of Bedrooms', 'Area Population', 'Price', 'Address'],
#       dtype='object')

house = data.select_dtypes(include=[np.number]).interpolate().dropna()
print(sum(house.isnull().sum()  != 0))
nulls = pd.DataFrame(house.isnull().sum().sort_values(ascending=False))
nulls.columns = ['Null Count']
nulls.index.name  = 'Feature'
print(nulls)

print(' correlation of all columns are :\n' + str(data[data.columns[:]].corr()['Price'].sort_values(ascending=False)))

sns.FacetGrid(data,size=7)\
.map(plt.scatter,'Avg. Area Number of Rooms','Price')\
.add_legend()
plt.show()

x = house.iloc[:,[0,1,2,3,4,5]]
y = house.iloc[:,-1]
print(x.shape, y.shape)

from sklearn import preprocessing
scaler = preprocessing.StandardScaler()
```

4.py 6.py 5.py 6tth.py

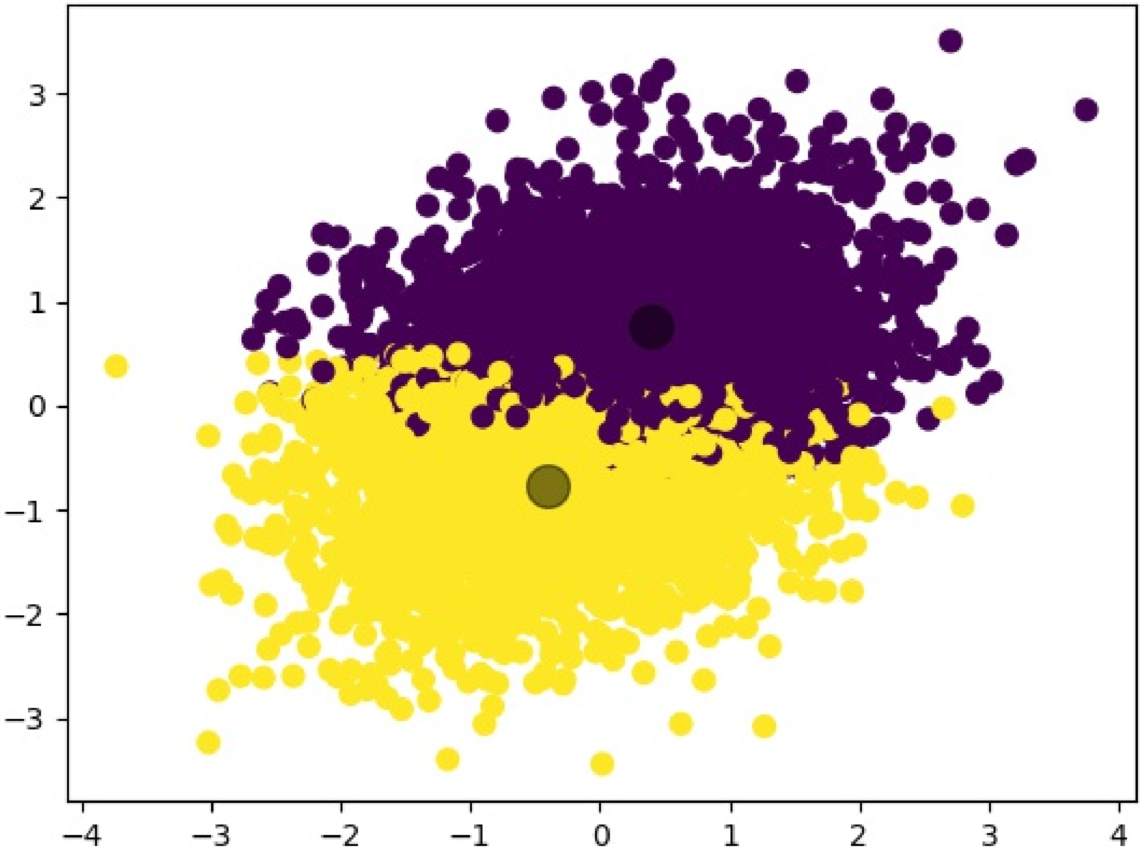
```python
from sklearn import preprocessing
scaler = preprocessing.StandardScaler()
from sklearn import metrics
scaler.fit(x)
X_scaled_array = scaler.transform(x)
X_scaled = pd.DataFrame(X_scaled_array, columns = x.columns)

from sklearn.cluster import KMeans
Sum_of_squared_distances = []
K = range(1,11)
for k in K:
    km = KMeans(n_clusters=k,init='k-means++',max_iter=300,n_init=10,random_state=0)
    km = km.fit(X_scaled)
    Sum_of_squared_distances.append(km.inertia_)


plt.plot(K, Sum_of_squared_distances, 'bx-')
plt.xlabel('k')
plt.ylabel('Sum_of_squared_distances')
plt.title('The Elbow Method showing the optimal k')
plt.show()


nclusters = 3 # this is the k in kmeans
seed = 0
K = range(2,10)
for k in K:
    km = KMeans(n_clusters=k, random_state=seed)
    km.fit(X_scaled)
# predict the cluster for each data point
    y_cluster_kmeans = km.predict(X_scaled)
    plt.scatter(X_scaled_array[:, 2], X_scaled_array[:, 5], c=y_cluster_kmeans, s=50)
    centers = km.cluster_centers_
    plt.scatter(centers[:, 2], centers[:, 5], c='black', s=200, alpha=0.5)
    plt.show()
    score = metrics.silhouette_score(X_scaled, y_cluster_kmeans)
    print('silhouette score for clusters ' +str(k)+' is : ' + str(score))
```



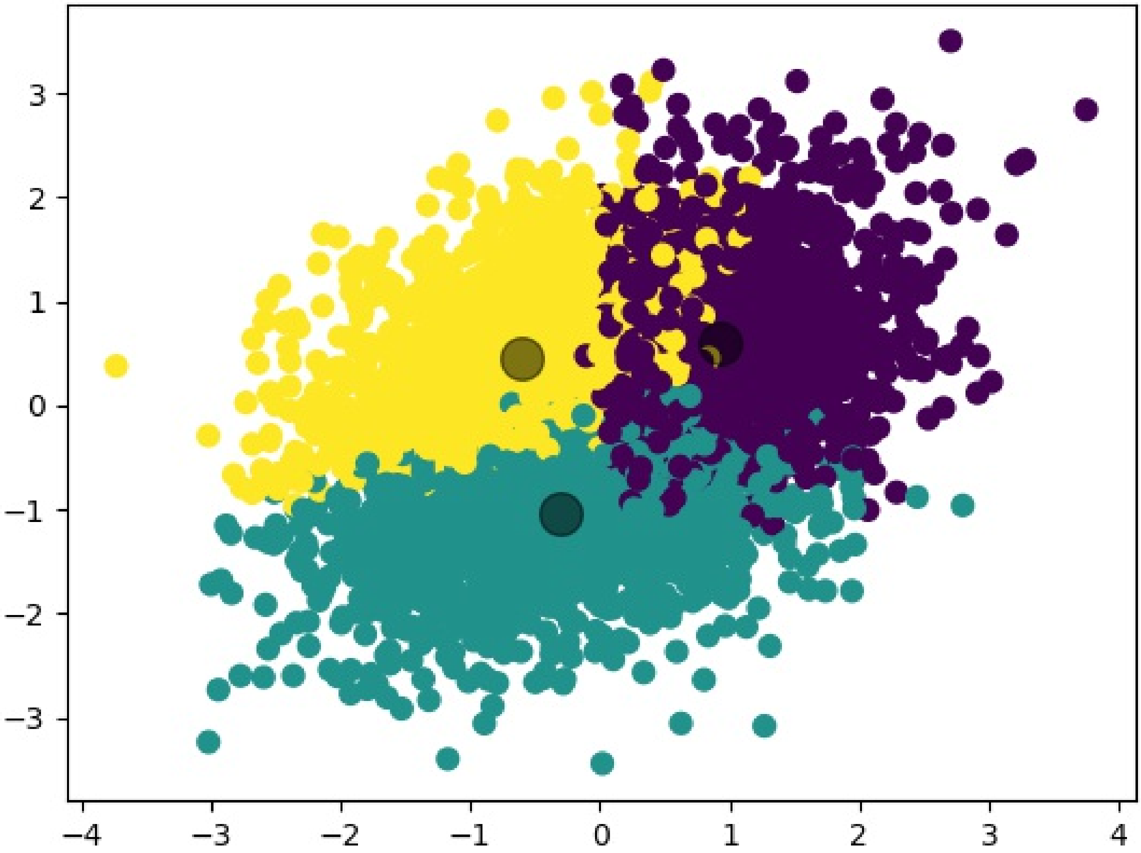
2500000
2000000
1500000
1000000
500000
0
Price
3
4
5
6
7
8
9
10
11
Avg. Area Number of Rooms



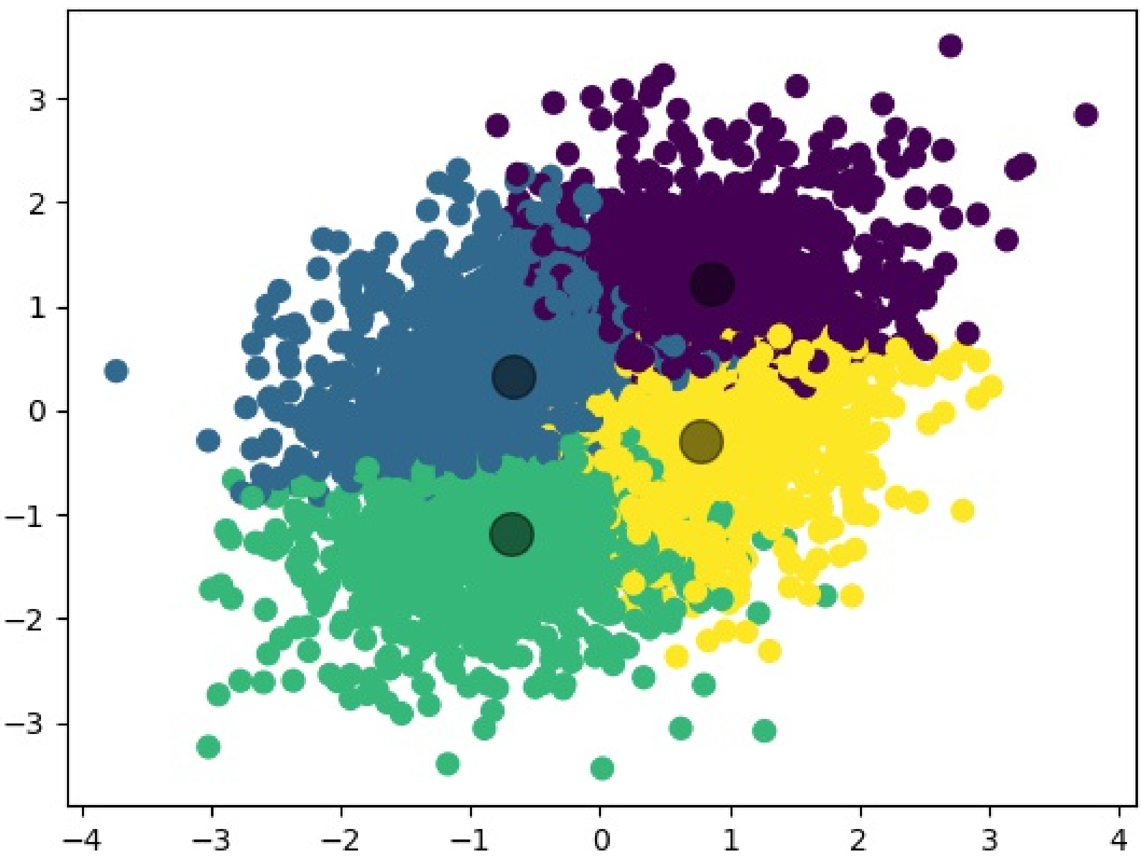
The Elbow Method showing the optimal k
Sum_of_squared_distances
30000
27500
25000
22500
20000
17500
15000
12500
2
4
6
8
10
k

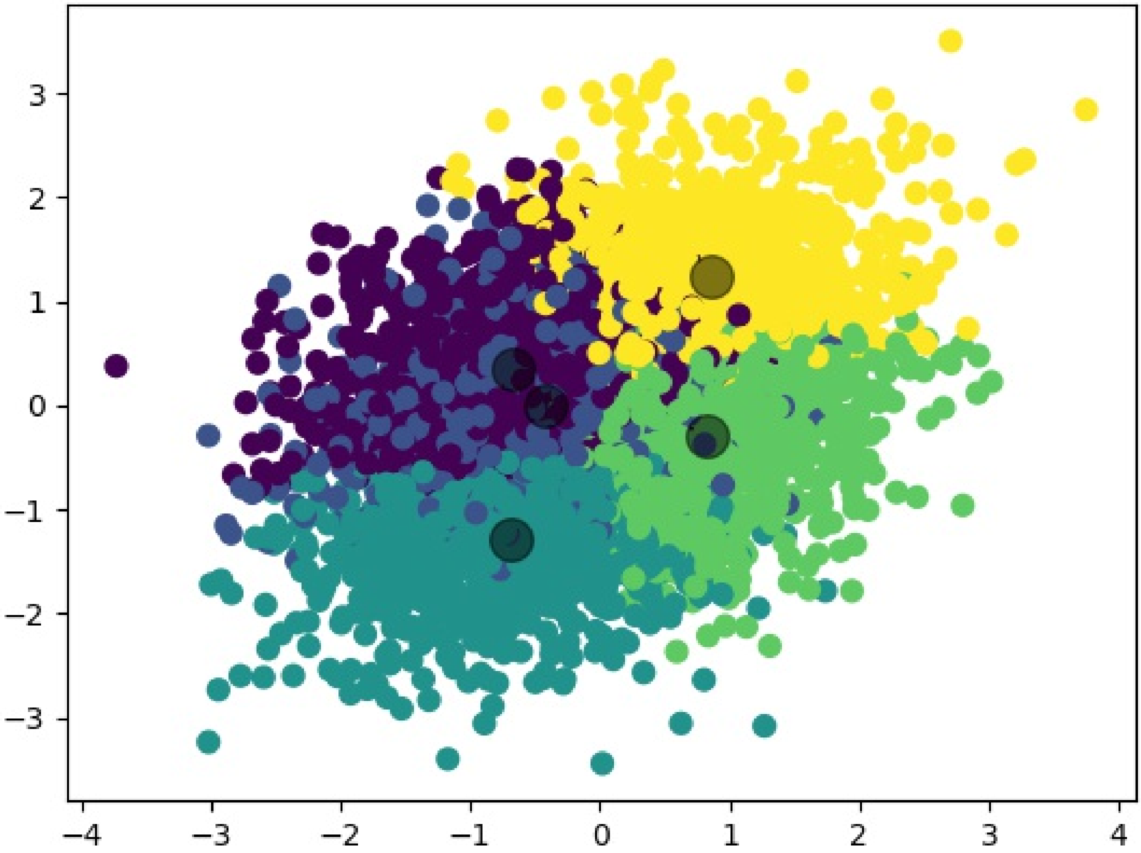


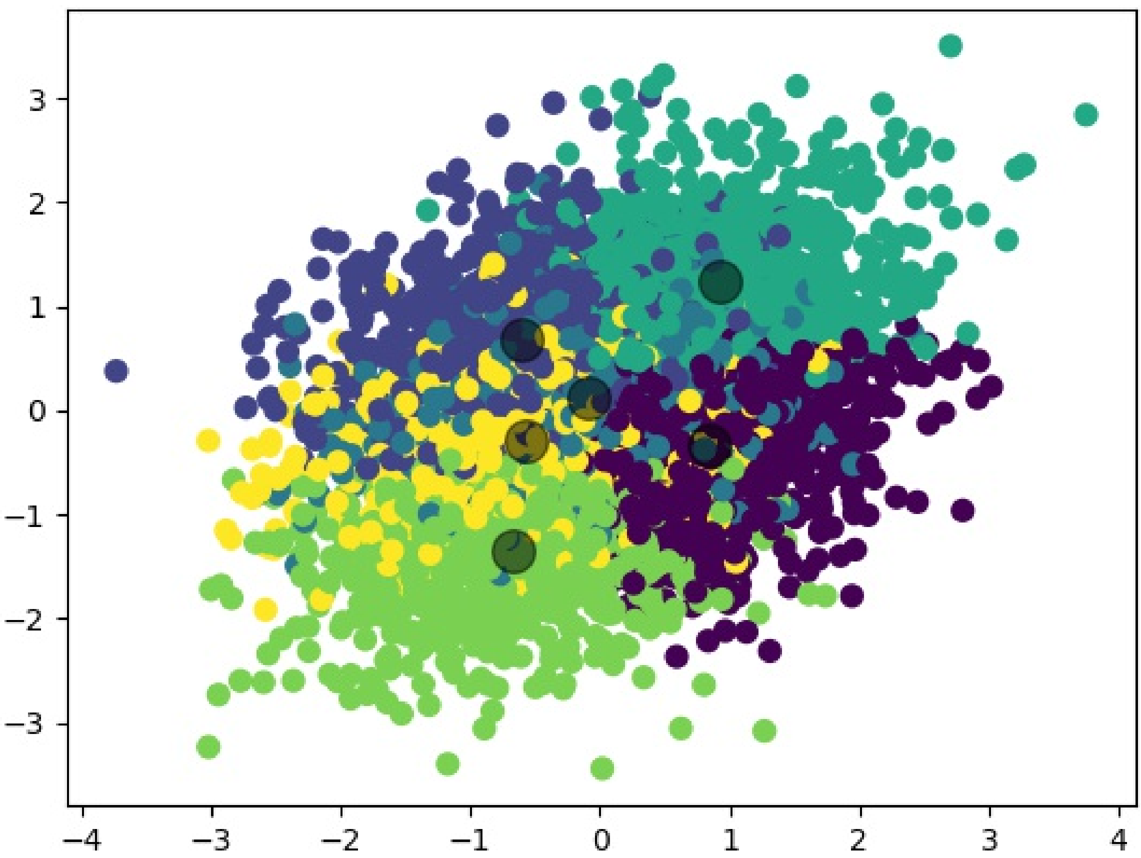
3
2
1
0
−1
−2
−3
−4
−3
−2
−1
0
1
2
3
4



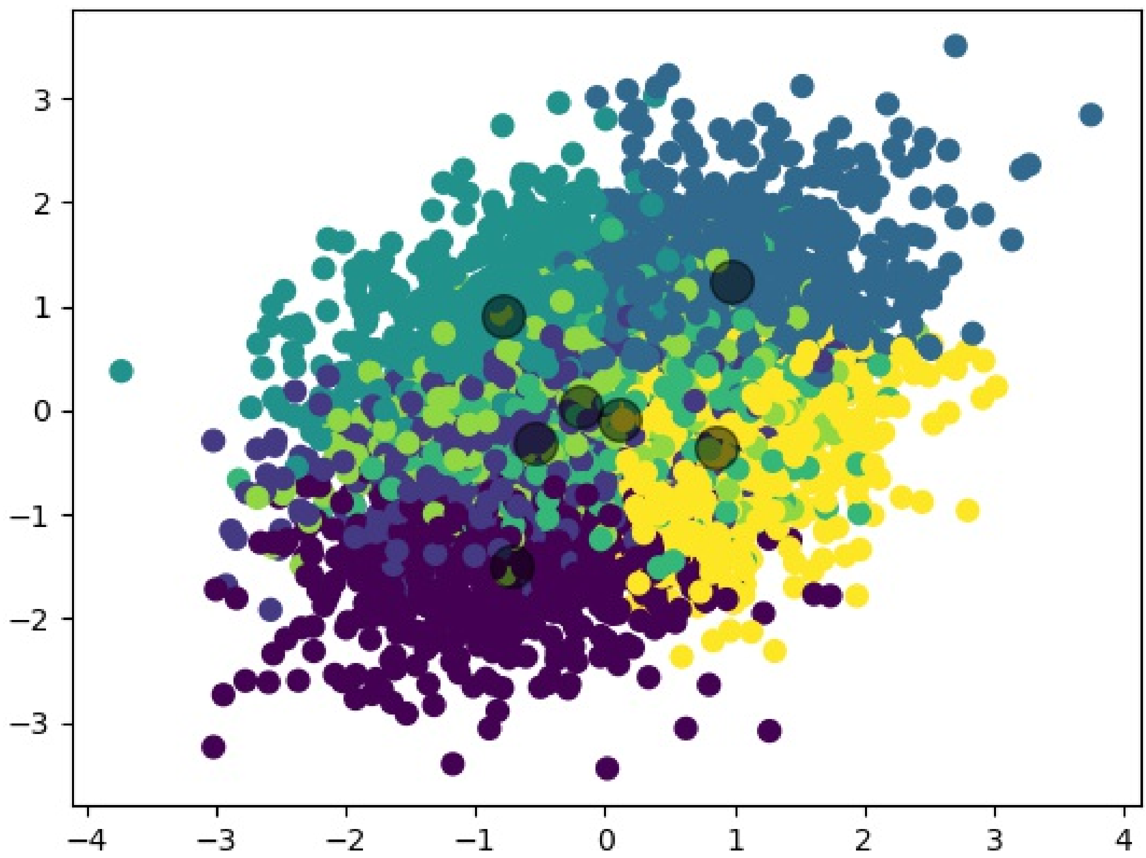
3
2
1
0
−1
−2
−3
−4
−3
−2
−1
0
1
2
3
4


3
2
1
0
−1
−2
−3
−4
−3
−2
−1
0
1
2
3
4

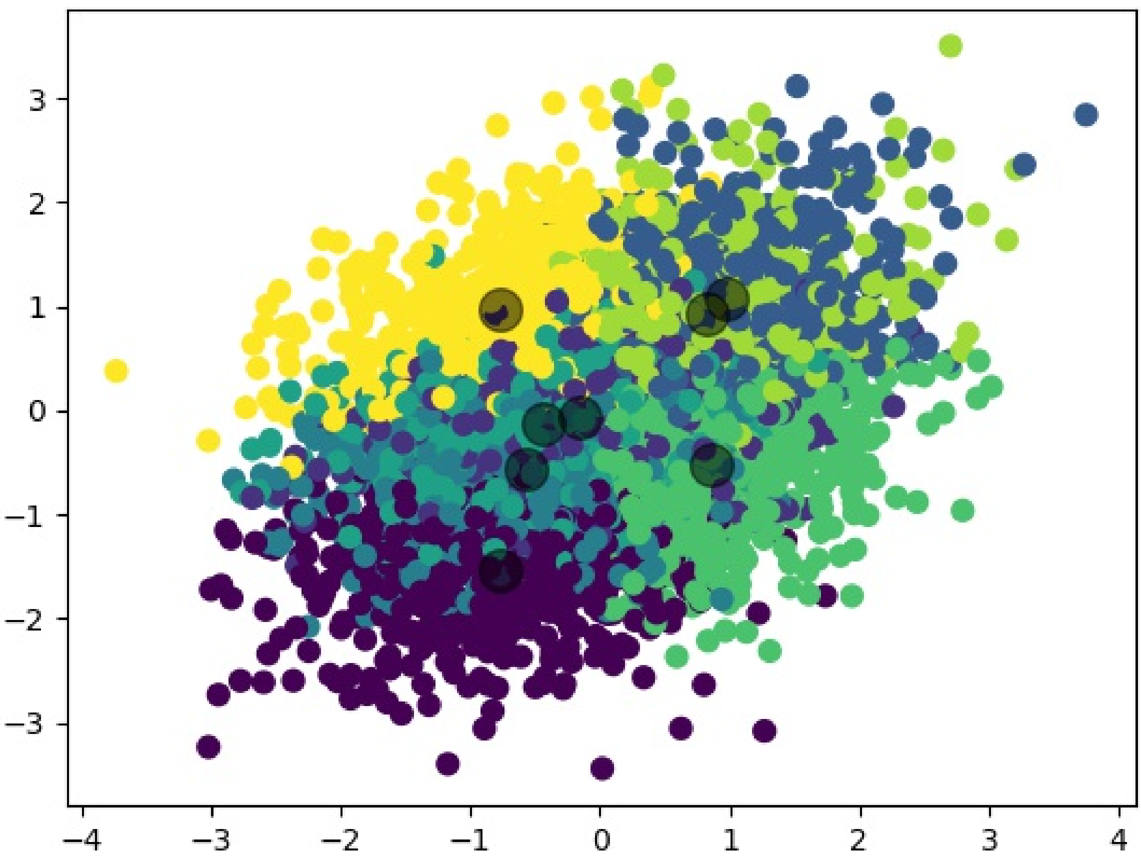

3
2
1
0
−1
−2
−3
−4
−3
−2
−1
0
1
2
3
4

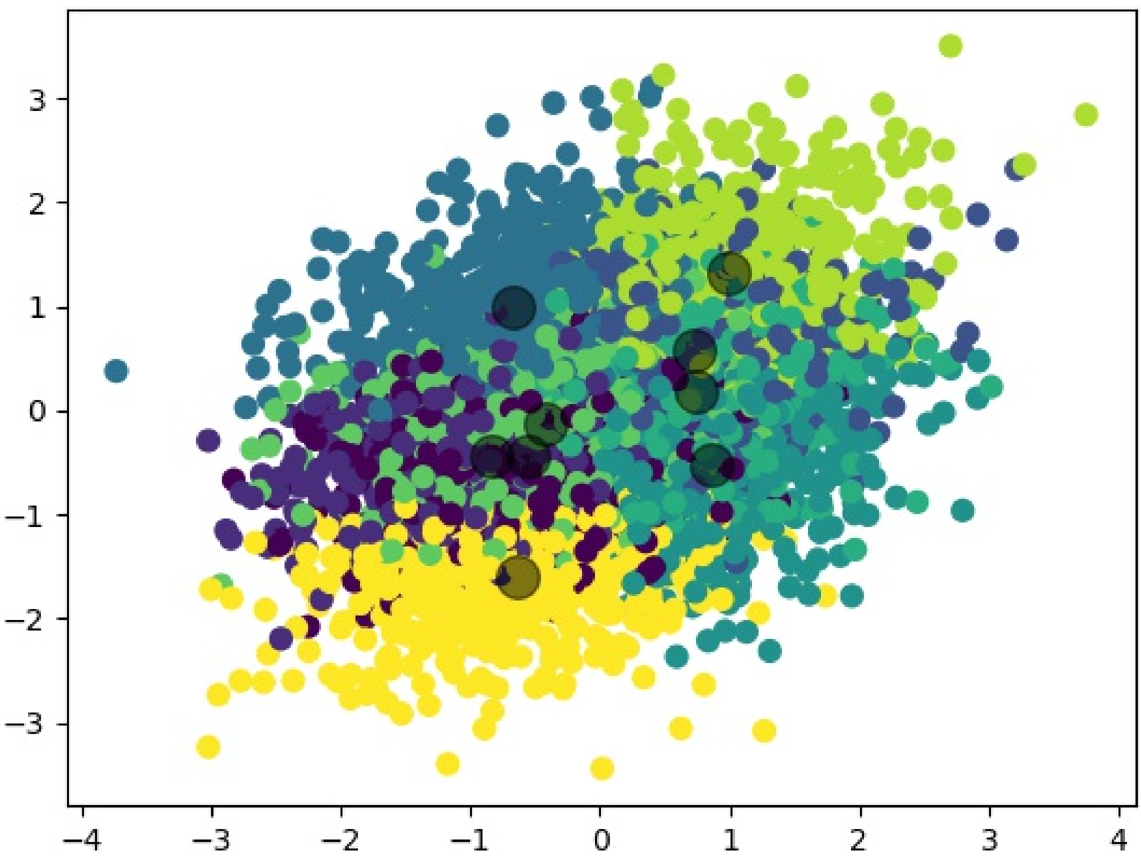


3
2
1
0
−1
−2
−3
−4
−3
−2
−1
0
1
2
3
4



3
2
1
0
−1
−2
−3
−4
−3
−2
−1
0
1
2
3
4


3
2
1
0
−1
−2
−3
−4
−3
−2
−1
0
1
2
3
4



```
C:\Users\gopichand\AppData\Local\Programs\Python\Python37-32\lib\site-packages\seaborn\axisgrid.py:230: UserWarning: The `size` paramter has been renamed to `height`;
  warnings.warn(msg, UserWarning)
(5000, 6) (5000,)
silhouette score for clusters 2 is : 0.19162386987794566
silhouette score for clusters 3 is : 0.1725904632708109
silhouette score for clusters 4 is : 0.159231305862676
silhouette score for clusters 5 is : 0.15495859919771884
silhouette score for clusters 6 is : 0.1504227088465428
silhouette score for clusters 7 is : 0.1509412651821385
silhouette score for clusters 8 is : 0.14689803096144988
silhouette score for clusters 9 is : 0.14420749380504008

Process finished with exit code 0
```