# Machine Learning - Assignment 5

Name: Idavalapati Vijay Taraka Ramarao
ID: 700742485
CRN: 13428

## Question1

(Provide only mathematical solutions for this question) Six points with the following attributes are given, calculate and find out clustering representations and dendrogram using Single, complete, and average link proximity function in hierarchical clustering technique.

ReadMe.txt | Assignment6_Question_1.py | Assignment6_Question_2.py

```python
# Q1 Six points with the following attributes are given, calculate and find out clustering
# representations and dendrogram using Single,
# complete, and average link proximity function in hierarchical clustering technique.
# Mathematical Part Solution is provided in Document, the below code is just for reference for Q1

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import scipy.cluster.hierarchy as shc
from scipy.spatial.distance import squareform, pdist
from matplotlib.pyplot import show


a = np.array([0.4005,0.2148,0.3457,0.2652,0.0789,0.4548])
b = np.array([0.5306,0.3854,0.3156,0.1875,0.4139,0.3022])

point = ['P1','P2','P3','P4','P5','P6']
data = pd.DataFrame({'Point':point, 'x cordinate':a, 'y cordinate':b})
data = data.set_index('Point')
print(data)

dist = pd.DataFrame(squareform(np.round(pdist(data[['x cordinate', 'y cordinate']]),4), 'euclidean'), columns=data.index.values,
print(dist)


print("\n")
plt.figure(figsize=(10,4))
plt.title("Dendrogram with Single inkage")
```

```python
print("\n")
plt.figure(figsize=(10,4))
plt.title("Dendrogram with Single inkage")
dend = shc.dendrogram(shc.linkage(data[['x cordinate', 'y cordinate']], method='single'), labels=data.index)
show()

plt.figure(figsize=(10,4))
plt.title("Dendrogram with Complete inkage")
dend = shc.dendrogram(shc.linkage(data[['x cordinate', 'y cordinate']], method='complete'), labels=data.index)
show()

plt.figure(figsize=(10,4))
plt.title("Dendrogram with Average inkage")
dend = shc.dendrogram(shc.linkage(data[['x cordinate', 'y cordinate']], method='average'), labels=data.index)
show()
```

**Output:**

Single Link Proximity:

- In **Single Linkage,** the distance between two clusters is the minimum distance between members of the two clusters

| | **p1** | **p2** | **p3** | **p4** | **p5** | **p6** |
|---|---|---|---|---|---|---|
| **p1** | 0 | 0.2357 | 0.2218 | 0.3688 | 0.3421 | 0.2347 |
| **p2** | 0.2357 | 0 | 0.1483 | 0.2042 | 0.1388 | 0.254 |
| **p3** | 0.2218 | 0.1483 | 0 | 0.1513 | 0.2843 | 0.11 |
| **p4** | 0.3688 | 0.2042 | 0.1513 | 0 | 0.2932 | 0.2216 |
| **p5** | 0.3421 | 0.1388 | 0.2843 | 0.2932 | 0 | 0.3921 |
| **p6** | 0.2347 | 0.254 | 0.11 | 0.2216 | 0.3921 | 0 |

smallest distance from above data is 0.11

so p3 and p6 forms first cluster

| | **p1** | **p2** | **p36** | **p4** | **p5** |
|---|---|---|---|---|---|
| **p1** | 0 | 0.2357 | 0.2218 | 0.3688 | 0.3421 |
| **p2** | 0.2357 | 0 | 0.1483 | 0.2042 | 0.1388 |
| **p36** | 0.2218 | 0.1483 | 0 | 0.1513 | 0.2843 |
| **p4** | 0.3688 | 0.2042 | 0.1513 | 0 | 0.2932 |
| **p5** | 0.3421 | 0.1388 | 0.2843 | 0.2932 | 0 |

smallest distance from above data is 0.1388 so p2 and p5 forms 2nd cluster

| | **p1** | **p25** | **p36** | **p4** |
|---|---|---|---|---|
| **p1** | 0 | 0.2357 | 0.2218 | 0.3688 |
| **p25** | 0.2357 | 0 | 0.1483 | 0.2042 |
| **p36** | 0.2218 | 0.1483 | 0 | 0.1513 |
| **p4** | 0.3688 | 0.2042 | 0.1513 | 0 |

smallest distance from above data is 0.1483

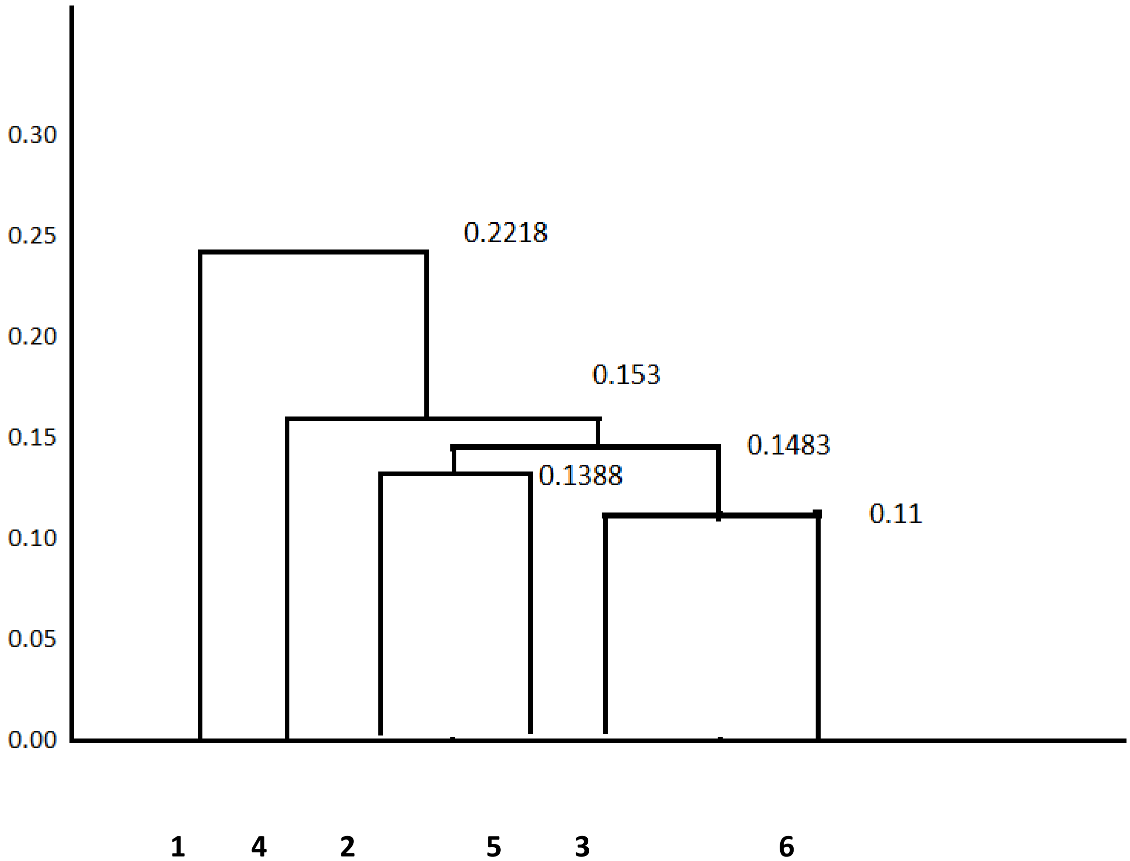
so p25 and p36 forms 3rdcluster

| | **p1** | **p(25)(36)** | **p4** |
|---|---|---|---|
| **p1** | 0 | 0.2218 | 0.3688 |
| **p(25)(36)** | 0.2218 | 0 | 0.1513 |
| **p4** | 0.3688 | 0.1513 | 0 |

smallest distance from above data is 0.153

so p(25)(36)and p4 forms 4thcluster

| | **p1** | **p4(25)(36)** |
|---|---|---|
| **p1** | 0 | 0.2218 |
| **p4(25)(36)** | 0.2218 | 0 |




Complete Link Proximity:

- In **Complete Linkage,** the distance between two clusters is the maximum distance between members of the two clusters

| | p1 | p2 | p3 | p4 | p5 | p6 |
|---|---|---|---|---|---|---|
| **p1** | 0 | 0.2357 | 0.2218 | 0.3688 | 0.3421 | 0.2347 |
| **p2** | 0.2357 | 0 | 0.1483 | 0.2042 | 0.1388 | 0.254 |
| **p3** | 0.2218 | 0.1483 | 0 | 0.1513 | 0.2843 | 0.11 |
| **p4** | 0.3688 | 0.2042 | 0.1513 | 0 | 0.2932 | 0.2216 |
| **p5** | 0.3421 | 0.1388 | 0.2843 | 0.2932 | 0 | 0.3921 |
| **p6** | 0.2347 | 0.254 | 0.11 | 0.2216 | 0.3921 | 0 |

smallest distance from above data is 0.11

so p3 and p6 forms first cluster

| | p1 | p2 | p36 | p4 | p5 |
|---|---|---|---|---|---|
| **p1** | 0 | 0.2357 | 0.2347 | 0.3688 | 0.3421 |
| **p2** | 0.2357 | 0 | 0.254 | 0.2042 | 0.1388 |
| **p36** | 0.2347 | 0.254 | 0 | 0.2216 | 0.3921 |
| **p4** | 0.3688 | 0.2042 | 0.2216 | 0 | 0.2932 |
| **p5** | 0.3421 | 0.1388 | 0.3921 | 0.2932 | 0 |

smallest distance from above data is 0.1388

so p2 and p5 forms 2nd cluster

| | p1 | p25 | p36 | p4 |
|---|---|---|---|---|
| **p1** | 0 | 0.3421 | 0.2347 | 0.3688 |
| **p25** | 0.3421 | 0 | 0.3921 | 0.2932 |
| **p36** | 0.2347 | 0.3921 | 0 | 0.2216 |
| **p4** | 0.3688 | 0.2932 | 0.2216 | 0 |

smallest distance from above data is 0.2216

so p25 and p36 forms 3rdcluster

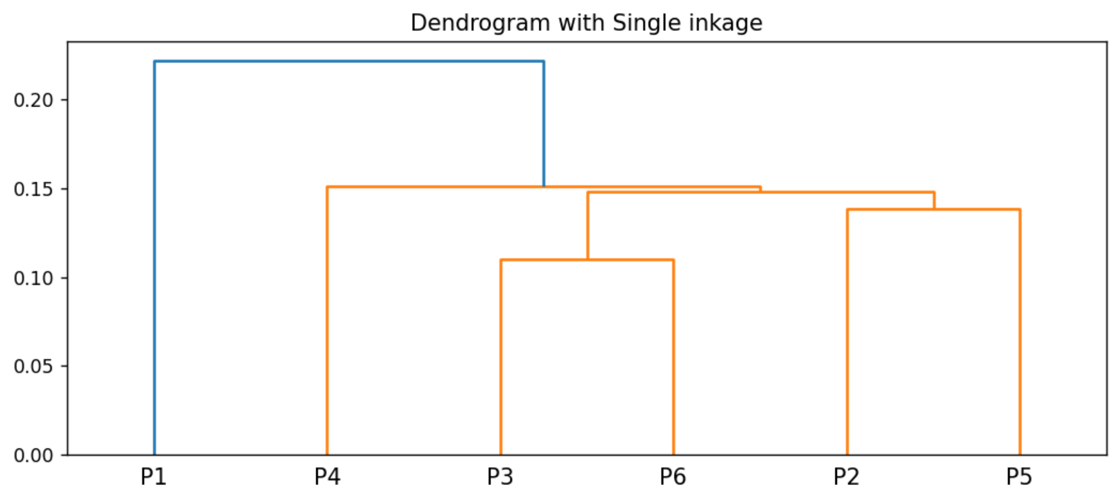
| | p1 | p(25)(36) | p4 |
|---|---|---|---|
| **p1** | 0 | 0.3421 | 0.3688 |
| **p(25)(36)** | 0.3421 | 0 | 0.2932 |
| **p4** | 0.3688 | 0.2932 | 0 |

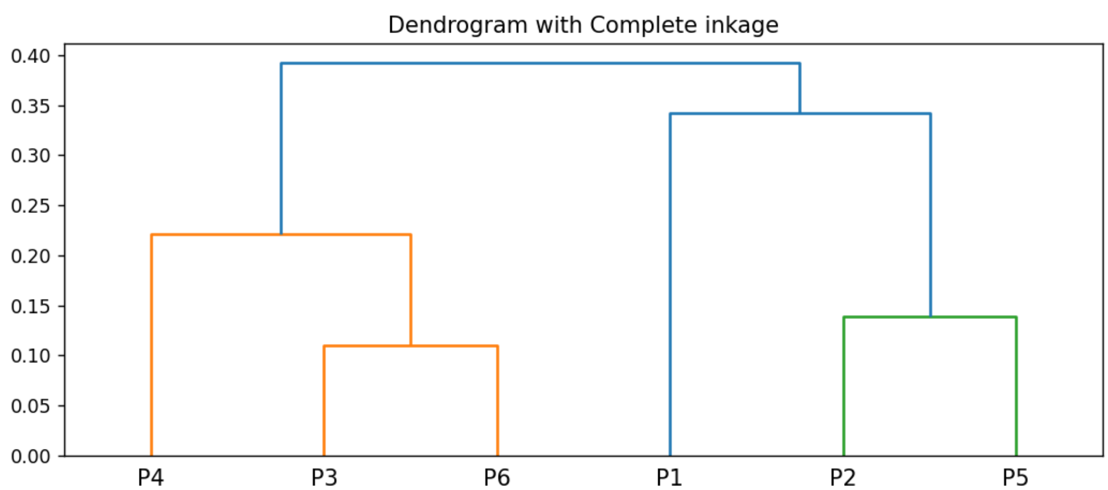
smallest distance from above data is 0.2932

so p(25)(36)and p1 forms 4thcluster

| | p1(25)(36) | p4 |
|---|---|---|
| **p1(25)(36)** | 0 | 0.1483 |
| **p4** | 0.3688 | 0 |

```
C:\Users\Administrator\Documents\GitHub\ML\venv\Scripts\python.exe C:/Users/Administrator/Documents/GitHub/ML/Assignment6/Assignment6_Question_1.py
       x cordinate  y cordinate
Point
P1          0.4005       0.5306
P2          0.2148       0.3854
P3          0.3457       0.3156
P4          0.2652       0.1875
P5          0.0789       0.4139
P6          0.4548       0.3022
        P1      P2      P3      P4      P5      P6
P1  0.0000  0.2357  0.2219  0.3688  0.3421  0.2348
P2  0.2357  0.0000  0.1483  0.2042  0.1389  0.2540
P3  0.2219  0.1483  0.0000  0.1513  0.2843  0.1099
P4  0.3688  0.2042  0.1513  0.0000  0.2932  0.2216
P5  0.3421  0.1389  0.2843  0.2932  0.0000  0.3921
P6  0.2348  0.2540  0.1099  0.2216  0.3921  0.0000
```



## Question2:

ReadMe.txt | Assignment6_Question_1.py | Assignment6_Question_2.py

```python
## 2) Use CC_GENERAL.csv given in the folder and apply:
# a) Preprocess the data by removing the categorical column and filling the missing values.
# b) Apply StandardScaler() and normalize() functions to scale and normalize raw input data.
# c) Use PCA with K=2 to reduce the input dimensions to two features.
# d) Apply Agglomerative Clustering with k=2,3,4 and 5 on reduced features and visualize
# result for each k value using scatter plot.
# e) Evaluate different variations using Silhouette Scores and Visualize results with a bar chart.

from sklearn import preprocessing
from sklearn.decomposition import PCA
from sklearn.cluster import AgglomerativeClustering
from sklearn.metrics import silhouette_score
import pandas as pd
import matplotlib.pyplot as plt

import warnings
warnings.filterwarnings("ignore")

import warnings
warnings.filterwarnings("ignore")


dataframe = pd.read_csv('datasets/CC GENERAL.csv')
print(dataframe.info())
```

```
print("\n")
print(dataframe.head())

print(dataframe.describe())

print("\n")
df = dataframe.drop(['CUST_ID'], axis=1)
print(df.head())

print("\n")
print(df.isnull().any())

print("\n")
df.fillna(dataframe.mean(), inplace=True)
print(df.isnull().any())
print(df.corr().style.background_gradient(cmap="Greens"))

x = df.iloc[:,0:-1]
y = df.iloc[:,-1]


scaler = preprocessing.StandardScaler()
scaler.fit(x)
X_scaled_array = scaler.transform(x)
```

```python
scaler = preprocessing.StandardScaler()
scaler.fit(x)
X_scaled_array = scaler.transform(x)
X_scaled_df = pd.DataFrame(X_scaled_array, columns = x.columns)

#Normalization is the process of scaling individual samples to have unit norm.
#This process can be useful if you plan to use a quadratic form such as the dot-product or any
# other kernel to quantify the similarity of any pair of samples.
X_normalized = preprocessing.normalize(X_scaled_df)
# Converting the numpy array into a pandas DataFrame
X_normalized = pd.DataFrame(X_normalized)

pca2 = PCA(n_components=2)
principalComponents = pca2.fit_transform(X_normalized)

principalDf = pd.DataFrame(data = principalComponents, columns = ['P1', 'P2'])

finalDf = pd.concat([principalDf, df[['TENURE']]], axis = 1)
print(finalDf.head())

plt.figure(figsize=(7,7))
plt.scatter(finalDf['P1'],finalDf['P2'],c=finalDf['TENURE'],cmap='prism', s =5)
plt.xlabel('pc1')
print(plt.ylabel('pc2'))
```

```python
ac2 = AgglomerativeClustering(n_clusters=2)

# Visualizing the clustering
plt.figure(figsize=(6, 6))
plt.scatter(principalDf['P1'], principalDf['P2'],
            c=ac2.fit_predict(principalDf), cmap='rainbow')
print(plt.show())

ac3 = AgglomerativeClustering(n_clusters=3)

# Visualizing the clustering
plt.figure(figsize=(6, 6))
plt.scatter(principalDf['P1'], principalDf['P2'],
            c=ac3.fit_predict(principalDf), cmap='rainbow')
print(plt.show())

ac4 = AgglomerativeClustering(n_clusters=4)

# Visualizing the clustering
plt.figure(figsize=(6, 6))
plt.scatter(principalDf['P1'], principalDf['P2'],
            c=ac4.fit_predict(principalDf), cmap='rainbow')
print(plt.show())
```

```python
            c=ac4.fit_predict(principalDf), cmap='rainbow')
print(plt.show())

ac5 = AgglomerativeClustering(n_clusters=5)

# Visualizing the clustering
plt.figure(figsize=(6, 6))
plt.scatter(principalDf['P1'], principalDf['P2'],
            c=ac5.fit_predict(principalDf), cmap='rainbow')
print(plt.show())

k = [2, 3, 4, 5]

# Appending the silhouette scores of the different models to the list
silhouette_scores = []
silhouette_scores.append(
    silhouette_score(principalDf, ac2.fit_predict(principalDf)))
silhouette_scores.append(
    silhouette_score(principalDf, ac3.fit_predict(principalDf)))
silhouette_scores.append(
    silhouette_score(principalDf, ac4.fit_predict(principalDf)))
silhouette_scores.append(
    silhouette_score(principalDf, ac5.fit_predict(principalDf)))

# Plotting a bar graph to compare the results
plt.bar(k, silhouette_scores)
plt.xlabel('Number of clusters', fontsize=20)
plt.ylabel('S(i)', fontsize=20)
print(plt.show())
```

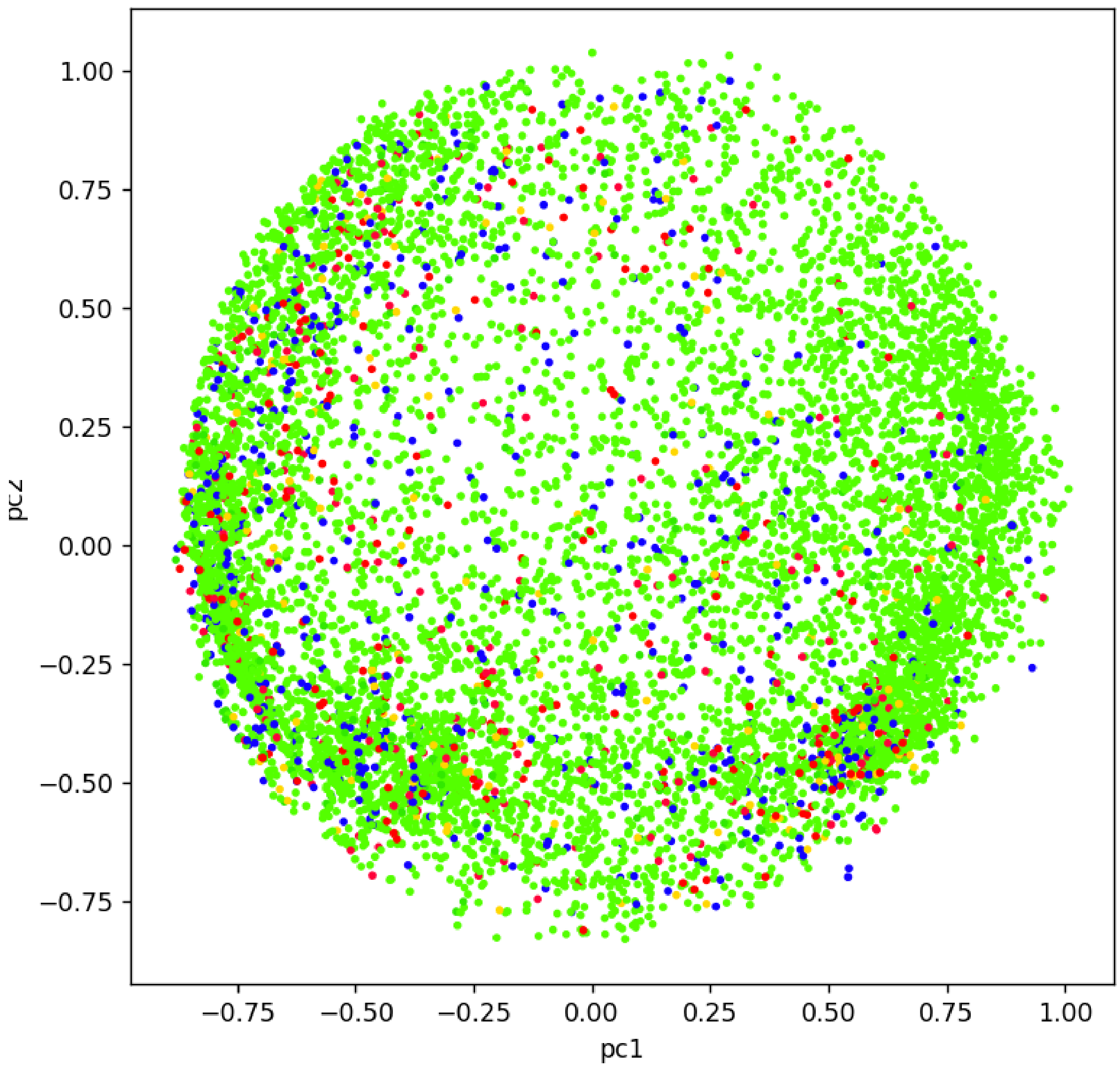
**Output:**

```
C:\Users\Administrator\Documents\GitHub\ML\venv\Scripts\python.exe C:/Users/Administrator/Documents/GitHub/ML/Assignment6/Assignment6_Question_2.py
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8950 entries, 0 to 8949
Data columns (total 18 columns):
 #   Column                            Non-Null Count  Dtype
---  ------                            --------------  -----
 0   CUST_ID                           8950 non-null   object
 1   BALANCE                           8950 non-null   float64
 2   BALANCE_FREQUENCY                 8950 non-null   float64
 3   PURCHASES                         8950 non-null   float64
 4   ONEOFF_PURCHASES                  8950 non-null   float64
 5   INSTALLMENTS_PURCHASES            8950 non-null   float64
 6   CASH_ADVANCE                      8950 non-null   float64
 7   PURCHASES_FREQUENCY               8950 non-null   float64
 8   ONEOFF_PURCHASES_FREQUENCY        8950 non-null   float64
 9   PURCHASES_INSTALLMENTS_FREQUENCY  8950 non-null   float64
 10  CASH_ADVANCE_FREQUENCY            8950 non-null   float64
 11  CASH_ADVANCE_TRX                  8950 non-null   int64
 12  PURCHASES_TRX                     8950 non-null   int64
 13  CREDIT_LIMIT                      8949 non-null   float64
 14  PAYMENTS                          8950 non-null   float64
 15  MINIMUM_PAYMENTS                  8637 non-null   float64
 16  PRC_FULL_PAYMENT                  8950 non-null   float64
 17  TENURE                            8950 non-null   int64
dtypes: float64(14), int64(3), object(1)
memory usage: 1.2+ MB
```

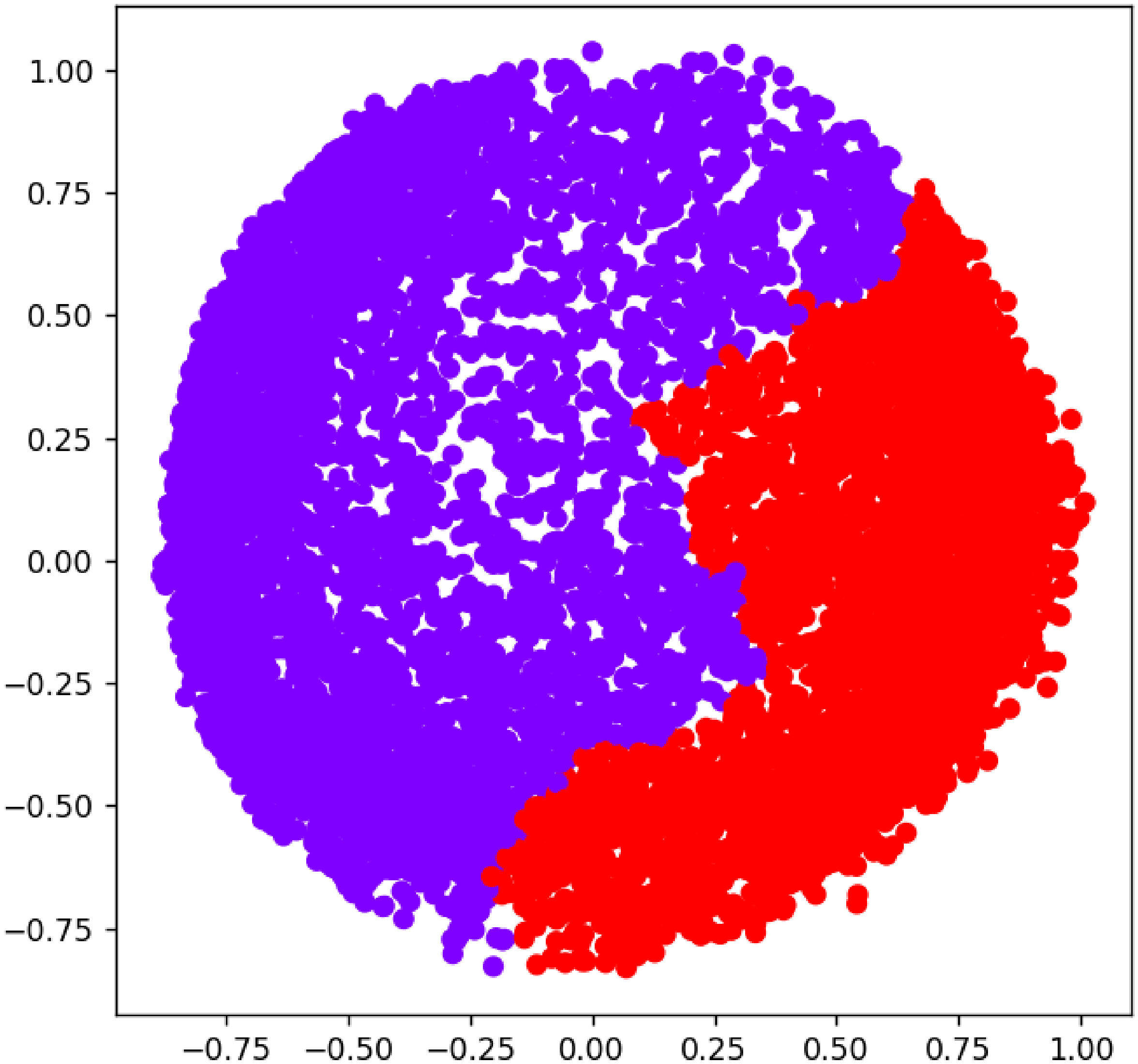
```
  CUST_ID      BALANCE  ...  PRC_FULL_PAYMENT  TENURE
0  C10001    40.900749  ...          0.000000      12
1  C10002  3202.467416  ...          0.222222      12
2  C10003  2495.148862  ...          0.000000      12
3  C10004  1666.670542  ...          0.000000      12
4  C10005   817.714335  ...          0.000000      12

[5 rows x 18 columns]
            BALANCE  BALANCE_FREQUENCY  ...  PRC_FULL_PAYMENT       TENURE
count   8950.000000        8950.000000  ...       8950.000000  8950.000000
mean    1564.474828           0.877271  ...          0.153715    11.517318
std     2081.531879           0.236904  ...          0.292499     1.338331
min        0.000000           0.000000  ...          0.000000     6.000000
25%      128.281915           0.888889  ...          0.000000    12.000000
50%      873.385231           1.000000  ...          0.000000    12.000000
75%     2054.140036           1.000000  ...          0.142857    12.000000
max    19043.138560           1.000000  ...          1.000000    12.000000

[8 rows x 17 columns]
```

```
       BALANCE  BALANCE_FREQUENCY  ...  PRC_FULL_PAYMENT  TENURE
0    40.900749           0.818182  ...          0.000000      12
1  3202.467416           0.909091  ...          0.222222      12
2  2495.148862           1.000000  ...          0.000000      12
3  1666.670542           0.636364  ...          0.000000      12
4   817.714335           1.000000  ...          0.000000      12

[5 rows x 17 columns]


BALANCE                             False
BALANCE_FREQUENCY                   False
PURCHASES                           False
ONEOFF_PURCHASES                    False
INSTALLMENTS_PURCHASES              False
CASH_ADVANCE                        False
PURCHASES_FREQUENCY                 False
ONEOFF_PURCHASES_FREQUENCY          False
PURCHASES_INSTALLMENTS_FREQUENCY    False
CASH_ADVANCE_FREQUENCY              False
CASH_ADVANCE_TRX                    False
PURCHASES_TRX                       False
CREDIT_LIMIT                         True
PAYMENTS                            False
MINIMUM_PAYMENTS                     True
PRC_FULL_PAYMENT                    False
TENURE                              False
dtype: bool
```

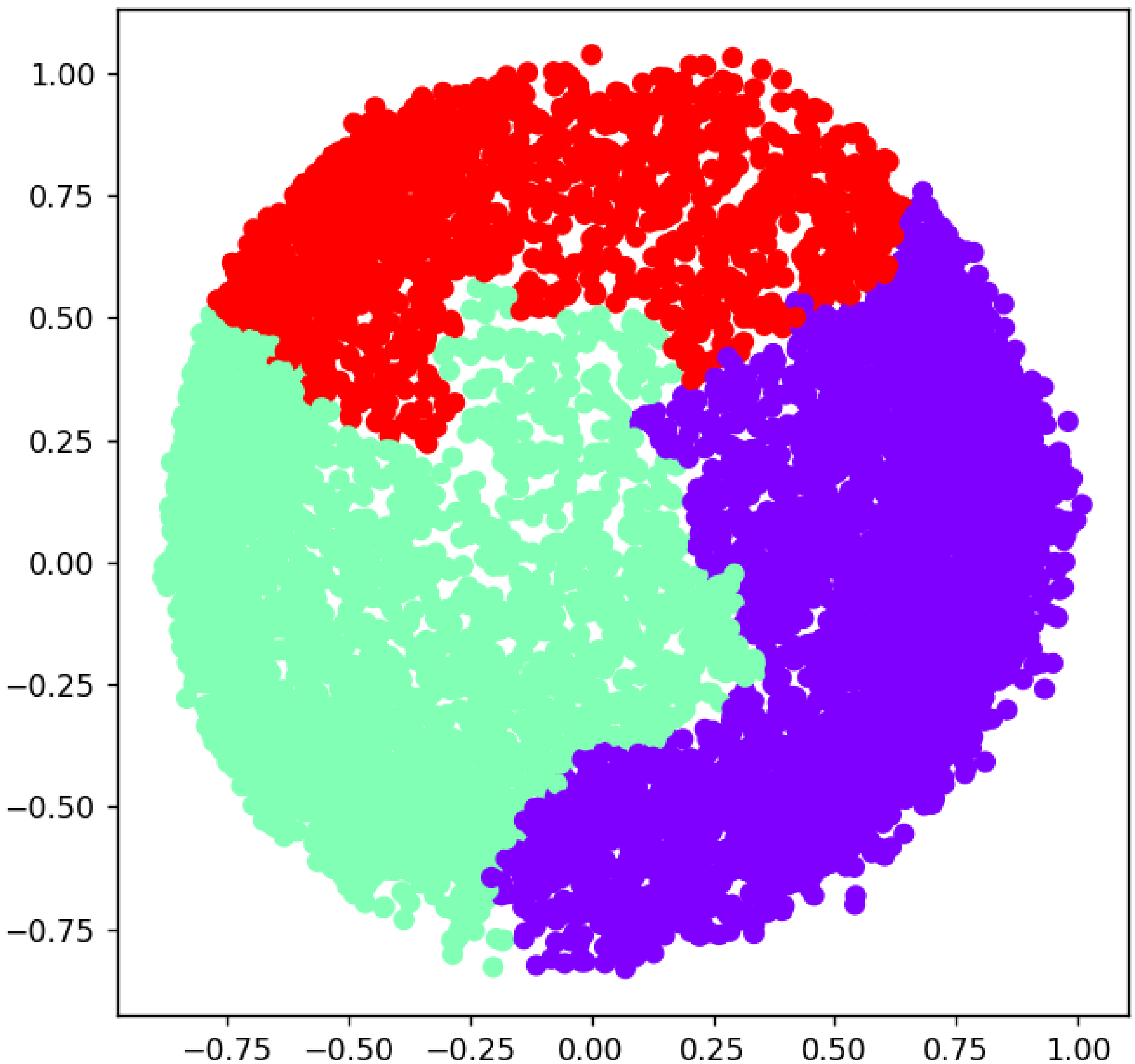
```
BALANCE                             False
BALANCE_FREQUENCY                   False
PURCHASES                           False
ONEOFF_PURCHASES                    False
INSTALLMENTS_PURCHASES              False
CASH_ADVANCE                        False
PURCHASES_FREQUENCY                 False
ONEOFF_PURCHASES_FREQUENCY          False
PURCHASES_INSTALLMENTS_FREQUENCY    False
CASH_ADVANCE_FREQUENCY              False
CASH_ADVANCE_TRX                    False
PURCHASES_TRX                       False
CREDIT_LIMIT                        False
PAYMENTS                            False
MINIMUM_PAYMENTS                    False
PRC_FULL_PAYMENT                    False
TENURE                              False
dtype: bool
<pandas.io.formats.style.Styler object at 0x00000178D9466230>
         P1        P2  TENURE
0 -0.488186 -0.677233      12
1 -0.517294  0.556075      12
2  0.334384  0.287313      12
3 -0.486616 -0.080781      12
4 -0.562175 -0.474770      12
Text(0, 0.5, 'pc2')
```



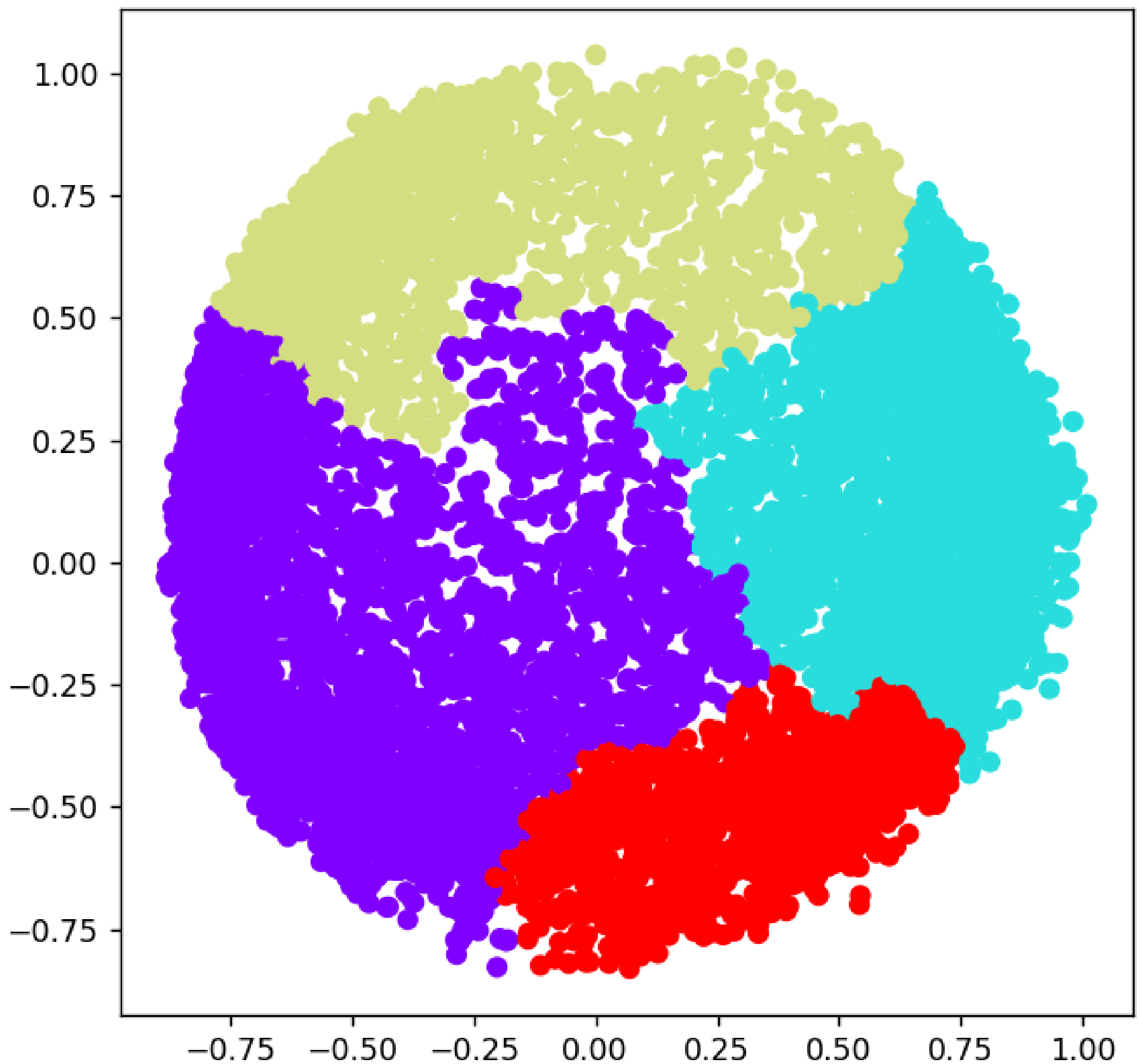
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
pc2
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00
pc1

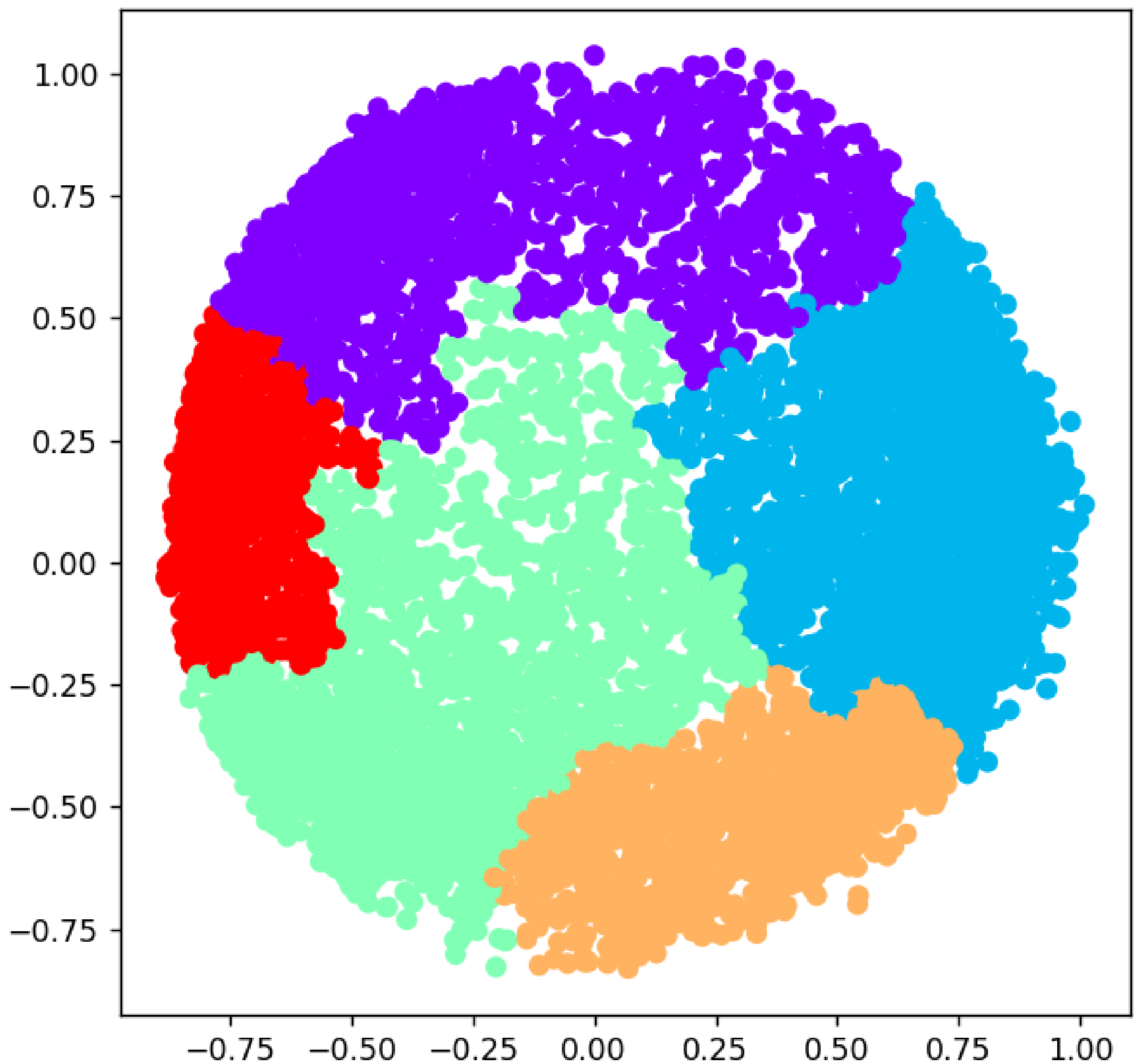


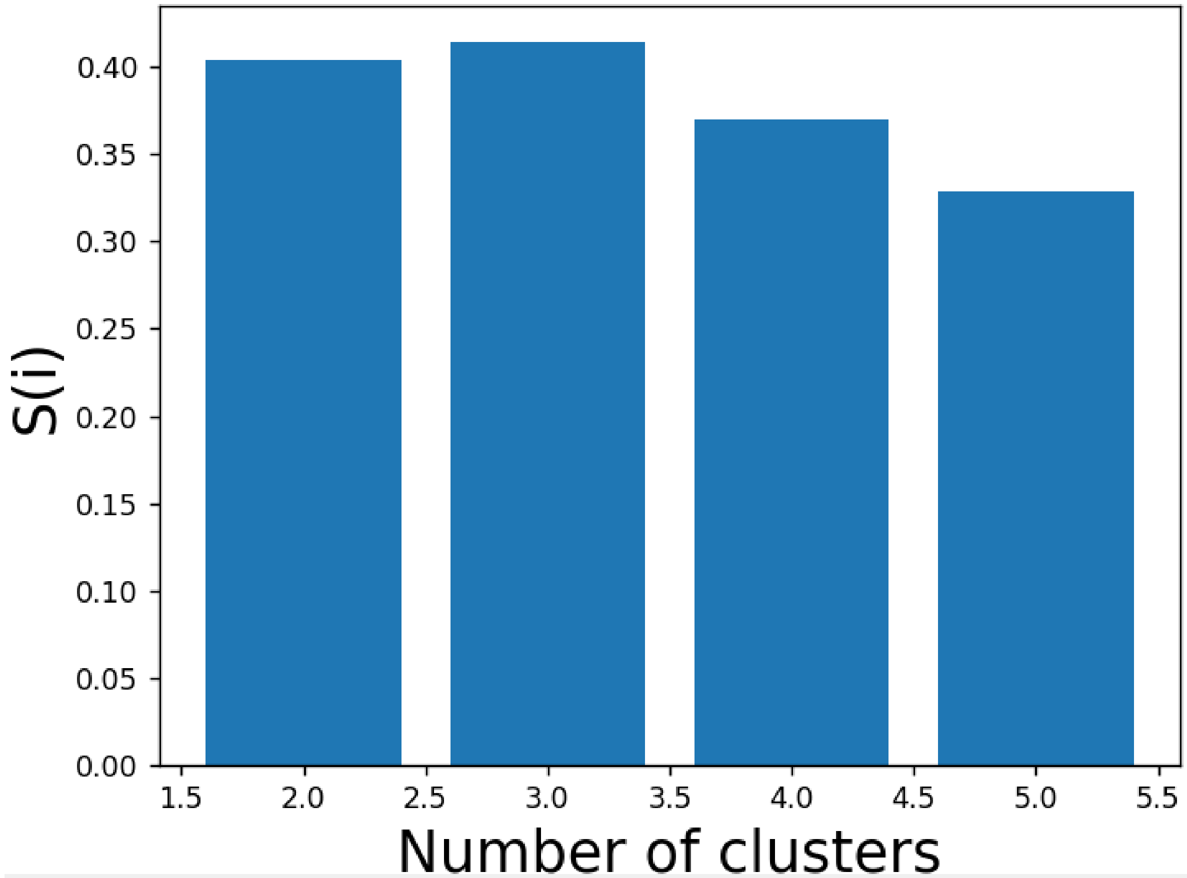
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00



1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00


1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00


1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00


S(i)
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
5.5
Number of clusters

**Related Links:**

SourceCode:

https://github.com/VijayTarakaRamarao/ML/tree/main/Assignment6

Recording:

https://github.com/VijayTarakaRamarao/ML/blob/main/Assignment4/MachineLearning_Assignment6.mp4