Internship Weekly Report - Week 6

Name: Sandeep Ravaji Patel

Domain: Data Science

Week Number: Week 6

Objective:

To understand clustering and dimensionality reduction techniques, specifically focusing on implementing K-Means clustering and performing PCA using Scikit-Learn.

Tasks Completed:

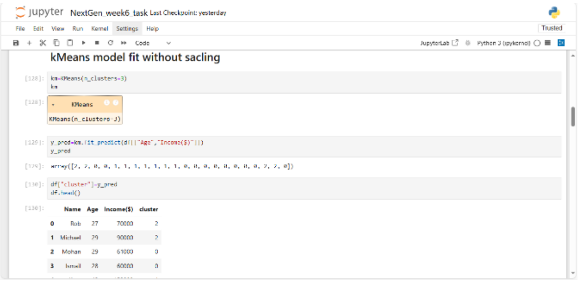
- Learned about unsupervised learning and its applications.
- Understood the concept of clustering and the K-Means algorithm.
- Implemented K-Means clustering using Scikit-Learn's KMeans class.
- Learned about dimensionality reduction and Principal Component Analysis (PCA).
- Performed PCA using Scikit-Learn's PCA class to reduce the dimensionality of a dataset.
- Visualized clusters and the results of PCA.
- Evaluated the performance of K-Means clustering.

Tools Used:

- Scikit-Learn
- Pandas
- NumPy
- Matplotlib
- Jupyter Notebook

Code Snippets / Design Screenshots

Example 1: K-Means Clustering Implementation


jupyter
NextGen_week6_task Last Checkpoint: yesterday
File Edit View Run Kernel Settings Help
Trusted
Code
JupyterLab
Python 3 (ipykernel)

kMeans model fit without sacling

```
[128]: km=KMeans(n_clusters=3)
       km
```

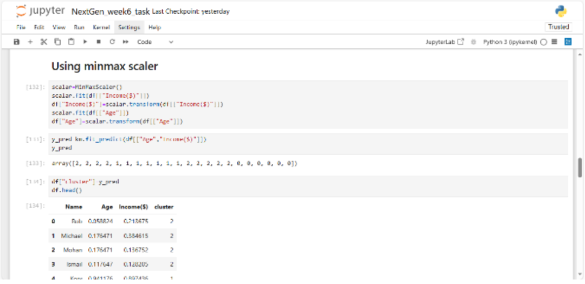
```
[128]: KMeans
       KMeans(n_clusters=3)
```

```
[129]: y_pred=km.fit_predict(df[["Age","Income($)"]])
       y_pred
```

```
[129]: array([2, 2, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 2, 2, 0])
```

```
[130]: df["cluster"]=y_pred
       df.head()
```

```
[130]:     Name  Age  Income($)  cluster
       0    Rob   27      70000        2
       1 Michael  29      90000        2
       2  Mohan   29      61000        0
       3  Ismail  28      60000        0
```


jupyter
NextGen_week6_task Last Checkpoint: yesterday
File Edit View Run Kernel Settings Help
Trusted
Code
JupyterLab
Python 3 (ipykernel)

Using minmax scaler

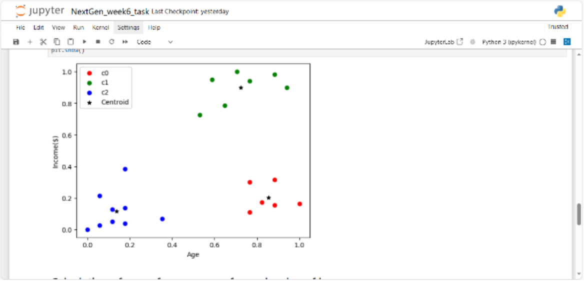
```
[132]: scalar=MinMaxScaler()
       scalar.fit(df[["Income($)"]])
       df["Income($)"]=scalar.transform(df[["Income($)"]])
       scalar.fit(df[["Age"]])
       df["Age"]=scalar.transform(df[["Age"]])
```

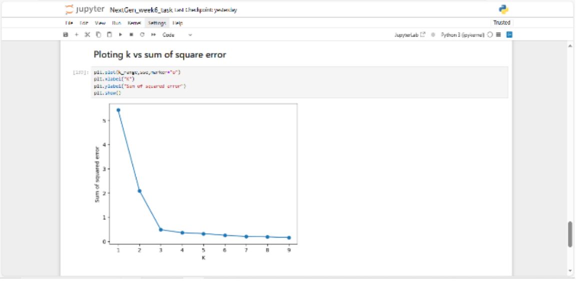
```
[133]: y_pred=km.fit_predict(df[["Age","Income($)"]])
       y_pred
```

```
[133]: array([2, 2, 2, 2, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 0, 0, 0, 0, 0, 0])
```

```
[134]: df["cluster"]=y_pred
       df.head()
```

```
[134]:     Name       Age  Income($)  cluster
       0     Rob  0.058824   0.213675        2
       1 Michael  0.176471   0.384615        2
       2   Mohan  0.176471   0.136752        2
       3  Ismail  0.117647   0.128205        2
       4    Kory  0.941176   0.897436        1
```

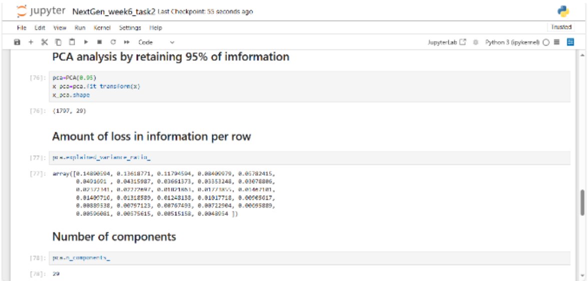

jupyter NextGen_week6_task Last Checkpoint: yesterday
File Edit View Run Kernel Settings Help
Trusted
Code
JupyterLab
Python 3 (ipykernel)
c0
c1
c2
Centroid
Income($)
Age
0.0
0.2
0.4
0.6
0.8
1.0


jupyter NextGen_week6_task Last Checkpoint: yesterday
File Edit View Run Kernel Settings Help
Trusted
Code
JupyterLab
Python 3 (ipykernel)
Sum of squared error
K
0
1
2
3
4
5
6
7
8
9

Ploting k vs sum of square error

```
plt.plot(k_range,sse,marker="o")
plt.xlabel("K")
plt.ylabel("Sum of squared error")
plt.show()
```

Example 2: PCA Implementation and Results


jupyter NextGen_week6_task2 Last Checkpoint: 55 seconds ago
File Edit View Run Kernel Settings Help
Trusted
Code
JupyterLab
Python 3 (ipykernel)

PCA analysis by retaining 95% of imformation

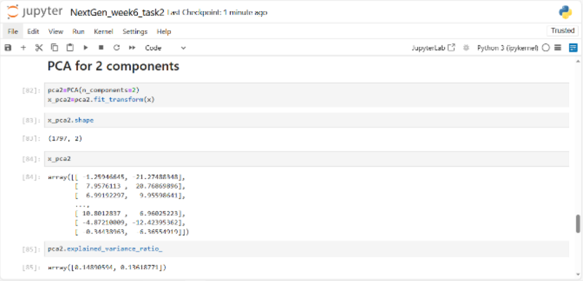
```
[76]: pca=PCA(0.95)
      x_pca=pca.fit_transform(x)
      x_pca.shape

[76]: (1797, 29)
```

Amount of loss in information per row

```
[77]: pca.explained_variance_ratio_

[77]: array([0.14890594, 0.13618771, 0.11794594, 0.08409979, 0.05782415,
             0.0491601 , 0.04315987, 0.03661373, 0.03353248, 0.03078806,
             0.02372341, 0.02272697, 0.01821863, 0.01773855, 0.01467101,
             0.01409716, 0.01318589, 0.01248138, 0.01017718, 0.00905617,
             0.00889538, 0.00797123, 0.00767493, 0.00722904, 0.00695889,
             0.00596081, 0.00575615, 0.00515158, 0.0048954 ])
```

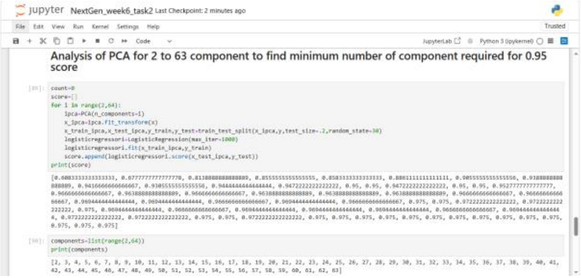
Number of components

```
[78]: pca.n_components_

[78]: 29
```


jupyter NextGen_week6_task2 Last Checkpoint: 1 minute ago
File Edit View Run Kernel Settings Help
Trusted
Code
JupyterLab
Python 3 (ipykernel)

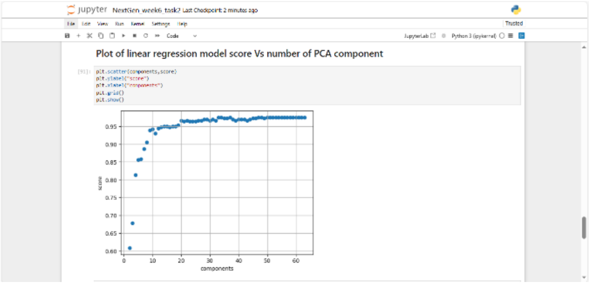
PCA for 2 components

```
[82]: pca2=PCA(n_components=2)
      x_pca2=pca2.fit_transform(x)

[83]: x_pca2.shape

[83]: (1797, 2)

[84]: x_pca2

[84]: array([[ -1.25946645, -21.27488348],
             [  7.9576113 ,  20.76869896],
             [  6.99192297,   9.95598641],
             ...,
             [ 10.8012837 ,   6.96025223],
             [ -4.87210809, -12.42395362],
             [ -0.34438963,  -6.36554919]])

[85]: pca2.explained_variance_ratio_

[85]: array([0.14890594, 0.13618771])
```


jupyter NextGen_week6_task2 Last Checkpoint: 2 minutes ago
File Edit View Run Kernel Settings Help
Trusted
Code
JupyterLab
Python 3 (ipykernel)

Analysis of PCA for 2 to 63 component to find minimum number of component required for 0.95 score

```
[89]: count=0
      score=[]
      for i in range(2,64):
          ipca=PCA(n_components=i)
          x_ipca=ipca.fit_transform(x)
          x_train_ipca,x_test_ipca,y_train,y_test=train_test_split(x_ipca,y,test_size=.2,random_state=30)
          logisticregressori=LogisticRegression(max_iter=1000)
          logisticregressori.fit(x_train_ipca,y_train)
          score.append(logisticregressori.score(x_test_ipca,y_test))
      print(score)

[0.6083333333333333, 0.6777777777777778, 0.8138888888888889, 0.8555555555555555, 0.8583333333333333, 0.8861111111111111, 0.9055555555555556, 0.9388888888
888889, 0.9416666666666667, 0.9305555555555556, 0.9444444444444444, 0.9472222222222222, 0.95, 0.95, 0.9472222222222222, 0.95, 0.95, 0.9527777777777777,
0.9666666666666667, 0.9638888888888889, 0.9666666666666667, 0.9638888888888889, 0.9638888888888889, 0.9638888888888889, 0.9666666666666667, 0.9666666666
666667, 0.9694444444444444, 0.9694444444444444, 0.9666666666666667, 0.9694444444444444, 0.9666666666666667, 0.975, 0.975, 0.9722222222222222, 0.9722222222
222222, 0.975, 0.9694444444444444, 0.9666666666666667, 0.9694444444444444, 0.9694444444444444, 0.9694444444444444, 0.9666666666666667, 0.969444444444444
4, 0.9722222222222222, 0.9722222222222222, 0.975, 0.975, 0.9722222222222222, 0.975, 0.975, 0.975, 0.975, 0.975, 0.975, 0.975, 0.975, 0.975, 0.975, 0.975,
0.975, 0.975, 0.975]

[90]: components=list(range(2,64))
      print(components)

[2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41,
42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63]
```



Plot of linear regression model score Vs number of PCA component

```
[91]: plt.scatter(components,score)
      plt.ylabel("score")
      plt.xlabel("components")
      plt.grid()
      plt.show()
```

Challenges Faced

1. **Understanding the mathematical concepts behind PCA:**
   - Initially found it challenging to grasp the concepts of eigenvectors, eigenvalues, and variance explained.
   - **Resolution:** Reviewed linear algebra concepts and PCA tutorials, and experimented with different datasets to visualize the effects of PCA.
2. **Choosing the optimal number of clusters for K-Means:**
   - Was unsure about how to determine the best value for 'k' in K-Means clustering.
   - **Resolution:** Implemented the elbow method to identify the optimal 'k' by plotting the within-cluster sum of squares for different values of 'k' and selecting the 'k' at the "elbow" point.

Learning Outcome

- Developed a solid understanding of unsupervised learning techniques, including clustering and dimensionality reduction.

- Implemented K-Means clustering and PCA using Scikit-Learn.
- Gained experience in applying these techniques to real-world datasets.
- Improved ability to analyze and interpret the results of clustering and dimensionality reduction.
- Learned how to visualize clusters and the effects of PCA.

Resources

- Scikit-Learn Documentation: https://scikit-learn.org/stable/
- Clustering Guide
- PCA Guide