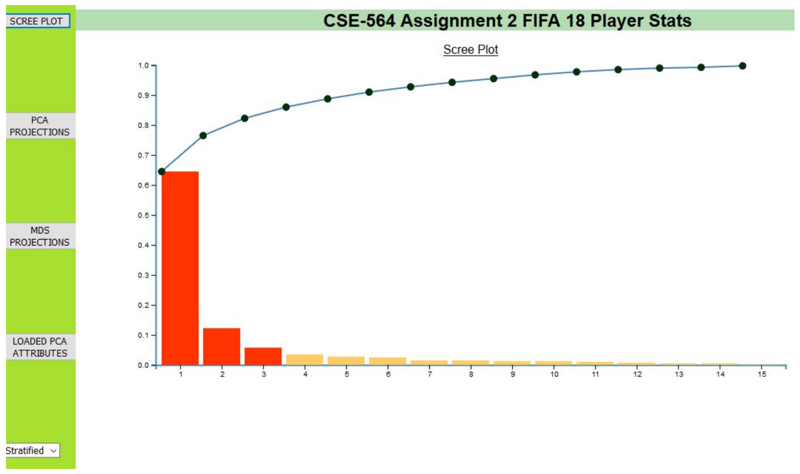
CSE 564 Assignment 2

Sayak Ghosh (112025780)



UI Elements

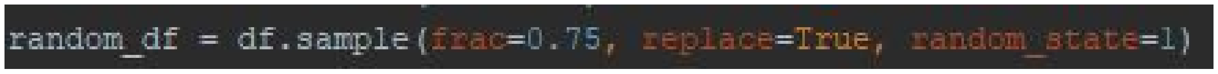
- Scree Plot (Button) - Displays the scree plot of the PCA components.
- PCA Projections(Button) - Displays the data projected on the two most important PCA components.
- MDS Projections(Button) - Displays the data projected on the major MDS components.
- Loaded PCA Attributes(Button) - Shows the most important features derived from the data.

Dataset: I used the FIFA 2018 players dataset found here. The original dataset contains 1839 rows and 15 attributes. All the columns has a range of 1-100.

Client Server: The server side rests on a Flask Framework . The visualizations on the client side are being rendered through D3 library in javascript.

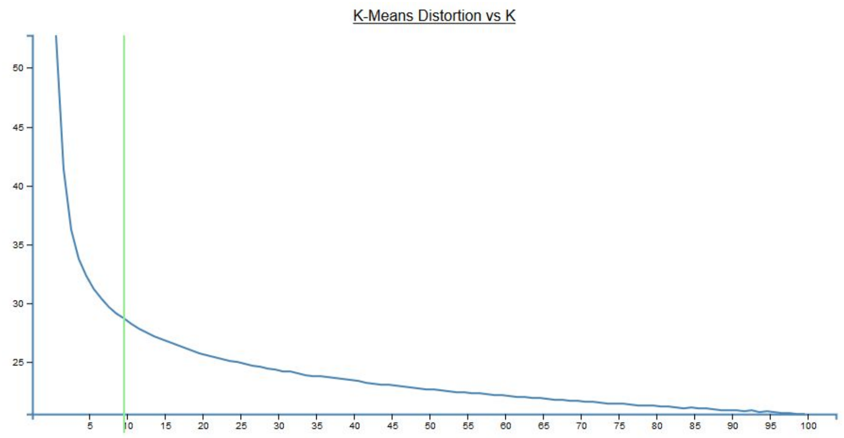
Task 1: Data clustering and decimation

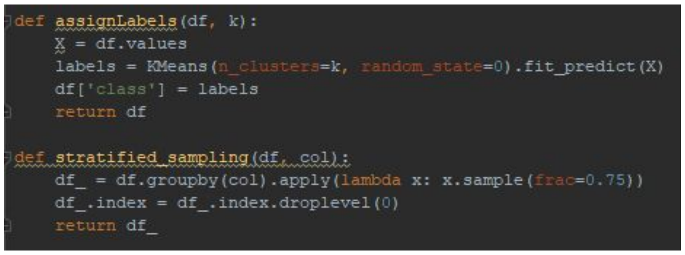
- Random Sampling - The data is sampled with replacement. Sampling frequency is 0.75



```
random_df = df.sample(frac=0.75, replace=True, random_state=1)
```

- Stratified Sampling - Cluster the data using K-Means data. I found out the optimal k for k-means by plotting the distortion against a range of values of K. The K for the corresponding elbow is chosen as the optimal number of clusters. The optimal number of clusters is chosen to be **10**. The data is clustered again using k-means with a k-value of 10. The data is then labeled with their nearest cluster centre followed by which they are grouped by their label. Uniform fraction of samples are then chosen for each group.



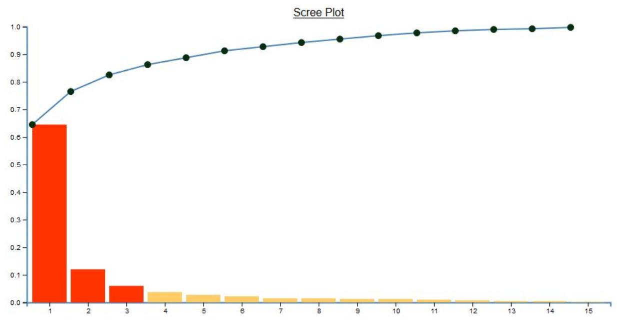


```
def assignLabels(df, k):
    X = df.values
    labels = KMeans(n_clusters=k, random_state=0).fit_predict(X)
    df['class'] = labels
    return df

def stratified_sampling(df, col):
    df_ = df.groupby(col).apply(lambda x: x.sample(frac=0.75))
    df_.index = df_.index.droplevel(0)
    return df_
```

Code Snippet for stratified sampling.

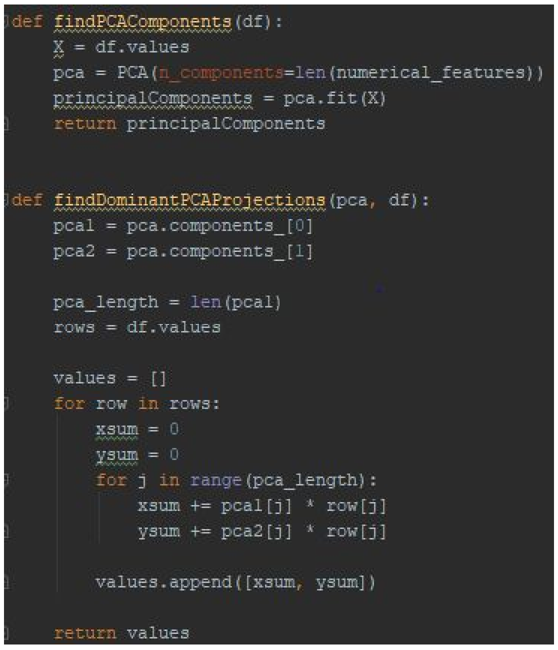
Task 2: Dimension Reduction

I have used PCA for reducing the number of dimensions of the data. PCA decomposition gives a number of PCA components . Each PCA component is a linear combination of the features of the data. Each PCA component is associated with an eigen value that determines the strength of that PCA component. The eigen values of the PCA components should add up to 1. I plot the scree plot to determine the number of components (intrinsic dimensionality) required to explain more than 80 percent of the variance in data.



As we can see here that **3** PCA components explain more than 80 percent of the variance in the data. Hence the intrinsic dimensionality is **3**.

Code snippet for finding PCA components and highest PCA loadings.



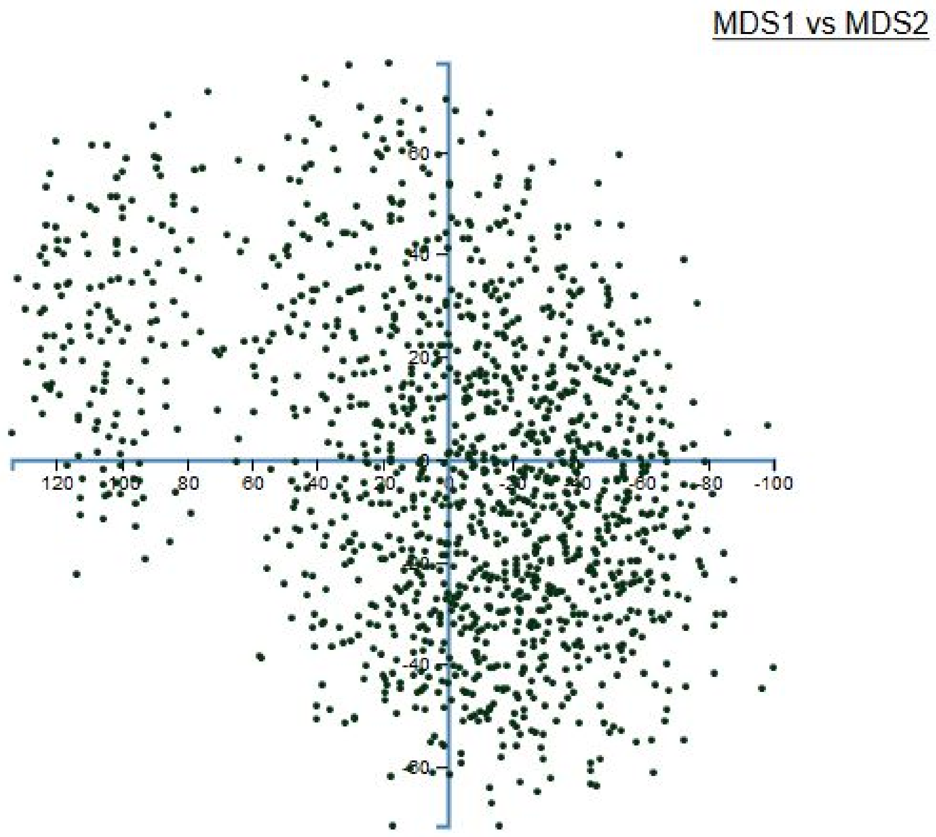
```
def findPCAComponents(df):
    X = df.values
    pca = PCA(n_components=len(numerical_features))
    principalComponents = pca.fit(X)
    return principalComponents


def findDominantPCAProjections(pca, df):
    pca1 = pca.components_[0]
    pca2 = pca.components_[1]

    pca_length = len(pca1)
    rows = df.values

    values = []
    for row in rows:
        xsum = 0
        ysum = 0
        for j in range(pca_length):
            xsum += pca1[j] * row[j]
            ysum += pca2[j] * row[j]

        values.append([xsum, ysum])

    return values
```

The attributes with the highest PCA loadings signifies the most important features in the data. They are illustrated below

The data is reconstructed the two highest PCA components and is projected on to the PCA components.

Task 3
Visualize the data projected into the top two PCA vectors via 2D scatterplot

Visualize the data via MDS (Euclidian & correlation distance) in 2D scatterplots



Interesting Observations:
As we can see from both the above 2 diagrams , there are 2 clusters . One big dense cluster to the the right and one smaller less dense cluster to the left.

Demo Link
https://youtu.be/oV0vFg56OMI