School of Engineering and Applied Science (SEAS), Ahmedabad University

B.Tech (CSE Semester VI):
Machine Learning (CSE 523)

Project Submission #3: Principal Component Analysis

Submission Deadline: April 05, 2020 (11:59 PM)

- **Group No.: 8**
- **Project Domain: Natural Language Processing**
- **Project Title: Sentimental Analysis on movie reviews**
- **Name of the group members:**
  1. Aditya Shah (1741007)
  2. Dhruvil Shah (1741024)
  3. Nilay Patel (1741038)
  4. Varun Patel (1741080)

Contents

1 Implementation code **2**
1.1 Code for features . . . 2
1.2 Code for PCA . . . 3

2 URL links **6**
2.1 Main folder . . . 6
2.2 Codes . . . 6
2.3 Data set . . . 6

3 Inference **7**
3.1 What we have done and why is it important? . . . 7
3.1.1 What we have done?: . . . 7
3.1.2 Why is PCA important?: . . . 7
3.2 How we have implemented? . . . 7
3.3 Analysis and Results . . . 7
3.3.1 Analysis: . . . 7
3.3.2 Results: . . . 9
3.3.3 Inference: . . . 13

1 Implementation code

1.1 Code for features

```
import pandas as pd
import numpy as np
from nltk.tokenize import word_tokenize
from nltk import pos_tag
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
from sklearn.preprocessing import LabelEncoder
from collections import defaultdict
from nltk.corpus import wordnet as wn
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn import model_selection, naive_bayes, svm
from sklearn.metrics import accuracy_score

np.random.seed(500)  #random seed
Corpus_csv = pd.read_csv("imdb_tr.csv",encoding='latin-1') #read csv file

Corpus_csv['text'].dropna(inplace=True)# All text to lower case. This is required as python interprets 'dog' and 'DOG' differently
Corpus_csv['text'] = [entry.lower() for entry in Corpus_csv['text']]# broken into words
Corpus_csv['text']= [word_tokenize(entry) for entry in Corpus_csv['text']]# Remove Stop words, Non-Numeric and perfom Word Stemming/Lemmenting.#
WordNetLemmatizer requires Pos tags to understand if the word is noun or verb or adjective etc. By default it is set to Noun
tagmap = defaultdict(lambda : wn.NOUN) #default  dictionary
tagmap['J'] = wn.ADJ
tagmap['V'] = wn.VERB
tagmap['R'] = wn.ADV
for index,entry in enumerate(Corpus_csv['text']):
    # Declaring Empty List to store the words that follow the rules for this step
    Final_words = []
    # Initializing WordNetLemmatizer() converting to its base form
    word_Lemmatized = WordNetLemmatizer()
    # pos_tag function below will provide the 'tag' i.e if the word is Noun(N) or Verb(V) or something else.
    for word, tag in pos_tag(entry):
        # Below condition is to check for Stop words and consider only alphabets
        if word not in stopwords.words('english') and word.isalpha():
            word_Final = word_Lemmatized.lemmatize(word,tagmap[tag[0]])
            Final_words.append(word_Final)
    # The final processed set of words for each iteration will be stored in 'text_final'
    Corpus_csv.loc[index,'text_final'] = str(Final_words)

Train_X, Test_X, Train_Y, Test_Y = model_selection.train_test_split(Corpus_csv['text_final'],Corpus_csv['polarity'],test_size=0.3)

Encoder_le = LabelEncoder() #convert to machine readable form
Train_Y = Encoder_le.fit_transform(Train_Y)
Test_Y = Encoder_le.fit_transform(Test_Y)

Tfidf_vect = TfidfVectorizer(max_features=5000) #to convert text into features
Tfidf_vect.fit(Corpus_csv['text_final'])
Train_X_Tfidf = Tfidf_vect.transform(Train_X) #to transform and replace it with new
Test_X_Tfidf = Tfidf_vect.transform(Test_X)  #to transform and replace it with new
```

```

print(Tfidf_vect.vocabulary_) #print new features
print(Train_X_Tfidf) #print occurence

```

1.2 Code for PCA

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import LabelEncoder
from sklearn.decomposition import PCA


df=pd.read_csv("/home/dhruvil/Downloads/pca_features_1.txt", delimiter=',', header=None, skiprows=0,
names=['Feature_name_1','Feature_name_2','Feature_name_3','Feature_name_4','Value'])


labe_Enc = LabelEncoder()                    #array or object declared to label encoder library
labe_Enc.fit(df['Feature_name_1'].astype(str))   #fitting of first column into le in form of string
df['Feature_name_1'] = labe_Enc.transform(df['Feature_name_1'].astype(str)) # transform column and replace old with new

labe_Enc.fit(df['Feature_name_2'].astype(str))  #fitting  of  second column into le  in form of string
df['Feature_name_2'] = labe_Enc.transform(df['Feature_name_2'].astype(str)) # transform column and replace old with new

labe_Enc.fit(df['Feature_name_3'].astype(str))   #fitting  of  third column into le  in form of string
df['Feature_name_3'] = labe_Enc.transform(df['Feature_name_3'].astype(str))# transform column and replace old with new

labe_Enc.fit(df['Feature_name_4'].astype(str)) #fitting  of  fourth column into le  in form of string
df['Feature_name_4'] = labe_Enc.transform(df['Feature_name_4'].astype(str)) # transform column and replace old with new



features = ['Feature_name_1','Feature_name_2','Feature_name_3','Feature_name_4']# Separating out the features
x = df.loc[:, features].values# Separating out the target
y = df.loc[:,['Value']].values  #assigning the target values to variable
x = StandardScaler().fit_transform(x)


pca = PCA(n_components=4)  #number of components
principalComponents = pca.fit_transform(x) # fitting and transforming features
principalDf = pd.DataFrame(data = principalComponents
             , columns = ['principal component 1','principal component 2','principal component 3','principal component 4'])
principalDf  #dataframe(2-D structure) output

```

```
pc1 = []  #new array or new list
pc1.append([principalDf['principal component 1']])   #entering first column of
    dataframe into new aray

pc2 = []   #new array or new list
pc2.append([principalDf['principal component 2']]) #entering second column of
    dataframe into new aray

pc3 = []   #new array or new list
pc3.append([principalDf['principal component 3']])  #entering third column of
    dataframe into new aray

pc4 = []   #new array or new list
pc4.append([principalDf['principal component 4']])  #entering fourth column of
    dataframe into new aray


fig = plt.figure(figsize = (10,10))
ax = fig.add_subplot(1,1,1)
t = ['Pricipal component 1,Principal comonent 2','Principal comonent 3,Principal
    comonent 4']
ax.grid()
ax.scatter(pc1,pc2)  #plotting collection of point of first two column
ax.scatter(pc3,pc4)   # plotting collection of point of last two column

ax.legend(t)
plt.show()

string_list = [] # empty list to contain columns with strings (words)
for colname, colvalue in principalDf.iteritems():
    if type(colvalue[1]) == str: #matching type as string
         string_list.append(colname)
# Get to the numeric columns by inversion
num_list = principalDf.columns.difference(string_list)

movie_num = principalDf[num_list]
#del movie # Get rid of movie df as we won't need it now
movie_num.head() #printing top 5 rows

movie_num = movie_num.fillna(value=0, axis=1)

X = movie_num.values
# Normalize the data for mean and variance
from sklearn.preprocessing import StandardScaler
X_std = StandardScaler().fit_transform(X)  #to make it easy for mean and variance
principalDf.plot(y= 'principal component 2', x ='principal component 1',kind='
    hexbin',gridsize=25, sharex=False, colormap='cubehelix', title='Hexbin of
    principal component')
principalDf.plot(y= 'principal component 4', x ='principal component 3',kind='
    hexbin',gridsize=25, sharex=False, colormap='cubehelix', title='Hexbin of
    principal component')

mean_vector = np.mean(X_std, axis=0) #mean
cov_matrix = np.cov(X_std.T)  #covariance
eigen_value, eigen_vector = np.linalg.eig(cov_matrix)  # to find eigenvalue and
    eigenvector

# Create a list of (eigenvalue, eigenvector) tuples
eig_pairs = [ (np.abs(eigen_value[i]),eigen_vector[:,i]) for i in range(len(
    eigen_value))]
```

```
91
92 # Sort from high to low
93 eig_pairs.sort(key = lambda x: x[0], reverse= False)
94
95 # Calculation of Explained Variance from the eigenvalues
96 tot = sum(eigen_value) #sum of eigenvalue
97 var_exp = [(i/tot)*100 for i in sorted(eigen_value, reverse=True)] # Individual
    explained variance
98 cum_var_exp = np.cumsum(var_exp) # Cumulative explained variance
99
100 # PLOT OUT THE EXPLAINED VARIANCES SUPERIMPOSED
101 plt.figure(figsize=(8, 5))
102
103 # plt.bar(range(4), var_exp, alpha=0.333, align='center', label='individual
    explained variance', color = 'g')
104 plt.step(range(4), cum_var_exp, where='mid',label='cumulative explained variance')
105 plt.ylabel('Explained variance ratio')
106 plt.xlabel('Principal components')
107 plt.title('Variance vs components')
108 plt.legend(loc='best')
109 plt.show()
110
111 # number of cluster = 4
112 x_n4d = pca.fit_transform(X_std)  # to fit and transform it to give normalized
    value
113 kmeans = KMeans(n_clusters=4)
114 # Compute cluster centers and predict cluster indices
115 X_clustered = kmeans.fit_predict(x_n4d) # find cluster indices and center
116
117 # Define our own color map
118 LABEL_COLOR_MAP = {0 : 'r',1 : 'g',2 : 'b',3:'y'}
119 label_color = [LABEL_COLOR_MAP[l] for l in X_clustered] #to asign each cluster a
    colour
120
121 # Plot the scatter digram
122 plt.figure(figsize = (7,7))
123 plt.scatter(x_n4d[:,0],x_n4d[:,3], c= label_color, alpha=0.5)
124 plt.title('Scattering of kmeans of principal components')
125 plt.show()
126
127 df = pd.DataFrame(x_n4d) #dataframe
128 df = df[[0,1,2,3]] #  visualise relationships between first 4 columns
129 df['X_cluster'] = X_clustered  #assigning  variable
130 #  Seaborn's pairplot  is called to visualize our KMeans clustering on the PCA
    projected data
131 sns.pairplot(df, hue='X_cluster',diag_kind='kde',height=1.85) #plot pairwise
    relationship
132 plt.title('Pairwise relationship of PCA projected data through KMeans')
133 plt.show() #showing the plot
134
```

2 URL links

2.1 Main folder

https://drive.google.com/open?id=1zJF4_tj4-jX2qp1HNJauWNnaDsINtP9j

2.2 Codes

https://drive.google.com/open?id=11oXoiSiT6Wav5xcZCQNgYa3kdlHNcxSE

2.3 Data set

https://drive.google.com/open?id=119LxZQbiJKkvyC82IfHlpTqyMSD3Pv2b

3 Inference

3.1 What we have done and why is it important?

3.1.1 What we have done?:

We have taken dataset of movie reviews. From that dataset, we have found features and store in a file. We have separate out the features and target values. Then we have found out Principal Component Analysis (PCA) through eigenvalues, eigenvectors, mean, covariance. Then we have used K-means clustering to plot the pairwise relationship of projected data(Seaborn pairplot).

3.1.2 Why is PCA important?:

PCA is important to reduce the problem of overfitting. Overfitting is caused when the no of features used in the training data set for training the model are very large in number. This can lead to creation of a model which is overly generalized. Therefore in many test cases it can give false results thereby declining the accuracy rate and increasing the error rate. PCA which is a technique of dimensionality reduction helps to extract out the most relevant features and removes the unnecessary or less relevant features.Thus it helps to build a model which is well generalized giving high accuracy rate.

3.2 How we have implemented?

We have taken dataset. We have converted all text into one case. Then we have declared empty list in which text after removing stopwords will be appended to that list. List will contain noun, adjective, verb, adverb. Then implemented train-test spilt on list and polarity. We have used label encoder (to convert it into machine readable form) and then used tfidf vectorizer that transform text to feature vector. Features are found and stored in file. We have seperate features and value. We have formed Dataframe. We have separate out string list and used only number list. We have standardized and normalized and stored it into variable. That variable is used to find mean, variance, eigenvalues, eigenvectosr, individual and cumultative explained variance. At last Kmeans clustering is applied. This is used to find cluster center and indices and then plot the graph. Then finally we have shown pairwise relationship of our PCA projected data(seaborn pairplot).

3.3 Analysis and Results

3.3.1 Analysis:

Consider a d-dimensional dataset containing n records.
Firstly, standardize the given dataset.
Now, compute the covariance matrix C or $\sum$, where $\sum \epsilon R^{dXd}$.

The covariance between any two features x_i and x_j o given by,

$\text{cov}(x_i , x_j) = \sigma_j = \frac{1}{n} \sum_{k=1}^{k=n}(x_i - \mu_i)(x_j - \mu_j)$

Here, μ_i and μ_j are sample means of features i and j respectively.

For the given dataset, $\sum$ will be,

$$\sum = \begin{bmatrix} \sigma_1^2 & \sigma_{12} & . & . & . & \sigma_{1d} \\ \sigma_{21} & \sigma_2^2 & . & . & . & \sigma_{2d} \\ . & . & . & & & . \\ . & . & & . & & . \\ . & . & & & . & . \\ \sigma_{d1} & \sigma_{d2} & . & . & . & \sigma_d^2 \end{bmatrix}$$

Next, obtain the eigenvalues and eigenvectors for the corresponding covariance matrix.
Suppose an eigen vector v which satisfies the following conditions:

$\sum$ v = λv
$\sum$ v = λIv
$\sum$ v - λIv = 0
($\sum$ - λI)v = 0.....(1)
This is similar to homogeneous system of linear equations.

For $\vec{v} \neq \vec{0}$ eq.(1) to be true.
$|\sum - \lambda I| = 0$
Solving the determinant we obtain d values of λ which represent eigenvalues corresponding to d eigenvalues. Corresponding to d eigenvalues we get d eigenvectors by putting value of λ in equation (1) and solving for v.

Now, out of d eigenvectors, select k eigenvectors corresponding to k highest eigenvalues.
Now, take these k eigenvectors in decreasing order of their corresponding eigenvalues and construct a projection matrix W, where W ϵR^{dXk}

The k columns of W represent principal components. Using W, we can transform a sample vector x into the PCA subspace obtaining $x^{'}$,

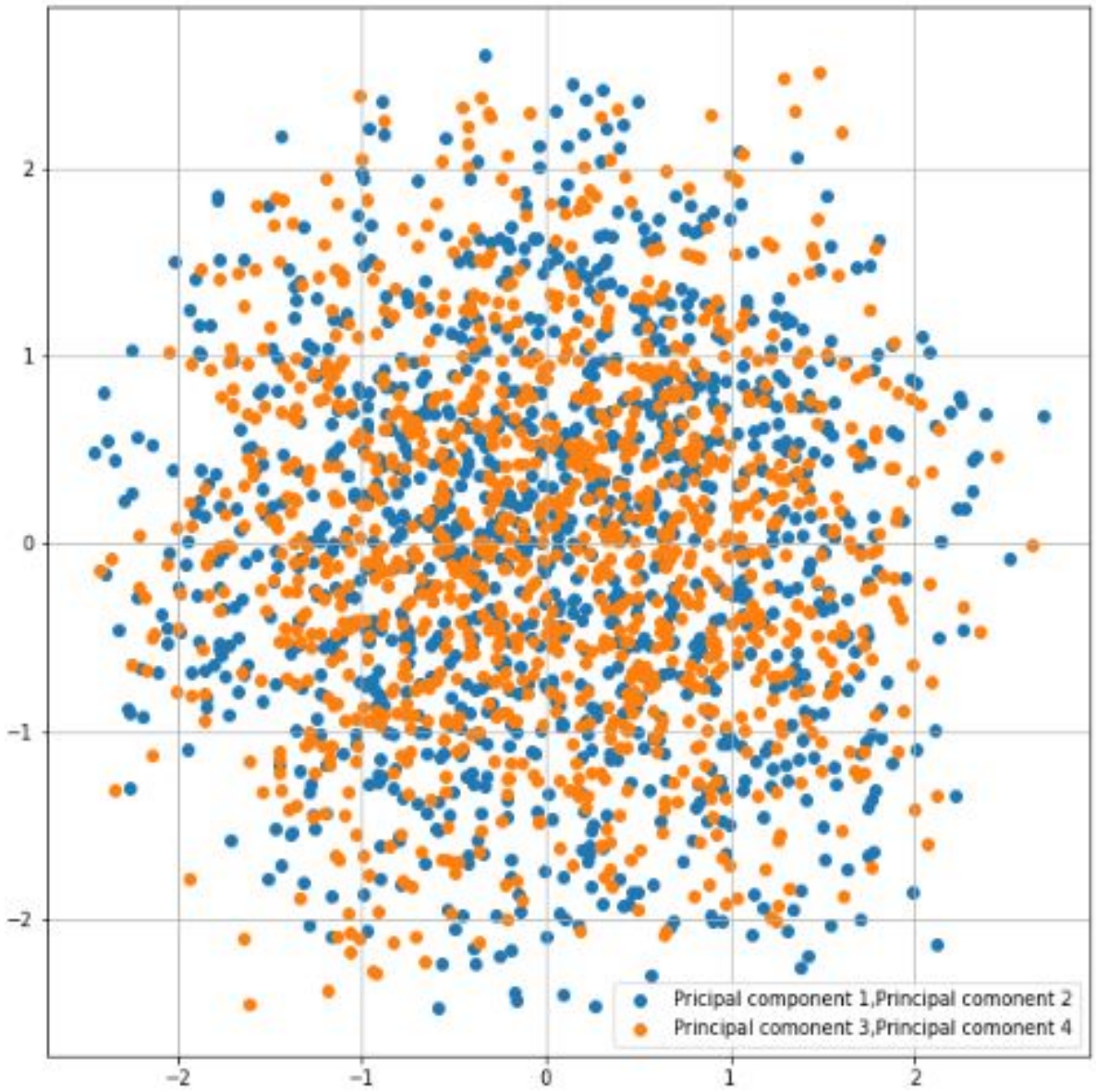
$x^{'}$ = x W
1Xk 1Xd dXk

Similarly, we can transform the entire dataset onto k principal components by calculating the matrix product,

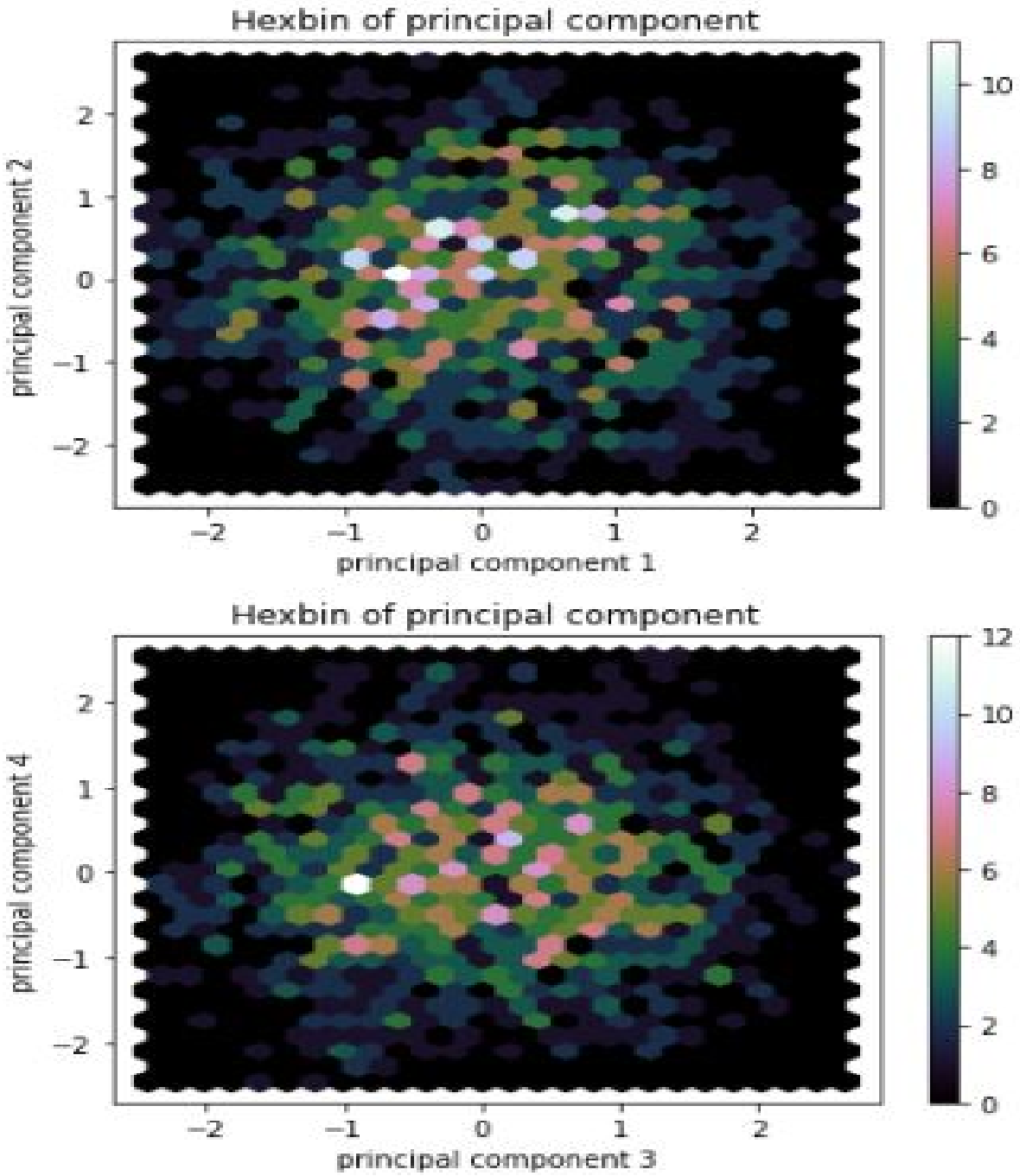
$X^{'}$ = XW
nXk nXd dXk

The dataset reduced to k features. Thus, our goal of dimensionality reduction is achieved.

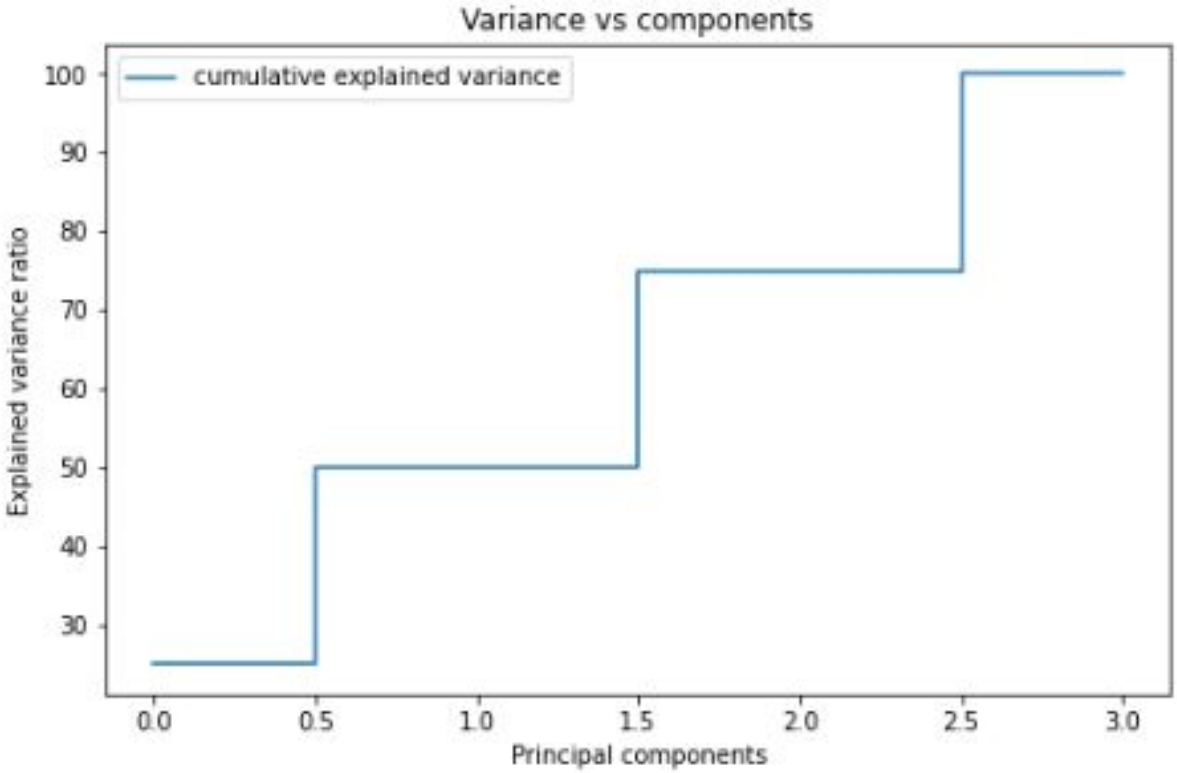
3.3.2 Results:



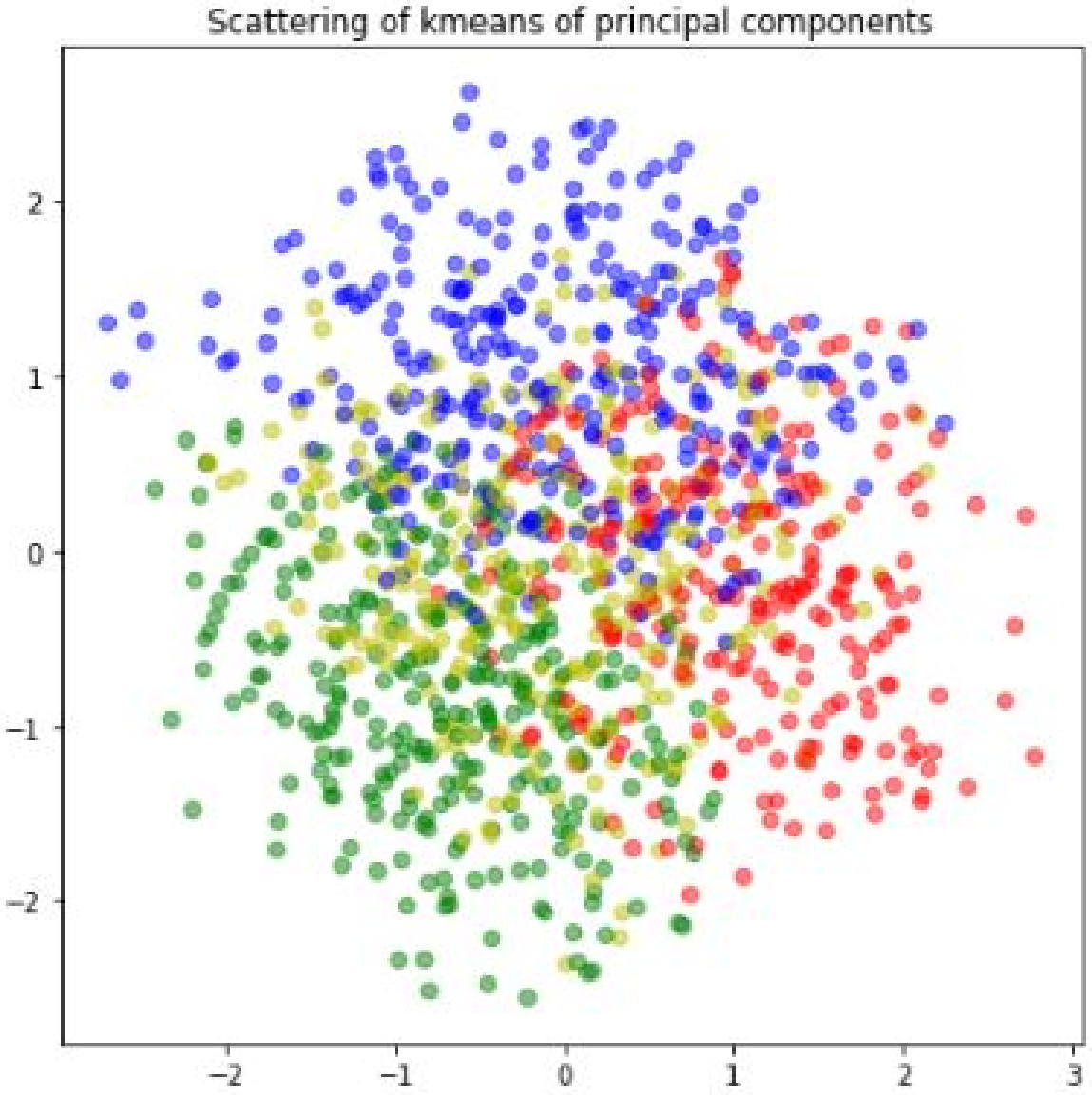
This graph shows the scattering of collection of points of 2 columns taken in pair present in dataframe.



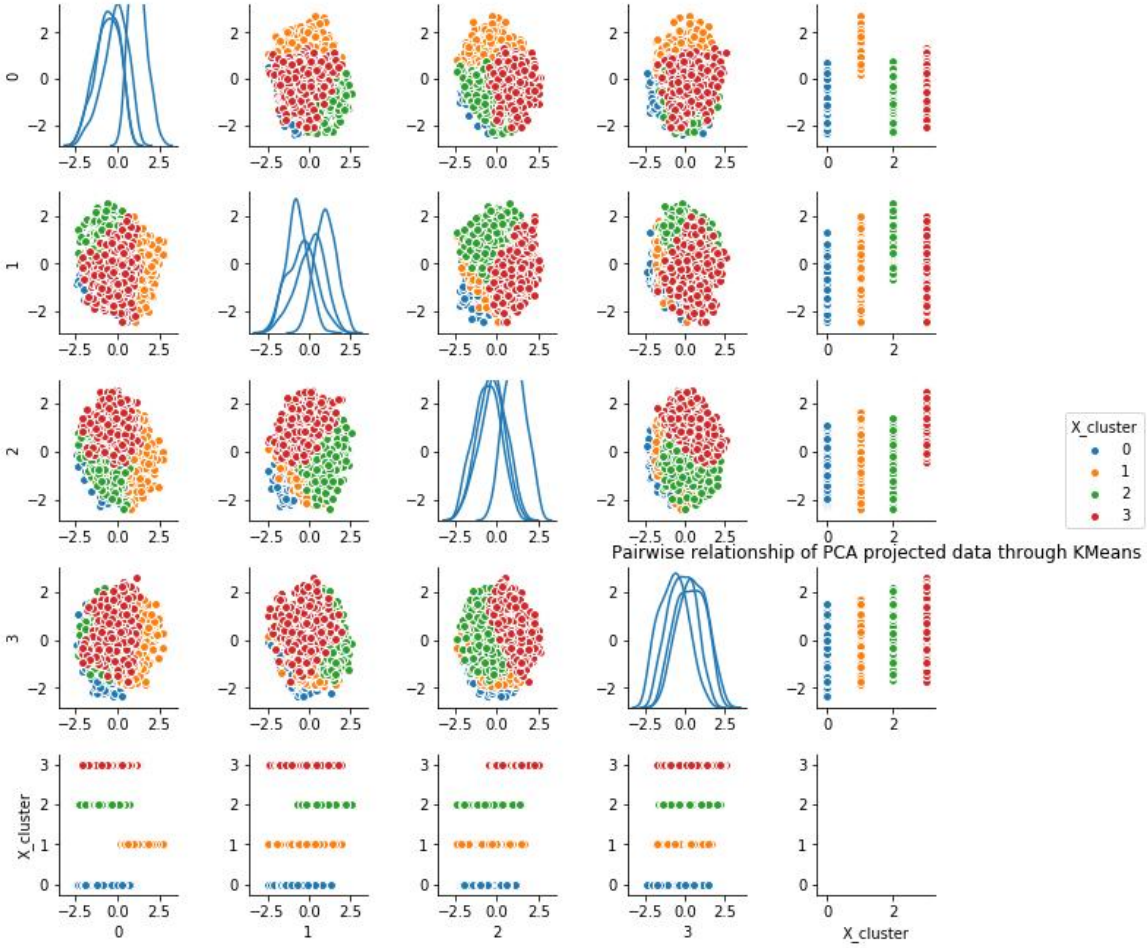
Hexbin plot represent the relationship of 2 numerical variables when you have a lot of data point. Instead of overlapping, the plotting window is split into several hexbins, and the number of points per hexbin is counted. The colour denotes this number of points.



It shows cumulative variance. It is found from individual variance and latter is found from sum of eigenvalues and eigenvalues.



Shows cluster indices and cluster centre of our PCA data. It also shows the normalized value of all the components.



Showing pairwise relationship to visualize our Kmeans clustering on PCA projected data(Seaborn pairplot)

3.3.3 Inference:

The accuracy of a model decreases if we decrease the number of parameters. For example, if we decrease the number of features the accuracy decreases. If we go on increasing the number of features, the accuracy increases but only up to a certain threshold value of parameters. If we increase the number of features above that threshold value , the accuracy decreases as model becomes overly generalized.