Visualization and Analysis of Machine Learning Methods on Yelp Dataset

January 27, 2022

1 Introduction

In this thesis we are going to investigate clustering methods on tezt data collected from Yelp user reviews.

1.1 Yelp

Yelp is a worldwide used website where users are able to view the local businesses and leave reviews about their customer experiences. Various business types are available such as restaurants, bars, cafes, hairdressers, spas, and gas stations. Yelp helps users to find the most popular businesses in their local area. People who need services search the companies on the website, review and give stars. Reviews are a big concern for the business, since they affect the decisions of the customer base.

Yelp website can be found in the following link [1].

1.2 Yelp Dataset

Yelp Dataset contains information about the business, reviews and users. Yelp has invited developers, engineers, statisticians, academics to analyse, visualize and manipulate their datasets. Their database is updated with thousand of new logs every day. The large amounts of data are collected by the company is open to the public on the company website. Yelp Dataset can be found in the following [2].

Data retrieved from the website consists of 5 JSON files containing business, review, user, checkin, tip and photo files in JSON format.

(1) business.json file contains business data including unique business id, name, address, city, state, postal code, location, star and review counts given by the users, attributes, categories and open hours.

(2) review.json file contains text data of reviews with the user_id that wrote the review and the business_id the review is written for. 'Funny, useful and cool' notes given by users for the business reviews made by users and stars value which is from 1 to 5.

(3) user.json file contains activity information about the users. Such as the count of reviews they have made and the notes they have received and the user_ids of their added friends.

(4) checkin.json file contains date and hours details of a business checkin. photo.json file contains photo data including the caption of the photo and its label.

(5) tip.json file contains tips written by a user on a business. Tips are shorter than reviews to convey quick suggestions.

2 Problem Definition

The purpose of the project is to investigate if there exists a quantifiable relationship between the review text and the numerical review by comparing the resulted cluster labels of each data frames. Unsupervised Learning Methods are used for labelization. In order to detect relations, chi-square tests are applied. The hypothesis of the project suggests that user notes and review text are related and user notes are assumed to have sensible labels.

3 Methodology

3.1 Supervised Learning

In this learning method, there is a function from inputs to outputs. In the Supervised Learning, the algorithm is trained with the given input called 'labelled data' and the output called 'labels'. As a result of the training, the algorithm creates a function that should match with the already existing function. When new data points are inserted into this new function, outputs are generated according to the developed model. In other words, this learning technique produces a function that maps between inputs (labelled data) and desired outputs.

Applications, where supervised learning is used, are image and object recognition, predictive analytics, customer sentiment analysis, spam detection. Details can be found in the following article [3].

3.1.1 Classification

Supervised Learning methods where the outputs came from a discrete set are called 'classification'.

3.1.2 Regression

When the outputs come from a continuous set, one can use methods such as linear regression, desicion tree regression or neural network methods.

3.1.3 Cross-validation

When the model has low quality, Underfitting or Overfitting may occur after new data is given to the model. Overfitting occurs when the model has learned the given training data, such that it performs poorly with different given inputs. Underfitting occurs when the algorithm could not relate the data meaningfully with the output. Deciding the best fitting model takes a lot of effort since various algorithms and parameters are used for testing. For the generated model to produce meaningful results, the model must be generated on a large dataset and tests should be done on a smaller portion of that large data. Cross-validation is used in order to measure how well the model fits the given data. Cross-validation is explained further in [4].

3.2 Unsupervised Learning

In Unsupervised Learning given data only has input values which is sometimes called 'unlabeled data'. Hence the parameters are uncertain while modelling. Since there are no output or 'labels',

outputs have to be generated according to a clustering of the unlabeled data. Unlike Supervised Learning the function from input to output must be calculated by the system. The system is expected to learn the relationships between the parameters from the examples by itself.

Unsupervised learning is mostly used for clustering problems. Applications of unsupervised learning are computer vision, medical imaging, anomaly detection, recommendation engines. Further details can be found in [5].

3.2.1 KMeans

KMeans is an Unsupervised Learning method of that is used for clustering. Input data is split into k disjoint subsets for a given KMeans parameter (K). The algorithm first randomly selects K number of data points as the center of each cluster. These points are called 'centroids'. The distance of each data point to each centroid is measured by the algorithm. Then, data points are labelled based on the centroid they are closest to. Then new centroids are calculated as the center of each cluster. Algorithm repeats until centroids stabilize.

Further details can be found in [6].

Inertia of KMeans Inertia is the determination of cluster number input for the best clustering model. KMeans algorithm selects the clusters based on an approach that minimizes the distance between points in a cluster while maximizing the distance between the clusters. The distance between points in a cluster is calculated by the 'within-cluster sum of squares (WCSS)' method. WCSS sums the square of distances between each data point and the centroid. A good data model must contain low inertia with a low number of clusters. However, the number of clusters and inertia are inversely proportional.

3.2.2 DBSCAN

DBSCAN is another Unsupervised Learning method used for clustering. It has two parameters: ϵ and Min_{sample}. First, the algorithm randomly selects a starting point. Data points in the neighbourhood that has the starting point in its center around the predetermined radius ϵ that consist of more data points than the predetermined minimum number of points Min_{sample} are selected as 'core points'. All the data points in the neighbourhoods that have the core points in their centers which ϵ as the radius and consists more points than the Min_{sample} value are selected as the new core points. The process is iterated until neighbourhoods around the core points do not have Min_{sample} number of points. Then a new starting point is randomly selected and the same calculations are repeated until all data points are selected. Clusters are formed by all overlapping disks of radius ϵ placed on core points.

There may be a starting point in a neighbourhood that does contain Min_{sample} number of points. Data points in these neighbourhoods are called 'Noise' and labelled as '-1' by default. These usually are deleted. DBSCAN terminates after all data points are labelled according to their clusters.

Further details can be found in [7]

3.3 Statistical Tests

3.3.1 Chi-square

The statistic that measures the mismatch between observed frequencies and expected frequencies is called chi-square statistics. If chi-square equals 0, the observed and expected frequencies have no relation. Chi-square value is greater when the match between the frequencies are larger. The p-value is the probability of obtaining a different chi-square from the current experiment. The p-value must be less than 0.05 for the test to be considered significant. When the p-value is above 0.05, it means that the test did not work properly.

4 Experiments

Experiments will be done using the Pyhon 3 [8] programming language and scikitlearn [9], pandas [10], numpy [11], scipy [12], matplotlib [13] libraries. We import these libraries below.

[1]:
```
import pandas as pd
import numpy as np
from sklearn import preprocessing
from mpl_toolkits import mplot3d
import matplotlib.pyplot as plt
from sklearn.cluster import DBSCAN
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
import scipy.stats as stats
import pandas as pd
import re
import json
from sklearn.feature_extraction.text import CountVectorizer
```

4.1 Subsampling

Review data retrieved from the website is approximately 6.77 GB. A sample of the data is required to process and manipulate the data easily.

The project was done on a computer that has 8 GB ram and python generated a Memory Error and can not read the whole review.json file. Normally, using the sample() function, it is possible to derive the desired rows of samples from the dataset. However, in this project, splitting large sets into pieces that have desired rows was possible using “Big Text Splitter”[14].

4.2 Conversion to Pandas Dataframe

After splitting the review part of the Yelp dataset into 1000 rows, we were able to load the JSON file as a pandas data frame using the read_json() function from the pandas library.

The converted data frame is named ‘df_r’, which will be used in further calculations. .head() method provides to see the first five rows of the data frame. df_r table includes columns review_id, user_id, business_id, stars, useful, funny, cool, text, date.

[2]:
```
review_json = 'yelp_academic_dataset_review_split.json'
df_r = pd.read_json(review_json, lines=True)
df_r.head()
```

[2]:
```
                review_id                 user_id             business_id  \
0  lWC-xP3rd6obsecCYsGZRg  ak0TdVmGKo4pwqdJSTLwWw  buF9druCkbuXLX526sGELQ
1  8bFej1QE5LXp4O05qjGqXA  YoVfDbnISlW0f7abNQACIg  RA4V8pr014UyUbDvI-LW2A
2  NDhkzczKjLshODbqDoNLSg  eC5evKn1TWDyHCyQAwguUw  _sS2LBIGNT5NQb6PD1Vtjw
3  T5fAqjjFooT4V0OeZyuk1w  SFQ1jcnGguO0LYWnbbftAA  0AzLzHfOJgL7ROwhdww2ew
4  sjm_uUcQVxab_EeLCqsYLg  0kA0PAJ8QFMeveQWHFqz2A  8zehGz9jnxPqXtOc7KaJxA

   stars  useful  funny  cool  \
0      4       3      1     1
1      4       1      0     0
2      5       0      0     0
3      2       1      1     1
4      4       0      0     0

                                                text                 date
0  Apparently Prides Osteria had a rough summer a… 2014-10-11 03:34:02
1  This store is pretty good. Not as great as Wal… 2015-07-03 20:38:25
2  I called WVM on the recommendation of a couple… 2013-05-28 20:38:06
3  I've stayed at many Marriott and Renaissance M… 2010-01-08 02:29:15
4  The food is always great here. The service fro… 2011-07-28 18:05:01
```

4.3 Feature Engineering

The notes 'funny', 'useful', 'cool' of the users will be filtered from the data frame and vectorized according to the business_id they were given for the next step of this project. To calculate the sum of every note that is given to each business, the group_by() function of pandas is used with the sum() function and loaded into the df_bfuc data frame. The table contains for columns: business_id string and funny, useful, cool integer values. In further calculations, each business_id row is considered as a three-dimensional vector.

[3]:
```
df_bfuc = df_r.groupby(['business_id'])[['funny' , 'useful' , 'cool']].sum()
df_bfuc.head()
```

[3]:
```
                        funny  useful  cool
business_id
-1ShItlulHnBsoOQWnblzw      0       1     0
-CmFN8rIN5T4Ic-Fvb8NlA      0       0     0
-IIvmjoEKa9Trhf4OxzJeA      0       0     0
-L69Ix0-xX4BlHA61fGvrQ      1       3     2
-Lk-Bf5heuKTl4n7bvptew      1       0     0
```

For better handling of the data, it is necessary to better understand the general view of its components. The sum of each 'funny', 'useful', 'cool' count gives the general picture of the df_bfuc dataframe using the .sum() function according to the business_id column. Sum of the df_bfuc

frame is loaded to the fucsum variable which can be seen as a pandas Series object using the type(fucsum) function.

[4]:
```
fucsum = df_bfuc.sum(axis=0)
type(fucsum)
```

[4]:
```
pandas.core.series.Series
```

The fucsum series contained two parts, first is the 'funny, useful, cool' attributes which are indexed as 'Notes'. The other part is the sum values that will be indexed as 'Count'. After naming the columns, fucsum is converted to a data frame.

'Useful' note is the most used one in the sample.

[5]:
```
fucsum.index.name = 'Notes'
fucsum1 = fucsum.reset_index(name='Count')
df_fucsum = pd.DataFrame(fucsum1)
df_fucsum.head()
```

[5]:

	Notes	Count
0	funny	357
1	useful	1033
2	cool	383

4.3.1 Normalization

As mentioned before, each business_id row of the df_bfuc data frame is an integer vector. However, the scope of values as we can see above are far from each other, which makes it harder to apply clustering operations and 3D visualizations.

Therefore, df_bfuc values should be compressed into a unit measure without disrupting the dimensions and distances. This method is called "Normalization" and can be applied using preprocessing package of the sklearn library. In order to prepare the df_bfuc data frame for the normalization, it is converted to a NumPy array.

The normalization method used in the following is L1 Norm which is the sum of the magnitudes of the vectors.

After applying the calculations below, an array called 'X_normalized' is obtained.

[6]:
```
x = df_bfuc.values

X_normalized = preprocessing.normalize(x, norm='l1')
X_normalized
```

[6]:
```
array([[0.  , 1.  , 0.  ],
       [0.  , 0.  , 0.  ],
       [0.  , 0.  , 0.  ],
       …,
       [0.  , 1.  , 0.  ],
       [0.  , 1.  , 0.  ],
```

```
       [0.25, 0.75, 0.  ]])
```

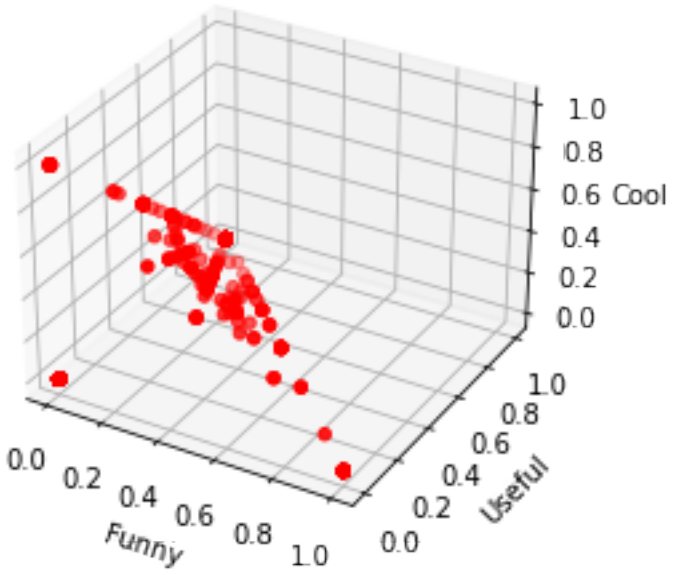
For visualizing the X_normalized vectors in the unit cube, 'plot3d' function is defined using the mplot3d package from the mpl_toolkits and matplotlib.pyplot package.

The plot3d function is called for all of the 3d visualizations in the project.

[7]:
```
def plot3d(data,color):

    fig = plt.figure()
    ax = plt.axes(projection ='3d')


    xs=data[:,0]
    ys=data[:,1]
    zs=data[:,2]

    ax.scatter(xs,ys,zs, c =color)
    ax.set_xlabel('Funny')
    ax.set_ylabel('Useful')
    ax.set_zlabel('Cool')
```

X_normalized vectors are plotted with the color red. The axis names are 'Funny', 'Useful' and 'Cool'. It is necessary to point out that scattered points in the plot do not represent a single vector. Since there are many vectors close to each other, points overlap and resemble a single scatter point.

[8]:
```
plot3d(X_normalized,'red')
```



In order to recognize patterns and similarities in the data two unsupervised learning methods will be applied. Then we will compare the labels obtained from two different methods.

4.4 DBSCAN

The first method that will be used to label the vectors is called Density-Based Spatial Clustering of Applications with Noise (DBSCAN), defined as an unsupervised learning method that can be used in vector arrays.

To cluster the X_normalized data, StandardScaler package from sklearn.preprocessing module and DBSCAN package from sklearn.cluster module must be imported. To calculate different variations of the DBSCAN method with different parameters, the dbscan() function is defined below. DBSCAN algorithm is applied to the vector array following the steps in [15] from the official scikitlearn website.

Returned db_labels array holds the label of each vector.

[9]:
```
def dbscan(array, epsilon, minsamples):
    db_cluster = StandardScaler().fit_transform(array)

    db= DBSCAN(eps = epsilon, min_samples = minsamples).fit(db_cluster)

    core_samples_mask = np.zeros_like(db.labels_, dtype=bool)
    core_samples_mask[db.core_sample_indices_] = True

    db_labels= db.labels_

    return db_labels
```

DBSCAN method is applied below to the X_normalized array with 0.25 epsilon and 100 min_samples values.

[10]:
```
db_labels1 = dbscan(X_normalized,0.25,100)
```

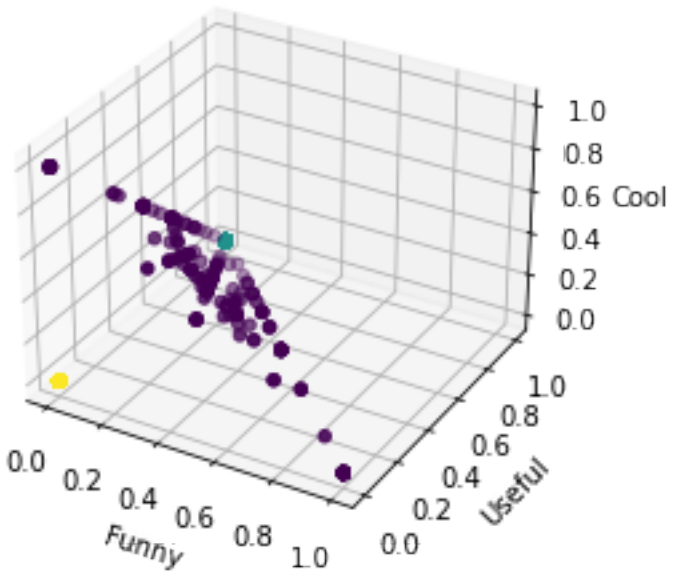
The results of the DBSCAN method is examined below using len, set, list, count functions.

[11]:
```
labelNum = len(set(db_labels1))
clusterNum = len(set(db_labels1)) - (1 if -1 in db_labels1 else 0)
noiseNum = list(db_labels1).count(-1)

print("Number of label types: %d" % labelNum)
print("Estimated number of clusters: %d" % clusterNum)
print("Estimated number of noise points: %d" % noiseNum)
print("Labels: %s" % set(db_labels1))
```

```
Number of label types: 3
Estimated number of clusters: 2
Estimated number of noise points: 248
Labels: {0, 1, -1}
```

Results of DBSCAN labels are plotted below. It is easily observed that one of the clusters is dominant to others.

[12]:
```
plot3d(X_normalized,db_labels1)
```



The count of each DBSCAN label is calculated and observed with pandas crosstab function.

[13]:
```
ct = pd.crosstab(db_labels1,'Count')
ct
```

[13]:
```
col_0  Count
row_0
-1       248
 0       186
 1       381
```

4.5 KMeans

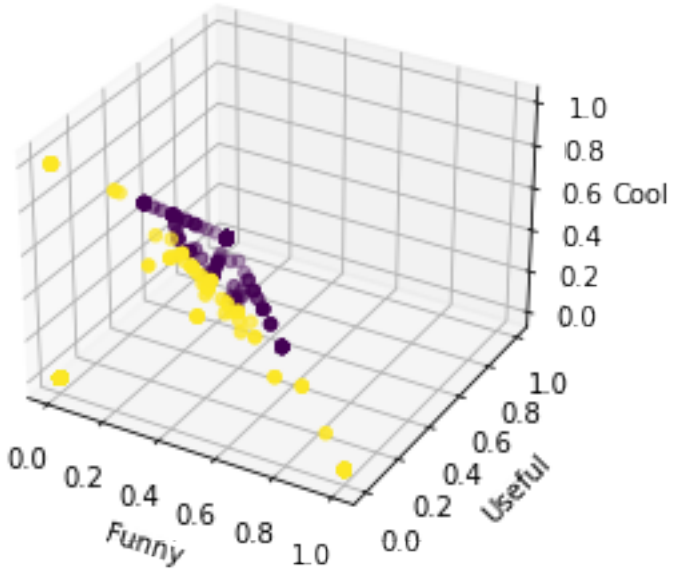
The other unsupervised clustering method that will be applied to the normalized 'funny', 'useful', 'cool' vector array is KMeans. The KMeans method will be used several times to analyze different cluster count results. Therefore, the kmeans() function is defined below using the KMeans package from sklearn.cluster module.

[14]:
```
def kmeans(normData,clusterCount):
    model = KMeans(n_clusters=clusterCount)
    model.fit(normData)
```

```
    return model.predict(normData)
```

KMeans is applied to the X_normalized vector array with 2 clusters. Labels are loaded into the kmeans2_labels array. To better understand the label distribution, vector are plotted according to the kmeans2_labels.

[15]:
```
kmeans2_labels = kmeans(X_normalized,2)
plot3d(X_normalized,kmeans2_labels)
```



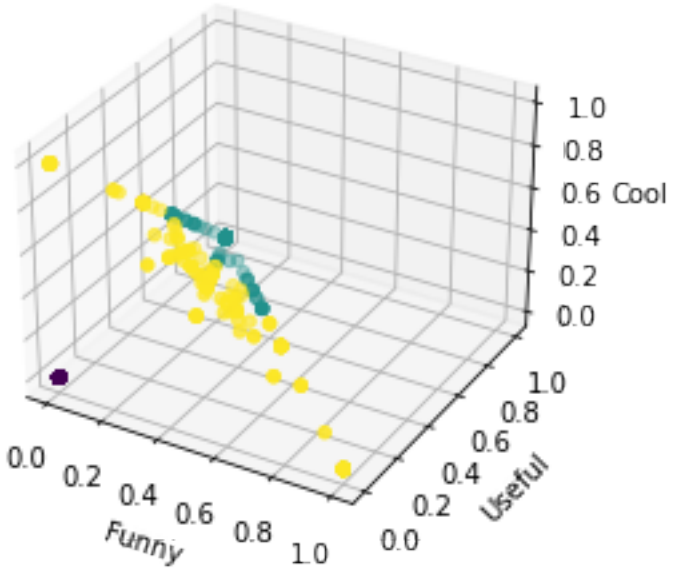
The count of each label within the kmeans2_labels array is calculated below using the crosstab function.

[16]:
```
ct = pd.crosstab(kmeans2_labels,'Count')
ct
```

[16]:
```
col_0  Count
row_0
0        324
1        491
```

In order to view the different variations of the KMeans method cluster numbers with 3 is plotted below.

[17]:
```
kmeans3_labels = kmeans(X_normalized,3)
plot3d(X_normalized, kmeans3_labels)
```



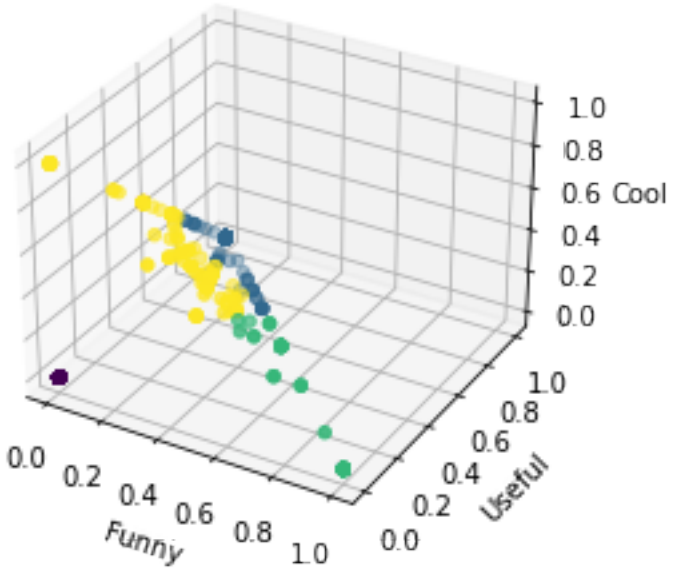
[18]:
```
ct1 = pd.crosstab(kmeans3_labels,'Count')
ct1
```

[18]:
```
col_0  Count
row_0
0        381
1        237
2        197
```

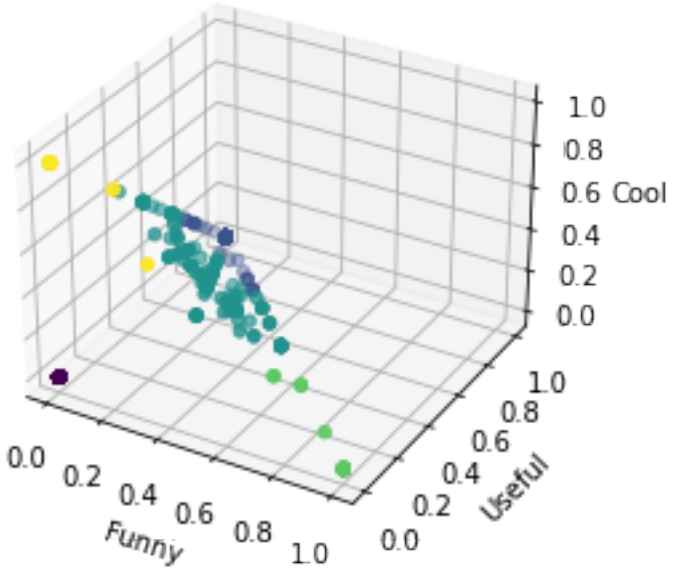
The same calculations for plotting and counting labels when KMeans is applied to the vectorized array with 4 and 5 clusters.

Results can be examined below.

[19]:
```
kmeans4_labels = kmeans(X_normalized,4)
plot3d(X_normalized,kmeans4_labels)
```



[20]:
```
kmeans5_labels = kmeans(X_normalized,5)
plot3d(X_normalized,kmeans5_labels)
```



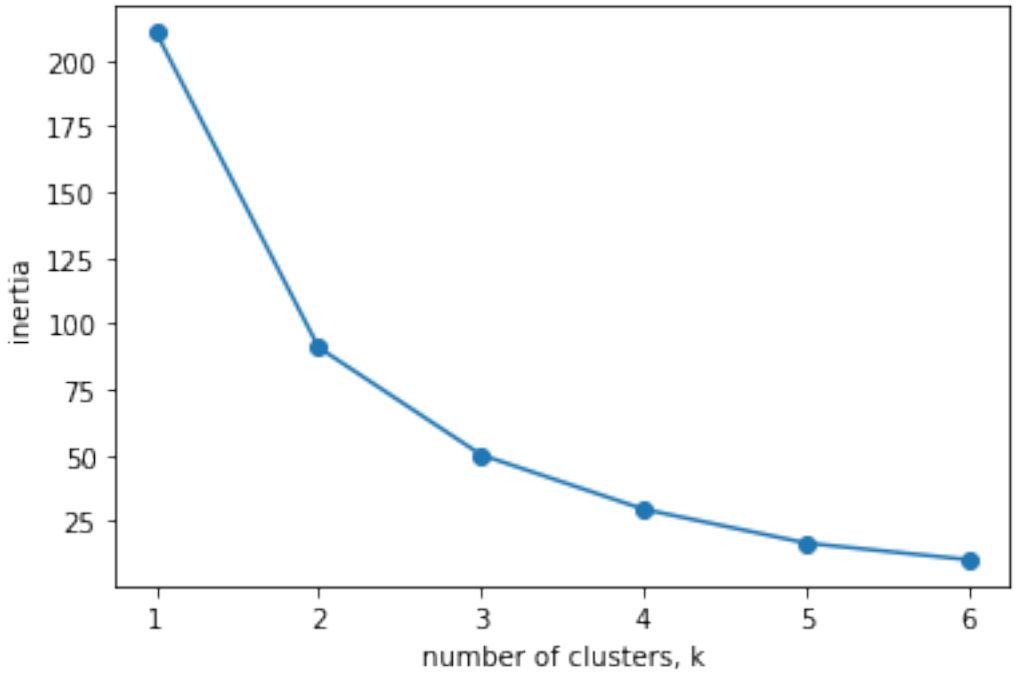
4.5.1 KMeans - Inertia

In order to view inertia values for each cluster within the designated range, a for loop has been created and the calculated inertia result is appended to the inertias array.

[21]:
```
ks = range(1, 7)
inertias = []

for k in ks:

    model = KMeans(n_clusters=k)

    model.fit(X_normalized)

    inertias.append(model.inertia_)
```

To find the optimal K for a dataset, use the Elbow method; find the point where the decrease in inertia begins to slow. ks and inertias values are plotted below to find the elbow point.

[22]:
```
plt.plot(ks, inertias, '-o')
plt.xlabel('number of clusters, k')
plt.ylabel('inertia')
plt.xticks(ks)
plt.show()
```



Although the elbow point is open to interpretation, it is most likely reached with 3 clusters. However, in further calculation considering the quality of clusters, both kmeans2_labels and kmeans3_labels will be used.

4.6 KMeans and DBSCAN Comparison

The quality of the clustering operations can be better observed by comparing the kmeans2_labels to dbscan_labels. Cross Tabulation of the data is used to compare the obtained label arrays. crosstab() function of the pandas library takes two array parameters and outputs a table containing index and column variables respectively.

[23]:
```
ct_3x3 = pd.crosstab(kmeans3_labels, db_labels1)
ct_3x3.index.name = 'kmeans3'
ct_3x3
```

[23]:
```
col_0     -1    0    1
kmeans3
0          0    0  381
1         51  186    0
2        197    0    0
```

The above result shows that KMeans labels 381 vectors as “0” and DBSCAN labels the same vectors as “1”. KMeans labels 197 vectors as “2” and DBSCAN labels the same vectors as “-1”. However, 237 vectors labelled as “1” from the KMeans method is separated to two parts with “-1” and “0” labels according to the DBSCAN method which consist of a majority of “0” labels. This differentiation lowers the quality of the labelling method results.

[24]:
```
ct_2x3 = pd.crosstab(kmeans2_labels, db_labels1)
ct_2x3.index.name = 'kmeans2'
ct_2x3
```

[24]:
```
col_0     -1    0    1
kmeans2
0        138  186    0
1        110    0  381
```

To compare DBSCAN and KMeans with 2 label types, DBSCAN should be reapplied with different parameters. After recalculating the DBSCAN method on the X_normalized array with 0.25 epsilon value and 200 minimum samples, we have obtained two label sets named “0” and “-1”. Labels are loaded into the db_label2 array.

[25]:
```
db_labels2 = dbscan(X_normalized, 0.25, 200)
print("Labels: %s" % set(db_labels2))
```

```
Labels: {0, -1}
```

Cross tabulation is used below to compare the kmeans2_labels and the db_labels2 arrays.

[26]:
```
ct_2x2 = pd.crosstab(kmeans2_labels, db_labels2)
ct_2x2.index.name = 'kmeans2'
```

```
ct_2x2
```

[26]:
```
col_0      -1    0
kmeans2
0         324    0
1         110  381
```

Cross tabulation of KMeans and DBSCAN methods with 2 data labels shows that one cluster from DBSCAN with 3 data labels remain the same however the others labelled as a whole new cluster.

4.7 Text Vectorization

The df_r data frame has a column called 'text' that involves customer reviews. In order to analyze the review information of each business, reviews must be grouped according to their matching busines_id. Text analysis will be done using every word from the reviews. However, the revies have nonrelated letters and special characters that shoul not be involved in the calculation.

To standardize and clean the 'text' values, calculations below are made using the regex library. Following sources are used [16] [17].

4.7.1 Text Preprocessing

First of all, 'text_edit' column that is the lowered version of the 'text' column is added to the df_r data frame.

[27]:
```
df_r['text_edit'] = df_r['text'].str.lower().values
```

Combinations containing word characters next to digits and single letters are cleaned using the replace function of regex.

[28]:
```
df_nw = df_r['text_edit'].replace(to_replace = '[\W\d]', value = ' ', regex =␣
↪True)
df_up = df_nw.replace(to_replace = '\s\w\s', value = ' ', regex = True)
```

The edited column is replaced with the "text_edit" column in the df_r data frame.

[29]:
```
df_r = df_r.assign(text_edit=df_up)
```

After computing the alterations on the text_edit column, results are grouped by their matching business_id. The goal here is to create a bag full of words for each business from their corresponding reviews. df_r is grouped according to business_id loading the business_id column to df_bid. Then df_bid is joined with the "text_edit" column. The joined columns are loaded into the text_list2 data frame.

[30]:
```
df_bid = df_r.groupby('business_id')
text_list2 = df_bid['text_edit'].agg(lambda column: " ".join(column))
```

To increase the readibility of the text_list2 dataframe, column that contains words is named "reviews". The data frame is written into a JSON file called "review.json".

[31]:
```
text_list3 = text_list2.reset_index(name="reviews")
text_list3.to_json('reviews.json', orient='records', lines=True)
```

4.7.2 Count Vectorizer

Count Vectorizer function is used to convert every row in a data frame to vectors. In order to calculate the vector values, the function finds all of the unique words in the data frame and creates a reference list from the bag of unique words. If a review does contain a word from the reference list, then the array of that row contains zero in the position of the word. If a review contains multiple words, the count of that word is written in the vector. Every row is converted to a vector according to the reference list. Further details can be found in the following scikitlearn documentation [18]

4.7.3 Converting Text to Vectors

CountVectorizer function must be imported from sklearn.feature_extraction module.

[32]:
```
from sklearn.feature_extraction.text import CountVectorizer


vectorizer = CountVectorizer()
X = vectorizer.fit_transform(text_list2)
vectorizer.get_feature_names_out()
```

[32]:
```
array(['__', 'aaaaaand', 'aback', …, 'zucchini', 'zzzzzzzz', 'â¼'],
      dtype=object)
```

After applying count vectorization that converts text_list2 data frame to the X dataframe, X must be converted to an array with the toarray() function of the NumPy library.

[33]:
```
x2 = X.toarray()
```

x2 array must be normalized using the same steps in the Normalization section after importing the preprocessing package from the sklearn module.

[34]:
```
X2_normalized = preprocessing.normalize(x2, norm='l1')
X2_normalized
```

[34]:
```
array([[0., 0., 0., …, 0., 0., 0.],
       [0., 0., 0., …, 0., 0., 0.],
       [0., 0., 0., …, 0., 0., 0.],
       …,
       [0., 0., 0., …, 0., 0., 0.],
       [0., 0., 0., …, 0., 0., 0.],
       [0., 0., 0., …, 0., 0., 0.]])
```

The goal of the project is to investigate similarities between user reviews and given notes. Therefore KMeans clusters of each data frame will be compared in the analysis section, using cross tabulation and chi-square test.

KMeans is applied to the vectorized array with 2 clusters to prepare the text labels for further analysis.

[35]:
```
text2_labels = kmeans(X2_normalized,2)
ct_text2 = pd.crosstab(text2_labels,'Count')
ct_text2
```

[35]:
```
col_0  Count
row_0
0        382
1        433
```

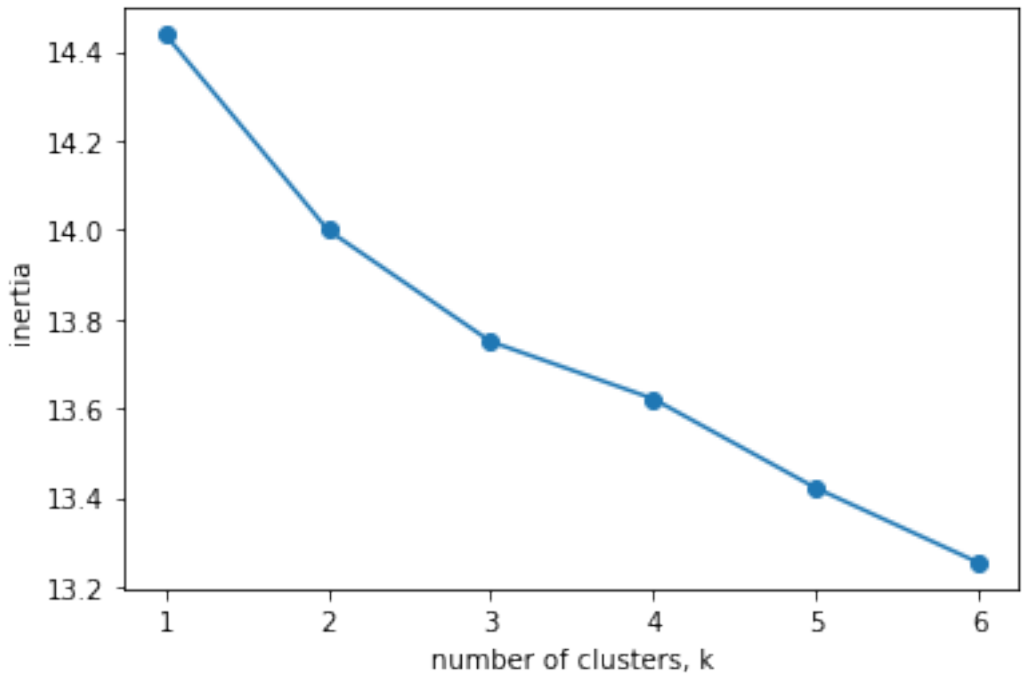
4.7.4 The Inertia of Text Vectorization

Recall from the "KMeans-Inertia" section that the inertia algorithm was used to help find the elbow point of a KMeans Inertia graph to guess the best clustering within the cluster range. Inertias of the text labels ranging from 1 to 7 clusters are calculated below.

[36]:
```
ks = range(1, 7)
inertias = []

for k in ks:

    model = KMeans(n_clusters=k)
    model.fit(X2_normalized)
    inertias.append(model.inertia_)
```

Inertia values of the text labels are plotted. The elbow point is estimated to be 4 clusters.

[37]:
```
plt.plot(ks, inertias, '-o')
plt.xlabel('number of clusters, k')
plt.ylabel('inertia')
plt.xticks(ks)
plt.show()
```



KMeans is applied to the array with 4 clusters below. The cluster on the elbow point of text labels will be compared to the funny, useful, cool valued business vectors and results will be analyzed in the analysis section.

[38]:
```
text4_labels = kmeans(X2_normalized,4)
ct_text4 = pd.crosstab(text4_labels,'Count')
ct_text4
```

[38]:
```
col_0  Count
row_0
0        318
1         14
2        263
3        220
```

5 Analysis

5.1 KMeans - DBSCAN on Notes

Cross-tabulations obtained from the KMeans and DBSCAN Comparison sections are analyzed below. The stats package of the scipy library is used for the statistical tests.

[39]:
```
print(ct_2x2)
print(stats.chisquare(ct_2x2, axis= None))
```

```
col_0     -1    0
kmeans2
0        324    0
1        110  381
Power_divergenceResult(statistic=472.0527607361963,
pvalue=5.431225839119839e-102)
```

The chi-square value of the ct_2x2 table is close to 472 and the p-value of the same table is far below than 0.05. It can be said that there is a strong relation between the labelling of KMeans and DBSCAN methods.

From the inertia graph plotted in the KMeans-Inertia section, the elbow point of the inertia was determined as 3. In order to compare the best cluster of notes labels with KMeans and DBSCAN, chi-square test is going to be used below.

[40]:
```
print(ct_3x3)
print(stats.chisquare(ct_3x3, axis= None))
```

```
col_0     -1    0    1
kmeans3
0          0    0  381
1         51  186    0
2        197    0    0
Power_divergenceResult(statistic=1627.3349693251535, pvalue=0.0)
```

The label distributions are sensible in the above table and the p-value is 0 meaning that there is not any different result obtained from the test. Chisquare value is far greater than 0, which means there is a very strong relation between the KMeans DBSCAN clusters.

5.2 Notes and Text Labels

Labels that are obtained with KMeans and Dbscan method from notes data and text data are compared below using cross tabulation and the stats.chisquare function.

[41]:
```
ct_2x2nt= pd.crosstab(kmeans2_labels, text2_labels)

print(ct_2x2nt)
```

```
col_0    0    1
row_0
0      150  174
1      232  259
```

[42]:
```
print(stats.chisquare(ct_2x2nt, axis= None))
```

```
Power_divergenceResult(statistic=37.42208588957055, pvalue=3.74605056323478e-08)
```

The labels of the vectorization are evenly split. '0' labelled vectors are more in KMeans clustering. Since the chi-square result is nearly 38, it can be understood that there is a weak relationship even though the distributions in the clusters are problematic.

Cross tabulation below has DBSCAN labels its rows and vectorization labels in its columns.

[43]:
```
ct_2x2dt= pd.crosstab(db_labels2, text2_labels)
print(ct_2x2dt)
```

```
col_0    0    1
row_0
-1     197  237
 0     185  196
```

[44]:
```
print(stats.chisquare(ct_2x2dt, axis= None))
```

```
Power_divergenceResult(statistic=7.669938650306749, pvalue=0.05334907419686734)
```

From the ct_2x2dt table, it can be seen that DBSCAN has labelled 53% of the data as "-1" and 43% as "0" while the Count Vectorizer function has labelled 53% of the data as "1" and 43% as "0". Since the label counts in the cross tabulation are almost evenly distributed, it can be said that these clusters are not sensible. According to the chi-square function results, the p-value is calculated close to 0.05, which means that this test can be considered as failed.

From the inertia graph plotted in the Inertia of Text Vectorization section the elbow point of the vectorization was determined as 4. In order to compare the best cluster of text labels with KMeans, kmeans2_labels and text4_labels are cross tabulated below.

[45]:
```
ct_2x4nt= pd.crosstab(kmeans2_labels, text4_labels)
print(ct_2x4nt)
```

```
col_0    0   1    2    3
row_0
0      131   2  100   91
1      187  12  163  129
```

[46]:
```
print(stats.chisquare(ct_2x4nt, axis= None))
```

```
Power_divergenceResult(statistic=301.7509202453988,
pvalue=2.5573984702216795e-61)
```

Text vectorization label distribution is 39%, 2%, 32%, 27% and KMeans label distribution of the notes are 40% and 60% respectively. The result of the chi-square test shows that there may be a strong relation between vectorization the clusters of text vectorization and notes. Since the p-value is far smaller than 0.05, the test is considered reliable.

6 Future Work

6.1 PCA Method

In earlier sections, cluster labels of notes data are plotted using a 3d plot function. This was possible since notes contained 3 attributes; 'funny', 'useful', 'cool'. Each attribute represented a dimension in the unit cube. Text labels are also a multidimensional array. Every word of the review dataset represented a dimension, therefore plotting the text labels was impossible using the defined 'plot3d' function. Principal component analysis (PCA) is a statistical method that summarizes the content in large data tables as a smaller set called "summary indexes" which helps the visualization and analysis to be done easily. PCA method will be used in the review dataset.

6.2 High Performance Computing

Since we had limited computing resources we needed sampling. Executing the Experiment codes on the entire Yelp Dataset requires a computer with a large RAM capacity. Istanbul Technical University has the National Center for High Performance Computing (UHeM) that provides academics and students with the use of supercomputing and data storage services. UHeM services will be used in order to apply the experiment of this project to the entire Yelp Dataset. Further information about the UHeM services can be found in the following [19]

7 References

[1] Yelp. (n.d.). Retrieved January 25, 2022, from https://www.yelp.com/

[2] Yelp open dataset. Yelp Dataset. (n.d.). Retrieved January 25, 2022, from https://www.yelp.com/dataset

[3] IBM Cloud Education. (n.d.). What is supervised learning? IBM. Retrieved January 25, 2022, from https://www.ibm.com/cloud/learn/supervised-learning

[4] 3.1. cross-validation: Evaluating estimator performance. scikit. (n.d.). Retrieved January 25, 2022, from https://scikit-learn.org/stable/modules/cross_validation.html

[5] IBM Cloud Education. (n.d.). What is unsupervised learning? IBM. Retrieved January 25, 2022, from https://www.ibm.com/cloud/learn/unsupervised-learning

[6] Sklearn.cluster.kmeans. scikit. (n.d.). Retrieved January 25, 2022, from https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html?highlight=kmeans#sklearn.cluster.KMeans

[7] Sklearn.cluster.DBSCAN. scikit. (n.d.). Retrieved January 25, 2022, from https://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html

[8] Van Rossum, G., & Drake, F. L. (2009). Python 3 Reference Manual. Scotts Valley, CA: CreateSpace.

[9] Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.

[10] McKinney, W., & others. (2010). Data structures for statistical computing in python. In Proceedings of the 9th Python in Science Conference (Vol. 445, pp. 51–56).

[11] Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., … Oliphant, T. E. (2020). Array programming with NumPy. Nature, 585, 357–362. https://doi.org/10.1038/s41586-020-2649-2

[12] Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., … SciPy 1.0 Contributors. (2020). SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. Nature Methods, 17, 261–272. https://doi.org/10.1038/s41592-019-0686-2

[13] Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. Computing in Science & Engineering, 9(3), 90–95.

[14] Bigtextfilesplitter. (n.d.). Retrieved January 25, 2022, from https://www.withdata.com/big-text-file-splitter/

[15] Demo of DBSCAN clustering algorithm. scikit. (n.d.). Retrieved January 25, 2022, from https://scikit-learn.org/stable/auto_examples/cluster/plot_dbscan.html

[16] Francisco, M. – S., Maureen, Zealand, P. – N., Pythia, Mark, Amrut, P., Italy, A. M. –, Maiorana, A., Bulgaria, michael –, Michael, California, Y. –, Yuri, Europe, T. –, Tom, Najam, Shannon, P. –, & Philip. (n.d.). Quick-start. Regex Cheat Sheet. Retrieved January 25, 2022, from https://www.rexegg.com/regex-quickstart.html

[17] Replace values in pandas dataframe using regex. GeeksforGeeks. (2020, December 29). Retrieved January 25, 2022, from https://www.geeksforgeeks.org/replace-values-in-pandas-dataframe-using-regex/

[18] Sklearn.feature_extraction.text.CountVectorizer. scikit. (n.d.). Retrieved January 25, 2022, from https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html

[19] National HPC Center. UHeM. (n.d.). Retrieved January 25, 2022, from https://en.uhem.itu.edu.tr/