**CS 7800 Information Retrieval**

**Text Mining**

**Abhishek Singh Thakur Nikhil Chavan**

**(U00933835) (U00969241)**

**Use case:** This lab firstly tends to extract features from ‘mini\_newsgroups’ directory, which contains 20 folders, every folder contains documents. Overall, there are 2000 documents upon which features are extracted and TF, IDF, and TFIDF values are stored in ‘libsvm’ format. Secondly, Classifiers such as Multinomial Naive Bayes, Bernoulli, KNeighbors, SVC are used to generate respective models and classifiers evaluation is calculated using cross-validation techniques. As a part of the third section, Feature Selection method SelectKBest which takes metrics Chi-Square and Mutual Information and are used to reduce the number of features and accordingly the with obtained feature data set with less features 4 classifiers models are generated and accordingly the performances are calculated using cross-validation techniques and graphs are generated for analysis and evaluation of how Feature Selection techniques works. As a final part, the Documents are clustered using KMeans Clustering and Agglomerative/Hierarchical Clustering Methods and respective quality checks are done using Silhouette Coefficient and Normalized Mutual Information metric scores. The graphs are plotted for 2 metric scores against 2 clustering algorithms for cluster size ranging from 2 to 25 for evaluation.

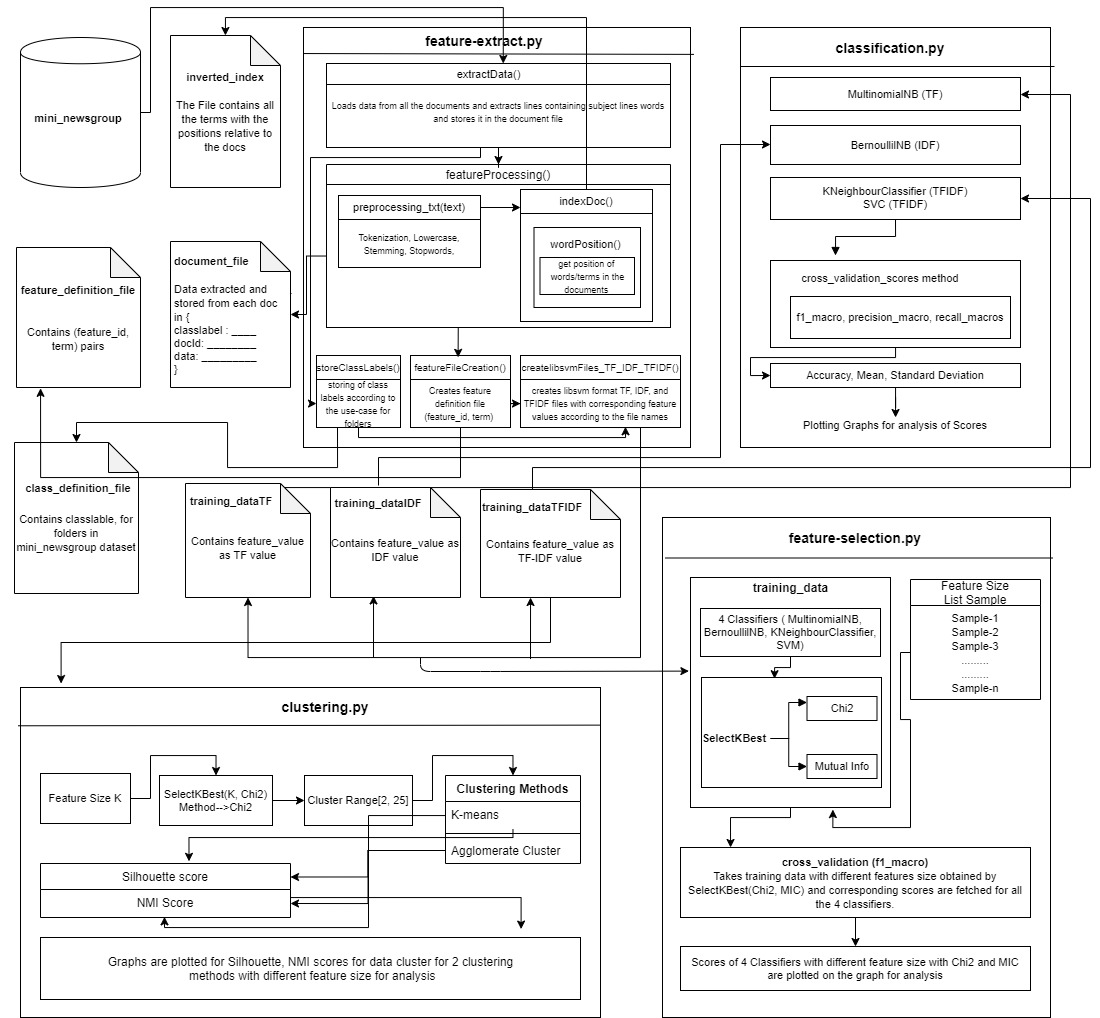
**Setup:** Downloading the mini\_newsgroups zipped file, installing python version which supports NLTK library and ‘sklearn’ since the classifiers, classifier evaluation techniques, feature selection techniques, document clustering methods, and clustering evaluation algorithms are imported from sklearn library. Any IDE which is of personal choice I’ve used Visual Studio Code for writing and compiling python code.

**Design Implementation:**

The text mining project is divided into 4 parts for this lab. The below screenshot shows the implementation of the code to achieve the below features which are parts of the lab.

***Feature Extraction:*** The code is implemented in feature-extraction.py where it takes 4 arguments which are the filename of the dataset and the filenames which stores classlabels, features and its featured and training data for TF, IDF, and TFIDF each. The extractData() method extracts the data from all the 2000 documents contained in respective in 20 folders of mini\_newsgroup dataset. In featureProcessing() for each document it considers lines which have Subject and Lines as the words in the line after preprocessing the data i.e. after performing tokenization, stemming, lowercase and correction of data. The entire data is extracted and stored in document\_file for easy retrieval containing classlabel, documentName and extracted Data. The classlabels are for the 20 folders are divided into 6 groups which are stored in the class\_defination\_file and it is done in storeClassLabels(). In indexDoc()Using data in document\_file an Inverted matrix is generated and stored in inverted\_index file for easy calculation of TF, IDF and TFIDF values. The inverted index data is used to fetch all the terms and each term is allocated with the featured, and the featureId, term pair is stored in feature\_defination\_file which is implemented in featureFileCreation() method. Finally, in createlibsvmFiles\_TF\_IDF\_TFIDF() method training data is generated in libsvm format i.e. < class label> <feature-id>:<feature-value> <feature-id>:<feature-value> ... in each of its line. However training\_data\_file.TF, training\_data\_file.IDF and training\_data\_file.TFIDFfiles contain <feature-value>  as the TF, IDF, and TFIDF respectively, values of the corresponding term in the document with its <feature-id> extracting from the feature\_defination\_file, the feature id and value pair is being sorted by its feature id before storing the information in the files.

***Classifiers:*** In classification.py the 4 classifiers used in this lab are Multinomial Naive Bayes classifier, Bernoulli classifier, KNeighbors classifier, Support Vector-machine Classifiers. The dataset upon which these classifiers are tested is considered from training\_data\_file.TF, training\_data\_file.IDF and training\_data\_file.TFIDFfiles. TF data is used for Multinomial Naive Bayes classifier, IDF data is used for Bernoulli classifier and TFIDF data is used for KNeighbors and SVC classifiers. Using cross\_validation\_scores() method which calculates Accuracy of the classifiers upon the respective training data scores are generated for each classifier with 3 metrics i.e. f1\_macro, precision\_macro, recall\_macros. The scores contain Mean and Standard Deviation values which are plotted for the analysis. The detailed explanation and analysis are discussed in the below sections.



***Feature Selection:*** In feature-selection.py the training data with reduced features are generated using SelectKBest algorithms which selects K best features and it is done using Chi2 and MI values. For some K sized feature data 2 training data are produced using Chi2 and MI metric with SelectKBest. The classifier models are generated for the 4 classifiers using the 2 generated K sized feature set data. Accordingly, in getScores() method cross-validation scores are generated for 4 classifiers, and aginst each metric i.e Chi2 and MI respective scores are plotted on graphs for 4 classifier score results and are analyzed to check the performance.

***Clustering:*** The clustering of the documents is performed with TFIDF files in clustering.py and it is done using KMeans and Agglomerative Clustering Methods. For K sized feature set extracted using SelectKBest using Chi2, the clustering methods are used upon with the range of 2 to 25. For each range of values, Silhouette and NMI scores are plotted against the KMeans and Agglomerative Clustering methods to check the performance of the clustering methods and its behaviour to validated which is optimal and why.

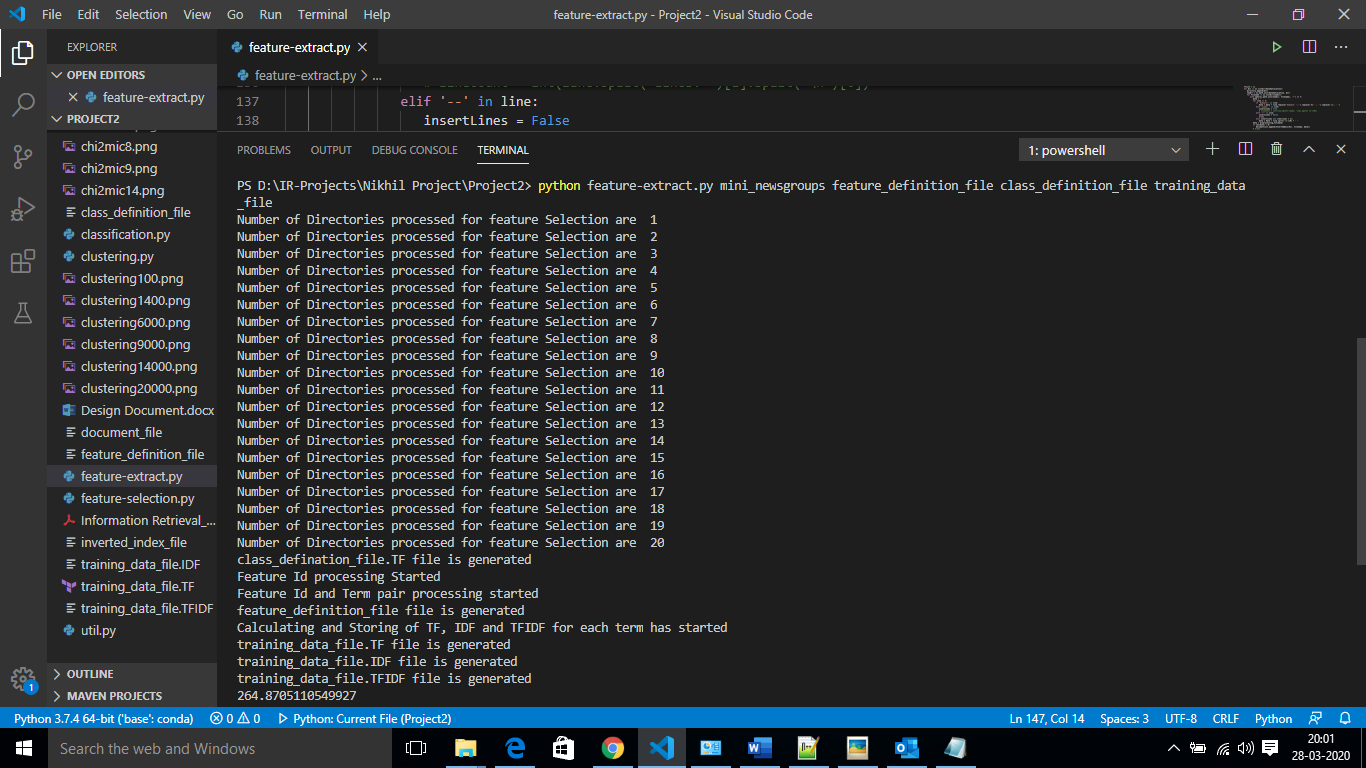
In detailed discussion on analysis for the graphs plotted in each section is done in below respective sections.

**Feature Extraction:**

python feature-extract.py mini\_newsgroups feature\_definition\_file class\_definition\_file training\_data\_file

The below arguments are passed as arguments to extract the features for the data inside it. Before processing each document, the data is firstly extracted by using the data in lines which have ‘Subject: ’ and ‘Lines: ’ in the document and later the data has been preprocessed i.e. applying lowercase, stemming, tokenizing and correcting the data.

* **mini\_newsgroups:** It contains 20 directories in it and each directory has several documents in it that are processed to extract the features. Overall, there are 2000 documents in the **mini\_newsgroups** folder to process.
* **feature\_definition\_file:** This file contains feature Id and Term pair in each line which are results after processing all the 2000 documents extracting its data applying lowercase, stemming, tokenizing and correcting the data. The feature Id is sorted and the terms are being a word in the documents which are considered as features.
* **class\_definition\_file:** This file contains the classlabel and directory pair which are in the mini\_newsgroups dataset folder. The documents processed are captured with their respective folder to which it belongs by its classlabel rather than its folder name. There are 6 groups divided as (comp.graphics, comp.os.ms-windows.misc, comp.sys.ibm.pc.hardware, comp.sys.mac.hardware, comp.windows.x), (rec.autos, rec.motorcycles, rec.sport.baseball, rec.sport.hockey), (sci.crypt, sci.electronics, sci.med, sci.space), (misc.forsale), (talk.politics.misc, talk.politics.guns, talk.politics.mideast), (talk.religion.misc, alt.atheism, soc.religion.christian) and are labelled from 0 to 5.
* **training\_data\_file:** There are 3 different files to be generated by concatenating **training\_data\_file to TF, IDF and, TFIDF.** Each of the 3 files contains the data in the libsvm format i.e. < class label> <feature-id>:<feature-value> <feature-id>:<feature-value> ... in each of its line. However **training\_data\_file.TF, training\_data\_file.IDF** and **training\_data\_file.TFIDF** files contain <feature-value>  as the TF, IDF and TFIDF respectively, values of the corresponding term in the document with its <feature-id> extracting from the feature\_defination\_file, the feature id and value pair is being sorted by its feature id before storing the information in the files. The data in libsvm format are later used as Training and Testing data and are applied on Classifiers and with the results obtained the classifiers are evaluated using cross-validation techniques. Finally, feature selection techniques are applied to extract the potential features to calculate the classifier evaluation and the results are analyzed. The TFIDF value is being used in Document Clustering in the below sections to find and analyze the results of clustering algorithms on this dataset.



The above screenshot generates 5 files according to the above description. There are 21245 features extracted with the data extracted from the documents. Besides, the TF, IDF and TFIDF files contain 1998 out of 2000 documents which shows that 2 documents are either missing Subject or Lines or both of the lines in it which are excluded from the files.

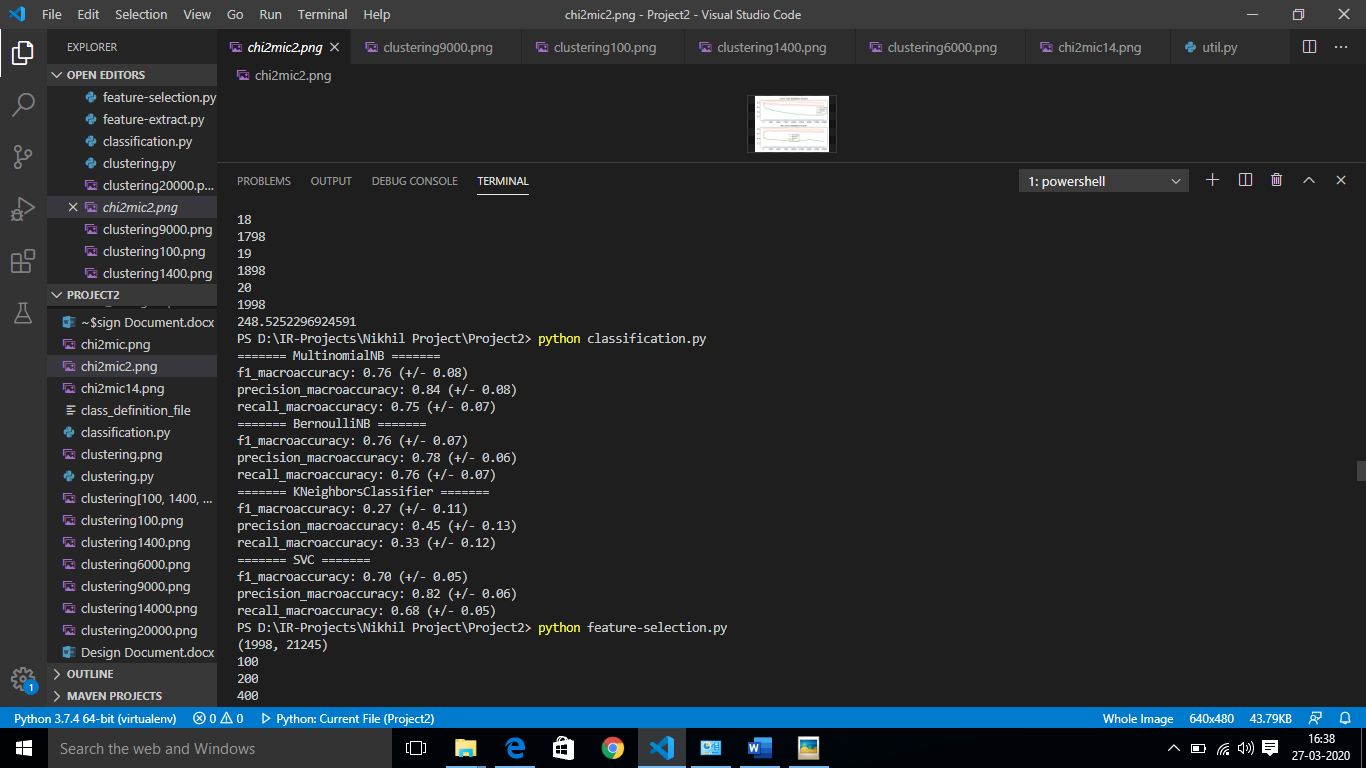
**Classifier Evaluation:** python classification.py

The classifiers used in this lab are Multinomial Naive Bayes classifier, Bernoulli classifier, KNeighbors classifier, Support Vector-machine Classifiers. The dataset upon which these classifiers are tested is considered from **training\_data\_file.TF, training\_data\_file.IDF** and **training\_data\_file.TFIDF** files. TF data is used for Multinomial Naive Bayes classifier, IDF data is used for Bernoulli classifier and TFIDF data is used for KNeighbors and SVC classifiers since these classifiers are modeled to work based upon Supervised and Unsupervised data and accordingly are data is considered.

The data in libsvm format in each of the files are firstly loaded and is divided into training and testing data. As discussed above classifier models are generated with the data provided accordingly. After the models are generated we evaluate the classifier models using cross validation techniques accordingly scores are generated by scores = cross\_val\_score(model, train\_data, test\_data, cv=5, scoring=crossValidation)

* model: It takes the Multinomial Naive Bayes, Bernoulli, KNeighbors, SVC Classifiers Models generated.
* train\_data: It takes training data extracted from TF, IDF and TFIDF files (libsvm format decoded)
* test\_data: It takes training data after splitting the data loaded from the files and split accordingly.
* cv: The 5-fold cross-validation (cv=5) uses [KFold](https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.KFold.html#sklearn.model_selection.KFold) or [StratifiedKFold](https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.StratifiedKFold.html#sklearn.model_selection.StratifiedKFold) strategies by default.
* scoring: It takes f1\_macro, precision\_macro, and recall\_macro as metric upon which for each classifier means and 2\*std of 5-fold scores are calculated respectively. Usually, F1 scores range > 0.5

The below screenshot is the results of mean and standard deviation obtained for all the 4 classifier models for different metrics i.e. for scoring methods taking f1\_macro, precision\_macro, and recall\_macro.



**Result Analysis:**

The results on our multiple feature dataSet the F1 score can be considered as an evaluation parameter since it is the harmonic mean of precision and recall. As we see in the above screenshot that Multinomial Naïve Bayes classifier has higher precision for the TF data provided. This shows that Multinomial NB is a nominal classifier for the dataset provided. Considering the precision to check the performance SVC, and Bernoulli NB comes after the Multinomial classifier model. Lastly, KNeighbors Classifier has the least F1, Precision and Recall scores i.e. the results after considering testing data on the model only predicts data correctly with 45% accuracy which is less than 50%. Considering the fact that the nominal model should show F1 scores range >0.5 then KNeighbors Classifier is not a good classifier for the input data provided. Concluding that Bernoulli and Multinomial classifier have high precision and recall which shows that for a given input data, these classifiers are nominal and predicts the results with high accuracy. However, if we want to consider high precision and low recall then KNN can be considered. Each classifier model can be tuned by adjusting the default parameters to achieve better F1 scores with better precision and recall values.

**Feature Selection:** python feature\_selection.py

This lab uses SelectKBest feature selection technique for dimensionality feature reduction, the technique uses 2 methods for this lab which are chi2 and mutual\_info\_classif. As we got 21245 features in our dataset. This feature selection technique SelectKBest takes k as an integer which produces the best of k features out of 21245 features for our dataset, and the ranking of the features is given based upon the Chi2 and Mutual Information scores. In this section, we apply SelectKBest algorithm for different k values and the dataset resulted are applied with the 4 classifiers that are Multinomial Naive Bayes, Bernoulli, KNeighbors, SVC Classifiers and models are generated with a resulted dataset. We apply cross validation for each k sized feature dataset upon all the 4 classifiers and for 2 methods in SelectKBest technique.

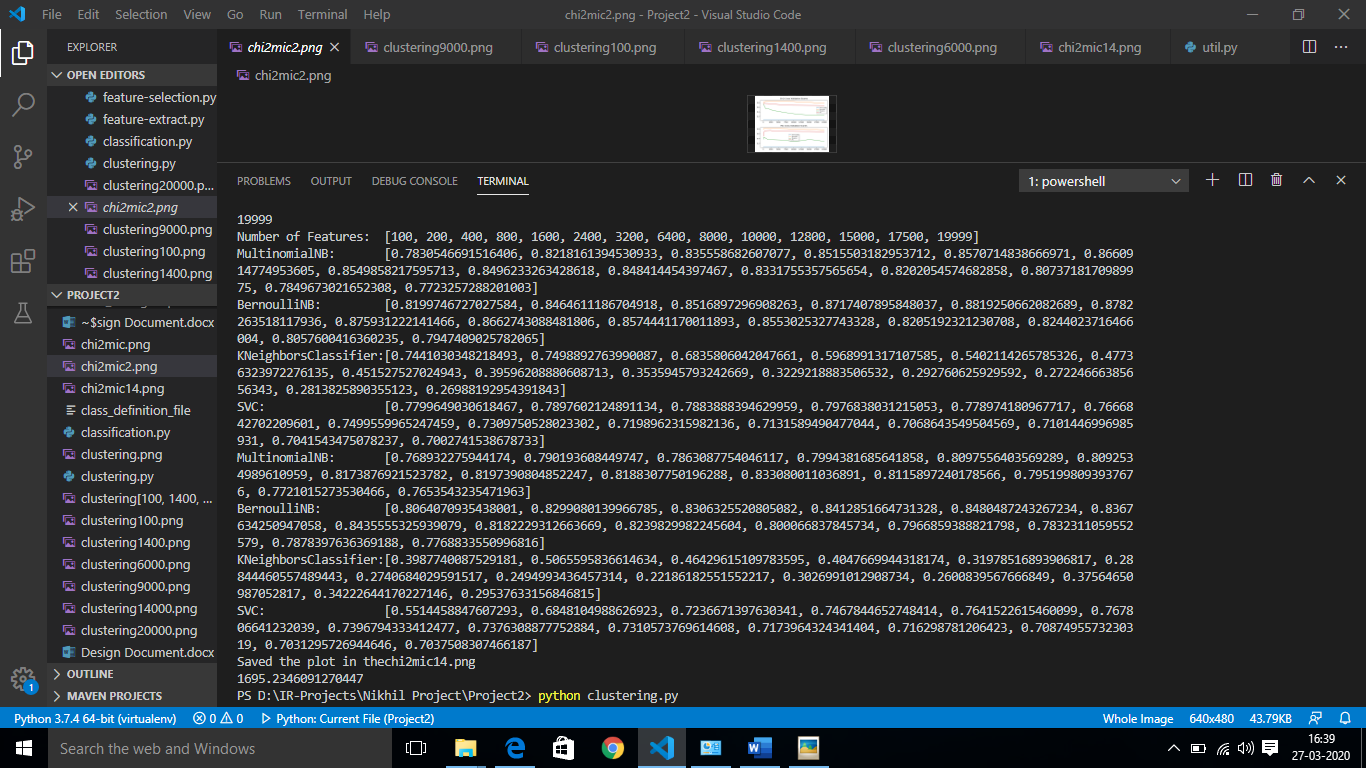
* **Chi-Square Method:** In feature selection, we aim to select features which are highly dependent on the response This method determines how much a calculated output varies with the expected value. i.e. when the calculated output is close to expected value then the Chi-Square value will be less. Thus chi2 method calculates the scores based upon how much a feature is dependent on final output. Higher the chi2 score higher is the importance and ranking of the feature, thus allowing us to know about the dependency.
* **Mutual information method:** MI measures how much information the presence/absence of a term contributes to making the correct classification decision in class. In simple terms how one features data gives information about another feature or the final output.

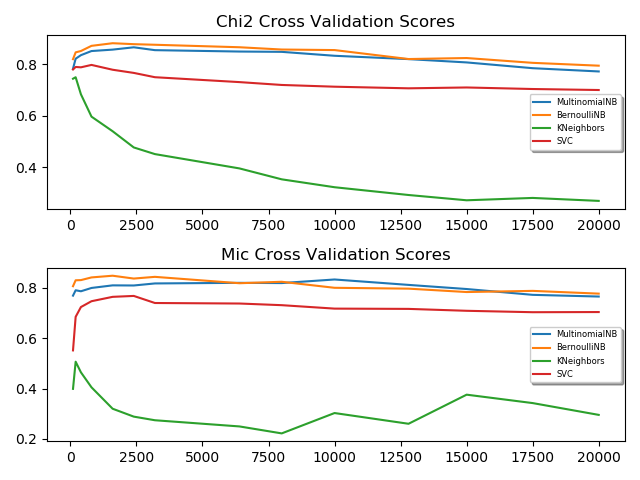
The results are finally plotted on a graph using matplot library to analyze the results on how the feature reduction techniques vary the performance of the classifier model i.e. to check which classifier performs well with less or high features. Below are few examples upon which cross-validation is performed on the feature sizes provided as list and respective data is being reduced using SelectKBest using Chi2 and Mutual Information method for feature reduction.

**Example: 1**

Number of features sizes in the list: 14

Feature Size list (k) = [100, 200, 400, 800, 1600, 2400, 3200, 6400, 8000, 10000, 12800, 15000, 17500, 19999]





**Result Analysis:**

As per the chi2 and MI score results and respective graphs plotted based upon the same result, it shows that the chi2 and MI scores for Multinomial and Bernoulli Classifiers are almost in similar, however, we could see both the graphs declining as the number of features increased, besides when best features are between 400 to 4000 both the classifiers perform better. On the other hand, SVC classifier, even though the scores decline as features increase which is similar, the model produces different chi2 and MI scores when features are between 100 to 400. Thus, proving that the SVC classifier model performs less when feature size is less or vice versa.

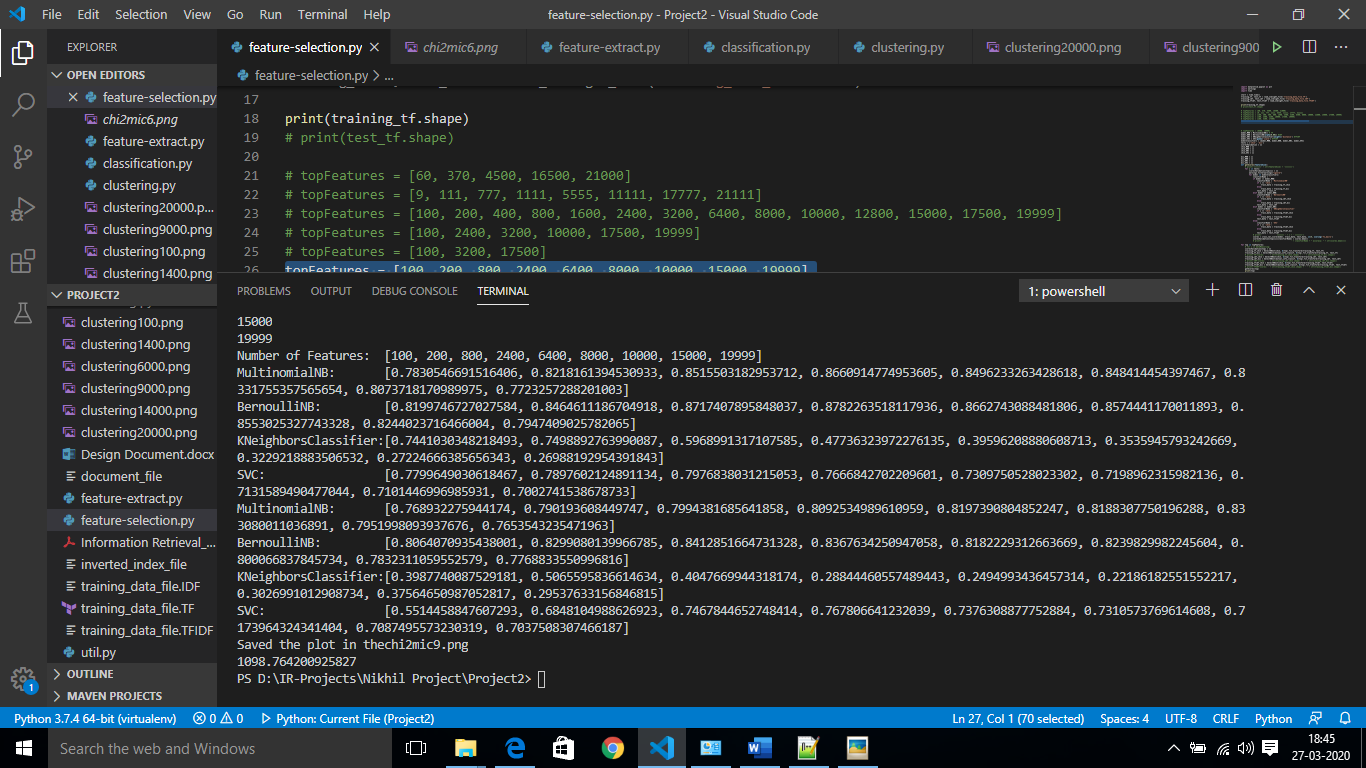
In contrast to other classifiers, KNeighbors Classifier produced abnormal chi2 and MI scores for different feature size, when feature size is between 100 to 400 the graph is steep. However, there is the rapid declination in the chi2 score graph as features increased, but MI scores show declination till features less than 8000 and later growth in KNeighbors Classifier performance. This concludes that KNeighbors is not the best given dataset it requires several tuning in the parameters available to perform better.

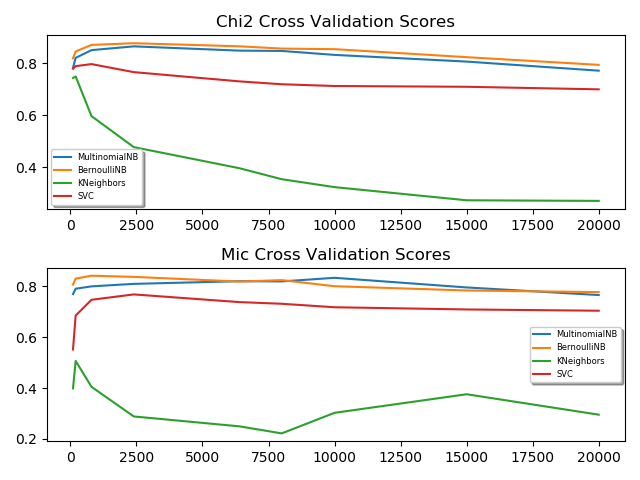
Finally, Chi-Square and MI scores based upon Dimensionality Feature Reduction behaved similarly for the given feature size for all the classifiers excluding KNeighbors. This proves the fact that considering the average number of features yields better performance of the classifiers.

**Example: 2**

Number of features sizes in the list: 9

Feature Size list (k) = [100, 200, 800, 2400, 6400, 8000, 10000, 15000, 19999]

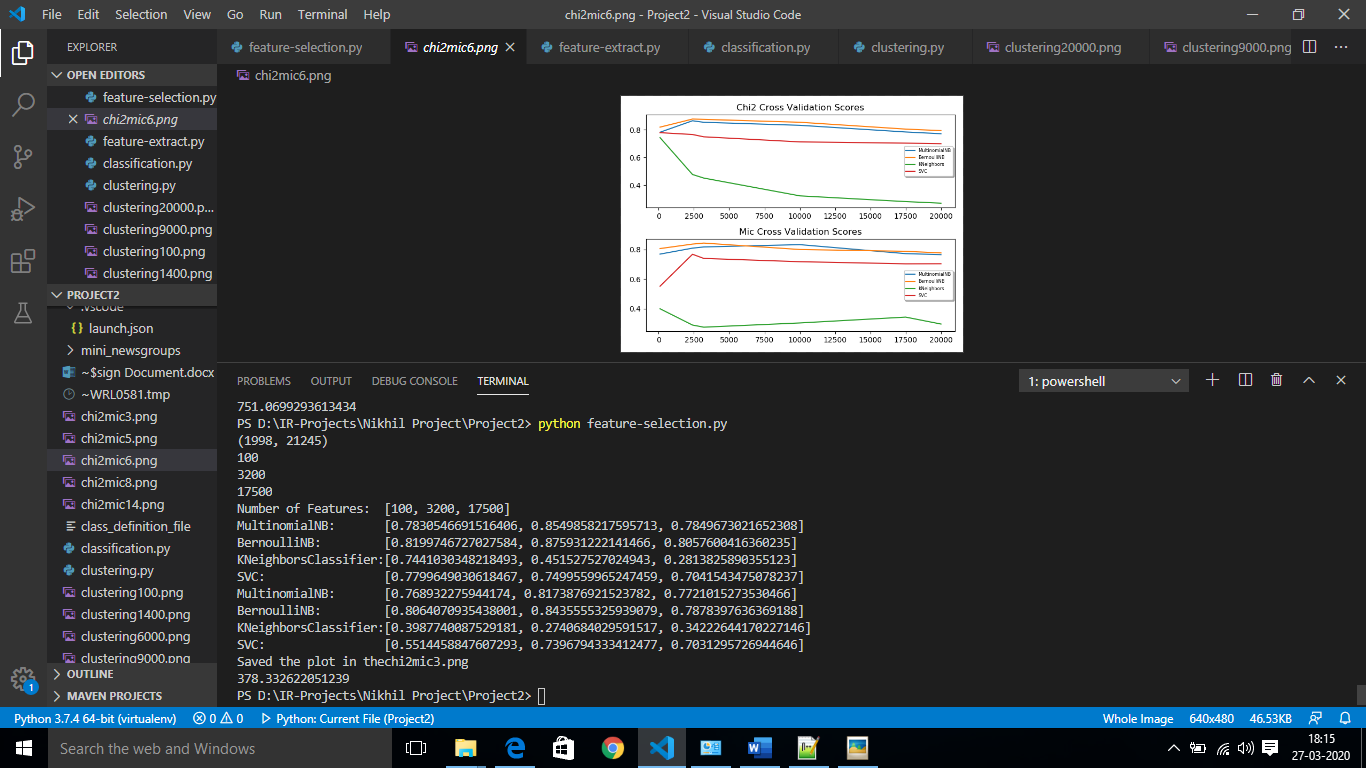


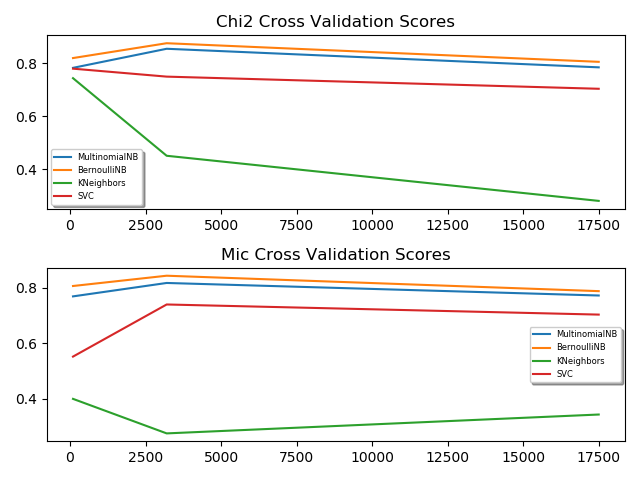


**Example: 3**

Number of features sizes in the list: 3

Feature Size list (k) = [100, 3200, 17500]

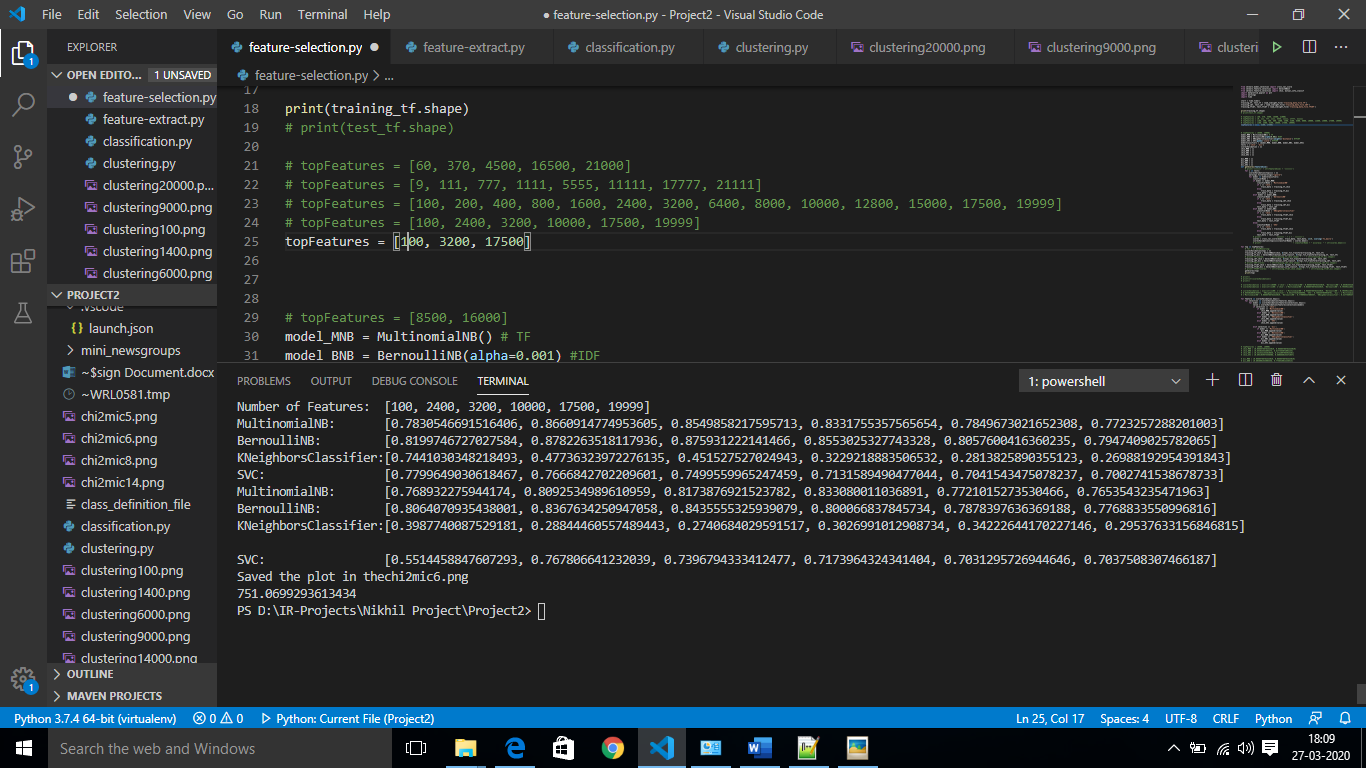


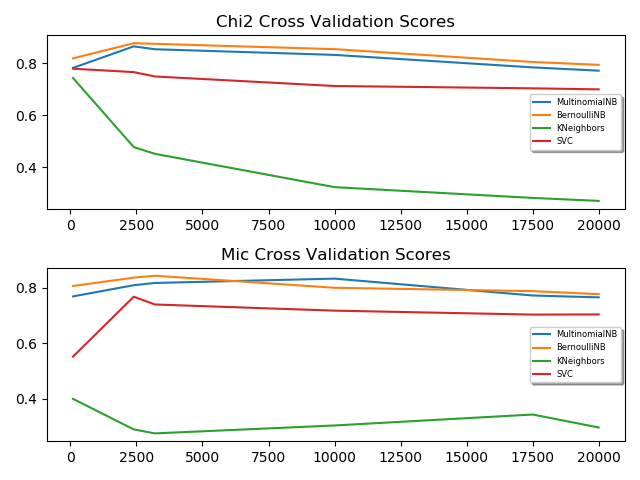


**Example: 4**

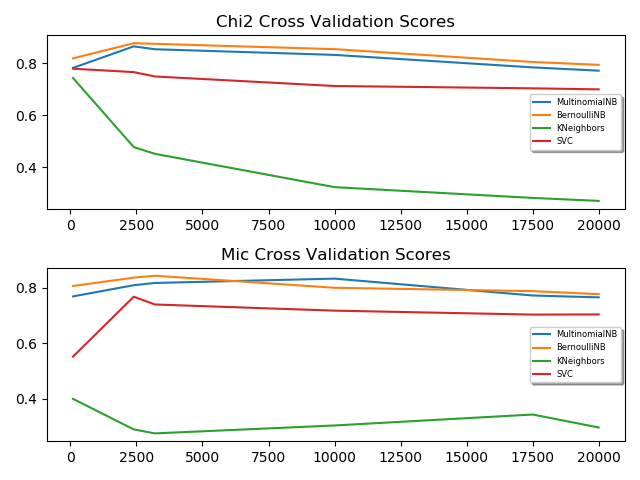
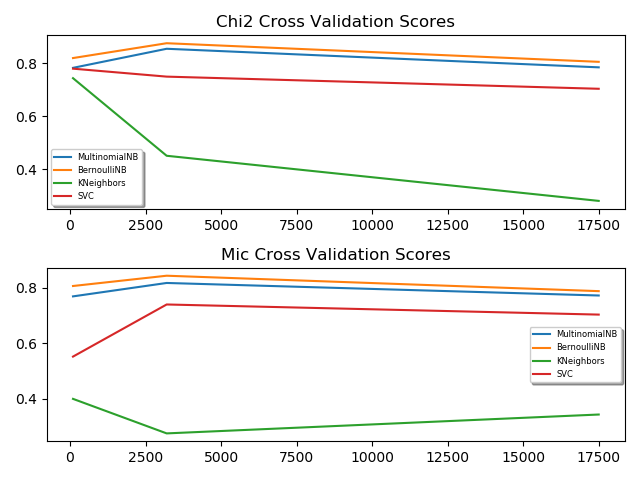
Number of features sizes in the list: 6

Feature Size list (k) = [100, 2400, 3200, 10000, 17500, 19999]

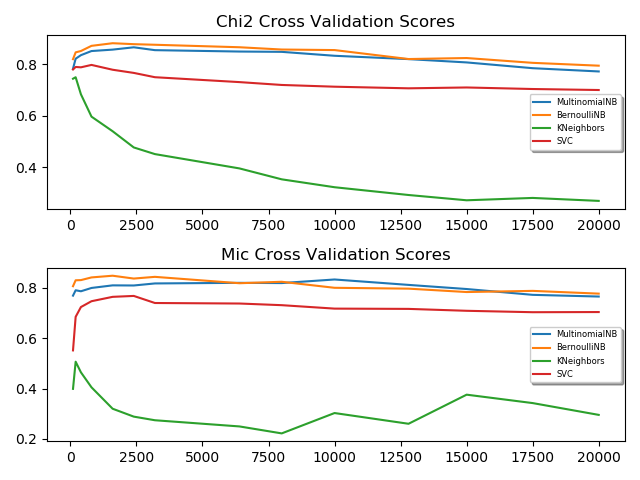
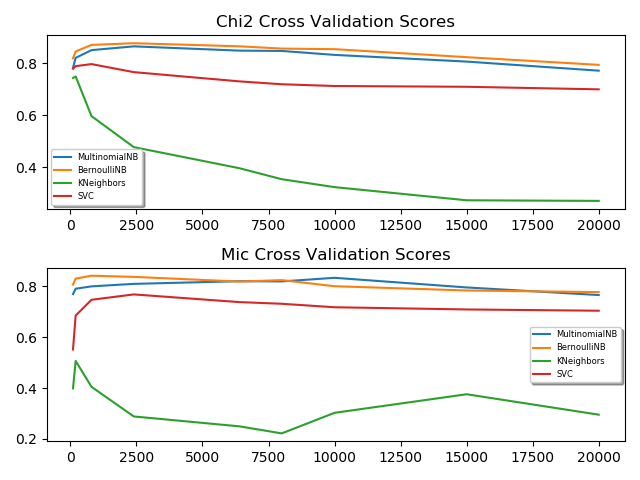




**Analysis:**



**(Example 3) (Example 4)**

**(Example 1) (Example 2)**

The above 4 graphs plotted for different feature size list and contain the chi2 and MI score distribution for different feature size vectors in different examples and high-level analysis of the 4 examples are discussed below.

* Example 1: = [100, 200, 400, 800, 1600, 2400, 3200, 6400, 8000, 10000, 12800, 15000, 17500, 19999]
* Example 2: = [100, 200, 800, 2400, 6400, 8000, 10000, 15000, 19999]
* Example 3: = [100, 3200, 17500]
* Example 4: = [100, 2400, 3200, 10000, 17500, 19999]

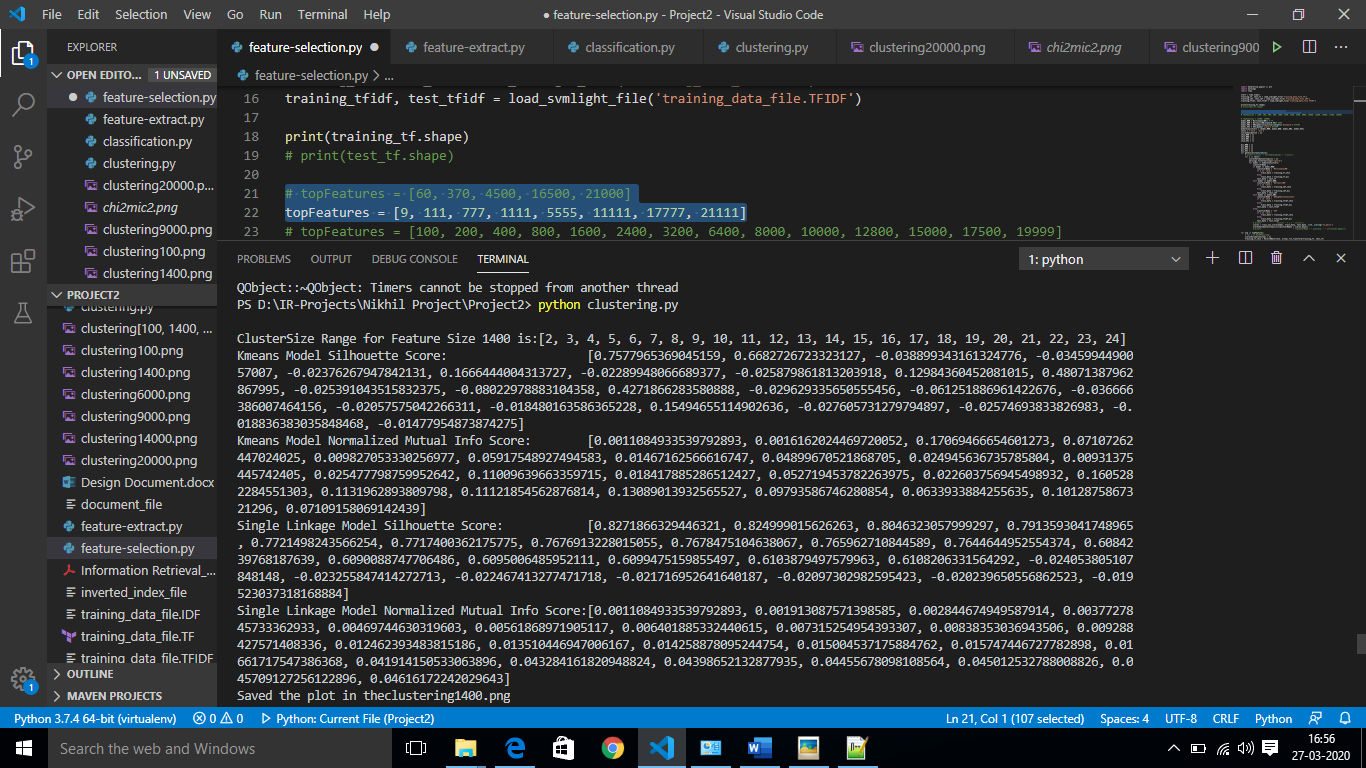
In all the 4 examples the chi2 scores for all the 4 classifiers declines after the number of features reach close to between 800 to 3500 for Multinomial, Bernoulli and SVC Classifier. The similar pattern is observed for MI scores as well for these classifier model, but the MI scores for SVC classifier shows sudden steeping when features size is almost 100 till features size is around 3000. SVC couldn’t find the information about the class by using a feature till the top features considered are around 3000.

However, KNeighbors Classifiers in all the examples performed poorly, there is a sudden declination in the chi2 and MI scores in all the examples till feature size is around 10000 and then a slight steep in the curves shows that the performance hike. However, a nominal F1 score a classifier to consider or perform better is above 0.5 which is satisfied in Multinomial and Bernoulli, SVC (till feature size < 10000). The KNN performed poorly and couldn’t be considered as a nominal classifier for such kind of dataset since there is a performance decline.

**Document Clustering:** python clustering.py

In this lab, the TFIDF file is used for document clustering, the TFIDF data is loaded and as we know that we have generated 21245 features in the Feature Extraction section. We apply Feature Selection SelectKBest technique which uses Chi2 method to obtain top K features for easier calculation of Document Clustering. We cluster the top K feature set using 2 methods KMeans Clustering and Agglomerative/Hierarchical Clustering Methods, later the performance of the clustering methods is evaluated using Silhouette Coefficient and Normalized Mutual Information metric scores. The graphs are plotted for 2 metric scores against 2 clustering algorithms for cluster size ranging from 2 to 25 for evaluation.

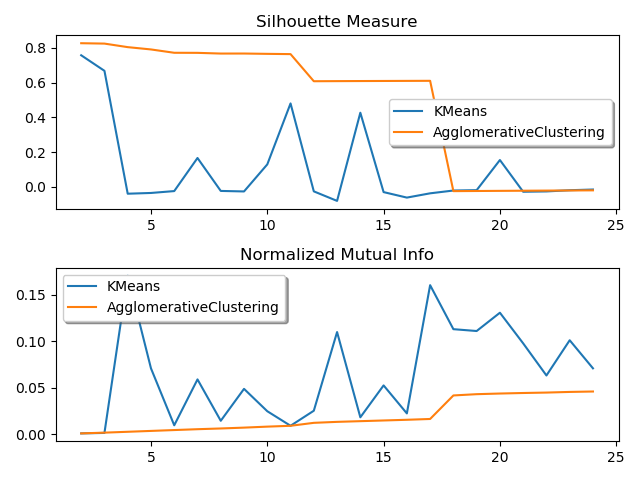
**Feature Size: 1400**

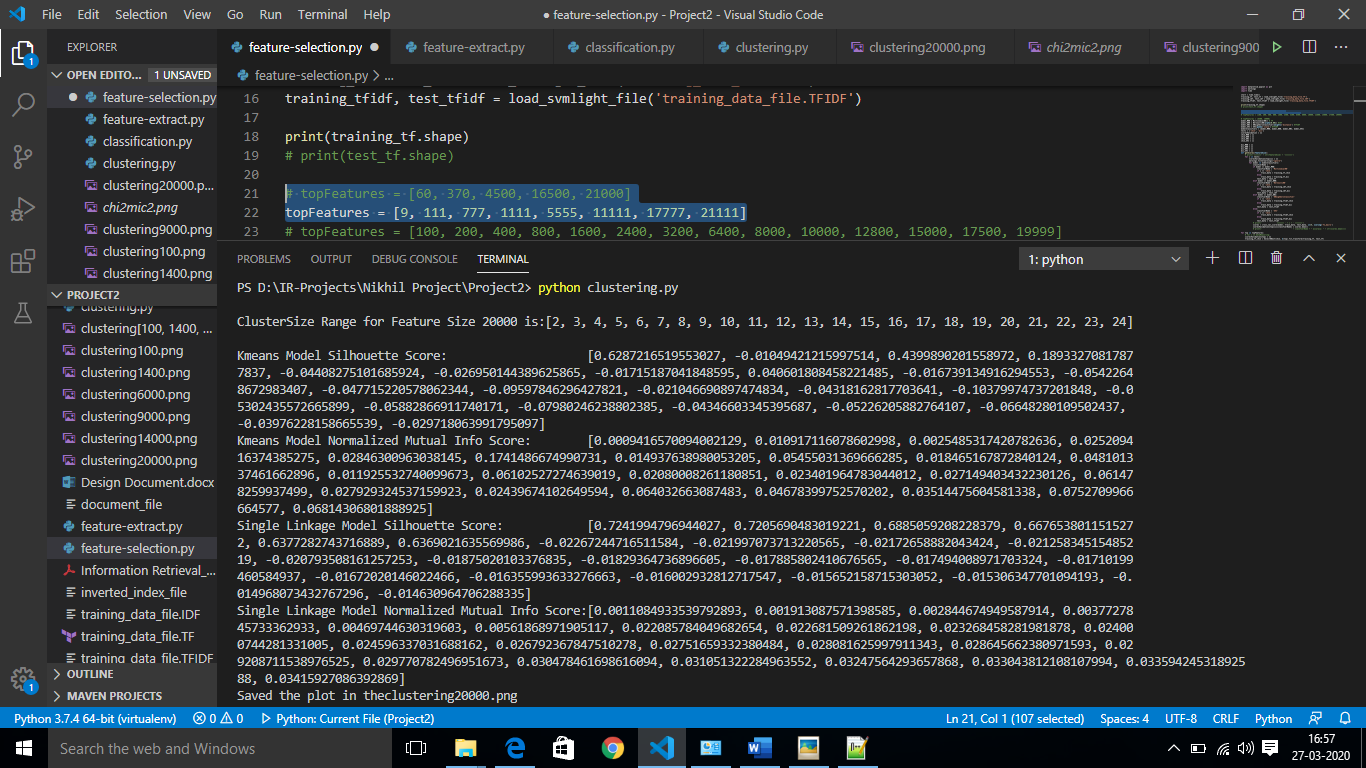


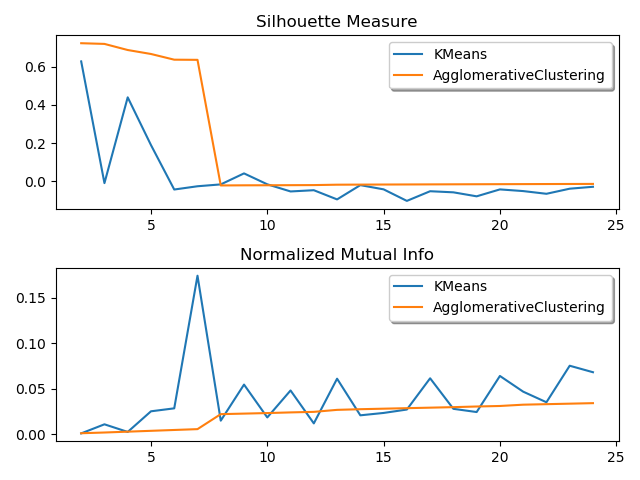
**Result Analysis:**

The SC scores seem to decline as the number of clusters increased in both hierarchical and KMeans clustering. However, the KMeans clustering method produces abnormal SC for different cluster sizes but overall it seems almost similar or declining. Besides, The NMI scores graphs are completely opposite to the SC scores graphs, here the MI scores tend to increase as the number of clusters increases for both clustering algorithms and KMeans performs abnormally as the clusters increase, but overall, it’s increasing.

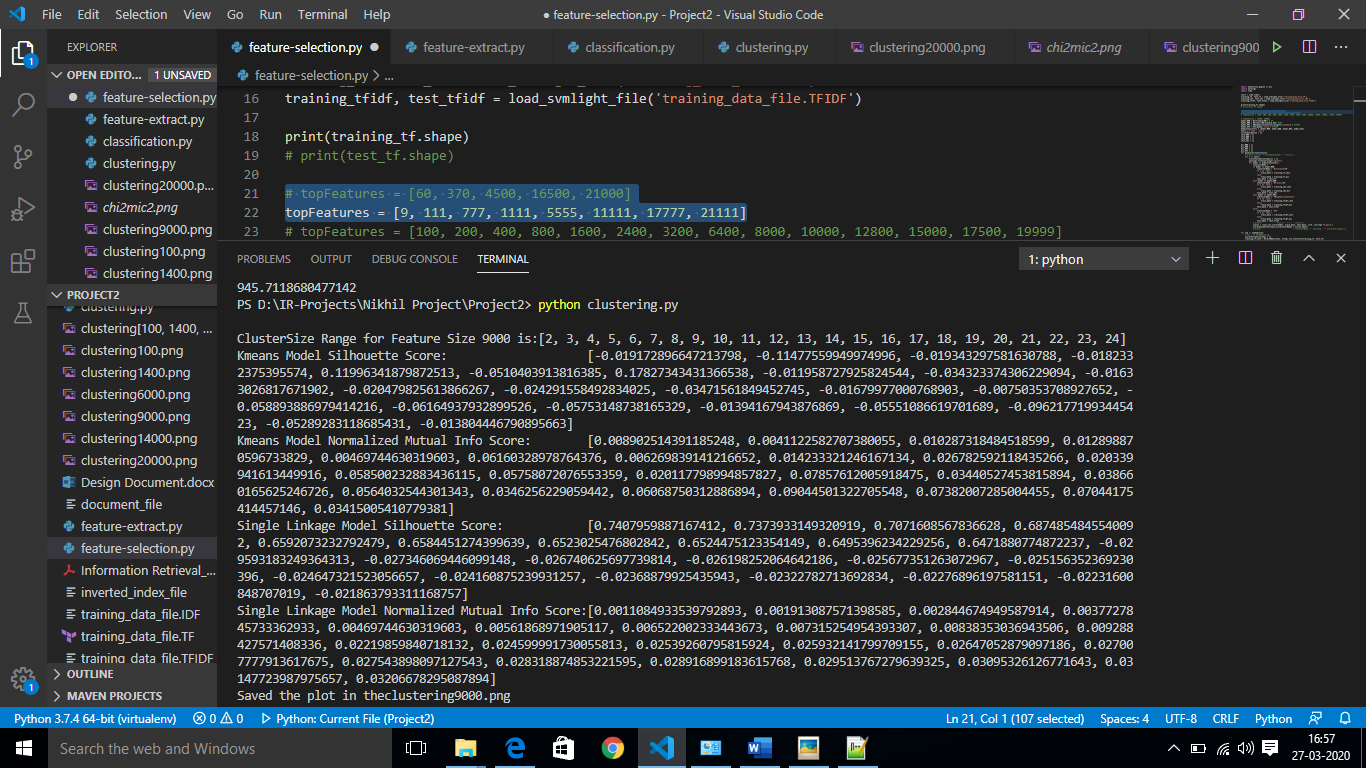
The opposite behaviour of the SC and NMI scores is because the calculation of SC doesn’t depend on the existing class labels and hence Silhouette Coefficient is generally higher for convex clusters such as density. When the number of clusters is less, the cluster tends to have high density whereas NMI score calculation depends on the class labels, therefore, there is an increase in the score as clusters increase i.e. the mutual dependence of the feature increases with the increase in the clusters. The below graphs shows the score distribution.

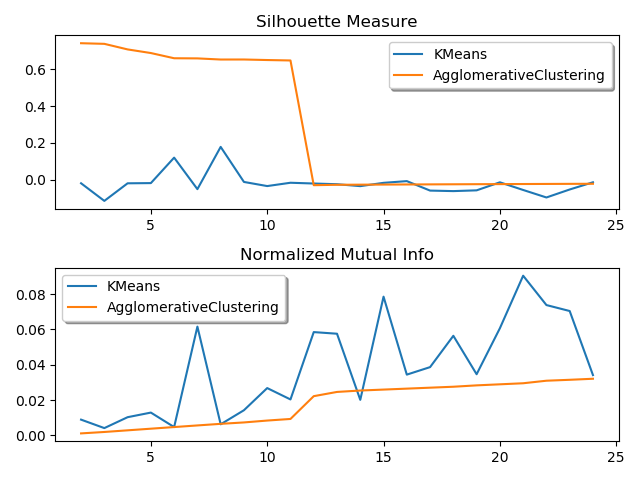
**Feature Size: 20000**



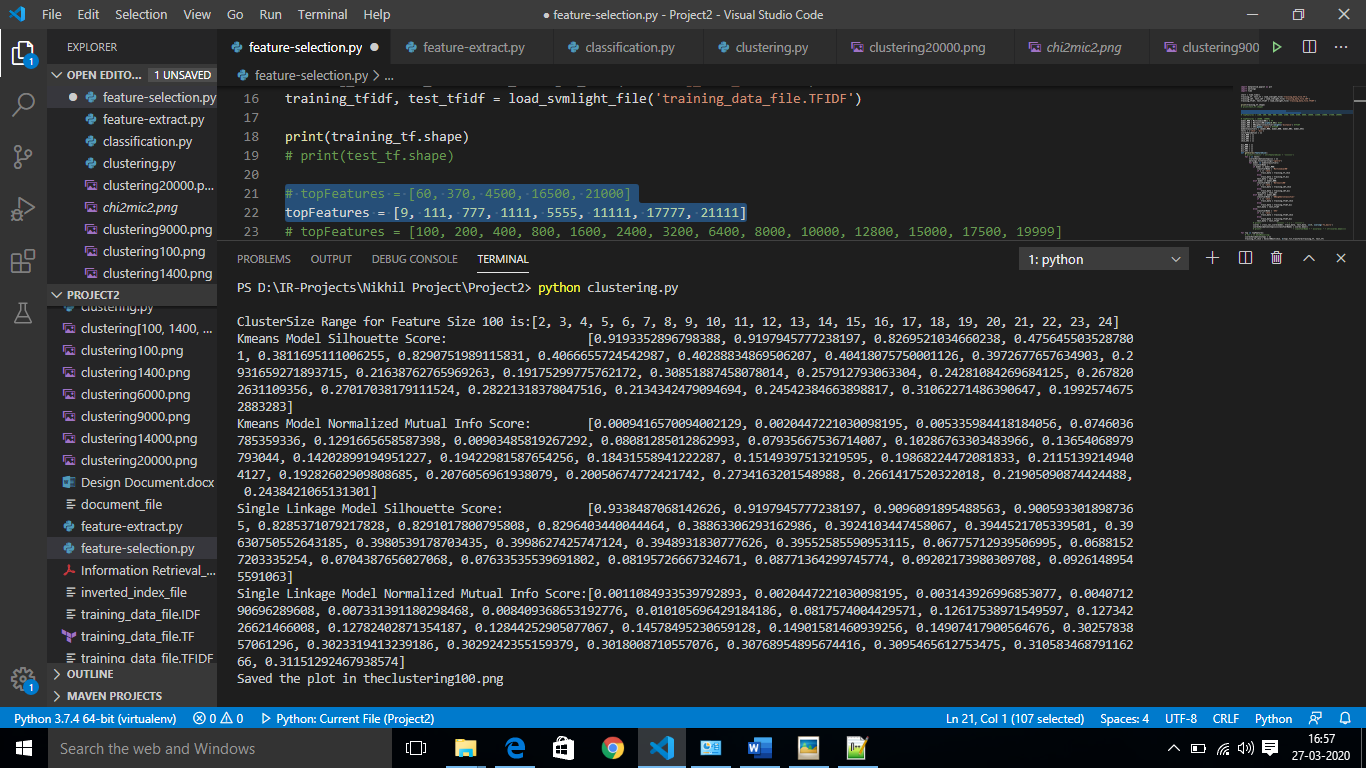


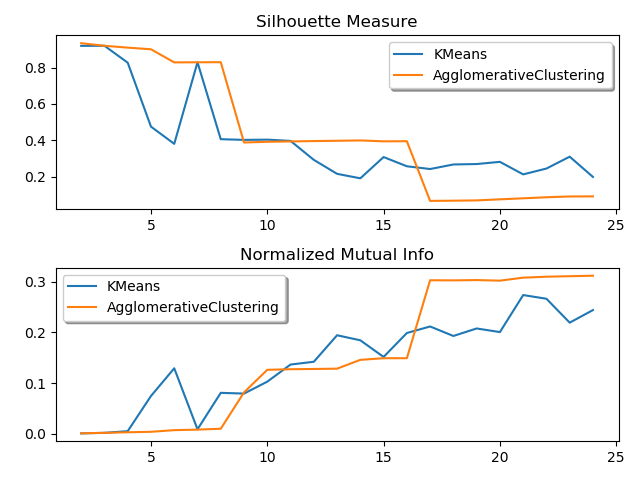
**Feature Size: 9000**



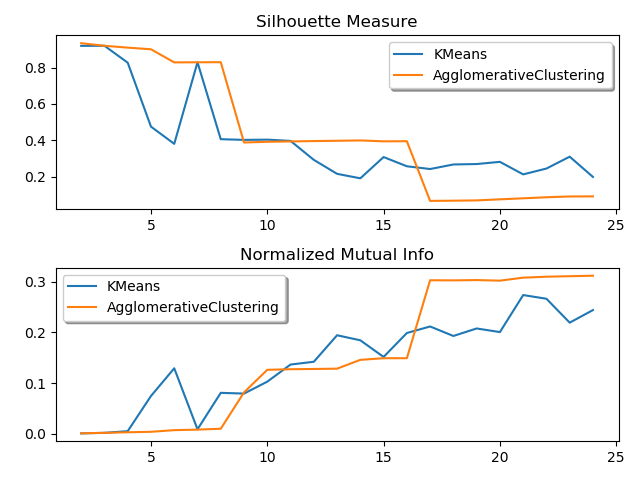
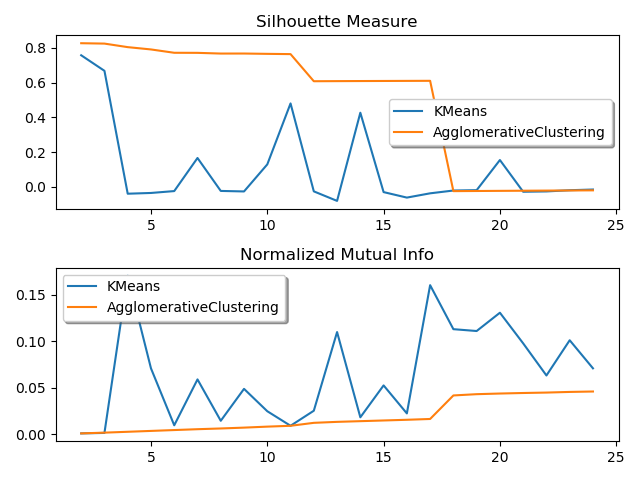


**Feature Size: 100**

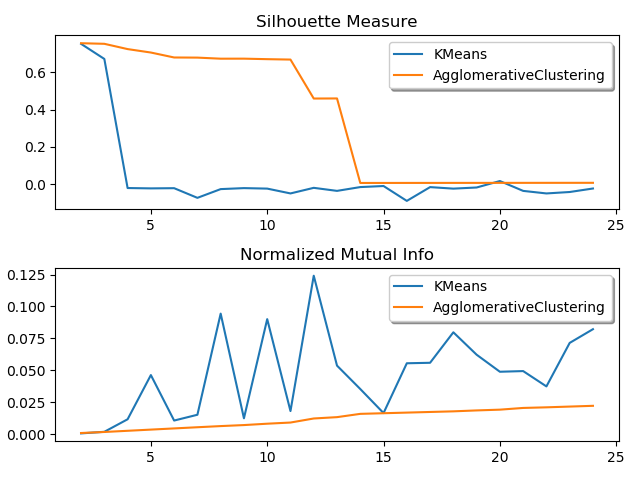
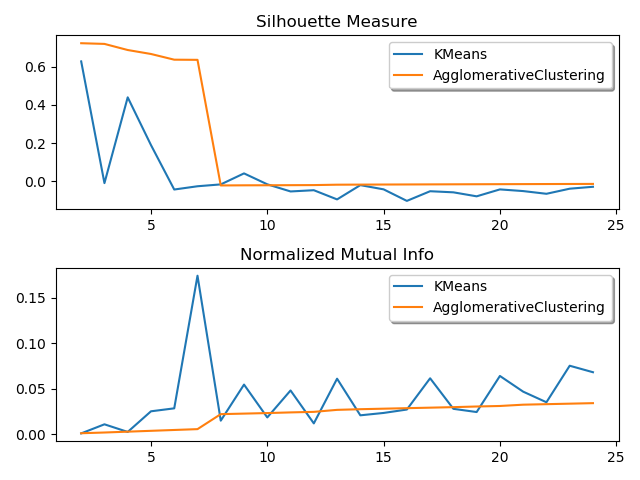




**Overall Analysis for Different Feature Size**

**Feature Size: 100 Feature Size: 1400**

**Feature Size: 9000 Feature Size: 20000**

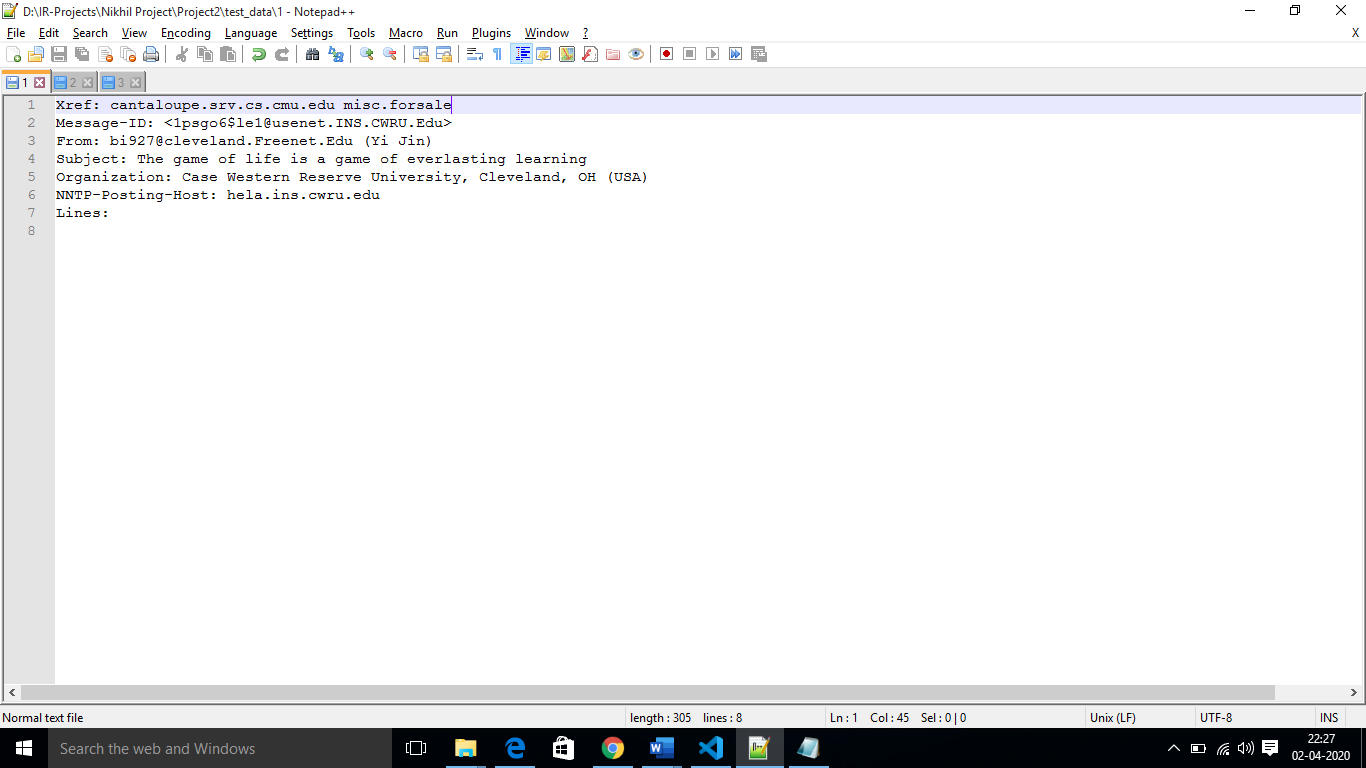
All the above graphs are plotted for different feature sizes for the same dataset i.e. TFIDF data. As the feature size increases the overall SC value for KMeans and Agglomerative Clustering decreases. However, for Agglomerative clustering, the SC declines rapidly at a less cluster size as feature size increases. For example, for K=20000 SC steeps down at cluster 8 and when K=1400 SC steeps down suddenly at cluster 18. Similarly, KMeans shows heavy abnormality when the feature size is the least and abnormity lessen for cluster range 5 to 25 as the feature size increases.

However, NMI scores for KMeans and Agglomerative Clustering keeps on increasing for different features sizes calculated against clusters ranging from 5 to 25. For feature size 100 there seems that Agglomerative Clustering performs better for higher clusters and in remaining all cases NMI scores of KMeans being abnormal it is higher and very rarely collides with NMI scores of the hierarchical clustering method.

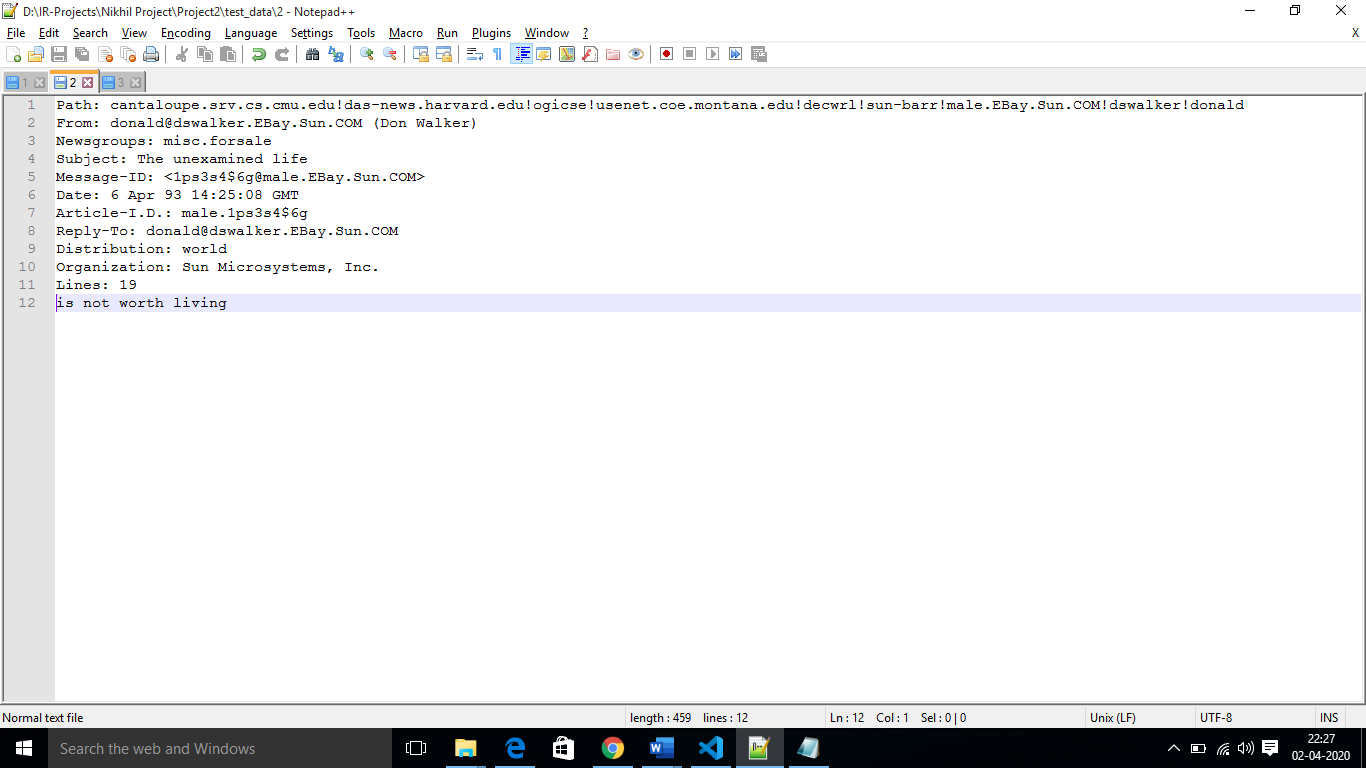
**Test Cases:**

Creating TF, IDF and TFIDF training data which is extracted from the documents stored in the test\_data directory and data present in the documents are

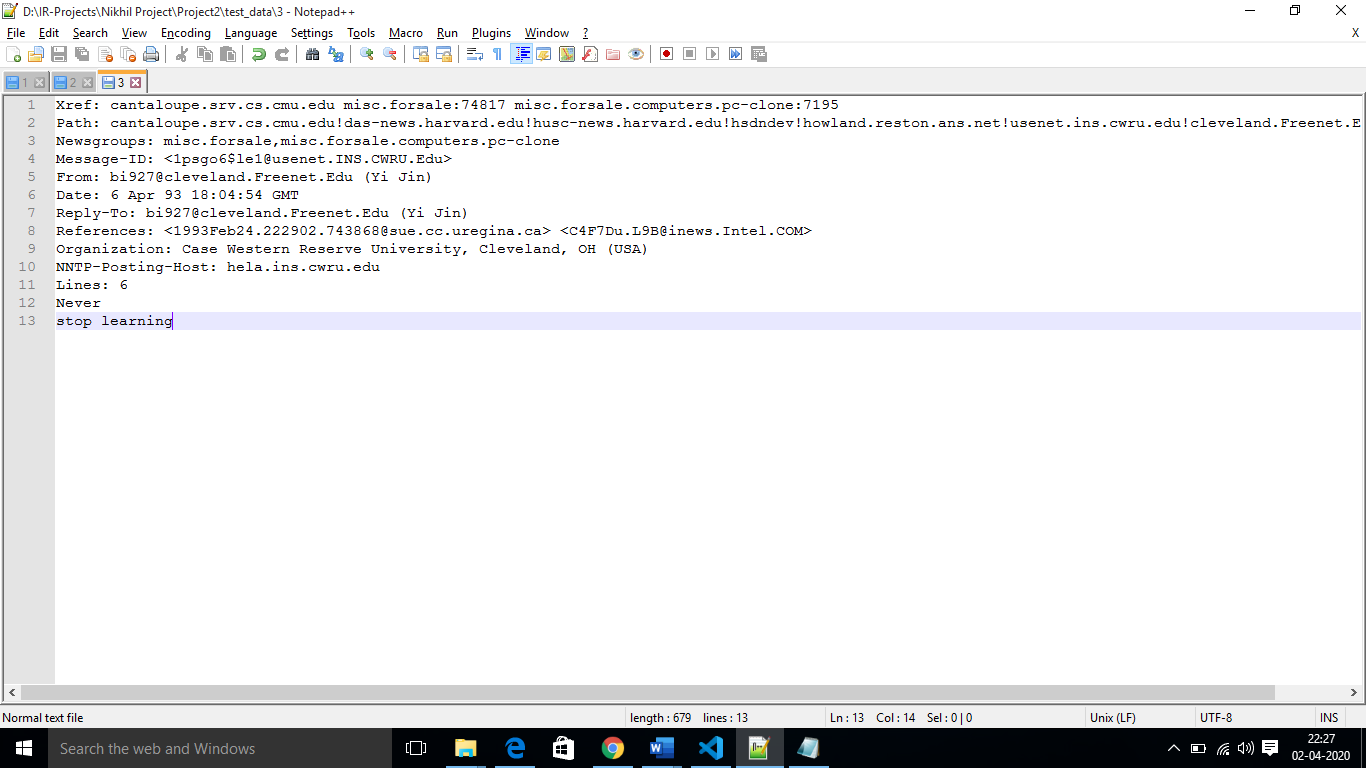
Document 1:



Document 2:



Document 3:

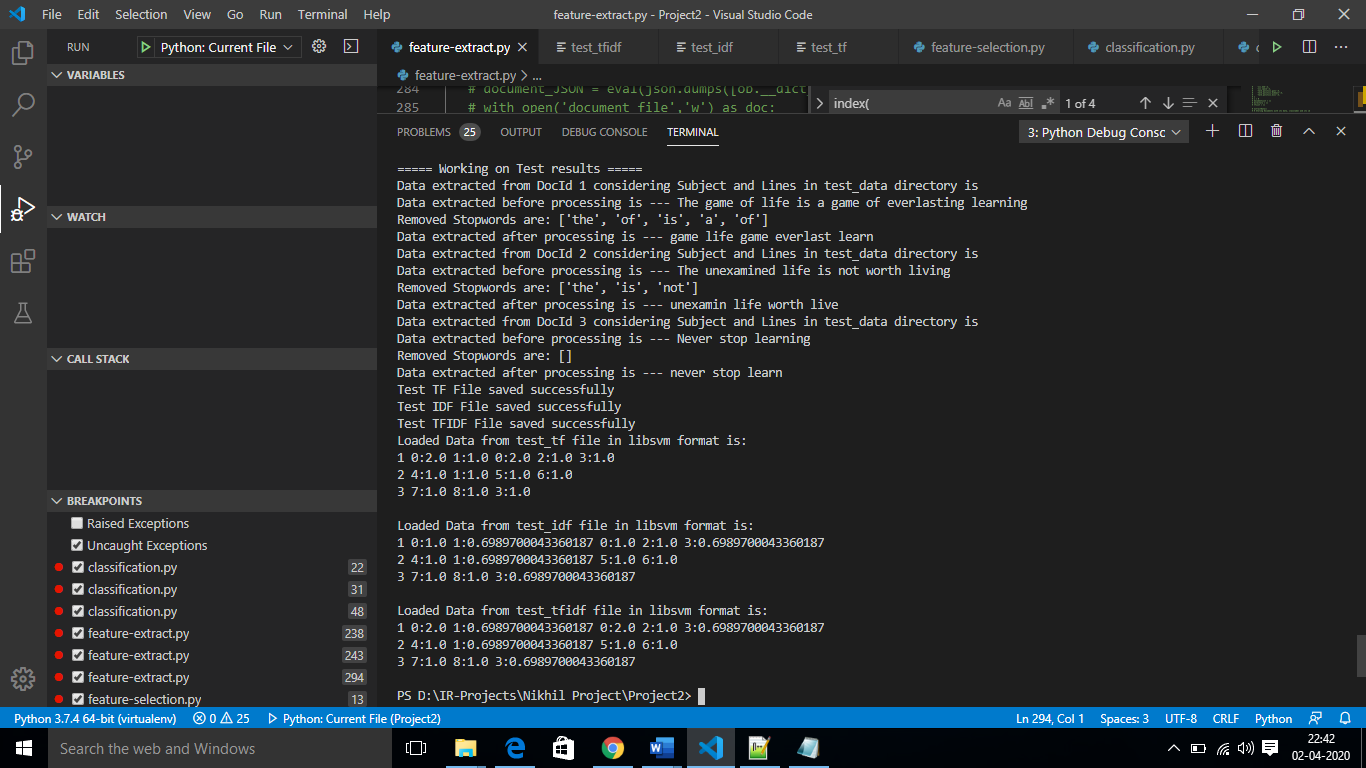


Data extracted before and after processing from the documents considering only Subject and Lines as the data

*Document 1: The game of life is a game of everlasting learning*

*Document 2: The unexamined life is not worth living*

*Document 3: Never stop learning*



Manual Inverted Index Result {'game': {'1': [1, 3]}, 'life': {'1': [2], '2': [2]}, 'everlast': {'1': [4]}, 'learn': {'1': [5], '3': [3]}, 'unexamin': {'2': [1]}, 'worth': {'2': [3]}, 'live': {'2': [4]}, 'never': {'3': [1]}, 'stop': {'3': [2]}}

Manually calculating TF, IDF and TFIDF values for words life and learn to cross check with the results.

Life = TF(life in query) \* IDF(life in Manual Result) = 1.0\* 0.6989700043360187= 0.6989700043360187

Learn = TF(learn in query) \* IDF(learn in Manual Result) = 1\* 0.6989700043360187= 0.6989700043360187

The same calculations are observed in the screenshot

To generate the files in libsvm format it each line in file is in the format of <class-label> <feature-id><feature-value> <feature-id><feature-value> ……… (till the last term of the processed data in the document)

The <class-label> refers to the filename. However, in the real dataset it refers to the class labels according to the use case.

<feature-id> : It corresponds to the index of the term in the Inverted Index matrix generated

<feature-value>: It contains values of TF, IDF, and TFIDF for generating TF, IDF, and TFIDF test files, which are later stored locally and loaded to check if the data saving is proper or not.

As we can see in the screenshot, we find data stored is the same as we calculated manually for the above scenario. Thus, ensuring that the Feature extraction feature is working perfectly.