

PROFESSIONAL TRAINING REPORT

At

**Sathyabama Institute of Science and Technology
(Deemed to be University)**

Submitted in partial fulfilment of the requirements for the award of
Bachelor of Engineering Degree in Computer Science and Engineering

By

Dinesh Sheelam

(Reg.No. 40731091)



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SCHOOL OF COMPUTING

SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY

JEPPIAAR NAGAR, RAJIV GANDHI SALAI,

CHENNAI – 600119, TAMILNADU

April 2023



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with Grade "A" by NAAC

(Established under Section 3 of UGC Act, 1956)

JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI- 600119

www.sathyabama.ac.in



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **Dinesh Sheelam (40731091)** who carried out the project entitled "**Duplicate Question Detection**" under my supervision from Jan 2023 to April 2023.

Internal Guide Name

Dr.A. Annie Micheal

Head of Department

Dr. L. Lakshmanan M.E., Ph.D.

Dr. S. Vigneshwari M.E., Ph.D.

Submitted for Viva voce Examination held on 17-04-2023

Internal Examiner

External Examiner

DECLARATION

I **Dinesh Sheelam** hereby declare that the Project Report entitled "**Duplicate Question Detection Using Natural Language Processing**" done by me under the guidance of **Dr.A.Annie Micheal** is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering.

DATE: 17-04-2023

PLACE: CHENNAI



SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management** of **SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. Sasikala M.E., Ph.D., Dean**, School of Computing , **Dr.S.Vigneshwari M.E., Ph.D.**, and **Dr.L.Lakshmanan M.E., Ph.D.**, Heads of the Department of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr.A.Annie Micheal** for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

TRAINING CERTIFICATE

COGNIBOT LABS
CERTIFICATE
OF TRAINING

THE CERTIFICATE IS PRESENTED TO

Dinesh Sheelam

40731091

from Sathyabama Institute of Science and Technology for successfully completing the 45 hours professional training on Artificial Intelligence - I (Machine Learning) conducted by Cognibot Labs.

Awarded on: October 7, 2022



Harikarasudhan
Trainer - Cognibot Labs

ABSTRACT

The code performs duplicate question detection using Natural Language Processing (NLP) techniques. The dataset of questions is preprocessed by converting to lowercase, removing punctuation and digits, removing stop-words, and stemming. the code splits the dataset into training and test sets using the `train_test_split` function. The training set is used to fit a TF-IDF vectorizer to transform the questions into a matrix of features.

KMeans clustering is then applied to group similar questions into clusters. The preprocessed questions are then transformed using the TF-IDF vectorizer to obtain a matrix of features. The KMeans clustering algorithm is applied to group similar questions into clusters. Duplicate questions are detected by computing cosine similarity between questions and identifying those with high similarity scores. The program prompts the user to enter a question, preprocesses it, and computes its similarity to the training set questions. If the question is found to be a duplicate, the program outputs the duplicate question(s) from the training set.

cosine similarity for duplicate question detection, the first step is to preprocess the questions. This involves removing stop words, punctuation, and special characters and then tokenizing the questions into words. Next, the words in each question are converted into vectors using techniques such as Word2Vec or GloVe.

These vectors represent the semantic meaning of each word and capture the context in which it is used.

Once the questions have been converted into vectors, their similarity can be calculated using cosine similarity. The cosine similarity measure calculates the cosine of the angle between two vectors in a high-dimensional space. The closer the cosine value is to 1, the more similar the two vectors are, and hence the more similar the questions are.

TABLE OF CONTENTS

CHAPTER No.	TITLE	PAGE NO
	ABSTRACT	I
	CONTENTS	II
	LIST OF FIGURES	IV
1	INTRODUCTION	1
	1.1 DUPLICATE QUESTION DETECTION	1
	1.2 ESTIMATING DATASET INFORMATION	1
	1.3 MACHINE LEARNING ALGORITHMS	2
2.	AIM AND SCOPE OF THE PRESENT INVESTIGATION	6
	2.1 AIM	6
	2.2 SCOPE	6
	2.3 ATTRIBUTES INFORMATION	6
	2.4 DATA PREPARATION	6

	2.5 READING DATASET	7
3.	EXPERIMENT AND ALGORITHMS	8
	3.1 K-MEANS CLUSTERING	8
	3.2. IMPORTED LIBRARIES	9
	3.3. LIBRARIES USED	9
	3.4. IMPLEMENTATION OF ALGORITHMS	13
4.	RESULTS AND DISCUSSION, PERFORMANCE ANALYSIS	16
5.	SUMMARY AND CONCLUSIONS	17
A	APPENDIX	18
B	SCREENSHOTS	19
C	SOURCE CODE	26
D	REFERENCES	31

LIST OF FIGURES

FIGURE NO.	FIGURE NAME	PAGE NO.
1.1	Excel sheet of the given questions dataset	2
1.2	TF-IDF Vectorizer	3
1.3	K-means clustering	5
3.1	Various python libraries for Machine Learning	7
3.2	Example implementation of Seaborn library	9
3.3	Architechture Diagram	11
3.4	Imported libraries in jupyter notebook	13
3.5	Testing and Training data	14

CHAPTER-1: INTRODUCTION

1.1 Duplicate Question Detection

Duplicate question detection is the process of identifying pairs of questions that have similar or identical meanings. It is commonly used in online question answering platforms to avoid redundancy and improve the user experience. By identifying and removing duplicate questions, the platform can provide a more diverse set of questions for users to explore and reduce the time and effort needed to search for information. Duplicate question detection can also be used in customer support services, where it is important to avoid redundant information and provide timely and accurate responses to customer inquiries. Additionally, it can be used in chatbots to improve their ability to understand and respond to user queries.

Overall, duplicate question detection is an important task in various domains and can help improve the efficiency and accuracy of question answering systems, customer support services, and chatbots. The techniques used in the above code can be adapted and extended to different datasets and applications to achieve this goal.

1.2 ESTIMATING DATASET INFORMATION

The given dataset contains 4999 rows and 1 column. It contains only one feature i.e. questions.

Dataset is given below (Fig.1.1)

questions.csv - Excel

File Home Insert Page Layout Formulas Data Review View Developer Help Tell me what you want to do

A1 question

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1	question																		
2	What is the step by step guide to invest in share market in india?																		
3	What is the story of Kohinoor (Koh-i-Noor) Diamond?																		
4	How can I increase the speed of my internet connection while using a VPN?																		
5	Why am I mentally very lonely? How can I solve it?																		
6	Which one dissolve in water quikly sugar, salt, methane and carbon di oxide?																		
7	Astrology: I am a Capricorn Sun Cap moon and cap rising...what does that say about me?																		
8	Should I buy tiago?																		
9	How can I be a good geologist?																		
10	When do you use ä, instead of ä—?																		
11	Motorola (company): Can I hack my Charter Motorola DCX3400?																		
12	Method to find separation of slits using fresnel biprism?																		
13	How do I read and find my YouTube comments?																		
14	What can make Physics easy to learn?																		
15	What was your first sexual experience like?																		
16	In which location did Mahatma Gandhi originate from?																		
17	Can you tell me where Mahatma Gandhi was born?																		
18	What does manipulation mean?																		
19	Why do girls want to be friends with the guy they reject?																		
20	From what place did Mahatma Gandhi come into the world?																		
21	Which is the best digital marketing institution in banglore?																		
22	Why do rockets look white?																		
23	What's causing someone to be jealous?																		
24	What are the questions should not ask on Quora?																		
25	How much is 30 kV in HP?																		
26	What does it mean that every time I look at the clock the numbers are the same?																		
27	What are some tips on making it through the job interview process at Medicines?																		
28	What is web application?																		
29	Does society place too much importance on sports?																		
30	What is best way to make money online?																		
31	How should I prepare for CA final law?																		
32	What's one thing you would like to do better?																		

questions

Ready Accessibility: Unavailable

Fig 1.1 - Excel sheet of the given questions dataset

1.3 COMMON MACHINE LEARNING ALGORITHMS AND GOALS

The variety of machine learning algorithms are used they are -

1.3.1 Text Preprocessing

Text preprocessing is a critical step in natural language processing (NLP) that involves cleaning and transforming raw text data into a format that is suitable for machine learning algorithms. Text preprocessing techniques are used to remove unwanted elements from the text data, such as punctuation, digits, and stopwords,

and to convert the text into a standard format that can be processed by machine learning algorithms.

Common text preprocessing techniques include:

1. Lowercasing: converting all characters in the text to lowercase to reduce the complexity of the text data.
2. Tokenization: breaking the text data into individual words or tokens.
3. Punctuation removal: removing all punctuation marks from the text data.
4. Digit removal: removing all digits from the text data.
5. Stopword removal: removing all common words such as "the", "and", "a", etc. that do not carry much meaning.
6. Stemming: reducing words to their base form or stem, e.g., converting "running", "runs", and "run" to "run".

1.3.2 TF-IDF Vectorizer (Term frequency and Inverse Document frequency):

A TfidfVectorizer is used to transform the preprocessed text data into a matrix of features. This algorithm assigns a weight to each word in the text data based on its frequency in the document and inverse frequency in the corpus.

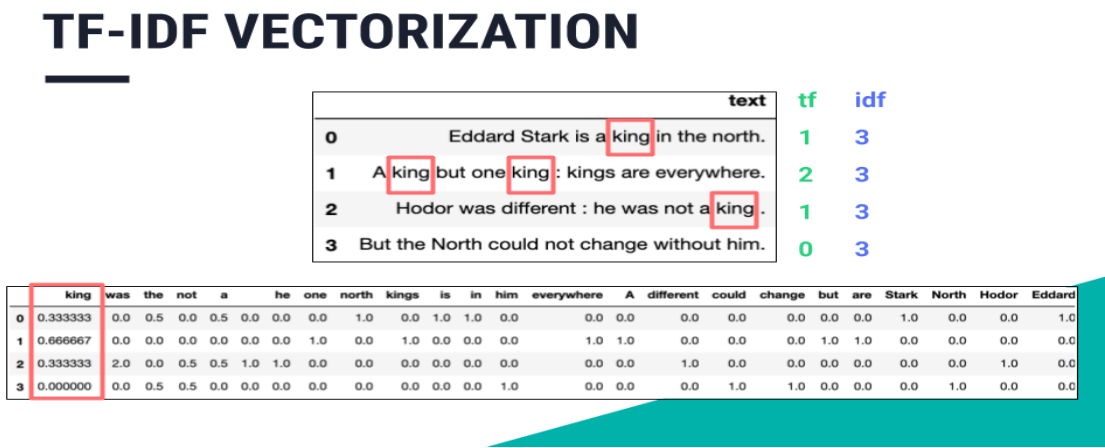


Fig – 1.2 TF-IDF Vectorizer

1.3.3 Cosine Similarity:

The cosine similarity algorithm is used to measure the similarity between pairs of questions. It computes the cosine of the angle between two vectors representing the text data, which gives a measure of their similarity.

1.3.4 Algorithm

1. Kmeans Clustering:

The KMeans algorithm is used to cluster similar questions together based on their similarity scores. It partitions the data into K clusters by minimizing the sum of squared distances between the points and their cluster centers.

K-means is a clustering algorithm that partitions a dataset into k clusters, where k is a user-defined parameter. The algorithm works by first randomly selecting k data points to be the initial centroids of the clusters. It then assigns each data point in the dataset to the nearest centroid, based on the Euclidean distance between the data point and the centroid. After all data points have been assigned to a cluster, the algorithm calculates the mean of all data points in each cluster, and moves the centroid to the new mean location. This process is repeated until the centroids no longer move significantly, or a maximum number of iterations is reached.

The K-means algorithm seeks to minimize the sum of squared distances between each data point and its assigned centroid. This objective function is known as the within-cluster sum of squares (WCSS) and is a measure of the compactness of each cluster. The algorithm iteratively updates the centroids to minimize the WCSS until convergence.

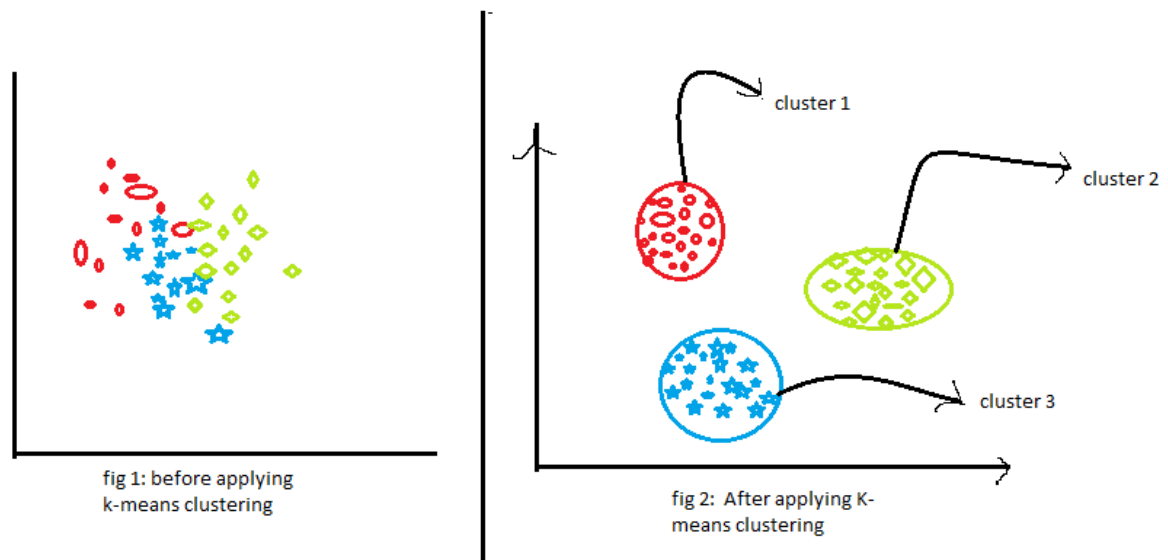


Fig – 1.3 K-means clustering

K-means has several advantages, including its simplicity, scalability, and efficiency on large datasets. However, it also has some limitations. One major limitation is that the algorithm requires the user to specify the number of clusters (k) in advance. Choosing the optimal value of k can be difficult and may require trial and error. Additionally, the algorithm can be sensitive to the initial placement of the centroids, and may converge to a suboptimal solution.

Despite its limitations, K-means is widely used in many applications, including image segmentation, market segmentation, and anomaly detection. Its simplicity and efficiency make it a popular choice for exploratory data analysis and as a baseline clustering algorithm for comparison with more advanced techniques.

To detect the duplicate questions in the given questions data set using NLP

CHAPTER-2

AIM AND SCOPE OF PRESENT INVESTIGATION

2.1 Aim:

To detect the duplicate questions in the given questions data set using NLP

2.2 Scope:

The scope of the code is to create a machine learning model that can detect duplicate questions by clustering similar questions together and comparing them using cosine similarity. The code uses natural language processing (NLP) techniques like tokenization, stop word removal, stemming, and vectorization to preprocess the text data and convert it into a numerical format that can be used by machine learning algorithms. The K-means clustering algorithm is used to cluster similar questions together based on their TF-IDF scores, and cosine similarity is used to compare new questions with the existing ones to detect duplicates. The code can be used to improve the functionality of question-answering systems, online forums, or any platform that deals with a large volume of text data by preventing repetitive or redundant questions.

2.3 Attributes Information :

The dataset only contains one attribute named "question" and it consists of 4999 questions. The code uses this dataset to demonstrate the process of duplicate question detection using text preprocessing, clustering, and similarity scoring.

2.4 Data Preparation :

In this project as a backbone tool python is used to carry out machine learning concepts. With the help of a software called Anaconda Navigator, a jupyter notebook is launched where it is already installed along with the navigator. jupyter notebook is an open-source web application that allows to create and share documents and has live code and also visualization.

After importing the required libraries the dataset will be read in the note book with help of data frame (two-dimensional labeled data structure with columns of potentially different types) and read_csv(desired file type).

2.5 Reading Dataset :

```
df = pd.read_csv("questions.csv")
```

```
In [9]: df = pd.read_csv("questions.csv")
```

The dataset provided cannot always be a fully valued set, in that case we need to prepare the data in such a way the machine understands what is the value that has been entered.

The dataset is now ready for creating a machine learning model.

```
In [10]: df
```

```
Out[10]:
```

	question
0	What is the step by step guide to invest in sh...
1	What is the story of Kohinoor (Koh-i-Noor) Dia...
2	How can I increase the speed of my internet co...
3	Why am I mentally very lonely? How can I solve...
4	Which one dissolve in water quikly sugar, salt...
...	...
4994	Who is currently winning the presidential elec...
4995	What has a better ROI, marketing on radio sta...
4996	Which mobile is good for 50k?
4997	Is the character Jane in the movie Predestinat...
4998	How true are near death experiences?

4999 rows × 1 columns

CHAPTER- 3

EXPERIMENT AND ALGORITHMS

We need to perform clustering to detect duplicate question. So, we used kmeans clustering algorithm for clustering process.

3.1 K-Means Clustering :

K-means is a popular unsupervised machine learning algorithm used for clustering. The algorithm partitions n observations into k clusters, where each observation belongs to the cluster with the nearest mean.

The algorithm works as follows:

1. Initialize k centroids randomly.
2. Assign each observation to the nearest centroid.
3. Recalculate the centroid of each cluster.
4. Repeat steps 2 and 3 until the centroids converge, or a specified number of iterations is reached.

K-means aims to minimize the sum of squared distances between each observation and its assigned centroid, known as the "within-cluster sum of squares" or "inertia". The algorithm is sensitive to the initial placement of the centroids and may converge to suboptimal solutions, so it is often run multiple times with different initializations to increase the chances of finding the global minimum of the inertia.

K-means is commonly used for data exploration, image segmentation, anomaly detection, and customer segmentation in marketing. It has some limitations, such as the requirement for a predetermined number of clusters, sensitivity to outliers and skewed data, and difficulty in handling non-linearly separable clusters.

3.2. Imported Libraries

Libraries are collections of prewritten code that users can use to optimize tasks. In project as python is used for implementation tool, it has the most libraries as compared to other programming languages. More than of 60% machine learning developers use and goes for python as it is easy to learn. As python has comparatively large collection of libraries let's Ilook at the libraries that came in handy for mammographic dataset.

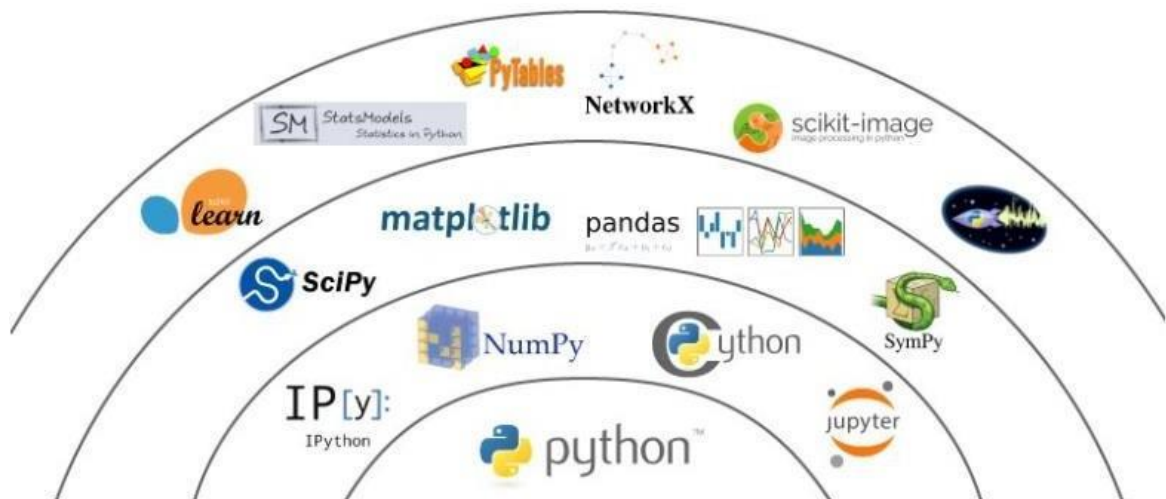


Fig 3.1- various python libraries for Machine Learning

3.3. Libraries Used :

1. Pandas: Pandas is a widely-used data analysis and manipulation library for python. It provides a lot of functions and methods that expedite the data analysis and preprocessing steps. IT also provides fast, flexible and expressive data structures working with relational or labeled or both easy and intuitive. Considered as fundamental high-level building block in performing practical, real-world data analysis in python. Has powerful tools like DataFrame and Series for analyzing.

2. Numpy :Numpy stands for Numerical Python, is a library consisting of multidimensional array objects and a collection of countless of routines for processing those arrays. Using this mathematical and logical operations on arrays can be performed. The difference in using Numpy from pandas is, it works on numerical data whereas pandas on tabular data.

3. Sklearn :sklearn stands for Scikit-learn, a machine learning library. It is imported for various classification, regression and clustering algorithms including k-means, random forest, support vector machines, gradient boosting and DBSCAN. It is designed using libraries Numpy and Scipy.

From the sklearn library and from the tree inside the library DecisionTreeClassifier. It is a class capable of performing multi-class classifier on a dataset. When compared with other classifiers, DecisionTreeClassifier takes input as two arrays: an array X, sparse or dense, of shape(n_samples, n_features) holding training samples and an array Y of integer values, shape (n_samples), holding class labels for training sample.

By default, sklearn train_test_split will make random partitions for two subsets.

We can also specify a random state for the operation. First, we need to split the dataset and then allocate the size for train and test. For this mammographic dataset we need train size as 80% (0.80) and test size as 20% (0.20) with the random state of 100.

4. Seaborn : Seaborn is a library built on top of matplotlib. It used for data visualization and exploratory data analysis. They work easily with dataframes and pandaslibrary. The graphs created can also be customized easily. It provides default styles and color palettes to make statistical plots more attractive. Also closely integrated to the data structures from pandas

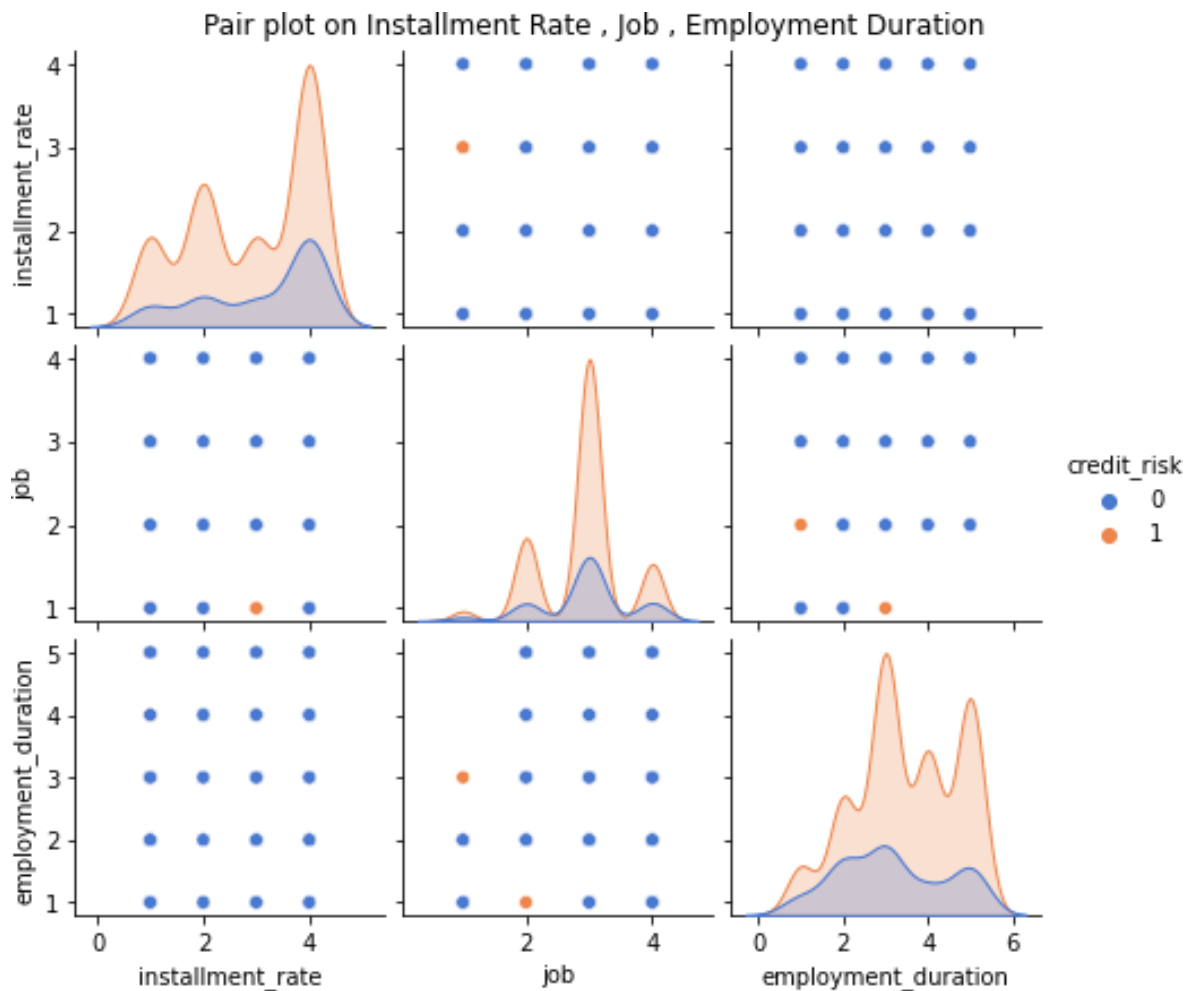


Fig 3.2 Example implementation of Seaborn library

5. Matplotlib.pyplot : Matplotlib.pyplot is a state-based interface to matplotlib. It provides a MATLAB-like way of plotting. It make changes to figures.

6. Regular expression: "re" stands for regular expression, which is a sequence of characters that forms a search pattern. It is commonly used in text processing and allows you to search and manipulate text based on patterns.

In the context of the code provided, the "re" library is being used to remove certain patterns from the text, such as removing digits and punctuation.

7. String: The string library provides a set of constants and functions related to string manipulation. The library contains various methods to manipulate and process strings, such as converting strings to lowercase or uppercase, stripping whitespace, finding substrings, replacing substrings, splitting strings, and more. Some of the commonly used functions in the string library include `ascii_letters`, `digits`, `punctuation`, `join()`, `replace()`, `lower()`, `upper()`, and `strip()`.

8.Nltk: NLTK stands for Natural Language Toolkit, and it is a popular open-source library used for natural language processing (NLP) tasks such as tokenization, stemming, lemmatization, part-of-speech tagging, and sentiment analysis. It provides a set of tools and resources to work with human language data, making it easier for developers and researchers to perform various NLP tasks efficiently. NLTK also includes a wide variety of corpora, grammars, and lexicons for different languages that can be used to train and test NLP models.

9.Word tokenisation: Word tokenization is the process of splitting a sentence or text into its constituent words or tokens. It is an essential step in natural language processing and allows for further analysis of text at a granular level. The NLTK library provides various tokenization tools, including `word_tokenize()`, which splits a sentence into words based on whitespace and punctuation.

8.SnowballStemmer: SnowballStemmer is a stemming algorithm that is used to extract the root word from a given word by following a set of rules. It is based on the Porter stemming algorithm and supports multiple languages. It works by iteratively removing suffixes from a word until it reaches the root form of the word. This is useful for text processing tasks such as natural language processing, information retrieval, and search engines.

9.Tfidf Vectorizer: TfidfVectorizer is a text feature extraction method that converts a collection of raw documents into a matrix of TF-IDF (Term Frequency-Inverse Document Frequency) features. The goal of the TF-IDF algorithm is to weight the importance of each word in a document based on its frequency of occurrence in the corpus (the collection of documents). This helps to identify important words that are unique to a particular document and can be used to classify or cluster similar documents. TfidfVectorizer is commonly used in natural language processing and

text mining tasks such as sentiment analysis, topic modeling, and document classification.

10. Cosine similarity:

Cosine similarity is a measure of similarity between two non-zero vectors of an inner product space. In natural language processing, it is commonly used to compute the similarity between two text documents represented as vectorized word frequency counts or TF-IDF values. It measures the cosine of the angle between the two vectors, where a value of 1 indicates that the vectors are identical, and a value of 0 indicates that the vectors are orthogonal

3.4 Implimentation of algorithm

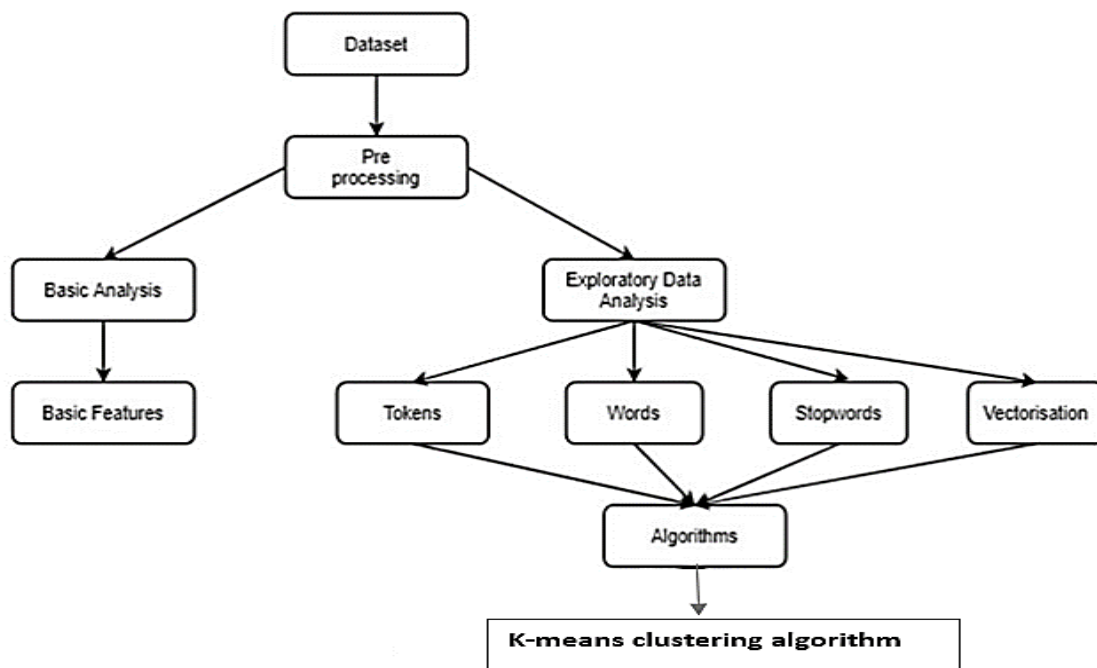


Fig.3.3 Architechture Diagram

Import the packages and classes you need.

Provide data to work with and eventually do appropriate transformations.

Create a regressor model and fit it with existing data.

Check the results of model fitting to know whether the model is satisfactory.

Apply the model for predictions.

```
In [1]: import pandas as pd
import numpy as np
```

```
In [2]: import re
import string
import nltk
```

```
In [3]: from nltk.tokenize import word_tokenize
word_tokenize
```

```
In [4]: from nltk.stem import SnowballStemmer
SnowballStemmer
```

Out[4]: nltk.stem.snowball.SnowballStemmer

```
In [5]: from sklearn.feature_extraction.text import TfidfVectorizer
TfidfVectorizer
```

Out[5]: sklearn.feature_extraction.text.TfidfVectorizer

```
In [6]: from sklearn.cluster import KMeans
KMeans
```

Out[6]: sklearn.cluster._kmeans.KMeans

```
In [7]: from sklearn.metrics.pairwise import cosine_similarity
cosine_similarity
```

Out[7]: <function sklearn.metrics.pairwise.cosine_similarity(X, Y=None, dense_output=True)>

```
In [8]: import matplotlib.pyplot as plt
```

```
In [9]: df = pd.read_csv("questions.csv")
```

Fig 3.4- Imported libraries in jupyter notebook

```
In [18]: from sklearn.model_selection import train_test_split
# Split the dataset into train and test sets
train_df, test_df = train_test_split(df, test_size=0.2, random_state=42)

# Feature extraction
tfidf_vectorizer = TfidfVectorizer()
tfidf_matrix_train = tfidf_vectorizer.fit_transform(train_df['cleaned_text'])
tfidf_matrix_test = tfidf_vectorizer.transform(test_df['cleaned_text'])
tfidf_matrix_test
```

```
Out[18]: <1000x5389 sparse matrix of type '<class 'numpy.float64'>'
         with 4430 stored elements in Compressed Sparse Row format>
```

```
In [19]: # Feature extraction
tfidf_vectorizer = TfidfVectorizer()
tfidf_matrix_train = tfidf_vectorizer.fit_transform(train_df['cleaned_text'])
tfidf_matrix_test = tfidf_vectorizer.transform(test_df['cleaned_text'])
tfidf_matrix_test
```

```
Out[19]: <1000x5389 sparse matrix of type '<class 'numpy.float64'>'
         with 4430 stored elements in Compressed Sparse Row format>
```

```
In [20]: train_df
```

```
Out[20]:
```

	question	cleaned_text
1738	What are the best books for basic electrical e...	best book basic electr engin
4943	What is the best way to make ourselves happy?	best way make happi
2916	What is it like to own and drive a Nissan GT-R?	like drive nissan gtr
1595	Why is dividing by zero an error or undefined?	divid zero error undefin
3214	Why do young women sext older men?	young women sext older men
...	...	...
4426	I am interested in a fashion designing program...	interest fashion design programm best institut...
466	Considering the current trend of information t...	consid current trend inform technolog futur pr...
3092	Are there any universities that teach film pro...	univers teach film product peopl filmmak experi
3772	Where should I promote my blog?	promot blog
860	How do you get a micro SIM card out of an iPho...	get micro sim card iphon

3999 rows × 2 columns

```
In [21]: test_df
```

```
Out[21]:
```

	question	cleaned_text
84	What are some of the high salary income jobs i...	high salari incom job field biotechnolog
2470	Where does Mexico get its gasoline from?	mexico get gasoline
2804	What are the advantages and disadvantages of s...	advantag disadvantage studi abroad one better
4987	What is Quickbooks tech support number in Ariz...	quickbook tech support number arizona
4924	How do I find a translator?	find translat
...	...	...
4486	What can I do to stop worrying what others think?	stop worri other think
1918	How can I speed up my Internet connectionn?	speed internet connectionn
4696	What's 1 popular or rock song that's in 3/4 time?	what popular rock song that time
3360	Why does Arvind Kejriwal always comment agains...	arvind kejrwi always comment modi
3886	Should I apply concealer or foundation first?	appli conceal foundat first

1000 rows × 2 columns

Fig 3.5- Testing and Training data

CHAPTER-4

RESULTS AND PERFORMANCE ANALYSIS

The code performs duplicate question detection using KMeans clustering and cosine similarity. It groups similar questions together based on their cosine similarity score and clusters them using KMeans. The result is a set of clusters where each cluster contains similar questions. This can be used to identify and remove duplicate questions from a dataset or to group similar questions together for further analysis.

We calculate cosine similarity of the user input question with the all questions present in the data set. Cosine similarities are placed in the one of the column in the data set. Now from this coulumn of cosine similarity scores we return the tuples which are having the highest cosine similarity because it represents how much close the both questions are there according to their systax and context.

The performance of the duplicate question detection system can be evaluated using metrics cosine cosine score. These metric is used to measure the accuracy of the system in correctly identifying duplicate questions.

Number of duplicate questions can be returned by setting the score in the if condition if we set the least score then we get the more questions because even if less similarity in the question.

Duplicate questions are detected by comparing the similarity between the user input question and the questions in the training set. If the similarity score is above a threshold of 0.69, the system considers the question a duplicate and returns the most similar question(s) from the training set.

The performance of the system can be evaluated by measuring its accuracy in detecting duplicate questions. The threshold value of 0.69 for similarity score is chosen arbitrarily and may not be optimal for all datasets. It is also worth noting that the system may not be able to detect duplicate questions with high accuracy if the training set is small or if the questions are too similar or ambiguous.

CHAPTER-5

SUMMARY AND CONCLUSIONS

This code is a Python script that performs question duplicate detection using clustering and cosine similarity. It first imports the required libraries such as pandas, numpy, nltk, and sklearn. The script then reads a CSV file named "questions.csv" that contains a column of questions.

The text in the questions is preprocessed using the `clean_text()` function, which converts the text to lowercase, removes punctuation, digits, and stopwords, and applies stemming. The preprocessed text is then used to create a TF-IDF matrix for feature extraction.

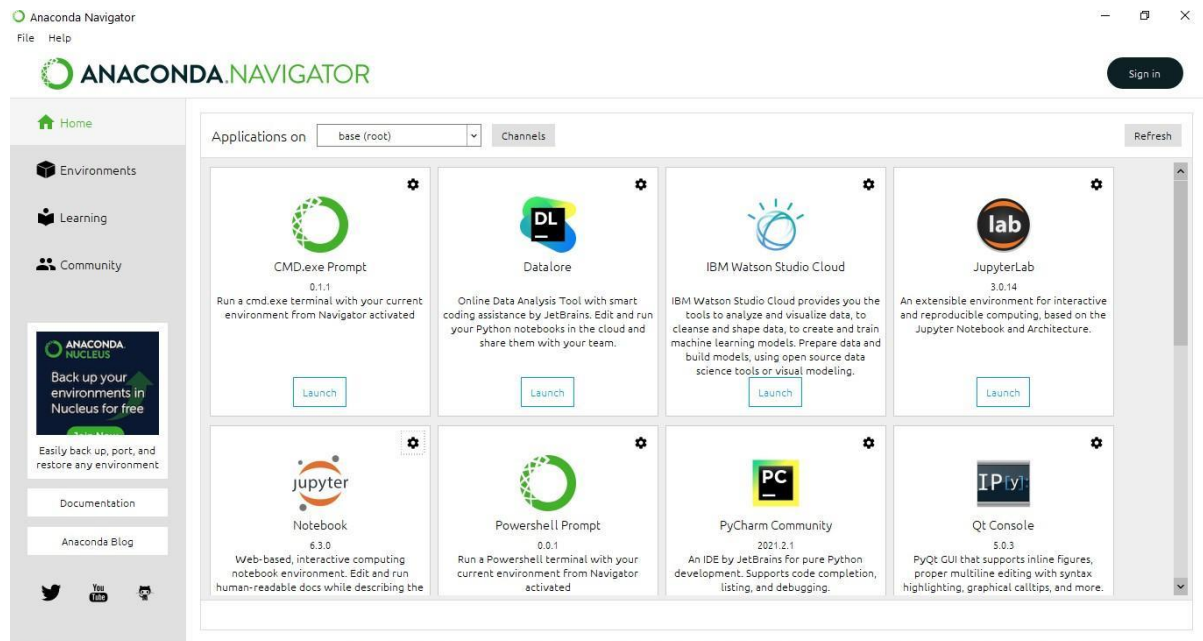
The script then splits the data into training and testing sets and performs clustering using KMeans algorithm with five clusters. The questions in the training set are then grouped into clusters and duplicates are detected by computing the pairwise cosine similarity scores between the questions in each cluster.

Finally, the script prompts the user to enter a question, preprocesses the input, computes the cosine similarity scores between the input question and the questions in the training set, and returns the most similar questions with a similarity score of at least 0.69. If no duplicates are found, the script prints a message indicating that the input question is not a duplicate.

In conclusion, this script provides a simple and effective approach for detecting duplicate questions in a dataset using clustering and cosine similarity.

WORKING ENVIRONMENT :

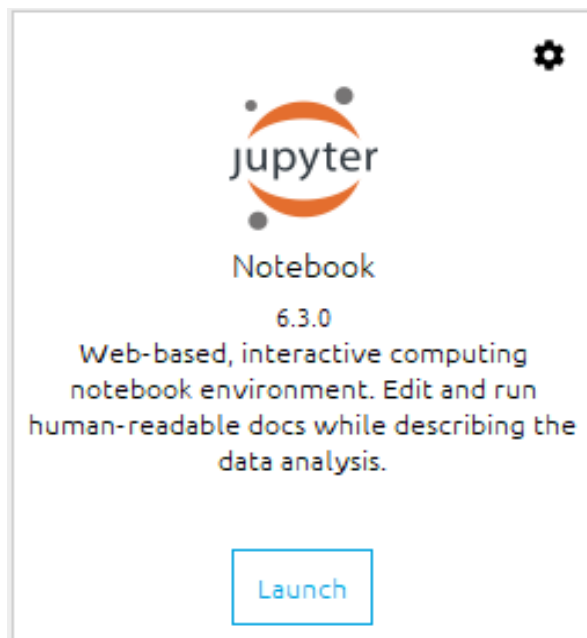
ANACONDA NAVIGATOR is desktop GUI used to launch applications and also manage packages in one place. Outlook



CODING ENVIRONMENT :

Jupyter notebook from the anaconda navigator is launched along with all the preinstalled packages for python.

APPENDIX



A.Screenshots and Outputs

```
In [1]: import pandas as pd
import numpy as np
```

```
In [2]: import re
import string
import nltk
```

```
In [3]: from nltk.tokenize import word_tokenize
word_tokenize
```

```
In [4]: from nltk.stem import SnowballStemmer
SnowballStemmer
```

```
Out[4]: nltk.stem.snowball.SnowballStemmer
```

```
In [5]: from sklearn.feature_extraction.text import TfidfVectorizer
TfidfVectorizer
```

```
Out[5]: sklearn.feature_extraction.text.TfidfVectorizer
```

```
In [6]: from sklearn.cluster import KMeans
KMeans
```

```
Out[6]: sklearn.cluster._kmeans.KMeans
```

```
In [7]: from sklearn.metrics.pairwise import cosine_similarity
cosine_similarity
```

```
Out[7]: <function sklearn.metrics.pairwise.cosine_similarity(X, Y=None, dense_output=True)>
```

```
In [8]: import matplotlib.pyplot as plt
```

```
In [9]: df = pd.read_csv("questions.csv")
```

In [10]: df

Out[10]:

	question
0	What is the step by step guide to invest in sh...
1	What is the story of Kohinoor (Koh-i-Noor) Dia...
2	How can I increase the speed of my internet co...
3	Why am I mentally very lonely? How can I solve...
4	Which one dissolve in water quikly sugar, salt...
...	...
4994	Who is currently winning the presidential elec...
4995	What has a better ROI, marketing on radio sta...
4996	Which mobile is good for 50k?
4997	Is the character Jane in the movie Predestinat...
4998	How true are near death experiences?

4999 rows × 1 columns

In [11]:

```
df.head()
```

Out[11]:

	question
0	What is the step by step guide to invest in sh...
1	What is the story of Kohinoor (Koh-i-Noor) Dia...
2	How can I increase the speed of my internet co...
3	Why am I mentally very lonely? How can I solve...
4	Which one dissolve in water quikly sugar, salt...

In [12]: df.sample()

Out[12]:

	question
3951	What percentage of people who have gender reas...

In [13]: df.tail()

Out[13]:

	question
4994	Who is currently winning the presidential elec...
4995	What has a better ROI, marketing on radio sta...
4996	Which mobile is good for 50k?
4997	Is the character Jane in the movie Predestinat...
4998	How true are near death experiences?

```
In [14]: # Data cleaning and preprocessing
def clean_text(text):

    # Convert to lowercase
    text = text.lower()

    # Remove punctuation
    text = re.sub(r'^\w\s', '', text)

    # Remove digits
    text = re.sub(r'\d+', '', text)

    # Remove stopwords
    stopwords = nltk.corpus.stopwords.words('english')
    tokens = word_tokenize(text)
    tokens = [token for token in tokens if token not in stopwords]

    # Stemming
    stemmer = SnowballStemmer('english')
    tokens = [stemmer.stem(token) for token in tokens]
    return " ".join(tokens)
```

```
In [15]: df['cleaned_text'] = df['question'].apply(clean_text)
```

```
In [16]: df['cleaned_text']
```

```
Out[16]: 0          step step guid invest share market india
1          stori kohinoor kohinoor diamond
2          increas speed internet connect use vpn
3          mental lone solv
4          one dissolv water quik sugar salt methan carbo...
...
4994         current win presidenti elect
4995         better roi market radio station facebook
4996         mobil good k
4997         charact jane movi predestin paradox time machin
4998         true near death experi
Name: cleaned_text, Length: 4999, dtype: object
```

```
In [17]: df
```

```
Out[17]:
```

	question	cleaned_text
0	What is the step by step guide to invest in sh...	step step guid invest share market india
1	What is the story of Kohinoor (Koh-i-Noor) Dia...	stori kohinoor kohinoor diamond
2	How can I increase the speed of my internet co...	increas speed internet connect use vpn
3	Why am I mentally very lonely? How can I solve...	mental lone solv
4	Which one dissolve in water quikly sugar, salt...	one dissolv water quik sugar salt methan carbo...
...	...	...
4994	Who is currently winning the presidential elec...	current win presidenti elect
4995	What has a better ROI, marketing on radio sta...	better roi market radio station facebook
4996	Which mobile is good for 50k?	mobil good k
4997	Is the character Jane in the movie Predestinat...	charact jane movi predestin paradox time machin
4998	How true are near death experiences?	true near death experi

4999 rows × 2 columns

```
In [18]: from sklearn.model_selection import train_test_split
# Split the dataset into train and test sets
train_df, test_df = train_test_split(df, test_size=0.2, random_state=42)

# Feature extraction
tfidf_vectorizer = TfidfVectorizer()
tfidf_matrix_train = tfidf_vectorizer.fit_transform(train_df['cleaned_text'])
tfidf_matrix_test = tfidf_vectorizer.transform(test_df['cleaned_text'])
tfidf_matrix_test
```

```
Out[18]: <1000x5389 sparse matrix of type '<class 'numpy.float64'>'
        with 4430 stored elements in Compressed Sparse Row format>
```

```
In [19]: # Feature extraction
tfidf_vectorizer = TfidfVectorizer()
tfidf_matrix_train = tfidf_vectorizer.fit_transform(train_df['cleaned_text'])
tfidf_matrix_test = tfidf_vectorizer.transform(test_df['cleaned_text'])
tfidf_matrix_test
```

```
Out[19]: <1000x5389 sparse matrix of type '<class 'numpy.float64'>'
        with 4430 stored elements in Compressed Sparse Row format>
```

```
In [20]: train_df
```

```
Out[20]:
```

	question	cleaned_text
1738	What are the best books for basic electrical e...	best book basic electr engin
4943	What is the best way to make ourselves happy?	best way make happi
2916	What is it like to own and drive a Nissan GT-R?	like drive nissan gtr
1595	Why is dividing by zero an error or undefined?	divid zero error undefin
3214	Why do young women sext older men?	young women sext older men
...	...	...
4426	I am interested in a fashion designing program...	interest fashion design programm best institut...
466	Considering the current trend of information t...	consid current trend inform technolog futur pr...
3092	Are there any universities that teach film pro...	univers teach film product peopl filmmak experi
3772	Where should I promote my blog?	promot blog
860	How do you get a micro SIM card out of an iPho...	get micro sim card iphon

3999 rows × 2 columns

```
In [21]: test_df
```

```
Out[21]:
```

	question	cleaned_text
84	What are some of the high salary income jobs i...	high salari incom job field biotechnolog
2470	Where does Mexico get its gasoline from?	mexico get gasolin
2804	What are the advantages and disadvantages of s...	advantag disadvantag studi abroad one better
4987	What is Quickbooks tech support number in Ariz...	quickbook tech support number arizona
4924	How do I find a translator?	find translat
...	...	...
4486	What can I do to stop worrying what others think?	stop worri other think
1918	How can I speed up my Internet connectionn?	speed internet connectionn
4696	What's 1 popular or rock song that's in 3/4 time?	what popular rock song that time
3360	Why does Arvind Kejriwal always comment agains...	arvind kejriw alway comment modi
3886	Should I apply concealer or foundation first?	appli conceal foundat first

1000 rows × 2 columns

```
In [22]: # Similarity calculation
```

```
similarity_matrix_train = cosine_similarity(tfidf_matrix_train)
similarity_matrix_train
```

```
Out[22]: array([[1.          , 0.09272997, 0.          , ..., 0.          , 0.          ,
        ],
        [0.09272997, 1.          , 0.          , ..., 0.          , 0.          ,
        ],
        [0.          , 0.          , 1.          , ..., 0.          , 0.          ,
        ],
        ...,
        [0.          , 0.          , 0.          , ..., 1.          , 0.          ,
        ],
        [0.          , 0.          , 0.          , ..., 0.          , 1.          ,
        ],
        [0.          , 0.          , 0.          , ..., 0.          , 0.          ,
        ],
        [0.          , 0.          , 0.          , ..., 0.          , 0.          ,
        ],
        [1.          , 1.          , 1.          , ..., 1.          , 1.          ,
        ]])
```

```
In [23]: # Clustering
```

```
num_clusters = 5
km = KMeans(n_clusters=num_clusters)
km.fit(tfidf_matrix_train)
clusters_train = km.labels_.tolist()
```

```
In [24]: # Duplicate detection
```

```
train_questions = train_df['question'].tolist()
clustered_questions_train = {i: [] for i in range(num_clusters)}
```

```
In [25]: for i, cluster in enumerate(clusters_train):
```

```
    clustered_questions_train[cluster].append(train_questions[i])
duplicates_train = []
```

```
In [26]: for cluster in clustered_questions_train:
```

```
    if len(clustered_questions_train[cluster]) > 1:
        group = clustered_questions_train[cluster]
        pairwise_similarity = similarity_matrix_train[np.ix_([train_questions.index(q) for q in group], [train_questions.index(q) for q in group])]
        indices = np.argmin(pairwise_similarity, axis=1)
        duplicates_train.extend([(group[i], group[j]) for i, j in zip(range(len(group)), indices) if i < indices[i]])
```

```
In [27]: # Print the duplicates in the training set
print("Duplicate questions in the training set:")
for pair in duplicates_train:
    print(pair)
```

```
Duplicate questions in the training set:
('What is it like to own and drive a Nissan GT-R?', 'Why is dividing by zero an error or undefined?')
('How can I improve my English writing skills by myself?', 'If I use a VPN while downloading torrents will I get caught?')
('What are the most annoying types of highly upvoted answers on Quora?', 'What are the trickiest questions asked in an interview?')
('Why are so many questions posted to Quora that are so easily answered by using Google?', 'How do you send a private message to someone you're following on Quora?')
('Quora: How do you post a question on Quora?', 'Why do people ask questions whose answer can be easily found on the internet?')
('How many questions have you asked on Quora?', 'Why does Quora succeed in spite of having some other Q and A websites like Yahoo Answers and Wiki Answers?')
('How do I earn from Quora?', 'What are the trickiest questions asked in an interview?')
('What is Quora's policy on expressing opinions and proselytizing to answer questions?', 'Are people just being ignorant when they ask here on Quora instead of simply typing it into say Google?')
('Why is Quora claiming my questions need improvement?', 'Why does Quora succeed in spite of having some other Q and A websites like Yahoo Answers and Wiki Answers?')
('Why is nobody answering my questions in Quora?', 'Are people just being ignorant when they ask here on Quora instead of simply typing it into say Google?')
('What was (or is) the stupidest question asked on Quora?', 'Why does Quora succeed in spite of having some other Q and A websites like Yahoo Answers and Wiki Answers?')
```

Input given to the model

```
In [28]: user_input = input("Enter a question: ")
```

Enter a question: where mahatma gandhi was born ?

```
In [29]: similarity_matrix_test = cosine_similarity(tfidf_matrix_test, tfidf_matrix_train)
test_questions = test_df['question'].tolist()
```

```
In [30]: cleaned_user_input = clean_text(user_input)
tfidf_user_input = tfidf_vectorizer.transform([cleaned_user_input])
similarity_scores = cosine_similarity(tfidf_user_input, tfidf_matrix_train)[0]
indices = np.argpartition(similarity_scores, -3)[-3:]
duplicates = []
```

Output from the model

```
In [31]: for i in range(len(indices)):
          if similarity_scores[indices[i]] >= 0.1:
              duplicate_question = train_questions[indices[i]]
              duplicates.append(duplicate_question)
          if duplicates == None:
              print("The question is not a duplicate.")
          if duplicates is not None:
              print("The question is similar to the following question in the training set:")

              for i in duplicates:
                  print(i)
```

The question is similar to the following question in the training set:

In which location did Mahatma Gandhi originate from?

From what place did Mahatma Gandhi come into the world?

Can you tell me where Mahatma Gandhi was born?

A. SOURCE CODE :

```
import pandas as pd
```

```
import numpy as np
```

```
import re
```

```
import string
```

```
import nltk
```

```
from nltk.tokenize import word_tokenize
```

```
word_tokenize
```

```
from nltk.stem import SnowballStemmer
```

```
SnowballStemmer
```

```
from sklearn.feature_extraction.text import TfidfVectorizer
```

```
TfidfVectorizer
```

```
from sklearn.cluster import KMeans
```

```
KMeans
```

```
from sklearn.metrics.pairwise import cosine_similarity
```

```
cosine_similarity
```

```

import matplotlib.pyplot as plt

df = pd.read_csv("questions.csv")

df

df.head()

df.sample()

df.tail()

# Data cleaning and preprocessing

def clean_text(text):

    # Convert to lowercase

    text = text.lower()

    # Remove punctuation

    text = re.sub(r'[^w\s]', "", text)

    # Remove digits

    text = re.sub(r'\d+', "", text)

    # Remove stopwords

    stopwords = nltk.corpus.stopwords.words('english')

    tokens = word_tokenize(text)

    tokens = [token for token in tokens if token not in stopwords]

    # Stemming

    stemmer = SnowballStemmer('english')

```

```

tokens = [stemmer.stem(token) for token in tokens]

return " ".join(tokens)

df['cleaned_text'] = df['question'].apply(clean_text)

df

from sklearn.model_selection import train_test_split

# Split the dataset into train and test sets

train_df, test_df = train_test_split(df, test_size=0.2, random_state=42)


# Feature extraction

tfidf_vectorizer = TfidfVectorizer()

tfidf_matrix_train = tfidf_vectorizer.fit_transform(train_df['cleaned_text'])

tfidf_matrix_test = tfidf_vectorizer.transform(test_df['cleaned_text'])

tfidf_matrix_test


# Similarity calculation

similarity_matrix_train = cosine_similarity(tfidf_matrix_train)

similarity_matrix_train


# Clustering

```

```

num_clusters = 5

km = KMeans(n_clusters=num_clusters)

km.fit(tfidf_matrix_train)

clusters_train = km.labels_.tolist()

import seaborn as sns

sns.scatterplot(clusters_train,clusters_train)


# Duplicate detection

train_questions = train_df['question'].tolist()

clustered_questions_train = {i: [] for i in range(num_clusters)}

for i, cluster in enumerate(clusters_train):

    clustered_questions_train[cluster].append(train_questions[i])

duplicates_train = []

for cluster in clustered_questions_train:

    if len(clustered_questions_train[cluster]) > 1:

        group = clustered_questions_train[cluster]

        pairwise_similarity = similarity_matrix_train[np.ix_([train_questions.index(q) for q in
group], [train_questions.index(q) for q in group])]

        indices = np.argmin(pairwise_similarity, axis=1)

        duplicates_train.extend([(group[i], group[j]) for i, j in zip(range(len(group)), indices) if
i < indices[i]])


# Print the duplicates in the training set

```

```

print("Duplicate questions in the training set:")

for pair in duplicates_train:

    print(pair)

user_input = input("Enter a question: ")

similarity_matrix_test = cosine_similarity(tfidf_matrix_test, tfidf_matrix_train)

test_questions = test_df['question'].tolist()


cleaned_user_input = clean_text(user_input)

tfidf_user_input = tfidf_vectorizer.transform([cleaned_user_input])

similarity_scores = cosine_similarity(tfidf_user_input, tfidf_matrix_train)[0]

indices = np.argsort(similarity_scores)[-3:]

duplicates = []

for i in range(len(indices)):

    if similarity_scores[indices[i]] >= 0.69:

        duplicate_question = train_questions[indices[i]]

        duplicates.append(duplicate_question)

if duplicates == None:

    print("The question is not a duplicate.")

if duplicates is not None:

    print("The questions is similar to the following question in the training set:")

    print(" ")

    for i in duplicates:

        print(i)

```

REFERNCES :

[1] For Learning Python Language<https://www.learnpython.org/>

[2] Scipy Lectures Notes

http://scipy-lectures.org/_downloads/ScipyLectures-simple.pdf

[3] Numpy

<https://numpy.org/doc/stable/user/quickstart.html>

[4] Hugging Face.com

Tfidf-Vectorizer

- - THE END - -