THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY
MSBD 5002 Data Mining and Knowledge Discovery
Fall 2021 Final Examination
Time: December 10th 11:59PM - Dec 13th 11:59AM (60 hours)

Exam Guidelines

- This is an open book examination.
- Exam Duration: December 10th 11:59PM to Dec 13th 11:59AM (60 hours).
- Turn In: You must submit your solutions via **canvas** before Dec 13th 11:59AM. In oder to avoid network congestion and submission failure, please submit your solutions in advance. **Any late submission will lead to zero mark directly.**
- You must pack all solutions together into one zip file, named as itsc_studentID_final.zip. The example of directory structure is shown in the Figure 1.
- **Academic integrity:**
  - Your program and report should be based on your own effort. Students cannot collaborate with anyone.
  - In case you seek help from any reference source, you should state it clearly in your report. Failure to do so is considered plagiarism which will lead to appropriate disciplinary actions.
  - **Plagiarism will lead to zero mark directly.**

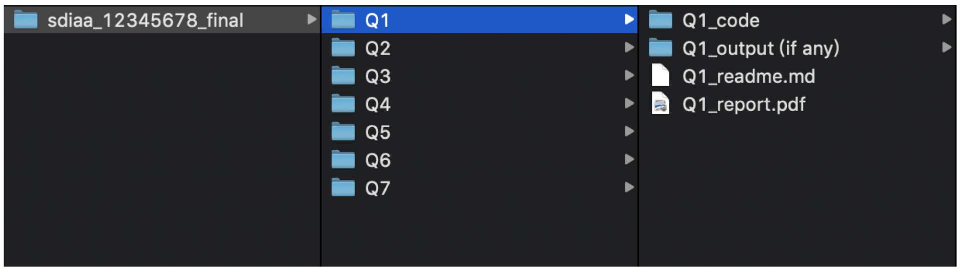

Figure 1: An example of the directory structure to be submitted.

Datasets

The data sets for Q1, Q2, Q3, Q4, Q5, Q6, and Q7 can be downloaded via the link or the canvas.

Grading

Your score will be given based on:

1. Correctness, completeness and clarity in your report.
2. Codes must be runnable, and code comments are necessary. Missing the necessary comments will be deducted a certain score.
3. Evaluation performance (if any).
4. You can get bonus if you propose a novel approach to any question.

Notes

Please read the following notes carefully.

1. Please read Exam Guidelines carefully.
2. Please work hard and arrange your time reasonably.
3. For programming language, in principle, python is preferred.
4. Computation of some questions is very large, students might use cloud computing platform, such as azure.
5. If your codes or answer refer to any blog, github, paper and so on, please provide the links in corresponding **QX_report.pdf**.
6. If you have any question, please send email to **mscbd5002spring20@gmail.com**. Questions from other channels will not be answered.

1 Community Detection (12 Points)

Community detection techniques are useful for social media algorithms to discover people with common interests and keep them tightly connected. In this problem, given a social network in a large European research institution, you are required to detect 42 communities (i.e., departments) among 1005 researchers of this institution.

1.1 Data Description

Two datasets `email-Eu-core-department-labels.txt` and `email-Eu-core.txt` are provided. Specifically, the file `email-Eu-core.txt` represents the social network generated using email data from a large European research institution. There is an edge (u, v) in the network if person u sent person v at least one email. Each line in this file records an email communication link between members of the institution:

- `source id` - id of the email sender;
- `target id` - id of the emial receiver.

The `email-Eu-core-department-labels.txt` file provides the ground-truth department membership labels. Each line in this file has the following information:

- `node id` - id of the node (a member of the institute);
- `department id` - id of the member's department (number in 0, 1, ..., 41).

1.2 Task Description

TASK 1: Based on `email-Eu-core.txt`, detect 42 communities among 1005 members of the institution. Please report **normalized mutual information (NMI)** of your clustering results.

TASK 2: Many real-world graphs are very large and complex, and it can be computationally infeasible to detect communities in such large-scale graphs. Please discuss the **time complexity** of your community detection algorithm in detail and list the possible directions for improving the efficiency of your algorithm.

TASK 3: Sampling algorithms are widely used in large-scale graph data analysis. Please check the related work on graph sampling and describe a method that can sample a representative subgraph from a large graph (coding is not required).

1.3 Submission

1. Write down your algorithm's principles and details in **Q1_report.pdf**. If your code refers to blogs, GitHub codes, papers or anything else, please cite them clearly.
2. Put your community detection results in **Q1_communities.csv**. Each line should be in the format of [node id, department id].
3. Pack all your codes for task 1 and task 2 in **Q1_code** folder.
4. Please state clearly how to run your program in **Q1_readme.md**.

1.4 Notes

1. NMI is a common evaluation metric for community detection. Please refer to the corresponding sklearn package or other resources if you are not familiar with NMI.

2. Please use appropriate notations such as the number of edges $|E|$ and the number of nodes $|V|$ when describing the time complexity of your algorithm.

2 COVID-19 Prediction (12 Points)

As the COVID-19 outbreak evolves, accurate forecasting continues to play an extremely important role in informing policy decisions. In this problem, you are required to analyze historical COVID-19 information and develop prediction models to forecast the the number of confirmed cases and deaths around the world.

2.1 Data Description

The two datasets `covid19_confirmed_global.txt` and `covid19_deaths_global.txt` record the number of confirmed cases and deaths from 22 Jan 2020 to 29 Nov 2021 in different regions. Each sample in the above datasets has the following information:

- `province/state`;
- `country/region`;
- The `latitude` and `longitude` of the region;
- The `cumulative number of confirmed cases or deaths` from 22 Jan 2020 to a specific day in the region.

2.2 Task Description

TASK 1: Based on the provided datasets, develop prediction models to forecast the number of **daily** confirmed new cases **and** deaths in **the US from 30 Nov 2021 to 6 Dec 2021**. Compare your predictions with the real number of daily confirmed cases and deaths provided by the Johns Hopkins University Center for Systems Science and Engineering (JHU CSSE).

TASK 2: Based on the `covid19_confirmed_global.txt` dataset, plot a figure to visualize the total number of **confirmed cases** around the world in **Oct 2021**. Please refer to Figure 2 as an example. Use **different colors or radius sizes** to distinguish the number of confirmed cases in different regions.

2.3 Submission

1. Write down your algorithm's principles and details in **Q2_report.pdf**. If your code refers to blogs, GitHub codes, papers or anything else, please cite them clearly.
2. Put your prediction results and heatmap in **Q2_report.pdf**.
3. Pack all your codes for task 1 and task 2 in **Q2_code** folder.
4. Please state clearly how to run your program in **Q2_readme.md**.

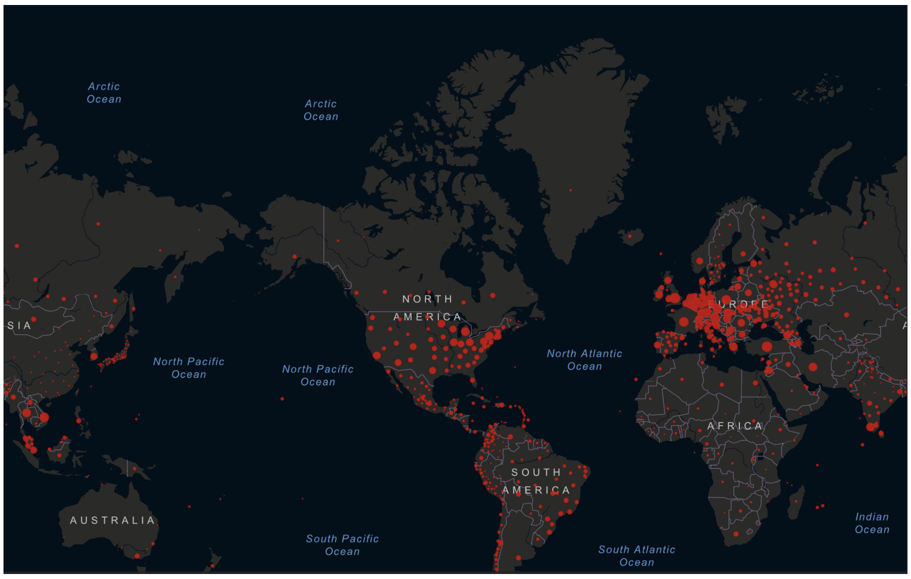

Figure 2: An example heat map visualizing the number of confirmed cases.

2.4 Notes

1. You **CAN** use any prediction algorithm that you know. But you are **NOT** allowed to directly copy the publicly available models.
2. For task 1, you are required to predict the number of newly confirmed cases and deaths **in each day** instead of the total number of confirmed cases and deaths in the seven consecutive days.
3. Be careful about missing data values.

3 Handwritten Digit Classification (12 Points)

3.1 Data Description

This dataset contains 9269 images about the handwritten digits. Each row is the data for one instance. The first 784 columns in a row correspond to the 784 features of an instance (a 28*28 image), and the last column is the lable, ranging from 0 to 9.

3.2 Task Description

Task 1: Please describe the dataset from the following aspects: (1) the distribution of different classes; (2) the difference between the images in the dataset and the original MNIST dataset.

Task 2: Please provide a classifier based on the given data, and we will test its performance on another test dataset. Please make your classifier store its predicted results in a file "Q3_predicted_results.csv", where each row contains one column for the output of an instance. Your score for this task will depend on your description of your method and the performance (the mean per-class accuracy) of your classifier. An example "Q3_predicted_results.csv" for a test dataset of two images is provided via the link and it is also on canvas.

3.3 Submission

1. Please write down your answer to Task 1 and your algorithm's principles and details (so that we know how and why your method is designed in that way) in **Q3_report.pdf**. If your code refers to blogs, GitHub codes, papers or anything else, please cite them clearly.

2. Please put all your codes in **Q3_code** folder.

3. Please state clearly how to run your program in **Q3_readme.md**.

3.4 Notes

You can use any algorithm that you know. You are **NOT** allowed to directly use any complete models (like a tool that directly solves this problem) which are publicly available without any detailed process.

4 Fraud Transaction Detection (12 Points)

4.1 Data Description

This dataset contains 1296676 transaction records. Some records are marked as fraud. Each record has 23 features. The detailed description about each feature can be found in “feature_description.txt” via this link or on canvas.

4.2 Task Description

Please provide a model to detect fraud transaction records based on the given dataset. We will measure the performance of your model using AUC (Area Under the ROC Curve) on another dataset. Please make your model store its predicted results in a file “Q4_predicted_results.csv”, where each row contains one column for the output of an instance. An example “Q4_predicted_results.csv” for a test dataset of two records is provided via the link and it is also on canvas.

You should also provide a report describing your algorithm’s principles and details, i.e., telling us how and why your method is designed in that way. Your score for this task depends on the model performance and your report.

4.3 Submission

1. Please name your report as **Q4_report.pdf**. If your code refers to blogs, GitHub codes, papers or anything else, please cite them clearly.

2. Please put all your codes in **Q4_code.pdf** folder.

3. Please state clearly how to run your program in **Q4_readme.md**.

4.4 Notes

You can use any algorithm that you know. You are **NOT** allowed to directly use any complete models (like a tool that directly solves this detection problem) which are publicly available without any detailed process.

5 Recommendation Systems (14 Points)

You have learned some basic models including user-based and item-based collaborative filtering methods in class. However, some features of items or users can also help to improve the performance of recommendation systems. In this problem, you are given a movie rating dataset which contains basic rating information, movie titles, movie genres and user information. You should try to figure out how to utilize these features to construct a recommendation system.

5.1 Data Description

You are provided with 4 datasets: `movies.csv`, `rating_train.csv`, `rating_test.csv`, and `users.csv`. Specifically, `movies.csv` contains the titles and genres of 3,883 movies; `rating_train.csv` includes user ratings to different movies; `rating_test.csv` is the test set containing movies that you are required to predict ratings for; `users.csv` introduces the basic information of 6,040 users. Please see `DATA_DESCRIPTION.txt` for more detailed description.

5.2 Task Description

TASK 1: Based on rating_train.csv and other relevant data in this question, build a recommendation system to predict user ratings for movies in rating_test.csv.

TASK 2: Compare your method with the collaborative filtering algorithm and report the advantage of your method.

5.3 Submission

1. Write down your algorithm's principles and details in **Q5_report.pdf**. If your code refers to blogs, GitHub codes, papers or anything else, please cite them clearly.

2. Your prediction result should named as **Q5_output.csv**. Each line represents the user's rating of the movie, which means your final output should contain 3 columns: 'UserID', 'MovieID', and 'Rating'.

3. Pack all your codes in **Q5_code** folder.

4. Please state clearly how to run your program in **Q5_readme.md**.

5.4 Notes

There will be some bonus score if you use some creative or the state-of-the-art models.

6 Data Augmentation in Natural Language Processing (14 Points)

Adequate training data can be a favorable precondition for training machine learning models. But in real-world problems, the data that can be used to train the model is often not enough. There are many tasks in natural language processing (NLP) from text classification to question answering, but whatever the task may be the amount of data available to train the model impacts the model performance significantly. The simplest option may be to obtain more data, yet acquiring and labeling additional observations can result in an expensive and time-consuming process that one wishes to avoid.

6.1 Problem 1

Suppose you are doing a text classification task in NLP. Unfortunately, your training dataset is insufficient. Please explain how you will expand the data via data augmentation before training. Please answer in details for at least **three** NLP data augmentation methods, and give at least **one** specific example to illustrate for each method.
Reference to other materials to answer this question is allowed, if you do so, please also list your references.

6.2 Problem 2

The ubiquitousness of smartphones enables people to make announcements of all kinds of emergencies that are observed in real-time, such as natural disasters, sudden social incidents, and outbursts of crisis. During times of such emergencies, social media platforms like Twitter have become an important communication channel. Consequently, monitoring such social media like Twitter has drawn more and more attention.
However, it can be ambiguous whether a tweet are indeed announcing a disaster. Take this example: "On plus side LOOK AT THE SKY LAST NIGHT IT WAS ABLAZE.", where the word "ABLAZE" is used explicitly but metaphorically. This is rather challenging to a machine program.
In this problem, you are required to code a machine learning model that predicts whether a Tweet is about real disasters or not.

6.2.1 Data Description

The dataset consists of 10,000 tweets that are hand classified. There are three files: `train.csv` - the training set; `test.csv` - the test set; and `sample_submission.csv` - a sample prediction result file in the correct format. Each sample in the train and test set has the following information:

- The `text` of a tweet;
- A `keyword` from that tweet (although this may be blank!);
- The `location` the tweet was sent from (may also be blank).

The column information are listed in below:

- `id` - a unique identifier for each tweet;
- `text` - the text of the tweet;
- `location` - the location the tweet was sent from (may be blank);
- `keyword` - a particular keyword from the tweet (may be blank);
- `target` - in train.csv only, this denotes whether a tweet is about a real disaster "1" or not "0".

6.2.2 Task Description

TASK 1: Check the target class distribution for the two classes 0 and 1. You should draw a histogram to show the target class distribution, then conclude if there is an imbalanced class and which one is the imbalanced class.

TASK 2: Generate some more training data for the less class, for example, 300 more samples, using one data augmentation method. Explain in details which data augmentation method you choose and how you use this method.

TASK 3: Perform text representation using any kind of methods to vectorize and represent the collection of texts. You may need to build the word index dictionary and the embedding dictionary, then to create an embedding matrix for the text classification model to learn the word representations.

TASK 4: You are predicting whether a given tweet is about a real disaster or not. If so, predict a `1`. If not, predict a `0`. Build an text classification model for this task. You are required to summarize your model structure as shown in below Figure 3:

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 embedding (Embedding)       (None, 50, 100)           1869500   
                                                                 
 spatial_dropout1d (SpatialD  (None, 50, 100)          0         
 ropout1D)                                                       
                                                                 
 lstm (LSTM)                 (None, 100)               80400     
                                                                 
 dense (Dense)               (None, 1)                 101       
                                                                 
=================================================================
Total params: 1,950,001
Trainable params: 80,501
Non-trainable params: 1,869,500
```

Figure 3: An example text classification model summary.

TASK 5: To examine how your data augmentation method works, use the augmented text data to train the model. Then, compare the performance with which trained on data without augmentation in ROC AUC score. With data augmentation, you should get a boost in the model performance (ROC AUC score).

6.3 Submission

1. For **Problem 1**, you should write up your answers in detail.

2. For **Problem 2**, pack up all code files in folder **Q6_code**.

3. You should write the pseudo code of your data augmentation algorithms in the report **Q6_report.pdf** and describe them in detail.

4. You should write the experiment settings in your report **Q6_report.pdf**. For example, if you split the dataset, you should write it clearly.

5. You should include the histogram of the target class distribution in your report **Q6_report.pdf** and draw a conclusion on the imbalanced class.

6. You should store your generated data samples after data augmentation separately in **Q6_generated.csv**.

7. You should store your corresponding classification results on augmented data in the file **Q6_results_augmented.csv** and on data without augmentation in the file **Q6_results_plain.csv**, referring to `sample_submission.csv`.

8. You should summarize your model structure in **Q6_report.pdf**.

9. You should compare the performance and provide an analysis in **Q6_report.pdf**.

10. Please state clearly how to run your program in **Q6_readme.md**.

6.4 Notes

1. For **Problem 1**, you do **NOT** need to code in the question. But you need to answer **in detail**.

2. For **Problem 2**, you are required to code for your data augmentation method and NLP model manually, but you are recommended to use any existing package and libraries of the classification model to test the effectiveness of your algorithms. For example, you feed the augmented data by different NLP data augmentation methods to the same NLP text classification model and compare the performance, to choose the best one.

3. Libraries such as `Keras`, `Scikit-Learn`, `Gensim`, and `nlpaug` are recommended.

4. You can use `GloVe` pre-trained corpus model to represent the words. You can use the pre-trained `GloVe` vectors which are trained on large corpora - an unsupervised learning algorithm developed by Stanford, which is provided in the zipped file. It is available in 3 varieties: 50D, 100D, and 200D. You can try 100D here. Each line in the file contains a word and its corresponding n-dimensional vector representation. You may need to prepare an embedding dictionary containing every word in the `GloVe` pre-trained vectors and their corresponding vector.

5. Do **NOT** validate using the augmented data. Before data augmentation, you should split the data into the train and validation set so that no samples in the validation set have been used for data augmentation.

6. If you are doing K-fold cross-validation, always keep the original sample and augmented sample in the same fold to avoid overfitting.

7. You are recommended to always try different augmentation approaches and check which one works better.

8. A mix of different augmentation methods is also appreciated but do not overdo it.

9. Experiment to determine the optimal number of samples to be augmented to get the best results.

10. If your pseudo code includes model designs, you can use a function to denote it directly instead of writing it in detail.

11. How much performance gain you want to achieve with your data augmentation method depends on yourself.

12. Keep in mind that data augmentation in NLP does not always help to improve model performance.

7 Sentiment Analysis and Opinion Mining (24 Points)

Sentiment analysis, also known as bias analysis or opinion mining, is a process of analyzing, processing, inference and reasoning subjective texts with emotional colors. With the ability of sentiment analysis, the positive and negative sentiment tendencies of certain natural language texts with subjective descriptions can be automatically determined. The corresponding results can illustrate the attitude of a speaker, writer, or other subject with respect to some topic or the overall contextual polarity or emotional reaction to a document, interaction, or event. Such attitude may be a judgement or evaluation (see appraisal theory), affective state (that is to say, the emotional state of the author or speaker), or the intended emotional communication (that is to say, the emotional effect intended by the author or interlocutor). Sentiment analysis has been a topic of great interests because it has various practical applications. For instance, user's feeds can be personalized based on their opinions on a variety of topics. Similarly, marketing agencies can use sentiment analysis to research the public's opinion of their company or grasp customer opinions on products. With the growing amount of publicly available information on the Internet, there is a vast amount of texts available to express opinions on review sites, forums, blogs, social media, and so on. Naturally, online social platforms like Twitter has been a popular place to for people to express their opinions about various topics, products, and events. In this question, you should explore different techniques for analyzing sentiments expressed in Twitter data. You will utilize standard data processing techniques on Twitter data to develop and train various learning models, ultimately reaching a conclusion about the most effective way to analyze sentiments.

7.1 Data Description

In order for a model to have decent performance, one needs a relatively good size of dataset to train. The dataset chosen here is the "`Sentiment140`", which originated from Stanford University. It contains 1,600,000 tweets extracted using the twitter api. The tweets have been annotated (0 = negative, 4 = positive) and they can be used to detect sentiments . It contains the following 6 fields:

- `target` - the polarity of the tweet (0 = negative, 2 = neutral, 4 = positive);
- `id` - the id of the tweet (2087);
- `date` - the date of the tweet (Sat May 16 23:58:44 UTC 2009);
- `flag` - the query (lyx). If there is no query, then this value is `NO_QUERY`;
- `user` - the user that tweeted (robotickilldozr);
- `text` - the text of the tweet (Lyx is cool).

7.2 Task Description

TASK 1: Check the target class distribution for the three classes 0, 2, and 4. You should draw a histogram to show the target class distribution, then conclude your observation.

Task 2: Start by dropping the columns in the data that are not helpful for sentiment analysis. List the columns you choose to drop and explain the reason why.

Task 3: Examine the length of the string in the `text` column. You should plot the length distribution of the tweets with box plot, to see the overall distribution of length of strings in each entry. Conclude that if they satisfy twitter's character limit 140.

Task 4: Start text data processing. Note that HTML encoding is not converted to text correctly, resulting in text field as "`&amp`", "`&quot`", and so on. The first step is to decode HTML in the tweets to general text.

Task 5: The second step of text data processing is to remove 1) the `@mention`, 2) the URL links, 3) the stop-words, and 4) other non letter characters including the hashtag "`#`", the punctuation marks, and the numbers. Though they may carry certain information such as the users mentioned, such information does not help in this task of sentiment analysis.

Task 6: The third step is to convert all text to lowercase.

Task 7: Deal with tokenization and stemming/lemmatization. Create the embedding matrix with either count vectorizer or Tfidf vectorizer. Save your cleaned data after the data processing as csv file.

Task 8: To provide a general idea of what kind of words are frequent in the corpus, perform text visualisation via word cloud for each class, respectively. Conclude on your observations from the generated word clouds with at least **two** example words and analyze their reasons to be of certain size in each cloud in detail. An example word cloud is shown in below Figure 4.

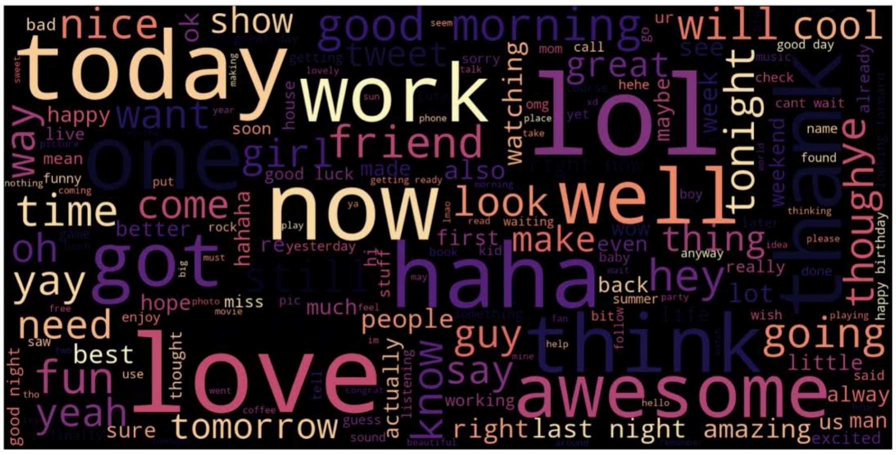

Figure 4: An example word cloud of positive words.

Task 9: Extract the term frequency data - what words are used in the tweets, and how many times they are used in the entire corpus. Note that if you transform the data with the fitted count vectorizer, you can directly get the term frequency from the sparse matrix. Save the extracted term frequency as csv file, exemplifying in below Table 1. The indexes are the token from the tweets dataset, and the numbers in the "negative" and "positive" columns represent how many times the token appears in negative tweets and positive tweets, respectively.

Table 1: An example of term frequency data.

	negative	**positive**	**total**
to	313,160	252,566	565,726
the	257,836	265,998	523,834
my	190,774	125,955	316,729
it	157,448	147,786	305,234

TASK 10: Perform tweet token visualisation to show how different the tokens in two different classes (positive, negative). Plot the top 50 words in negative tweets using bar chart and analyze the results in detail.

TASK 11: Given a tweet from a user, you are predicting the sentiment expressed in the tweet. Build at least **two** classical machine learning methods (i.e., Logistic Regression, Naive Bayes, Support Vector Machines, etc.) and at one **two** deep learning models (i.e., Recurrent Neural Network (RNN), Long Short Term Memory (LSTM) Network, etc.) for this task. You are required to summarize your method parameter and model structure as shown in Figure 3.

TASK 12: Report and summarize your method performance in a table and analyze the findings in detail. You should compare the performance of different models with the model accuracy score. An example is show in below Table 2.

Table 2: An example of model performance.

Method	**Accuracy(%)**
Logistic Regression	71
Naive Bayes	65
SVM	57
LSTM	86

7.3 Submission

1. Pack up all code files in folder **Q7_code**.
2. You should write the pseudo code of your data processing pipeline and your models in the report **Q7_report.pdf** and describe them in detail.
3. You should write the experiment settings in your report **Q7_report.pdf**. For example, if you split the dataset, you should write it clearly.
4. You should include the histogram of the target class distribution in your report **Q7_report.pdf** and draw a conclusion.
5. You should include the box plot of the tweet text length in your report **Q7_report.pdf** and draw a conclusion on if they satisfy twitter's character limit 140.
6. You should store your cleaned data after data processing separately in **Q7_clean.csv**.
7. You should include the word clouds of each class, receptively, in your report **Q7_report.pdf** and conclude on your observations with at least **two** example words and analyze their reasons to be of certain size in each cloud in detail.
8. You should store the extracted term frequency data separately in **Q7_freq.csv**.

9. You should include the bar chart of the top 50 negative tweets in your report **Q7_report.pdf** and analyze the results in detail.

10. You should summarize your method parameter and model structure in **Q7_report.pdf**.

11. You should summarize and compare the performance of different models and provide an detailed analysis in **Q7_report.pdf**.

12. Please state clearly how to run your program in **Q7_readme.md**.

7.4 Notes

1. You are required to code for your data processing and your sentiment analysis models manually, but you are recommended to use any existing package and libraries of the classification model to test the effectiveness of your algorithms. For example, you feed the clean data after various stages of data processing methods to the different sentiment analysis model and compare their performance, to choose the best one.

2. Libraries such as `Keras`, `Scikit-Learn`, `nltk`, `WordCloud`, `BeautifulSoup`, `Gensim` are recommended.

3. You can always try more text data processing approaches and check which one works better.

4. If your pseudo code includes model designs, you can use a function to denote it directly instead of writing it in detail.