Reflection report

Database Creation

1. Business rules [1]

The data has seventeen columns, and the content is articles sent on reddit. The user table needs to have the name of the user, the number of posts made by the user, the registration time and the user upvote ratio, all of which are also relevant to the user. Similarly, it can be divided into those related to sub-sections and those related to posts. With the business rules extracted, the data can be simply divided into three sections. This is shown in the diagram below.

user_info table	**subreddit_info table**	**post_info table**
user_id	subreddit_id	post_id
user_name	subreddit_name	post_title
user_num_post	subreddit_created_at	post_author_id
user_registered_at	subreddit_description	post_subreddit_id
user_upvote_ratio	subreddit_faved_by	posted_at
	subreddit_numb_members	post_num_comments
	subreddit_numb_posts	post_score
		post_total_awards_recived
		post_upvote_ratio
		selftext

2. Data integrity [2]

In data integrity aspect,

1) Set user_register_at as date format and post_at to datetime format because of its precision to the exact time and then set the other data to its corresponding data type for the domain constraint. Since selftext is a long string, the longtext data type is used here to store it. For fractional numbers, such as post_upvote_ratio, a float is used. Here are the SQL statements used to set the data type:

```
user_registered_at date,
posted_at datetime,
post_upvote_ratio float,
selftext longtext,
```

2) When creating tables like user_info, subreddit_info, post_info and favorite_user_info, the unique values, user_id, subreddit_id, post_id and reddit_id are created as primary keys, which represents a uniqueness constraint as well as a non-null constraint. This achieves the entity integrity constraint. Take the user_info table as an example.

```
user_id int unsigned not null,
primary key (user_id)
```

3) Because every post has an author and a subreddit, implement the business rule via foreign key and achieve the referential integrity constraint by set post_author_id and post_subreddit_id in the post_info table as foreign keys and reference to the primary keys user_id and subreddit_id in the user_info and subreddit_info tables respectively. This way, in

future table modifications, only the data need to be modified and not the database structure. Modifying data is not only less laborious but also less technically demanding.

```
constraint `fk_author` foreign key (post_author_id) references user_info(user_id)
on delete restrict on update cascade,

constraint `fk_subreddit` foreign key (post_subreddit_id) references
subreddit_info(subreddit_id) on delete restrict on update cascade
```

By implementing data integrity constraints, it helps to ensure that data stored in the database can be found and linked to other data. This ensures that the entire data set can be recovered and searched when needed. It enhances data stability, provides optimal performance and makes it reusable and easy to maintain.

3. Normalization

In terms of database normalisation, in order to remove duplicate data and increase data consistency. Three normalisations were implemented.[3]

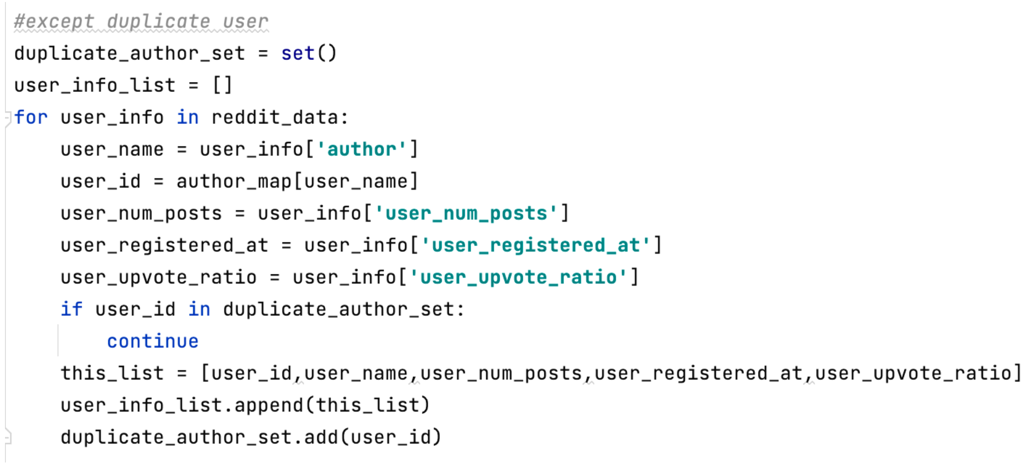
1NF: Remove multivalued attribute, i.e., remove duplicate columns, because there are no duplicate semantic columns in the data, each cell contains a single value, then 1NF end.

2NF: That is, remove the partial dependencies and make each non-primary key attribute completely dependent on the entire primary key function. To achieve this, the 17 columns of data are divided into three tables, the user_info table related to user, the post_info table related to post, and the subreddit_info table related to subreddit.

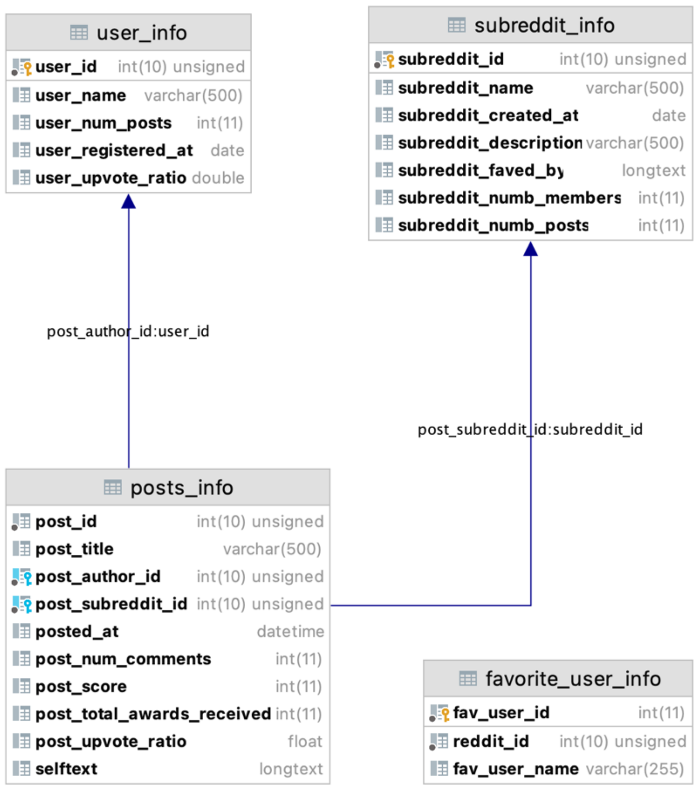
3NF: The purpose of the 3NF is to remove transitive dependencies. It was found that the sub_fav_users in subreddit_info table are only related to subreddit_name, so a new table was created with fav_user_id, subreddit_id and user_name, each corresponding to the id of their favourite subreddit.

4. Reverse engineering and bad data management [4]

In the creation of the pipeline, take the example of creating a user table, setting up a set to store users that have already appeared, so that duplicate storage can be avoided, which is a reflection of the management of duplicate data. Considering the bad data handling, this experiment simply transcoded all text to utf8 and import emoji for better display. The table contains some emoji and some special characters which should be filtered out or replaced by corresponding text using a filter to achieve clean data. This is an area that needs to be improved in the future.



```python
#except duplicate user
duplicate_author_set = set()
user_info_list = []
for user_info in reddit_data:
    user_name = user_info['author']
    user_id = author_map[user_name]
    user_num_posts = user_info['user_num_posts']
    user_registered_at = user_info['user_registered_at']
    user_upvote_ratio = user_info['user_upvote_ratio']
    if user_id in duplicate_author_set:
        continue
    this_list = [user_id,user_name,user_num_posts,user_registered_at,user_upvote_ratio]
    user_info_list.append(this_list)
    duplicate_author_set.add(user_id)
```


user_info
user_id int(10) unsigned
user_name varchar(500)
user_num_posts int(11)
user_registered_at date
user_upvote_ratio double
subreddit_info
subreddit_id int(10) unsigned
subreddit_name varchar(500)
subreddit_created_at date
subreddit_description varchar(500)
subreddit_faved_by longtext
subreddit_numb_members int(11)
subreddit_numb_posts int(11)
post_author_id:user_id
post_subreddit_id:subreddit_id
posts_info
post_id int(10) unsigned
post_title varchar(500)
post_author_id int(10) unsigned
post_subreddit_id int(10) unsigned
posted_at datetime
post_num_comments int(11)
post_score int(11)
post_total_awards_received int(11)
post_upvote_ratio float
selftext longtext
favorite_user_info
fav_user_id int(11)
reddit_id int(10) unsigned
fav_user_name varchar(255)

Reverse engineering the entire database using DataGrip results in a diagram like this, which is what the entire database will look like once it has been built.

By reverse engineering [4] the relationships between data tables can be clearly understood. It allows to visualise and clearly understand entities and their attributes, primary and foreign keys in tables, and the relationships between tables. For example, the author in post_info is related to the user in user_info and the subreddit in post_info is related to the subreddit in subreddit_info. Also, the base ratio of the links can be clearly shown, for example the relationship between post_info and user_info, and subreddit_info is a one-to-many relationship.

5. Example of partial code explanation to create a user table

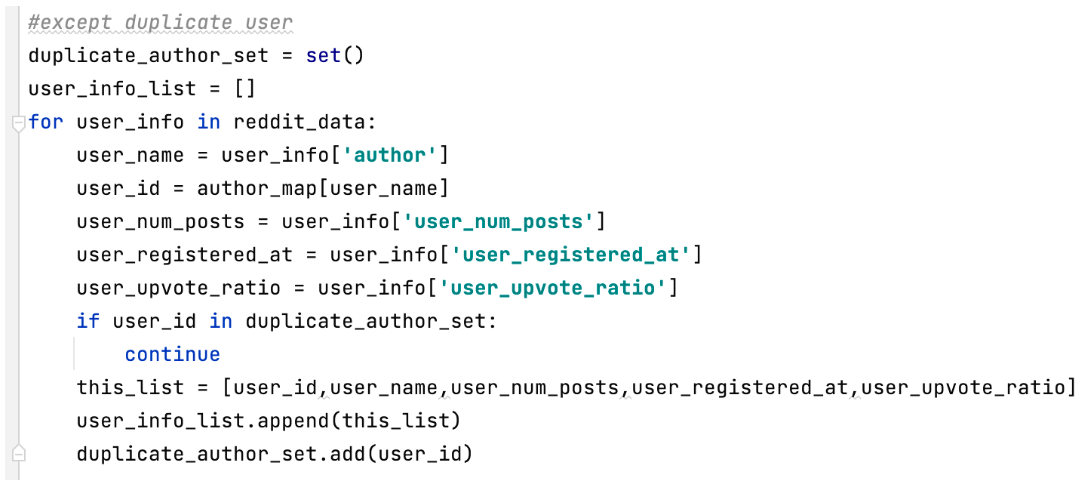
```
import csv
import json

json_content = []
with open('data_portfolio_21.csv', encoding='utf8', newline='') as f:
    reader = csv.DictReader(f)
    for row in reader:
        json_content.append(row)

# Write json to file
with open('riddit.json', 'w', encoding='utf8') as f:
    json.dump(json_content, f, indent=4)
```

Convert csv files to json format for a more visual and clear presentation. The structure of data in the file is also much clearer.

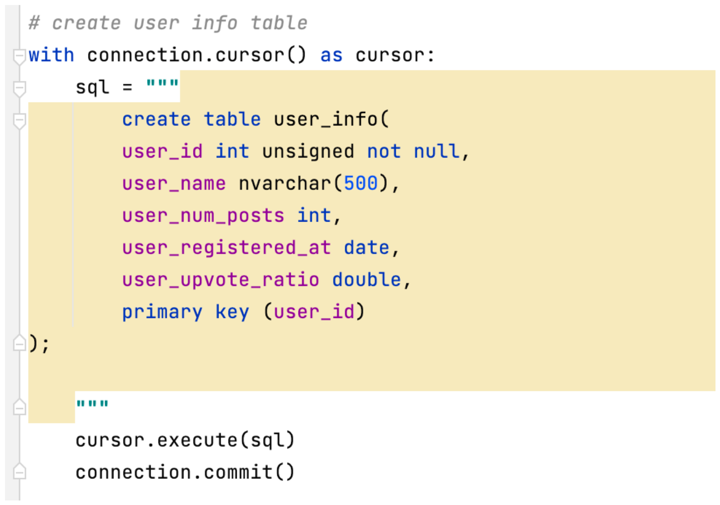
`reddit_data = json.load(open('riddit.json'))` Read json files

```
#USER INFORMATION
# user info into a table
author_map = {}
for author_names in reddit_data:
    author_name = author_names['author']
    if not author_name in author_map:
        author_map[author_name] = len(author_map)
```

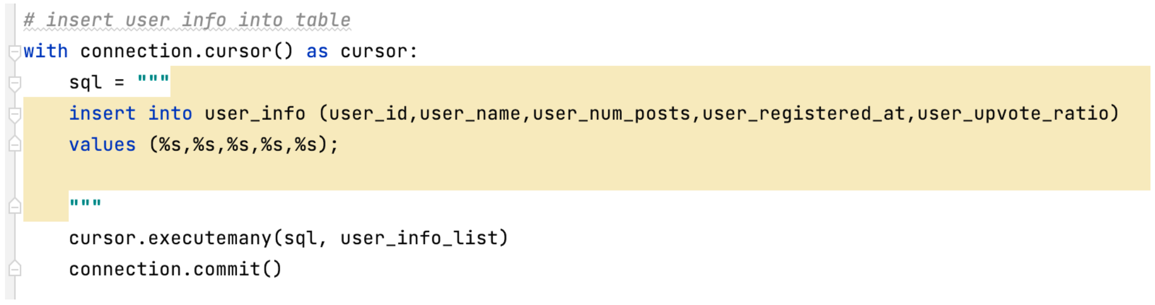
Create a *dict{}* to store the number of users.


```python
#except duplicate user
duplicate_author_set = set()
user_info_list = []
for user_info in reddit_data:
    user_name = user_info['author']
    user_id = author_map[user_name]
    user_num_posts = user_info['user_num_posts']
    user_registered_at = user_info['user_registered_at']
    user_upvote_ratio = user_info['user_upvote_ratio']
    if user_id in duplicate_author_set:
        continue
    this_list = [user_id,user_name,user_num_posts,user_registered_at,user_upvote_ratio]
    user_info_list.append(this_list)
    duplicate_author_set.add(user_id)
```

Use a for loop to read the data about the user and store it in a list.


```python
# create user info table
with connection.cursor() as cursor:
    sql = """
        create table user_info(
        user_id int unsigned not null,
        user_name nvarchar(500),
        user_num_posts int,
        user_registered_at date,
        user_upvote_ratio double,
        primary key (user_id)
);

    """
    cursor.execute(sql)
    connection.commit()
```

Create a user table and perform the declaration of primary keys and data types to achieve data integrity and normalisation of the created table.


```python
# insert user info into table
with connection.cursor() as cursor:
    sql = """
    insert into user_info (user_id,user_name,user_num_posts,user_registered_at,user_upvote_ratio)
    values (%s,%s,%s,%s,%s);

    """
    cursor.executemany(sql, user_info_list)
    connection.commit()
```

Insert the data from the list into the table.

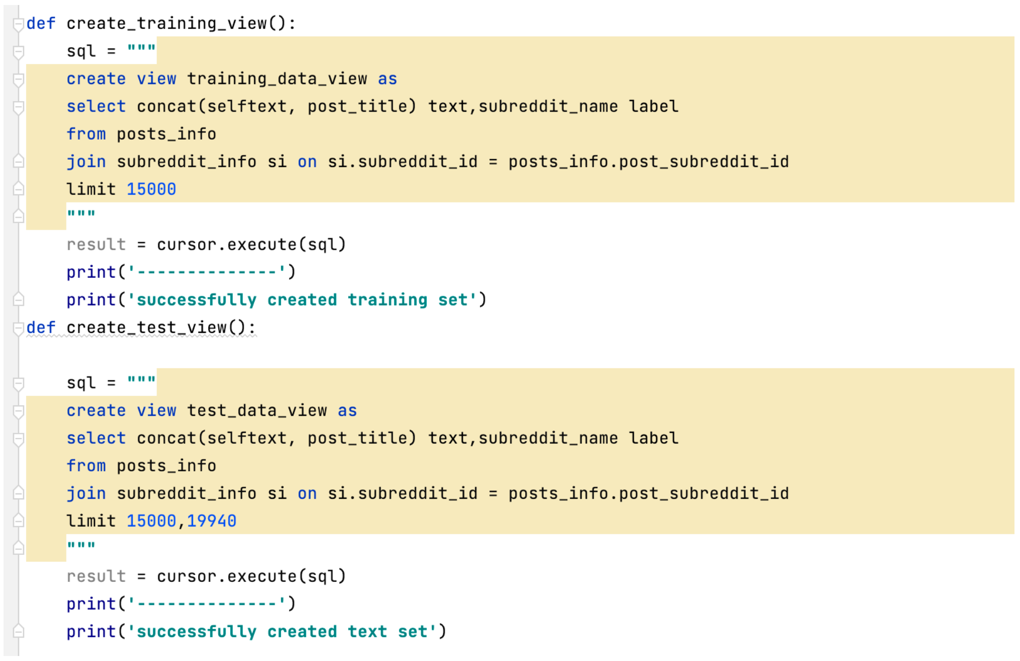
6. Reflection

Compared to parsing a csv file directly, it is much clearer to read the corresponding values in json format, rather than reading the relevant data after splitting it up. Moreover, the json format is faster to read, the code is simpler to understand and more modular, which has a positive effect on subsequent modifications and maintenance. However, when reading subreddit_fav_by, the json format is not available and the csv file must be read and the text cut and stored. In terms of table creation, 3NF does not fully implement it and does not add a foreign key to the reddit_id in the fav_user_info table. This will need to be upgraded at a later stage.

ML Application

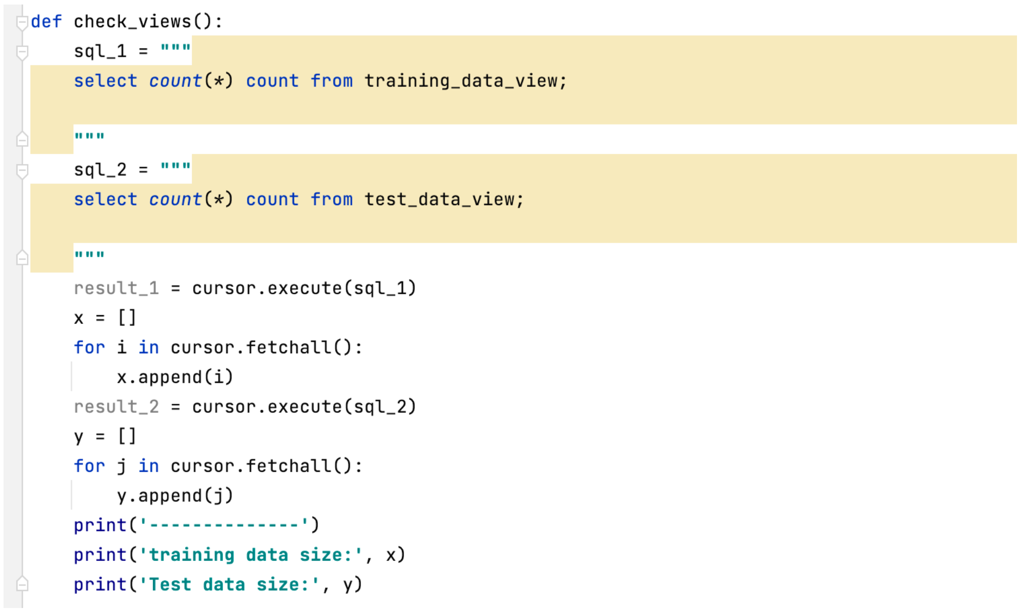
1. Preliminary data segmentation and preparation

First, two views are created as training_data_view and test_data_view, the entire data was manually divided into a training set and a test set in the ratio of 8:2 by means of SQL statements. The code is shown in the diagram below. Since most of the selftext data is NULL, the selftext and post_title is combined and referred to as text, which increases the richness of the data.



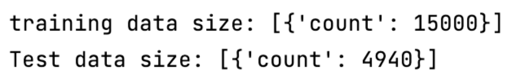
```
def create_training_view():
    sql = """
    create view training_data_view as
    select concat(selftext, post_title) text,subreddit_name label
    from posts_info
    join subreddit_info si on si.subreddit_id = posts_info.post_subreddit_id
    limit 15000
    """
    result = cursor.execute(sql)
    print('--------------')
    print('successfully created training set')
def create_test_view():

    sql = """
    create view test_data_view as
    select concat(selftext, post_title) text,subreddit_name label
    from posts_info
    join subreddit_info si on si.subreddit_id = posts_info.post_subreddit_id
    limit 15000,19940
    """
    result = cursor.execute(sql)
    print('--------------')
    print('successfully created text set')
```

Secondly, check the size of views.



```
def check_views():
    sql_1 = """
    select count(*) count from training_data_view;

    """
    sql_2 = """
    select count(*) count from test_data_view;

    """
    result_1 = cursor.execute(sql_1)
    x = []
    for i in cursor.fetchall():
        x.append(i)
    result_2 = cursor.execute(sql_2)
    y = []
    for j in cursor.fetchall():
        y.append(j)
    print('--------------')
    print('training data size:', x)
    print('Test data size:', y)
```

This function allows the size of the data set to be output, giving a ratio of almost 8:2.The output is shown below.



```
training data size: [{'count': 15000}]
Test data size: [{'count': 4940}]
```

2. Data import and feature engineering [6]

The next step is to export the data from the database, and then import it into a list using the *cursor.fetchall()* method, then transform this list into a DataFrame [5]. After the transformation of the data frame has been achieved, the text containing the covid as well as the coronavirus has to be marked. Text containing keywords is marked with a 1, the rest with a 0. The anonymous function *lambda x* is used here to implement this.

```
data_frame_training['label'] = data_frame_training['label'].apply(
    lambda x: 1 if (x.lower().find('covid') != -1 or x.lower().find('coronavirus') != -1) else 0
).astype(int)
```

Declare the four variables as *training_data_x, test_data_x, training_data_y, test_data_y*, where *training_data_x, training_data_y*, is the original data plan divided as training model, used when fit model, while *test_data_x, , test_data_y* is not involved in the training of the model, but used to evaluate the training of the model, used when scoring.

Random sampling is performed via the *DataFrame.sample()* method, but the ratio is set to 1, with the aim of breaking up the order of the entire data set.

```
count_vectorizer = CountVectorizer(analyzer='word', token_pattern=r'\w{1,}')
count_vectorizer.fit(data_frame_training['text'])
training_data_count = count_vectorizer.transform(training_data_x)
test_data_count = count_vectorizer.transform(test_data_x)
```

Then, feature engineering is performed. The text data is transformed into feature vectors and new features are created based on the existing data. In this experiment, the count vector is chosen as the feature and the CountVectorizer method is used to implement feature conversion and feature extraction [7]. For each training text, it only considers how often each word appears in that training text.

CountVectorizer converts the words in the text into a word frequency matrix and it calculates the number of occurrences of each word through the *fit()* and *transform()* function, a word frequency matrix.

3. Training models

After completing the feature engineering, the model is trained using the features created earlier. In this experiment, Linear Classifier [8], (Random Forest Classifier) [9], SVM Classifier [10], naive bayes [11], and decision tree Classifier [12] were used. The binary file generated by the random forest was discarded because it exceeded the upper limit of the database storage capacity.

```
print('loading svm classifier')
# SVM Classifier
begin_time = time()
svm_classifier_model = SVC()
svm_classifier_model.fit(training_data_count, training_data_y)
predictions = svm_classifier_model.predict(test_data_count)
result_svm_classifier = metrics.accuracy_score(predictions, test_data_y)
end_time = time()
svm_classifier_model_time = end_time - begin_time
```

The *training_data_count*, which is the shape training vector, and the target value *training_data_y* are fitted to the SVM classifier by the *fit()* function. *predict()* method to return the trained prediction results. In this experiment, the time taken to train the model, the accuracy score of the model, was the main certain criterion. The running speed can be obtained by adding the *time()* function as the start and end time before and after the training model. Accuracy scores are obtained via *metrics.accuracy_score()* where the parameters are the true label and the predicted label.

After training, the trained model is saved into MongoDB by serialisation. Serialise the model data via the *pickle.dump()* method and this can be stored in the database.

```
logistic_regression_binaries = pickle.dumps(logistic_regression_model)
svm_classifier_binaries = pickle.dumps(svm_classifier_model)
naive_bayes_binaries = pickle.dumps(naive_bayes_model)
# random_forest_binaries = pickle.dumps(random_forest_model)
decision_tree_classifier_binaries = pickle.dumps(decision_tree_classifier_model)
```

The model names, binary data, scores and speed times are stored in MongoDB and can be read from MongoDB and restored for later use for text prediction and classification.

```
collection.remove()

collection.save({'model_name': 'logistic_regression',
                 'binaires_data': logistic_regression_binaries,
                 'score': result_logistic_regression,
                 'time': logistic_regression_model_time
                })
```

By comparison, the highest rated mod is SVM Classifier, and the fastest rated mod is Linear Classifier. The SVM is the model that takes the longest time but have the highest score, as it has the advantage of requiring fewer samples and is better suited to solving text classification problems. SVM Classifier are good at coping with the case of linearly indistinguishable sample data, mainly through relaxation variables (penalty variables) and kernel function techniques, and this part is the essence of SVM [13].

The ML components were tested by ablation experiment, the Linear Classifier was found to be the optimal model in this experiment.

4. Python – MongoDB – Browser

With the flask framework [14], python can be combined with a browser.

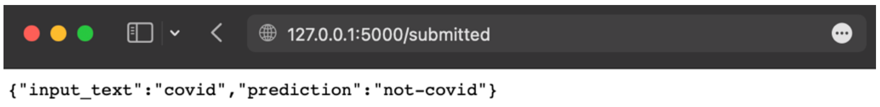
```
@app.route('/report', methods=['GET', 'POST'])
def retrieve_results():
    best_scoring_model_top3 = collection.find({}, {"model_name": 1}).sort("score", -1).limit(3)
    fast_model_top_3 = collection.find({}, {"model_name": 1}).sort("time", 1).limit(3)
```

Sorting queries on data in MongoDB via python makes it easy to get the fastest model as well as the highest scoring model.

```
global model_loaded
load_model = collection.find()
for i in load_model:
    model_loaded = pickle.loads(i['binaires_data'])
    name = i['model_name']
    print(name)
```

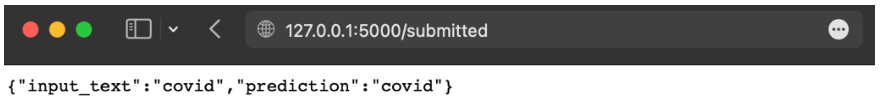
The binaries stored in MongoDB are deserialized in the *retrieve_results()* function via the *pickle.loads()* method. and stored in the variable model_loaded for prediction.

Prediction bias after adding a K Neighbour Classifier to the covid-or-not function.



```
{"input_text":"covid","prediction":"not-covid"}
```

After commenting it out the prediction becomes accurate.



```
{"input_text":"covid","prediction":"covid"}
```

5. Reflection

An attempt was made to automatically split the dataset from the test set using the *sklearn.model_selection .train_test_split()* method. But the accuracy was only about 85 percent. In addition, the label of the test set should be guaranteed to be 100% accurate, but the number of samples is too large and the text can only be labelled using the relevant function, which does not guarantee 100% accuracy. This is also an area that needs to be upgraded in the later stages.

Ethics and Bias in Data-driven Solutions

Based on the covid-or-not procedure done for this experiment, an analysis was made based on the UK data to the framework [15]. The analysis is as follows.

This system is explicitly intended to perform predictive classification of texts to determine whether they are covidentially relevant, which is a way of **Define and understand public benefit and user need**. The source of the data is reddit, which is publicly available and legal, so **Comply with the law**. In terms of **Review the quality and limitations of the data**, the project data is accurate, representative and scaled for use, but this is only for the analysis of reddit user data and should be collected from other forums such as Quora Medium, etc. Given the **Evaluate and consider wider policy implications**, plans should be put in place to ensure that data insights are used responsibly. This would also need to be developed by a specialist person or team in the relevant area. [16]

Involve Diverse Expertise was considered to be the weakest one.

The completion of an application on screening public opinion should cover different disciplines and fields and involve different experts. Only working in a diverse, multidisciplinary team with a broad skill set can contribute to the success of a data or technology project. As a software engineering student with no background in law or sociology, I may encounter difficulties in the application of software in the future.

In terms of transparency, external stakeholders should be involved, such as reddit users, government departments who want to use the software to analyse public opinion, covid-related people and people with expertise in law, sociology, etc.

In terms of accountability, this project should have people with the appropriate expertise to do the corresponding work, e.g., legal advice, social communication advice, etc. Information on expert advice and project team structure should be published where necessary. The public or their representatives should be able to monitor and control the projections made by the software and the screening of public opinion.

In terms of fairness, consideration should be given to whether the application of this software is fair to those who are unfortunate enough to be infected with the New Coronavirus. Whether it will lead to a discriminatory impact on society towards those suffering from New Coronavirus, where the virus originated or where the variant originated. Whether it will lead to antagonism between certain members of society. [17]

Solution for fairness: This system can be used to make important, life-changing decisions in many sensitive environments; therefore, it is vital to ensure that these decisions do not reflect discriminatory behaviour towards certain groups or populations.

References:

[1] Hernandez, M.J., 2013. Database design for mere mortals: a hands-on guide to relational database design. Pearson Education.

[2] Talend Real-Time Open Source Data Integration Software. 2021. *What is Data Integrity and Why Is It Important?*. [online] Available at: <https://www.talend.com/resources/what-is-data-integrity/> [Accessed 11 June 2021].

[3] Guru99.com. 2021. *What is Normalization in SQL? 1NF, 2NF, 3NF, BCNF Database (DBMS) Example*. [online] Available at: <https://www.guru99.com/database-normalization.html> [Accessed 11 June 2021].

[4] Masoud, F.A., Khattab, H. and Al-Karazoon, M., 2005. University of Jordan Case Tool (Uj-Case-Tool) for Database Reverse Engineering. *World Academy of Science, Engineering and Technology*, *9*, pp.28-31.

[5] Pandas.pydata.org. 2021. *Intro to data structures — pandas 1.2.4 documentation*. [online] Available at: <https://pandas.pydata.org/pandas-docs/stable/user_guide/dsintro.html> [Accessed 11 June 2021].

[6] Scikit-learn.org. 2021. *1. Supervised learning — scikit-learn 0.24.2 documentation*. [online] Available at: <https://scikit-learn.org/stable/supervised_learning.html#supervised-learning> [Accessed 11 June 2021].

[7] Sekhar, S.M., Siddesh, G.M., Raj, M. and Manvi, S.S., 2021. Protein class prediction based on Count Vectorizer and long short term memory. *International Journal of Information Technology*, *13*(1), pp.341-348.

[8] Mladenić, D., Brank, J., Grobelnik, M. and Milic-Frayling, N., 2004, July. Feature selection using linear classifier weights: interaction with classification models. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval* (pp. 234-241).

[9] Pal, M., 2005. Random forest classifier for remote sensing classification. *International journal of remote sensing*, *26*(1), pp.217-222.

[10] Madzarov, G., Gjorgjevikj, D. and Chorbev, I., 2009. A multi-class SVM classifier utilizing binary decision tree. *Informatica*, *33*(2).

[12] Safavian, S.R. and Landgrebe, D., 1991. A survey of decision tree classifier methodology. *IEEE transactions on systems, man, and cybernetics*, *21*(3), pp.660-674.

[13] Medium. 2021. *Top 4 advantages and disadvantages of Support Vector Machine or SVM*. [online] Available at: <https://dhirajkumarblog.medium.com/top-4-advantages-and-disadvantages-of-support-vector-machine-or-svm-a3c06a2b107> [Accessed 11 June 2021].

[14] Grinberg, M., 2018. *Flask web development: developing web applications with python*. " O'Reilly Media, Inc.".

[15] GOV.UK. 2021. *Data Ethics Framework*. [online] Available at: <https://www.gov.uk/government/publications/data-ethics-framework> [Accessed 11 June 2021].

[16] Nicol Turner-Lee, a., 2021. *Algorithmic bias detection and mitigation: Best practices and policies to reduce consumer harms*. [online] Brookings. Available at: <https://www.brookings.edu/research/algorithmic-bias-detection-and-mitigation-best-practices-and-policies-to-reduce-consumer-harms/> [Accessed 11 June 2021].

[17] Mehrabi, N., Morstatter, F., Saxena, N., Lerman, K. and Galstyan, A., 2019. A survey on bias and fairness in machine learning. *arXiv preprint arXiv:1908.09635*.