

Yishan Li

Xinyue Yu

DSCI 551

Dec 3rd 2021

# DSCI 551 Final Report

## 1 Project Members

Yishan Li:

Email: liyishan@usc.edu

Undergraduate major: Mathematics - Computer Science.

Background knowledge and skills: Python, machine learning, HTML5, CSS, JavaScript.

Current Program: Applied Data Science.

Xinyue Yu:

Email: yuxinyue@usc.edu

Undergraduate major: Computer Science.

Background knowledge and skills: Database, front end programming.

Current Program: Communication Data Science.

## 2 Goal

Predicting if a patient has Alzheimer's Disease based on body index and MRI scans using machine learning.

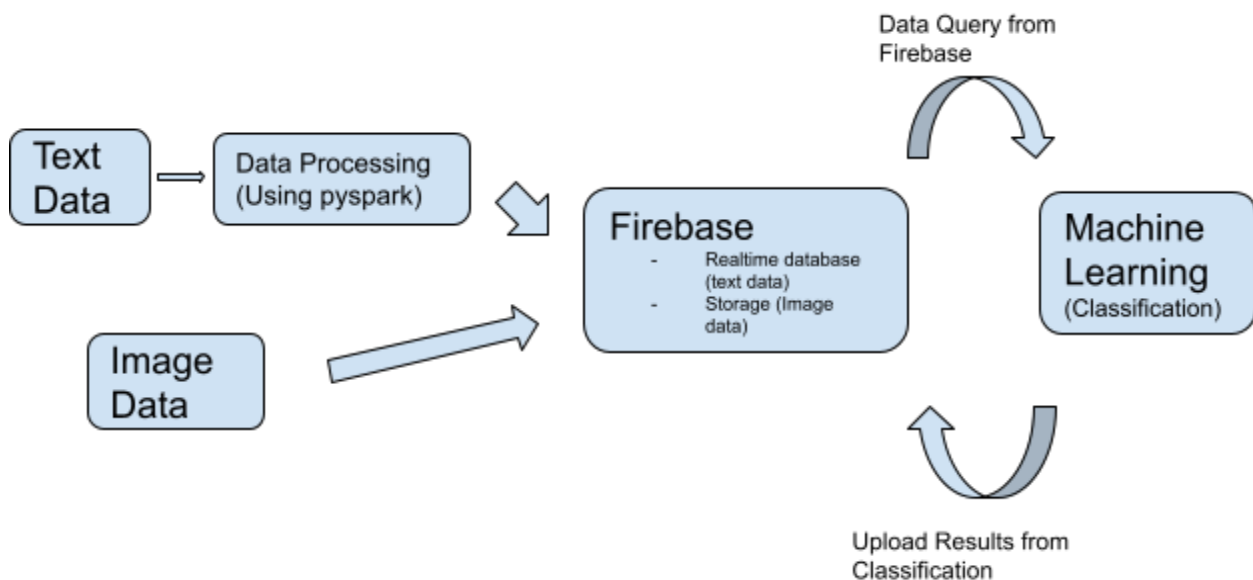
## 3 Background

Alzheimer's disease is a brain disorder that slowly destroys memory and thinking skills and, eventually, the ability to carry out the simplest tasks. In most people with the disease —

those with the late-onset type symptoms first show in their mid-60s. Rarely, early-onset Alzheimer's appears between a person's 30s and mid-60s. Alzheimer's disease is the most common cause of dementia among older adults. We aim to identify and distinguish features in the MRI image and the text data extracted from the MRI image using machine learning to determine if a patient has Alzheimer's disease. We want to address the importance of this seeming distant disease for those who have grandparents around the age of 70 like us.

Our website serves mainly for two goals: let the user predict the probability of getting Alzheimer's by using the *Alzheimer's Prediction tool* webpage where they can provide their own health information (which will not be recorded and uploaded to database for privacy reasons), and compare and see the prediction results of our classification model on MRI scans.

## 4 Architecture



Graph 1.1

## 5 Data Preparation and Data Management

### 5.1 Data Source

- Text data ([Open Access Series of Imaging Studies \(OASIS\)](#))
- Non-text data (<https://www.kaggle.com/legendahmed/alzheimermridataset>)

### 5.2 Data Preparation

#### - Text Data

The text dataset is cleaned using Python. The libraries we use include *pyspark*, *pandas*, *sklearn*, *seaborn*, *plotly*, *json*, and *requests*. The data is imported into spark dataframe. The unnecessary features of text data like Subject ID, MRI ID, etc. are dropped, some of the columns are renamed, and the gender categorical feature is altered with the encoding method using pyspark functions like `drop()`, `filter()`, `when()`, etc.

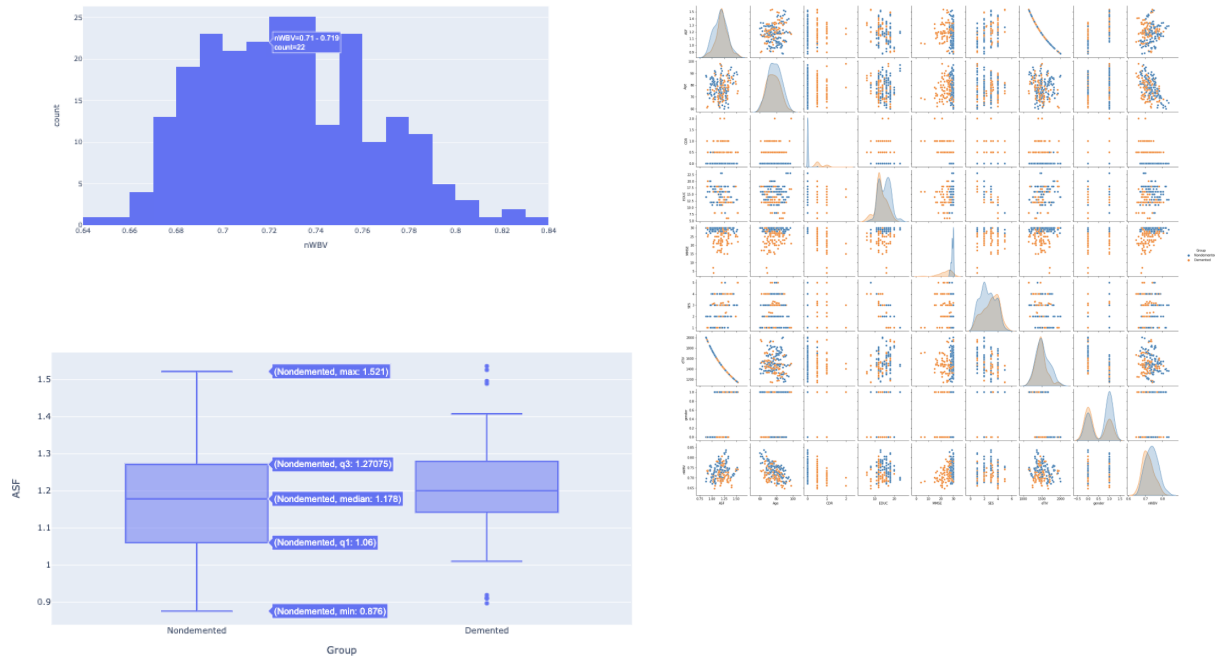
We conducted data imputation using Iterative Imputer from sklearn to fill in the missing values of columns. The data has been splitted into train and test data, and converted to JSON format and then stored into the realtime database provided by Firebase.

	Group	Age	EDUC	SES	MMSE	CDR	eTIV	nWBV	ASF	gender
0	Nondemented	87	14	2.0	27.0	0.0	1987	0.696	0.883	0
1	Nondemented	88	14	2.0	30.0	0.0	2004	0.681	0.876	0
2	Demented	75	12	NaN	23.0	0.5	1678	0.736	1.046	0
3	Demented	76	12	NaN	28.0	0.5	1738	0.713	1.010	0
4	Demented	80	12	NaN	22.0	0.5	1698	0.701	1.034	0

Text Data  
Preview

We also did some data exploratory analysis for training and testing data using matplotlib, seaborn, and plotly. For example, generating the pair plots hued by class (demented and nondemented) to identify possible significant relationship, histogram of

each column to see data distribution, and boxplots of each column for the two classes we have for data to identify the max, min, and outliers. (Below Graph 2.1 demonstrate the examples of histogram, boxplot, and pair plot in data exploration process)



Graph 2.1

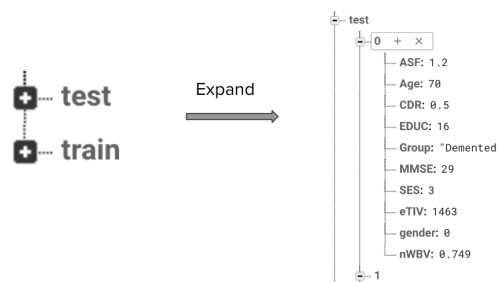
### - Image Data

The MRI scans are splitted into train and test sets for classification.

## 5.3 Data Storage

### - Text Data

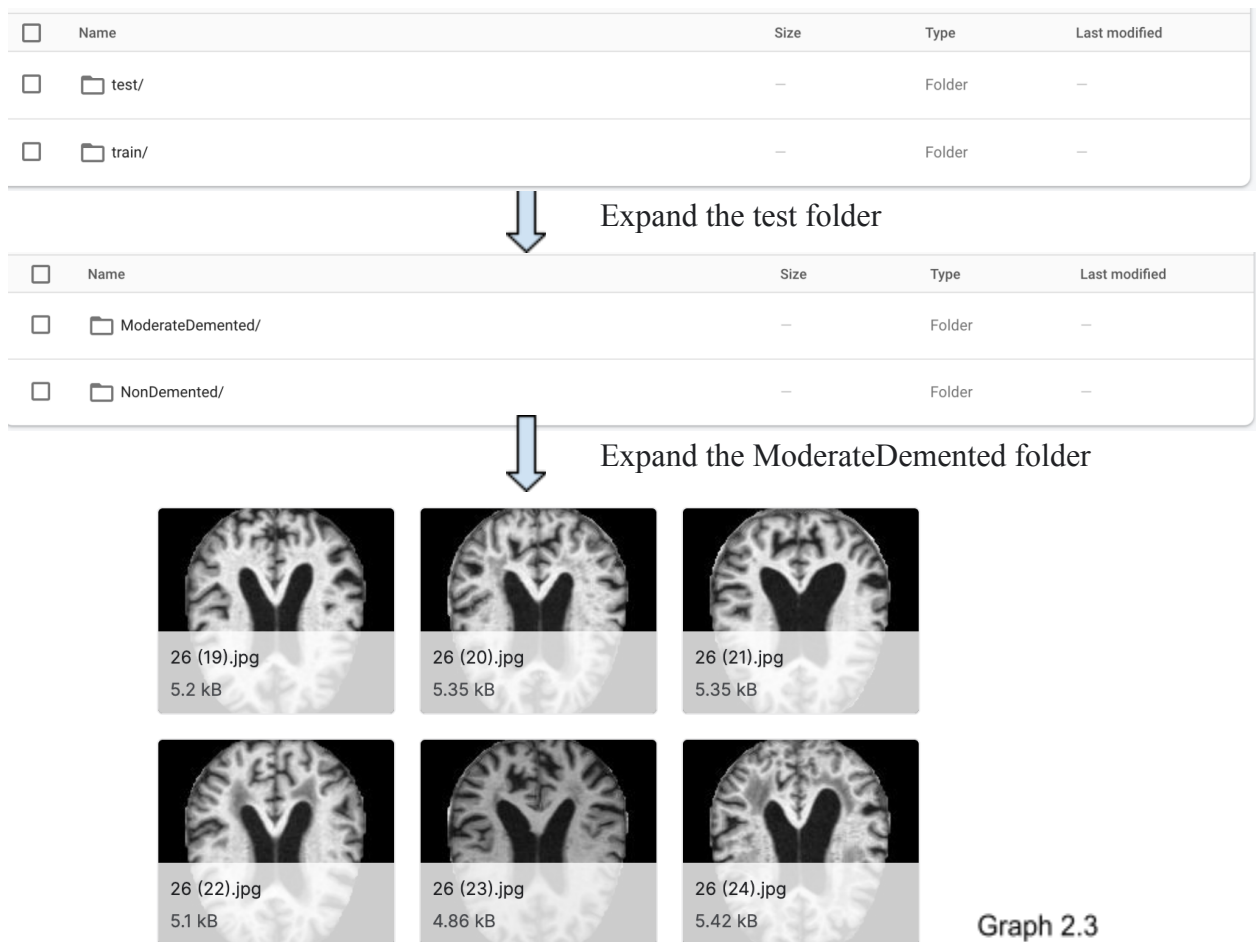
The text data is converted to json format, splitted into train and test sets, and uploaded to Firebase Realtime Database. Graph 2.2 is a snapshot from our database.



Graph 2.2

## - Image Data

The image data is divided into training and testing sets and then stored in separate folders based on their class using cloud data storage, Firebase Storage. Graph 2.3 is a snapshot from our firebase storage.



Graph 2.3

## 6 Data Retrieving, Machine Learning, and Data Storage for Results

### 6.1 Data Retrieving

#### - Text Data

We use the library requests to retrieve the text data from the firebase realtime database. Graph 3.1 shows how test data is retrieved from the database.

```
response = requests.get(test_url)
text_test = response.json()
test_t = pd.DataFrame.from_dict(text_test, orient='columns')
```

Graph 3.1

### - Image Data

We use the library prebase to retrieve the image data from the firebase storage.

Graph 3.2 shows the setup code to retrieve the image data from storage.

```
config = {
    "apiKey": "AIzaSyCuiVp0qI3AZhI4IqV-i_Uskamxq8591Ss",
    "authDomain": "finalproject-42236.firebaseio.com",
    "databaseURL": "https://finalproject-42236-default-rtdb.firebaseio.com",
    "projectId": "finalproject-42236",
    "storageBucket": "finalproject-42236.appspot.com",
    "messagingSenderId": "80230405227",
    "appId": "1:80230405227:web:487eef8725a91a467d8d2e",
    "measurementId": "G-CX9FMSLLEL",
    "serviceAccount": "finalproject-42236-firebase-adminsdk-yd1bm-b03cb2ada1.json"
}
```

Graph 3.2

```
firebase = pyrebase.initialize_app(config)
storage = firebase.storage()
```

## 6.2 Machine Learning

### - Text Data

For the machine learning part of text data, we choose to use the supervised learning - classification, since we already know the labels, which is the column *Group* in this case, and our goal is to distinguish between the demented and nondemented group based on input data, classification is the most suitable learning we should choose.

To conduct the machine learning we choose to use two kinds of classifiers: Random Forest and Naive Bayes. The reason we choose to use two classifiers is because the users can compare the performance of two classifiers; the prediction results, the predictions probabilities for two classes, and the accuracy of two classifiers are shown in our webpage, which will be illustrated in detailed in the 7th part of our final report - Web UI Design.

The models are built using the sklearn library and fit with training data obtained from the Firebase realtime database. Then the capability of the models are being tested

with the testing dataset and the user input. The prediction results from the testing dataset are uploaded to the realtime database.

#### - **Image Data**

We also use classification for the non-text data, MRI scans. As stated above, the goal of our project is to distinguish between the demented and nondemented group, so classification is the most adequate choice. Instead of using the sklearn library, we choose to use the tensorflow for classification on the image data.

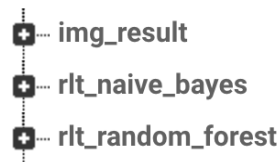
The data is splitted into three sets when conducting classification. Training set, validation set, and testing set. The data are preprocessed to be more ideal for a neural network by rescaling RGB channel values with standardization. Our model is made up with three convolution blocks and uses `tf.keras.Sequential()` to group a linear stack of layers into the prediction model.

The model is trained and tuned using the training and validation data, and tested on the testing set. The prediction results are also being stored into the Firebase realtime database.

### **6.3 Data Storage for Results**

The prediction results from two classifiers for text data and the prediction result from the model of non text data are all stored into Firebase. Graph 4.1 is a snapshot of the database.

finalproject-42236-default-rtdb

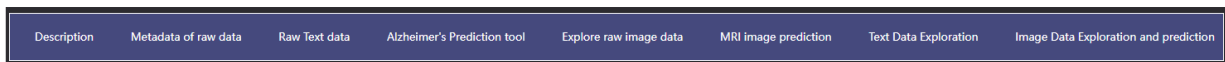


Graph 4.1

## 7 Web UI Design

### 7.1 Basic Information

The webpage of our final project is done by HTML, CSS, and JavaScript. “style.css” file contains all the design details of our webpage. For example, how we implemented the navigation bar, and how we do the image, text alignment. You can click each option in the navigation bar to see the functions of our project. We have “Description”, “Metadata of Raw Data”, “Raw Text Data”, “Alzheimer’s Prediction Tool”, “Explore Raw Image Data”, “MRI Image Prediction”, “Text Data Exploration” and “Image Data Exploration and Prediction” in the navigation bar. See Graph 4.2.



Graph 4.2

For each page, you can click on the navigation bar or click backwards to go back to the page that you want to explore or you have explored before, but for the “Alzheimer’s Prediction Tool”, you have to click the backwards sign from your browser because this was an external website; also, for “Explore Raw Image Data”, you can click “Go back” on the top left corner to go back or backwards on the browser.

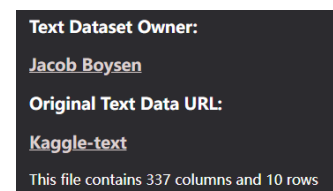
### 7.2 Each section below is corresponding to a web page of our website

- **Description**

This web page is done by HTML and shows the Introduction of our project and website.

- **Metadata of Raw Data**

In this web page, we include the author and the link of our text data and image data and provide hyperlinks to the



Graph 5.1



authors and the data. For the text data part, we also include the columns and rows of this dataset. (Graph 5.1)

The middle part of this webpage introduces the abbreviation and the full name of each column, which lets the users know the meaning of the columns and search for the medical terminologies. (Graph 5.2)

ASF: Atlas Scaling Factor
Age: Age
Group: Demented or not demented
SES: Socioeconomic Status
EDUC: Years of Education
nMBV: Normalize Whole Brain Volume
eTIV: Estimated total intracranial volume
CDR: Clinical Dementia Rating
MMSE: Mini Mental State Examination
gender: 0.0 as male, 1.0 as female

Graph 5.2

#### - Raw Text Data

In this web page, we have an interactive grid called AG-Grid to let the users explore with the text dataset that we use. Each column contains the data we can explore, and this grid allows users to sort and filter the data. To use this grid, users can click on each column to

Atlas Scaling Factor	Age ↑	Clinical Dementia Rati...
1.273	60	0
1.252	60	0
1.331	61	0
1.337	61	0
0.897	61	1
1.16	61	0
1.274	62	0

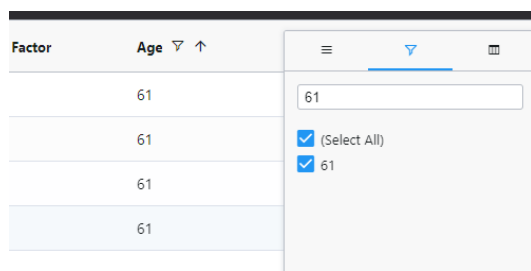
Graph 5.3

sort the whole grid by this column in an ascending order. The image below shows when we click the “Age” column, all the records

will be sorted based on this column in an ascending order (Graph 5.3). Also, on the right most part of each column, our users can click the icon to pin the column, and auto resize the columns (Graph 5.4). On the second

Age ↑	
60	Pin Column
60	Autosize This Column
61	Autosize All Columns
61	Reset Columns

Graph 5.4



Factor	Age ▼ ↑
	61
	61
	61
	61

Graph 5.5

icon, we allow our users to filter the data to see all the information of the people(demented and non-demented people) involved in this data to match those with yours or your relative's personal information.

Once the users have filtered the data they want, they can export the file into an excel file and store it in their own computer.



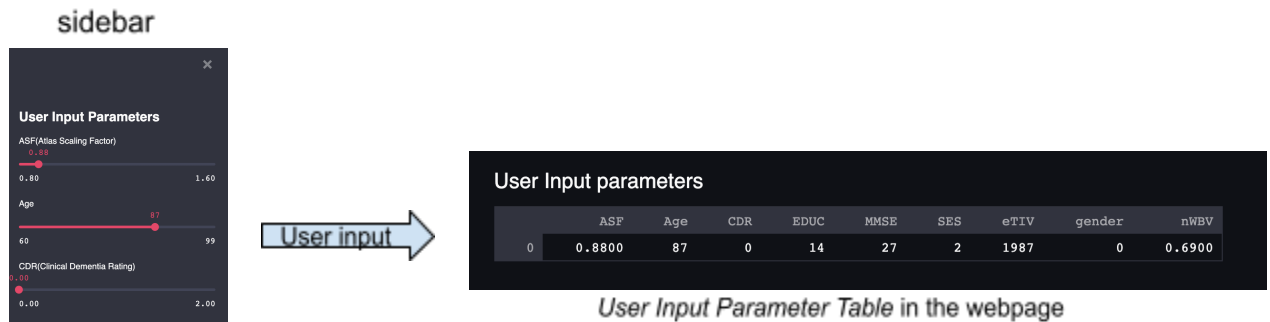
Graph 5.6

#### - Alzheimer's Prediction tool

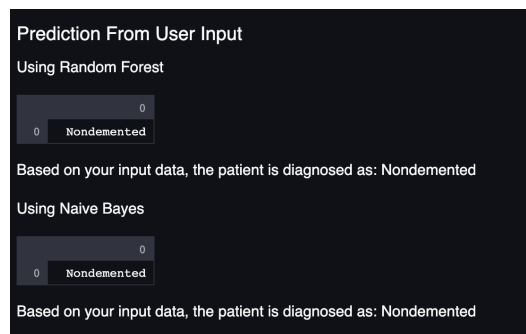
This webpage is built using python script and library streamlit, which will turn my python script into a shareable web app, and it is deployed using Heroku. Besides viewing it on our website, you can also see it with this link <https://finallit.herokuapp.com/>. This webpage allows us to retrieve training data, build models (Random Forest Classifier and Naive Bayes Classifier) and conduct classification, and upload results in real time.

In this web page, we provide a brief description about how to use this app in the beginning. There is a sidebar in the webpage, and the users are allowed to put in their own data which will also be reflected in the *User Input Parameters* table. This user input data will be put into the trained models and the models will generate predictions based on these inputs (Graph 5.7). The predictions, probabilities of prediction for each class, and

the interpretation of the results for each classifier are shown in the tables for comparison (Graph 5.8).



Graph 5.7



Graph 5.8

The prediction results of test data and the true label of test data are also demonstrated in this webpage. Eventually, the prediction results from test data will be uploaded to the Firebase realtime database in real time but not the prediction result from user input in consideration of the patient's privacy.

You can see the python script in the folder: Code File/textml.py

#### - Explore Raw Image Data

This page shows the comparison between demented and non-demented people's MRI images. We can use the arrow signs on the left and right side of each image as well as the button under the image to see the images and compare those with your own images. Notably, demented people's MRI images will have a butterfly-shaped hole at the center of the image, while those non-demented people don't have.

### - **MRI Image Prediction**

The MRI Image Prediction page includes the information about the image data we used to build the model, how we built the model, and a sample of prediction results from test data.

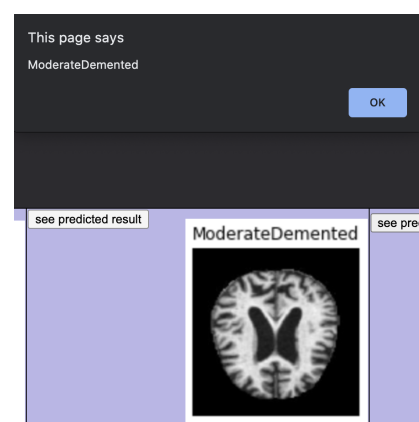
In the Sample Prediction Results section of the webpage, each MRI scan has a true label on its top, and if you click on the button '*see prediction result*' in the same

cell of the table, it will prompt the prediction result from our classification model (Graph 5.9).

### - **Text Data Exploration**

This webpage is made from a python notebook. We provide some interactive graphs using the plotly library that will allow users to have a closer look of our text data. Users can put their mouse on the graph, and detailed data information pertaining to that area will be shown automatically, and users can also zoom in the graph. The first part of the webpage is the histogram for each column of the text data. The second part of the webpage is the boxplot of each column of the text data except the Group column, since

Graph 5.9



the boxplots are hued by the class. Both the first part and the second part are separated by plots for training data and testing data.

- **Image Data Exploration and Prediction**

The *Image Data Exploration and Prediction* webpage contains detailed information about how the classification model for the MRI scans can be built and it is also made from a python notebook. Thus, if the user wishes to conduct similar experiments, they can refer to this webpage.

## **8 Conclusion**

### **8.1 Lessons & Challenges & Experiences & Skills Gained**

To make the website, we have learned a lot and built our web pages from scratch. Yishan had previous experience with html, but Xinyue didn't. So Xinyue learnt the functions of JQuery, JavaScript, HTML and CSS file suffixes for the basic outline of the website.

To learn how to fully use the interactive function of the AG-Grid is also somewhat challenging. We learnt the basic grid knowledge from youtube and followed through its instructions. However, when we tried to explore the interactive function and tried to make our grid look like that, we found that it won't appear on our website as it should be. But later, we opened the webpage of this undone website, and inspected it, and found out it was the license issue. So we went to their website and found the license that we exactly needed, problem solved.

The process of finding tools to build an online real time machine learning application was difficult, but we managed to find streamlit which allowed us to do it. However, it took time and effort in deploying this app. After several failures, we were able to use Heroku to publish it. This

can also be very helpful in building future data science projects. You can view this repository to see how our machine learning app is being deployed <https://github.com/yishan575757/lit2>.

It was also our first time using cloud storage (Firebase Storage). It took us several tries to properly query the image data from the storage. We were eventually able to set up the appropriate configuration in accessing the data.

Image classification was also new to us, but the tensorflow library and its documentation helped us a lot that allowed us to conduct machine learning successfully.

## **8.2 Conclusion**

From this project, we have learnt how to build our own website from scratch, how to make use of the learning resources online while we stuck in the middle of technical issues, how to generate a website using Python with the help of streamlit, and how to clean, select, and change the columns name of the text data by using Spark dataframe. As for now, machine learning for text data has a relatively high accuracy compared to image data and generates more reliable prediction results and this can be improved in the future if we are able to find more MRI scans, or find a method to tune our model to become more accurate.

In conclusion, we are able to fulfill our goal to predict if a patient has Alzheimer's Disease based on body index and MRI scans using machine learning. Our web UI was able to meet requirements, and we utilize the firebase to store data and the inference results. We also used pyspark to extract features from text data, and our website is intuitive enough to be used for a non technical user.

Demo video link:

<https://drive.google.com/file/d/1xe19MADzuxsrofOv2F0jmyUUs3TwfDq0/view?usp=sharing>

Demo slides link:

[https://docs.google.com/presentation/d/1r5jh1cZh\\_HjXSAq5DHpn3L7fJFkve42sPcfYX7C2tMc/edit?usp=sharing](https://docs.google.com/presentation/d/1r5jh1cZh_HjXSAq5DHpn3L7fJFkve42sPcfYX7C2tMc/edit?usp=sharing)

## 9 Reference

Interactive graph

<https://plotly.com/python/>

Webpage

[https://www.w3schools.com/html/html\\_images.asp](https://www.w3schools.com/html/html_images.asp)

[https://www.w3schools.com/html/html\\_tables.asp](https://www.w3schools.com/html/html_tables.asp)

[https://www.w3schools.com/tags/tag\\_button.asp](https://www.w3schools.com/tags/tag_button.asp)

AG-grid

<https://www.ag-grid.com/javascript-data-grid/getting-started/>

<https://www.youtube.com/watch?v=KS-wg5zfCXc>

Pyspark

<https://spark.apache.org/docs/3.1.1/api/python/index.html>

Streamlit

<https://streamlit.io/>

<https://towardsdatascience.com/a-quick-tutorial-on-how-to-deploy-your-streamlit-app-to-heroku-874e1250dadd>

Machine Learning

<https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html>

[https://scikit-learn.org/stable/modules/generated/sklearn.naive\\_bayes.GaussianNB.html](https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html)

<https://www.tensorflow.org/tutorials/images/classification>

<https://scikit-learn.org/stable/modules/generated/sklearn.impute.IterativeImputer.html#sklearn.impute.IterativeImputer>