**MACHINE LEARNING**

Ankit Butola

# Table of Contents

[Introduction 4](#_bookmark0)

[Task 2 5](#_bookmark1)

[Task 2 (Data Splitting) 12](#_bookmark2)

[Task 2 (Classifier 1: Decision Tree) 12](#_bookmark3)

[Task 2 (Classifier 2: Random Forest) 13](#_bookmark4)

[Task 2 (Classifier 3: CNN Model) 14](#_bookmark5)

[Task 2 (Classifier 3: Model Comparison and Result) 19](#_bookmark6)

[Task 3 20](#_bookmark7)

[Word Cloud Illustration 24](#_bookmark8)

[Wordcloud for TESLA 25](#_bookmark9)

[Wordcloud for GOOGLE: 26](#_bookmark10)

[Wordcloud for LINKEDIN 26](#_bookmark11)

[Word cloud for AMAZON 27](#_bookmark12)

[Text MINING 27](#_bookmark13)

**Group Report**

As a part of Machine Learning CA2,

For the task 2, based on hand gesture, we were aware of the approach and we decided to include three modifiers namely, (a). Decision Tree (b). Random Forest and (c). CNN model.

In case of task 3, we were supposed to use the google API and fetch the company reviews (any four company of our choice) and present a in word cloud illustration, but we faced the problem to get the entire reviews using API, as we were able to fetch max 5 reviews for each company, we tried various approach, we even contacted some other class mates to understand but eventually we decided to fetch the review using web scraping as we were running out of time to submit the assignment.

# Introduction

In this assignment, we will be covering two things,

Firstly, the hand gesture recognition app, for hand gesture application, we will be using the sign\_mnist dataset which is available on Moodle, we will be scripting the entire code in colab using python along with use of various packages like,

**Pandas**: to structure our dataset, manipulate the datatype (if required)

**Numpy**: to perform various mathematical operations while using arrays and to make the data structure powerful.

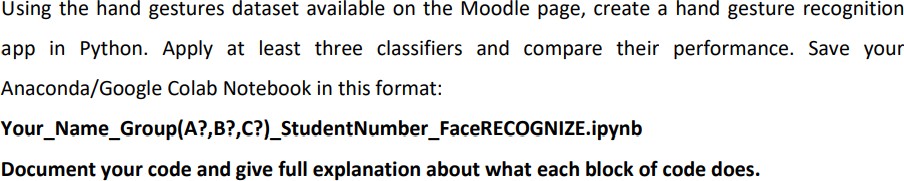
**Matplotlib.pyplot**: for plotting graphs.

**Seaborn**: for analysing and exploring the data and for its visualization

Furthermore, we will compare the accuracy of test and train dataset using three different models, Decision Tree, Random Forest and the CNN model.

For the second part, we be creating an google API using developer account, and we be downloading the reviews for the four company’s having reviews more than 1000 per company and using the text mining technique we will identify the top 10 mostly used words from the reviews that we will be downloading, and finally we will illustrate the output in a Word Cloud Illustration format with a condition applicable that the size of the word will get huge if it is mostly used from the overall reviews.

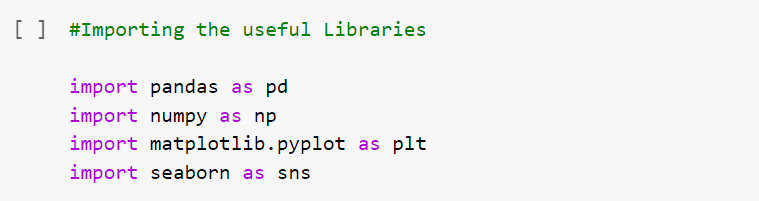
# Task 2



For this we have firstly downloaded all the pre-requites like the dataset and the useful information document related to sign\_mist available on moodle.

Step 1:

We have imported all the useful libraries,



**Pandas**: to structure our dataset, manipulate the datatype (if required)

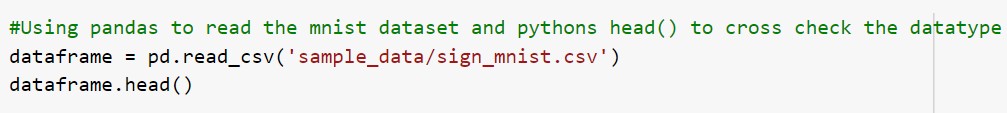
**Numpy**: to perform various mathematical operations while using arrays and to make the data structure powerful.

**Matplotlib.pyplot**: for plotting graphs.

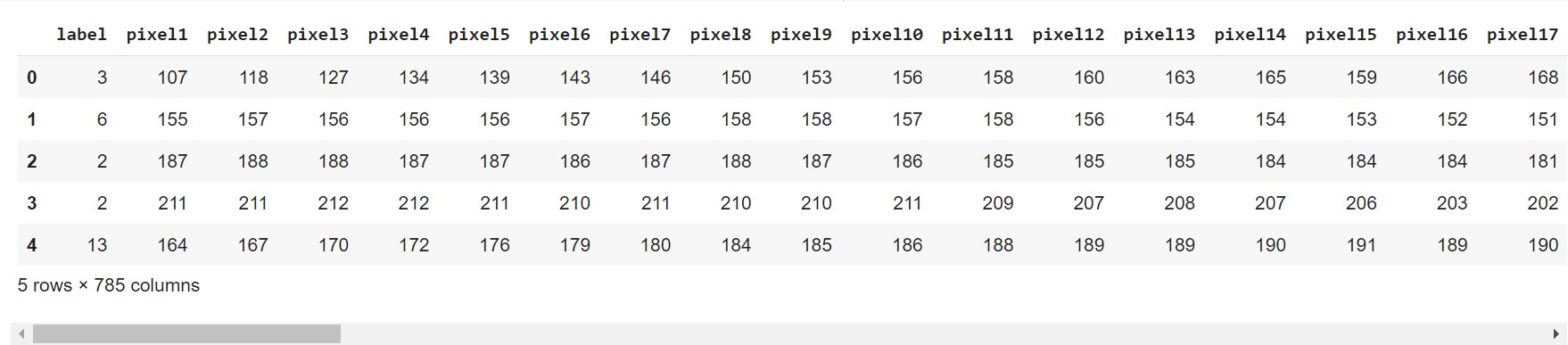
**Seaborn**: for analysing and exploring the data and for its visualization

Step 2:

Reading the mnist dataset using pandas, as it’s the fastest way to read the CSV data in pandas data frame format and quickly testing the dataset to check the datatype of the dataset using the head() function. Here we are storing the data into dataframe variable.



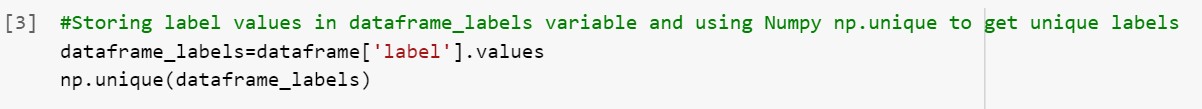
Output:



Step 3:

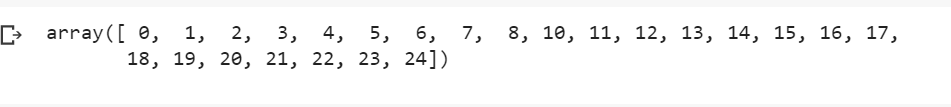
Now, we will check the unique values of label from our dataset, for which we will be using NumPy functions,

np.unique(): to get unique element of the array (in our case unique values of labels)



Output:

So, we have total 25 unique values for label starting from 0 to 24



Step 4:

Checking the shape of our dataframe, for this we will use pandas shape function to get the numbers of rows and the columns



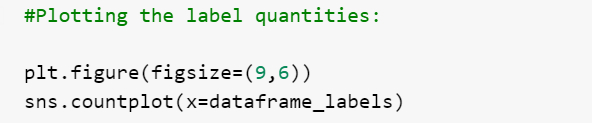
Output:



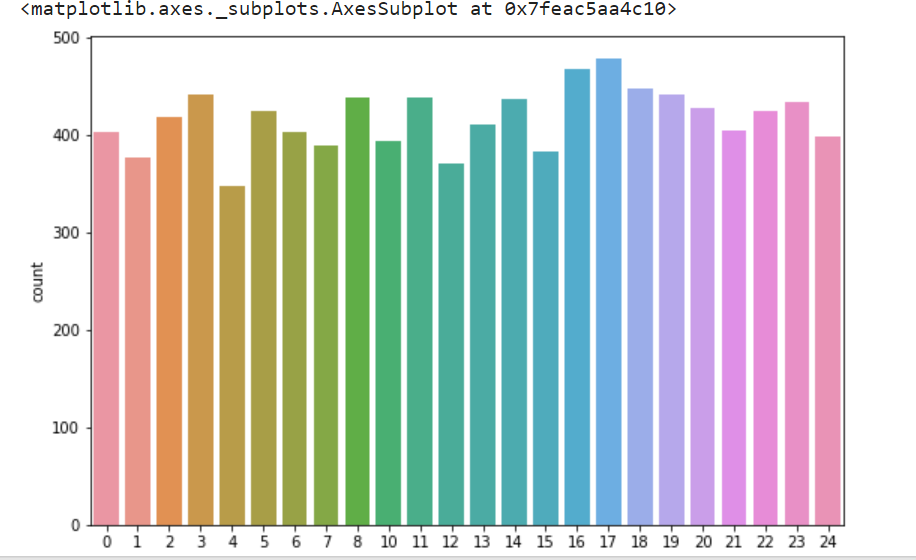
We have 10000 rows and 785 columns in our dataset.

Step 5:

Let us now check the quantities for each labels using countplot() function from seaborn.

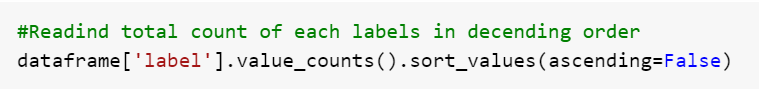


Output:

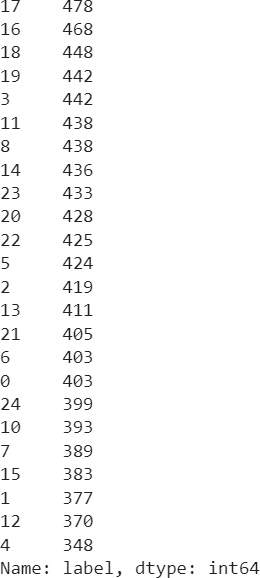


Step 6:

Making the above step easier to read, we will see the counts for each labels in descending order.

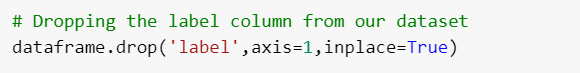


Output:



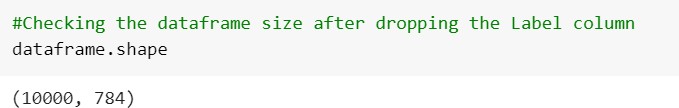
Step 7:

Dropping the label column from our dataset, as we will be working on pixels which are stored in the variables available except for the label column.



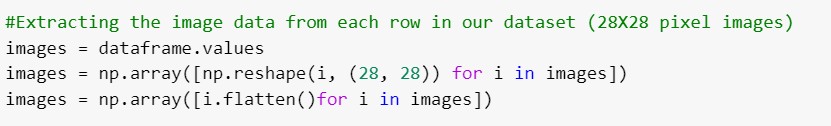
Step 8:

Cross checking the dataframe shape, for which we should have 784 columns and not 785.



Step 9:

Now, since we have deleted the un wanted column (label), we can now extract the image from our dataset which is stored in a pixel code format, for this we will storing our dataframe values in images variable and then using for loop to execute entire data set and with a pixel of 28X28



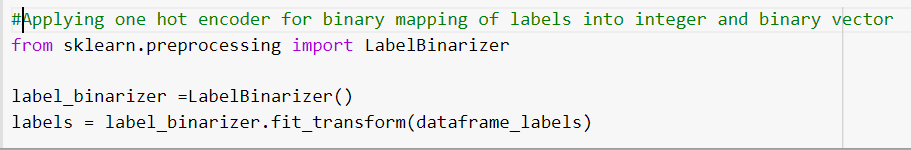
Step 10:

Applying one hot encoder, in our dataset we have 0-24 labels, we will firstly do the integer encoding to convert out dataset into a binary vector having 25 values which will look something like,

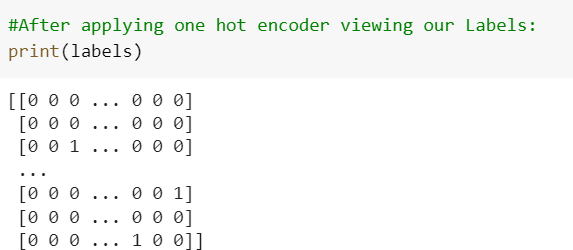
0: [1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]

1: [0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0] and so on.

We will be using automatic method to define the binary mapping of labels into integer and binary vectors using the LabelBinarizer() from sklearn.preprocessing.



Output:



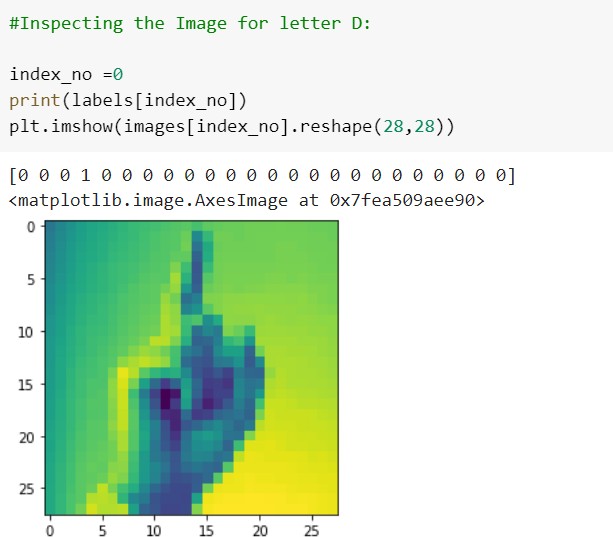
Step 11:

Now, since we have already binarized our dataset we can now proceed to check the images. Let us check,

a. [0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0] which means “D”

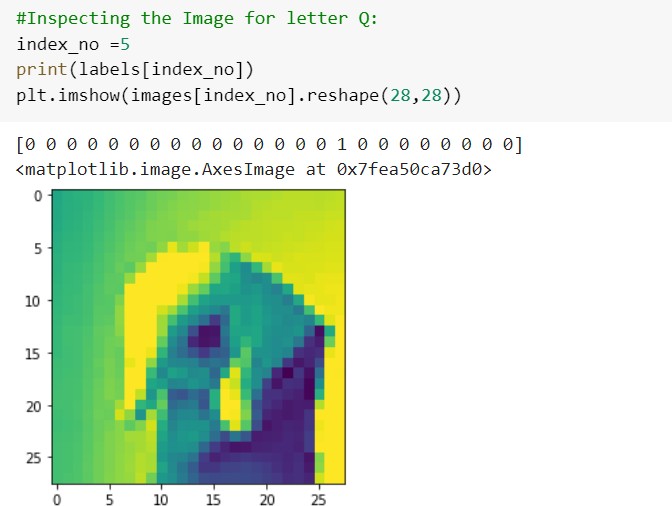


Output:

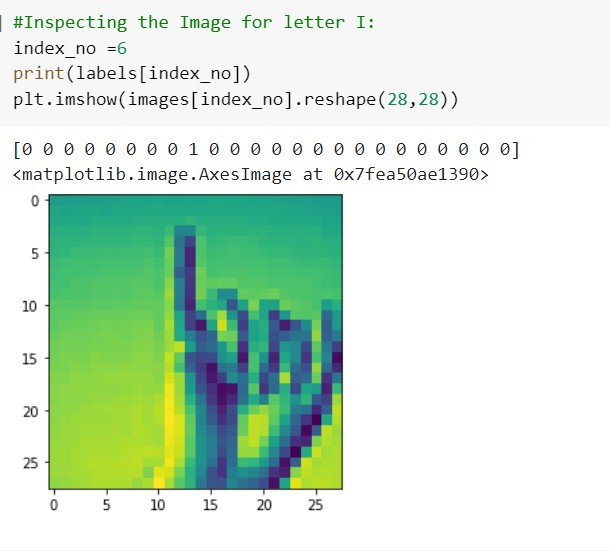


b. [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0] which means “Q”





c. [0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0] which means “I”

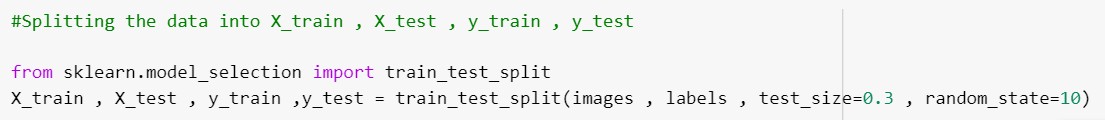


# Task 2 (Data Splitting)

Before creating the model, we need to split the data into test and train dataset which will help us to further evaluate the performance of the prediction done for the unbiased models (that we will be developing in the next step)

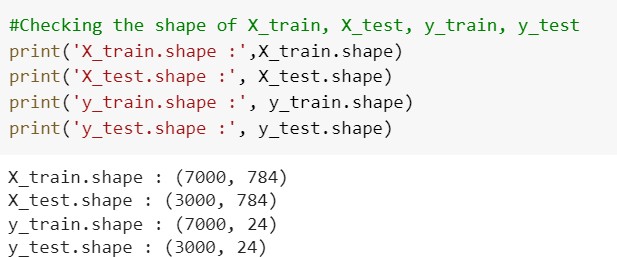
Step 12:

For splitting the data we will be using train\_test\_split() function from sklearn.model\_selection package.



Step 13:

Checking the shape of X\_train, X\_test, y\_train, y\_test

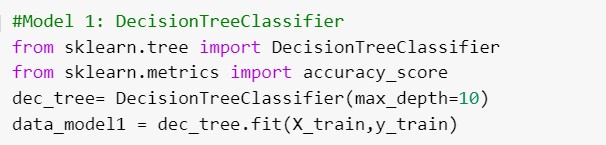


# Task 2 (Classifier 1: Decision Tree)

Step 14:

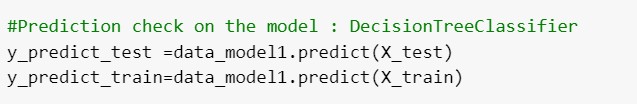
The first Classifier we are going to use is, The Decision Tree, as it is simple to understand and for this we will be making use of [DecisionTreeClassifier](https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html#sklearn.tree.DecisionTreeClassifier) function from sklearn.tree package and to calculate the accuracy we will be using accuracy\_score from the same sklearn.metrics package.

The [DecisionTreeClassifier](https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html#sklearn.tree.DecisionTreeClassifier) take 2 parameters having X\_train,y\_train as an input array and after fitting the we can predict the accuracy.



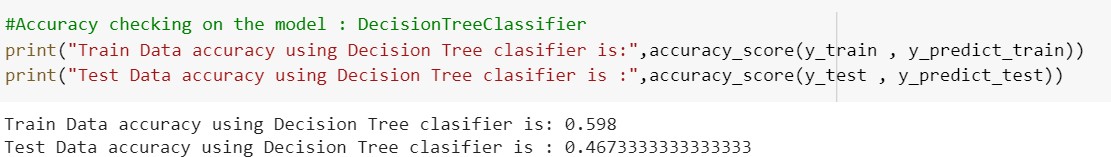
Step 15:

In the above step we have already fitted our model, now we will predict the model



Step 16:

Checking the accuracy of the model (Decision Tree)



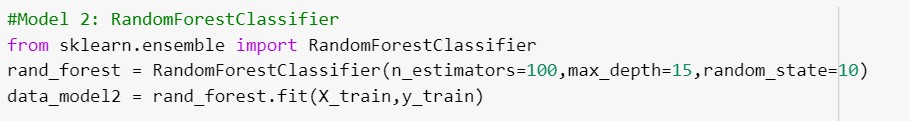
The train and test data accuracy we received through Decision Tree Classifier is 0.59 and

0.47 respectively.

# Task 2 (Classifier 2: Random Forest)

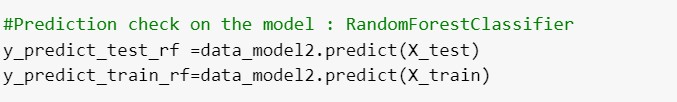
Step 17:

For random forest classifier, we will be using scikit-learn library provided in machine learning. We will be using RandomForestClassifier() function to fit our model which takes two parameters X\_train,y\_train.



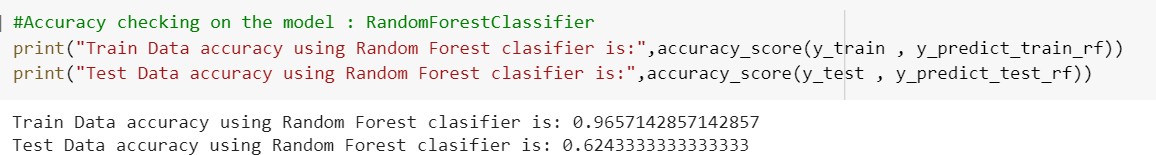
Step 18:

In the above step we have already fitted our model, now we will predict the model.



Step 19:

Checking the accuracy of the model (Random Forest)



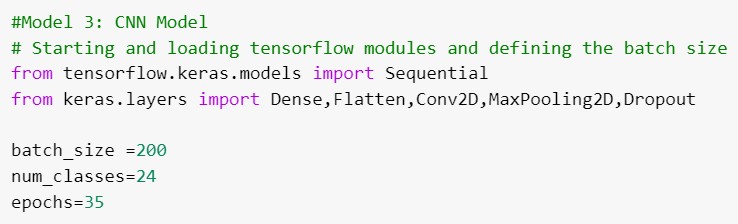
The train and test data accuracy we received through Random Forest Classifier is 0.96 and

0.62 respectively.

# Task 2 (Classifier 3: CNN Model)

Step 20:

For CNN model, we will be importing the important library from tensorFlow and make use of Keras, we will initially scale the image and resize and reshape it to set our hand gesture (28X28 pixel), later, we will create and the train our model.



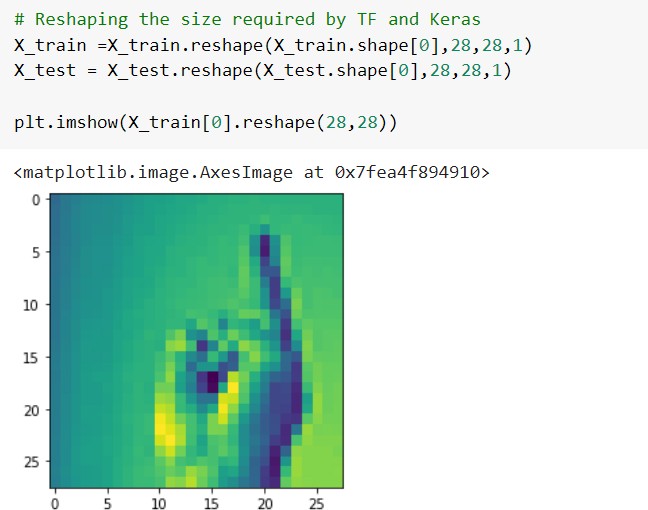


Step 21:

Checking the result for letter “D”, which should be



Output:



Step 22:

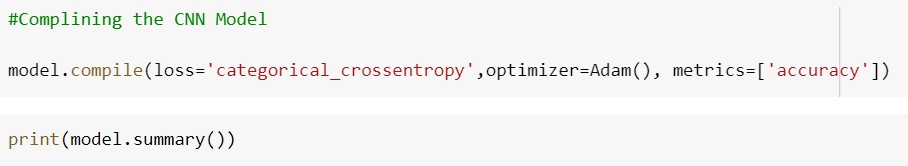
Here we will be creating the sequential model as we are dealing with pattern dataset, we will be using the layer pattern Convolution2D, MaxPooling2D which will help to define the input window size where we can define the height, width, colour of an images. We will also configure the inputs of shape (28, 28,1).

We will also add the top dense layer, so first we will flatten the three-dimensional output to one-dimensional and add another dense layer at the top, this will help us to feed the final tensor outcome and will result in completion of the model.

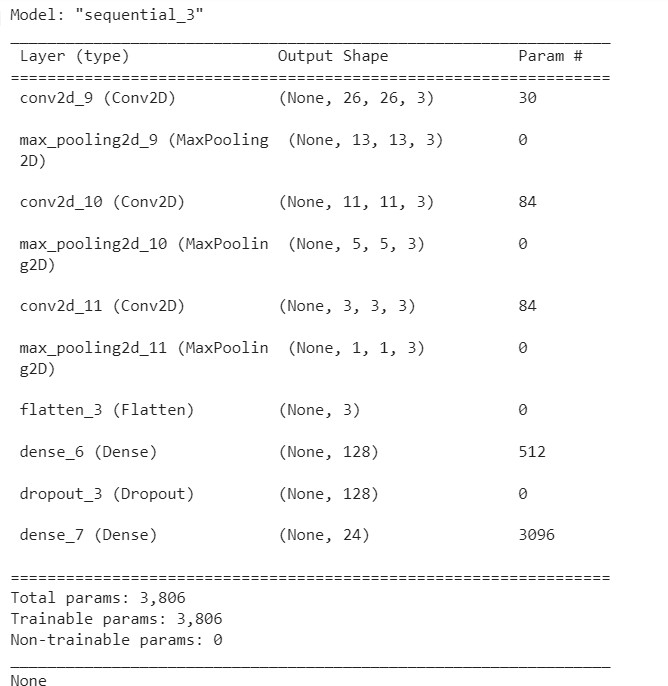


Step 23:

Compiling the CNN model and checking the summary of the complied model



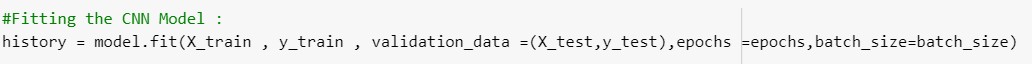
Output:



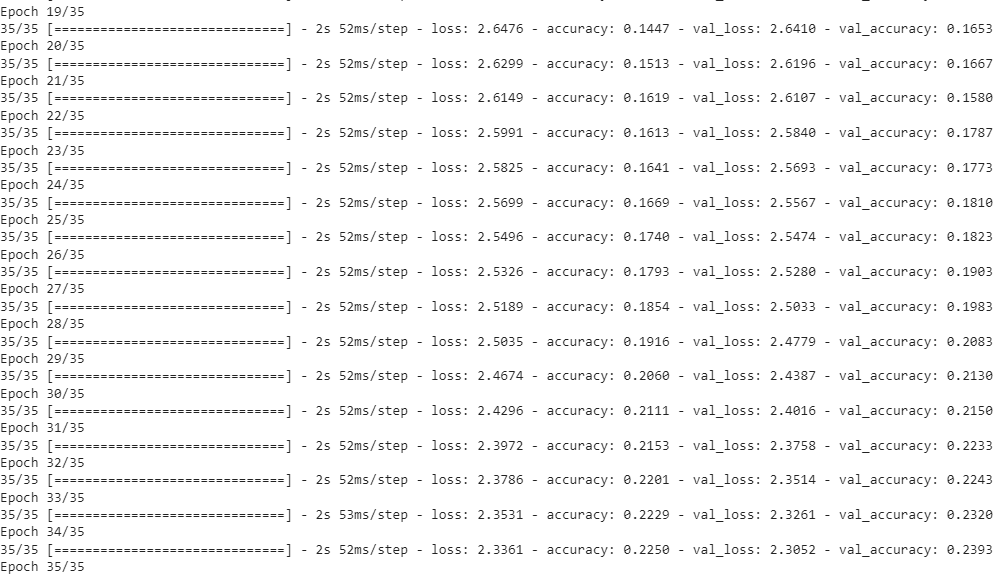
Above output, is the actual architecture of our model, where we can see that (3,3,3) outputs were been flatten from the vector of shape before arriving two-dimensional layer.

Step 24:

Fitting the CNN model

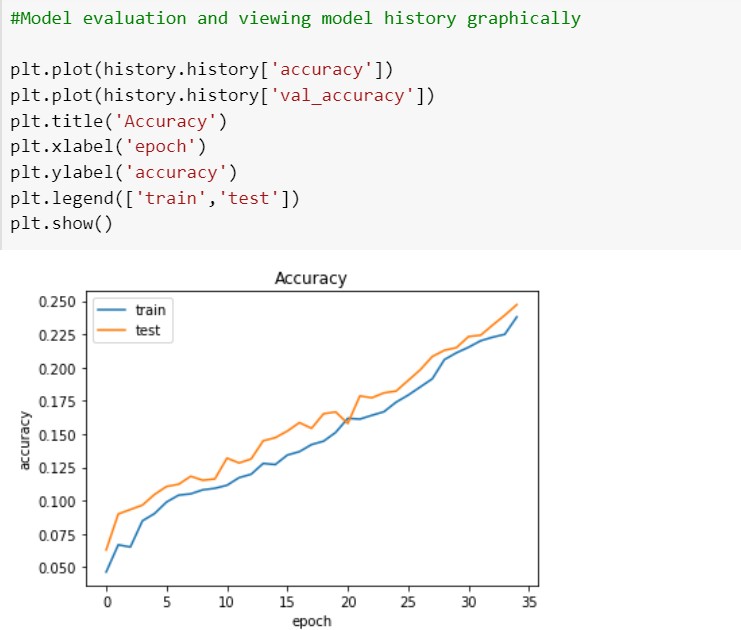


Output:



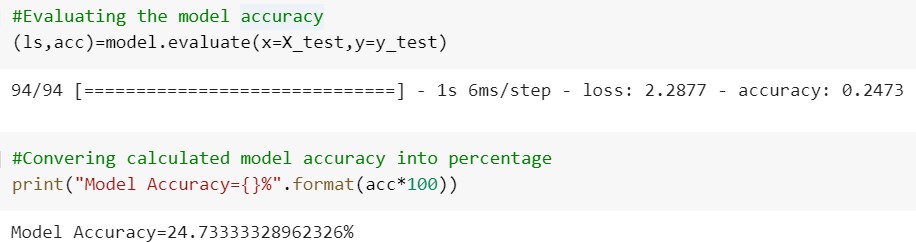
Step 25:

Evaluating the CNN Model



Step 26:

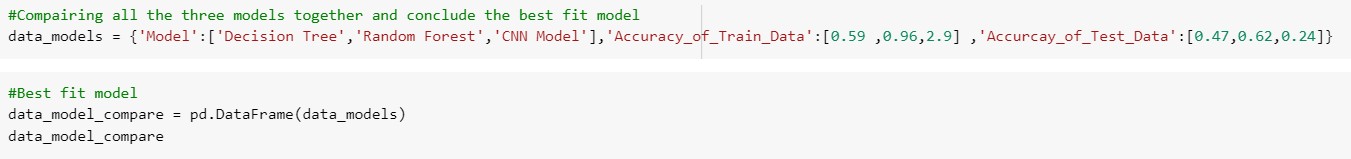
Evaluating the model accuracy



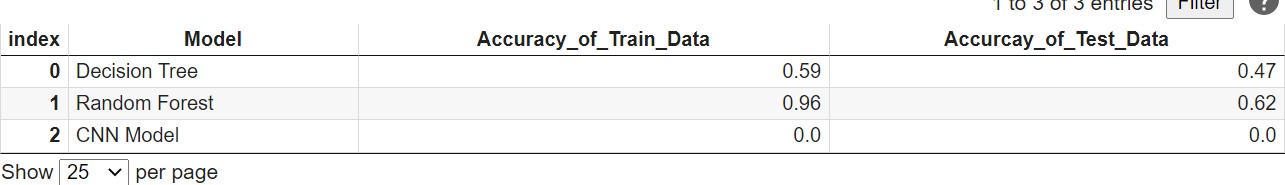
# Task 2 (Classifier 3: Model Comparison and Result)

Step 27:

In this step, we will compare all the three models that we have used initially and find the best fir model for our data set.



Output:



As per the above result, we can say that Random Forest is the best Model fit which has 96% of accuracy.

# Task 3



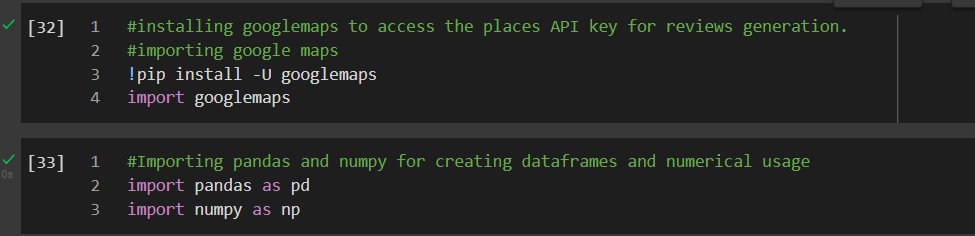
As a solution to the above task, We have created the Google-places API key and used ity to fetch the reviews of the companies like google, linkedin, amazon and tesla. Since the API is letting us fetch limited number of reviews(2 or 5), we have used the google-play-scraper library to fetch the reviews of the different companies from play store. After that, we have used the extracted data of reviews to create the WordCloud and find the most used words for a particular company. We have successfully completed the python code for this task in colab notebook.

All the pre-requites like the useful information document is available on moodle CA2 detail page.

Below are the steps involved in creating the WordCloud and fetching different company reviews.

Step 1:

We have imported the relevant libraries.

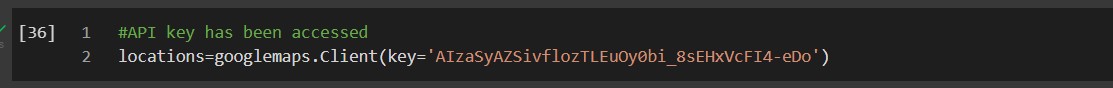


Google maps has been installed and imported to access the google\_places API for review generation.

Pandas for working with dataframes. Numpy for the numerical usage

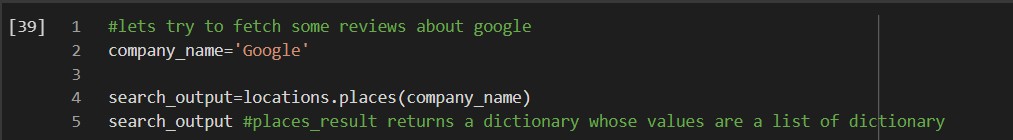
Step 2:

Accessing the locations using the API key.



Step 3:

Fetching the values for the company “google”



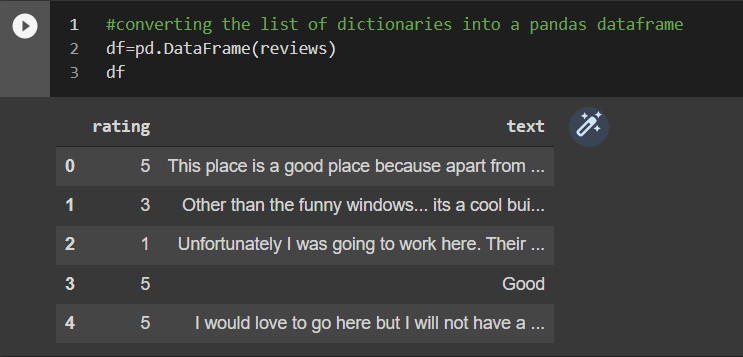
Step4:

Checking the number of reviews fetched.



Step 5:

Display the fetched reviews in a dataframe:

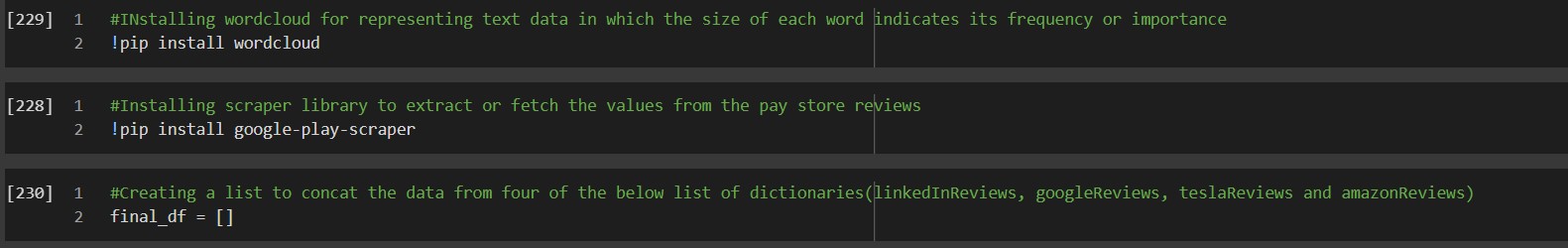


**Google api was not able to scrape more than 2 or 5 record of reviews. Hence Used Google- play-scraper library to fetch the reviews from play store for the four compay apps naming GOOGLE, LINKEDIN, TESLA and AMAZON.**

Below are the steps to fetch the company reviews with the use of google-play-scraper Library:

Step 1:

Installing the required libraries and creating a list namly final\_df to store all the compay reviews collectively.



Step 2:

Importing the google play scraper with some of its important functions sort and reviews\_all And fetching the data for the company linkedin using reviews\_all function.

from google\_play\_scraper import Sort, reviews\_all

linkedInReviews = reviews\_all('com.linkedin.android', sleep\_milliseconds=0,

lang='en', country='ie',

sort=Sort.MOST\_RELEVANT, # defaults to Sort.MOST\_RELEVANT count=1000,

)

Step 3:

Repeating the step 2 for other three companies as well

googleReviews = reviews\_all('com.google.android.googlequicksearchbox', sleep\_milliseconds=0,

lang='en', country='ie',

sort=Sort.MOST\_RELEVANT, # defaults to Sort.MOST\_RELEVANT

)

teslaReviews = reviews\_all('com.teslamotors.tesla', sleep\_milliseconds=0,

lang='en', country='ie',

sort=Sort.MOST\_RELEVANT, # defaults to Sort.MOST\_RELEVANT count=1000,

)

amazonReviews = reviews\_all('in.amazon.mShop.android.shopping', sleep\_milliseconds=0,

lang='en', country='ie',

sort=Sort.MOST\_RELEVANT, # defaults to Sort.MOST\_RELEVANT count=1000,

)

Step 4:

Created Dataframes for each company containing the data respectively and printing the 1000 reviews for each company. We will be transferring only 1000 records per company.

tesla\_dataframe = pd.DataFrame(teslaReviews[0:1000]) # print(tesla\_dataframe['content'])

google\_dataframe = pd.DataFrame(googleReviews[0:1000]) # print(google\_dataframe['content'])

linkedin\_dataframe = pd.DataFrame(linkedInReviews[0:1000]) # print(linkedin\_dataframe['content'])

amazon\_dataframe = pd.DataFrame(amazonReviews[0:1000]) # print(amazon\_dataframe['content'])

list1 = [tesla\_dataframe,google\_dataframe,linkedin\_dataframe,amazon\_dat aframe]

for val in list1:

print(val['content'])

output:

0

1

2

3

4

It's amazing after the update. I does have a f... There are issues with setting the time for sch... Omg this app is utterly awful since the update... Always takes way too long to "wake up" (minimu... Mapping Long Trips! I wish Tesla would let us ...

...

995 it used to be good and fast. the new version i... 996 Takes FOREVER to "wake up" which makes turning... 997 This new update is terrible! Put it back to th... 998 The new version is awful. Previously the app c... 999 Car update is great, solar is dreadful. I just... Name: content, Length: 1000, dtype: object

0

1

2

3

4

Easy to use and if you do the enquiry several ... Everything was fine until the last update on B... As mentioned by others posters, this app is dr... That Google assistant is the biggest pain in t... Every time there is update it jams my phone up...

...

995 I have a Pixel 2 xl. The latest update to the ... 996 Mixed feelings. The ads are insane! The search...

997 App is on my phone and now has decided

Step 5:

# Word Cloud Illustration

Here, we have imported wordcloud library for createing the wordcloud for each compay with 1000 reviews and matplotlib for diplaying the wordcloud.

Tokens has been used to store the individual word from each review.

from wordcloud import WordCloud, STOPWORDS import matplotlib.pyplot as plt

import pandas as pd

for value in list1: comment\_words = '' stopwords = set(STOPWORDS)

# iterate through the csv file final\_df[0][0]['content'] print("#############################################################"

)

for val in value['content']:

# typecaste each val to string val = str(val)

# split the value

tokens = []

tokens = val.split()

# Converts each token into lowercase for i in range(len(tokens)):

tokens[i] = tokens[i].lower()

comment\_words += " ".join(tokens)+" " wordcloud = WordCloud(width = 800, height = 800,

background\_color ='white', stopwords = stopwords,

min\_font\_size = 10).generate(comment\_words) # plot the WordCloud image

plt.figure(figsize = (8, 8), facecolor = None) plt.imshow(wordcloud)

plt.axis("off") plt.tight\_layout(pad = 0)

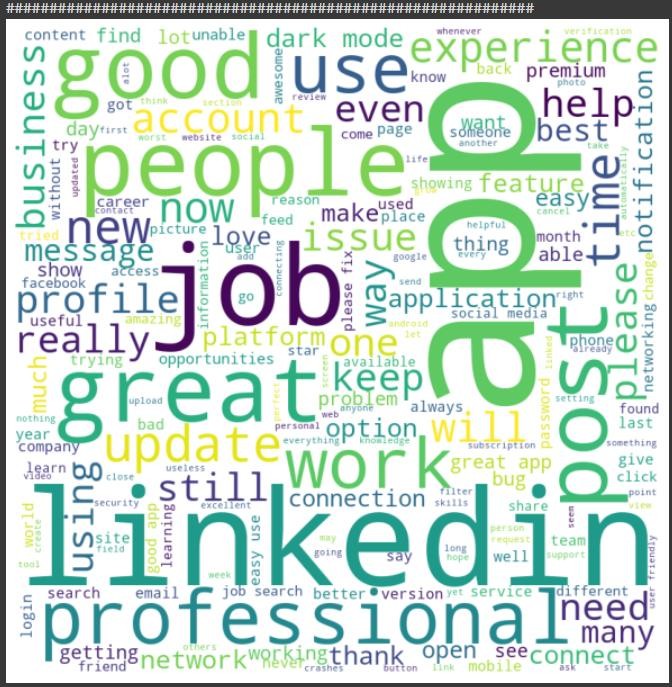
plt.show()

OUTPUT:

# Wordcloud for TESLA:

# Wordcloud for GOOGLE:

# Wordcloud for LINKEDIN:



# Word cloud for AMAZON:

**Step 6:**

# Text MINING

Importing the libraries used for text mining and tokenization: We have also used regex in the below code

import re import nltk

nltk.download('stopwords')

from nltk.corpus import stopwords

from nltk.stem.porter import PorterStemmer

We have used Tokenization which means a large string of review is being splitted into words and stored in a list to that we can pass this list to the frequency distribution function of nltk stem porter.

All the company reviews are converted into lowercase to get the correct frequency of each review.

Splitting each review and Storing in a list namely “Google\_corpus” and finally passing the it into the frequency distribution function of nltk library.

for company in list1: Google\_corpus = []

print("############################################################## ###")

for i in range(0, 1000): # Keeping only alphabets

company\_review = re.sub('[^a-zA-Z]', ' ', company['content'][i] ) # All aplhabets must be lower case only

company\_review = company\_review.lower() # Split review in different words company\_review = company\_review.split() # Remove non-

significant words .set is used when review is large like an article # Stemming - only keep root word

ps = PorterStemmer()

company\_review = [ps.stem(word) for word in company\_review if not w ord in set(stopwords.words('english'))]

# Convert the list review back to string #company\_review = ' '.join(company\_review) # Append the cleaned review to list corpus for indword in company\_review:

Google\_corpus.append(indword) word\_freq = nltk.FreqDist(Google\_corpus) wordcloud\_data = word\_freq.most\_common(10) print(wordcloud\_data)

Printing the frequency of top ten most used word for each company. OUTPUT:

Word frequency for tesla\_dataframe:

#################################################################

[('app', 915), ('car', 476), ('updat', 446), ('work', 353), ('time', 303), ('use', 265), ('version',

254), ('charg', 253), ('new', 237), ('connect', 206)] #################################################################

Word frequency for google\_dataframe:

#################################################################

[('googl', 1054), ('app', 665), ('updat', 649), ('search', 395), ('work', 393), ('phone', 336),

('use', 319), ('fix', 251), ('time', 244), ('get', 244)]

Word frequency for linkedin\_dataframe:

#################################################################

[('app', 618), ('linkedin', 271), ('use', 234), ('job', 231), ('great', 173), ('work', 147), ('get',

145), ('connect', 141), ('good', 131), ('peopl', 129)]

Word frequency for amazon\_dataframe:

#################################################################

[('amazon', 671), ('app', 662), ('order', 375), ('product', 315), ('time', 303), ('custom', 297),

('deliveri', 261), ('servic', 250), ('day', 219), ('use', 189)]

In conclusion, we are able to make the visual representations of words that give greater prominence to words that appear more frequently for each company (i.e tesla, amazon, google, and linkedin)

**THANKYOU**

