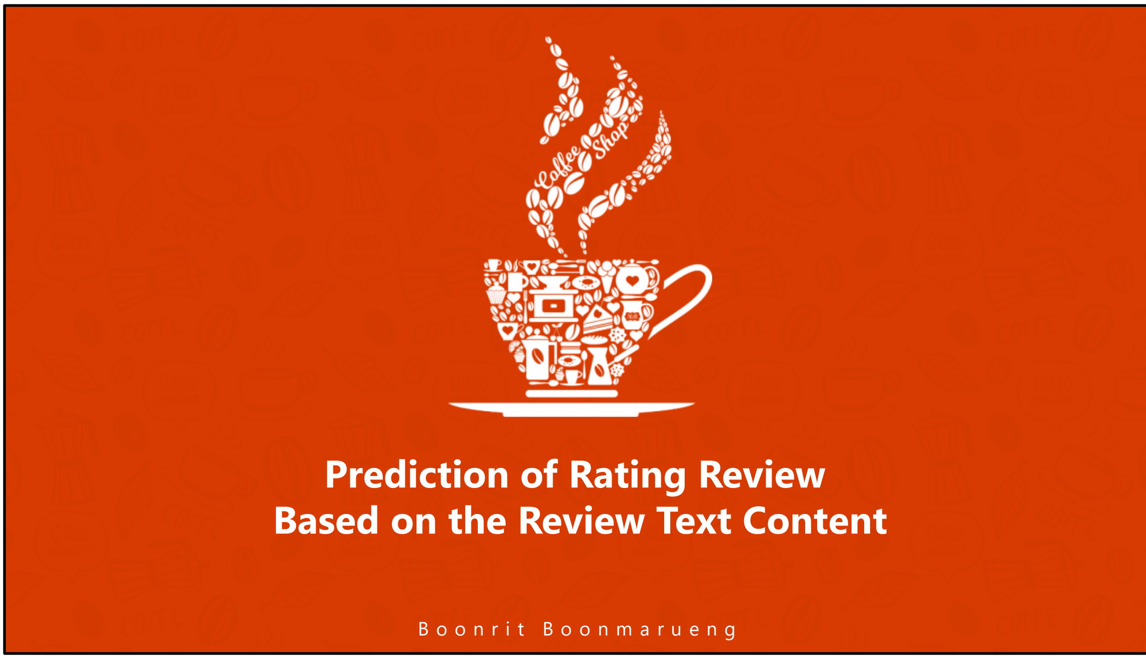


Good evening teachers and classmates. My name Oat. It’s a pleasure to have you with us in my presentation today.
The topic of presentation is

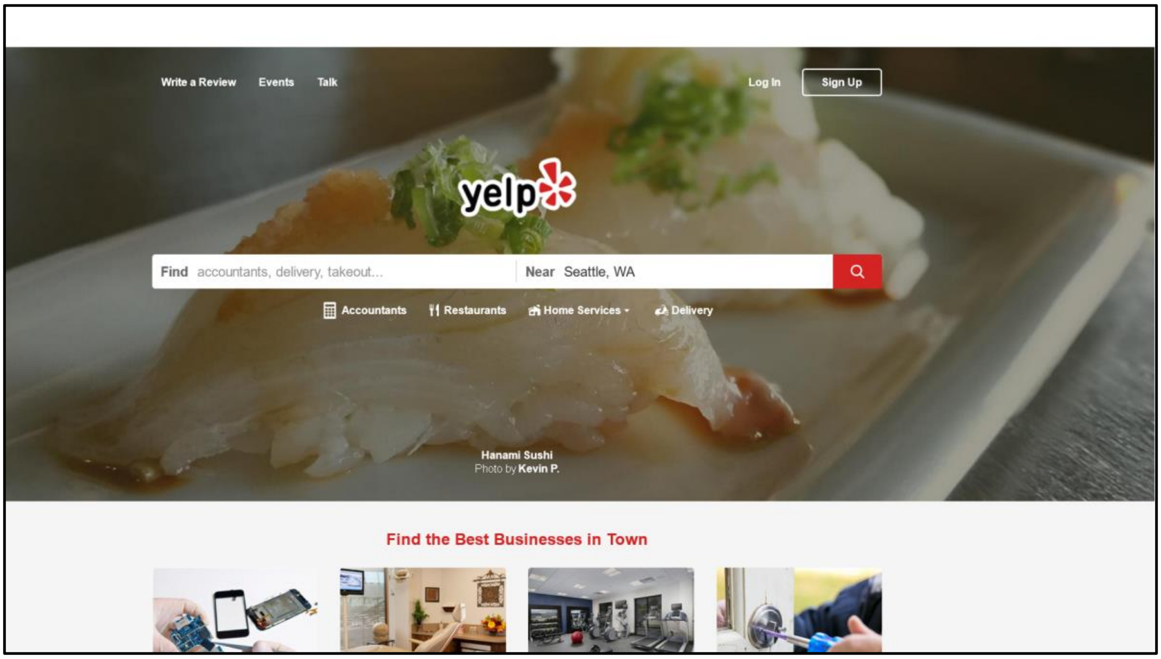
The Prediction of Rating Review Base on the **Re**view text **Con**tent.

Nowadays, E-**com**merce is **dev**eloping fast, as a result of product or service reviews have grown immediately on the online **plat**form. Some of you may be **fam**iliar with these applications. …. And …

So, the problem is many reviews make it **dif**ficult for businesses to automatcly classify them into different semantic orientations (positive, negative, and neutral), or rating the customer’s reviews.

However, these reviews is great value in re**flect**ing their customer’s **op(**อบ)inions. And customer needs.

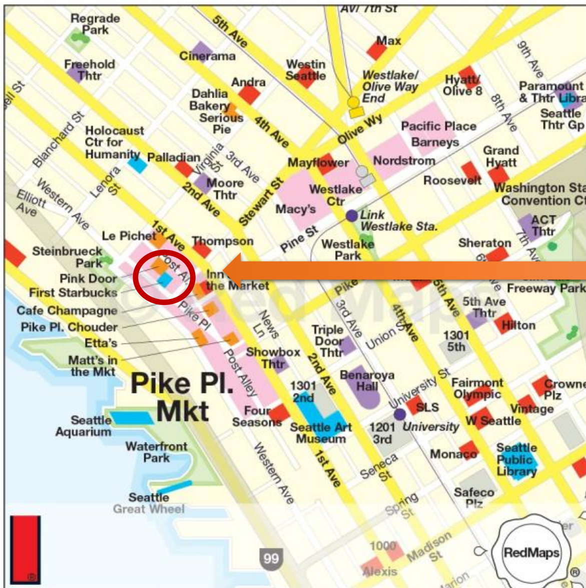
Therefore, many companies are paying their at**ten**tion in text analysis.



Today, we are going analyze the customer reviews that I chose form Yelp.
This platform is a very popular in USA by writing a review for products/services and giving their satisfaction in range 1-5 stars.

Some of you here may be able to guess from the first page of my presentation that have the figure of coffee cup.
Yes, Today, I will present the text classification about the review of Coffee Shop.

What coffee brand/coffee shop that is very popular today? Both in Thailand and outside
Of course, the first brand that very well know is the Starbucks. Shop



SEATTLE by Red Maps City Guides

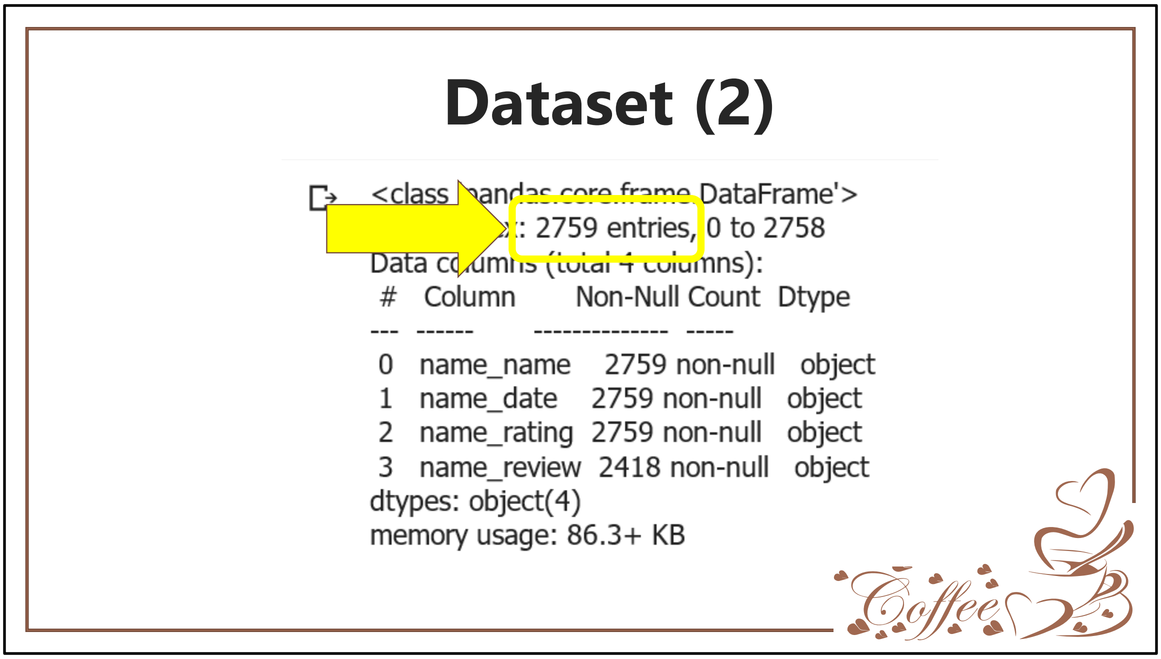
5

The data that I used in this project came from the review of First Starbucks coffee shop located in the pike place market.
At Seattle, USA

Dataset (1)

	name_name	name_date	name_rating	name_review
0	Sarah B.	7/18/2006	1 star rating	NaN
1	Joshua M.	2006-11-11 00:00:00	1 star rating	I'm not reviewing this particular one per se, ...
2	Matthew C.	2006-03-12 00:00:00	3 star rating	This is the first Starbucks ever! I can't beli...
3	Erik T.	12/26/2006	3 star rating	Definitely a tourist trap, and definitely wort...
4	Lorena R.	2/13/2007	3 star rating	Ok so I'm not a Starbucks person- no triple sh...

This is one example of customer's review on the Yelp website.
I extracted every review from this website, then construct datafFrame with columns look like this figure:
Consisting of name, date, **rat**ings and review text.



At the beginning, the number of reviews to be analyzed is approximately two thousand and seven hundred rows.

Cleaning

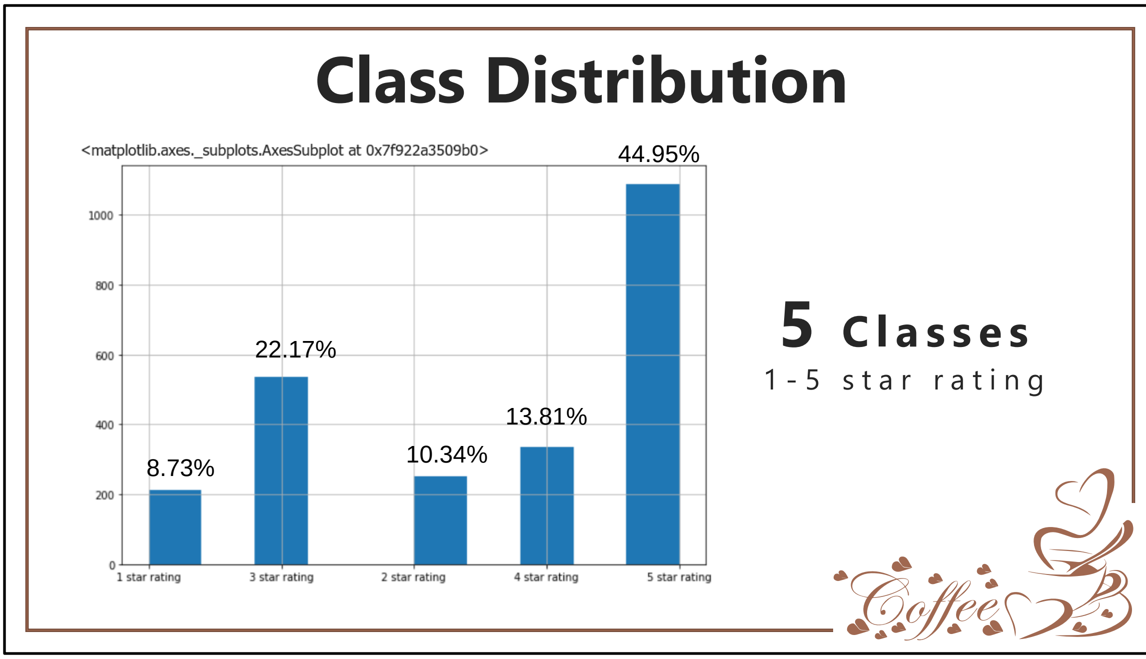
	name_name	name_date	name_rating	name_review
0	Sarah B.	7/18/2006	1 star rating	NaN
1	Joshua M.	2006-11-11 00:00:00	1 star rating	I'm not reviewing this particular one per se, ...
2	Matthew C.	2006-03-12 00:00:00	3 star rating	This is the first Starbucks ever! I can't beli...
3	Erik T.	12/26/2006	3 star rating	Definitely a tourist trap, and definitely wort...
4	Lorena R.	2/13/2007	3 star rating	Ok so I'm not a Starbucks person- no triple sh...

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2759 entries, 0 to 2758
Data columns (total 4 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   name_name    2759 non-null   object
 1   name_date    2759 non-null   object
 2   name_rating  2759 non-null   object
 3   name_review  2418 non-null   object
dtypes: object(4)
memory usage: 86.3+ KB
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2418 entries, 0 to 2417
Data columns (total 5 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   index        2418 non-null   int64
 1   name_name    2418 non-null   object
 2   name_date    2418 non-null   object
 3   name_rating  2418 non-null   object
 4   name_review  2418 non-null   object
dtypes: int64(1), object(4)
memory usage: 94.6+ KB
```

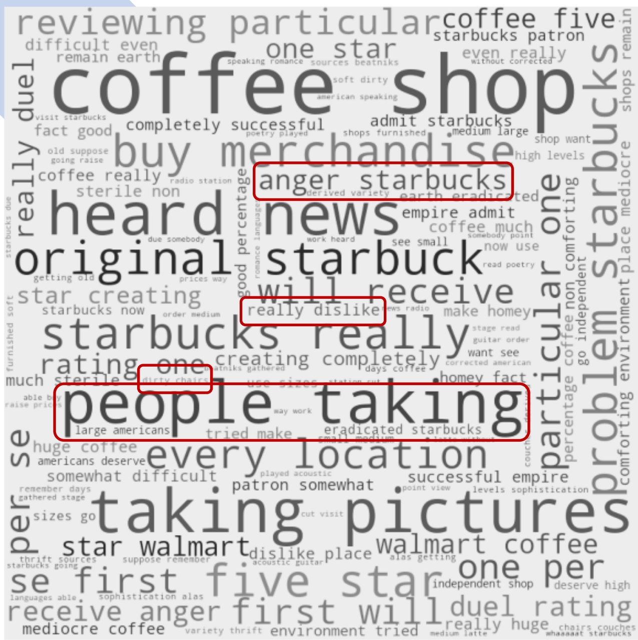
Coffee

But from the initial inspection, I found out that some reviews that doesn't have a review text written, but they still give the rating
To solve this problem, I filter out these reviews that make the number of rows to be reduced to around two thousand and four hundred rows



For class Distribution.
Review ratings are distinguished in the range of 1-5 stars.
With most of the distribution at 5 stars
Followed by 3 stars
And finally 1 star

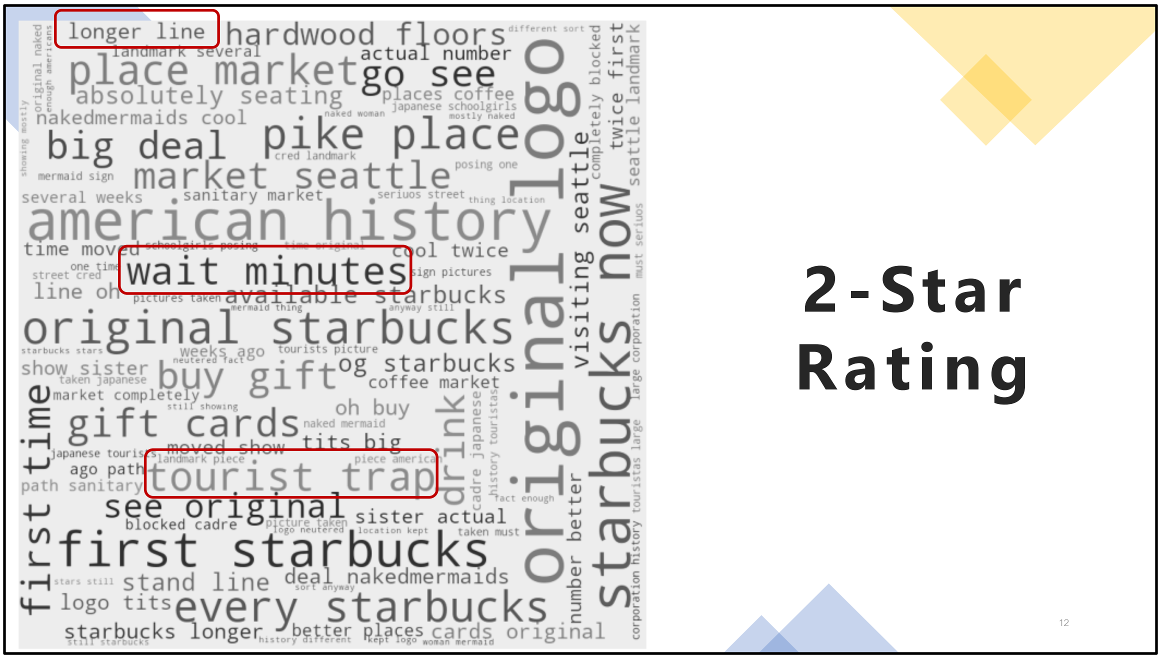
Before making a model, I tried to make Word Cloud for each class.
To see the word or sentence that they highly mention.



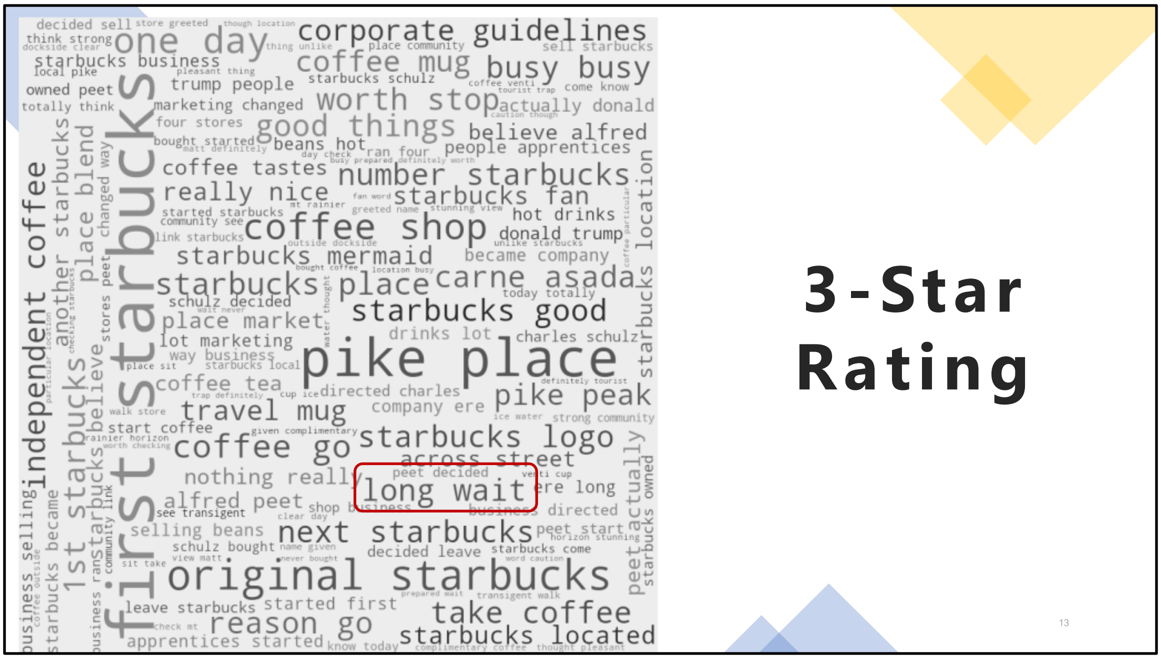
1-Star Rating

11

By the first group
People Talking I/ dirty chairs



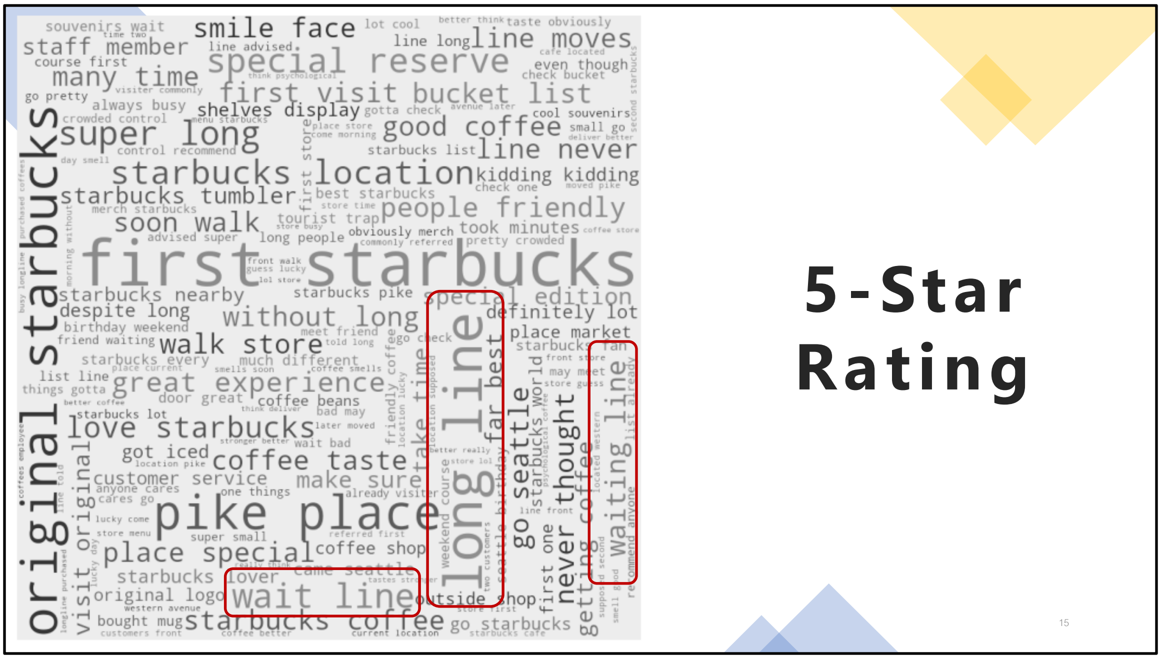
The next group talked about long queues and very long lines.
Including mention that the Starbucks is a trap



Group 3 also talks about long queues.

4-Star Rating

14



Again in group of 5 stars, also mention about.....

In summary, one of the most mention about problems from Group 1-5 is the long q and long waiting time that customers have to experience here

Advantages of Word Clouds :

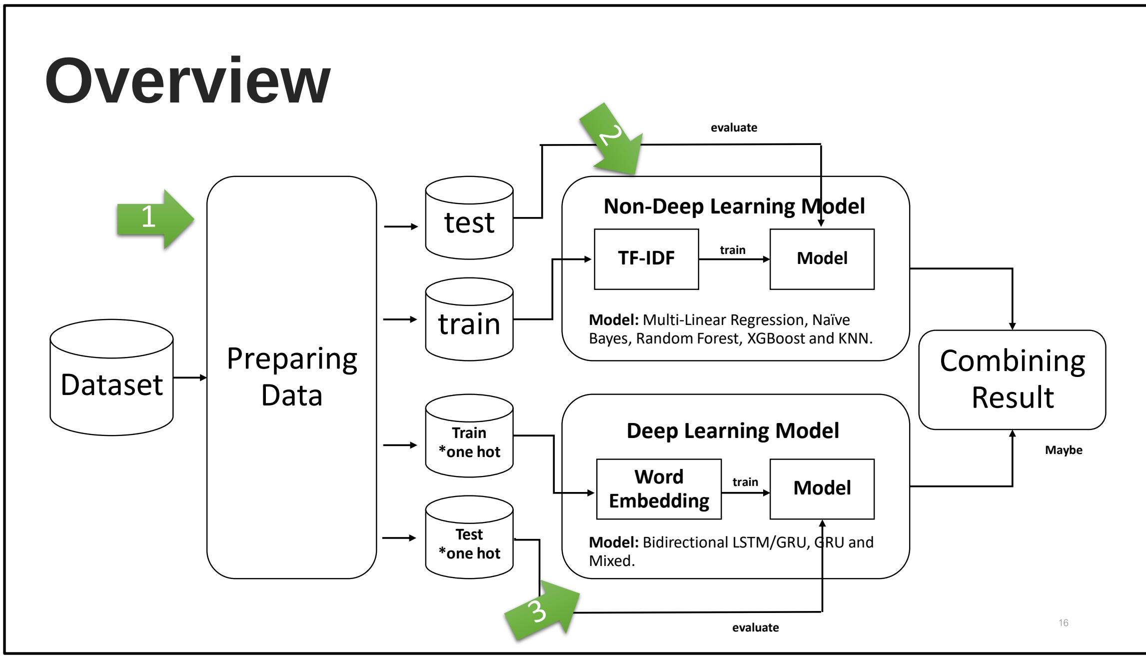
1. Analyzing customer and employee feedback.

2. Identifying new SEO keywords to target.

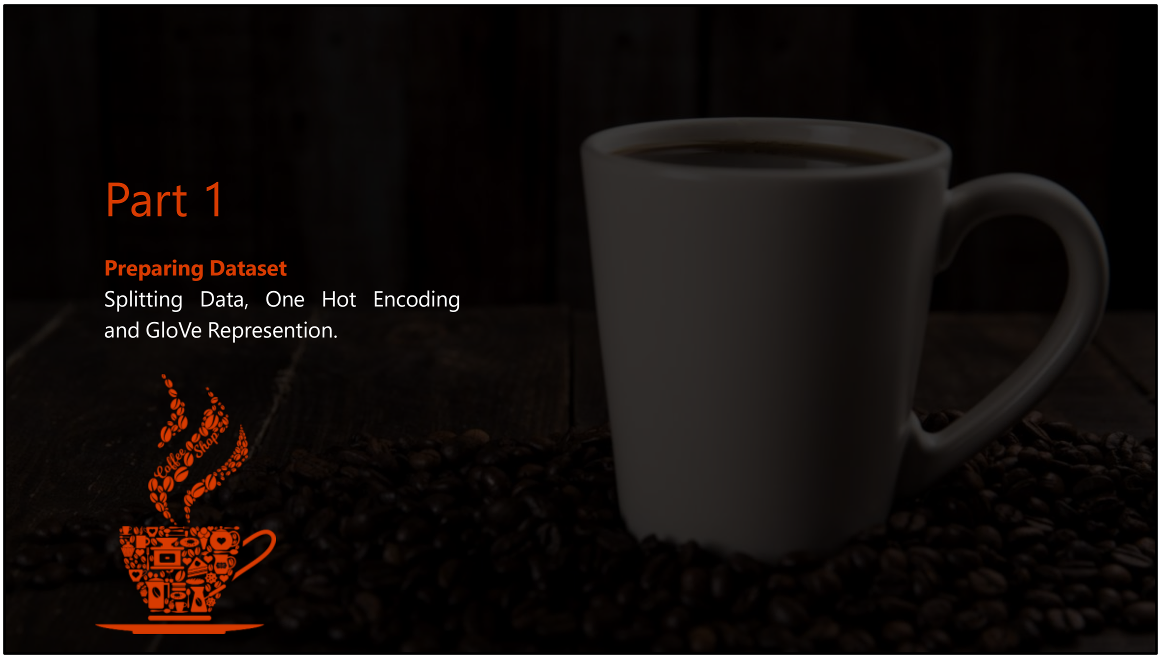
Drawbacks of Word Clouds :

1. Word Clouds are not perfect for every situation.
2. Data should be optimized for context.

Ref: https://www.geeksforgeeks.org/generating-word-cloud-python/



Okay, Let's move in to training and testing the models.
This slide shows an overview of my work. Which is divided into 3 main parts 1. Preparation data 2. Creating a non-DEF model consists of .....
And finally, the model ...



First Part, Preparing dataset for the model.
Such as

Splitting Data

```
X1 = all_reviews['name_review']
y1 = all_reviews['name_rating']
review_train1, review_test1, label_train1, label_test1 = train_test_split(X1, y1, test_size=0.5, random_state=38)
```

By training/ testing models I divided the data into 50 50% and fixed the random state at 38 to avoid the Bias from dividing the data.

One Hot Encoding

	name_name	name_date	name_rating	name_review
0	Sarah B.	7/18/2006	1 star rating	NaN
1	Joshua M.	2006-11-11 00:00:00	1 star rating	I'm not reviewing this particular one per se, ...
2	Matthew C.	2006-03-12 00:00:00	3 star rating	This is the first Starbucks ever! I can't beli...
3	Erik T.	12/26/2006	3 star rating	Definitely a tourist trap, and definitely wort...
4	Lorena R.	2/13/2007	3 star rating	Ok so I'm not a Starbucks person- no triple sh...

```
class_names = ['1 star rating', '2 star rating', '3 star rating', '4 star rating', '5 star rating']
```

```
y = train[class_names].values
y
```

```
array([[0, 0, 0, 0, 1],
       [0, 0, 1, 0, 0],
       [0, 0, 1, 0, 0],
       ...,
       [0, 0, 0, 0, 1],
       [0, 1, 0, 0, 0],
       [0, 0, 0, 0, 1]], dtype=uint8)
```

Next step, As we know that Deep Learning models need the numeric input. I transformed the rating from category in to 5-class Binary as show in figure below.

https://www.quora.com/How-is-GloVe-different-from-word2vec

GloVe

Embedding File:	glove.6B.100d.txt
The number of word:	400,000 words
Embedding size:	100

```
the -0.038194 -0.24487 0.72812 -0.39961 0.083172 0.043953 -0.39141 0.3344 -0.57545 0.087459 0.28787 -0.06731 0.30906 -0.26384
, -0.10767 0.11053 0.59812 -0.54361 0.67396 0.10663 0.038867 0.35481 0.06351 -0.094189 0.15786 -0.81665 0.14172 0.21939 0.5850
. -0.33979 0.20941 0.46348 -0.64792 -0.38377 0.038034 0.17127 0.15978 0.46619 -0.019169 0.41479 -0.34349 0.26872 0.04464 0.421
of -0.1529 -0.24279 0.89837 0.16996 0.53516 0.48784 -0.58826 -0.17982 -1.3581 0.42541 0.15377 0.24215 0.13474 0.41193 0.67043
to -0.1897 0.050024 0.19084 -0.049184 -0.089737 0.21006 -0.54952 0.098377 -0.20135 0.34241 -0.092677 0.161 -0.13268 -0.2816 0.
and -0.071953 0.23127 0.023731 -0.50638 0.33923 0.1959 -0.32943 0.18364 -0.18057 0.28963 0.20448 -0.5496 0.27399 0.58327 0.204
in 0.085703 -0.22201 0.16569 0.13373 0.38239 0.35401 0.01287 0.22461 -0.43817 0.50164 -0.35874 -0.34983 0.055156 0.69648 -0.17
a -0.27086 0.044006 -0.02026 -0.17395 0.6444 0.71213 0.3551 0.47138 -0.29637 0.54427 -0.72294 -0.0047612 0.040611 0.043236 0.2
" -0.30457 -0.23645 0.17576 -0.72854 -0.28343 -0.2564 0.26587 0.025309 -0.074775 -0.3766 -0.057774 0.12159 0.34384 0.41928 -0.
's 0.58854 -0.2025 0.73479 -0.68338 -0.19675 -0.1802 -0.39177 0.34172 -0.60561 0.63816 -0.26695 0.36486 -0.40379 -0.1134 -0.58
for -0.14401 0.32554 0.14257 -0.099227 0.72536 0.19321 -0.24188 0.20223 -0.89599 0.15215 0.035963 -0.59513 -0.051635 -0.014428
- -1.2557 0.61036 0.56793 -0.96596 -0.45249 -0.071696 0.57122 -0.31292 -0.43814 0.90622 0.06961 -0.053104 0.25029 0.27841 0.77
that -0.093337 0.19043 0.68457 -0.41548 -0.22777 -0.11803 -0.095434 0.19613 0.17785 -0.020244 -0.055409 0.33867 0.79396 -0.047
on -0.21863 -0.42664 0.5196 0.0043103 0.58045 -0.10873 -0.37726 0.4566 -0.60627 -0.075773 0.11306 0.17703 0.1605 0.074514 0.63
is -0.54264 0.41476 1.0322 -0.40244 0.46691 0.21816 -0.074864 0.47332 0.080996 -0.22079 -0.12808 -0.1144 0.50891 0.11568 0.028
```

To construct the embedding vector, I used Glove Corpus that has 4 hundred thousand words and 100 embedding size/features to represent the words.

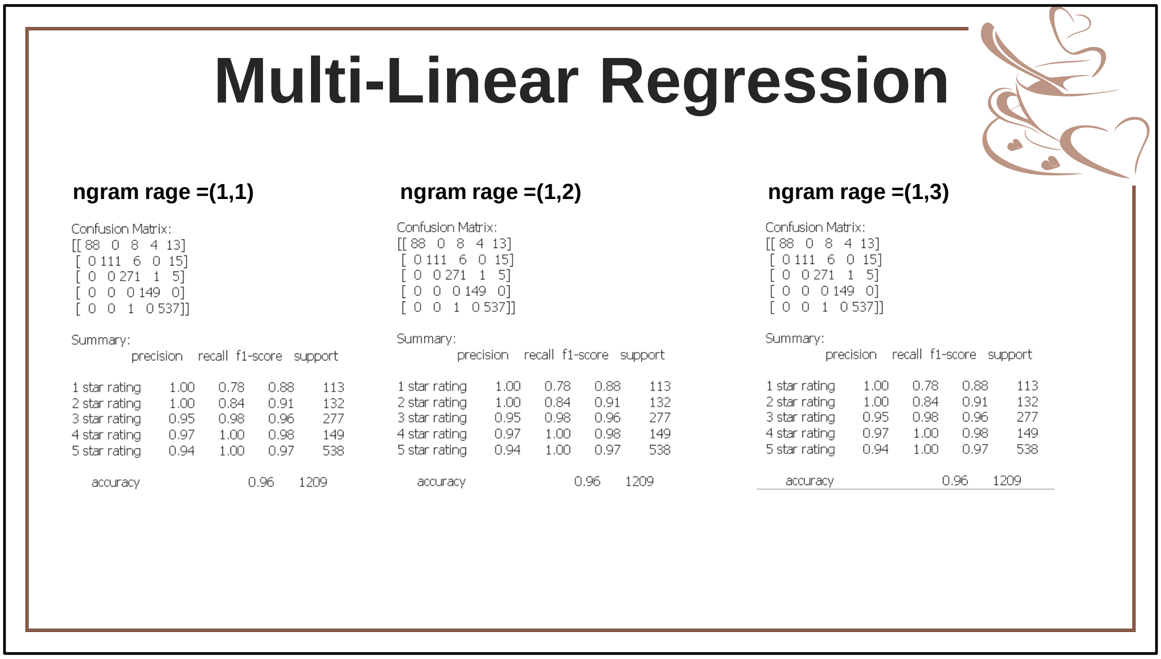

Part 2
Non-Deep Learning Model
Multi-Linear Regression, Naïve Bayes,
Random Forest, XGBoost and KNN

Creating Sklearn Pipeline

1
2

```
pipeline = Pipeline([
    ('Tf-Idf', TfidfVectorizer(ngram_range=(1,3), analyzer=text_process)),
    ('classifier', LogisticRegression(solver='newton-cg', multi_class='multinomial'))
])
```

I use pipeline tools from the sklearn library. The pipeline structure consists of 2 parts.
First. Creating the TFIDF matrix, which has ngrame parameters and text process such as cut, lower case, and removing special character.
And the second part The model that will be used to create the classifier.



Multi-Linear Regression

ngram rage =(1,1)

Confusion Matrix:
```
[[ 88   0   8   4  13]
 [  0 111   6   0  15]
 [  0   0 271   1   5]
 [  0   0   0 149   0]
 [  0   0   1   0 537]]
```

Summary:

	precision	recall	f1-score	support
1 star rating	1.00	0.78	0.88	113
2 star rating	1.00	0.84	0.91	132
3 star rating	0.95	0.98	0.96	277
4 star rating	0.97	1.00	0.98	149
5 star rating	0.94	1.00	0.97	538
accuracy			0.96	1209

ngram rage =(1,2)

Confusion Matrix:
```
[[ 88   0   8   4  13]
 [  0 111   6   0  15]
 [  0   0 271   1   5]
 [  0   0   0 149   0]
 [  0   0   1   0 537]]
```

Summary:

	precision	recall	f1-score	support
1 star rating	1.00	0.78	0.88	113
2 star rating	1.00	0.84	0.91	132
3 star rating	0.95	0.98	0.96	277
4 star rating	0.97	1.00	0.98	149
5 star rating	0.94	1.00	0.97	538
accuracy			0.96	1209

ngram rage =(1,3)

Confusion Matrix:
```
[[ 88   0   8   4  13]
 [  0 111   6   0  15]
 [  0   0 271   1   5]
 [  0   0   0 149   0]
 [  0   0   1   0 537]]
```

Summary:

	precision	recall	f1-score	support
1 star rating	1.00	0.78	0.88	113
2 star rating	1.00	0.84	0.91	132
3 star rating	0.95	0.98	0.96	277
4 star rating	0.97	1.00	0.98	149
5 star rating	0.94	1.00	0.97	538
accuracy			0.96	1209

At the beginning, with the creation of

the MLR model, I started by changing the parameters of ngram from 1 – 3 nagram.
I found that using only 1

ngram only gives a high acc value at 96
And if we considered the error, it is found
Most errors occurred at class

1-3 stars.

Naïve Bayes

ngram rage =(1,1)

```
Confusion Matrix:
[[ 88   0   2   2  21]
 [  0 111   1   0  20]
 [  0   0 264   1  12]
 [  0   0   0 136  13]
 [  0   0   1   0 537]]
```

Summary:

	precision	recall	f1-score	support
1 star rating	1.00	0.78	0.88	113
2 star rating	1.00	0.84	0.91	132
3 star rating	0.99	0.95	0.97	277
4 star rating	0.98	0.91	0.94	149
5 star rating	0.89	1.00	0.94	538
accuracy			0.94	1209

ngram rage =(1,2)

```
Confusion Matrix:
[[ 88   0   2   2  21]
 [  0 111   1   0  20]
 [  0   0 264   1  12]
 [  0   0   0 136  13]
 [  0   0   1   0 537]]
```

Summary:

	precision	recall	f1-score	support
1 star rating	1.00	0.78	0.88	113
2 star rating	1.00	0.84	0.91	132
3 star rating	0.99	0.95	0.97	277
4 star rating	0.98	0.91	0.94	149
5 star rating	0.89	1.00	0.94	538
accuracy			0.94	1209

ngram rage =(1,3)

```
Confusion Matrix:
[[ 88   0   2   2  21]
 [  0 111   1   0  20]
 [  0   0 264   1  12]
 [  0   0   0 136  13]
 [  0   0   1   0 537]]
```

Summary:

	precision	recall	f1-score	support
1 star rating	1.00	0.78	0.88	113
2 star rating	1.00	0.84	0.91	132
3 star rating	0.99	0.95	0.97	277
4 star rating	0.98	0.91	0.94	149
5 star rating	0.89	1.00	0.94	538
accuracy			0.94	1209

NB, results in accuracy less than MLR.
However, it was found that increasing the number of words in the analysis did not increase any accuracy.
And most false predictions from 1-5 stars are predictd the results as Class 5 Rating

Random Forest

ngram rage =(1,1)

```
Confusion Matrix:
[[ 90   1  15   2   5]
 [  0 111  10   1  10]
 [  0   0 275   1   1]
 [  0   0   0 149   0]
 [  0   0   3   1 534]]

Summary:
              precision    recall  f1-score   support

1 star rating      1.00      0.80      0.89       113
2 star rating      0.99      0.84      0.91       132
3 star rating      0.91      0.99      0.95       277
4 star rating      0.97      1.00      0.98       149
5 star rating      0.97      0.99      0.98       538

     accuracy                          0.96      1209
```

ngram rage =(1,2)

```
Confusion Matrix:
[[ 89   0  14   3   7]
 [  0 111   9   3   9]
 [  0   0 274   1   2]
 [  0   0   0 149   0]
 [  0   0   3   1 534]]

Summary:
              precision    recall  f1-score   support

1 star rating      1.00      0.79      0.88       113
2 star rating      1.00      0.84      0.91       132
3 star rating      0.91      0.99      0.95       277
4 star rating      0.95      1.00      0.97       149
5 star rating      0.97      0.99      0.98       538

     accuracy                          0.96      1209
```

ngram rage =(1,3)

```
Confusion Matrix:
[[ 88   2  15   2   6]
 [  0 111   8   1  12]
 [  0   0 275   1   1]
 [  0   0   0 149   0]
 [  0   0   2   1 535]]

Summary:
              precision    recall  f1-score   support

1 star rating      1.00      0.78      0.88       113
2 star rating      0.98      0.84      0.91       132
3 star rating      0.92      0.99      0.95       277
4 star rating      0.97      1.00      0.98       149
5 star rating      0.97      0.99      0.98       538

     accuracy                          0.96      1209
```

For RF that uses the concept of voting of multiple DT
I found that the result is very high acc at 96
And most false prediction occurred in1-2 stats

XGBoost

ngram rage =(1,1)

Confusion Matrix:

```
[[ 92   4   6   5   6]
 [  1 112   6   2  11]
 [  0   0 270   1   6]
 [  0   0   0 149   0]
 [  0   1   3   0 534]]
```

Summary:

	precision	recall	f1-score	support
1 star rating	0.99	0.81	0.89	113
2 star rating	0.96	0.85	0.90	132
3 star rating	0.95	0.97	0.96	277
4 star rating	0.95	1.00	0.97	149
5 star rating	0.96	0.99	0.98	538
accuracy			0.96	1209

ngram rage =(1,2)

Confusion Matrix:

```
[[ 92   4   6   5   6]
 [  1 112   6   2  11]
 [  0   0 270   1   6]
 [  0   0   0 149   0]
 [  0   1   3   0 534]]
```

Summary:

	precision	recall	f1-score	support
1 star rating	0.99	0.81	0.89	113
2 star rating	0.96	0.85	0.90	132
3 star rating	0.95	0.97	0.96	277
4 star rating	0.95	1.00	0.97	149
5 star rating	0.96	0.99	0.98	538
accuracy			0.96	1209

ngram rage =(1,3)

Confusion Matrix:

```
[[ 92   4   6   5   6]
 [  1 112   6   2  11]
 [  0   0 270   1   6]
 [  0   0   0 149   0]
 [  0   1   3   0 534]]
```

Summary:

	precision	recall	f1-score	support
1 star rating	0.99	0.81	0.89	113
2 star rating	0.96	0.85	0.90	132
3 star rating	0.95	0.97	0.96	277
4 star rating	0.95	1.00	0.97	149
5 star rating	0.96	0.99	0.98	538
accuracy			0.96	1209

XGBoost too, has a high acc value of 96
And most error in 1-3 stars

K-Nearest Neighbor

```
array([[ 89,   3,   4,   1,  16],
       [  1, 110,   3,   0,  18],
       [  0,   0, 260,   1,  16],
       [  0,   0,   8, 137,   4],
       [  0,   0,   1,   0, 537]])
```

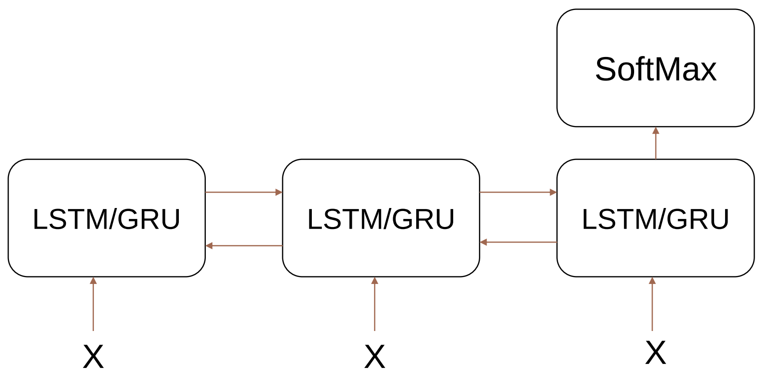
```
acc = (89+110+260+137+537)*100/len(y_true)
acc = np.round(acc,3)
print("ACC:  "+str(acc)+"%")
```

```
ACC: 93.714%
```

For the final model, the KNN that I created the TDIDF for each review and used DF Cosin Similarity to
Measure the similarity of each review. It gave the result only 93%
And most of the errors that occur are predicted as 5 stars class

Alright, Let’s move on the third part of my presentation.

Overview of Bidirectional



`embedding_matrix (2300, 100)`

I chose the Biditectional because using bidirectional will create relationship from run inputs in two ways, both one from past to future and one from future to past.
It gonna help to capture the retation of each word and give me high chance to get best result.

So, I designed Structure of my work as this figure: 3 consecutive layers of LSTM/GRU and Finally use Softmax to classify into 5 class 1-5 stars

Bidirectional LSTM/LSTM

LSTM-LSTM-LSTM

```
array([[ 89,   6,   5,   7,   6],
       [  0, 111,   5,   8,   8],
       [  0,   0, 264,  10,   3],
       [  0,   0,   3, 146,   0],
       [  1,   0,   0,   5, 532]])
```

```
1209/1209 [==============================] - 2s 2ms/step
1209/1209 [==============================] - 0s 169us/step
```

	precision	recall	f1-score	support
0	0.96	0.80	0.87	113
1	0.99	0.83	0.90	132
2	0.88	0.97	0.92	277
3	0.92	0.90	0.91	149
4	0.96	0.99	0.98	538
accuracy			0.94	1209

```
Epoch 30/30
967/967 [==============================] - 23s 24ms/step - loss: 0.1389 - accuracy: 0.9667 - val_loss: 0.0908 - val_accuracy: 0.9917
<keras.callbacks.callbacks.History at 0x7f7cb0051cc0>
```

In the first case, choose 3 consecutive layers of LSTM.
It gave the acc at 94
But the surprising point is that the accuracy of validation acc is greater than training and testing acc.
Maybe validation set consists of "easier" examples than the training and testing set.

Bidirectional GRU/GRU

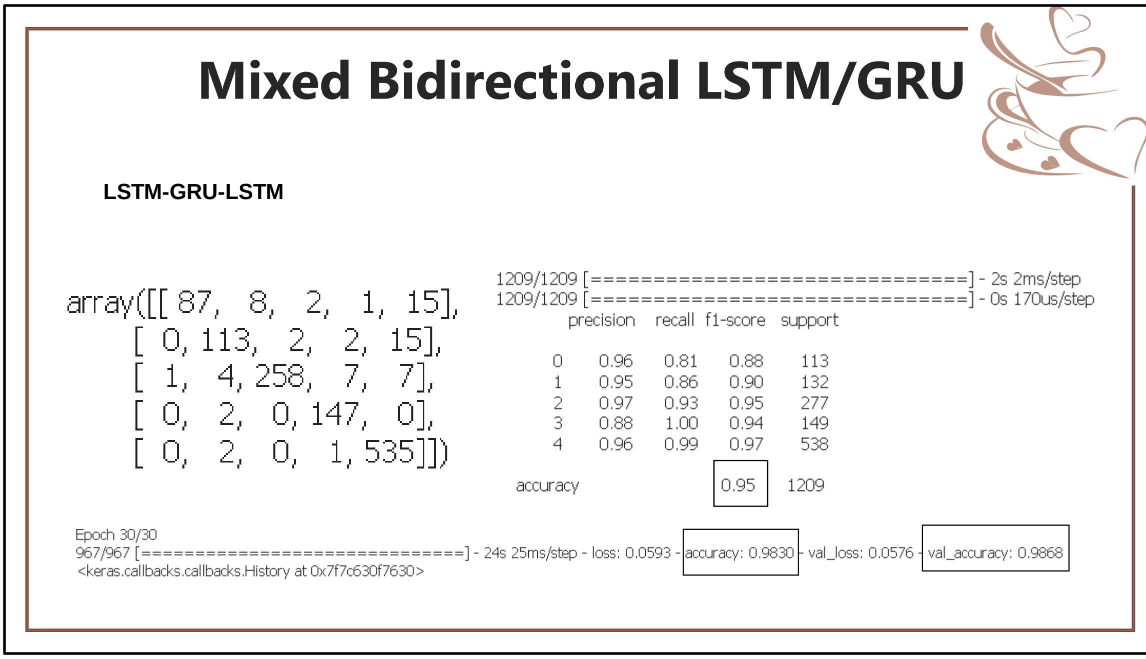
GRU-GRU-GRU

```
array([[ 90,   2,   0,   4,  17],
       [  6, 112,   1,   2,  11],
       [  6,   0, 260,  11,   0],
       [  0,   0,   0, 149,   0],
       [  2,   1,   1,   0, 534]])
```

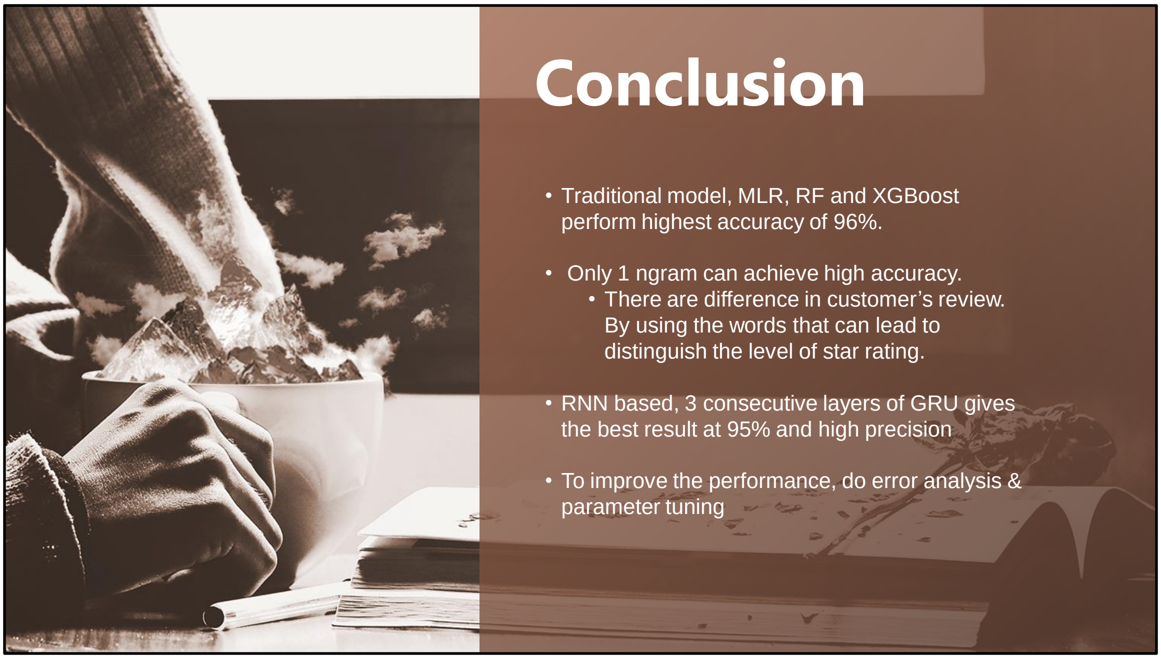
```
1209/1209 [==============================] - 3s 2ms/step
1209/1209 [==============================] - 0s 198us/step
              precision    recall  f1-score   support

           0       0.99      0.81      0.89       113
           1       0.99      0.85      0.91       132
           2       0.99      0.95      0.97       277
           3       0.96      1.00      0.98       149
           4       0.92      0.99      0.95       538

    accuracy                           0.95      1209
```

```
Epoch 30/30
967/967 [==============================] - 30s 31ms/step - loss: 0.0354 - accuracy: 0.9917 - val_loss: 0.0358 - val_accuracy: 0.9926
<keras.callbacks.callbacks.History at 0x7f7c6405de10>
```

Next 3 layers of GRU.
It give acc at 95. And most of false prediction occurred in class 1-2 stars.



Finally 3 layers of Mixed LSTM and GRU.
It give acc at 95. And most of false prediction still occurred in class 1-2 stars.
However, this model gives low precision at class 4 star 0.88.



Let's me summarize, From the tradition model part, MRL, RF and Xgboost give the highest acc at 96% and also precision, I think the reason is the effect of ensemble method and data is not very complicated. There is a group of words that can specify the level of each class.

For the RNN based model,

we can combine the result from each model, and voting the final result.

Finally, Future improvement,, we have to do error analysis by focus on error that come from false prediction and do the parameter tuning for finding best parameters.

THANK YOU!

CONTACT US AT:

Boonrit Boonmarueng

Boonrit.b@mail.kmutt.ac.th

+66 870212463