Cloud & Big Data
SWE 485

‘Classification of Customers Reported Complaints to Hungerstation Twitter Account’

Team number: 3
Section : 54132

Student name	Id
Nouf Alyousef	437200783
Reem Almalki	437200685
Raghad Alkhudhair	437204150
Rahaf Alqahtani	437201900
Amjad Algarei	437201798

Submitted Date: 19 / 04/ 2020

Table of Contents

List of Figures

List of Tables

Phase 1: Data Collection

• Project description

1) Brief description of Hungerstation company

Hungerstation is the most popular platform in the online food delivery industry. The company started in 2012 in Khobar-Saudi Arabia. It covers more than 4000 restaurants in more than 70 cities in Saudi Arabia and Bahrain. It is known for its fast and simple ordering experience which allows customers to select their favorite restaurant and the ability to pay in cash, credit cards, or other methods. It is available as a website and as a mobile application operating on both iOS and Android[1].

2) Problem description

Based on the nature of services that Hungerstation provides, multiple challenges are expected to arise that may affect the company's reputation and make it lose its customers. Some examples of the complaints or problems that may be reported by customers could be regarding incomplete/wrong/late orders, online payment problems or delivery problems as well as food quality. As mentioned above, with the huge scope of the company and the numerous number of customers, it is important to know what are the issues that customers mostly complain about in order to spend more resources in solving the issues that happen more frequently first. In this project, the customers' complaints will be classified into different categories based on the issue each complaint is about, by searching for some keywords in each tweet or complaint using Python libraries, then analyzing those tweets. This will help the company prioritize the issues to be solved based on their customers' points of view which will help them know their real weaknesses and thus overcome them.

3) Objectives of the analysis

The main objective of the analysis is to enable hungerstation to know the classification of customer complaints with different criteria including the timeline at which most raised complaints and others criteria , in order to allow Hungerstation to take a step further and make decisions regarding their services.

In order to get the customers' opinions, we will use Twitter which is an open source social media platform. Twitter is one of the main platforms used by Saudi people to express their opinions. Therefore, we will use Twitter API to extract at least 2000 tweets about Hungerstation in Arabic language. Then, the data will be cleaned by using Pandas library to specify the needed data for each tweet which are the tweet's content, and tweet's date. Then, we will use sentiment analysis using TextBlob library to extract only the negative tweets in order to prepare data to use classification technique.

Tweets will be classified into three categories which are “Payment complains” , “Delivery complains” and “Orders complains”. Then a simple manipulation will be applied using the libraries mentioned in the table to specify which year, month, day and hours that have the most complaints rate.

4) Tools and libraries

The table below describes the libraries that will be used in the project and the purpose of each as well:

List of libraries 1 Table

Library name	Purpose of use
TextBlob	1-To count words that contain complaints as mentioned above in order to identify the major problem that most customers suffer from. 2-To analyze the opinion of the customers about Hungerstation. [1]
Scikit	1- To classify tweets into three categories as mentioned above. [2]
NumPy	1- To create matrices to be used for the classification supplied from Scikit library. [3]
Pandas	1- To clean and arrange data, which helps to facilitate the understanding of data and its decoding. 2- To separate and group data according to specific criteria which are city, restaurant name and year. 3- To filter data that is analyzed and remove data that we do not want to analyze according to specific criteria. [4] 4- To explore the data extracted
TweePy & GetOldTweets3	1- To extract tweets from twitter. [5][9]
string	1- It's helpful to remove punctuations.[6]
re	1- It's helpful to remove duplicate letters.[7]

datetime	1- It's helpful to split the date to at least 3 arguments - year, month, and day.[8]

5) Development environment

The development was performed using Jupyter notebook to write Python scripts used for extracted, exploring, classifying, analyzing and visualizing data.

- **Data Collection**

Around 2000 tweets were extracted from Twitter using Twitter API. The extracted tweets all mentioned '@hunger_care' which is the twitter account specified for customer care and complaints for Hungerstation Company. In future phases, these tweets will be analyzed and classified.

Phase 2: Data preprocessing

- **Data Exploration**

After exploring the existing data we found that we needed more tweets so we have collected around 15000 new tweets and we explored them using Pandas library to get more general information about the data as shown in Figure(1). Also, we used string library, re library and datetime library as mentioned in table (1)

Data Exploration

```
In [ ]: allData = pd.read_csv("allTweets2019_2020.csv")

In [ ]: # how much data do I have?
        len(allData)

In [ ]: # Print a concise summary of a DataFrame
        allData.info()

In [ ]: # number of non-NA values
        allData.count()

In [ ]: # Retrieve list of columns
        allData.columns

In [ ]: #Generates descriptive statistics or Summary Statistic of the numeric columns
        allData.describe()
```

Figure 1 Data Exploration

In order to prepare the data for the next phase we needed to solve some quality issues as shown in the table below:

Table 2 Quality issues

Quality issues	Why it's need to be fixed	How it's fixed
1- Remove hunger care replies	Because we are only interested in knowing customers opinions	By using loop to go through all the content and get index of tweets that contain" hunger_care" then remove them using method "drop()"
2- Remove numbers , punctuations and duplicate letters from tweets	Since most of the tweets contain the order number which will not be useful in the analysis. Also, because we need to make the data more structured and meaningful.	By importing the library "string" which helps with removing punctuations using " string.punctuation". Also,by importing the library "re" which helps with removing repeated letters using "re.sub". And, to remove the number we used .isdigit()
3- Remove unwanted columns	To reduce the amount of data which is not helpful in the analysis or most of its values are null	By using "drop()" method and the column name
4- Remove tweets that don't contain keywords	Because we need to reduce the amount of data which will not help in the analysis	By using loop to go through all the content and remove the tweets that don't contain any keywords[1]
5- Remove null tweets	Because we need to reduce the amount of data which will not help in the analysis	By using loop to go through all the content and remove tweets with null value
6 - Replace some NaN values with Not given	To replace null values by Not given instead of deleting the whole tweet.	By using "fillna()" method

[1] ("صرف" - "كاش" -"فيزا" -"طلب خطأ" -"طلب غلط" -"دفع" - "بطاقة" -"تأخير" -"متأخر" - "بارد" -"مندوب" -"سائق" طلب ناقص" -"الطلب ناقص" -"التوصيل" -"التطبيق" - "كود" -"ماوصل" -"ما وصل" -"الموقع" -"اللوكيشن" - "عطل" -"معلق" -" - "الوقت" - "لا يستجيب" -"مايرد" -"ما يرد" - "مشغول" -"المحفظة" -"مطعم" "تأخير" - "الغاء" - "الغى" -"استرجاع" -"فلوس"")

7- Separate date to four columns (year,month,day,hour)	Because we need to see the most days and times in which complaints occured by customers.	By import datetime to separate it into 4 columns which are year , month, day and hour and remove the column of date .

We have three files as shown on the table below

Table 3 Files

Files	Description
1- allTweetsBeforeCleaningxcel.xlsx	contain the original data before cleaning
2- finalCode.ipynb	contain the source code of data exploration and cleaning.
3- FinalTweetsAfterCleaning.xlsx	contain the original data after cleaning.

Phase 3: Data Analysis and modeling

- **Descriptive Analysis:**

In order to help solving the problem defined in phase one, which is determining which categories of complaints are tweeted by Humberston's customers mentioning @Hunger_carer account on twitter, we analyzed the data from several aspects.

First, in order to understand the data, a look at the head of the data showing the columns and the first five rows in provided in the below cell.

Figure 2 Data Columns

```
In [296]:
1 tokenizer = RegexpTokenizer(r'\w+')
2 allData["tokens"] = allData["cleanTweets"].apply(tokenizer.tokenize)

In [297]:
1 allData.head()

Out[297]:
```

	Unnamed: 0	Tweets	Username	Mentions	To	cleanTweets	Hour	Day	Month	Year	tokens
0	0	ما وصلني شي من الي وعدتوه بالخاص	Talal_A_M	Not given	hunger_care	ما وصلني شي من الي وعدتوه بالخاص	23	19	12	2019	[ما, وصلني, شي, من, الي, وعدتوه, بالخاص]
1	14	انا قبل شوي طلبت بالموقع وطريقة الدفع عن طريق ...	mjoodh_0121	Not given	hunger_care	انا قبل شوي طلبت بالموقع وطريقة الدفع عن طريق ...	23	19	12	2019	[انا, قبل, شوي, طلبت, بالموقع, وطريقة, الدفع, ...]
2	20	تاخر الطلب وارسلت لكم وهذا الرد وبعد ماكلمت ال...	rn21212121	Not given	hunger_care	تاخر الطلب وارسلت لكم وهذا الرد وبعد ماكلمت ال...	22	19	12	2019	[تاخر, الطلب, وارسلت, لكم, وهذا, الرد, وبعد, م...]
3	22	تتمنى تشوفون وتتاكدون من كلامي لاني كلمت وشفت ...	rn21212121	Not given	hunger_care	تتمنى تشوفون وتتاكدون من كلامي لاني كلمت وشفت ب...	22	19	12	2019	[تتمنى, تشوفون, وتتاكدون, من, كلامي, لاني, كلمت...]
4	28	هذا هو انا الحين مابي الطلب ابي الفلوس الي راح...	joj_a1993	Not given	hunger_care	هذا هو انا الحين مابي الطلب ابي الفلوس الي راح...	22	19	12	2019	[هذا, هو, انا, الحين, مابي, الطلب, ابي, الفلوس...]

The above cell output shows 11 columns which are:

Table 4 Columns Descriptions

Column	Description
Unndamed: 0	Index of the tuple in the original data before eliminating tweets based on cleaning process in phase 2
Tweets	Content of the tweet
Username	Username of the tweet writer
Mentions	Username of the mentioned account in the tweet
To	Username of whom the tweet is wrote to (in this case it is always @Hunger_care)
cleanTweets	The content of tweets after applying the cleaning processes in phase 2
Hour	Hour the tweet posted at
Day	Day the tweet posted on
Month	Month the tweet posted in
Year	Year the tweet posted in
Tokens	Words used in the tweet separated in an array

Furthermore, in understanding the data, the cell -Figure 3 - below shows the vocabulary size in the tweets and the used words in total, as well as the maximum length of a tweet (in words).

```
In [298]:
all_words = [word for tokens in allData["tokens"] for word in tokens]
sentence_lengths = [len(tokens) for tokens in allData["tokens"]]

VOCAB = sorted(list(set(all_words)))

print("%s words total, with a vocabulary size of %s" % (len(all_words), len(VOCAB)))
print("Max sentence length is %s" % max(sentence_lengths))
```

```
35414 words total, with a vocabulary size of 7397
Max sentence length is 58
```

Figure 3 Vocabulary Size

In the Following part, we will provide answers based on the analysis we did for the following questions:

What are the top ten words customers use in tweets?

From the below bar chart in Figure 4, we can see the most used words in the collected tweets is “ الطلب” following with "من" following with the Hungerstation customer care account username.

```
In [301]:
word_df.set_index('Word', inplace=True)
```

```
In [302]:
# visualize the top 20 words

fig, ax = plt.subplots()

ax.tick_params(axis='x', labelsize=15,rotation=45)
ax.tick_params(axis='y', labelsize=10)

ax.set_xlabel('Word', fontsize=15)
ax.set_ylabel('Count' , fontsize=15)
ax.set_title('Distribution of words', fontsize=15, fontweight='bold')

word_df.plot(ax=ax, kind='bar')
```

```
Out[302]: <matplotlib.axes._subplots.AxesSubplot at 0x1a22a12510>
```

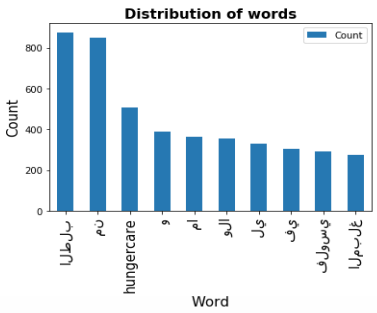

Figure 4 Distribution of Words

The below cell -Figure 5- shows the number of occurrences of each word in the whole collected tweets.

```
In [304]:
1  For_words= [ word for tokens in allData["tokens"] for word in tokens]

In [305]:
1 for_word_counter = Counter(For_words)

In [306]:
1 for_word_counter.most_common(10)

Out[306]: [('875 ,'الطلب'),
 ('847 ,'من'),
 ('hungercare', 507),
 ('387 ,'و'),
 ('361 ,'ما'),
 ('353 ,'ولا'),
 ('328 ,'لي'),
 ('302 ,'في'),
 ('290 ,'فلوسي'),
 ('274 ,'المبلغ')]
```

Figure 5 Words Distribution in Numbers

From the cell above we can see that most of the words are prepositions, so it is better to ignore these prepositions since they do not indicate an important fact nor tell an effective story about the data.

So, the below - Figure 6 -shows the most occurred words in the collected tweets after ignoring prepositions.

```
In [307]:
1 # stop words
2 ignore = {'من','عن','هذا','في',
3           'ما','مع','على','لي','او','ولا','انا' ,'الي' ,'و' ,'ان','الا','بس','او'}

In [308]:
1 # remove stop words from the list of words
2 for word in ignore:
3     if word in for_word_counter:
4         del for_word_counter[word]

In [309]:
1 for_word_counter.most_common(10)

Out[309]: [('875 ,'الطلب'),
 ('hungercare', 507),
 ('290 ,'فلوسي'),
 ('274 ,'المبلغ'),
 ('266 ,'طلبت'),
 ('216 ,'المطعم'),
 ('197 ,'التوصيل'),
 ('195 ,'تم'),
 ('179 ,'الفلوس'),
 ('169 ,'التطبيق')]
```

Figure 6 Distribution without Prepositions

The above figure -Figure 6- shows that most complaints mentioned the word "الطلب"" along with the customer care account username, and following with the word "فلوسي" and "المبلغ" which gives an indication that most complaints are related to payment issues, and therefore the company should put more effort on making the payment process much easier and trusted.

What are the least ten common words customers use in tweets?

The figure -Figure 7- below shows the least occurred words in the collected tweets, which are all occurred once only.

```
In [310]: For_words= [ word for tokens in allData["tokens"] for word in tokens]

In [311]: for_word_counter = Counter(For_words)

In [312]: for_word_counter.most_common()[-10:]

Out[312]: [('1', 'استردادة'),
           ('1', 'مازالت'),
           ('1', 'رسايل'),
           ('1', 'لسحب'),
           ('1', 'مخترق'),
           ('1', 'هنقرستيشنمخترق'),
           ('1', 'لتقديم'),
           ('1', 'واسترجاع'),
           ('1', 'واقل'),
           ('1', 'حظر')]
```

Figure 7 Least Occurred Words

The below cell -Figure 8- shows the ten least common words after ignoring the prepositions.

```
In [313]: # stop words
          ignore = {'من','عن','هذا','في',
                    'ما','حظر','مازالت','مع','على','لي','او','ولا','انا','الي','و','ان','الا','بس','او'}

In [314]: # remove stop words from the list of words
          for word in ignore:
              if word in for_word_counter:
                  del for_word_counter[word]

In [315]: for_word_counter.most_common()[-10:]

Out[315]: [('1', 'الإلكتروني'),
           ('1', 'بقدرة'),
           ('1', 'استردادة'),
           ('1', 'رسايل'),
           ('1', 'لسحب'),
           ('1', 'مخترق'),
           ('1', 'هنقرستيشنمخترق'),
           ('1', 'لتقديم'),
           ('1', 'واسترجاع'),
           ('1', 'واقل')]
```

Figure 8 Least occurred without Prepositions

At what hours most tweets are posted?

From the below cell – Figure 9- we can see the bar chart that shows the number off tweets in each hour. Most tweets are posted at 6 and 7 pm. Which may suggest that the company should hire more employees at these peak hours.

```
In [51]:
occurances=df["Hour"].value_counts()
hours=df["Hour"].value_counts().keys()

plt.ylabel("Number of complaints")
plt.xlabel("Hour")
plt.bar(hours, occurances)
plt.figure(figsize=(70,70))
plt.show()
```

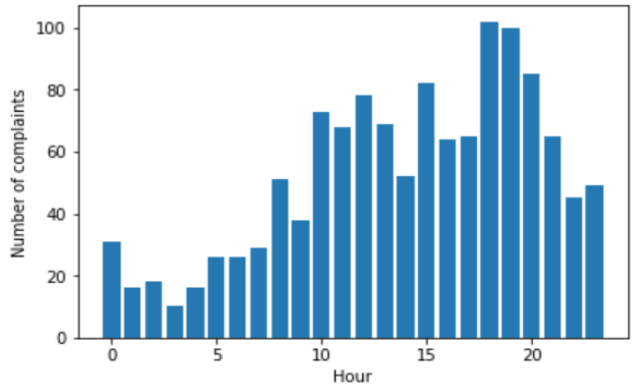

Figure 9 Peak Hours

How many tweets are negative tweets and how many are positive?

After applying Mazajak framework to classify the tweets into poaitive, negative, and neutral. Since we are mostly interested in negative and positive tweets, neutral tweets were eliminated and the value counts of the rest of the two classification in inducated in the cell below.

```
In [5]:
df.Sentiment.value_counts()

Out[5]: negative    1170
        positive      88
        Name: Sentiment, dtype: int64
```

Figure 10 Sentiment Counts

In what months the negative and positive tweets are posted?

The below graph -Figure 11- shows the months that most positive or negative tweets are posted on.

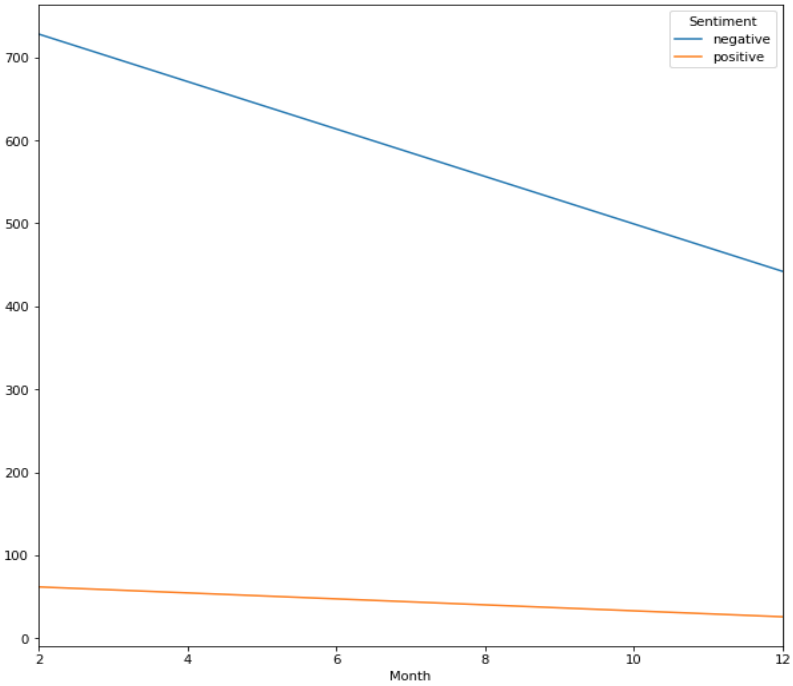

Figure 11 Peak Months

How many tweets in each category?

After dividing the tweets in three categories (Payment, Order, Delivery) in order to know which issue that causes customer to complain about more, the below cell shows the number of tweets in each category. The payment category is the top.

```
In [50]: 1 df.Category.value_counts()

Out[50]: Payment     543
         order       506
         Deliviry    209
         Name: Category, dtype: int64
```

Figure 12 Category Counts

In what months the tweets are posted based on their categories?

The below graph shows the months that most tweets are posted on based on their categories.

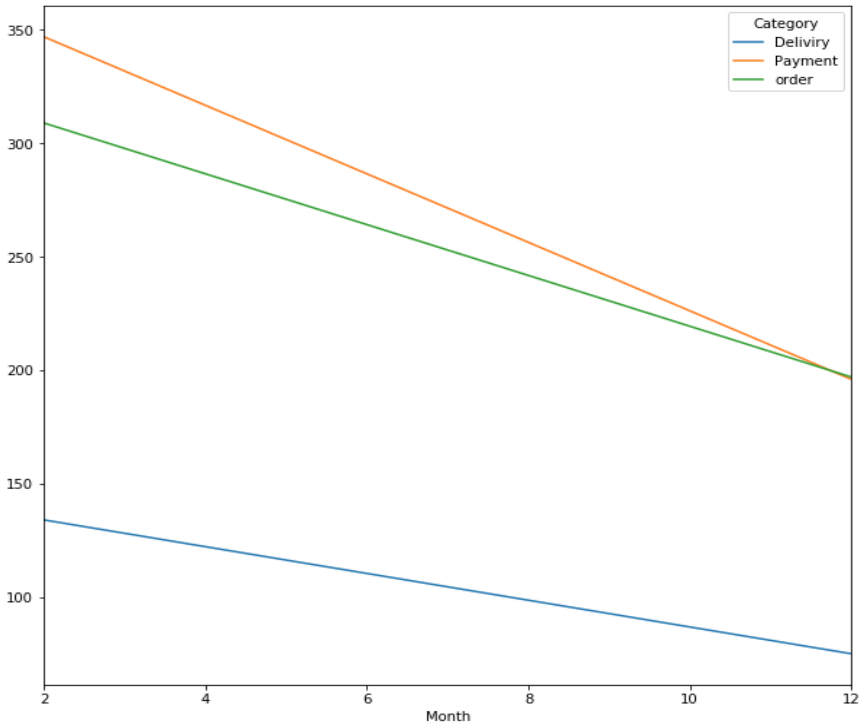

Figure 13 Peak Months with Category

- **Predictive Analysis:**

We applied a predictive model for classifying the sentiment of the tweet (positive, negative), and another predictive mode for classifying the category of the tweet (Payment, Delivery or Order).

1. Predictive models for the tweet's sentiment

After applying Mazajak on all tweets to give each of the a sentiment, and then eliminating neutral tweets, we divided the data into 50% training set and 50% testing set. And the applied several models and evaluated their performance.

A) Naïve-Bayes

Naïve-Bayes is a classification method based on Bayes' Theorem that assumes independence between predictors. A Naive Bayes classifier assumes that the existence of a feature has no relation with the existence of any other feature [10].

For example, a fruit may be considered to be a banana if it is yellow, long, and about 6 inches tall. Even if these features depend on each other or upon the presence of the other features, all of these properties independently contribute to the probability that this fruit is a banana and that is why it is known as 'Naive'[10].

Naive Bayes is easy to perform and fits big data sets. Besides its simplicity, Naive Bayes outperform even highly sophisticated classification methods [10].

However, after applying the Naïve-Bayes model to our data in the below cell, we got the below performance evaluation.

In [94]:

```
# create the classifer and fit the training data and lables
classifier_nb = MultinomialNB().fit(X_train.todense(),y_train)

print("MultinomialNB accuracy: %.2f"%classifier_nb.score(X_test.todense(), y_test))

# #do a 10 fold cross-validation
results_nb = cross_val_score(classifier_nb, X.todense(),target, cv=10)
print("\n10-fold cross-validation:")
print(results_nb)

print("The average accuracy of the MultinomialNB classifier is : %.2f" % np.mean(resul

print("\nConfusion matrix of the MultinomialNB classifier:")
predicted_nb = classifier_nb.predict(X_test.todense())
print(confusion_matrix(y_test,predicted_nb))


print("\nClassification_report of MultinomialNB classifier:")
print(classification_report(y_test,predicted_nb))
print("--------------------------------------------------------------------------------")
```

```
MultinomialNB accuracy: 0.92

10-fold cross-validation:
[0.92857143 0.92857143 0.92857143 0.92857143 0.92857143 0.92857143
 0.92857143 0.92857143 0.936      0.936      ]
The average accuracy of the MultinomialNB classifier is : 0.93

Confusion matrix of the MultinomialNB classifier:
[[578   0]
 [ 51   0]]

Classification_report of MultinomialNB classifier:
              precision    recall  f1-score   support

           0       0.92      1.00      0.96       578
           1       0.00      0.00      0.00        51

    accuracy                           0.92       629
   macro avg       0.46      0.50      0.48       629
weighted avg       0.84      0.92      0.88       629
```

Figure 14 Naive-Bayes results

From the above figure we can see that the Naïve-Bayes model gave an average accuracy of 93%.

Furthermore, the graph below shows the ROC of the Naïve-Bayes.

In [95]:
```
# calculate the fpr and tpr for all thresholds of the classification
probs = classifier_nb.predict_proba(X_test)
preds = probs[:,1]

fpr, tpr, threshold = metrics.roc_curve(y_test, preds)
roc_auc = metrics.auc(fpr, tpr)
```

In [96]:
```
plt.title("ROC Naive Bayes")
plt.plot(fpr , tpr, 'b', label = 'AUC = %0.2f' % roc_auc)
plt.legend(loc = 'lower right')
plt.plot([0,1] , [0, 1] , 'r--')
plt.xlim([0,1])
plt.ylim([0,1])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show
```

Out[96]: <function matplotlib.pyplot.show(*args, **kw)>

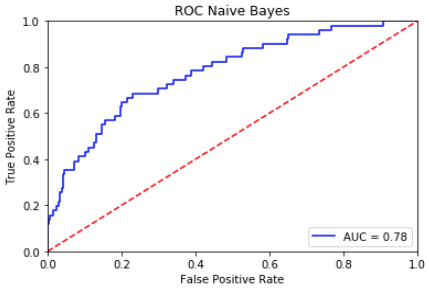

Figure 15 Naive-Bayes ROC

B) SVM

Support Vector Machine, known as SVM for short, is a classification technique used in machine learning. In the SVM algorithm, each data item is plotted as a point in n-dimensional space (where n is number of features) with the value of each feature being the value of a particular coordinate. Then, classification is performed using the hyper-plane that differentiates the two classes [11].

After applying the SVM model to our data in the below cell, we got the below performance evaluation.

In [97]:

```
# create the classifer and fit the training data and lables
classifier_svm = svm.SVC(kernel='linear', C=1, probability=True).fit(X_train,y_train)

print("SVM accuracy: %.2f"%classifier_svm.score(X_test, y_test))

#do a 10 fold cross-validation
results_svm = cross_val_score(classifier_svm, X,target, cv=10)
print("\n10-fold cross-validation:")
print(results_svm)

print("The average accuracy of the SVM classifier is : %.2f" % np.mean(results_svm))

print("\nConfusion matrix of the SVM classifier:")
predicted_svm = classifier_svm.predict(X_test)
print(confusion_matrix(y_test,predicted_svm))


print("\nClassification_report of SVM classifier:")
print(classification_report(y_test,predicted_svm))
print("--------------------------------------------------------------------------------")
```

```
SVM accuracy: 0.93

10-fold cross-validation:
[0.92857143 0.94444444 0.94444444 0.94444444 0.94444444 0.93650794
 0.91269841 0.94444444 0.944      0.936     ]
The average accuracy of the SVM classifier is : 0.94

Confusion matrix of the SVM classifier:
[[576   2]
 [ 44   7]]

Classification_report of SVM classifier:
              precision    recall  f1-score   support

           0       0.93      1.00      0.96       578
           1       0.78      0.14      0.23        51

    accuracy                           0.93       629
   macro avg       0.85      0.57      0.60       629
weighted avg       0.92      0.93      0.90       629
```

Figure 16 SVM results

From the above figure we can see that the Naïve-Bayes model gave an average accuracy of 94%.

The graph below shows the ROC for the SVM model.

```
In [98]:
# calculate the fpr and tpr for all thresholds of the classification
probs = classifier_svm.predict_proba(X_test)
preds = probs[:,1]

fpr, tpr, threshold = metrics.roc_curve(y_test, preds)
roc_auc = metrics.auc(fpr, tpr)
```

```
In [99]:
plt.title("ROC SVM")
plt.plot(fpr , tpr, 'b', label = 'AUC = %0.2f' % roc_auc)
plt.legend(loc = 'lower right')
plt.plot([0,1] , [0, 1] , 'r--')
plt.xlim([0,1])
plt.ylim([0,1])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show
```

```
Out[99]: <function matplotlib.pyplot.show(*args, **kw)>
```

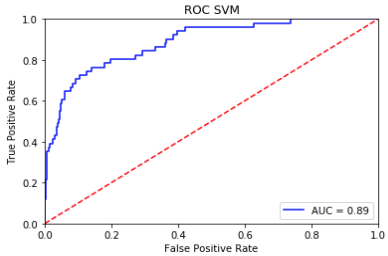

Figure 17 SVM ROC

C) Logistic

Logistic regression is named for the function used at the core of the method, the logistic function [12].

The logistic function, also called the sigmoid function was developed by statisticians to describe properties of population growth in ecology, rising quickly and maxing out at the carrying capacity of the environment. It's an S-shaped curve that can take any real-valued number and map it into a value between 0 and 1, but never exactly at those limits [12].

After applying the logistic regression model to our data in the below cell, we got the below performance evaluation.

In [101]:

```
print("Logisitic Accuracy: %.2F"%classifier_log.score(X_test, y_test))
results_log = cross_val_score(classifier_log, X,target, cv=10)


print("\n10-fold cross-validation:")
print(results_log)


print("The average accuracy of the Logisitic classifier is : %.2f" % np.mean(results_l
print("\nConfusion matrix of the Logisitic classifier:")
predicted_log= classifier_log.predict(X_test)
print(confusion_matrix(y_test,predicted_log))


print("\nClassification_report of Logisitic classifier:")
print(classification_report(y_test,predicted_log))
print("--------------------------------------------------------------------------------")
```

```
Logisitic Accuracy: 0.92

10-fold cross-validation:
[0.92857143 0.94444444 0.92857143 0.92857143 0.92857143 0.92857143
 0.92857143 0.93650794 0.936      0.936     ]
The average accuracy of the Logisitic classifier is : 0.93

Confusion matrix of the Logisitic classifier:
[[578   0]
 [ 51   0]]

Classification_report of Logisitic classifier:
              precision    recall  f1-score   support

           0       0.92      1.00      0.96       578
           1       0.00      0.00      0.00        51

    accuracy                           0.92       629
   macro avg       0.46      0.50      0.48       629
weighted avg       0.84      0.92      0.88       629
```

Figure 18 Logistic results

From the above figure we can see that the Naïve-Bayes model gave an average accuracy of 93%. The graph below shows the ROC for the logistic model.

In [102]:
```
# calculate the fpr and tpr for all thresholds of the classification

probs = classifier_log.predict_proba(X_test)
preds = probs[:,1]

fpr, tpr, threshold = metrics.roc_curve(y_test, preds)
roc_auc = metrics.auc(fpr, tpr)
```

In [103]:
```
# polt AUC
plt.title('ROC Logisitic ')
plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % roc_auc)
plt.legend(loc = 'lower right')
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0, 1])
plt.ylim([0, 1])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
```

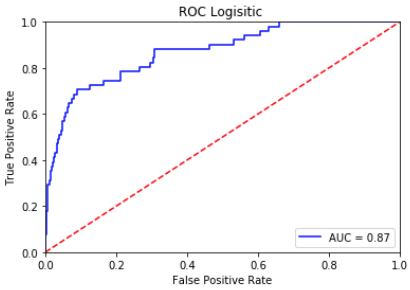

Figure 19 Logistic ROC

Based on the results above, the SVM gave the highest accuracy among the used model which is 94%. Therefore, we chose the SVM to predict the tweet sentiment.

2. Predictive models for the tweet's category

First, we classified the tweets into three categories (payment, order, deliver) based on the occurrence of the related key words in the below cell.

In [109]:
```
filter1 = data['tokens']

categoryList = []
for x in filter1:
    if ("بارد" in x or "بالخاص" in x or "خدمة" in x or
        "الغاء" in x  or "تكنسلت" in x  or "اطلب" in x or "اطلب" in x or "اطلب" in x
       or "الأكل" in x or "ناقص" in x or "الخدمة" in x or "طلبي" in x or "جودة" in x
       or "الجودة" in x or "وجبة" in x or "طلبت" in x or "نتيجة" in x or "شكواي" in x
       or "حسبي" in x or "فشل" in x or "وجبتي" in x or "المطعم" in x or "خايس" in x
       or "مجيوس" in x or "جاء" in x or "غلط" in x ):
        categoryList.append('order')


    elif ("تاخر" in x or "السائق" in x or "السواق" in x or
        "واسلوبه" in x  or "المندوب" in x  or "وقت" in x or "التوصيل" in x or "اكثر" in x
       or "بالسائقين" in x or "بالغلط" in x or "لعنوان" in x or "عنوان" in x or "ثاني" in x
       or "توصيل" in x or "مشوار" in x or "موظفكم" in x or "ساعه" in x or "لمندوبين" in x
       or "ولسي" in x or "المندوب" in x or "المندوبين" in x or "ادب" in x or "بالتوصيل" in x
       or "التسليم" in x or "استلم" in x or "دقيقه" in x or "دقيقة" in x):
        categoryList.append('Deliviry')

    else: categoryList.append('Payment')


data['Category']= categoryList
```

Figure 20 Category Filtering

Then we applied the same three models we applied previously on the tweet sentiment, after dividing 50% of the data into training set and 50% of the data into testing set.

A) Naïve-Bayes

In predicting the tweet category, the naïve-Bayes gave an average accuracy of 73% and below in the performance report.

```
MultinomialNB accuracy: 0.69

10-fold cross-validation:
[0.7480315  0.7007874  0.7480315  0.72222222 0.75396825 0.67460317
 0.752      0.768      0.712      0.75       ]
The average accuracy of the MultinomialNB classifier is : 0.73

Confusion matrix of the MultinomialNB classifier:
[[ 14  10  82]
 [  7 179  85]
 [  3   9 240]]

Classification_report of MultinomialNB classifier:
              precision    recall  f1-score   support

           0       0.58      0.13      0.22       106
           1       0.90      0.66      0.76       271
           2       0.59      0.95      0.73       252

    accuracy                           0.69       629
   macro avg       0.69      0.58      0.57       629
weighted avg       0.72      0.69      0.66       629
```

Figure 21 Naive-Bayes results with Categories

B) SVM

The performance report of the SVM is shown below after applying it in tweet category prediction. The SVM gave a 93% average accuracy.

```
SVM accuracy: 0.86

10-fold cross-validation:
[0.87401575 0.93700787 0.88188976 0.92063492 0.96031746 0.93650794
 0.944      0.944      0.968      0.93548387]
The average accuracy of the SVM classifier is : 0.93

Confusion matrix of the SVM classifier:
[[ 76  26   4]
 [  6 257   8]
 [ 16  29 207]]

Classification_report of SVM classifier:
              precision    recall  f1-score   support

           0       0.78      0.72      0.75       106
           1       0.82      0.95      0.88       271
           2       0.95      0.82      0.88       252

    accuracy                           0.86       629
   macro avg       0.85      0.83      0.84       629
weighted avg       0.86      0.86      0.86       629
```

Figure 22 SVM Results with Categories

C) Logistic

The logistic regression gave 89% average accuracy as shown below in the classification report.

```
10-fold cross-validation:
[0.86614173 0.85826772 0.85826772 0.88095238 0.93650794 0.88095238
 0.904      0.888      0.944      0.91129032]
The average accuracy of the Logisitic classifier is : 0.89

Confusion matrix of the Logisitic classifier:
[[ 56  40  10]
 [  3 260   8]
 [  7  38 207]]

Classification_report of Logisitic classifier:
              precision    recall  f1-score   support

           0       0.85      0.53      0.65       106
           1       0.77      0.96      0.85       271
           2       0.92      0.82      0.87       252

    accuracy                           0.83       629
   macro avg       0.85      0.77      0.79       629
weighted avg       0.84      0.83      0.83       629
```

Figure 23 Logistic Results with Categories

Based on the results shown previously, SVM gave the highest average accuracy also when applied on predicting the tweet category. Therefore, the team chose the SVM model.

Phase 4: Visualization and Findings

• Visualizations and Findings

This section contains different visualization graphs that describe the data we classified by our models in the previous sections, and some of the findings we reached to from the graphs as well.

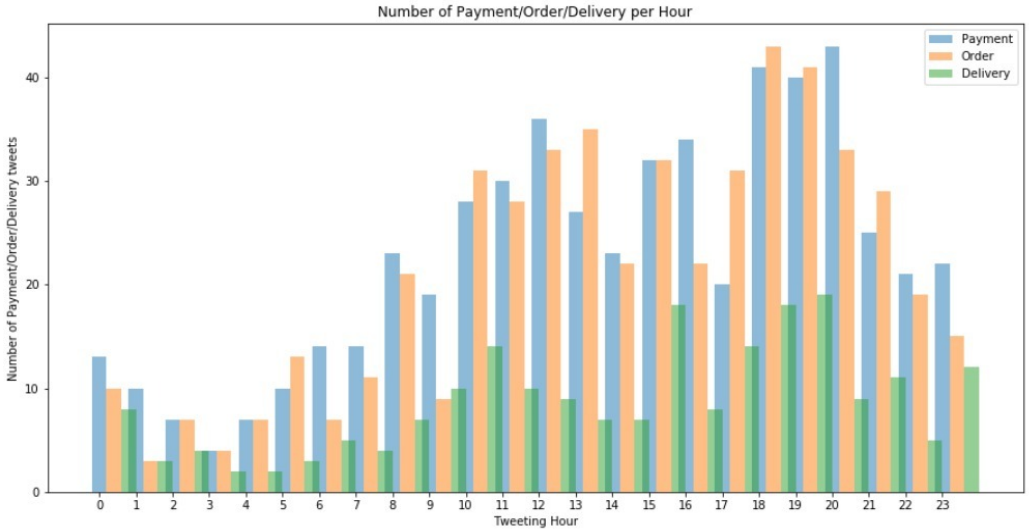

Figure 24 Tweeting Hours Graph

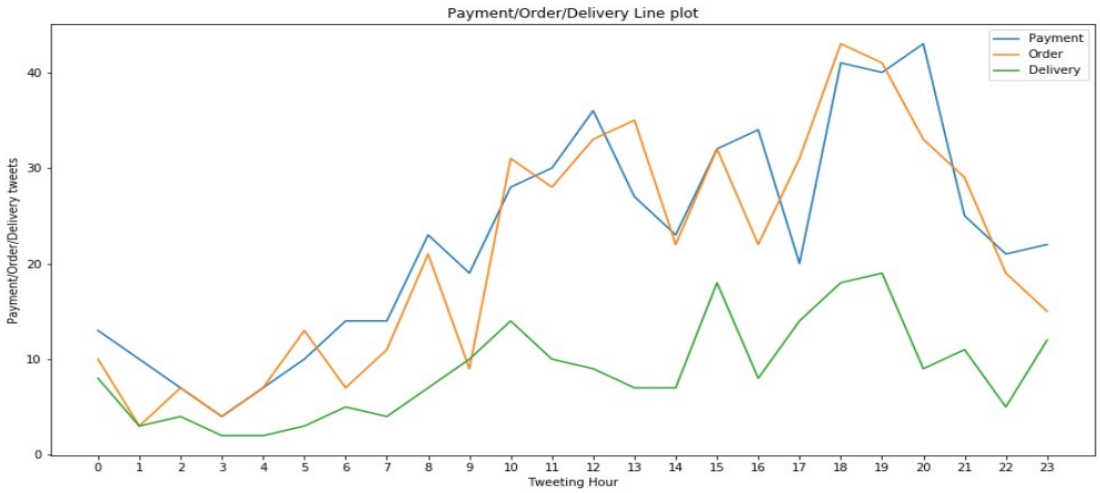

Figure 25 Tweeting Hours Graph #2

From the two visualizations above, we found that

- The number of tweets about order and payment complaints reached more than 40 tweets, while the number of tweets about delivery complaints never exceeded 20 tweets.

- The number of order complaints is the highest in the period between [18 pm - 20 pm] and the least in the period between [0 am - 7 am]. And the highest hour with the most order complaints is at 18:00.

- The number of payment complaints is the highest in the period between [18pm - 20pm] and the least in the period between [0 am-7am]. Moreover, the highest hour with the most payment complaints is at 20:00.

- The number of delivery complaints is the highest in the period between [18 pm - 20 pm] and the least in the period between [0 am - 9 am]. And the highest hours with the most delivery complaints are at 20 pm and 16 pm.

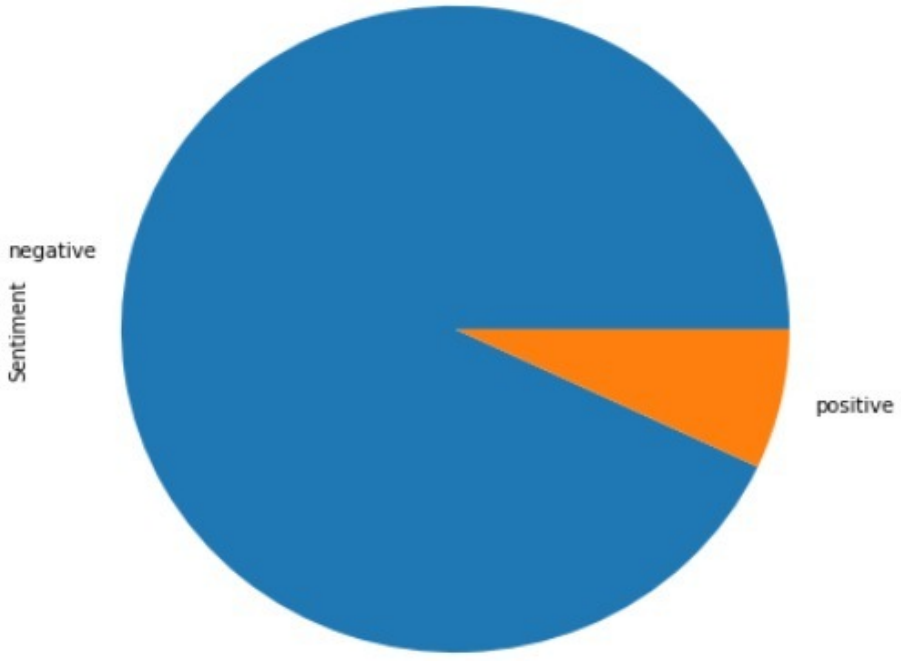

Figure 26 Sentiment Pie Chart

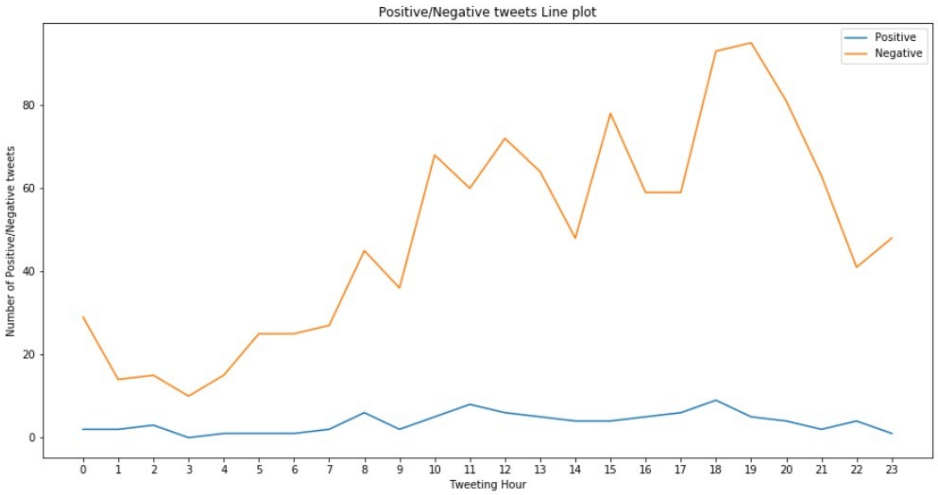

Figure 27 Tweeting Hours Based on Sentiment

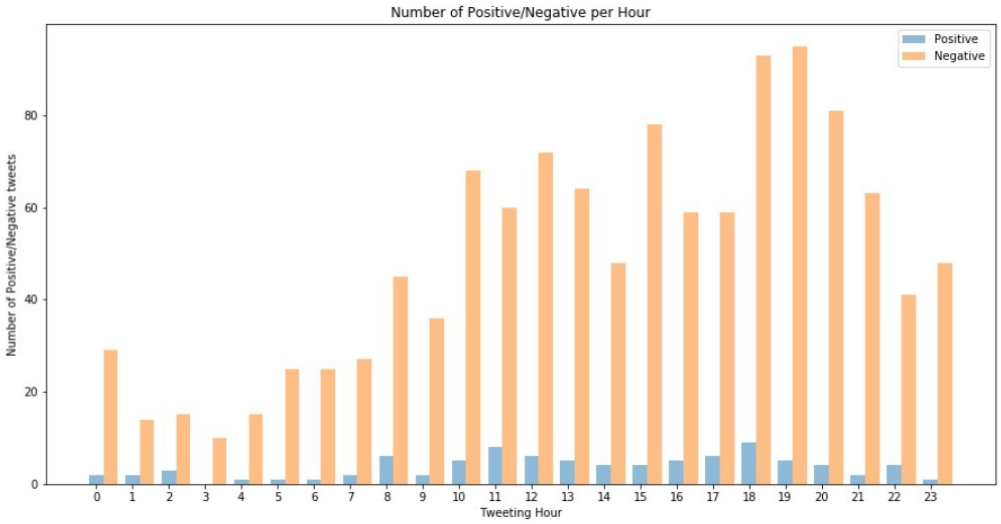

Figure 28 Tweeting Hours Based on Sentiment #2

According to the three visualizations above, we found the majority tweets about Hungerstation company shows that most people feel frustrated with the Hungerstation app. Also, we found that:

- The ratio of the complaints represents the largest number of tweets and reaches more than 75%.
- The number of complaints is the highest in the period between [18 pm - 20 pm] and the least in in the period between [1 am - 4 am].
- In general, the number of complaints increased in the late-night hours.
- The number of positive tweets never exceeded twenty tweets in one hour.
- The number of negative tweets reached more than 80 tweets in one hour.

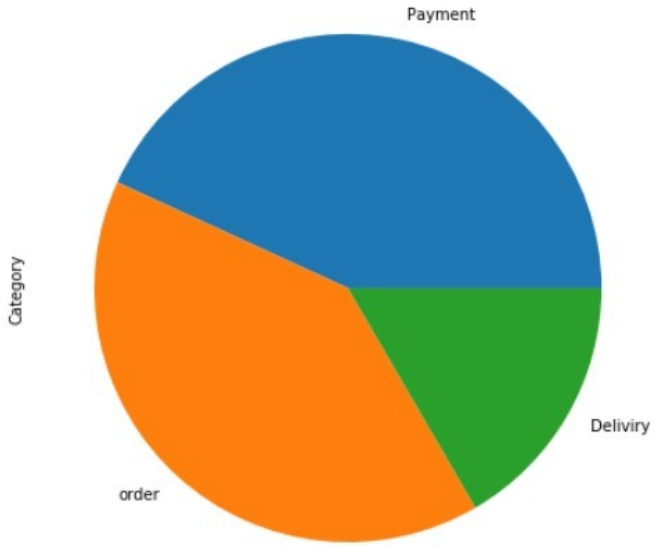

Figure 29 Category Pie Chart

After we did the visualization above, we found that the number of order complaints is the most, then payment complaints and the least is delivery complaints.

- ## Recommendations

  Depending on the analysis of Hungerstation client's tweets, we can conclude that there are critical hours in the day that the client's complaints increase in, these complaints mostly are about payment and order which could affect the company reputation and customer satisfaction. In order to improve the service, we suggest providing some instructions to guide clients, this could reduce complaints as well as increasing and training the staff could be helpful too(delivery and help desk). Moreover, for the critical hours we believe that the company should put more effort on increasing the quality of service at all levels and especially in the critical hours where the number of complaints increase and the demand for hiring more staff increases consequently.

- **Frameworks and libraries:**

Table 5 Phase#4 Frameworks & Libraries

Framework/Library Name	Purpose of Use
Mazajak	To classify the sentence sentiment into positive, negative or neutral [13].
RegexpTokenizer	To divide the tweet into words [14].
matplotlib.pyplot	To plot the data into visual graphs [15].
Counter	To count the number of objects in each list [16].
requests	To send HTTP requests using python [17].
json	To convert between python list and json files [18].
Scikit	1- To classify tweets into three categories as mentioned above. [2]
Pandas	1- To clean and arrange data, which helps to facilitate the understanding of data and its decoding. 2- To separate and group data according to specific criteria which are city, restaurant name and year. 3- To filter data that is analyzed and remove data that we do not want to analyze according to specific criteria. [4] 4- To explore the data extracted
Numpy	1- To create matrices to be used for the classification supplied from Scikit library. [3]

We have three files as shown on the table below

Table 6 Phase#4 files

Files	Description
1- ClassificationPhaseFinalTweets.csv	contain the data after cleaning and classification.
2- DataVisualization.ipynb	contain the source code of data classification.

References :

[1].Stack Abuse. (2020). *Python for NLP: Introduction to the TextBlob Library*. [online] Available at: https://stackabuse.com/python-for-nlp-introduction-to-the-textblob-library/ [Accessed 18 Feb. 2020].

[2].Stack Abuse. (2020). *Python for NLP: Introduction to the TextBlob Library*. [online] Available at: https://stackabuse.com/python-for-nlp-introduction-to-the-textblob-library/ [Accessed 18 Feb. 2020].

[3].Numpy.org. (2020). *NumPy — NumPy*. [online] Available at: https://numpy.org/ [Accessed 18 Feb. 2020].

[4].Team, D. (2020). *15 Latest Pandas Features - What Makes Python Pandas Unique? - DataFlair*. [online] DataFlair. Available at: https://data-flair.training/blogs/python-pandas-features/ [Accessed 18 Feb. 2020].

[5].Medium. (2020). *Tweepy: a Python Library for the Twitter API*. [online] Available at: https://medium.com/@jasonrigden/tweept-a-python-library-for-the-twitter-api-9d0537dcebd4 [Accessed 18 Feb. 2020].

[6].string, B. and Johnston, L., 2020. *Best Way To Strip Punctuation From A String*. [online] Stack Overflow. Available at: <https://stackoverflow.com/questions/265960/best-way-to-strip-punctuation-from-a-string> [Accessed 13 March 2020].

[7].Regular-expressions.info. 2020. *Python Re Module - Use Regular Expressions With Python - Regex Support*. [online] Available at: <https://www.regular-expressions.info/python.html> [Accessed 13 March 2020].

[8].Glykas, S., 2020. *Working With Datetime Objects And Timezones In Python — Agile Actors*. [online] agile actors. Available at: <http://blog.agileactors.com/blog/2018/2/22/working-with-datetime-objects-and-timezones-in-python> [Accessed 13 March 2020].

[9].PyPI. 2020. *Getoldtweets3*. [online] Available at: <https://pypi.org/project/GetOldTweets3/> [Accessed 14 March 2020].

[10] 6. R, "Learn Naive Bayes Algorithm | Naive Bayes Classifier Examples", *Analytics Vidhya*, 2020. [Online]. Available: https://www.analyticsvidhya.com/blog/2017/09/naive-bayes-explained/. [Accessed: 04- Apr- 2020].

[11] U. code), "Understanding Support Vector Machines(SVM) algorithm (along with code)", *Analytics Vidhya*, 2020. [Online]. Available: https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/. [Accessed: 04- Apr- 2020].

[12] J. Brownlee, "Logistic Regression for Machine Learning", *Machine Learning Mastery*, 2020. [Online]. Available: https://machinelearningmastery.com/logistic-regression-for-machine-learning/. [Accessed: 04- Apr- 2020].

[13] "Mazajak", *Mazajak.inf.ed.ac.uk*, 2020. [Online]. Available: http://mazajak.inf.ed.ac.uk:8000/. [Accessed: 04- Apr- 2020].

[14] "Code Faster with Line-of-Code Completions, Cloudless Processing", *Kite.com*, 2020. [Online]. Available: https://kite.com/python/docs/nltk.RegexpTokenizer. [Accessed: 04- Apr- 2020].

[15] "matplotlib.pyplot — Matplotlib 3.1.2 documentation", *Matplotlib.org*, 2020. [Online]. Available: https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.pyplot.html. [Accessed: 04- Apr- 2020].

[16] "Python Counter - Python Collections Counter - JournalDev", *JournalDev*, 2020. [Online]. Available: https://www.journaldev.com/20806/python-counter-python-collections-counter. [Accessed: 04- Apr- 2020].

[17] "Using the Requests Library in Python", *Python For Beginners*, 2020. [Online]. Available: https://www.pythonforbeginners.com/requests/using-requests-in-python. [Accessed: 04- Apr- 2020].

[18] "JSON — The Hitchhiker's Guide to Python", *Docs.python-guide.org*, 2020. [Online]. Available: https://docs.python-guide.org/scenarios/json/. [Accessed: 04- Apr- 2020].