UTM
UNIVERSITI TEKNOLOGI MALAYSIA

Twitter US Airlines Sentiment Analysis

Group Members:

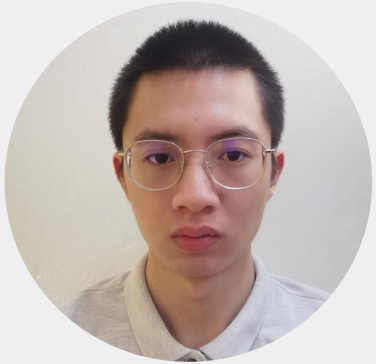
1) Tan Fei Zhi (MCS221016)
2) Mohd Fikri bin Mohd Hanim (MCS211043)
3) Mohamad Nor Azni bin Zainal Abidin (MCS211050)
4) Teh Hung Wei (MCS212011)

Team Members

Fei Zhi
MCS221016
Master of Data Science

Fikri
MCS211043
Master of Data Science

Azni
MCS211050
Master of Data Science



Hung Wei
MCS212011
Master of Computer Science

Executive Summary

Airline companies provide air transportation for passengers and cargo, with the goal of making travel more convenient.

Many reports of issues with US airlines in social media and news. Satisfaction and opinions of passengers are crucial for enhancing airlines services. sentiment analysis is conducted in this study to interpret and understand these opinions,

This study is conducted to accurately extract passengers' opinions from a large number of unstructured review texts from Twitter and classify them into sentiment classes.

Background of the Case

The US Department of Transportation revealed that Frontier Airlines was one of six carriers - and the only US-based one - that will have to reimburse customers over $600 million for flights that were delayed or canceled since March 2020.

Receiving many negative feedback from passengers on twitters. Customer satisfaction is a crucial factor in a company's success. Analyzing customer sentiment from written feedback can be time-consuming and labor-intensive for airlines.

Problems Statement

01

What are passengers' opinions on the service provided by US airlines? (positive & negative words)

02

What are the reasons for negative reviews that are tweeted by passengers?

03

What are the main issues happened to each airline?

04

Which machine learning algorithm is suitable for sentiment analysis?

Objectives

Objective 1

To identify the main issues of airlines companies based on positive and negative tweets

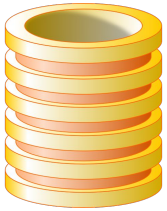
Objective 2

To utilize the insights gained from sentiment analysis study for airlines to enhance services

Objective 3

To identify the best machine algorithm for airline sentiment analysis

Dataset



This data originally came from Crowdflower's Data for Everyone library.

As the original source says,

> A sentiment analysis job about the problems of each major U.S. airline. Twitter data was scraped from February of 2015 and contributors were asked to first classify positive, negative, and neutral tweets, followed by categorizing negative reasons (such as "late flight" or "rude service").

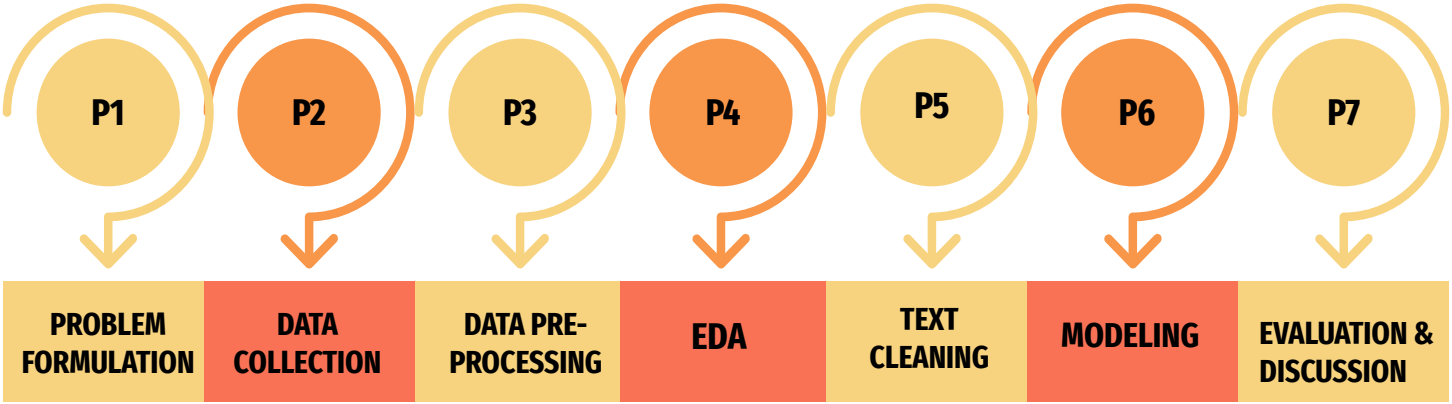
Data Source: (Source: Kaggle | Twitter US Airline Sentiment)

```
df.shape
```

```
(14640, 15)
```

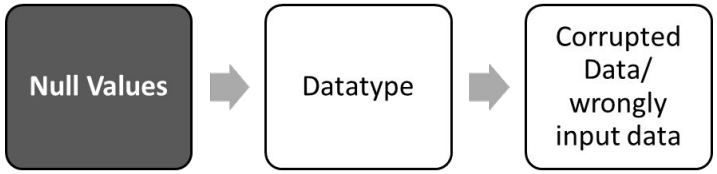
```
0   tweet_id
1   airline_sentiment
2   airline_sentiment_confidence
3   negativereason
4   negativereason_confidence
5   airline
6   airline_sentiment_gold
7   name
8   negativereason_gold
9   retweet_count
10  text
11  tweet_coord
12  tweet_created
13  tweet_location
14  user_timezone
```

Overview of Methodology



Note: P=Process

DATA PRE-PROCESSING



Check total of null values

```
df.isnull().sum()
```

```
tweet_id                          0
airline_sentiment                 0
airline_sentiment_confidence      0
negativereason                 5462
negativereason_confidence      4118
airline                           0
airline_sentiment_gold        14600
name                              0
negativereason_gold           14608
retweet_count                     0
text                              0
tweet_coord                   13621
tweet_created                     0
tweet_location                 4733
user_timezone                  4820
dtype: int64
```

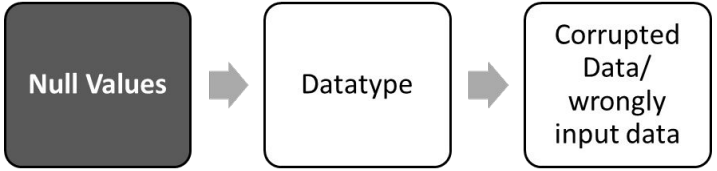
Check percentage of null values

```
((df.isnull() | df.isna()).sum() * 100 / df.index.size).round(2)
```

```
tweet_id                          0.00
airline_sentiment                 0.00
airline_sentiment_confidence      0.00
negativereason                   37.31
negativereason_confidence        28.13
airline                           0.00
airline_sentiment_gold           99.73
name                              0.00
negativereason_gold              99.78
retweet_count                     0.00
text                              0.00
tweet_coord                      93.04
tweet_created                     0.00
tweet_location                   32.33
user_timezone                    32.92
dtype: float64
```

If percentage of null values > 90%, the columns will be dropped since there are too many missing values.

DATA PRE-PROCESSING



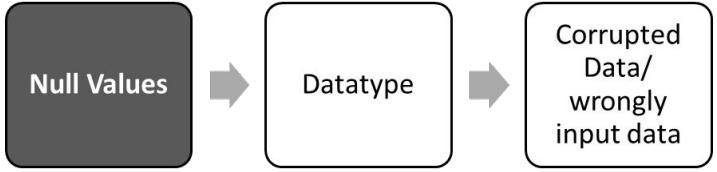

```
df.drop('airline_sentiment_gold', inplace=True, axis=1)
df.drop('negativereason_gold', inplace=True, axis=1)
df.drop('tweet_coord', inplace=True, axis=1)
```

```
df.shape
```

```
(14640, 12)
```

There are three attributes (airline_sentiment_gold, negativereason_gold and tweet_cord) dropped in this case. Now, the number of attributes is decreasing from 15 to 12.

DATA PRE-PROCESSING



Fill null values with NaN

```
df=df.mask(df == '')
```

	tweet_id	airline_sentiment	airline_sentiment_confidence	negativereason	negativereason_confidence	airline
0	570306133677760513	neutral	1.0000	NaN	NaN	Virgin America
1	570301130888122368	positive	0.3486	NaN	0.0000	Virgin America
2	570301083672813571	neutral	0.6837	NaN	NaN	Virgin America
3	570301031407624196	negative	1.0000	Bad Flight	0.7033	Virgin America
4	570300817074462722	negative	1.0000	Can't Tell	1.0000	Virgin America

DATA PRE-PROCESSING

Check data types

```
df.dtypes
```

```
tweet_coord                 object
tweet_created               object
tweet_location              object
```

Convert twee_creted from object to datetime

```
df['tweet_created'] = pd.to_datetime(df['tweet_created'])
```

```
tweet_created    datetime64[ns, pytz.FixedOffset(-480)]
```

Exploratory Data Analysis(EDA)

On the EDA process we analysis about:

1. Check the airlines in dataset.
2. Calculate the tweet times for each airline companies.
3. Data Visualization
    a. Bar Chart
    b. Pie Chart
    c. Wordcloud

DataFrame(). When working with a single column of a DataFrame, the unique() method is utilised and returns all of the column's unique items. The method outputs a DataFrame that includes the distinct column elements and their accompanying index labels.

Check the airlines in dataset

```
df.airline.unique()
```

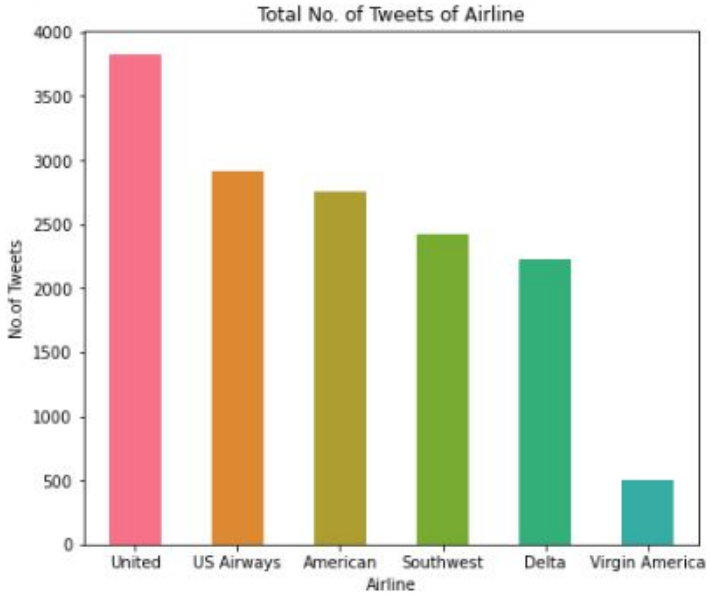
```
array(['Virgin America', 'United', 'Southwest', 'Delta', 'US Airways',
       'American'], dtype=object)
```

Exploratory Data Analysis(EDA)

On this analysis, the finding is United has the most number of tweets while Virgin America has the least number of tweets.

```
colors=sns.color_palette('husl',10)
pd.Series(df['airline']).value_counts().plot(kind="bar",color=colors,figsize=(7,6),fontsize=10,rot=0,title='Total No. of Tweets of Airline')
plt.xlabel('Airline',fontsize=10)
plt.ylabel('No.of Tweets',fontsize=10)
```

```
Text(0, 0.5, 'No.of Tweets')
```



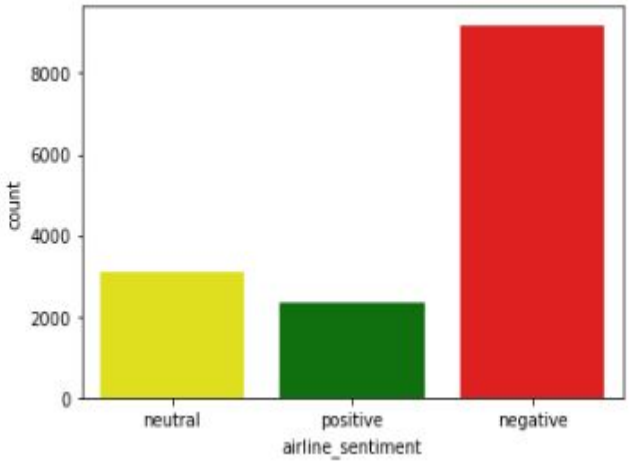
Exploratory Data Analysis(EDA)

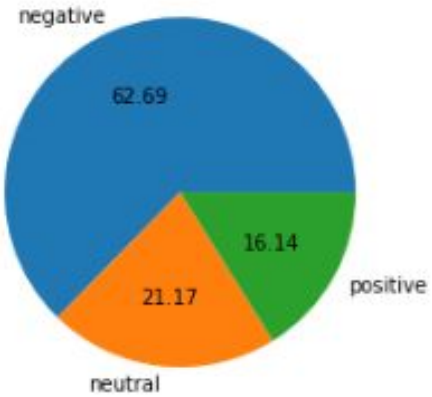
Next is about the level of sentiment. We find that data have 3 level of sentiment.

1. Negative - 9178
2. Neutral - 3099
3. Positive - 2363

```
df["airline_sentiment"].value_counts()
```

```
negative    9178
neutral     3099
positive    2363
Name: airline_sentiment, dtype: int64
```





The negative reactions of passengers toward airlines recorded the highest with 9178 counts (62.69%) while the positive reactions recorded the lowest with 2363 counts (16.14%). The balance is neutral with 3099 counts (21.17%).

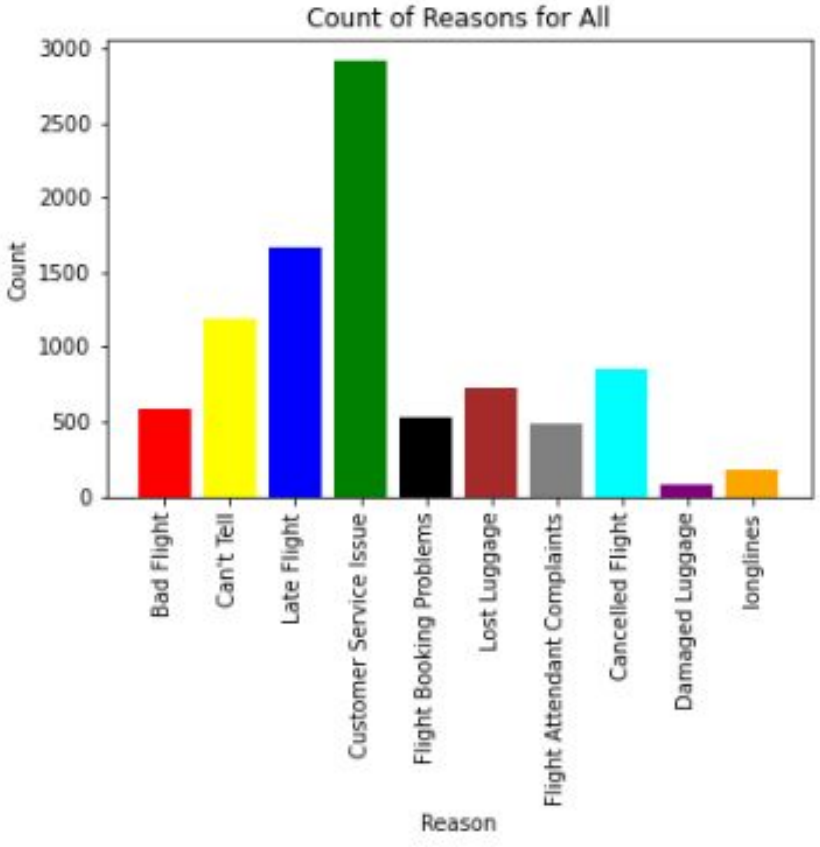
Exploratory Data Analysis(EDA)

-Problem Statement 02
-Objective 1

From dataset, what we get on the negative analysis is about the

1. Bad flight
2. Can't tell
3. Late flight
4. Customer service issue
5. Flight booking problems
6. Lost luggage
7. Flight attendant complaints
8. Cancelled flight
9. Damaged luggage
10. Longlines

Where the top issue is about customer service issue.
The least issue is about damaged luggage.



Exploratory Data Analysis(EDA)

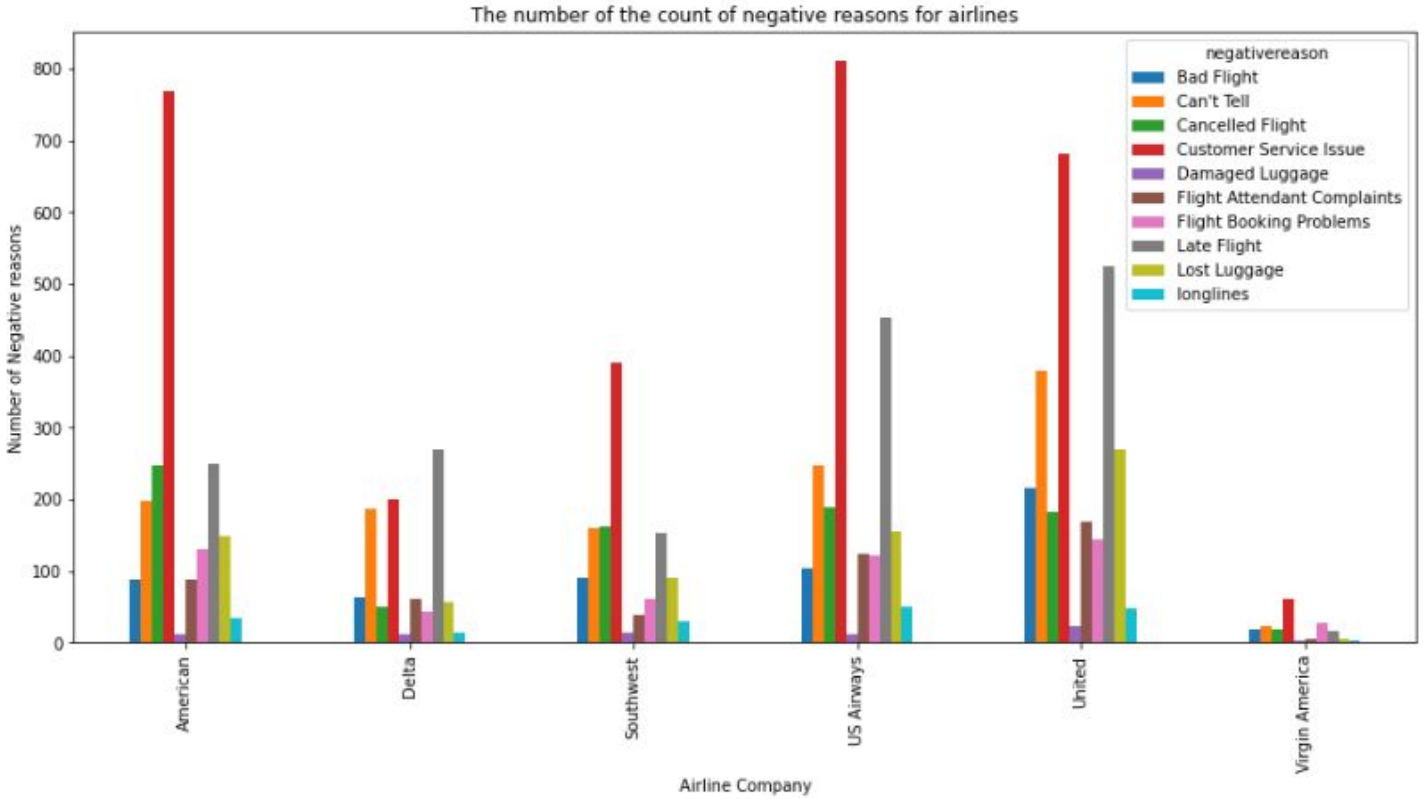
-Problem Statement 02
-Objective 1

This is about the count of negative reason by each airline.

Where the top issue is about customer service issue and second follow by late flight issue.

The US Airways is the highest from other airline on the negative reason.

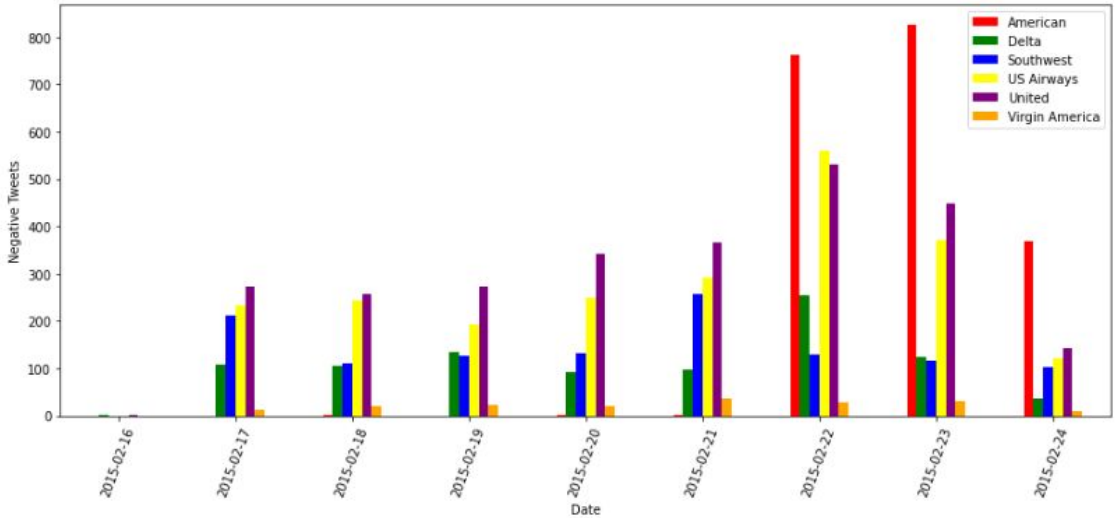
The Virgin America is low negative reason then the other airline.



Exploratory Data Analysis(EDA)

-Problem Statement 03
-Objective 2

The date and the reason of the same day for both airlines are among the pieces of information that might be analysed.



tweet_created	airline	airline_sentiment	
2015-02-16	Delta	negative	1
		neutral	1
	United	negative	2
2015-02-17	Delta	negative	108
		neutral	86
			...
2015-02-24	United	neutral	49
		positive	25
	Virgin America	negative	10
		neutral	6
		positive	13

Length: 136, dtype: int64

Exploratory Data Analysis(EDA)

-Problem Statement 01
-Objective 1

WordCloud is one of the data sets that we can examine. It is a graphic depiction of a text's most commonly occurring terms. The word will appear in tweets more frequently the larger it is shown.

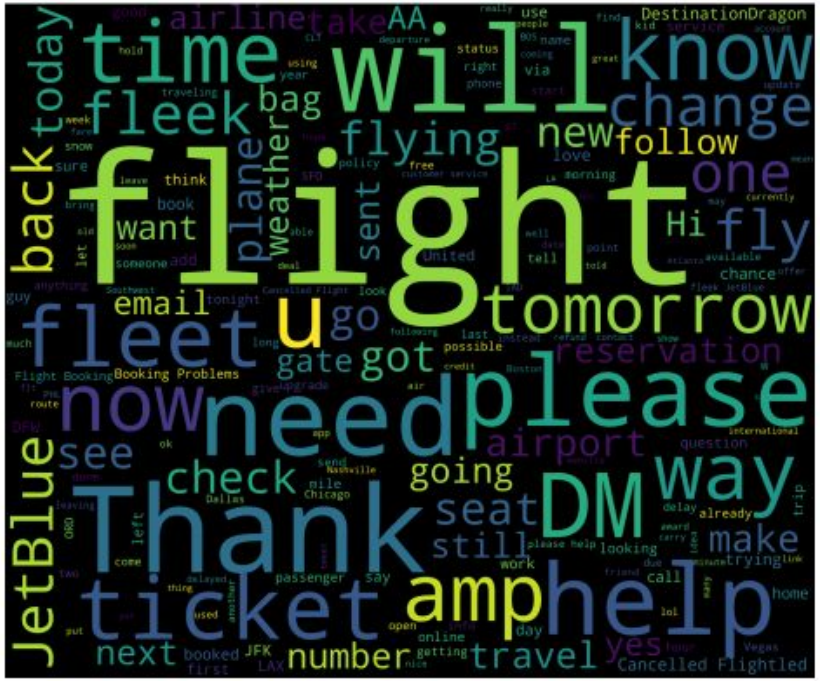
```
new_df=df[df['airline_sentiment']=='negative']
words = ' '.join(new_df['text'])
cleaned_word = " ".join([word for word in words.split()
                            if 'http' not in word
                                and not word.startswith('@')
                                and word != 'RT'
                            ])
wordcloud = WordCloud(stopwords=STOPWORDS,
                      background_color='red',
                      width=3000,
                      height=2500
                     ).generate(cleaned_word)
plt.figure(1,figsize=(12, 12))
plt.imshow(wordcloud)
plt.axis('off')
plt.show()
```

The word that appeared in the tweets with the greatest negative connotations was highlighted in red.

Exploratory Data Analysis(EDA)

-Problem Statement 01
-Objective 1

The term that appeared in the tweets with the most neutral connotations was put below in black to symbolise neutrality.



Exploratory Data Analysis(EDA)

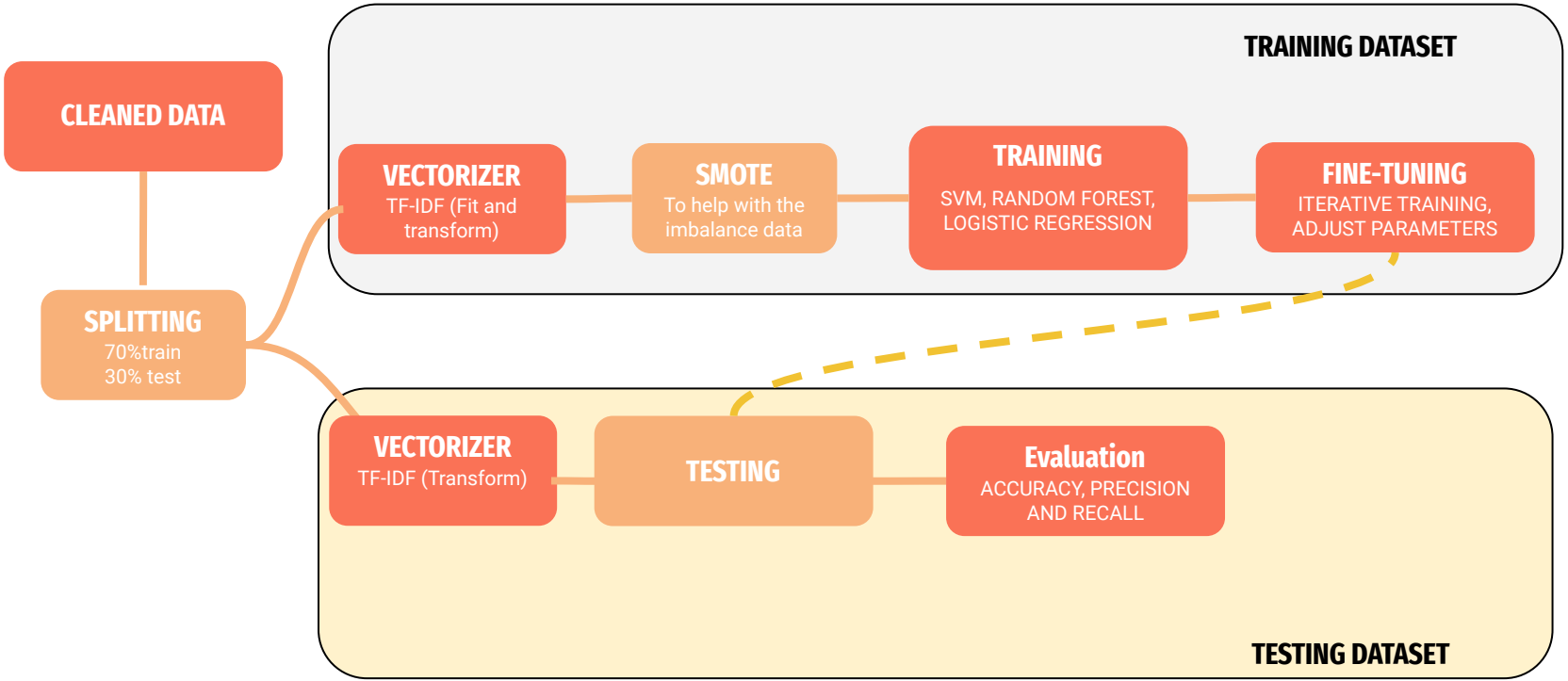
-Problem Statement 01
-Objective 1

The word that appeared in tweets with the most positive connotations was highlighted in green to signify positivity (see illustration below).

Text Pre-processing

Stopwords	@AmericanAir hook me up with a free trip to Barbados and I will tell you the secret beaches to see	'@AmericanAir hook free trip Barbados I tell secret beaches see'
URL	@AmericanAir: Bet these birds wish they'd flown south for the #winter... http://t.co/tY9C0Gae2o?€? Lol	@AmericanAir: Bet these birds wish they'd flown south for the #winter... Lol
Punctuation	@AmericanAir I'm sure they did. It's certainly chilly back East today! ????	@AmericanAir I'm sure they did It's certainly chilly back East today
@username	@AmericanAir oh it's nothing my issue was already resolved and I am in the air	" oh it's nothing my issue was already resolved and I am in the air"
emojis	-	-
alphaneumeric	@VirginAmerica Love the team running Gate E9 at LAS tonight	VirginAmerica Love the team running Gate E 9 at LAS tonight
Decontraction	@VirginAmerica Your site doesn't work on iPhone or iPad.don't have a computer #help	VirginAmerica Your site does not work on iPhone or iPad. do not have a computer #help

MACHINE LEARNING PROCESS



Preparation for ML Modelling Process

1. Main library used: sklearn, matplotlib
2. Recap: from previous part, we had obtained the clean dataset.

```
X = df['final_text']
Y = df['airline_sentiment']
```

```
: # 70,30
X_train , X_test , Y_train , Y_test = train_test_split(X , Y , test_size=0.3)
```

3. Splitting ratio=> 70:30

```
tfid = TfidfVectorizer(use_idf=True)
X_train_vect =  tfid.fit_transform(X_train)

#the first column shows where and what is the unique word located in the dataframe.
#just fit X_final to our ML algorithm, and it would become sample and features for row and column respectively.
print(X_train_vect)
```

4. Apply TFI-DF (Term frequency-inverse document frequency)
5. Perform SMOTE oversampling

```
smote = SMOTE()
X_sm,Y_sm = smote.fit_resample(X_train_vect,Y_train)
```

FEATURE VECTOR

In machine learning and natural language processing, a feature vector is a numerical representation of a set of features or attributes of an object or instance. The feature vector may also include other attributes of the text, such as the length of the text, the presence of certain words or phrases, or the context in which the text appears.

TF - IDF

•numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus.

•used to convert a collection of documents (a corpus) into numerical vectors that represent the tf-idf values of the words in each document

•help identify the words and phrases that are most strongly associated with a particular sentiment (positive, negative, or neutral)

- $\text{IDF}(t) = \log \frac{1+n}{1+\text{df}(t)} + 1$

TF - IDF

```
tfid = TfidfVectorizer(use_idf=True)
X_final =  tfid.fit_transform(X)
#the first column, shows where and what is the unique word located in the dataframe. (rows
#just feed X_final to our ML algorithm, and it would become sample and features for row ar
print(X_final)
```

Python

```
Output exceeds the size limit. Open the full output data in a text editor
  (0, 8284)     1.0
  (1, 9441)     0.5715144216199972
  (1, 3310)     0.3059111495163962
  (1, 1864)     0.4991825649009825
  (1, 123)      0.4357510390241188
  (1, 7213)     0.3751365529646431
  (2, 9945)     0.36395420765197306
  (2, 435)      0.34633875964424976
  (2, 9457)     0.34712077682710796
  (2, 6316)     0.29810885808388093
  (2, 5876)     0.45320923958874404
  (2, 6240)     0.48311623162963613
  (2, 9780)     0.31531756249018034
```

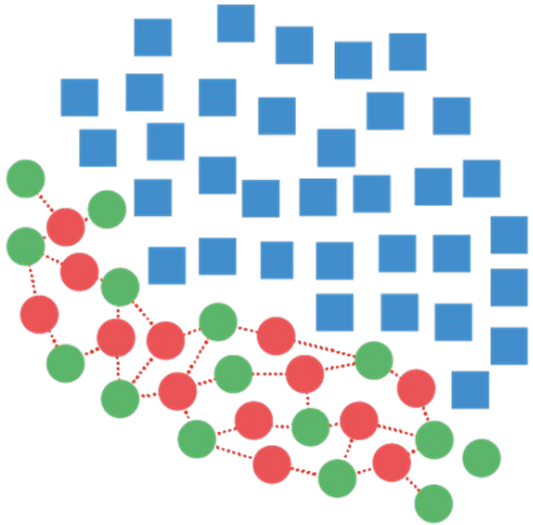
```
  (0, 8284)     1.0
  (1, 9441)     0.5715144216199972
  (1, 3310)     0.3059111495163962
```

(Document ID, Token iD)

Tf-idf score

SMOTE

Original Dataset

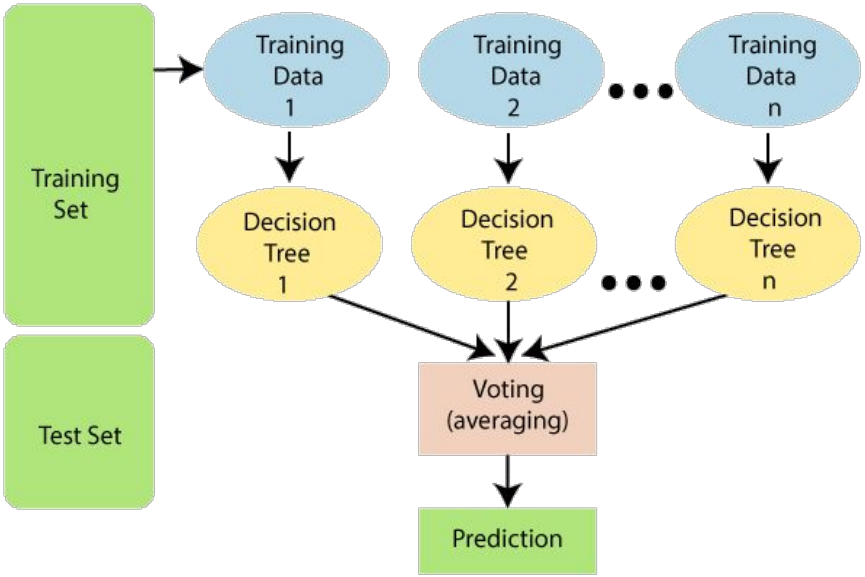


Generating Samples

Resampled Dataset

RANDOM FOREST

"A random forest is an ensemble machine learning model that consists of a collection of decision trees. The basic idea behind a decision tree is to split the training data into subsets based on certain features and use these splits to make predictions. A random forest works by training many decision trees on random subsets of the training data and then averaging the predictions made by each tree to make a final prediction."

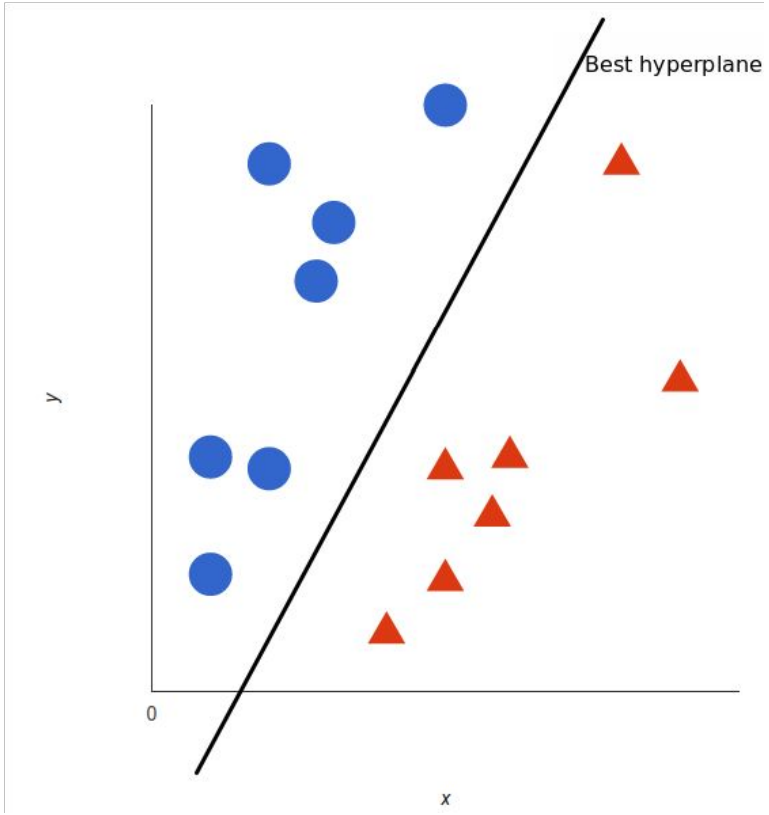


SVM

Constructing an the best hyperplanes

"A support vector machine constructs a hyperplane or set of hyperplanes in a high- or infinite-dimensional space, which can be used for classification, regression, or other tasks. Intuitively, a good separation is achieved by the hyperplane that has the largest distance to the nearest training-data points of any class, since in general the larger the margin the lower the generalization error of the classifier."

Bioinspire Algorithm and application (2004)



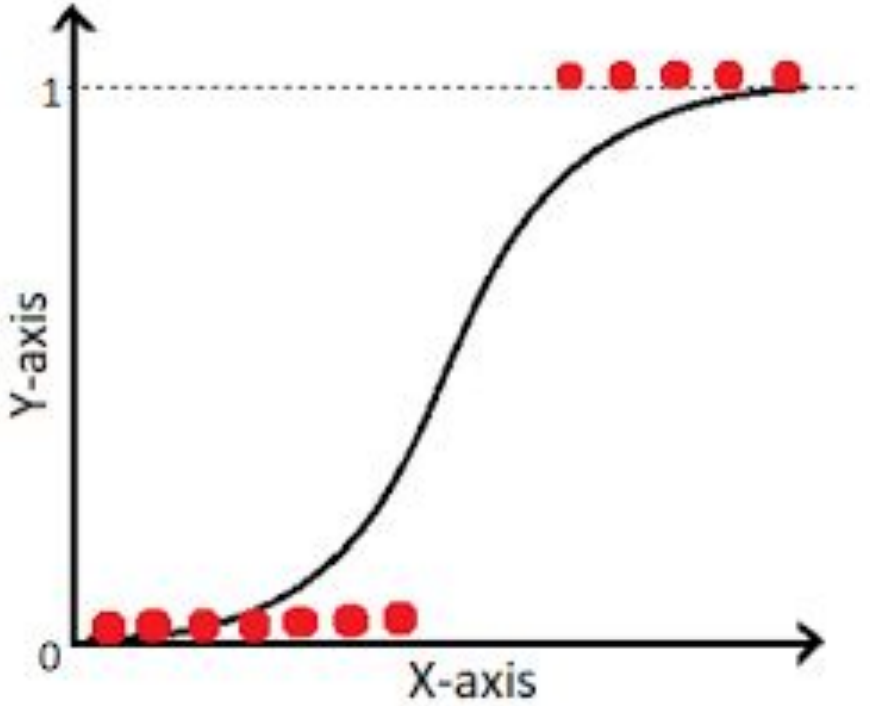
SVM

```
class sklearn.svm.SVC(*, C=1.0, kernel='rbf', degree=3, gamma='scale', coef0=0.0, shrinking=True,
probability=False, tol=0.001, cache_size=200, class_weight=None, verbose=False, max_iter=-1,
decision_function_shape='ovr', break_ties=False, random_state=None)                    [source]
```

c	1 , 10 , 100, 1000	Penalty parameter, (High C, high penalty)
Kernel	Linear , polynomial, Radial Basis function,	Map the data into higher-dimensional space
Gamma	1, 0.1 , 0.01	Determine how much influence a single training example has

LOGISTIC REGRESSION

"This type of statistical model (also known as *logit model*) is often used for classification and predictive analytics. Logistic regression estimates the probability of an event occurring, such as voted or didn't vote, based on a given dataset of independent variables. Since the outcome is a probability, the dependent variable is bounded between 0 and 1. In logistic regression, a logit transformation is applied on the odds—that is, the probability of success divided by the probability of failure."



Classification and Modelling

- Random Forest (RF)
- Support Vector Machine (SVM)
- Logistic Regression (LR)

Random Forest

```
from sklearn.ensemble import RandomForestClassifier
rf_model = RandomForestClassifier(n_estimators=30,max_features="sqrt",random_state=None)

#n_estimators value was measured in next few sections
#selected based on accuracy score and lesser training time

rf_model.fit(X_sm,Y_sm)
X_test_vect =  tfid.transform(X_test)
Y_pred_rf = rf_model.predict(X_test_vect)
print('done')
```

```
rf_model.fit(X_sm,Y_sm)
X_test_vect =  tfid.transform(X_test)
```

Classification Report

```
#from sklearn.metrics import accuracy_score, confusion_matrix, classification_report, plot_confusion_matrix
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
import matplotlib.pyplot as plt
cr_rf = classification_report(Y_test,Y_pred_rf)
print(cr_rf)
```

```
i=10
x_rf=[]
y_rf=[]
while i <= 100:
#     print(f"train {i}")
    rf_model = RandomForestClassifier(n_estimators=i,max_features="sqrt",random_state=None)
    rf_model.fit(X_sm,Y_sm)
    Y_pred_rf = rf_model.predict(X_test_vect)
    Accuracy_rf = accuracy_score(Y_test,Y_pred_rf)
    x_rf.append(i)
    y_rf.append(Accuracy_rf)
    print(i , " : " , Accuracy_rf)
    i+=5
```

SVM

```
from sklearn.svm import SVC
from sklearn.model_selection import GridSearchCV
```

```
param_grid_SVM = {'C': [10, 100, 1000],
              'gamma': [1, 0.1, 0.01],
              'kernel': ['rbf', 'linear']}
```

```
grid_SVM = GridSearchCV(SVC(), param_grid_SVM, refit = True, verbose = 3)
```

```
grid_SVM.fit(X_train_vect, Y_train)
```

```
# print best parameter after tuning
print(grid_SVM.best_params_)

# print how our model looks after hyper-parameter tuning
print(grid_SVM.best_estimator_)
```

```
{'C': 10, 'gamma': 0.1, 'kernel': 'rbf'}
SVC(C=10, gamma=0.1)
```

```
#Fitting according to the best params
svm_clf2 = SVC(kernel = 'rbf', C = 10, gamma = 0.1)
svm_clf2.fit(X_test_vect, Y_test)
```

Logistic Regression

```
from sklearn.linear_model import LogisticRegression
LR=LogisticRegression()
Y_pred_LR = LR.fit(X_train_vect, Y_train).predict(X_test_vect)
print(confusion_matrix(Y_test, Y_pred_LR))
print('\n')
print('Accuracy for LR (before optimize) is ', accuracy_score(Y_test,Y_pred_LR),"\n")
print(classification_report(Y_test, Y_pred_LR))
```

```
solvers_LR = ['newton-cg','sag', 'saga','lbfgs']
c_LR = [100, 10, 1.0, 0.1, 0.01]
penalty_LR = ['l1', 'l2']
```

```
#for optimization process, the main hyperparameter tuned were solver, penalty, and C
# solver  : 'newton-cg','sag', 'saga','lbfgs' ------------->default : Lbfgs
#    C    : 100, 10, 1.0, 0.1, 0.01 ---------------------->default : 1.0
# penalty : Decide later #l1 : Lasso(absolute magnitude) #L2 : Ridge(squared magnitude) --->default: L2
from sklearn.model_selection import GridSearchCV
param_LR = dict(solver=solvers_LR,C=c_LR, penalty = penalty_LR )
grid_LR = GridSearchCV(estimator=LR, param_grid=param_LR, n_jobs=-1, verbose=3, scoring='accuracy',error_score=0)
# fitting the model for grid search
grid_LR.fit(X_train_vect, Y_train)
# print best parameter after tuning
print("Best parameters measured :",grid_LR.best_params_ , "\n")
grid_y_pred_LR = grid_LR.predict(X_test_vect)
# print classification report for the best param
print("The optimized LR model : \n",classification_report(Y_test, grid_y_pred_LR))
```

Results and Findings

- Random Forest - Accuracy = 0.9087

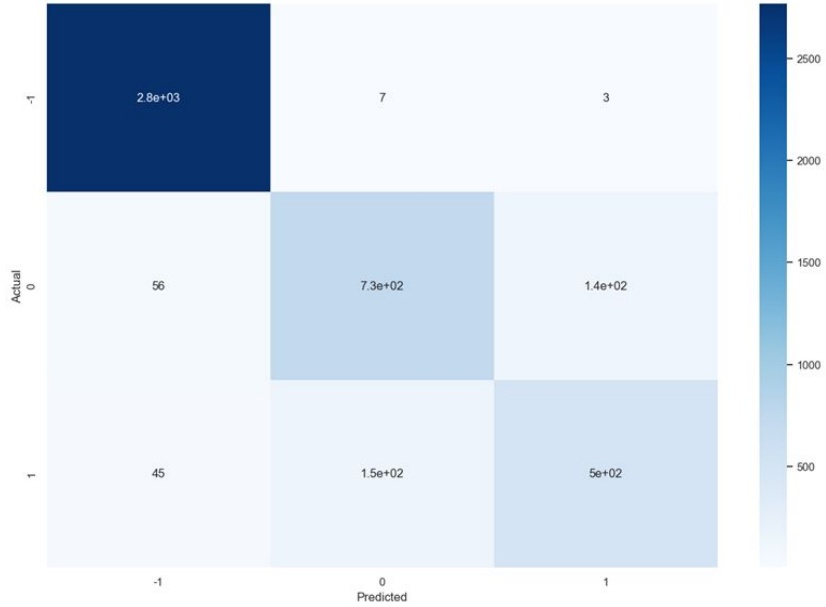
Classification Report

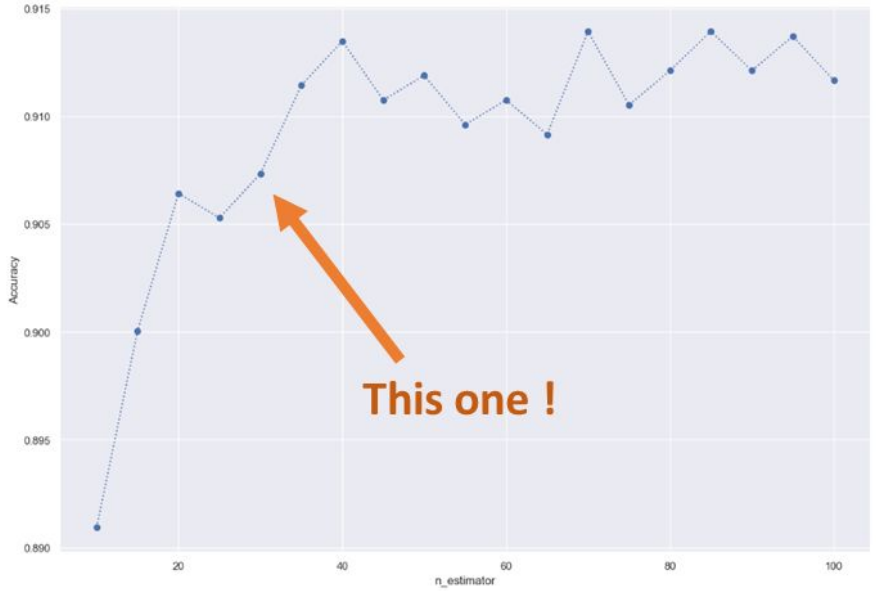
```
In [56]: #from sklearn.metrics import accuracy_score, confusion_matrix, classification_report, plot_confusion_matrix
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
import matplotlib.pyplot as plt
cr_rf = classification_report(Y_test,Y_pred_rf)
print(cr_rf)
```

```
              precision    recall  f1-score   support

          -1       0.96      1.00      0.98      2777
           0       0.82      0.79      0.80       924
           1       0.78      0.72      0.75       691

    accuracy                           0.91      4392
   macro avg       0.85      0.83      0.84      4392
weighted avg       0.91      0.91      0.91      4392
```



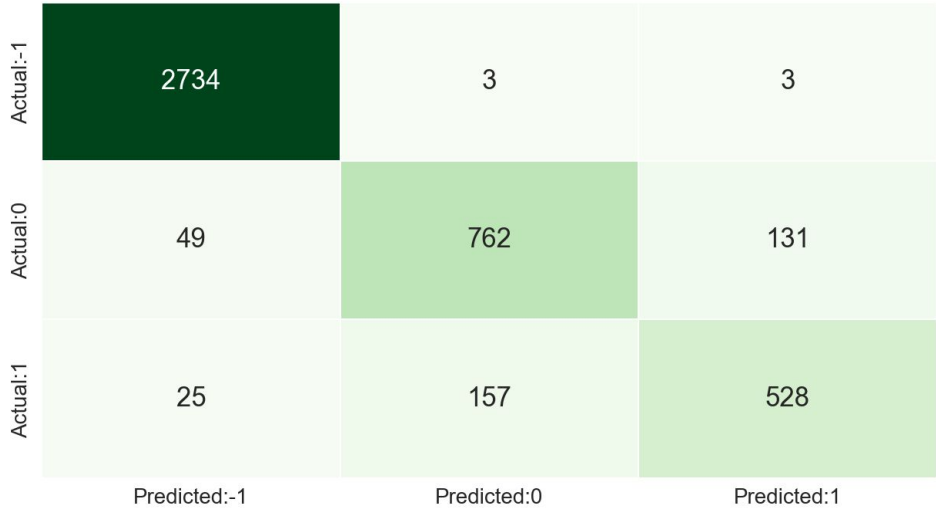


Results and Findings

- SVM - Accuracy = 0.92

```
{'C': 10, 'gamma': 0.1, 'kernel': 'rbf'}
SVC(C=10, gamma=0.1)
```

	precision	recall	f1-score	support
-1	0.97	1.00	0.99	2824
0	0.82	0.82	0.82	897
1	0.82	0.73	0.77	671
accuracy			0.92	4392
macro avg	0.87	0.85	0.86	4392
weighted avg	0.92	0.92	0.92	4392

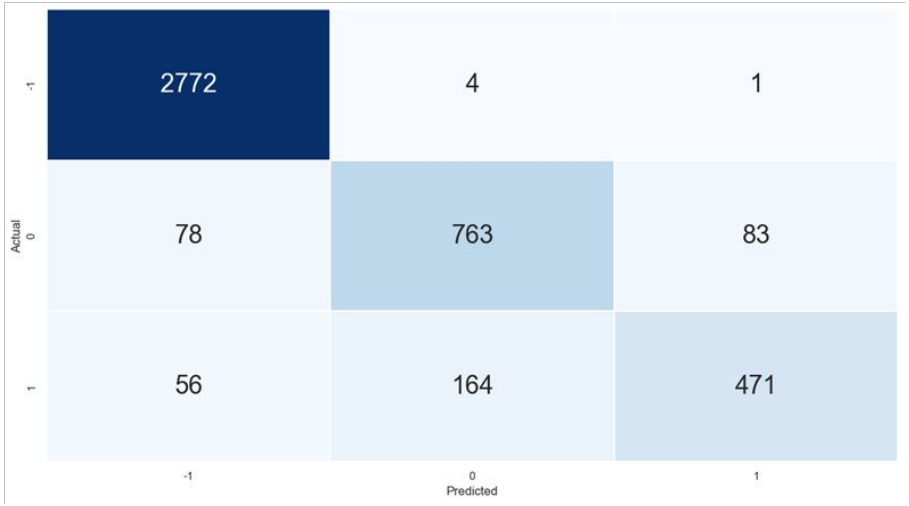


Results and Findings

- Logistic Regression- Accuracy = 0.9187

```
Best parameters measured : {'C': 1.0, 'penalty': 'l1', 'solver': 'saga'}
```

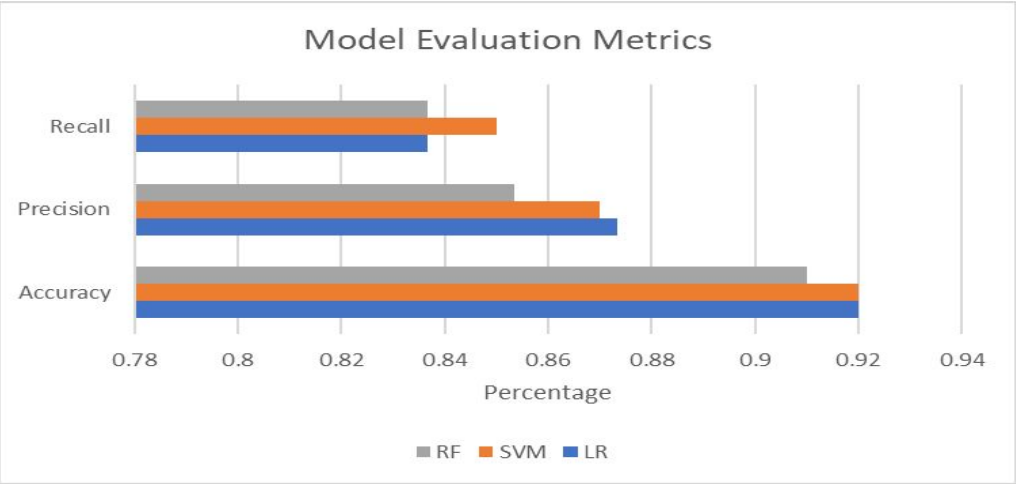
	precision	recall	f1-score	support
-1	0.95	1.00	0.98	2777
0	0.82	0.83	0.82	924
1	0.85	0.68	0.76	691
accuracy			0.91	4392
macro avg	0.87	0.84	0.85	4392
weighted avg	0.91	0.91	0.91	4392



-Problem Statement 04
-Objective 3

Results and Findings - Summary

	LR	SVM	RF
Accuracy	0.92	**0.92**	0.91
Precision (-1,0,1)	0.95,0.82,0.85	**0.97,0.82,0.82**	0.96,0.82,0.78
Recall (-1,0,1)	1.00,0.83,0.68	**1.00,0.82,0.73**	1.00,0.79,0.72



$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Recall} = \frac{TP}{TP + FN}$$

Conclusion

In this study, we have completed Sentiment Analysis on Twitter US Airlines. The purpose of this study is to know what are the opinions of customers towards US airlines based on twitter data. It might be difficult to determine the primary difficulties facing airline firms based on favourable and unfavourable tweets since there are a lot of opinions and comments towards airlines. Sentiment analysis can help quickly and accurately process large amounts of text data, saving time and resources that would otherwise be spent manually reviewing and analyzing the data.

The problem statement and objectives have been defined and briefly explained in the section above. The dataset used is a ready dataset that obtained from (Source: Kaggle | Twitter US Airline Sentiment). First of all, the data pre-processing , EDA and text pre-processing using TFI-DF and SMOTE have been conducted to make the data more meaningful to be used in the next phases. After completing with data processing part, machine learning took place to train and test the dataset using Sentiment Analysis method. The detailed processes can be referred to the slides above. There are three Machine Learning algorithms are selected including Random Forest, Support Vector Machine and Logistic Regression.

Conclusion (Cont..)

In a nutshell, opinions of customers are really significant for airlines to know their services. Sentiment analysis is suitable to identify the opinions of the passengers towards airline. At the end of the study, the objectives of the study have been achieved and the problem statement have been answered by the works that have been done in this study. Besides that, we summarized our results and findings in the slide with comparing on the algorithms. SVM shows the best performance in this study. By using sentiment analysis to identify customer sentiment towards airlines, airlines companies can quickly identify and address customer complaints or concerns. This can lead to improved customer satisfaction and loyalty. The insights gained from this study may help airlines to enhance services. Lastly, the work distribution is attached at the next page. Everyone is committed to the project and showing high cooperation.

Work Distribution

	Fei Zhi	Fikri	Azni	Hung Wei
Data Collection	✓	✓	✓	✓
Data Pre-processing & EDA	✓			
Text Cleaning	✓	✓		
Machine Learning		✓	✓	✓
Results and Discussions			✓	✓
Slides	✓		✓	✓