Sentiment Analysis of Tweets

Oleh :

Dhoni Hanif Supriyadi

1. Business Understanding

Pada tahapan ini, seorang Data Scientist akan menelaah bisnisnya terlebih dahulu seperti apa yang dibicarakan dalam bisnis ini? Apa bisnis ini termasuk klasifikasi atau regresi? Dan hal lainnya. Bisnis ini membicarakan tentang Sentiment Analysis terhadap twitter. Seperti yang kita ketahui, beberapa kata tidak dapat ditolerir seperti kata – kata yang negative sehingga hal ini butuh kita nilai bahwa kata – kata ini buruk. Sebelum itu, kita harus mengumpulkan datanya terlebih dahulu dan menganalisisnya terlebih dahulu.

2. Data Collection

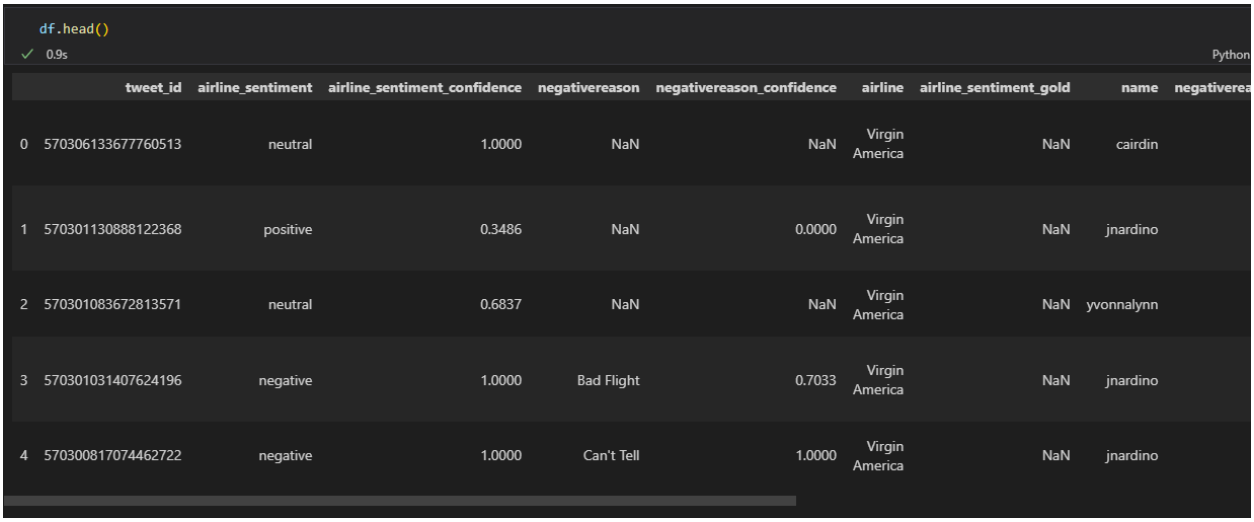
Hal pertama yang harus kita lakukan adalah mengumpulkan datanya terlebih dahulu. Disini, kita sudah mendapatkan datanya melalui kaggle.com. Lalu, kita akan lakukan pembacaan datanya. Namun, sebelum itu, mari import library yang dibutuhkan terlebih dahulu.

```
# Basic Operation
import pandas as pd
import numpy as np

# Text Preprocessing & Cleaning
from sklearn.feature_extraction.text import TfidfVectorizer
from nltk.corpus import stopwords
import re
from wordcloud import WordCloud,STOPWORDS
from nltk import SnowballStemmer


from sklearn.model_selection import train_test_split # Split Data
from imblearn.over_sampling import SMOTE # Handling Imbalanced

# Model Building
from sklearn.naive_bayes import MultinomialNB
from sklearn.svm import SVC


from sklearn.metrics import classification_report , confusion_matrix , accuracy_score # Performance Metrics


# Data Visualization
import matplotlib.pyplot as plt
from wordcloud import WordCloud
import seaborn as sns
import warnings


warnings.filterwarnings('ignore')
%matplotlib inline
✓ 2.1s
```

Disini, kita membutuhkan beberapa library seperti pandas, numpy, scikit-learn, nltk, re, wordcloud, seaborn, warnings, dan matplotlib.

Selanjutnya, mari lakukan pembacaan data seperti berikut.

Data berhasil diload. Sekarang, cek 5 data dari atas.



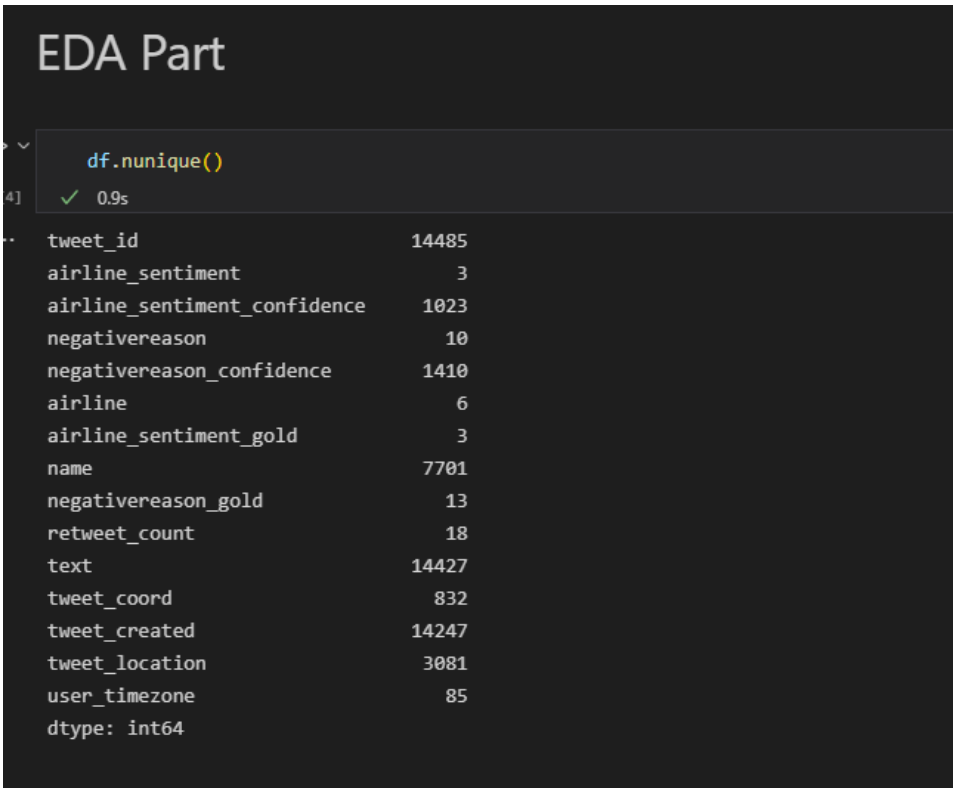
```
df.head()
```

✓ 0.9s Python

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

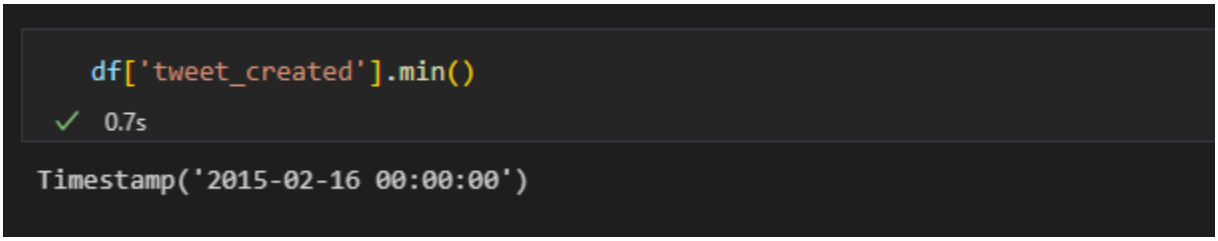
3. Exploratory Data Analysis (EDA)

Hal pertama yang harus kita lakukan adalah mengecek jumlah nilai unique seperti berikut.



Disini dikatakan bahwa tweet_id, text, dan tweet_created memiliki nilai unik yang sangat besar. Seperti yang kita ketahui, ada beberapa variabel yang menyimpan data dengan format yang salah seperti variabel tweet_created. Oleh karena itu, tweet_created memiliki jumlah nilai unik yang sangat besar sehingga kita harus memperbaiki format data tersebut seperti berikut.

Data minimal dari tweet_created adalah sebagai berikut.



Sedangkan data maksimal dari tweet_created adalah sebagai berikut.

```
df['tweet_created'].max()
✓ 0.5s
Timestamp('2015-02-24 00:00:00')
```

Lalu, mari cek kembali jumlah nilai unik dari variabel tweet_created seperti berikut.

```
df['tweet_created'].nunique()
✓ 0.7s
9
```

Setelah kita perbaiki format data, maka jumlah nilai unik berubah menjadi 9. Selanjutnya, mari cek jumlah tweets berdasarkan tweet_created seperti berikut.

```
numberoftweets = df.groupby('tweet_created').size()
✓ 0.7s

numberoftweets
✓ 0.9s
tweet_created
2015-02-16       4
2015-02-17    1408
2015-02-18    1344
2015-02-19    1376
2015-02-20    1500
2015-02-21    1557
2015-02-22    3079
2015-02-23    3028
2015-02-24    1344
dtype: int64
```

Data terbesar ada pada tanggal 22 February 2015, sedangkan data terkecil berada pada tanggal 16 February 2015. Sekarang, mari cek missing values seperti berikut.

```
print("Percentage null or na values in df")
((df.isnull() | df.isna()).sum() * 100 / df.index.size).round(2)
```

✓ 0.1s

```
Percentage null or na values in df

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
```

Dalam hal ini, kita menampilkan missing value dari masing – masing data dalam bentuk persentase. Missing value terbesar berada pada data negativereason_gold yaitu sebesar 99.78%. beberapa data juga memiliki missing value yang sangat besar seperti airline_sentiment_gold dan tweet_coord. Mari hapus data – data ini.

```
del df['tweet_coord']
del df['airline_sentiment_gold']
del df['negativereason_gold']
df.head()
```

✓ 0.1s Python

	tweet_id	airline_sentiment	airline_sentiment_confidence	negativereason	negativereason_confidence	airline	name	retweet_count	text	tw
0	570306133677760513	neutral	1.0000	NaN	NaN	Virgin America	cairdin	0	@VirginAmerica What @dhepburn said.	
1	570301130888122368	positive	0.3486	NaN	0.0000	Virgin America	jnardino	0	@VirginAmerica plus you've added commercials t...	
2	570301083672813571	neutral	0.6837	NaN	NaN	Virgin America	yvonnalynn	0	@VirginAmerica I didn't today... Must mean I n...	
3	570301031407624196	negative	1.0000	Bad Flight	0.7033	Virgin America	jnardino	0	@VirginAmerica it's really aggressive to blast...	

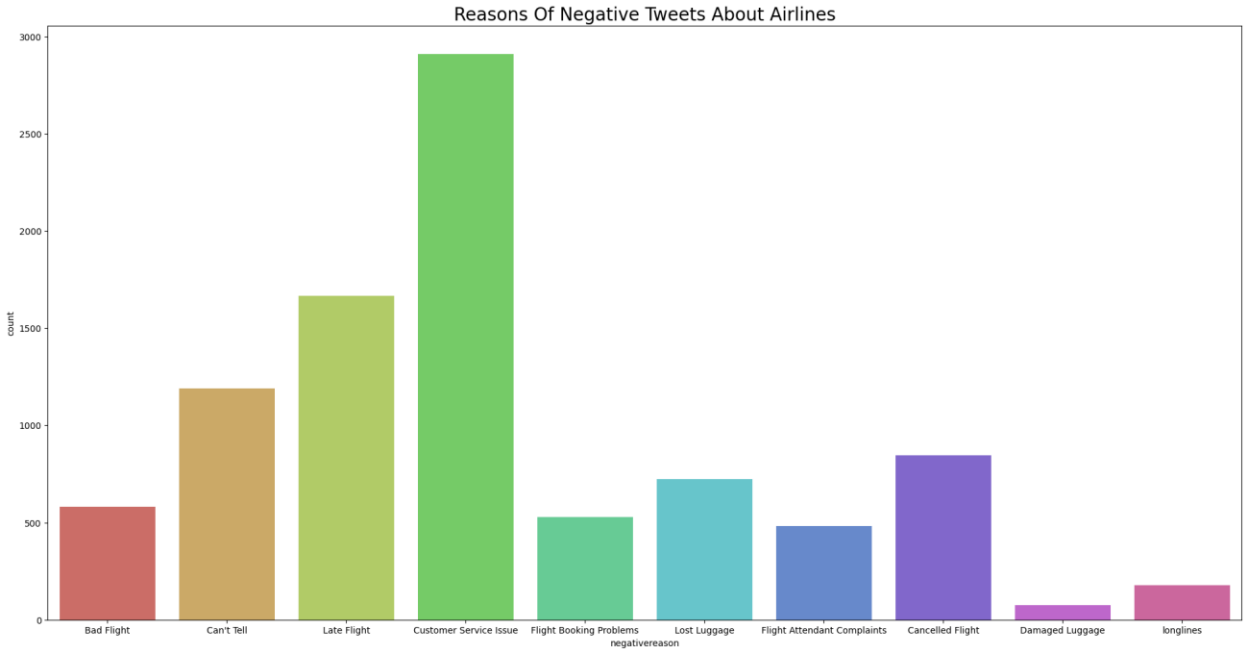
Sekarang, data – data yang memiliki missing value yang sangat banyak telah dihapus. Selanjutnya, mari tampilkan jumlah dari nilai negative reason seperti berikut.

Reasons Of Negative Tweets

```
print('Reasons Of Negative Tweets :','green')
print(df.negativereason.value_counts())

plt.figure(figsize = (24, 12))
sns.countplot(x = 'negativereason', data = df, palette = 'hls')
plt.title('Reasons Of Negative Tweets About Airlines', fontsize = 20)
plt.show()
```

[14] ✓ 0.4s

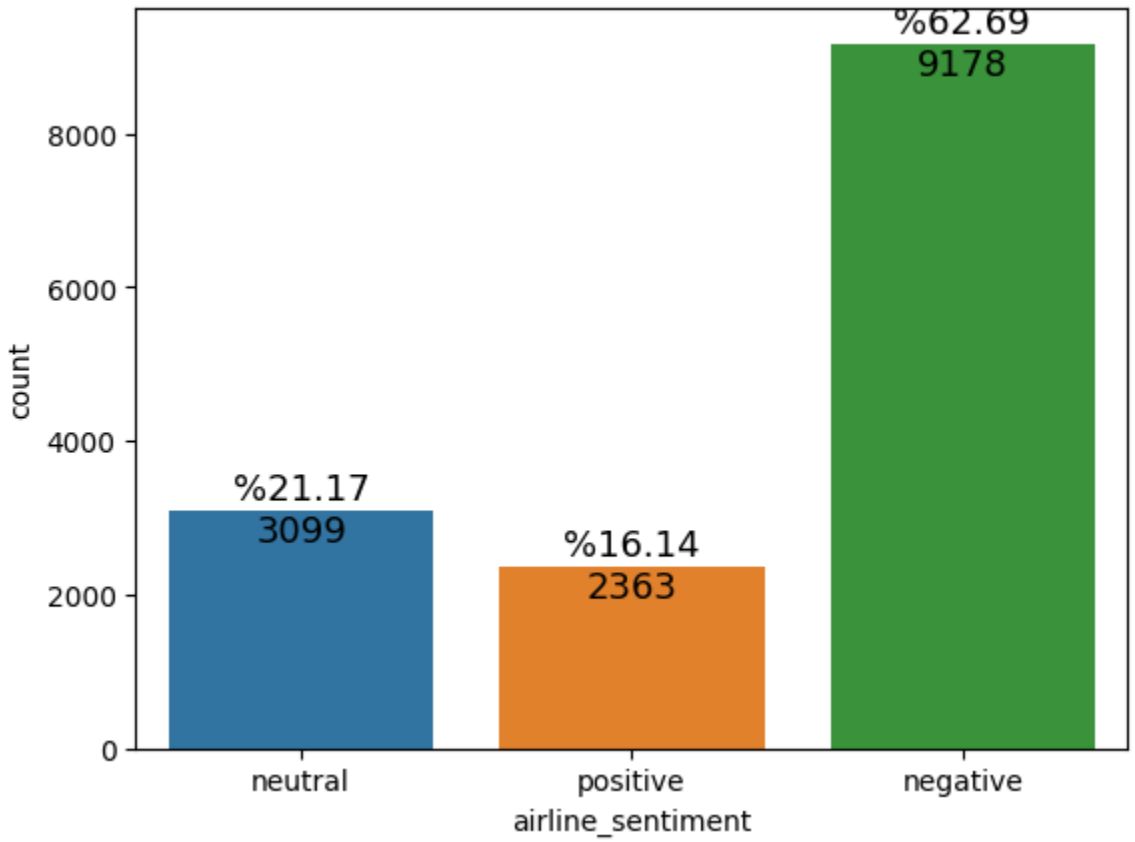
```
Reasons Of Negative Tweets : green
Customer Service Issue         2910
Late Flight                    1665
Can't Tell                     1190
Cancelled Flight                847
Lost Luggage                    724
Bad Flight                      580
Flight Booking Problems         529
Flight Attendant Complaints     481
longlines                       178
Damaged Luggage                  74
Name: negativereason, dtype: int64
```



Seperti yang kita lihat, data terbesar berada pada Customer Service Issue yaitu sebesar 2910 data dan data terkecil berada pada Damage Luggage yaitu sebesar 74 data. Selanjutnya, mari tampilkan label dari data.

```python
def labels(ax, df, xytext=(0, 0)):
    for bar in ax.patches:
        ax.annotate('%{:.2f}\n{:.0f}'.format(100*bar.get_height()/len(df),bar.get_height()), (bar.get_x() + bar.get_width() / 2,
                    bar.get_height()), ha='center', va='center',
                    size=13, xytext=xytext,
                    textcoords='offset points')

ax = sns.countplot(data =df, x = "airline_sentiment");
labels(ax, df)
# inbalanced data
```

✓ 0.2s



Seperti yang terlihat, data negative memiliki angkat yang sangat besar yaitu 62.69 % dari keseluruhan data sedangkan data positif hanya 16.14 % dan data neutral hanya 21.17 % sehingga dapat kita katakana bahwa data inblance.

Selanjutnya, mari tampilkan nilai unik dari data airline.

```python
df['airline'].unique()
```

✓ 0.4s

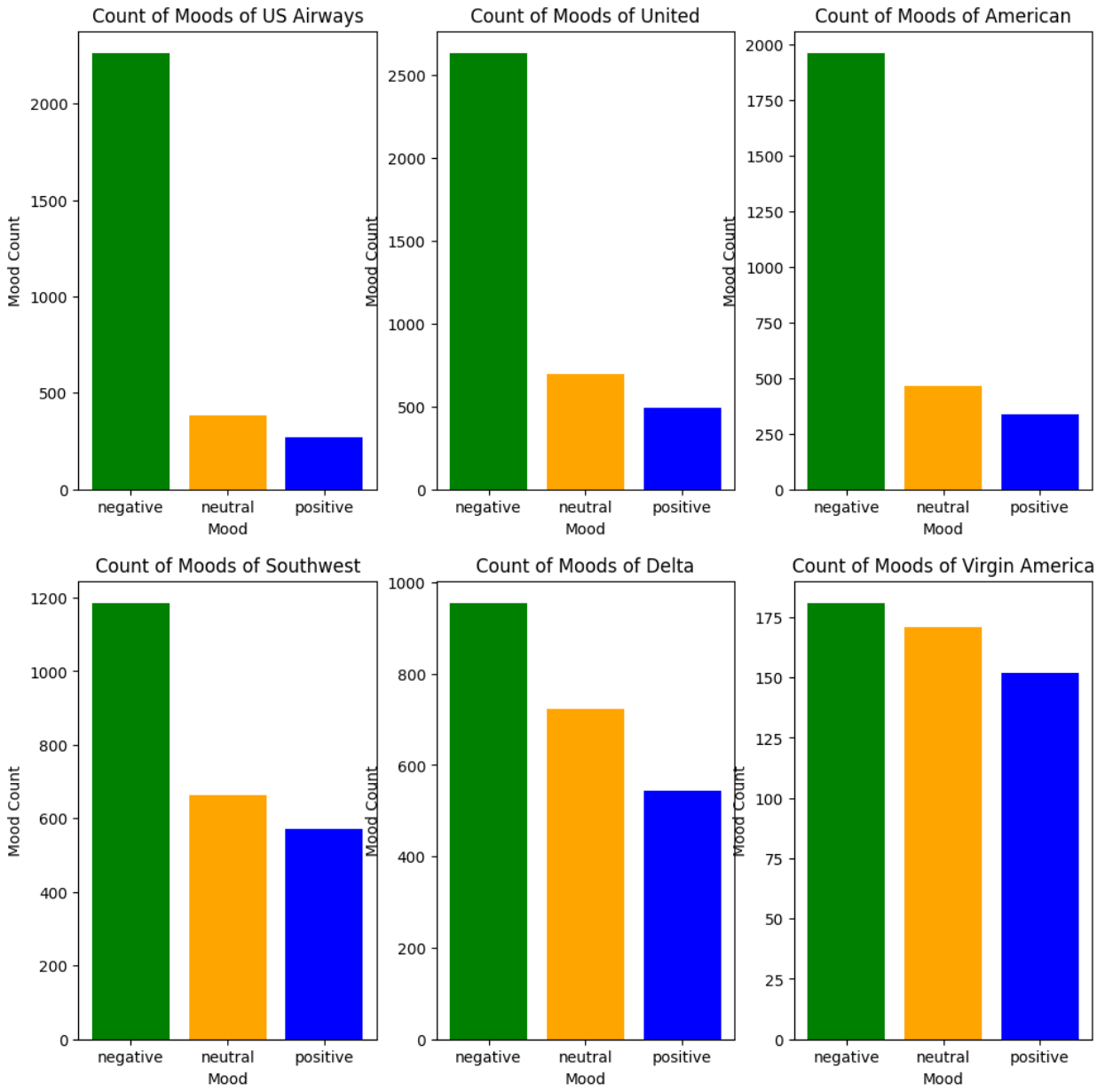
```
array(['Virgin America', 'United', 'Southwest', 'Delta', 'US Airways',
       'American'], dtype=object)
```

Data airline memiliki nilai unik yaitu seperti yang terlihat diatas. Lalu, tampilkan data tersebut berdasarkan mood seperti berikut.

```
print("Total number of tweets for each airline \n ",df.groupby('airline')['airline_sentiment'].count().sort_values(ascending=False))
airlines= ['US Airways','United','American','Southwest','Delta','Virgin America']
plt.figure(1,figsize=(12, 12))
for i in airlines:
    indices= airlines.index(i)
    plt.subplot(2,3,indices+1)
    new_df=df[df['airline']==i]
    count=new_df['airline_sentiment'].value_counts()
    Index = [1,2,3]
    plt.bar(Index,count, color=['green','orange','blue'])
    plt.xticks(Index,['negative','neutral','positive'])
    plt.ylabel('Mood Count')
    plt.xlabel('Mood')
    plt.title('Count of Moods of '+i)
```

✓ 1.1s

```
Total number of tweets for each airline
  airline
United            3822
US Airways        2913
American          2759
Southwest         2420
Delta             2222
Virgin America     504
Name: airline_sentiment, dtype: int64
```



Seperti yang terlihat, data negative adalah data terbesar di semua tempat yang terdaftar sedangkan data positif memiliki angka yang paling rendah. Data terbesar juga dimiliki oleh United sedangkan data terkecil dimiliki oleh Virgin America. Selanjutnya mari ubah index menjadi tweet_created.

```
date = df.reset_index()
#convert the Date column to pandas datetime
date.tweet_created = pd.to_datetime(date.tweet_created)
#Reduce the dates in the date column to only the date and no time stamp using the 'dt.date' method
date.tweet_created = date.tweet_created.dt.date
date.tweet_created.head()
df = date
day_df = df.groupby(['tweet_created','airline','airline_sentiment']).size()
# day_df = day_df.reset_index()
day_df
```

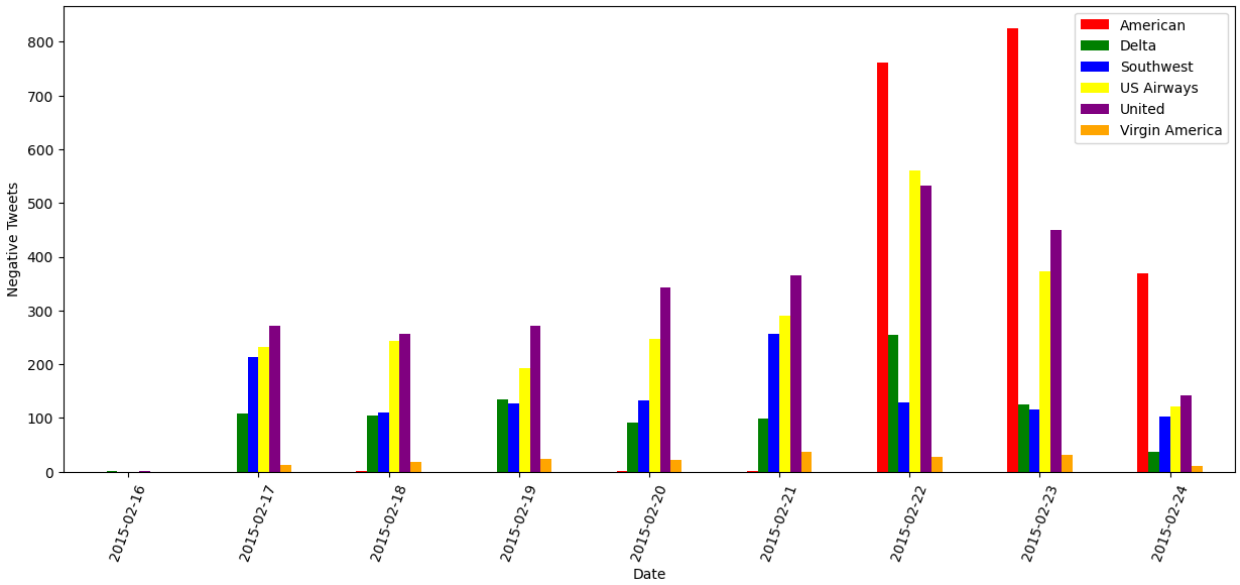
✓ 0.2s

```
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
```

Kita telah berhasil mengubah index menjadi tanggal. Selanjutnya, mari tampilkan data tersebut berdasarkan tanggal dan airline.

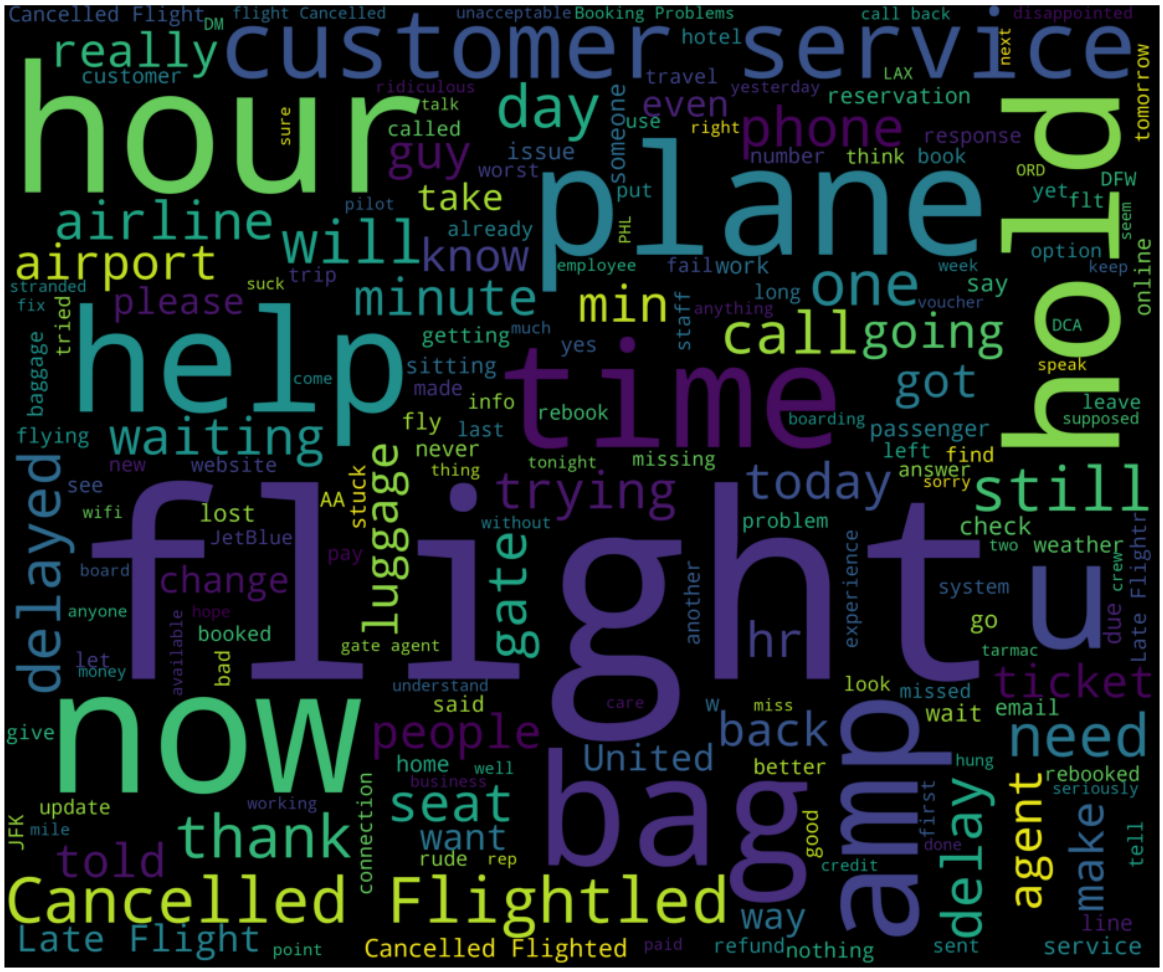
```
day_df = day_df.loc(axis=0)[:,:,'negative']

#groupby and plot data
ax2 = day_df.groupby(['tweet_created','airline']).sum().unstack().plot(kind = 'bar', color=['red', 'green', 'blue','yellow','purple','orange'], figsize = (1
labels = ['American','Delta','Southwest','US Airways','United','Virgin America']
ax2.legend(labels = labels)
ax2.set_xlabel('Date')
ax2.set_ylabel('Negative Tweets')
plt.show()
```

✓ 0.4s Python



Seperti yang kita lihat, data pada tanggal 23 Februari 2015 memiliki data paling besar disbanding seluruh data. Amerika menduduki data paling tinggi pada tanggal 22, 23, dan 24 Februari 2015. Namun, pada tanggal lain, amerika menduduki angka paling rendah. United memiliki kencederungan sama dari tanggal ke tanggal tetapi mengalami penurunan pada tanggal 23 dan 24 Februari 2015. Data virgin adalah data yang paling rendah dari hampir seluruh data di beberapa tanggal.

Selanjutnya, mari tampilkan wordcloud pada data negative seperti berikut.



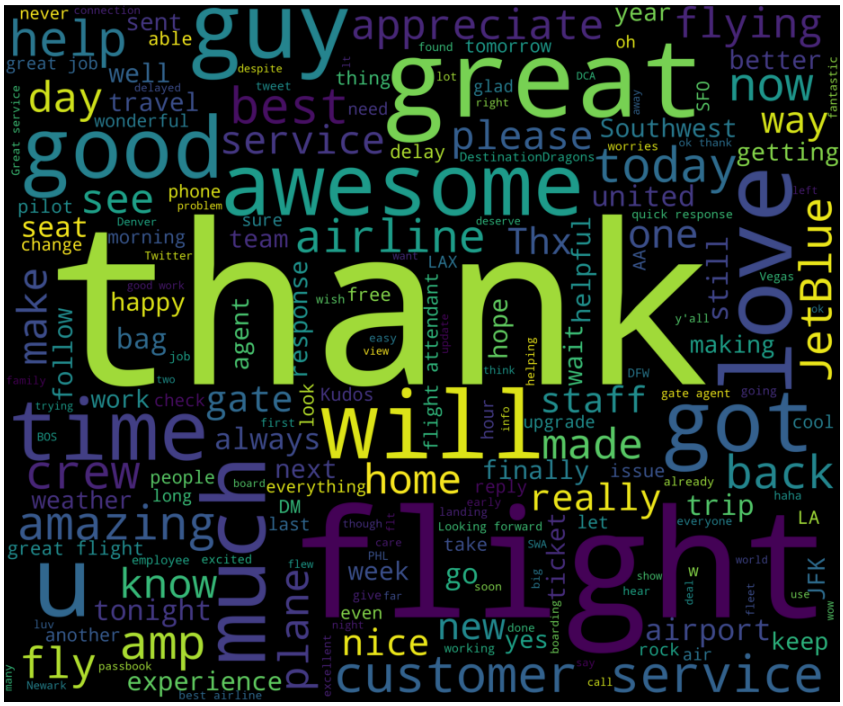
Seperti yang kita lihat, semakin banyak kata yang sama pada data, semakin besar pula kata tersebut. Disini, terlihat kata flight adalah kata yang sangat besar sehingga dapat kita katakan bahwa kata flight mewakili hampir seluruh data. Namun, beberapa kata terlihat sangat kecil yang artinya hanya ada sedikit data mengenai hal tersebut pada negative wordcloud.

Selanjutnya, mari tampilkan wordcloud pada positive sentiments

WordCloud of Positive sentiments

```
new_df=df[df['airline_sentiment']=='positive']
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='black',
                      width=3000,
                      height=2500
                     ).generate(cleaned_word)
plt.figure(1,figsize=(12, 12))
plt.imshow(wordcloud)
plt.axis('off')
plt.show()
```

21] ✓ 18.9s



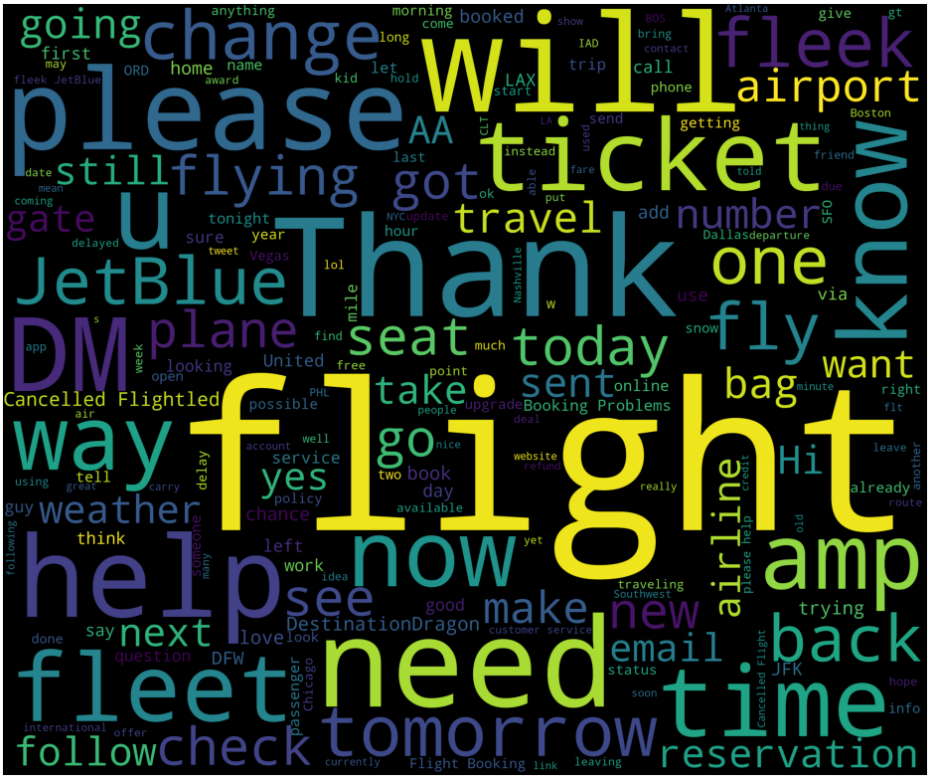
Seperti yang terlihat, kata thank adalah kata yang paling besar sehingga dapat kita katakan bahwa kata thank mewakili hampir seluruh data positive. Namun, kata flight juga termasuk kata yang besar. Sama halnya seperti pada data negative, kata flight pada positive juga sangat banyak.

Selanjutnya, mari tampilkan wordcloud pada neutral sentiments.

WordCloud of Neutral sentiments

```
new_df=df[df['airline_sentiment']=='neutral']
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='black',
                      width=3000,
                      height=2500
                     ).generate(cleaned_word)
plt.figure(1,figsize=(12, 12))
plt.imshow(wordcloud)
plt.axis('off')
plt.show()
```

✓ 20.3s



Seperti yang kita lihat, kata flight disini juga mewakili hampir seluruh data neutral sehingga dapat kita simpulkan bahwa kata flight mewakili hampir seluruh data pada data negative, positive, dan neutral.

4. Data Preparation

Pada tahapan ini, kita akan menyiapkan datanya terlebih dahulu dan melakukan beberapa tahapan terhadap data seperti stopwords, filtering, stemming, dan vektorisasi sebelum melatihnya pada machine learning. Pertama, kita ubah data negative, positive dan neutral menjadi bilangan diskrit seperti berikut.

```
# convert Sentiments to 0,1,2
def convert_Sentiment(sentiment):
    if  sentiment == "positive":
        return 2
    elif sentiment == "neutral":
        return 1
    elif sentiment == "negative":
        return 0
```

✓ 0.5s

```
# Apply convert_Sentiment function
df.airline_sentiment = df.airline_sentiment.apply(lambda x : convert_Sentiment(x))
```

✓ 0.6s

Lalu, lakukan stopwords pada data seperti berikut.

```
# Remove stop words
def remove_stopwords(text):
    text = ' '.join([word for word in text.split() if word not in (stopwords.words('english'))])
    return text

# Remove url
def remove_url(text):
    url = re.compile(r'https?://\S+|www\.\S+')
    return url.sub(r'',text)

# Remove punct
def remove_punct(text):
    table = str.maketrans('', '', string.punctuation)
    return text.translate(table)

# Remove html
def remove_html(text):
    html=re.compile(r'<.*?>')
    return html.sub(r'',text)

# Remove @username
def remove_username(text):
    return re.sub('@[^\s]+','',text)

# Remove emojis
def remove_emoji(text):
    emoji_pattern = re.compile("["
                               u"\U0001F600-\U0001F64F"  # emoticons
                               u"\U0001F300-\U0001F5FF"  # symbols & pictographs
                               u"\U0001F680-\U0001F6FF"  # transport & map symbols
                               u"\U0001F1E0-\U0001F1FF"  # flags (iOS)
                               u"\U00002702-\U000027B0"
                               u"\U000024C2-\U0001F251"
                               "]+", flags=re.UNICODE)
    return emoji_pattern.sub(r'', text)
```

```python
# Decontraction text
def decontraction(text):
    text = re.sub(r"won\'t", " will not", text)
    text = re.sub(r"won\'t've", " will not have", text)
    text = re.sub(r"can\'t", " can not", text)
    text = re.sub(r"don\'t", " do not", text)

    text = re.sub(r"can\'t've", " can not have", text)
    text = re.sub(r"ma\'am", " madam", text)
    text = re.sub(r"let\'s", " let us", text)
    text = re.sub(r"ain\'t", " am not", text)
    text = re.sub(r"shan\'t", " shall not", text)
    text = re.sub(r"sha\n't", " shall not", text)
    text = re.sub(r"o\'clock", " of the clock", text)
    text = re.sub(r"y\'all", " you all", text)
    text = re.sub(r"n\'t", " not", text)
    text = re.sub(r"n\'t've", " not have", text)
    text = re.sub(r"\'re", " are", text)
    text = re.sub(r"\'s", " is", text)
    text = re.sub(r"\'d", " would", text)
    text = re.sub(r"\'d've", " would have", text)
    text = re.sub(r"\'ll", " will", text)
    text = re.sub(r"\'ll've", " will have", text)
    text = re.sub(r"\'t", " not", text)
    text = re.sub(r"\'ve", " have", text)
    text = re.sub(r"\'m", " am", text)
    text = re.sub(r"\'re", " are", text)
    return text

# Seperate alphanumeric
def seperate_alphanumeric(text):
    words = text
    words = re.findall(r"[^\W\d_]+|\d+", words)
    return " ".join(words)
```

```python
# Seperate alphanumeric
def seperate_alphanumeric(text):
    words = text
    words = re.findall(r"[^\W\d_]+|\d+", words)
    return " ".join(words)

def cont_rep_char(text):
    tchr = text.group(0)

    if len(tchr) > 1:
        return tchr[0:2]

def unique_char(rep, text):
    substitute = re.sub(r'(\w)\1+', rep, text)
    return substitute

def char(text):
    substitute = re.sub(r'[^a-zA-Z]',' ',text)
    return substitute

# combaine negative reason with  tweet (if exsist)
df['final_text'] = df['negativereason'].fillna('') + ' ' + df['text']


# Apply functions on tweets
df['final_text'] = df['final_text'].apply(lambda x : remove_username(x))
df['final_text'] = df['final_text'].apply(lambda x : remove_url(x))
df['final_text'] = df['final_text'].apply(lambda x : remove_emoji(x))
df['final_text'] = df['final_text'].apply(lambda x : decontraction(x))
df['final_text'] = df['final_text'].apply(lambda x : seperate_alphanumeric(x))
df['final_text'] = df['final_text'].apply(lambda x : unique_char(cont_rep_char,x))
df['final_text'] = df['final_text'].apply(lambda x : char(x))
df['final_text'] = df['final_text'].apply(lambda x : x.lower())
df['final_text'] = df['final_text'].apply(lambda x : remove_stopwords(x))
```

✓ 1m 2.3s Python

Kita berhasil melakukan stopwords pada data. Selanjutnya, melakukan stemming seperti berikut.

```
sbs = SnowballStemmer(language='english')
```
✓ 0.6s

```
def stemmer(text):
    text = [sbs.stem(word) for word in text.split(' ')]
    text = " ".join(text)
    return text

df.final_text = df.final_text.apply(stemmer)
```
✓ 2.2s

Dan mengecek hasilnya seperti berikut.

```
# result
df['final_text']
```
✓ 0.4s

```
0                                                said
1                            plus ad commerci experi tacki
2                      today must mean need take anoth trip
3          bad flight realli aggress blast obnoxi enterta...
4                             ca tell realli big bad thing
                              ...
14635                      thank got differ flight chicago
14636     custom servic issu leav minut late flight warn...
14637                pleas bring american airlin blackberri
14638     custom servic issu money chang flight answer p...
14639     ppl need know mani seat next flight plz put us...
Name: final_text, Length: 14640, dtype: object
```

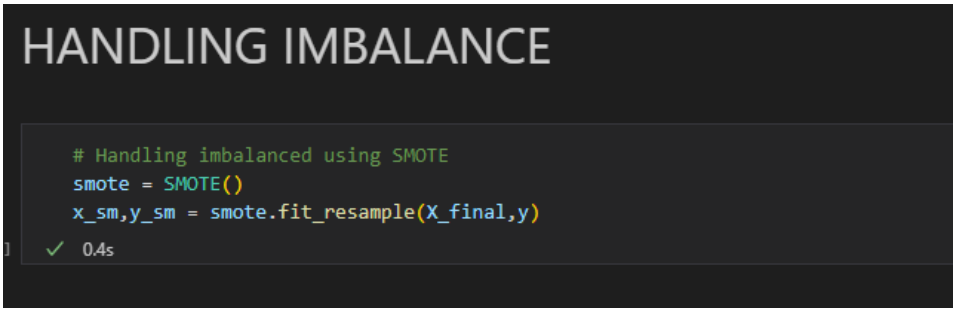
Data berhasil kita stemming. Selanjutnya, kita lakukan vektorisasi pada data dengan TfidfVectorizer().

```
X = df['final_text']
y = df['airline_sentiment']
```
✓ 0.6s

Apply TFIDF on cleaned tweets

```
tfid = TfidfVectorizer()
X_final =  tfid.fit_transform(X)
```
✓ 0.2s

Data berhasil divektorisasi. Selanjutnya, lakukan smote pada data guna meng-handle data imbalance atau data tidak imbang seperti berikut.

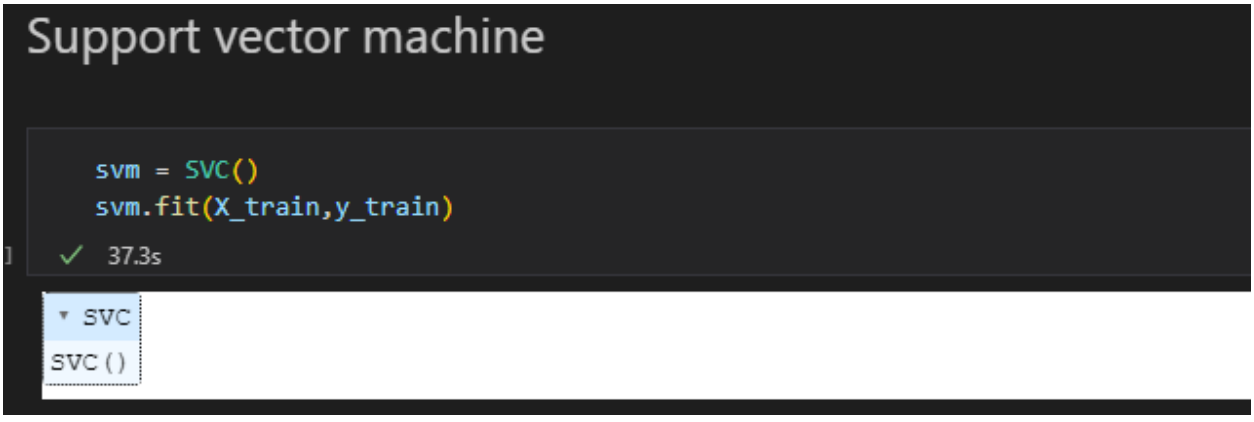


Lalu, bagi dataset menjadi 2 yaitu data training dan data testing seperti berikut.

Data telah dibagi menjadi data training dan data testing dengan rasio 75:25 dan random_state 3. Selanjutnya, kita lakukan tahap pembuatan model.

5. Building Model & Evaluation Model

Pada tahapan ini, kita lakukan pembuatan model dan mengevaluasinya. Kita memakai 2 model yaitu Support Vector Machine (SVM) dan Naïve Bayes. Pertama, kita buat model Support Vector Machine (SVM) seperti berikut.



Lalu, cek akurasi pada model ini seperti berikut.

```
svm_prediction =  svm.predict(X_test)
[34] ✓ 7.1s

accuracy_score(svm_prediction,y_test)
[35] ✓ 0.4s
··· 0.9355026147588611
```

Model menghasilkan akurasi 93 % pada data testing. Selanjutnya lakukan pembuatan model Naïve Bayes seperti berikut.

Naive Bayes

```
nb = MultinomialNB()
nb.fit(X_train,y_train)
6] ✓ 0.7s

▾ MultinomialNB
MultinomialNB()
```

Lalu, cek akurasi pada model ini.

```
nb_prediction =  nb.predict(X_test)
✓ 0.6s

accuracy_score(nb_prediction,y_test)
✓ 0.4s
0.852992446252179
```

Model ini memiliki akurasi yang lebih rendah dibanding model Support Vector Machine (SVM) sehingga kita dapat menggunakan model Support Vector Machine (SVM). Sebelum itu, mari evaluasi model Support Vector Machine (SVM) seperti berikut.

VISUALIZE BEST MODEL PERFORMENCE

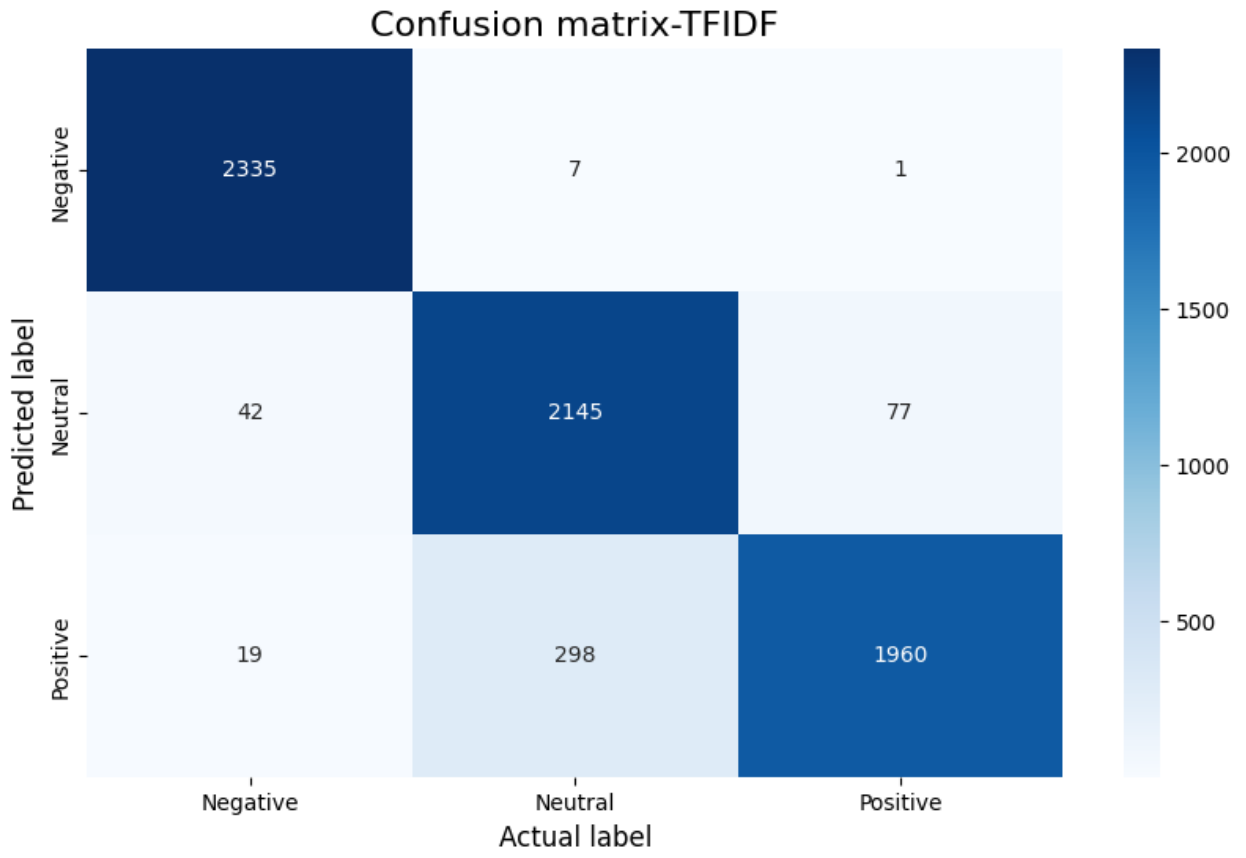
```
cr = classification_report(y_test, svm_prediction)
```

✓ 0.7s

```
print("Classification Report:\n----------------------\n", cr)

cm = confusion_matrix(y_test, svm_prediction)


# plot confusion matrix
plt.figure(figsize=(10,6))
sentiment_classes = ['Negative', 'Neutral', 'Positive']
sns.heatmap(cm, cmap=plt.cm.Blues, annot=True, fmt='d',
            xticklabels=sentiment_classes,
            yticklabels=sentiment_classes)
plt.title('Confusion matrix-TFIDF', fontsize=16)
plt.xlabel('Actual label', fontsize=12)
plt.ylabel('Predicted label', fontsize=12)
plt.show()
```

✓ 0.5s

```
Classification Report:
----------------------
              precision    recall  f1-score   support

           0       0.97      1.00      0.99      2343
           1       0.88      0.95      0.91      2264
           2       0.96      0.86      0.91      2277

    accuracy                           0.94      6884
   macro avg       0.94      0.93      0.93      6884
weighted avg       0.94      0.94      0.94      6884
```



Seperti yang kita lihat, model memiliki akurasi sekitar 94% dengan True Positive jauh lebih besar dibanding True Negative dan begitupun juga True Negative sehingga kita dapat memakai model ini untuk kebutuhan selanjutnya.

Terimakasih