Import Library

```
pip install emoji
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting emoji
  Downloading emoji-1.7.0.tar.gz (175 kB)
     |████████████████████████████████| 175 kB 26.5 MB/s
Building wheels for collected packages: emoji
  Building wheel for emoji (setup.py) ... done
  Created wheel for emoji: filename=emoji-1.7.0-py3-none-any.whl size=171046 sha256=95d426dbcc4d81ac6f7e4cd4b021c9d5aa2
  Stored in directory: /root/.cache/pip/wheels/8a/4e/b6/57b01db010d17ef6ea9b40300af725ef3e210cb1acfb7ac8b6
Successfully built emoji
Installing collected packages: emoji
Successfully installed emoji-1.7.0
```

```
pip install contractions
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting contractions
  Downloading contractions-0.1.72-py2.py3-none-any.whl (8.3 kB)
Collecting textsearch>=0.0.21
  Downloading textsearch-0.0.21-py2.py3-none-any.whl (7.5 kB)
Collecting pyahocorasick
  Downloading pyahocorasick-1.4.4-cp37-cp37m-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (106 kB)
     |████████████████████████████████| 106 kB 32.5 MB/s
Collecting anyascii
  Downloading anyascii-0.3.1-py3-none-any.whl (287 kB)
     |████████████████████████████████| 287 kB 62.7 MB/s
Installing collected packages: pyahocorasick, anyascii, textsearch, contractions
Successfully installed anyascii-0.3.1 contractions-0.1.72 pyahocorasick-1.4.4 textsearch-0.0.21
```

```
import pandas as pd
import numpy as np
import emoji
```

```
import contractions
import re
from bs4 import BeautifulSoup
import matplotlib.pyplot as plt
%matplotlib inline
import nltk

import nltk
nltk.download("punkt")
from nltk.tokenize import word_tokenize
nltk.download('stopwords')
from nltk.corpus import stopwords
nltk.download('wordnet')

from nltk.stem import LancasterStemmer, WordNetLemmatizer
from nltk import pos_tag
from nltk.corpus import wordnet
nltk.download('sentiwordnet')
from sklearn import svm
from sklearn.svm import SVC
from nltk.corpus import sentiwordnet as swn
import pickle

from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import KFold, StratifiedKFold, cross_val_score
from sklearn.metrics import classification_report,confusion_matrix, accuracy_score
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data] Downloading package sentiwordnet to /root/nltk_data...
[nltk_data]   Unzipping corpora/sentiwordnet.zip.
```

```
df = pd.read_csv('ebay_reviews.csv')
df
```

	category	review title	review content	rating
0	Headsets	Wireless gaming headset	This gaming headset ticks all the boxes # look...	5
1	Headsets	Good for those with a big head, low budget	Easy setup, rated for 6 hours battery but mine...	3
2	Headsets	MezumiWireless Gaming Headset	I originally bought this wireless headset for ...	5
3	Headsets	HW- S2 great headset.	This is my 2nd Mezumi headset, It kills the fi...	5
4	Headsets	BEST HEADPHONES I'VE PURCHASED IN MY ENTIRE LIFE	This is probably the best headset I've purchas...	5
...	...	...	...	...
44751	Racks & Holders	Utensil holder	Reasonably priced but a little flimsy	3
	Racks &			

Preprocessing

1. Cleaning Data

```
def html_remover(data):
    beauti = BeautifulSoup(data,'html.parser')
    return beauti.get_text()
```

```
def convert_emoji(data):
    return emoji.demojize(data)

def url_remover(data):
    return re.sub('(http|https):\/\/\S+', '',data)

def remove_round_brackets(data):
    return re.sub('\(.*?\)','',data)

def remove_punc(data):
    document = re.sub(r'[^\w\s]','', data)
    return document

def white_space(data):
    return ' '.join(data.split())

def text_lower(data):
    return data.lower()

def contraction_replace(data):
    return contractions.fix(data)

def remove_number(data):
    return re.sub(r"\d+", "", data)

def remove_singl_char(data):
    return re.sub(r"\b[a-zA-Z]\b", "", data)

def web_associated(data):
    new_data = html_remover(data)
    new_data = convert_emoji(new_data)
    new_data = url_remover(new_data)
    new_data = remove_round_brackets(new_data)
    new_data = remove_punc(new_data)
    new_data = white_space(new_data)
    new_data = text_lower(new_data)
    new_data = contraction_replace(new_data)
    new_data = remove_number(new_data)
```

```
    new_data = remove_singl_char(new_data)
    return new_data

df['cleaning data'] = df['review content'].apply(web_associated)
```

```
/usr/local/lib/python3.7/dist-packages/bs4/__init__.py:273: UserWarning: "b'.'" looks like a filename, not markup. You
  ' Beautiful Soup.' % markup)
/usr/local/lib/python3.7/dist-packages/bs4/__init__.py:273: UserWarning: "b'..'" looks like a filename, not markup. You
  ' Beautiful Soup.' % markup)
```

```
df.head()
```

	category	review title	review content	rating	cleaning data
0	Headsets	Wireless gaming headset	This gaming headset ticks all the boxes # look...	5	this gaming headset ticks all the boxes looks ...
1	Headsets	Good for those with a big head, low budget	Easy setup, rated for 6 hours battery but mine...	3	easy setup rated for hours battery but mine h...
2	Headsets	MezumiWireless Gaming Headset	I originally bought this wireless headset for ...	5	originally bought this wireless headset for m...
			This is my 2nd Mezumi headset. It		this is my nd mezumi headset it

2. Tokenizing

```
def tokenize(data):
    return nltk.word_tokenize(data)
```

```
df['tokenizing'] = df['cleaning data'].apply(tokenize)
```

3. Negation Handling

```
def Negation(sentence):
    temp = int(0)
    for i in range(len(sentence)):
        if sentence[i-1] in ['not',"n't"]:
            antonyms = []
            for syn in wordnet.synsets(sentence[i]):
                syns = wordnet.synsets(sentence[i])
                w1 = syns[0].name()
                temp = 0
                for l in syn.lemmas():
                    if l.antonyms():
                        antonyms.append(l.antonyms()[0].name())
                max_dissimilarity = 0
                for ant in antonyms:
                    syns = wordnet.synsets(ant)
                    w2 = syns[0].name()
                    syns = wordnet.synsets(sentence[i])
                    w1 = syns[0].name()
                    word1 = wordnet.synset(w1)
                    word2 = wordnet.synset(w2)
                    if isinstance(word1.wup_similarity(word2), float) or isinstance(word1.wup_similarity(word2), int):
                        temp = 1 - word1.wup_similarity(word2)
                    if temp>max_dissimilarity:
                        max_dissimilarity = temp
                        antonym_max = ant
                        sentence[i] = antonym_max
                        sentence[i-1] = ''
    while '' in sentence:
        sentence.remove('')
    return sentence
```

```
import nltk
nltk.download('omw-1.4')
```

```
[nltk_data] Downloading package omw-1.4 to /root/nltk_data...
True
```

```
df['negation'] = df['tokenizing'].apply(Negation)
```

Stopword

```
def stopword(data):
    nltk.download('stopwords')
    clean = []
    for i in data:
        if i not in stopwords.words('english'):
            clean.append(i)
    return clean
```

```
df['stopword'] = df['negation'].apply(stopword)
```

```
Streaming output truncated to the last 5000 lines.
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package stopwords to /root/nltk_data...
```

4. Lemmatization

[ ] ↳ *3 cells hidden*

5. Pos Tagging

[ ] ↳ *6 cells hidden*

Pelabelan

TextBlob

[ ] ↳ *5 cells hidden*

Vader Sentiment

[ ] ↳ *8 cells hidden*

Split Data

[ ] ↳ *2 cells hidden*

EKSTRAKSI FITUR: Term presence

```
count_vect = CountVectorizer(binary=True)
X_train_counts = count_vect.fit_transform(data_label)
print(X_train_counts.shape)
count_vect.vocabulary_
```

```
(44756, 32506)
{'this': 28335,
 'gaming': 12677,
 'headset': 13937,
 'ticks': 28445,
 'all': 2633,
 'the': 28164,
```

```
'boxes': 4907,
'looks': 17150,
'grate': 13269,
'built': 5221,
'to': 28587,
'last': 16517,
'excellent': 10872,
'sound': 26303,
'mic': 18103,
'comfortable': 6712,
'wear': 30685,
'easy': 9927,
'set': 25191,
'up': 29734,
'what': 30830,
'more': 18586,
'could': 7552,
'you': 31594,
'ask': 3424,
'for': 12051,
'setup': 25213,
'rated': 22793,
'hours': 14392,
'battery': 4097,
'but': 5327,
'mine': 18233,
'has': 13798,
'lasted': 16519,
'sessions': 25190,
'over': 20155,
'good': 13065,
'loudness': 17213,
'from': 12338,
'earcups': 9875,
'seal': 24921,
'and': 2896,
'thick': 28296,
'padding': 20344,
'around': 3323,
'ears': 9903,
'person': 20864,
'with': 31098,
```

```
 'med': 17872,
 'big': 4457,
 'head': 13903,
 'would': 31276,
 'look': 17139,
 'funny': 12469,
 'on': 19788,
 'small': 25935,
```

```
X_train_TP = count_vect.transform(X_train)
X_test_TP = count_vect.transform(X_test)
```

```
print(X_train_TP)
```

```
  (0, 2896)     1
  (0, 4873)     1
  (0, 5353)     1
  (0, 8924)     1
  (0, 12338)    1
  (0, 12794)    1
  (0, 12815)    1
  (0, 13934)    1
  (0, 15551)    1
  (0, 18517)    1
  (0, 18871)    1
  (0, 19365)    1
  (0, 19432)    1
  (0, 19465)    1
  (0, 19788)    1
  (0, 19827)    1
  (0, 20941)    1
  (0, 28245)    1
  (0, 30594)    1
  (0, 30998)    1
  (1, 2416)     1
  (1, 2896)     1
  (1, 4097)     1
  (1, 4152)     1
  (1, 4298)     1
  :     :
  (26852, 13841)        1
```

```
(26852, 14872)        1
(26852, 15323)        1
(26852, 15613)        1
(26852, 16876)        1
(26852, 16990)        1
(26852, 17224)        1
(26852, 18954)        1
(26852, 19865)        1
(26852, 20958)        1
(26852, 21929)        1
(26852, 24508)        1
(26852, 24683)        1
(26852, 24974)        1
(26852, 25223)        1
(26852, 26112)        1
(26852, 28164)        1
(26852, 28489)        1
(26852, 28587)        1
(26852, 30154)        1
(26852, 30850)        1
(26852, 31098)        1
(26852, 31224)        1
(26852, 31259)        1
(26852, 31594)        1
```

KLASIFIKASI dengan term presence

```
SVM_Clasifier = svm.SVC(C=1.0, kernel='linear', degree=3, gamma='auto')

# fitting data latih pada classifier
SVM_Clasifier.fit(X_train_TP,y_train)
# memprediksi label pada set data uji
predictions_SVM_TP = SVM_Clasifier.predict(X_test_TP)

# Menggunakan fungsi accuracy_score untuk mendapat nilai akurasi
print('Confusion Matrix: \n',confusion_matrix(y_test, predictions_SVM_TP))
print()
```

```
print('Accuracy: ', accuracy_score(y_test, predictions_SVM_TP))
```

```
Confusion Matrix:
 [[  463    91   268]
 [   47  1869    85]
 [  274   102 14704]]

Accuracy:  0.9515723621739374
```

EKSTRAKSI FITUR: TF-IDF

```
vectorizer = TfidfVectorizer()
X = vectorizer.fit_transform(data_label)
features_train_transformed = vectorizer.transform(X_train)
features_test_transformed = vectorizer.transform(X_test)
print(features_train_transformed)
```

```
  (0, 30998)    0.13864144681984028
  (0, 30594)    0.3052228986401106
  (0, 28245)    0.15041903846597365
  (0, 20941)    0.2856753269703874
  (0, 19827)    0.1351725362091631
  (0, 19788)    0.10360645891226611
  (0, 19465)    0.15760160429789655
  (0, 19432)    0.25909898703161555
  (0, 19365)    0.26036805345351166
  (0, 18871)    0.0917117881043199
  (0, 18517)    0.1715262165984092
  (0, 15551)    0.07711214590024702
  (0, 13934)    0.5349042293370069
  (0, 12815)    0.2584802312211341
  (0, 12794)    0.29418740972610896
  (0, 12338)    0.129843810759904
  (0, 8924)     0.21978448345196488
  (0, 5353)     0.14738063971833018
  (0, 4873)     0.14246495241436954
  (0, 2896)     0.06459055873377202
  (1, 31098)    0.08339968118563554
```

```
  (1, 30850)    0.1271662397235786
  (1, 30575)    0.28488956287462275
  (1, 29819)    0.2376250757959866
  (1, 29285)    0.23623328251824366
  :     :
  (26852, 20958)        0.0868411777547107
  (26852, 19865)        0.1649929452605165
  (26852, 18954)        0.2603462786431715
  (26852, 17224)        0.09185808100565548
  (26852, 16990)        0.11370211696098893
  (26852, 16876)        0.08837575737789898
  (26852, 15613)        0.05159310599229723
  (26852, 15323)        0.3080124455866271
  (26852, 14872)        0.07075384099492378
  (26852, 13841)        0.15627370255685713
  (26852, 13798)        0.1901372538162533
  (26852, 13269)        0.2516740902663807
  (26852, 12661)        0.12741006011224043
  (26852, 12275)        0.23853239997345266
  (26852, 11702)        0.19596409458684458
  (26852, 11382)        0.1157897150705048
  (26852, 10773)        0.13582861583144784
  (26852, 10746)        0.11450474507400692
  (26852, 9440) 0.11581863009611014
  (26852, 9403) 0.13380303003102784
  (26852, 8504) 0.22495719979885334
  (26852, 5327) 0.07297891540865092
  (26852, 3521) 0.09645936741797435
  (26852, 3439) 0.21338209402387953
  (26852, 2896) 0.0964904377493778
```

KLASIFIKASI dengan TF-IDF

```
SVM_Clasifier = svm.SVC(C=1.0, kernel='linear', degree=3, gamma='auto')
SVM_Clasifier.fit(features_train_transformed,y_train)
```

```
SVC(gamma='auto', kernel='linear')
```

```
#Evaluate the model on the training data set

predictions_SVM_Tfidf = SVM_Clasifier.predict(features_test_transformed)
print(classification_report(y_test, predictions_SVM_Tfidf))
print('Confusion Matrix: \n',confusion_matrix(y_test, predictions_SVM_Tfidf))
print()
print('Accuracy: ', accuracy_score(y_test, predictions_SVM_Tfidf))
```

```
              precision    recall  f1-score   support

    Negative       0.86      0.30      0.45       822
     Neutral       0.89      0.87      0.88      2001
    Positive       0.95      0.99      0.97     15080

    accuracy                           0.95     17903
   macro avg       0.90      0.72      0.77     17903
weighted avg       0.94      0.95      0.94     17903

Confusion Matrix:
 [[  248    88   486]
 [   14  1737   250]
 [   25   117 14938]]

Accuracy:  0.9452605708540468
```

Evaluation with K-Fold and Classification Report:

```
from sklearn.model_selection import KFold

# vectorizer = TfidfVectorizer()
# X = vectorizer.fit_transform(data_label)
kf = KFold(n_splits=10, shuffle=True, random_state=42)
scores = []
for fold, (train_index, test_index) in enumerate(kf.split(data_label,label), 1):
    X_train, X_test = data_label[train_index], data_label[test_index]
    y_train, y_test = label[train_index], label[test_index]
```

```
    vectorizer = TfidfVectorizer()
    vectorizer.fit_transform(data_label)
    features_train_transformed = vectorizer.transform(X_train)
    features_test_transformed = vectorizer.transform(X_test)

    SVM_Clasifier = svm.SVC(C=1.0, kernel='linear', degree=3, gamma='auto')
    SVM_Clasifier.fit(features_train_transformed,y_train)
    predictions_SVM_Tfidf = SVM_Clasifier.predict(features_test_transformed)


#     accuracy = accuracy_score(y_true_test, y_pred)
#     precision = precision_score(y_true_test, y_pred)
#     recall = recall_score(y_true_test, y_pred)
#     f1 = f1_score(y_true_test, y_pred)

    #print to file
#     acc_oversampled(LLmark, featExt, accuracy)
#     cr_oversampled(LLmark, featExt, precision, recall, f1)

    print(f'# Fold {fold}, Train set: {len(train_index)}, Test set:{len(test_index)}')
    print(classification_report(y_test, predictions_SVM_Tfidf), "\n")
```

```
 # Fold 1, Train set: 40280, Test set:4476
              precision    recall  f1-score   support

    Negative       0.89      0.36      0.51       184
     Neutral       0.92      0.90      0.91       509
    Positive       0.96      0.99      0.98      3783

    accuracy                           0.96      4476
   macro avg       0.92      0.75      0.80      4476
weighted avg       0.95      0.96      0.95      4476


 # Fold 2, Train set: 40280, Test set:4476
              precision    recall  f1-score   support

    Negative       0.84      0.35      0.50       217
     Neutral       0.92      0.88      0.90       504
```

```
    Positive       0.96      0.99      0.97      3755

    accuracy                           0.95      4476
   macro avg       0.90      0.74      0.79      4476
weighted avg       0.95      0.95      0.94      4476


# Fold 3, Train set: 40280, Test set:4476
              precision    recall  f1-score   support

    Negative       0.84      0.35      0.49       217
     Neutral       0.90      0.89      0.89       494
    Positive       0.96      0.99      0.97      3765

    accuracy                           0.95      4476
   macro avg       0.90      0.74      0.78      4476
weighted avg       0.94      0.95      0.94      4476


# Fold 4, Train set: 40280, Test set:4476
              precision    recall  f1-score   support

    Negative       0.90      0.40      0.56       204
     Neutral       0.92      0.90      0.91       494
    Positive       0.96      0.99      0.98      3778

    accuracy                           0.96      4476
   macro avg       0.93      0.76      0.81      4476
weighted avg       0.95      0.96      0.95      4476


# Fold 5, Train set: 40280, Test set:4476
              precision    recall  f1-score   support

    Negative       0.89      0.45      0.60       214
     Neutral       0.93      0.91      0.92       498
    Positive       0.96      0.99      0.98      3764

    accuracy                           0.96      4476
   macro avg       0.93      0.78      0.83      4476
weighted avg       0.96      0.96      0.95      4476
```

```
print('\n\nCross-Validation accuracy: %.3f +/- %.3f' %(np.mean(scores), np.std(scores)))
```

```
Cross-Validation accuracy: nan +/- nan
/usr/local/lib/python3.7/dist-packages/numpy/core/fromnumeric.py:3441: RuntimeWarning: Mean of empty slice.
  out=out, **kwargs)
/usr/local/lib/python3.7/dist-packages/numpy/core/_methods.py:189: RuntimeWarning: invalid value encountered in double_
  ret = ret.dtype.type(ret / rcount)
/usr/local/lib/python3.7/dist-packages/numpy/core/_methods.py:263: RuntimeWarning: Degrees of freedom <= 0 for slice
  keepdims=keepdims, where=where)
/usr/local/lib/python3.7/dist-packages/numpy/core/_methods.py:223: RuntimeWarning: invalid value encountered in true_di
  subok=False)
/usr/local/lib/python3.7/dist-packages/numpy/core/_methods.py:254: RuntimeWarning: invalid value encountered in double_
  ret = ret.dtype.type(ret / rcount)
```

```
review_input_transformed = vectorizer.transform(data_label)
SVM_prediction = SVM_Clasifier.predict(review_input_transformed)
```

```
import collections, numpy
print("Jumlah review: {}".format(len(data_label)))
SVM = collections.Counter(predictions_SVM_Tfidf)
print("Hasil Klasifikasi SVM : ", SVM)

results = pd.DataFrame({
    "Labeled_Data" : data_label,
    "Label" : SVM_prediction
 })
results.to_csv("Hasil_SVM2.csv", index = False)
```

```
Jumlah review: 44756
Hasil Klasifikasi SVM :  Counter({'Positive': 3899, 'Neutral': 492, 'Negative': 84})
```