SENTIMENT ANALYSIS USING PYTHON (updated)

THAO TRAN (★★★★★)

THU TRAN (★★★)

HUITING WU (★★★★)

1. DATA PROCESSING
2. WORDCLOUD
3. TF-IDF SCORE
4. MACHINE LEARNING MODEL
5. SUMMARY

1 DATA PROCESSING

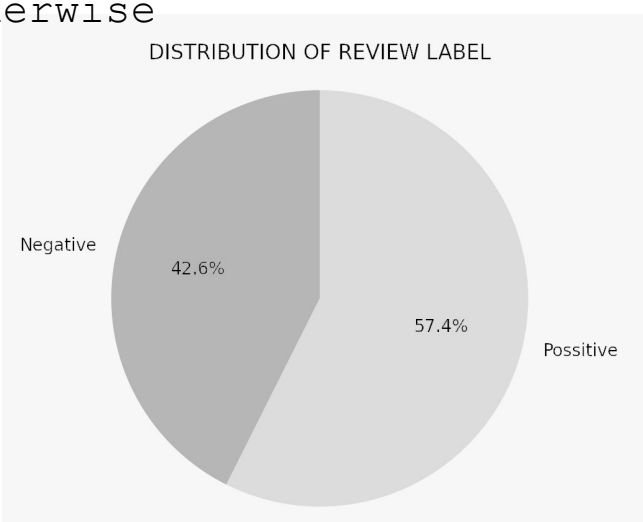
PREVIOUS

CHECK DUPLICATE

Forget to check the duplicates

REVIEW LABEL

Label 1 ('Positive') for rating 5 stars, and label 0 ('Negative') for otherwise



IMPROVEMENT

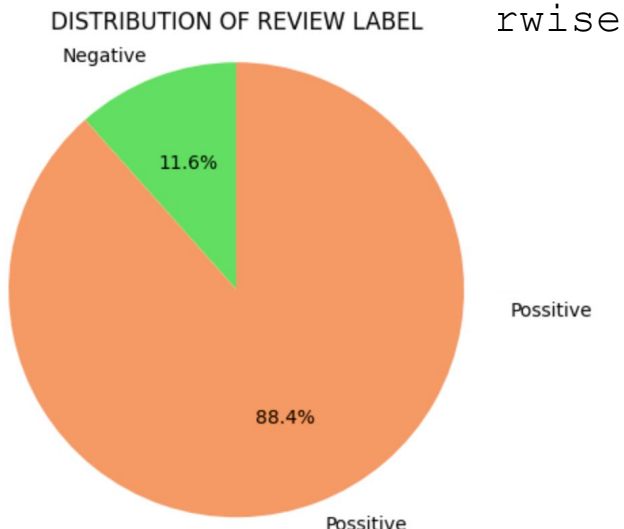
CHECK DUPLICATE

```
# check duplicate
print(data.shape)
data.drop_duplicates(inplace=True)
print(data.shape)
```

```
(9976, 2)
(7868, 2)
```

REVIEW LABEL

Label 1 ('Positive') for rating 3 stars and above,
and label 0 (rwise



1 DATA PROCESSING

IMPROVEMENT

MODIFY THE STOPWORDS

1

```
# Expansion of english contractions
def contraction_expansion(content):
    content = re.sub(r"won\'t", "would not", content)
    content = re.sub(r"can\'t", "can not", content)
    content = re.sub(r"don\'t", "do not", content)
    content = re.sub(r"shouldn\'t", "should not", content)
    content = re.sub(r"needn\'t", "need not", content)
    content = re.sub(r"hasn\'t", "has not", content)
    content = re.sub(r"haven\'t", "have not", content)
    content = re.sub(r"weren\'t", "were not", content)
    content = re.sub(r"mightn\'t", "might not", content)
    content = re.sub(r"didn\'t", "did not", content)
    content = re.sub(r"n\'t", " not", content)
    content = re.sub(r"productread", "product read", content)
    return content
```

1

DATA PROCESSING

IMPROVEMENT

DATA CLEANING AND MODIFY THE STOPWORDS

2

```
#Removing special character
def remove_special_character(content):
    return re.sub('\W+',' ', content )
```

3

```
# Removing URL's
def remove_url(content):
    return re.sub(r'http\S+', '', content)
```

4

```
#Customize stopword as per data
from nltk.corpus import stopwords
stop_words = stopwords.words('english')
new_stopwords = ["would","shall","could","might","battery","headphone",
                 "product","bluetooth","bass","headphones", "read"]
stop_words.extend(new_stopwords)
stop_words.remove("not")
stop_words=set(stop_words)
print(stop_words)
```

1 DATA PROCESSING

IMPROVEMENT

DATA CLEANING AND MODIFY THE STOPWORDS

5

```
#Removing the stopwords from text
def remove_stopwords(content):
    clean_data = []
    for i in content.split():
        if i.strip().lower() not in stop_words and i.strip().lower().isalpha():
            clean_data.append(i.strip().lower())
    return " ".join(clean_data)
```

```
#Data preprocessing
def data_cleaning(content):
    content = contraction_expansion(content)
    content = remove_special_character(content)
    content = remove_url(content)
    content = remove_stopwords(content)
    return content
```

1

DATA PROCESSING

IMPROVEMENT

DATA CLEANING AND MODIFY THE STOPWORDS

```
#Data cleaning
data2['reviews_clean']=data2['review'].apply(data_cleaning)
data2.head(5)
```

	review	rating	label	reviews_clean
0	It was nice produt. I like it's design a lot. ...	5	1	nice produt like design lot easy carry looked ...
1	awesome sound....very pretty to see this nd th...	5	1	awesome sound pretty see nd sound quality good...
2	awesome sound quality. pros 7-8 hrs of battery...	4	1	awesome sound quality pros hrs life including ...
3	I think it is such a good product not only as ...	5	1	think good not per quality also design quite g...
4	awesome bass sound quality very good bettary l...	5	1	awesome sound quality good bettary long life p...

2

WORDCLOUD

PREVIOUS

POSITIVE REVIEW

IMPROVEMENT

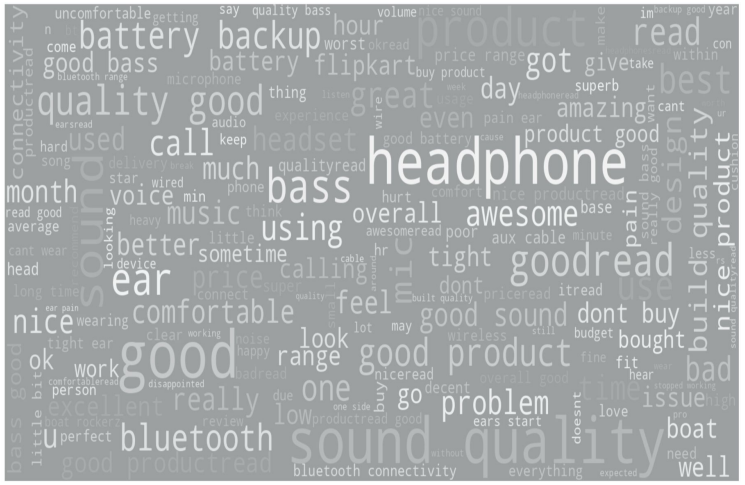
POSITIVE REVIEW

2

WORDCLOUD

PREVIOUS

NEGATIVE REVIEW



IMPROVEMENT

NEGATIVE REVIEW

2

WORDCLOUD

PREVIOUS

POSSITIVE REVIEW

WORDS	COUNT
good	1843
product	1402
sound	1289
quality	1216
bass	943
nice	752
best	659
awesome	610
productread	521
read	477

NEGATIVE REVIEW

WORDS	COUNT
good	1847
sound	1220
quality	1127
product	928
bass	644
battery	398
bluetooth	395
nice	347
use	342
headphone	342

IMPROVEMENT

POSSITIVE REVIEW

WORDS	COUNT
good	3967
sound	2487
quality	2309
not	1810
nice	1152
best	1080
awesome	992
price	843
also	636
backup	569

NEGATIVE REVIEW

WORDS	COUNT
not	681
good	213
sound	206
quality	200
buy	130
working	125
bad	122
ears	102
worst	81
mic	78

3

TF-IDF SCORE

- **Term Frequency - Inverse Document Frequency (TF-IDF)** is a widely used statistical method in natural language processing and information retrieval.
- **Term Frequency**: TF of a term or word is the number of times the term appears in a document compared to the total number of words in the document.
- **Inverse Document Frequency:** IDF of a term reflects the proportion of documents in the X corpus that contain the term. Words unique to a small percentage of documents (e.g., technical jargon terms) receive higher importance values than words common across all documents (e.g., a, the, and).

3

TF-IDF SCORE

$$TF = \frac{\text{number of times the term appears in the document}}{\text{total number of terms in the document}}$$

$$IDF = log(\frac{\text{number of the documents in the corpus}}{\text{number of documents in the corpus contain the term}})$$

$$TF\text{-}IDF = TF * IDF$$

3

TF-IDF SCORE

- **There are 10000 reviews in the dataset. Only 100 reviews contains the term “Love”. And the term “Love” appears 20 times in one of the reviews which contains 100 words in total.**

TF = 20/100 = 0.2

IDF = log(10000/100) = 2

TF-IDF score of “Love” in that document is TF*IDF = 0.2*2 = 0.4

3

TF-IDF SCORE

```python
from sklearn.feature_extraction.text import TfidfVectorizer

documents = [
    "This is the first document.",
    "This document is the second document.",
    "And this is the third one.",
    "Is this the first document?",
]

# Create the TF-IDF vectorizer
vectorizer = TfidfVectorizer()

# Fit and transform the documents
tfidf_matrix = vectorizer.fit_transform(documents)

# Get the feature names (words)
feature_names = vectorizer.get_feature_names_out()

# Create a dense matrix representation for easier understanding
dense_matrix = tfidf_matrix.todense()

# Display the results
print("Feature Names (Words):", feature_names)
print("TF-IDF Matrix:")
print(dense_matrix)
```

EXAMPLE

```
Feature Names (Words): ['and' 'document' 'first' 'is' 'one' 'second' 'the' 'third' 'this']
TF-IDF Matrix:
[[0.         0.46979139 0.58028582 0.38408524 0.         0.
  0.38408524 0.         0.38408524]
 [0.         0.6876236  0.         0.28108867 0.         0.53864762
  0.28108867 0.         0.28108867]
 [0.51184851 0.         0.         0.26710379 0.51184851 0.
  0.26710379 0.51184851 0.26710379]
 [0.         0.46979139 0.58028582 0.38408524 0.         0.
  0.38408524 0.         0.38408524]]
```

RESOURCES

TF-IDF — Term Frequency-Inverse Document Frequency by Fatih Karabiber

https://www.learndatasci.com/glossary/tf-idf-term-frequency-inverse-document-frequency/#:~:text=Using%20scikit%2Dlearn-,What%20is%20TF%2DIDF%3F,%2C%20relative%20to%20a%20corpus

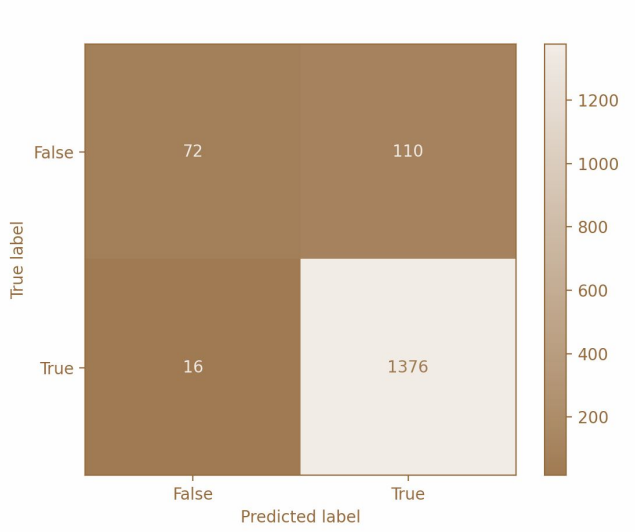
4

MACHINE LEARNING MODEL

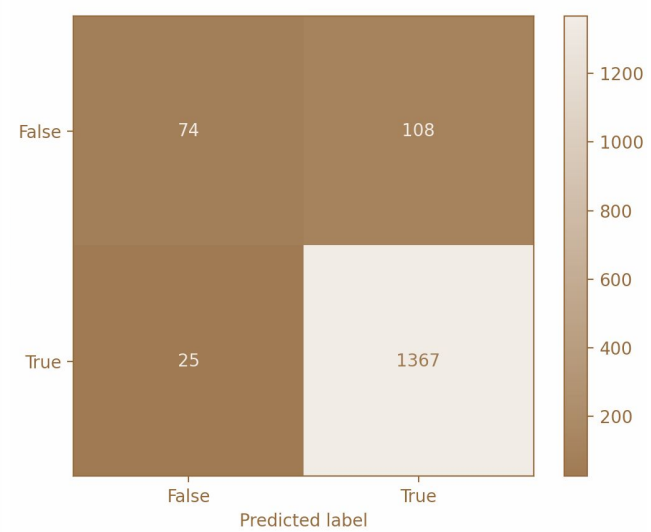
DATA SPLITING

```
X_train1, X_test1, y_train1, y_test1 = train_test_split(X1, data2['label'],
                                test_size=0.2,
                                stratify=data2['label'],
                                random_state = 42)
```

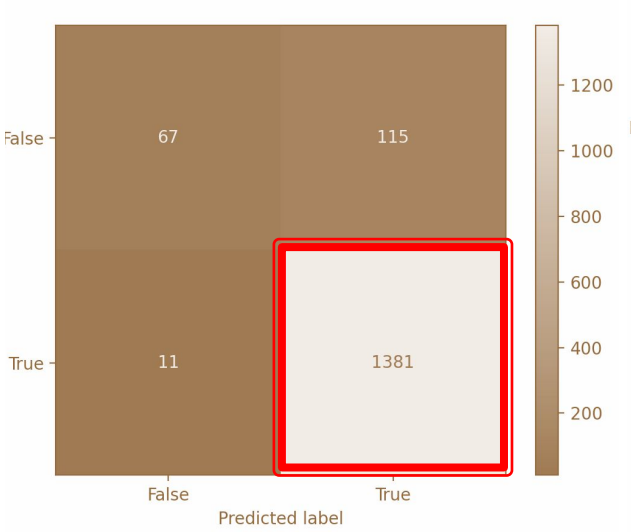
CONFUSION MATRIX



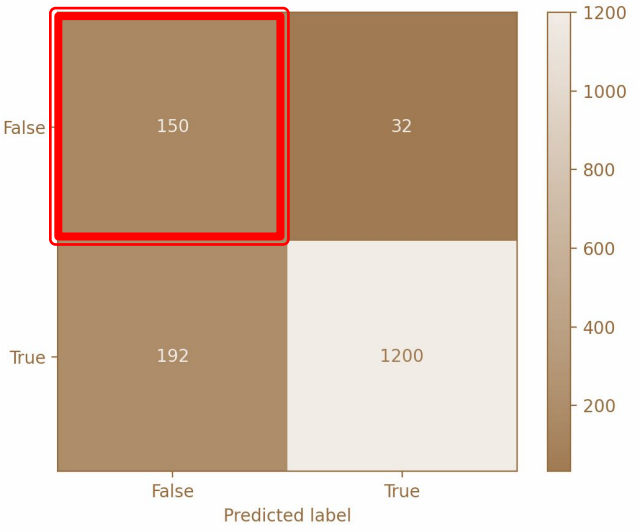
LOGISTIC REGRESSION



RANDOM FOREST



SUPPORT VECTOR MACHINES



XGBOOST

4

MACHINE LEARNING MODEL

CROSS VALIDATION

HYPER PARAMETER TUNNING

```
from sklearn.model_selection import RandomizedSearchCV
from scipy.stats import randint
import numpy as np
from sklearn.metrics import make_scorer,classification_report,f1_score, roc_auc_score, precision_score, recall_score

# Define the hyperparameter grid
par_grid = dict(max_depth = randint(low=2, high=6), n_estimators=randint(low=50, high=150), \
    min_samples_split = randint(low=2, high=6), max_samples=[0.3, 0.5, 0.7, 0.9] )
```

RANDOMIZEDSEARCHCV
(using R.Forest estimator)

```
from sklearn.ensemble import RandomForestClassifier

# Instantiate the classifier
rf = RandomForestClassifier()

# Instantiate the RandomizedSearchCV object
random_search = RandomizedSearchCV(rf, par_grid, cv=5, scoring='roc_auc', n_jobs=-1,random_state=42)

# Fit to the data
cv_model = random_search.fit(X_train1, y_train1)

# Print the best parameters and the best score
print(f"Best parameters: {random_search.best_params_}")
auc_rcv=random_search.best_score_
print(f"AUC score: {random_search.best_score_:.2f}")
```

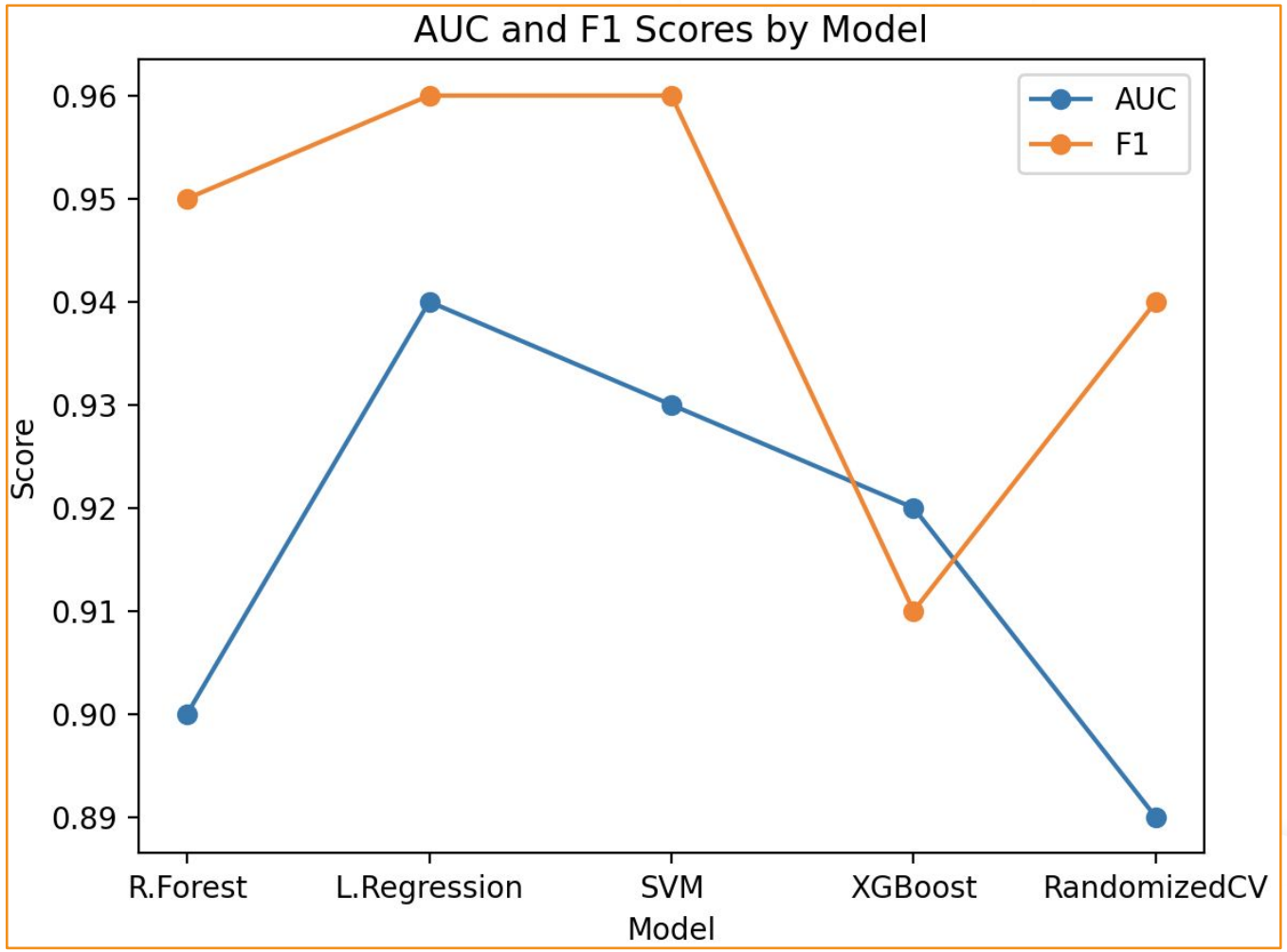
OUTPUT

```
Best parameters: {'max_depth': 4, 'max_samples': 0.9, 'min_samples_split': 5, 'n_estimators': 132}
AUC score: 0.89
```

4 MACHINE LEARNING MODEL

METRIC TABLE

Model	AUC	F1
Random Forest	0.9	0.95
Logistic Regression	0.94	0.96
Support Vector Machine	0.93	0.96
XGBoost	0.92	0.91
RandomizedCV	0.89	0.94



5

SUMMARY

- Some steps that we use to update our model
  - Remove the duplicate rows
  - Change ‘label’ of positive and negative review
  - Modify the remove “stopwords”
  - Better wordcloud visualization.
  - Change the splitting data to 80:20
  - Use different method of machine learning model: Logistic Regression, Random Forest, SVM, XGBoost, RandomizedSearchCV.
  - Use different metrics to evaluate the model: AUC, F1-score
- **CONCLUSION:**

  After doing all steps above, the best machine learning model to predict “positive” and “negative” reviews is logistic regression with the AUC = 0.94, and F1-score = 0.96