Indonesian Journal of Computer Science
ISSN 2549-7286 (*online*)
Jln. Khatib Sulaiman Dalam No. 1, Padang, Indonesia
Website: ijcs.stmikindonesia.ac.id | E-mail: ijcs@stmikindonesia.ac.id

Sentiment Analysis Performance Value Optimization Using Hyperparamater Tunning With Grid Search On Shopee App Reviews

Muhammad Luthfi Al-Ghifari, Ken Ditha Tania*
luthfidiky011102@gmail.com, kenyta.tania@gmail.com
Universitas Sriwijaya

Article Information

Submitted : 11 Sep 2023
Reviewed: 18 Sep 2023
Accepted : 1 Oct 2023

Keywords

Sentiment Analysis, Shopee, TF-IDF, Hyperparameter Tuning, Gridsearch

Abstract

The rapid development of technology today has provided convenience for us in today's civilization. One of these developments is the invention of the internet due to high internet penetration and rapid growth in mobile usage, online shopping has increased tremendously. This online shopping is now often referred to as e-commerce. E-commerce is one of the trade models that has been widened under the effect of extensive use of technology. Specifically, e-commerce refers to the usage of the Internet or other networks. Shopee is one of the popular marketplaces in Indonesia that has the highest number of visitors of 129 million per month and can be downloaded on the Google Play Store. Play Store itself has several features such as Reviews that can allow users to give opinions. All complaints and opinions from shopee users can be channeled into this feature. With this a research aims to optimize the performance value of sentiment analysis with the Term Frequency-Inverse Document Frequency (TF-IDF) method and Hyperparameter Tuning with Gridsearch for the Shopee application on the Google Play Store. Based on research the reviews resulting in 3000 data where 2015 user data is positive and 985 data is negative. Testing data was split by a ratio of 90:10 for 300 data test in each classification model to find the accuracy score. With hyperparameter tuning using gridsearch we can see the result of each accuracy score of KNN, DCT, RF, and LR is increasing from 0.73 to 0.77, 0.823 to 0.826, 0.856 to 0.87, and 0.856 to 0.866. This indicated that among the machine learning model that had been tuning using gridsearch, KNN is the one that highly increased.

A. Introduction

The rapid development of technology today has provided convenience for us in today's civilization. One of these developments is the invention of the internet [1]. In the recent years, due to high internet penetration and rapid growth in mobile usage, online shopping has increased tremendously and has become very popular [2]. Internet technology has made all human activities very practical and fast, this online shopping is now often referred to as e-commerce.

E-commerce is one of the trade models that has been widened under the effect of extensive use of technology. Specifically, e-commerce refers to the usage of the Internet or other networks (e.g., intranets) to purchase, sell, trade data, goods, or services. This already revolutionized the way businesses operate and consumers shop[3], [4]. With the advancement of technology, e-commerce platforms such as Shopee, Tokopedia, Bukalapak have emerged as popular online marketplaces that connect sellers and buyers in a convenient and efficient way [5]. Therefore e-commerce is one of the shopping media that is often used by people today to make it easier for them to buy goods from afar without going to the shopping place itself, people who want to sell goods or services and shop simply click on the screen of their gadget, then just sit back and wait for the goods to reach their hands.

Shopee is one of the popular marketplaces in Indonesia where Shopee is a mobile marketplace application that has the highest number of visitors of 129 million per month [6]. Shopee is an electronic buying and selling application that can be downloaded on the Google Play Store. Play Store itself is a site that provides several applications ranging from music, movies, books and various categories that can be downloaded online [7]. Play Store itself has several features such as Reviews that can allow users to give opinions or comments on other people's work. Rating that can make users give value to an application. All complaints and opinions from shopee users can be channeled into this feature on the Google Play Store [8]. Sentiment analysis is carried out to see user responses to the Shopee application, whether the reviews are good or bad if we look from the perspective of comments or ratings obtained by each review.

In Table 1, there is attached some research about sentiment analysis. This literature search using some keywords such as “Sentiment Analysis using Decision Tree, Naïve Bayes on the Shopee”, “Sentiment Analysis Using K-Nearest Neighbor, SVM, and TF-IDF”, and “Gridsearch Sentiment Analysis using KNN, Random Forest, Logistic Regression, Naives Bayes, and SVM”.

Based on the literature review, most of research is focused on getting better accuracy using various algorithm. Some of the method using tuning model with gridsearch hyperparameter, TF-IDF, and N-Gram. SVM, Decision Tree, Naïve Bayes, KNN, Logistic Regression, and Random Forest are the algorithm used with hyperparameter tuning in Gridsearch for getting better accuracy. By adding a hyperparameter optimization will get a better result to determine hyperparameter efficiency in choosing parameter in sentiment analysis[9] Thus, after obtaining the best parameters, we can choose what machine learning models are accurate based on the accuracy score.

Table 1. Literature Review [10]

Reference	Topic	Accuracy Score	Precision Score	Recall Score	Results

[11]	Sentiment analysis on the shopee application rating using the decision tree method with SMOTE	99.91%	99.98%	99.88%	Better results using SMOTE
[12]	Sentiment analysis shopee reviews on twitter using KNN	Testing data 70 : 10 = 83% 80 : 20 = 90%	-	-	Using the KNN method, the accuracy obtained by testing 10 k values in the distribution of training data and testing data is 70%: 30% having an accuracy of 83% and 80%: 20% have an accuracy of 90%.
[13]	Sentiment analysis of game product on shopee using the TF-IDF method and naive bayes classifier	80.94%	60.33%	80.22%	Combining TF IDF with Naïve bayes make the sentiment analysis reviews a good accuracy
[14]	Implementation of feature extraction using the TF-IDF method and N-Gram model to analyze sentiment reviews	88.4%	87.3%	88.4%	SVM algorithm with TF-IDF feature extraction using unigram shows great potential
[9]	Propose a hybrid approach using the random forest classifier and the grid search method for customer feedback sentiment prediction	90.02%	85.93%	96.13%	Increasing accuracy of Random forest using tuning number of maximum trees in forest and depth of tree
[15]	Optimizing K-Nearest Neighbor based breast cancer detection using Hyperparameter tuning	Before tuning : 90.10% After tuning : 94.35%	-	-	The accuracy of the optimized model with tuned hyper-parameters is 94.35%, while the accuracy of the KNN model with default hyper-parameters is 90.10%. This indicates that hyper-parameter tuning enhances the

					accuracy and performance
[16]	Peningkatan Kinerja Akurasi Prediksi Penyakit Diabetes Mellitus Menggunakan Metode Grid Seacrh pada Algoritma Logistic Regression	Before tuning : 78% After tuning : 80%	Before tuning : 77% After tuning : 79%	Before tuning : 77% After tuning : 79%	The results of research on improving the performance of prediction accuracy prediction performance of diabetes mellitus disease against the comparison of Logistic Regression model comparison without Grid Search and with Grid Search there is a significant increase
[17]	Comparing the performance of naive bayes and svm classifiers and to identify the significant hyperparameters for the classifiers.	NB : 68.70% SVM : 85.65%	NB : 83.22% SVM : 87.60%	NB : 75.36% SVM : 88.47%	SVM has a better performance than Naive Bayes based on sentiment analysis of healthcare companies' stock comments

Through the background that has been described previously, this research aims to optimize the performance value of sentiment analysis with the Term Frequency-Inverse Document Frequency (TF-IDF) method and Hyperparameter Tuning with Gridsearch for the Shopee application on the Google Play Store.

B. Research Method

The method used in solving the problem in this research is by analyzes user reviews on the Shopee application on the Google Play site, in producing a good accuracy level performance value, the selected model will be carried out Hyperparameter Tuning with Gridsearch in weighting using the Term Frequency-Inverse Document Frequency (TF-IDF) method which can be seen in Figure 1.

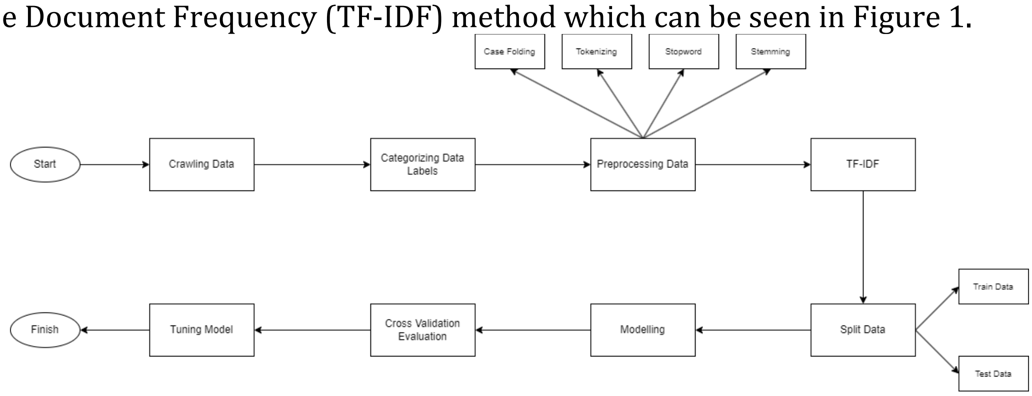


Figure 1. Research Stages [18]

2.1. *Crawling Data*

The data collection process is obtained from the Google Play Store site by taking raing data on the Shopee application. Data collection is obtained by performing scraping techniques using the Python algorithm on Google Colab.

```
[ ]  !pip install google-play-scraper

     Collecting google-play-scraper
       Downloading google_play_scraper-1.2.4-py3-none-any.whl (28 kB)
     Installing collected packages: google-play-scraper
     Successfully installed google-play-scraper-1.2.4
```

Figure 2. Installing google play scrapper package

```
[ ]  from google_play_scraper import app

     import pandas as pd

     import numpy as np

[ ]  from google.colab import drive
     drive.mount('/content/drive')

     Mounted at /content/drive
```

Figure 3. Import library needed

```
[ ]  from google_play_scraper import Sort, reviews

     result, continuation_token = reviews(
         'com.shopee.id',
         lang='id', # defaults to 'en'
         country='id', # defaults to 'us'
         sort=Sort.NEWEST, # defaults to Sort.MOST_RELEVANT you can use Sort.NEWEST to get newst reviews
         count=3000, # defaults to 100
         filter_score_with=None # defaults to None(means all score) Use 1 or 2 or 3 or 4 or 5 to select certain score
     )
```

Figure 4. Retrieve reviews data in shopee app

```
my_df.to_csv('/content/drive/MyDrive/shopee_relevant_1.csv', index=False)
```

Figure 5. Saving data in form of a csv file

We can see the example data that has been crawled in figure 6 below.

	userName	score	at	content
0	deny wibi	5	2023-07-18 18:41:50	Masukan agar dibantu aktifasi spaylater dan sp...
1	Bango Ranca	5	2023-07-18 18:38:51	barangnya bagus2.toko nya amanah
2	Alya Safitri	3	2023-07-18 18:37:50	Shoppe ko suka ngelag2 gitu si, kadang kalo pe...
3	Sendy Wardhana	5	2023-07-18 18:35:39	Gratis ongkir
4	Suhendri Bowo	5	2023-07-18 18:34:11	Keren
...	...	...	...	...
2995	el piazon	5	2023-07-16 11:41:24	Bagus
2996	Harry Sunarwan	5	2023-07-16 11:40:50	sangat suka
2997	Nurah Siti	5	2023-07-16 11:40:17	sangat membantu
2998	Tika Lasari	5	2023-07-16 11:39:44	Lebih tingkatkn lagi pelayanan shope
2999	Iqbal Super502	1	2023-07-16 11:39:39	We rasah marakke bocilku badmood suu cokk tak ...

3000 rows × 5 columns

Figure 6. Some of the result of the reviews data taken

2.2. Categorizing Data Labels

In categorizing the data labels, 3,000 review data were taken. Reviews with ratings 4 and 5 are labeled as positive sentiment, ratings 1, 2, 3 as negative sentiment automatically. Obtained review data with a label of 2015 positive sentiment or around 67.2% and 985 negative sentiment or around 32.8%.

```python
sentiment = []
for index, row in my_df.iterrows():
    if row["score"] == 5:
        sentiment.append(1)
    elif row["score"] ==4:
        sentiment.append(1)
    else:
        sentiment.append(0)

my_df["sentiment"] = sentiment
my_df
```

	userName	score	at	content	sentiment
0	deny wibi	5	2023-07-18 18:41:50	Masukan agar dibantu aktifasi spaylater dan sp...	1
1	Bango Ranca	5	2023-07-18 18:38:51	barangnya bagus2.toko nya amanah	1
2	Alya Safitri	3	2023-07-18 18:37:50	Shoppe ko suka ngelag2 gitu si, kadang kalo pe...	0
3	Sendy Wardhana	5	2023-07-18 18:35:39	Gratis ongkir	1
4	Suhendri Bowo	5	2023-07-18 18:34:11	Keren	1
...	...	...	...	...	...
2995	el piazon	5	2023-07-16 11:41:24	Bagus	1
2996	Harry Sunarwan	5	2023-07-16 11:40:50	sangat suka	1
2997	Nurah Siti	5	2023-07-16 11:40:17	sangat membantu	1
2998	Tika Lasari	5	2023-07-16 11:39:44	Lebih tingkatkn lagi pelayanan shope	1
2999	Iqbal Super502	1	2023-07-16 11:39:39	We rasah marakke bocilku badmood suu cokk tak ...	0

3000 rows × 5 columns

Figure 7. Result of data that has been labeled

```python
my_df['sentiment'].value_counts()
```

```
1    2015
0     985
Name: sentiment, dtype: int64
```

Figure 8. Positive and negative reviews comparison

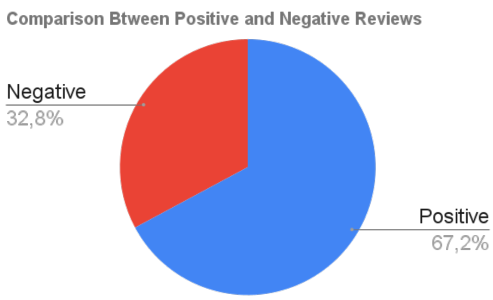


Figure 9. Plot pie positive and negative reviews

2.3. Preprocessing Data

Preprocessing is one of the important stages for data in the mining process. The data used in the mining process is not always in an ideal condition for processing [19]. The goal is to clean, transform, and prepare the data to make it compatible for use in the model, thus improving the performance and predictive results of the model.

```python
my_df=my_df[['content', 'sentiment']]#because we only need the content and sentiment columns then we do another column filter
```

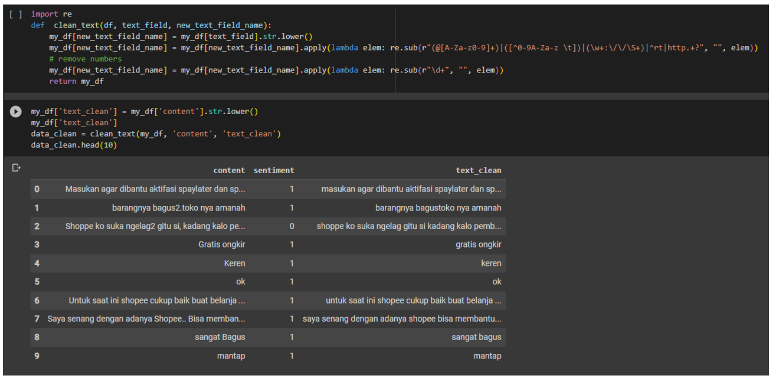
```python
my_df.head(5)
```

	content	sentiment
0	Masukan agar dibantu aktifasi spaylater dan sp...	1
1	barangnya bagus2.toko nya amanah	1
2	Shoppe ko suka ngelag2 gitu si, kadang kalo pe...	0
3	Gratis ongkir	1
4	Keren	1

Figure 10. Column data that will be preprocessed

In this stage the processed data will be corrected and delete some unnecessary data. The stages of data preprocessing are as follows.

2.3.1. *Case Folding* is the process of converting all letters to small so that the letters become uniform.



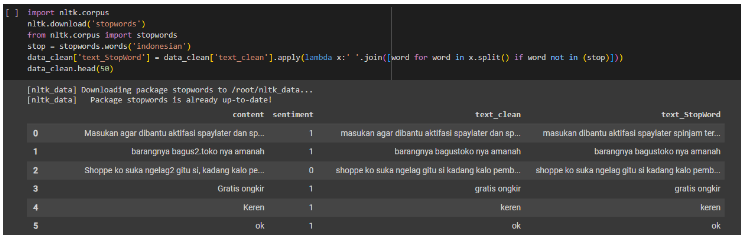
```
import re
def  clean_text(df, text_field, new_text_field_name):
    my_df[new_text_field_name] = my_df[text_field].str.lower()
    my_df[new_text_field_name] = my_df[new_text_field_name].apply(lambda elem: re.sub(r"(@[A-Za-z0-9]+)|([^0-9A-Za-z \t])|(\w+:\/\/\S+)|^rt|http.+?", "", elem))
    # remove numbers
    my_df[new_text_field_name] = my_df[new_text_field_name].apply(lambda elem: re.sub(r"\d+", "", elem))
    return my_df
```

```
my_df['text_clean'] = my_df['content'].str.lower()
my_df['text_clean']
data_clean = clean_text(my_df, 'content', 'text_clean')
data_clean.head(10)
```

	content	sentiment	text_clean
0	Masukan agar dibantu aktifasi spaylater dan sp...	1	masukan agar dibantu aktifasi spaylater dan sp...
1	barangnya bagus2.toko nya amanah	1	barangnya bagustoko nya amanah
2	Shoppe ko suka ngelag2 gitu si, kadang kalo pe...	0	shoppe ko suka ngelag gitu si kadang kalo pemb...
3	Gratis ongkir	1	gratis ongkir
4	Keren	1	keren
5	ok	1	ok
6	Untuk saat ini shopee cukup baik buat belanja ...	1	untuk saat ini shopee cukup baik buat belanja ...
7	Saya senang dengan adanya Shopee.. Bisa memban...	1	saya senang dengan adanya shopee bisa membantu...
8	sangat Bagus	1	sangat bagus
9	mantap	1	mantap

Figure 11. Program code for case folding

2.3.2. *Stopword* is the filtering of words that represent the document so that words that are considered unimportant are discarded. Examples of unimportant words are "in", "and", "to", "from", etc.

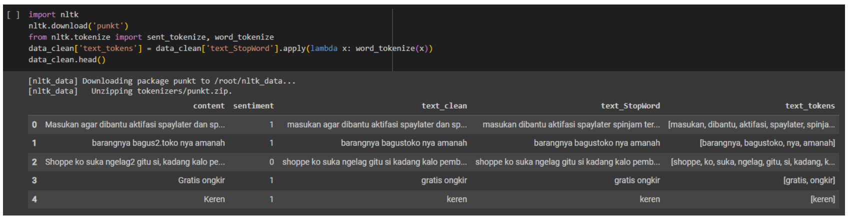


```
import nltk.corpus
nltk.download('stopwords')
from nltk.corpus import stopwords
stop = stopwords.words('indonesian')
data_clean['text_StopWord'] = data_clean['text_clean'].apply(lambda x:' '.join([word for word in x.split() if word not in (stop)]))
data_clean.head(50)

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

	content	sentiment	text_clean	text_StopWord
0	Masukan agar dibantu aktifasi spaylater dan sp...	1	masukan agar dibantu aktifasi spaylater dan sp...	masukan dibantu aktifasi spaylater spinjam ter...
1	barangnya bagus2.toko nya amanah	1	barangnya bagustoko nya amanah	barangnya bagustoko nya amanah
2	Shoppe ko suka ngelag2 gitu si, kadang kalo pe...	0	shoppe ko suka ngelag gitu si kadang kalo pemb...	shoppe ko suka ngelag gitu si kadang kalo pemb...
3	Gratis ongkir	1	gratis ongkir	gratis ongkir
4	Keren	1	keren	keren
5	ok	1	ok	ok

Figure 12. Program code for stopword

2.3.3. *Tokenizing* each word will be separated based on the spaces found.

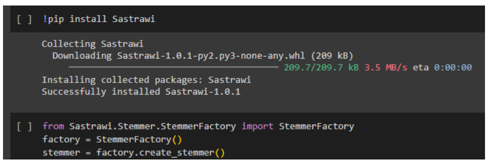


```
import nltk
nltk.download('punkt')
from nltk.tokenize import sent_tokenize, word_tokenize
data_clean['text_tokens'] = data_clean['text_StopWord'].apply(lambda x: word_tokenize(x))
data_clean.head()

[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
```

	content	sentiment	text_clean	text_StopWord	text_tokens
0	Masukan agar dibantu aktifasi spaylater dan sp...	1	masukan agar dibantu aktifasi spaylater dan sp...	masukan dibantu aktifasi spaylater spinjam ter...	[masukan, dibantu, aktifasi, spaylater, spinja...
1	barangnya bagus2.toko nya amanah	1	barangnya bagustoko nya amanah	barangnya bagustoko nya amanah	[barangnya, bagustoko, nya, amanah]
2	Shoppe ko suka ngelag2 gitu si, kadang kalo pe...	0	shoppe ko suka ngelag gitu si kadang kalo pemb...	shoppe ko suka ngelag gitu si kadang kalo pemb...	[shoppe, ko, suka, ngelag, gitu, si, kadang, k...
3	Gratis ongkir	1	gratis ongkir	gratis ongkir	[gratis, ongkir]
4	Keren	1	keren	keren	[keren]

Figure 13. Program code for tokenizing

2.3.4. *Stemming* converting a compound word into a base word.



```
!pip install Sastrawi

Collecting Sastrawi
  Downloading Sastrawi-1.0.1-py2.py3-none-any.whl (209 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 209.7/209.7 kB 3.5 MB/s eta 0:00:00
Installing collected packages: Sastrawi
Successfully installed Sastrawi-1.0.1
```

```
from Sastrawi.Stemmer.StemmerFactory import StemmerFactory
factory = StemmerFactory()
stemmer = factory.create_stemmer()
```

Figure 14. Package for stemming

```
# stemmed
def stemmed_wrapper(term):
    return stemmer.stem(term)

term_dict = {}
for document in data_clean['text_tokens']:
    for term in document:
        if term not in term_dict:
            term_dict[term] = ' '

for term in term_dict:
    term_dict[term] = stemmed_wrapper(term)

# apply stemmed term to dataframe
def get_stemmed_term(document):
    return [term_dict[term] for term in document]

data_clean['komen_stemmed'] = data_clean['text_tokens'].apply(get_stemmed_term)
print(data_clean['komen_stemmed'].tail(10))
```

```
2990                                    [mantab, fitur, yg]
2991                                       [bantu, bangett]
2992    [recomended, perlu, mudah, up, date, teruuuss,...
2993    [apk, yng, bgus, harga, nya, apk, shopee, mura...
2994                                               [mantap]
2995                                                [bagus]
```

Figure 15. Program code for stemming

2.4. Term Frequency-Inverse Document Frequency (TF-IDF)

Term Frequency - Inverse Document Frequency (TF-IDF) is a widely used statistical method in natural language processing and information retrieval. It measures how important a term is within a document relative to a collection of documents (i.e., relative to a corpus). Words within a text document are transformed into importance numbers by a text vectorization process [20]. By combining it, we can get a TF-IDF score for each word in the document. This score reflects how important the context of word in document and the overall selection. Words with high TF-IDF scores tend to have more importance in the document.

```
import sklearn
from sklearn.feature_extraction.text import CountVectorizer

max_features = 1000
databaru= preprocessed_data['komen_stemmed'].astype(str)


cvect = CountVectorizer(max_features=max_features, ngram_range=(1,3), binary=True)
# calculated TF only
TF_vector = cvect.fit_transform(databaru).toarray()
```

Figure 16. Program word to perform term weighting

```
[ ]  X= TF_vector
     Y= preprocessed_data['sentiment']
```

Figure 17. Define X and Y for modelling

2.5. Split Data

To evaluate the prediction results, test data taken from the dataset is used. In this research, the ratio of training data and test data is 90:10[21]. The division of which is generated through scikit-learn with the 90% of 2700 train data and 10% of 300 test data will be used.

```
[ ]  #splitting data
     from sklearn.model_selection import train_test_split
     X_train,X_test,Y_train,Y_test= train_test_split(X,Y,test_size=0.1,random_state=0)
```

Figure 18. Program code for splitting data

2.6. Modelling

We will determine the model with a dataset that has been split based on 4 existing algorithms, namely, K-Nearest Neighbor, Decision Tree, Random Forest, and Logistic Regression[22], [23]. Here we will look for the accuracy value of each algorithm on the dataset and then evaluate with Cross Validation to get the highest accuracy results at the model tuning stage.

```python
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression

from sklearn import metrics
from matplotlib import pyplot as plt
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
```

Figure 19. Import package for modelling

Table 2. Program code for classification each model

Modelling

Model 1 : K-Nearest Neighbor

```python
y_pred_knn = knn_model.predict(X_test)
print('Classification Report for K Nearest Neighbor')
print(' ')
print(classification_report(Y_test, y_pred_knn))
print('-'*50)
```

Model 2 : Decision Tree

```python
y_pred_dct = dct_model.predict(X_test)
print('Classification Report for Decision Tree')
print(' ')
print(classification_report(Y_test, y_pred_dct))
print('-'*50)
```

Model 3 : Random Forest

```python
y_pred_rdf = rdf_model.predict(X_test)
print('Classification Report for Random Forest')
print(' ')
print(classification_report(Y_test, y_pred_rdf))
print('-'*50)
```

Model 4 : Logistic Regression

```python
y_pred_lr = lr_model.predict(X_test)
print('Classification Report for Logistic Regression')
print(' ')
print(classification_report(Y_test, y_pred_lr))
print('-'*50)
```

2.7. Cross Validation Evaluation

Cross-validation evaluation adalah a method used in machine learning to measure model performance more accurately by utilizing available data more efficiently. This technique provides the ability to estimate model performance on unseen data not used while training, tuning model hyperparameter, resolving the issues about third split, and avoid instability of sampling [24]. In gridsearch, every combination of a preset list of values of hyperparameters is tried, such that the best combination is chosen based on the cross-validation score [25]. By using cross-validation at each step of the evaluation, you ensure that you evaluate the model's performance more accurately and objectively across a wide range of data.

```python
from sklearn.model_selection import KFold, cross_val_score

k_folds_3 = KFold(n_splits = 3)
k_folds_4 = KFold(n_splits = 4)
k_folds_5 = KFold(n_splits = 5)
```

Figure 20. Import package for Cross Validation

Table 3. Program code for each cross validation modelling

Cross Validation Evaluation Code

Model 1 : K-Nearest Neighbor

```python
# K Nearest Neighbor
scores_knn_3 = cross_val_score(knn_model, X, Y, cv = k_folds_3)
print("Average CV Score for n_splits = 3 : ", scores_knn_3.mean())

scores_knn_4 = cross_val_score(knn_model, X, Y, cv = k_folds_4)
print("Average CV Score for n_splits = 4 : ", scores_knn_4.mean())

scores_knn_5 = cross_val_score(knn_model, X, Y, cv = k_folds_5)
print("Average CV Score for n_splits = 5 : ", scores_knn_5.mean())
```

Model 2 : Decision Tree

```python
# Decision Tree
scores_dct_3 = cross_val_score(dct_model, X, Y, cv = k_folds_3)
print("Average CV Score for n_splits = 3 : ", scores_dct_3.mean())

scores_dct_4 = cross_val_score(dct_model, X, Y, cv = k_folds_4)
print("Average CV Score for n_splits = 4 : ", scores_dct_4.mean())

scores_dct_5 = cross_val_score(dct_model, X, Y, cv = k_folds_5)
print("Average CV Score for n_splits = 5 : ", scores_dct_5.mean())
```

Model 3 : Random Forest

Model 4 : Logistic Regression

```
[ ] # Random Forest
    scores_rdf_3 = cross_val_score(rdf_model, X, Y, cv = k_folds_3)
    print("Average CV Score for n_splits = 3 : ", scores_rdf_3.mean())

    scores_rdf_4 = cross_val_score(rdf_model, X, Y, cv = k_folds_4)
    print("Average CV Score for n_splits = 4 : ", scores_rdf_4.mean())

    scores_rdf_5 = cross_val_score(rdf_model, X, Y, cv = k_folds_5)
    print("Average CV Score for n_splits = 5 : ", scores_rdf_5.mean())
```

```
[ ] # Logistic Regression
    scores_lr_3 = cross_val_score(lr_model, X, Y, cv = k_folds_3)
    print("Average CV Score for n_splits = 3 : ", scores_lr_3.mean())

    scores_lr_4 = cross_val_score(lr_model, X, Y, cv = k_folds_4)
    print("Average CV Score for n_splits = 4 : ", scores_lr_4.mean())

    scores_lr_5 = cross_val_score(lr_model, X, Y, cv = k_folds_5)
    print("Average CV Score for n_splits = 5 : ", scores_lr_5.mean())
```

2.8. Tuning Model

Tuning model is a step to provide parameters to the model that has the highest accuracy based on datasets that have been searched for k-fold or cross validation values. The evaluation results of the tuning model are carried out with the GridsearchCV library from Python to select the best parameters[21]. It is a method to determine the optimal hyperparameters of a model which aids in higher accurate prediction. Gridsearch function also consists of a scoring parameter which helps in specifying the metric to be assessed on [26] This avoids the manual trial-and-error method and allows you to search for parameter combinations in a more systematic and efficient manner. However, this technique can be time-consuming, especially if there are many hyperparameters to be determined and the search space is large.

```
[ ] from sklearn.model_selection import GridSearchCV
```

Figure 21. Import package for gridsearch

Table 4. Best model for each model

Get Best Model

Model 1 : K-Nearest Neighbor

```
[ ] knn_model.get_params()

    {'algorithm': 'auto',
     'leaf_size': 30,
     'metric': 'minkowski',
     'metric_params': None,
     'n_jobs': None,
     'n_neighbors': 5,
     'p': 2,
     'weights': 'uniform'}

    parameters_knn = {
                    # 'algorithm': ['auto', 'ball_tree', 'kd_tree', 'brute'],
                    # 'leaf_size': [10, 20, 30, 40, 50],
                    'metric': ['minowski', 'euclidean', 'canberra'],
                    # 'metric_params': ['dict', None],
                    # 'n_jobs': [None, -1],
                    'n_neighbors': [2, 3, 4, 5, 6, 7],
                    # 'p': [1, 2, 3, 4, 5, 6, 7],
                    'weights': ['uniform', 'distance']
                    }
```

Model 2 : Decision Tree

```
[ ] dct_model.get_params()

    {'ccp_alpha': 0.0,
     'class_weight': None,
     'criterion': 'gini',
     'max_depth': None,
     'max_features': None,
     'max_leaf_nodes': None,
     'min_impurity_decrease': 0.0,
     'min_samples_leaf': 1,
     'min_samples_split': 2,
     'min_weight_fraction_leaf': 0.0,
     'random_state': None,
     'splitter': 'best'}

    parameters_dct = {
                    # 'ccp_alpha': [0.0, 0.1, 0.5, 1.0],
                    # 'class_weight': ['balanced', 'balanced_subsample', None],
                    'criterion': ['gini', 'entropy', 'log_loss'],
                    'max_depth': [1, 2, 3, 4, 5, None],
                    'max_features': [1, 2, 3, 1.0, 2.0, 3.0, 'auto', 'sqrt', 'log2', None],
                    'max_leaf_nodes': [1, 2, 3, 4, 5, None],
                    # 'min_impurity_decrease': 0.0,
                    # 'min_samples_leaf': 1,
                    # 'min_samples_split': 2,
                    # 'min_weight_fraction_leaf': 0.0,
                    # 'random_state': [0, 42, 123, None],
                    # 'splitter': ['best', 'random']
                    }
```

Model 3 : Random Forest

```
[ ] rdf_model.get_params()

    {'bootstrap': True,
     'ccp_alpha': 0.0,
     'class_weight': None,
     'criterion': 'gini',
     'max_depth': None,
     'max_features': 'sqrt',
     'max_leaf_nodes': None,
     'max_samples': None,
     'min_impurity_decrease': 0.0,
     'min_samples_leaf': 1,
     'min_samples_split': 2,
     'min_weight_fraction_leaf': 0.0,
     'n_estimators': 100,
     'n_jobs': None,
     'oob_score': False,
     'random_state': None,
     'verbose': 0,
     'warm_start': False}

    parameters_rdf = {
                    # 'bootstrap': [False, True],
                    # 'ccp_alpha': [0.0, 0.1, 0.2, 0.3, 0.4, 0.5],
                    # 'class_weight': ['balanced', 'balanced_subsample'],
                    'criterion': ['gini', 'entropy', 'log_loss'],
                    # 'max_depth': [1, 2, 3, 4, 5, None],
                    'max_features': ['sqrt', 'log2', None],
                    # 'max_leaf_nodes': [1, 2, 3, 4, 5, None],
                    # 'max_samples': [1, 2, 3, 4, 5, None],
                    # 'min_impurity_decrease': 0.0,
                    # 'min_samples_leaf': 1,
                    # 'min_samples_split': 2,
                    # 'min_weight_fraction_leaf': 0.0,
                    'n_estimators': [10, 20, 30, 40, 50],
                    # 'n_jobs': None,
                    # 'oob_score': [True False],
                    # 'random_state': [0, 42, 123, None],
                    # 'verbose': [0, 1, 10, 100],
                    # 'warm_start': [False, True]
```

Model 4 : Logistic Regression

```
[ ] lr_model.get_params()

    {'C': 1.0,
     'class_weight': None,
     'dual': False,
     'fit_intercept': True,
     'intercept_scaling': 1,
     'l1_ratio': None,
     'max_iter': 100,
     'multi_class': 'auto',
     'n_jobs': None,
     'penalty': 'l2',
     'random_state': None,
     'solver': 'lbfgs',
     'tol': 0.0001,
     'verbose': 0,
     'warm_start': False}

    parameters_lr = {
                    'C': [0.01, 0.1, 1.0, 10.0, 100.0],
                    # 'class_weight': ['balanced', 'balanced_subsample', None]
                    # 'dual': [True, False],
                    # 'fit_intercept': [True, False],
                    # 'intercept_scaling': [0.01, 0.1, 1.0, 10.0, 100.0],
                    # 'l1_ratio': [0, 1, None],
                    'max_iter': [10, 20, 30, 40, 50],
                    # 'multi_class': ['auto', 'ovr', 'multinomial'],
                    'n_jobs': [None, -1],
                    # 'penalty': ['l1', 'l2', 'elasticnet', None],
                    # 'random_state': [0, 42, 123, None],
                    # 'solver': ['newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga'],
                    'tol': [0.1, 0.01, 0.001, 0.0001, 0.00001],
                    # 'verbose': [1, 10, 100],
                    # 'warm_start': [True, False]
                    }
```

C. Result and Discussion

3.1. Accuracy Score

After splitting the train data and test data, we will search for the score accuracy of each model with the coding that has been made before with the number of test data, namely 300.

Table 5. Accuracy score each classification model

Accuracy Score Before Tuning

Model 1 : K-Nearest Neighbor

```
Classification Report for K Nearest Neighbor

              precision    recall  f1-score   support

           0       0.82      0.26      0.40       102
           1       0.72      0.97      0.83       198

    accuracy                           0.73       300
   macro avg       0.77      0.62      0.61       300
weighted avg       0.75      0.73      0.68       300

--------------------------------------------------

[ ] knn_score = knn_model.score(X_test, Y_test)
    knn_score

0.73
```

Model 2 : Decision Tree

```
Classification Report for Decision Tree

              precision    recall  f1-score   support

           0       0.74      0.74      0.74       102
           1       0.86      0.87      0.87       198

    accuracy                           0.82       300
   macro avg       0.80      0.80      0.80       300
weighted avg       0.82      0.82      0.82       300

--------------------------------------------------

[ ] dct_score = dct_model.score(X_test, Y_test)
    dct_score

0.8233333333333334
```

Model 3 : Random Forest

```
Classification Report for Random Forest

              precision    recall  f1-score   support

           0       0.80      0.76      0.78       102
           1       0.88      0.90      0.89       198

    accuracy                           0.86       300
   macro avg       0.84      0.83      0.84       300
weighted avg       0.86      0.86      0.86       300

--------------------------------------------------

rdf_score = rdf_model.score(X_test, Y_test)
rdf_score

0.8566666666666667
```

Model 4 : Logistic Regression

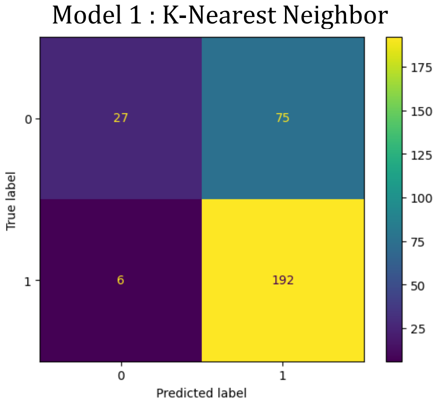
```
Classification Report for Logistic Regression

              precision    recall  f1-score   support

           0       0.84      0.72      0.77       102
           1       0.86      0.93      0.90       198

    accuracy                           0.86       300
   macro avg       0.85      0.82      0.83       300
weighted avg       0.86      0.86      0.85       300

--------------------------------------------------

[ ] lr_score = lr_model.score(X_test, Y_test)
    lr_score

0.8566666666666667
```

As we can see that testing on test data generate accuracy score for K-Nearest Neighbor 0.73, Decision tree 0.823, Random forest 0.856, Logistic Regression 0.856. Proved that distance beetwen each model is not much different. Here is the visualization for Confusion Matrix for each model in Table 5.

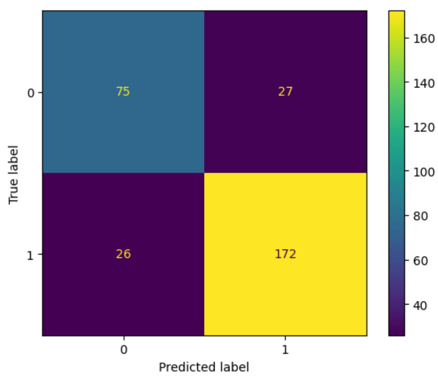
Table 6. Confusion Matrix for each model

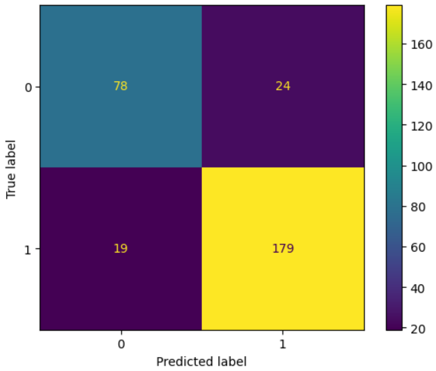
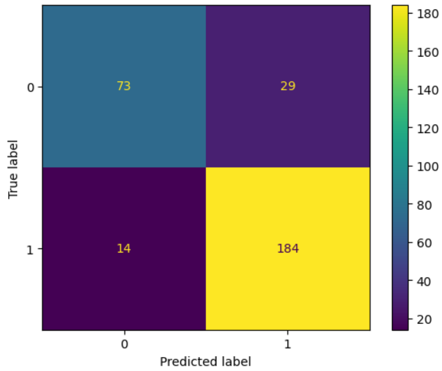
Confusion Matrix

Model 1 : K-Nearest Neighbor



Model 2 : Decision Tree



Model 3 : Random Forest	Model 4 : Logistic Regression
	

3.2. Cross Validation Score

For cross validation score for each model, we use k-fold for n_splits 3,4,5 fold. Within that fold we will use the highest score beetwen three fold.

Table 7. Cross validation score for each model

Result of Cross Validation

Model 1 : K-Nearest Neighbor

```
Average CV Score for n_splits = 3 :  0.7513333333333333
Average CV Score for n_splits = 4 :  0.7576666666666667
Average CV Score for n_splits = 5 :  0.7316666666666667
```

Model 2 : Decision Tree

```
Average CV Score for n_splits = 3 :  0.81
Average CV Score for n_splits = 4 :  0.8046666666666666
Average CV Score for n_splits = 5 :  0.8110000000000002
```

Model 3 : Random Forest

```
Average CV Score for n_splits = 3 :  0.839
Average CV Score for n_splits = 4 :  0.8433333333333334
Average CV Score for n_splits = 5 :  0.842
```

Model 4 : Logistic Regression

```
Average CV Score for n_splits = 3 :  0.8569999999999999
Average CV Score for n_splits = 4 :  0.8553333333333334
Average CV Score for n_splits = 5 :  0.851
```

3.3. Hyperparameter Tunning

After we input the best model that already searched with get_params, then we use gridsearch alongside the highest cv score for each model.

Table 8. Gridsearch best model for each classification

Using Best Model

Model 1 : K-Nearest Neighbor

```
grid = GridSearchCV(estimator = knn_model, param_grid = parameters_knn, cv=4)

best_model_knn = grid.fit(X_train, Y_train)

best_model_knn.best_params_

{'metric': 'euclidean', 'n_neighbors': 3, 'weights': 'uniform'}

model_knn_new = KNeighborsClassifier(metric='euclidean', n_neighbors=3, weights='uniform')
knn_model_new = model_knn_new.fit(X_train, Y_train)
```

Model 2 : Decision Tree

```
grid = GridSearchCV(estimator = dct_model, param_grid = parameters_dct, cv=5)

best_model_dct = grid.fit(X_train, Y_train)

best_model_dct.best_params_

{'criterion': 'gini',
 'max_depth': None,
 'max_features': 1.0,
 'max_leaf_nodes': None}

model_dct_new = DecisionTreeClassifier(criterion='gini', max_depth=None, max_features=1.0, max_leaf_nodes=None)
dct_model_new = model_dct_new.fit(X_train, Y_train)
```

Model 3 : Random Forest

```
grid = GridSearchCV(estimator = rdf_model, param_grid = parameters_rdf, cv=4)

best_model_rdf = grid.fit(X_train, Y_train)

best_model_rdf.best_params_

{'criterion': 'log_loss', 'max_features': 'log2', 'n_estimators': 50}

model_rdf_new = RandomForestClassifier(criterion='log_loss', max_features='log2', n_estimators=50)
rdf_model_new = model_rdf_new.fit(X_train, Y_train)
```

Model 4 : Logistic Regression

```
grid = GridSearchCV(estimator = lr_model, param_grid = parameters_lr, cv=3)

best_model_lr = grid.fit(X_train, Y_train)

best_model_lr.best_params_

{'C': 10.0, 'max_iter': 10, 'n_jobs': None, 'tol': 0.1}

model_lr_new = LogisticRegression(C=10.0, max_iter=10, n_jobs=None, tol=0.1)
lr_model_new = model_lr_new.fit(X_train, Y_train)
```

3.3.1 Best Score for Each Model

After getting the best model for each classification model, we will re-input the accuracy score using the new model from gridsearch to see if the accuracy will increase or not.

Table 9. New score after hyperparameter tuning

New Accuracy Score

Model 1 : K-Nearest Neighbor

```
y_pred_knn_new = knn_model_new.predict(X_test)
print('Classification Report for K Nearest Neighbor')
print(' ')
print(classification_report(Y_test, y_pred_knn_new))
print('-'*50)

Classification Report for K Nearest Neighbor

              precision    recall  f1-score   support

           0       0.81      0.42      0.55       102
           1       0.76      0.95      0.84       198

    accuracy                           0.77       300
   macro avg       0.79      0.69      0.70       300
weighted avg       0.78      0.77      0.75       300

--------------------------------------------------

[ ] knn_new_score = knn_model_new.score(X_test, Y_test)
    knn_new_score

0.77
```

Model 2 : Decision Tree

```
y_pred_dct_new = dct_model_new.predict(X_test)
print('Classification Report for Decision Tree')
print(' ')
print(classification_report(Y_test, y_pred_dct_new))
print('-'*50)

Classification Report for Decision Tree

              precision    recall  f1-score   support

           0       0.74      0.75      0.75       102
           1       0.87      0.86      0.87       198

    accuracy                           0.83       300
   macro avg       0.81      0.81      0.81       300
weighted avg       0.83      0.83      0.83       300

--------------------------------------------------

[ ] dct_new_score = dct_model_new.score(X_test, Y_test)
    dct_new_score

0.8266666666666667
```

Model 3 : Random Forest

```
y_pred_rdf_new = rdf_model_new.predict(X_test)
print('Classification Report for Random Forest')
print(' ')
print(classification_report(Y_test, y_pred_rdf_new))
print('-'*50)

Classification Report for Random Forest

              precision    recall  f1-score   support

           0       0.82      0.78      0.80       102
           1       0.89      0.91      0.90       198

    accuracy                           0.87       300
   macro avg       0.86      0.85      0.85       300
weighted avg       0.87      0.87      0.87       300

--------------------------------------------------

[ ] rdf_new_score = rdf_model_new.score(X_test, Y_test)
    rdf_new_score

0.87
```

Model 4 : Logistic Regression

```
y_pred_lr_new = lr_model_new.predict(X_test)
print('Classification Report for Logistic Regression')
print(' ')
print(classification_report(Y_test, y_pred_lr_new))
print('-'*50)

Classification Report for Logistic Regression

              precision    recall  f1-score   support

           0       0.84      0.75      0.79       102
           1       0.88      0.93      0.90       198

    accuracy                           0.87       300
   macro avg       0.86      0.84      0.85       300
weighted avg       0.87      0.87      0.86       300

--------------------------------------------------

[ ] lr_new_score = lr_model_new.score(X_test, Y_test)
    lr_new_score

0.8666666666666667
```

3.3.2 Comparison

```
===================================Comparison Model===================================

--------------------------------------------------------------------------------------
Accuracy model K Nearest Neighbor
1. Baseline Model                                             : 0.73
2. Modeling 2 (After Hyperparameter Tuning)                   : 0.77
--------------------------------------------------------------------------------------

Accuracy model Decision Tree
1. Baseline Model                                             : 0.8233333333333334
2. Modeling 2 (After Hyperparameter Tuning)                   : 0.8266666666666667
--------------------------------------------------------------------------------------

Accuracy model Random Forest
1. Baseline Model                                             : 0.8566666666666667
2. Modeling 2 (After Hyperparameter Tuning)                   : 0.87
--------------------------------------------------------------------------------------

Accuracy model Logistic Regression
1. Baseline Model                                             : 0.8566666666666667
2. Modeling 2 (After Hyperparameter Tuning)                   : 0.8666666666666667
--------------------------------------------------------------------------------------
```

Figure 22. Comparison beetwen modelling after hyperparameter tuning

D. Conclusion

After conducting research on "Sentiment Analysis Performance Value Optimization Using Hyperparamater Tunning With Grid Search on Shopee App Reviews", it can be concluded that in sentiment analysis of shopee application reviews on Google Play Store from 3000 data that has been crawled, as much as 2015 user data is positive or around 67.2% of the data gives 4-star or 5-star reviews and the remaining 32.8% or 985 data is negative when we make calssification for each modelling, for the data used in this study 300 data used for test data after being split by a ratio of 90:10 will be used for each classification model such as K-Nearest Neighbor, Decision Tree, Random Forest, and Logistic Regression. But with hyperparameter tuning using gridsearch we can see the result of each accuracy score of KNN, DCT, RF, and LR is increasing from 0.73 to 0.77, 0.823 to 0.826, 0.856 to 0.87, and 0.856 to 0.866. This indicated that among the machine learning model that had been tuning using gridsearch, KNN is the one that highly increased.

E. References

[1] R. Wahyudi *et al.*, "Analisis Sentimen pada review Aplikasi Grab di Google Play Store Menggunakan Support Vector Machine," *JURNAL INFORMATIKA*, vol. 8, no. 2, 2021, [Online]. Available: http://ejournal.bsi.ac.id/ejurnal/index.php/ji

[2] R. V. Karthik and S. Ganapathy, "A fuzzy recommendation system for predicting the customers interests using sentiment analysis and ontology in e-commerce," *Appl Soft Comput*, vol. 108, Sep. 2021, doi: 10.1016/j.asoc.2021.107396.

[3] E. Turban, J. Whiteside, D. King, and J. Outland, "Introduction to Electronic Commerce and Social Commerce." [Online]. Available: http://www.springer.com/series/10099

[4] *Unpacking E-commerce*. OECD Publishing, 2019.

[5] A. Salsabila Putri and R. Zakaria, "Analisis Pemetaan E-Commerce Terbesar Di Indonesia Berdasarkan Model Kekuatan Ekonomi Digital," 2020.

[6] T. R. iPrice, "[ANALISIS] Kilas Balik Persaingan E-Commerce Indonesia Tahun 2017," *iPrice*, Dec. 21, 2017. https://iprice.co.id/trend/insights/kilas-balik-e-commerce-indonesia-2017/ (accessed Aug. 19, 2023).

[7] N. Herlinawati *et al.*, "Analisis Sentimen Zoom Cloud Meetings Di Play Store Menggunakan Naïve Bayes Dan Support Vector Machine," 2020.

[8] M. Rezki, D. N. Kholifah, M. Faisal, R. Suryadithia, U. Bina, and S. Informatika, "Analisis Review Pengguna Google Meet dan Zoom Cloud Meeting Menggunakan Algoritma Naïve Bayes." [Online]. Available: http://ejournal.bsi.ac.id/ejurnal/index.php/infortech264

[9] S. George and B. Sumathi, "Grid Search Tuning of Hyperparameters in Random Forest Classifier for Customer Feedback Sentiment Prediction," 2020. [Online]. Available: www.ijacsa.thesai.org

[10] S. Muhammad Isa, R. Suwandi, and Y. Pricilia Andrean, "Optimizing the Hyperparameter of Feature Extraction and Machine Learning Classification Algorithms," 2019. [Online]. Available: www.ijacsa.thesai.org

[11] C. Cahyaningtyas, Y. Nataliani, and I. R. Widiasari, "Analisis sentimen pada rating aplikasi Shopee menggunakan metode Decision Tree berbasis SMOTE," *AITI: Jurnal Teknologi Informasi*, vol. 18, no. Agustus, pp. 173–184, 2021.

[12] Z. Rais, R. N. Said, and R. Ruliana, "Text Classification on Sentiment Analysis of Marketplace SHOPEE Reviews On Twitter Using K-Nearest Neighbor (KNN) Method," *JINAV: Journal of Information and Visualization*, vol. 3, no. 1, pp. 1–8, Jul. 2022, doi: 10.35877/454ri.jinav1389.

[13] R. Kosasih and A. Alberto, "Sentiment analysis of game product on shopee using the TF-IDF method and naive bayes classifier," *ILKOM Jurnal Ilmiah*, vol. 13, no. 2, pp. 101–109, Aug. 2021, doi: 10.33096/ilkom.v13i2.721.101-109.

[14] M. E. Purbaya, D. P. Rakhmadani, Maliana Puspa Arum, and Luthfi Zian Nasifah, "Implementation of n-gram Methodology to Analyze Sentiment Reviews for Indonesian Chips Purchases in Shopee E-Marketplace," *Jurnal RESTI (Rekayasa Sistem dan Teknologi Informasi)*, vol. 7, no. 3, pp. 609–617, Jun. 2023, doi: 10.29207/resti.v7i3.4726.

[15] T. A. Assegie, "An optimized K-Nearest neighbor based breast cancer detection," *Journal of Robotics and Control (JRC)*, vol. 2, no. 3, pp. 115–118, May 2021, doi: 10.18196/jrc.2363.

[16] M. I. Gunawan, "JEPIN (Jurnal Edukasi dan Penelitian Informatika) Peningkatan Kinerja Akurasi Prediksi Penyakit Diabetes Mellitus Menggunakan Metode Grid Seacrh pada Algoritma Logistic Regression," 2020.

[17] K. Chong and N. Shah, "Comparison of Naive Bayes and SVM Classification in Grid-Search Hyperparameter Tuned and Non-Hyperparameter Tuned Healthcare Stock Market Sentiment Analysis." [Online]. Available: www.ijacsa.thesai.org

[18] J. Khatib Sulaiman Dalam No, I. Firman Ashari, M. Daffa, S. Ali, and I. Teknologi Sumatera, "Sentiment Analysis of Tweets About Allowing Outdoor Mask Wear Using Naïve Bayes and TextBlob," *Indonesian Journal of Computer Science Attribution*, vol. 12, no. 3, pp. 2023–1092, 2023.

[19] R. Apriani *et al.*, "Analisis Sentimen Dengan Naïve Bayes Terhadap Komentar Aplikasi Tokopedia," 2019.

[20] F. Karabiber, "TF-IDF Term Frequency-Inverse Document Frequency," *LearnDataSci*, Apr. 06, 2021. https://www.learndatasci.com/glossary/tf-idf-term-frequency-inverse-document-frequency/ (accessed Aug. 20, 2023).

[21] W. Wijanarto and S. P. Brilianti, "Peningkatan Performa Analisis Sentimen Dengan Resampling dan Hyperparameter pada Ulasan Aplikasi BNI Mobile," *Jurnal Eksplora Informatika*, vol. 9, no. 2, pp. 140–153, Mar. 2020, doi: 10.30864/eksplora.v9i2.333.

[22] M. M. RAMADHAN, I. S. SITANGGANG, F. R. NASUTION, and A. GHIFARI, "Parameter Tuning in Random Forest Based on Grid Search Method for Gender Classification Based on Voice Frequency," *DEStech Transactions on Computer Science and Engineering*, no. cece, Oct. 2017, doi: 10.12783/dtcse/cece2017/14611.

[23] E. Kabir Hashi and M. Shahid Uz Zaman, "Developing a Hyperparameter Tuning Based Machine Learning Approach of Heart Disease Prediction," *Journal of Applied Science & Process Engineering*, vol. 7, no. 2, 2020.

[24] M. Alhamid, "What is Cross-Validation?," *Towards Data Science*, Dec. 25, 2020. https://towardsdatascience.com/what-is-cross-validation-60c01f9d9e75 (accessed Aug. 19, 2023).

[25] I. Priyadarshini and C. Cotton, "A novel LSTM–CNN–grid search-based deep neural network for sentiment analysis," *Journal of Supercomputing*, vol. 77, no. 12, pp. 13911–13932, Dec. 2021, doi: 10.1007/s11227-021-03838-w.

[26] G. T. Reddy *et al.*, "An Ensemble based Machine Learning model for Diabetic Retinopathy Classification," in *International Conference on Emerging Trends in Information Technology and Engineering, ic-ETITE 2020*, Institute of Electrical and Electronics Engineers Inc., Feb. 2020. doi: 10.1109/ic-ETITE47903.2020.235.