

Natural Language Processing with Disaster Tweets

Case Study 6 Text Mining

Group 1:



Hutari Andini
2206820680



Galank Zamer
2206051393



Rachelle Melody
d'Lyra Soentara
2206051456



Kartika Rizkia
Zuhrah
2206027993



Luthfi Athallah
2206826980

Context of the Case Study

kaggle



Natural Language Processing with Disaster Tweets

Text
Classification

Prediction

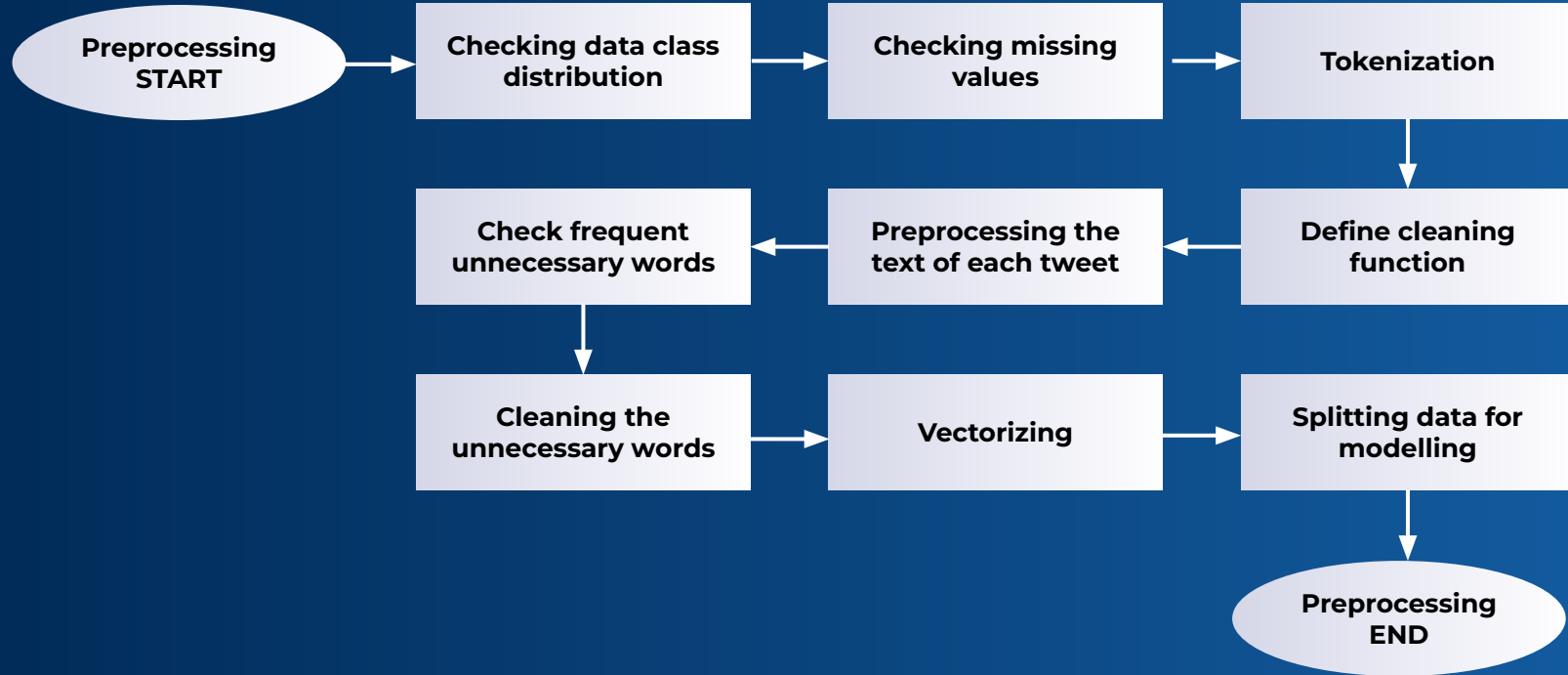
Evaluation

We are challenged to build a machine learning model that predicts which Tweets are about real disasters and which one's aren't. We have a dataset of 10,000 tweets that were hand classified.

<https://twitter.com/AnyOtherAnnaK/status/629195955506708480>

Data Preprocessing

Preparing data to make it ready to analyze



Some highlight in Data Preprocessing

Tokenization, Cleaning, and Vectorizing

1

Tokenization

```
# fungsi clean words
def clean_words(text, extra_stops = None):
    text = re.sub(r'(http.*)|@', '', text)
    tokens = word_tokenize(text)
    stops = stopwords.words('english') + (extra_stops if extra_stops else [])
    clean_tokens = [token.lower() for token in tokens if token.lower() not in stops and token.isalnum()]
    return ' '.join(clean_tokens)
```

Pada function clean_words, sudah terdapat cleaning text berupa penghilangan link, stopwords, dan juga tokenisasi tweet yang berupa pemenggalan kalimat di twitter menjadi per kata.

2

Cleaning

```
# Mengubah ke huruf kecil (lowercase) dan menghapus tanda baca, karakter aneh, dan strip
def preprocess(text):
    text = text.lower() #lowercase text
    text = text.strip() #Menghapus leading/trailing whitespace
    text = re.sub('@(\w+)', 'atUser', text) #mengubah @user menjadi atUser
    text = re.sub('#(\w+)', 'r'\1', text) #menghapus hashtag di depan suatu kata
    text = re.compile('<.*>').sub('', text) #menghapus HTML tags/markups
    text = re.compile('[%s]' % re.escape(string.punctuation)).sub(' ', text) #Replace punctuation with space. Careful
    text = re.sub('\s+', ' ', text) #Menghapus extra space dan tabs
    text = re.sub('110-01531' ' ' text) #110-01 matches and digit 10 to 10000
```

Tahapan cleaning berupa pengubahan karakter-karakter yang tidak memiliki informasi, menghilangkan punctuation, menyeragamkan kapitalisasi pada text, dan juga menghilangkan spasi berupa white space.

3

Vectorizing

```
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import f1_score

vectorizer = CountVectorizer()

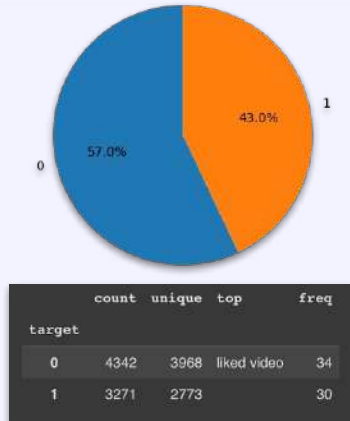
vectors = vectorizer.fit_transform(df_train['clean_text'])
test_vectors = vectorizer.transform(df_test['clean_text'])
```

Dengan menggunakan CountVectorizer function pada sklearn, dataset akan di vektorisasi setelah dilakukan *cleaning*. Dataset yang akan di fit menggunakan vektorisasi ini adalah pada df_train maupun df_test.

Sneak peaks clean_text!

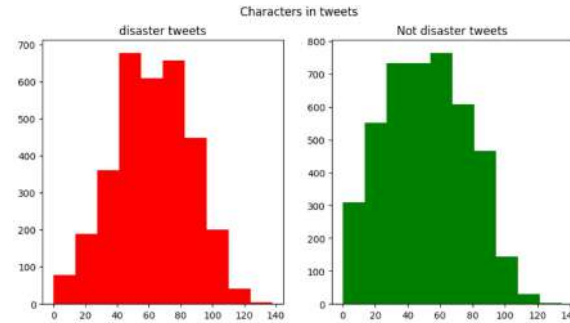
	id	text	target	clean_text
0	1	Our Deeds are the Reason of this #earthquake M...	1	deeds reason earthquake may allah forgive us
1	4	Forest fire near La Ronge Sask. Canada	1	forest fire near la ronge sask canada

1 Distribution of Target Variable



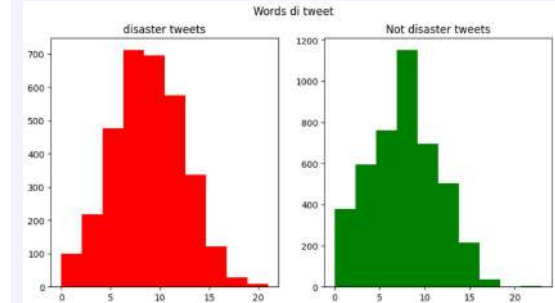
- Jumlah tweets di non disaster lebih banyak daripada tweets disaster
- Jumlah **unique < count** artinya ada beberapa repetisi di text dari tweets
- Kosong di 'top' pada disaster artinya kemungkinan tidak ada single phrase yang dominan

2 Characters di tweets



- Distribusi dari jumlah Characters **Disaster dan non Disaster tweets mirip**
- Secara keseluruhan, **non Disaster memiliki character lebih banyak**
- Nilai terbanyaknya berpusat di **approximately 40-70 characters**

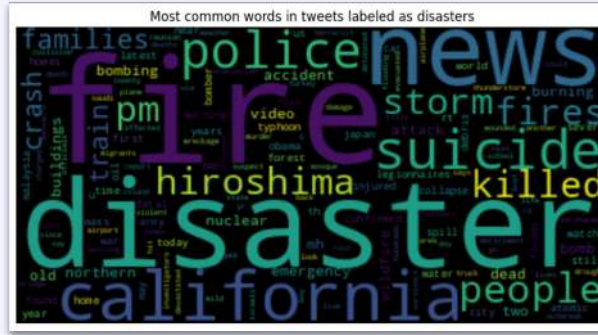
3 Words di tweets



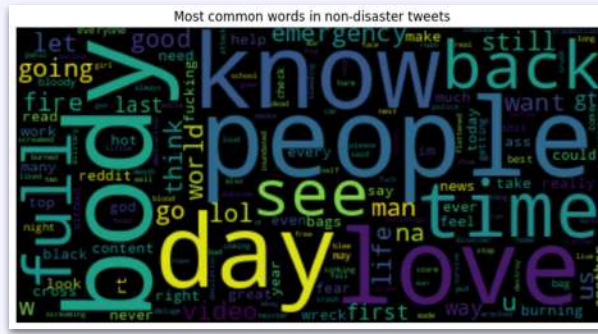
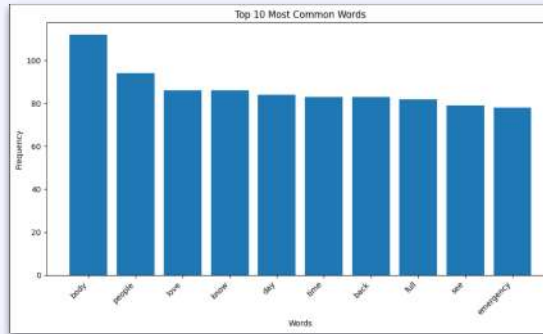
- Distribusi dari jumlah words *Disaster* secara kasar **mendekati normal**
- Sedangkan *non Disaster tweets* memiliki **frekuensi ekstrim di sekitar 7-9 words**
- Secara general, jumlah words *non Disaster* **lebih banyak daripada Disaster**

Top 10 Most Common Words

Words	Frequency
the	175
of	115
and	110
a	108
to	105
in	102
that	98
on	88
it	88
you	85



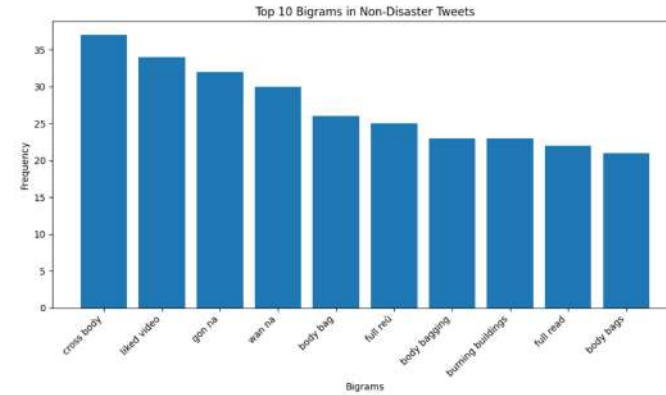
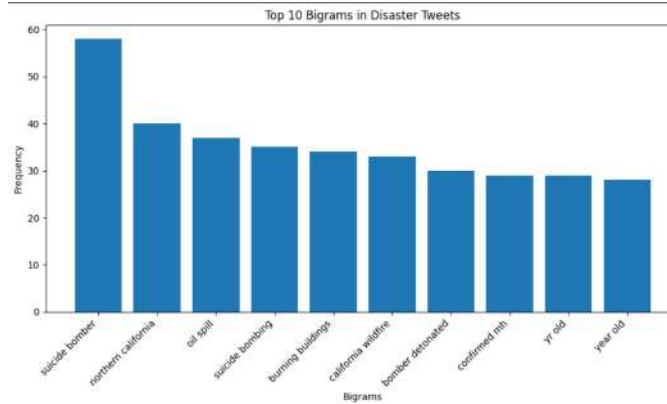
- *fire*
- *disaster*
- *news*
- *california*
- *suicide*



- *body*
- *people*
- *love*
- *know*
- *day*

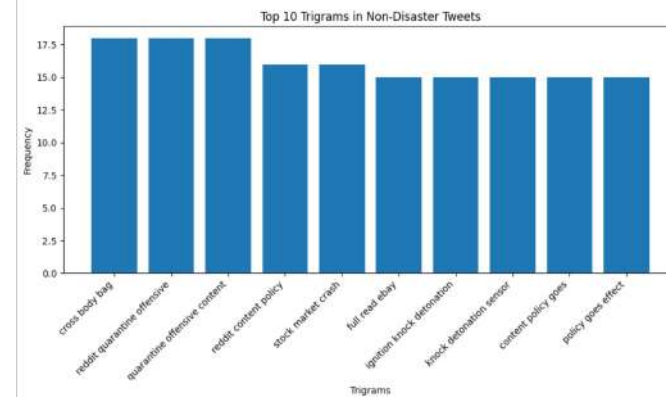
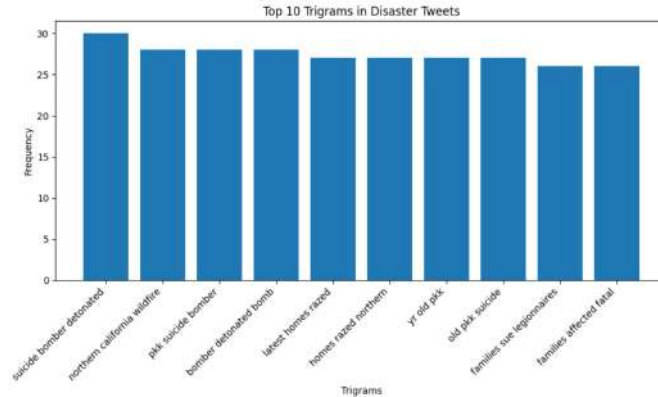
1

Bigrams

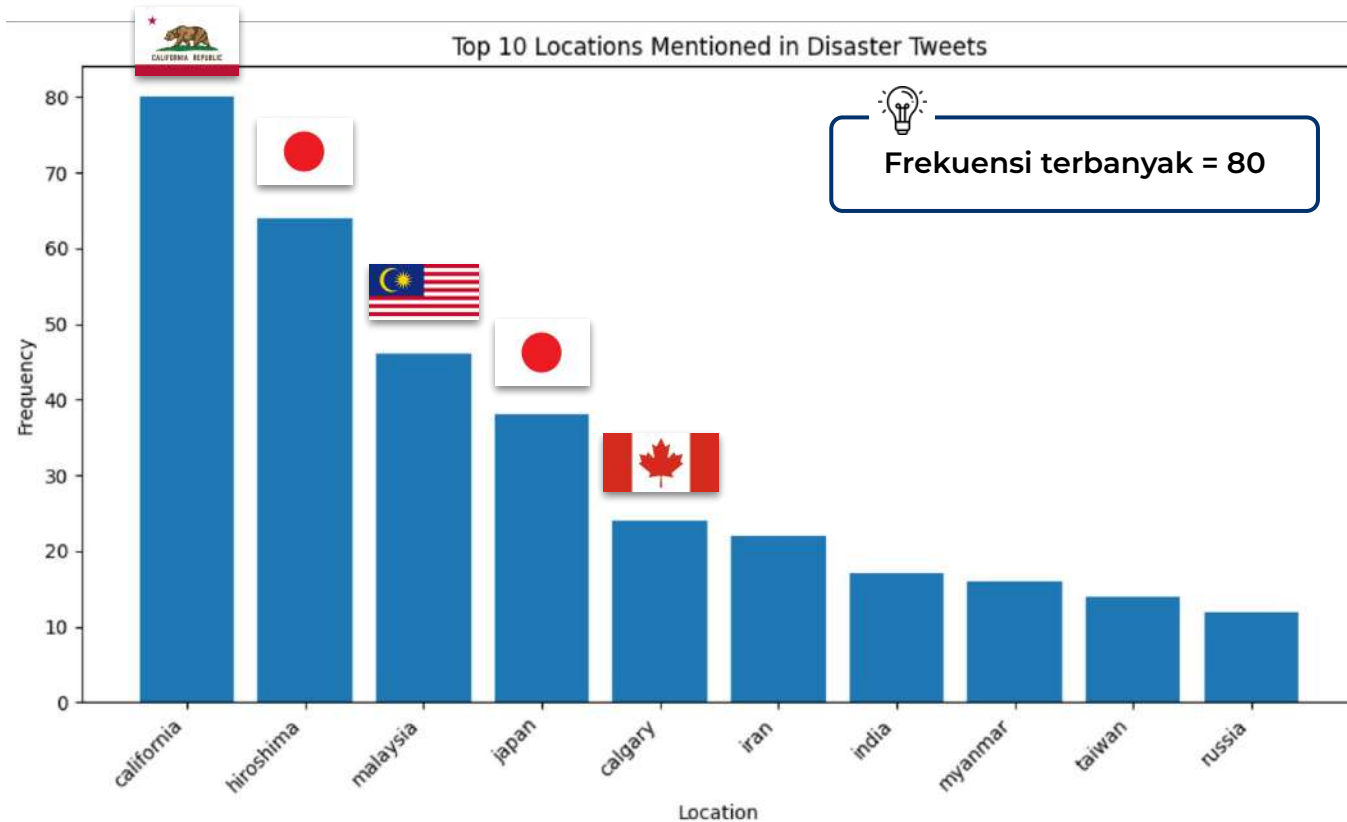


2

Trigrams



Top 10 Locations Mentioned in Disaster Tweets



Pada bagian ini data akan dilatih dan selanjutnya diuji. Pertama-tama, data dibagi menjadi data train dan data test. Kemudian akan dilakukan pemodelan, sebagai berikut:

Random Forest

	Precision	Recall	F1-Score	Support
0	78%	84%	81%	874
1	76%	68%	72%	649
Accuracy			77%	1523
Macro Avg	77%	76%	76%	1523
Weighted Avg	77%	77%	77%	1523

Pada bagian ini data akan dilatih dan selanjutnya diuji. Pertama-tama, data dibagi menjadi data train dan data test. Kemudian akan dilakukan pemodelan, sebagai berikut:

Support Vector Machine (SVM)

	Precision	Recall	F1-Score	Support
0	79%	91%	84%	874
1	84%	67%	74%	649
Accuracy			80%	1523
Macro Avg	81%	79%	79%	1523
Weighted Avg	81%	80%	80%	1523

Pada bagian ini data akan dilatih dan selanjutnya diuji. Pertama-tama, data dibagi menjadi data train dan data test. Kemudian akan dilakukan pemodelan, sebagai berikut:

Logistic Regression

	Precision	Recall	F1-Score	Support
0	80%	86%	83%	874
1	79%	72%	75%	649
Accuracy			80%	1523
Macro Avg	80%	79%	79%	1523
Weighted Avg	80%	80%	80%	1523

Pada bagian ini data akan dilatih dan selanjutnya diuji. Pertama-tama, data dibagi menjadi data train dan data test. Kemudian akan dilakukan pemodelan, sebagai berikut:

Gradient Boosting Classifier

	Precision	Recall	F1-Score	Support
0	71%	93%	81%	874
1	84%	50%	63%	649
Accuracy			75%	1523
Macro Avg	78%	71%	72%	1523
Weighted Avg	77%	75%	73%	1523

Pada bagian ini data akan dilatih dan selanjutnya diuji. Pertama-tama, data dibagi menjadi data train dan data test. Kemudian akan dilakukan pemodelan, sebagai berikut:

XGBoost

	Precision	Recall	F1-Score	Support
0	76%	90%	82%	874
1	82%	61%	70%	649
Accuracy			78%	1523
Macro Avg	79%	76%	76%	1523
Weighted Avg	79%	78%	77%	1523

Pada bagian ini data akan dilatih dan selanjutnya diuji. Pertama-tama, data dibagi menjadi data train dan data test. Kemudian akan dilakukan pemodelan, sebagai berikut:

Naive Bayes

	Precision	Recall	F1-Score	Support
0	80%	83%	82%	874
1	76%	72%	74%	649
Accuracy			78%	1523
Macro Avg	78%	78%	78%	1523
Weighted Avg	78%	78%	78%	1523

Pada bagian ini data akan dilatih dan selanjutnya diuji. Pertama-tama, data dibagi menjadi data train dan data test. Kemudian akan dilakukan pemodelan, sebagai berikut:

K Nearest Neighbor (KKN)

	Precision	Recall	F1-Score	Support
0	68%	66%	67%	874
1	56%	59%	57%	649
Accuracy			63%	1523
Macro Avg	62%	62%	62%	1523
Weighted Avg	63%	63%	63%	1523

Pada bagian ini data akan dilatih dan selanjutnya diuji. Pertama-tama, data dibagi menjadi data train dan data test. Kemudian akan dilakukan pemodelan, sebagai berikut:

Decision Tree

	Precision	Recall	F1-Score	Support
0	76%	81%	78%	874
1	72%	66%	69%	649
Accuracy			74%	1523
Macro Avg	74%	73%	74%	1523
Weighted Avg	74%	74%	74%	1523

Model Evaluation

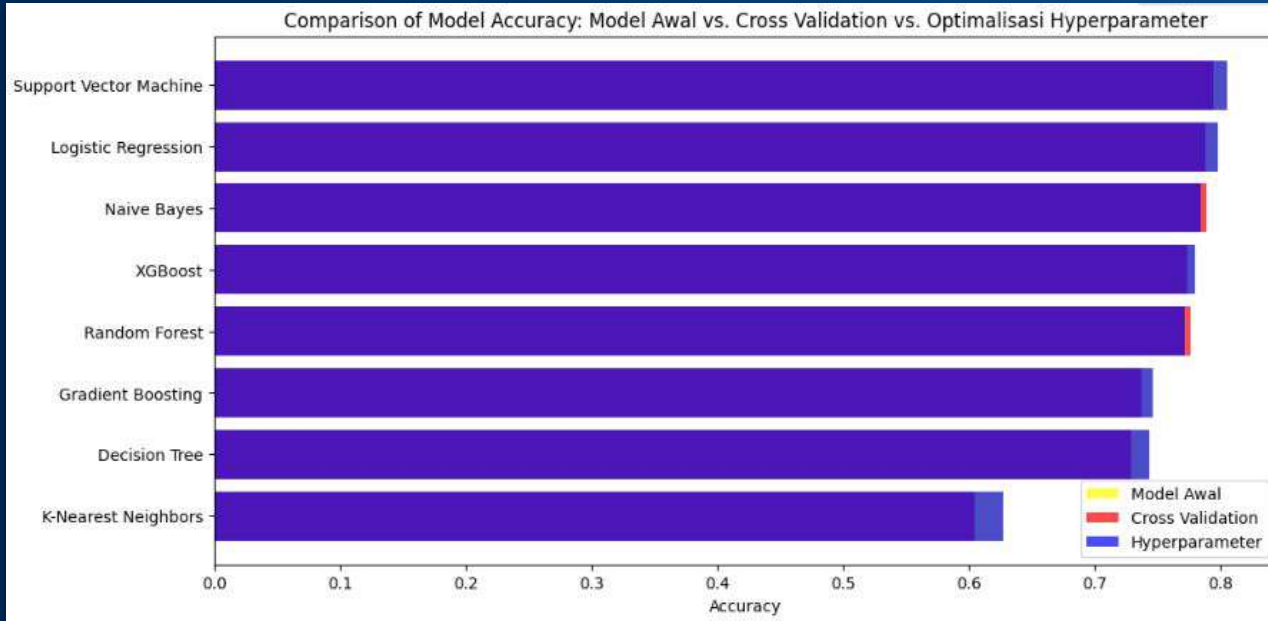


Tabel Nilai Akurasi Model Awal, Model dengan Cross-Validation, dan Model dengan Hyperparameter Tuning

Nama Model	Model Awal	Model dengan Cross-Validation (k-fold=5)	Model dengan Hyperparameter Tuning
Support Vector Machine (SVM)	80.5%	79.5%	80.5%
Logistic Regression	79.8%	78.8%	79.8%
Naive Bayes	78.5%	78.9%	78.5%
XGBoost	78.0%	77.4%	78.0%
Random Forest	77.2%	77.6%	77.2%
Gradient Boosting	74.7%	73.7%	74.7%
Decision Tree	74.4%	72.9%	74.4%
K-Nearest Neighbor	62.7%	60.4%	62.7%

Model Evaluation


Grafik Perbandingan Nilai Akurasi Model Awal, Model dengan Cross-Validation, dan Model dengan Hyperparameter Tuning



Sebagian besar model (kecuali Naive Bayes dan Random Forest) memiliki nilai akurasi model dengan Hyperparameter Tuning yang **lebih besar** dibandingkan model dengan Cross-Validation (k-fold=5).

Model Evaluation

Interpretasi Perbandingan Nilai Akurasi Model Awal, Model dengan Cross-Validation, dan Model dengan Hyperparameter Tuning



Dapat dilihat bahwa nilai akurasi model **Support Vector Machine (SVM)** menjadi yang tertinggi untuk semua evaluasi.



Artinya, model SVM dapat **memprediksi label dengan benar sebanyak 80.5%** dari sampel.

Membentuk model ensemble dari tiga model terbaik

```
# Membuat Voting Classifier (menggunakan hard voting, di mana model memilih kelas mayoritas)
ensemble_model = VotingClassifier(
    estimators=[('svm', svm_model), ('logreg', logreg_model), ('nb', nb_model)],
    voting='hard' # Bisa menggunakan 'soft' untuk probabilistic voting jika model mendukung
)

# Melatih ensemble model
ensemble_model.fit(X_train, y_train)

# Prediksi dengan ensemble model
y_pred_ensemble = ensemble_model.predict(X_test)

# Menghitung akurasi
accuracy = accuracy_score(y_test, y_pred_ensemble)

# Menampilkan hasil evaluasi
print(f"Ensemble Model Accuracy: {accuracy}")

Ensemble Model Accuracy: 0.8030203545633617
```

Ensemble ketiga model terbaik, nyatanya tidak memberikan nilai akurasi yang lebih baik dari model SVM dengan hyperparameter tuning.

Prediction

Kita memprediksi data test dengan model terbaik yaitu model Support Vector Machine (SVM) dengan optimalisasi hyperparameter

```
[4] # Menggunakan model SVM terbaik setelah hyperparameter tuning
    optimal_svm_model = grid_search_svm.best_estimator_

    # Melakukan prediksi pada df_test
    df_test['target'] = optimal_svm_model.predict(df_test)

    # Menyimpan hasil prediksi dalam format yang sesuai untuk Kaggle
    Submission1 = df_test[['id', 'target']]

    # Menyimpan ke file CSV
    Submission1.to_csv('Submission1.csv', index=False)
```

474

Team1_DMBIUI24



0.79803

2

2d



Your Best Entry!

Your most recent submission scored 0.79803, which is an improvement of your previous score of 0.79282. Great job!

Tweet this



Thank You

By Group 1