# UJIAN AKHIR SEMESTER

# DATA SCIENCE B

## BREAST CANCER CLASSIFICATION & OPINION MINING CLASSIFICATION


**Oleh :**

**AXL ADILLA**

**20/466397/PPA/05963**

**PROGRAM MAGISTER ILMU KOMPUTER**

**DEPARTEMEN ILMU KOMPUTER DAN ELEKTRONIKA**

**FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM**

**UNIVERSITAS GADJAH MADA**


**2021**

## 1. BREAST CANCER CLASSIFICATION

a. Permasalahan Breast Cancer Classification adalah merupakan permasalahan penyakit serus yang cukup banyak diriset oleh banyak peneliti dengan berbagai metode (Kajala & Jain, 2020), cukup banyak riset yang menunjukan hasil akurasi yang tinggi yaitu diatas 90%. Berdasarkan data yang diperoleh, data yang didapat adalah fitur dari mammograms dengan 29 fitur (radius_mean, texture_mean, perimeter_mean, area_mean, smoothness_mean, compactness_mean, concavity_mean, concave points_mean, symmetry_mean, fractal_dimension_mean, radius_se, texture_se, perimeter_se, area_se, smoothness_se, compactness_se, concave points_se, symmetry_se, fractal_dimension_se, radius_worst, texture_worst, perimeter_worst, area_worst, smoothness_worst, compactness_worst, concavity_worst, concave points_worst, symmetry_worst, fractal_dimension_worst) dan target berupa kelas M untuk Malignant (Ganas) dan B untuk Benign (Jinak).

b. Berdasarkan paper (Kajala & Jain, 2020; Patgiri et al., 2019) menyatakan metode SVM memiliki akurasi yang baik dalam mengklasifikasikan Breast Cancer dengan data fitur dari mammograms, Kelebihan SVM dibanding algoritma lain seperti random forest, naïve bayes, logistic regression, maupun neural network adalah ketahanan terhadap prediksi (Patgiri et al., 2019) dimana Patgiri menyatakan algoritma seperti random forest, naïve bayes, logistic regression, maupun neural network sangat dipengaruhi data training yang digandakan, beberapa metode mampu mencapai 100% akurasi dengan mengaturan data tersebut dan metode SVM salah satu metode yang tidak mengalami perubahan terlalu signifikan antara data training normal maupun digandakan dibandingkan metode lainnya dengan akurasi dari 92% ke 98%.

c. Preprocessing yang perlu dilakukan pada Dataset adalah menyeimbangkan memeriksa apakah ana data kosong atau null dengan potongan kode :

```
dataset.isna().sum()
```

Diketahui bahwa semua data lengkap maka langkah selanjutnya adalah menyeimbangkan antara kelas Malignant dan Benign dengan :

```
count_class_b, count_class_m = dataset['diagnosis'].value_counts()
```

Dimana menunjukan data Benign sejumlah 357 dan Malignant sejumlah 212, maka dilakukan oversampling pada data Malignant sehingga data seimbang

```
dataset_class_m_over = dataset_class_m.sample(count_class_b, replace=True)
```

menghasilkan Benign sejumlah 357 dan Malignant sejumlah 357. Kemudian dilakukan split data antara data fitur dan data target.

```
dataset_x = dataset.drop('diagnosis', axis=1)
dataset_y = dataset['diagnosis']
```

Dilanjutkan dengan Split antara data training dan data Test

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(dataset_x, dataset_y, random_state=0, test_size=0.7 )
```

Langkah terakhir adalah melakukan encoding terhadap data target / diagnose dengan 1 untuk Malignant dan 0 untuk Benign

```
def diagnosa_to_label(diagnosa):
  if diagnosa == 'M':
        return 1
  else:
        return 0
y_train_label = y_train.apply(diagnosa_to_label)
y_train_label
```

d. Ekstraksi Fitur yang dilakukan adalah melakukan normalisasi data dengan MaxMinScaler untuk setiap fitur agar masing-masing fitur seimbang dan tidak memberikan bobot lebih pada classifier

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
x_train_scaled = scaler.fit_transform(x_train)
```

e. Feature Selection yang digunakan pada tugas ini adalah Wrapper Method secara forward / maju secara berulang-ulang dari memilih 1 fitur hingga menggunakan semua (29) fitur untuk mencari & mendapatkan fitur apa saja yang paling berpengaruh pada klasifikasi Breast Cancer

```
from sklearn.feature_selection import SequentialFeatureSelector
from sklearn.metrics import accuracy_score
from sklearn.svm import SVC
```

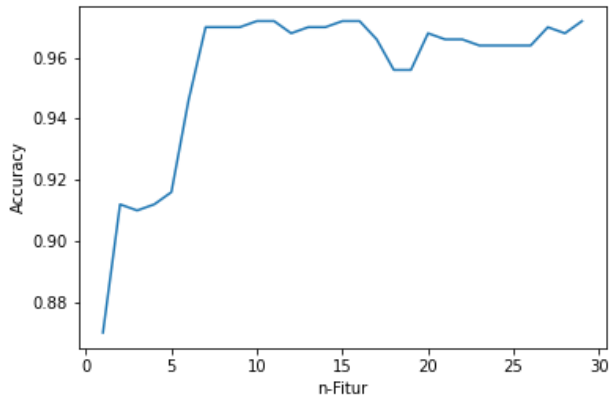
```
clf = SVC()
for n_fitur in range(1, len(x_train.columns)):
    #Fungsi untuk memilih n-fitur dari 29 fitur
     sfs = SequentialFeatureSelector(clf, n_features_to_select=n_fitur, direction='forward')
     sfs.fit(x_train_scaled, y_train_label)
     new_x_train_scaled = sfs.transform(x_train_scaled)
     y_test_label = y_test.apply(diagnosa_to_label)
    x_test_scaled = scaler.transform(x_test)
    new_x_test_scaled = sfs.transform(x_test_scaled)

    clf.fit(new_x_train_scaled, y_train_label)
    y_pred = clf.predict(new_x_test_scaled)
    accuracy_model = accuracy_score(y_test_label, y_pred)
```

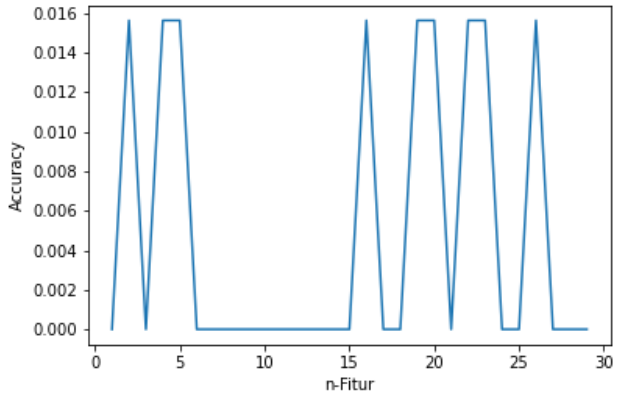
f. Pembangunan model dengan menggunakan SVM, pada penelitian ini pembangunan model dilakukan berulang-ulang untuk mencari bagaimana pengaruh fitur terhadap akurasi prediksi dan waktu pembangunan model dengan

```
from sklearn.feature_selection import SequentialFeatureSelector
from sklearn.metrics import accuracy_score
from sklearn.svm import SVC
import time
clf = SVC()
for n_fitur in range(1, len(x_train.columns)):
    #Fungsi untuk memilih n-fitur dari 29 fitur
    sfs = SequentialFeatureSelector(clf, n_features_to_select=n_fitur, direction='forward')
    sfs.fit(x_train_scaled, y_train_label)
    …
    start_train = time.process_time()
    clf.fit(new_x_train_scaled, y_train_label)
    y_pred = clf.predict(new_x_test_scaled)
    end_train = time.process_time() - start_train
    accuracy_model = accuracy_score(y_test_label, y_pred)
```

g. Evaluasi model yang didapat adalah secara umum sesuai pada Gambar 1 dan Gambar 2 di bawah, dengan detail pada tabel 1 dibawah. Pada Tabel 1. menunjukan kecenderungan makin banyak fitur yang digunakan akurasi makin tinggi namun mencapai stagnansi setelah digunakan 7 fitur yaitu concave points_mean, area_se, smoothness_se, radius_worst, texture_worst, area_worst, smoothness_worst dengan akurasi 97% . Namun dari segi waktu tidak menunjukan suatu pengaruh berarti dan sangat fluktuatif namun tidak signifikan dan dapat diabaikan




Gambar 1 Grafik Akurasi vs N-Fitur




Gambar 2 Grafik Waktu Eksekusi Vs N-Fitur

Untuk Code Breast Cancer Classification Dapat dilihat di https://github.com/AxlAdilla/UAS-DataScience

Tabel 1 Detail Evaluasi Model

| Jumlah Fitur | Akurasi | Time | Fitur yang digunakan |
|---|---|---|---|
| 1 | 0.87 | 0 | area_worst |
| 2 | 0.912 | 0 | area_worst; smoothness_worst |
| 3 | 0.91 | 0 | smoothness_se; area_worst; smoothness_worst |
| 4 | 0.912 | 0 | area_se; smoothness_se; area_worst; smoothness_worst |
| 5 | 0.916 | 0 | area_se; smoothness_se; radius_worst; area_worst; smoothness_worst |
| 6 | 0.946 | 0 | area_se; smoothness_se; radius_worst; texture_worst; area_worst; smoothness_worst |
| 7 | 0.97 | 0.015625 | concave points_mean; area_se; smoothness_se; radius_worst; texture_worst; area_worst; smoothness_worst |
| 8 | 0.97 | 0.015625 | radius_mean; concave points_mean; area_se; smoothness_se; radius_worst; texture_worst; area_worst; smoothness_worst |
| 9 | 0.97 | 0.015625 | radius_mean; concavity_mean; concave points_mean; area_se; smoothness_se; radius_worst; texture_worst; area_worst; smoothness_worst |
| 10 | 0.972 | 0 | radius_mean; concavity_mean; concave points_mean; area_se; smoothness_se; compactness_se; radius_worst; texture_worst; area_worst; smoothness_worst |
| 11 | 0.972 | 0.015625 | radius_mean; texture_mean; concavity_mean; concave points_mean; area_se; smoothness_se; compactness_se; radius_worst; texture_worst; area_worst; smoothness_worst |
| 12 | 0.968 | 0 | radius_mean; texture_mean; concavity_mean; concave points_mean; symmetry_mean; area_se; smoothness_se; compactness_se; radius_worst; texture_worst; area_worst; smoothness_worst |
| 13 | 0.97 | 0 | radius_mean; texture_mean; perimeter_mean; concavity_mean; concave points_mean; symmetry_mean; area_se; smoothness_se; compactness_se; radius_worst; texture_worst; area_worst; smoothness_worst |
| 14 | 0.97 | 0 | radius_mean; texture_mean; perimeter_mean; concavity_mean; concave points_mean; symmetry_mean; texture_se; area_se; smoothness_se; compactness_se; radius_worst; texture_worst; area_worst; smoothness_worst |
| 15 | 0.972 | 0 | radius_mean; texture_mean; perimeter_mean; concavity_mean; concave points_mean; symmetry_mean; texture_se; area_se; smoothness_se; compactness_se; symmetry_se; radius_worst; texture_worst; area_worst; smoothness_worst |
| 16 | 0.972 | 0 | radius_mean; texture_mean; perimeter_mean; area_mean; concavity_mean; concave points_mean; symmetry_mean; texture_se; area_se; smoothness_se; compactness_se; symmetry_se; radius_worst; texture_worst; area_worst; smoothness_worst |
| 17 | 0.966 | 0 | radius_mean; texture_mean; perimeter_mean; area_mean; concavity_mean; concave points_mean; symmetry_mean; radius_se; texture_se; area_se; smoothness_se; compactness_se; symmetry_se; radius_worst; texture_worst; area_worst; smoothness_worst |
| 18 | 0.956 | 0 | radius_mean; texture_mean; perimeter_mean; area_mean; compactness_mean; concavity_mean; concave points_mean; symmetry_mean; radius_se; texture_se; area_se; smoothness_se; compactness_se; symmetry_se; radius_worst; texture_worst; area_worst; smoothness_worst |
| 19 | 0.956 | 0.015625 | radius_mean; texture_mean; perimeter_mean; area_mean; compactness_mean; concavity_mean; concave points_mean; symmetry_mean; radius_se; texture_se; area_se; smoothness_se; compactness_se; symmetry_se; radius_worst; texture_worst; area_worst; smoothness_worst; concave points_worst |
| 20 | 0.968 | 0.015625 | radius_mean; texture_mean; perimeter_mean; area_mean; compactness_mean; concavity_mean; concave points_mean; symmetry_mean; radius_se; texture_se; perimeter_se; area_se; smoothness_se; compactness_se; symmetry_se; radius_worst; texture_worst; area_worst; smoothness_worst; concave points_worst |
| 21 | 0.966 | 0 | radius_mean; texture_mean; perimeter_mean; area_mean; compactness_mean; concavity_mean; concave points_mean; symmetry_mean; radius_se; texture_se; perimeter_se; area_se; smoothness_se; compactness_se; symmetry_se; fractal_dimension_se; radius_worst; texture_worst; area_worst; smoothness_worst; concave points_worst |

| | | | |
|---|---|---|---|
| 22 | 0.966 | 0.015625 | radius_mean; texture_mean; perimeter_mean; area_mean; compactness_mean; concavity_mean; concave points_mean; symmetry_mean; radius_se; texture_se; perimeter_se; area_se; smoothness_se; compactness_se; concave points_se; symmetry_se; fractal_dimension_se; radius_worst; texture_worst; area_worst; smoothness_worst; concave points_worst |
| 23 | 0.964 | 0.015625 | radius_mean; texture_mean; perimeter_mean; area_mean; compactness_mean; concavity_mean; concave points_mean; symmetry_mean; radius_se; texture_se; perimeter_se; area_se; smoothness_se; compactness_se; concave points_se; symmetry_se; fractal_dimension_se; radius_worst; texture_worst; perimeter_worst; area_worst; smoothness_worst; concave points_worst |
| 24 | 0.964 | 0 | radius_mean; texture_mean; perimeter_mean; area_mean; compactness_mean; concavity_mean; concave points_mean; symmetry_mean; radius_se; texture_se; perimeter_se; area_se; smoothness_se; compactness_se; concave points_se; symmetry_se; fractal_dimension_se; radius_worst; texture_worst; perimeter_worst; area_worst; smoothness_worst; concavity_worst; concave points_worst |
| 25 | 0.964 | 0.015625 | radius_mean; texture_mean; perimeter_mean; area_mean; smoothness_mean; compactness_mean; concavity_mean; concave points_mean; symmetry_mean; radius_se; texture_se; perimeter_se; area_se; smoothness_se; compactness_se; concave points_se; symmetry_se; fractal_dimension_se; radius_worst; texture_worst; perimeter_worst; area_worst; smoothness_worst; concavity_worst; concave points_worst |
| 26 | 0.964 | 0 | radius_mean; texture_mean; perimeter_mean; area_mean; smoothness_mean; compactness_mean; concavity_mean; concave points_mean; symmetry_mean; radius_se; texture_se; perimeter_se; area_se; smoothness_se; compactness_se; concave points_se; symmetry_se; fractal_dimension_se; radius_worst; texture_worst; perimeter_worst; area_worst; smoothness_worst; compactness_worst; concavity_worst; concave points_worst |
| 27 | 0.97 | 0 | radius_mean; texture_mean; perimeter_mean; area_mean; smoothness_mean; compactness_mean; concavity_mean; concave points_mean; symmetry_mean; radius_se; texture_se; perimeter_se; area_se; smoothness_se; compactness_se; concave points_se; symmetry_se; fractal_dimension_se; radius_worst; texture_worst; perimeter_worst; area_worst; smoothness_worst; compactness_worst; concavity_worst; concave points_worst; symmetry_worst |
| 28 | 0.968 | 0 | radius_mean; texture_mean; perimeter_mean; area_mean; smoothness_mean; compactness_mean; concavity_mean; concave points_mean; symmetry_mean; radius_se; texture_se; perimeter_se; area_se; smoothness_se; compactness_se; concave points_se; symmetry_se; fractal_dimension_se; radius_worst; texture_worst; perimeter_worst; area_worst; smoothness_worst; compactness_worst; concavity_worst; concave points_worst; symmetry_worst; fractal_dimension_worst |
| 29 | 0.972 | 0 | radius_mean; texture_mean; perimeter_mean; area_mean; smoothness_mean; compactness_mean; concavity_mean; concave points_mean; symmetry_mean; fractal_dimension_mean; radius_se; texture_se; perimeter_se; area_se; smoothness_se; compactness_se; concave points_se; symmetry_se; fractal_dimension_se; radius_worst; texture_worst; perimeter_worst; area_worst; smoothness_worst; compactness_worst; concavity_worst; concave points_worst; symmetry_worst; fractal_dimension_worst |

## 2. OPINION MINING CLASSIFICATION

a. Pada permasalahan Opinion Mining Classification yang diberikan adalah NLP Bahasa Indonesia permasalahan Sentiment Analysis Classification, dengan dataset Twitter tentang tentang gojek.

b. Sudah banyak Teknik yang berkembang pada permasalahan Sentiment Analysis Classification, diantaranya adalah dengan SVM (Lidya et al., 2015) maupun Neural Network (Kim, 2014; Paliwal et al., 2018), pada penelitian ini penulis akan membandingkan keduanya dengan ekstraksi fitur word2vec jenis FastText. Untuk mengklasifikasi sentiment data Twitter Gojek.

c. Langkah Preprocessing Secara umum adalah menyeimbangkan Kelas pada dataset yang tidak seimbang yaitu kelas 0 sejumlah 3062 dan kelas 1 sejumlah 938, dilakukan oversampling pada data kelas 1

```
dataset_class_0 = dataset[dataset['sentimen'] == 0]
dataset_class_1 = dataset[dataset['sentimen'] == 1]
dataset_class_1_over = dataset_class_1.sample(3062, replace=True)
dataset = pd.concat([dataset_class_0, dataset_class_1_over])
dataset['sentimen'].value_counts()
```

Dilakukan Split data antara data test dan train

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(dataset_x, dataset_y, random_state=0, test_size=0.7 )
```

Dilanjutkan dengan Langkah Preprocessing NLP yaitu (A. & Sonawane, 2016):

- menghilangkan URL, @ dan hashtag

```
import re
def remove_link(text):
text = re.sub(r'https?:\/\/.*[\r\n]*', '', text, flags=re.MULTILINE)
text = re.sub(r'\S*.com\S*', '', text, flags=re.MULTILINE)
text = re.sub(r'@\S*\s', '', text, flags=re.MULTILINE)
text = re.sub(r'\S*#\S*', '', text, flags=re.MULTILINE)
return text
```

- menghilangkan emoji

```
import emoji
def remove_emoji(text):
  text = emoji.get_emoji_regexp().sub("", text)
  return text
```

- menghilnagkan symbol dan angka

```
def remove_number_symbol(text):
   text = re.sub(r'\d.', '', text, flags=re.MULTILINE)
   text = re.sub(r'[^a-zA-Z]', ' ', text, flags=re.MULTILINE)
   text = re.sub(r'\s+', ' ', text, flags=re.MULTILINE)
   return text
```

- menghilangkan stopword

```
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
def remove_stopwords(text):
   tokens = word_tokenize(text)
   listStopword =  set(stopwords.words('indonesian'))
   removed = []
   for t in tokens:
      if t not in listStopword:
         removed.append(t)
   return " ".join(removed)
```

- melakukan stemming

```
from Sastrawi.Stemmer.StemmerFactory import StemmerFactory
sastrawi_factory = StemmerFactory()
sastrawi_stemmer = sastrawi_factory.create_stemmer()
def stemming(text):
   text = sastrawi_stemmer.stem(text)
   return text
```

d. Metode Ekstraksi Data Text yang akan dilakukan adalah dengan Word Embedding dengan Fasttext dimana akan memetakan teks ke vector dengan 100 dimensi, secara sederhana potongan kode sebagai berikut

```
import fasttext

fasttext_model = fasttext.load_model("cc.id.100.bin")
fasttext_model.get_sentence_vector(text)
```

e. Feature Selection digunakan semua vector yang dihasilkan dari word2vec

f. Model yang digunakan adalah SVM adalah sebagai berikut

```
def preprocessing_data(text):
    text = remove_link(text)
    text = remove_emoji(text)
    text = remove_number_symbol(text)
    text = remove_stopwords(text)
    text = stemming(text)

    return text

import fasttext
fasttext_model = fasttext.load_model("cc.id.100.bin")
x_test_vector = [ create_vector(fasttext_model, x)  for x in x_test]
x_train_vector = [ create_vector(fasttext_model, x)  for x in x_train]

from sklearn.svm import SVC
from sklearn.model_selection import cross_val_score,cross_validate

clf = SVC()
clf.fit(x_train_vector,y_train)

from sklearn.metrics import confusion_matrix, classification_report, accuracy_score
y_pred = clf.predict(x_test_vector)

classif_report = classification_report(y_test, y_pred)
confus_matrix = confusion_matrix(y_test, y_pred)

print(classif_report)
print(confus_matrix)
```

Sedangkan pada model Neural Network sebagai berikut

```
from torch import nn
import torch
from transformers import AutoModel

class ClassifierModel(nn.Module):
    def __init__(self, input_size, device):
```

```
        super(ClassifierModel, self).__init__()
        self.input_size = input_size
        self.device = device

        self.out = nn.Sequential(
            nn.Linear(int(self.input_size), 100),
            nn.ReLU(),
            nn.Linear(100, 50),
            nn.ReLU(),
            nn.Linear(50, 2)
        )
        self.sigmoid = nn.Sigmoid()

    def forward(self, vector):
        out = self.out(vector)
        sigmoid = self.sigmoid(out)
        sigmoid_reshape = torch.reshape(sigmoid, (sigmoid.shape[0], sigmoid.shape[-1]))
        return sigmoid_reshape
```

Untuk code terdapat pada file jupyter notebook

g. Kesimpulan pada tugas Sentiment Analysis adalah

Pada perbandingan model SVM dan Neural Network menghasilkan hasil akurasi yang serupa yaitu 68%

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.69 | 0.65 | 0.67 | 2147 |
| 1 | 0.67 | 0.71 | 0.69 | 2140 |
| accuracy | | | 0.68 | 4287 |
| macro avg | 0.68 | 0.68 | 0.68 | 4287 |
| weighted avg | 0.68 | 0.68 | 0.68 | 4287 |

Gambar 3 Report Test Model Neural Network

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.67 | 0.71 | 0.69 | 2147 |
| 1 | 0.69 | 0.65 | 0.67 | 2140 |
| accuracy | | | 0.68 | 4287 |
| macro avg | 0.68 | 0.68 | 0.68 | 4287 |
| weighted avg | 0.68 | 0.68 | 0.68 | 4287 |

Gambar 4 Report Test Model SVM

Namun jika dilihat dari akurasi data latihnya model neural network memiliki akurasi lebih tinggi yaitu 84% dibanding SVM 75%

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.82 | 0.86 | 0.84 | 738 |
| 1 | 0.85 | 0.82 | 0.83 | 731 |
| accuracy | | | 0.84 | 1469 |
| macro avg | 0.84 | 0.84 | 0.84 | 1469 |
| weighted avg | 0.84 | 0.84 | 0.84 | 1469 |

Gambar 5 Report Train Model Neural Network

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.73 | 0.79 | 0.76 | 915 |
| 1 | 0.77 | 0.71 | 0.74 | 922 |
| accuracy | | | 0.75 | 1837 |
| macro avg | 0.75 | 0.75 | 0.75 | 1837 |
| weighted avg | 0.75 | 0.75 | 0.75 | 1837 |

Gambar 6 Report Train Model SVM

Model Neural Network mampu mendapatkan akurasi tinggi pada train namun memiliki akurasi yang berbeda jauh dengan data test nya hingga 16% sehingga ada kecenderungan overfitting, sedangkan SVM lebih mendekati antara akurasi train dengan test nya dengan perbedaan 7%

Untuk Code Sentiment Analysis Classification Dapat dilihat di https://github.com/AxlAdilla/UAS-DataScience

## 3. DAFTAR PUSTAKA


A., V., & Sonawane, S. S. (2016). Sentiment Analysis of Twitter Data: A Survey of Techniques. *International Journal of Computer Applications*, *139*(11), 5–15. https://doi.org/10.5120/ijca2016908625

Kajala, A., & Jain, V. K. (2020). Diagnosis of Breast Cancer using Machine Learning Algorithms-A Review. *2020 International Conference on Emerging Trends in Communication, Control and Computing (ICONC3)*, 1–5. https://doi.org/10.1109/ICONC345789.2020.9117320

Kim, Y. (2014). Convolutional Neural Networks for Sentence Classification. *ArXiv:1408.5882 [Cs]*. http://arxiv.org/abs/1408.5882

Lidya, S. K., Sitompul, O. S., & Efendi, S. (2015). *SENTIMENT ANALYSIS PADA TEKS BAHASA INDONESIA MENGGUNAKAN SUPPORT VECTOR MACHINE (SVM) DAN K-NEAREST NEIGHBOR (K-NN)*. 8.

Paliwal, S., Khatri, S. K., & Sharma, M. (2018). *Sentiment Analysis and Prediction Using Neural Networks*. 8.

Patgiri, R., Nayak, S., Akutota, T., & Paul, B. (2019). Machine Learning: A Dark Side of Cancer Computing. *ArXiv:1903.07167 [Cs, Stat]*. http://arxiv.org/abs/1903.07167