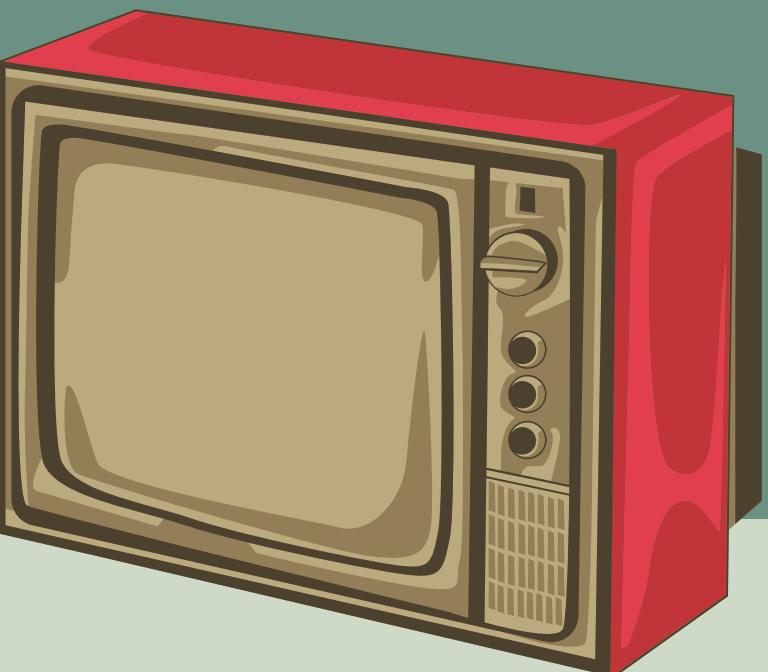


DESTY NURUL ANITSA EWAKO

**Analisis Deteksi Sentiment Acara TV
Indonesia dengan Membandingkan
Algoritma Machine Learning dan Fine
Tuning Bert Tensorflow**



Latar Belakang

Latar belakang = Acara televisi yang beragam disajikan di berbagai stasiun televisi memiliki tingkat kualitas yang berbeda-beda.

Sentimen masyarakat dapat dijadikan sebagai salah satu indikator oleh stasiun televisi untuk menentukan kualitas suatu acara. Pada twitter dapat dilakukan proses penggalian informasi mengenai sentimen masyarakat terhadap kualitas acara yang ditayangkan.

Salah satu teknik penggalian informasi pada twitter adalah analisis sentimen. Pada projek kali ini dilakukan analisis deteksi sentimen acara tv dengan algoritma machine learning dan transfer learning.



Tujuan

1. Mengetahui cara kerja dalam menganalisis sentimen dengan python
2. Mengetahui proses dalam klasifikasi sentimen di NLP dengan metode Algoritma Machine Learning dan Fine Tuning Bert Tensorflow
3. Mengetahui perbandingan algoritma dari segi proses komputasi dan akurasi
4. Program mampu melakukan analisis deteksi sentimen acara tv berdasarkan kalimatnya

Urgensi

Urgensi = Membuat Analisis Deteksi Sentiment Acara TV Indonesia dengan Membandingkan proses komputasi dan akurasi dari Algoritma Machine Learning dengan Fine Tuning Bert Tensorflow

Data Acquisition

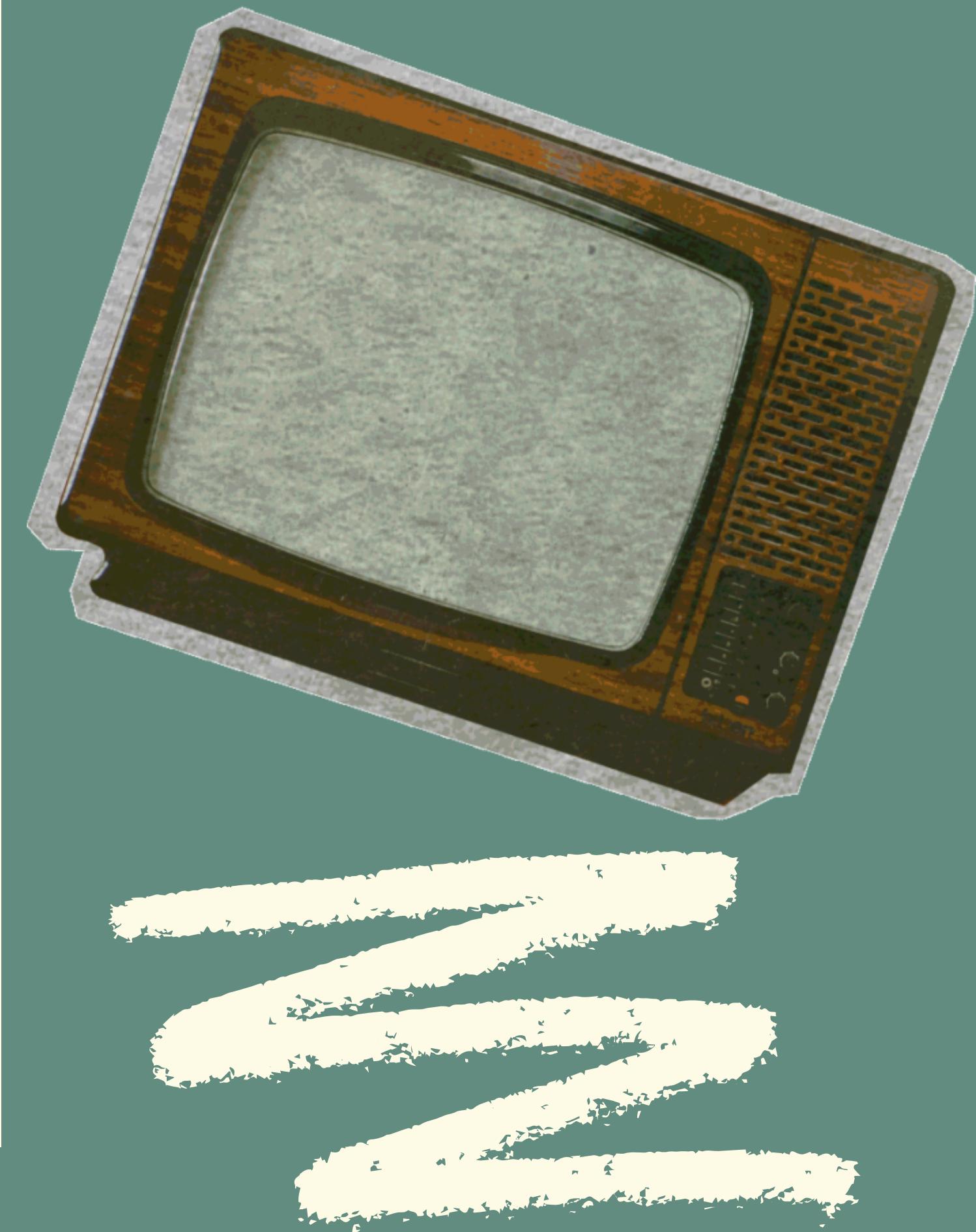
Data di peroleh secara Public Datasets, data sudah tersedia melalui berbagai sumber seperti google, kaggle, github. Dataset kali ini diambil dari github https://github.com/rizalespe/Dataset-Sentimen-Analisis-Bahasa-Indonesia/blob/master/dataset_tweet_sentiment_ayangan_tv.csv

Penjelasan Sentiment

- 0: Negative
- 1: Positive

Dataset

- X = Text Tweet
- Y = Sentiment



Preprocessing Data

Algoritma Machine Learning

Case Folding

```
[ ] import re

# Buat fungsi untuk langkah case folding
def casefolding(text):
    text = text.lower()                                # Mengubah teks menjadi lower case
    text = re.sub(r'https?://\S+|www\.\S+', '', text)  # Menghapus URL
    text = re.sub(r'[-+]?[0-9]+', '', text)            # Menghapus angka
    text = re.sub(r'[\^w\s]', '', text)                # Menghapus karakter tanda baca
    text = text.strip()
    return text
```

Case Folding proses dalam text preprocessing yang dilakukan untuk menyeragamkan karakter pada data.

```
▶ def text_normalize(text):
    text = ' '.join([key_norm[key_norm['singkat'] == word]['hasil'].values[0]
                    if (key_norm['singkat'] == word).any()
                    else word for word in text.split()])
    text = str.lower(text)
    return text
```

Kumpulan kata yang sudah dinormalisasikan, dari kata tidak baku/ singkatan menjadi kata baku

Filtering (Stopword Removal)

```
[ ] from nltk.tokenize import sent_tokenize, word_tokenize
from nltk.corpus import stopwords

stopwords_ind = stopwords.words('indonesian')
```

Stopword Removal / proses filtering, pemilihan kata-kata penting dari hasil token yaitu kata-kata apa saja yang digunakan untuk mewakili dokumen.

```
# Buat fungsi untuk langkah stopword removal

more_stopword = ['tsel', 'gb', 'rb', 'pd', 'gt']
stopwords_ind = stopwords_ind + more_stopword

def remove_stop_words(text):
    clean_words = []
    text = text.split()
    for word in text:
        if word not in stopwords_ind:
            clean_words.append(word)
    return " ".join(clean_words)
```

Stemming

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

# Buat fungsi untuk langkah stemming bahasa Indonesia
def stemming(text):
    text = stemmer.stem(text)
    return text
```

Stemming yaitu proses pengurangan dimensi pada data, dengan membuang imbuhan pada kata baku menjadi kata dasarnya.

Preprocessing Data

Algoritma Machine Learning

Text Preprocessing Pipeline

+ Code

```
# Buat fungsi untuk menggabungkan seluruh langkah text preprocessing
def text_preprocessing_process(text):
    text = casefolding(text)
    text = text_normalize(text)
    text = remove_stop_words(text)
    text = stemming(text)
    return text

[ ] %%time
data['clean_teks'] = data['Text Tweet'].apply(text_preprocessing_process)

# Perhatikan waktu komputasi ketika proses text preprocessing

CPU times: user 1min 10s, sys: 255 ms, total: 1min 10s
Wall time: 1min 11s
```

Membuat fungsi baru yaitu Text Preprocessing yang menyimpan variabel dari fungsi yang telah dibuat, dan menjalankan semua Fungsi Text Preprocessing yang telah dibuat pada semua data teks sentimen acara tv.

Preprocessing Data

Algoritma Transfer Learning Bert

03 Text Preprocessing

```
[ ] import re

def text_preprocessing(text):
    text = text.lower()                                     # Mengubah teks menjadi lower case
    text = re.sub(r'https?://\S+|www\.\S+', '', text)      # Menghapus URL
    text = re.sub(r'[-+]?[0-9]+', '', text)                # Menghapus angka
    text = re.sub(r'[\^\\w\\s]', '', text)                  # Menghapus karakter tanda baca
    text = text.strip()                                    # Menghapus whitespaces
    return text
```

```
[ ] %time data['Text Tweet'] = data['Text Tweet'].apply(text_preprocessing)
```

```
CPU times: user 4.4 ms, sys: 1 µs, total: 4.4 ms
Wall time: 4.28 ms
```

Berbeda dengan Algoritma Machine Learning, Text Preprocessing Transfer Learning Bert hanya melakukan case folding untuk menyeragamkan karakter pada data. Kemudian diaplikasikan pada semua text pada data sentimen acara tv.

Extraction dan Selection Feature

Algoritma Machine Learning

Feature Extraction

```
▶ from sklearn.feature_extraction.text import TfidfVectorizer  
  
tf_idf = TfidfVectorizer(ngram_range=(1,1))  
tf_idf.fit(X)  
  
⌚ TfidfVectorizer()  
  
[ ] # Melihat Jumlah Fitur  
print(len(tf_idf.get_feature_names_out()))  
  
1175
```

Proses mengubah teks menjadi vector menggunakan metode TF-IDF, Ngram yang digunakan unigram/perkata untuk membuat token pada data yang dijadikan fitur atau atribut, pada data sentimen acara tv didapatkan fitur 1175

Feature Selection

Feature Selection (Chi Square)

```
[ ] # Mengubah nilai data tabular tf-idf menjadi array agar dapat dijalankan  
X = np.array(data_tf_idf)  
y = np.array(y)  
  
[ ] from sklearn.feature_selection import SelectKBest  
from sklearn.feature_selection import chi2  
  
# Ten features with highest chi-squared statistics are selected  
chi2_features = SelectKBest(chi2, k=650)  
X_kbest_features = chi2_features.fit_transform(X, y)  
  
# Reduced features  
print('Original feature number:', X.shape[1])  
print('Reduced feature number:', X_kbest_features.shape[1])  
  
Original feature number: 1175  
Reduced feature number: 650
```

Mengurangi dimensi pada fitur dengan menggunakan chi-square, nilai chi-square, semakin tinggi nilainya maka semakin baik fiturnya k = 650, artinya 650 fitur yang memiliki skore teratas yang akan digunakan sebagai kbest featuralnya.

Load Tokenizer

Algoritma Transfer Learning Bert

05 Load Tokenizer

```
[ ] # Tentukan pre-trained model yang akan digunakan untuk fine-tuning  
# Daftar model dapat ditemukan pada https://huggingface.co  
  
PRE_TRAINED_MODEL = 'indobenchmark/indobert-base-p2' # https://huggingface.co/indobenchmark/indobert-base-p2  
  
[ ] from transformers import BertTokenizer  
  
bert_tokenizer = BertTokenizer.from_pretrained(PRE_TRAINED_MODEL) # Load tokenizer dari pre-trained model  
  
Downloading: 100% [229k/229k, 00:00<00:00, 177kB/s]  
Downloading: 100% [112/112, 00:00<00:00, 3.75kB/s]  
Downloading: 100% [2.00/2.00, 00:00<00:00, 67.2B/s]  
Downloading: 100% [1.53k/1.53k, 00:00<00:00, 31.8kB/s]  
  
[ ] # Lihat vocabulary dari pre-trained model yang telah di load sebelumnya  
vocabulary = bert_tokenizer.get_vocab()
```

Pretrained model yang digunakan adalah indobert, menggunakan BertTokenizer dari pretrained model.



Menentukan max_length kalimat pada semua data, diambil nilai tengahnya karena agar tidak terlalu banyak data yang hilang, dan tidak terlalu banyak padnya.

Input Formatting

Algoritma Transfer Learning Bert

Teks harus dipecah menjadi token sebelum dimasukkan ke BERT, kemudian token tersebut harus dipetakan ke indeks pada kosakata tokenizer yang telah di load sebelumnya. Tokenisasi harus dilakukan oleh tokenizer yang disertakan dengan BERT.

Input Formatting pada Data Latih & Uji

```
[ ] # Buat fungsi untuk menggabungkan langkah tokenisasi,  
#menambahkan special tokens untuk keseluruhan data  
#sebagai input formatting ke model BERT  
def convert_example_to_feature(sentence):  
    return bert_tokenizer.encode_plus(  
        sentence,  
        add_special_tokens = True,  
        padding = 'max_length',  
        truncation = 'longest_first',  
        max_length = MAX_LEN,  
        return_attention_mask = True,  
        return_token_type_ids=True  
)
```

```
# Buat fungsi untuk memetakan input hasil input formatting agar sesuai dengan model BERT  
def map_example_to_dict(input_ids, attention_masks, token_type_ids, label):  
    return {  
        "input_ids": input_ids, # Sebagai token embedding  
        "token_type_ids": token_type_ids, # Sebagai segment embedding  
        "attention_mask": attention_masks, # Sebagai filter informasi mana yang kalkulasi oleh model  
    }, label
```

```
# Buat fungsi untuk iterasi pada setiap kalimat pada keseluruhan data
def encode(data):
    input_ids_list = []
    token_type_ids_list = []
    attention_mask_list = []
    label_list = []

    for sentence, label in data.to_numpy():

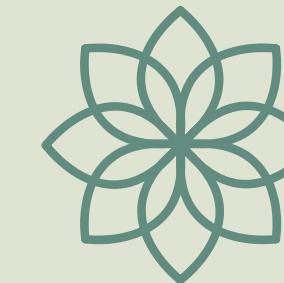
        bert_input = convert_example_to_feature(sentence)

        input_ids_list.append(bert_input['input_ids'])
        token_type_ids_list.append(bert_input['token_type_ids'])
        attention_mask_list.append(bert_input['attention_mask'])
        label_list.append([label])

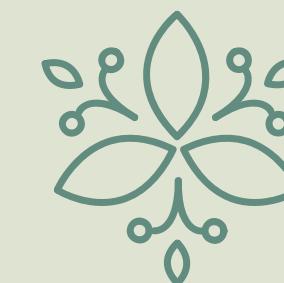
    return tf.data.Dataset.from_tensor_slices(
        (input_ids_list, attention_mask_list,
         token_type_ids_list, label_list)).map(map_example_to_dict)
```

Membuat fungsi encode untuk melakukan iterasi pada setiap kalimat pada data

Model Machine Learning



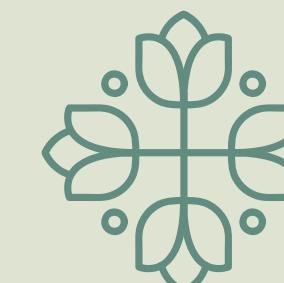
Split data yang digunakan
80% Training dan 20% Testing



Algoritma Supervised Learning
Naive Bayes



Parameter model yang
digunakan Kbest_Feature 650
fitur dengan skor terbaik



Evaluasi Model dengan Cross
Validaryin mencapai 88%

Modeling

```
# Split arrays or matrices into random train and test subsets.  
# https://scikit-learn.org/stable/modules/generated/sklearn.model\_selection.train\_test\_split.html  
  
X_train, X_test, y_train, y_test = train_test_split(X_kbest_features,  
                                                y, test_size=0.2,  
                                                random_state=40)  
  
# Training the model  
  
algorithm = MultinomialNB()  
model = algorithm.fit(X_train, y_train) # Fitkan (latih) algoritma  
  
dump(model, filename='model_1.joblib')
```

```
# Gunakan model yang telah di latih untuk memprediksi label pada data uji  
model_pred = model.predict(X_test)  
  
# Tampilkan hasil prediksi label dari model  
model_pred  
  
array([1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0,  
      1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0,  
      1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1,  
      1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1])  
  
# Tampilkan label sebenarnya pada data uji (actual label)  
y_test  
  
array([1, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0,  
      1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0,  
      1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0,  
      1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0])
```

Model Evaluation

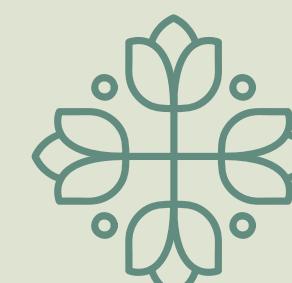
05 Model Evaluation

```
▶ # Hitung jumlah data yang berhasil di prediksi model  
# & jumlah data yang salah di prediksi  
prediksi_benar = (model_pred == y_test).sum()  
prediksi_salah = (model_pred != y_test).sum()  
  
print('Jumlah prediksi benar\t:', prediksi_benar)  
print('Jumlah prediksi salah\t:', prediksi_salah)  
  
accuracy = prediksi_benar / (prediksi_benar + prediksi_salah)*100  
print('Akurasi pengujian\t:', accuracy, '%')  
  
Jumlah prediksi benar : 67  
Jumlah prediksi salah : 13  
Akurasi pengujian : 83.75 %
```

Cross Validation

```
from sklearn.model_selection import ShuffleSplit # bisa pilih beberapa teknik cross validation  
from sklearn.model_selection import cross_val_score # untuk mengetahui performa model pada cross validation  
  
cv = ShuffleSplit(n_splits=10, test_size=0.2, random_state=50)  
  
cv_accuracy = (cross_val_score(model, X_kbest_features, y, cv=cv, scoring='accuracy'))  
  
avg_accuracy = np.mean(cv_accuracy)  
  
print('Naive Bayes')  
print('Akurasi setiap split:', cv_accuracy)  
print('Rata-rata akurasi pada cross validation:', avg_accuracy)  
  
Naive Bayes  
Akurasi setiap split: [0.8125 0.9625 0.9 0.8375 0.9 0.825 0.9375 0.8875 0.9125 0.85 ]  
Rata-rata akurasi pada cross validation: 0.8825
```

Model Transfer Learning Bert



Input formatting menggunakan fungsi sebelumnya pada data keseluruhan data, megubah data ke format bert pada data yang akan ditrain, test, dan validasi

Model yang digunakan,
`TFBertForSequenceClassification`
Parameter yang digunakan,
primizer adam, learning rate $5e-5$

Akurasi yang didapatkan dari
fine tuning untuk train 100%
dan Testing 95%

Model

```
# Lakukan input formatting menggunakan fungsi sebelumnya pada data keseluruhan data
train_encoded = encode(df_train).batch(BATCH_SIZE) #mengubah data ke format bert
test_encoded = encode(df_test).batch(BATCH_SIZE)
val_encoded = encode(df_val).batch(BATCH_SIZE)
```

```
[35] # Tentukan nilai hyperparameter untuk fine-tuning
      EPOCHS = 5
      BATCH_SIZE = 32
      LEARNING_RATE = 5e-5
```

```
%%time
bert_history = bert_model.fit(train_encoded, epochs=EPOCHS, batch_size=BATCH_SIZE, validation_data=val_encoded)

Epoch 1/5
10/10 [=====] - 2s 170ms/step - loss: 0.0024 - accuracy: 1.0000 - val_loss: 0.3911 - val_accuracy: 0.9000
Epoch 2/5
10/10 [=====] - 2s 170ms/step - loss: 0.0029 - accuracy: 1.0000 - val_loss: 0.2748 - val_accuracy: 0.9250
Epoch 3/5
10/10 [=====] - 2s 170ms/step - loss: 0.0011 - accuracy: 1.0000 - val_loss: 0.2314 - val_accuracy: 0.9250
Epoch 4/5
10/10 [=====] - 2s 171ms/step - loss: 8.4794e-04 - accuracy: 1.0000 - val_loss: 0.2476 - val_accuracy: 0.9500
Epoch 5/5
10/10 [=====] - 2s 172ms/step - loss: 7.0444e-04 - accuracy: 1.0000 - val_loss: 0.2544 - val_accuracy: 0.9500
CPU times: user 7.95 s, sys: 673 ms, total: 8.62 s
Wall time: 10.3 s
```

Evaluation Model

```
from transformers import TFBertForSequenceClassification

# Load model
bert_model = TFBertForSequenceClassification.from_pretrained(PRE_TRAINED_MODEL, num_labels=2)
```

```
# Tentukan optimizer dengan learning rate tertentu
# Paper aslinya menggunakan Adam Optimizer
optimizer = tf.keras.optimizers.Adam(learning_rate=LEARNING_RATE)

# Karena tidak menggunakan one-hot vectors, sehingga loss function
# dapat menggunakan sparse categorical cross entropy
loss = tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True)
metric = tf.keras.metrics.SparseCategoricalAccuracy('accuracy')

# Compile model
bert_model.compile(optimizer=optimizer, loss=loss, metrics=[metric])
```

```
score = bert_model.evaluate(test_encoded)

print("Test Accuracy:", score[1])

2/2 [=====] - 0s 44ms/step - loss: 0.4094 - accuracy: 0.9250
Test Accuracy: 0.925000011920929

predicted_raw = bert_model.predict(test_encoded)

y_pred = np.argmax(predicted_raw['logits'], axis=1)
y_true = np.array(df_test['Sentiment'])

from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report

accuracy_score(y_true, y_pred)

0.925
```

Performa Model

MODEL MACHINE LEARNING

- Saat melakukan text preprocessing pada semua data membutuhkan waktu 1min 11s
- untuk melakukan tokenisasi dengan mengubah kalimat ke fitur/kata menggunakan ngram, sedangkan TF-IDF mengubah token tersebut kedalam vektor.
- Waktu yang dibutuhkan pada saat training cepat komputasi ringan
- Model yang digunakan algoritma Naive bayes, didapatkan akurasi 88%

MODEL TRANSFER LEARNING BERT

- Saat melakukan text preprocessing pada semua data membutuhkan waktu 25.4 ms
- Tokenizer menggunakan bert yang berisikan pretrained model.
- Waktu yang dibutuhkan pada saat training / fine tuning membutuhkan waktu 10.3
- Hasil akurasi dari bert 95%

I N S I G T

- Model ML cocok digunakan untuk mencari topik dan peringkasan data.
- Model ML tidak memperhatikan urutan pada kalimat dan bersifat low semantik

- Model Transfer Learning cocok digunakan dalam pengklasifikasian teks .
- Model Transfer Learning Bert memperhatikan semantik kata dan urutanya

- Model ML memiliki waktu komputasi yang lebih ringan dan cepat
- Model ML memiliki akurasi 88% dan kurang powerfull karena bersifat low semantik.

- Model Transfer Learning Bert memiliki waktu komputasi yang ringan dan cepat
- Model Transfer Learning Bert memiliki akurasi 95%
- Model powerfull karena memiliki akurasi yang tinggi dan dapat memahami kalimat.

Kesimpulan

Model Transfer Learning Bert cocok digunakan untuk kasus klasifikasi pada sentimen acara telivisi karena lebih powerfull.

