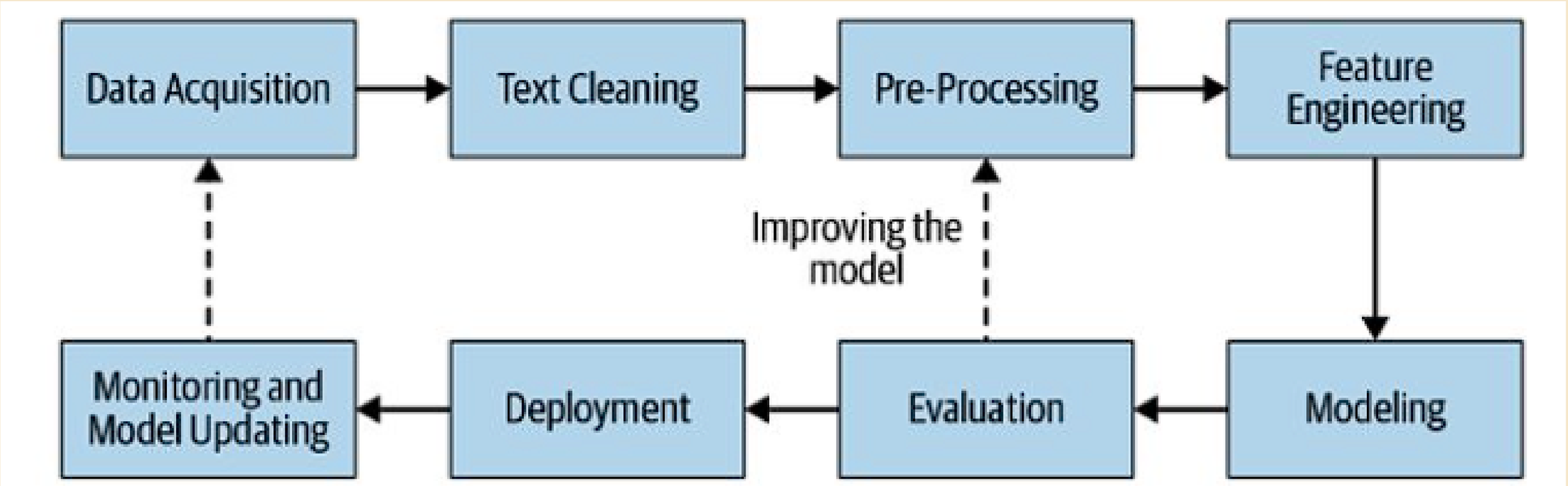
# PERBANDINGAN ALGORITMA MACHINE LEARNING TRADISIONAL DAN LSTM PADA ANALISIS SENTIMENT KOMENTAR INSTAGRAM


**BY FAYZA APRILIZA - BROYDEN**

Natural Language Processing (NLP)

https://www.linkedin.com/in/fayzaapriliza/

# NLP PIPELINE

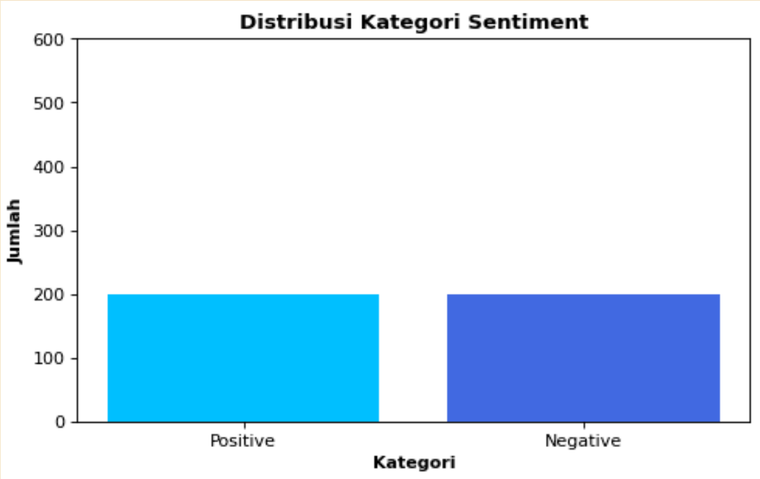



# INTRODUCTION

**Latar Belakang, Masalah, dan Urgensi**

Instagram merupakan platform media sosial yang memiliki jumlah pengguna cukup banyak. Pengguna instagram dapat saling berinteraksi melalui komentar pada postingan. Akan tetapi, tidak semua komentar memiliki sentimen positif. Ada banyak komentar negatif yang dapat merugikan. Oleh karena itu, perlu mengetahui sentimen dari suatu komentar, apakah positif atau negatif. Hal ini bertujuan untuk mengetahui apa yang disukai dan tidak disukai sehingga dapat mengambil tindakan yang sesuai.

# DATA ACQUISITION




Dataset yang digunakan adalah data komentar dari Instagram sebanyak 400 data yang terdiri atas 3 variabel, yakni Id, Sentiment, dan Intagram Comment Text. Dengan distribusi antar sentiment positif dan negatif berjumlah sama besar.

# DATA PREPROCESSING

## ML TRADISIONAL

Rename Column

```
data.rename(columns = {'Instagram Comment Text':'text'}, inplace = True)
```

Drop Column

```
data.drop(['Id'], axis=1, inplace=True)
data.head()
```

Convert Sentiment to Numeric

```
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()
data['Sentiment'] = le.fit_transform(data['Sentiment'])
data.head()
```

# DATA PREPROCESSING

## ML TRADISIONAL

### 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'#[A-Za-z0-9]+','', text)          # Menghapus hastag
  text = re.sub(r'[-+]?[0-9]+', '', text)           # Menghapus angka
  text = re.sub(r'[^\w\s]',' ', text)                # Menghapus karakter tanda baca
  text = re.sub("\s\s+" , " ", text)
  text = text.strip()
  return text
```

### Word Normalization

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

# DATA PREPROCESSING

## ML TRADISIONAL

### Stopword Removal

```
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

```
def stemming(text):
  text = stemmer.stem(text)
  return text
```

# DATA PREPROCESSING

## LSTM, BI-LSTM

Text Preprocessing

```
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'#[A-Za-z0-9]+','', text)           # Menghapus hastag
  text = re.sub(r'[-+]?[0-9]+', '', text)            # Menghapus angka
  text = re.sub(r'[^\w\s]',' ', text)                 # Menghapus karakter tanda baca
  text = re.sub("\s\s+" , " ", text)
  text = text.strip()
  text = text_normalize(text)
  return text
```

# FEATURE ENGINEERING

## ML TRADISIONAL

Feature Extraction dengan TF-IDF

```
tf_idf = TfidfVectorizer(ngram_range=(1,1))
tf_idf.fit(X)
```

```
▾ TfidfVectorizer
TfidfVectorizer()
```

Feature Selection (k = 800)

```
chi2_features = SelectKBest(chi2, k=800)
X_kbest_features = chi2_features.fit_transform(X, y)
```

## LSTM, BI-LSTM

Word Embedding dengan Word2Vec

```
EMBEDDING_SIZE = 100    # Dimensi word vektor / neuron pada projection (hidden) layer
WINDOW_SIZE = 5         # Window size. Jarak maksimum antara kata saat ini dan yang diprediksi dalam sebuah
MIN_WORD  = 1           # Model akan mengabaikan semua kata dengan frekuensi total lebih rendah dari ini (opsional)
EPOCH = 10              # Jumlah iterasi (epoch).
SG = 1                  # Strategi algoritma pelatihan: 1 untuk skip-gram, 0 untuk CBOW
NEGATIVE = 5            # Negative sampling. Jika 0, negative sampling tidak digunakan

%%time

# Proses training Word2Vec

from gensim.models import Word2Vec, FastText, KeyedVectors
model_word2vec = Word2Vec(sentences, vector_size=EMBEDDING_SIZE, sg=SG, min_count=MIN_WORD, window=WINDOW_SIZE, negative=
```

# MODELLING

### MultinomialNB

```
mnb = MultinomialNB()
```

### Random Forest

```
rf = RandomForestClassifier()
```

### Logistic Regression

```
lr = LogisticRegression()
```

### SVM

```
svm = svm.SVC()
```

### LSTM

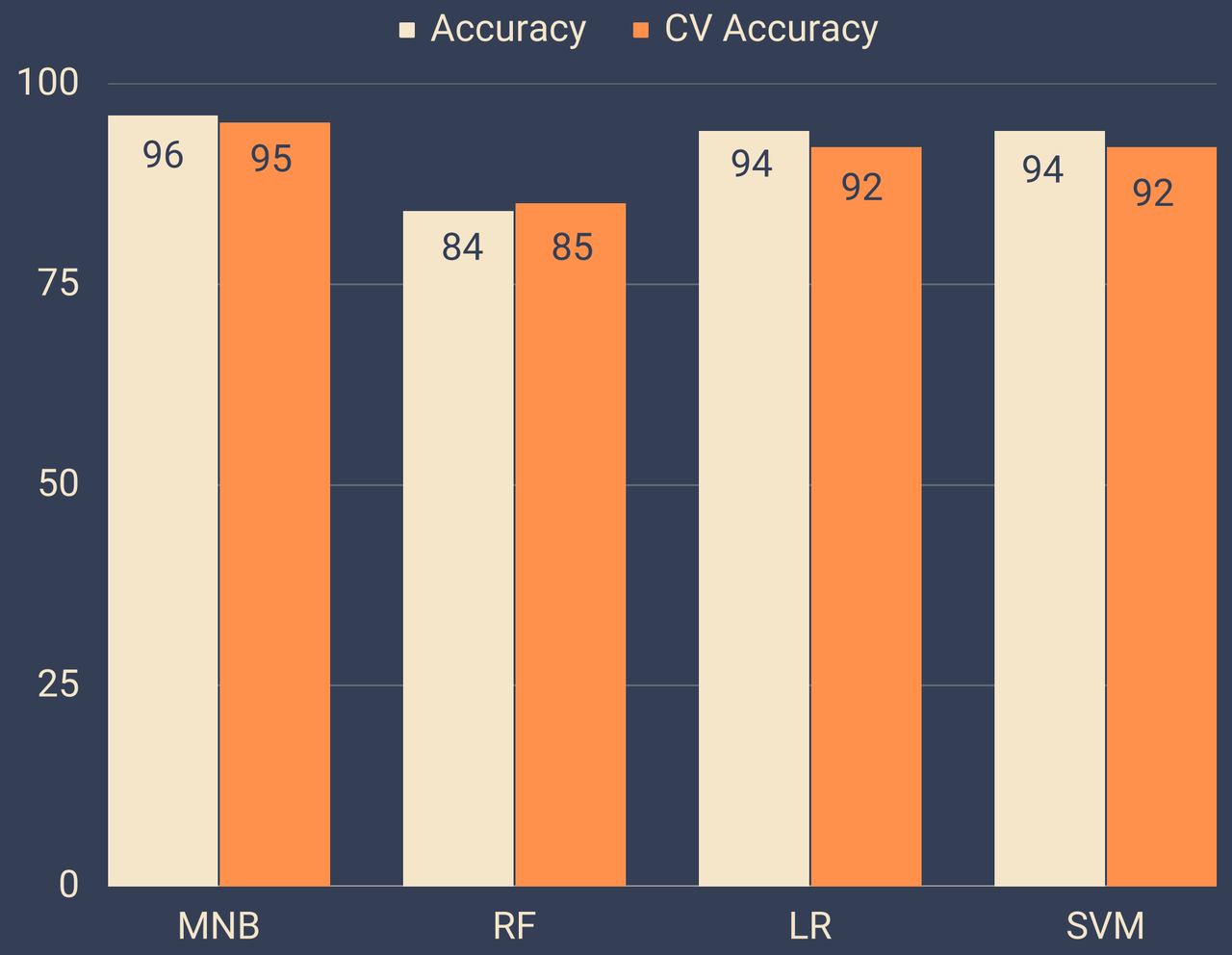
```
model_LSTM = Sequential()
model_LSTM.add(Embedding(
    input_dim = WV_DICTIONARY_SIZE,
    input_length = MAX_SEQ_LENGTH,
    output_dim = 64))
model_LSTM.add(LSTM(64))
model_LSTM.add(Dropout(0.9))
model_LSTM.add(Dense(2, activation='sigmoid'))
```

### Bi-LSTM + Word2Vec

```
model_BiLSTM_w2v = Sequential()
model_BiLSTM_w2v.add(Embedding(
    input_dim = WV_DICTIONARY_SIZE,
    output_dim = EMBEDDING_SIZE,
    input_length = MAX_SEQ_LENGTH,
    # trainable = False,
    embeddings_initializer = Constant(EMBEDDING_MATRIX)))
model_BiLSTM_w2v.add(Bidirectional(LSTM(64)))
# model_BiLSTM_w2v.add(LSTM(32))
# model_LSTM.add(Dropout(0.9))
model_BiLSTM_w2v.add(Dense(2, activation='sigmoid'))
```

# EVALUATION

Machine Learning Tradisional (MultinomialNB, Random Forest, Logistic Regression, Support Vector Machine)

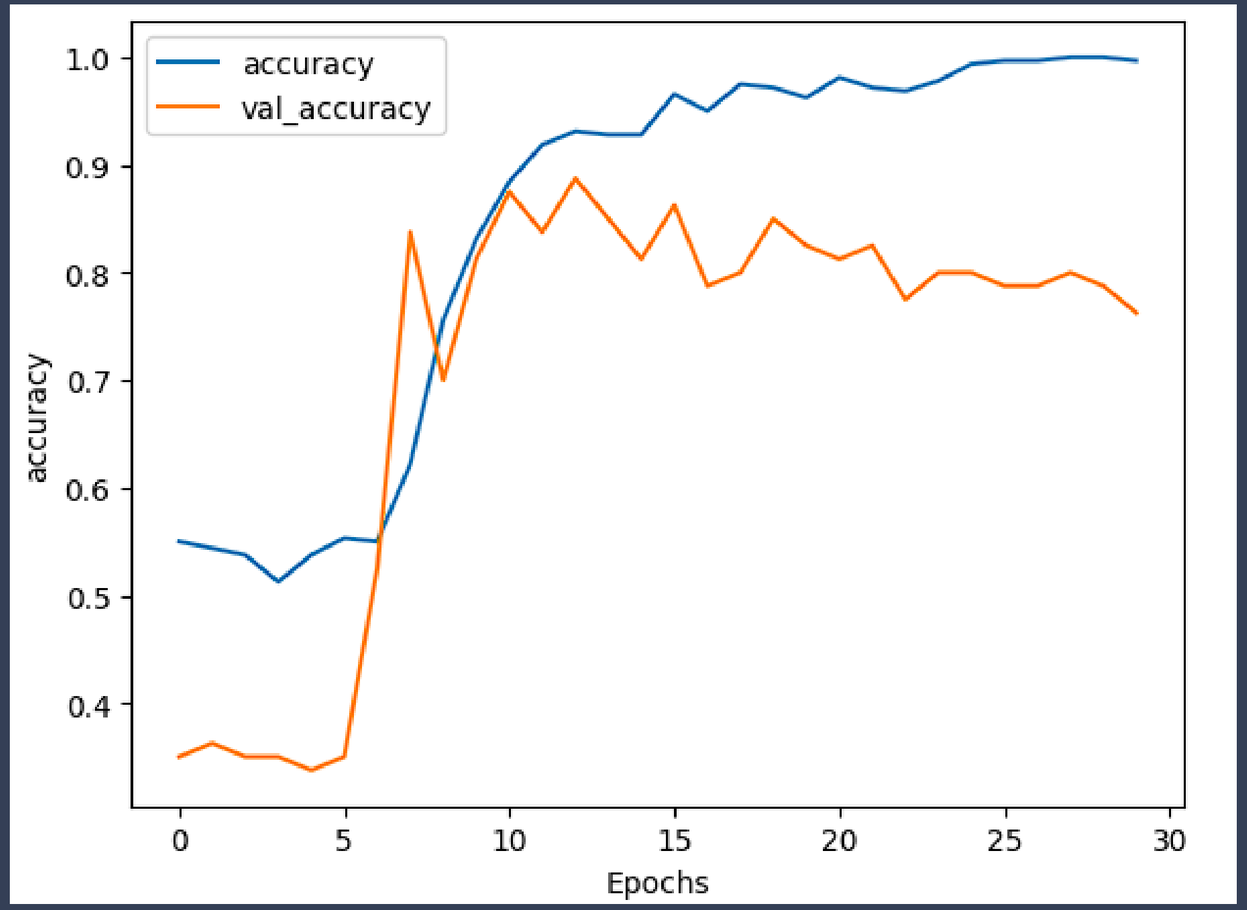


# EVALUATION

LSTM

Hyperparameter

- EPOCHS = 30
- BATCH_SIZE = 32
- Optimizer = Adam(epsilon=2e-8)
- Metrics = accuracy
- loss= categorical_crossentropy



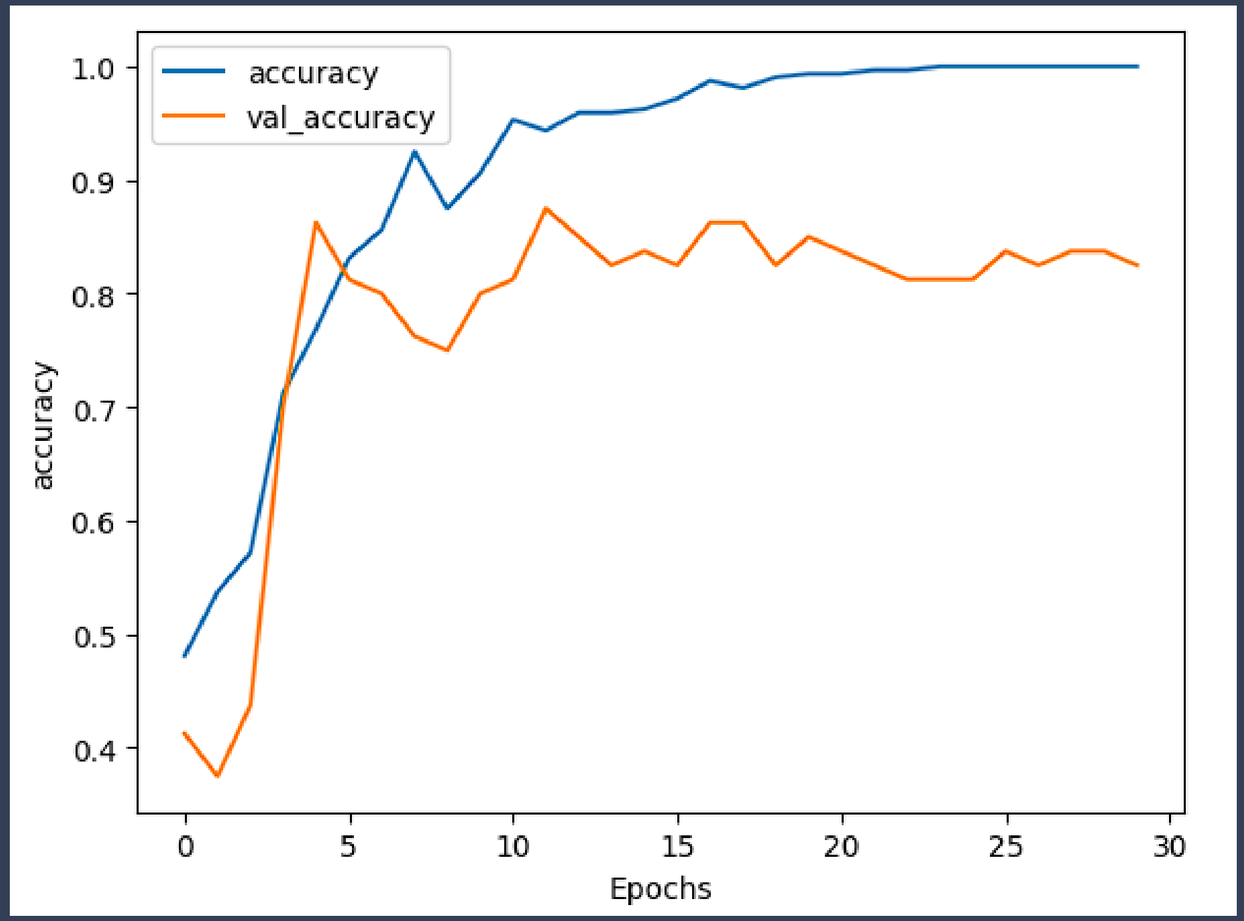

Diperoleh hasil akurasinya sebesar 76%

# EVALUATION

Bi-LSTM + Word2Vec

Hyperparameter

- EPOCHS = 30
- BATCH_SIZE = 32
- Optimizer = Adam(epsilon=2e-8)
- Metrics = accuracy
- loss= categorical_crossentropy




Diperoleh hasil akurasinya sebesar 82%

# CONCLUSION


Dengan menggunakan teknik preprocessing yang sama diperoleh MultinomialNB merupakan algoritma yang memiliki akurasi tertinggi dibandingkan algoritma ML tradisional lainnya. Sedangkan, Bi-LSTM memperoleh akurasi lebih baik dibandingkan LSTM dengan hyperparameter dan proses training yang sama.

MultinomialNB dengan TF-IDF n_gram=(1,1) memiliki akurasi sebesar 96% sedangkan Bi-LSTM dengan Word2Vec memiliki akurasi sebesar 82%.

Jadi, MultinomialNB memiliki akurasi yang paling baik tetapi belum memiliki kemampuan menangkap makna dari suatu kata karena masih menggunakan TF-IDF.

Kampus Merdeka INDONESIA JAYA
orbit FUTURE ACADEMY Skills For Future Jobs
Institut Teknologi Telkom Purwokerto
Bridging Technology for Humanity

# THANK YOU

BY FAYZA APRILIZA