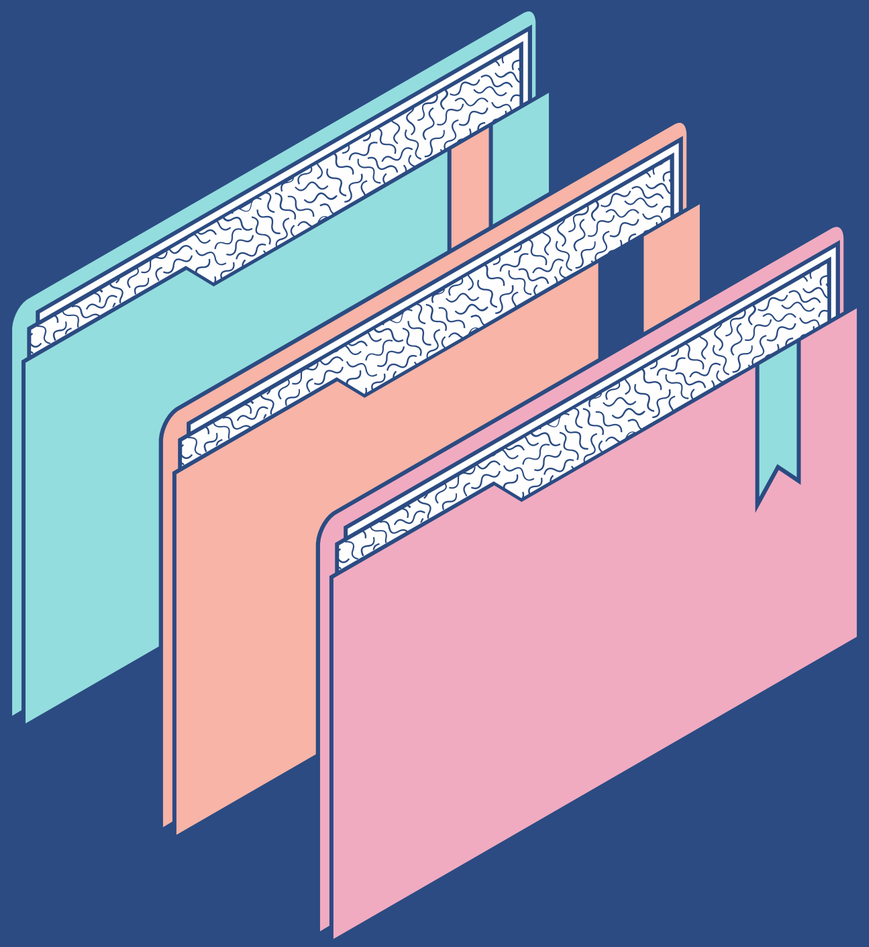
BINAR PLATINUM CHALLENGE

Sentiment Analysis with LSTM and ANN Model

Zulfikar Ramadhan
Hadziq Mufid Mahmud



Introduction

- Melakukan analisis sentimendengan model dengan dua algoritma (LSTM dan ANN) pada setiap masukan (file csv dan json text). Dengan mendeployment berupa Flask API dan Demo dengan Streamlit.

- Sentiment Analysis adalah proses menganalisisteks digital untuk menentukan apakah nada emosional pesan tersebut positif, negatif, atau netral.

- Sentiment analysis menggunakan NLP untuk mengidentifikasi sentimen dan memberikan interpretasi hasilnya.

Roadmap Sentiment Analysis

1 Prepare Dataset

2 Cleansing Data

3 Feature Extraction

4 Training Model

5 Evaluation Model

6 Deployment with Flask API and Streamlit

7 Sentiment Analysis

Prepare Dataset

Melihat dataset yang tidak seimbang, kami memutuskan untuk menambah dataset hanya untuk label neutral dan negative.

Link
https://github.com/ridife/dataset-idsa/blob/master/Indonesian%20Sentiment%20Twitter%20Dataset%20Labeled.csv

Masing masing label berjumlah +- 6000

```
#Count value (Sentimen) data 1
train_df1['Sentimen'].value_counts()
```

```
positive    6383
negative    3412
neutral     1138
Name: Sentimen, dtype: int64
```

```
#Read data 2
train_df2 = pd.read_csv("/content/Indonesian Sentiment Twitter Dataset Labeled.csv", sep='\t')
train_df2
```

```
#Drop label positive (sentimen=1) data 2
train_df2 = train_df2[train_df2['sentimen'] < 1]
train_df2
```

```
#Drop duplicates data 2
train_df2 = train_df2.drop_duplicates()
train_df2
```

```
#Count value (sentimen) data 2
train_df2['sentimen'].value_counts()
```

```
#Replace label 0 to neutral and -1 to negative
train_df2['Sentimen'] = np.where(train_df2['sentimen']== 0, 'neutral', 'negative')
train_df2.drop('sentimen', axis=1, inplace=True)
train_df2
```

```
#Combine data 1 and data 2
train_df = pd.concat([train_df1, train_df2])
train_df
```

```
#Count value (Sentimen) final data training
train_df['Sentimen'].value_counts()
```

```
positive    6383
neutral     6222
negative    6103
Name: Sentimen, dtype: int64
```

Cleansing Data

´Agar model menjadi lebih akurat, maka dilakukan cleansing untuk emoticon dan punctuation

´Dikarenakan banyak ejaan yang tak baku dan tak sesuai KBBI, maka kami meminimalisir itu dengan word normalization

```
#Cleansing data (tweet)
train_df['Tweet'] = train_df['Tweet'].str.strip()
train_df['Tweet'] = train_df['Tweet'].replace(r'([A-Z]+)\s(\d+)', r'\1\2', regex=True)
train_df['Tweet'] = train_df['Tweet'].str.replace(r'\\x[A-Za-z0-9./:)(*&^%$#@!_;]+', '')
train_df['Tweet'] = train_df['Tweet'].str.replace(r'[^\w\d\s]+', '')
train_df['Tweet'] = train_df['Tweet'].str.lower()
train_df['Tweet'] = train_df['Tweet'].str.replace(r'user', '')
train_df['Tweet'] = train_df['Tweet'].str.replace(r'_', '')
train_df
```

```
#Read alay dictionary
alay_dict_df = pd.read_csv('/content/colloquial-indonesian-lexicon.csv')
alay_dict_df
```

```
alay_dict = alay_dict_df.groupby('formal')['slang'].apply(list).to_dict()
```

```
with open("alay_dict.json", "w") as outfile:
  json.dump(alay_dict, outfile)
```

```
from google.colab import files
files.download ("alay_dict.json")
```

```
with open("alay_dict.json") as f:
  alay_dict = json.load(f)
```

```
keyword_processor = KeywordProcessor()
keyword_processor.add_keywords_from_dict(alay_dict)

def replace_alay(text):
  return keyword_processor.replace_keywords(text)
```

```
train_df['Tweet'] = train_df['Tweet'].apply(replace_alay)
train_df
```

Feature Extraction on LSTM Model

Tokenization adalah proses untuk mengubah kata - angka. Setelah itu dilakukan pad sequence, yaitu mengubah tokenization menjadi bentuk list. Untuk Sentimen, kami melakukan one hot encoder

```
results = Counter()
hitung_sp_char = df['Tweet'].str.lower().str.split().apply(results.update)
print(len(results))
```

```
29170
```

```
max_features = 29200
tokenizer = Tokenizer(num_words=max_features, split=' ')
tokenizer.fit_on_texts(df['Tweet'].values)
X = tokenizer.texts_to_sequences(df['Tweet'].values)
```

```
with open('token.pickle','wb') as handle:
  pickle.dump(tokenizer,handle,protocol=pickle.HIGHEST_PROTOCOL)
  print('tokenizer.pickle has been created')
```

```
tokenizer.pickle has been created
```

```
find_max_list(X)
```

```
95
```

```
XX = pad_sequences(X, maxlen=128) #padding
```

```
Y = pd.get_dummies(df['Sentimen']).values
```

```
print(Y)
```

```
[[0 0 1]
 [0 1 0]
 [0 0 1]
 ...
 [1 0 0]
 [0 1 0]
 [0 1 0]]
```

Feature Extraction on ANN Model

Process dengan Bag of Words lalu diubah lagi dengan TF-IDF dan mengganti sentimen dengan angka

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

     count_vect = CountVectorizer()
     X_train_counts = count_vect.fit_transform(train_df['Tweet'])
     X_train_counts.shape
```

```
(18708, 29083)
```

```
[ ]  from sklearn.feature_extraction.text import TfidfTransformer

     tf_transformer = TfidfTransformer(use_idf=False).fit(X_train_counts)
     X_train_tf = tf_transformer.transform(X_train_counts)
     X_train_tf.shape
```

```
(18708, 29083)
```

```
[ ]  X = X_train_tf
     y = train_df.Sentimen
```

```
[ ]  sentimen = {'neutral': 0, 'positive': 1, 'negative': 2}
     y = y.replace(sentimen)
```

Train Model LSTM

```
X_Train, X_Test, y_train, y_test = train_test_split(XX,Y,test_size=0.2,random_state=42)
```

```
embed_dim = 128
lstm_out = 196

"""model = Sequential()
model.add(Embedding(max_features, embed_dim))
model.add(SpatialDropout1D(0.4))
model.add(LSTM(lstm_out, dropout=0.2, recurrent_dropout=0.2, input_length=XX.shape[1]))
model.add(Dense(2,activation='softmax'))
model.compile(loss = 'categorical_crossentropy', optimizer='adam',metrics = ['accuracy'])
print(model.summary())"""

units = 196

model = Sequential()
model.add(Embedding(max_features,embed_dim, input_length=XX.shape[1]))
model.add(LSTM(units,dropout=0.2, recurrent_dropout=0.2))
model.add(Dense(3,activation='softmax'))
```

```
model.compile(loss='categorical_crossentropy', optimizer='adam',metrics=['accuracy',Precision(), Recall()])
print(model.summary())
```

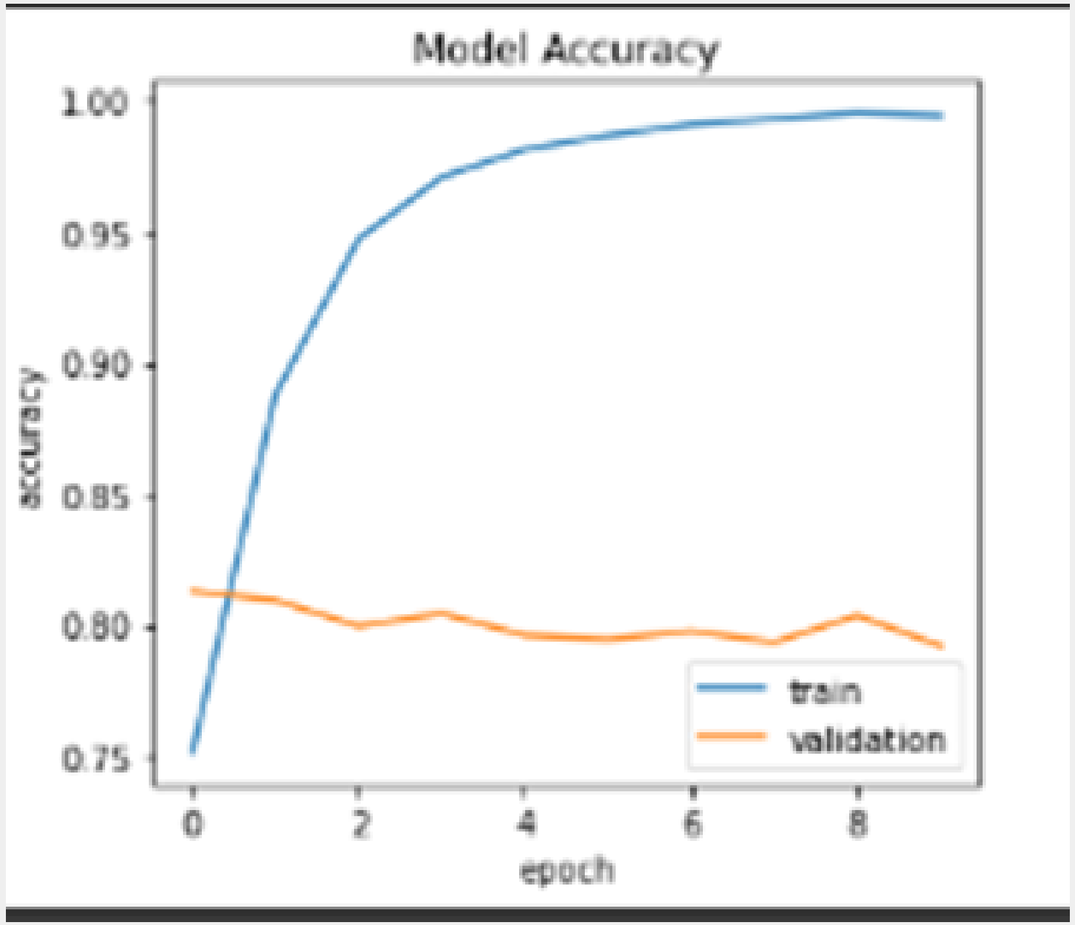
Train Model ANN

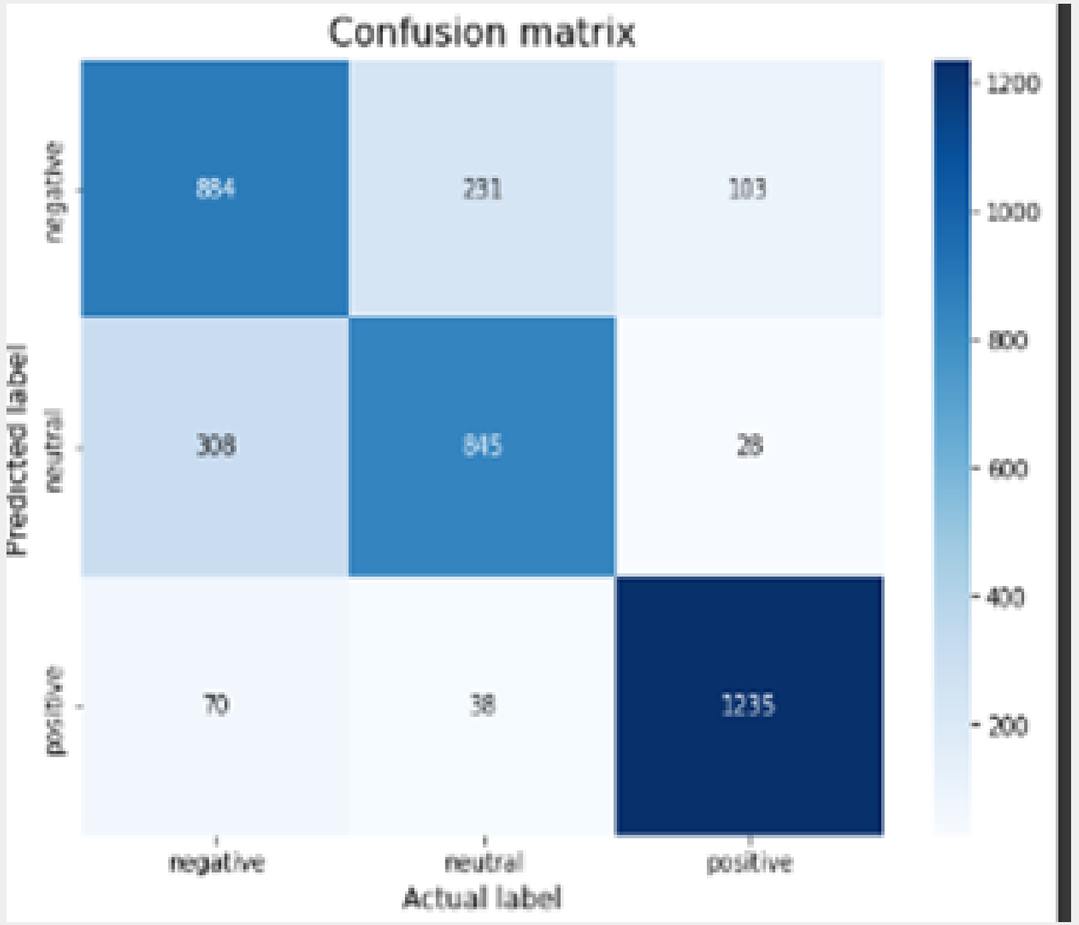
```
[ ] from sklearn.neural_network import MLPClassifier

    ann_model = MLPClassifier(hidden_layer_sizes=(256,128,64,36), activation="relu", random_state=42, solver="adam", alpha=0.00001, early_stopping=True, max_iter=1000, verbose=True)
    ann_model.fit(X_train, y_train)

    print("Training is done")
```

```
from sklearn.metrics import classification_report

test = ann_model.predict(X_test)

print("Testing is done")

print(classification_report(y_test, test))
```

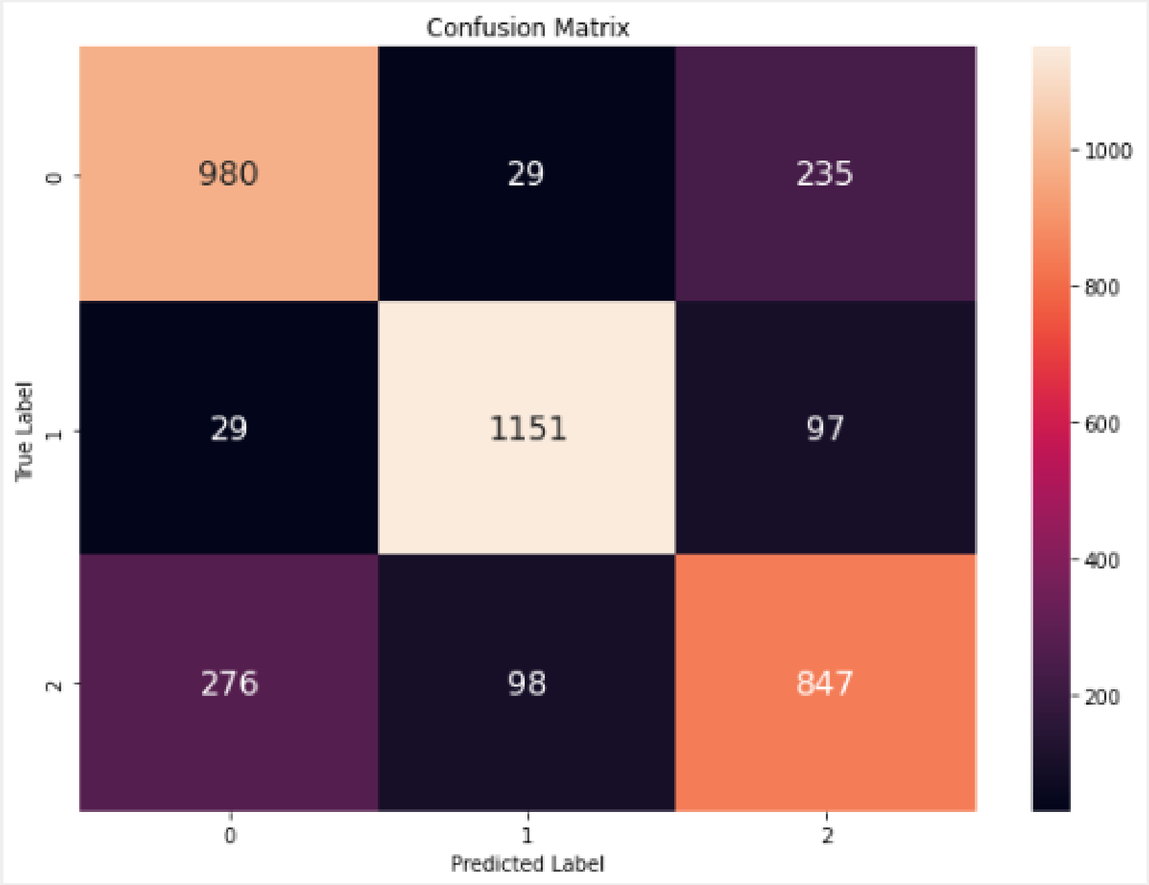
```
Testing is done
              precision    recall  f1-score   support

           0       0.76      0.79      0.78      1244
           1       0.90      0.90      0.90      1277
           2       0.72      0.69      0.71      1221

    accuracy                           0.80      3742
   macro avg       0.79      0.79      0.79      3742
weighted avg       0.80      0.80      0.80      3742
```

Evaluation Model LSTM

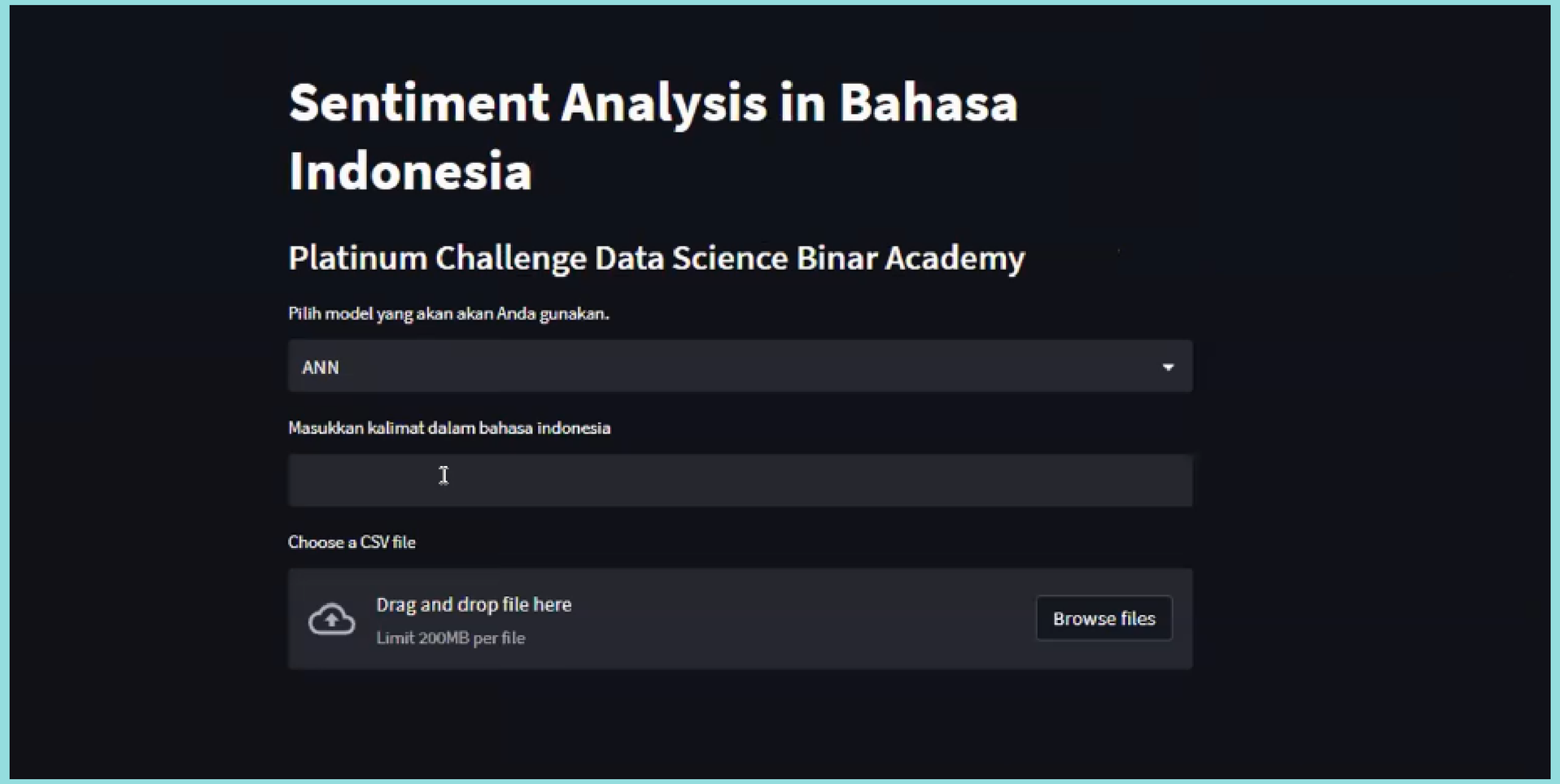




Evaluation Model ANN



Deployment with API and Demo on Streamlit



Conclusion

Machine Learning mampu mengklasifikasikan sentimen dari ribuan data tweet dalam hitungan menit. Meskipun akurasinya hanya 80% Machine Learning mampu melakukan sentimen dalam waktu yang cepat.