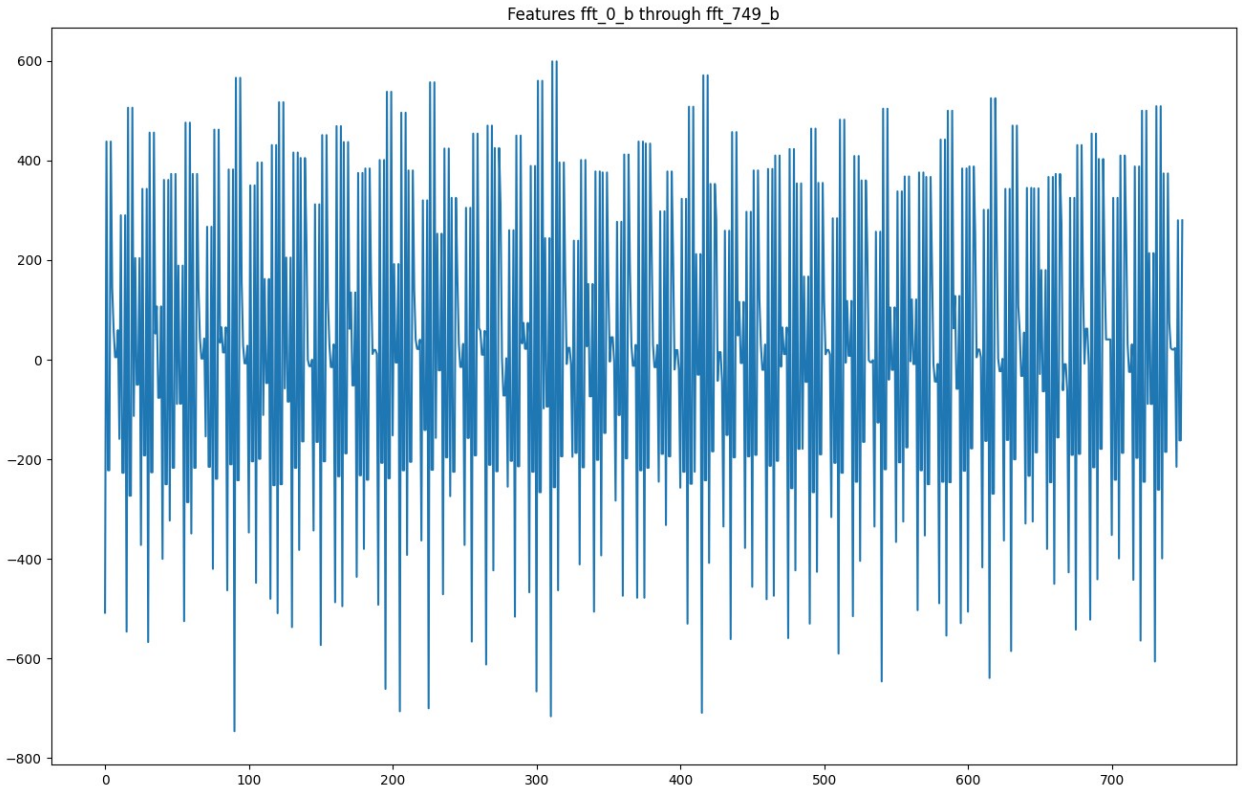
Project ML

```
import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split

import tensorflow as tf

from sklearn.metrics import confusion_matrix, classification_report

data = pd.read_csv('../content/emotions.csv')

data
```

{"type":"dataframe","variable_name":"data"}

```
sample = data.loc[0, 'fft_0_b':'fft_749_b']

plt.figure(figsize=(16, 10))
plt.plot(range(len(sample)), sample)
plt.title("Features fft_0_b through fft_749_b")
plt.show()
```




```
data['label'].value_counts()
```

```
label
NEUTRAL     716
NEGATIVE    708
POSITIVE    708
Name: count, dtype: int64
```

```
label_mapping = {'NEGATIVE': 0, 'NEUTRAL': 1, 'POSITIVE': 2}
```

Preprocessing

```
def preprocess_inputs(df):
    df = df.copy()

    df['label'] = df['label'].replace(label_mapping)

    y = df['label'].copy()
    X = df.drop('label', axis=1).copy()

    X_train, X_test, y_train, y_test = train_test_split(X, y,
train_size=0.7, random_state=123)

    return X_train, X_test, y_train, y_test
```

```
X_train, X_test, y_train, y_test = preprocess_inputs(data)
```

```
<ipython-input-63-3f2c0931f06d>:4: FutureWarning: Downcasting behavior
in `replace` is deprecated and will be removed in a future version. To
retain the old behavior, explicitly call
`result.infer_objects(copy=False)`. To opt-in to the future behavior,
set `pd.set_option('future.no_silent_downcasting', True)`
  df['label'] = df['label'].replace(label_mapping)
```

```
X_train
```

{"type":"dataframe","variable_name":"X_train"}

Modeling

```
inputs = tf.keras.Input(shape=(X_train.shape[1],))

# Wrap tf.expand_dims in a Lambda layer
expand_dims = tf.keras.layers.Lambda(lambda x: tf.expand_dims(x,
axis=2))(inputs)

gru = tf.keras.layers.GRU(256, return_sequences=True)(expand_dims)

flatten = tf.keras.layers.Flatten()(gru)

outputs = tf.keras.layers.Dense(3, activation='softmax')(flatten)


model = tf.keras.Model(inputs=inputs, outputs=outputs)
print(model.summary())
```

```
Model: "functional_4"
```

```
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳
━━━━━━━━━━━━━━━━━┓
┃ Layer (type)                          ┃ Output Shape                 ┃
Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇
━━━━━━━━━━━━━━━━━┩
│ input_layer_8 (InputLayer)            │ (None, 2548)                 │
0 │
├───────────────────────────────────────┼──────────────────────────────┼
─────────────────┤
│ lambda_4 (Lambda)                     │ (None, 2548, 1)              │
0 │
├───────────────────────────────────────┼──────────────────────────────┼
─────────────────┤
│ gru_4 (GRU)                           │ (None, 2548, 256)            │
198,912 │
```

```
├───────────────────────────────────────┼───────────────────────────────┤
─────────────────┤
│ flatten_4 (Flatten)                   │ (None, 652288)                │
0 │
├───────────────────────────────────────┼───────────────────────────────┤
─────────────────┤
│ dense_4 (Dense)                       │ (None, 3)                     │
1,956,867 │
└───────────────────────────────────────┴───────────────────────────────┴
─────────────────┘

 Total params: 2,155,779 (8.22 MB)

 Trainable params: 2,155,779 (8.22 MB)

 Non-trainable params: 0 (0.00 B)

None
```

```
model.compile(
    optimizer='adam',
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy']
)

history = model.fit(
    X_train,
    y_train,
    validation_split=0.2,
    batch_size=32,
    epochs=50,
    callbacks=[
        tf.keras.callbacks.EarlyStopping(
            monitor='val_loss',
            patience=5,
            restore_best_weights=True
        )
    ]
)
```

```
Epoch 1/50
38/38 ━━━━━━━━━━━━━━━━━━━━ 6s 125ms/step - accuracy: 0.6867 - loss:
41.4652 - val_accuracy: 0.8930 - val_loss: 14.5967
Epoch 2/50
38/38 ━━━━━━━━━━━━━━━━━━━━ 5s 124ms/step - accuracy: 0.9349 - loss:
3.0479 - val_accuracy: 0.9365 - val_loss: 3.9214
Epoch 3/50
38/38 ━━━━━━━━━━━━━━━━━━━━ 5s 121ms/step - accuracy: 0.9339 - loss:
3.1856 - val_accuracy: 0.8629 - val_loss: 8.7536
Epoch 4/50
38/38 ━━━━━━━━━━━━━━━━━━━━ 5s 125ms/step - accuracy: 0.9459 - loss:
```

```
2.5055 - val_accuracy: 0.9465 - val_loss: 4.1846
Epoch 5/50
38/38 ━━━━━━━━━━━━━━━━━━━━ 5s 118ms/step - accuracy: 0.9873 - loss:
0.2460 - val_accuracy: 0.9431 - val_loss: 3.9431
Epoch 6/50
38/38 ━━━━━━━━━━━━━━━━━━━━ 4s 117ms/step - accuracy: 0.9864 - loss:
0.3796 - val_accuracy: 0.9732 - val_loss: 1.2569
Epoch 7/50
38/38 ━━━━━━━━━━━━━━━━━━━━ 5s 120ms/step - accuracy: 0.9836 - loss:
0.2925 - val_accuracy: 0.9498 - val_loss: 4.3082
Epoch 8/50
38/38 ━━━━━━━━━━━━━━━━━━━━ 5s 116ms/step - accuracy: 0.9707 - loss:
2.1224 - val_accuracy: 0.9565 - val_loss: 1.6326
Epoch 9/50
38/38 ━━━━━━━━━━━━━━━━━━━━ 5s 118ms/step - accuracy: 0.9955 - loss:
0.1921 - val_accuracy: 0.9398 - val_loss: 5.3997
Epoch 10/50
38/38 ━━━━━━━━━━━━━━━━━━━━ 4s 117ms/step - accuracy: 0.9818 - loss:
2.0692 - val_accuracy: 0.9465 - val_loss: 6.0800
Epoch 11/50
38/38 ━━━━━━━━━━━━━━━━━━━━ 5s 114ms/step - accuracy: 0.9887 - loss:
0.2426 - val_accuracy: 0.9431 - val_loss: 4.5341
```

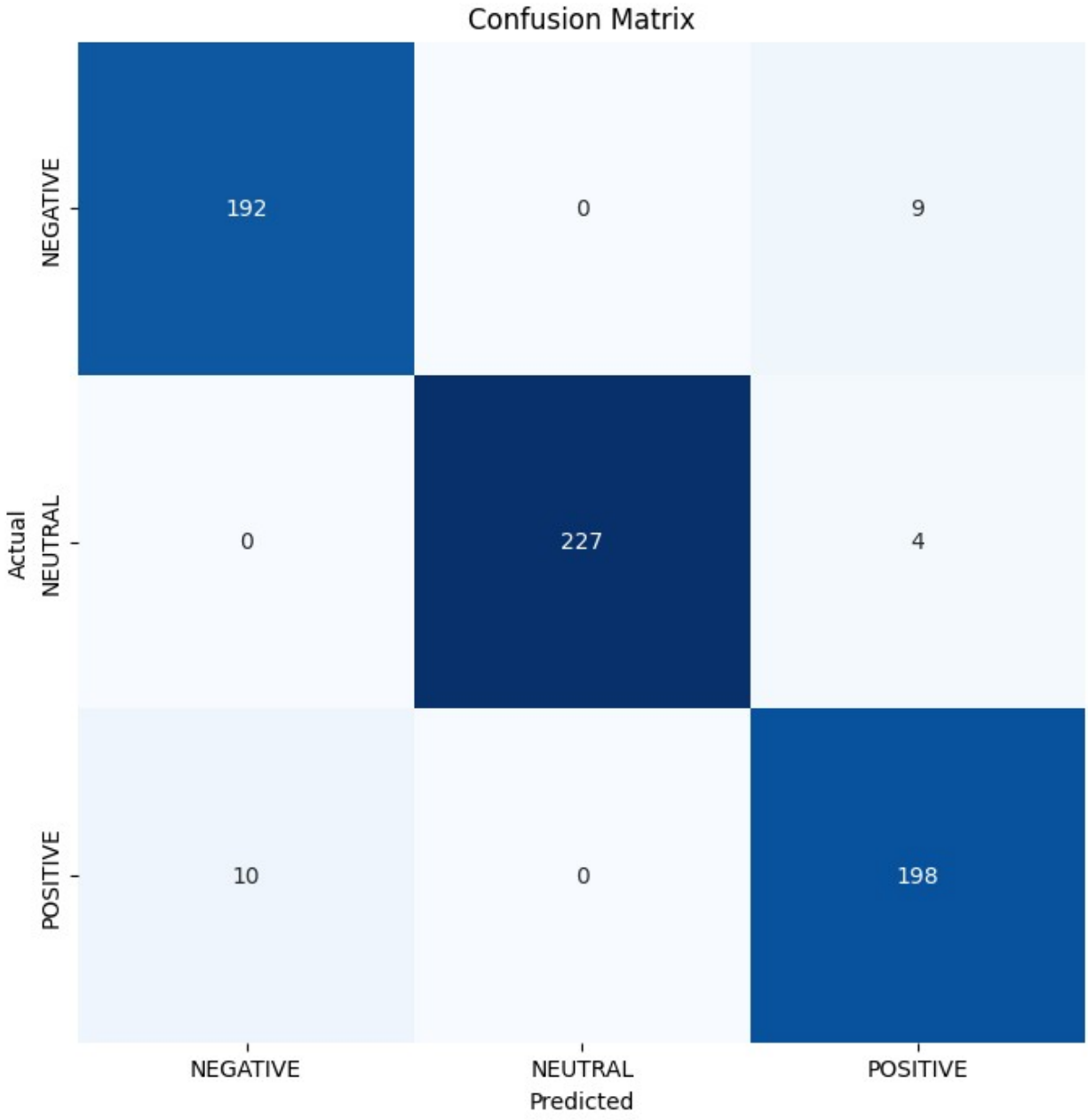
Results

```
model_acc = model.evaluate(X_test, y_test, verbose=0)[1]
print("Test Accuracy: {:.3f}%".format(model_acc * 100))
```

```
Test Accuracy: 96.406%
```

```
y_pred = np.array(list(map(lambda x: np.argmax(x),
model.predict(X_test))))

cm = confusion_matrix(y_test, y_pred)
clr = classification_report(y_test, y_pred,
target_names=label_mapping.keys())

plt.figure(figsize=(8, 8))
sns.heatmap(cm, annot=True, vmin=0, fmt='g', cbar=False, cmap='Blues')
plt.xticks(np.arange(3) + 0.5, label_mapping.keys())
plt.yticks(np.arange(3) + 0.5, label_mapping.keys())
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix")
plt.show()

print("Classification Report:\n----------------------\n", clr)
```

```
20/20 ━━━━━━━━━━━━━━━━━━━━ 1s 43ms/step
```




```
Classification Report:
----------------------
              precision    recall  f1-score   support

    NEGATIVE       0.95      0.96      0.95       201
     NEUTRAL       1.00      0.98      0.99       231
    POSITIVE       0.94      0.95      0.95       208

    accuracy                           0.96       640
```

```
   macro avg       0.96      0.96      0.96       640
weighted avg       0.96      0.96      0.96       640
```