Mental Health Classification

Chue Wai Wai Phyo - Group 20

Agenda

- Introduction
- Dataset Selection & Preprocessing
- Implementation & Use of Pre-trained Model using Hugging Face
- Results & insights
- Conclusion
- Reference

01

Introduction

Topic

The project focuses on disease diagnosis from medical text records, particularly in the field of mental health. It uses machine learning models to classify mental health conditions such as anxiety, depression, stress, bipolar disorder, and suicidal tendencies based on text data.

Application

Once optimized, the model can be applied to real-world scenarios, such as:

- Assisting therapists by identifying potential mental health issues from counseling transcripts.
- Providing initial diagnostic support for mental health helplines.
- Helping healthcare organizations analyze patient records efficiently.

The Model

kingabzpro / **Llama-3.1-8B-Instruct-Mental-Health-Classification** like 5

Question Answering | Transformers | Safetensors | suchintikasarkar/sentiment-analysis-for-mental-health | English | llama | text-generation | mental_health | llama-3.1 | text-generation-inference

Inference Endpoints | License: apache-2.0

Model card | Files and versions | Community | Train | Deploy | Use this model

Edit model card

Llama-3.1-8B-Instruct-Mental-Health-Classification

This model is a fine-tuned version of meta-llama/Meta-Llama-3.1-8B-Instruct on an suchintikasarkar/sentiment-analysis-for-mental-health dataset.

Tutorial

Get started with the new Llama models and customize Llama-3.1-8B-It to predict various mental health disorders from the text by following the Fine-Tuning Llama 3.1 for Text Classification tutorial.

Use with Transformers

```
from transformers import AutoTokenizer,AutoModelForCausalLM,pipeline
import torch

model_id = "kingabzpro/Llama-3.1-8B-Instruct-Mental-Health-Classification"

tokenizer = AutoTokenizer.from_pretrained(model_id)

model = AutoModelForCausalLM.from_pretrained(
        model_id,
        return_dict=True,
```

Downloads last month
525

Safetensors Model size 8.03B params | Tensor type FP16

Inference Examples

Question Answering

This model does not have enough activity to be deployed to Inference API (serverless) yet. Increase its social visibility and check back later, or deploy to Inference Endpoints (dedicated) instead.

Model tree for kingabzpro/Llama-3.1-8B-Instruct-Mental-Health-Classif…

Quantizations 2 models

Spaces using kingabzpro/Llama-3.1-8B-Instruct-Mental-Health-Classif… 7

featherless-ai/try-this-model | Granther/try-this-model | kingabzpro/mental-disorder-classification | emekaboris/try-this-model | SC999/NV_Nemotron | DexterSptizu/Llama-3.1-8B-Instruct-Mental-Health-Classification | JackHoltone/try-this-model

Results

```
100%|██████████| 300/300 [03:24<00:00,  1.47it/s]

Accuracy: 0.913
Accuracy for label Normal: 0.972
Accuracy for label Depression: 0.913
Accuracy for label Anxiety: 0.667
Accuracy for label Bipolar: 0.800
```

Classification Report:

	precision	recall	f1-score	support
Normal	0.92	0.97	0.95	143
Depression	0.93	0.91	0.92	115
Anxiety	0.75	0.67	0.71	27
Bipolar	1.00	0.80	0.89	15
accuracy			0.91	300
macro avg	0.90	0.84	0.87	300
weighted avg	0.91	0.91	0.91	300

Confusion Matrix:

```
[[139   3   1   0]
 [  5 105   5   0]
 [  6   3  18   0]
 [  1   2   0  12]]
```

02

Dataset Selection & Preprocessing

Data Loading & Exploring the dataset before data prepocessing

Objective: The project aimed to predict mental illness category from text data using machine learning models.

The dataset contained 53k mental health records and has 3 columns including unnamed, statement and status. And we can see there are missing values in the statement column. There are seven categories of classification in "status" column.

```
[ ] import pandas as pd

    df = pd.read_csv("hf://datasets/AhmedSSoliman/sentiment-analysis-for-mental-health-Combined-Data/sentiment-analysis-for-mental-health-Combined Data.csv")
```

```
sample_df= df.sample(frac=1/4, random_state = 42).reset_index(drop=True)
```

```
[ ] # Check the structure and summary
    print(sample_df.info())
    print(sample_df.describe())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 13261 entries, 0 to 13260
Data columns (total 3 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Unnamed: 0  13261 non-null  int64
 1   statement   13155 non-null  object
 2   status      13261 non-null  object
dtypes: int64(1), object(2)
memory usage: 310.9+ KB
None
         Unnamed: 0
count  13261.000000
mean   26517.650856
std    15389.528523
min        1.000000
25%    13066.000000
50%    26719.000000
75%    39827.000000
max    53033.000000
```

```
#Checking how the data look like
sample_df.head()
```

	Unnamed: 0	statement	status
0	22261	Just as the the title says. I feel like one is...	Depression
1	41400	a blackened sky encroached tugging behind it m...	Depression
2	20065	It gives you insomnia, which in turn makes you...	Depression
3	30036	Hello all, I'm a new submitter to this channel...	Normal
4	780	Thank God the CB is over for Eid	Normal

```
[ ] sample_df['status'].value_counts()
```

```
[ ]
```

status	count
Normal	4154
Depression	3860
Suicidal	2572
Anxiety	966
Bipolar	711
Stress	697
Personality disorder	301

dtype: int64

```
[ ] sample_df.shape
```

```
(13261, 3)
```

```
[ ] #Dropping the unwanted column
    data = sample_df.drop(columns=["Unnamed: 0"])
```

```
    #Checking the missing data
    data.isnull().sum()
```

	0
statement	106
status	0

dtype: int64

```
    #Dropping the missing data
    data = data.dropna()
```

```
[ ] #Checking the duplicated data
    data.duplicated().sum()
```

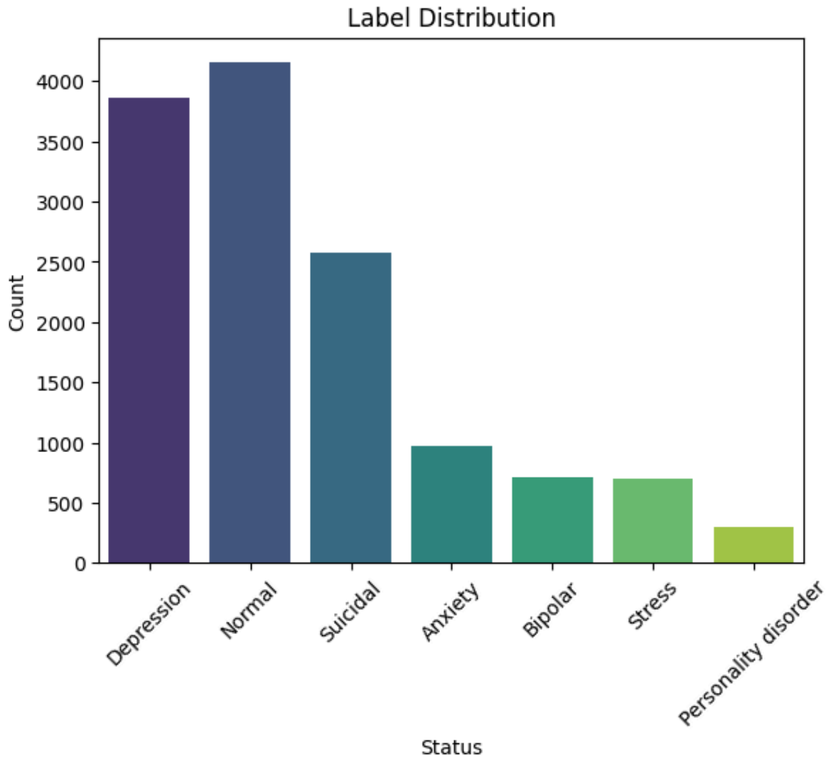
```
129
```

```
[ ] #Dropping the duplicated data
    data.drop_duplicates(inplace=True)
```

```
[ ] #Checking the Final Shape
    data.shape
```

```
(13026, 2)
```

```
[ ] #Resets the index of the DataFrame after dropping rows.
    data = data.reset_index(drop=True)
```




```
def remove_stop_words(text):
    # Tokenize the text and filter out stop words
    words = [word for word in text.split() if word.lower() not in stop_words]
    return ' '.join(words)
def lemmatize_text(text):
    doc = nlp(text)  # Process text with SpaCy
    lemmatized_text = ' '.join([token.lemma_ for token in doc if not token.is_punct])  # Skip punctuation
    return lemmatized_text
def stem_word(text):  # Stemming
    stemmer = nltk.stem.PorterStemmer()
    words = [stemmer.stem(word) for word in text.split()]
    return ' '.join(words)
def clean_text(text):
    #Check if text is string type before applying re.sub
    if isinstance(text, str):
        text = re.sub(r"http\S+|www\S+|https\S+", '', text, flags=re.MULTILINE)  # Remove URLs
        text = re.sub(r'\@\w+|\#', '', text)  # Remove mentions and hashtags
        text = re.sub(r"[^a-zA-Z\s]", '', text)  # Remove special characters
        return text.lower().strip()  # Convert to lowercase and strip whitespace
    else:
        #Handle non-string values (e.g., NaN) by returning an empty string or a placeholder
        return ""

def preprocess_text(text):
    """Apply all preprocessing steps to the text."""
    text = clean_text(text)            # Step 1: Clean the text
    text = remove_stop_words(text)     # Step 2: Remove stop words
    text = lemmatize_text(text)        # Step 3: Lemmatize text
    text = stem_word(text)             # Step 4: Apply stemming
    return text
# Apply preprocessing to the DataFrame
df = sample_df.copy()
for i, sentence in enumerate(tqdm(sample_df["statement"], desc="Processing Text")):
    df.loc[i, "statement"] = preprocess_text(sentence)
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
/usr/local/lib/python3.10/dist-packages/spacy/util.py:1740: UserWarning: [W111] Jupyter notebook detected:
  warnings.warn(Warnings.W111)
Processing Text: 100%|██████████| 13261/13261 [03:15<00:00, 67.85it/s]
```

03

Implementation & Use of Pre-trained Model using Hugging Face

```python
if torch.cuda.is_available():
    print(f"PyTorch GPU: {torch.cuda.get_device_name(0)}")
else:
    print("PyTorch GPU not available.")

print(f"TensorFlow GPUs: {len(tf.config.list_physical_devices('GPU'))}")
spacy.prefer_gpu()
print("SpaCy is using GPU:", spacy.require_gpu())

# PyTorch Example
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
tensor = torch.randn(3, 3).to(device)
print("PyTorch Tensor:", tensor)

# TensorFlow Example
with tf.device('/GPU:0'):
    result = tf.matmul(tf.constant([[1.0, 2.0]]), tf.constant([[3.0], [4.0]]))
print("TensorFlow Result:", result)

# Hugging Face Example
tokenizer = AutoTokenizer.from_pretrained("bert-base-uncased")
model = AutoModel.from_pretrained("bert-base-uncased").to(device)
inputs = tokenizer("Hello, GPU!", return_tensors="pt").to(device)
outputs = model(**inputs)
print("Hugging Face Output:", outputs.last_hidden_state.shape)

# SpaCy Example
nlp = spacy.load("en_core_web_sm")
doc = nlp("GPU-powered text processing!")
print("SpaCy Tokens:", [token.text for token in doc])
```

```
PyTorch GPU: Tesla T4
TensorFlow GPUs: 1
SpaCy is using GPU: True
PyTorch Tensor: tensor([[-0.4837, -1.0205, -1.0612],
        [-0.9411, -1.4880,  0.3098],
        [-1.1435,  0.8090,  0.9276]], device='cuda:0')
TensorFlow Result: tf.Tensor([[11.]], shape=(1, 1), dtype=float32)
Hugging Face Output: torch.Size([1, 7, 768])
SpaCy Tokens: ['GPU', '-', 'powered', 'text', 'processing', '!']
```

```python
# Downloading model
model_id = "kingabzpro/Llama-3.1-8B-Instruct-Mental-Health-Classification"

tokenizer = AutoTokenizer.from_pretrained(model_id)

model = AutoModelForCausalLM.from_pretrained(
    model_id,
    return_dict=True,
    low_cpu_mem_usage=True,
    torch_dtype=torch.float16,
    device_map="auto",
    trust_remote_code=True,
)


text = "I can't sleep at night. I think about my past decisions and blame myself for it."
prompt = f"""Classify the text into Normal, Depression, Suicidal, Anxiety, Bipolar, Stress,
Personality disorder and return the answer as the corresponding mental health disorder label.
text: {text}
label: """.strip()

# Update: Using 'generate' directly instead of pipeline
# We call 'generate' method directly instead of using pipeline
# with 'text-generation' task which appears to be causing issues with 'prefix'.
input_ids = tokenizer(prompt, return_tensors="pt").input_ids.to(model.device)
outputs = model.generate(input_ids, max_new_tokens=2, do_sample=True, temperature=0.1)
generated_text = tokenizer.decode(outputs[0], skip_special_tokens=True)

print(generated_text)
```

```
label: Depression
```

04

Results & insights

```
[ ] from sklearn.metrics import ConfusionMatrixDisplay
    from sklearn.metrics import accuracy_score, classification_report
```

```
[ ] y_pred = df["statement"][:2000].apply(get_label) # Only predict the first 2000 values
```

```
[ ] # Replace incorrect predictions
    y_pred = ["Suicidal" if label == "Suic" else label for label in y_pred]
```

```
[ ] # Accuracy check
    print("Accuracy score:", accuracy_score(sample_df["status"][:2000], y_pred))
    print("Classification report: \n", classification_report(sample_df["status"][:2000], y_pred))
```

```
Accuracy score: 0.6405
Classification report:
                      precision    recall  f1-score   support

             Anxiety       0.68      0.67      0.67       138
             Bipolar       0.74      0.65      0.69        99
          Depression       0.55      0.81      0.65       613
              Normal       0.71      0.97      0.82       609
Personality disorder       1.00      0.04      0.07        52
              Stress       0.94      0.14      0.25       105
            Suicidal       0.84      0.05      0.10       384

            accuracy                           0.64      2000
           macro avg       0.78      0.48      0.47      2000
        weighted avg       0.70      0.64      0.57      2000
```

Accuracy:

The overall accuracy of the model is 64.05%.

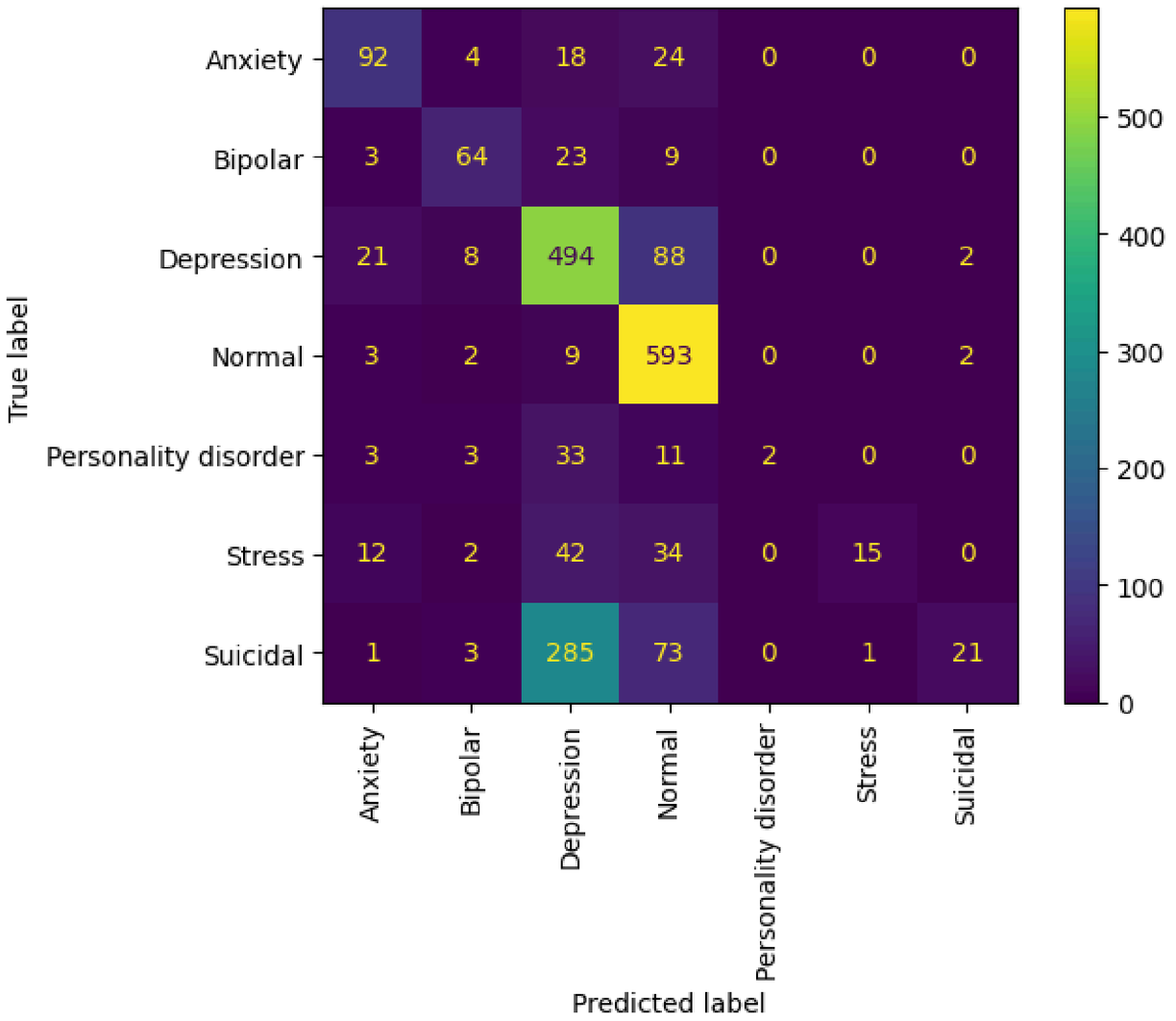
Classification Report Insights:

Precision: It focuses on how "accurate" the model's positive predictions are.

Recall: It focuses on how well the model finds all the actual positive cases.

f1-score: F1-score is high only if both precision and recall are high.

If one of these is low (e.g., high precision but low recall), the F1-score will also be low, reflecting that the model is either overpredicting or underpredicting positives.



Confusion Matrix Observations:

- **"Normal" and "Depression" dominate predictions:**"Normal" has the highest number of correct predictions (593), and "Depression" is often correctly classified (494).
- **Severe misclassifications for "Suicidal":** Most "Suicidal" cases are misclassified as "Depression" (285 cases).
- **Poor performance on minor classes:** Classes like "Personality disorder" and "Stress" suffer from low recall and high misclassification rates, likely due to class imbalance.

05

Conclusion

- **The model shows reasonable accuracy (64.05%)**
- **Fix Class Imbalance:** The model needs more data for smaller groups like "Personality disorder" and "Stress" or adjustments to treat all groups more equally.
- **Improve Differentiation:** The model confuses similar categories, like "Suicidal" and "Depression," so it may need better input features or more detailed training examples.
- **Focus on Smaller Groups:** We need to make sure the model recognizes less common categories better, not just the big ones like "Normal" or "Depression."

06

Reference

Hamza, R. (2024). *Large Language Models Driven Projects in the Real World* [Lecture]. Python for Data Science, Tokyo International University.

Hamza, R. (2024). *Sentiment Analysis* [Lecture]. Python for Data Science, Tokyo International University.

Soliman, A. S. (n.d.). *Sentiment analysis for mental health: Combined data* [Dataset]. Hugging Face. Retrieved from https://huggingface.co/datasets/AhmedSSoliman/sentiment-analysis-for-mental-health-Combined-Data

Kingabzpro. (n.d.). *Llama-3.1-8B-Instruct-Mental-Health-Classification* [Machine learning model]. Hugging Face. Retrieved from https://huggingface.co/kingabzpro/Llama-3.1-8B-Instruct-Mental-Health-Classification

OpenAI. (2024). *ChatGPT* (November 2024 version) [Large language model]. Retrieved from https://chat.openai.com/

Thank You!

Any Questions?