TASK REPORT

By:- Sreeram Vipparla

Task 1

TODO:

- Apply object detection algorithms to identify various elements within the meme images.
- Catalogue the types of objects detected and analyse their frequency and distribution across the dataset.

For this task I have used the YOLOv8 model to classify the images and catalogue the frequency.

1)Exploratory Analysis on the training data

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8500 entries, 0 to 8499
Data columns (total 4 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   id      8500 non-null   int64
 1   img     8500 non-null   object
 2   label   8500 non-null   int64
 3   text    8500 non-null   object
dtypes: int64(2), object(2)
memory usage: 265.8+ KB
```

```
[ ]  data.nunique()
```

```
id       8500
img      8500
label       2
text     7072
dtype: int64
```

Here, the text label only has 7072 unique values of 8500 values

.

Furthermore, I did analysis to find the exact duplicates in the text column

```
exact_duplicates=duplicates[duplicates.duplicated(subset='text',keep=False)]
exact_duplicates
```

	id	img	label	text
5635	52407	img/52407.png	0	"where did you learn to make kool-aid like tha...
1423	79451	img/79451.png	1	"where did you learn to make kool-aid like tha...
7274	61037	img/61037.png	1	a girl asks her mom, "why am i black and you'r...
3791	1653	img/01653.png	0	a girl asks her mom, "why am i black and you'r...
7064	7239	img/07239.png	0	a head diaper is required when you have shit f...
...	...	...	...	...
4785	60759	img/60759.png	1	you see this pig? it's called muhammad
6005	9432	img/09432.png	0	you've been hungerstruck!!
6302	36789	img/36789.png	0	you've been hungerstruck!!
3512	23415	img/23415.png	0	your purchase of $19.99 comes to $21.36 after ...
2530	21693	img/21693.png	1	your purchase of $19.99 comes to $21.36 after ...

554 rows × 4 columns

For the reference, here are the two images with same text

```
from google.colab.patches import cv2_imshow
import cv2

image_path3="/content/drive/MyDrive/hateful_memes/img/52407.png"
image_path4="/content/drive/MyDrive/hateful_memes/img/79451.png"
img3=cv2.imread(image_path3)
img4=cv2.imread(image_path4)

img3_resized=cv2.resize(img3, (300, 300))
img4_resized=cv2.resize(img4, (300, 300))

concatenated_img=cv2.hconcat([img3_resized, img4_resized])
cv2_imshow(concatenated_img)
```

But here the image on the left has a label 0(Not-Hateful) and the image on the right has a label 1(Hateful).
So based on this we also need to understand what is going on in the images.

For the image classification, I have used the YOLOv8 model and have resized the image to 224 x 224 to make the process faster. On performing the object using YOLOv8.m model,

```python
import matplotlib.pyplot as plt

# Convert sorted_object_counts to a dictionary for easy access
sorted_object_counts_dict = dict(sorted_object_counts)

plt.figure(figsize=(40, 40))
plt.bar(sorted_object_counts_dict.keys(),sorted_object_counts_dict.values(),color='lightblue')
plt.xlabel('Object Type')
plt.ylabel('Count')
plt.title('Object Type Counts')
plt.xticks(rotation=45)
plt.show()
```

Here in the bar graph, we cannot infer anything useful in site other than that only the person object has a disproportionate high count around 19500 and the next highest object count is a tie of about 1500.

In order to get more analysis on the frequency, I have decided to subdivide the person class and introduced new factors which are mentioned below:-

1)Gender

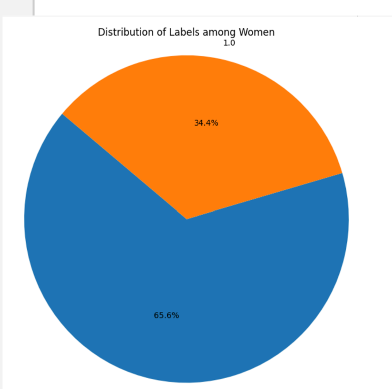
2)Emotion

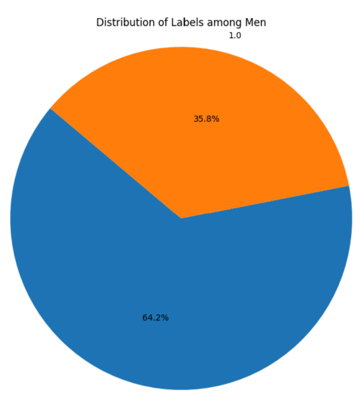
3)Animal

I pretrained a YOLOv8 model for each based on the annotated dataset found on Roboflow. The gender and emotion model has a good enough precision and recall, but the animal detector has an accuracy of about 65% and a recall of 35%.

Given below are the results of the detailed analysis:

1)<u>Gender Analysis</u>

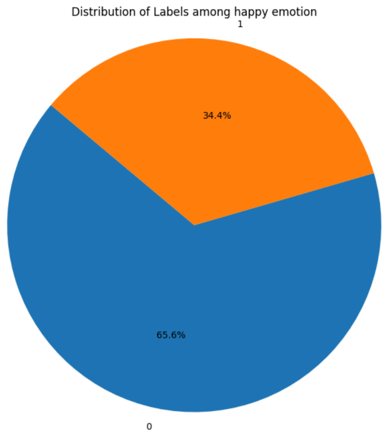


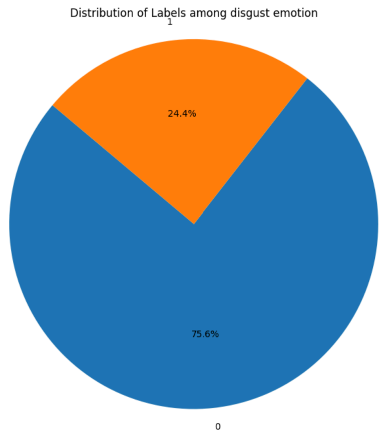


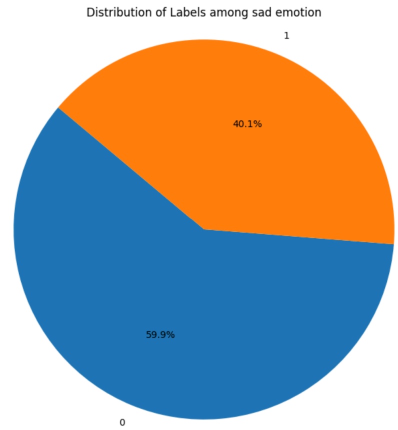
Distribution of Label 0 by Gender

Distribution of Label 1 by Gender

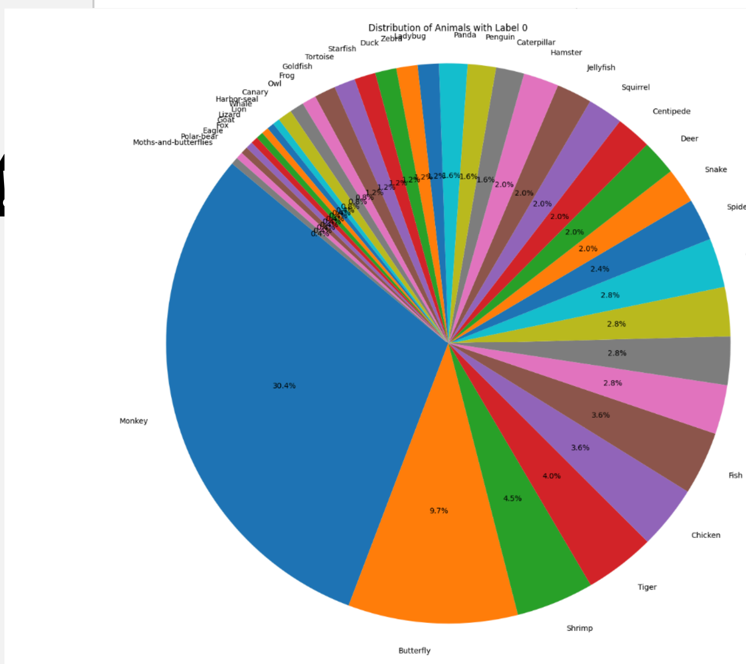
2)Emotion Analysis

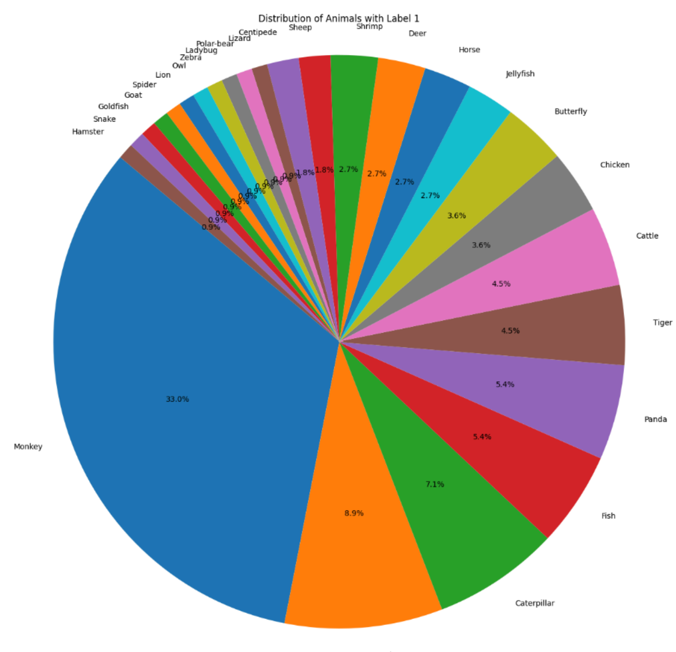






3)Animal Analysis





TASK 2

TODO:

- Determine how text overlays influence the object detection process.

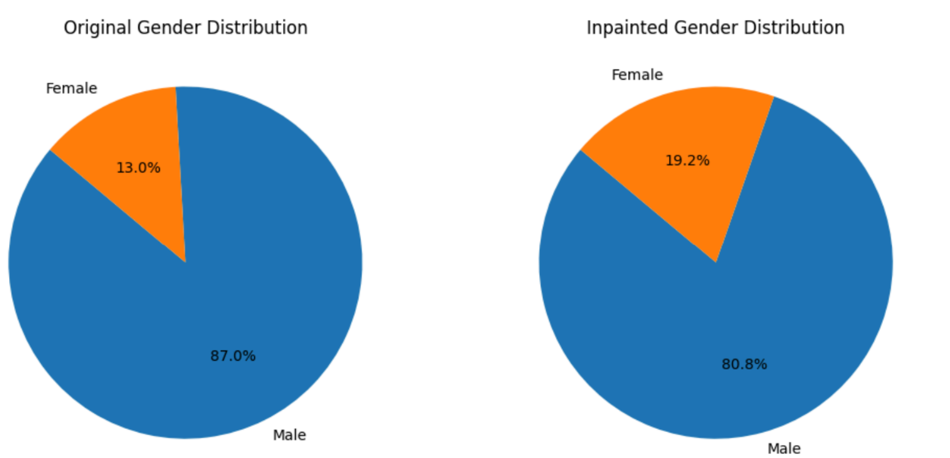
To answer this task, for the removal of text from the images I have used keras_ocr to detect the text and then created a mask to then inpaint the image so that the text is removed.

NOTE:- At the beginning while inpainting I converted the images to grey scale as it greatly reduces the compute time but later when I ran the object detection model, I obtained inaccurate results while checking for emotion and animals. On further research, I found out that the images have to be colored.

When I executed the inpainting algorithm on the color images, my gpu always timed out. For the small dataset analysis, I chose a dataset of 3000 random images from the image dataset.

The results on the custom dataset:

1)Gender Analysis:

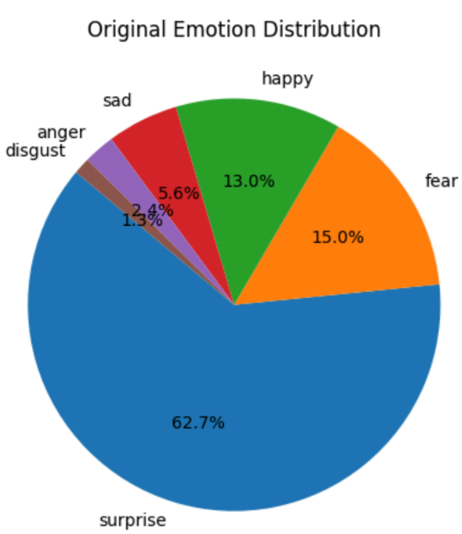


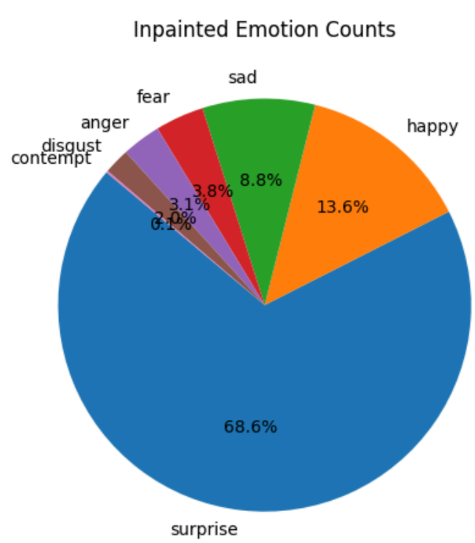
I have created a boolean mask and filtered the merged data frame to identify the values of rows where 'gender_original' and 'inpaint_inpaint' values are not the same.

```
[ ] mask=merged_data['gender_original']!=merged_data['gender_inpaint']
    mismatched_rows=merged_data[mask]

    print(mismatched_rows)
    total_occurrences=len(mismatched_rows)
    print("Total occurrences of mismatched rows:",total_occurrences)
```

```
       image_id gender_original  prob_original gender_inpaint  prob_inpaint
43    21853.png          Female           0.86           Male          0.77
166   52746.png            Male           0.84         Female          0.76
201   06415.png            Male           0.98         Female          0.71
206   45286.png          Female           0.84           Male          0.78
210   52490.png            Male           0.92         Female          0.80
...         ...             ...            ...            ...           ...
1858  42903.png            Male           0.77         Female          0.77
1882  43680.png            Male           0.88         Female          0.71
1953  82731.png            Male           0.71         Female          0.75
2002  52316.png            Male           0.81         Female          0.76
2026  01256.png            Male           0.81         Female          0.74

[71 rows x 5 columns]
Total occurrences of mismatched rows: 71
```

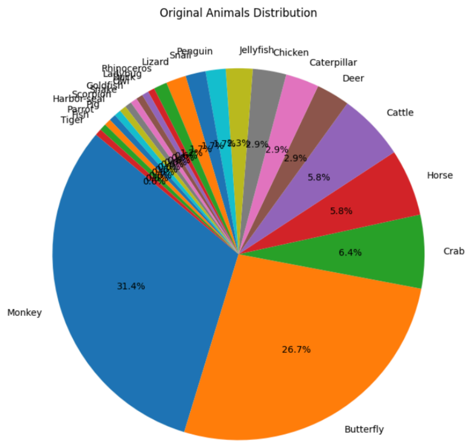
2)Emotion Analysis:

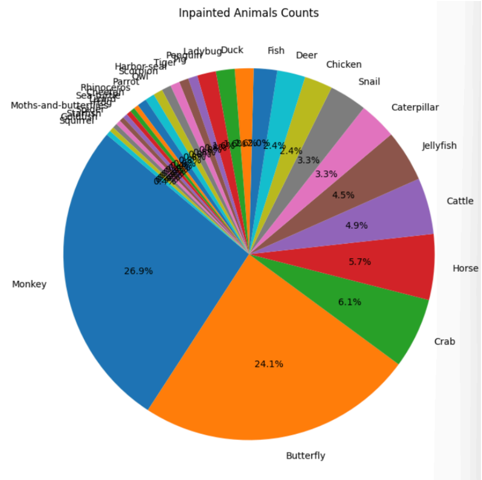





```
mask=merged_data['emotion_original']!=merged_data['emotion_inpaint']
mismatched_rows=merged_data[mask]

print(mismatched_rows)
total_occurrences=len(mismatched_rows)
print("Total occurrences of mismatched rows:",total_occurrences)
```

```
       image_id emotion_original  prob_original emotion_inpaint  prob_inpaint
3     27450.png            happy           0.71        surprise          0.95
37    98720.png              sad           0.75        surprise          0.89
46    82137.png             fear           0.78             sad          0.74
47    82137.png             fear           0.76             sad          0.74
48    41063.png             fear           0.89        surprise          0.79
...         ...              ...            ...             ...           ...
1367  65240.png             fear           0.73        contempt          0.78
1375  43506.png            happy           0.76        surprise          0.88
1378  43506.png         surprise           0.74           happy          0.77
1381  92613.png          disgust           0.80           happy          0.71
1382  92613.png            happy           0.71         disgust          0.81

[255 rows x 5 columns]
Total occurrences of mismatched rows: 255
```





3)Animals Analysis:

```
mask=merged_data['animal_original']!=merged_data['animal_inpaint']
mismatched_rows=merged_data[mask]

print(mismatched_rows)
total_occurrences=len(mismatched_rows)
print("Total occurrences of mismatched rows:",total_occurrences)
```

```
      image_id animal_original  prob_original animal_inpaint  prob_inpaint
127  87341.png            Crab           0.78       Scorpion          0.83
393  70825.png             Owl           0.80       Squirrel          0.80
Total occurrences of mismatched rows:  2
```

TASK-3

TODO:

A classification task to classify the meme is hateful or not.

```
import pandas as pd
from sklearn.utils import resample

print('Original class distribution:',data_train_data['label'].value_counts())

majority_class=data_train_data[data_train_data['label']==0]
minority_class=data_train_data[data_train_data['label']==1]

majority_class_downsampled=resample(majority_class,replace=False,n_samples=len(minority_class),random_state=42)

balanced_data=pd.concat([majority_class_downsampled, minority_class])

print('Balanced class distribution:',balanced_data['label'].value_counts())
```

```
Original class distribution: 0    5481
1    3019
Name: label, dtype: int64
Balanced class distribution: 0    3019
1    3019
Name: label, dtype: int64
```

After analysing the distribution of labels, we find that the label class 1 is the majority class. So to make the trained data balanced I have created a new data frame where both the label classes each have the same value of 3019 followed by the undersampling of the majority class.

Then created the respective pickle files for the text list

```
with open('/content/drive/MyDrive/train_text_data.pickle', 'wb') as f:
    pickle.dump(train_text_data, f)

with open('/content/drive/MyDrive/dev_seen_text.pickle', 'wb') as f:
    pickle.dump(dev_seen_text, f)

with open('/content/drive/MyDrive/test_data_text.pickle', 'wb') as f:
    pickle.dump(test_data_text, f)
```

Similarly, the same has been done to the labels as well as train, val and test images

```
with open('/content/drive/MyDrive/data_train_labels.pickle', 'wb') as f:
    pickle.dump(data_train_labels, f)

with open('/content/drive/MyDrive/dev_seen_labels.pickle', 'wb') as f:
    pickle.dump(dev_seen_labels, f)

with open('/content/drive/MyDrive/test_data_labels.pickle', 'wb') as f:
    pickle.dump(test_data_labels, f)
```

```
[ ] with open('/content/drive/MyDrive/train_image_data.pickle','wb') as f:
        pickle.dump(train_image_data, f)

    with open('/content/drive/MyDrive/val_image_data.pickle','wb') as f:
        pickle.dump(val_image_data, f)

    with open('/content/drive/MyDrive/test_image_data.pickle','wb') as f:
        pickle.dump(test_image_data, f)
```

For the preprocessing of the images, I have applied the InceptionV3 model that is trained on ImageNet to get the features of the images effectively.

```
from keras.applications.inception_v3 import InceptionV3
from keras.applications.inception_v3 import preprocess_input

model = InceptionV3(weights='imagenet')
model_new = Model(model.input, model.layers[-2].output)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/i
96112376/96112376 [==============================] - 1s 0us/step
```

```
[ ] def encode(PIL_img):

        img = PIL_img.resize((299, 299))
        x = image.img_to_array(img)
        x = np.expand_dims(x, axis=0)
        x = preprocess_input(x)

        image_feature_vector = model_new.predict(x)
        image_feature_vector = np.reshape(image_feature_vector, image_feature_vector.shape[1])

        return image_feature_vector
```

This process has been done for each of the train, test and validation images.

MODEL

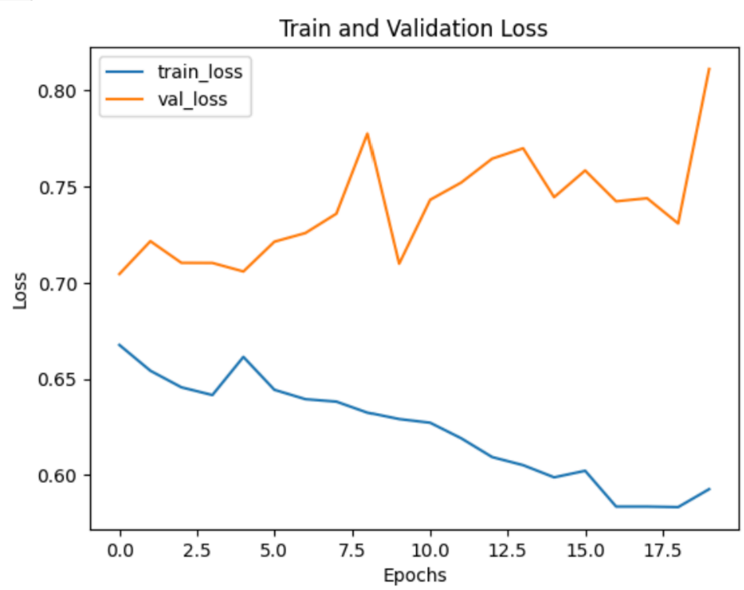
The model chosen here is a Multi-Modal Model, specifically a LSTM

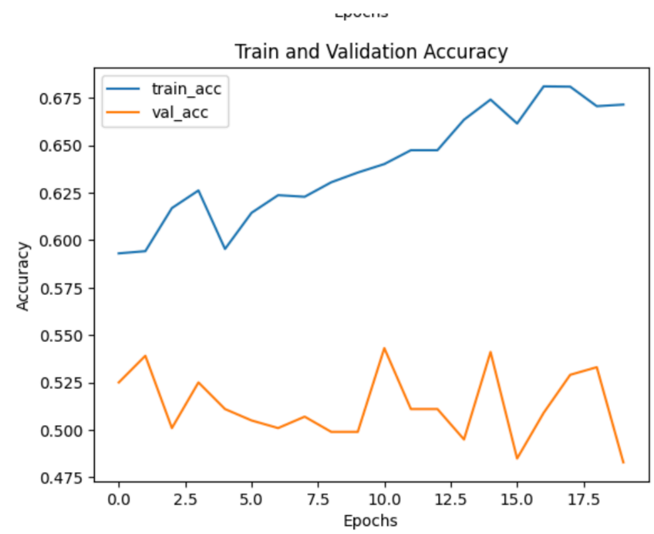
```
Model: "model"
__________________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to
==================================================================================================
 input_1 (InputLayer)           [(None, 2048)]       0           []

 input_2 (InputLayer)           [(None, 768)]        0           []

 dropout (Dropout)              (None, 2048)         0           ['input_1[0][0]']

 reshape (Reshape)              (None, 768, 1)       0           ['input_2[0][0]']

 dense (Dense)                  (None, 1024)         2098176     ['dropout[0][0]']

 dropout_1 (Dropout)            (None, 768, 1)       0           ['reshape[0][0]']

 dense_1 (Dense)                (None, 512)          524800      ['dense[0][0]']

 lstm (LSTM)                    (None, 512)          1052672     ['dropout_1[0][0]']

 add (Add)                      (None, 512)          0           ['dense_1[0][0]',
                                                                  'lstm[0][0]']

 dense_2 (Dense)                (None, 512)          262656      ['add[0][0]']

 dense_3 (Dense)                (None, 256)          131328      ['dense_2[0][0]']

 dense_4 (Dense)                (None, 1)            257         ['dense_3[0][0]']

==================================================================================================
Total params: 4069889 (15.53 MB)
Trainable params: 4069889 (15.53 MB)
Non-trainable params: 0 (0.00 Byte)
__________________________________________________________________________________________________
```

LSTM was particularly chosen for this task because they are well-suited to handle sequential data with long-term dependencies and also have the ability to capture temporal dependencies.

For the loss function I have chosen binary cross entropy along with adam optimizer.

The model has been trained for 20 epochs obtaining a final training accuracy of 67.14% and a value accuracy of 54.309%.


Train and Validation Loss
train_loss
val_loss
Loss
Epochs


Train and Validation Accuracy
train_acc
val_acc
Accuracy
Epochs

TASK 4

TODO:

Try to predict whether or not a meme is toxic, based on the sentiment of the caption. Is the caption enough for this task? Share your performance. What other improvements do you think you could make?.

Here to find the sentiment analysis of the text in each of the dataframe, I preprocessed the text data using the nltk library.

```python
import pandas as pd
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer
import string

nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')


def preprocess_text(text):
    tokens=word_tokenize(text.lower())

    # Remove punctuation
    table=str.maketrans('', '',string.punctuation)
    tokens=[w.translate(table) for w in tokens]

    # Remove remaining tokens that are not alphabetic
    tokens=[word for word in tokens if word.isalpha()]

    # Remove stopwords
    stop_word=set(stopwords.words('english'))
    tokens=[word for word in tokens if not word in stop_words]

    # Lemmatization
    lemmatizer=WordNetLemmatizer()
    tokens=[lemmatizer.lemmatize(word) for word in tokens]


    preprocessed_text=' '.join(tokens)

    return preprocessed_text
```

Since the text data has been cleaned, I performed sentiment analysis on the text and assigned the following numeric values to the following 3 sentiments:

1. Positive: +1
2. Neutral: 0
3. Negative: -1

The label data was imbalanced in the train_data dataframe and I undersampled the majority class label 1.

PERFORMING UNIMODAL ANALYSIS

I used Logistic Regression, Decision Trees and Random Forest to find out the accuracy.

Before running the model, I have used TF-IDF for vectorization of features, extraction and dimensionality reduction.

1. LOGISTIC REGRESSION

On performing Logistic Regression, the following accuracies have been obtained for the validation and test set.

```
Test Unseen Accuracy: 0.6245
Test seen Accuracy: 0.544
Dev seen Accuracy: 0.536
Dev Unseen Accuracy: 0.6203703703703703
```

2. DECISION TREE

On performing Decision Tree, the following accuracies have been obtained for the validation and test set. Additionally, I have set a few hyperparameters where the max depth is 20, minimum sample split is 2 and the minimum sample leaf is 2.

```
Test Unseen Accuracy: 0.62
Test seen Accuracy: 0.543
Dev seen Accuracy: 0.532
Dev Unseen Accuracy: 0.62
```

3. RANDOM FOREST

On performing Random Forest, the following accuracies have been obtained for the validation and test set. Additionally, I have set a few hyperparameters where the number of estimators is 100, maximum depth is 20, minimum sample split is 2 and minimum sample leaf is 2 .

```
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble,
  warn(
Test Unseen Accuracy: 0.6215
Test Seen Accuracy: 0.545
Dev Seen Accuracy: 0.538
Dev Unseen Accuracy: 0.6203703703703703
```

OVERVIEW

On analysis of the accuracy obtained, it can be inferred that uni modal models are not enough and we have to use a multi modal model along with sentiment analysis to obtain a higher accuracy.

NOTE- I sincerely apologize for not being able to complete the bonus task 4 since I was short on time.

Improvements

1) The multi modal can be constructed using sentiment analysis for getting a high accuracy.