Lab 2 Kaggle Competition Report: Hu Zhuochen (NCCU 113308106)

Part I: Data Preprocessing

Step 1: Extract tweet content with user ID

The first thing I did is to process the tweets stored in JSON format by extracting essential fields, specifically the tweet ID and text. I then organized these information into a structured format using a Pandas DataFrame, making it suitable for further analysis. This step is crucial because sentiment analysis relies on the content of tweets to determine the emotions.

Code

```python
import json
import pandas as pd

tweets = []

with open('/kaggle/input/dataset/tweets_DM.json', 'r') as file:
    for line in file:
        try:
            json_data = json.loads(line)
            tweet = json_data.get('_source', {}).get('tweet', {})
            text = tweet.get('text', None)
            tweet_id = tweet.get('tweet_id', None)
            if text:
                tweets.append({"tweet_id": tweet_id, "text": text})
        except json.JSONDecodeError as e:
            print(f"Error decoding line: {e}")

df_tweets = pd.DataFrame(tweets)

print(df_tweets)
```

Output

```
         tweet_id                                               text
0        0x376b20  People who post "add me on #Snapchat" must be ...
1        0x2d5350  @brianklaas As we see, Trump is dangerous to #...
2        0x28b412  Confident of your obedience, I write to you, k...
3        0x1cd5b0                Now ISSA is stalking Tasha 😂😂😂 <LH>
4        0x2de201  "Trust is not the same as faith. A friend is s...
...           ...                                                ...
1867530  0x316b80  When you buy the last 2 tickets remaining for ...
1867531  0x29d0cb  I swear all this hard work gone pay off one da...
1867532  0x2a6a4f  @Parcel2Go no card left when I wasn't in so I ...
1867533  0x24faed  Ah, corporate life, where you can date <LH> us...
1867534  0x34be8c              Blessed to be living #Sundayvibes <LH>

[1867535 rows x 2 columns]
```

Step 2: Combining Datasets

With the extracted tweet content, I made use of the common column of "tweet_id" and first merged df_tweets with df_id, the data file containing user ID and identification for train-test split.

<table>
<tr><th>Code</th><th>Output</th></tr>
<tr><td>

```
df_id = pd.read_csv('/kaggle/input/dataset/data_identification.csv')
df_id_tweets = pd.merge(df_tweets, df_id, on='tweet_id', how='left')

print(df_id_tweets)
```

</td><td>

```
         tweet_id                                               text  \
0        0x376b20  People who post "add me on #Snapchat" must be ...
1        0x2d5350  @brianklaas As we see, Trump is dangerous to #...
2        0x28b412  Confident of your obedience, I write to you, k...
3        0x1cd5b0               Now ISSA is stalking Tasha 😂😂😂 <LH>
4        0x2de201  "Trust is not the same as faith. A friend is s...
...           ...                                                ...
1867530  0x316b80  When you buy the last 2 tickets remaining for ...
1867531  0x29d0cb  I swear all this hard work gone pay off one da...
1867532  0x2a6a4f  @Parcel2Go no card left when I wasn't in so I ...
1867533  0x24faed  Ah, corporate life, where you can date <LH> us...
1867534  0x34be8c              Blessed to be living #Sundayvibes <LH>

        identification
0                train
1                train
2                 test
3                train
4                 test
...                ...
1867530           test
1867531           test
1867532           test
1867533          train
1867534          train

[1867535 rows x 3 columns]
```

</td></tr>
</table>

Then, I split the combined dataframe into train and test sets according to their identification before appending the other csv file containing emotions for the training data onto df_train.

Code

```python
df_emo = pd.read_csv('/kaggle/input/dataset/emotion.csv')
df_train = df_id_tweets[df_id_tweets['identification'] == 'train']
df_test = df_id_tweets[df_id_tweets['identification'] == 'test']

df_train_emo = pd.merge(df_train, df_emo, on='tweet_id', how='left')

print(df_train_emo)
print(df_test)
```

Output

```
          tweet_id                                               text  \
0         0x376b20  People who post "add me on #Snapchat" must be ...
1         0x2d5350  @brianklaas As we see, Trump is dangerous to #...
2         0x1cd5b0                Now ISSA is stalking Tasha 😂😂😂 <LH>
3         0x1d755c  @RISKshow @TheKevinAllison Thx for the BEST TI...
4         0x2c91a8           Still waiting on those supplies Liscus. <LH>
...            ...                                                ...
1455558   0x321566  I'm SO HAPPY!!! #NoWonder the name of this sho...
1455559   0x38959e  In every circumtance I'd like to be thankful t...
1455560   0x2cbca6  there's currently two girls walking around the...
1455561   0x24faed  Ah, corporate life, where you can date <LH> us...
1455562   0x34be8c                Blessed to be living #Sundayvibes <LH>

        identification       emotion
0                train  anticipation
1                train       sadness
2                train          fear
3                train           joy
4                train  anticipation
...                ...           ...
1455558          train           joy
1455559          train           joy
1455560          train           joy
1455561          train           joy
1455562          train           joy

[1455563 rows x 4 columns]
```

```
          tweet_id                                               text  \
2         0x28b412  Confident of your obedience, I write to you, k...
4         0x2de201  "Trust is not the same as faith. A friend is s...
9         0x218443  When do you have enough ? When are you satisfi...
30        0x2939d5  God woke you up, now chase the day #GodsPlan #...
33        0x26289a  In these tough times, who do YOU turn to as yo...
...            ...                                                ...
1867525   0x2913b4  "For this is the message that ye heard from th...
1867529   0x2a980e  "There is a lad here, which hath five barley l...
1867530   0x316b80  When you buy the last 2 tickets remaining for ...
1867531   0x29d0cb  I swear all this hard work gone pay off one da...
1867532   0x2a6a4f  @Parcel2Go no card left when I wasn't in so I ...

        identification
2                 test
4                 test
9                 test
30                test
33                test
...                ...
1867525           test
1867529           test
1867530           test
1867531           test
1867532           test

[411972 rows x 3 columns]
```

Part II: Exploratory Data Analysis

In classification problems, it is crucial to identify any class imbalances. This plot revealed a clear imbalance in the training set, with "joy" being the most dominant class and "anger" having the fewest entries. Recognizing this disparity is essential, as it will allow me to account for the unequal distribution of samples during model training to ensure that the classifiers' performance is not skewed or distorted by the inherent imbalance in the dataset.

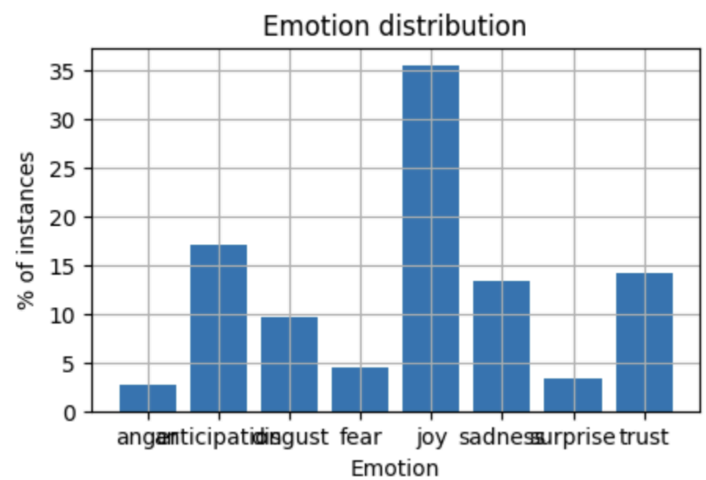
Code

```
train_df.groupby(['emotion']).count()['tweet_id']
```

```
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt

labels = train_df['emotion'].unique()
post_total = len(train_df)
df1 = train_df.groupby(['emotion']).count()['tweet_id']
df1 = df1.apply(lambda x: round(x*100/post_total,3))

fig, ax = plt.subplots(figsize=(5,3))
plt.bar(df1.index,df1.values)

#arrange
plt.ylabel('% of instances')
plt.xlabel('Emotion')
plt.title('Emotion distribution')
plt.grid(True)
plt.show()
```

Output

```
emotion
anger             39867
anticipation     248935
disgust          139101
fear              63999
joy              516017
sadness          193437
surprise          48729
trust            205478
Name: tweet_id, dtype: int64
```



Part III: Feature Engineering

For this task, TF-IDF is chosen as the vectorizer because it is more ideal than the simpler and frequency-based BOW approach due to the nature of tweets, which often include redundant or frequently occurring words such as “the,” “and,” or even platform-specific terms like “RT” (retweet). These words add little to no value for tasks like sentiment analysis but can dominate models that rely on simpler techniques like BOW, which treats all words equally regardless of their contextual importance.

To tokenize the tweet content effectively, the TweetTokenizer from the NLTK is used. This specialized tokenizer is tailored for social media text, which often includes unique elements such as emojis, hashtags, mentions, and contractions. Unlike the standard word_tokenize, which treats all text uniformly, the TweetTokenizer handles these elements appropriately, preserving emojis and hashtags as individual tokens. This ensures that features like sentiment expressed through emojis or contextual clues embedded in hashtags are retained, which is crucial for tasks like sentiment analysis.

The attached code is the final version after several runs, where modifications include dropping tokens starting with special symbols including # and @ as well as numbers. While I am aware of the importance of hashtags (indicating topics) and @ (tagging users) in social media text, the initial token size including these exceeded 100k, which is too computationally intensive. As such, I decided to drop this and only focus on Emojis as well as proper English words (with an assumption that our tweet data is mainly in English).

Code	Output

Code

```python
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
from nltk.tokenize import TweetTokenizer
from nltk.corpus import words
import re
import emoji
import nltk

nltk.download('punkt')
nltk.download('words')
english_vocab = set(words.words())

def is_emoji(token):
    return any(char in emoji.EMOJI_DATA for char in token)

tokenizer = TweetTokenizer()

def tokenize_tweet(tweet):
    # Tokenize the tweet
    tokens = tokenizer.tokenize(tweet)
    filtered_tokens = [
        token for token in tokens
        if re.match(r'^[A-Za-z0-9]+$', token) and token.lower() in english_vocab
        or is_emoji(token)
    ]
    return filtered_tokens

tfidf = TfidfVectorizer(tokenizer=tokenize_tweet, stop_words='english')

X_train_tfidf = tfidf.fit_transform(train_df['text'])
X_test_tfidf = tfidf.transform(test_df['text'])

print(f"Train data TF-IDF shape: {X_train_tfidf.shape}")
print(f"Test data TF-IDF shape: {X_test_tfidf.shape}")
```

```python
feature_names = tfidf.get_feature_names_out()
print(feature_names[500:600])
print(feature_names[32200:32300])
```

Output

```
Train data TF-IDF shape: (1455563, 32340)
Test data TF-IDF shape: (411972, 32340)
```

```
['afterwork' 'afterworld' 'aga' 'agama' 'agape' 'agar' 'agarwal' 'age'
 'aged' 'agee' 'ageless' 'agen' 'agency' 'agenda' 'agent' 'ager'
 'aggrandize' 'aggrandizement' 'aggrandizer' 'aggravate' 'aggravating'
 'aggravation' 'aggregate' 'aggregation' 'aggregator' 'aggress'
 'aggression' 'aggressive' 'aggressively' 'aggressiveness' 'aggressor'
 'aggrieved' 'agha' 'aghast' 'agile' 'agility' 'aging' 'agitate'
 'agitation' 'agitator' 'aglow' 'agnostic' 'ago' 'agog' 'agon' 'agonize'
 'agonizingly' 'agony' 'agora' 'agoraphobia' 'agouti' 'agrarian' 'agree'
 'agreeability' 'agreeable' 'agreed' 'agreeing' 'agreement' 'agricultural'
 'agriculture' 'agrimony' 'agronomy' 'agua' 'aguacate' 'aguinaldo'
 'aguish' 'agust' 'agy' 'ah' 'aha' 'ahead' 'ahem' 'ahimsa' 'ahluwalia'
 'ahoy' 'ahsan' 'ai' 'aid' 'aide' 'aiel' 'ailing' 'ailment' 'aim' 'aiming'
 'aimless' 'aimlessly' 'aint' 'air' 'airbrush' 'aircraft' 'aircrew'
 'airdrop' 'aire' 'airfield' 'airhead' 'airing' 'airlift' 'airliner'
 'airmail' 'airman']
['🛳' '🛴' '🛵' '🛶' '🛷' '🛸' '🤐' '🤑' '🤒' '🤓' '🤔' '🤕' '🤖' '🤗' '🤘' '🤙' '🤚' '🤛'
 '🤜' '🤝' '🤞' '🤟' '🤠' '🤡' '🤢' '🤣' '🤤' '🤥' '🤦' '🤧' '🤨' '🤩' '🤪' '🤫' '🤬' '🤭'
 '🤮' '🤯' '🤰' '🤱' '🤲' '🤳' '🤴' '🤵' '🤶' '🤷' '🤸' '🤹' '🤺' '🤼' '🤽' '🤾' '🥀' '🥁'
 '🥂' '🥃' '🥄' '🥅' '🥇' '🥈' '🥉' '🥊' '🥋' '🥌' '🥐' '🥑' '🥒' '🥓' '🥔' '🥕' '🥖' '🥗'
 '🥘' '🥙' '🥚' '🥛' '🥜' '🥝' '🥞' '🥟' '🥠' '🥡' '🥢' '🥣' '🥤' '🥥' '🥦' '🥧' '🥨' '🥩'
 '🥪' '🥫' '🦀' '🦁' '🦂' '🦃' '🦄' '🦅' '🦆' '🦇']
```

Part IV: Model Training

The process of model training began with a Decision Tree Classifier due to its ease of interpretation and relatively good performance for baseline modeling. Decision trees provide a clear and intuitive structure for understanding how predictions are made, making them an ideal starting point for analyzing complex datasets.

To address computational limitations, dimensionality reduction and sampling were employed. The original dataset consisted of 32,340 tokens and 1,455,563 training entries, which proved too large to handle due to insufficient computational power when the code consistently crashes. To mitigate this, Truncated Singular Value Decomposition was applied to reduce the dimensionality of the TF-IDF feature set to 100 components. Additionally, a subset of 50,000 samples was randomly selected from the training data to further manage resource demands without compromising the model's ability to learn key patterns.

The class imbalance issue identified earlier in the EDA stage was also addressed using class weights. By calculating balanced weights for each emotion class based on their relative frequencies in the sampled training data, the model was adjusted to ensure that minority classes were not overshadowed during training.

However, the score on Kaggle was not very ideal with the Decision Tree Classifier, obtaining merely 0.27.

<table>
<tr><th>Code</th><th>Output</th></tr>
<tr><td><pre>from sklearn.tree import DecisionTreeClassifier
from sklearn.utils.class_weight import compute_class_weight
from sklearn.decomposition import TruncatedSVD
import numpy as np

n_components = 100
svd = TruncatedSVD(n_components=n_components, random_state=42)

X_train_tfidf_reduced = svd.fit_transform(X_train_tfidf)
X_test_tfidf_reduced = svd.transform(X_test_tfidf)

sample_size = 50000
sample_indices = np.random.choice(X_train_tfidf_reduced.shape[0], size=sample_size, replace=False)

X_train_sampled = X_train_tfidf_reduced[sample_indices]
y_train_sampled = train_df['emotion'].iloc[sample_indices]

class_weights = compute_class_weight(class_weight='balanced',
                                     classes=np.unique(y_train_sampled),
                                     y=y_train_sampled)
class_weight_dict = dict(zip(np.unique(y_train_sampled), class_weights))

dt_model = DecisionTreeClassifier(class_weight=class_weight_dict, random_state=42)
dt_model.fit(X_train_sampled, y_train_sampled)

y_pred = dt_model.predict(X_test_tfidf_reduced)
test_df['predicted_emotion'] = y_pred
print(test_df.head())</pre></td>
<td><pre>   tweet_id   text \
2   0x28b412 Confident of your obedience, I write to you, k...
4   0x2de201 "Trust is not the same as faith. A friend is s...
9   0x218443 When do you have enough ? When are you satisfi...
30   0x2939d5 God woke you up, now chase the day #GodsPlan #...
33   0x26289a In these tough times, who do YOU turn to as yo...
...   .
1867525 0x2913b4 "For this is the message that ye heard from th...
1867529 0x2a980e "There is a lad here, which hath five barley l...
1867530 0x316b80 When you buy the last 2 tickets remaining for ...
1867531 0x29d0cb I swear all this hard work gone pay off one da...
1867532 0x2a6a4f @Parcel2Go no card left when I wasn't in so I ...

        identification predicted_emotion
2   test   joy
4   test   joy
9   test   sadness
30   test   joy
33   test   anticipation
...   .
1867525   test   joy
1867529   test   anticipation
1867530   test   joy
1867531   test   joy
1867532   test   disgust

[411972 rows x 4 columns]</pre></td></tr>
</table>

The code was then updated by incorporating multiple classifiers into an ensemble model using a Voting Classifier. This technique combines the strengths of several algorithms to improve overall prediction performance. Specifically, a Logistic Regression, a Random Forest Classifier, and an XGBoost Classifier were trained. The decision to include these 3 models in the ensemble was based on their complementary strengths (Logistic Regression's simplicity and interpretability, Random Forest and XGBoost's efficiency and predictive power) and their track record of delivering strong performance in similar text classification tasks.

To optimize the model's performance while staying within computational constraints, multiple trial-and-error runs were conducted, systematically varying the number of components and sample size. The aim was to strike a balance between capturing sufficient feature complexity and ensuring that the process did not exceed the system's capacity. The number of components for dimensionality reduction using Truncated SVD was incrementally raised to 750 in this final iteration. This adjustment allowed the model to retain more nuanced patterns in the TF-IDF features without overloading the system. Similarly, the sample size was fine-tuned through multiple runs, beginning with smaller subsets of the training data and scaling up to 100,000 samples.

This approach increased the model performance on test data to 0.29.

Code

```
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
import xgboost as xgb
from sklearn.ensemble import VotingClassifier
from sklearn.decomposition import TruncatedSVD
from sklearn.utils.class_weight import compute_class_weight
import numpy as np

n_components = 750
svd = TruncatedSVD(n_components=n_components, random_state=42)

X_train_tfidf_reduced = svd.fit_transform(X_train_tfidf)
X_test_tfidf_reduced = svd.transform(X_test_tfidf)

sample_size = 100000
sample_indices = np.random.choice(X_train_tfidf_reduced.shape[0], size=sample_size, replace=False)

X_train_sampled = X_train_tfidf_reduced[sample_indices]
y_train_sampled = train_df['emotion'].iloc[sample_indices]

class_weights = compute_class_weight(class_weight='balanced',
                                     classes=np.unique(y_train_sampled),
                                     y=y_train_sampled)
class_weight_dict = dict(zip(np.unique(y_train_sampled), class_weights))

ensemble_model = VotingClassifier(estimators=[
    ('lr', LogisticRegression(max_iter=1000, class_weight='balanced', random_state=42)),
    ('rf', RandomForestClassifier(n_estimators=100, max_depth=10, random_state=42, class_weight='balanced')),
    ('xgb', xgb.XGBClassifier(n_estimators=100, max_depth=6, random_state=42, use_label_encoder=False))
], voting='hard')

ensemble_model.fit(X_train_sampled, y_train_sampled)

y_pred_ensemble = ensemble_model.predict(X_test_tfidf_reduced)

test_df['predicted_emotion'] = y_pred_ensemble
test_df[['tweet_id', 'predicted_emotion']].rename(columns={'tweet_id': 'id'}).to_csv('Submissions_ensemble.csv', index=False)

print(test_df)
```