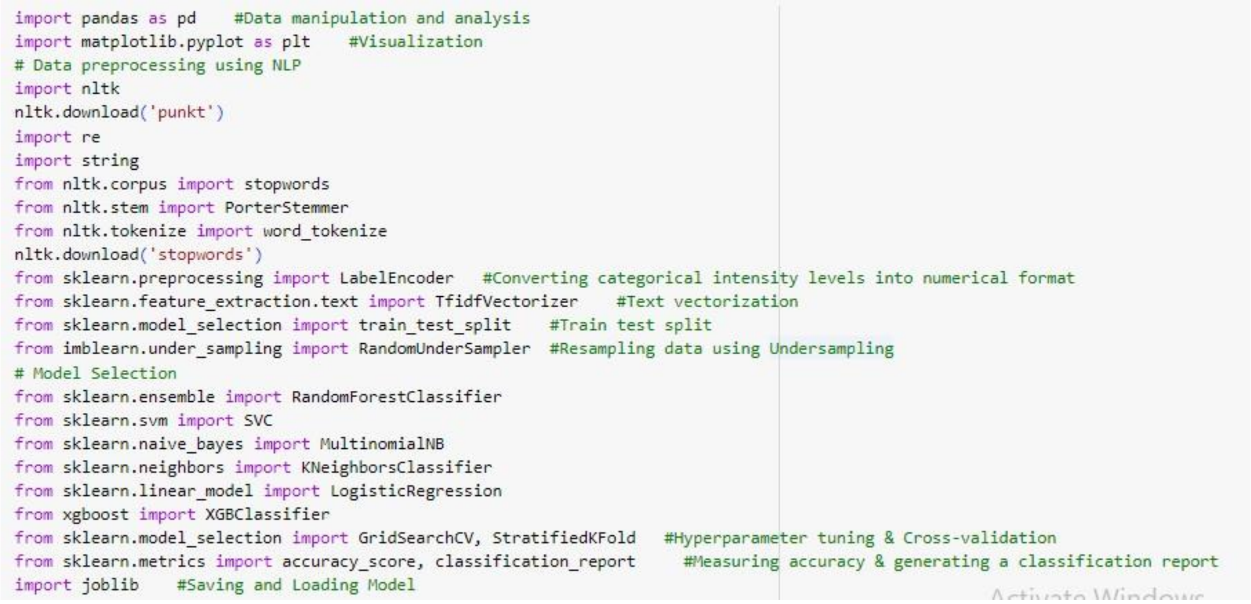
Capstone Project: Intensity Analysis (Build your own model using NLP and Python)

OBJECTIVE: Develop an intelligent system using NLP to predict the intensity in the text reviews. By analyzing various parameters and process data, the system will predict the intensity where its happiness, angriness or sadness. This predictive capability will enable to proactively optimize their processes, and improve overall customer satisfaction.

STEP 1: IMPORT NECESSARY LIBRARIES:

- We have import the all necessary libraries as bellow mentioned:

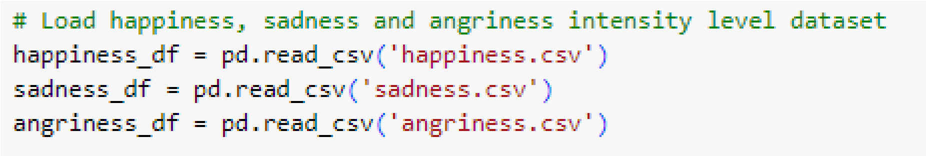


```
import pandas as pd    #Data manipulation and analysis
import matplotlib.pyplot as plt    #Visualization
# Data preprocessing using NLP
import nltk
nltk.download('punkt')
import re
import string
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
from nltk.tokenize import word_tokenize
nltk.download('stopwords')
from sklearn.preprocessing import LabelEncoder    #Converting categorical intensity levels into numerical format
from sklearn.feature_extraction.text import TfidfVectorizer    #Text vectorization
from sklearn.model_selection import train_test_split    #Train test split
from imblearn.under_sampling import RandomUnderSampler  #Resampling data using Undersampling
# Model Selection
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.naive_bayes import MultinomialNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from xgboost import XGBClassifier
from sklearn.model_selection import GridSearchCV, StratifiedKFold   #Hyperparameter tuning & Cross-validation
from sklearn.metrics import accuracy_score, classification_report    #Measuring accuracy & generating a classification report
import joblib    #Saving and Loading Model
```

STEP 2 : DATA EXPLORATION

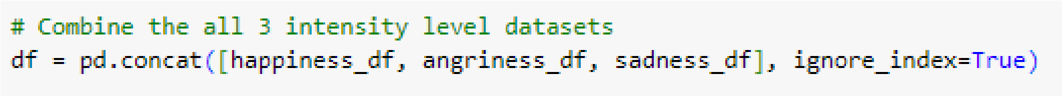
Data Collection:

- We have loaded all 3 CSV (happiness , sadness and angriness) files were given as below :

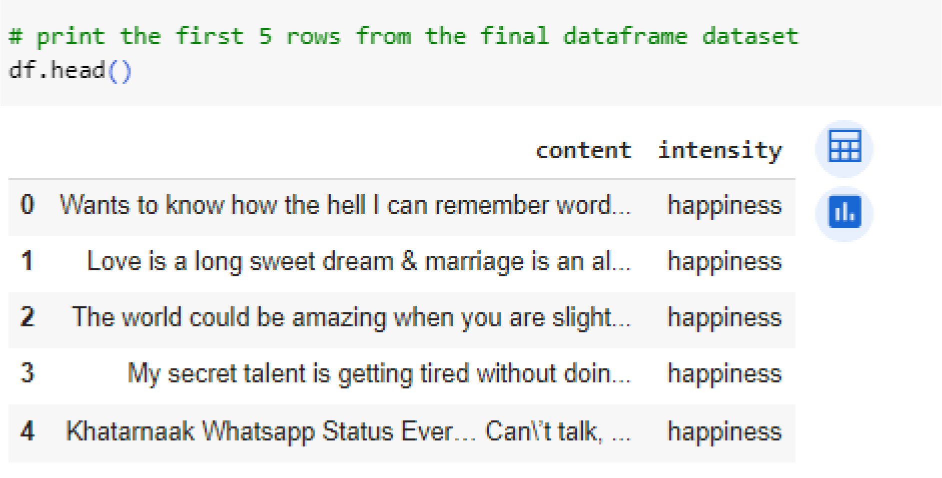


```
# Load happiness, sadness and angriness intensity level dataset
happiness_df = pd.read_csv('happiness.csv')
sadness_df = pd.read_csv('sadness.csv')
angriness_df = pd.read_csv('angriness.csv')
```

- After that we combines all 3 csv files in single file as below:

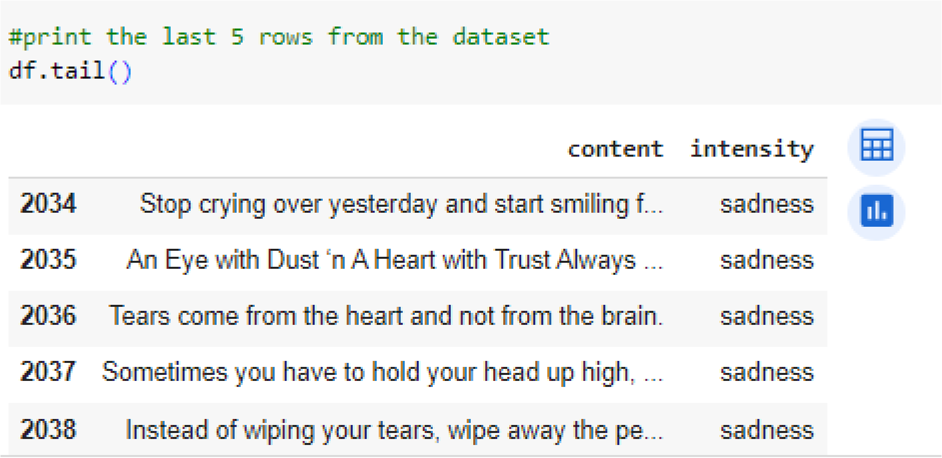


```
# Combine the all 3 intensity level datasets
df = pd.concat([happiness_df, angriness_df, sadness_df], ignore_index=True)
```



```
# print the first 5 rows from the final dataframe dataset
df.head()
```

	content	intensity
0	Wants to know how the hell I can remember word...	happiness
1	Love is a long sweet dream & marriage is an al...	happiness
2	The world could be amazing when you are slight...	happiness
3	My secret talent is getting tired without doin...	happiness
4	Khatarnaak Whatsapp Status Ever… Can\'t talk, ...	happiness

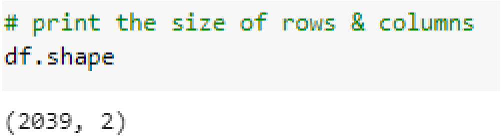


```
#print the last 5 rows from the dataset
df.tail()
```

	content	intensity
2034	Stop crying over yesterday and start smiling f...	sadness
2035	An Eye with Dust ‘n A Heart with Trust Always ...	sadness
2036	Tears come from the heart and not from the brain.	sadness
2037	Sometimes you have to hold your head up high, ...	sadness
2038	Instead of wiping your tears, wipe away the pe...	sadness

Understand the Data:

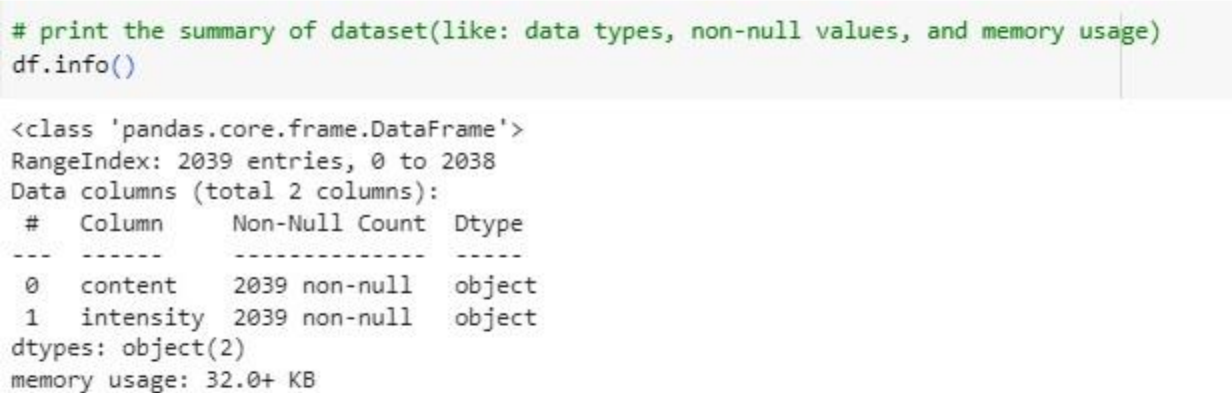
- We check the dataset's size, data types of each variable as below:



```
# print the size of rows & columns
df.shape
```

```
(2039, 2)
```

Insight: We Observed that 2039 rows * 2 columns and 0 Non-null values are available in our combined dataset.



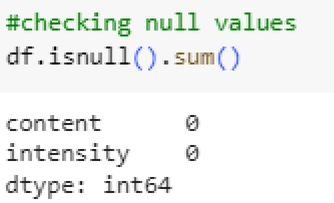
```
# print the summary of dataset(like: data types, non-null values, and memory usage)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2039 entries, 0 to 2038
Data columns (total 2 columns):
 #   Column     Non-Null Count  Dtype 
---  ------     --------------  ----- 
 0   content    2039 non-null   object
 1   intensity  2039 non-null   object
dtypes: object(2)
memory usage: 32.0+ KB
```

Insight: We can observe that both content and intensity columns had Object data types, 2039 non-null entries and DataFrame is using 32.0+ kilobytes of memory.

Identify and Handling Missing Data if available

- We checked the null values as below:



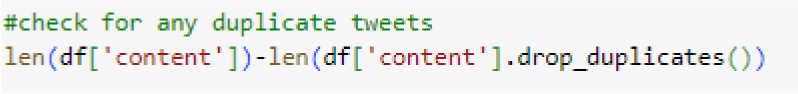
```
#checking null values
df.isnull().sum()
```

```
content      0
intensity    0
dtype: int64
```

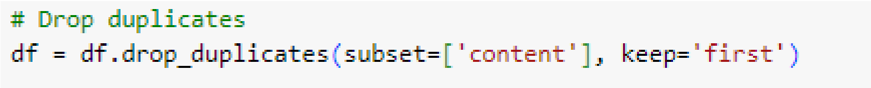
Insight: we observed no null values are present.

Identify and Handling Duplicate data if available

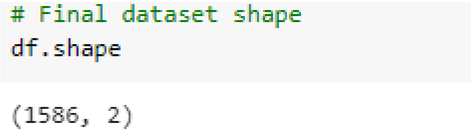
- We have checked the duplicate values as below:



```
#check for any duplicate tweets
len(df['content'])-len(df['content'].drop_duplicates())
```



```
# Drop duplicates
df = df.drop_duplicates(subset=['content'], keep='first')
```



```
# Final dataset shape
df.shape
```

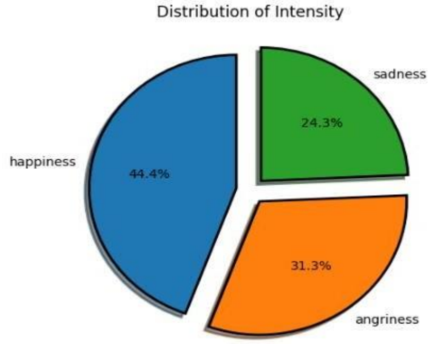
```
(1586, 2)
```

Insight: We observed 453 content rows are duplicates so we drop them and now we get 1586 columns * 2 columns in our final dataset

Visual Exploration:

- Will perform visualizations of frequency distribution of categorical variables by Pie chart as below

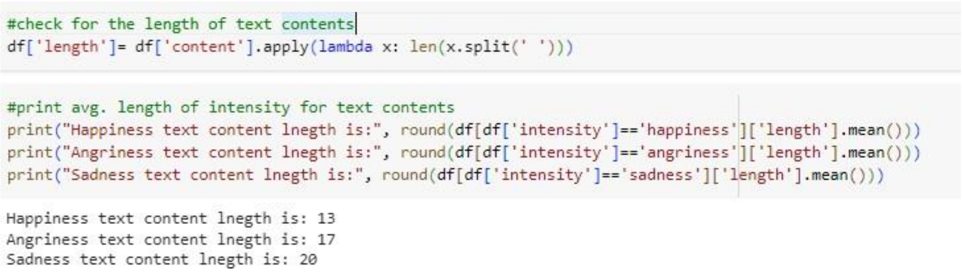
Distribution of Intensity column:



Insight: We observed that intensity distributed in 3 levels, happiness: 44.4% , Sadness: 24.3% and Angriness: 31.3% which means our Intensity levels are imbalanced so we need to deal with it.
Here one class has significantly fewer instances than the other, so we choose undersampling technique to balance this.

Distribution of Text Content column:

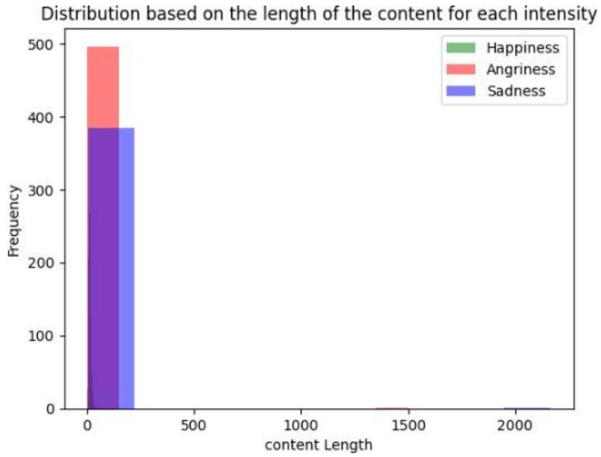
- We check the length of content for each column using lambda function and then we plot the histogram:



```
#check for the length of text contents
df['length']= df['content'].apply(lambda x: len(x.split(' ')))
```

```
#print avg. length of intensity for text contents
print("Happiness text content lnegth is:", round(df[df['intensity']=='happiness']['length'].mean()))
print("Angriness text content lnegth is:", round(df[df['intensity']=='angriness']['length'].mean()))
print("Sadness text content lnegth is:", round(df[df['intensity']=='sadness']['length'].mean()))
```

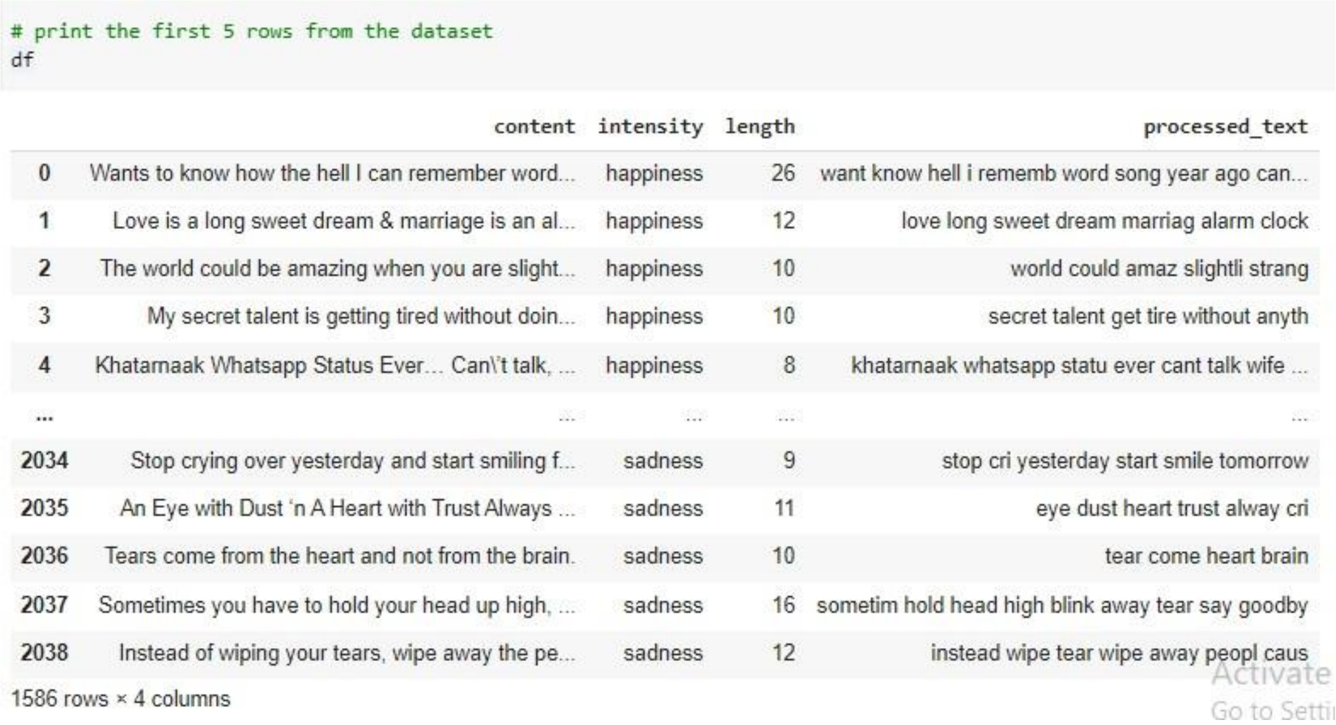
```
Happiness text content lnegth is: 13
Angriness text content lnegth is: 17
Sadness text content lnegth is: 20
```



Insight: We observed that length of Happiness intensity for text content is: 13, length of Angriness intensity for text content is: 17 and length of Sadness intensity for text content is: 20 & then we plot the histogram.

STEP 3: DATA CLEANING AND TOKENIZATION USING NLP

- It’s a crucial step in natural language processing (NLP) to clean and transform raw text data into a format suitable for analysis



```
# print the first 5 rows from the dataset
df
```

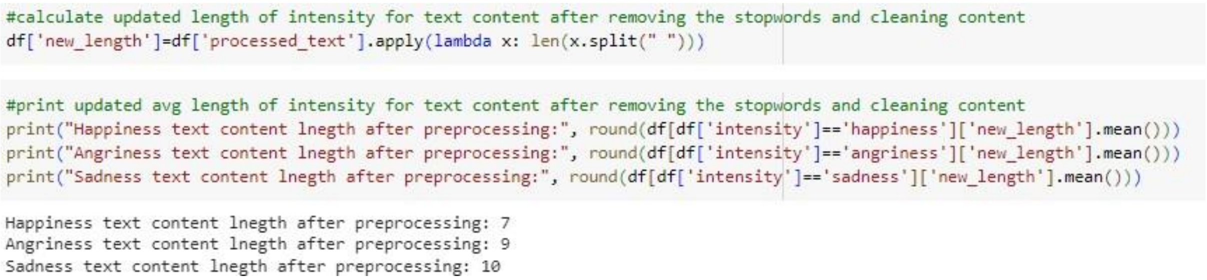
	content	intensity	length	processed_text
0	Wants to know how the hell I can remember word...	happiness	26	want know hell i rememb word song year ago can...
1	Love is a long sweet dream & marriage is an al...	happiness	12	love long sweet dream marriag alarm clock
2	The world could be amazing when you are slight...	happiness	10	world could amaz slightli strang
3	My secret talent is getting tired without doin...	happiness	10	secret talent get tire without anyth
4	Khatarnaak Whatsapp Status Ever… Can\'t talk, ...	happiness	8	khatarnaak whatsapp statu ever cant talk wife ...
...	...	...	...	...
2034	Stop crying over yesterday and start smiling f...	sadness	9	stop cri yesterday start smile tomorrow
2035	An Eye with Dust ‘n A Heart with Trust Always ...	sadness	11	eye dust heart trust alway cri
2036	Tears come from the heart and not from the brain.	sadness	10	tear come heart brain
2037	Sometimes you have to hold your head up high, ...	sadness	16	sometim hold head high blink away tear say goodby
2038	Instead of wiping your tears, wipe away the pe...	sadness	12	instead wipe tear wipe away peopl caus

1586 rows × 4 columns

Insight: We remove non-word characters, special characters, emojis, and unwanted symbols, single characters except for 'a' and 'i', stop words and perform stemming, Convert text into lowercase, Replace multiple spaces with a single space and done tokenization and then we applied this to text content column.

Distribution of Text Content column after text cleaning:

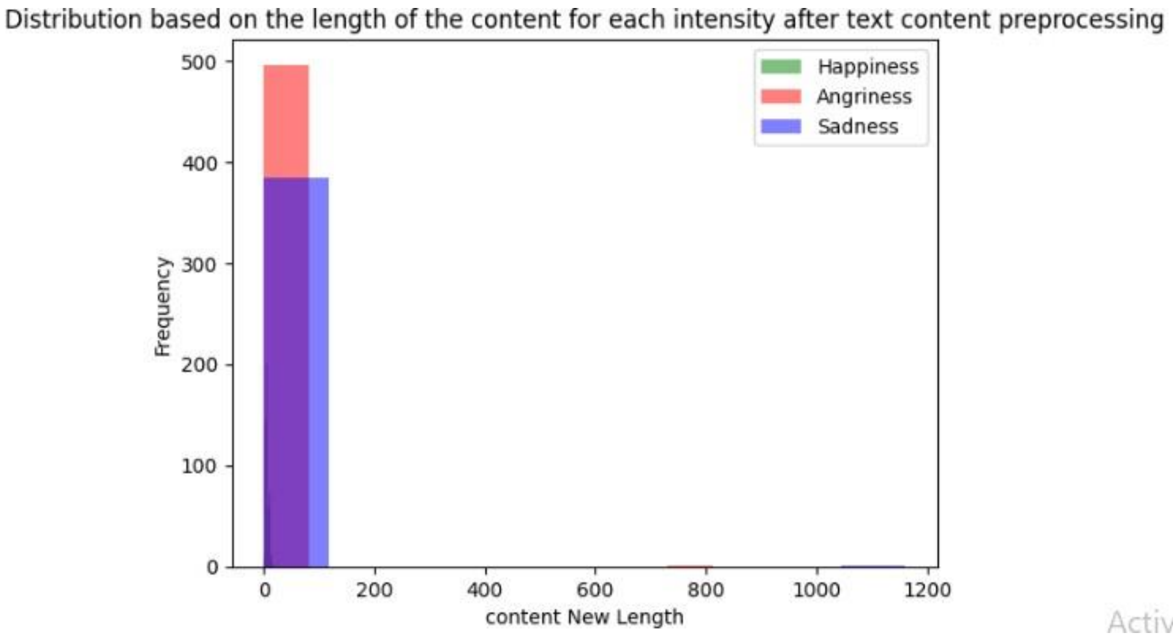
- After cleaning we checked the length of content for each column using lambda function and then we plot the histogram:



```
#calculate updated length of intensity for text content after removing the stopwords and cleaning content
df['new_length']=df['processed_text'].apply(lambda x: len(x.split(" ")))
```

```
#print updated avg length of intensity for text content after removing the stopwords and cleaning content
print("Happiness text content lnegth after preprocessing:", round(df[df['intensity']=='happiness']['new_length'].mean()))
print("Angriness text content lnegth after preprocessing:", round(df[df['intensity']=='angriness']['new_length'].mean()))
print("Sadness text content lnegth after preprocessing:", round(df[df['intensity']=='sadness']['new_length'].mean()))
```

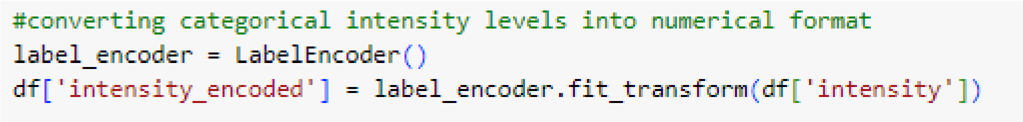
```
Happiness text content lnegth after preprocessing: 7
Angriness text content lnegth after preprocessing: 9
Sadness text content lnegth after preprocessing: 10
```



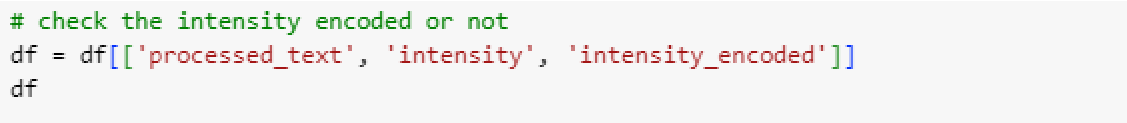
Insight: We observed that after text content preprocessing the length of Happiness intensity for text content is: 7, length of Angriness intensity for text content is: 9 and length of Sadness intensity for text content is: 10 & then we plot the histogram of that.

STEP 4: FEATURE ENGINEERING & FEATURE SELECTION

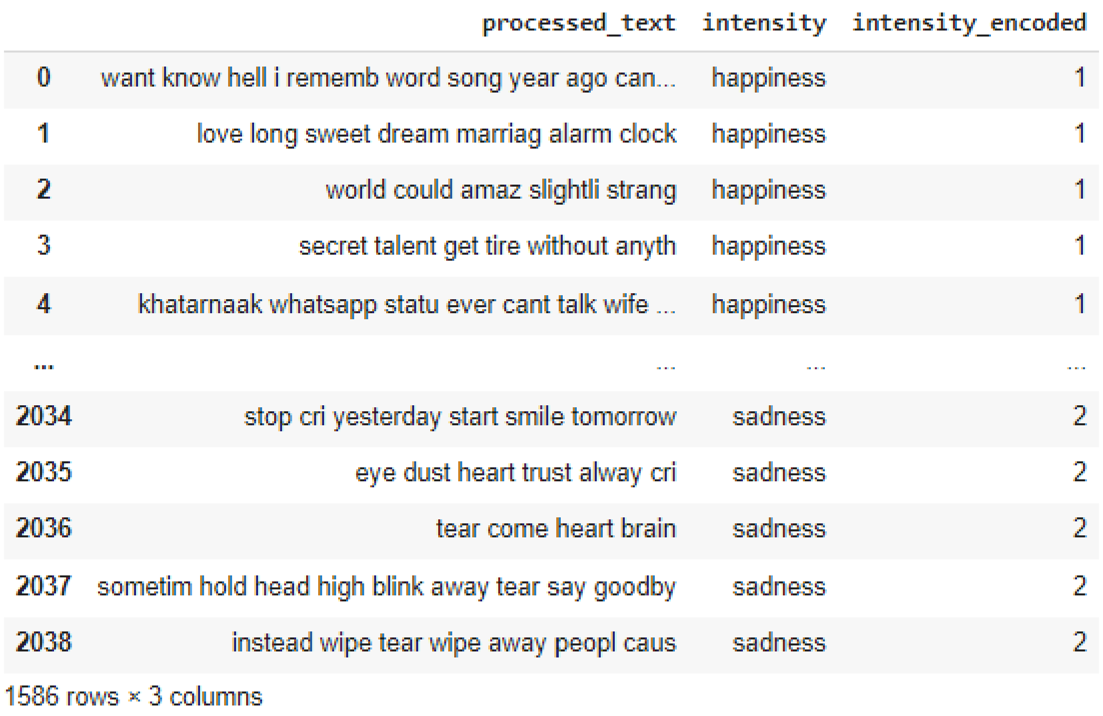
- Feature engineering in the context of TF-IDF vectorization involve creating new features or selecting a subset of existing features to improve model performance
- We transformed the categorical (intensity column) into numerical values



```
#converting categorical intensity levels into numerical format
label_encoder = LabelEncoder()
df['intensity_encoded'] = label_encoder.fit_transform(df['intensity'])
```



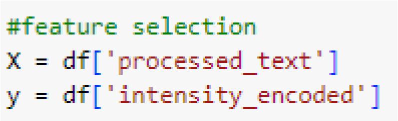
```
# check the intensity encoded or not
df = df[['processed_text', 'intensity', 'intensity_encoded']]
df
```



	processed_text	intensity	intensity_encoded
0	want know hell i rememb word song year ago can...	happiness	1
1	love long sweet dream marriag alarm clock	happiness	1
2	world could amaz slightli strang	happiness	1
3	secret talent get tire without anyth	happiness	1
4	khatarnaak whatsapp statu ever cant talk wife ...	happiness	1
...	...	...	...
2034	stop cri yesterday start smile tomorrow	sadness	2
2035	eye dust heart trust alway cri	sadness	2
2036	tear come heart brain	sadness	2
2037	sometim hold head high blink away tear say goodby	sadness	2
2038	instead wipe tear wipe away peopl caus	sadness	2

1586 rows × 3 columns

Insight: We observed that our intensity levels are successfully encoded as follows: happiness encoded as 1, angriness encoded as 0 and sadness encoded as 2.

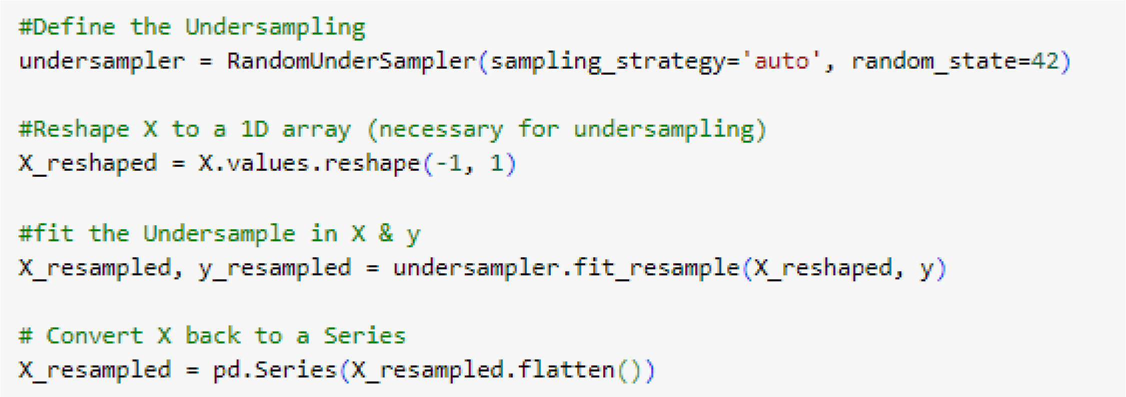


```
#feature selection
X = df['processed_text']
y = df['intensity_encoded']
```

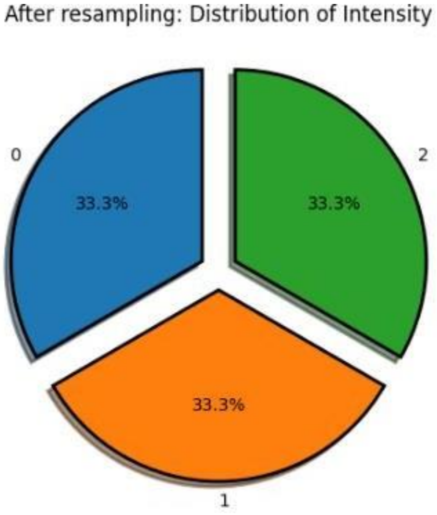
Insight: We done feature selection (X & y) by choosing subset of existing features to improve our model performance.

STEP 5: DEALING WITH IMBALANCED DATASET USING UNDERSAMPLING

- As we see earlier our intensity levels are imbalanced means one class has significantly fewer instances than the other, so we choose undersampling technique to balance this.



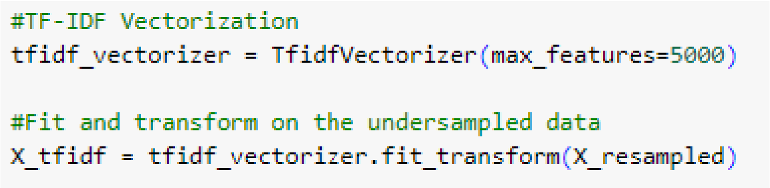
```
#Define the Undersampling
undersampler = RandomUnderSampler(sampling_strategy='auto', random_state=42)

#Reshape X to a 1D array (necessary for undersampling)
X_reshaped = X.values.reshape(-1, 1)

#fit the Undersample in X & y
X_resampled, y_resampled = undersampler.fit_resample(X_reshaped, y)

# Convert X back to a Series
X_resampled = pd.Series(X_resampled.flatten())
```



Insight: As we see now Intensity levels are equally distributed as 33.3% which means our intensity levels are balanced.

TF-IDF Vectorization:

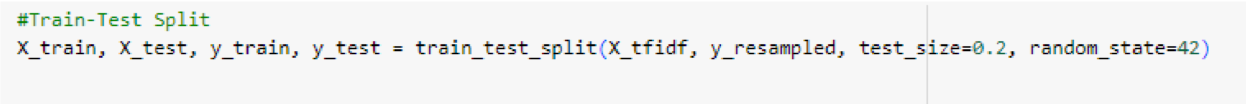
- TF-IDF Vectorization a numerical statistic used in information retrieval and natural language processing to reflect the importance of a word in a document relative to a collection of documents.
- TF-IDF Vectorizer we use to convert the text data into TF-IDF features.



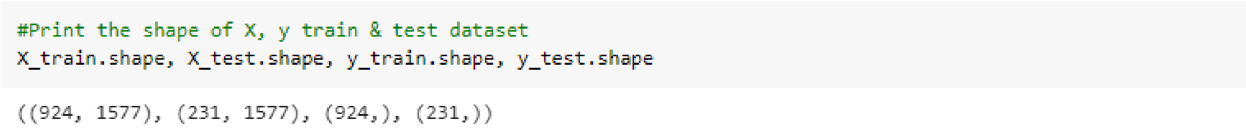
```
#TF-IDF Vectorization
tfidf_vectorizer = TfidfVectorizer(max_features=5000)

#Fit and transform on the undersampled data
X_tfidf = tfidf_vectorizer.fit_transform(X_resampled)
```

Split the Dataset:

- We Split the dataset into 80% training and 20% testing because of small size of datasets (1586*2).



```
#Train-Test Split
X_train, X_test, y_train, y_test = train_test_split(X_tfidf, y_resampled, test_size=0.2, random_state=42)
```



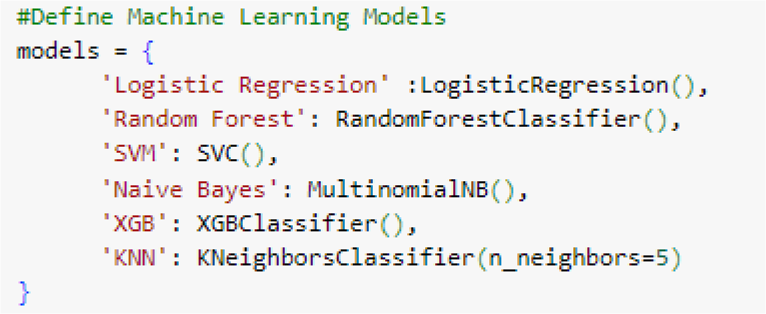
```
#Print the shape of X, y train & test dataset
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

```
((924, 1577), (231, 1577), (924,), (231,))
```

STEP 6: MODEL DEVELOPMENT AND EVALUATION

- We have small dataset size of 1586 columns * 2 rows so we can choose basic ML techniques like Logistic Regression, Random Forest, SVM, Naïve Baye, KNN and XGBoost.

Model selection

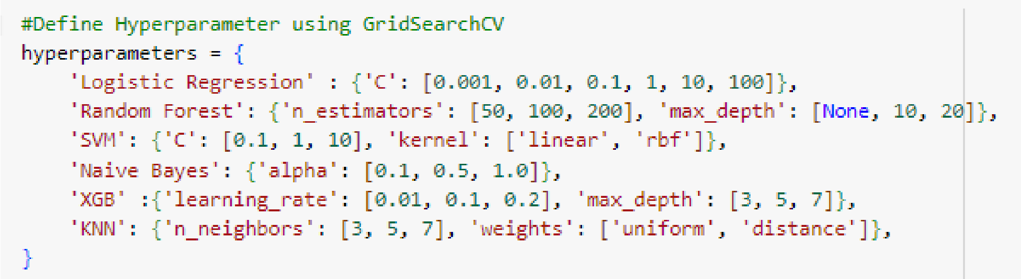


```
#Define Machine Learning Models
models = {
      'Logistic Regression' :LogisticRegression(),
      'Random Forest': RandomForestClassifier(),
      'SVM': SVC(),
      'Naive Bayes': MultinomialNB(),
      'XGB': XGBClassifier(),
      'KNN': KNeighborsClassifier(n_neighbors=5)
}
```

STEP 7: HYPERPARAMETER TUNING & CROSS VALIDATION FOR MODEL IMPROVEMENT

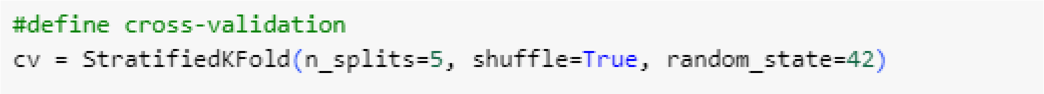
Hyperparameter Tuning

- For better improvement of model we used hyperparameter tuning and cross validation.
- Define the Hyperparameter using GridSearchCV method



```
#Define Hyperparameter using GridSearchCV
hyperparameters = {
    'Logistic Regression' : {'C': [0.001, 0.01, 0.1, 1, 10, 100]},
    'Random Forest': {'n_estimators': [50, 100, 200], 'max_depth': [None, 10, 20]},
    'SVM': {'C': [0.1, 1, 10], 'kernel': ['linear', 'rbf']},
    'Naive Bayes': {'alpha': [0.1, 0.5, 1.0]},
    'XGB' :{'learning_rate': [0.01, 0.1, 0.2], 'max_depth': [3, 5, 7]},
    'KNN': {'n_neighbors': [3, 5, 7], 'weights': ['uniform', 'distance']},
}
```

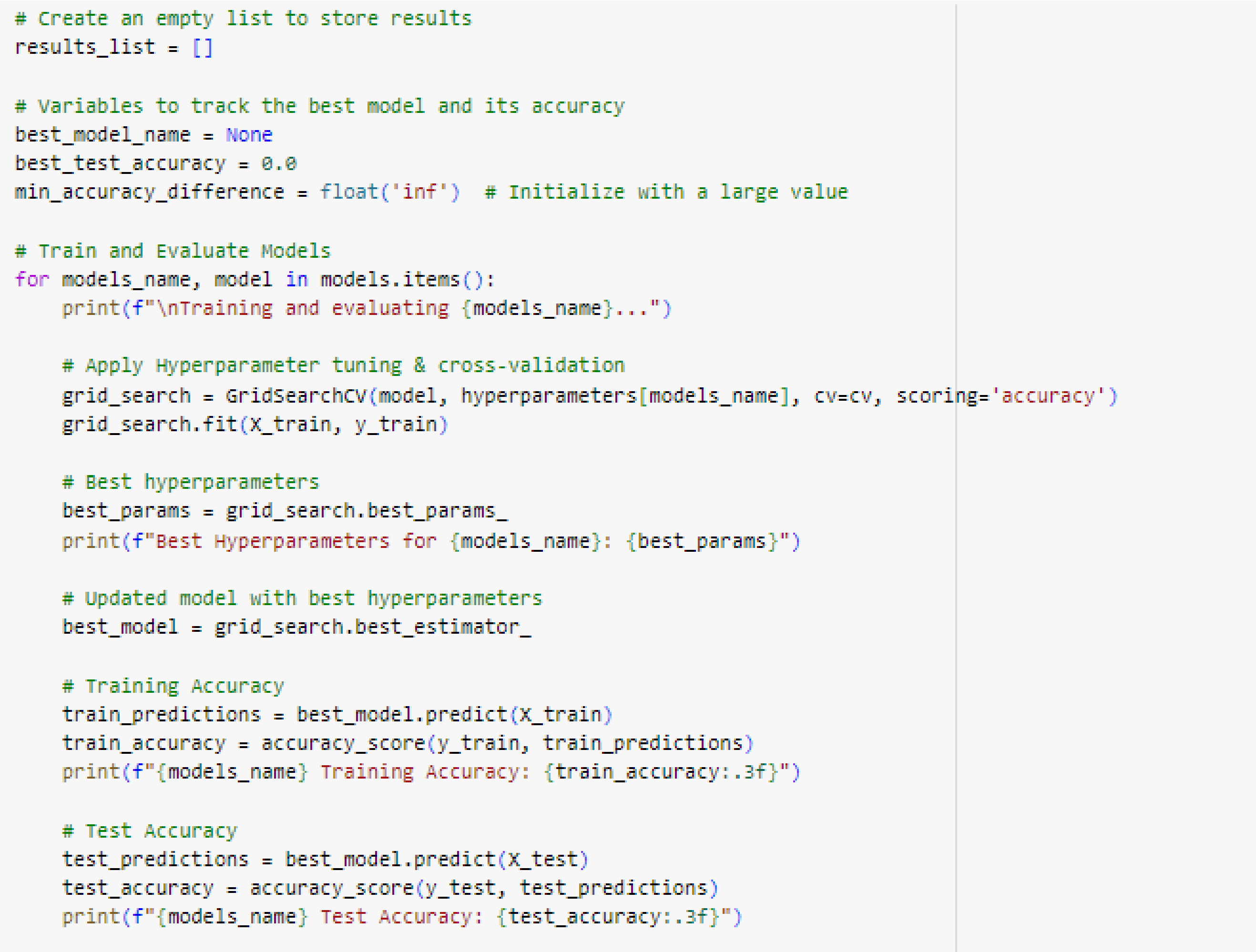
Cross Validation:



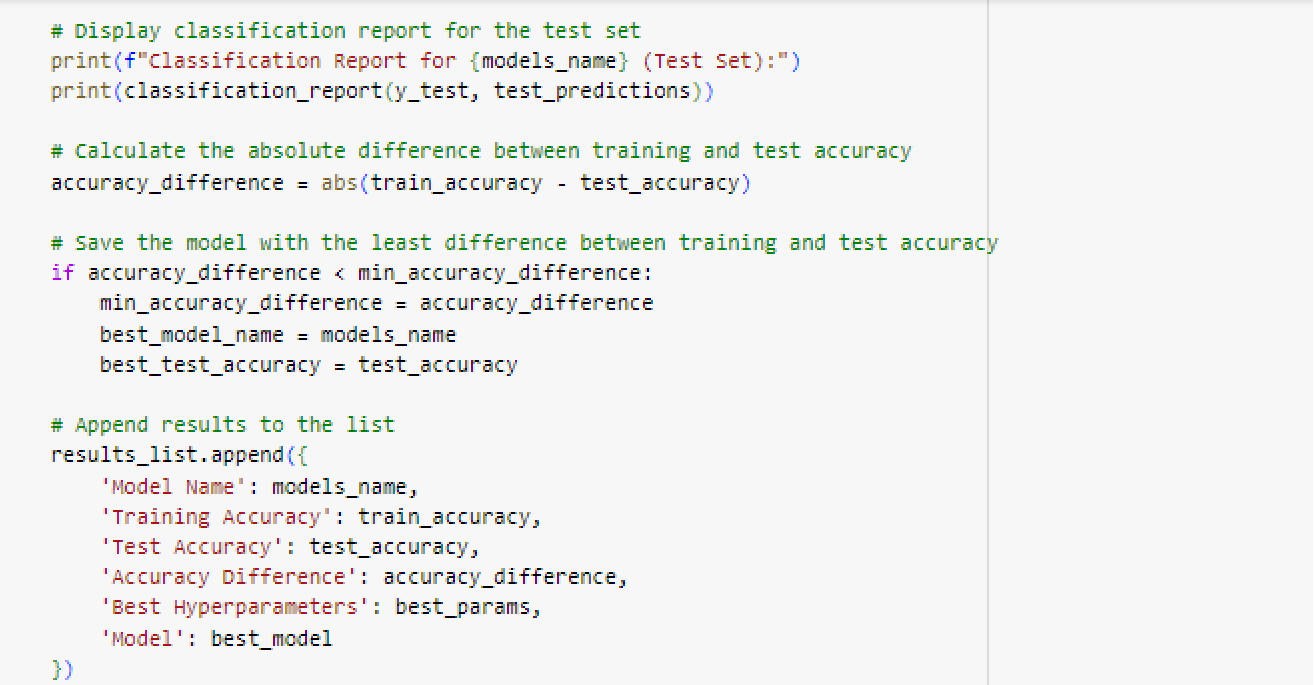
```
#define cross-validation
cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
```

Insight: We choose Stratifield K-Fold cross validation method because we wants better model generalization & reduce the bias.

Train the machine learning models:



```
# Create an empty list to store results
results_list = []

# Variables to track the best model and its accuracy
best_model_name = None
best_test_accuracy = 0.0
min_accuracy_difference = float('inf')  # Initialize with a large value

# Train and Evaluate Models
for models_name, model in models.items():
    print(f"\nTraining and evaluating {models_name}...")

    # Apply Hyperparameter tuning & cross-validation
    grid_search = GridSearchCV(model, hyperparameters[models_name], cv=cv, scoring='accuracy')
    grid_search.fit(X_train, y_train)

    # Best hyperparameters
    best_params = grid_search.best_params_
    print(f"Best Hyperparameters for {models_name}: {best_params}")

    # Updated model with best hyperparameters
    best_model = grid_search.best_estimator_

    # Training Accuracy
    train_predictions = best_model.predict(X_train)
    train_accuracy = accuracy_score(y_train, train_predictions)
    print(f"{models_name} Training Accuracy: {train_accuracy:.3f}")

    # Test Accuracy
    test_predictions = best_model.predict(X_test)
    test_accuracy = accuracy_score(y_test, test_predictions)
    print(f"{models_name} Test Accuracy: {test_accuracy:.3f}")
```

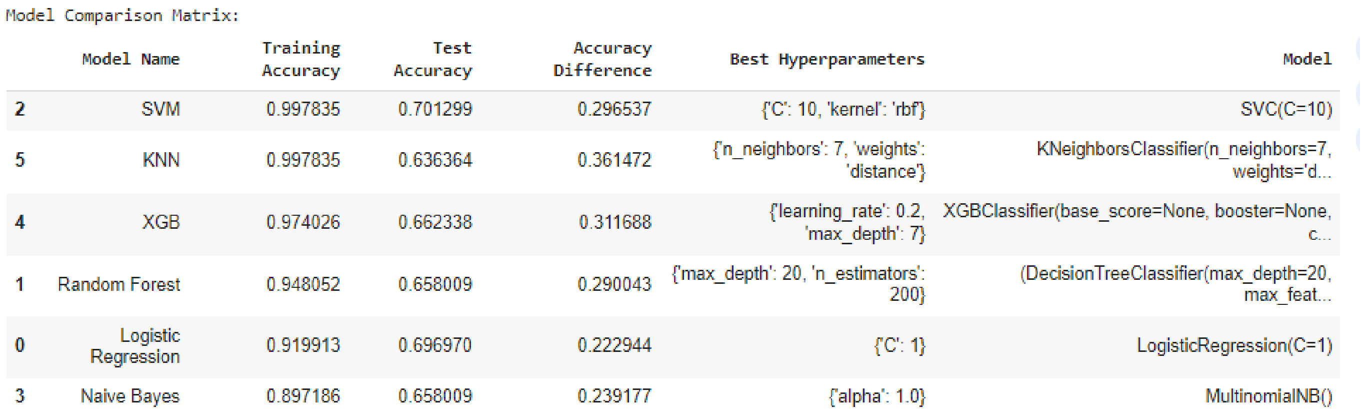


```
    # Display classification report for the test set
    print(f"Classification Report for {models_name} (Test Set):")
    print(classification_report(y_test, test_predictions))

    # Calculate the absolute difference between training and test accuracy
    accuracy_difference = abs(train_accuracy - test_accuracy)

    # Save the model with the least difference between training and test accuracy
    if accuracy_difference < min_accuracy_difference:
        min_accuracy_difference = accuracy_difference
        best_model_name = models_name
        best_test_accuracy = test_accuracy

    # Append results to the list
    results_list.append({
        'Model Name': models_name,
        'Training Accuracy': train_accuracy,
        'Test Accuracy': test_accuracy,
        'Accuracy Difference': accuracy_difference,
        'Best Hyperparameters': best_params,
        'Model': best_model
    })
```

Insight: We trained all 5 selected machine learning models (Logistic Regression, Random Forest, SVM, Naïve Baye, KNN and XGBoost) using best Hyperparameter and Cross validation methods.

Model Evaluation:

- We evaluated the all 5 selected models on the test set and print Training, testing accuracy, accuracy difference andbest Hyperparameters



Model Comparison Matrix:

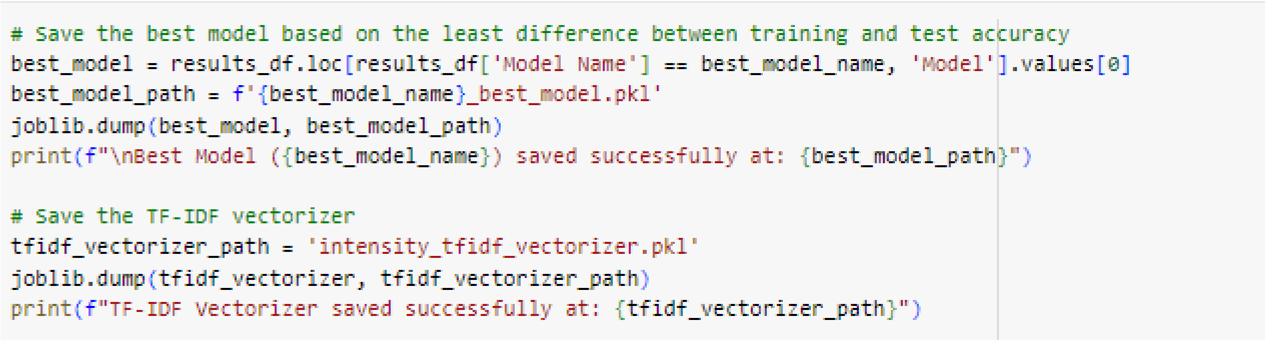
	Model Name	Training Accuracy	Test Accuracy	Accuracy Difference	Best Hyperparameters	Model
2	SVM	0.997835	0.701299	0.296537	{'C': 10, 'kernel': 'rbf'}	SVC(C=10)
5	KNN	0.997835	0.636364	0.361472	{'n_neighbors': 7, 'weights': 'distance'}	KNeighborsClassifier(n_neighbors=7, weights='d...
4	XGB	0.974026	0.662338	0.311688	{'learning_rate': 0.2, 'max_depth': 7}	XGBClassifier(base_score=None, booster=None, c...
1	Random Forest	0.948052	0.658009	0.290043	{'max_depth': 20, 'n_estimators': 200}	(DecisionTreeClassifier(max_depth=20, max_feat...
0	Logistic Regression	0.919913	0.696970	0.222944	{'C': 1}	LogisticRegression(C=1)
3	Naive Bayes	0.897186	0.658009	0.239177	{'alpha': 1.0}	MultinomialNB()

Insight: We can observe clearly from our model comparison matrix that, the model with the highest test accuracy is SVM with 70.13%. However, it's crucial to consider the accuracy difference as well to identify potential overfitting. A lower accuracy difference suggests better generalization to unseen data.

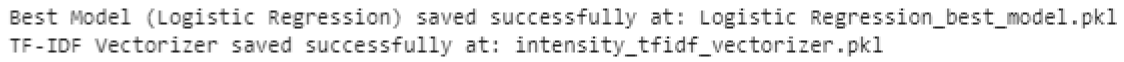
In this case, the Logistic Regression model has a relatively good test accuracy (69.70%) and a lower accuracy difference (22.29%) compared to other models. Therefore, considering both accuracy and the accuracy difference, the Logistic Regression model appears to be a reasonable choice.

STEP 8: SAVING THE MODEL

- We saved the best mode & TF-IDF vectorizer file using joblib



```
# Save the best model based on the least difference between training and test accuracy
best_model = results_df.loc[results_df['Model Name'] == best_model_name, 'Model'].values[0]
best_model_path = f'{best_model_name}_best_model.pkl'
joblib.dump(best_model, best_model_path)
print(f"\nBest Model ({best_model_name}) saved successfully at: {best_model_path}")

# Save the TF-IDF vectorizer
tfidf_vectorizer_path = 'intensity_tfidf_vectorizer.pkl'
joblib.dump(tfidf_vectorizer, tfidf_vectorizer_path)
print(f"TF-IDF Vectorizer saved successfully at: {tfidf_vectorizer_path}")
```

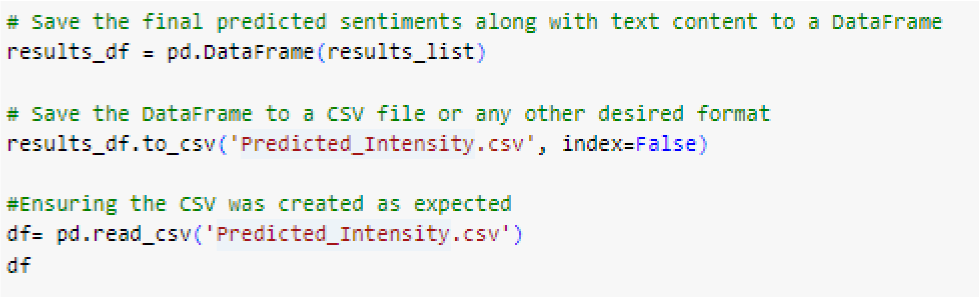


```
Best Model (Logistic Regression) saved successfully at: Logistic Regression_best_model.pkl
TF-IDF Vectorizer saved successfully at: intensity_tfidf_vectorizer.pkl
```

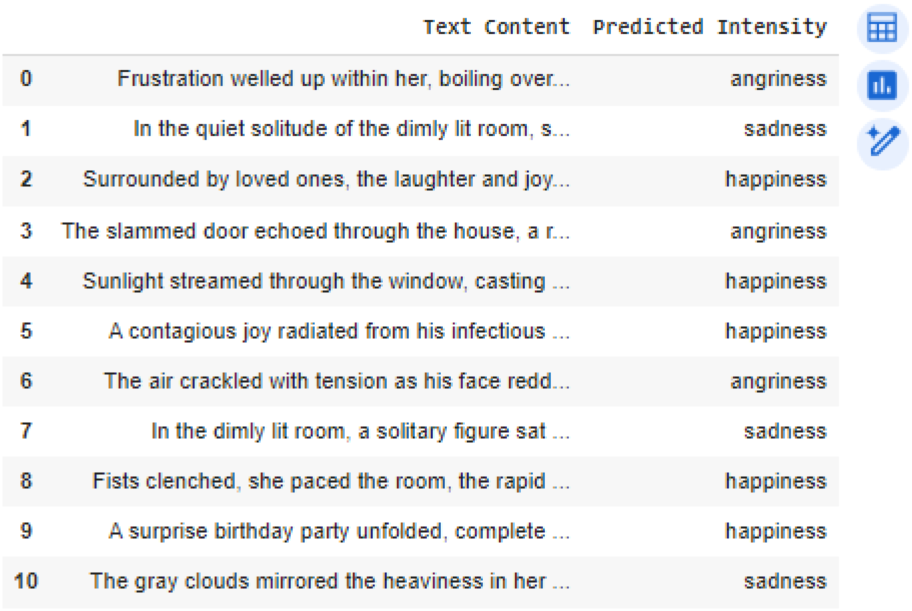
Insight: We got the best model based on less accuracy difference is L Logistic Regression and tfidf vectorizor file.

STEP 9: TEST & VALIDATE THE MODEL

- We load the saved pickle files of high accuracy best model (SVM) and tfidf vectorizor file(intensity_tfidf_vectorizer), test the few text contents to predict the intensity for trained model validation & save the result in CSV file(Predicted_Intensity).



```
# Save the final predicted sentiments along with text content to a DataFrame
results_df = pd.DataFrame(results_list)

# Save the DataFrame to a CSV file or any other desired format
results_df.to_csv('Predicted_Intensity.csv', index=False)

#Ensuring the CSV was created as expected
df= pd.read_csv('Predicted_Intensity.csv')
df
```



	Text Content	Predicted Intensity
0	Frustration welled up within her, boiling over...	angriness
1	In the quiet solitude of the dimly lit room, s...	sadness
2	Surrounded by loved ones, the laughter and joy...	happiness
3	The slammed door echoed through the house, a r...	angriness
4	Sunlight streamed through the window, casting ...	happiness
5	A contagious joy radiated from his infectious ...	happiness
6	The air crackled with tension as his face redd...	angriness
7	In the dimly lit room, a solitary figure sat ...	sadness
8	Fists clenched, she paced the room, the rapid ...	happiness
9	A surprise birthday party unfolded, complete ...	happiness
10	The gray clouds mirrored the heaviness in her ...	sadness

Insight: We observe that our Logistic Regression trained model predict the correct intensity of assigned text contents.

Conclusion: Model with the highest test accuracy is SVM with 70.13%. However,
The Logistic Regression model has a relatively good test accuracy (69.70%) and a lower accuracy difference (22.29%) compared to other models. Therefore, considering both accuracy and the accuracy difference, the Logistic Regression model appears to be a reasonable choice.