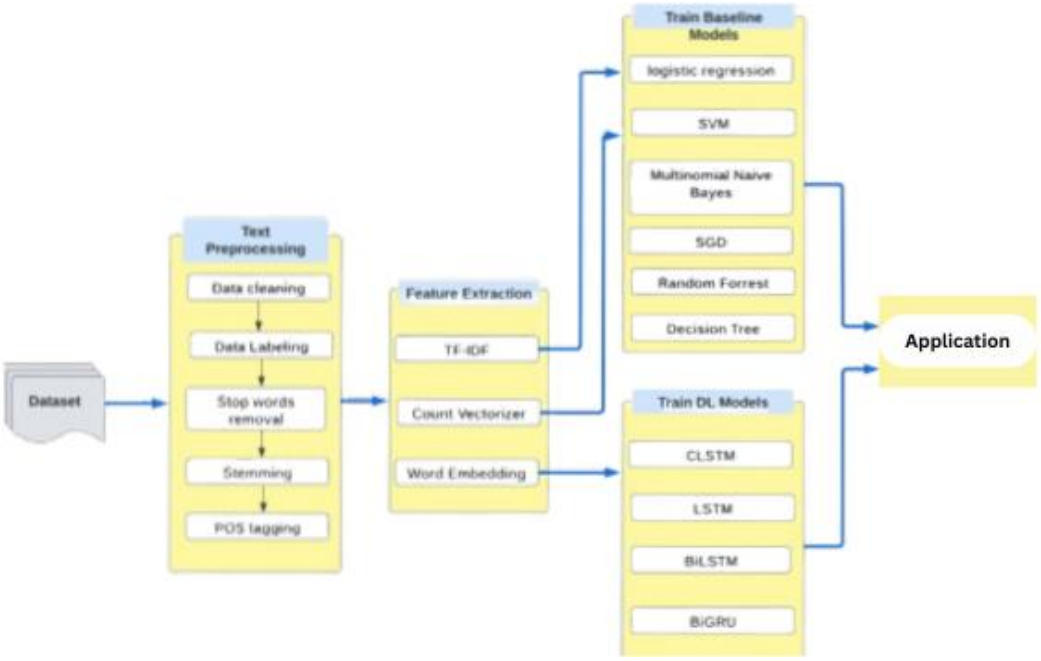
System Architecture:

Sentiment analysis system architecture entails several processes in the process of analyzing text data and sentiment assessments. At the beginning, the system collects data in two steps. The first step is data extraction where the raw data, which is often text data originated from social media and other platforms where people can express their feelings that is collected together with the specific labels. This data goes through a thorough preprocessing phase, which entails actions such as cleaning to remove noise and unnecessary information, tokenization, which splits text into words that have uniformity.

The process of preprocessing will be dealing with stopwords removal, lemmatization, and vectorization, for changing the text information into numerical format that suite for machine learning. Moreover, techniques like taking the N-grams into account and the POS tagging that can help in capturing the implied sentiment may be used. From preprocessing, the system moves to model selection and training where the appropriate machine learning model will be selected and trained using the preprocessed data with the aim of achieving the needed model performance. The model so trained is appraised through the use of various measures to appraise the performance of the model, making sure that the changes are done where necessary for the sake of accuracy and relevance. In the next stage, the model is adopted and further polished to meet the required accuracy level. Finally, deployment of the model into an application or platform for real-time sentiment analysis will be the last step.



Multiclass Sentiment Analysis:

The multiclass sentiment analysis in with Random Forest, SVM and Logistic Regression models involve data collection and readiness as a first step. A dataset inferred from text data labeled with five sentiment categories (positive, negative, very positive, very negative, neutral) is collected and preprocessed. Hence, the process entails cleaning the text, splitting it into individual words, removing the stopwords, and either lemmatizing or stemming the words to bring them to similar forms for analysis. Subsequently, the feature extraction is used to transform the text data which was processed into the numerical features. This is done by using the techniques like TF-IDF or word embeddings such as Word2Vec or glove. These numerical characteristics do for the text data what numbers do for machine learning models.

The dataset is then cut into training and test sets. Three different models are trained on the training set. Random Forest, SVM, and LR. The RandomForestClassifier, SVC, and LogisticRegression from the sklearn library are used for each model in this study. The hyperparameter tuning is executed through GridSearchCV or RandomizedSearchCV recipes to enhance the models' performance. The trained and adjusted models are assessed by the test set. Performance metrics such as precision, recall, accuracy, and F1 score are employed to estimate the effectiveness of the model in classifying sentiment categories. In this case, the most efficient model is used for implementation.

Following are the columns of Pre Processed Data.

```
Index(['Unnamed: 0', 'id', 'coa', 'commentId', 'txt', 'like', 'nolike', 'date',
       'uid', 'nchar',
       ...
       'X1f41e', 'X1f41d', 'X1f41f', 'X1f41a', 'X1f95d', 'X1f959', 'X1f982',
       'total.emojiCount', 'uiddate', 'optimized_sentiment_score'],
      dtype='object', length=359)
```

	txt	optimized_sentiment_score	tokens
0	good	0.0	[good]
1	Nervous tachycardia all the day long	0.0	[nervous, tachycardia, day, long]
2	Tired	0.0	[tired]
3	Напряжённый денёк	0.0	[напряжённый, денёк]
4	Сегодня пяяяятница😁	0.0	[сегодня, пяяяятница, x1f601]

Sentiment Category:

```
df['sentiment_category'] = df['tokens'].apply(lambda x: get_sentiment_category(sum(sia.polarity_scores(word)['compound'] for word in x) / len(x) if len(x) > 0 else 0))


# DataFrame with the new column
print(df.head())
```

```
                                txt  optimized_sentiment_score  \
0                              good                        0.0
1  Nervous tachycardia all the day long                        0.0
2                             Tired                        0.0
3                  Напряжённый денёк                        0.0
4          Сегодня пяяяятница&#x1f601;                        0.0

                          tokens  positive_words  negative_words  \
0                         [good]               1               0
1  [nervous, tachycardia, day, long]               0               1
2                         [tired]               0               1
3             [напряжённый, денёк]               0               0
4     [сегодня, пяяяятница, x1f601]               0               0

   neutral_words  very_positive_words  very_negative_words sentiment_category
0              0                    0                    0          Positive
1              3                    0                    0          Negative
2              0                    0                    0          Negative
3              2                    0                    0           Neutral
4              3                    0                    0           Neutral
```

Random Forest :

```
# Display classification report
print(classification_report(y_test, y_pred))
```

```
Accuracy: 0.9930926216640502
               precision    recall  f1-score   support

     Negative       0.98      0.96      0.97      1467
      Neutral       1.00      1.00      1.00      8622
     Positive       0.98      0.99      0.98      2618
Very Negative       1.00      1.00      1.00         6
Very Positive       1.00      0.85      0.92        27

     accuracy                           0.99     12740
    macro avg       0.99      0.96      0.98     12740
 weighted avg       0.99      0.99      0.99     12740
```

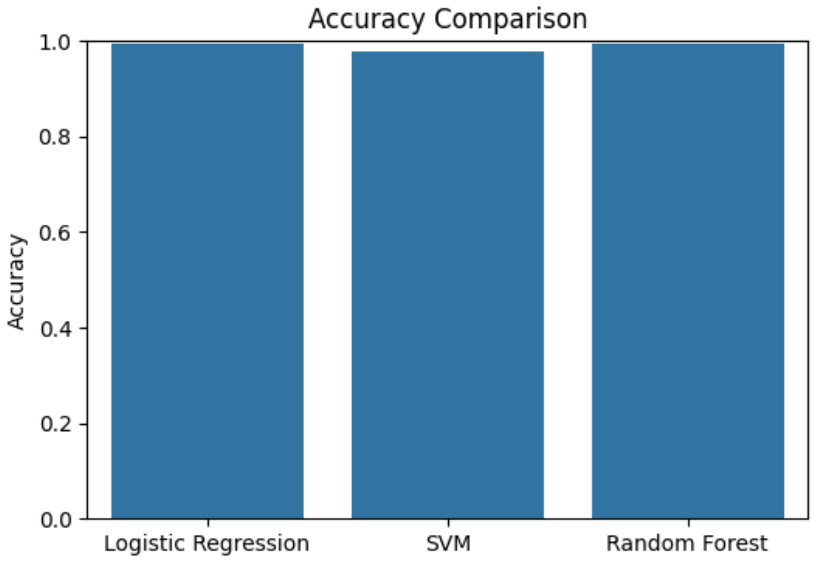
Support Vector Machine:

```
Support Vector Machine Accuracy: 0.9793563579277865
Support Vector Machine Classification Report:
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_(
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_(
  _warn_prf(average, modifier, msg_start, len(result))
               precision    recall  f1-score   support

     Negative       0.94      0.93      0.94      1467
      Neutral       0.99      1.00      0.99      8622
     Positive       0.96      0.97      0.96      2618
Very Negative       0.00      0.00      0.00         6
Very Positive       0.00      0.00      0.00        27

     accuracy                           0.98     12740
    macro avg       0.58      0.58      0.58     12740
 weighted avg       0.98      0.98      0.98     12740
```

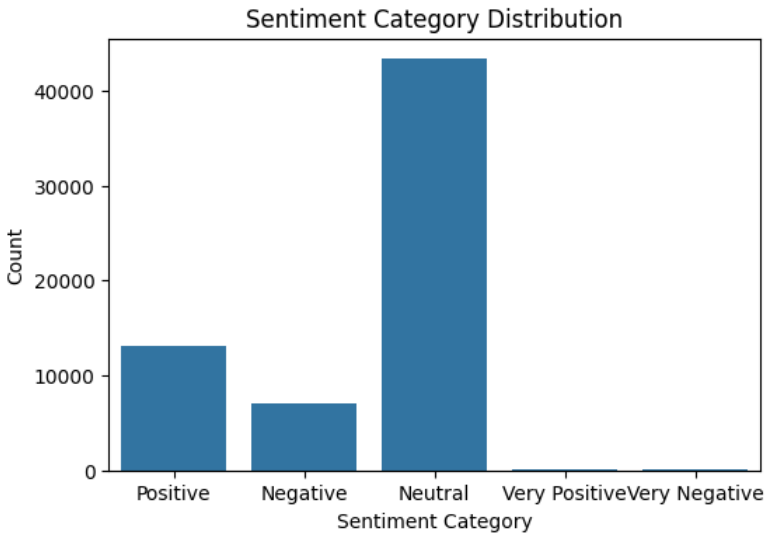
Logistic Regression:

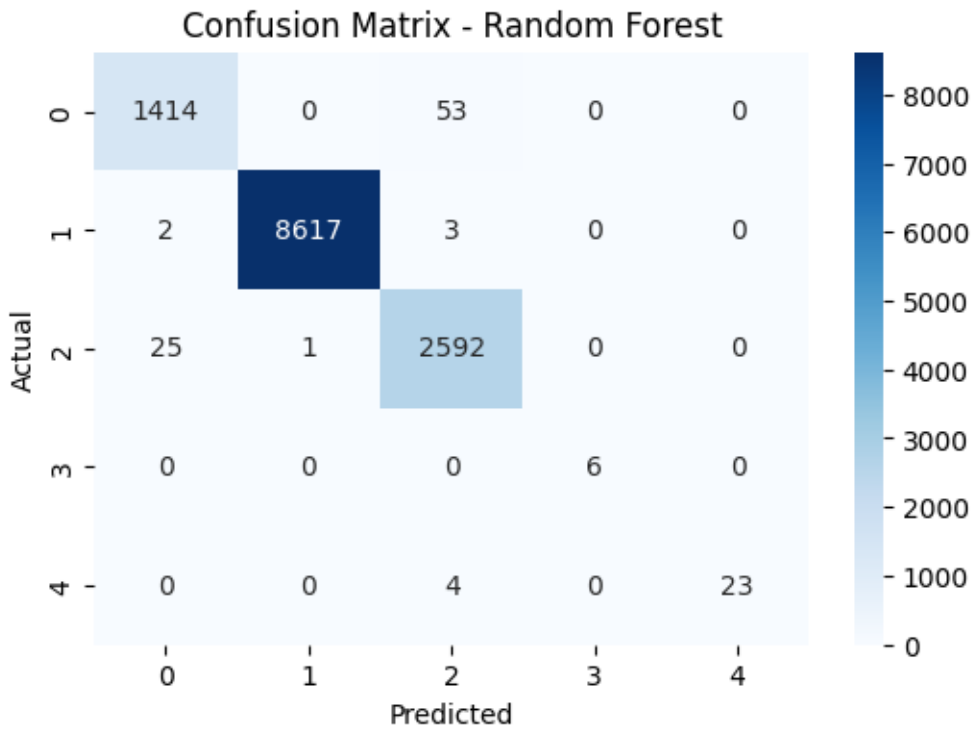
```
Logistic Regression Accuracy: 0.9934850863422292
Logistic Regression Classification Report:
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_class
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_class
  _warn_prf(average, modifier, msg_start, len(result))
              precision    recall  f1-score   support

     Negative       0.99      0.96      0.98      1467
      Neutral       1.00      1.00      1.00      8622
     Positive       0.98      0.99      0.99      2618
Very Negative       0.00      0.00      0.00         6
Very Positive       0.88      0.85      0.87        27

     accuracy                           0.99     12740
    macro avg       0.77      0.76      0.77     12740
 weighted avg       0.99      0.99      0.99     12740
```

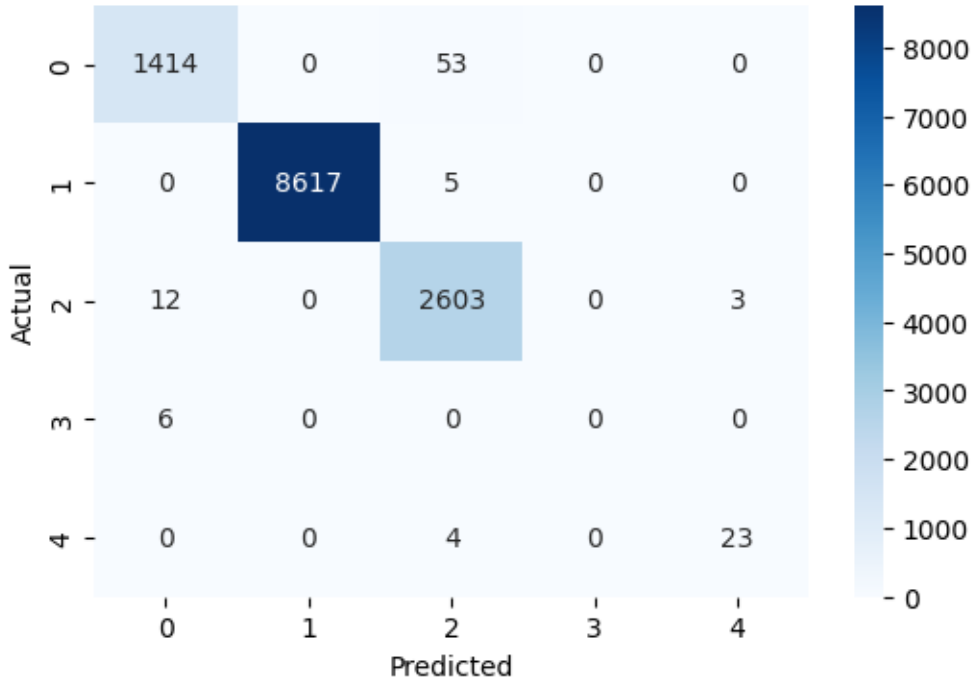
Accuracies:

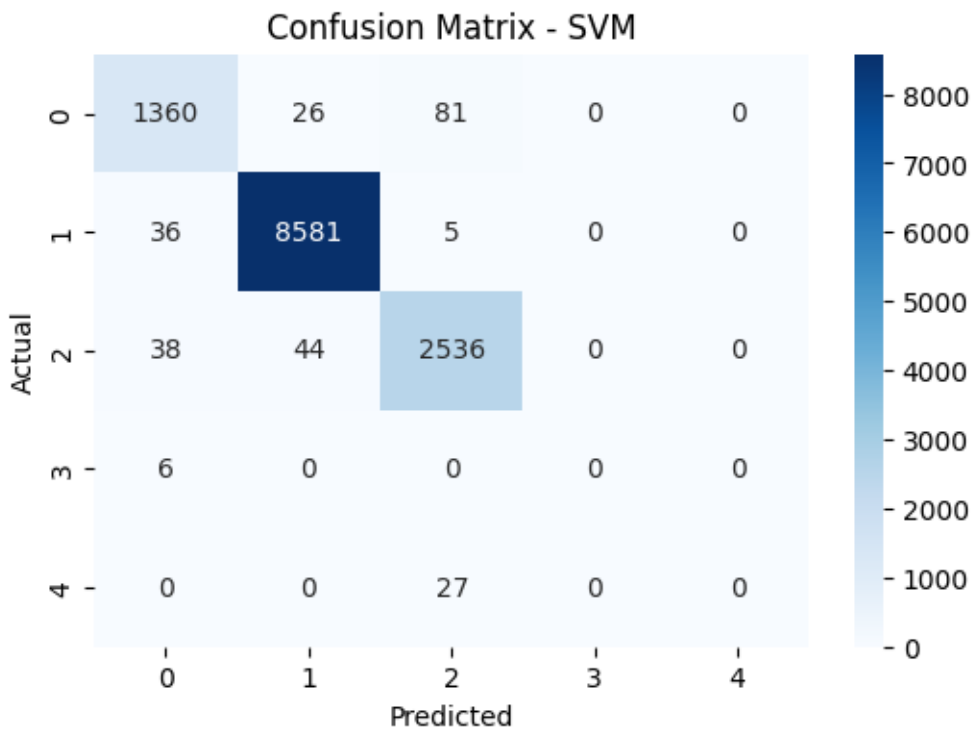


Category distribution by semtiments:




Confusion Matrix - Random Forest
Actual
Predicted
0
1
2
3
4
1414
0
53
0
0
2
8617
3
0
0
25
1
2592
0
0
0
0
0
6
0
0
0
4
0
23
8000
7000
6000
5000
4000
3000
2000
1000
0


Confusion Matrix - Logistic Regression
Actual
Predicted
0
1
2
3
4
1414
0
53
0
0
0
8617
5
0
0
12
0
2603
0
3
6
0
0
0
0
0
0
4
0
23
8000
7000
6000
5000
4000
3000
2000
1000
0


Confusion Matrix - SVM
Actual
Predicted
0
1
2
3
4
1360
26
81
0
0
36
8581
5
0
0
38
44
2536
0
0
6
0
0
0
0
0
0
27
0
0
8000
7000
6000
5000
4000
3000
2000
1000
0