Electrical and Computer Engineering Department ENCS3340 Artificial Intelligence, First Semester, 2022-2023 Machine Learning Project

Tweet Emotion Detection

Prepared by:

Name: Lana Batnij **ID:**1200308

Name: Yasmeen kamal **ID:**1201146

Name: Anas Hawamda **ID:**1202121

Instructor: Dr.Aziz Qaroush

Data :16/2/2022
Section: 2

Abstract:

The aim of this project is to create an application that predict the emotion expressed in a tweet based on extracted features from the tweet content, using the machine learning like decision trees, naïve bayes, and neural networks, and that to help humans to make decisions weather the tweets is positive related to the subject or negative.

Table of contents:

Table of Figures:

Theory:

Machine Learning:

In machine learning, Precision, Recall, Accuracy, and F1 Score are metrics used to evaluate the performance of a classification model. These metrics help to determine how well a model is performing in terms of its ability to correctly identify the positive and negative classes.

- Precision: Precision is the measure of the proportion of true positive predictions out of all the positive predictions made by the model. It is calculated as True Positives / (True Positives + False Positives). Precision measures how often the model is correct when it predicts that a sample is positive.

- Recall: Recall is the measure of the proportion of true positive predictions out of all the actual positive samples. It is calculated as True Positives / (True Positives + False Negatives). Recall measures the ability of the model to correctly identify positive samples.

- Accuracy: Accuracy is the measure of the proportion of correct predictions out of all the predictions made by the model. It is calculated as (True Positives + True Negatives) / (True Positives + False Positives + False Negatives + True Negatives). Accuracy measures the overall performance of the model.

- F1 Score: F1 Score is the harmonic mean of precision and recall. It is calculated as 2 * (Precision * Recall) / (Precision + Recall). F1 Score combines both precision and recall into a single metric, and is often used when both metrics are equally important.

In summary, Precision, Recall, Accuracy, and F1 Score are important metrics used to evaluate the performance of a classification model, and understanding their meanings and how to calculate them can help you to better interpret the results of your model.

Forest Decision trees:

A random forest is simply a collection of decision trees whose results are aggregated into one final result. Their ability to limit overfitting without substantially increasing error due to bias is why they are such powerful models. One-way Random Forests reduce variance is by training on different samples of the data [2].

Advantages to using decision trees:

- ✓ Easy to interpret and make for straightforward visualizations.
- ✓ The internal workings are capable of being observed and thus make it possible to reproduce work.
- ✓ Can handle both numerical and categorical data.
- ✓ Perform well on large datasets
- ✓ Are extremely fast

Support vector machine:

In machine learning, support vector machines are supervised learning models with associated learning algorithms that analyze data for classification and regression analysis. Developed at AT&T Bell Laboratories by Vladimir Vapnik with colleagues SVMs are one of the most robust prediction methods, being based on statistical learning frameworks or VC theory proposed by Vapnik (1982, 1995) and Chervonenkis (1974). Given a set of training examples, each marked as belonging to one of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary . SVM maps training examples to points in space so as to maximise the width of the gap between the two categories. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall.

In addition to performing linear classification, SVMs can efficiently perform a non-linear classification using what is called the kernel trick, implicitly mapping their inputs into high-dimensional feature spaces.

When data are unlabelled, supervised learning is not possible, and an unsupervised learning approach is required, which attempts to find natural clustering of the data to groups, and then map new data to these formed groups. The support vector clustering[2] algorithm, created by Hava Siegelmann and Vladimir Vapnik, applies the statistics of support vectors, developed in the support vector machines algorithm, to categorize unlabeled data [1].

Procedure and our Idea:

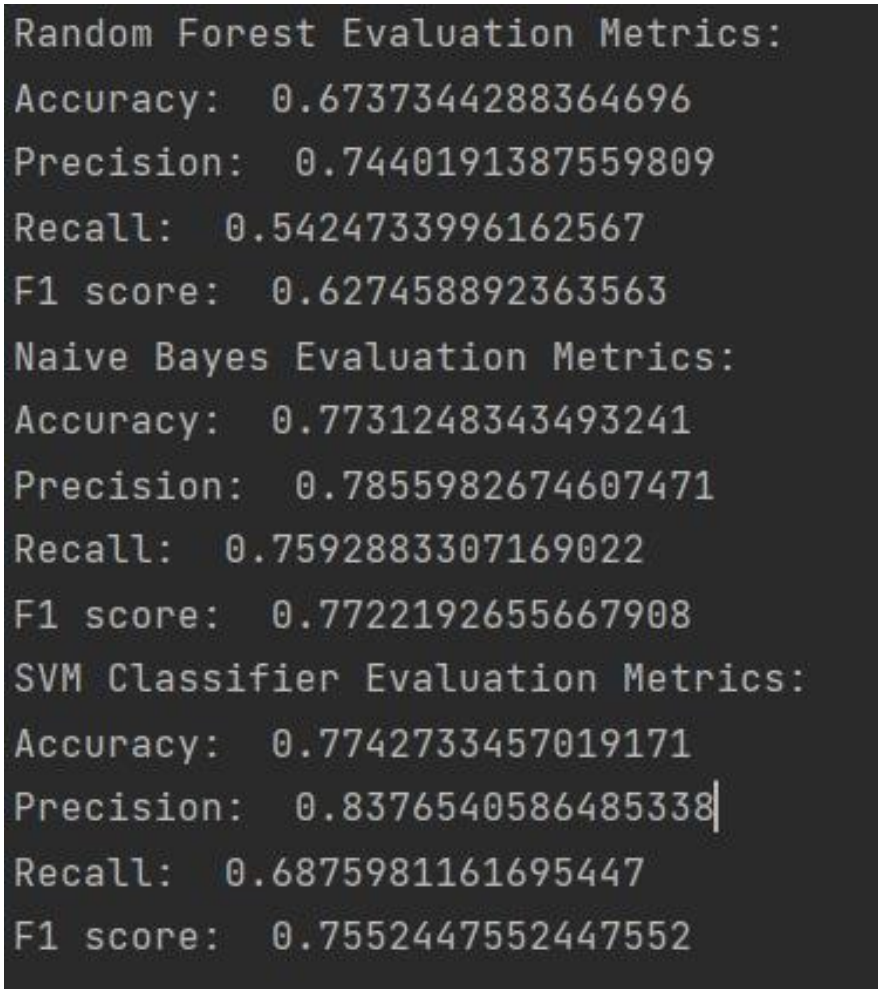
Basically, our idea combines between using decision tree the random Forest, Naïve Bayes algorithm, and the Support vector machine algorithm.

Firstly, we process the Data as we read and analyzed all the data from the file using the pandas Library using python language programming, after that we transform the data from String to binary so that we can handle it easily the basic idea was if the tweet is positive the value will be one, else if it is negative it is going to be zero.

After that extract the features, in this stage we decided to use the classical learning method such that SVM, Naïve bayes, and the Random Forest decision trees.

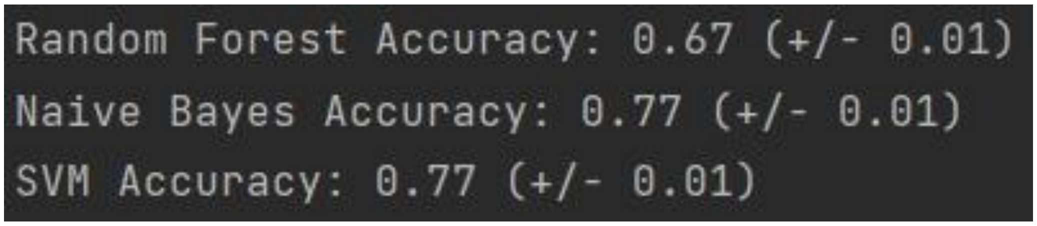
After we decide the Training algorithms that we find the most suitable. We wrote the code for the Random Forest decision tree with limits to avoid the over fitting , same as SVM and Naïve bayes , then we do exactly as the application demand we applied 25% testing the process and the 75% training the model. After this we did let the data be as token counts array and applied the algorithms codes, lastly, we show the results.

Results:



```
Random Forest Evaluation Metrics:
Accuracy:  0.6737344288364696
Precision:  0.7440191387559809
Recall:  0.5424733996162567
F1 score:  0.627458892363563
Naive Bayes Evaluation Metrics:
Accuracy:  0.7731248343493241
Precision:  0.7855982674607471
Recall:  0.7592883307169022
F1 score:  0.7722192655667908
SVM Classifier Evaluation Metrics:
Accuracy:  0.7742733457019171
Precision:  0.8376540586485338
Recall:  0.6875981161695447
F1 score:  0.7552447552447552
```

Figure 1.1



```
Random Forest Accuracy: 0.67 (+/- 0.01)
Naive Bayes Accuracy: 0.77 (+/- 0.01)
SVM Accuracy: 0.77 (+/- 0.01)
```

Figure 1.2

Appendix A:

```
import pandas as pd

from sklearn.model_selection import train_test_split, cross_val_score

from sklearn.feature_extraction.text import CountVectorizer

from sklearn.ensemble import RandomForestClassifier

from sklearn.naive_bayes import MultinomialNB

from sklearn import svm

from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score


# Load data into pandas dataframes

P_Tweets = pd.read_csv('PositiveTweets.tsv', sep='\t', names=['Class', 'text'])

N_Tweets = pd.read_csv('NegativeTweets.tsv', sep='\t', names=['Class', 'text'])


# Concatenate the dataframes and convert Classes to binary values

Tweets = pd.concat([P_Tweets, N_Tweets], ignore_index=True)

Tweets['Class'] = Tweets['Class'].map({'pos': 1, 'neg': 0})

# Split the data into training and testing sets

CrossTrain, CrossTest, ClassicalTrain, ClassicalTest = train_test_split(Tweets['text'], Tweets['Class'],
test_size=0.25)


# Transform the text into a matrix of token counts

vectorizer = CountVectorizer()

CrossTrain_counts = vectorizer.fit_transform(CrossTrain)

CrossTest_counts = vectorizer.transform(CrossTest)


rfc = RandomForestClassifier(n_estimators=200, max_depth=10, random_state=42,
min_samples_split=10, max_features='sqrt')

rfc.fit(CrossTrain_counts, ClassicalTrain)

nb = MultinomialNB()

nb.fit(CrossTrain_counts, ClassicalTrain)
```

```
svm_clf = svm.SVC(kernel='linear', C=0.1, random_state=42)
svm_clf.fit(CrossTrain_counts, ClassicalTrain)
Cross_RFC = cross_val_score(rfc, CrossTrain_counts, ClassicalTrain, cv=5)
Cross_NB = cross_val_score(nb, CrossTrain_counts, ClassicalTrain, cv=5)
Cross_SVM = cross_val_score(svm_clf, CrossTrain_counts, ClassicalTrain, cv=5)
print("Results for cross-validation:")
print("Random Forest Accuracy: %0.2f (+/- %0.2f)" % (Cross_RFC.mean(), Cross_RFC.std() * 2))
print("Naive Bayes Accuracy: %0.2f (+/- %0.2f)" % (Cross_NB.mean(), Cross_NB.std() * 2))
print("SVM Accuracy: %0.2f (+/- %0.2f)" % (Cross_SVM.mean(), Cross_SVM.std() * 2))

# Predict test set using each classifier
Classical_RFC = rfc.predict(CrossTest_counts)
Classical_NB = nb.predict(CrossTest_counts)
Classical_SVM = svm_clf.predict(CrossTest_counts)
print("Results for  classical method:")
print("Random Forest Evaluation Metrics:")
print("Accuracy: ", accuracy_score(ClassicalTest, Classical_RFC))
print("Precision: ", precision_score(ClassicalTest, Classical_RFC))
print("Recall: ", recall_score(ClassicalTest, Classical_RFC))
print("F1 score: ", f1_score(ClassicalTest, Classical_RFC))

print("Naive Bayes Evaluation Metrics:")
print("Accuracy: ", accuracy_score(ClassicalTest, Classical_NB))
print("Precision: ", precision_score(ClassicalTest, Classical_NB))
print("Recall: ", recall_score(ClassicalTest, Classical_NB))
print("F1 score: ", f1_score(ClassicalTest, Classical_NB))

print("SVM Classifier Evaluation Metrics:")
print("Accuracy: ", accuracy_score(ClassicalTest, Classical_SVM))
print("Precision: ", precision_score(ClassicalTest, Classical_SVM))
print("Recall: ", recall_score(ClassicalTest, Classical_SVM))
print("F1 score: ", f1_score(ClassicalTest, Classical_SVM))
```

References:

[1] https://en.wikipedia.org/wiki/Support_vector_machine

[2] https://towardsdatascience.com/decision-trees-and-random-forests-df0c3123f991