Twitter Feed Sentiment Analysis

BIA 678 A- Big Data Technologies

Presented by: Group 10:

Pratik Upreti	MS in Computer Science
Savleen Kaur	MS in Computer Science
Sanyam Cheraya	MS in Business Intelligence & Analytics
Vidya Natraj	MS in Information Systems

Table of Contents

Introduction

As a part of this team project, we are performing sentiment analysis of tweets by giving each tweet a sentiment score. The score will be based on keywords from the tweet and will be marked in four scales of measurement – Positive, Neutral, Negative, and Irrelevant. The sentiment analysis is done using PySpark and Python. PySpark is run on Databricks and is used for parallelization. The main application of performing sentiment analysis is to analyze a Twitter feed (social media) and understand how customers or end-users perceive or feel about a brand or a service. Any negative mentions could warrant corrective action before the situation escalates further. We will be using advanced text mining techniques such as machine learning models or Natural Language Processing to analyze tweets and give the tweets a score. The four Machine Learning models used are K Nearest Neighbors (KNN), Support Vector Classifier (SVC), Multinomial Naïve Bayes, and Random Forest. Finally, we will evaluate the four Machine Learning models on model precision and share our thoughts on which model will work best.

Tools and Technologies

Databricks: Databricks is a cloud-based platform that simplifies the management of complex data. Cluster creation for data processing was previously only available on Amazon Web Services and Google Cloud, but it is now available on Azure as well. Databricks is a platform for developing, testing, and deploying machine learning and analytics applications to help businesses improve their performance. The Databricks Machine Learning platform may be used to train models, track models via experiments, create feature tables, and distribute, manage, and serve models.

PySpark: PySpark is a useful language for a variety of tasks, including exploratory data analysis at scale, constructing machine learning pipelines, and developing ETLs for a data platform. PySpark is an excellent programming language for developing more scalable data analytics and pipelines.

Parallelization is achieved using PySpark in both local mode and cluster mode, what spark does is it divides data into partitions for parallel processing. When we run a Spark program on a cluster environment parallel processing occurs. For example, when we run our PySpark program on AWS Cluster, if we increase the cluster size, we get to see that it will help in decreasing the time taken for processing.

Pandas:

- It is the core library for scientific computing in Python
- It creates a multidimensional array object, and tools for working with these arrays

Numpy:

- It imports data of various file formats such as csv, excel etc.
- It allows data manipulation like join, merge, concatenation etc.

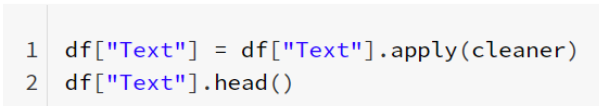
Exploratory Data Analysis

Understanding the data

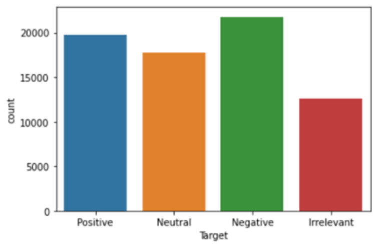
It is the Twitter dataset that will be used to perform the sentiment analysis of the tweets posted by the people. There is a total of 74682 rows and 4 columns. There are in total 4 columns in the data set. Id, which is just a serial number, given to the tweet. Next is the Entity for which the tweet is posted for example Call of duty, League of legends, etc. next up is Target which defines whether a particular tweet will be positive, negative, irrelevant or neutral. This is key in the analysis as it will impact the training the model and closely predicting the target for the text in the test dataset. The last column is the text column which has the corresponding tweets. The sentiment analysis will be performed based on the tweets in the text column.

Data Cleaning

As discussed above, the data file has four fields – Tweet ID, Entity, Tweet Description and finally the sentiment score. Data cleaning is handled within the code, and this handles punctuations, stop words (a, the, is etc.), lemmatize words, and stemming words to produce variants of the base word. Below is a sample of the code used in PySpark.



```
df["Text"] = df["Text"].apply(cleaner)
df["Text"].head()
```



The bar chart above is an analysis of the tweet feed after the data has been cleaned, where the tweets have been categorized in the four-point scale. We can see that that over 20000 tweets have been categorized as negative by the model, followed by positive, neutral and irrelevant.

Data preparation

The next step is to prepare the data into a form which will be accepted as an input to the model designed. Our Validation dataset has 74682 rows with 4 columns. The following column names, "Id", "Entity", "Target", and "Text" were added to the code. We are passing Target and Text column to our models for further classification to understand if the tweet was positive/negative or neutral.

```
In [3]:  columns = ["Id","Entity","Target","Text"]
         data = pd.read_csv("/kaggle/input/twitter-entity-sentiment-analysis/twitter_traini
         ng.csv",
                             names=columns,header=None)
```

```
In [4]:  data.head()
```

<u>Data Splitting</u>

Splitting the dataset into testing and training data is the primary part of any machine learning model. But before that, we need to perform some data cleaning. First, a new column sentiment has been created, giving the sentiment a value of 1, 0, -1 based on whether the target is positive, negative, or neutral, respectively. There are some unnecessary words that need to be removed. The text is split into a list of words called tokens. The stop words are removed using the nltk library and using lemmatizer and porter stemmer to convert words into generic form for example both "making" and "make" will have the same generic word "make" so that it is not counted twice. After that, we joined these pre-processed tokens back to their original text form. We are now ready to split the data into test and training data. We took 70% as training data and 30% as test data.

<u>Methodology</u>

In this paper we are attempting to use the same Twitter feed dataset and perform a model comparison. Details on the respective models can be found below.

<u>KNN</u>

- KNN known as K nearest neighbors is a supervised Machine Learning model that is a powerful classification algorithm used in pattern recognition
- K nearest neighbors stores all available cases and classifies new cases based on similarity measure (example: distance function)
- An object is classified based on majority votes for its neighbor classes
- We pass on values like 1, 3, 5, 7 etc. to KNN till the point accuracy curve doesn't get flattened, the moment it starts to flatten, we stop
- The K values are always odd as even values may lead to a tie in vote
- It is simple and intuitive algorithm
- The input consists of the k closest training examples in a data set. The output depends on whether KNN is used for classification or regression
- In KNN classification, the output is a class membership. An object is classified by a plurality vote of its neighbors, with the object being assigned to the class most common among

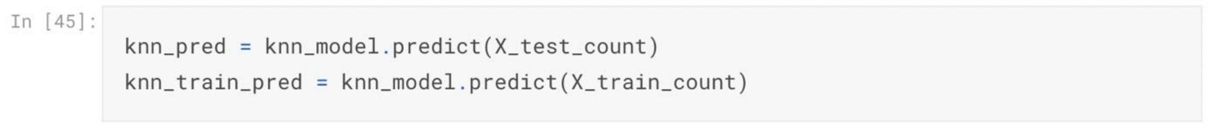
its k nearest neighbors. If k = 1, then the object is simply assigned to the class of that single nearest neighbor

- In KNN regression, the output is the property value for the object. This value is the average of the values of k nearest neighbors

Advantages of KNN:

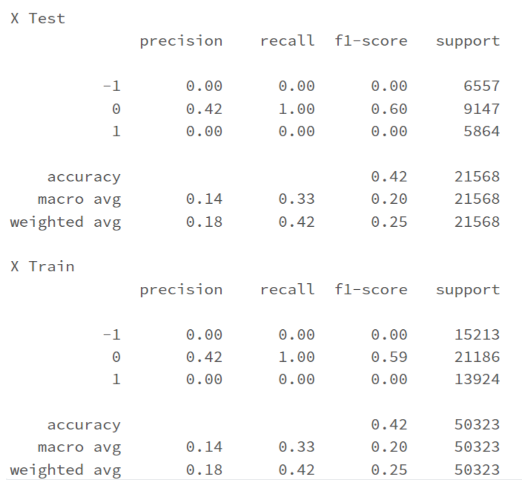
- Simple to implement and intuitive to understand
- Can learn non-linear decision boundaries when used for classification and regression. Can come up with a highly flexible decision boundary adjusting the value of K.
- **No training time for classification/regression:** The KNN algorithm has no explicit training step and all the work happens during prediction
- **Constantly evolves with new data:** Since there is no explicit training step, as we keep adding new data to the dataset, the prediction is adjusted without having to retrain a new model
- **Single Hyperparameters:** There is a single hyperparameter, the value of K. This makes hyper parameter tuning easy.
- **Choice of distance metric:** There are many distance metrics to choose from. Some popular distance metrics used are Euclidean, Manhattan, Minkowski, hamming distance and so on.

Snippet of the code:

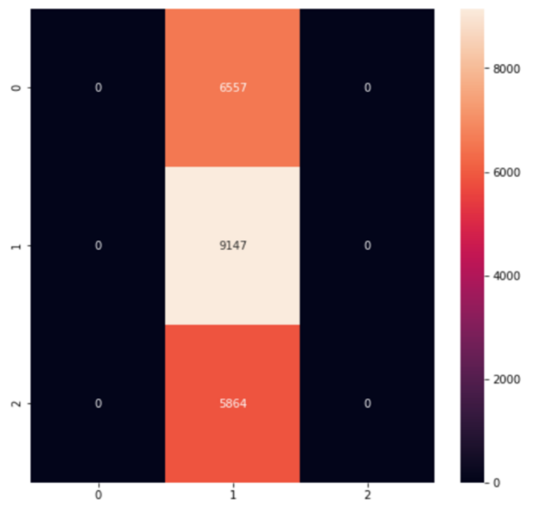


```
In [45]:
knn_pred = knn_model.predict(X_test_count)
knn_train_pred = knn_model.predict(X_train_count)
```

Output: Confusion Matrix and Heatmap of KNN:



```
X Test
              precision    recall  f1-score   support

          -1       0.00      0.00      0.00      6557
           0       0.42      1.00      0.60      9147
           1       0.00      0.00      0.00      5864

    accuracy                           0.42     21568
   macro avg       0.14      0.33      0.20     21568
weighted avg       0.18      0.42      0.25     21568

X Train
              precision    recall  f1-score   support

          -1       0.00      0.00      0.00     15213
           0       0.42      1.00      0.59     21186
           1       0.00      0.00      0.00     13924

    accuracy                           0.42     50323
   macro avg       0.14      0.33      0.20     50323
weighted avg       0.18      0.42      0.25     50323
```



Multinomial Naïve Bayes

- Multinomial Naive Bayes is one of algorithms whose method is based on applying Bayes theorem
- Bayes theorem calculates probability P(A|B) where A is the class of the possible outcomes and B is the given instance which must be classified
- Here below the formula of naive bayes:

  P(A|B) = P(A) * P(B|A)/P(B)

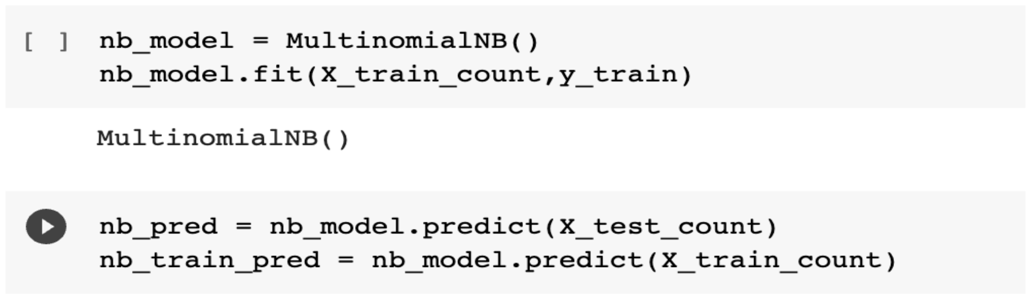
  Where:
  P(A|B): measure of how often A and B are observed to occur together

  P(B|A): measures of how often B occur in A (likelihood)
  P(A): measure of how often A is observed to occur in general (prior probability)
  P(B): measure of how often B is observed to occur in general (marginal likelihood)

  Advantages of Multinomial NB algorithm are:

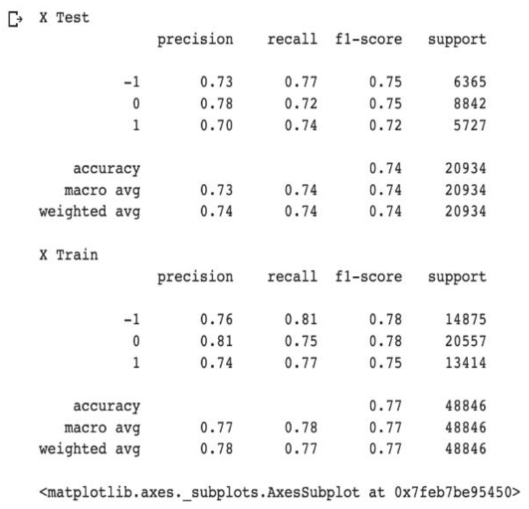
- To learn the parameters, a limited amount of training data is required
- When compared to sophisticated models, it can be taught quickly
- It is straightforward to implement because all that is required is probability calculation. This method is applicable to both continuous and discrete data
- It is simple and can be used to forecast real-time applications
- It is extremely scalable and can easily handle massive datasets

Snippet of the code:

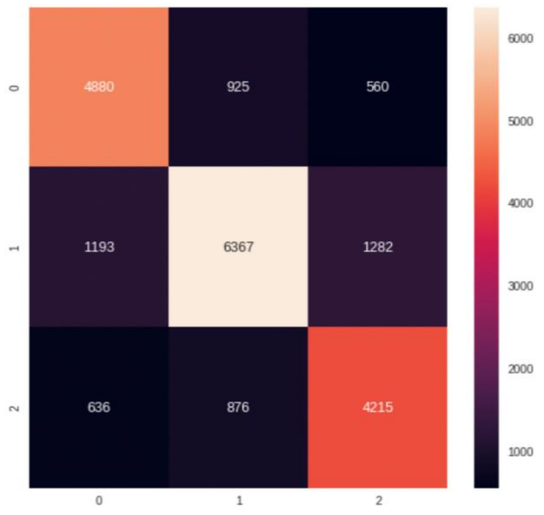


```
[ ]  nb_model = MultinomialNB()
     nb_model.fit(X_train_count,y_train)

     MultinomialNB()

     nb_pred = nb_model.predict(X_test_count)
     nb_train_pred = nb_model.predict(X_train_count)
```

Output: Confusion Matrix and Heatmap of Multinomial NB:



```
X Test
              precision    recall  f1-score   support

          -1       0.73      0.77      0.75      6365
           0       0.78      0.72      0.75      8842
           1       0.70      0.74      0.72      5727

    accuracy                           0.74     20934
   macro avg       0.73      0.74      0.74     20934
weighted avg       0.74      0.74      0.74     20934

X Train
              precision    recall  f1-score   support

          -1       0.76      0.81      0.78     14875
           0       0.81      0.75      0.78     20557
           1       0.74      0.77      0.75     13414

    accuracy                           0.77     48846
   macro avg       0.77      0.78      0.77     48846
weighted avg       0.78      0.77      0.77     48846

<matplotlib.axes._subplots.AxesSubplot at 0x7feb7be95450>
```



Random Forest

- Random forest is a supervised machine learning algorithm that is widely used for classification and regression
- Theoretically, it builds decision trees on different samples and takes their majority vote for classification
- There will be several decision trees for the homogenous subsets of the dataset and the outcome is based on the majority vote for the prediction of the decision tree
- One of the key features of the random forest model is that, it is able to address the issue of high variance and an entire forest is being trained (hence the name)
- Training a large forest does not necessarily improve the model, an out of bag error as a function of number of trees can help in arriving at the optimal number of trees

Advantages of Random Forest

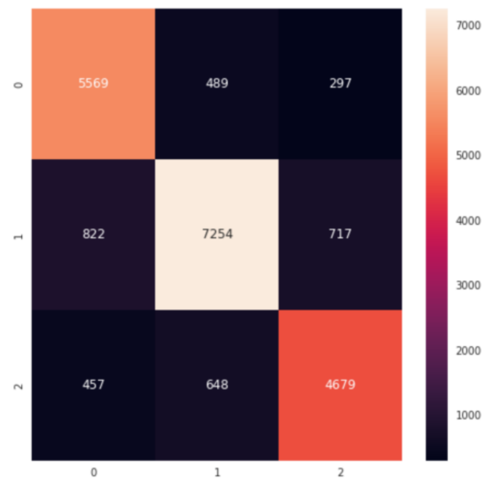
- It is a data robust algorithm, can handle different types of data and requires minimal to no data preprocessing
- Also, the entire dataset can be used to train, evaluate the model and calculate the test (generalization) error from the same dataset
- There is no need to split data into a training and holdout set. This is due to the bagging process that allows random datasets to be picked with replacement, so there will always be a set of points that wasn't used to create the tree.

Snippet of the code

```
rf_tuned = RandomForestClassifier(max_depth = 12,
                                  max_features = 7,
                                  min_samples_split = 2).fit(X_train_count,y_train)
rf_pred = rf_tuned.predict(X_test_count)
rf_train_pred = rf_tuned.predict(X_train_count)
```

Output: Confusion Matrix and Heatmap of Random Forest:

```
X Test
              precision    recall  f1-score   support

          -1       0.81      0.88      0.84      6355
           0       0.86      0.82      0.84      8793
           1       0.82      0.81      0.82      5784

    accuracy                           0.84     20932
   macro avg       0.83      0.84      0.83     20932
weighted avg       0.84      0.84      0.84     20932

X Train
              precision    recall  f1-score   support

          -1       0.89      0.94      0.92     14883
           0       0.93      0.91      0.92     20603
           1       0.92      0.90      0.91     13355

    accuracy                           0.92     48841
   macro avg       0.92      0.92      0.92     48841
weighted avg       0.92      0.92      0.92     48841
```



Support Vector Classifier/Machine

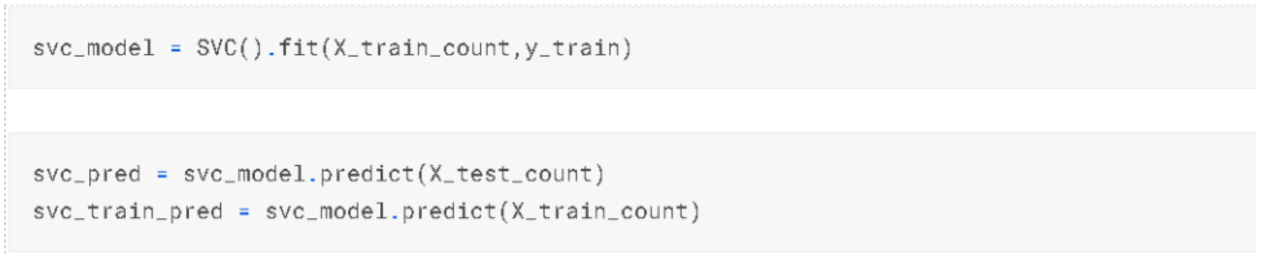
- SVM or Support Vector Machines are the supervised machine learning models which help in analyzing the data for classification and regression
- It helps in making the data linearly separable by mapping the high dimension feature and categorizing them further

- This algorithm is mostly used to deal with classification problems. The main goal of this algorithm is to find the best separable line which will divide the space we are working on into two different spaces.
- There are a variety of parameters and attributes used in this algorithm which are gamma, c, etc. and may have different effects depending on the type of model.

<u>Advantages of SVM</u>

- They are highly effective and efficient with high dimension spaces as they deal with data having high dimension features
- More efficient in cases where the size of dimensions is greater than the number of samples
- It is also memory efficient since it uses a subset of training data in the decision function also known as Support Vectors

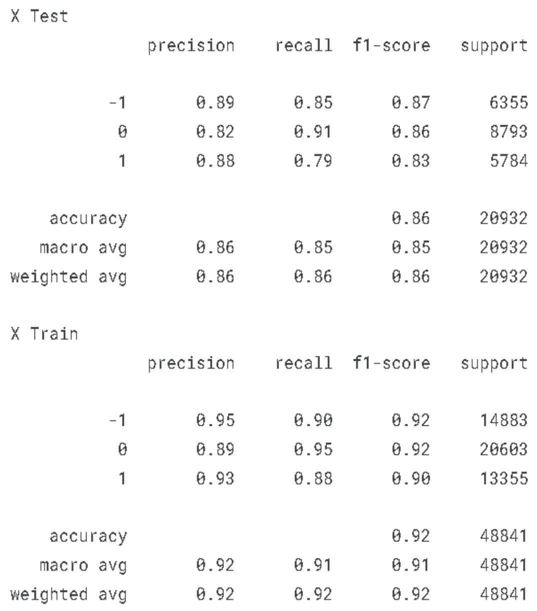
Snippet of code



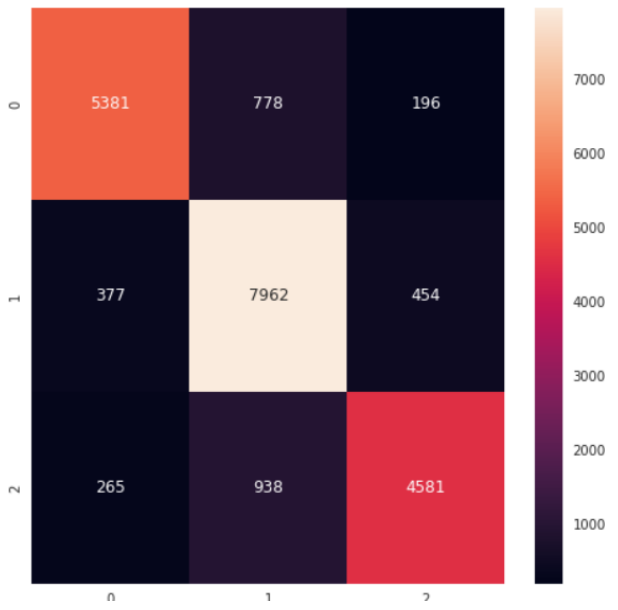
```
svc_model = SVC().fit(X_train_count,y_train)
```

```
svc_pred = svc_model.predict(X_test_count)
svc_train_pred = svc_model.predict(X_train_count)
```

Output: Confusion Matrix, and Heatmap of Support Vector Machine:

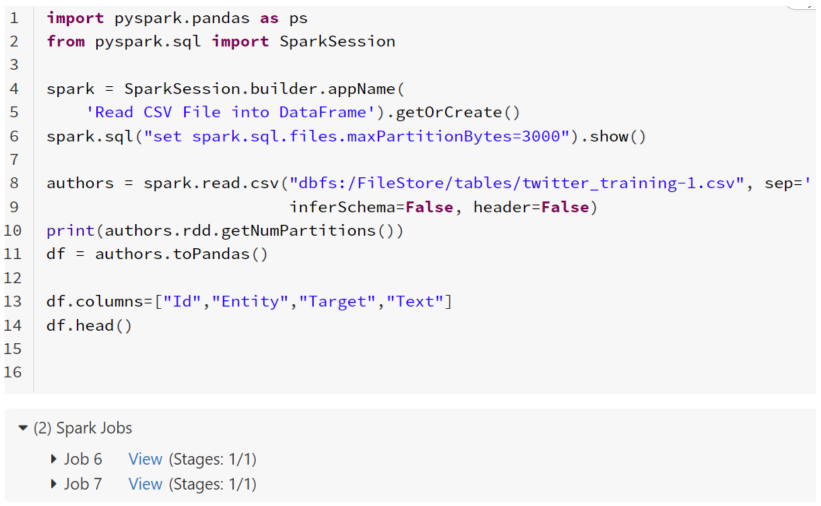


```
X Test
              precision    recall  f1-score   support

          -1       0.89      0.85      0.87      6355
           0       0.82      0.91      0.86      8793
           1       0.88      0.79      0.83      5784

    accuracy                           0.86     20932
   macro avg       0.86      0.85      0.85     20932
weighted avg       0.86      0.86      0.86     20932

X Train
              precision    recall  f1-score   support

          -1       0.95      0.90      0.92     14883
           0       0.89      0.95      0.92     20603
           1       0.93      0.88      0.90     13355

    accuracy                           0.92     48841
   macro avg       0.92      0.91      0.91     48841
weighted avg       0.92      0.92      0.92     48841
```



Leveraging Parallelization in PySpark

The Spark application on databricks runs on a cluster, where the cluster manager assigns tasks to workers. Spark manages data through RDDs using partitions that parallelize distributed data processing by reading data into an RDD from a node that is closest to it. Below is a snapshot of the code from Databricks where the data is partitioned across two worker nodes. We have specified the maximum number of partitions in bytes required as a text file. Based on the partition set, two jobs are created in parallel. Jobs are further divided into stages which are further spawned into multiple tasks. The biggest advantage of using partitioning is accelerating the data processing time as data is processed in parallel.

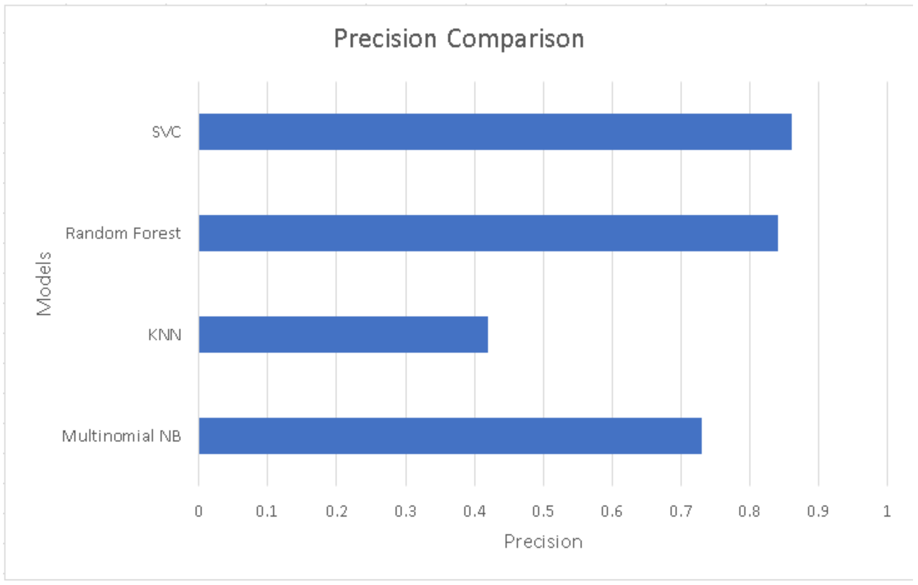


```
import pyspark.pandas as ps
from pyspark.sql import SparkSession

spark = SparkSession.builder.appName(
    'Read CSV File into DataFrame').getOrCreate()
spark.sql("set spark.sql.files.maxPartitionBytes=3000").show()

authors = spark.read.csv("dbfs:/FileStore/tables/twitter_training-1.csv", sep='
                         inferSchema=False, header=False)
print(authors.rdd.getNumPartitions())
df = authors.toPandas()

df.columns=["Id","Entity","Target","Text"]
df.head()
```

```
(2) Spark Jobs
    Job 6    View (Stages: 1/1)
    Job 7    View (Stages: 1/1)
```

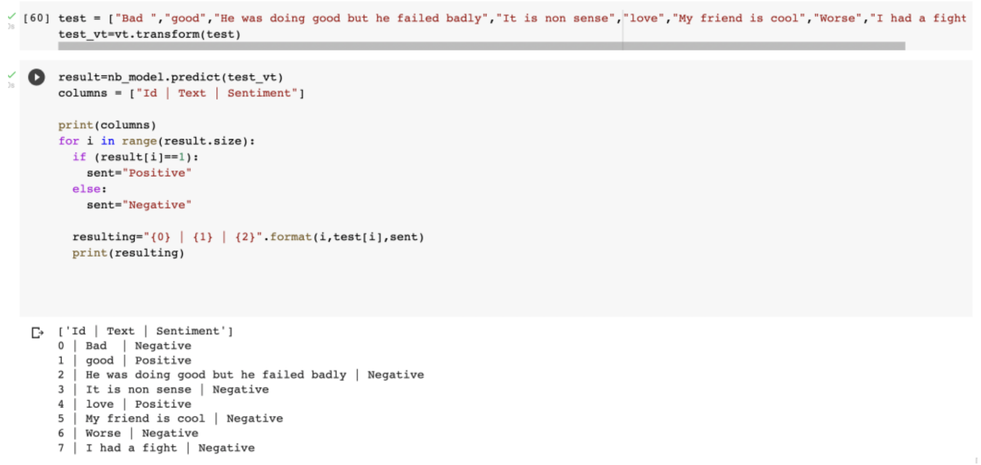
Result Analysis and Model Prediction

The four models produced varied results when the accuracy scores in the test dataset were compared. The below screenshot compares the accuracy for the four ML models. SVC turned out to have the highest accuracy with a value of 86%, followed by Random Forest, Multinomial NB and KNN. Measuring the accuracy of a classification model is a simple way of measuring the effectiveness of the model. In other words, it describes how the model performs across all classes, but it can be misleading by itself. Our takeaway is that we should not rely totally only on this metric to evaluate how well a specific model is performing.



Testing our Model

We tried testing out models by passing a set of random user inputs to the code to check if our model was able to associate correct sentiment to our text. Based on the result, our model was able to accurately predict the sentiment associated with the text. For example, a statement **“bad”** was given a Negative sentiment, whereas statement **“love”** or **“good”** were given positive sentiment.

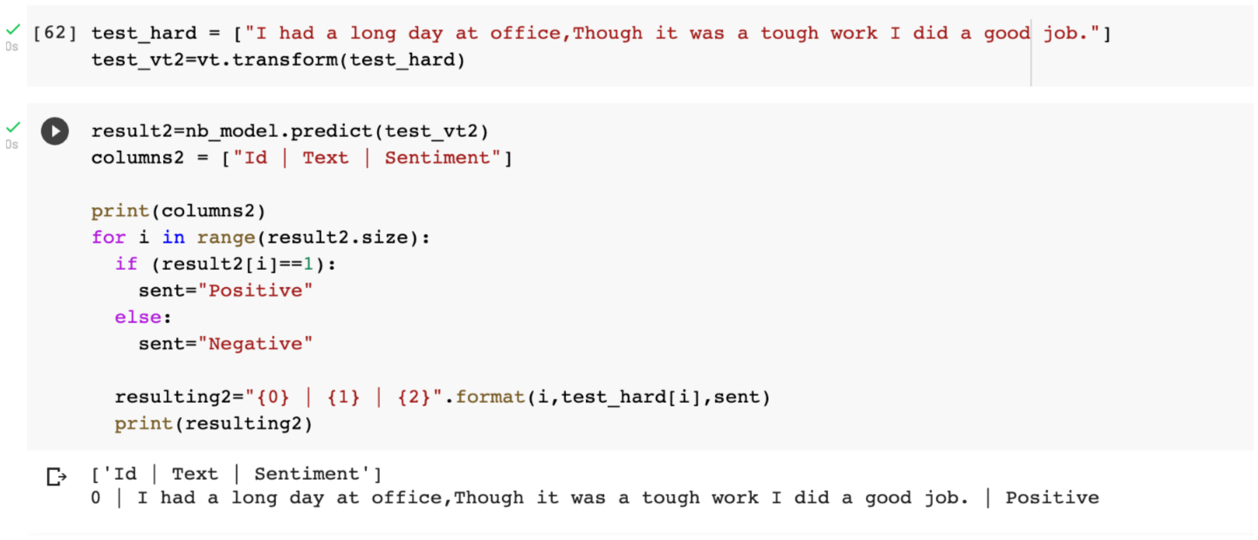


```
[60] test = ["Bad ","good","He was doing good but he failed badly","It is non sense","love","My friend is cool","Worse","I had a fight
     test_vt=vt.transform(test)
```

```
result=nb_model.predict(test_vt)
columns = ["Id | Text | Sentiment"]

print(columns)
for i in range(result.size):
  if (result[i]==1):
    sent="Positive"
  else:
    sent="Negative"

  resulting="{0} | {1} | {2}".format(i,test[i],sent)
  print(resulting)
```

```
['Id | Text | Sentiment']
0 | Bad  | Negative
1 | good | Positive
2 | He was doing good but he failed badly | Negative
3 | It is non sense | Negative
4 | love | Positive
5 | My friend is cool | Negative
6 | Worse | Negative
7 | I had a fight | Negative
```

We also tried testing our model using long sentences with a mixture of both positive and negative words like **“tough”** and **“good”.** The model was successfully able to identify the overall sentiment of the sentence.



```
[62] test_hard = ["I had a long day at office,Though it was a tough work I did a good job."]
     test_vt2=vt.transform(test_hard)
```

```
result2=nb_model.predict(test_vt2)
columns2 = ["Id | Text | Sentiment"]

print(columns2)
for i in range(result2.size):
  if (result2[i]==1):
    sent="Positive"
  else:
    sent="Negative"

  resulting2="{0} | {1} | {2}".format(i,test_hard[i],sent)
  print(resulting2)
```

```
['Id | Text | Sentiment']
0 | I had a long day at office,Though it was a tough work I did a good job. | Positive
```

Conclusion

Sentiment analysis can be extremely useful for various purposes like commercial applications, marketing analysis, public relations, product reviews, net promoter scoring, product feedback, and customer service. As a user, we use various products and we have our own sentiments and opinions about them, we share those opinion by using social platforms (twitter, FB). In this case we are the customer however as a future IT employee we will be working for such industries/products so we will have experience of the both worlds as a customer as well as the creator of a product. In order to achieve complete customer satisfaction and improve our brand, sentiment analysis can be of great help by giving insights into the dataset.

Twitter is a popular social media for sharing opinions and many people openly share their views about brands and social issues, hence we chose to use a twitter dataset for our analysis. Also, it is easy to group and collate data which share similar opinions with a hashtag. So, as we wanted to train our model, we chose a dataset which consists of thousands of tweets. We implemented sentiment analysis using four different models – KNN, SVC, Random Forest and Multinomial Naïve bayes. After training our model we compared the precision of all our models and we concluded that SVM and Random Forest worked fairly well as their precision values were 86% and 84% respectively. Whereas, precision of Multinomial Naïve Bayes was found to be 73% and KNN performed poorly on our dataset with a precision of 42%. Also, to achieve parallelization we used PySpark so, based on the partition set, two jobs were created in parallel. Parallelization is extremely beneficial while dealing with large datasets and it helped in decreasing the time taken for processing.

Areas where sentiment analysis can be useful are:

1. A company that uses social media to its advantage

It creates new ways to become popular online, especially among young people. The trending topics and mention features on Twitter are really useful.

2. Improving client service

Few things are more important than providing outstanding customer service. Customer service can make or ruin a company's reputation. Scraping Twitter for genuine feedback allows you to improve your business by getting feedback from customers.

3.Training for chatbots

Sentiment analysis has benefits that go beyond supporting your human staff. If your website has a chatbot, sentiment analysis can improve it as well. This is because it can teach your chatbot to recognize and respond to the customer based on their mood.

References

Online References

[1] https://www.techopedia.com/definition/29695/sentiment-analysis

[2] https://towardsdatascience.com/sentiment-analysis-concept-analysis-and-applications-6c94d6f58c17

[3] https://developer.twitter.com/en/docs/tutorials/how-to-analyze-the-sentiment-of-your-own-tweets

Research papers

[1] A survey on sentiment analysis challenges - Doaa Mohey El-Din Mohamed Hussein

[2] Sentiment analysis algorithms and applications: A survey -Walaa Medhat , Ahmed Hassan , Hoda Korashy