[BD] Final Project README

Link to the Dataset
https://raw.githubusercontent.com/mandalbiswadip/data_storage/main/labeled_data.csv

The offline code can be run normally. The logistic regression is under Logistic regression_all.py and the naive bayes under NB_MR.py.

Alternatively, you may view the code on the following Databricks links:

Logistic Regression:
https://databricks-prod-cloudfront.cloud.databricks.com/public/4027ec902e239c93eaaa8714f173bcfc/3208215397689650/3799201722184814/1993809047708771/latest.html
L1 Logistic Regression:
https://databricks-prod-cloudfront.cloud.databricks.com/public/4027ec902e239c93eaaa8714f173bcfc/3208215397689650/2034321820104559/1993809047708771/latest.html
L2 Logistic Regression:
https://databricks-prod-cloudfront.cloud.databricks.com/public/4027ec902e239c93eaaa8714f173bcfc/3208215397689650/430692435418764/1993809047708771/latest.html
Naive Bayes:
https://databricks-prod-cloudfront.cloud.databricks.com/public/4027ec902e239c93eaaa8714f173bcfc/2591969351302898/749049199183343/1386004499334152/latest.html

The online streaming setup is described as follows:
1. launch your kafka environment using the usual commands:
In one terminal window

```
bin/zookeeper-server-start.sh config/zookeeper.properties
```

In another:

```
bin/kafka-server-start.sh config/server.properties
```

2. Then create two topics. Call them test and finaltopic.

```
bin/kafka-topics.sh --create --topic test
--bootstrap-server localhost:9092

bin/kafka-topics.sh --create --topic finaltopic
--bootstrap-server localhost:9092
```

3. Create a directory named checkpoint at the same level as the programs you are running. I have already provided one, but in case running the program throws errors, create an empty directory named checkpoint and rerun the program twice.

4. Launch your ELK.
Logstash should be configured with the logstash.conf file given in the submission.

5. Then, run perform_streaming using:

```
sh <path_to_spark_submit> --packages
org.apache.spark:spark-sql-kafka-0-10_2.12:3.1.1
<path_to_perform_streaming.py> <bootstrap-servers>
<checkpoint-dir> <input-topic> <output-topic>
```

for example for us:

```
sh /Users/Hema/Library/Python/3.8/bin/spark-submit
--packages
org.apache.spark:spark-sql-kafka-0-10_2.12:3.1.1
/Users/Hema/bdproject/perform_streaming.py
localhost:9092 checkpoint test finaltopic
```

Similarly, run tweet_fetcher

```
sh <path_to_spark_submit> --packages
org.apache.spark:spark-sql-kafka-0-10_2.12:3.1.1
<path_to_tweet_fetcher.py> <query>
```

For us it was:

```
sh /Users/Hema/Library/Python/3.8/bin/spark-submit
--packages
org.apache.spark:spark-sql-kafka-0-10_2.12:3.1.1
/Users/Hema/bdproject/tweet_fetcher.py elon
```

6. Go to Kibana, create an index pattern (hw3_final_index is what you need to match to) and visualize in the dashboard.