**Data-Intensive Computing**

**LAB 3: DATA ANALYTICS PIPELINE USING APACHE SPARK**

**Done by:**

**Charanya Sudharsanan – 50245956 - csudhars**

**Prachi Shah – 50248748 – pshah**

**Directory Location:**

Titanic Data Analysis

Data Collection and Cleaning

Lab3

Feature Engineering

Multi Class Classification

Testing

Readme

**Environment:**

1. **OS - MacOS**
2. **Spark installation for MacOS**
3. **Language used - Python**

**Commands:**

1. **Navigate to the folder with the file myNaive.py**
2. **Run the following code on terminal/cmd:**

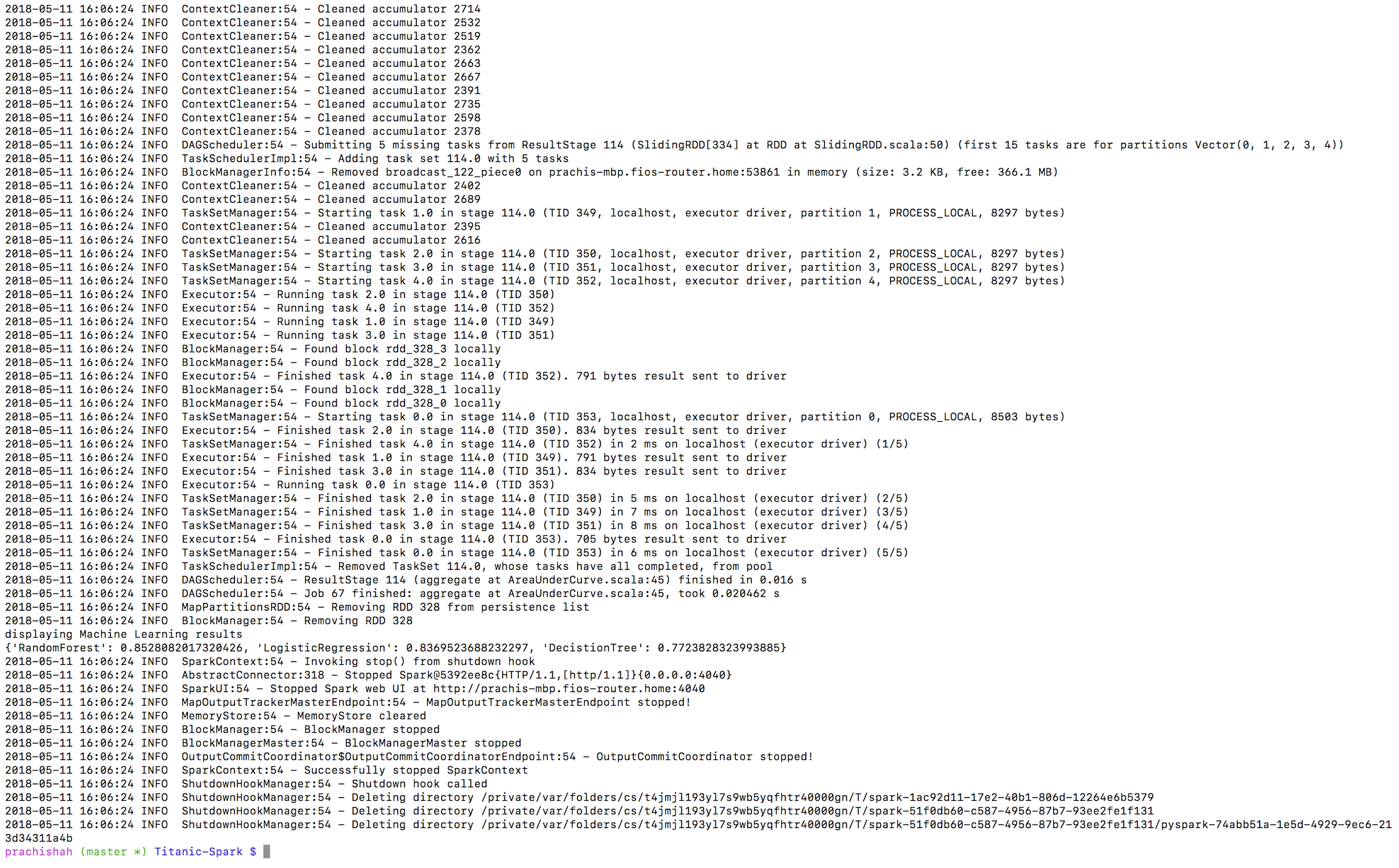
**spark-submit myNaive.py**

**TITANIC DATA ANALYSIS:**

1. **We load the dataset by reading the csv file as text in RDD data format. We load both the training and test data set.**
2. **We then transform RDD into dataframe for further manipulation.**
3. **We proceed by cleaning the data, where we remove non characters and normalize the data by filling in missing data.**
4. **Feature Engineering is performed by extracting their titles that’ll be used as labels. We index the categorical labels to passing it to machine learning algorithms .**
5. **Convert the features into vectors to apply MLLIB.**
6. **Split data into training and test data set and apply models from the ML/MLLIB and tune the parameters.**

**Input Files: Lab3 - > Titanic Data Analysis -> train.csv, test.csv**

**Output Screenshot after running it on pyspark:**

****

**DATA COLLECTION AND CLEANING:**

**Collecting data from NY Times:**

* Articles from NY Times API are collected using keyword Politics/Sports/Media/Business for specified period of time. We later divide them into test and train sets.
* Removed urls containing videos and scraped only the content of the articles
* Tweets are cleaned for further data manipulation.. (Cases, special characters, spaces, one letter words, redundant words are removed).
* Put each of the articles into a seperate text file and put them on seperate folders each for one category.

Data Collection Scripts : Lab 3 - > Data Collection and Cleaning -> Data Collection Scripts

Articles Collected : Lab 3 - > Data Collection and Cleaning -> Data -> Politics/Sports/Media/Business

**We collect some articles that we will use to test the model on unseen data.**

**Test Data Collected : Lab3 ->** Data Collection and Cleaning -> Test Data

**Further Cleaning is done on pyspark to remove stop words , nouns, prepositions, conjuncts etc to get key words using the nltk package.**

**Data Cleaning on pyspark :**

****

**The Stop words are removed and redundant words like modal verbs, prepositions, nouns etc are removed on pyspark.**

**Data consolidation:**

**To enable further computations on data, all the articles are consolidated into one csv file. The structure of the file is the following:**

**Column 1 - The article text**

**Column 2 - Label of the article [1 - Sports, 2 - Business, 3 - Media, 4 - Politics]**

**The file is called final.csv**

**Feature Engineering :**

**In pyspark , we preprocess the data by looping through each of the folder and collecting top 25 words for each category . The data is cleaned and all redundant words are removed as in above screenshot . We write the top 25 words into a csv file . These serve as features for the category.**

**Similary we do the same on each category of data and arrive at 25 X 4 features. These features are obtained from clean data.**

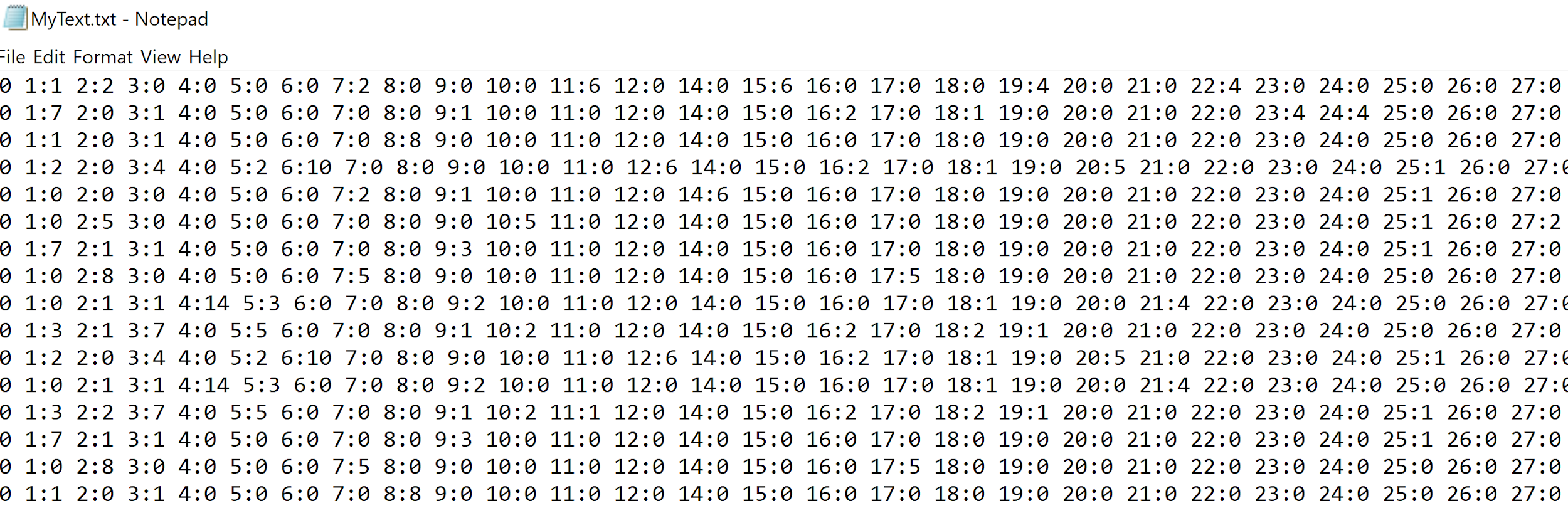
**pyspark file to calculate wordcount and generate csv :** Lab 3 - > Feature Engineering -> wordc.py

**Feature files :** Lab 3 - > Feature Engineering -> Feature ->bus\_wc.csv/med\_wc.csv/pol\_wc.csv/sp\_wc.csv

**Article list** -> Lab3 -> Feature Engineering -> traininglist.csv

**Now we create a document Matrix using these csv files and our articles .**

**Document matrix input -> Lab3 -> Feature Engineering -> MyText.txt**

****

**Scripts used to create the csv , article list and document matrix : Lab3 -> Feature Engineering -> Script -> csvgen.py , DocMatgen.py**

**Multi Class Classification :**

**We now use Naive Bayes and Logistic Regression to perform the classification by giving as an input our document matrix in the form of "libsvm". The data is split into training and testing data. The training data is fit to the model and we test the accuracy of the model on the testing data.**

**PySpark Scripts for Logistic regression and Naive Bayes :**

**Logistic Regression - Lab3 -> Multi Class Classification -> Logistic regression -> LR.py**

**Naive Bayes - Lab3 -> Multi Class Classification -> Naive bayes -> MyNaive.ipynb**

**Data cleaning and Spark processing for Logistic Regression:**

Input - final.csv

The input is first loaded in an RDD. This RDD is then converted to a Spark dataframe, with two columns, ‘FileContent’ and ‘label’.

Then, the following steps are done -

1. The FileContent data is Tokenised using RegexTokenizer. This will help in filtering stop words in the next step.
2. The Stop words are removed, using nltk package.
3. The term frequence (TF) of the FileContent are calculated, using the Spark ML package. The number of features are set to 20. This means that each token is hashed to one of the 20 hashcodes. And TF for each hash is calculated.
4. The Inverse Document Frequency is calculated, using the Spark ML package.
5. Each word then has a TF-IDF score.
6. The data is split into train and test data
7. The logistic regression model is made and training data
8. To evaluate the accuracy of the model, the test data is fit on the model. And the accuracy is calculated using the evaluator.

The following part is done for testing the model on unseen/new articles:

1. The steps 1 to 5 are followed as before.
2. The data is then fit in the model from the previous part and the accuracy is calculated.

Following are the results:

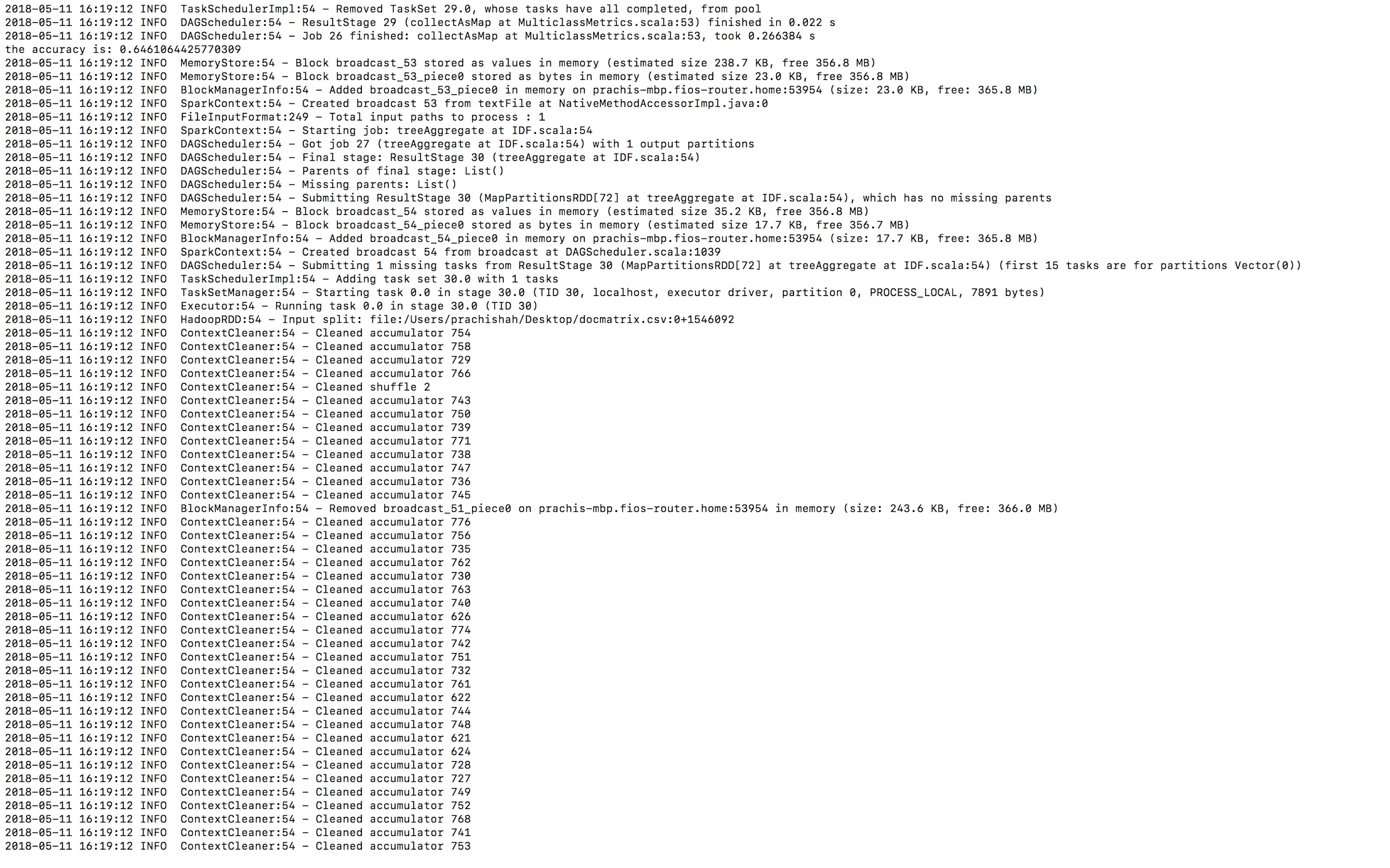
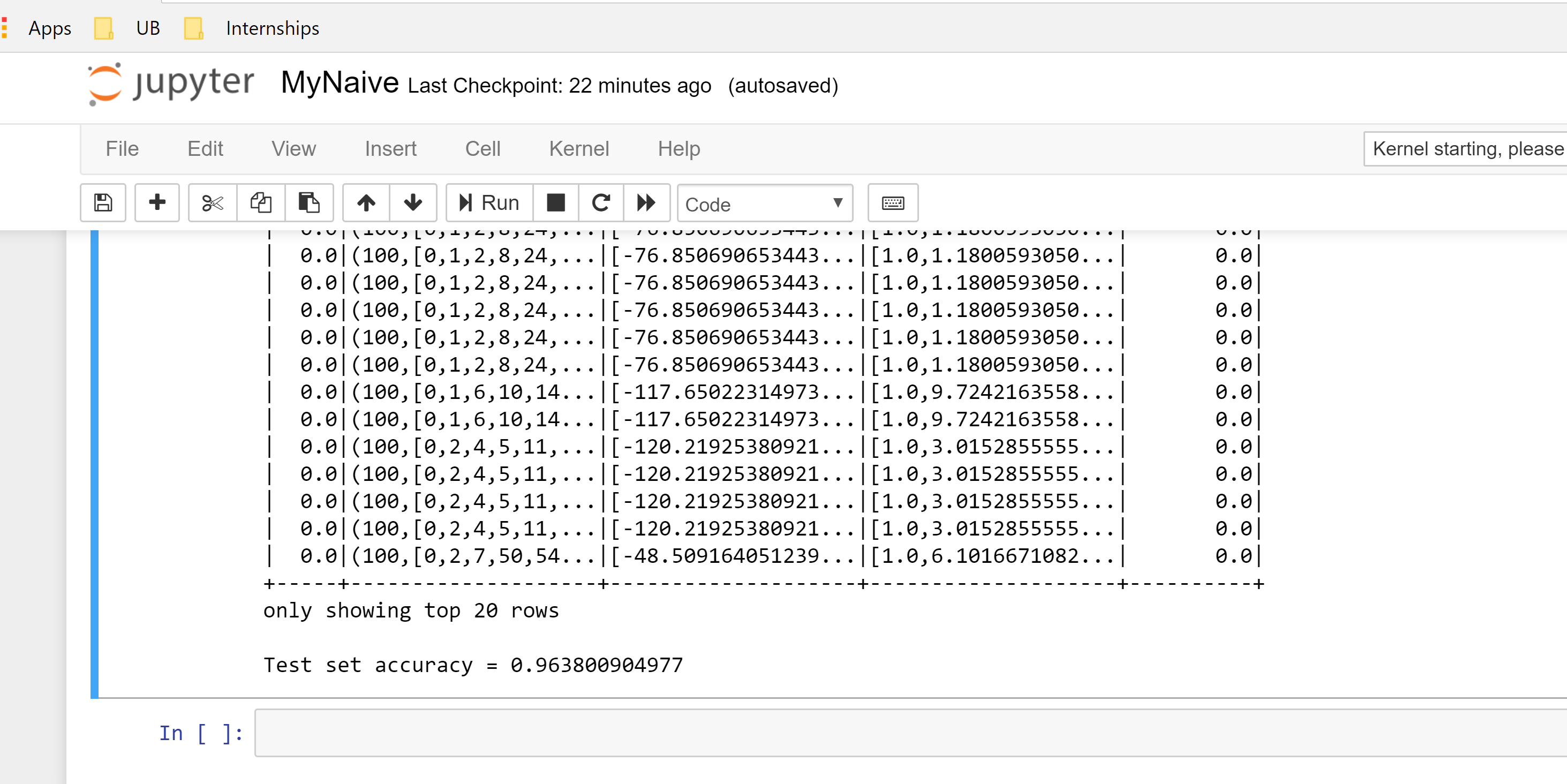


Fig1 - Accuracy for testing data

**Naïve Bayes :**

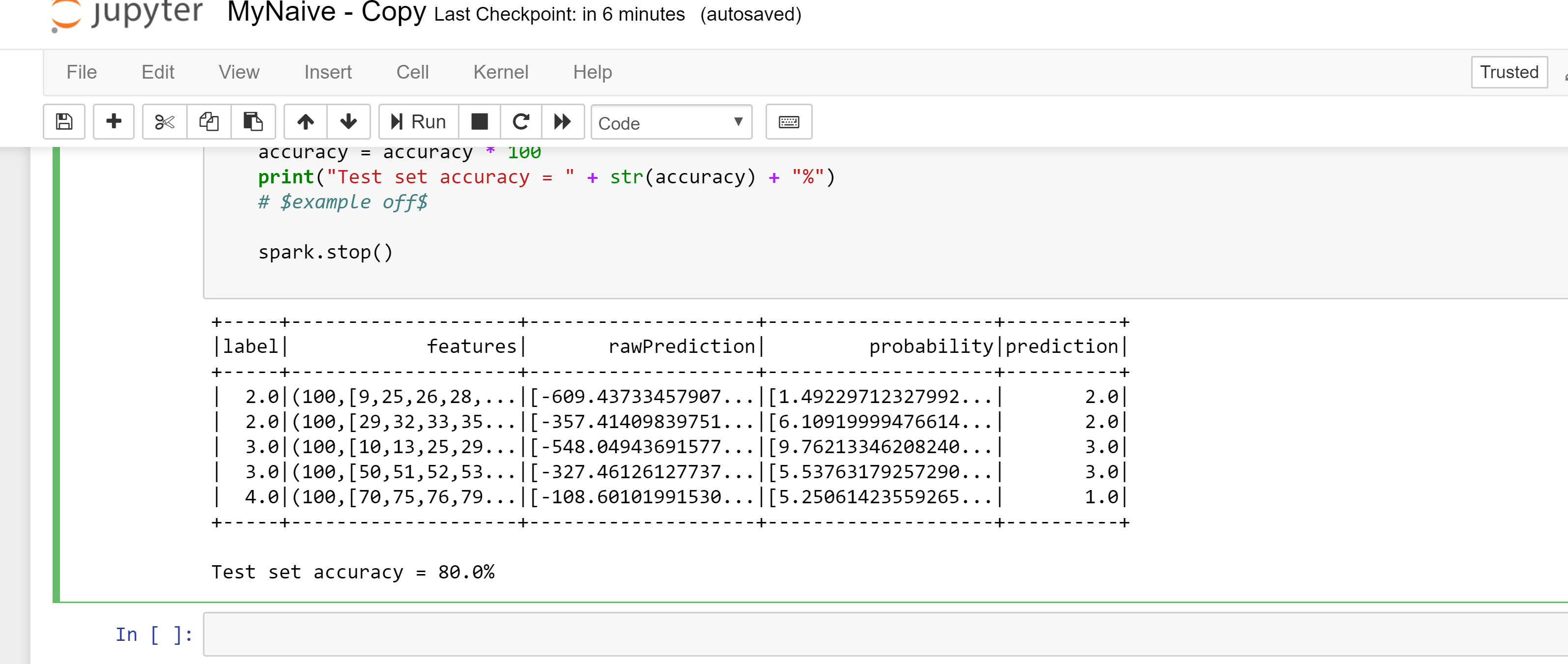


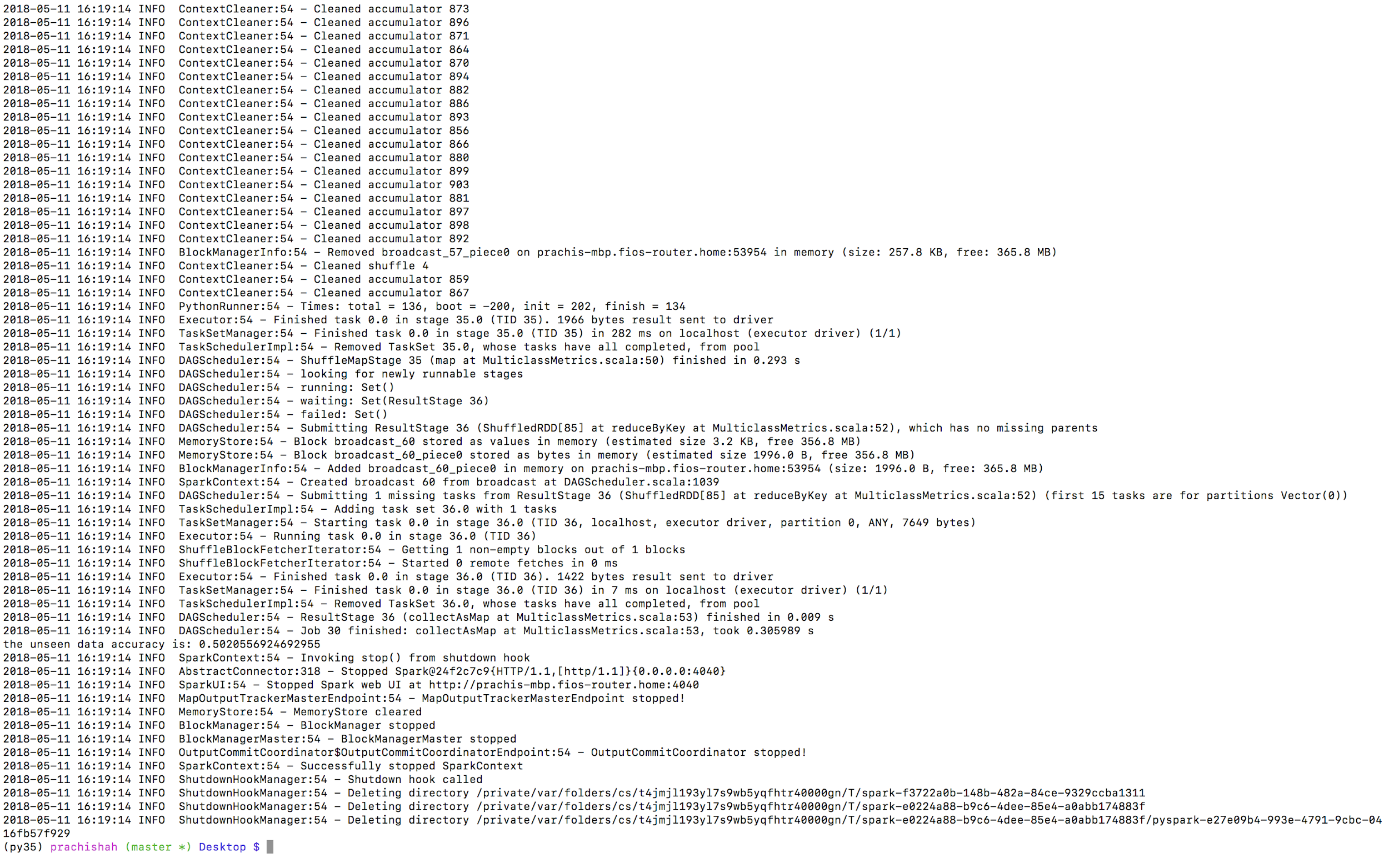
**Testing:**

**We test the above model on new unseen data. The test data has already been collected in the above step and kept separately.**

**This data is unseen and can be used to check the accuracy of the two models on unseen data.**

**Naïve Bayes : Tested on unseen Data**



**Logistic regression : unseen Data Testing**

**Test Set Accuracy for Naïve Bayes: 80%**

**Test Set Accuracy for Logistic Regression : 50%**

**We tested on different sizes of data to get an optimum accuracy.**

**Testing for naïve bayes - > lab3 -> Testing**

**Testing for Logistic regression - > Done Along with test data .**

**Conclusion: Thus , Data has been collected from NYTimes api , cleaned and consolidated in pyspark and passed through Machine learning classification models after feature engineering to get an optimum accuracy.**