MEF UNIVERSITY

TEXT CLASSIFICATION USING APACHE SPARK

Capstone Project

Umut Rezan Azizoğlu

İSTANBUL, 2018

MEF UNIVERSITY

TEXT CLASSIFICATION USING APACHE SPARK

Capstone Project

Umut Rezan Azizoğlu

Advisor: Prof. Dr. Özgür Özlük

İSTANBUL, 2018

MEF UNIVERSITY

Name of the project: Text Classification Using Apache Spark
Name/Last Name of the Student: Umut Rezan Azizoğlu
Date of Thesis Defense: 28/12/2018

I hereby state that the graduation project prepared by Umut Rezan Azizoğlu has been completed under my supervision. I accept this work as a "Graduation Project".

dd/mm/yyyy
Prof. Dr. Özgür Özlük

I hereby state that I have examined this graduation project by Azizoğlu has been completed under my supervision. I accept this work as a "Graduation which is accepted by his supervisor. This work is acceptable as a graduation project and the student is eligible to take the graduation project examination.

dd/mm/yyyy
Prof. Dr. Özgür Özlük

Director
of
Big Data Analytics Program

We hereby state that we have held the graduation examination of _____ and agree that the student has satisfied all requirements.

THE EXAMINATION COMMITTEE

Committee Member	Signature
1. Prof. Dr. Özgür Özlük	……..
2. ……..	……..

Academic Honesty Pledge

I promise not to collaborate with anyone, not to seek or accept any outside help, and not to give any help to others.

I understand that all resources in print or on the web must be explicitly cited.

In keeping with MEF University's ideals, I pledge that this work is my own and that I have neither given nor received inappropriate assistance in preparing it.

U. Rezan Azizoğlu 28/12/2018 Signature

EXECUTIVE SUMMARY

TEXT CLASSIFICATION USING APACHE SPARK

Umut Rezan Azizoğlu

Advisor: Prof. Dr. Özgür Özlük

DECEMBER, 2018, 29 page

One of the biggest problems of enterprises which are marketplace e-commerce business model with social platform; The improper communication of their social platform is the negative impact of the customer experience and the damage of the brand's value both materially and morally.

As the number of daily commentaries is in numbers that cannot be read manually with optimal human resources in terms of company profitability, the interpretation modules in social market places are left unconscious.

With this Project; established a model that prevents sentences that spoil the customer experience in their social platforms.

Both data preparation and machine learning model were developed on Databricks notebook, using the apache spark platform with SparkML libraries and Pyspark language. The “Text Classification” approach is adopted when determining the model.

Key Words: text classification, sentiment analysis, Apache Spark, Python (Pyspark), Databricks

ÖZET

APACHE SPARK İLE METİN SINIFLANDIRMA

Umut Rezan AZİZOĞLU

Tez Danışmanı: Prof. Dr. Özgür Özlük

ARALIK,2018, 29 sayfa

Sosyal platformu bulunan,elektronik pazar yeri iş modeliyle çalışan girişimlerin, en büyük problemlerinden biri; sosyal mecralarında ki uygunsuz yorumların, müşteri deneyimini olumsuz etki etmesi ve girişimin marka değerinin hem maddi hem manevi zarar görmesidir.

Günlük yorum sayılarının şirket karlılığı açısından, optimal insan kaynağı ile manuel olarak okunamayacak sayılarda olması nedeniyle çoğunlukla sosyal pazar yerlerinde ki yorumlaşma modülleri deyim yerindeyse başıboş bırakılmaktadır.

Bu Proje ile; bu durumu çözmek amacıyla girişimlerin sosyal mecralarında müşteri deneyimini bozan cümleleri engelleyen bir model geliştirilmiştir.

Hem mevcut datanın hazırlığı, hem de Makine öğrenmesi modeli; databricks notebook kullanılarak, Apache Spark üzerinden Python(Pyspark) dili ile sparkml kütüphaneleri kullanılarak geliştirilmiştir. Model belirlenirken metin sınıflandırma yaklaşımı benimsenmiştir.

Anahtar Kelimeler: metin kategorileştirme, sentiment analizi, Apache Spark, Python (Pyspark), Databricks

TABLE OF CONTENTS

1. INTRODUCTION

One of the widely used natural language processing task in different business problems is “Text Classification”. The goal of text classification is to automatically classify the text documents into one or more defined categories.

“Text Classification” have 2 common type. Fist one is topic classification, categorizing a text document into one of a predefined set of topics. In many topic classification problems, this categorization is based primarily on keywords in the text.

Another common type of text classification is sentiment analysis, whose goal is to identify the polarity of text content: the type of opinion it expresses. This can take the form of a binary like/dislike rating, or a more granular set of options, such as a star rating from 1 to 5. [14]

Some examples of text classification are:

- Understanding audience sentiment from social media,
- Detection of spam and non-spam emails,
- Auto tagging of customer queries, and
- Categorization of news articles into defined topics.[14]

This project will focus on sentiment analysis approach, Sentiment analysis aims to estimate the **sentiment polarity** of a body of text based solely on its content. The sentiment polarity of text can be defined as a value that says whether the expressed opinion is **positive** (*polarity=1*), **negative** (*polarity=0*), or neutral. [15]

Like other machine learning approach on “Text Classification”, sentiment analysis has higher recall but a lower precision.[2] The project also will descibe how overcome this issue

1.1. About Data

Data are provided from Modacruz. Modacruz is a leading secondhand fashion, marketplace e-commerce. Since business model is C2C, users can communicate with the product under the listed products through the comment feature. There are approximately 22500 interpretations in Modacruz daily.

In this study, the used dataset encloses the randomly selected comments. The first, dataset contains a total of 34552 comments. And these comments are labeled. Although 5912 of the comments do not matches with the policies of the site, 28640 of them have not seen any harm.

And second iteration process – The reason of this methodology explained in the project document in “The Project Concerns” section- contains a total of x comments. Although x of the comments do not comply with the policies of the site, x of them have not seen any harm.

The dataset has two columns;

i. Label: label of comments. If it is approvable, takes 0 if not 1
ii. Text: content text of comments

2. DATABRICKS AND APACHE SPARK

2.1. Databricks Notebook

Azure Databricks is an Apache Spark-based analytics platform optimized for the Microsoft Azure cloud services platform. Designed with the founders of Apache Spark, Databricks is integrated with Azure to provide one-click setup, streamlined workflows, and an interactive workspace that enables collaboration between data scientists, data engineers, and business analysts. Databricks providing a zero-management cloud platform built around Spark that delivers fully managed Spark clusters, an interactive workspace for exploration and visualization, a production pipeline scheduler, and a platform for powering Spark-based applications. [11]

You can use Scala, Python, Java, Sql, or R in Databricks notebooks or a same notebook. Notebooks can be shared by multiple sessios, Libraries can be imported and called in notebooks.

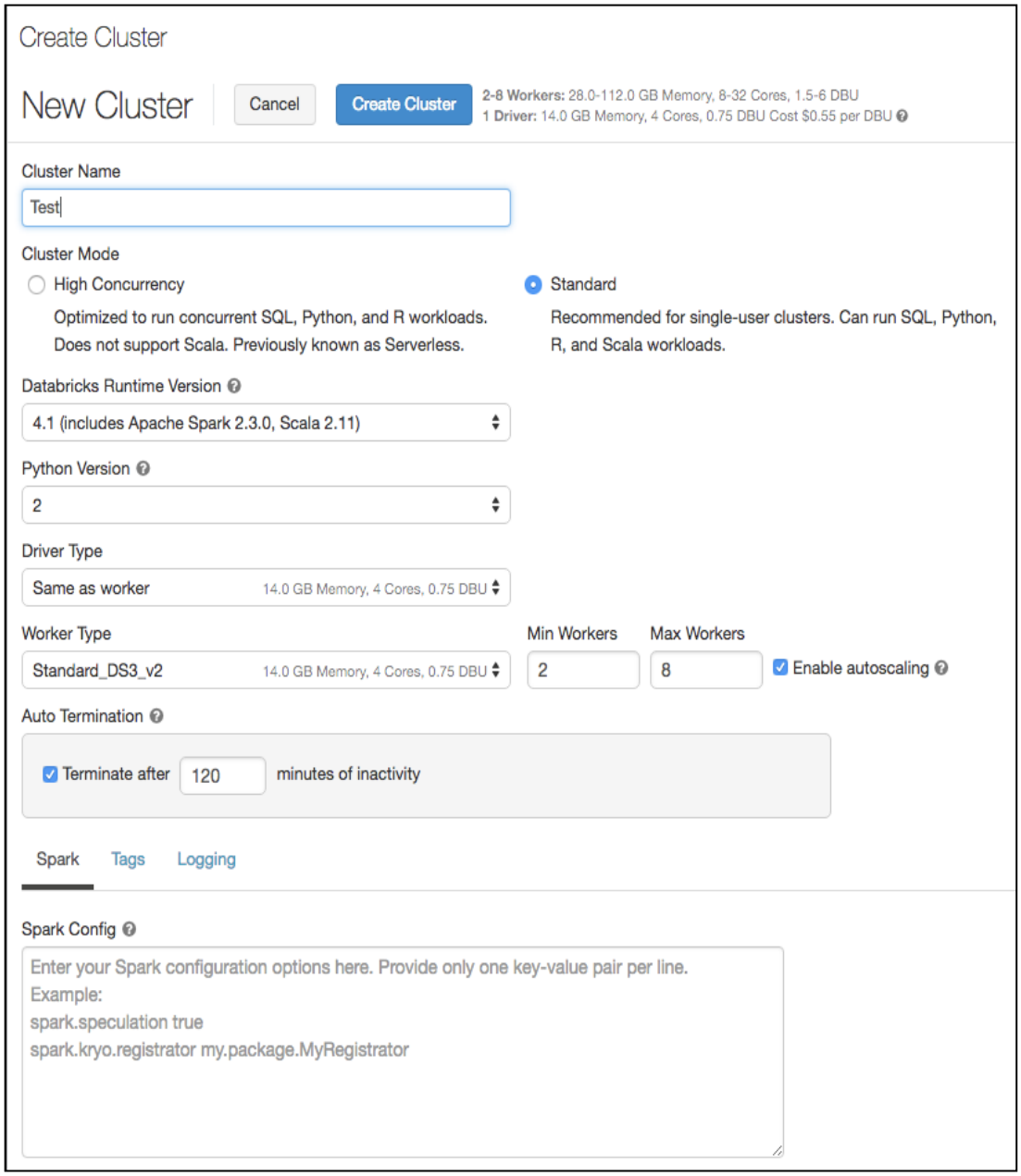

Figure 1**: Databricks cluster configuration screen**

2.2. Apache Spark

Apache Spark is an open-source cluster computing framework for big data processing. Just like Hadoop MapReduce, it also works with the system to distribute data across the cluster and process the data in parallel. Apache Spark achieves high performance for both batch and streaming data, using a state-of-the-art DAG (directed

acyclic graph) execution engine that supports cyclic data flow and in-memory computing.[9]

Spark uses master/slave architecture i.e. one central coordinator and many distributed workers. The central coordinator is called the driver. The driver runs in its own java process. These drivers communicate with a potentially large number of distributed workers called executors. Each executor is a separate java process. A Spark Application is a combination of driver and its own executors. With the help of cluster manager, a Spark Application is launched on a set of machines. Standalone Cluster Manager is the default built in cluster manager of Spark. Apart from its built-in cluster manager, Spark works with some open source cluster manager like Hadoop Yarn, Apache Mesos etc. Spark is 100 times faster than Bigdata Hadoop and 10 times faster than accessing data from disk. [13]

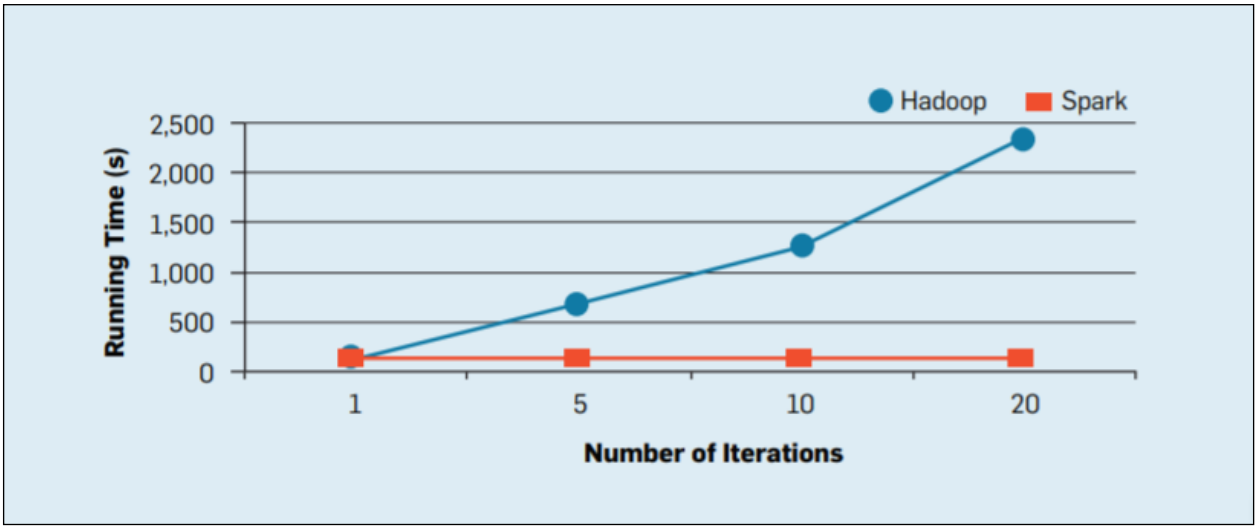

Figure 2: **Performance of logistic regression in Hadoop MapReduce vs. Spark for 100GB of data on 50 m2.4xlarge EC2 nodes.[10]**

2.2.1. Spark ML

Spark comes with a library containing common machine learning (ML) functionality, called Spark ML, previously MLlib. Spark ML provides multiple types of machine learning algorithms, including classification, regression, clustering, and collaborative filtering, as well as supporting functionality such as model evaluation and data import. All these methods are designed to scale out across a cluster. Recently, SparkML switched its primary data source from RDDs to dataframes to take advantage of the dataframe functionality, which provides a more intuitive interface and improved processing. In its current state, SparkML only supports parallel algorithms that run well on clusters, so it provides a somewhat limited set that should be mainly used on large datasets. For smaller datasets, other tools such as Python's scikit-learn or R should be used. [3]

2.2.2. Spark Machine Learning Pipeline

In general, a machine learning pipeline describes the process of writing code, releasing it to production, doing data extractions, creating training models, and tuning the algorithm. It should be a continuous process as a team works on their ML platform. But for Apache Spark a pipeline is an object that puts transform, evaluate, and fit steps into one object pyspark.ml.Pipeline. [16]

MLlib standardizes APIs for machine learning algorithms to make it easier to combine multiple algorithms into a single pipeline. A Pipeline is specified as a sequence of stages, and each stage is either a Transformer or an Estimator. These stages are run in order, and the input DataFrame is transformed as it passes through each stage. For Transformer stages, the transform() method is called on the DataFrame. For Estimator stages, the fit() method is called to produce a Transformer (which becomes part of the PipelineModel, or fitted Pipeline), and that Transformer's transform() method is called on the DataFrame. [9]

A big benefit of using ML Pipelines is hyperparameter optimization. With paramgridbuilder function, it is possible to tune both feature extraction parameters and ml algorithm parameters.

Also Parameters belong to specific instances of Estimators and Transformers. For example, if we have two Logistic Regression instances lr1 and lr2, then we can build

a ParamMap with both maxIter parameters specified: ParamMap(lr1.maxIter -> 10, lr2.maxIter -> 20). This is useful if there are two algorithms with the maxIter parameter in a Pipeline.

Pipelines are a simple and effective way to manage complex machine learning workflows. Overall, the Pipeline API is a major step in making machine learning scalable and easy.

3. TEXT CLASSIFICATIONS STEPS

In the following sections of the study, the text categorization project will be explained step by step.

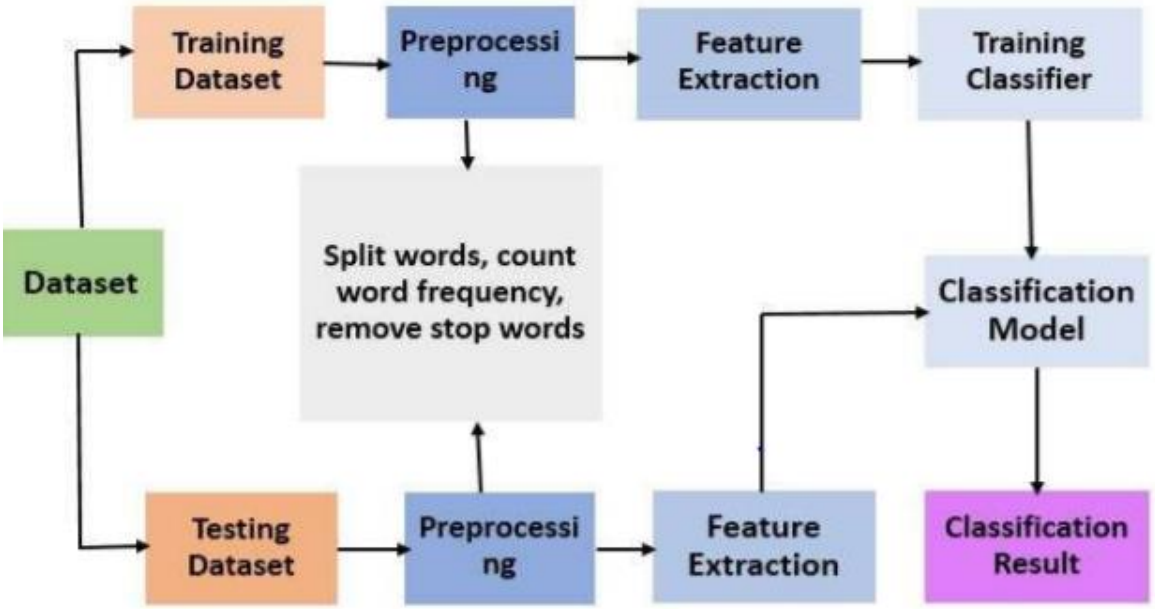

Figure-3**: Traditional Text Classification Steps**

3.1. RegexTokenizer

Tokenizer performs the tokenization on a string of words that are separated by spaces and returns an array of words. If there is a need to perform tokenization with a different delimiter, then you can use RegexTokenizer. A regex based tokenizer that extracts tokens either by using the provided regex pattern to split the text (default) or repeatedly matching the regex (if gaps is false). Optional parameters also allow filtering tokens using a minimal length. It returns an array of strings that can be empty.[9]

Regextokinezer is not just provide tokenizing stage, it also takes role part of data cleaning and text normalization stage for text categorization.

In this project, pattern parameter used for split and clean words. Pattern is shown below which is used in this project:

```
pattern=' |,|;|-|_|\*|\t|\!|\.|\*|\:|\(|\|\"|\&|\$|\|\#|\}|\]|\[|\)|\{|\/|\'|<|>'
```

Generally, words, which are not match with company policy, are using with punctuation (for example “d*ş&rd& buluşalım”), for this reason regextokenizer plays important role.

Also Uppercase and Lowercase usage is not standard in the dataset; so we need to convert all words or letters to Lowercase with “toLowercase” parameters.

3.2. Stops Words Remover

Stop words are words which should be excluded from the input, typically because the words appear frequently and don’t carry as much meaning.

StopWordsRemover takes as input a sequence of strings (e.g. the output of a Tokenizer) and drops all the stop words from the input sequences. The list of stopwords is specified by the stopWords parameter. Default stop words for some languages are accessible by calling StopWordsRemover.loadDefaultStopWords(language), for which available options are “danish”, “dutch”, “english”, “finnish”, “french”, “german”, “hungarian”, “italian”, “norwegian”, “portuguese”, “russian”, “spanish”, “swedish” and “turkish”. A boolean parameter caseSensitive indicates if the matches should be case sensitive (false by default). [9]

These are Turkish stop words:

```
['acaba', 'ama', 'aslında', 'az', 'bazı', 'belki', 'biri',
'birkaç', 'birşey', 'biz', 'bu', 'çok', 'çünkü', 'da', 'daha', 'de',
'defa', 'diye', 'eğer', 'en', 'gibi', 'hem', 'hep', 'hepsi', 'her',
'hiç', 'için', 'ile', 'ise', 'kez', 'ki', 'kim', 'mı', 'mu', 'mü',
'nasıl', 'ne', 'neden', 'nerde', 'nerede', 'nereye', 'niçin', 'niye',
'o', 'sanki', 'şey', 'siz', 'şu', 'tüm', 've', 'veya', 'ya', 'yani']
```

It is possible to add manually some special words; But in this project isn't used default stop words, decided to remove characters at below, Cause as mentioned before data have too much punctuation and too many words not grammatical. For this reason, we need special approach.

Dataset contains, in terms of data analysis, too many meaningless symbols; which are shown below. So; In project, these symbols are removed from dataset.

```
stopWordstr = ['♡', '□', '😑', '🙏', '🌸', '🙌', '😇', '😋', '😢', '☆', '🌼', '👧',
'🏵', '💗', '💞', '😋', '🌺', '😅', '🖼', '👧', '✔', '😌🙏', '😍', '👍', ':)', '👏',, '🤧',
'😤', '😑', ':(', '😋', '👌', '💏', '✌', ':))', ':)))', '🤡','😉','😄']
```

3.3. Term frequency & inverse document frequency (TF-IDF)

TF-IDF is an efficient and simple algorithm for matching words in a text to documents that are relevant to that text. From the data collected, we see that TF-IDF returns documents that are highly relevant to a particular text. If a user were to input a text for a particular topic, TF-IDF can find documents that contain relevant information on the text. Furthermore, encoding TF-IDF is straightforward, making it ideal for forming the basis for more complicated algorithms and text retrieval systems (Berger et al, 2000). Despite its strength, TF-IDF has its limitations. In terms of synonyms, notice that TF-IDF does not make the jump to the relationship between words. Going back to (Berger & Lafferty, 1999), if the user wanted to find information about, say, the word dışardan, TF-IDF would not consider documents that might be relevant to the query but instead use the word "dşrdn". For large document collections, this could present an escalating problem.[4]

In Spark (TF-IDF) is a feature vectorization method widely used in text mining to reflect the importance of a term to a document in the corpus. Denote a term by t, a document by d, and the corpus by D. Term frequency TF(t,d) is the number of times that term tt appears in document dd, while document frequency DF(t,D) is the number of documents that contains term tt. If we only use term frequency to measure the importance, it is very easy to over-emphasize terms that appear very often but carry little information about the document, e.g. “de”, “ve”, and “ben”. If a term appears very often across the corpus, it means it doesn’t carry special information about a particular document. Inverse document frequency is a numerical measure of how much information a term provides:

$$IDF(t,D) = \log|D|+1 / DF(t,D)+1$$

Where |D||D| is the total number of documents in the corpus. Since logarithm is used, if a term appears in all documents, its IDF value becomes 0. Note that a smoothing term is applied to avoid dividing by zero for terms outside the corpus. The TF-IDF measure is simply the product of TF and IDF:

$$TFIDF(t,d,D)=TF(t,d)\cdot IDF(t,D)$$

There are several variants on the definition of term frequency and document frequency. In Spark MLlib separate TF and IDF to make them flexible.

HashingTF is a Transformer which takes sets of terms and converts those sets into fixed-length feature vectors. In text processing, a “set of terms” might be a bag of words. HashingTF utilizes the hashing trick. [9]

Hashing Trick is especially suitable for the almost linearly separable training set, where the training set is large and very high dimensional. Hashing trick is a complementary variation of kernel trick hashing trick hashes very high dimensional input vector to a lower dimensional feature space. The new feature space preserves sparsity and consumes less space compared to that of the original input matrix. [5]

3.4. VectorAssembler

VectorAssembler is a transformer that combines a given list of columns into a single vector column. It is useful for combining raw features and features generated by different feature transformers into a single feature vector, in order to train ML models like

logistic regression and decision trees. VectorAssembler accepts the following input column types: all numeric types, boolean type, and vector type. In each row, the values of the input columns will be concatenated into a vector in the specified order. [9]

The VectorAssembler is used to concatenate all attributes into a single vector. This vector attribute is placed under Features in the data frame.

4. PERFORMANCE OF CLASSIFICATION ALGORITHM

In this study, selected machine learning algorithm was Logistic regression. Also studied with Naïve Bayes, Support Vector Machine and XGboost Classifier. But Logistic regression gave best solutions.

4.1. Configuration

When working with Databricks, the configuration of the Spark Cluster affects the performance as well as the parameters of the relevant ML algorithm. Therefore, specifying the relevant configuration; contains information on the scope of the study.

Here are Configuration of Cluster:

- Databricks Runtime Version: 4.1 (includes Apache Spark 2.3.0, Scala 2.11)
- Driver Type: 32.0 GB Memory, 16 Cores, 2 DBU
- Worker Type: 32.0 GB Memory, 16 Cores, 2 DBU
- Number of workers: 2

4.2. Parameter Tuning

One of the most critical points that determine performance in ML studies is to tune the parameters of the relevant algorithm.

Although the relevant parameters vary according to the content of the data set; The study showed that for text classification, if TF-IDF fetaure extraction is used, the most important effect is the numfeature parameter.

In the study, the numfeature parameter was set as 2^{21}.

Another effective parameter is the regparam parameter of logistic regression, regparam is L2 regularization, its penalizing models for being too complex.

In the study, the regparam parameter was set as 0.006.

4.3. Evaluation Metrics and Comparison of Algorithms

There are different measures to evaluate the performance of classification algorithms. In the Project, the confusion matrix is choosen for performance metrics and consider the following measures Test Error,Accuracy,Weightened Precision, Weightened Recall, F-measure.

Table -1 shows us the performance of the Logistic Regression model used in the project. And the table show us, the performances of Naive Bayes, SVM and XGboost. The most successful results were obtained by Logistic Regression.

After Logistic regression, SVM gave best result.

Algorithms/Metrics	Test Error	Accurracy	W-Precision	W-Recall	F-Measure
Logistic regression	0.0645	0.9355	0.9281	0.9355	0.9318
Naive Bayes	0.1435	0.8565	0.8777	0.8565	0.8639
SVM	0.0834	0.9166	0.9141	0.9166	0.9150
Xgboost	0.1286	0.8714	0.8739	0.8714	0.8458

Table-1 **Comparison of metrics of the Algorithms**

Noticeable point is Precision is not as good as Accuracy and Recall in Logistic Regression model, although, it is better than other algorithms.

Precision is a more important metric for Modacruz's business model. Blocking user’s comments also means disrupting the flow of purchases. Therefore, the correct positive predictions out of all the positive predictions takes important role.

The project dataset is unbalanced, although it was not preferred to use stratified sample in the project because all of the data was needed for an accurate modeling. Weighted precision and recall were preferred when evaluating the project due to unbalance problem.

Getting the precision and recall for each class, and weight by the count of data of each class. That will give the weighted precision and recall.

The formula of weighted precision is:

$$(pc1*|c1|)+(pc2*|c2|)|c1|+|c2|$$

pc1:precision of class 1

pc2: precision of class 2

c1: count of data of class 1

c2: count of data of class 2

5. CONCERNS OF THE PROJECT

5.1. More Data

The Project; has some concerns that need to solve. First one is labelling process; the process take too much time and it is not effective for labor. At the same time, model gives good accuracy, but precision is not good as accuracy, this situation, disrupts model efficiency.

For developing the model, the project need that much more data. For this reason, the project suggests iterative process to overcome this issue.

When selecting the data to be added, The project approach is the iteration process based on the outputs of the first model. To improve the model, the data that the model predicted, but observed as type I and type II errors were corrected and added to the previous data.

A limited number of data had to be selected in the reading of all data due to the identified difficulties. Therefore, instead of randomly selecting the data, instead of using label estimation. Probabilities are used; which are produced by the model.

The histogram of the two-day data was analyzed to decide which data range to select.

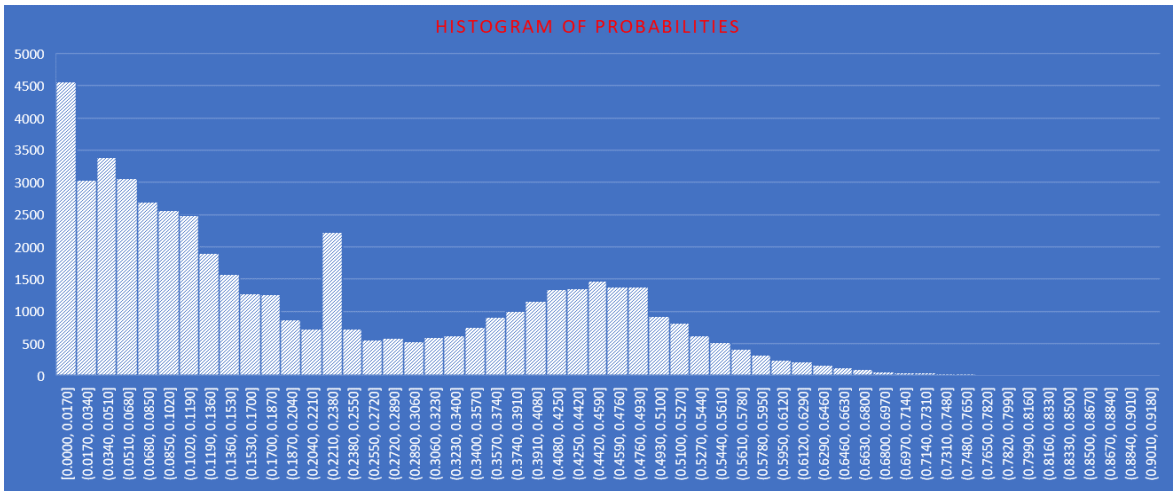

Figure-4: **Distribution of probability**

What Indicated of histograms, it was decided to read the labels under 0.017 and above the 0.50 probability and detect Type I and Type II errors.

After re-labelling data, model was re-trained. And this process repeated 4 times. After the last iteration, observed that the model was not further developed.

5.2. “Number” Problem

In some comments contain numbers, such as telephone numbers, and sharing telephone numbers do not matches with the policies of the site.

However, it was observed that the model was not very successful in estimating numbers. Because the data set also contains acceptable, too many numbers. To overcome this issue, two different method was used.

5.2.1 Multi-class Classification

Firstly, all non-acceptable number is labeled as ‘3’ and model is re-established with multi-class approach. But model is not performed as good as binary model.

5.2.2 Mark The Data

Another approach is to convert the numbers to any text. For this process, regexp_replace functions importing from pyspark.sql.functions. and all numbers in the “Text” column was replaced with “sayi”.

Here is code of this process:

```
df.withColumn('Text', regexp_replace('Text', '\d+', 'sayi'))
```

The desired result could not be achieved with this approach.

6. CONCLUSION

In this project, provided a solution, how to sort out undesirable text, if a company can not overcome the volume of the data. Whole project was designed with “Big Data” approach. Not just provided solution for volume it is also provided how to overcome Data variety (In this project it is “Text”) and Data velocity. For e-commerce business Real-Time processing is “have to”. At this point, serializing the model and productionized is take key role.

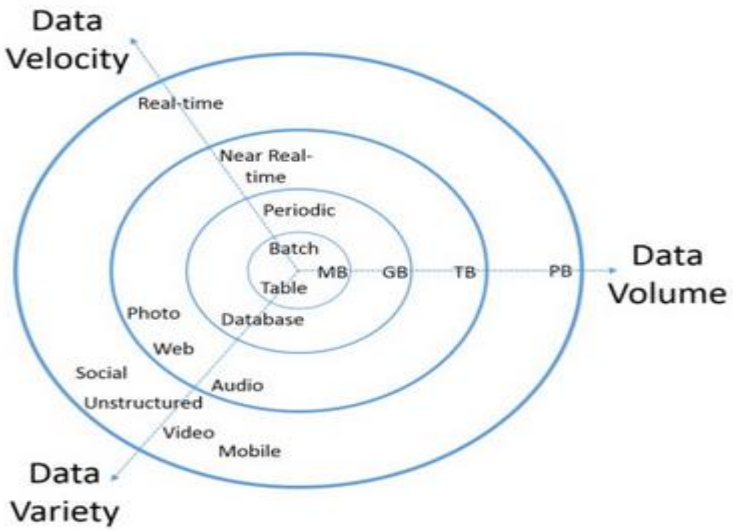

Figure-5**: 3V of Big Data**

For big data, Apache Spark is provided high performance, it is both handles easily, Data volume, Data variety and Data Velocity. Databricks notebook is developed by creator of Spark, so it matches well with Spark platform. Provides practical solutions such as easy management of Spark platform and the combination of many languages such as Scala, Python (Pyspark), Java,SQL and R language.

For our data set Logistic Regression gives best solution than Naïve Bayes, Support Vector Machine and XGboost Classifier.

Text Classifications have some methods to get achievement, for project dataset, given process is gives best solution, but each business may have different interpretation jargon, so comparing different algorithms takes important role.

To overcome load of labeling process and “Text Classification” lower precision issue, project suggest that iterative labeling process. Start modelling with optimal size of data and productionize, output of model, then labeling much more data which are contain Type I and Type II error with evaluating probability not label prediction.

8. REFERENCES AND CITING

- Yiming Yang (1999). *An Evaluation of Statistical Approaches to Text Categorization.* School of Computer Science, Carnegie Mel lon University Information Retrieval Volume 1 Issue 1-2, [1]
- Miji K Raju1, Sneha T Subrahmanian2, T.Sivakumar3, (2017). *A Comparative Survey on Different Text Categorization Techniques*, International Journal of Computer Science and Engineering Communications, Volume.5, Issue.3 1612-1618 [2]
- Zaharia, M., Franklin, M. J., Ghodsi, A., Gonzalez, J., Shenker, S., Stoica, I., Venkataraman, S. (2016). *Apache Spark: A Unified Engine for Big Data Processing*. Communications of the ACM. 59(11), 56–65. [3]
- Juan Ramos (2010). *Using TF-IDF to Determine Word Relevance in Document Queries*, Department of Computer Science, Rutgers University. [4]
- Robertson, S. (2004). *Understanding inverse document frequency: on theoretical arguments for IDF. Journal of Documentation, 503–520.* [5]
- Ravi, K., Ravi, V., & Shivakrishna, B. (2018). *Sentiment Classification Using Paragraph Vector and Cognitive Big Data Semantics on Apache Spark*. IEEE 17th International Conference on Cognitive Informatics & Cognitive Computing (ICCI*CC). [6]
- Luu, Hien (2018). Machine Learning with Spark SpringerLink, Springer. [7]
- Adam Simitos, (2016). *Text Mining in Twitter with Spark and Scala*. International Hellenic University. [8]
- https://spark.apache.org/ [9]
- https://docs.databricks.com/getting-started/concepts.html
- https://docs.microsoft.com/en-us/azure/azure-databricks/ [11]
- https://developers.google.com/machine-learning/guides/text-classification/
- https://data-flair.training/blogs/how-apache-spark-works/ [13]
- https://www.analyticsvidhya.com/blog/2018/04/a-comprehensive-guide-to-understand-and-implement-text-classification-in-python/ [14]

- https://medium.com/data-from-the-trenches/text-classification-the-first-step-toward-nlp-mastery-f5f95d525d73 [15]
- https://www.bmc.com/blogs/introduction-to-sparks-machine-learning-pipeline/ [16]

APPENDIX A

PYTHON (PYSPARK) SCRIPT

```
import pandas as pd
import numpy as np
from pyspark.sql import Row
from pyspark import SparkContext,SparkConf
from pyspark.sql import SparkSession
import nltk

from pyspark.ml.feature import HashingTF, IDF,
RegexTokenizer,StopWordsRemover,VectorAssembleR

from pyspark.ml import Pipeline
from pyspark.ml.classification import LogisticRegression,
from mleap.sklearn.preprocessing.data import FeatureExtractor, LabelEncoder,
from pyspark.ml.evaluation import
RegressionEvaluator,MulticlassClassificationEvaluator,BinaryClassificationEvaluator
from pyspark.ml.tuning import ParamGridBuilder, TrainValidationSplit,CrossValidator
from pyspark.mllib.regression import LabeledPoint
from pyspark.sql.functions import *

df = spark.table("df")
df = df.na.drop()
display(df.groupby('label').count())

train_df, test_df = df.randomSplit([0.65, 0.35], seed = 2018)

print("Training Dataset Count: " + str(train_df.count()))
print("Test Dataset Count: " + str(test_df.count()))
```

```
stopWordstr = ['♡', '◘', '😑', '🙏', '🌸', '🙌', '😇', '😊', '😢', '☆', '🌼', '👧', '🏵', '💟', '💞', '😊', '💮', '😅', '💷', '👧', '🙏🙏🙏', '🙏🙏', '✔', '😌🙏', '😍', '👍', '👍👍', '👍👍👍', ':)', '👏', '👏👏', '👏👏👏', '🤧', '😂', '😑😑', '😑', ':(', '😊', '👌👌', '👌', '🙇', '✌', ':))', ':)))', '😸','😉','😁','😭']

regexTokenizer = RegexTokenizer(inputCol="Text", outputCol="words", pattern=' |,|;|-|_|\*|\t|\!|\.|\*|\:|\(|\|\"|\&|\$|\|\#|\}|\]|\[|\)|\{|\/|\'|<|>',toLowercase=True)

remover = StopWordsRemover(inputCol="words", outputCol="filtered",stopWords =stopWordstr)

hashtf = HashingTF(inputCol="filtered", outputCol='tf')

idf = IDF(inputCol='tf', outputCol="tffeatures")

va = VectorAssembler(inputCols=["tf", "tffeatures"], outputCol="features")

lr = LogisticRegression()

pipelinelr = Pipeline(stages=[regexTokenizer,remover,hashtf, idf, va,lr])

paramGrid = (ParamGridBuilder()
        .addGrid(lr.regParam, [0.0001,0.006,0.003,0.01,0.03])
        .addGrid(idf.minDocFreq,[2,3,4])
        .addGrid(hashtf.numFeatures, [2**3,2**18,2**21])
        .addGrid(hashtf.binary, [True,False])
        .addGrid(lr.fitIntercept, [True,False])
        .addGrid(lr.standardization, [True,False])
        .addGrid(lr.elasticNetParam, [0.01,0.05,0.1])
        .addGrid(lr.aggregationDepth, [2,3])
        .addGrid(lr.maxIter,[5,1000])
        .addGrid(lr.family,['binomial'])
```

```
        .addGrid(lr.tol,[1e-06,1e-01])
        .build()  )

cvlr = CrossValidator(estimator=pipelinelr, evaluator=MulticlassClassificationEvaluator(),
estimatorParamMaps=paramGrid)

cvModel = cvlr.fit(train_df)
modellr = cvModel.bestModel

modellr.stages[4].extractParamMap()

predictions = modellr.transform(train_df)


predictions = predictions.select(col("label").cast("Float"),col("prediction"))
evaluator = MulticlassClassificationEvaluator(labelCol="label",
predictionCol="prediction", metricName="accuracy")
accuracy = evaluator.evaluate(predictions)
print("Test Error = %g" % (1.0 - accuracy))


evaluator = MulticlassClassificationEvaluator(labelCol="label",
predictionCol="prediction", metricName="accuracy")
accuracy = evaluator.evaluate(predictions)
print("Accuracy = %g" % accuracy)

evaluatorf1 = MulticlassClassificationEvaluator(labelCol="label",
predictionCol="prediction", metricName="f1")
f1 = evaluatorf1.evaluate(predictions)
print("f1 = %g" % f1)
```

```
evaluatorwp = MulticlassClassificationEvaluator(labelCol="label",
predictionCol="prediction", metricName=" weightedPrecision ")
wp = evaluatorwp.evaluate(predictions)
print("weightedPrecision = %g" % wp)

evaluatorwr = MulticlassClassificationEvaluator(labelCol="label",
predictionCol="prediction", metricName="weightedRecall ")
wr = evaluatorwr.evaluate(predictions)
print("weightedRecall = %g" % wr)
```