Mohmed Shah

5982571

ASSIGNMENT 1 - PROJECT 2: naïve BAYES FROM SCRATCH

Table of Contents

[Introduction 3](#_Toc88070517)

[Preprocessing 3](#_Toc88070518)

[Design and Implementation 10](#_Toc88070519)

[Implementation and Training and Testing performance 15](#_Toc88070520)

[Conclusion 25](#_Toc88070521)

  
  
**Assignment Cover Sheet**

|  |  |
| --- | --- |
| **Subject Code: CSCI316** |  |
| **Subject Name: BIG DATA MINING TECHNIQUES AND IMPLEMENTATION** |  |
| **Submission Type: Online** |  |
| **Assignment Title: Assignment 1 Project 2 Naïve Bayes** |  |
| **Student Name: M.S.Saalik Shah** |  |
| **Student Number: 5982571** |  |
| **Student Phone/Mobile No. 0507815780** |  |
| **Student E-mail: msss950**@uowmail.edu.au |  |
| **Lecturer Name: Dr Farhad Oroumchian** |  |
| **Due Date: 17/11/2021** |  |
| **Date Submitted: 17/11/2021** |  |

|  |  |
| --- | --- |
| **PLAGIARISM:** The penalty for deliberate plagiarism is FAILURE in the subject. Plagiarism is cheating by using the written ideas or submitted work of someone else. UOWD has a strong policy against plagiarism.  The University of Wollongong in Dubai also endorses a policy of non-discriminatory language practice and presentation.  **PLEASE NOTE:**STUDENTS MUST RETAIN A COPY OF ANY WORK SUBMITTED | **DECLARATION:** I/We certify that this is entirely my/our own work, except where I/we have given fully-documented references to the work of others, and that the material contained in this document has not previously been submitted for assessment in any formal course of study. I/we understand the definition and consequences of plagiarism.  **Signature of Student:** |

|  |  |  |
| --- | --- | --- |
| |  | | --- | | **Optional Marks:** | | **Comments:** | |

https://my.uowdubai.ac.ae/images/scissors.gif

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | **Lecturer Assignment Receipt**(To be filled in by student and retained by Lecturer upon return of assignment) | | | **Subject:** | **Assignment Title:** | | **Student Name:** | **Student Number:** | | **Due Date:** | **Date Submitted:** | | **Signature of Student:** | | |

https://my.uowdubai.ac.ae/images/scissors.gif

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | **Student Assignment Receipt** (To be filled in and retained by Student upon submission of assignment) | | | **Subject:** | **Assignment Title:** | | **Student Name:** | **Student Number:** | | **Due Date:** | **Date Submitted:** | | **Signature of Lecturer** | | |

# Introduction

Since this is an experimentation of our own Naïve bayes. I wanted to compare and see which methods had given me a better result and whether there is any correlation that makes the method create an excellent model and then document all my findings and end it with my conclusion to the discoveries found.

**Part 1**: Binary classes

**Part 2**: Multi classes or multi-Label

# Preprocessing

1. **Binary classes:**
2. **Unbalanced dataset:** -At the start we had **used sample clustering** to reduce the dataset to 8000 samples where 6000 was used for training and 2000 for testing as mentioned for the task, the preprocessing was done only for the “Type” attribute since it was in the string format which resulted in an excellent accuracy but not a satisfied recall since the dataset was not a balanced set (more on this in the design and Implementation section).

Then we tried to bin all the columns to get a better result, **except** the UDI and product ID (since they did not contribute at all for any improvement) **and** the different error types since they were the target classes. After binning, all the columns were differentiated by low, medium, and high **except** Type since it wasalready differentiated and “machine failure” column which was having binary values of 1 and 0 hence it was not binned which resulted in a good accuracy but an excellent recall. But this was still not a good classifier as the dataset was not balanced.

1. **Balanced Dataset:** -
2. **Under Sampling Dataset:** - We had first under sampled our dataset by **increasing** value 1 to 5000 from 339 and **reducing** value 0 to 5000 from 10000. This brings us to 10,000 instances to which we then **clustered sample**

the dataset to reduce the dataset to 8000 samples where 6000 was used for training and 2000 for testing as mentioned for the task, the preprocessing was done only for the “Type” attribute since it was in the string format which resulted to a very good accuracy, recall and precision.

Then we tried to bin all the columns to get a better result, **except** the UDI and product ID (since they did not contribute at all for any improvement) **and** the different error types since they were the target classes. After binning, all the columns were differentiated by low, medium, and high **except** Type since it wasalready differentiated and “machine failure” column which was having binary values of 1 and 0 hence it was not binned which resulted in a good accuracy, recall and precision. But it was **lower** than the previous experiment **without** binning.

1. **Over Sampling Dataset:** - We had first over sampled our dataset by **increasing** value 1 to 10,000 from 339 and **kept** value 0 to 10,000. This brings us to 20,000 instances to which we then **clustered sample**

the dataset **twice** by first reducing the 20,000 instances to 10,000 and then from 10,000 instances to 8000 samples where 6000 was used for training and 2000 for testing as mentioned for the task, the preprocessing was done only for the “Type” attribute since it was in the string format which resulted to a very good accuracy, recall and precision.

Then we tried to bin all the columns to get a better result, **except** the UDI and product ID (since they did not contribute at all for any improvement) **and** the different error types since they were the target classes. After binning, all the columns were differentiated by low, medium, and high **except** Type since it wasalready differentiated and “machine failure” column which was having binary values of 1 and 0 hence it was not binned which resulted in a good accuracy, recall and precision. But it was **lower** than the previous experiment **without** binning.

1. **Multi-classes or Multi-label**: - A new approach was decided to be taken into getting a good model, and this was multi-class. This approach was done by joining all the different Machine failure columns together and so now all the different Machine failure columns are in the Machine failure column.

To combine all the Machine failure columns together we first need to understand our dataset. We have **9631 instances** which are **labeled as 0** and **399 labeled as 1.**

So now we need to **know what type of failures** are there, because there is also a **possibility for a combination of two failures** or more. So to get the different combinations we will change all the 1s to the name of the failure and 0s to an empty value ""

Graphical user interface, text

Description automatically generated

Based on the above results we see that there are 12 labels, and out of those 11 labels are machine failures and 1 label is no machine failures. So now let’s check the frequency of each label

Table

Description automatically generated

From the above results we see that the **No machine failure should be equal to 9661** but we get 9652 which shows there must be something more in the data.

By going through the whole dataset based on the failures we see **that when RNF failure occurs the machine does not fail.**

**Table

Description automatically generated**

Based on the above results we see that RNF failure does not cause any machine failure but may require fixing or repairs which is beyond the scope of the project, but RNF is a failure but not machine failure.

But it still does not solve the issue of No machine errors being 9652 as shown above. So, this could mean that there could occur a machine failure without any of the failures occurring. In short, if all the known failures are 0, the machine can still fail. Let's find out if this is true.

Table

Description automatically generated

**So based on the above results we were right, there are instances where the machine failure occurs but none**

**of the other failures occur or are set to 1, so we need to distinguish it to another type. Hence bringing**

**the total types to 13 labels which are**

**0 = No Error**

**1 = TWFPWFOSF**

**2 = TWFRNF**

**3 = TWFOSF**

**4 = PWFOSF**

**5 = HDFPWF**

**6 = HDFOSF**

**7 = TWF**

**8 = RNF**

**9 = PWF**

**10 = OSF**

**11 = HDF**

**12 = UD, Machine failure equal to 1 without any known failure occuring, so we categorize it as UnDefined**

So now let’s try to classify the multi-label dataset

We will be doing it differently since we ran into a few errors when using the above dataset since there were **NaN** values that were difficult to remove and hence, we had to make a few changes to the code as well.

1. **Unbalanced dataset:** -At the start we had **used sample clustering** to reduce the dataset to 8000 samples where 6000 was used for training and 2000 for testing as mentioned for the task, the preprocessing was done only for the “Type” attribute since it was in the string format which resulted in a very bad model having a bad accuracy. This is **because of the different approach which is Probabilistic Naïve bayes** (more on this in the design and Implementation section).

Then we tried to bin all the columns to get a better result, **except** the UDI and product ID (since they did not contribute at all for any improvement) **and** the different error types since they were the target classes. After binning, all the columns were differentiated by low, medium, and high **except** Type since it wasalready differentiated hence it was not binned which resulted in an excellent accuracy but a bad recall. But this was still not a good classifier as the dataset was not balanced.

1. **Balanced Dataset:** -
2. **Under Sampling Dataset:** - We had **first under sampled** our dataset by

making all the labels have equal instances of 1000 which brings us to 13,000 instances since we have 13 labels then we **clustered sample** the dataset to reduce the dataset to 8060 samples where 6045 was used for training and 2015 for testing as mentioned for the task, the preprocessing was done only for the “Type” attribute since it was in the string format which resulted to an **excellent accuracy**, recall and precision.

**Then we tried to bin all the columns** to get a better result, **except** the UDI and product ID (since they did not contribute at all for any improvement) **and** the different error types since they were the target classes. After binning, all the columns were differentiated by low, medium, and high **except** Type since it wasalready differentiated hence it was not binned which resulted in a bad accuracy, recall and precision. The probabilistic model was unable to properly find insights since all of them were binned and hence the model had bad metrics.

1. **Over Sampling Dataset:** - We had **first over sampled** our dataset by

making all the labels have equal instances of 10000 which brings us to 13,0000 instances since we have 13 labels to which we then **clustered sample** the dataset **twice** by first reducing the 130,000 instances to 13,000 and then from 13,000 instances to 8060 samples where 6045 was used for training and 2015 for testing as mentioned for the task,, the preprocessing was done only for the “Type” attribute since it was in the string format which resulted to an **excellent accuracy**, recall and precision.

**Then we tried to bin all the columns** to get a better result, **except** the UDI and product ID (since they did not contribute at all for any improvement) **and** the different error types since they were the target classes. After binning, all the columns were differentiated by low, medium, and high **except** Type since it wasalready differentiated hence it was not binned which resulted in a bad accuracy, recall and precision. The probabilistic model was unable to properly find insights since all of them were binned and hence the model had bad metrics.

Preprocessing Conclusion

1. **Binary class**
2. For **Binary class of unbalanced dataset**:

Without Binning: Excellent accuracy, Recall and Precision

With Binning: Slightly better accuracy, Recall and Precision when compared

**Overall best**: Binary class of unbalanced dataset with Binning

1. For **Binary class of balanced dataset using Under sampling**:

Without Binning: good accuracy, Recall and Precision

With Binning: Slightly low accuracy, Recall and Precision when compared

**Overall best**: Binary class of balanced dataset using Under sampling without Binning

1. For **Binary class of balanced dataset using Over sampling**:

Without Binning: good accuracy, Recall and Precision

With Binning: Slightly low accuracy, Recall and Precision when compared

**Overall best**: Binary class of balanced dataset using Over sampling without Binning

1. **Multi-class or Multi label**
2. For **Multi class of unbalanced dataset**:

Without Binning: Very bad accuracy, Recall and Precision

With Binning: Excellent accuracy, Recall and Precision when compared

**Overall best**: Multi class of unbalanced dataset with Binning

1. For **Multi class of balanced dataset using Under sampling**:

Without Binning: Excellent accuracy, Recall and Precision

With Binning: very bad accuracy, Recall and Precision when compared

**Overall best**: Multi class of balanced dataset using Under sampling without Binning

1. For **Multi class of balanced dataset using Over sampling**:

Without Binning: Excellent accuracy, Recall and Precision

With Binning: very bad accuracy, Recall and Precision when compared

**Overall best**: Multi class of balanced dataset using Over sampling without Binning

# Design and Implementation

We created **two different types of Gaussian Naïve Bayes** algorithms because when we had used the gaussian Naïve Bayes using multi class (multi label) we ran into errors due to the low frequency (two of the labels had only one instance).

So we created a Gaussian naïve bayes that **focuses on log probabilities**, we use this because when we **calculate the variance (standard deviation**) for the previous Gaussian naïve bayes we get errors since the labels with low frequency don’t have enough data for calculating the variance and when we under sample or over sample, the variance is 0 or lower than 0 resulting to a very small value and hence cause an error in our prediction since the model believes that the label with low variance is the best label for the particular instance that is to be predicted.

Hence, we had to create a step-by-step implementation that **focuses on the log probabilities** which prevents from having an error since we are using log probabilities for each feature and not mean and variance.

1. **Gaussian naïve Bayes (works well for continuous data)**
2. First we choose the target labels for that we use y. unique and then we have to set the features and the instances, below is the code

Text

Description automatically generated with medium confidence

1. We calculate the mean, standard deviation and prior this is used to be calculated for each attribute or feature since this is how gaussian distribution works

Text

Description automatically generated

1. Then we calculate by separating the target labels and then calculating the mean, standard deviation and prior for each instance of the dataset

Graphical user interface, text

Description automatically generated

1. Then we use gaussian probability distribution by calculating the posterior which is

P (A|B) which is the probability of A being true if b is true and class probabilities.

Graphical user interface, text, application

Description automatically generated

1. **Gaussian naïve bayes (using log probabilities)**
2. First, we calculate the log probabilities with additive smoothing

Graphical user interface, text, application

Description automatically generated

1. Then we store the number of instances and features, we store the number of instance to go through each row and then calculate the log probabilities for each feature by the formula P(A|B) = P(B|A) P(A) / P(B)

Graphical user interface, text, application, email

Description automatically generated

1. Now we compute the probabilities for each feature based on the instances

Graphical user interface, text, application

Description automatically generated

# Implementation and Training and Testing performance

1. **Binary class**
2. For **Binary class of unbalanced dataset**:

Without Binning: Excellent accuracy, Recall and Precision

A screenshot of a computer

Description automatically generated with medium confidence

-------------------------------------------------------------------------------------------------------------------------------

With Binning: Slightly better accuracy, Recall and Precision when compared

A screenshot of a computer

Description automatically generated with medium confidence

**Overall best**: Binary class of unbalanced dataset with Binning

-------------------------------------------------------------------------------------------------------------------------------

1. For **Binary class of balanced dataset using Under sampling**:

Without Binning: good accuracy, Recall and Precision

A screenshot of a computer

Description automatically generated with medium confidence

With Binning: Slightly low accuracy, Recall and Precision when compared

A screenshot of a computer

Description automatically generated with medium confidence

**Overall best**: Binary class of balanced dataset using Under sampling without Binning

-------------------------------------------------------------------------------------------------------------------------------

1. For **Binary class of balanced dataset using Over sampling**:

Without Binning: good accuracy, Recall and Precision

Table

Description automatically generated

With Binning: Slightly low accuracy, Recall and Precision when compared

A screenshot of a computer

Description automatically generated with medium confidence

**Overall best**: Binary class of balanced dataset using Over sampling without Binning

1. **Multi-class or Multi label**
2. For **Multi class of unbalanced dataset**:

Without Binning: Very bad accuracy, Recall and Precision

Table

Description automatically generated

With Binning: Excellent accuracy, Recall and Precision when compared

Table

Description automatically generated

**Overall best**: Multi class of unbalanced dataset with Binning

1. For **Multi class of balanced dataset using Under sampling**:

Without Binning: Excellent accuracy, Recall and Precision

Table

Description automatically generated

With Binning: very bad accuracy, Recall and Precision when compared

Table

Description automatically generated

**Overall best**: Multi class of balanced dataset using Under sampling without Binning

1. For **Multi class of balanced dataset using Over sampling**:

Without Binning: Excellent accuracy, Recall and Precision

Table

Description automatically generated

With Binning: very bad accuracy, Recall and Precision when compared

Table

Description automatically generated

**Overall best**: Multi class of balanced dataset using Over sampling without Binning

# Conclusion

The best models for each case

1. **Binary class**
2. For **Binary class of unbalanced dataset**:

**Overall best**: Binary class of unbalanced dataset with Binning

1. For **Binary class of balanced dataset using Under sampling**:

**Overall best**: Binary class of balanced dataset using Under sampling without Binning

1. For **Binary class of balanced dataset using Over sampling**:

**Overall best**: Binary class of balanced dataset using Over sampling without Binning

1. **Multi-class or Multi label**
2. For **Multi class of unbalanced dataset**:

**Overall best**: Multi class of unbalanced dataset with Binning

1. For **Multi class of balanced dataset using Under sampling**:

**Overall best**: Multi class of balanced dataset using Under sampling without Binning

1. For **Multi class of balanced dataset using Over sampling**:

**Overall best**: Multi class of balanced dataset using Over sampling without Binning