Master's Degree Thesis

Mechanical Engineering

Rolling Bearing Damage Characterization

A Machine Learning Approach

Milad Rahmani Tootkaboni

Student Number: S257786

Supervisor: *(Politecnico di Torino)*

Professor Alessandro Fasana

Advisors: *(University of Massachusetts Dartmouth)*

Professor Mazdak Pour A Tootkaboni

Professor Arghavan Louhghalam

Academic Year: 2020/21

Contact Info:

Supervisor: *(Politecnico di Torino)*

Alessadnro Fasana *(Full Professor)*

Department of Mechanical and Aerospace Engineering (DIMEAS)

+39 0110903397 / 3397
alessandro.fasana@polito.it
Corso Duca Degli Abruzzi 24, 10129, Turin, ITALY

Advisors: *(University of Massachusetts Dartmouth)*

Mazdak Pour A Tootkaboni *(Associate Professor)*

Department of Civil and Environmental Engineering

+1 508-999-8465
mtootkaboni@umassd.edu
285 Old Westport Road – Dartmouth, MA 02747-2300, USA

Arghavan Louhghalam *(Associate Professor)*

Department of Civil and Environmental Engineering

+1 508-999-8491
arghavan.louhghalam@umassd.edu
285 Old Westport Road – Dartmouth, MA 02747-2300, USA

Acknowledgements

This is written to acknowledge all the professors helped me on working on my thesis, guided me professionally and invested their time to clarify all the vague points.

I would like to thank professor *Alessandro Fasana* as the supervisor given me the chance to cover my thesis with a fabulous topic under his supervision and also providing me a marvelous data set. I highly appreciate professors *Mazdak Pour A Tootkaboni* and *Arghavan Louhghalam* from *University of Massachusetts Dartmouth* as the advisors for accepting me to derive benefits from their outstanding knowledge.

I would like to dedicate this thesis to my beloved parents who have been always caring to me and given me the chance to be a better human being.

Table of Contents

Abstract

This thesis aims to clarify the defect classification of roller bearings. The first chapter gives an introduction on bearings. The main test done at *Politecnico di Torino* is briefly explained on the second chapter. The basic matrices formation which are used for further final models matrices is presented in chapter three. On chapter four different models and their different factors are completely defined. Data preparation for machine learning process such as train and test data sets are fully described in chapter five. Machine learning process and all the pertaining results are presented in chapter six. The concept of feature selection is fully described in chapter seven. In chapter eight the conclusion is presented. At the end the extra information is presented in four separated appendices.

Chapter One

Introduction

1.1. General Definition

Bearings are components used to facilitate the rotational movement of machine parts so as to reduce friction between rotary and fixed parts as well as preventing parts from getting defected and damaged.

The classification of rolling bearings will be based on different issues for instance, working conditions, operations, etc. but the most significant one depends on the type of the rolling elements. From this perspective a bearing could be divided into two different types, ***Ball Bearing*** for which the rolling element is spherical, and ***Roller Bearing*** for which the rolling element is cylindrical. However, each type contains many different subsets, the main aim is to know the general difference between the two classes.

1.2. History of Bearings[1]

40 BC: Wooden type of ball bearing to support a table, Roman Nemi ships, Italy.

15th century: Maiden use of ball bearing in aerospace illustrated in the drawings of a helicopter by Leonardo da Vinci.

17th century: First description of caged bearings by Galileo.

Mid 1740: Invention of first caged roller bearing by John Harrison.

1794: The first modern type of ball bearing by a British inventor, Philip Vaughan.

1869: The first type of radial ball bearing by a French bicycle mechanic, Jules Suriray. It was used in the winner bicycle of the world's first race in Paris-Rouen.

1883: The creation of an independent bearing industry by Friedrich Fischer, the founder of FAG.

1898: The invention of tapered roller bearing by Henry Timken.

1907: The modern self-aligning ball bearing by Sven Wingquist of SKF.

1934: The invention of wire race bearing by Erich Franke.

1972: Invention of V-grooved bearing guide wheels, a linear motion bearing, by Bud Wisecarver.

Early 1980's: The invention of first bi-material plain bearing by Robert Schroeder.

1.3. Applications and Concerns

Today, both ball and roller bearings are widely used in different fields of industry. From home appliances, dental industry, to more complicated fields such as automobile industries, aerospace and etc.

Bearings like all other types of mechanical components have a lifetime range, which depends on different factors. For instance, working conditions, existence of abrasive particles, erosive and corrosive environments, humidity, maintenance etc.

For different situations there will be different concerns about how to optimize this lifetime by considering the conditions in which the bearings are working.

To ease this concern, firstly the conditions should be defined and studied, then by considering the data, a proper solution must be defined in order to identify the obstacle. Having found the problem, it is easy to deal with and eliminate that to optimize the bearing lifetime. This will lead to cost and time savings which are the most important factors in different industries.

[1] From https://www.kginternational.com/

1.4. Dataset

This research is done based on the article published on ***Mechanical Systems and Signal Processing*** journal[2].

Article name: ***The Politecnico di Torino rolling bearing test rig: Description and analysis of open access data***.

Article authors: ***Alessandro Paolo Daga, Alessandro Fasana, Stefano Marchesiello, Luigi Garibaldi***[3].

The research has been taken place and the test has been conducted in the ***Dynamic and Identification Research Group (DIRG)*** of the ***Department of Mechanical and Aerospace Engineering*** at ***Politecnico di Torino***.

The main article by ***PoliTo*** works on two main different tests, ***Variable Speed and Load*** test and ***Endurance*** test.

However, on this thesis only the data of ***Variable Speed and Load*** test is used.

The complementary information is available on:

ftp://ftp.polito.it/people/DIRG_Bearing_Data/

1.5. Scope

This research aims to identify the characterization of roller bearings by means of the data introduced before.

To clarify, if a new data set is provided this research helps us to identify either the type of working condition or the class of defect by means of ***Machine Learning Classification*** methods.

To give some extra explanation, for instance, if a company as a client provides a set of data and asks to find the working condition or the defect class of the bearing, it could be possible to create a model to make an acceptable guess.

To do so, firstly, it is needed to explain some parts of the main test and how data are interpreted. Telling about the different matrices which are extracted in the lab. Also giving some information about the layout of the test hardware.

Secondly, providing information about the classification of data is useful to give a better insight of the test.

Thirdly, defining new pieces of data, for having a better understanding about the model is needed.

To achieve such goal, some methods of ***Machine Learning*** is applied so the next step is introducing the methods which are carried out to make the models.

[2] https://www.journals.elsevier.com/mechanical-systems-and-signal-processing

[3] E-mail addresses: alessandro.daga@polito.it (A.P. Daga), alessandro.fasana@polito.it (A. Fasana), stefano.marchesiello@polito.it (S. Marchesiello), luigi.garibaldi@polito.it (L. Garibaldi).

Then the configuration of train and test matrices are defined.

Finally, when the creation of model is finished the comparison of different situations is shown.

The application which is used to figure out the model is ***MATLAB***[4] by ***MathWorks®***.

[4] https://www.mathworks.com/

Chapter Two

Main Test Explanation

In this chapter the main test done in Politecnico di Torino is briefly explained to vividly clarify the machine learning procedure. All the data provided in this chapter are from (the complete information could be found on):

ftp://ftp.polito.it/people/DIRG_BearingData/

2.1. Test Hardware Setup

In this section two different pieces of hardware, i.e. test rig and sensors as well as the acquisition system are described.

2.1.1. Test Rig

The test rig consists of different parts as follows: *(Figure 2.1)*

Spindle: which guarantees the rotation of the shaft. The spindle

is fixed to a rigid support which rests on a massive steel base plate.

Shaft: on which three bearings are mounted. The applicable force is exerted to the shaft. The shaft is hollow and bearings on the shaft are lubricated from inside.

Bearings: the outer rings of two identical bearings **B1** and **B3** are fixed on two supports and the inner rings are attached to the shaft. The inner ring of bearing **B2** is attached to the shaft also. However, the outer ring is in connection with the force applying system. *(Table 2.1)*

Sledge and Static Load Cell: the force applying system that are connected to the outer ring of **B2** and have the role of load exertion.

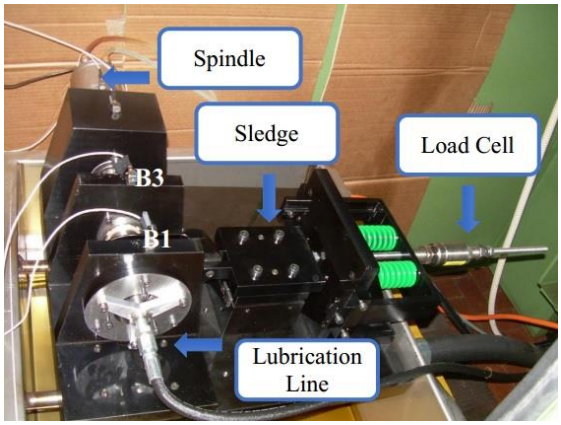


Figure 2.1 - a) Overall view of the test rig

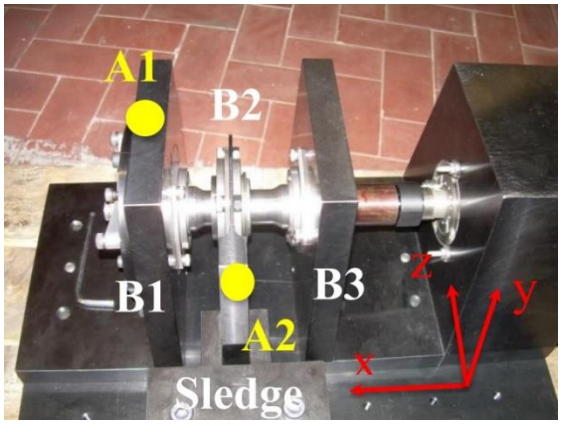


Figure 2.1 - b) Accelerometers positions and the reference system

2.1.2. Sensors

In this test two identical accelerometers are used and mounted on two positions **A1** and **A2** as it is illustrated in *(Figure 2.1 – b)* in the way that **A1** is located on the support of the damaged bearing **B1** which undergoes the test and **A2** is on the support of the larger bearing **B2** which is the position of external load application.

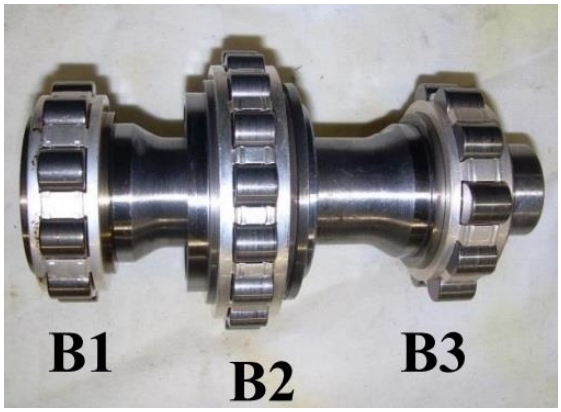


Figure 2.1 - c) Shaft and its three roller bearings

	Pitch Diameter D (mm)	Roller Diameter d (mm)	Contact Angle Φ (○)	Rolling Elements Z
B1 & B3	40.5	9.0	0	10
B2	54.0	8.0	0	16

Table 2.1 - Properties of roller bearings

Accelerometers are the triaxial IEPE type:

Frequency range: 1-12000 Hz (amplitude ±5%, phase ±10○)

Nominal resonant frequency: 55 kHz

Nominal sensitivity: 1 mV/ms^{-2}

The radial force on the second bearing is measured by means of the static load cell with the sensitivity of 0.499 mV/N. *(Figure 2.2)*

Figure 2.2 - a) The triaxial IEPE accelerometer

2.1.3. Acquisition

The acquisition is achieved by means of OR38 signal analyzer, an OROS production, which has the accuracy on the input channel: phase ±0.02○, amplitude ±0.02dB, frequency ±0.005%.

The analogue to digital transformation is done by means of a 24 bits delta-sigma convertor. The range of each channel is set between minimum (±17 mV) and maximum (±40 mV) to avoid saturation of channels.

In this test six channels are defined based on two accelerometers located on **A1** and **A2**. Each accelerometer measures data in three dimensions *x, y* and *z*. *(Table 2.2)*

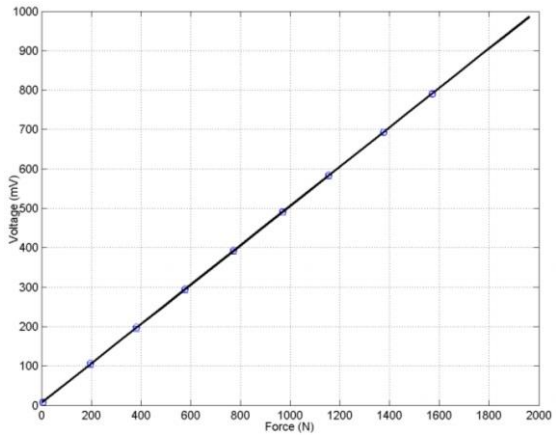


Figure 2.2 - b) Calibration curve of the static load cell

	Channel 1	**Channel 2**	**Channel 3**	**Channel 4**	**Channel 5**	**Channel 6**
Direction	Axial, x	Radial, y	Radial, z	Axial, x	Radial, y	Radial, z
Accelerometer No.	A1	A1	A1	A2	A2	A2
Channel label	x_1	y_1	z_1	x_2	y_2	z_2

Table 2.2 - Direction of the measured accelerations

2.2. Description of the Variable Speed and Load Test

For this test there are seven different classes based on defects. The class **0A** is for undamaged case in which there is no defect (i.e. healthy condition). The defects are produced by a Rockwell tool in three defect sizes of 450, 250 and 150 µm on the inner ring respectively for **1A**, **2A** and **3A** and three defect sizes of 450, 250 and 150 µm on the roller respectively for **4A**, **5A** and **6A**. *(Table 2.3)*

In each class there are different working conditions based on two factors, speed and load. There are four different loads **0**, **1000**, **1400** and **1800 N** and five different rotational speeds **100**, **200**, **300**, **400** and **500 Hz**. *(Table 2.4)*

Due to power limitations of the spindle controller, it is not possible to have 1800 N for 400 Hz and also 1400 and 1800 N for 500 Hz.

Having considered all the aspects, there are 17 different working condition for each class of defect.

For this test the sampling frequency is f_s = 51200 Hz and the record duration is T = 10 s. The number of acquisition points will be $f_s \times T = 512000$. By considering six channels the final matrix for each working condition is 512000×6.

Class Name	Defect	Defect Size (µm)
0A	NO defect (Healthy)	---
1A	Defect on the inner ring	450
2A	Defect on the inner ring	250
3A	Defect on the inner ring	150
4A	Defect on the roller	450
5A	Defect on the roller	250
6A	Defect on the roller	150

Table 2.3 - Different classes definition

		Nominal Speed (Hz)				
		100	**200**	**300**	**400**	**500**
Nominal Load (N)	**0**	1	5	9	13	16
	1000	2	6	10	14	17
	1400	3	7	11	15	---
	1800	4	8	12	---	---

Table 2.4 – Different working conditions definition

Chapter Three

Basic Matrices Definition

As it was shown in the previous chapter, the dataset for each working condition is presented in the form of a 512000×6 matrix *(Figure 3.1)*.

The rows represent acquisition points and the columns are the channels and the whole test is done in 10s.

It is worth mentioning that for seven classes of defect from **0A** to **6A** and different working conditions from **1** to **17** (both classes and working conditions were explained in the previous chapter *Table – 2.3* and *Table – 2.4*), there are 119 different matrices of 512000×6.

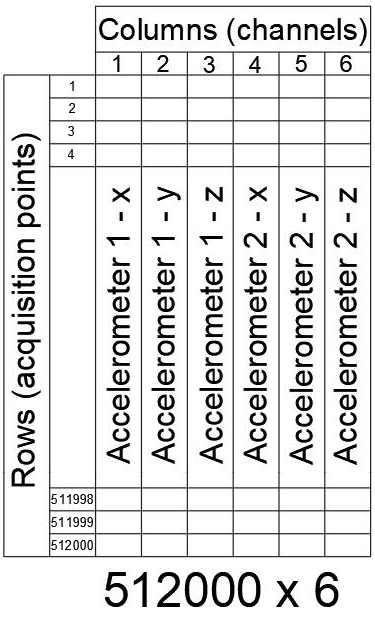


Figure 3.1 - Main data set matrix

3.1. Main Matrix Summarization

Now to have a better performance, each matrix is summarized to chunks. In this paper two different types of chunks are considered for data summarization.

These two types of chunks are considered to have a better comparison between 100 and 200 data point matrices when they are used to form the train and test matrices in machine learning procedure *(Figure 3.2)*.

3.1.1. 100 Chunks – 0.1s

If a record duration of T=10s is divided into 100 chunks, each chunk is equal to 0.1s. From previous chapter it is known that the sampling frequency is equal to f_s=51200 Hz. It means in each second 51200 acquisition points are recorded.

Now for each chunk of 0.1s the number of acquisition points are 5120. Each set of 5120×6 acquisition points is converted to 1×6 set of numbers called data points. It means that each data point should be obtained from the data pertaining to each chunk's 5120 acquisition points.

This results in a final matrix of 100×6. Each matrix represents a working condition data summarized from 512000 acquisition points to 100 final data points.

3.1.2. 200 Chunks – 0.05s

If a record duration of T=10s is divided into 200 chunks, each chunk is equal to 0.05s. From previous chapter it is known that the sampling frequency is equal to f_s=51200 Hz. It means in each second 51200 acquisition points are recorded.

Now for each chunk of 0.05s the number of acquisition points are 2560. Each set of 2560×6 acquisition points is converted to 1×6 set of numbers called data points. It means that each data point should be obtained from the data pertaining to each chunk's 2560 acquisition points.

This results in a final matrix of 200×6. Each matrix represents a working condition data summarized from 512000 acquisition points to 200 final data points.

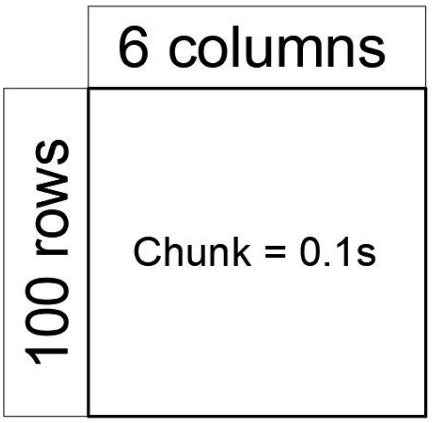


Figure 3.2 - a) Matrix 100x6 - chunk 0.1s

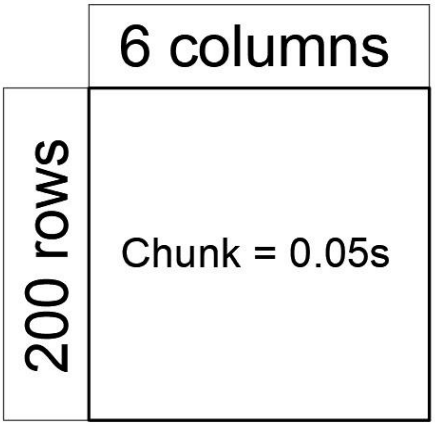


Figure 3.2 - b) Matrix 200x6 - chunk 0.05s

Chapter Four

Models Definition

In this chapter the different models are defined based on different features. Features are added gradually to see what difference they make on the accuracy of the machine learning methods.

As it was introduced in the previous chapter, all the models are made based on the summarized matrices, both for 100 and 200 data points.

Models are created based on different criteria. Apart from different chunks and different features, the working conditions and classes of defects are also important in forming the models.

To emphasize, it must be said that the models introduced and made in this chapter are the reference for upcoming machine learning analysis and they are not used directly in analysis procedure.

In the upcoming sections the different criteria in models creation are introduced and explained in details.

4.1. Chunks

In this section the first criterion of model creation, which is *chunk*, is introduced in two divisions:

- 100 chunks – 0.1s
- 200 chunks – 0.05s

4.2. Features

In this section the second criterion of model creation, which is *feature*, is introduced in four divisions as follows:

4.2.1. Mean

The first feature is *Mean* of the data. To form the first type of matrix for each working condition, the *Mean* value of each chunk's acquisition points should be calculated. Each set of acquisition points form a 1×6 data point. The *Mean* value data point forms the columns of one to six. At last there would be two types of matrices a 100×6 for 100 chunks of 0.1s and a 200×6 for 200 chunks of 0.05s. *(Figure 4.1)*

4.2.2. Mean and RMS

The second feature is *RMS* of the data. To form the second type of matrix for each working condition, the *Mean* and *RMS* values of each chunk's acquisition points should be calculated. Each set of acquisition points form a 1×12 data point. The *Mean* value data point forms the columns of one to six and the *RMS* value data point forms the columns of seven to twelve. At last there would be two types of matrices a 100×12 for 100 chunks of 0.1s and a 200×12 for 200 chunks of 0.05s. *(Figure 4.2)*

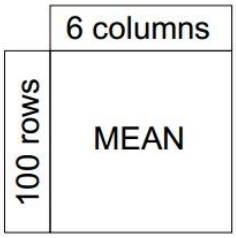


Figure 4.1 - a) Mean matrix - 100 chunks

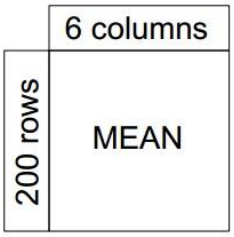


Figure 4.1 - b) Mean matrix - 200 chunks

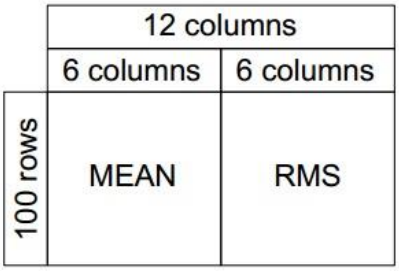


Figure 4.2 - a) Mean and RMS matrix - 100 chunks

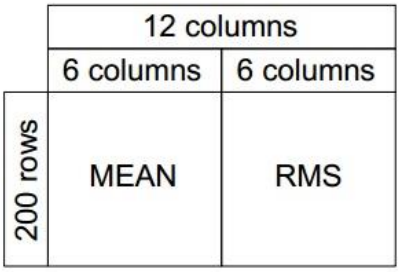


Figure 4.2 - b) Mean and RMS matrix - 200 chunks

4.2.3. Mean, RMS and Kurtosis

The third feature is *Kurtosis* of the data. To form the third type of matrix for each working condition, the *Mean, RMS* and *Kurtosis* values of each chunk's acquisition points should be calculated. Each set of acquisition points form a 1×18 data point. The *Mean* value data point forms the columns of one to six, the *RMS* value data point forms the columns of seven to twelve and the *Kurtosis* value data point forms the columns of thirteen to eighteen. At last there would be two types of matrices a 100×18 for 100 chunks of 0.1s and a 200×18 for 200 chunks of 0.05s. *(Figure 4.3)*

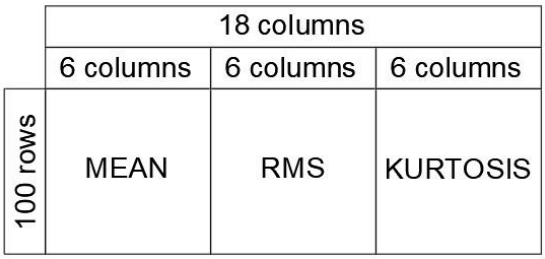


Figure 4.3 - a) Mean, RMS and Kurtosis matrix - 100 chunks

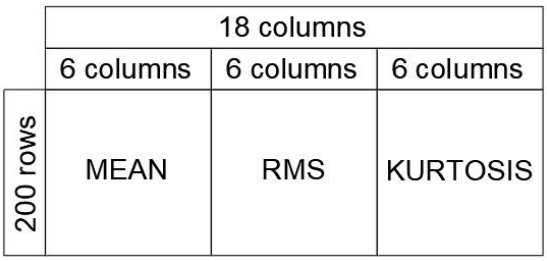


Figure 3.3 - b) Mean, RMS and Kurtosis matrix - 200 chunks

4.2.4. Mean, RMS, Kurtosis and Skewness

The fourth feature is *Skewness* of the data. To form the fourth type of matrix for each working condition, the *Mean, RMS, Kurtosis* and *Skewness* values of each chunk's acquisition points should be calculated. Each set of acquisition points form a 1×24 data point. The *Mean* value data point forms the columns of one to six, the *RMS* value data point forms the columns of seven to twelve, the *Kurtosis* value data point forms the columns of thirteen to eighteen and the *Skewness* value data point forms the columns of nineteen to twenty-four. At last there would be two types of matrices a 100×24 for 100 chunks of 0.1s and a 200×24 for 200 chunks of 0.05s. *(Figure 4.4)*

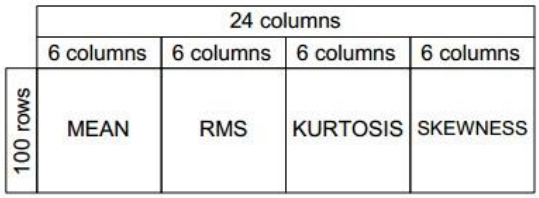


Figure 4.4 - a) Mean, RMS, Kurtosis and Skewness matrix - 100 chunks

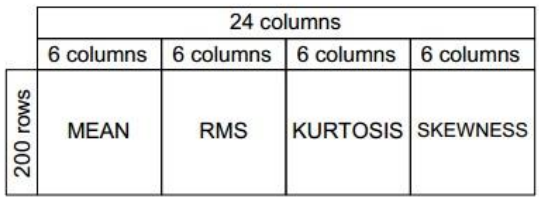


Figure 4.4 - b) Mean, RMS, Kurtosis and Skewness matrix - 200 chunks

4.3. Approaches

In this section the third criterion of model creation, which is *approach*, is introduced in four divisions as follows:

4.3.1. Same Working Condition

The first approach is *Same Working Condition*. It means that both speed and load are known but the class of defect is unknown. In this part the matrix is formed out of seven different matrices from classes **0A** to **6A**. The final matrix for 100 data points has 700 rows in which the first 100 rows of matrix are dedicated to class matrix **0A** and the last 100 rows are for class matrix **6A.** Also the final matrix for 200 data points has 1400 rows in which the first 200 rows of matrix are dedicated to class matrix **0A** and the last 200 rows are for class matrix **6A.** *(Figure 4.5)*

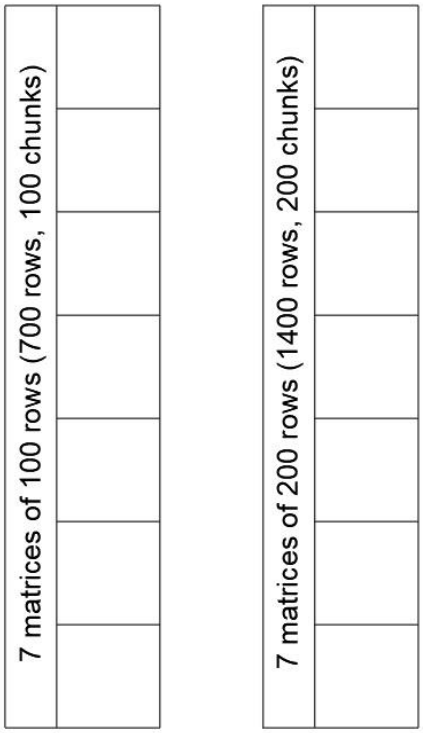


Figure 4.4 - Same working conditions final matrices

4.3.2. Same Class of Defect

The second approach is *Same Class of defect*. It means that the class of defect is known but both speed and load are unknown. In this part the matrix is formed out of seventeen different matrices from working conditions **1** to **17**. The final matrix for 100 data points has 1700 rows in which the first 100 rows of matrix are dedicated to working condition matrix **1** and the last 100 rows are for working condition matrix **17.** Also the final matrix for 200 data points has 3400 rows in which the first 200 rows of matrix are dedicated to working condition matrix **1** and the last 200 rows are for working condition matrix **17.** *(Figure 4.6)*

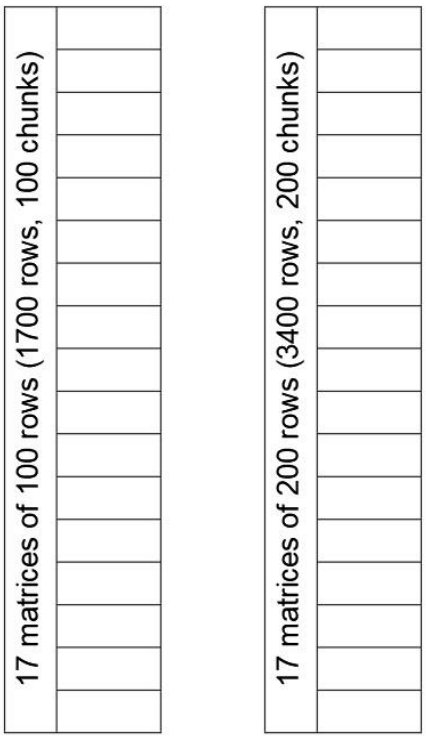


Figure 4.5 - Same class of defect final matrices

4.3.3. Same Load

The third approach is *Same Load*. It means that just the applied load is known but both speed and class of defect are unknown. In this part the matrix formation depends on the exerted load. For each class of defect there are four different loads of **0**, **1000**, **1400** and **1800 N**. For **0** and **1000 N** there are five different conditions, for **1400 N** there are four different conditions and for **1800 N** there are three different conditions. The final matrix for 100 data points has:

- 3500 rows (for **0** and **1000 N**) in which the first 500 rows of matrix are dedicated to class matrices **0A** and the last 500 rows are for class matrices **6A**.
- 2800 rows (for **1400 N**) in which the first 400 rows of matrix are dedicated to class matrices **0A** and the last 400 rows are for class matrices **6A**.
- 2100 rows (for **1800 N**) in which the first 300 rows of matrix are dedicated to class matrices **0A** and the last 300 rows are for class matrices **6A**.

Also the final matrix for 200 data points has:

- 7000 rows (for **0** and **1000 N**) in which the first 1000 rows of matrix are dedicated to class matrices **0A** and the last 1000 rows are for class matrices **6A**.
- 5600 rows (for **1400 N**) in which the first 800 rows of matrix are dedicated to class matrices **0A** and the last 800 rows are for class matrices **6A**.
- 4200 rows (for **1800 N**) in which the first 600 rows of matrix are dedicated to class matrices **0A** and the last 600 rows are for class matrices **6A**. *(Figure 4.7)*

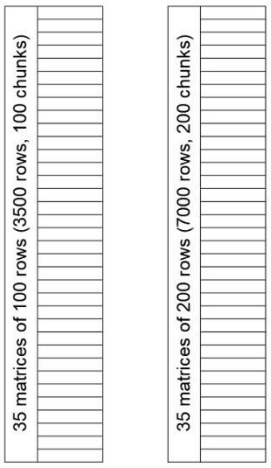


Figure 4.6 – a) Same load final matrices (0 and 1000 N)

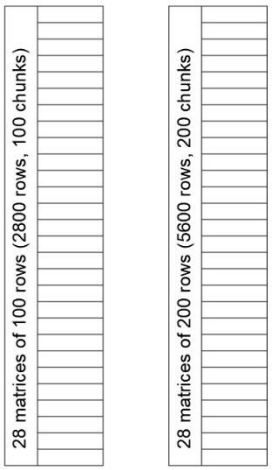


Figure 4.7 - b) Same load final matrices (1400 N)

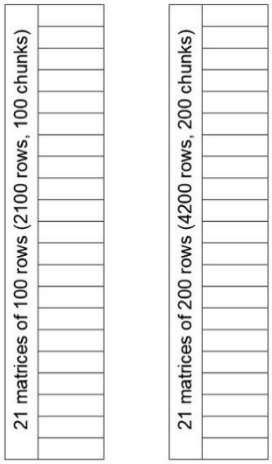


Figure 4.7 - c) Same load final matrices (1800 N)

4.3.4. Same Speed

The fourth approach is *Same Speed*. It means that just the applied speed is known but both load and class of defect are unknown. In this part the matrix formation depends on the exerted speed. For each class of defect there are five different speeds of **100**, **200**, **300**, **400** and **500 Hz**. For **100**, **200** and **300 Hz** there are four different conditions, for **400 Hz** there are three different conditions and for **500 Hz** there are two different conditions. The final matrix for 100 data points has:

- 2800 rows (for **100**, **200** and **300 Hz**) in which the first 400 rows of matrix are dedicated to class matrices **0A** and the last 400 rows are for class matrices **6A**.
- 2100 rows (for **400 Hz**) in which the first 300 rows of matrix are dedicated to class matrices **0A** and the last 300 rows are for class matrices **6A**.
- 1400 rows (for **500 Hz**) in which the first 200 rows of matrix are dedicated to class matrices **0A** and the last 200 rows are for class matrices **6A**.

Also the final matrix for 200 data points has:

- 5600 rows (for **100**, **200** and **300 Hz**) in which the first 800 rows of matrix are dedicated to class matrices **0A** and the last 800 rows are for class matrices **6A**.
- 4200 rows (for **400 Hz**) in which the first 600 rows of matrix are dedicated to class matrices **0A** and the last 600 rows are for class matrices **6A**.
- 2800 rows (for **500 Hz**) in which the first 400 rows of matrix are dedicated to class matrices **0A** and the last 400 rows are for class matrices **6A**. *(Figure 4.8)*

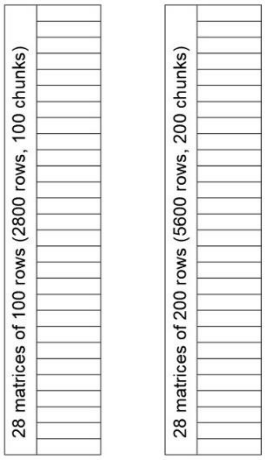


Figure 4.8 - a) Same speed final matrices (100,200 and 300 Hz)

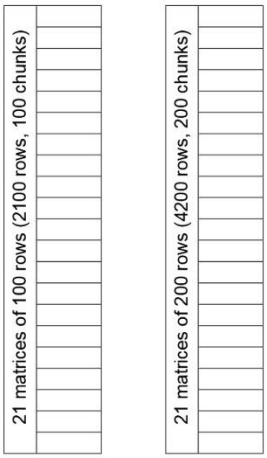


Figure 4.8 - b) Same speed final matrices (400 Hz)

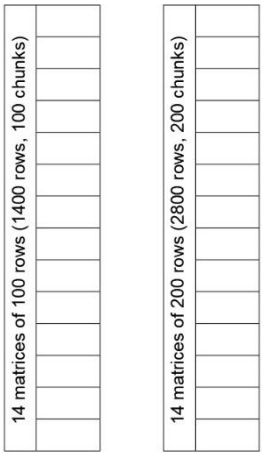


Figure 4.7 - c) Same speed final matrices (500 Hz)

4.4. Models

Having explained the model factors, now this is time to introduce the models. The general format of models is as follows:

X_*y*_nnn

X: indicates the model name based on different features. For four models X is defined as follows:

M: this model has only one feature and that is *Mean*. Obviously as it was represented before, this model is six-dimensional (the final matrix has six columns).

MR: this model has two features *Mean* and *RMS*. Obviously as it was represented before, this model is twelve-dimensional (the final matrix has twelve columns).

MRK: this model has three features *Mean*, *RMS* and *Kurtosis*. Obviously as it was represented before, this model is eighteen-dimensional (the final matrix has eighteen columns).

MRKS: this model has four features *Mean*, *RMS*, *Kurtosis* and *Skewness*. Obviously as it was represented before, this model is twenty-four-dimensional (the final matrix has twenty-four columns).

y: this factor indicates the approaches discussed in previous sections. *y* is defined as follows:

W7: same working condition

C17: same class of defect

L35: same load (0 and 1000 N)

L28: same load (1400 N)

L21: same load (1800 N)

S28: same speed (100, 200 and 300 Hz)

S21: same speed (400 Hz)

S14: same speed (500 Hz)

nnn: this factor is a three-digit number indicating the number of data points:

100: for 100 chunks of 0.1s

200: for 200 chunks of 0.05s

For instance, **MRK_W7_100** introduces the model with three features of *Mean*, *RMS* and *Kurtosis*, for the *Same Working Condition* approach which is defined by 100 data points. The final matrix is 700×18.

Another example could be **MRKS_L28_200** introduces the model with four features of *Mean*, *RMS*, *Kurtosis* and *Skewness*, for the *Same Load* approach (1400 N) which is defined by 200 data points. The final matrix is 5600×24.

Full 32 different models are listed in ***Appendix A***.

4.5. Models Selection

The approaches to be discussed are:

Same working condition:

7 (200 Hz – 1400 N)

Same class of defect:

2A (defect on inner ring - 250µm)

Same load:

1800 N

Same speed:

400 Hz

4.6. Dimensionality Reduction

Based on the models introduced in the previous section, the dimensions of final matrices vary from six to twenty-four. This means that it is not possible to illustrate the data in a two or three dimensional space.

On the other hand, it is needed to know how different clusters of data points are located with respect to the others.

The reason of dimensionality reduction is to find that how different working conditions in a model's final matrix are independently preserved. More the clusters are separated, better results of classification performance by means of machine learning. As it will be seen in the upcoming parts, the clusters generally get more separated while new features are added to the models.

This better separation leads in a higher prediction performance and lower misclassification error while classification procedure is done.

Both *prediction performance* and *misclassification error* is completely explained in chapter six.

To ease this concern, some dimensionality reduction methods are used. These methods keep maximum possible data with lowest amount of data loss to make it achievable to have the data in two or three dimensions. Afterwards, it is obvious the data could be plotted to have a better understanding of clusters' positions.

One of the best dimensionality reduction methods which is simple to understand is *principal component analysis (PCA)*. In upcoming section this method is briefly explained.

4.6.1. Principal Component Analysis (PCA)[1]

As it was mentioned before *Principal Component Analysis* is a method for dimensionality reduction and is used when the data set is high-dimensional.

The principal components of a data set in a q-dimensional space are a set of q direction vectors where the i^{th} vector is the direction of the best fitting line that keeps the highest possible data as well as being orthogonal to the previous $i - 1$ vectors. The best fitting line is so that minimizes the *average squared*

[1] https://en.wikipedia.org/

distance of data to the line. Principal component analysis computes the principal components of a data set and executes a basis change. Most of the time the maximum data possible is kept within first few components and the rest could be ignored without losing a lot of vital information. Another definition for the direction line is so that maximizes the variance of data projection. From a mathematical point of view principal components are *eigenvectors* of the data's *covariance matrix* or simply they can be achieved by *Singular Value Decomposition (SVD)* of the data matrix.

4.6.2. PCA Description and Comments

As it was mentioned the reason of dimensionality reduction is to find that how different clusters of data points as representatives of different working conditions in a model's final matrix are independently preserved. Better the clusters are separated, the clusters are separated, higher accuracy of classification performance.

In the upcoming parts some PCA figures are depicted and discussed.

To avoid over explanation only models of 200 chunks and only two models with least and most number of features (i.e. **M** and **MRKS**) are discussed in this chapter.

- **Approach W (Same Working Condition):**

For the first comparison the *Same Working Condition* approach is discussed. As it is obvious for model **M** the ranges of PCs vary slightly and two couples of clusters are almost located in the same regions. This matter could affect the classification procedure by considering the wrong data points of each cluster clearly results in misclassification error. However, in **MRKS** model clusters are almost completely separated from each other in ranges of PCs over three times larger in comparison with the previous model **M**.

- **Approach S (Same Speed)**

As the second comparison the *Same Speed* approach is selected. As it is obvious for model **M** again the ranges of PCs vary slightly and twenty-one clusters are located in a region with length of sixteen and width of almost four. While on the other hand, in **MRKS** model clusters are almost completely separated from each other in ranges of PCs pretty much larger (almost 200 and 60 for length and width respectively) in comparison with the previous model **M**.

Having a larger area for cluster would undoubtedly help to have a better performance on classification.

In the upcoming pages, the PCA results as figures are illustrated. The differences between two different models (**M** and **MRKS**) for two approaches (*Same Working Condition* and *Same Speed*) show the improvement in clusters' separation which leads to the least misclassification error.

M_W7_200:

Figure 4.8 - a) Same working condition - Mean

MRKS_W7_200:

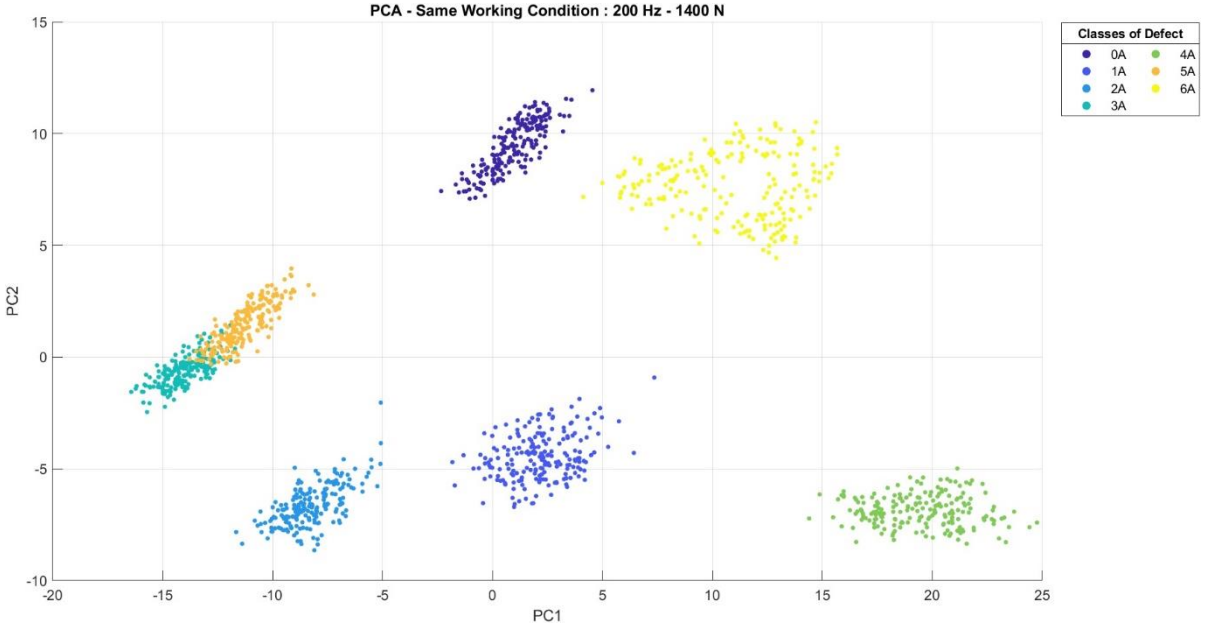


Figure 4.8 - b) Same working condition – Mean, RMS, Kurtosis, Skewness

M_S21_200:

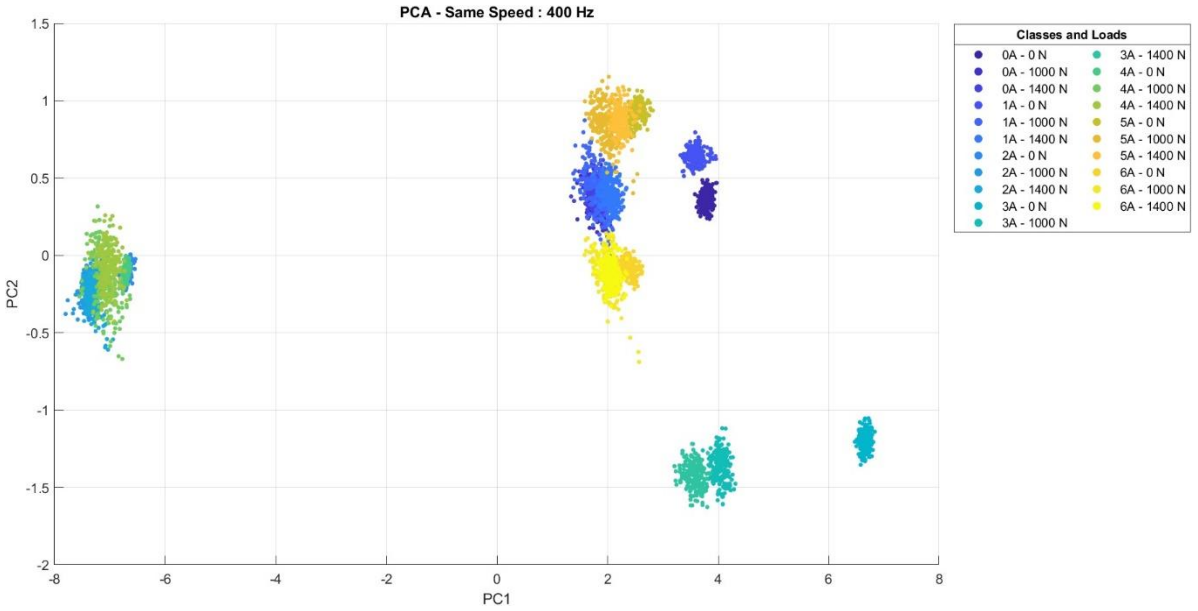


Figure 4.9 - a) Same speed - Mean

MRKS_S21_200:

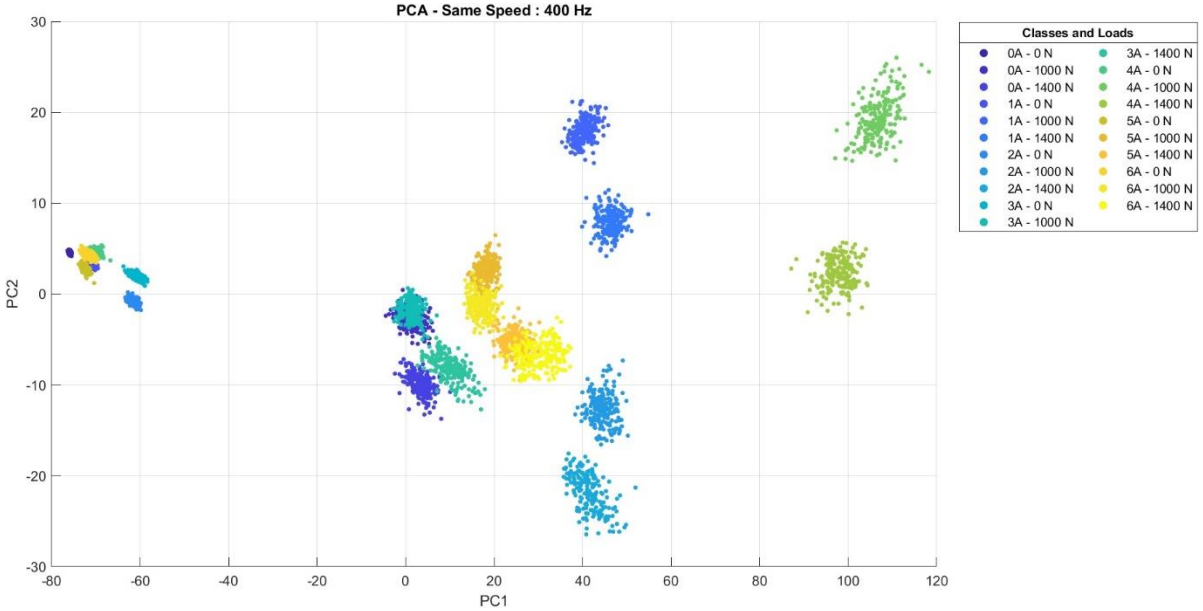


Figure 4.9 - b) Same speed – Mean, RMS, Kurtosis, Skewness

For all models with different criteria such as chunk and approaches, this improvement can be seen by adding different features.

It is conspicuous that the higher number of features (i.e. higher dimensions) would lead to a better machine learning classification procedure.

Here only two approaches are discussed as examples. Full different models' PCA illustrations for different approaches, chunks and features are depicted in ***Appendix B***.

Chapter Five

Data Preparation for Machine Learning

In this chapter the models are prepared for *Machine Learning* procedure. The train and test data creation is completely figured out as well as clarifying the models to be used and also presenting the algorithms to be applied on the data.

Before data preparation is completely interpreted, it is worth explaining machine learning briefly.

5.1. Machine Learning[1]

It is a subset of *Artificial Intelligence (AI)* focusing on model creation by means of learning from a data set and developing its accuracy without being programed for.

The aim is training the algorithms with the primary data set to find patterns based on which the new set of data is used to make predictions.

[1] https://www.ibm.com/cloud/learn/machine-learning

5.1.1. Machine Learning Methods

Based on methods, machine learning is divided into two main categories, *Supervised Learning* and *Unsupervised Learning*.

Supervised Learning is trained based on a label as the supervision indicator. It means the data is classified based on labeled data. For instance, *Regression* and *Classification* in general are two main types of supervised learning.

Unsupervised Learning is trained based on unlabeled data. This method uses data to find meaningful features to create classes and also labels itself. As an example, *Clustering* is a general category for unsupervised learning.

5.1.2. Machine Learning Steps

First step is training data set preparation. The model will learn the pattern to predict the new data based on the training data and its validation.

Second step is to choose a proper algorithm to train the model.

Third step is training and creating the model based on the chosen algorithm and the training data.

Fourth and the last step is to create test data set in order to examine and improve the model.

5.1.3. Machine Learning Algorithms

For different methods there are several different algorithms. The most important ones are as follows:

Regression:

Linear Regression
Lasso Regression
Logistic Regression
Multivariate Regression
Multiple Regression

Classification:

Naïve-Bayes Classifier
K-Nearest Neighbors Classifier
Decision Tree Classifier
Support Vector Machines (SVM)
Discriminant analysis Classifier

Clustering:

K-Means Clustering
Mean-Shift Clustering
Density-Based Spatial Clustering
Expectation-Maximization
Hierarchical Clustering
Neural Networks

In this thesis, while the aim is to classify the data and make a model for predicting the future data, the *Classification* is used which is a subset of *Supervised Learning* method.

In chapter six the classification procedure is fully explained.

5.2. Train and Test Matrices Initial Creation

Each data set which is used in machine learning process is better to be divided into two main categories one bigger part with more data points for *Train* and *Validation* of the model and a smaller set of data for *Test*.

Usually 70-80 percent of the data is used for train and validation, and the rest is dedicated to test data.

In this paper the portion is divided to 75 percent for train and validation and the rest 25 percent for test. It means for each subset of four data points, three are dedicated to train matrix and the fourth one is allocated to test matrix. (*Figure 5.1)*

5.2.1. Train and Test Matrices Creation for chunks of 0.1s

As it was mentioned before for each working condition with chunks of 0.1s there are 100 data points. To create train and test matrices there are 25 subsets of four data points. It means that the train matrix contains 75 rows and the test matrix has 25 rows. Each train matrix is validated by means of 10-fold cross validation. Having created the model, the test data is used as the model's new input to predict the different classes of data.

5.2.2. Train and Test Matrices Creation for chunks of 0.05s

For each working condition with chunks of 0.05s there are 200 data points. To create train and test matrices there are 50 subsets of four data points. It means that the train matrix contains 150 rows and the test matrix has 50 rows. Each train matrix is validated by means of 10-fold cross validation. Having created the model, the test data is used as the model's new input to predict the different classes of data.

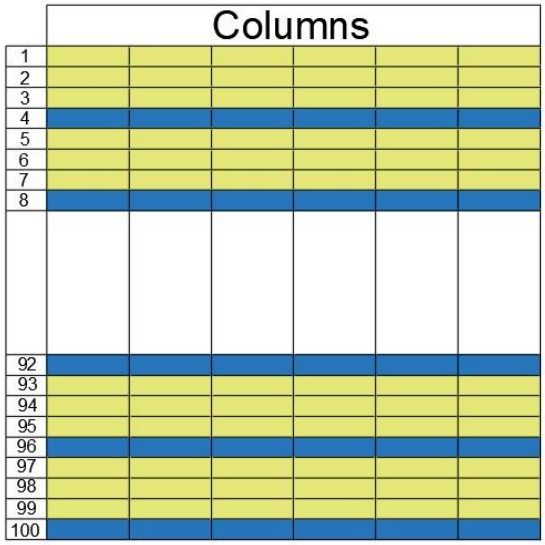


Figure 5.1 - a) Train and Test Matrices for 100 chunks

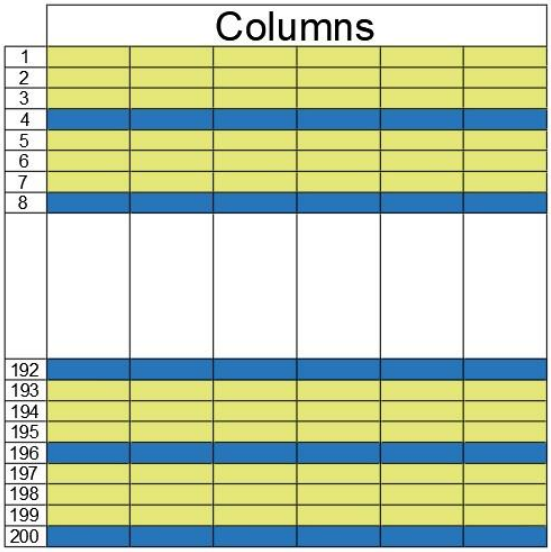


Figure 5.2 - b) Train and Test Matrices for 200 chunks

5.3. Validation[2]

In machine learning, validation technique is used to find how perfectly a model reacts to new data set. In two ways it could be helpful:

- To find out which algorithm or parameters are needed to use.
- To avoid overfitting.

There are two main methods for validation. *Holdout validation* and *K-fold Cross-Validation*.

5.3.1. Holdout Validation

This method is used when the data set is divided into two groups of train and validation. Almost 70-80 percent for train and the rest for validation.

5.3.2. K-Fold Cross-Validation

This method is used when the data set is divided into k groups. One group for validation and k-1 for train. This is repeated for k times and the average is the outcome.

In this thesis cross-validation is preferred, because it provides the chance to train data on multiple train and validation parts. The number of folds is considered 10.

[2] https://explore.mathworks.com/
https://medium.com/

5.4. Train and Test Matrices for Different Models

In this section the train and test matrices formations are discussed for different models with different chunks and approaches.

As it was mentioned before, four approaches to be discussed are:

W7 (200 Hz – 1400 N)
C17 (2A)
L21 (1800N)
S21 (400 Hz)

And also it was mentioned that the dimensions for:

M is six
MR is twelve
MRK is eighteen
MRKS is twenty-four

5.4.1. Label Vector

To prepare the data for machine learning process, each set of data is needed to be introduced by a set of labels.

To create label vector for different models the number of different working condition in each final matrix is important. Each working condition is labeled by a positive integer number starting from one. For instance, approach **W7** is labelled **one** to **seven**, approach **C17** is labelled **one** to **seventeen**, and approach **L21** & **S21** are labelled **one** to **twenty-one.**

5.4.2. Train matrices

As mentioned before train matrix for 100 data points has 75 rows and for 200 data points has 150 rows. (75 percent of data for train and validation)

For different approaches the matrices are like:

W7: this approach is for same working condition and the final matrix has 525 rows for 100 chunks and 1050 rows for 200 chunks. (*Figure 5.2)*

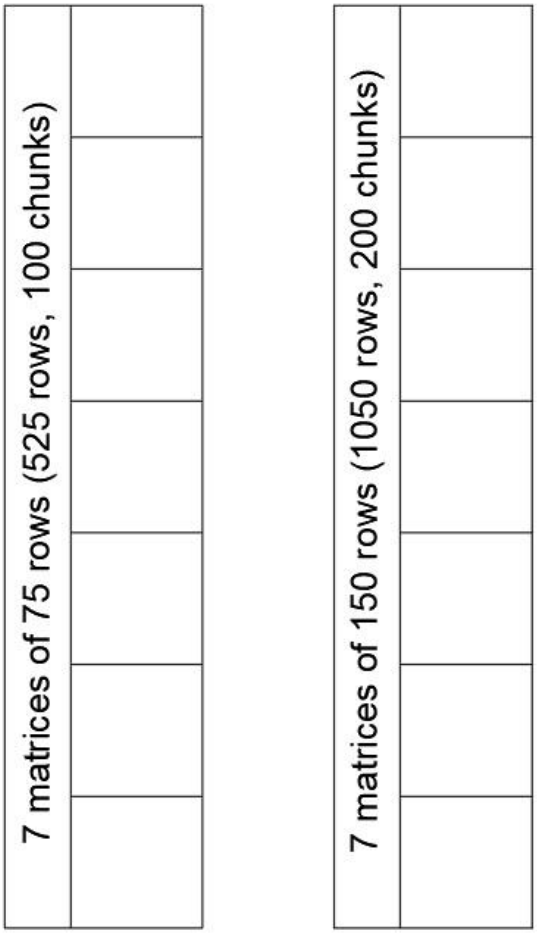


Figure 5.2 - Train matrices for same working condition - 100 and 200 chunks

C17: this approach is for same class of defect and the final matrix has 1275 rows for 100 chunks and 2550 rows for 200 chunks. (*Figure 5.3)*

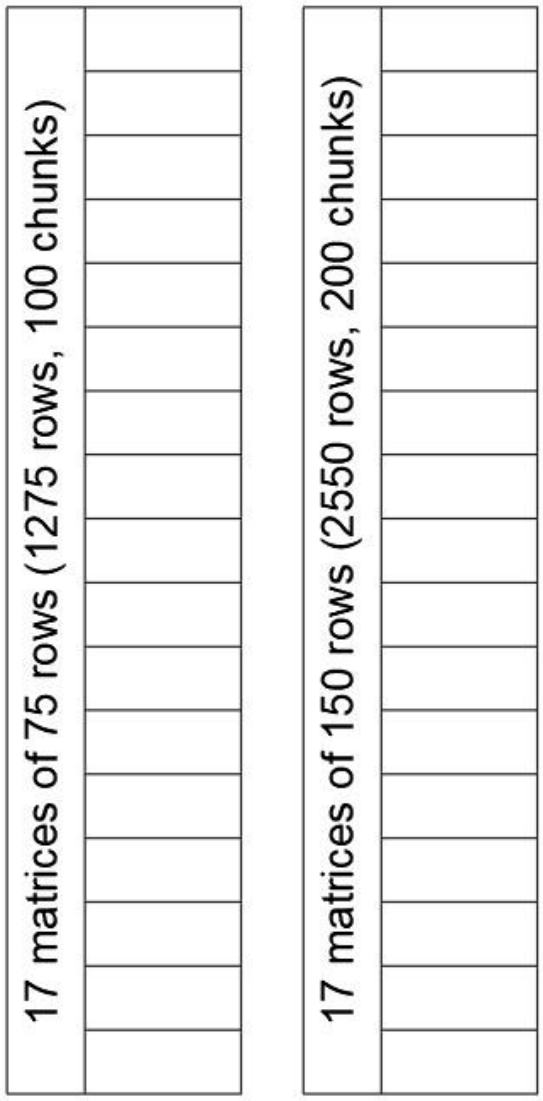


Figure 5.3 - Train matrices for same class of defect - 100 and 200 chunks

L21 & S21: this approach is for same load and same speed and the final matrix has 1575 rows for 100 chunks and 3150 rows for 200 chunks. (*Figure 5.4)*

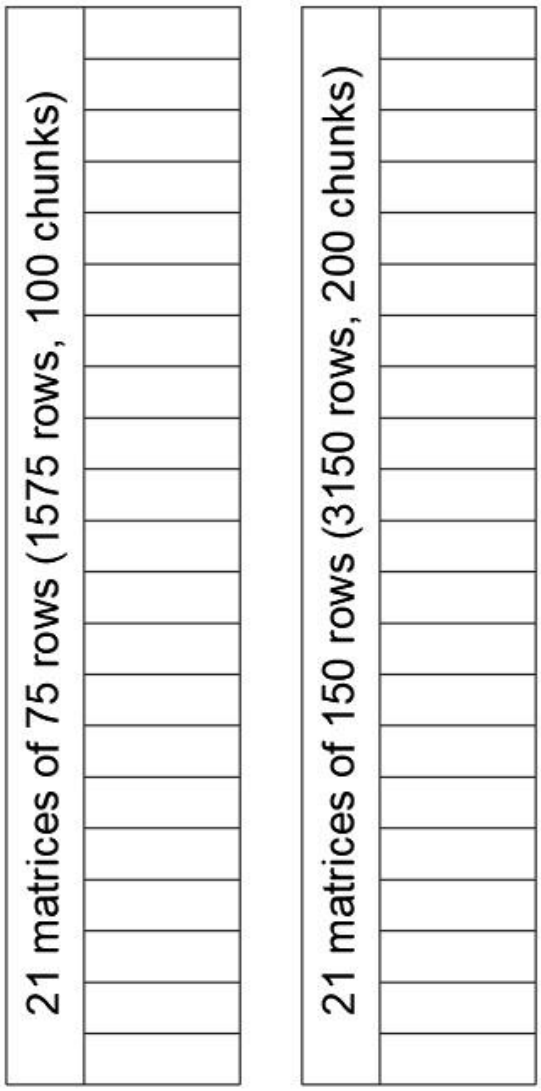


Figure 5.4 - Train matrices for same load & same speed - 100 and 200 chunks

Based on the models, the final matrices are like:

M: the final matrix to be used as the machine learning model creation input has six dimensions.

For example, **M_W7_100** is 6×525 or another one such as **M_L21_200** is 6×3150.

MR: the final matrix to be used as the machine learning model creation input has twelve dimensions.

For instance, **MR_C17_200** is 12×2550 or another one such as **MR_S21_100** is 12×1575.

MRK: the final matrix to be used as the machine learning model creation input has eighteen dimensions.

As an example, **MRK_C17_100** is 18×1275 or another one such as **MRK_S21_200** is 18×3150.

MRKS: the final matrix to be used as the machine learning model creation input has twenty-four dimensions.

To give an example, **MRKS_W7_200** is 24×1050 or another one such as **MRKS_L21_100** is 24×1575.

5.4.3. Test matrices

As mentioned before test matrix for 100 data points has 25 rows and for 200 data points has 50 rows. (25 percent of data for test)

For different approaches the matrices are like:

W7: this approach is for same working condition and the final matrix has 175 rows for 100 chunks and 350 rows for 200 chunks. (*Figure 5.5)*

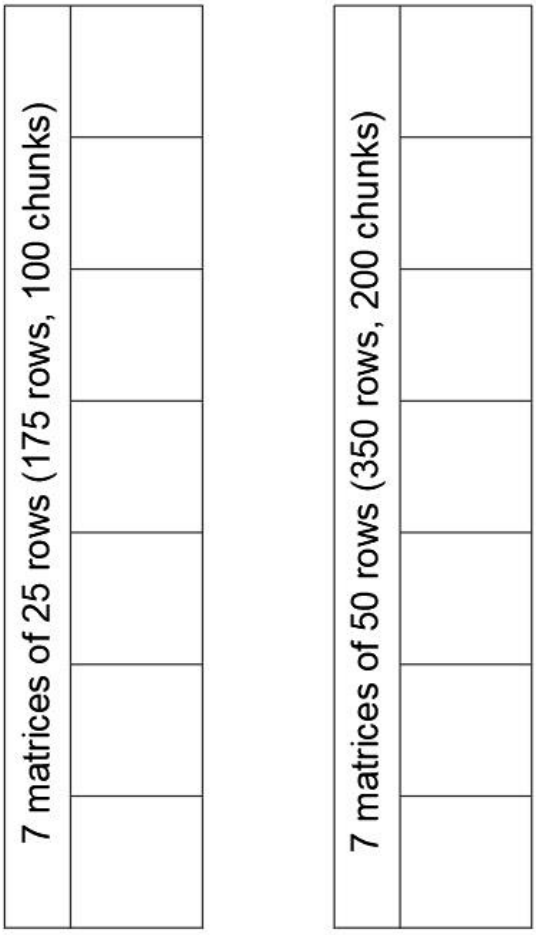


Figure 5.5 - Test matrices for same working condition - 100 and 200 chunks

C17: this approach is for same class of defect and the final matrix has 425 rows for 100 chunks and 850 rows for 200 chunks. (*Figure 5.6)*

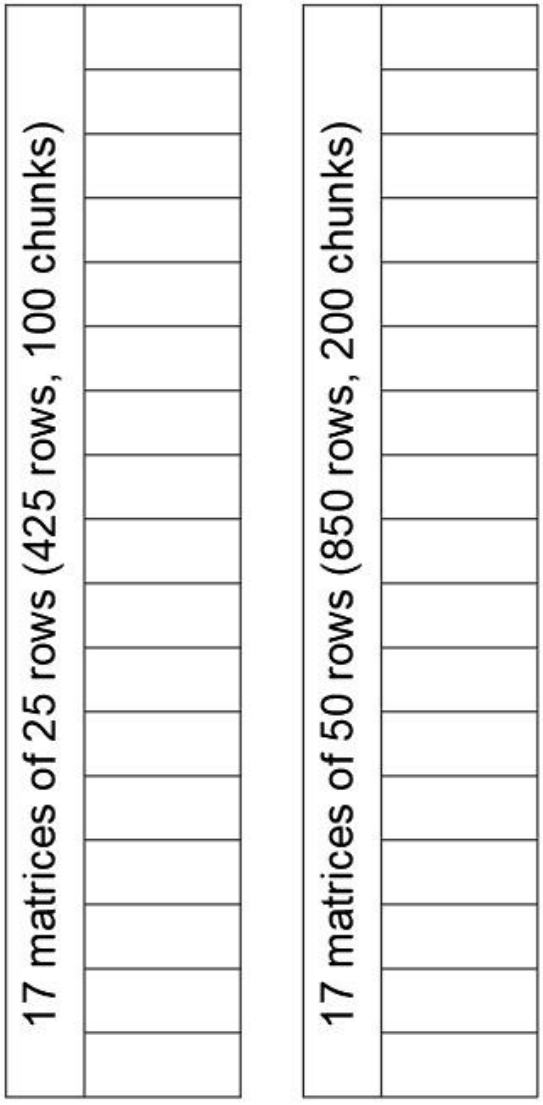


Figure 5.6 - Test matrices for same class of defect - 100 and 200 chunks

L21 & S21: this approach is for same load and same speed and the final matrix has 525 rows for 100 chunks and 1050 rows for 200 chunks. (*Figure 5.7)*

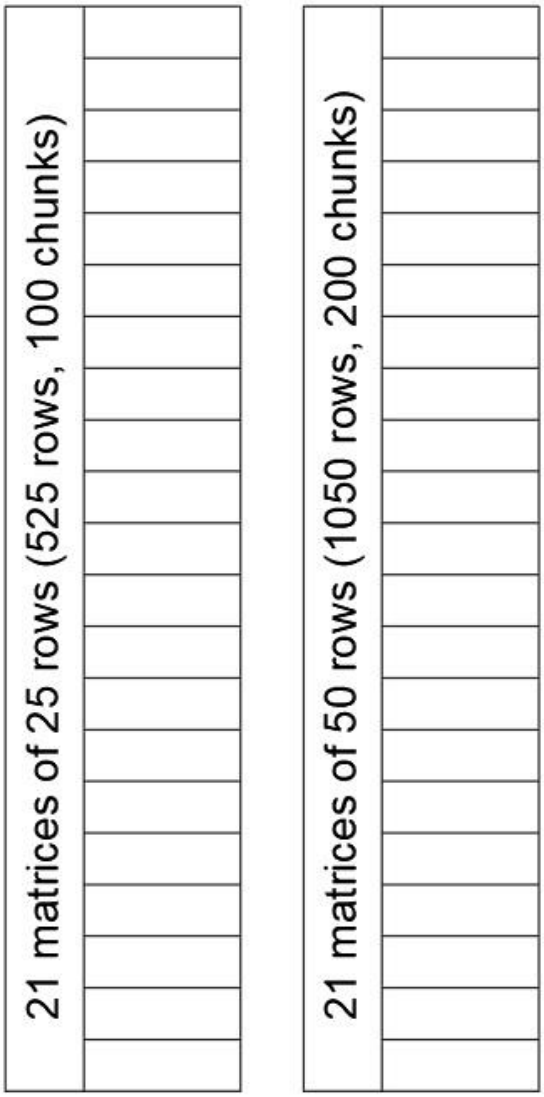


Figure 5.7 - Test matrices for same load & same speed - 100 and 200 chunks

Having created the machine learning model, it is time to examine its precision in predicting the classes by means of test matrices.

M: the final matrix to be used has six dimensions.

For example, **M_W7_100** is 6×175 or another one such as **M_L21_200** is 6×1050.

MR: the final matrix to be used has twelve dimensions.

For instance, **MR_C17_200** is 12×850 or another one such as **MR_S21_100** is 12×525.

MRK: the final matrix to be used has eighteen dimensions.

As an example, **MRK_C17_100** is 18×425 or another one such as **MRK_S21_200** is 18×1050.

MRKS: the final matrix to be used has twenty-four dimensions.

To give an example, **MRKS_W7_200** is 24×350 or another one such as **MRKS_L21_100** is 24×525.

Chapter Six

Machine Learning Process Results

In this chapter the machine learning process results are provided. But before presenting the results it is useful to give some information about the algorithms to be used in this process.

As it was mentioned in the previous chapter the machine learning is divided in two categories based on methods, supervised and unsupervised.

Because of being labeled, our data is considered as supervised learning type. The field of supervision is related to classes so the algorithms to be discussed are those pertaining to classification.

Prior to the introduced classifiers there are five important ones that are:

Naïve-Bayes Classifier

K-Nearest Neighbors Classifier

Decision Tree Classifier

Support Vector Machines (SVM)

Discriminant analysis Classifier

In the following section these classifiers are briefly explained.

6.1. Classification Algorithms

6.1.1. Naïve-Bayes

These classifiers are a group of probabilistic classifiers based on Bayes' theorem with strong independence among features. In other words, Naïve Bayes classifier presumes that the presence of a unique feature is not related to presence of any other features.

Apart from its simplicity, Naïve Bayes is efficiently useful for large data sets and outperforms many other highly sophisticated methods.

6.1.2. K-Nearest Neighbors

This algorithm is a non-parametric classification method in which the input is made out of *k* closest training examples and the output is classified by means of its *k* nearest neighbors.

A useful technique for classification could be considering weights for neighbors' contribution in the way that the nearer neighbors have effect more than average in comparing to the further neighbors.

6.1.3. Decision Tree

This classifier uses a decision tree in order to go from observations to target results.

In these classifiers, leaves are class labels, however, branches represent features result in class labels.

6.1.4. Support Vector Machines (SVM)

This classifier is one of the most powerful prediction algorithms which is a non-probabilistic linear classifier. SVM trains the data to groups of points in space in order to maximize the distance between two categories. Then new examples are brought into the same space and assigned to a category based on the position they are placed.

Apart from the linear classification, SVM can perform a non-linear classification by means of *kernel* trick for high-dimensional space.

6.1.5. Discriminant Analysis

This classifier is used to find a combination of features which categorizes the data into two or more classes. Each group must have a score on predictor measures as well as a score on group measures. This acts in distributing data into classes of the same type.

This classification algorithm has two main forms, linear and quadratic, using linear and quadratic decision surfaces respectively. Quadratic form is the more general type of the linear one.

6.2. Train and Validation of Machine Learning Models

In this section the procedure and results of different models with different approaches and different chunks are explained[1].

By assuming that the precision (not compulsorily accuracy) of models with more data points (200 chunks) is higher (because of the larger number of train and test data points), in this section just the 200-chunk models are discussed.

Another assumption is that the models with lowest and highest number of features are discussed. (i.e. **M** with six dimensions and **MRKS** with twenty-four dimensions)

To clarify, from 32 possible models, only eight are going to be studied.

To have a better understanding of different models charts and figures, it is worth introducing each group number (each matrix) in each model.

W7 model:

1: 0A – 200 Hz – 1400 N
2: 1A – 200 Hz – 1400 N
3: 2A – 200 Hz – 1400 N
4: 3A – 200 Hz – 1400 N
5: 4A – 200 Hz – 1400 N
6: 5A – 200 Hz – 1400 N
7: 6A – 200 Hz – 1400 N

C17 model:

1: 2A – 100 Hz – 0 N
2: 2A – 100 Hz – 1000 N
3: 2A – 100 Hz – 1400 N
4: 2A – 100 Hz – 1800 N
5: 2A – 200 Hz – 0 N
6: 2A – 200 Hz – 1000 N
7: 2A – 200 Hz – 1400 N
8: 2A – 200 Hz – 1800 N
9: 2A – 300 Hz – 0 N
10: 2A – 300 Hz – 1000 N
11: 2A – 300 Hz – 1400 N
12: 2A – 300 Hz – 1800 N
13: 2A – 400 Hz – 0 N
14: 2A – 400 Hz – 1000 N
15: 2A – 400 Hz – 1400 N
16: 2A – 500 Hz – 0 N
17: 2A – 500 Hz – 1000 N

L21 model:

1: 0A – 100 Hz – 1800 N
2: 0A – 200 Hz – 1800 N
3: 0A – 300 Hz – 1800 N
4: 1A – 100 Hz – 1800 N
5: 1A – 200 Hz – 1800 N
6: 1A – 300 Hz – 1800 N
7: 2A – 100 Hz – 1800 N
8: 2A – 200 Hz – 1800 N
9: 2A – 300 Hz – 1800 N
10: 3A – 100 Hz – 1800 N
11: 3A – 200 Hz – 1800 N
12: 3A – 300 Hz – 1800 N
13: 4A – 100 Hz – 1800 N
14: 4A – 200 Hz – 1800 N
15: 4A – 300 Hz – 1800 N
16: 5A – 100 Hz – 1800 N
17: 5A – 200 Hz – 1800 N
18: 5A – 300 Hz – 1800 N
19: 6A – 100 Hz – 1800 N
20: 6A – 200 Hz – 1800 N
21: 6A – 300 Hz – 1800 N

[1] Results of full models are listed in Appendix D.

S21 model:

1: 0A – 400 Hz – 0 N
2: 0A – 400 Hz – 1000 N
3: 0A – 400 Hz – 1400 N
4: 1A – 400 Hz – 0 N
5: 1A – 400 Hz – 1000 N
6: 1A – 400 Hz – 1400 N
7: 2A – 400 Hz – 0 N
8: 2A – 400 Hz – 1000 N
9: 2A – 400 Hz – 1400 N
10: 3A – 400 Hz – 0 N
11: 3A – 400 Hz – 1000 N
12: 3A – 400 Hz – 1400 N
13: 4A – 400 Hz – 0 N
14: 4A – 400 Hz – 1000 N
15: 4A – 400 Hz – 1400 N
16: 5A – 400 Hz – 0 N
17: 5A – 400 Hz – 1000 N
18: 5A – 400 Hz – 1400 N
19: 6A – 400 Hz – 0 N
20: 6A – 400 Hz – 1000 N
21: 6A – 400 Hz – 1400 N

All validations are done based on *Cross Validation* technique with *10-folds*.

6.3. Algorithms Selection Based on Train Accuracies and Results

Having trained the models, the subsets of each algorithm accuracies are compared to choose the best and the worst for discussion.

6.3.1. Naïve-Bayes

Both *Gaussian* and *Kernel Naïve Bayes* have acceptable performances.

6.3.2. K-Nearest Neighbors

Like previous algorithm, all the subsets of this algorithm have almost the same and accepted performance except *Coarse KNN* which has a moderate accuracy.

6.3.3. Decision Tree

Both *Fine* and *Medium Tree* subsets have very good performances, however, the worst subset among all algorithms is the *Coarse Tree*.

6.3.4. Support Vector Machines (SVM)

All subsets of *SVM* have great performances in general and one of two best subsets among all algorithms is *Medium Gaussian SVM*.

6.3.5. Discriminant analysis

Both *Linear* and *Quadratic Discriminant* have great performances and the second best subset among all algorithms is *Quadratic Discriminant*.

Four algorithms are: Coarse Tree (*weak*), Coarse KNN (*moderate*), Quadratic Discriminant and Medium Gaussian SVM (*powerful*).

6.4. Data Interpretation

In this chapter, the data related to the machine learning procedure and different algorithms' train accuracies are given in the form of confusion matrix.

6.4.1. Confusion Matrix[2]

In supervised learning method of machine learning, data could be provided in some special tables which allow a visual representation of an algorithm's performance. In such matrices the rows show the actual class while the columns are representatives for predicted classes.

In confusion matrix there could be four different outcomes, based on either the class is predicted correctly or not.

True Positive (TP): dedicated to the portion for which the data is correctly accepted.

True Negative (TN): dedicated to the portion for which the data is correctly rejected.

False Positive (FP): dedicated to the portion for which the predicted class is wrongly accepted.

False Positive (FN): dedicated to the portion for which the predicted class is wrongly rejected.

If any of these outcomes calculated in the form of percentages other than numbers, they are presented in the rate forms as TPR, TNR, FPR and FNR.

6.4.2. Classification Learner App

For having a better visual presentation for different models' confusion matrices the *Classification Learner App* from **MATLAB** is used.

To have a comparison between different models, two models with lowest and highest dimensions (**M** and **MRKS**) are discussed for the *Same Speed* approach and 200 chunks.

As it can be seen, the train accuracy of different models from weakest to the best are illustrated and it is vividly clear that the performance of machine learning procedure for MRKS model is notably higher in general.

Full confusion matrices of **M** and **MRKS** models for different approaches and 200 chunks are depicted in ***Appendix C***.

[2] https://en.wikipedia.org/

M_S21_200

Coarse Tree (*Train Accuracy: 23.6%*)

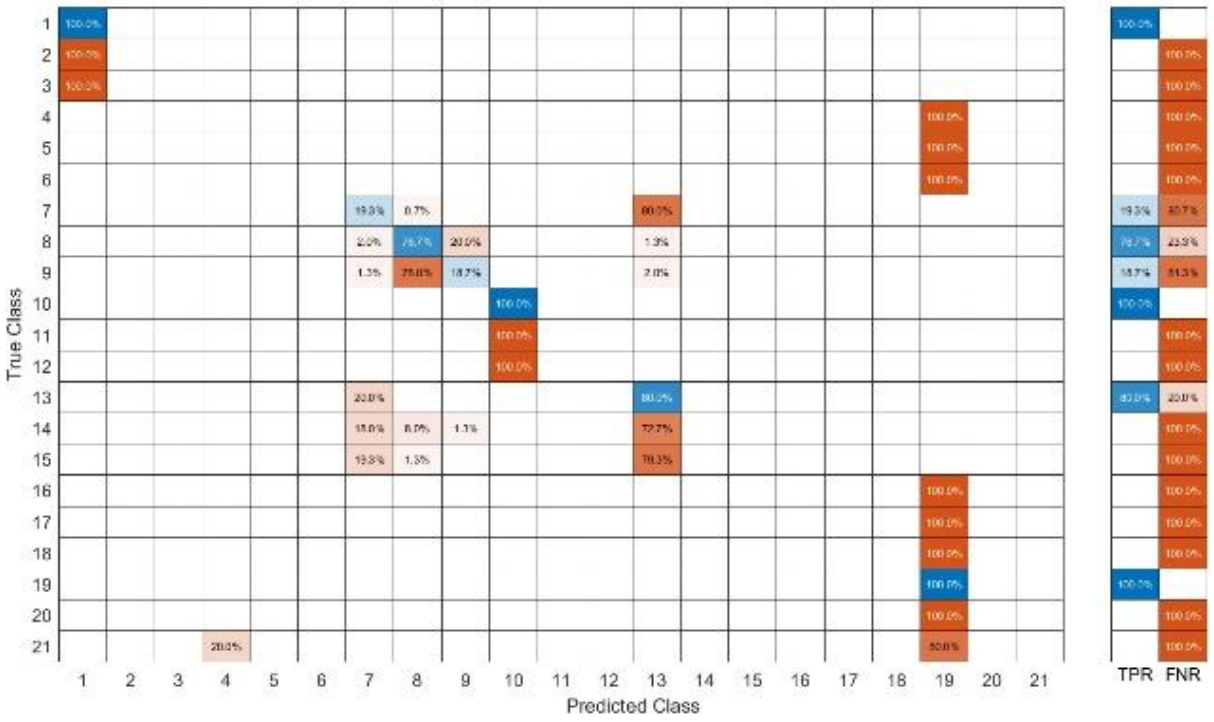


Figure 6.1 – a) Coarse tree confusion matrix

Coarse KNN (*Train Accuracy: 65.3%*)

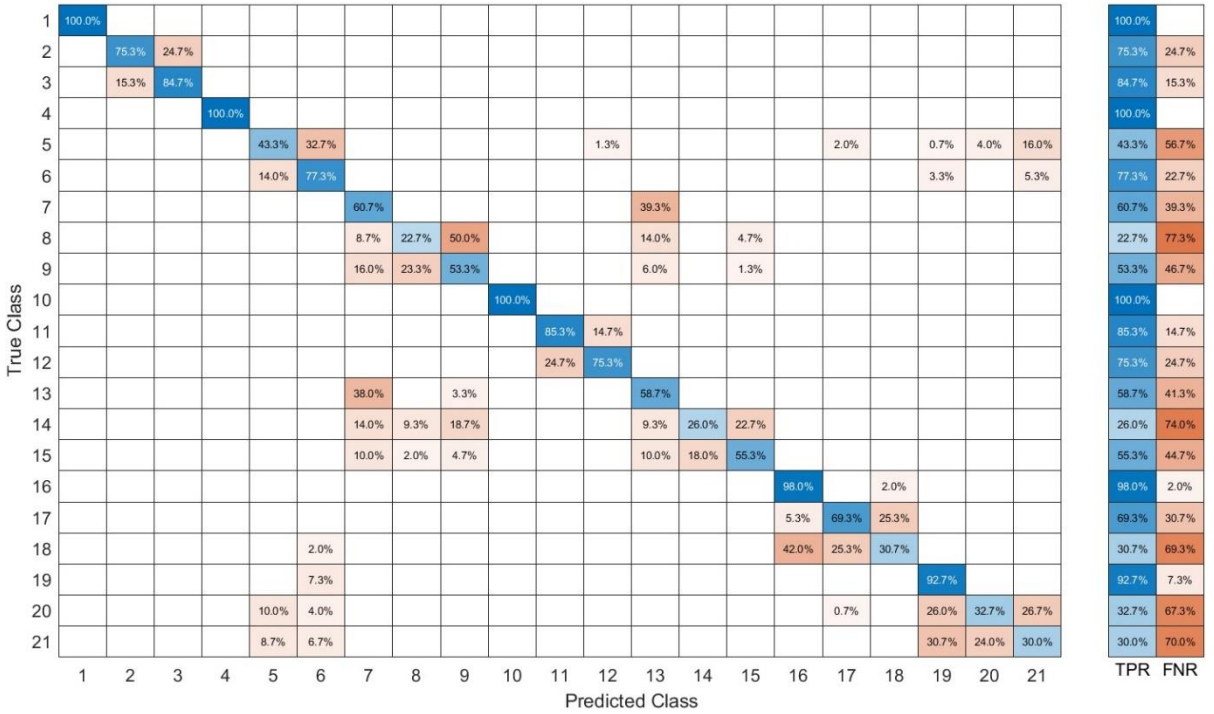


Figure 6.1 – b) Coarse KNN confusion matrix

M_S21_200

Quadratic Discriminant (*Train Accuracy: 90.9%*)

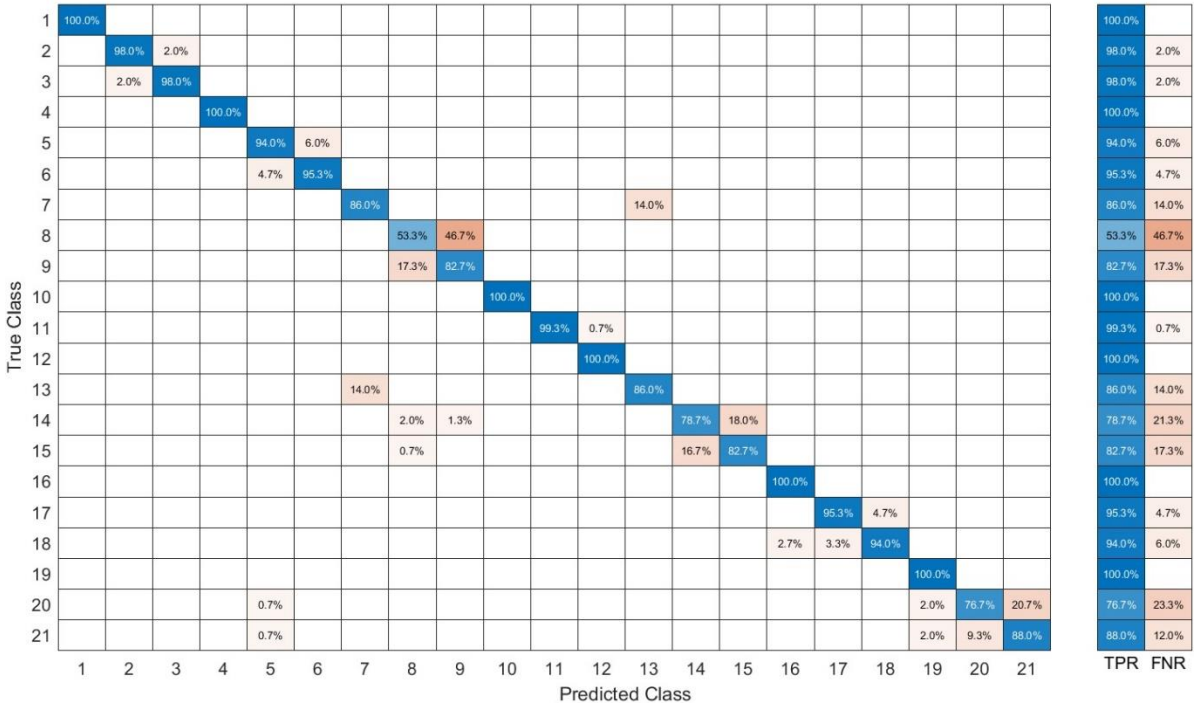


Figure 6.1 – c) Quadratic discriminant confusion matrix

Medium Gaussian SVM (*Train Accuracy: 90.3%*)

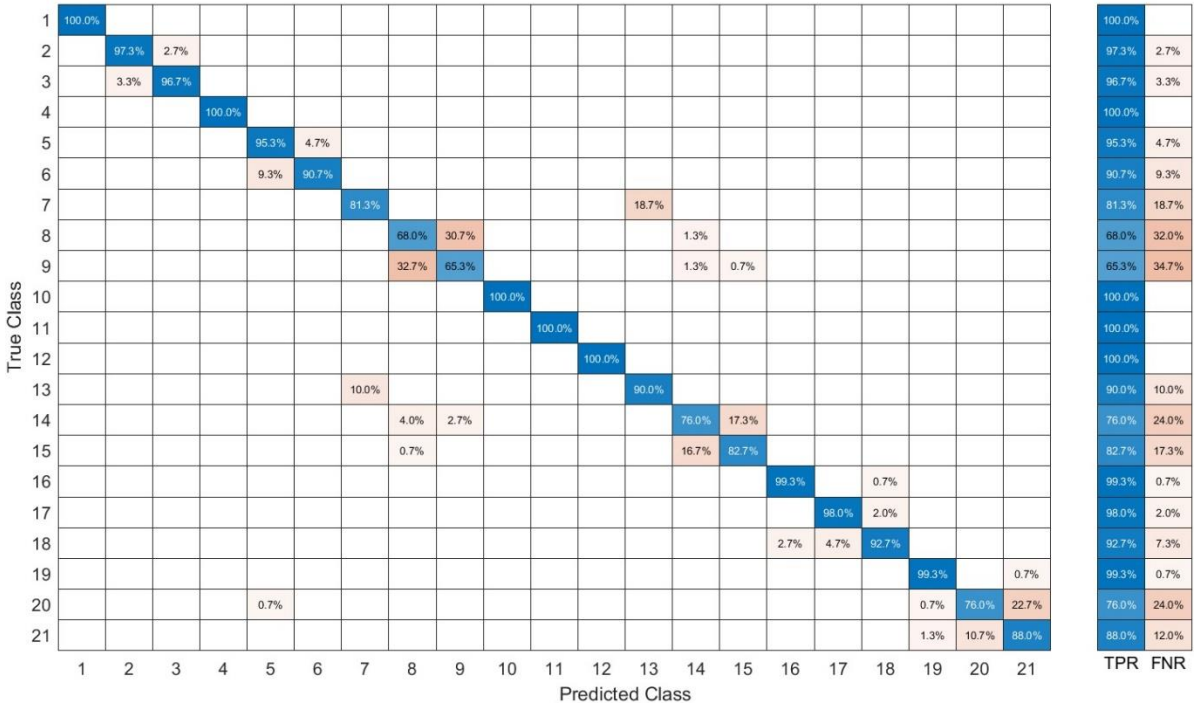


Figure 6.1 – d) Medium Gaussian SVM confusion matrix

MRKS_S21_200

Coarse Tree (*Train Accuracy: 23.8%*)

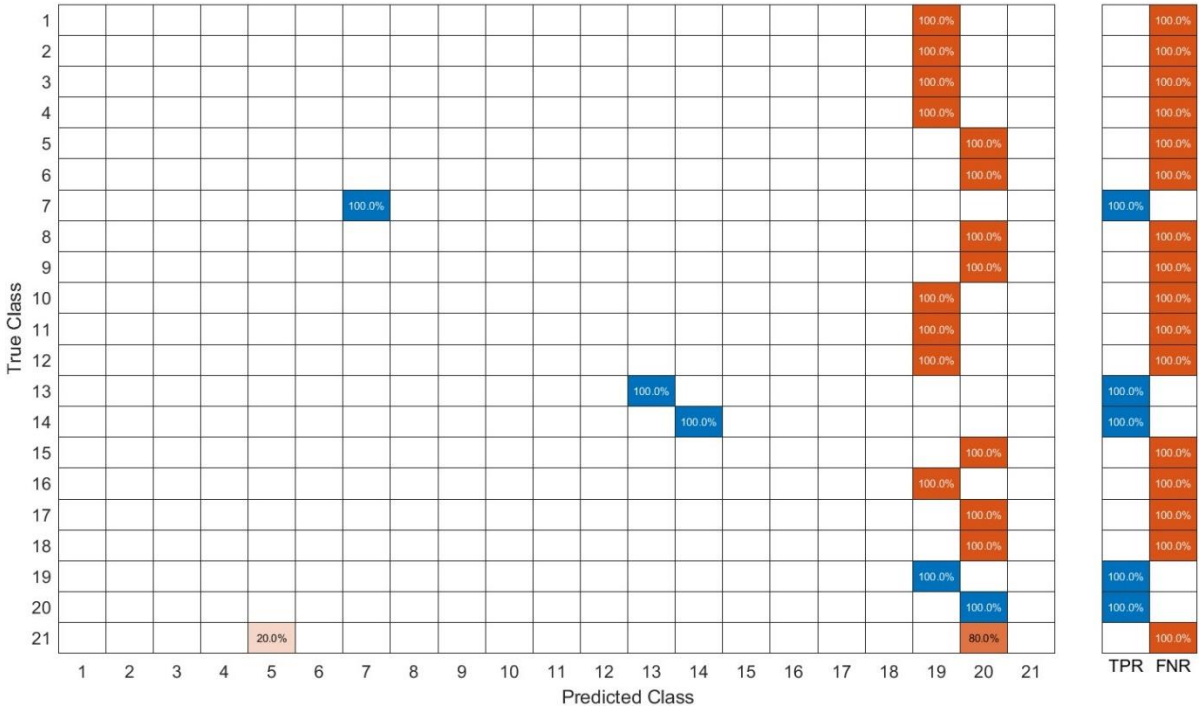


Figure 6.2 – a) Coarse tree confusion matrix

Coarse KNN (*Train Accuracy: 90.3%*)

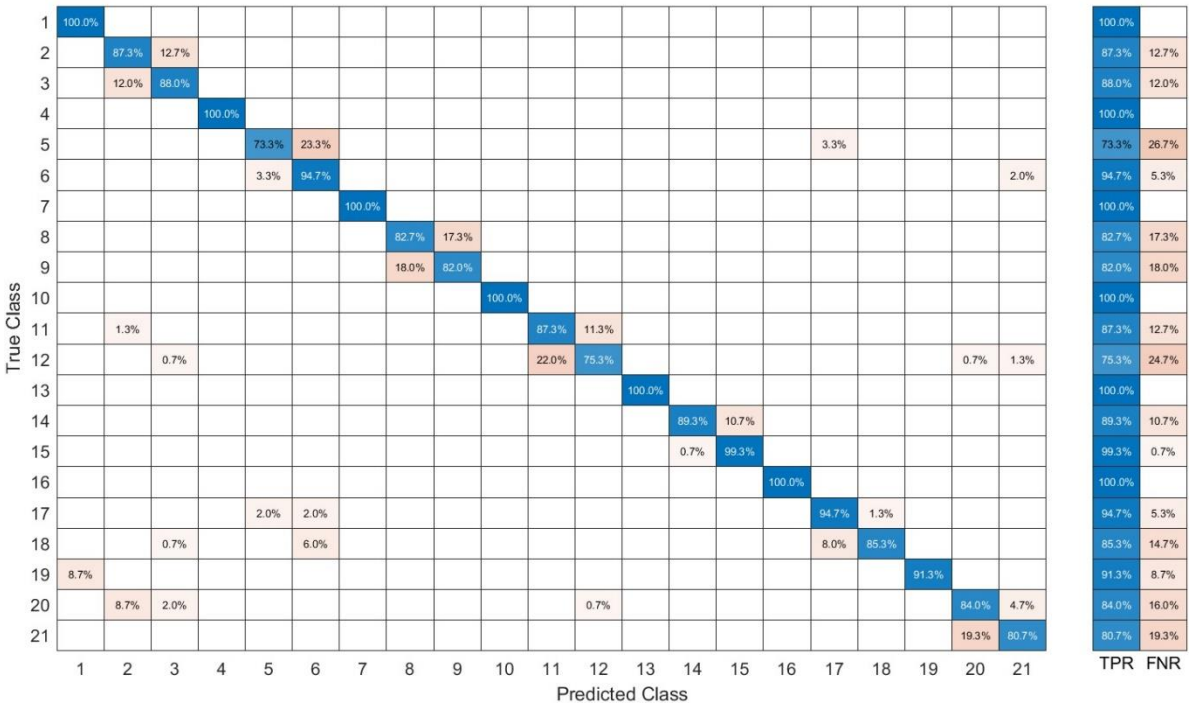


Figure 6.2 – b) Coarse KNN confusion matrix

MRKS_S21_200

Quadratic Discriminant (*Train Accuracy: 100.0%*)

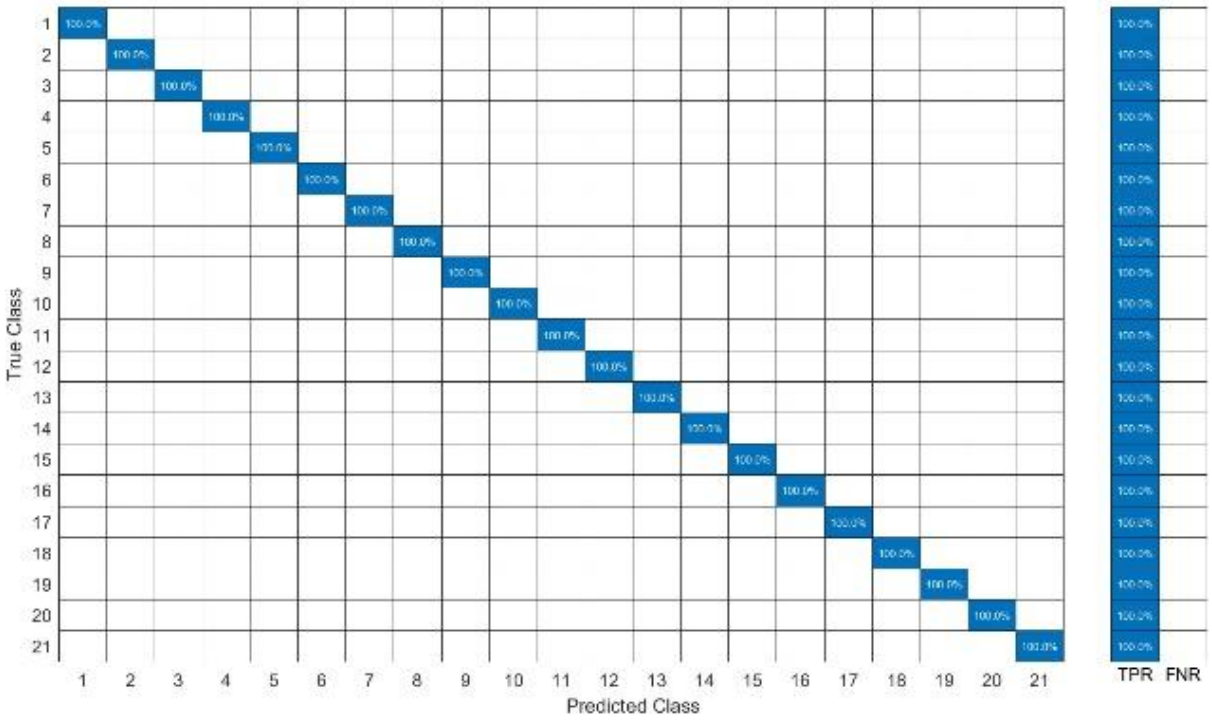


Figure 6.2 – c) Quadratic discriminant confusion matrix

Medium Gaussian SVM (*Train Accuracy: 99.9%*)

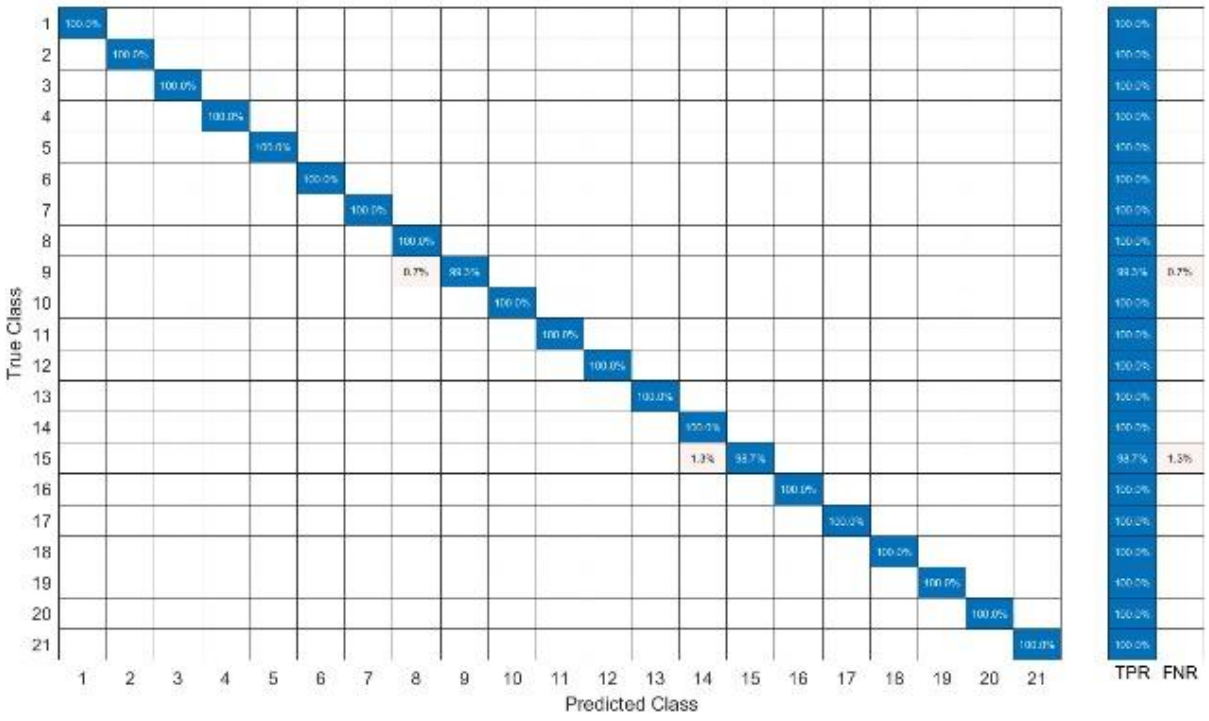


Figure 6.2 – d) Medium Gaussian SVM confusion matrix

6.5. Test Results

Having created the models, it is time to estimate their performances by inducing the test data into prediction function and find out how well the models are working.

Before presenting the result, it is noteworthy to mention two definitions.

By considering each test matrix source, the matrix classification label is known. By comparing the results with the original label vectors there could be two possible options, first the class is predicted wrongly and second it is predicted correctly.

6.5.1. Prediction Performance

If the test data is predicted correctly regarding the classification label vector, the outcome of correct items shows the performance of the model. Higher number of correct predicted items shows a better model.

6.5.2. Misclassification Error

If the test data is predicted wrongly regarding the classification label vector, the outcome of incorrect items shows the misclassification error of the model. Lower number of incorrect predicted items shows a better model.

6.5.3. Test Prediction

Having trained the data and exported the created model, then it is time to predict the outcome of the test data. The model can be called *TrainedModel* and used in the prediction function.

The total test matrix is introduced as *TTest*. Also a label vector ***q*** is created for each TTest matrix with the same procedure explained for train data by considering that the size of these label vectors comparing to train data label vectors are one third due to the percentages explained for train and test data creation (75 percent for train and 25 percent for test).

```
P=TrainedModel.predictFcn(TTest)
```

The outcome of prediction function is called *P*, it will have the same size as *q*. Vector *q* as the reference and vector *P* as the predicted data are going to be compared with each other.

The number of correct items are considered for the *Prediction Performance* and also the wrong items show the *Misclassification Error*.

The result of different models for different algorithms are provided in the upcoming tables.

M_W7_200	Coarse Tree	Coarse KNN	Quadratic Discriminant	Medium Gaussian SVM
Train Accuracy	71.3	81.0	98.4	99.0
Prediction Performance	71.43	82.57	96.57	97.43
Misclassification Error	28.57	17.43	3.43	2.57
MRKS_W7_200	**Coarse Tree**	**Coarse KNN**	**Quadratic Discriminant**	**Medium Gaussian SVM**
Train Accuracy	71.4	99.8	100.0	100.0
Prediction Performance	71.43	100.00	100.00	100.00
Misclassification Error	28.57	0.00	0.00	0.00

Table 6.1 - Same working condition

M_C17_200	Coarse Tree	Coarse KNN	Quadratic Discriminant	Medium Gaussian SVM
Train Accuracy	26.3	79.8	88.3	88.5
Prediction Performance	25.88	80.71	87.18	89.41
Misclassification Error	74.12	19.29	12.82	10.59
MRKS_C17_200	**Coarse Tree**	**Coarse KNN**	**Quadratic Discriminant**	**Medium Gaussian SVM**
Train Accuracy	29.4	84.4	98.6	97.7
Prediction Performance	29.41	83.29	97.65	97.53
Misclassification Error	70.59	16.71	2.35	2.47

Table 6.2 - Same class of defect

M_L21_200	Coarse Tree	Coarse KNN	Quadratic Discriminant	Medium Gaussian SVM
Train Accuracy	23.8	74.3	97.0	96.8
Prediction Performance	23.81	76.00	97.05	97.24
Misclassification Error	76.19	24.00	2.95	2.76
MRKS_L21_200	**Coarse Tree**	**Coarse KNN**	**Quadratic Discriminant**	**Medium Gaussian SVM**
Train Accuracy	23.8	97.2	100.0	100.0
Prediction Performance	23.81	98.19	100.0	99.90
Misclassification Error	76.19	1.81	0.00	0.10

Table 6.3 - Same load

M_S21_200	Coarse Tree	Coarse KNN	Quadratic Discriminant	Medium Gaussian SVM
Train Accuracy	23.6	65.3	90.9	90.3
Prediction Performance	23.43	65.81	91.05	91.05
Misclassification Error	76.57	34.19	8.95	8.95
MRKS_S21_200	**Coarse Tree**	**Coarse KNN**	**Quadratic Discriminant**	**Medium Gaussian SVM**
Train Accuracy	23.8	90.3	100.0	99.9
Prediction Performance	23.81	91.90	100.00	100.00
Misclassification Error	76.19	8.10	0.00	0.00

Table 6.4 - Same speed

As it can be seen, in general models with higher number of features present a better predictions and higher performances in comparison with the lower dimensional models.

Also prediction of same working condition, because of having known load and speed from different classes of defect is more accurate than the same class of defect.

Accuracy of same load and same speed are also the equal in general and there is not a highly remarkable difference between them.

Based on algorithms, quadratic discriminant and medium Gaussian SVM are presenting an almost similar great performance, however, coarse KNN is operating moderately. The worst algorithm is coarse tree which has the highest accuracy for the models with lower number of predictors. It shows by growing the number of predictors the accuracy of this algorithm falls drastically.

Full machine learning results of 32 different models are tabulated in ***Appendix D***.

Chapter Seven

Feature Selection

In this chapter the matter of feature selection is discussed. However, in statistics feature selection is applied for high dimensional datasets, here as a comparative complementary part is explained and discussed.

Based on recent developments, it is not unusual to work with high dimensional datasets varying from hundreds of features up to tens of thousands. Working with a lot of features may sometimes ends in an accurate calculation but undoubtedly it will have some drawbacks as well.

To explain more, sometimes the high number of features cause a high cost of computation and also may lead to less accurate models and results. It becomes important to choose some more important features to simultaneously reduce the costs of evaluation as well as keeping the redundant features out in order to have the optimum accuracy.

This could be done by means of features selection algorithms. However, statistically this thesis dataset is not very high dimensional.

7.1. Feature Selection Methods[1]

In machine learning, there could be two ways to think about features selection.

First it could be based on supervised or unsupervised methods. The supervised techniques use target variables, however, the unsupervised ignore them.

The second way is based on three different algorithms: *Filter Methods*, *Wrapper Methods* and *Embedded Methods*.

7.1.1. Filter Methods

Filter methods are used for pre training step. The selection takes place based on the importance of features for their correlation with the outcome rather than being dependent on the machine learning algorithms.

These methods could be used for both classification and regression and both categorical and continuous features could be studied.

7.1.2. Wrapper Methods

These methods are used to create a subset of features and training the model then by considering the inferences of the model some features are added or removed from the subset. The problem of wrapper methods is that these methods are bound to research and are computationally expensive.

7.1.3. Embedded Methods

These methods are combinations of filter and wrapper methods. These methods have their own built-in algorithms for feature selection.

Both wrapper and embedded methods are mostly used for regression and mostly applicable for continuous features.

7.1.4. Differences Between Filter Methods and Wrapper Methods

Filter methods find the relevance of features by considering their correlation with dependent variable, however, wrapper methods measure the convenience of a feature subset.

Filter methods are much faster and also wrapper methods are computationally highly expensive.

Statistical methods are used for filter methods while cross validation is used for wrapper methods.

Wrapper methods are more likely to experience overfitting comparing to the filter methods.

[1] https://machinelearningmastery.com/
https://www.analyticsvidhya.com/

7.2. Methods to be used[2]

In this paper two algorithms of filter method are briefly explained, used and the results are discussed.

7.2.1. Chi-Square

In statistics chi-square is used to find the independence of two items. If we have two variables one as observed and the other as expected, chi-square will specify that how these two variables affect each other and cause deviation.

If we consider the independent features as predictors and dependent ones as responses, in feature selection the goal is to find the features with highest dependence on response.

When the observed and expected counts are very close to each other, the features are considered independent and the chi-square value is smaller. To clarify, higher the chi-square value, more dependence of feature on the response.

7.2.2. Maximum Relevance Minimum Redundancy (MRMR)

This algorithm has become widely popular after being used by Uber engineers.

Maximum relevance minimum redundancy (MRMR) is called so because at each level of iteration the feature which has the maximum relevance with respect to target and the minimum redundancy with respect to the previous iterations is aimed to be selected.

7.3. Models for Feature Selection

The highest dimensions of datasets are dedicated to **MRKS** (with twenty-four) models. The feature selection methods are applied on these models.

To have a comparison between the featured models and the models with lowest dimensions **M** (with six) six best features are selected to create a new sub model of MRKS.

The new matrices are formed out of MRKS models by considering the order of features after applying the feature selection methods. The best six features make the six columns of the new six dimensional matrices respectively.

In the upcoming pages, four approaches are considered as well as four machine learning algorithms and also the best model MRKS the worst model M and two featured models of chi-square and MRMR derived from MRKS model.

The train accuracies, prediction performances and also the misclassification errors are reported as follows:

[2] https://towardsdatascience.com/

M_W7_200	Coarse Tree	Coarse KNN	Quadratic Discriminant	Medium Gaussian SVM
Train Accuracy	71.3	81.0	98.4	99.0
Prediction Performance	71.43	82.57	96.57	97.43
Misclassification Error	28.57	17.43	3.43	2.57
MRKS_W7_200	**Coarse Tree**	**Coarse KNN**	**Quadratic Discriminant**	**Medium Gaussian SVM**
Train Accuracy	71.4	99.8	100.0	100.0
Prediction Performance	71.43	100.00	100.00	100.00
Misclassification Error	28.57	0.00	0.00	0.00
Chi-Square	**Coarse Tree**	**Coarse KNN**	**Quadratic Discriminant**	**Medium Gaussian SVM**
Train Accuracy	71.4	98.3	100.0	100.0
Prediction Performance	87.14	96.38	96.67	96.67
Misclassification Error	12.86	3.62	3.33	3.33
MRMR	**Coarse Tree**	**Coarse KNN**	**Quadratic Discriminant**	**Medium Gaussian SVM**
Train Accuracy	71.2	97.0	100.0	100.0
Prediction Performance	87.14	95.62	96.67	96.57
Misclassification Error	12.86	4.38	3.33	3.43

Table 7.1 - Same working condition

M_C17_200	Coarse Tree	Coarse KNN	Quadratic Discriminant	Medium Gaussian SVM
Train Accuracy	26.3	79.8	88.3	88.5
Prediction Performance	25.88	80.71	87.18	89.41
Misclassification Error	74.12	19.29	12.82	10.59
MRKS_C17_200	**Coarse Tree**	**Coarse KNN**	**Quadratic Discriminant**	**Medium Gaussian SVM**
Train Accuracy	29.4	84.4	98.6	97.7
Prediction Performance	29.41	83.29	97.65	97.53
Misclassification Error	70.59	16.71	2.35	2.47
Chi-Square	**Coarse Tree**	**Coarse KNN**	**Quadratic Discriminant**	**Medium Gaussian SVM**
Train Accuracy	26.3	79.1	88.3	88.7
Prediction Performance	25.88	80.71	87.18	89.41
Misclassification Error	74.12	19.29	12.82	10.59
MRMR	**Coarse Tree**	**Coarse KNN**	**Quadratic Discriminant**	**Medium Gaussian SVM**
Train Accuracy	29.4	62.9	91.6	88.4
Prediction Performance	29.41	63.18	91.41	89.76
Misclassification Error	70.59	36.82	8.59	10.24

Table 7.2 - Same class of defect

M_L21_200	Coarse Tree	Coarse KNN	Quadratic Discriminant	Medium Gaussian SVM
Train Accuracy	23.8	74.3	97.0	96.8
Prediction Performance	23.81	76.00	97.05	97.24
Misclassification Error	76.19	24.00	2.95	2.76
MRKS_L21_200	**Coarse Tree**	**Coarse KNN**	**Quadratic Discriminant**	**Medium Gaussian SVM**
Train Accuracy	23.8	97.2	100.0	100.0
Prediction Performance	23.81	98.19	100.0	99.90
Misclassification Error	76.19	1.81	0.00	0.10
Chi-Square	**Coarse Tree**	**Coarse KNN**	**Quadratic Discriminant**	**Medium Gaussian SVM**
Train Accuracy	23.8	74.3	97.0	96.8
Prediction Performance	23.81	76.00	97.05	97.24
Misclassification Error	76.19	24.00	2.95	2.76
MRMR	**Coarse Tree**	**Coarse KNN**	**Quadratic Discriminant**	**Medium Gaussian SVM**
Train Accuracy	23.8	80.2	99.7	99.0
Prediction Performance	23.81	80.00	99.71	99.14
Misclassification Error	76.19	20.00	0.29	0.86

Table 7.3 - Same load

M_S21_200	Coarse Tree	Coarse KNN	Quadratic Discriminant	Medium Gaussian SVM
Train Accuracy	23.6	65.3	90.9	90.3
Prediction Performance	23.43	65.81	91.05	91.05
Misclassification Error	76.57	34.19	8.95	8.95
MRKS_S21_200	**Coarse Tree**	**Coarse KNN**	**Quadratic Discriminant**	**Medium Gaussian SVM**
Train Accuracy	23.8	90.3	100.0	99.9
Prediction Performance	23.81	91.90	100.00	100.00
Misclassification Error	76.19	8.10	0.00	0.00
Chi-Square	**Coarse Tree**	**Coarse KNN**	**Quadratic Discriminant**	**Medium Gaussian SVM**
Train Accuracy	23.6	65.2	90.8	90.4
Prediction Performance	23.43	65.81	91.05	91.05
Misclassification Error	76.57	34.19	8.95	8.95
MRMR	**Coarse Tree**	**Coarse KNN**	**Quadratic Discriminant**	**Medium Gaussian SVM**
Train Accuracy	23.8	75.1	95.5	94.5
Prediction Performance	23.81	75.90	96.86	95.33
Misclassification Error	76.19	24.10	3.14	4.67

Table 7.4 - Same speed

As it is obvious the tables are representing the data related to the lowest dimensional model **M**, the highest dimensional model **MRKS**, and two feature selection methods which are formed based on the MRKS model features. At last the dimensionality of M model, chi-square and MRMR are the same (six dimensional)

For each model three items of train accuracy, prediction performance and misclassification error are reported explicitly for all four machine learning algorithms discussed in the previous chapter, coarse tree (weakest), coarse KNN (moderate), quadratic discriminant and medium Gaussian SVM (two strongest).

Except the first approach and that is same working condition for which both feature selection methods chi-square and MRMR show better performances in comparison with the lowest dimensional model **M**, for the other approaches the chi-square method almost has the same performance as the model **M**, however, the performance of MRMR method is better than the model.

This shows that by means of applying MRMR method it could be reachable to have a model with the calculation simplicity of **M** model which leads to a lower computational costs, however, the accuracy and performance are way better than the simple model.

Full machine learning results of eight different models and their pertaining feature selection models are tabulated in ***Appendix D***.

Chapter Eight

Conclusion

In this final chapter it is concluded that this work aims to help classification in different situations.

The data used for this paper is based on the research done in Politecnico di Torino and the details could be found at:

ftp://ftp.polito.it/people/DIRG_BearingData/.

At the beginning the test is shortly explained and also the main data is introduced.

Having introduced the main data, different elements of models are explained and model creation is clarified.

The models are made based on three different elements, *Chunks*, *Approaches* and *Features*.

Chunks are based on two divisions, 100 chunks of 0.1s and 200 chunks of 0.05s. Approaches are based on four categories of same working condition, same class of defect, same load and same speed. Features are based on four different categories of mean, RMS,

kurtosis and skewness. Models are made based on combination of these features.

Finally, there will be 32 different models which are used in this paper. These models are listed in ***Appendix A***.

For having a better visual understanding of how clusters of different working conditions are positioned in comparison with the other ones in different models, there should be illustrations but for this aim at most three dimensions are allowed. However, the models are at least six dimensional. To ease this problem a new method of dimensionality reduction is introduced which maintain the highest data possible with lowest data loss. *PCA* is the method used for dimensionality reduction and full illustrations are depicted in ***Appendix B***.

Having created the models, it is time to go through the data preparation for machine learning procedure. For different models there are different sizes of final matrices, but, all the train and test matrices are made based on 75 percent and 25 percent of main final matrix data points respectively.

After data preparation, different methods and algorithms of machine learning are explained. Supervised and unsupervised learning are discussed, and classification which is a subset of supervised learning is selected for machine learning process.

Different algorithm of classification is introduced such as *Decision Tree*, *Naïve Bayes*, *Discriminant Analysis*, *K-Nearest Neighbors* and *Support Vector Machines (SVM)*. The validation of train matrices is based on *Cross-Validation 10 fold*.

To avoid excess of data only eight comparative models out of 32 possible ones are discussed in machine learning section. Models with lowest and highest dimensions **M** and **MRKS** and only for chunks of 0.05s which means 200 data points are selected. The full list of all 32 models with their all machine learning results are tabulated in ***Appendix C***.

At last the subject of feature selection is introduced for sake of simplicity, and avoiding computational costs. For this aim three different methods are explained which are *Filter Methods*, *Wrapper Methods* and *Embedded Methods*. Finally, two algorithms of filter methods are selected for feature selection process they are *Chi-Square* and *MRMR*.

Having completed the feature selection models, they are trained and tested with the same procedures of machine learning to find out how feature selection affects the results.

To avoid excess of data only four models out of eight possible ones are discussed in feature selection section. **MRKS** models and only for chunks of 0.05s which means 200 data points are selected. The full list of all eight models with their all machine learning results are tabulated in ***Appendix D***.

This thesis shows it is possible to guess either the class of defect or the working condition even if only speed or load is known by means of the data provided by the client.

Appendix A

Models

1. Same Working Condition

M_W7_100
M_W7_200
MR_W7_100
MR_W7_200
MRK_W7_100
MRK_W7_200
MRKS_W7_100
MRKS_W7_200

2. Same Class of Defect

M_C17_100
M_ C17_200
MR_ C17_100
MR_ C17_200
MRK_ C17_100
MRK_ C17_200
MRKS_ C17_100
MRKS_ C17_200

3. Same Load

M_L21_100
M_ L21_200
MR_ L21_100
MR_ L21_200
MRK_ L21_100
MRK_ L21_200
MRKS_ L21_100
MRKS_ L21_200

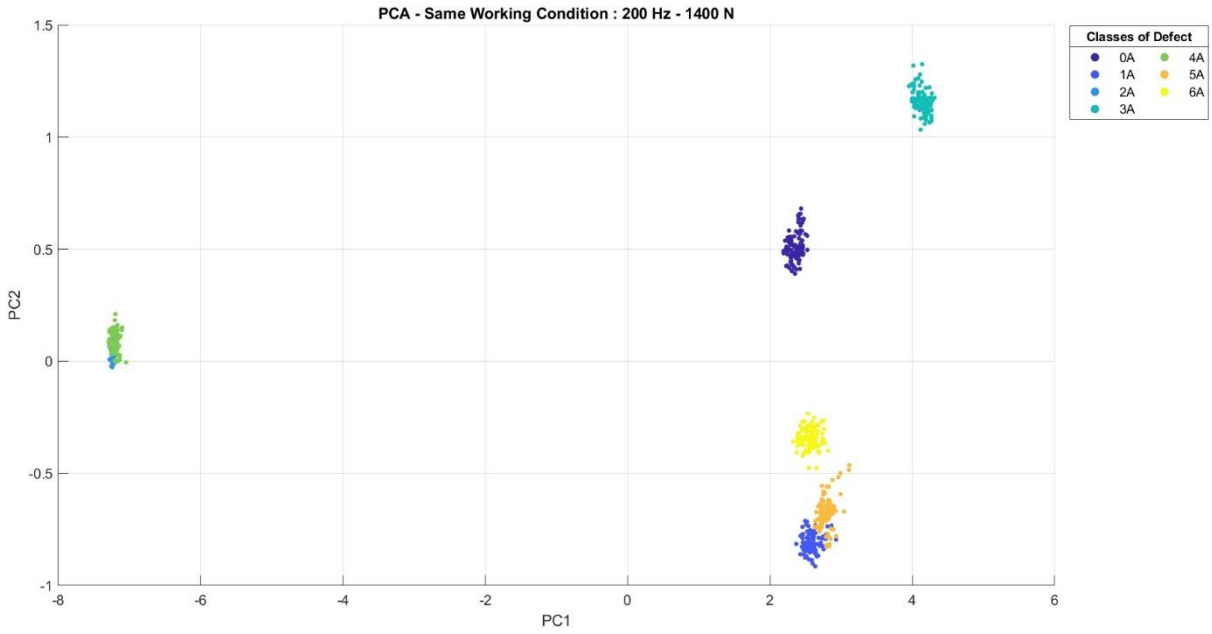
4. Same Speed

M_ S21_100
M_ S21_200
MR_ S21_100
MR_ S21_200
MRK_ S21_100
MRK_ S21_200
MRKS_ S21_100
MRKS_ S21_200

Appendix B

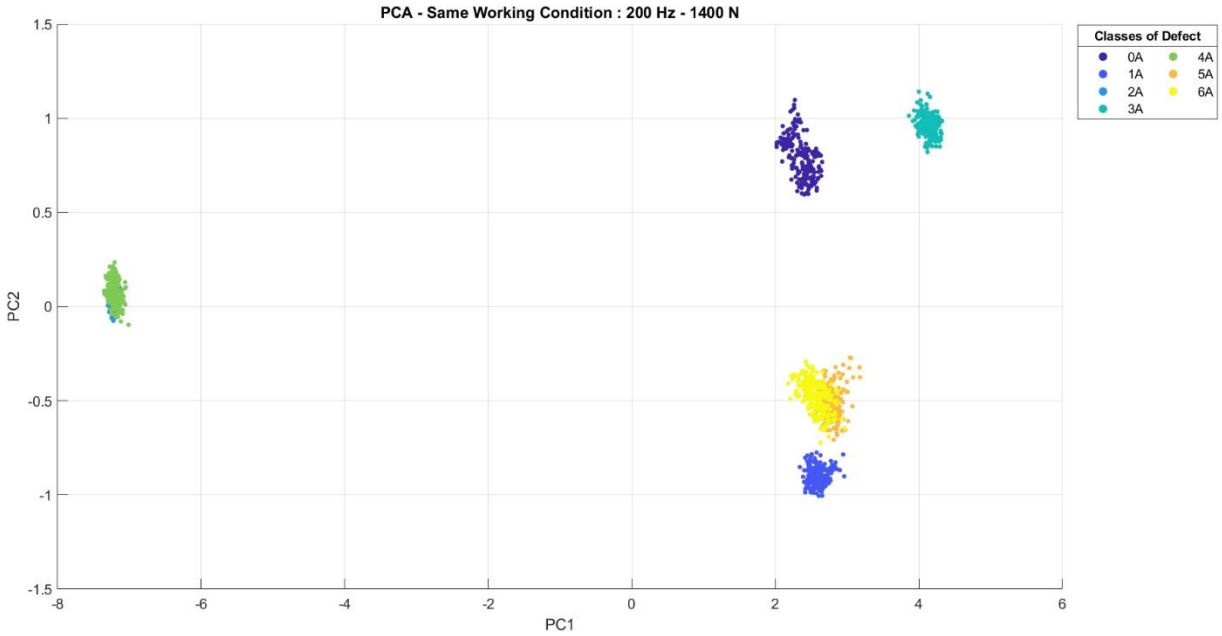
PCA Results

M_W7_100:



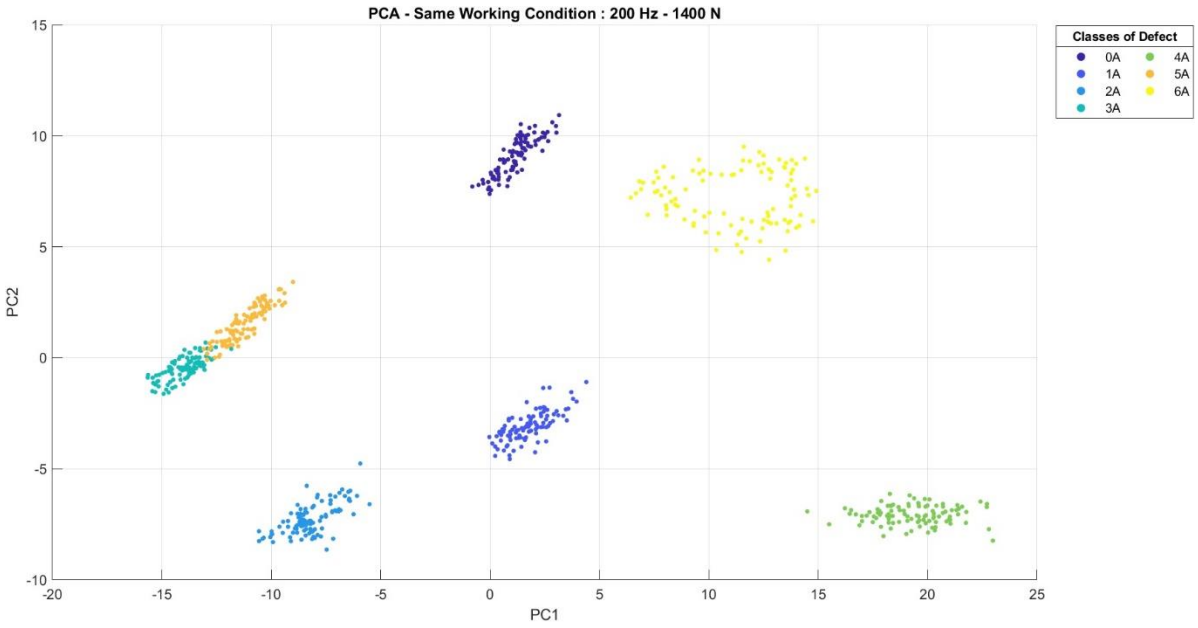
Same Working Condition - Mean - 100 Chunks

M_W7_200:



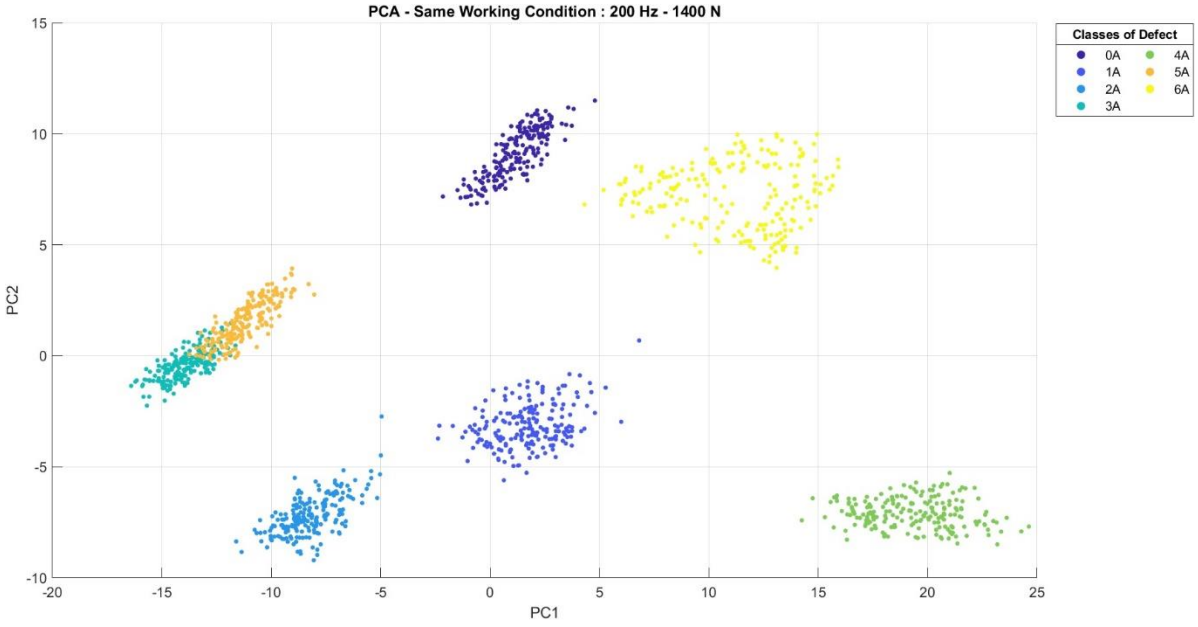
Same Working Condition - Mean - 200 Chunks

MR_W7_100:



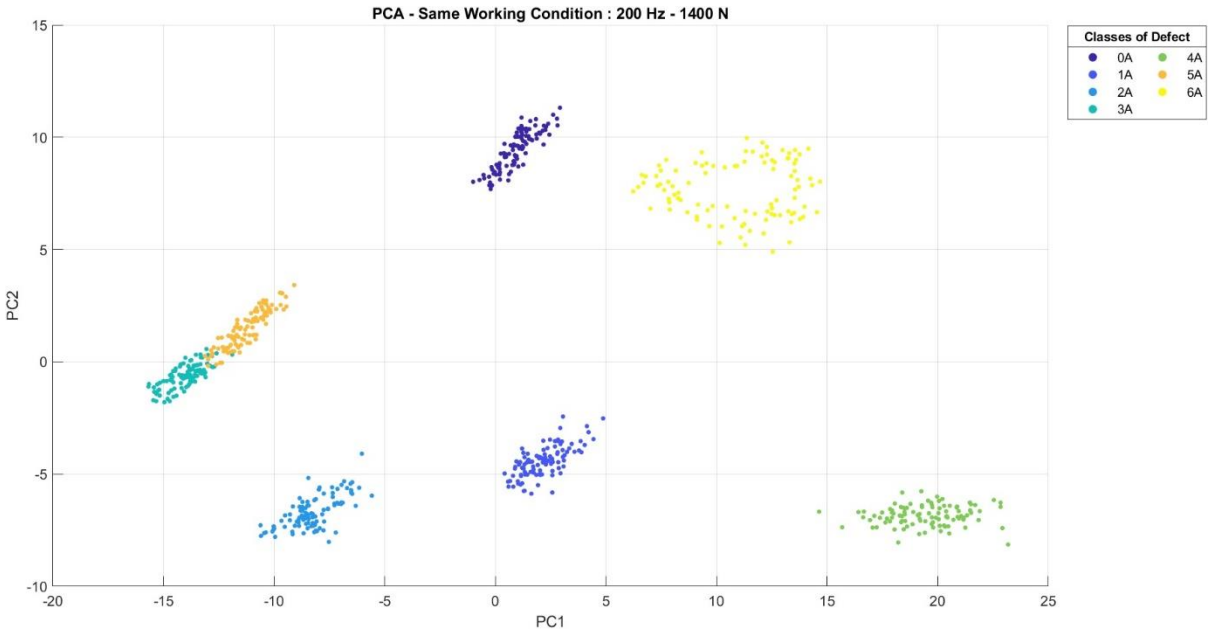
Same Working Condition - Mean, RMS - 100 Chunks

MR_W7_200:



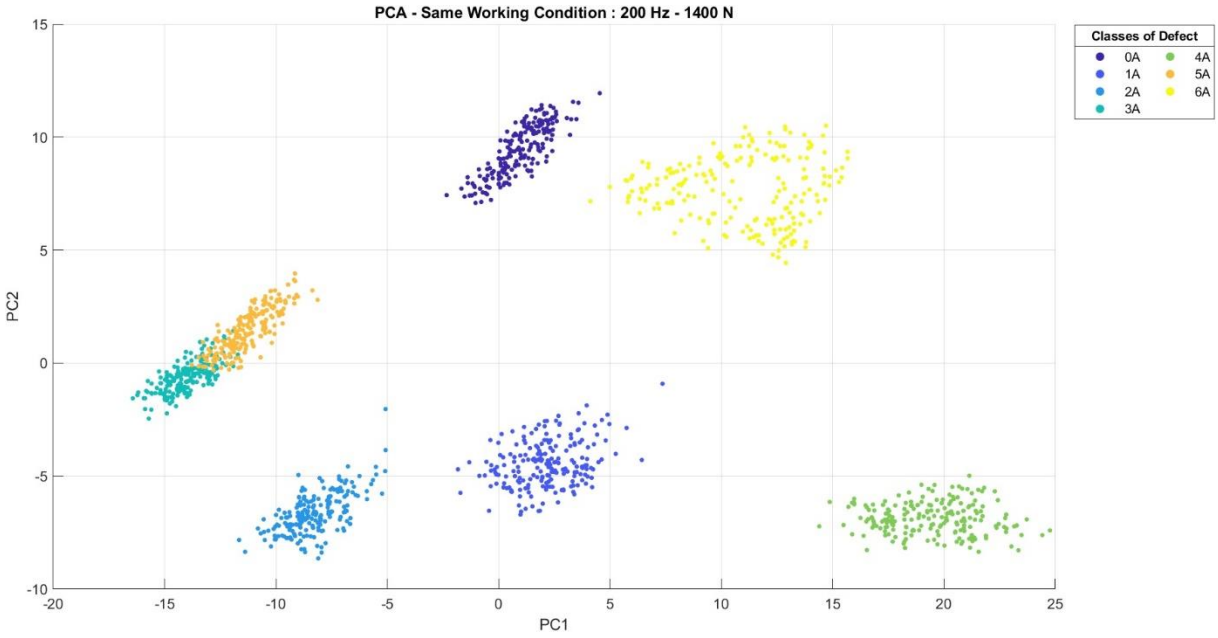
Same Working Condition - Mean, RMS - 200 Chunks

MRK_W7_100:



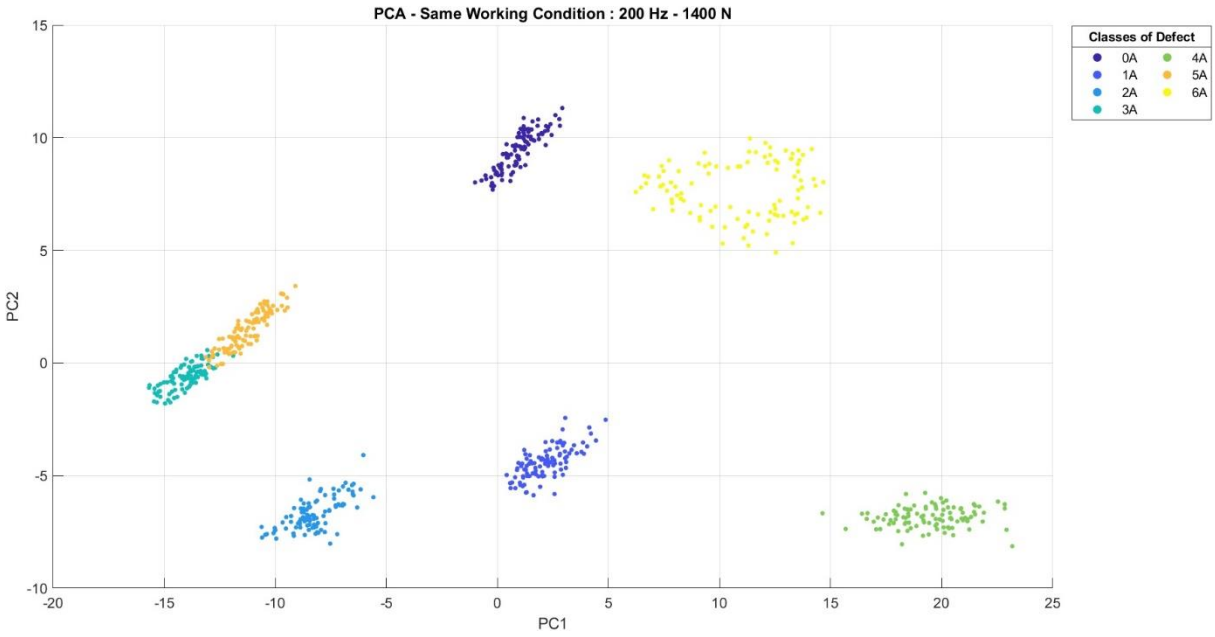
Same Working Condition - Mean, RMS, Kurtosis - 100 Chunks

MRK_W7_200:



Same Working Condition - Mean, RMS, Kurtosis - 200 Chunks

MRKS_W7_100:



Same Working Condition - Mean, RMS, Kurtosis, Skewness - 100 Chunks

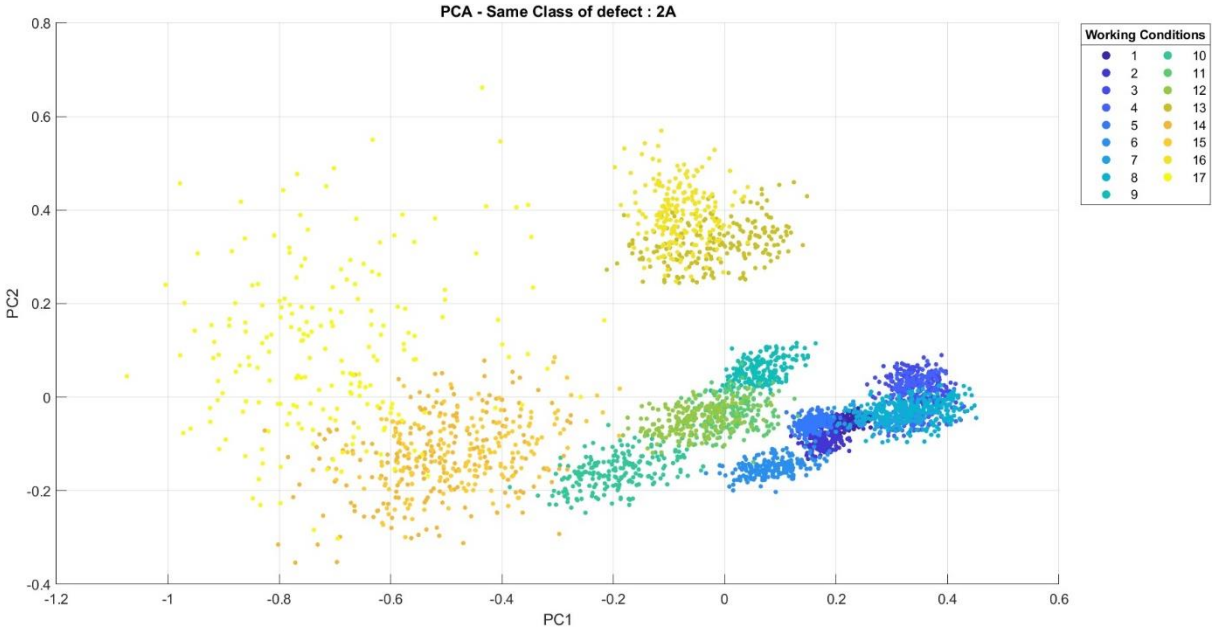
MRKS_W7_200:

Same Working Condition - Mean, RMS, Kurtosis, Skewness - 200 Chunks

M_C17_100:

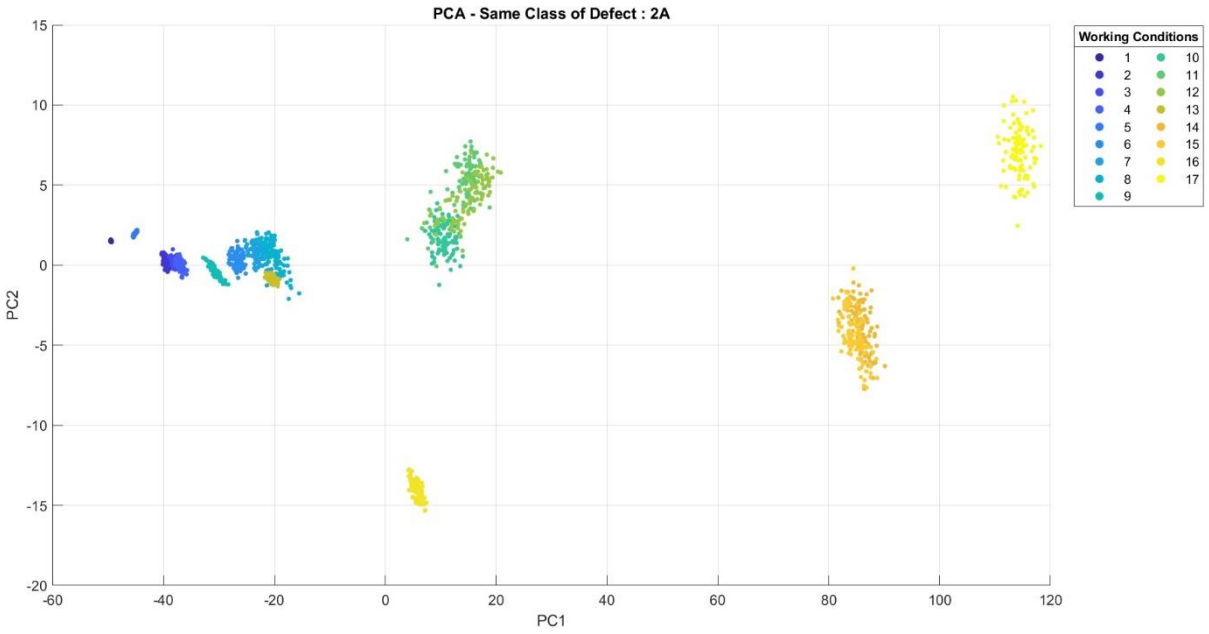
Same Class of Defect - Mean - 100 Chunks

M_C17_200:



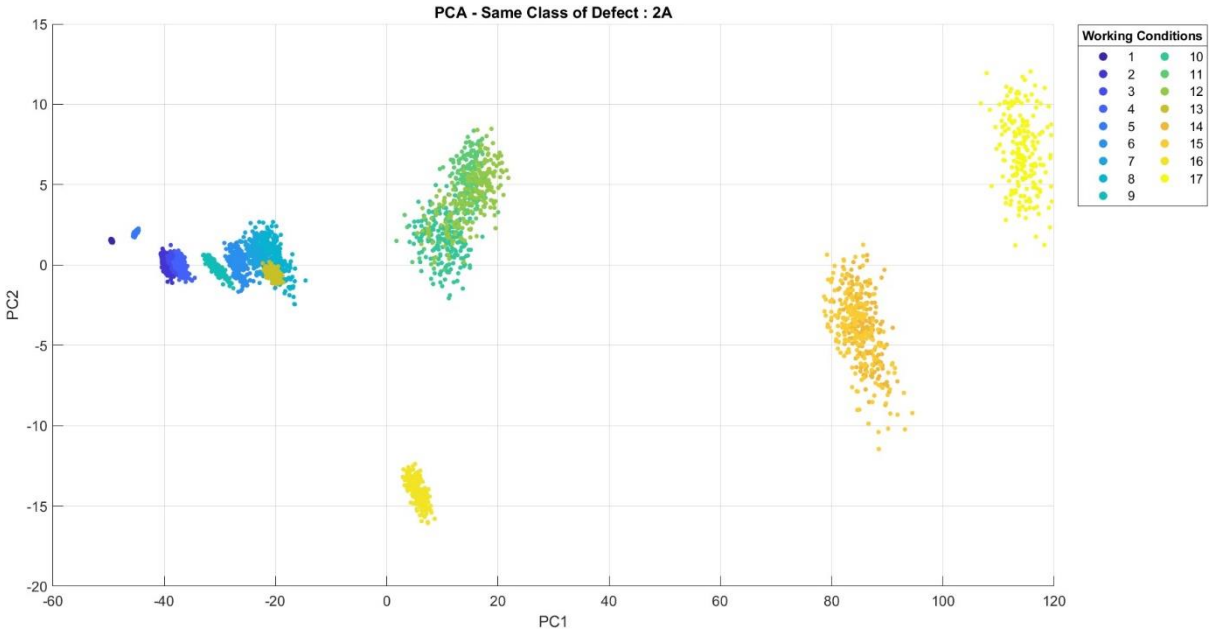
Same Class of Defect - Mean - 200 Chunks

MR_C17_100:



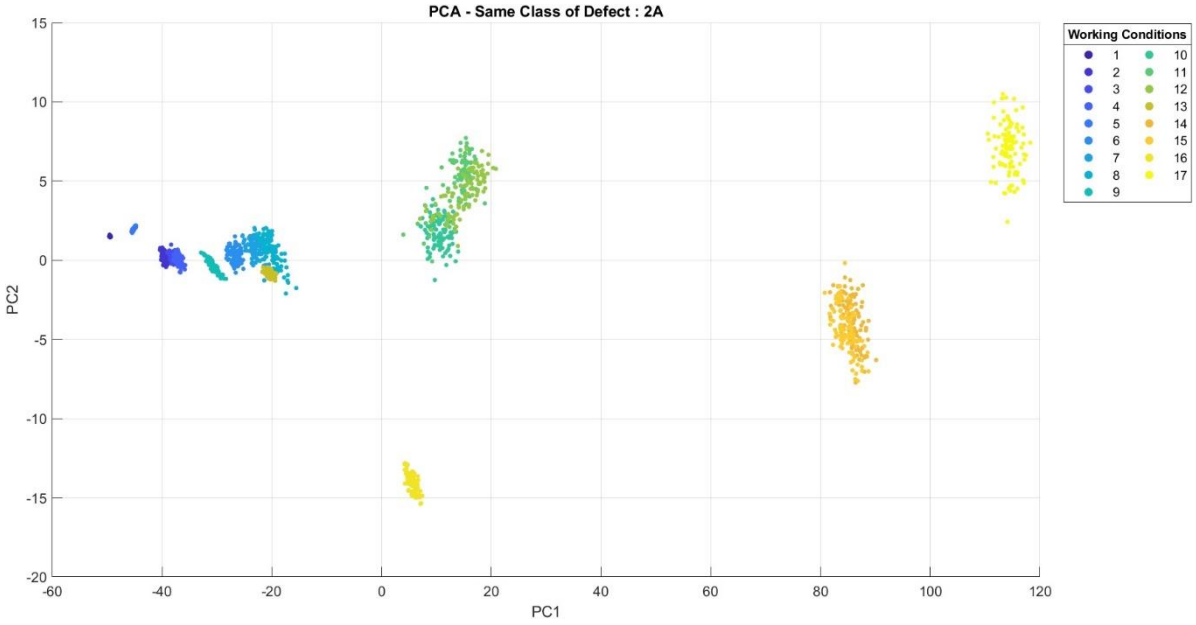
Same Class of Defect - Mean, RMS - 100 Chunks

MR_C17_200:



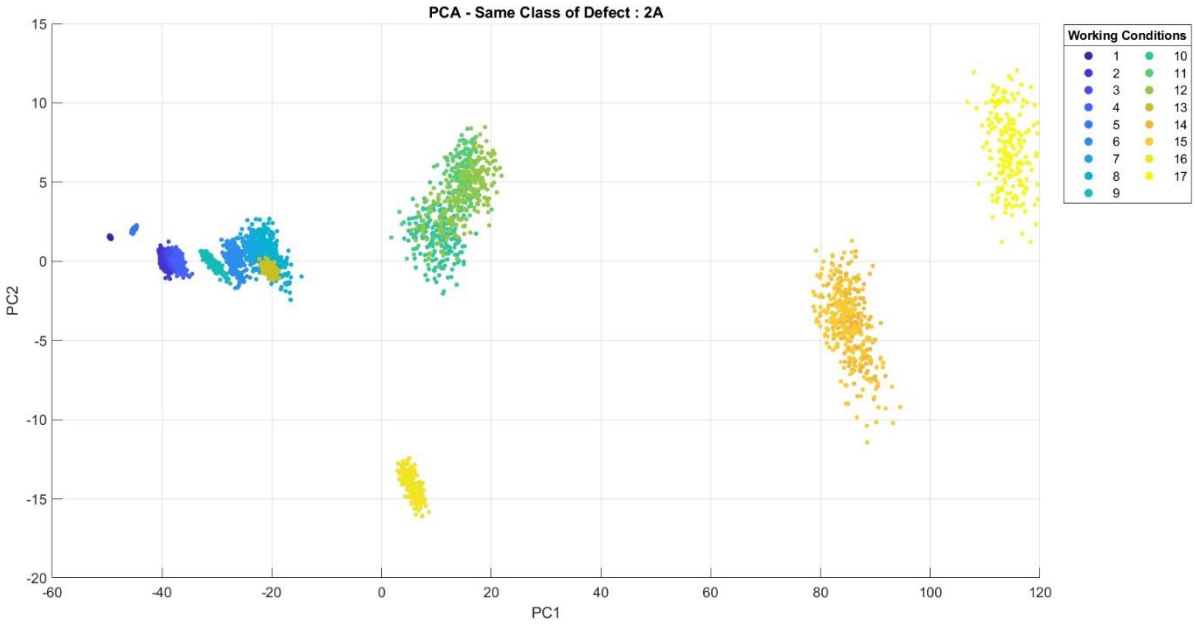
Same Class of Defect - Mean, RMS - 200 Chunks

MRK_C17_100:



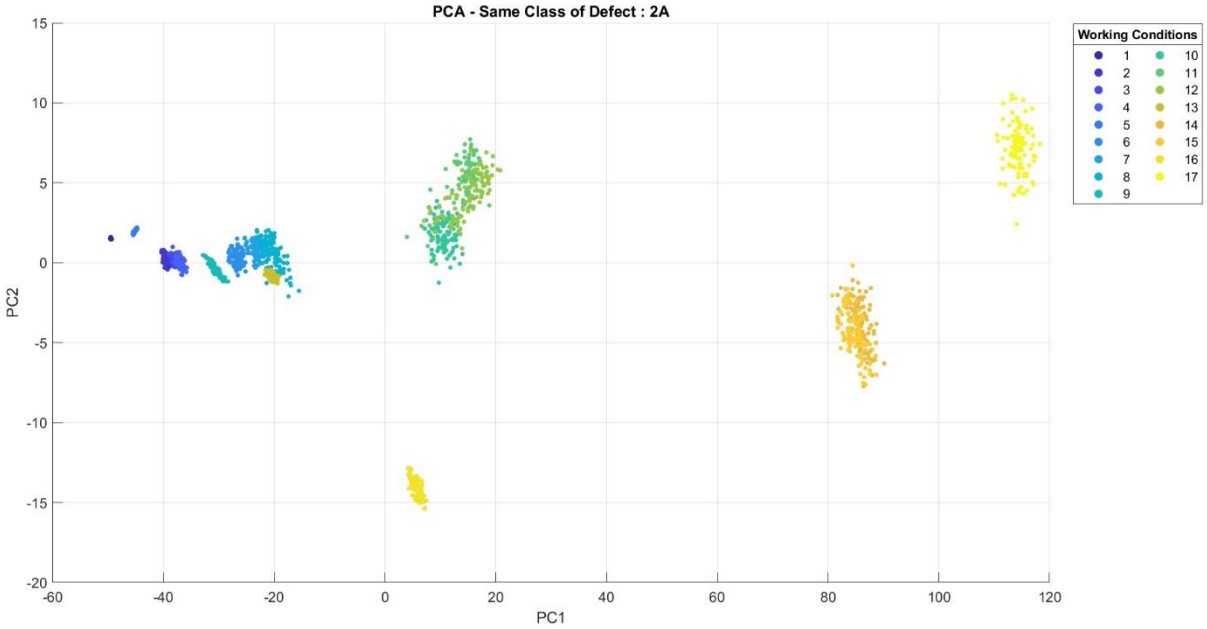
Same Class of Defect - Mean, RMS, Kurtosis - 100 Chunks

MRK_C17_200:



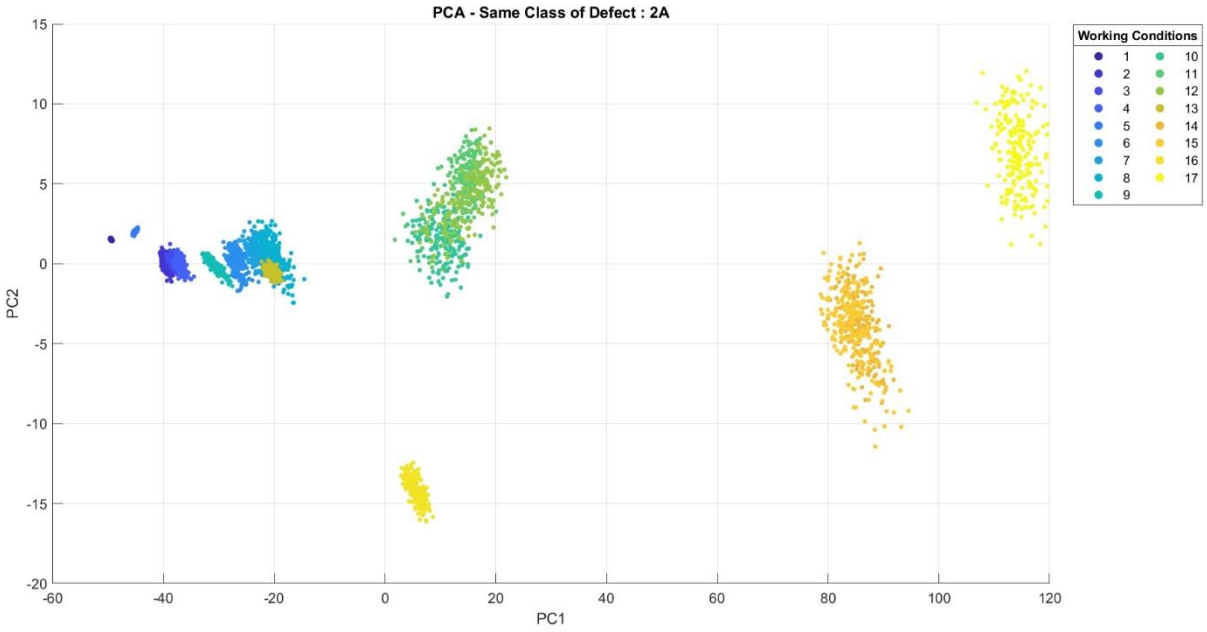
Same Class of Defect - Mean, RMS, Kurtosis - 200 Chunks

MRKS_C17_100:



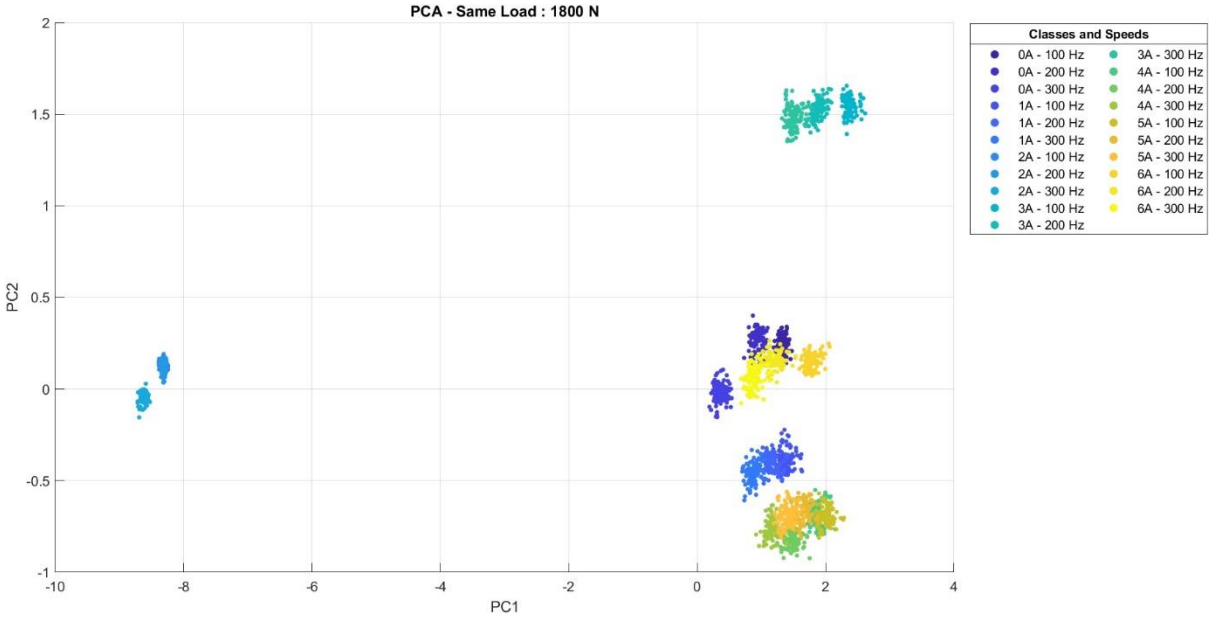
Same Class of Defect - Mean, RMS, Kurtosis, Skewness - 100 Chunks

MRKS_C17_200:



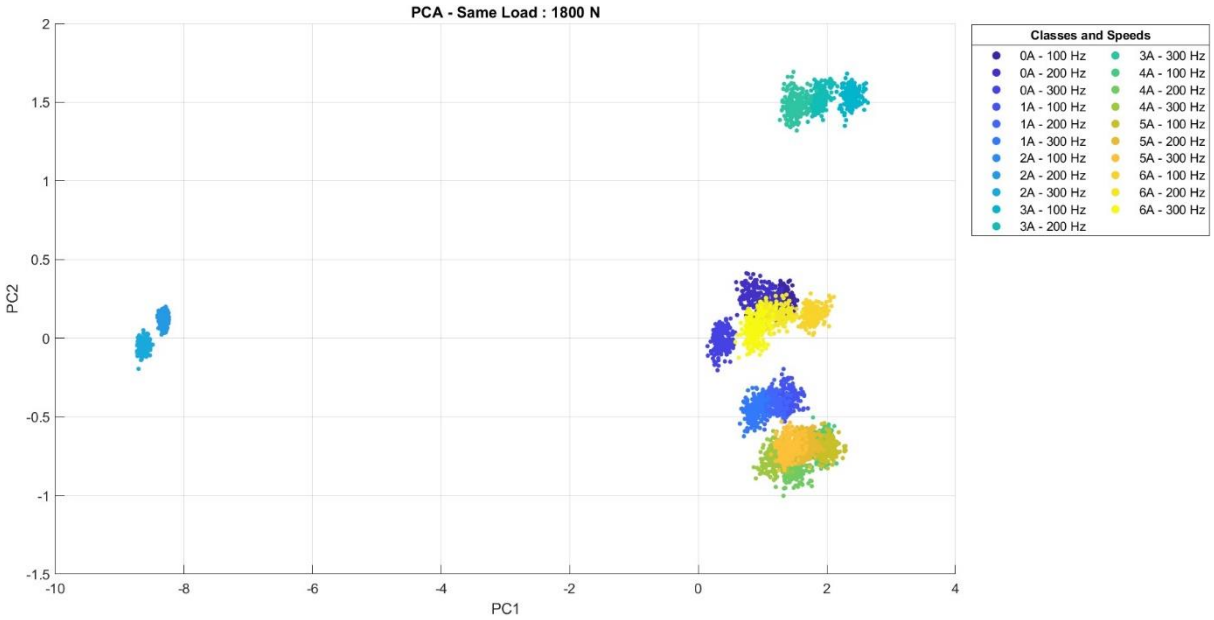
Same Class of Defect - Mean, RMS, Kurtosis, Skewness - 200 Chunks

M_L21_100:



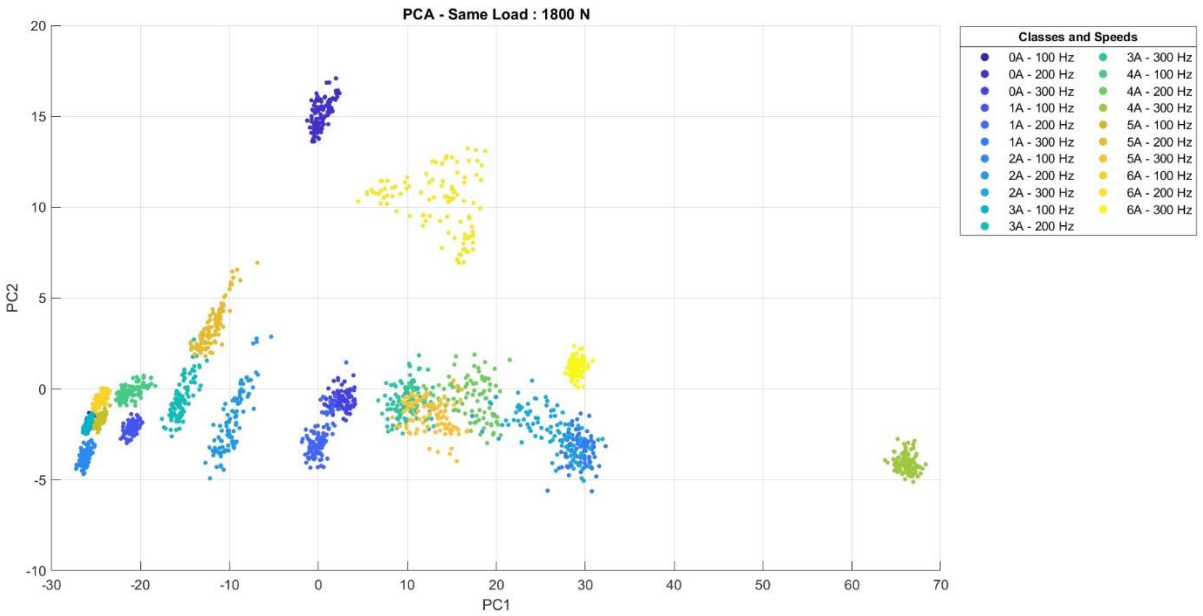
Same Load - Mean - 100 Chunks

M_L21_200:



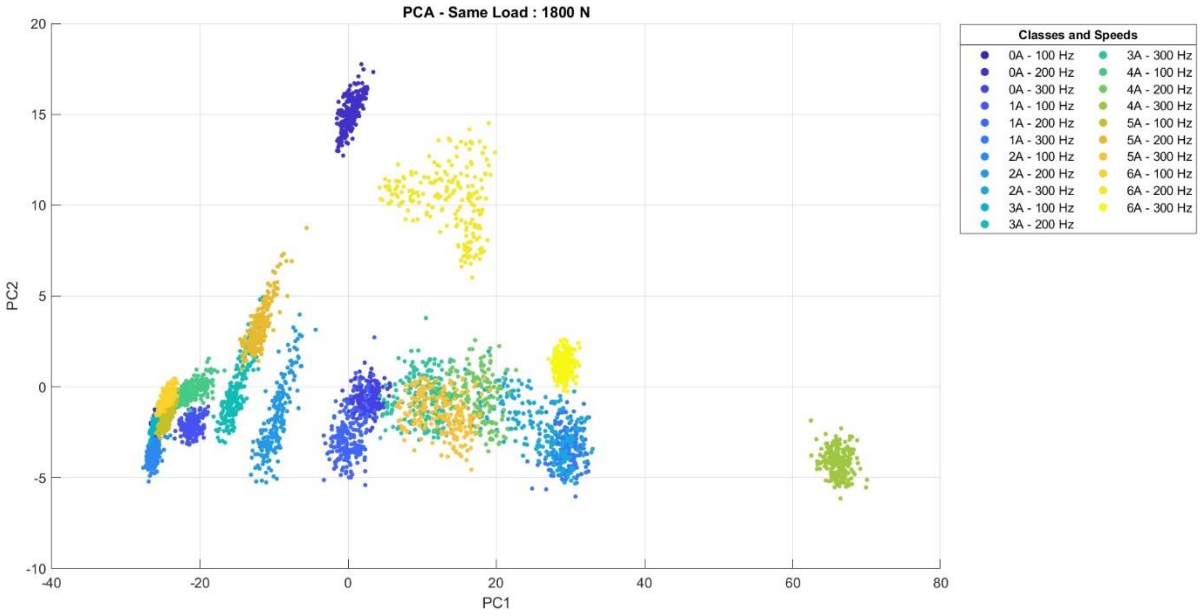
Same Load - Mean - 200 Chunks

MR_L21_100:



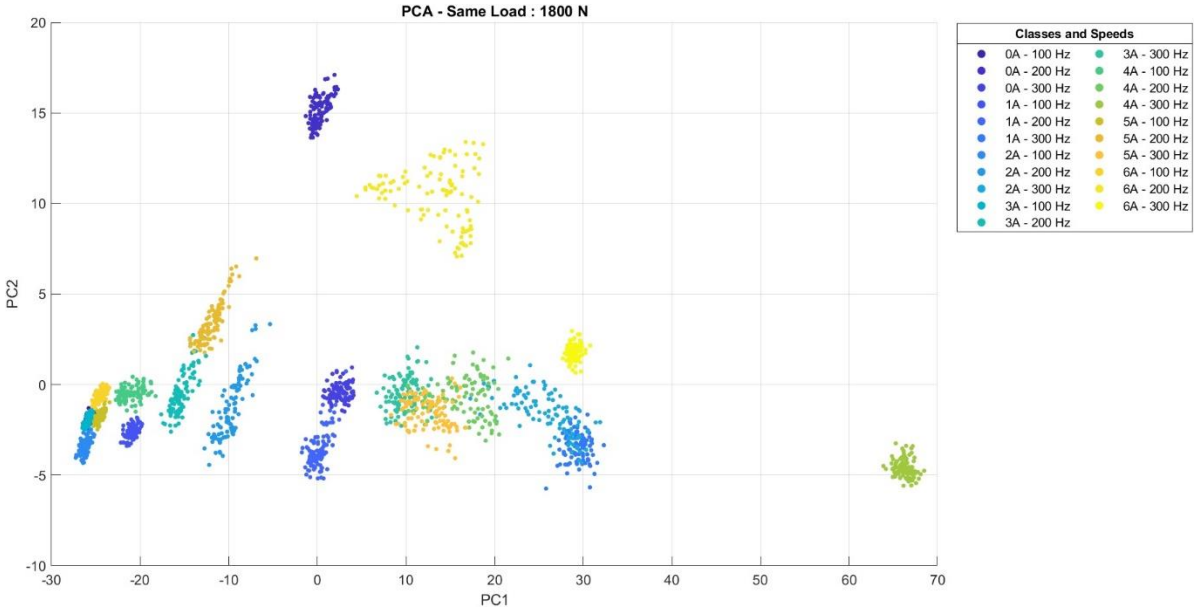
Same Load - Mean, RMS - 100 Chunks

MR_L21_200:



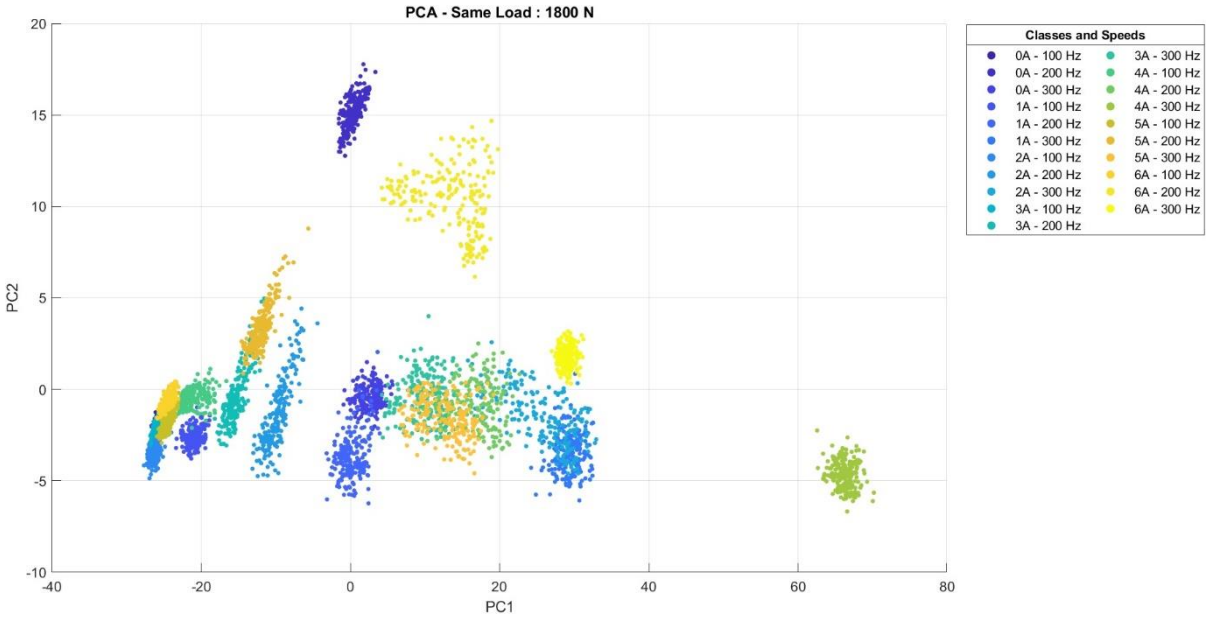
Same Load - Mean, RMS - 200 Chunks

MRK_L21_100:



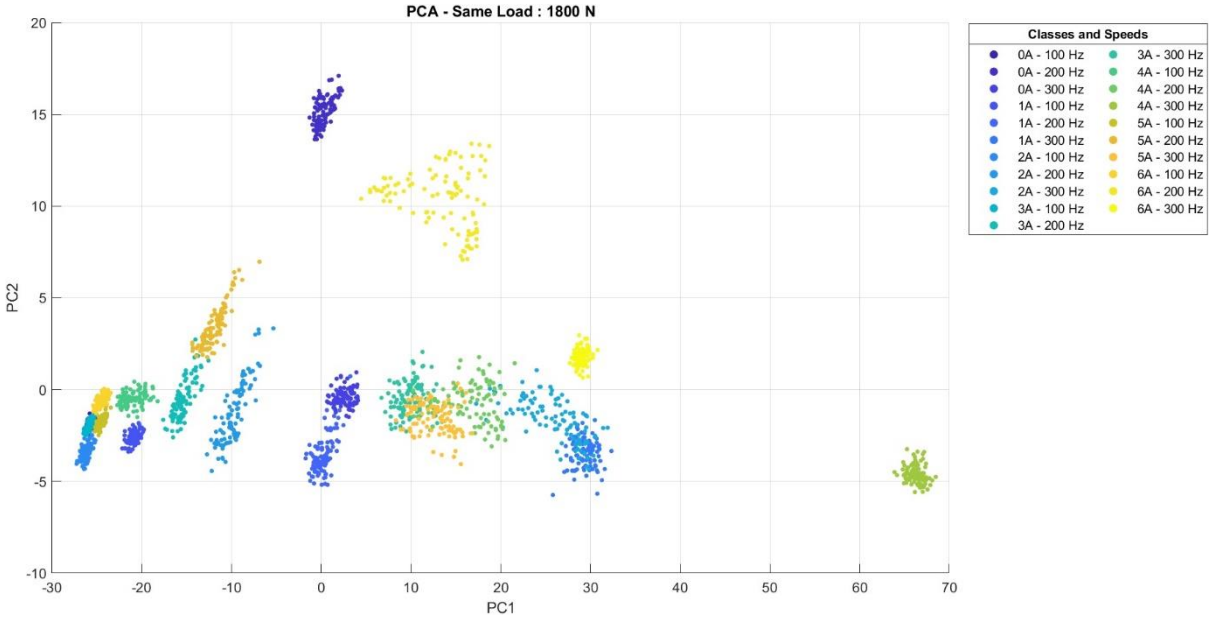
Same Load - Mean, RMS, Kurtosis - 100 Chunks

MRK_L21_200:



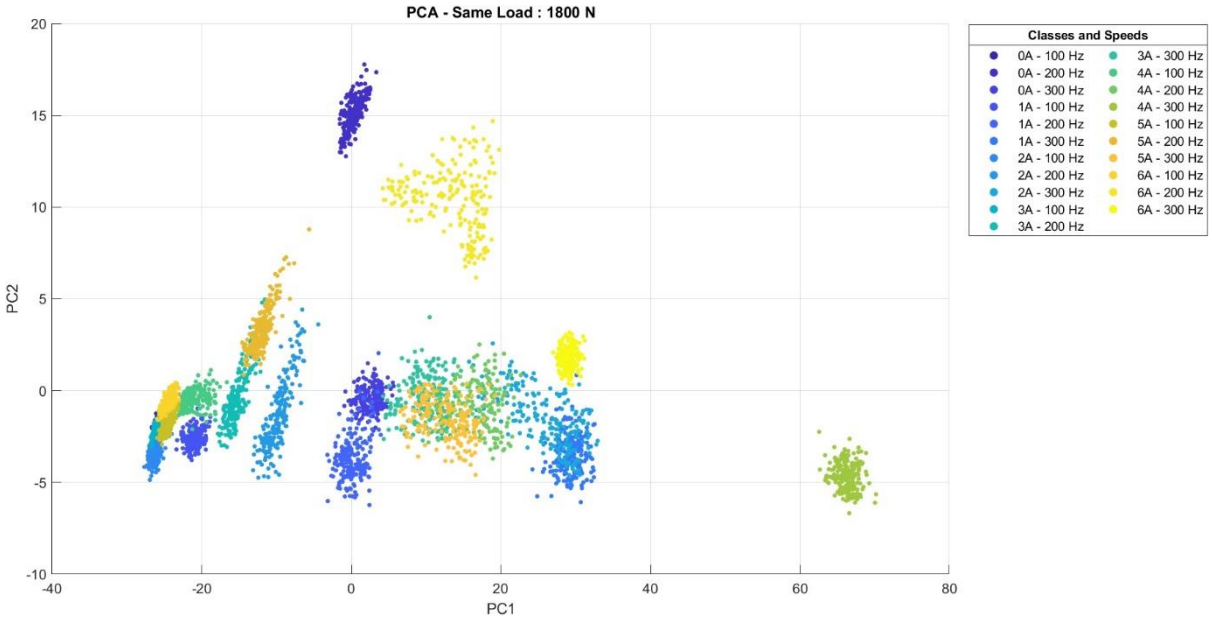
Same Load - Mean, RMS, Kurtosis - 200 Chunks

MRKS_L21_100:



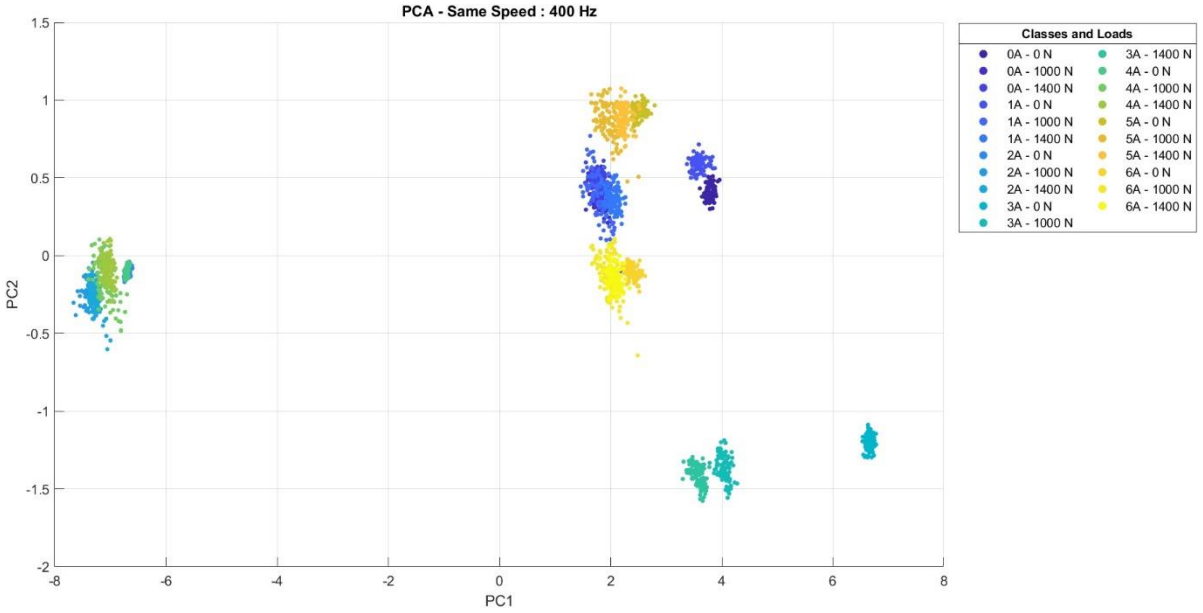
Same Load - Mean, RMS, Kurtosis, Skewness - 100 Chunks

MRKS_L21_200:



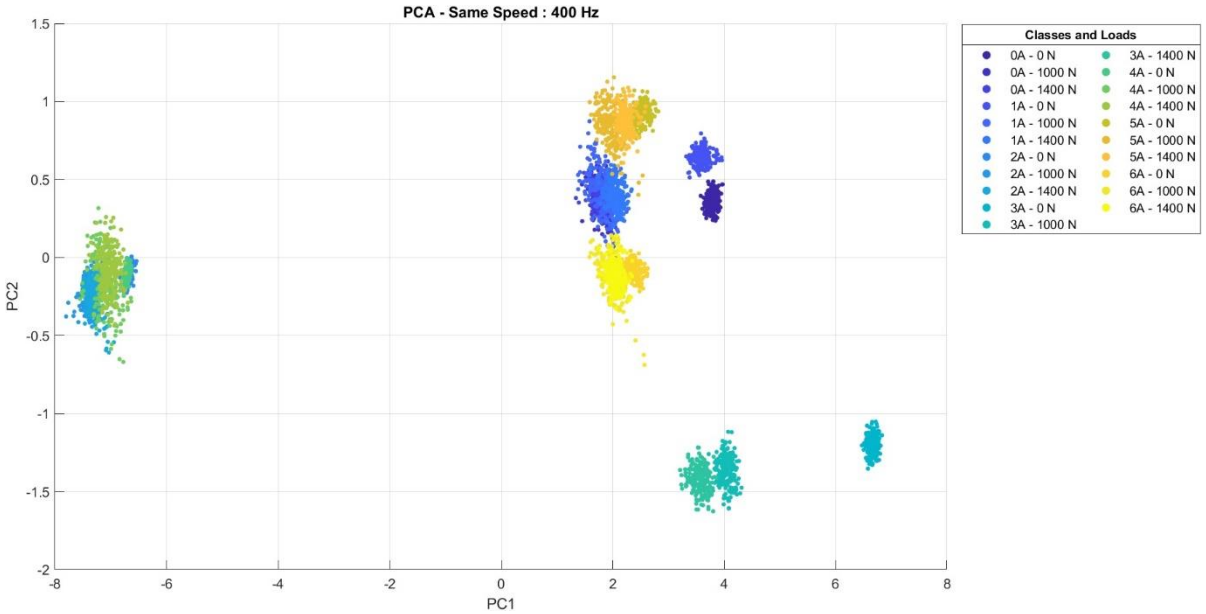
Same Load - Mean, RMS, Kurtosis, Skewness - 200 Chunks

M_S21_100:



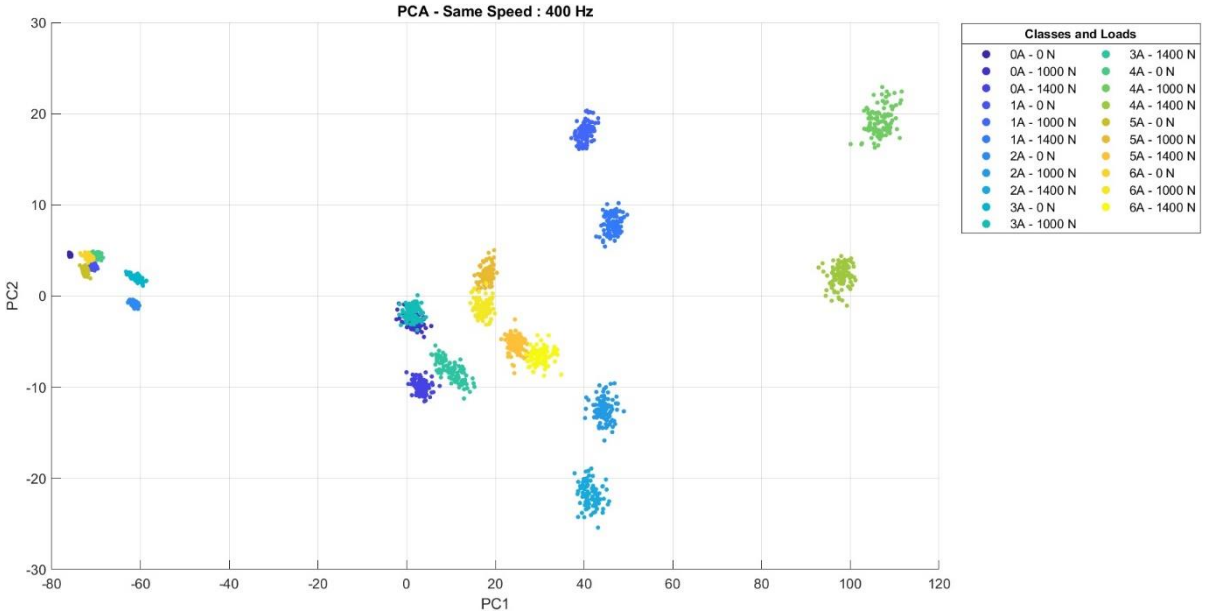
Same Speed - Mean - 100 Chunks

M_S21_200:



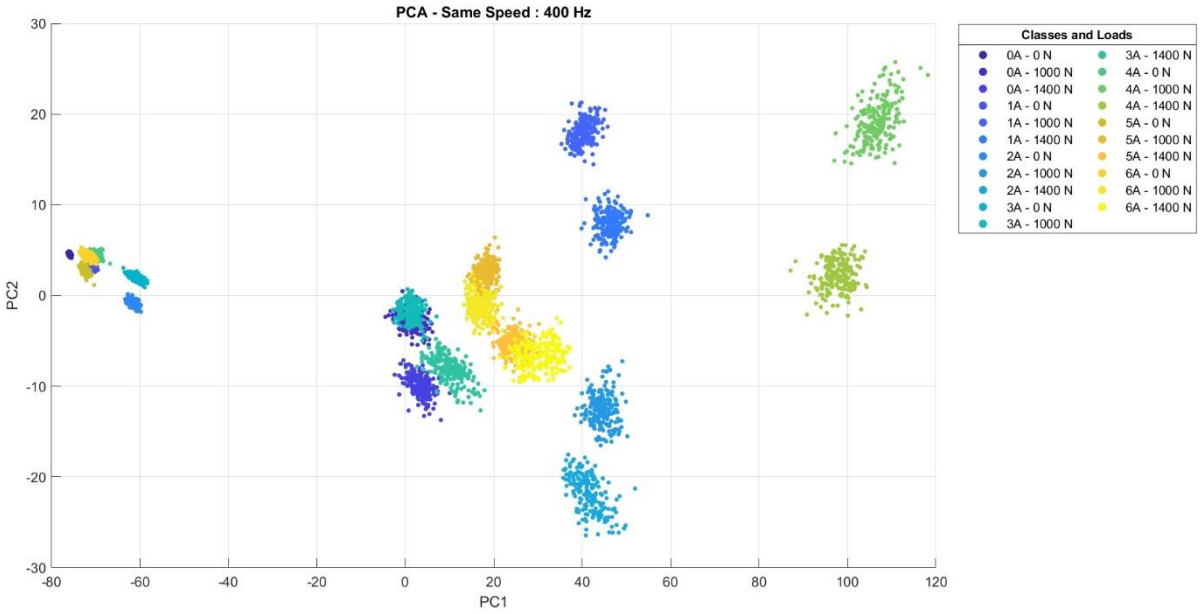
Same Speed - Mean - 200 Chunks

MR_S21_100:



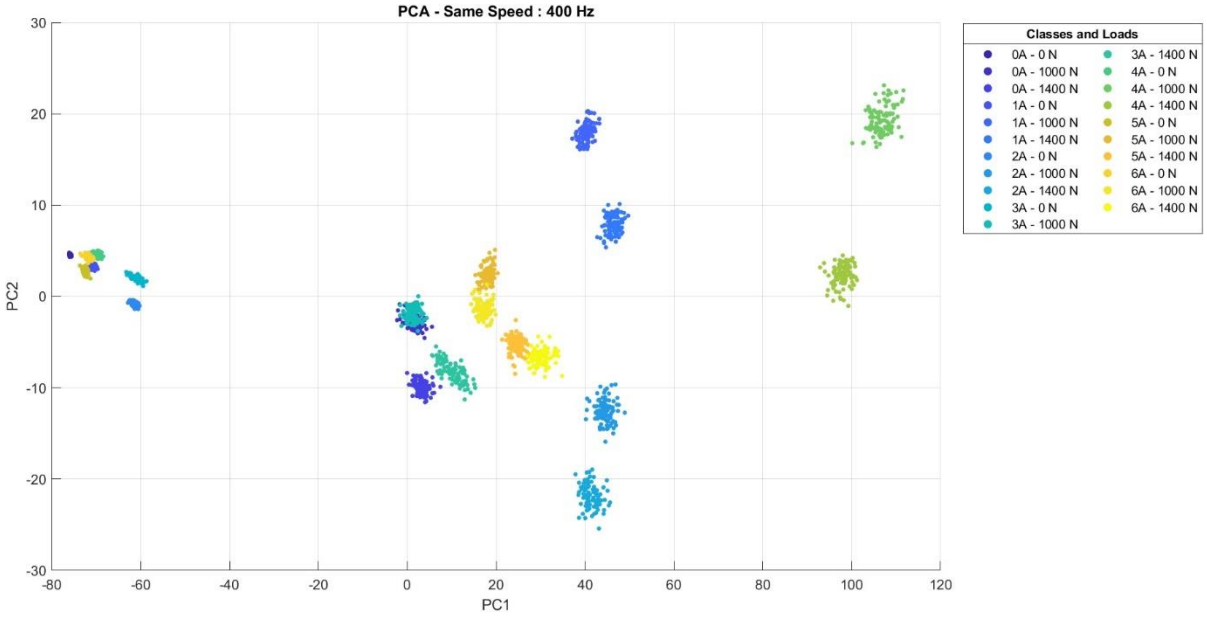
Same Speed - Mean, RMS - 100 Chunks

MR_S21_200:



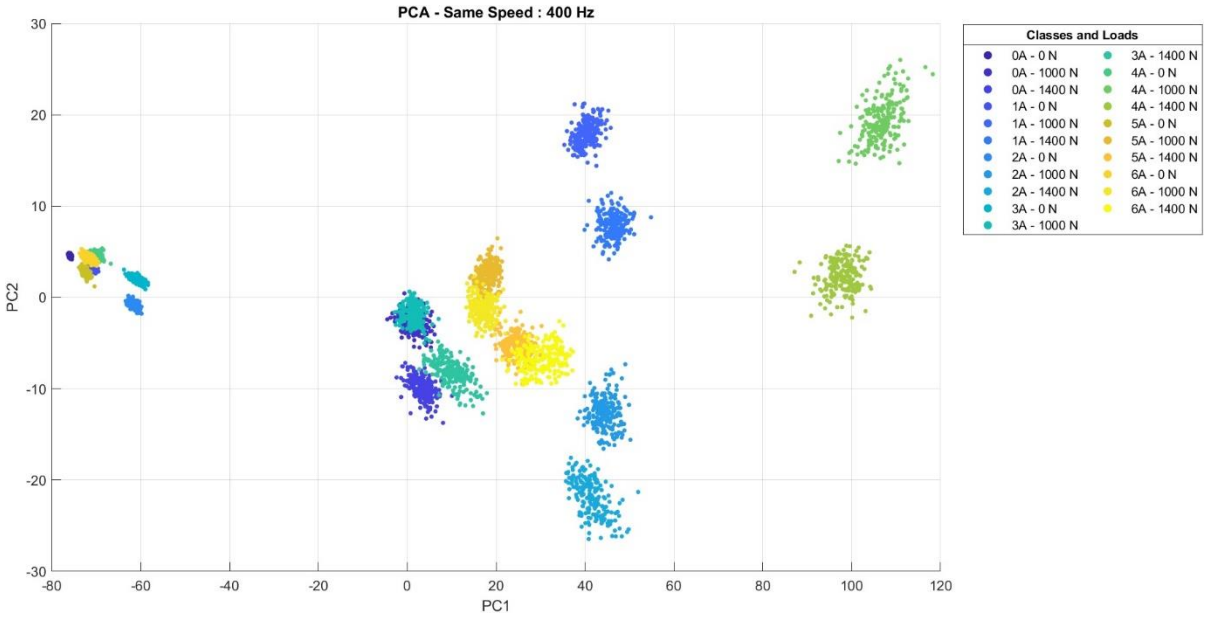
Same Speed - Mean, RMS - 200 Chunks

MRK_S21_100:



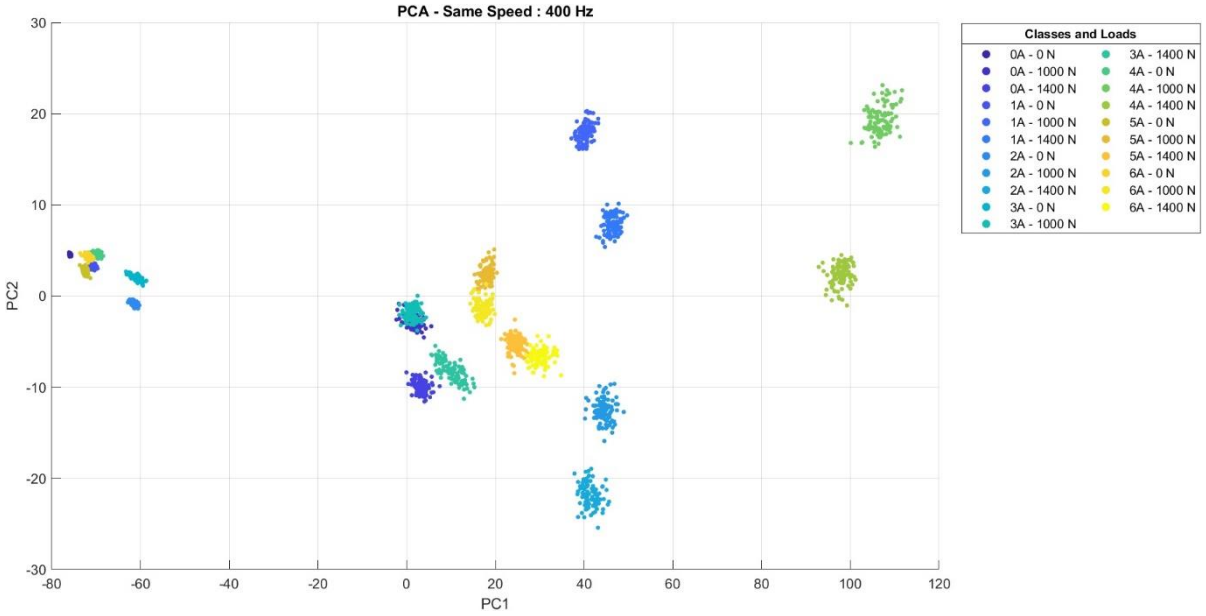
Same Speed - Mean, RMS, Kurtosis - 100 Chunks

MRK_S21_200:



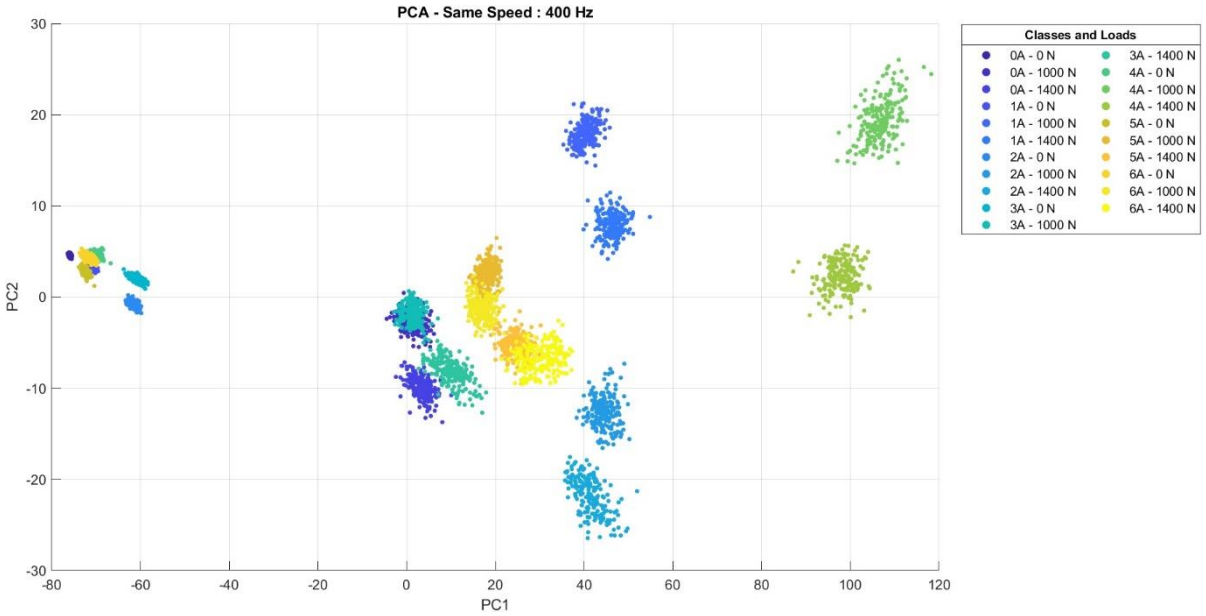
Same Speed - Mean, RMS, Kurtosis - 200 Chunks

MRKS_S21_100:



Same Speed - Mean, RMS, Kurtosis, Skewness - 100 Chunks

MRKS_S21_200:



Same Speed - Mean, RMS, Kurtosis, Skewness - 200 Chunks

Appendix C

Models with ***lowest*** and ***highest*** dimensions (**M** & **MRKS**) Confusion Matrices – 200 chunks

M_W7_200

Coarse Tree (*Train Accuracy: 71.3%*)

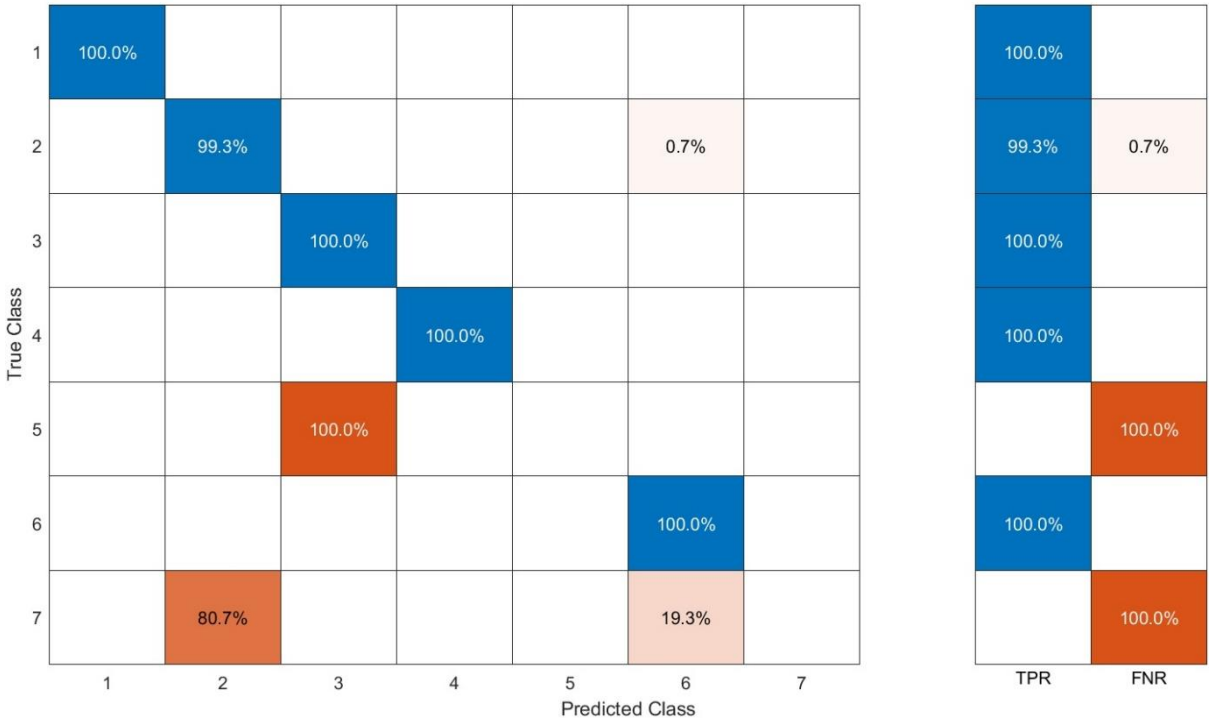


Figure 1 – a) Coarse tree confusion matrix

Coarse KNN (*Train Accuracy: 81.0%*)

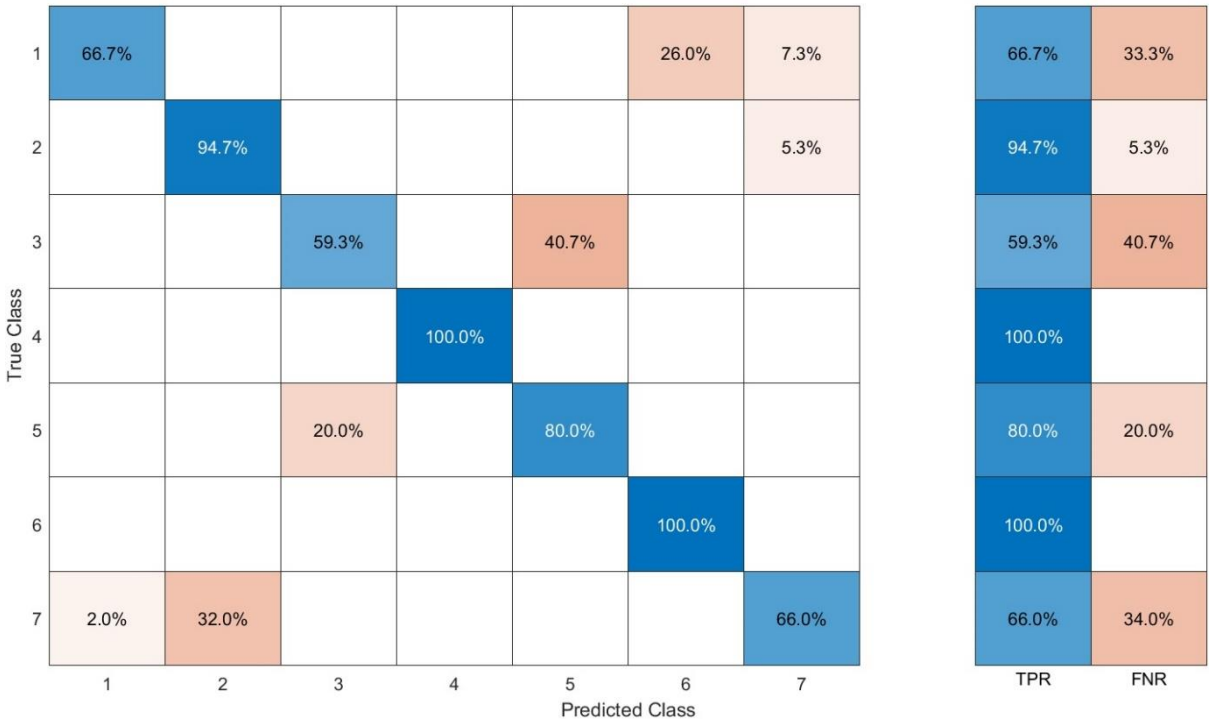


Figure 1 – b) Coarse KNN confusion matrix

M_W7_200

Quadratic Discriminant (*Train Accuracy: 98.4%*)

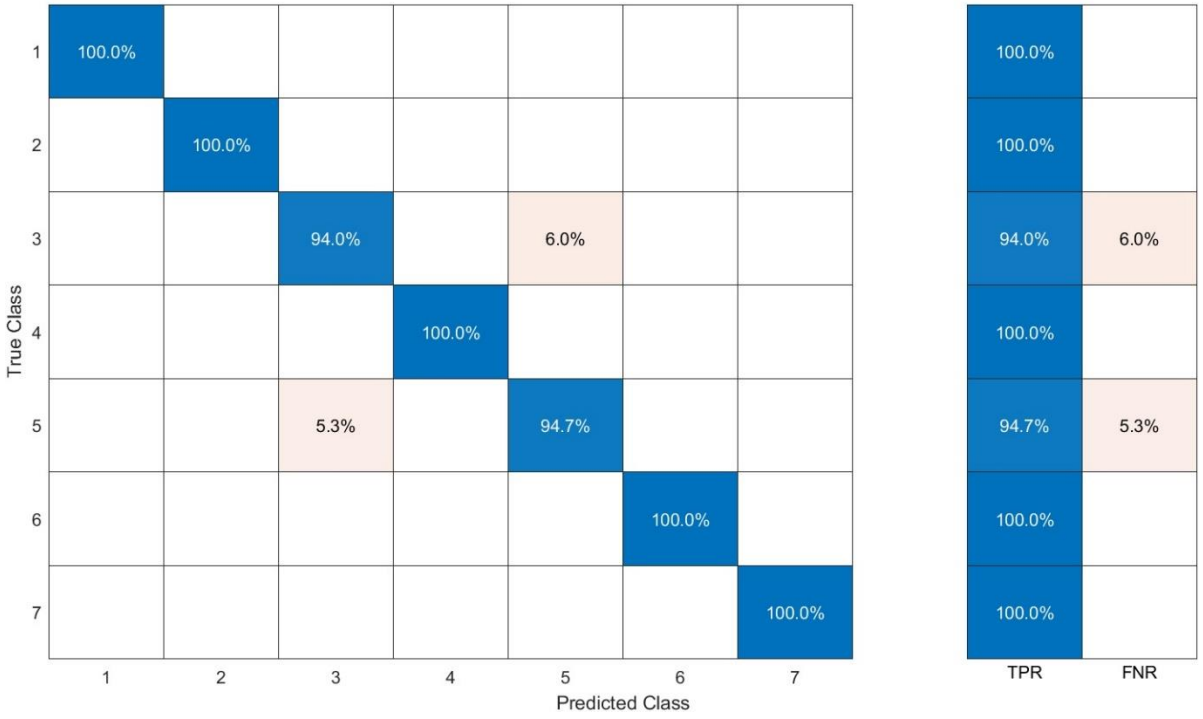


Figure 1 – c) Quadratic discriminant confusion matrix

Medium Gaussian SVM (*Train Accuracy: 99.0%*)

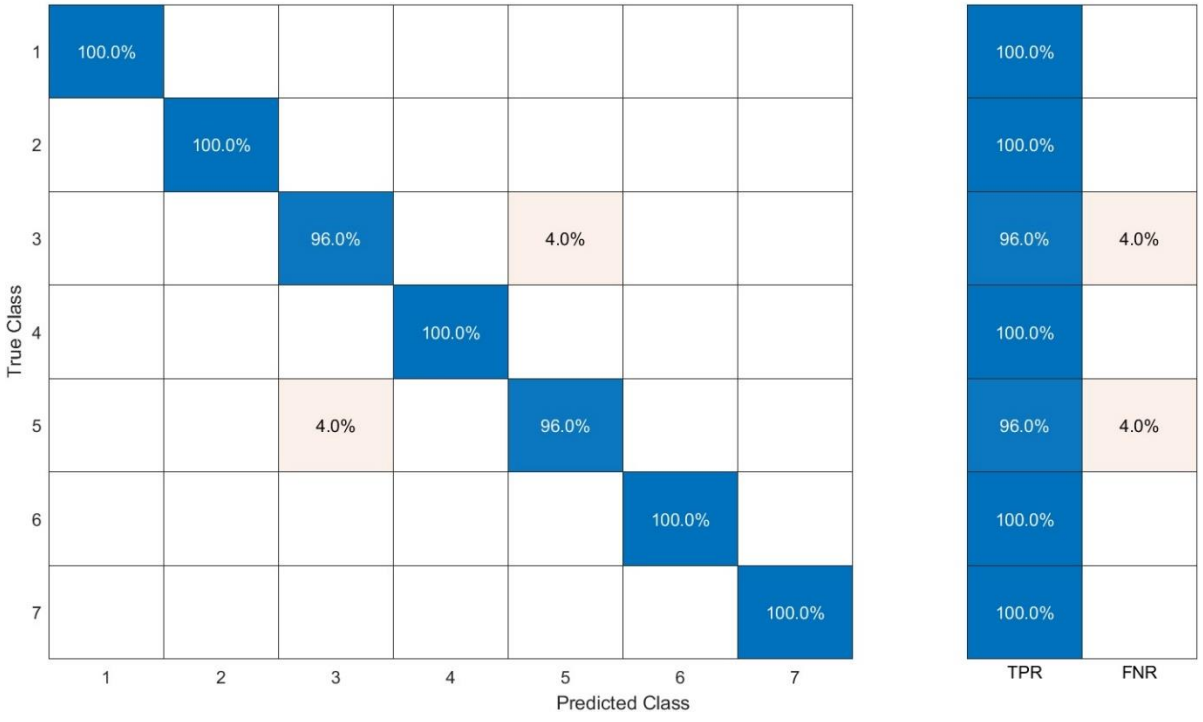


Figure 1 – d) Medium Gaussian SVM confusion matrix

MRKS_W7_200

Coarse Tree (*Train Accuracy: 71.4%*)

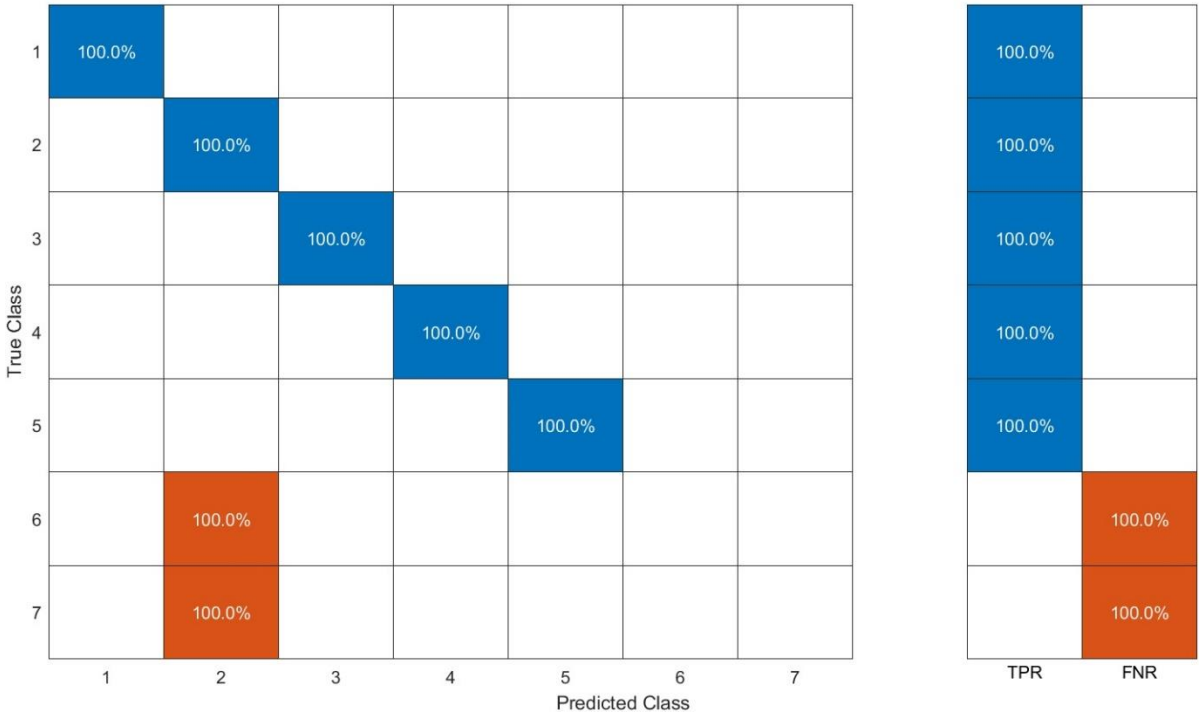


Figure 2 – a) Coarse tree confusion matrix

Coarse KNN (*Train Accuracy: 99.8%*)

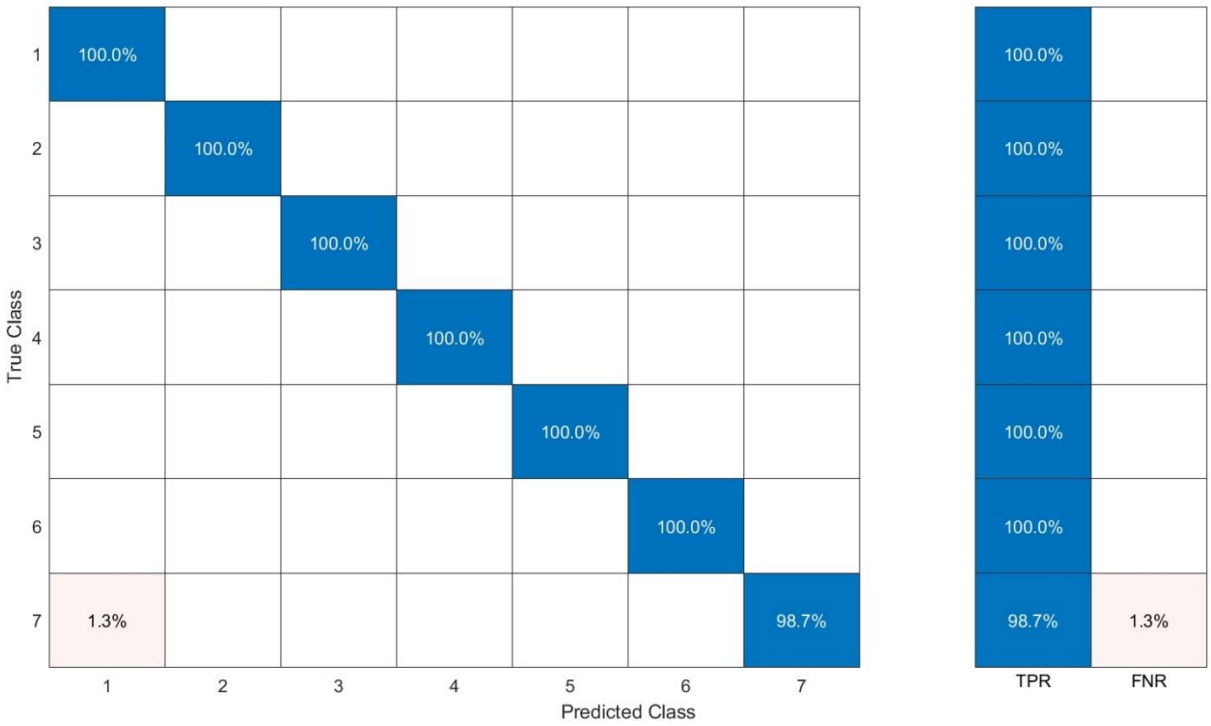


Figure 2 – b) Coarse KNN confusion matrix

MRKS_W7_200

Quadratic Discriminant (*Train Accuracy: 100.0%*)

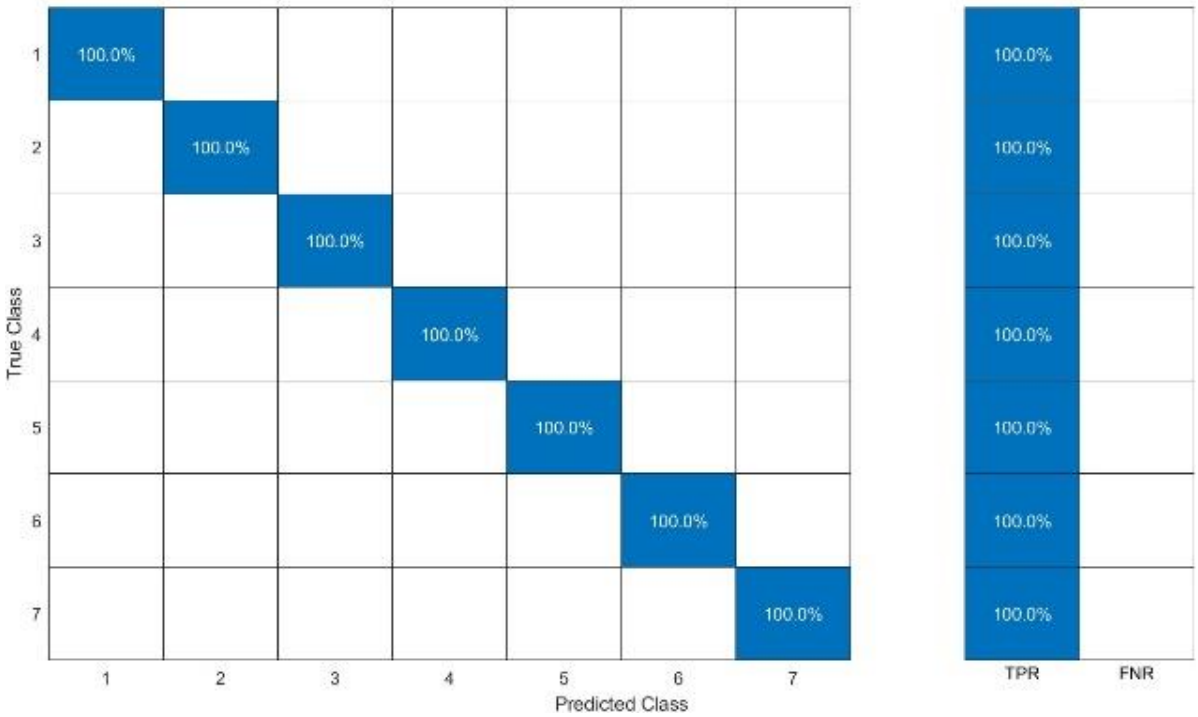


Figure 2 – c) Quadratic discriminant confusion matrix

Medium Gaussian SVM (*Train Accuracy: 100.0%*)

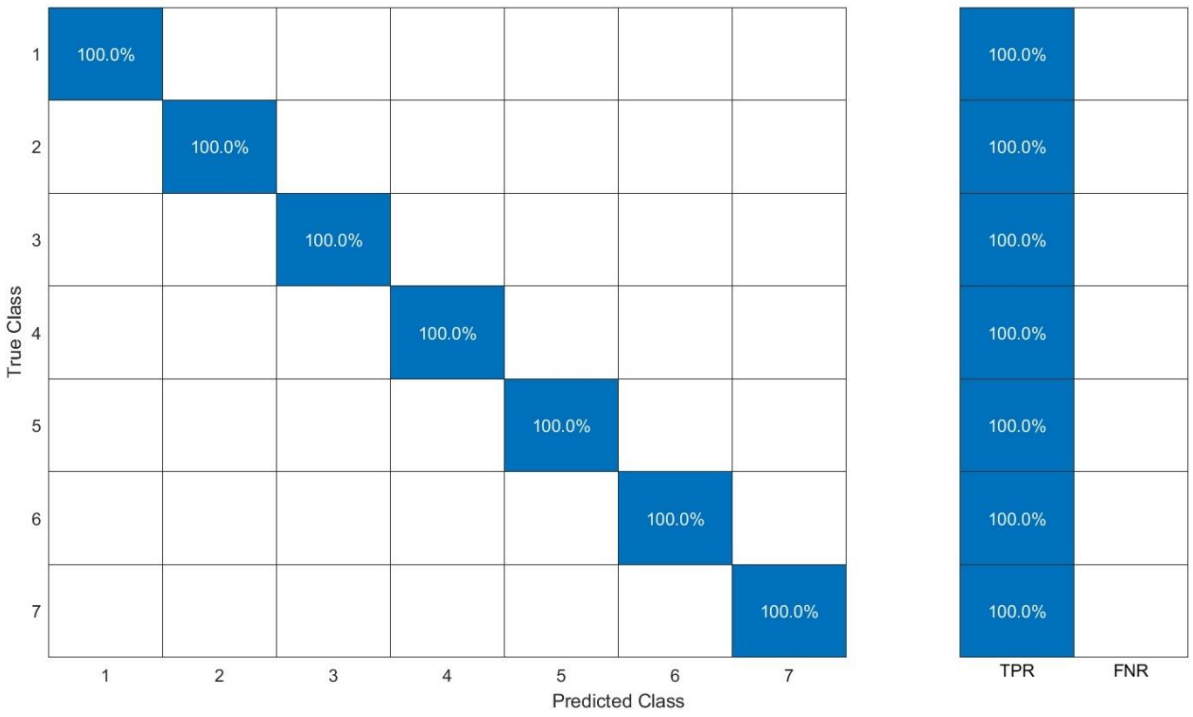


Figure 2 – d) Medium Gaussian SVM confusion matrix

M_C17_200

Coarse Tree (*Train Accuracy: 26.3%*)

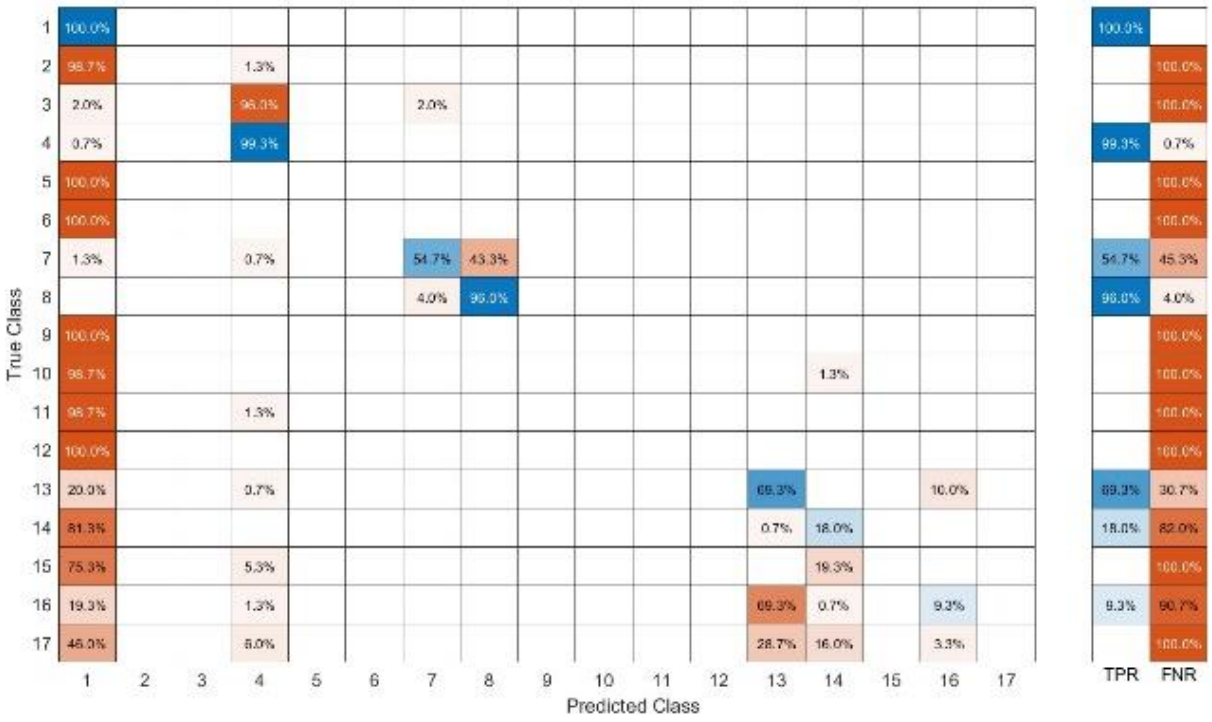


Figure 3 – a) Coarse tree confusion matrix

Coarse KNN (*Train Accuracy: 79.8%*)

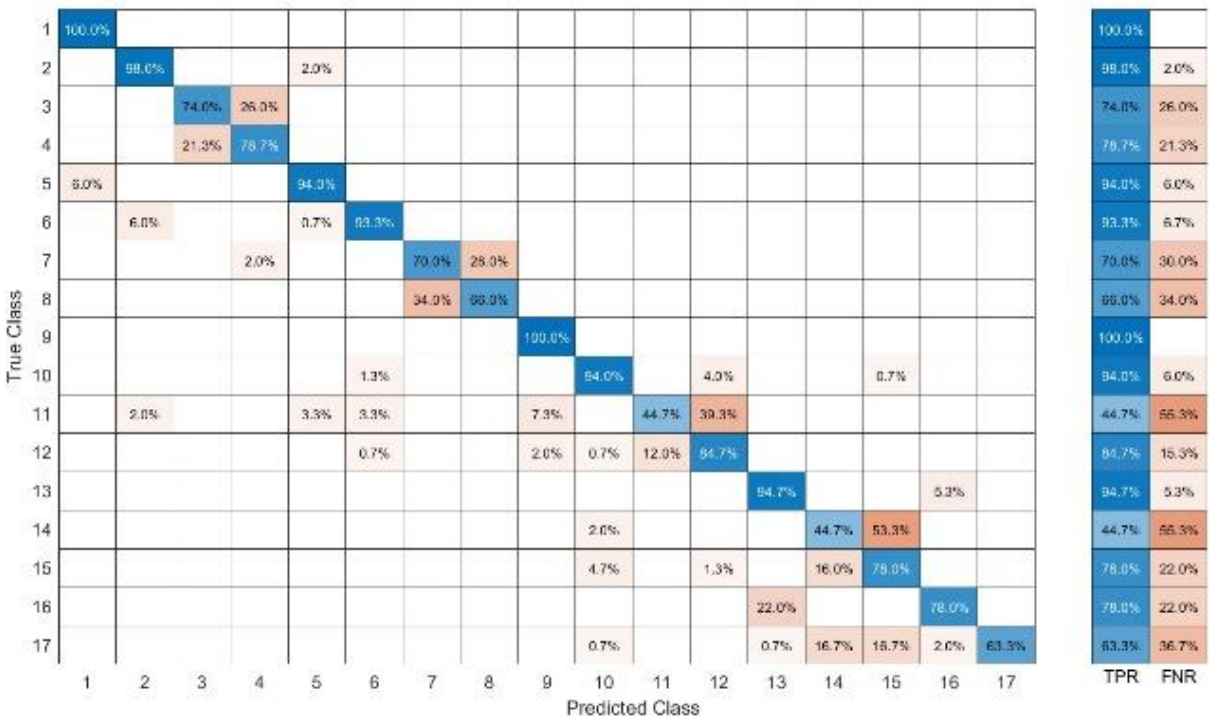


Figure 3 – b) Coarse KNN confusion matrix

M_C17_200

Quadratic Discriminant (*Train Accuracy: 88.3%*)

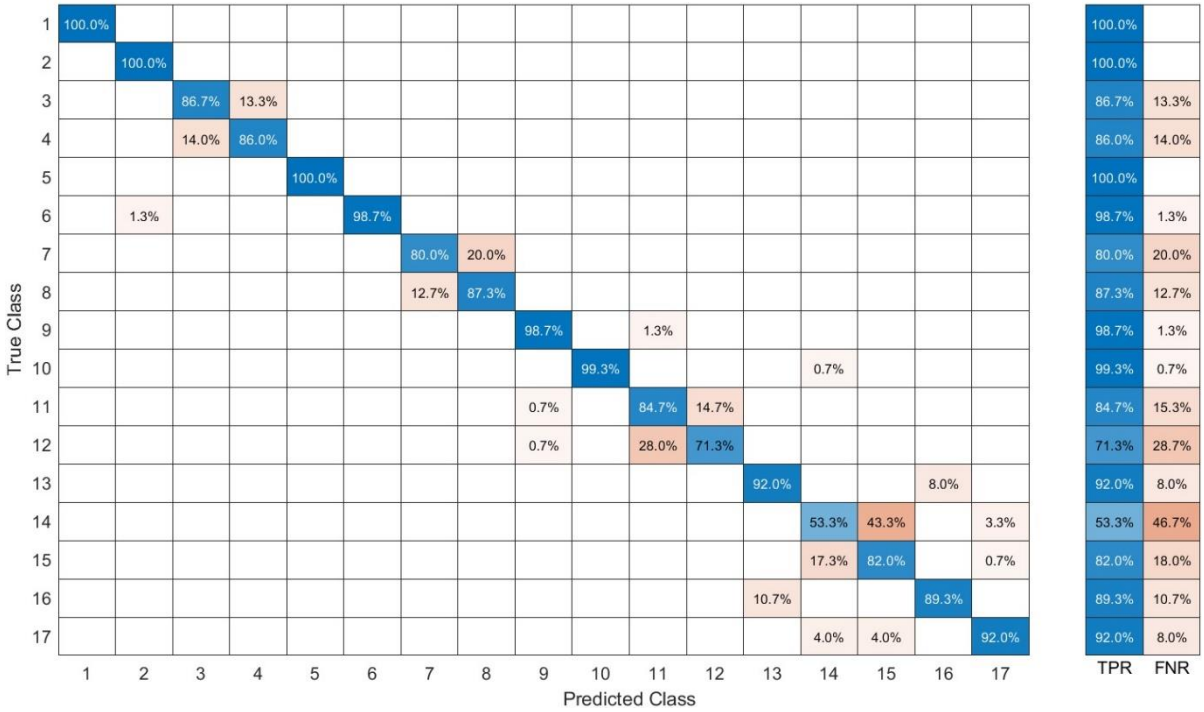


Figure 3 – c) Quadratic discriminant confusion matrix

Medium Gaussian SVM (*Train Accuracy: 88.5%*)

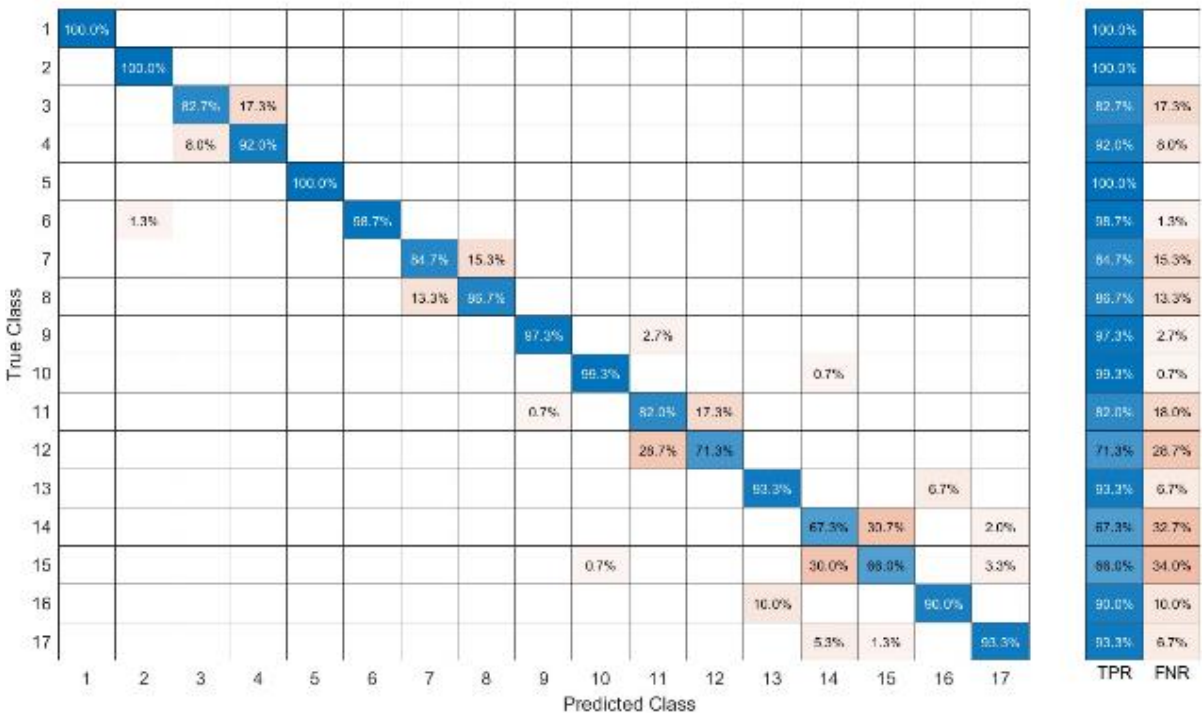


Figure 3 – d) Medium Gaussian SVM confusion matrix

MRKS_C17_200

Coarse Tree (*Train Accuracy: 29.4%*)

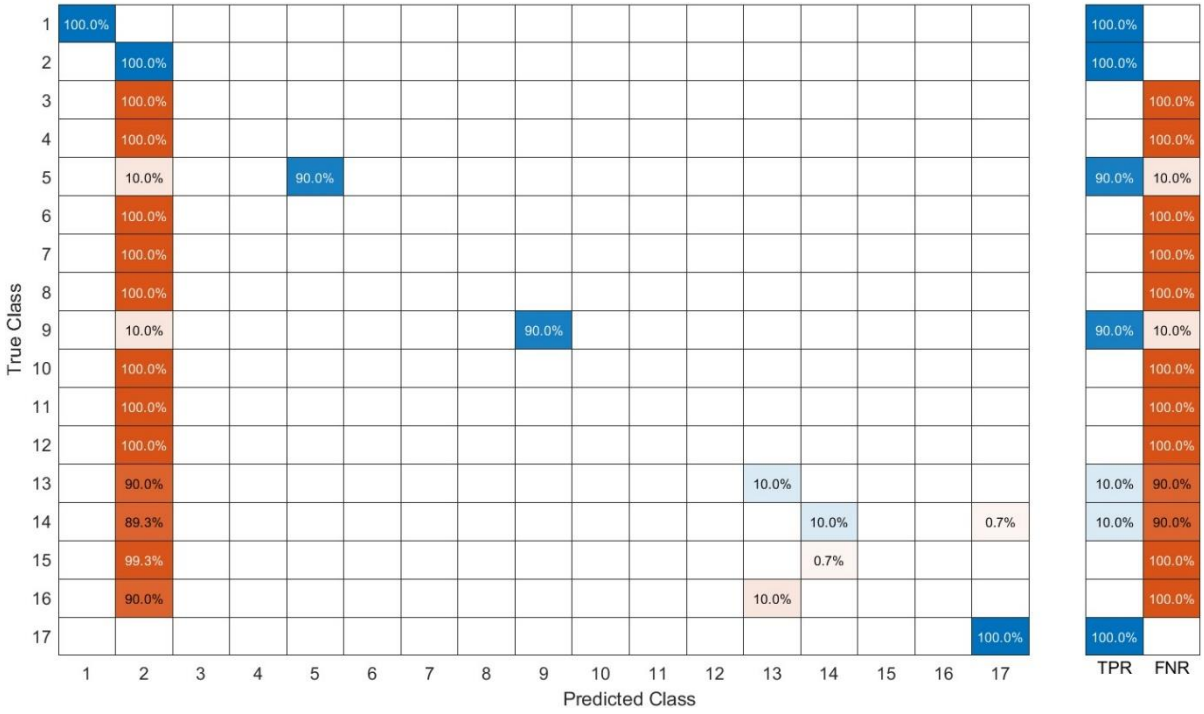


Figure 4 – a) Coarse tree confusion matrix

Coarse KNN (*Train Accuracy: 84.4%*)

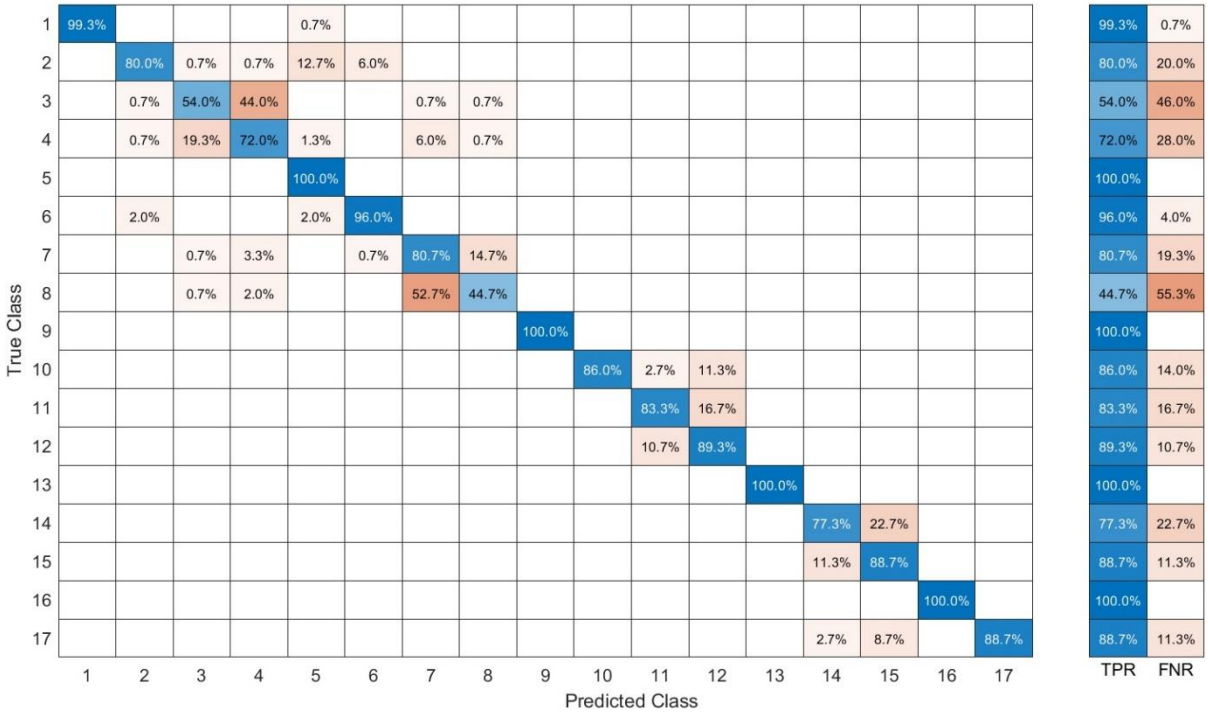


Figure 4 – b) Coarse KNN confusion matrix

MRKS_C17_200

Quadratic Discriminant (*Train Accuracy: 98.6%*)

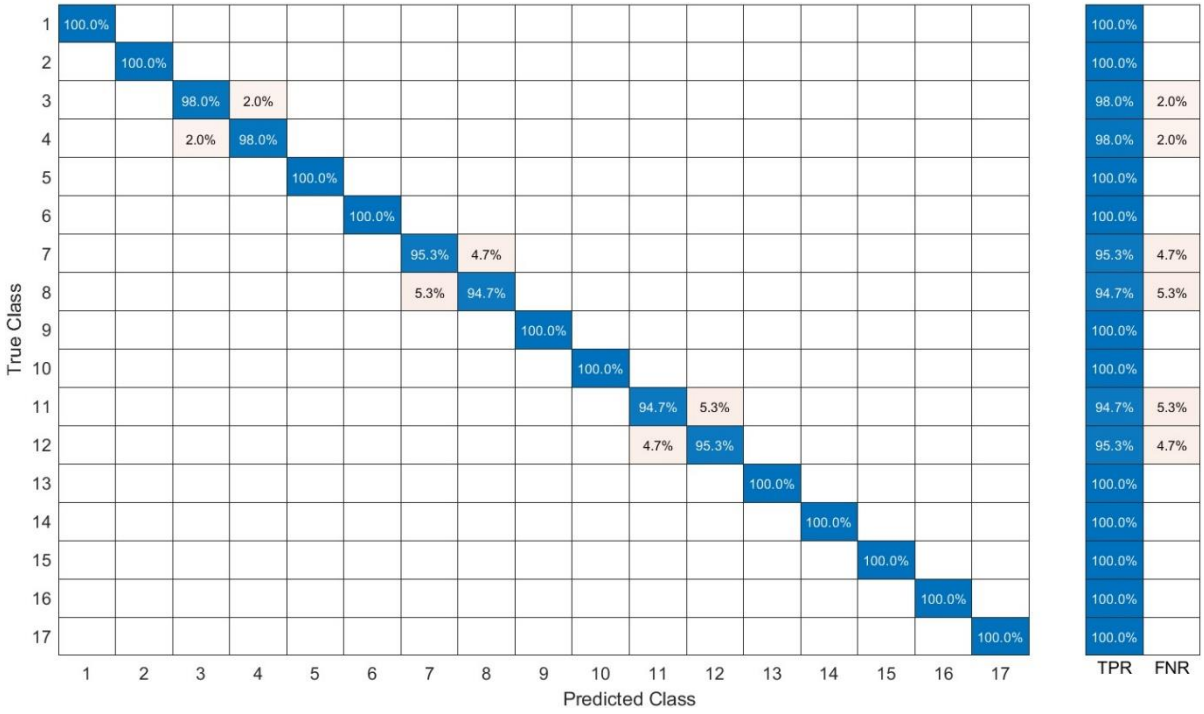


Figure 4 – c) Quadratic discriminant confusion matrix

Medium Gaussian SVM (*Train Accuracy: 97.7%*)

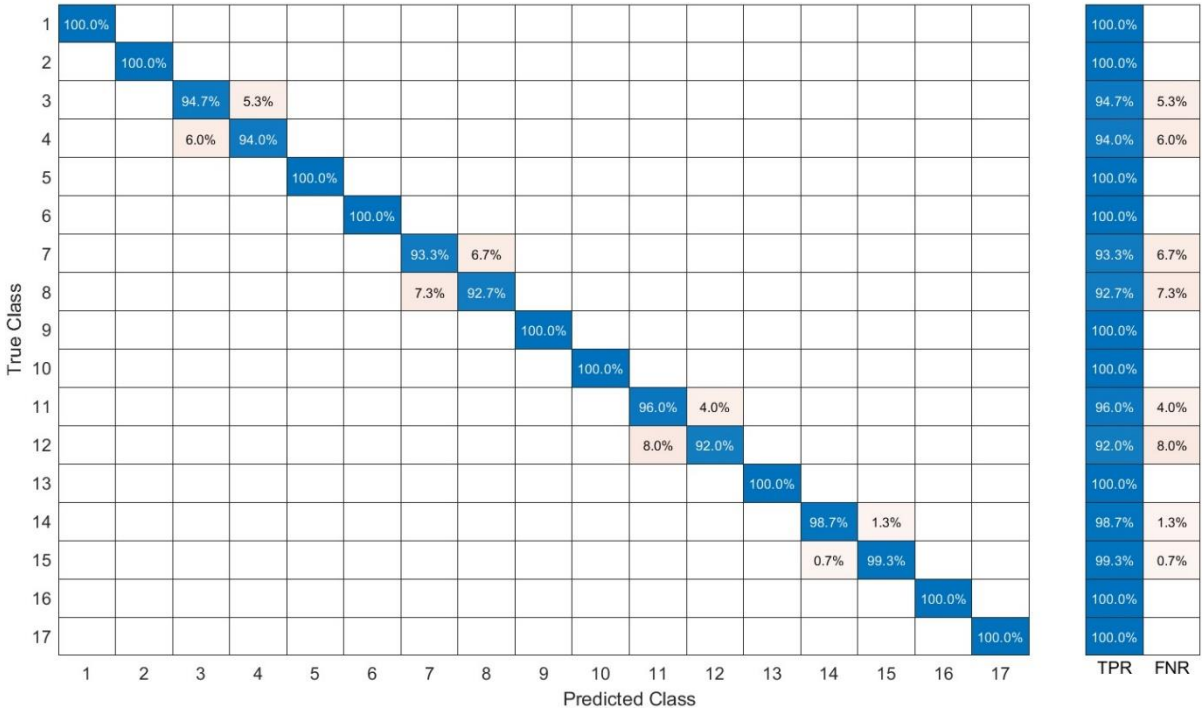


Figure 4 – d) Medium Gaussian SVM confusion matrix

M_L21_200

Coarse Tree (*Train Accuracy: 23.8%*)

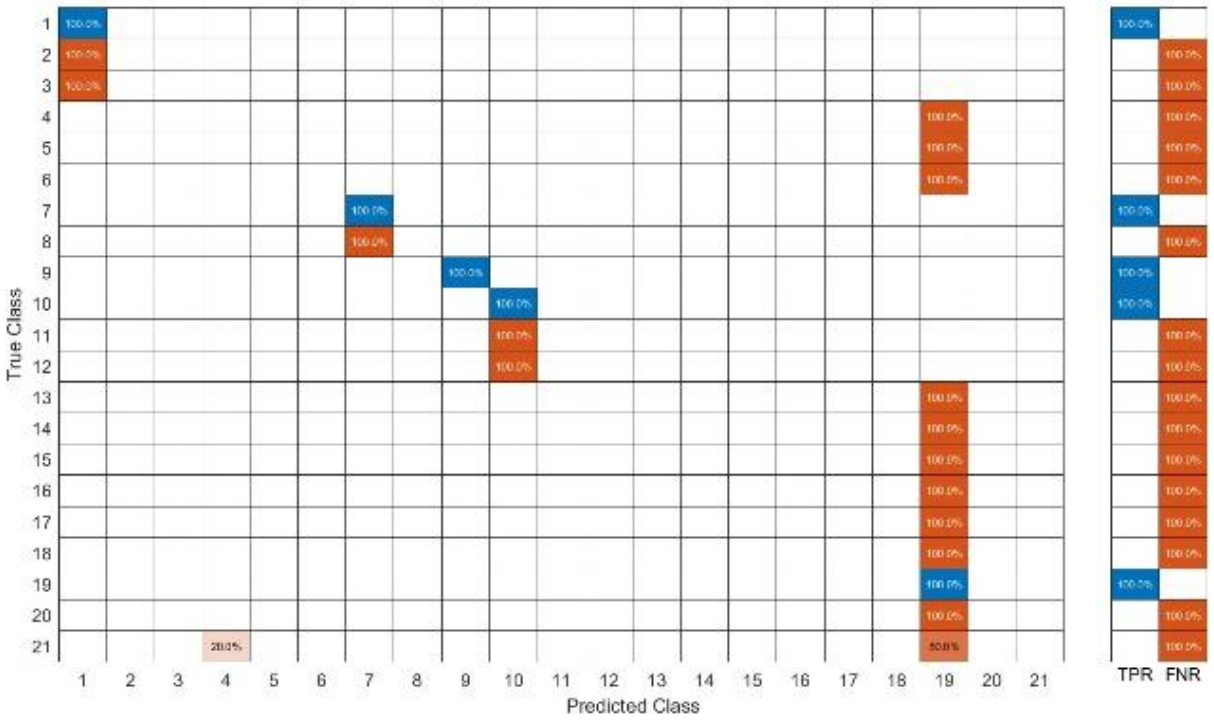


Figure 5 – a) Coarse tree confusion matrix

Coarse KNN (*Train Accuracy: 74.3%*)

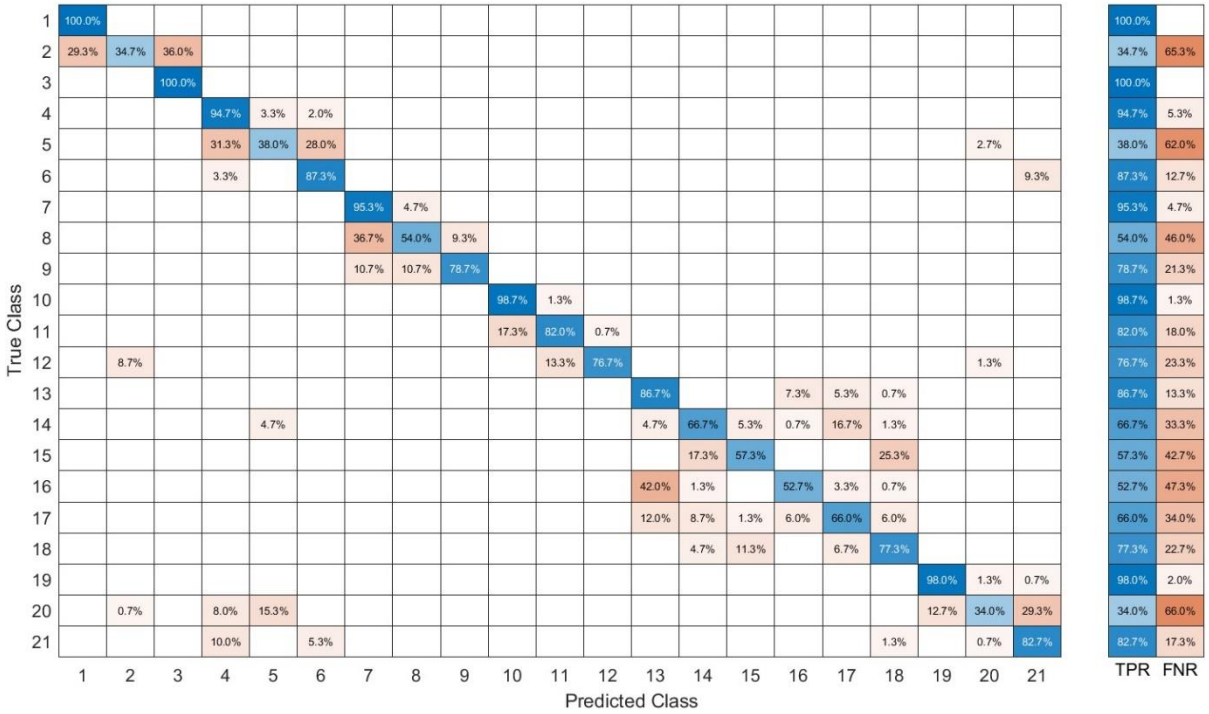


Figure 5 – b) Coarse KNN confusion matrix

M_L21_200

Quadratic Discriminant (*Train Accuracy: 97.0%*)

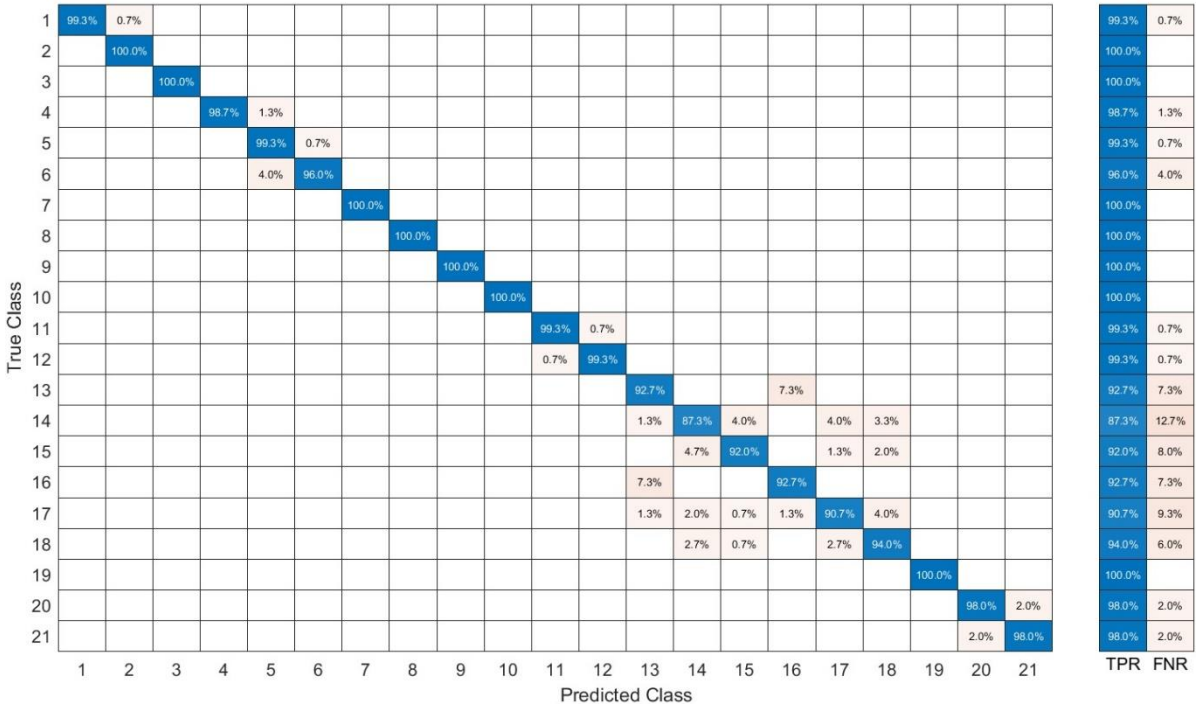


Figure 5 – c) Quadratic discriminant confusion matrix

Medium Gaussian SVM (*Train Accuracy: 96.8%*)

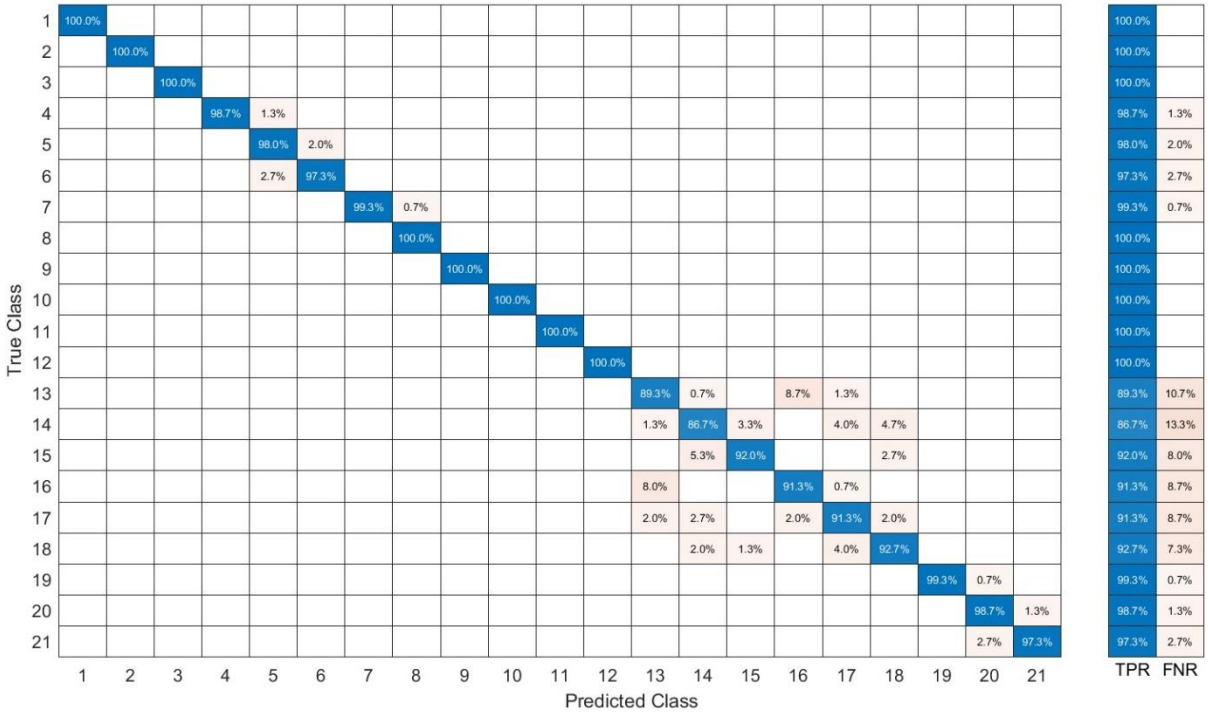


Figure 5 – d) Medium Gaussian SVM confusion matrix

MRKS_L21_200

Coarse Tree (*Train Accuracy: 23.8%*)

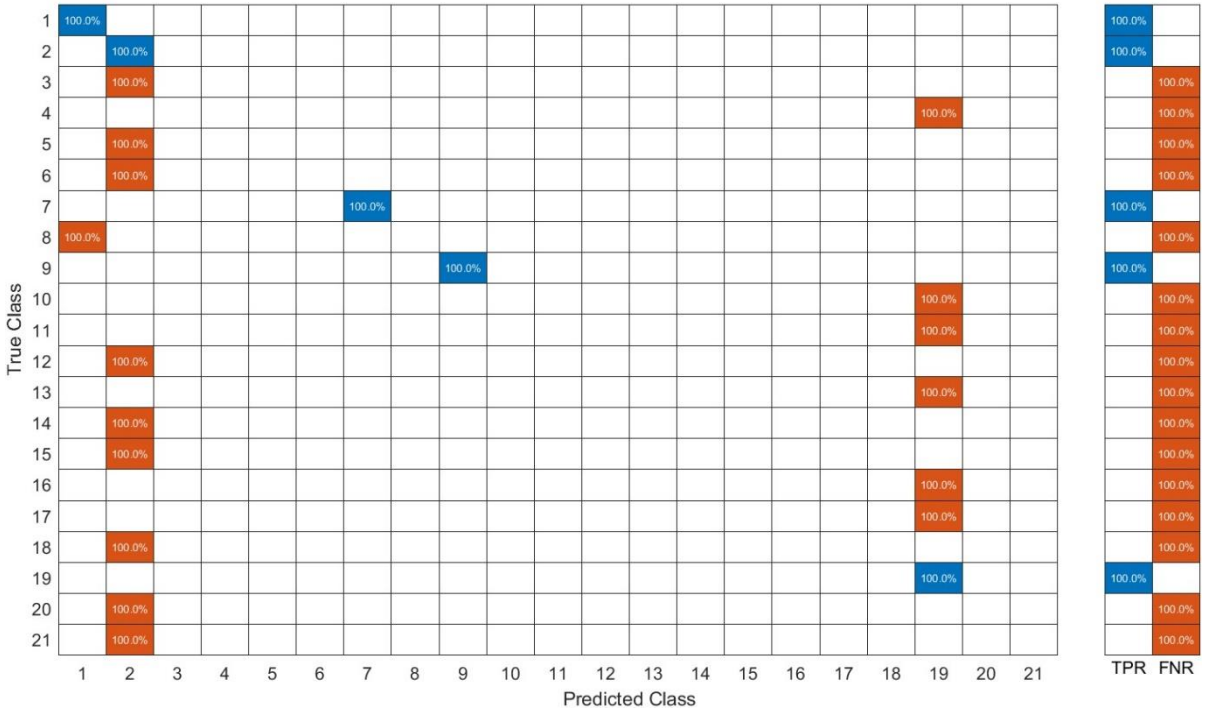


Figure 6 – a) Coarse tree confusion matrix

Coarse KNN (*Train Accuracy: 97.2%*)

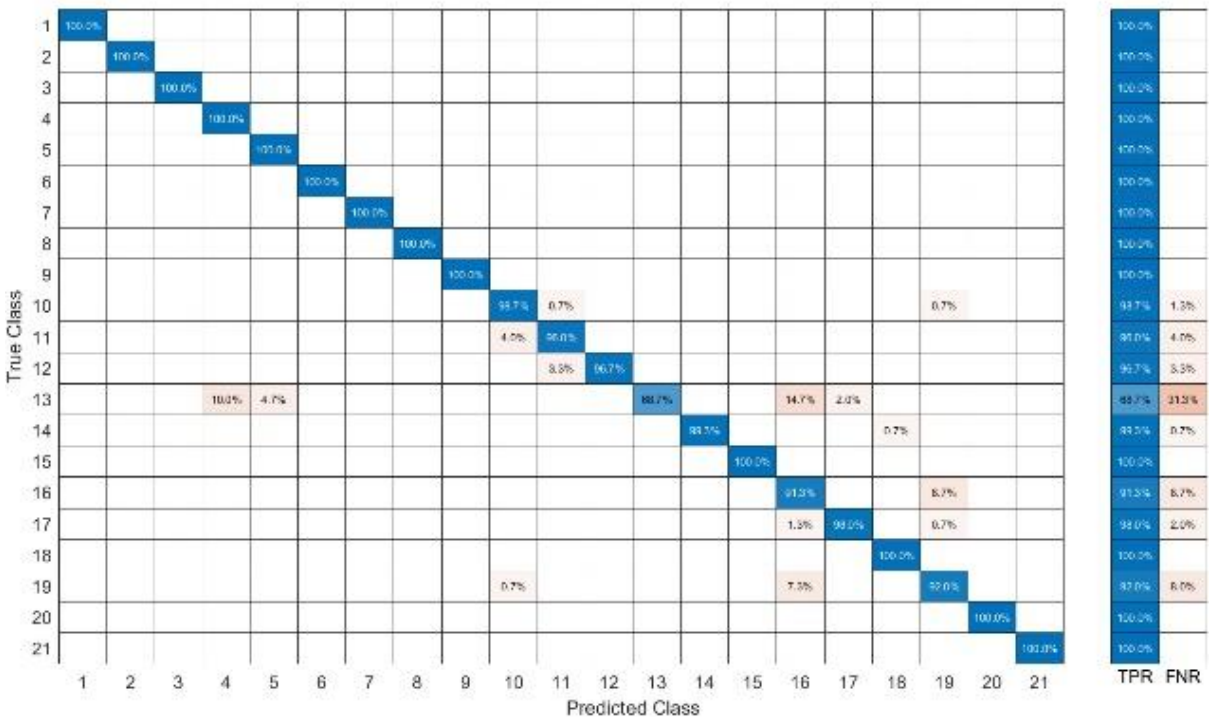


Figure 6 – b) Coarse KNN confusion matrix

MRKS_L21_200

Quadratic Discriminant (*Train Accuracy: 100.0%*)

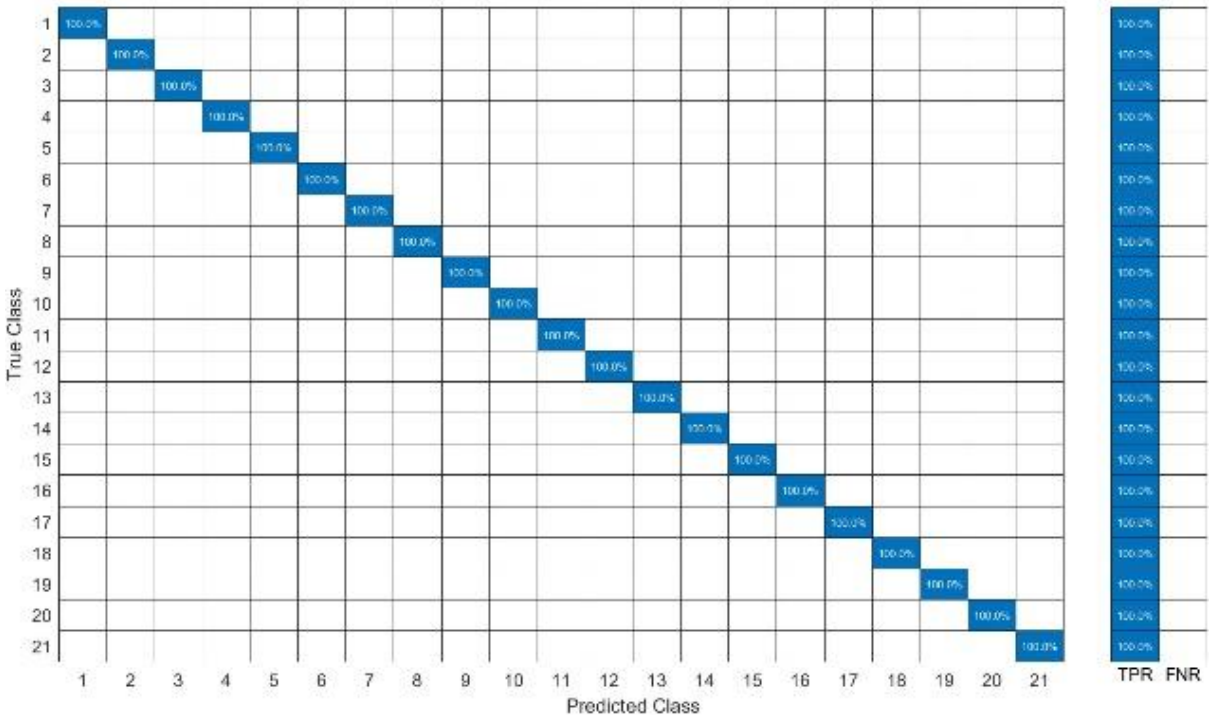


Figure 6 – c) Quadratic discriminant confusion matrix

Medium Gaussian SVM (*Train Accuracy: 100.0%*)

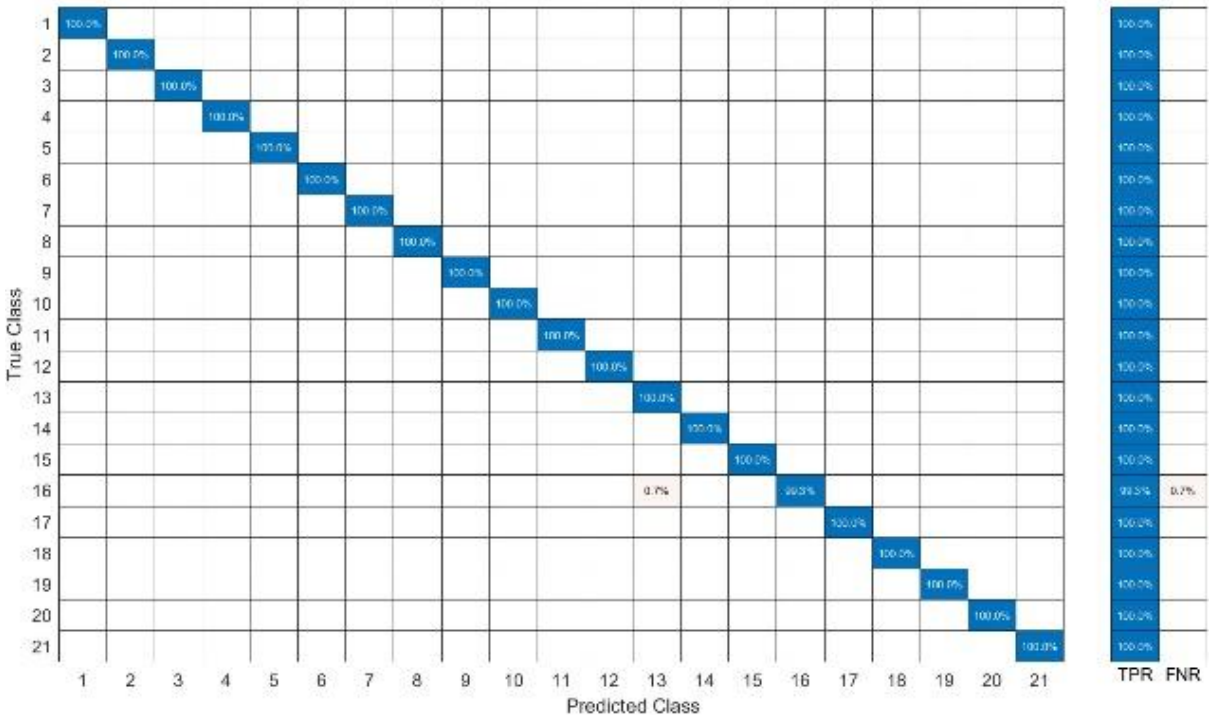


Figure 6 – d) Medium Gaussian SVM confusion matrix

M_S21_200

Coarse Tree (*Train Accuracy: 23.6%*)

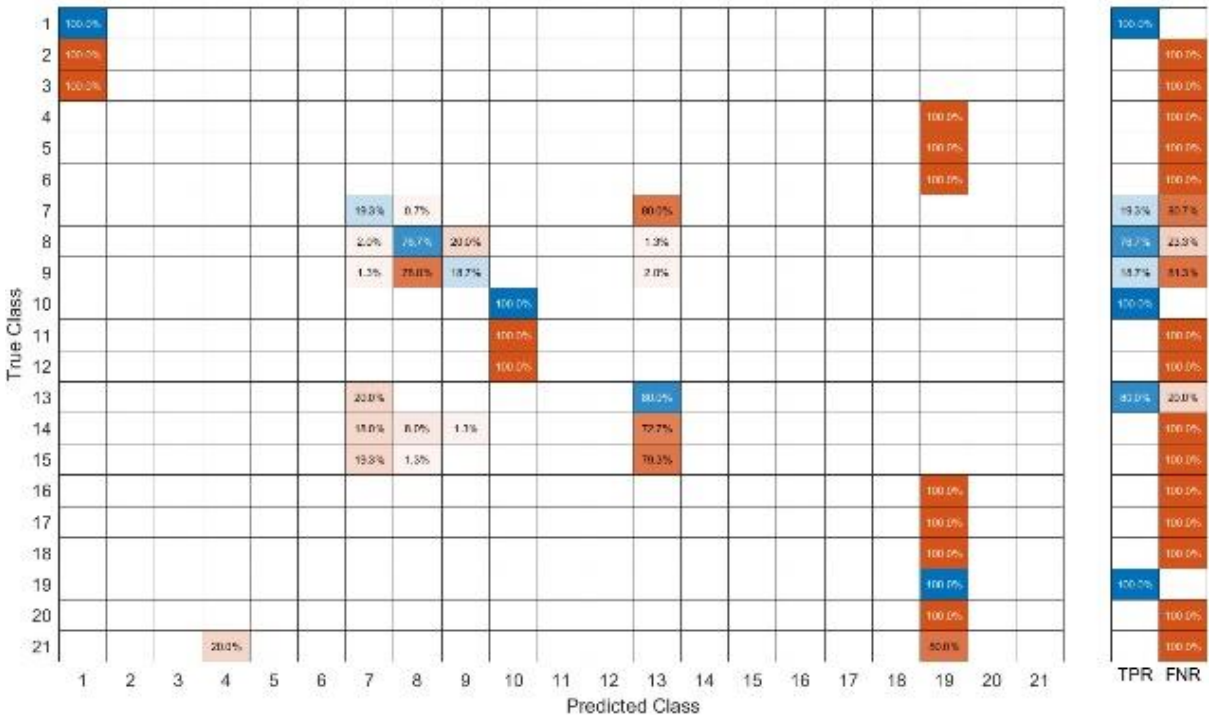


Figure 7 – a) Coarse tree confusion matrix

Coarse KNN (*Train Accuracy: 65.3%*)

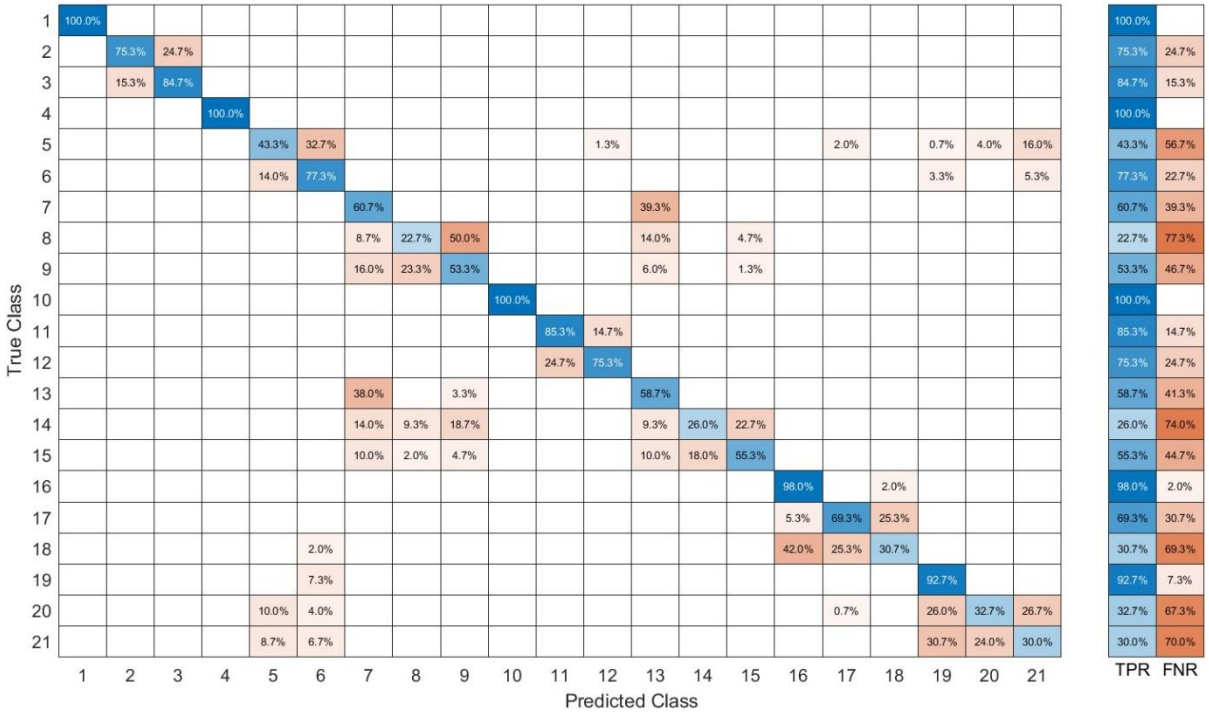


Figure 7 – b) Coarse KNN confusion matrix

M_S21_200

Quadratic Discriminant (*Train Accuracy: 90.9%*)

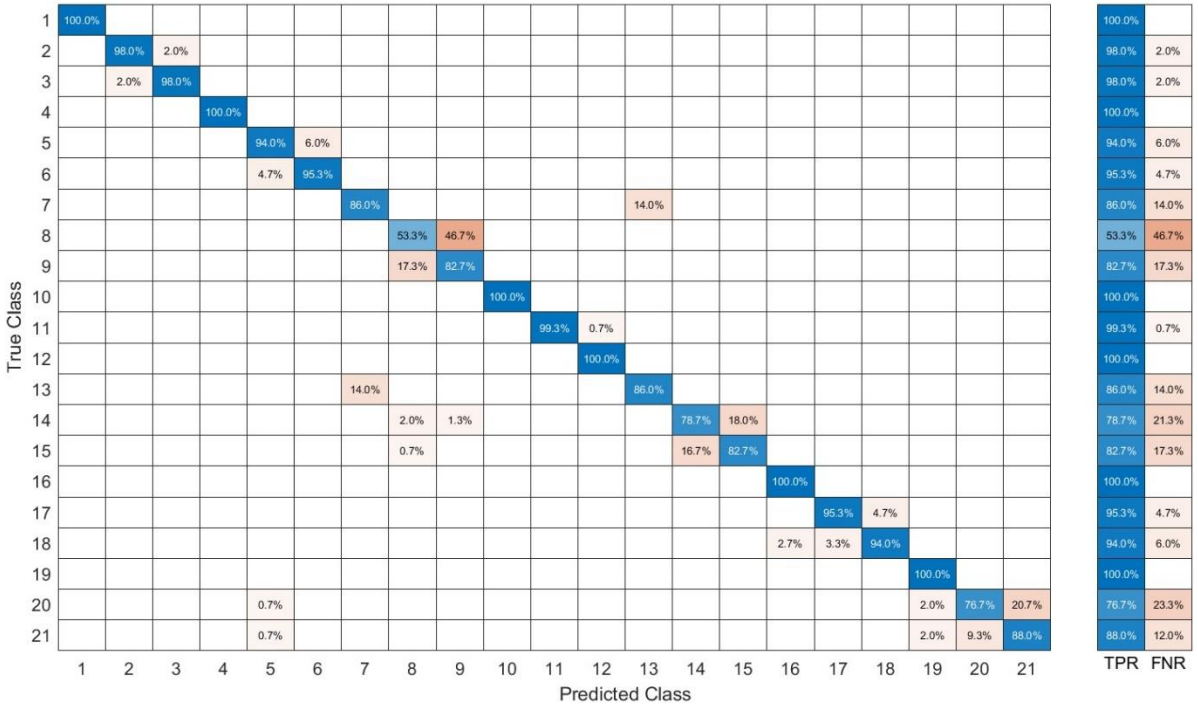


Figure 7 – c) Quadratic discriminant confusion matrix

Medium Gaussian SVM (*Train Accuracy: 90.3%*)

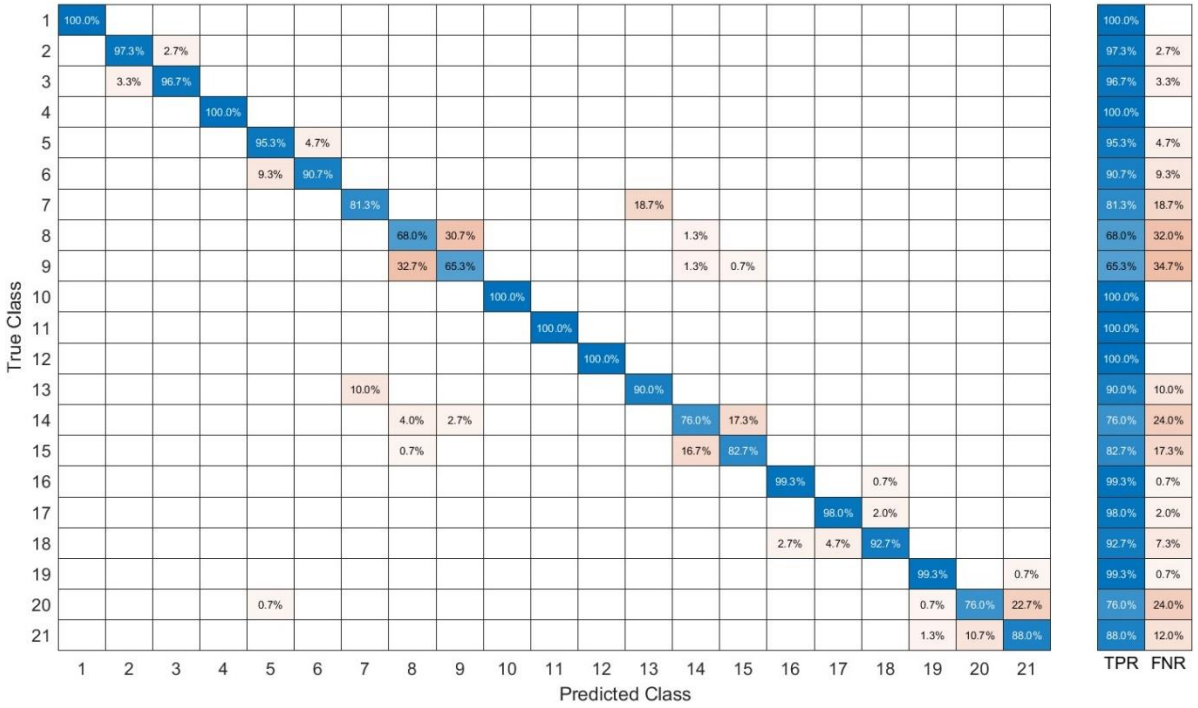


Figure 7 – d) Medium Gaussian SVM confusion matrix

MRKS_S21_200

Coarse Tree (*Train Accuracy: 23.8%*)

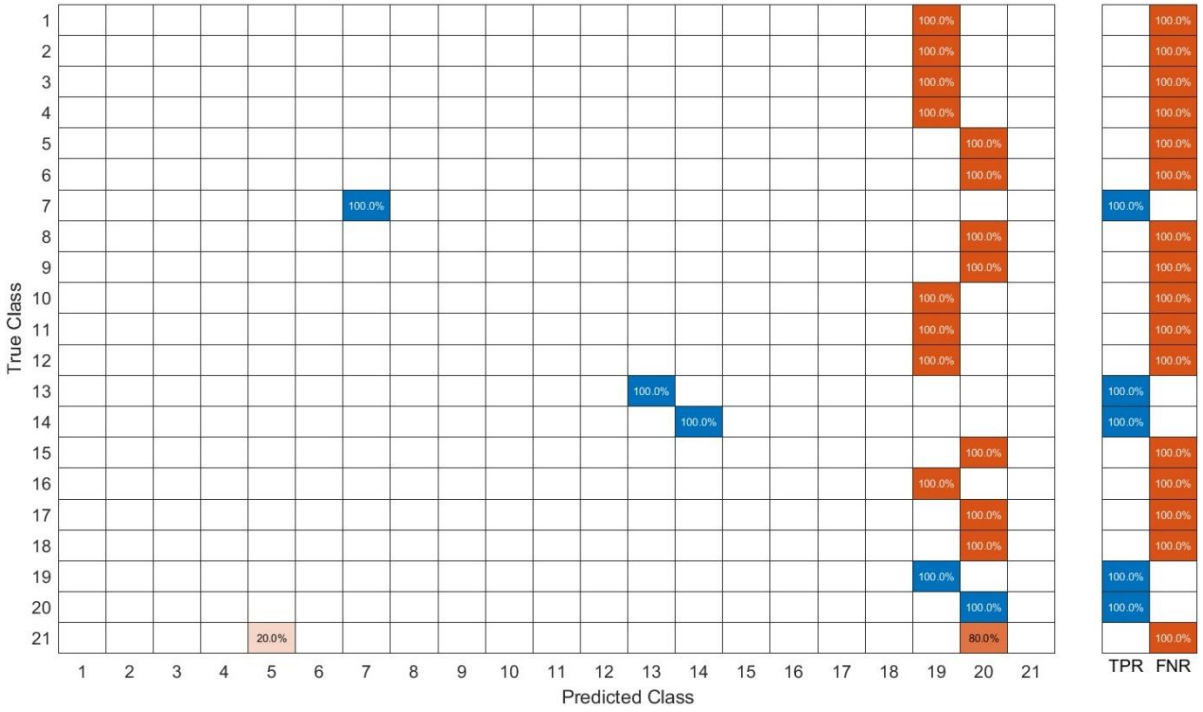


Figure 8 – a) Coarse tree confusion matrix

Coarse KNN (*Train Accuracy: 90.3%*)

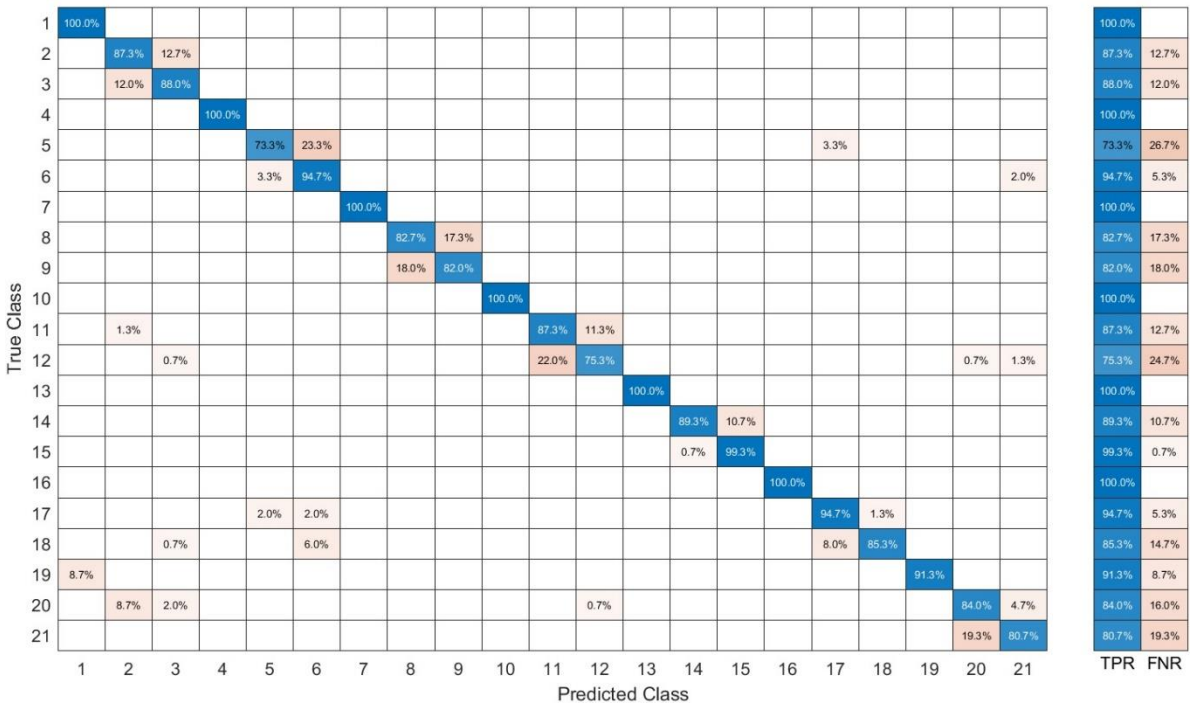


Figure 8 – b) Coarse KNN confusion matrix

MRKS_S21_200

Quadratic Discriminant (*Train Accuracy: 100.0%*)

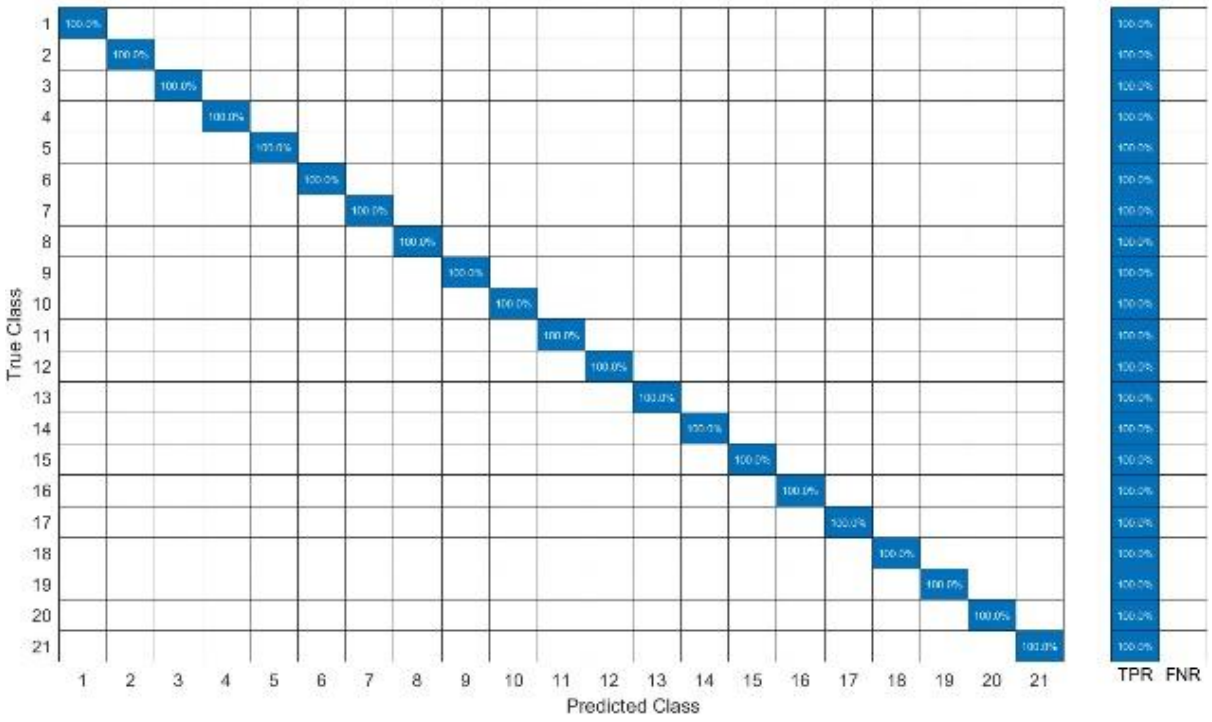


Figure 8 – c) Quadratic discriminant confusion matrix

Medium Gaussian SVM (*Train Accuracy: 99.9%*)

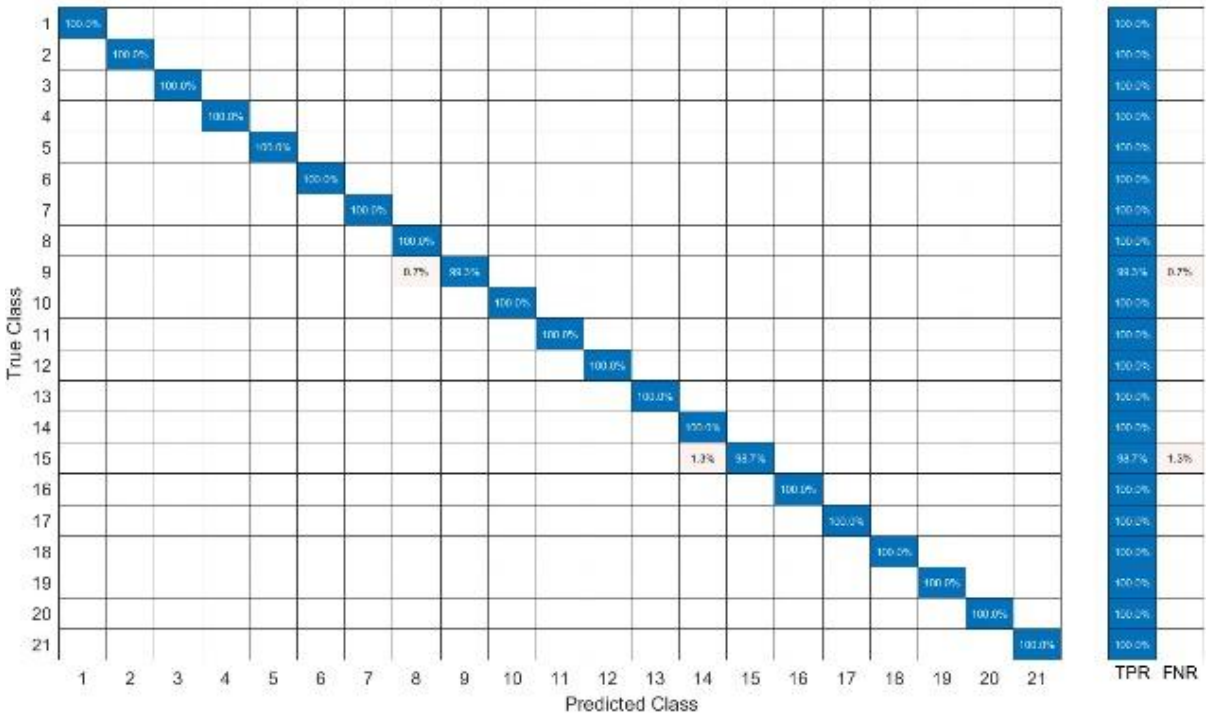


Figure 8 – d) Medium Gaussian SVM confusion matrix

Appendix D

Machine Learning Results

M_W7_100	Coarse Tree	Coarse KNN	Quadratic Discriminant	Medium Gaussian SVM
Train Accuracy	67.2	73.9	98.9	99.2
Prediction Performance	65.71	77.71	97.71	97.71
Misclassification Error	34.29	22.29	2.29	2.29
M_W7_200	**Coarse Tree**	**Coarse KNN**	**Quadratic Discriminant**	**Medium Gaussian SVM**
Train Accuracy	71.3	81.0	98.4	99.0
Prediction Performance	71.43	82.57	96.57	97.43
Misclassification Error	28.57	17.43	3.43	2.57

MR_W7_100	Coarse Tree	Coarse KNN	Quadratic Discriminant	Medium Gaussian SVM
Train Accuracy	69.3	99.2	100.0	100.0
Prediction Performance	71.43	99.43	100.00	100.00
Misclassification Error	28.57	0.57	0.00	0.00
MR_W7_200	**Coarse Tree**	**Coarse KNN**	**Quadratic Discriminant**	**Medium Gaussian SVM**
Train Accuracy	71.4	99.9	100.0	100.0
Prediction Performance	71.43	100.00	100.00	100.00
Misclassification Error	28.57	0.00	0.00	0.00

MRK_W7_100	Coarse Tree	Coarse KNN	Quadratic Discriminant	Medium Gaussian SVM
Train Accuracy	69.5	99.2	100.0	100.0
Prediction Performance	71.43	100.00	100.00	100.00
Misclassification Error	28.57	0.00	0.00	0.00
MRK_W7_200	**Coarse Tree**	**Coarse KNN**	**Quadratic Discriminant**	**Medium Gaussian SVM**
Train Accuracy	71.4	100.0	100.0	100.0
Prediction Performance	71.43	100.00	100.00	100.00
Misclassification Error	28.57	0.00	0.00	0.00

MRKS_W7_100	Coarse Tree	Coarse KNN	Quadratic Discriminant	Medium Gaussian SVM
Train Accuracy	69.5	99.0	100.0	100.0
Prediction Performance	57.14	100.00	100.00	100.00
Misclassification Error	42.86	0.00	0.00	0.00
MRKS_W7_200	**Coarse Tree**	**Coarse KNN**	**Quadratic Discriminant**	**Medium Gaussian SVM**
Train Accuracy	71.4	99.8	100.0	100.0
Prediction Performance	71.43	100.00	100.00	100.00
Misclassification Error	28.57	0.00	0.00	0.00

M_C17_100	Coarse Tree	Coarse KNN	Quadratic Discriminant	Medium Gaussian SVM
Train Accuracy	24.8	79.2	90.4	91.1
Prediction Performance	23.53	81.88	91.29	91.76
Misclassification Error	76.47	18.12	8.71	8.24
M_C17_200	**Coarse Tree**	**Coarse KNN**	**Quadratic Discriminant**	**Medium Gaussian SVM**
Train Accuracy	26.3	79.8	88.3	88.5
Prediction Performance	25.88	80.71	87.18	89.41
Misclassification Error	74.12	19.29	12.82	10.59

MR_C17_100	Coarse Tree	Coarse KNN	Quadratic Discriminant	Medium Gaussian SVM
Train Accuracy	27.5	85.4	99.5	99.1
Prediction Performance	29.41	86.59	99.76	100.00
Misclassification Error	70.59	13.41	0.24	0.00
MR_C17_200	**Coarse Tree**	**Coarse KNN**	**Quadratic Discriminant**	**Medium Gaussian SVM**
Train Accuracy	29.4	87.9	98.9	98.8
Prediction Performance	29.41	88.94	98.35	98.24
Misclassification Error	70.59	11.06	1.65	1.76

MRK_C17_100	Coarse Tree	Coarse KNN	Quadratic Discriminant	Medium Gaussian SVM
Train Accuracy	27.8	82.7	99.5	97.9
Prediction Performance	29.41	84.24	99.53	99.06
Misclassification Error	70.59	15.76	0.47	0.94
MRK_C17_200	**Coarse Tree**	**Coarse KNN**	**Quadratic Discriminant**	**Medium Gaussian SVM**
Train Accuracy	29.4	83.2	98.6	97.4
Prediction Performance	29.41	83.29	98.00	97.18
Misclassification Error	70.59	16.71	2.00	2.82

MRKS_C17_100	Coarse Tree	Coarse KNN	Quadratic Discriminant	Medium Gaussian SVM
Train Accuracy	27.8	85.5	98.6	98.2
Prediction Performance	23.53	84.94	98.82	98.71
Misclassification Error	76.47	15.06	1.18	1.29
MRKS_C17_200	**Coarse Tree**	**Coarse KNN**	**Quadratic Discriminant**	**Medium Gaussian SVM**
Train Accuracy	29.4	84.4	98.6	97.7
Prediction Performance	29.41	83.29	97.65	97.53
Misclassification Error	70.59	16.71	2.35	2.47

M_L21_100	Coarse Tree	Coarse KNN	Quadratic Discriminant	Medium Gaussian SVM
Train Accuracy	22.5	68.3	99.0	99.0
Prediction Performance	23.81	67.62	99.24	99.05
Misclassification Error	76.19	32.38	0.76	0.95
M_L21_200	**Coarse Tree**	**Coarse KNN**	**Quadratic Discriminant**	**Medium Gaussian SVM**
Train Accuracy	23.8	74.3	97.0	96.8
Prediction Performance	23.81	76.00	97.05	97.24
Misclassification Error	76.19	24.00	2.95	2.76

MR_L21_100	Coarse Tree	Coarse KNN	Quadratic Discriminant	Medium Gaussian SVM
Train Accuracy	22.5	97.0	100.0	100.0
Prediction Performance	23.81	97.52	100.00	100.00
Misclassification Error	76.19	2.48	0.00	0.00
MR_L21_200	**Coarse Tree**	**Coarse KNN**	**Quadratic Discriminant**	**Medium Gaussian SVM**
Train Accuracy	23.8	98.1	100.0	100.0
Prediction Performance	23.81	98.76	100.00	100.00
Misclassification Error	76.19	1.24	0.00	0.00

MRK_L21_100	Coarse Tree	Coarse KNN	Quadratic Discriminant	Medium Gaussian SVM
Train Accuracy	22.2	96.1	100.0	99.9
Prediction Performance	23.81	97.14	100.00	100.00
Misclassification Error	76.19	2.86	0.00	0.00
MRK_L21_200	**Coarse Tree**	**Coarse KNN**	**Quadratic Discriminant**	**Medium Gaussian SVM**
Train Accuracy	23.8	97.4	100.0	99.9
Prediction Performance	23.81	97.81	100.00	99.90
Misclassification Error	76.19	2.19	0.00	0.10

MRKS_L21_100	Coarse Tree	Coarse KNN	Quadratic Discriminant	Medium Gaussian SVM
Train Accuracy	22.2	96.3	100.0	99.9
Prediction Performance	11.90	96.48	100.0	99.90
Misclassification Error	88.10	3.52	0.00	0.10
MRKS_L21_200	**Coarse Tree**	**Coarse KNN**	**Quadratic Discriminant**	**Medium Gaussian SVM**
Train Accuracy	23.8	97.2	100.0	100.0
Prediction Performance	23.81	98.19	100.0	99.90
Misclassification Error	76.19	1.81	0.00	0.10

M_S21_100	Coarse Tree	Coarse KNN	Quadratic Discriminant	Medium Gaussian SVM
Train Accuracy	22.2	60.5	92.1	91.8
Prediction Performance	23.81	62.48	92.57	93.14
Misclassification Error	76.19	37.52	7.43	6.86
M_S21_200	**Coarse Tree**	**Coarse KNN**	**Quadratic Discriminant**	**Medium Gaussian SVM**
Train Accuracy	23.6	65.3	90.9	90.3
Prediction Performance	23.43	65.81	91.05	91.05
Misclassification Error	76.57	34.19	8.95	8.95

MR_S21_100	Coarse Tree	Coarse KNN	Quadratic Discriminant	Medium Gaussian SVM
Train Accuracy	22.2	85.7	100.0	100.0
Prediction Performance	23.81	88.19	100.00	100.00
Misclassification Error	76.19	11.81	0.00	0.00
MR_S21_200	**Coarse Tree**	**Coarse KNN**	**Quadratic Discriminant**	**Medium Gaussian SVM**
Train Accuracy	23.8	91.4	100.0	100.0
Prediction Performance	23.81	94.86	100.00	100.00
Misclassification Error	76.19	5.14	0.00	0.00

MRK_S21_100	Coarse Tree	Coarse KNN	Quadratic Discriminant	Medium Gaussian SVM
Train Accuracy	22.2	88.8	100.0	99.9
Prediction Performance	23.81	93.14	100.00	100.00
Misclassification Error	76.19	6.86	0.00	0.00
MRK_S21_200	**Coarse Tree**	**Coarse KNN**	**Quadratic Discriminant**	**Medium Gaussian SVM**
Train Accuracy	23.8	90.6	100.0	99.9
Prediction Performance	23.81	90.19	100.00	100.00
Misclassification Error	76.19	9.81	0.00	0.00

MRKS_S21_100	Coarse Tree	Coarse KNN	Quadratic Discriminant	Medium Gaussian SVM
Train Accuracy	22.2	91.7	100.0	99.9
Prediction Performance	23.81	92.29	100.00	99.90
Misclassification Error	76.19	7.71	0.00	0.10
MRKS_S21_200	**Coarse Tree**	**Coarse KNN**	**Quadratic Discriminant**	**Medium Gaussian SVM**
Train Accuracy	23.8	90.3	100.0	99.9
Prediction Performance	23.81	91.90	100.00	100.00
Misclassification Error	76.19	8.10	0.00	0.00

Appendix E

Features Selection Results

M_W7_100	Coarse Tree	Coarse KNN	Quadratic Discriminant	Medium Gaussian SVM
Train Accuracy	67.2	73.9	98.9	99.2
Prediction Performance	65.71	77.71	97.71	97.71
Misclassification Error	34.29	22.29	2.29	2.29
MRKS_W7_100	**Coarse Tree**	**Coarse KNN**	**Quadratic Discriminant**	**Medium Gaussian SVM**
Train Accuracy	69.5	99.0	100.0	100.0
Prediction Performance	57.14	100.00	100.00	100.00
Misclassification Error	42.86	0.00	0.00	0.00
Chi-Square	**Coarse Tree**	**Coarse KNN**	**Quadratic Discriminant**	**Medium Gaussian SVM**
Train Accuracy	69.3	96.0	100.0	100.0
Prediction Performance	71.43	97.71	100.00	100.00
Misclassification Error	28.57	2.29	0.00	0.00
MRMR	**Coarse Tree**	**Coarse KNN**	**Quadratic Discriminant**	**Medium Gaussian SVM**
Train Accuracy	69.5	86.7	100.0	100.0
Prediction Performance	71.43	88.00	100.00	100.00
Misclassification Error	28.57	12.00	0.00	0.00

M_W7_200	Coarse Tree	Coarse KNN	Quadratic Discriminant	Medium Gaussian SVM
Train Accuracy	71.3	81.0	98.4	99.0
Prediction Performance	71.43	82.57	96.57	97.43
Misclassification Error	28.57	17.43	3.43	2.57
MRKS_W7_200	**Coarse Tree**	**Coarse KNN**	**Quadratic Discriminant**	**Medium Gaussian SVM**
Train Accuracy	71.4	99.8	100.0	100.0
Prediction Performance	71.43	100.00	100.00	100.00
Misclassification Error	28.57	0.00	0.00	0.00
Chi-Square	**Coarse Tree**	**Coarse KNN**	**Quadratic Discriminant**	**Medium Gaussian SVM**
Train Accuracy	71.4	98.3	100.0	100.0
Prediction Performance	87.14	96.38	96.67	96.67
Misclassification Error	12.86	3.62	3.33	3.33
MRMR	**Coarse Tree**	**Coarse KNN**	**Quadratic Discriminant**	**Medium Gaussian SVM**
Train Accuracy	71.2	97.0	100.0	100.0
Prediction Performance	87.14	95.62	96.67	96.57
Misclassification Error	12.86	4.38	3.33	3.43

M_C17_100	Coarse Tree	Coarse KNN	Quadratic Discriminant	Medium Gaussian SVM
Train Accuracy	24.8	79.2	90.4	91.1
Prediction Performance	23.53	81.88	91.29	91.76
Misclassification Error	76.47	18.12	8.71	8.24
MRKS_C17_100	**Coarse Tree**	**Coarse KNN**	**Quadratic Discriminant**	**Medium Gaussian SVM**
Train Accuracy	27.8	85.5	98.6	98.2
Prediction Performance	23.53	84.94	98.82	98.71
Misclassification Error	76.47	15.06	1.18	1.29
Chi-Square	**Coarse Tree**	**Coarse KNN**	**Quadratic Discriminant**	**Medium Gaussian SVM**
Train Accuracy	25.0	78.8	90.0	91.0
Prediction Performance	23.53	81.88	91.29	91.76
Misclassification Error	76.47	18.12	8.71	8.24
MRMR	**Coarse Tree**	**Coarse KNN**	**Quadratic Discriminant**	**Medium Gaussian SVM**
Train Accuracy	27.5	65.6	96.1	91.8
Prediction Performance	29.41	63.76	96.24	90.82
Misclassification Error	70.59	36.24	3.76	9.18

M_C17_200	Coarse Tree	Coarse KNN	Quadratic Discriminant	Medium Gaussian SVM
Train Accuracy	26.3	79.8	88.3	88.5
Prediction Performance	25.88	80.71	87.18	89.41
Misclassification Error	74.12	19.29	12.82	10.59
MRKS_C17_200	**Coarse Tree**	**Coarse KNN**	**Quadratic Discriminant**	**Medium Gaussian SVM**
Train Accuracy	29.4	84.4	98.6	97.7
Prediction Performance	29.41	83.29	97.65	97.53
Misclassification Error	70.59	16.71	2.35	2.47
Chi-Square	**Coarse Tree**	**Coarse KNN**	**Quadratic Discriminant**	**Medium Gaussian SVM**
Train Accuracy	26.3	79.1	88.3	88.7
Prediction Performance	25.88	80.71	87.18	89.41
Misclassification Error	74.12	19.29	12.82	10.59
MRMR	**Coarse Tree**	**Coarse KNN**	**Quadratic Discriminant**	**Medium Gaussian SVM**
Train Accuracy	29.4	62.9	91.6	88.4
Prediction Performance	29.41	63.18	91.41	89.76
Misclassification Error	70.59	36.82	8.59	10.24

M_L21_100	Coarse Tree	Coarse KNN	Quadratic Discriminant	Medium Gaussian SVM
Train Accuracy	22.5	68.3	99.0	99.0
Prediction Performance	23.81	67.62	99.24	99.05
Misclassification Error	76.19	32.38	0.76	0.95
MRKS_L21_100	**Coarse Tree**	**Coarse KNN**	**Quadratic Discriminant**	**Medium Gaussian SVM**
Train Accuracy	22.2	96.3	100.0	99.9
Prediction Performance	11.90	96.48	100.0	99.90
Misclassification Error	88.10	3.52	0.00	0.10
Chi-Square	**Coarse Tree**	**Coarse KNN**	**Quadratic Discriminant**	**Medium Gaussian SVM**
Train Accuracy	22.5	68.5	98.9	98.9
Prediction Performance	23.81	67.62	99.24	99.05
Misclassification Error	76.19	32.38	0.76	0.95
MRMR	**Coarse Tree**	**Coarse KNN**	**Quadratic Discriminant**	**Medium Gaussian SVM**
Train Accuracy	22.2	80.4	99.3	98.6
Prediction Performance	23.81	82.29	98.86	95.24
Misclassification Error	76.19	17.71	1.14	4.76

M_L21_200	Coarse Tree	Coarse KNN	Quadratic Discriminant	Medium Gaussian SVM
Train Accuracy	23.8	74.3	97.0	96.8
Prediction Performance	23.81	76.00	97.05	97.24
Misclassification Error	76.19	24.00	2.95	2.76
MRKS_L21_200	**Coarse Tree**	**Coarse KNN**	**Quadratic Discriminant**	**Medium Gaussian SVM**
Train Accuracy	23.8	97.2	100.0	100.0
Prediction Performance	23.81	98.19	100.0	99.90
Misclassification Error	76.19	1.81	0.00	0.10
Chi-Square	**Coarse Tree**	**Coarse KNN**	**Quadratic Discriminant**	**Medium Gaussian SVM**
Train Accuracy	23.8	74.3	97.0	96.8
Prediction Performance	23.81	76.00	97.05	97.24
Misclassification Error	76.19	24.00	2.95	2.76
MRMR	**Coarse Tree**	**Coarse KNN**	**Quadratic Discriminant**	**Medium Gaussian SVM**
Train Accuracy	23.8	80.2	99.7	99.0
Prediction Performance	23.81	80.00	99.71	99.14
Misclassification Error	76.19	20.00	0.29	0.86

M_S21_100	Coarse Tree	Coarse KNN	Quadratic Discriminant	Medium Gaussian SVM
Train Accuracy	22.2	60.5	92.1	91.8
Prediction Performance	23.81	62.48	92.57	93.14
Misclassification Error	76.19	37.52	7.43	6.86
MRKS_S21_100	**Coarse Tree**	**Coarse KNN**	**Quadratic Discriminant**	**Medium Gaussian SVM**
Train Accuracy	22.2	91.7	100.0	99.9
Prediction Performance	23.81	92.29	100.00	99.90
Misclassification Error	76.19	7.71	0.00	0.10
Chi-Square	**Coarse Tree**	**Coarse KNN**	**Quadratic Discriminant**	**Medium Gaussian SVM**
Train Accuracy	22.2	60.8	91.8	91.8
Prediction Performance	23.81	62.48	92.57	93.14
Misclassification Error	76.19	37.52	7.43	6.86
MRMR	**Coarse Tree**	**Coarse KNN**	**Quadratic Discriminant**	**Medium Gaussian SVM**
Train Accuracy	22.5	66.9	92.3	90.2
Prediction Performance	23.81	69.90	95.43	94.48
Misclassification Error	76.19	30.10	4.57	5.52

M_S21_200	Coarse Tree	Coarse KNN	Quadratic Discriminant	Medium Gaussian SVM
Train Accuracy	23.6	65.3	90.9	90.3
Prediction Performance	23.43	65.81	91.05	91.05
Misclassification Error	76.57	34.19	8.95	8.95
MRKS_S21_200	**Coarse Tree**	**Coarse KNN**	**Quadratic Discriminant**	**Medium Gaussian SVM**
Train Accuracy	23.8	90.3	100.0	99.9
Prediction Performance	23.81	91.90	100.00	100.00
Misclassification Error	76.19	8.10	0.00	0.00
Chi-Square	**Coarse Tree**	**Coarse KNN**	**Quadratic Discriminant**	**Medium Gaussian SVM**
Train Accuracy	23.6	65.2	90.8	90.4
Prediction Performance	23.43	65.81	91.05	91.05
Misclassification Error	76.57	34.19	8.95	8.95
MRMR	**Coarse Tree**	**Coarse KNN**	**Quadratic Discriminant**	**Medium Gaussian SVM**
Train Accuracy	23.8	75.1	95.5	94.5
Prediction Performance	23.81	75.90	96.86	95.33
Misclassification Error	76.19	24.10	3.14	4.67