

université  
PARIS-SACLAY



INSTITUT  
POLYTECHNIQUE  
DE PARIS

INRAE

INTERNSHIP REPORT FOR DATA  
SCIENCES MASTER'S DEGREE

---

DEEP LEARNING AND GENOMIC  
SELECTION

---

*By:*

Ronny Tonato

ronny\_75@hotmail.com

*Internship supervisor:*

Eric Barrey

eric.barrey@inrae.fr

*Internship co-supervisor:*

Blaise Hanczar

blaise.hanczar@univ-evry.fr

*Performed at:*

BIGE Team, GABI Unit-INRAE, Jouy-en-Josas

September 2022

# Contents

<b>1</b>	<b>Introduction to the research problem</b>	<b>10</b>
1.1	Notions of genomics . . . . .	10
1.2	Problem context . . . . .	11
1.3	State-of-the-art . . . . .	11
1.4	Hypothesis . . . . .	12
1.5	Objectives . . . . .	12
<b>2</b>	<b>Material and methods</b>	<b>14</b>
2.1	Dataset . . . . .	14
2.1.1	Data description . . . . .	14
2.1.2	Data pre-processing . . . . .	15
2.1.3	Descriptive statistics . . . . .	15
2.2	Models . . . . .	16
2.2.1	Genomic Best Linear Unbiased Prediction (GBLUP) . . . . .	16
2.2.2	Multilayer Perceptron (MLP) . . . . .	17
2.2.3	Value Imputation and Mask Estimation (VIME) . . . . .	19
2.2.3.1	Problem formulation . . . . .	19
2.2.3.2	Self supervised learning approach . . . . .	20
2.2.3.3	Semi-supervised learning approach . . . . .	21
2.3	Training, validation and test sets . . . . .	22
<b>3</b>	<b>Results and discussion</b>	<b>24</b>
3.1	Model training . . . . .	24
3.1.1	Hyper-parameters initialization . . . . .	24
3.1.1.1	MLP model . . . . .	24
3.1.1.2	VIME model . . . . .	24
3.1.2	Learning curves . . . . .	25
3.2	Model performance . . . . .	27
3.2.1	Hyper-parameters tuning . . . . .	27
3.2.2	Improvement of learning curves . . . . .	27
3.3	Model comparison . . . . .	28
3.3.1	MSE analysis . . . . .	29
3.3.2	Performance of prediction by phenotype . . . . .	31
3.3.3	Correlation analysis between GBLUP and other models . . . . .	31
<b>4</b>	<b>Conclusion</b>	<b>34</b>
<b>5</b>	<b>Annexes</b>	<b>35</b>
	<b>Appendices</b>	<b>57</b>
<b>A</b>	<b>Glossary of genetics terminology</b>	<b>58</b>
	<b>Bibliography</b>	<b>60</b>

# List of Figures

2.1	An example of MLP with eight input variables $(x_1, \dots, x_8)$ , four output variables $(y_1, y_2, y_3, y_4)$ , and two hidden layers with three neurons each one	17
2.2	Schema of self supervised learning framework	21
2.3	Schema of semi-supervised learning framework	22
3.1	Learning curves for MLP initial model	26
3.2	Learning curves for VIME (only supervised) initial model	26
3.3	Learning curves for VIME initial model	26
3.4	Learning curves for MLP model with best hyper-parameters	28
3.5	Learning curves for VIME model with best hyper-parameters	28
3.6	Comparison between MLP and VIME (only supervised) by mean and standard deviation of MSE	29
3.7	Comparison between MLP and VIME by mean and standard deviation of MSE	29
3.8	Comparison between VIME (only supervised) and VIME by mean and standard deviation of MSE	30
3.9	Boxplot for comparison of MSE between true and predicted values across different models	30
3.10	Correlation analysis between MLP and GBLUP	33
3.11	Correlation analysis between VIME (only supervised) and GBLUP	33
3.12	Correlation analysis between VIME and GBLUP	33
5.1	Histogram of YD for traits 1 and 2	36
5.2	Histogram of YD for traits 3 and 4	36
5.3	Histogram of YD for traits 5 and 6	36
5.4	Histogram of YD for traits 7 and 8	37
5.5	Histogram of YD for traits 9 and 10	37
5.6	Histogram of YD for traits 11 and 12	37
5.7	Histogram of YD for traits 13 and 14	38
5.8	Histogram of YD for traits 15 and 16	38
5.9	Histogram of YD for traits 17 and 18	38
5.10	Histogram of YD for traits 19 and 20	39
5.11	Histogram of YD for traits 21 and 22	39
5.12	Histogram of YD for traits 23 and 24	39
5.13	Histogram of YD for traits 25 and 26	40
5.14	Histogram of YD for traits 27 and 28	40
5.15	Histogram of YD for traits 29 and 30	40
5.16	Histogram of YD for traits 31 and 32	41
5.17	Histogram of YD for trait 33	41
5.18	Learning curves for MLP across different values of the hyper-parameter learning rate	42
5.19	MSE performance for VIME (only supervised) across different values of the hyper-parameter learning rate	43

5.20	MSE performance for VIME (only supervised) across different values of the hyper-parameter $p_m$ . . . . .	43
5.21	MSE performance for VIME (only supervised) across different values of the hyper-parameter $\alpha$ . . . . .	44
5.22	MSE performance for VIME across different values of the hyper-parameter learning rate . . . . .	44
5.23	MSE performance for VIME across different values of the hyper-parameter $p_m$ . . . . .	45
5.24	MSE performance for VIME across different values of the hyper-parameter $\alpha$	45
5.25	MSE performance for VIME across different values of the hyper-parameter $\beta$	46
5.26	MSE performance for VIME across different values of the hyper-parameter $K$ . . . . .	46
5.27	Box plot for comparison of Pearson's correlation between true and predicted values across different models . . . . .	50
5.28	True vs predicted values of YD obtained by MLP model for trait 1 . . . . .	51
5.29	True vs predicted values of YD obtained by MLP model for trait 2 . . . . .	51
5.30	True vs predicted values of YD obtained by MLP model for trait 3 . . . . .	51
5.31	True vs predicted values of YD obtained by MLP model for trait 4 . . . . .	52
5.32	True vs predicted values of YD obtained by MLP model for trait 5 . . . . .	52
5.33	True vs predicted values of YD obtained by VIME (only supervised) model for trait 1 . . . . .	53
5.34	True vs predicted values of YD obtained by VIME (only supervised) model for trait 2 . . . . .	53
5.35	True vs predicted values of YD obtained by VIME (only supervised) model for trait 3 . . . . .	54
5.36	True vs predicted values of YD obtained by VIME (only supervised) model for trait 4 . . . . .	54
5.37	True vs predicted values of YD obtained by VIME (only supervised) model for trait 5 . . . . .	54
5.38	True vs predicted values of YD obtained by VIME model for trait 1 . . . . .	55
5.39	True vs predicted values of YD obtained by VIME model for trait 2 . . . . .	55
5.40	True vs predicted values of YD obtained by VIME model for trait 3 . . . . .	55
5.41	True vs predicted values of YD obtained by VIME model for trait 4 . . . . .	56
5.42	True vs predicted values of YD obtained by VIME model for trait 5 . . . . .	56

# List of Tables

3.1	ANOVA test for comparison of MSE between true and predicted values across different models . . . . .	31
3.2	Tukey's test for comparison of MSE between true and predicted values across different models . . . . .	31
5.1	MSE between true and predicted values for each trait . . . . .	47
5.2	Slope of linear regression between true and predicted values for each trait .	48
5.3	Pearson's correlation between true and predicted values for each trait . . .	49
5.4	ANOVA test for comparison of Pearson's correlation between true and predicted values across different models . . . . .	50
5.5	Tukey's test for comparison of Pearson's correlation between true and predicted values across different models . . . . .	50

# Anti-plagiarism declaration

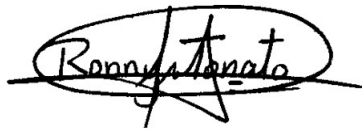
I, the undersigned Ronny Tonato, hereby certify and declare on my honour that:

- The results described in this report are the outcome of my work.
- I am the author of this report.
- I have not used third party sources or results without clearly identifying and referencing according to the recommended bibliographic rules.

## **mention to copy**

*I declare that this work can not be suspected of plagiarism.*

Signature:

A handwritten signature in black ink that reads "Ronny Tonato". The signature is written in a cursive style and is enclosed within a hand-drawn oval shape.

# Acknowledgement

First of all, I would particularly like to thank my internship supervisor, Mr Eric Barrey, for his availability, understanding, and relevant and constructive advice.

I would also like to thank my internship co-supervisor, Mr Blaise Hanczar, for his availability and wise advice.

I would like to thank my parents, Dayse, Marco and Sulay, my brothers, Diego and Jordan, my family, and all my friends in Ecuador and France who encouraged me during this year of study.

I would also like to thank the BIGE team for their welcome and their availability during my internship.

Finally, I am grateful to University Paris-Saclay, the Polytechnic Institute of Paris, and their highly skilled teachers for providing me with a solid technical background and knowledge.

# About the internship

As part of my studies at University Paris-Saclay and the Polytechnic Institute of Paris, I did a 5-month internship to end my master studies.

This report represents my work done during the internship at Animal Genetics and Integrative Biology (GABI) unit from INRAE, in Centre Île-de-France-Jouy-en-Josas. I worked in Equine Integrative Biology and Genetics (BIGE) team. Moreover, my internship tutor was Eric Barrey, a Senior Research Scientist at GABI unit and head of BIGE team, and my internship co-tutor was Blaise Hanczar, a Professor of Computer Science at IBISC laboratory from University of Evry Val d'Essonne.

GABI is a recherche unit, whose scientific interests aim at understanding and exploiting animal genetic variability to analyze the construction of phenotypes, the interactions with microbial ecosystems and more broadly with the environment, within the context of the agroecological transition. This unit is made up of more than 170-180 scientists, professors, engineers, technicians and students from both organisations and other technical and professional organisations and notably IDELE, ALLICE, IFCE, ENVA and University of Paris. GABI has several field of competences related to animal bio and genetics; which are quantitative, molecular and population genetics; functional genomics; biostatistics; bioinformatics; fundamental biology; experimentation on model and farm animals; mouse transgenesis; immunophenotyping; analysis of microbiomes and their metagenomes; integrative approaches for data analyses at all levels and scales.

On the other hand, the areas of expertise of BIGE team are quantitative genetics approach, breeding evaluation, bioinformatics, molecular genetics, functional genomics, molecular biology and integrative biology approaches of muscular exercise. For example, GWAS analyses on many different phenotypes for sports performances; genomic selection on performances; accurate genotyping; fin phenotyping techniques (gait, morphology, metabolomics, ...); analysis of microRNA and non-coding RNA; direct sequencing of mitochondrial RNA and DNA; analysis of mtDNA variability; mitochondrial functional genomics; analysis of the intestinal microbiota; integrative analysis of heterogeneous omics data.

The internship subject was part of GenIALearn project, which was supported by the Metaprogram INRAE DIGIT BIO, and where I also worked in collaboration with IBISC laboratory from University of Evry Val d'Essonne. In addition, the internship had a more applied approach, where I had to learn some notions and concepts of biology and genetics to have a better understanding of the dataset used to build the different models, which has been very helpful and enriching for me. Furthermore, I applied my skills and knowledge acquired during the two years of master to construct the different deep learning models proposed.



# Introduction

Nowadays, the use of deep learning algorithms in different areas has improved and increased considerably. As a proof of that, we can show examples such as: self-driving cars, natural language processing and speech recognition, computer vision, face detection and recognition, self-service solutions, machine translation, medical imaging, object identification, etc.

In particular, we can notice some useful advances in the area of biology and genetics, for example the alphafold project [1], which is an actually big challenge in biology and consists of the idea that a protein's structure must be determined by its amino acid sequence, that is, we have to predict the protein's structure using as input data the information related to its sequences as well as amino acid residue structure pairs.

The goal of this internship is to apply the methodology of deep learning for genomics selection. In other words, we want to use neural networks models to predict the phenotypes of dairy traits for animals using their genotyping data (SNP information). However, data used as input for the different models is given by a matrix representation of SNP information, which is too large and therefore, we are also interested in optimizing this matrix in a new space that let us to get a better representation of it.

On the first chapter, we introduce in detail the research problem of the internship, and for that, first, we talk about some notions and concepts related to genetics and genome, which gives us a better understanding about what we are going to work and delay during this work. Then, we explain the reasons why we are interested in applying deep learning to genomics selection, where we present an example of an application of deep learning in biology, which let us understand the predictive power of deep learning models. After, I introduce the objectives of the study and finally, I describe the state-of-the-art related to the different models proposed to predict phenotypes from genotyping data in recent years.

On the second chapter, I introduce the baseline model GBLUP (Genomic best linear unbiased prediction) ([2], [3] and [4]). Then, I present the MLP (Multi-layer Perceptron) [4], where GBLUP works better than MLP for this predictive task [5]. After, I introduce the VIME (Value Imputation and Mask Estimation) methodology [6] used to delay with our objective related to the optimization of the representation of genotyping data. Finally, I describe the dataset used for our study. Then, I explain the pre-processing of data done to get it ready for evaluating the performance of different models.

On the third chapter, I start describing the hyper-parameters selection for each proposed model before the training phase. Then, I evaluate the performance of the different models through the score of mean squared error (MSE) and I will compare the results obtained among the diverse models. Furthermore, the implementation of the models will be done in the programming language Python (<https://www.python.org/>), in particular, I will be working with TensorFlow (<https://www.tensorflow.org/>) and scikit-learn

(<https://scikit-learn.org/stable/>) packages.

On the last chapter, I discuss the conclusions I have reached on the basis of the results presented in the previous chapter. Additionally, I give some suggestions and recommendations about what methods and models may be tried or applied, which could let us to get a better performance regarding to our models proposed in this internship.

# Chapter 1

## Introduction to the research problem

Before starting to talk about the research problematic, we will discuss about some notions and concepts related to genetics and genome, which will help us familiarize ourselves with the terms of the problematic we are trying to face in this work.

### 1.1 Notions of genomics

Briefly, the deoxyribonucleic acid (DNA) is the support for genetic information at the nuclear and mitochondrial level in eukaryotes. A molecule of DNA is composed by two long chains of polynucleotides, where each one of these chains consists of 4 types of sub-units (nucleotides) called adenine (A), cytosine (C), guanine (G) and thymine (T). Several molecules will bind to it and transcribe this sequence into messenger RNA (called mRNA). The translation of this new mRNA makes it possible to produce proteins that will perform the various tasks necessary for the development, functioning and reproduction of living beings.

A gene is a sequence of nucleotides whose expression, i.e. transcription into mRNA and translation into protein, affects the characteristics of an organism. There are at least 22000 genes in bovines, representing only 27% of its genome. For an individual or animal, its genome is defined as the complete set of its information in the DNA. There are several versions of a gene, each version is called an allele. The genotype of an individual is the allelic composition of the genes under study. Bovines as humans beings have two copies of each gene, one from the father, one from the mother, so they have a maximum of two different alleles at the most.

The phenotype is the set of observable characters of an individual. This term sometimes refers to a single observable character such as size. In cells, DNA is collected as chromosomes. Each bovine has 30 pairs, from his parents, for a total of 3 billion pairs of DNA bases.

Simple nucleotide polymorphism: SNP is a variation on a base pair, at a specific locus, with a allelic frequency greater than 1%. For example: in a given population: For example: in a given population, on the 1st chromosome at locus N°20, 80% of the population has an A while the remaining 20% a G. The least frequent allele (G) has a frequency greater than 1%, we speak of SNP at this position, the two alleles being A or G.

## 1.2 Problem context

GWAS (Genomic wide association study) is an analysis of the statistical association between hundreds of thousands of polymorphisms (SNPs) and a phenotypic trait. These studies, which made a big leap in genetic research, were allowed by the invention of DNA genotyping microarray. Previously, this type of research was conducted on a small number of polymorphisms belonging to a small number of genes. For this purpose, they compare DNA sequences of several individuals with different phenotypes for the same trait. Then, their interest is to try to understand what influence the region (or regions) of the genome with a variant associated to the trait under study. Sometimes, the associated genes or regions have functions that are not yet known, or whose relationship with the phenotype is not obvious. Therefore, their next step is to study how these genes are involved in the phenotype.

Genomic prediction aims to predict the phenotype of an individual using their genotype information. We can find several applications about it in literature: we can try to predict the risk of disease in human beings [7]; while in plants and animals, they seek for the genetic value of the individual [8], that is, the probability of obtaining offspring with interesting phenotypic traits for breeding, for example.

By the way, deep learning methods are a class of machine learning techniques capable of identifying highly complex patterns in large dataset [9]. These kinds of models, both supervised and unsupervised, have found various applications in areas such as biology and genomics: for example, these models can be used as a fully data-driven refinement of bioinformatics tools, to predict the effect of non-coding variants or to get richer representations to reveal the structure of high-dimensional data [10]. We can also apply deep learning methods for predicting the sequence specificity of DNA- and RNA-binding proteins and of enhancer and cis-regulatory regions, methylation status, gene expression and control of splicing [9].

In particular, we are interested in the application of deep learning models for genomic selection because these models can capture non-linear genetic components and make multi-traits prediction, which can not be approached by classic statistical methods used to solve this task. Would it then be possible to predict multi-(diary)traits of animals from their genotype?

## 1.3 State-of-the-art

The best models of genomic predictions at present are statistical methods such as GBLUP (Genomic best unbiased linear prediction) and Bayesian approaches, which are very efficient for genomic selection on unique quantitative trait or few traits genetically related. However, these models can not consider massive and heterogeneous data, non-linear genetic components, and multi-traits prediction.

A multilayer perceptron (MLP) is one of the basic deep learning models, which will be seen in Chapter 2. This model is an artificial neuronal network which is really suitable for genomic selection because it can explain non-linear genetic components and make multi-traits prediction from genotype data.

Montesinos-López et al. [5] made the comparison of 3 deep learning models: MLP, CNN (convolutional neural network), LCNN (local convolutional neural network) against 3 baseline (or classic) models: GBLUP, BayesA and EGBLUP (extend GBLUP) to analyze their predictive ability on different traits for the simulated and real Arabidopsis dataset.

The prediction performance was measured in terms of Pearson’s correlation. The results showed that there is no clear superiority of deep learning in terms of prediction power compared to conventional genome-based prediction models. Nevertheless, there is clear evidence that deep learning algorithms capture non-linear patterns more efficiently than conventional genome based [5].

van Hilten et al. [11] proposed a new open-source deep learning framework known as GenNet for predicting phenotypes from genetic data. In this model, interpretable and memory-efficient neural networks are constructed by incorporating prior biological knowledge. The effectiveness of GenNet is demonstrated across multiple datasets and for multiple phenotypes.

Fumeron [12] did his M2 internship at BIGE team with collaboration of IBISC laboratory last year related to the prediction of the phenotypes (sport performance) from genotyping data of horses. He realized that adding pedigree information to the MLP model (known as MLP5), it helps to improve a little the model performance regarding to the model without it. However, the prediction performance of this model does not work better than baseline models in all phenotypes, which is not enough to guarantee that deep learning models work better than classical statistical methods.

DeepNull is a new architecture proposed et implemented by Zachary et al. [13], which combines the non-linear effects of covariates measured by a MLP model and the additive effects given by a linear model (it can even be a GWAS additive model). DeepNull is compared against a standard GWAS as baseline model on ten phenotypes from the UK BioBank, which is measured by the Pearson’s correlation (squared) between true and predicted value of phenotypes. The results showed that DeepNull improves phenotype prediction compared to Baseline because of potential covariate interactions or higher order terms can be easily include in DeepNull, which does not always happen with Baseline model. So, DeepNull is one of the best deep learning models to predict phenotypes from genotype data for now.

## 1.4 Hypothesis

The classical statistical methods for genomic prediction only take into account additive genetic effects, and as we will see later these models need the calculation of a genetic relation matrix only if we study several traits at the same time, and which is computationally expensive. In addition, these models do not allow predictions of multiple phenotypes at once and are not useful for capturing the nonlinear effects that are present in most cases. Therefore, the deep learning models to be developed will be under the assumption that there is the presence of additive genetic effects, and they will also be used for multi-traits prediction, that is, the estimation of performance for several phenotypes at once.

## 1.5 Objectives

As I have explained before, the research problem consists of genomic selection for animals, that is, we want to predict certain phenotypes of animals from their genotyping data by using deep learning architectures. So, in this work we propose to accomplish the following objectives:

- Predict the performance for phenotypes (dairy) of animals from their genotyping data.

- Optimize the representation of genotyping data (SNP information).

The first objective we propose to achieve is related to the main application of these models where the dataset used corresponds to the only domestic animals with massive data base of genotypes and traits of production recorded on a large population of most 100.000 animals. However, the second one is the most important objective because as we will see later, the input data is a large matrix, which is computationally expensive and therefore we will try to optimize this matrix before training (pre-processing) a neural network to make the best use of the linear and non-linear genetic information contained in the medium to high density genotyping data.

## Chapter 2

# Material and methods

In this section I describe the material and methods used to try to solve and achieve our objectives, i.e., the algorithms or models and the dataset. First, I describe the dataset used for evaluating our proposed models. Then, I introduce different models for achieving our problem. Finally, I explain how the data is divided into training, validation and testing sets for using the different models and obtaining the results to be seen in the next chapter.

### 2.1 Dataset

First of all, I must clarify the dataset used in this work is confidential and it was proportioned by GenEval (Genetic Evaluation of farm animals), which is INRAE's partner. Therefore, the partial or total distribution of data is prohibited.

#### 2.1.1 Data description

The dataset used for this study consists of information about 113599 Holstein breed female cows. As **genotype data**, I have the SNP information, which is described by a large matrix. For each animal, I have its ID and 53469 joined values, where the  $i$ th-value corresponds to the number of alleles (2) carried by the individual to the  $i$ th-variant and it's encoded as follows:

- 0 for homozygote 1
- 1 for heterozygote
- 2 for homozygote 2

Additionally, I also have the information on the chromosome and position of the  $i$ th-variant regard to the SNP information. A brief representation of all these data could be appreciated in the annexes.

As **phenotype data**, I will work with the performance of 35 dairy traits measured by the variable Yield Deviation, and its construction is detailed in the annexes. Each trait (encoded by CAR: caractère, trait in English) is described as follows:

- CAR 1 to 5: 5 dairy production traits.
- CAR 6: somatic cell score.
- CAR 7 to 12: 6 fertility traits.
- CAR 13 to 33: 21 dairy morphology traits.
- CAR 34: occurrence of clinical mastitis before the 150th day of lactation.

- CAR 35: functional longevity.

In addition, I also have the weights associated to the variable Yield Deviation ( $w\_YD$ ) for each trait, and its construction is also explained in the annexes.

For each animal, I have its ID, status (what type of population it represents) values of the variables YD and  $w\_YD$  for each trait.

### 2.1.2 Data pre-processing

First, I decided to pre-process all available information for the present study to deal with missing data, inappropriate format of data or wrong data type.

So, I started analyzing whether or not there are missing information regarding to the variables YD and  $w\_YD$  for all traits. Then, I computed and observed that for the trait 6 (CAR 6) I have two individuals with missing values for YD and  $w\_YD$ . Moreover, I decided to delete these two animals and its values associated for all traits.

For the trait 34 (CAR 34), I observe there are 29294 animals with missing values of both variables, which corresponds to 25.79% of all individuals. Furthermore, I also saw there are 4968 animals with missing values of both variables for the trait 35 (CAR 35), and which corresponds to 4.37% of all population considered for the study.

I therefore decided not to work with these two traits since it is not possible to infer the missing data by the nature of the information received, i.e., I do not take these two traits into account for the current study.

On the other hand, I also identified 12 individuals with a few missing typing in the SNP information. So, I decided to delete all values from the SNP information associated to these individuals and also that of the two individuals found in the analysis of the phenotype data.

Afterwards, I try to the separate all joined values (53469 data) for each animal to get a matrix representation of SNP information available. In the end, I have 113585 animals with their genotype and phenotype data available for the present work.

### 2.1.3 Descriptive statistics

Now, I study the behavior of Yield deviation (YD) for traits 1 to 33 in order to identify which probability distribution the data are fitted to. For this purpose, I calculate the histogram of YD for each trait, which can be seen in Figures 5.1 to 5.17.

The results show that the distribution of adjusted performance (YD) for traits 7, 8, 10 and 11 is not normally distributed, which is explained by the fact that these traits are binary traits. In addition, traits 13 and 21 are discrete traits with 5 modalities whose distribution is not normal (class 3 over-represented in the distribution of raw performance compared to a normal distribution), which explains a non-normal distribution of adjusted performance (YD).

The distribution of YD for other traits seems to be normal. I also verified all these assumptions by a Jarque-Bera test, which confirmed my results.

So, I decide to normalize the variable YD for all traits to have a better appreciation in terms of numerical value with respect to the analysis of results, which will be seen later in Chapter 3.



## 2.2 Models

### 2.2.1 Genomic Best Linear Unbiased Prediction (GBLUP)

Genomic best linear unbiased prediction (GBLUP) is a method that utilizes genomic relationships to estimate the genetic merit of an individual (or animal). To get this purpose, I use a genomic relationship matrix (GRM), which is estimated from DNA marker information. This matrix defines the covariance between individuals based on observed similarity at the genomic level, rather than on expected similarity based on pedigree, so that more accurate predictions of merit can be made.

The GBLUP method was introduced by VanRaden [2] and Habier et al. [3]. In practice, we use the following linear mixed model to compute the GBLUP:

$$\mathbf{Y} = \mathbf{X}\boldsymbol{\beta} + \mathbf{Z}\mathbf{u} + \boldsymbol{\varepsilon}, \quad (2.1)$$

where  $\mathbf{Y}$  is the vector of response variables (phenotypes) of order  $n \times 1$ ,  $\mathbf{X}$  is the design matrix of fixed effects of order  $n \times p$ ,  $\boldsymbol{\beta}$  is the vector of fixed effects of order  $p \times 1$ ,  $\mathbf{Z}$  is the design matrix of random effects (allocating records to genetic values) of order  $n \times q$ ,  $\mathbf{u}$  is the vector of random effects (or additive genetic effects for an individual) and  $\boldsymbol{\varepsilon}$  is a vector of residuals distributed as  $\mathcal{N}(0, \mathbf{R})$ , where  $\mathbf{R}$  is a variance-covariance matrix of residual effects of dimension  $n \times n$ . Furthermore, we have  $\text{Var}(\mathbf{u}) = \mathbf{G}\sigma_u^2$  where  $\mathbf{G}$  is the genomic relationship matrix, and  $\sigma_u^2$  is the genetic variance for this model.

A solution to estimate jointly parameters  $\hat{\boldsymbol{\beta}}$  and  $\hat{\mathbf{u}}$  from (2.1) is given for solving the following mixed model equation

$$\begin{pmatrix} \mathbf{X}^\top \mathbf{R}^{-1} \mathbf{X} & \mathbf{X}^\top \mathbf{R}^{-1} \mathbf{Z} \\ \mathbf{Z}^\top \mathbf{R}^{-1} \mathbf{X} & \mathbf{Z}^\top \mathbf{R}^{-1} \mathbf{Z} + \sigma_u^{-2} \mathbf{G}^{-1} \end{pmatrix} \begin{pmatrix} \hat{\boldsymbol{\beta}} \\ \hat{\mathbf{u}} \end{pmatrix} = \begin{pmatrix} \mathbf{X}^\top \mathbf{R}^{-1} \mathbf{y} \\ \mathbf{Z}^\top \mathbf{R}^{-1} \mathbf{y} \end{pmatrix}, \quad (2.2)$$

where the solution obtained from (2.2) for  $\hat{\mathbf{u}}$  is known as GBLUP.

So, the key to this method is using a well-estimated matrix  $\mathbf{G}$  to solve (2.2). To do that, I present below some methods introduced by VanRaden [2] in his paper to compute the matrix  $\mathbf{G}$  and which are illustrated in the book of Montesinos-Lopez et al. [4].

1. The first method computes the GRM as

$$\mathbf{G} = \frac{\mathbf{X}\mathbf{X}^\top}{p}, \quad (2.3)$$

where  $\mathbf{X}$  is the matrix of marker genotypes of dimensions  $J \times p$  coded by

$$x = \begin{cases} 0 & \text{if the SNP is homozygous for the major allele} \\ 1 & \text{if the SNP is heterozygous} \\ 2 & \text{if the SNP is homozygous for the other allele} \end{cases}$$

2. The second method is similar to the first one, but first each marker is centered by twice the minor allele frequency:

$$\mathbf{G} = \frac{(\mathbf{X} - \boldsymbol{\mu}_E)(\mathbf{X} - \boldsymbol{\mu}_E)^\top}{2 \sum_{j=1}^p p_j(1 - p_j)}, \quad (2.4)$$

where  $p_j$  is the minor allele frequency of SNP  $j = 1, \dots, p$  and  $\boldsymbol{\mu}_E$  the expected value of matrix  $\mathbf{X}$  under the Hardy–Weinberg equilibrium [14], from estimates of allelic frequencies, that is,  $\boldsymbol{\mu}_E = \mathbf{1}_J[2p_1, \dots, 2p_p]$ . The term  $2 \sum_{j=1}^p p_j(1 - p_j)$  is the sum of the variance estimates of each marker and makes GRM analogous to the numerator relationship matrix [2].

3. In the third method, the GRM should be calculated as follow

$$\mathbf{G} = \frac{\mathbf{Z}\mathbf{Z}^\top}{p}, \quad (2.5)$$

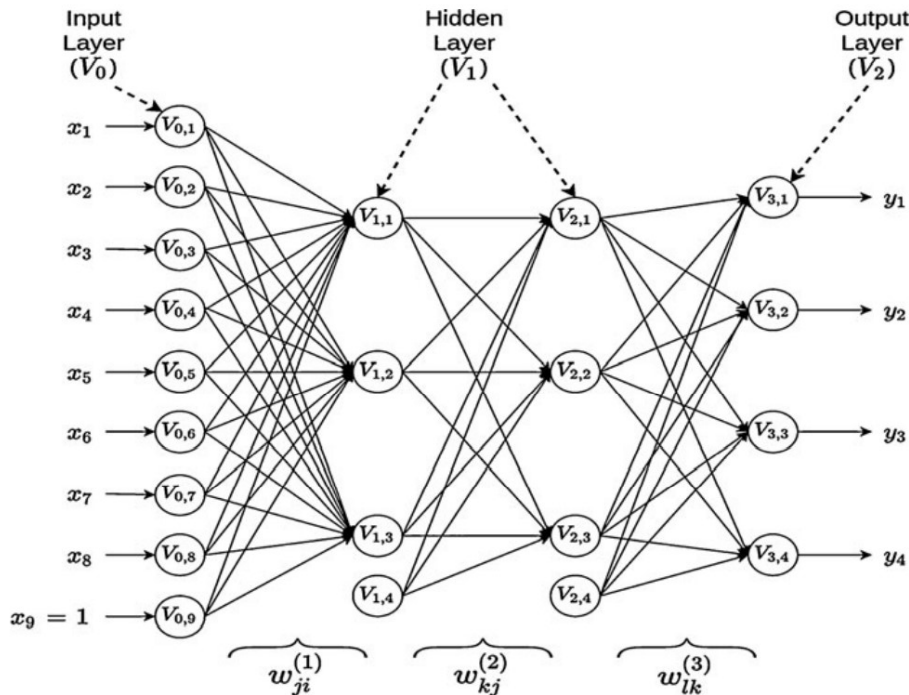
where  $\mathbf{Z}$  is the matrix of scaled SNP codes and  $p$  is the number of SNPs, that is  $z_{ij} = (x_{ij} - 2p_j)\sqrt{2p_j(1-p_j)}$ .

### 2.2.2 Multilayer Perceptron (MLP)

A multilayer perceptron (MLP) is an artificial neural network (ANN), which is composed of input layers, (2 or more) hidden layers and output layers where each layer has a specific number of nodes (or neurons). The MLP is a fully connected feedforward neural network because current neurons from each layer (starting from the first hidden layer) are connected with all neurons from the previous layer (starting from the input layer) until the output layer (i.e. fully connected) and the information transmitted from the input layer through hidden layers to the output layer travels in only one direction, in this case forward (feedforward).

In Figure 2.1 we provide an example of this type of neural networks to get a better understanding of the main components used to construct MLP models. In this neuronal network we have that

- $x_1, \dots, x_8$  represent the information received as input for the model.
- $w_{j,i}^{(1)}, w_{k,j}^{(2)}, w_{l,k}^{(3)}$  with  $i = 1, \dots, 9$ ;  $j, k = 1, \dots, 3$ ;  $l = 1, \dots, 4$  are the weights that modifies the received information from the input and can be interpreted as gains that can attenuate or amplify the values that they wish to propagate toward the neuron [4].
- $w_{j,9}^{(1)}, w_{k,4}^{(2)}, w_{l,4}^{(3)}$  with  $j, k = 1, \dots, 3$ ;  $l = 1, \dots, 4$  are the weights associated to  $x_9, V_{1,4}$  and  $V_{2,4}$ , respectively, which are known as the term intercept or bias.
- $y_1, \dots, y_4$  are the outputs of the neural network.



**Figure 2.1:** An example of MLP with eight input variables ( $x_1, \dots, x_8$ ), four output variables ( $y_1, y_2, y_3, y_4$ ), and two hidden layers with three neurons each one

In the following equations, I describe the analytical form of the model given in Figure 2.1 for  $N$  outputs neurons, with  $D$  inputs neurons,  $M_1$  hidden neurons (units) in hidden layer 1 and  $M_2$  hidden units in hidden layer 2:

$$V_{1,j} = g_1 \left( \sum_{i=1}^D w_{ji}^{(1)} x_i \right) \quad \text{for } j = 1, \dots, M_1 \quad (2.6)$$

$$V_{2,k} = g_2 \left( \sum_{j=1}^{M_1} w_{kj}^{(2)} V_{1,j} \right) \quad \text{for } k = 1, \dots, M_2 \quad (2.7)$$

$$y_l = g_3 \left( \sum_{k=1}^{M_2} w_{lk}^{(3)} V_{2,k} \right) \quad \text{for } l = 1, \dots, N \quad (2.8)$$

where (2.6) gives the output of each of the neurons in the first hidden layer, (2.7) produces the output of each of the neurons in the second hidden layer, and (2.8) produces the output of each response variable of interest. The coefficients  $b_{1,j}$ ,  $b_{2,k}$ , and  $b_{3,l}$  (or weights associated to  $V_{0,9}$ ,  $V_{1,4}$  and  $V_{2,4}$ , respectively) are the term intercept for the hidden layers 1, 2, and the output layer, respectively. The weights  $w_{j,i}^{(1)}$ ,  $w_{k,j}^{(2)}$ ,  $w_{l,k}^{(3)}$  are used to the learning process of the model,  $g_1$ ,  $g_2$  and  $g_3$  are the activation functions in hidden layers 1, 2, and the output layer, respectively.

As we have seen, the activation functions are used to propagate the output of one layer's nodes forward to the next layer (up to and including the output layer) [4]. The most common functions are sigmoid ( $\sigma$ ), softmax, tanh and ReLU (rectified linear unit).

$$\sigma(z) = \frac{1}{1 + e^{-z}}, \quad \tanh(z) = \frac{e^{2z} - 1}{e^{2z} + 1}, \quad \text{ReLU}(z) = \max(0, z),$$

$$\text{softmax}(z_i) = \frac{e^{z_i}}{1 + \sum_{j=1}^C e^{z_j}}, \quad \text{for } i = 1, \dots, C.$$

In neural networks, it's usually common to use a ReLU function as activation function in the output of each hidden layer. Depending on the problem or task to solve, we can apply an identity function (regression problem), sigmoid function (binary classification problem) or softmax function (multiclass classification problem) for the output layer.

Considering our previous example, the problem to solve is explained as follows: given a training data  $(x_1, y_1, x_2, y_2, \dots, x_m, y_m)$ , we want to find the parameters

$$\theta = (w_{j,i}^{(1)}, w_{k,j}^{(2)}, w_{l,k}^{(3)}, b_{1,j}, b_{2,k}, b_{3,l})$$

that minimizes the empirical loss  $J(\theta)$ :

$$J(\theta) = \frac{1}{m} \sum_{i=1}^m \mathcal{L}(\hat{y}_i, y_i)$$

where  $\hat{y}_i$  is the prediction of the model obtained by (2.8) and  $\mathcal{L}$  is some standard loss function, in our case  $\mathcal{L}$  is the Mean Squared Error (MSE):

$$\mathcal{L}(\hat{y}, y) = \frac{1}{2} (\hat{y} - y)^2 \quad \text{for } y \in \mathbb{R}.$$

The training process of the neural network follows the next steps:

- For an initialization of weights value, the loss and predictions are calculated by forward-propagation.

- We calculate the gradients of  $J(\theta)$  with respect to the parameters  $\theta$  through the back-propagation algorithm, which let us to compute the chain-rule by storing intermediate (and re-use derivatives).
- We apply stochastic gradient descent methods (as gradient descent) to update the parameters  $\theta$  using a fixed learning rate.

By the way, during the training phase there could be some problems such as: overfitting (or high variance), underfitting, vanishing (or exploding) gradient, covariate shift, etc. Therefore, we suggest some alternatives to deal with these problems:

- To avoid the poor convergence during the updating of parameters, we apply the first-order method Adam on mini batches of training set.
- To deal with the problem of the fastest the gradient diminishes/vanishes during backpropagation and therefore the network could stop learning, we initialize the weights randomly, use a ReLU function as activation function, and apply a dropout regularization.
- To avoid the covariate shift and allow to learn the network faster, we apply a batch normalization.

## 2.2.3 Value Imputation and Mask Estimation (VIME)

Deep learning models via supervised learning on large labeled datasets have showed several successes in a variety of applications (such as image classification [15], object detection [16], and language translation [17]). However, the collection of large labeled datasets is really expensive and even impossible in numerous domains. In these settings, even tough, there is often an abundance of unlabeled data available - datasets are often collected from a large population, but target labels are only available for a small group of people [6]. These datasets offer huge opportunities for self- and semi-supervised learning algorithms, which can take advantage of the unlabeled data to further improve the performance of a predictive model. Unfortunately, actual self- and semi-supervised learning algorithms are ineffective for tabular data because they heavily rely on the spatial or semantic structure of image or language data. So, I am going to present the systematic self- and semi-supervised learning framework for tabular data proposed by Yoon et al. [6], which they refer collectively as VIME (Value Imputation and Mask Estimation).

### 2.2.3.1 Problem formulation

First, I am going to introduce the general formulation of self- and semi-supervised learning presented by Yoon et al. [6]. We consider a small labeled dataset  $\mathcal{D}_l = \{\mathbf{x}_i, y_i\}_{i=1}^{N_l}$  and a large unlabeled dataset  $\mathcal{D}_u = \{\mathbf{x}_i\}_{i=1}^{N_l+N_u}$  where  $N_u \gg N_l$ ,  $\mathbf{x}_i \in \mathcal{X} \subseteq \mathbb{R}^d$  and  $y_i \in \mathcal{Y}$ . We assume every input feature  $\mathbf{x}_i$  in  $\mathcal{D}_l$  and  $\mathcal{D}_u$  is sampled i.i.d. from a feature distribution  $p_X$ , and the labeled data pairs  $(\mathbf{x}_i, y_i)$  in  $\mathcal{D}_l$  are drawn from a joint distribution  $p_{X,Y}$ . When only limited labeled samples from  $p_{X,Y}$  are available, a predictive model  $f : \mathcal{X} \rightarrow \mathcal{Y}$  solely trained by supervised learning is likely to overfit the training samples since the empirical supervised loss  $\sum_{i=1}^{N_l} l(f(\mathbf{x}_i), y_i)$  we minimize deviates significantly from the expected supervised loss  $\mathbb{E}_{(\mathbf{x},y) \sim p_{X,Y}} [l(f(\mathbf{x}), y)]$  where  $l(\cdot, \cdot)$  is some standard supervised loss function.

### Self supervised learning

In general, self-supervised learning constructs an encoder function  $e : \mathcal{X} \rightarrow \mathcal{Z}$  that takes a sample  $\mathbf{x} \in \mathcal{X}$  and returns an informative representation  $\mathbf{z} = e(\mathbf{x}) \in \mathcal{Z}$ . The representation

$\mathbf{z}$  is optimized to solve a pretext task defined with a pseudo-label  $y_s \in \mathcal{Y}_s$  and a self-supervised loss function  $l_{ss}$ .

We define the pretext predictive model as  $h : \mathcal{Z} \rightarrow \mathcal{Y}_s$ , which is trained jointly with the encoder function  $e$  by minimizing the expected self-supervised loss function  $l_{ss}$  as follows

$$\min_{e,h} \mathbb{E}_{(\mathbf{x}_s, y_s) \sim p_{X_s, Y_s}} [l_{ss}(y_s, (h \circ e)(\mathbf{x}_s))] \quad (2.9)$$

where  $p_{X_s, Y_s}$  is a pretext distribution that generates pseudo-labeled samples  $(\mathbf{x}_s, y_s)$  for training the encoder  $e$  and pretext predictive model  $h$ .

### Semi-supervised learning

Semi-supervised learning optimizes the predictive model  $f$  by minimizing the supervised loss function jointly with some unsupervised loss function defined over the output space  $\mathcal{Y}$ . Formally, semi-supervised learning is formulated as an optimization problem as follows

$$\min_f \mathbb{E}_{(\mathbf{x}, y) \sim p_{X, Y}} [l(y, f(\mathbf{x}))] + \beta \cdot \mathbb{E}_{\mathbf{x} \sim p_X, \mathbf{x}' \sim \tilde{p}_X(\mathbf{x}'|\mathbf{x})} [l_u(f(\mathbf{x}), f(\mathbf{x}'))] \quad (2.10)$$

where  $l_u : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}$  is an unsupervised loss function, and a hyper-parameter  $\beta \geq 0$  is introduced to control the trade-off between the supervised and unsupervised losses.  $\mathbf{x}'$  is a perturbed version of  $\mathbf{x}$  assumed to be drawn from a conditional distribution  $\tilde{p}_X(\mathbf{x}'|\mathbf{x})$ .

Now, I am going to present the framework developed by Yoon et al. [6] for self- and semi-supervised learning, and which are used on this work.

#### 2.2.3.2 Self supervised learning approach

The idea of this framework is to introduce two pretext tasks: *feature vector estimation* and *mask vector estimation*. The goal is to optimize a pretext model to recover an input sample (a feature vector) from its corrupted variant, at the same time as estimating the mask vector that has been applied to the sample.

In this framework, the two pretext tasks share a single pretext distribution  $p_{X_s, Y_s}$ . First, a mask vector generator outputs a binary mask vector  $\mathbf{m} = [m_1, \dots, m_d]^\top \in \{0, 1\}^d$  where  $m_j$  is randomly sampled from a Bernoulli distribution with probability  $p_m$  (i.e.  $p_{\mathbf{m}} = \prod_{j=1}^d \text{Bern}(m_j|p_m)$ ). Then a pretext generator  $g_m : \mathcal{X} \times \{0, 1\}^d \rightarrow \mathcal{X}$  takes a sample  $\mathbf{x}$  from  $\mathcal{D}_u$  and a mask vector  $\mathbf{m}$  as input, and generates a masked sample  $\tilde{\mathbf{x}}$ . The generating process of  $\tilde{\mathbf{x}}$  is given by

$$\tilde{\mathbf{x}} = g_m(\mathbf{x}, \mathbf{m}) = \mathbf{m} \odot \bar{\mathbf{x}} + (1 - \mathbf{m}) \odot \mathbf{x} \quad (2.11)$$

where the  $j$ -th feature of  $\bar{\mathbf{x}}$  is sampled from the empirical distribution

$$\hat{p}_{X_j} = \frac{1}{N_u} \sum_{i=N_l+1}^{N_l+N_u} \delta(x_j = x_{i,j}), \quad (2.12)$$

and where  $x_{i,j}$  is the  $j$ -th feature of the  $i$ -th sample in  $\mathcal{D}_u$ .

Following the convention of self-supervised learning, the encoder  $e$  first transforms the masked and corrupted sample  $\tilde{\mathbf{x}}$  to a representation  $\mathbf{z}$ , then a pretext predictive model will be introduced to recover the original sample  $\mathbf{x}$  from  $\mathbf{z}$ . The resulting sample  $\tilde{\mathbf{x}}$  may not contain any information about the missing features and even hard to identify which features are missing. To solve such a challenging task, we first divide it into two sub-tasks (pretext tasks):

1. *Mask vector estimation*: predict *which* features have been masked;
2. *Feature vector estimation*: predict the values of the features that have been corrupted.

Then, a separate pretext predictive model is introduced for each pretext task. Both models operate on top of the representation  $\mathbf{z}$  given by the encoder  $e$  and try to estimate  $\mathbf{m}$  and  $\mathbf{x}$  collaboratively. The two models and their functions are,

- **Mask vector estimator**,  $s_m : \mathcal{Z} \rightarrow [0, 1]^d$ , takes  $\mathbf{z}$  as input and outputs a vector  $\hat{\mathbf{m}}$  to predict which features of  $\tilde{\mathbf{x}}$  have been replaced by a noisy counterpart (i.e.,  $\mathbf{m}$ );
- **Feature vector estimator**,  $s_r : \mathcal{Z} \rightarrow \mathcal{X}$ , takes  $\mathbf{z}$  as input and returns  $\hat{\mathbf{x}}$ , an estimate of the original sample  $\mathbf{x}$ .

The encoder  $e$  and the pretext predictive models (in this case, the two estimators  $s_m$  and  $s_r$ ) are trained jointly in the following optimization problem,

$$\min_{e, s_m, s_r} \mathbb{E}_{\mathbf{x} \sim p_X, \mathbf{m} \sim p_{\mathbf{m}}, \tilde{\mathbf{x}} \sim g_{\mathbf{m}}(\mathbf{x}, \mathbf{m})} [l_m(\mathbf{m}, \hat{\mathbf{m}}) + \alpha \cdot l_r(\mathbf{x}, \hat{\mathbf{x}})] \quad (2.13)$$

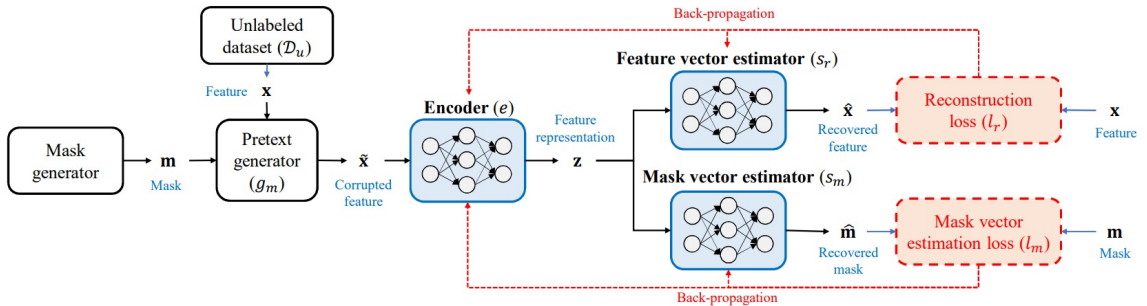
where  $\mathbf{m} = (s_m \circ e)(\tilde{\mathbf{x}})$  and  $\hat{\mathbf{x}} = (s_r \circ e)(\tilde{\mathbf{x}})$ . In addition, we can notice that the self-supervised loss function  $l_{ss}$  (given in Equation (2.9)) is decomposed as the sum of two losses, where the first loss function  $l_m$  is the sum of the binary cross-entropy losses for each dimension of the mask vector

$$l_m(\mathbf{m}, \hat{\mathbf{m}}) = -\frac{1}{d} \left[ \sum_{j=1}^d m_j \log [(s_m \circ e)_j(\tilde{\mathbf{x}})] + (1 - m_j) \log [1 - (s_m \circ e)_j(\tilde{\mathbf{x}})] \right], \quad (2.14)$$

and the second loss function  $l_r$  is the reconstruction loss, given by

$$l_r(\mathbf{x}, \hat{\mathbf{x}}) = \frac{1}{d} \left[ \sum_{j=1}^d (x_j - (s_r \circ e)_j(\tilde{\mathbf{x}}))^2 \right]. \quad (2.15)$$

The trade-off between the two losses is adjusted by the hyperparameter  $\alpha$ . The entire self-supervised learning framework introduced by Yoon et al. [6] is illustrated in Figure 2.2.



**Figure 2.2:** Schema of self supervised learning framework

### 2.2.3.3 Semi-supervised learning approach

Now, I show how the encoder function  $e$  from the previous sub-subsection can be used in semi-supervised learning. Let  $f_e = f \circ e$  and  $\hat{y} = f_e(\mathbf{x})$ . The predictive model  $f$  is trained by minimizing the objective function,

$$\mathcal{L}_{final} = \mathcal{L}_s + \beta \cdot \mathcal{L}_u. \quad (2.16)$$

The supervised loss  $\mathcal{L}_s$  is given by

$$\mathcal{L}_s = \mathbb{E}_{(\mathbf{x}, y) \sim p_{XY}} [l_s(y, f_e(\mathbf{x}))] \quad (2.17)$$

where  $l_s$  is the standard supervised loss function, in our case  $l_s$  is the mean squared error. The unsupervised (consistency) loss  $\mathcal{L}_u$  is defined between original samples ( $\mathbf{x}$ ) and their reconstructions from corrupted and masked samples ( $\tilde{\mathbf{x}}$ ),

$$\mathcal{L}_u = \mathbb{E}_{\mathbf{x} \sim p_X, \mathbf{m} \sim p_{\mathbf{m}}, \tilde{\mathbf{x}} \sim g_m(\mathbf{x}, \mathbf{m})} [(f_e(\tilde{\mathbf{x}}) - f_e(\mathbf{x}))^2] \quad (2.18)$$

This consistency loss is inspired by the idea in consistency regularizer: encouraging the predictive model  $f$  to return the similar output distribution when its inputs are perturbed. However, the perturbation in this framework is learned through the self-supervised framework while in the previous works, the perturbation is from a manually chosen distribution, such as rotation.

For a fixed sample  $\mathbf{x}$ , the inner expectation in Equation (2.18) is taken with respect to  $p_{\mathbf{m}}$  and  $g_m(\mathbf{x}, \mathbf{m})$  and could be interpreted as the variance of the predictions of corrupted and masked samples.  $\beta$  is another hyperparameter to adjust the supervised loss  $\mathcal{L}_s$  and the consistency loss  $\mathcal{L}_u$ . In each iteration of training, for each sample  $\mathbf{x} \in \mathcal{D}_u$  in the batch, we create  $K$  augmented samples  $\tilde{\mathbf{x}}_1, \dots, \tilde{\mathbf{x}}_K$  by repeating the operation in Equation (3)  $K$  times. Every time the sample  $\mathbf{x} \in \mathcal{D}_u$  is used in a batch, these augmented samples are recreated. Then, the stochastic approximation of  $\mathcal{L}_u$  is given as

$$\hat{\mathcal{L}}_u = \frac{1}{N_b K} \sum_{i=1}^{N_b} \sum_{k=1}^K [(f_e(\tilde{\mathbf{x}}_{i,k}) - f_e(\mathbf{x}_i))^2] = \frac{1}{N_b K} \sum_{i=1}^{N_b} \sum_{k=1}^K [(f(\tilde{\mathbf{z}}_{i,k}) - f(\mathbf{z}_i))^2] \quad (2.19)$$

where  $N_b$  is the batch size. During training, the predictive model  $f$  is regularized to make similar predictions on  $\mathbf{z}_i$  and  $\mathbf{z}_{i,k}, k = 1, \dots, K$ . After training  $f$ , the output for a new test sample  $\mathbf{x}^t$  is given by  $\hat{y} = f_e(\mathbf{x}^t)$ . The entire semi-supervised learning framework introduced by Yoon et al. [6] is illustrated in Figure 2.3.

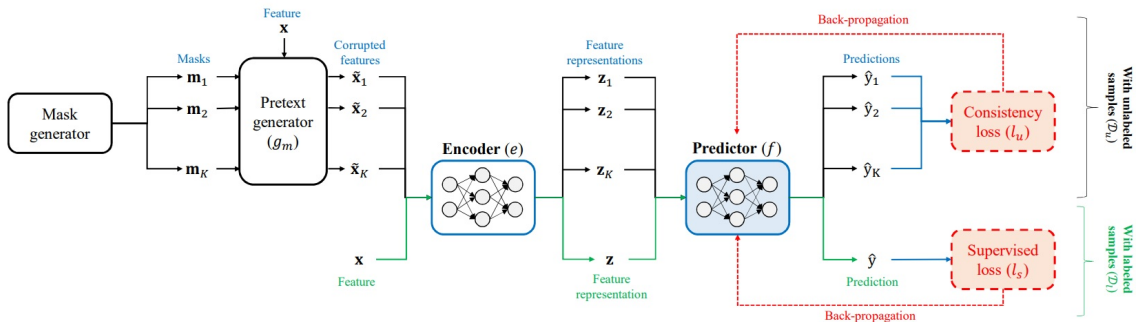


Figure 2.3: Schema of semi-supervised learning framework

## 2.3 Training, validation and test sets

As I said before, there are 113585 animals with their genotyping and phenotype data available for the present work. These 113585 female animals, regarding to the available phenotype data, were separated into

- a learning population (status = "A")
- a first validation population for selecting the best hyperparameters (status = "B")
- a second validation population (= "test") for testing the model chosen with the best hyperparameters (status = "V")

So, I also apply this split to the genotyping data by identifying the animals' ID from each status of the phenotype data (training set: "A", validation set: "B", testing set: "V"). According to this division, we get a training data set of 93484 animals, a validation data set of 10086 animals and a test data set of 10015 animals, which will be used to train, valid and evaluate the different models.



# Chapter 3

## Results and discussion

### 3.1 Model training

Before starting the training of the models, I must choose the hyper-parameters and parameters to initialize them. The initial architecture for the MLP and VIME models is presented below.

I should also clarify that the calculation of YD predictions for each trait by a single-trait GBLUP model was performed separately and the comparison of its results with the other models will be shown later.

#### 3.1.1 Hyper-parameters initialization

##### 3.1.1.1 MLP model

The initial architecture of MLP model is described as follow:

- 3 hidden layers: 1000/500/500 neurons, activation function: ReLU function
- Batch Normalization layer
- Dropout layers: 0.3/0.3/0.3 rate.
- Output layer of 33 neurons with Identity function as activation function
- Optimizer Adam with learning rate  $lr = 10^{-3}$ . Batch size: 500. Epochs: 100.
- Loss function: Mean Squared Error (MSE)

##### 3.1.1.2 VIME model

For the VIME model, first I take 35000 samples from training dataset as unlabeled samples and the rest as labeled samples (58484 samples). Then, I introduce the architecture of the self-supervised framework used to train the encoder:

- **Encoder:**
  - 3 hidden layers: 1000/500/500 neurons, activation function: ReLU
  - Batch Normalization layer
  - Dropout layer
- **Feature estimator:**
  - 1 hidden layer of 10000 neurons.
  - 1 output layer of 53469 neurons.

- A ReLU function as activation function for the hidden and output layers.
- **Mask estimator:**
  - 1 hidden layer of 10000 neurons with ReLU function as activation function.
  - 1 output layer of 53469 neurons with Sigmoid function as activation function.
- Optimizer Adam with learning rate  $lr = 10^{-3}$ . Batch size: 500. Epochs: 30.
- $\alpha = 1$
- $p_m = 0.2$

For the semi self-supervised framework, we use the encoder architecture already trained on the self-supervised framework with 35000 unlabeled samples.

Initially, I remove the part of the calculation of consistency loss ( $l_u$ ) with respect to unlabeled samples on the semi-supervised framework for now. This model will be called VIME (only supervised).

Therefore, I want to compare the prediction performance by including or not including the unsupervised part on the semi-supervised framework.

The semi self-supervised framework for VIME (only supervised) is described as

- **Predictor:**
  - 1 output layer of 33 neurons with Identity function as activation function
- Supervised loss ( $l_s$ ) is given by MSE.
- Optimizer Adam with learning rate  $lr = 10^{-3}$ . Batch size: 500. Epochs: 20.

The semi self-supervised framework for VIME complete is similar to VIME (only supervised) and is described as

- **Predictor:**
  - 1 output layer of 33 neurons with Identity function as activation function
- $K = 3$
- $\beta = 1$
- Optimizer Adam with learning rate  $lr = 10^{-3}$ . Batch size: 500. Epochs: 15.

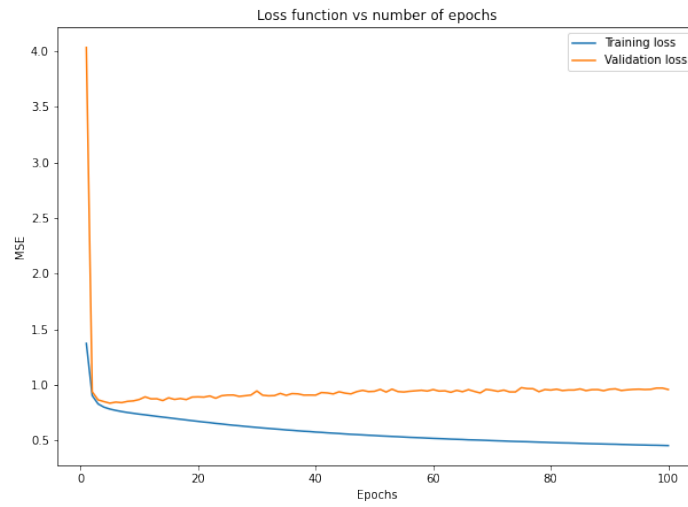
### 3.1.2 Learning curves

After having fixed the initial parameters and hyper-parameters of each model, I proceed to perform the training in each of them according to the algorithms described previously. During each epoch, I compute the training loss (with respect to training set) and validation loss (with respect to validation set) for later graphical use later.

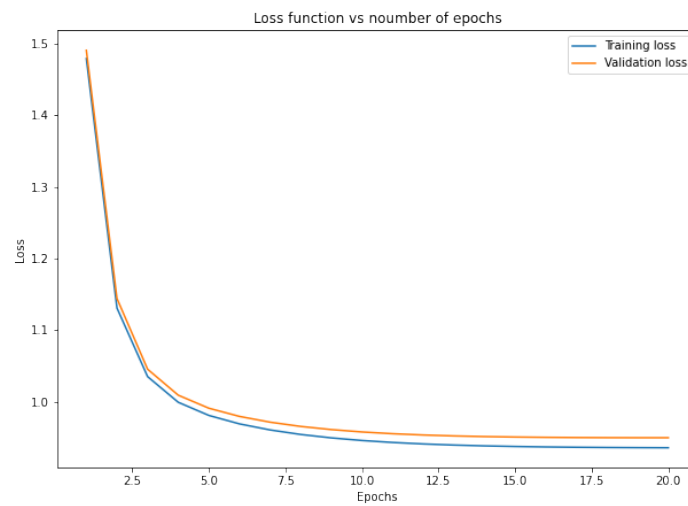
In the next figures, we can see the behavior of loss function (training and validation losses) according to the number of epochs for each model.

As we can see in Figure 3.1, for the MLP model, the training loss decreases as the number of epochs passes, which does not happen with the validation loss, which descends rapidly but then presents a growing trend. This therefore leads to an overfitting problem. In Figure 3.2, we have that for the VIME (only supervised) model, the training and validation losses decrease as the number of epochs passes, which does not lead to an

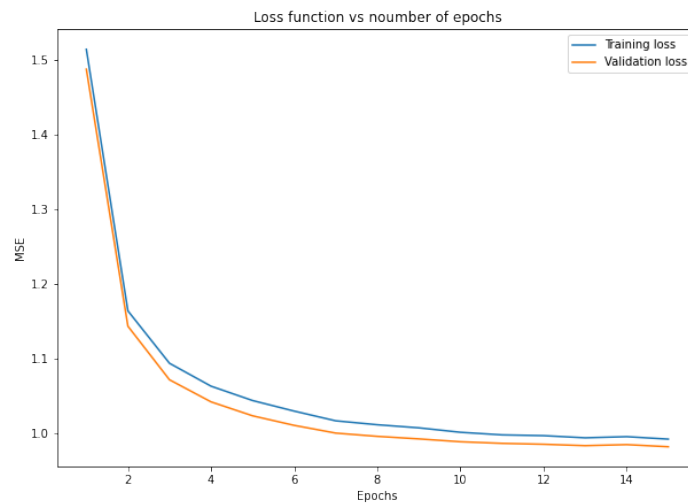
overfitting problem since the model has a good performance on the training and validation sets. We can see a similar behavior for the VIME model in Figure 3.3.



**Figure 3.1:** Learning curves for MLP initial model



**Figure 3.2:** Learning curves for VIME (only supervised) initial model



**Figure 3.3:** Learning curves for VIME initial model

## 3.2 Model performance

As we have seen before, there is an overfitting problem during the training of MLP model, but not with VIME (only supervised) and VIME models. So, I will try to reduce this problem for MLP model through the search of best hyper-parameters. I will also search the best hyper-parameters for VIME (only supervised) and VIME models to improve the results obtained previously. The performance of the different models will be measured by the mean squared error (MSE) calculated between true and predicted values on the validation set.

### 3.2.1 Hyper-parameters tuning

For MLP model, I vary the learning rate  $lr \in \{10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}\}$ . As we can see in Figure 5.18, the performance with  $lr = 10^{-4}$  improves a little with respect to the initial model. So, I decide to fix the value for the learning rate as  $lr = 10^{-4}$ . Then, I try to change the rate for dropout layers, where I got that is better to work with the same dropout layers: 0.3/0.3/0.3 rate. Besides, I decreased the number of epochs to 15 for training. So, the best MLP model is the one that has a learning rate  $lr = 10^{-4}$  and dropout layers: 0.3/0.3/0.3 rate.

For the VIME (only supervised) model, I fix the hyper-parameters  $\alpha = 1, p_m = 0.2$  and I vary the learning rate  $lr \in \{10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}\}$ . As we can see in Figure 5.19, the best performance is obtained with  $\hat{lr} = 10^{-3}$ . Then, I fix  $\hat{lr} = 10^{-3}, \alpha = 1$  and I vary  $p_m \in \{0.2, 0.4, 0.5, 0.6, 0.7, 0.9\}$ , where the best performance is obtained with  $\hat{p}_m = 0.2$  in Figure 5.20. Finally, I fix  $\hat{lr} = 10^{-3}, \hat{p}_m = 0.2$  and I vary  $\alpha \in \{0.1, 0.5, 1, 1.5, 3, 5, 7, 10\}$ , where the best performance is obtained with  $\hat{\alpha} = 1$  in Figure 5.21, although there is a small difference with  $\alpha = 5$ , which is not significant. Therefore, the best hyper-parameters for the VIME (only supervised) model are  $\hat{lr} = 10^{-3}, \hat{p}_m = 0.2, \hat{\alpha} = 1$  that match the initial hyper-parameters.

On the other hand, I fix the hyper-parameters  $\alpha = 1, p_m = 0.2, \beta = 1, K = 3$  on the VIME model, and I vary the learning rate  $lr \in \{10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}\}$ . As we can see in Figure 5.22, the best performance is obtained with  $\hat{lr} = 10^{-3}$ . Then, I fix  $\hat{lr} = 10^{-3}, \alpha = 1, \beta = 1, K = 3$  and I vary  $p_m \in \{0.2, 0.5, 0.8\}$ , where the best performance is obtained with  $\hat{p}_m = 0.2$  in Figure 5.23. After, I fix  $\hat{lr} = 10^{-3}, \hat{p}_m = 0.2, \beta = 1, K = 3$  and I vary  $\alpha \in \{0.5, 1, 2, 3, 5\}$ , where the best performance is obtained with  $\hat{\alpha} = 3$  in Figure 5.24. Now, I fix  $\hat{lr} = 10^{-3}, \hat{p}_m = 0.2, \hat{\alpha} = 3, K = 3$  and I vary  $\beta \in \{0.5, 1, 2, 3, 5\}$ , where the best performance is obtained with  $\hat{\beta} = 0.5$  in Figure 5.25. Finally, I fix  $\hat{lr} = 10^{-3}, \hat{p}_m = 0.2, \hat{\alpha} = 3, \hat{\beta} = 0.5$  and I vary  $K \in \{1, 3, 5\}$ , where the best performance is obtained with  $\hat{K} = 5$  in Figure 5.26. Therefore, the best hyper-parameters for the VIME model are  $\hat{lr} = 10^{-3}, \hat{p}_m = 0.2, \hat{\alpha} = 3, \hat{\beta} = 0.5$  and  $\hat{K} = 5$ .

### 3.2.2 Improvement of learning curves

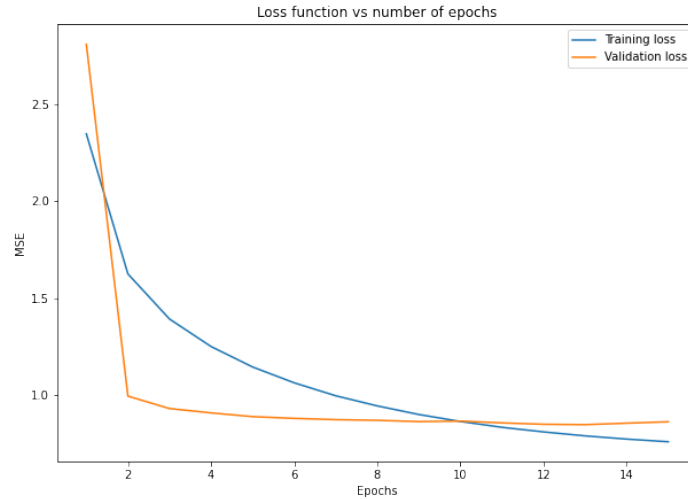
After having chosen the best hyper-parameters of each model, I proceed to perform again the training in each one of them, and I also calculate the training loss and validation loss. In the next Figures, we can see the new behavior of loss function (training and validation losses) according to the number of epochs for each model.

I must stress that there is no graph of the improved learning curves for the VIME (only supervised) model, as the hyper-parameters selected are the same as those initially chosen.

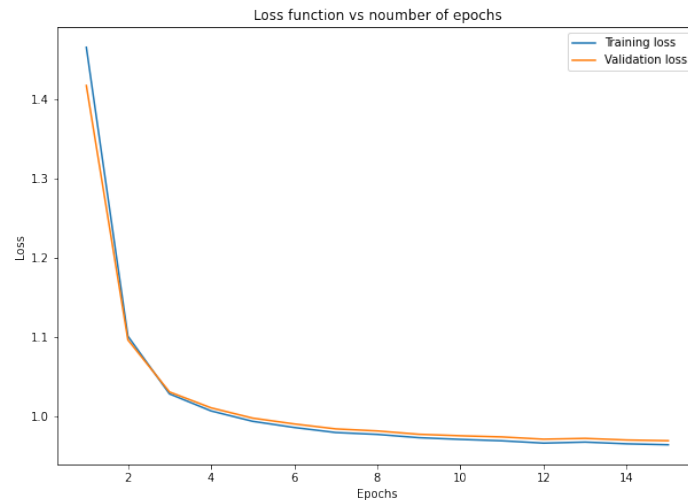
As we can see in Figure 3.4, for the MLP model, the training loss decreases as the number of epochs passes, which happens slowly with the validation loss because it de-

scends rapidly but then it presents a slow descent. It seems that there is no an overfitting problem, and this was one of the best models I could find.

In Figure 3.5, we have that the learning of the VIME model with the best hyper-parameters improves a little with respect to the initial model, and besides, there is no overfitting problem.



**Figure 3.4:** Learning curves for MLP model with best hyper-parameters



**Figure 3.5:** Learning curves for VIME model with best hyper-parameters

### 3.3 Model comparison

After finding the best hyper-parameters for each model, I proceed to evaluate the performance of each model as a function of the number of training data, that is, it is expected that as the amount of data for the training is increased, the performance of the model is better. To do this, I will take the training set into 10 partitions: 5000, 10000, 15000, 20000, 25000, 30000, 35000, 40000, 45000 and 58484 data since I use 35000 samples from training set to train the encoder of the self-supervised framework.

For each of these partitions, I will train the model according to the current partition of training set, and I compute the MSE between true and predicted values on the testing set. I repeat these calculations 10 times for MLP and VIME (only supervised).

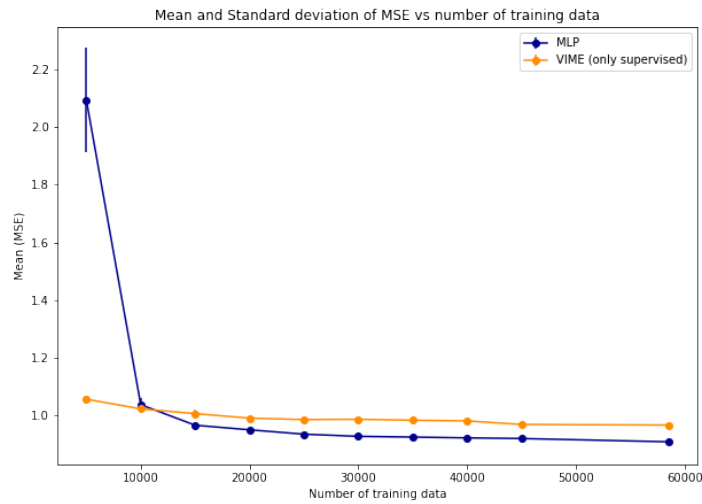
For the VIME model, I repeat these calculations 4 times due to long execution times. Then, I choose 4 values from MSE calculated for MLP and VIME (only supervised) when comparing these models with VIME.

### 3.3.1 MSE analysis

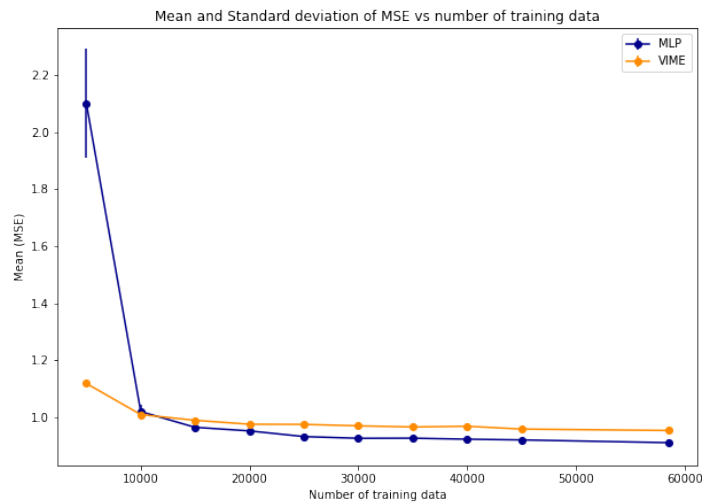
The following figures show the mean and standard deviation of the MSE calculated on the testing set as a function of the number of training data for each model.

In Figure 3.6 we can see MLP has a slightly better performance than slightly. We also have a similar behavior in Figure 3.7 where VIME has a slightly worse performance than MLP. Besides, we notice that the deviation standard of MSE for MLP at the beginning is bigger with respect to the other two models.

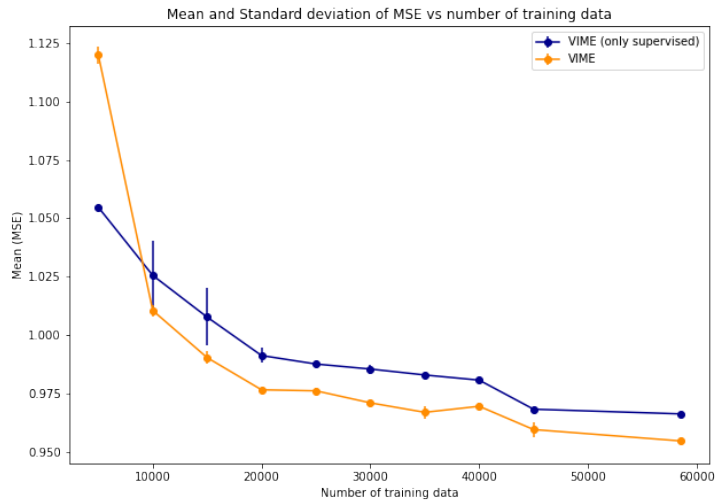
Figure 3.8 shows that VIME has a better performance than VIME (only supervised) and therefore, VIME also has the best performance of the three models. Besides, the deviation standard of MSE for VIME is smaller than that of VIME (only supervised).



**Figure 3.6:** Comparison between MLP and VIME (only supervised) by mean and standard deviation of MSE



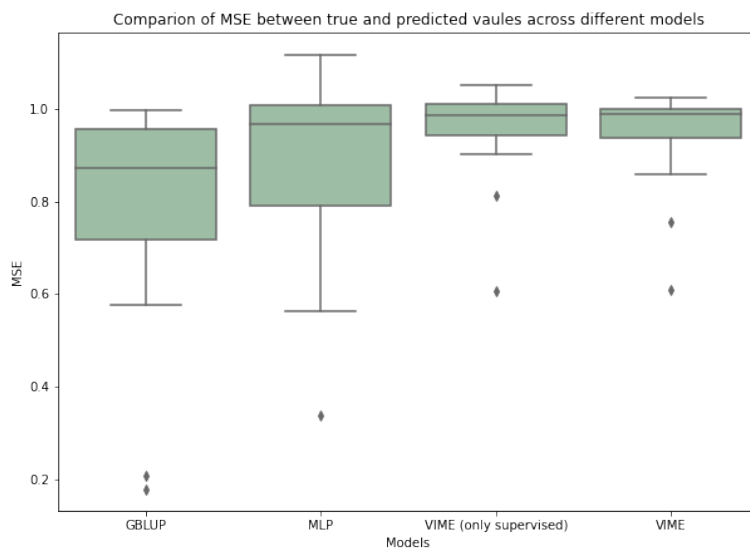
**Figure 3.7:** Comparison between MLP and VIME by mean and standard deviation of MSE



**Figure 3.8:** Comparison between VIME (only supervised) and VIME by mean and standard deviation of MSE

On the other hand, in Table 5.1 we can see the MSE calculated between true and predicted values on the testing set for each trait and with respect to each model: GBLUP, MLP, VIME (only supervised) and VIME. Here, we can notice that traits 4, 5, 1, 27 and 19 have the best performance in terms of MSE, while traits 11, 22 and 15 have the worst performance in terms of MSE.

In Figure 3.9 we can see the boxplot for comparison of MSE between true and predicted values across different models, where we can notice there is a little difference on the mean of MSE between models. So, I decided to perform an ANOVA test to contrast this hypothesis, and the results of this test are shown in Table 3.1. The  $p$  value obtained from ANOVA test is significant ( $p < 0.05$ ), and therefore, we conclude that there are significant differences among models.



**Figure 3.9:** Boxplot for comparison of MSE between true and predicted values across different models

sum_sq	df	F	Pr(>F)
0.536754	3.0	8.831117	0.000023
2.593274	128.0	NaN	NaN

**Table 3.1:** ANOVA test for comparison of MSE between true and predicted values across different models

From ANOVA analysis, we know that model differences are statistically significant in terms of MSE, but ANOVA does not tell which models are significantly different from each other. To know the pairs of significant different models, I will perform multiple pairwise comparison analysis for all unplanned comparison using Tukey’s test, and the results of this test can be seen in Table 3.2.

Below results from Tukey’s test suggest all pairwise comparisons between GBLUP and other models reject the null hypothesis of no variation in means (of MSE) between pairs of models ( $p < 0.05$ ) and indicates statistical significant differences, that is, GBLUP performance in terms of MSE differs statistically significant from other models.

group1	group2	Diff	Lower	Upper	q-value	p-value
GBLUP	MLP	0.096679	0.005457	0.187900	3.901835	0.033244
GBLUP	VIME (only supervised)	0.161721	0.070500	0.252943	6.526866	0.001000
GBLUP	VIME	0.149606	0.058384	0.240828	6.037914	0.001000
MLP	VIME (only supervised)	0.065042	-0.026179	0.156264	2.625031	0.252321
MLP	VIME	0.052927	-0.038294	0.144149	2.136079	0.435698
VIME (only supervised)	VIME	0.012115	-0.079106	0.103337	0.488952	0.900000

**Table 3.2:** Tukey’s test for comparison of MSE between true and predicted values across different models

### 3.3.2 Performance of prediction by phenotype

In Figures 5.28 to 5.42 we can see the plot and fit of a linear regression between true values and predicted values of YD obtained on the testing set for each model: MLP, VIME (only supervised) and VIME, and certain phenotypes chosen (CAR1 to CAR5). I also computed the value of  $R^2$ , adjusted  $R^2$  and the slope for each linear regression.

In Table 5.2 we can see the value of slope obtained from the linear regression between true values and predicted values of YD for all traits and each model.

We can see that the best fit of predictions with real values of YD for MLP model is obtained with traits 4 and 5 where the values of the slope are 0.7755 and 0.7131, respectively, which means the predicted values are strongly correlated with true values on these traits. We also got  $R^2 = 0.7832$  and an adjusted  $R^2 = 0.7831$  for trait 4, and we had  $R^2 = 0.7399$  and an adjusted  $R^2 = 0.7398$  for trait 5.

For VIME (only supervised) and VIME models, we get the best fit between predictions and true values of YD with trait 4 (CAR4), where the values of slope are 0.4658 and 0.3548, respectively. We also got a  $R^2 = 0.4081$  and an adjusted  $R^2 = 0.408$  for VIME (only supervised) model, and we had  $R^2 = 0.4163$  and an adjusted  $R^2 = 0.4162$  for VIME model.

So, I can notice that each model has the best performance on trait 4 with respect to other dairy phenotypes, which I also observed in the results obtained from Tables 5.1 and 5.3, that is, the lowest value of MSE and the highest value of Pearson’s correlation are obtained on trait 4.

### 3.3.3 Correlation analysis between GBLUP and other models

In Table 5.3 we can see the Pearson’s correlation calculated between true and predicted values on the testing set for each trait and with respect to each model: GBLUP, MLP,



VIME (only supervised) and VIME. Here, we can notice that traits 1, 2, 4, 5 and 20 are the most correlated in terms of Pearson's correlation because the first four traits are genetic additives, while traits 7, 10 and 11 are the least correlated.

On the other hand, in Figures 3.10, 3.11 and 3.12 we can see the plot and fit of a linear regression for correlations between GBLUP and each model: MLP, VIME (only supervised) and VIME, respectively. I also computed the value of  $R^2$ , adjusted  $R^2$  and the slope for each linear regression.

We can notice correlations of MLP make a good fit with respect to GBLUP because I obtained  $R^2 = 0.998$  and an adjusted  $R^2 = 0.9979$ . I also got a slope of 0.9896, which is the biggest value with respect to the other ones computed.

The correlations of VIME (only supervised) also make a good fit with respect to GBLUP because I obtained  $R^2 = 0.9377$  and an adjusted  $R^2 = 0.9336$ . I also got a slope of 0.6178. The correlations of VIME make a good fit with respect to GBLUP because I obtained  $R^2 = 0.9325$  and an adjusted  $R^2 = 0.9280$ . I also got a slope of 0.6354.

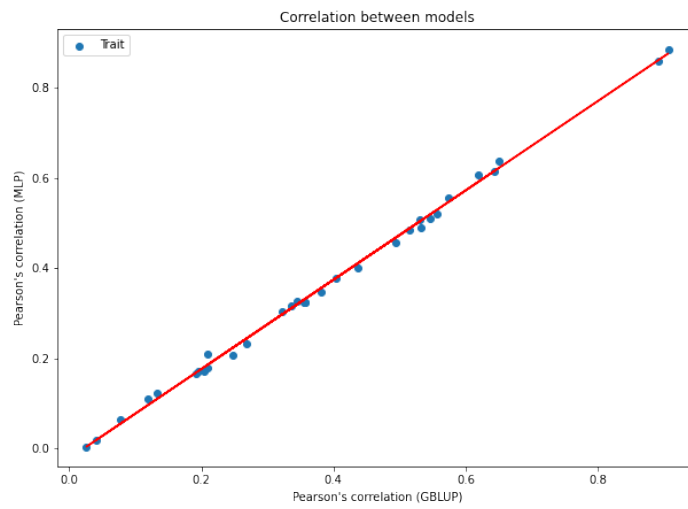
Besides, I notice that traits 4 and 5 are a little far from the regression line with respect to the VIME (only supervised) and VIME models, but it is not statistically significant.

In Figure 5.27 we can see the boxplot for comparison of Pearson's correlation between true and predicted values across different models, where we can notice there is a little difference on the mean Pearson's correlation between models. So, I decided to perform an ANOVA test to contrast this hypothesis, and the results of this test are shown in Table 5.4. The  $p$  value obtained from ANOVA test is significant ( $p < 0.05$ ), and therefore, we conclude that there are significant differences among models.

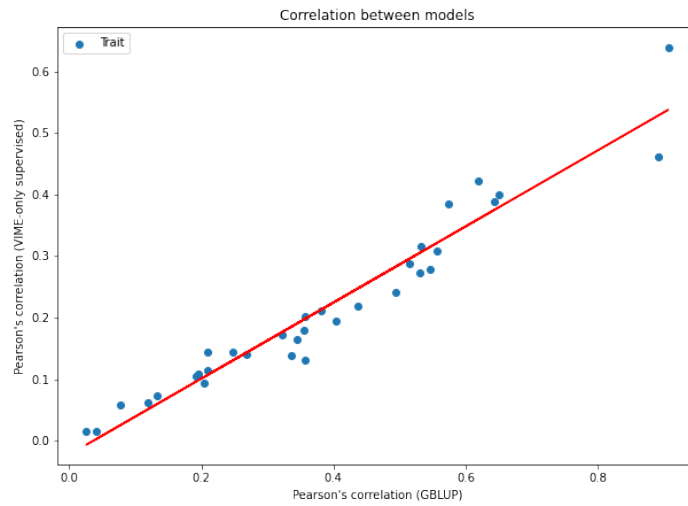
As we know model differences are statistically significant in terms of Pearson's correlation. So, I will perform multiple pairwise comparison analysis for all unplanned comparison using Tukey's test to identify the pairs of significant different models, and the results of this test can be seen in Table 5.5.

From Tukey's test, we know that excepts GBLUP-MLP and VIME (only supervised)-VIME, all pairwise comparisons for models reject null hypothesis of no variation in means (of Pearson's correlation) between pairs of models ( $p < 0.05$ ) and indicates statistical significant differences, that is, GBLUP and MLP performance in terms of Pearson's correlation differs statistically significant from the two VIME models.

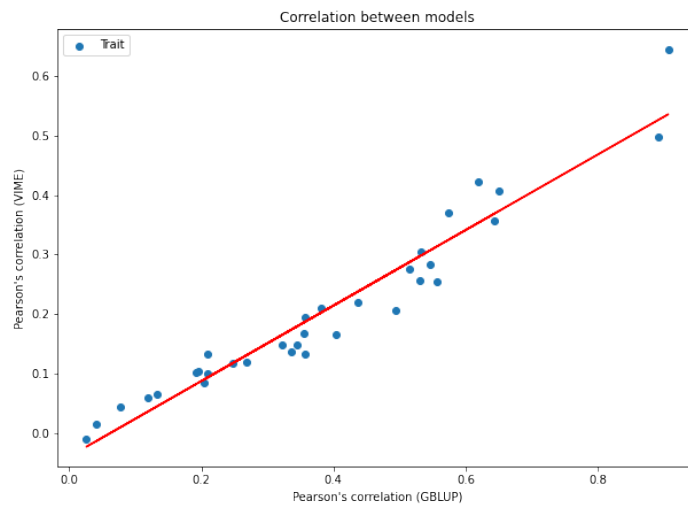
Finally, I notice MLP model has the best fit with respect to GBLUP model in terms of Pearson's correlation although the training of MLP was not good enough. However, the performance of VIME model is also good and could be improved if we try to augment the number of samples on unlabeled samples used to train the encoder architecture and the semi-supervised learning framework.



**Figure 3.10:** Correlation analysis between MLP and GBLUP



**Figure 3.11:** Correlation analysis between VIME (only supervised) and GBLUP



**Figure 3.12:** Correlation analysis between VIME and GBLUP

## Chapter 4

# Conclusion

Almost all dairy traits (CAR1 to CAR5) perform moderately well because the mean value of MSE was 0.8049 for GBLUP, 0.9016 for MLP, 0.9666 for VIME (only supervised), and 0.9545 for VIME, which is still a little high. In spite of it, we can assure that the first objective is met. In addition, we have seen that these traits are the most correlated among all the traits due to their nature as genetic additives.

We also notice that trait 4 (CAR4) is the best predicted phenotype among MLP and the two VIME models because it has the lowest value of MSE (0.1779 for GBLUP; 0.3369 for MLP; 0.6072 for VIME (only supervised); 0.6096 for VIME) among all dairy traits, which implies this trait is more suitable to represent this group of phenotypes.

On the other hand, we can see the worst predicted phenotypes are traits 7, 10 and 11, which correspond to fertility traits. These traits had the highest values (near/upper to 1) of MSE in each model, and this implies that the models presented in this work were not good enough to capture their effects and make a good estimate of them from the available genotyping data.

Regarding to the second objective we can note that it is not completely fulfilled since the MLP model has a better performance than two VIME models proposed to optimize the genotype data matrix in terms of the MSE. Furthermore, in terms of the Pearson's correlation, it seems that the correlations of the MLP model have a slightly better fit with respect to the correlations of the GBLUP model presented. This leads us to ask ourselves what would happen if we increased the number of unlabeled data for both self- and semi-supervised framework approaches training, that is, it is recommended to increase the amount of data available to have a better performance compared to the one presented in this work.

If our results have not been able to demonstrate a clear advantage of deep learning for genomic prediction with respect to classic statistical models, these results interested the scientists to whom I presented my work. This study will be continued by BIGE, G2B and MIA-Paris teams after I finish my internship for the purpose of publishing a scientific article, which will allow to cover and/or apply certain methods or ideas that could not be carried out in it, such as, for example, try new approaches to optimize the calculation time of the proposed VIME models or it is necessary to increase the training base to improve the results.

The use of artificial intelligence in biology is increasing recently. As we have seen in this work, it is possible to apply self- and semi-supervised learning algorithms for our predictive task. It is advisable to use this genotype data in other deep learning architectures in order to, for example, create new artificial genotypes from a trained model to use them later in predictive tasks.

# Chapter 5

## Annexes

### Calculation of the variable Yield Deviation and its weight

First, we consider a genetic evaluation model (in this case genomic evaluation SingleStep). So, we take an example of a trait measured several times in an animal's life (e.g., quantity of milk per lactation); and we suppose that the animal  $i$  has 3 performances described by the following equations:

$$\begin{aligned}y_{i,1} &= m_{1,a} + m_{2,b} + m_{3,c} + p_i + a_i + e_{i,1} \\y_{i,2} &= m_{1,a'} + m_{2,b'} + m_{3,c'} + p_i + a_i + e_{i,2} \\y_{i,3} &= m_{1,a''} + m_{2,b''} + m_{3,c''} + p_i + a_i + e_{i,3}\end{aligned}$$

where

- $m_k$  : correspond to the identified environmental effects impacting the phenotype.
- $p_i$  : it is the permanent environmental effect of the animal  $i$  (non-genetic) which has an identical impact on the 3 performances.
- $a_i$  : it is the additive genetic value of the animal  $i$ .
- $e_{i,j}$  : residual of the model for the performance  $j$  considered

Furthermore, we have the weight of performance  $j$  for each animal  $i$  given by  $w_{i,j}$  with  $j = 1, \dots, 3$ .

Then, we compute the adjusted performances:

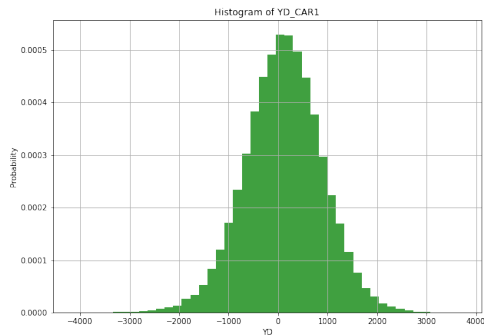
$$\begin{aligned}y_{adj_{i,1}} &= y_{i,1} - \hat{m}_{1,a} - \hat{m}_{2,b} - \hat{m}_{3,c} - \hat{p}_i \\y_{adj_{i,2}} &= y_{i,2} - \hat{m}_{1,a'} - \hat{m}_{2,b'} - \hat{m}_{3,c'} - \hat{p}_i \\y_{adj_{i,3}} &= y_{i,3} - \hat{m}_{1,a''} - \hat{m}_{2,b''} - \hat{m}_{3,c''} - \hat{p}_i\end{aligned}$$

Therefore, the variable Yield Deviation for an animal  $i$  ( $YD_i$ ) is computed by:

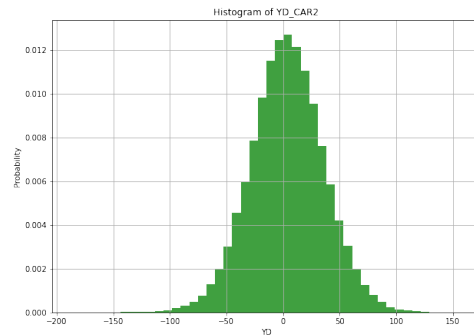
$$YD_i = \frac{\sum_{j=1}^3 w_{i,j} y_{adj_{i,j}}}{\sum_{j=1}^3 w_{i,j}}.$$

Otherwise, the weight of Yield Deviation for an animal  $i$  ( $wYD_i$ ) is the equivalent number of performances, calculated from the weights of each performance of  $i$ , the size of contemporary groups, repeatability and heritability of the trait.

## Histogram of YD for all traits (1 to 33)

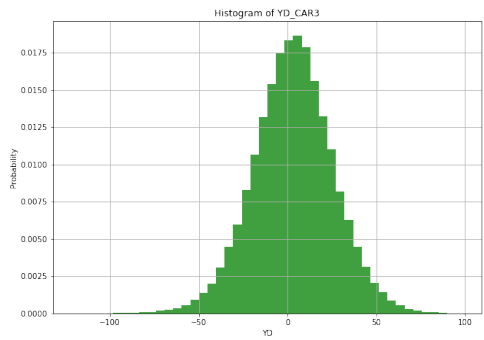


(a) Trait 1

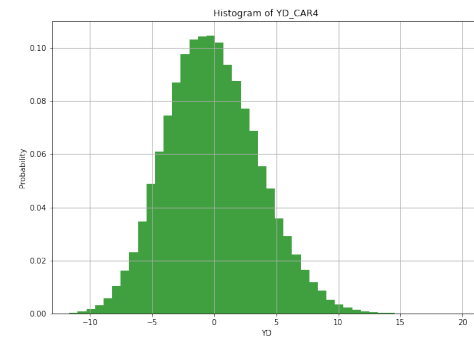


(b) Trait 2

**Figure 5.1:** Histogram of YD for traits 1 and 2

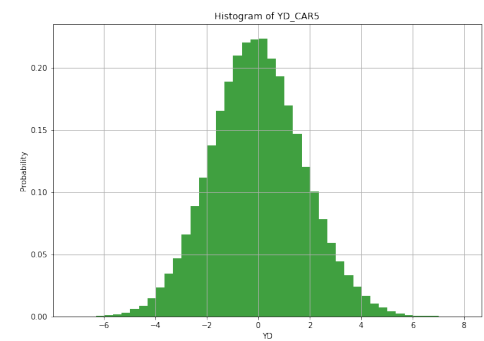


(a) Trait 3

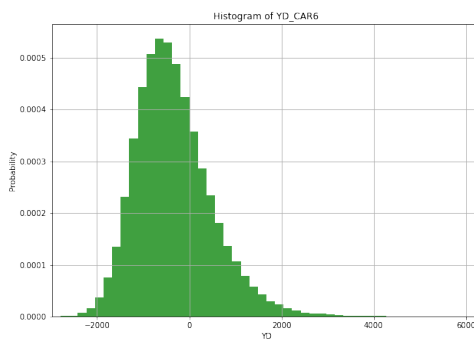


(b) Trait 4

**Figure 5.2:** Histogram of YD for traits 3 and 4

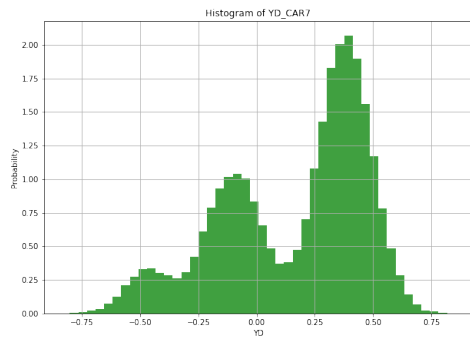


(a) Trait 5

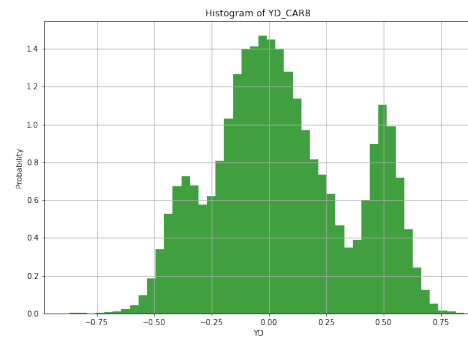


(b) Trait 6

**Figure 5.3:** Histogram of YD for traits 5 and 6

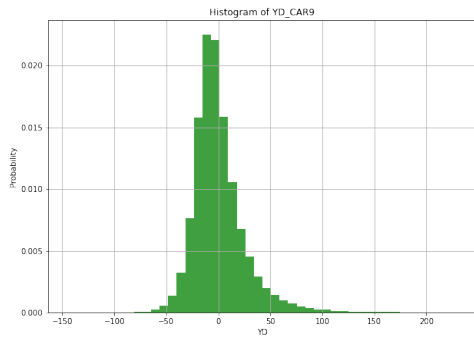


(a) Trait 7

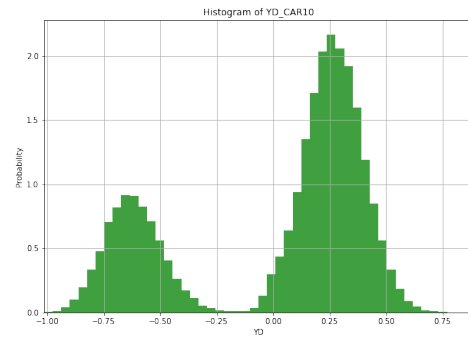


(b) Trait 8

**Figure 5.4:** Histogram of YD for traits 7 and 8

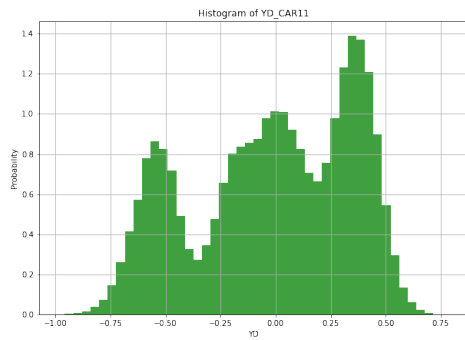


(a) Trait 9

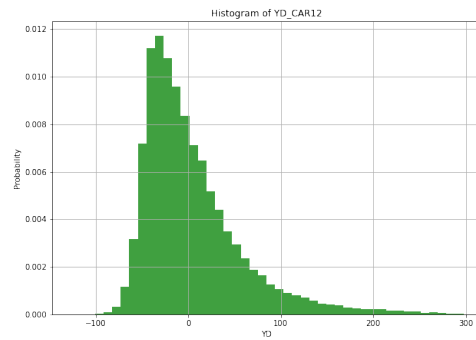


(b) Trait 10

**Figure 5.5:** Histogram of YD for traits 9 and 10

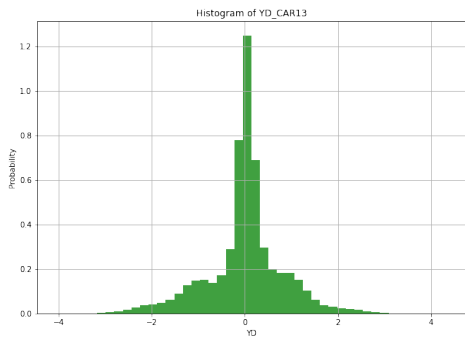


(a) Trait 11

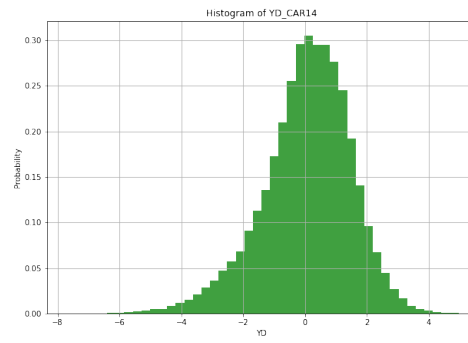


(b) Trait 12

**Figure 5.6:** Histogram of YD for traits 11 and 12

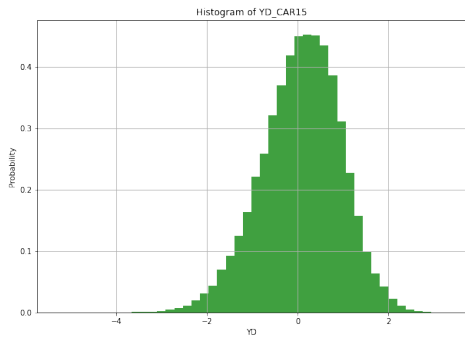


(a) Trait 13

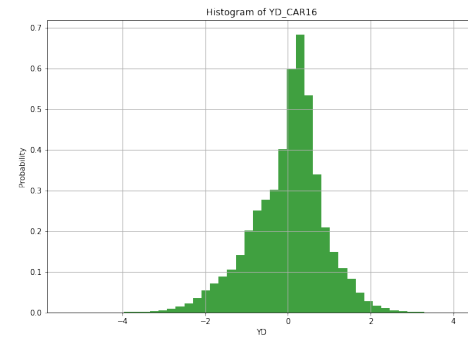


(b) Trait 14

**Figure 5.7:** Histogram of YD for traits 13 and 14

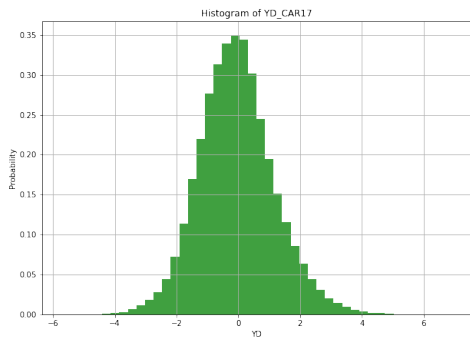


(a) Trait 15

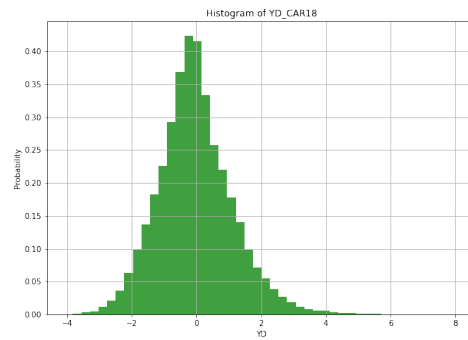


(b) Trait 16

**Figure 5.8:** Histogram of YD for traits 15 and 16

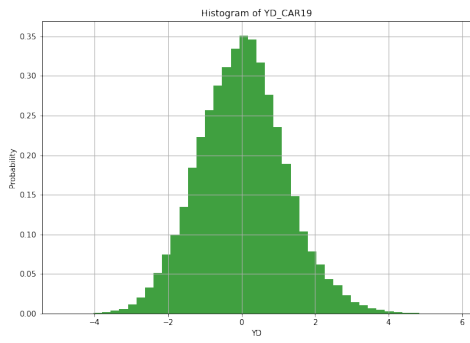


(a) Trait 17

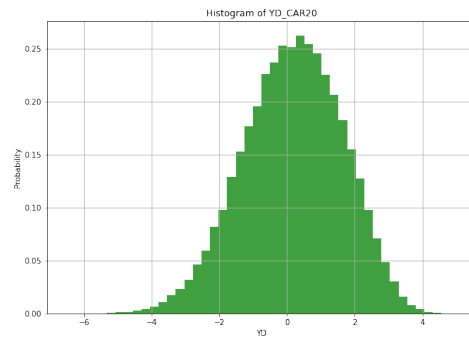


(b) Trait 18

**Figure 5.9:** Histogram of YD for traits 17 and 18

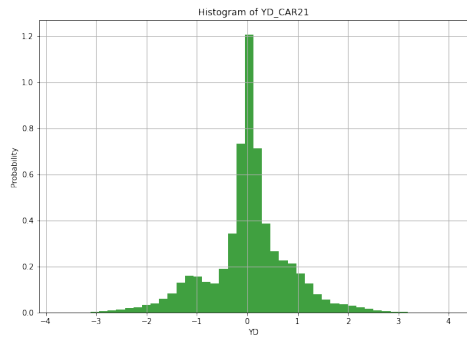


(a) Trait 19

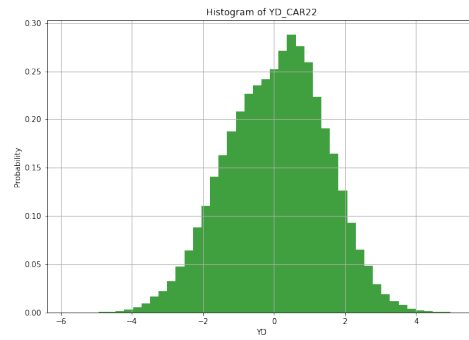


(b) Trait 20

**Figure 5.10:** Histogram of YD for traits 19 and 20

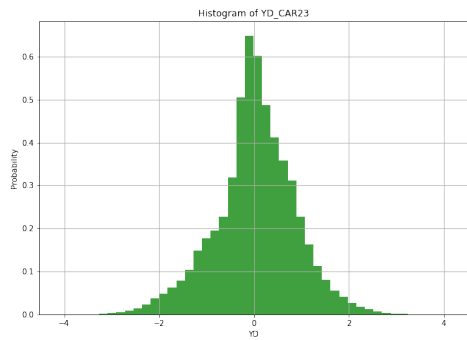


(a) Trait 21

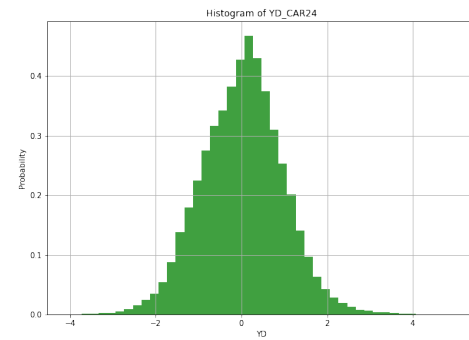


(b) Trait 22

**Figure 5.11:** Histogram of YD for traits 21 and 22



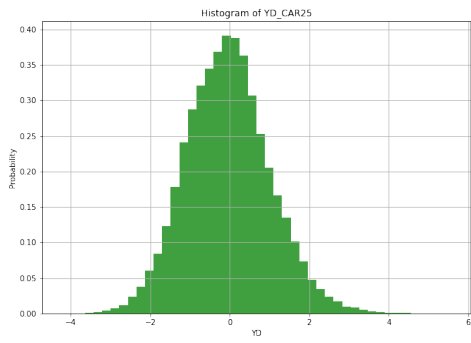
(a) Trait 23



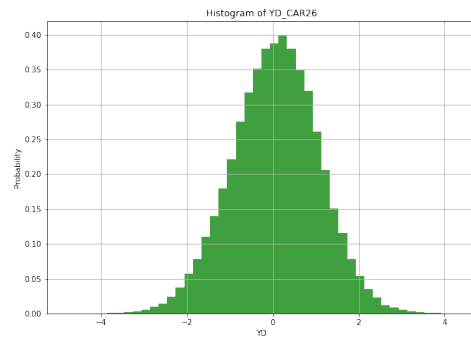
(b) Trait 24

**Figure 5.12:** Histogram of YD for traits 23 and 24



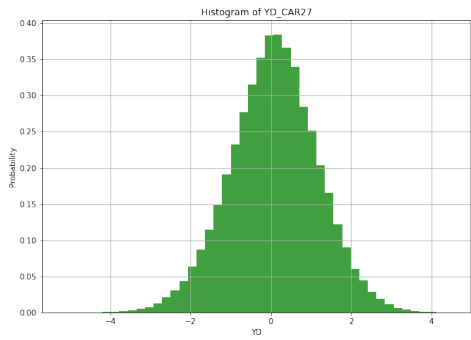


(a) Trait 25

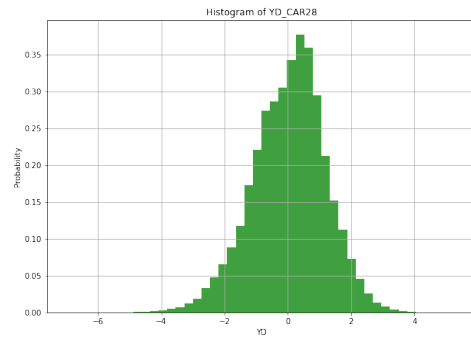


(b) Trait 26

**Figure 5.13:** Histogram of YD for traits 25 and 26

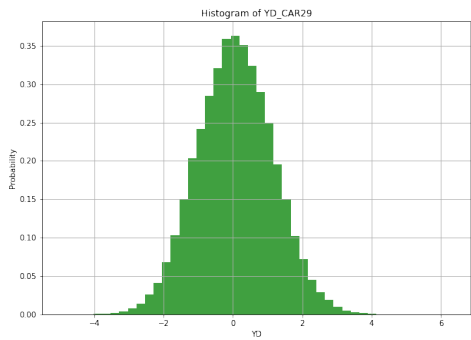


(a) Trait 27

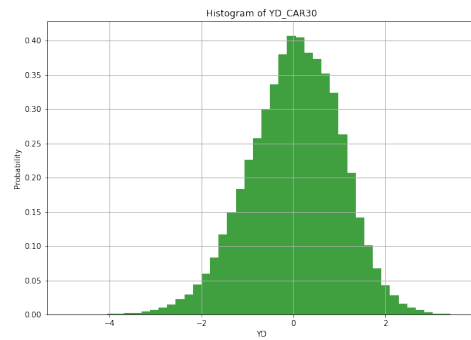


(b) Trait 28

**Figure 5.14:** Histogram of YD for traits 27 and 28

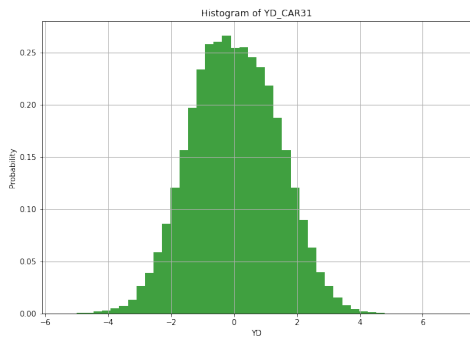


(a) Trait 29

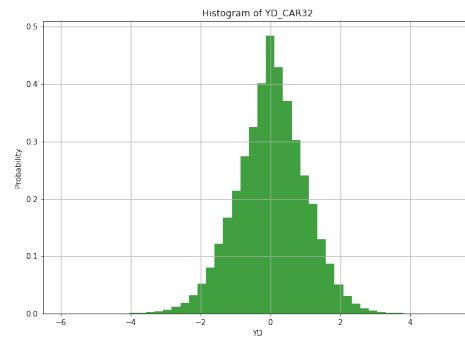


(b) Trait 30

**Figure 5.15:** Histogram of YD for traits 29 and 30

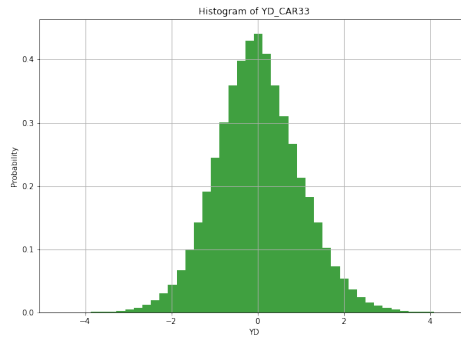


(a) Trait 31



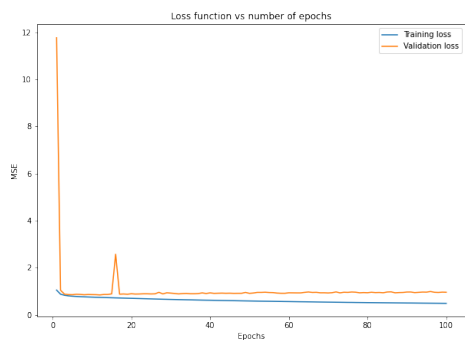
(b) Trait 32

**Figure 5.16:** Histogram of YD for traits 31 and 32

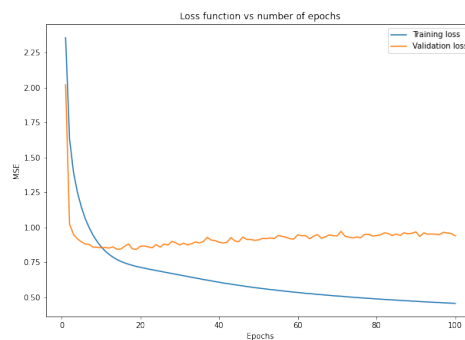


**Figure 5.17:** Histogram of YD for trait 33

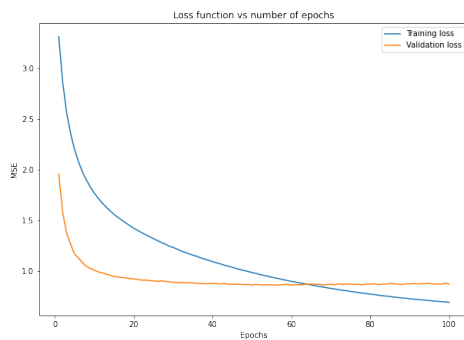
# Hyper-parameters tuning for MLP



(a)  $lr = 10^{-2}$



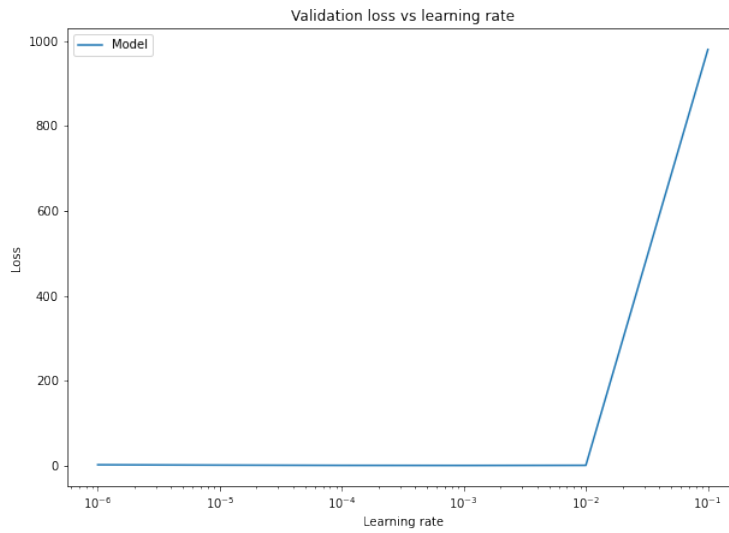
(b)  $lr = 10^{-4}$



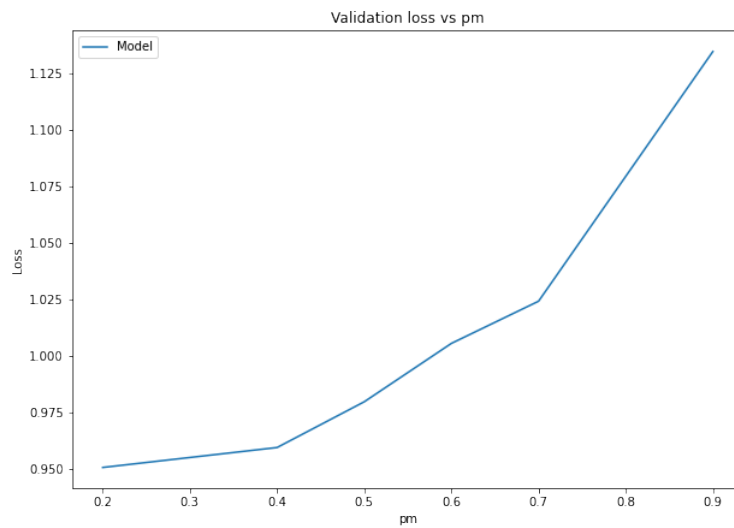
(c)  $lr = 10^{-5}$

**Figure 5.18:** Learning curves for MLP across different values of the hyper-parameter learning rate

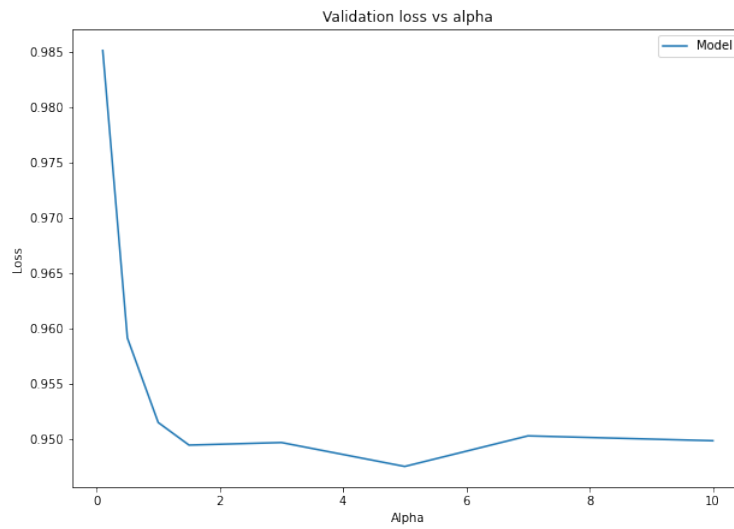
## Hyper-parameters tuning for VIME only supervised



**Figure 5.19:** MSE performance for VIME (only supervised) across different values of the hyper-parameter learning rate

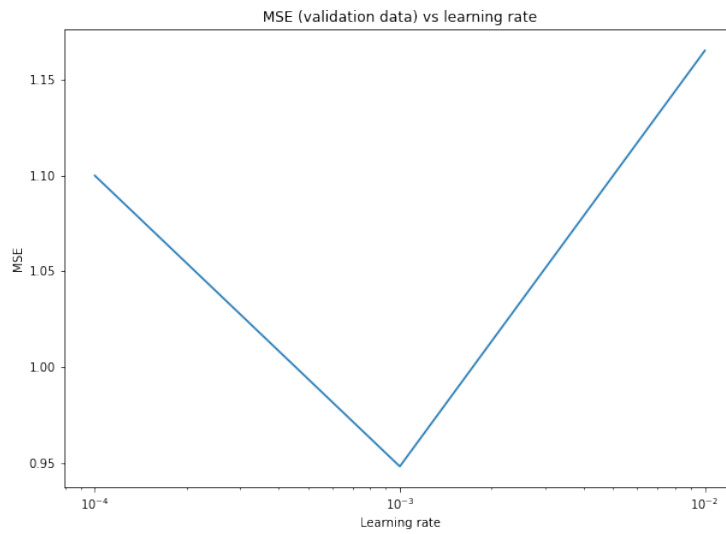


**Figure 5.20:** MSE performance for VIME (only supervised) across different values of the hyper-parameter  $p_m$

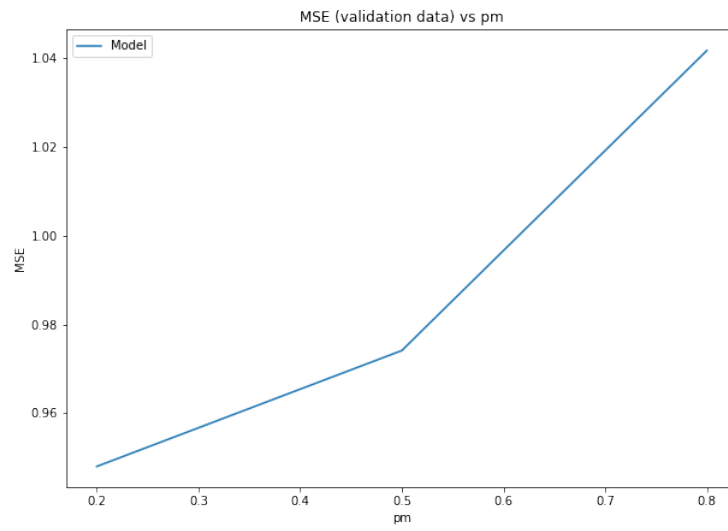


**Figure 5.21:** MSE performance for VIME (only supervised) across different values of the hyper-parameter  $\alpha$

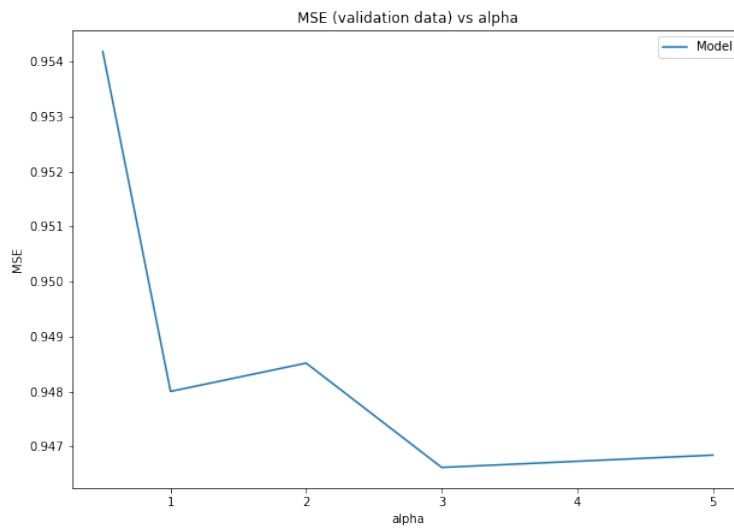
## Hyper-parameters tuning for VIME



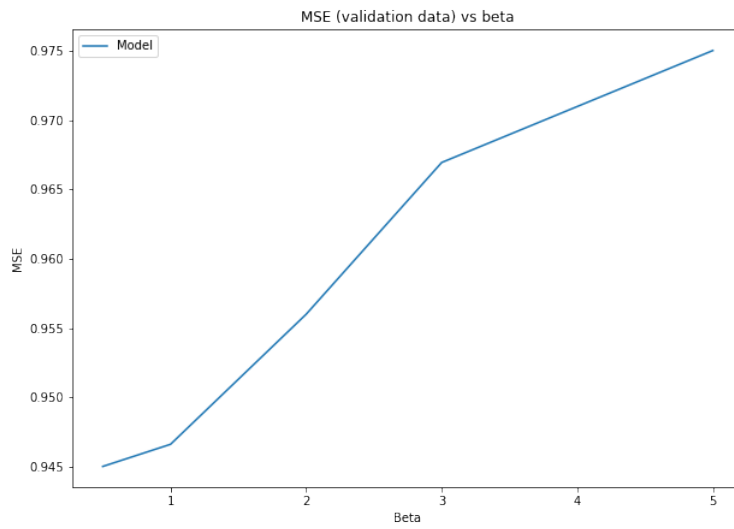
**Figure 5.22:** MSE performance for VIME across different values of the hyper-parameter learning rate



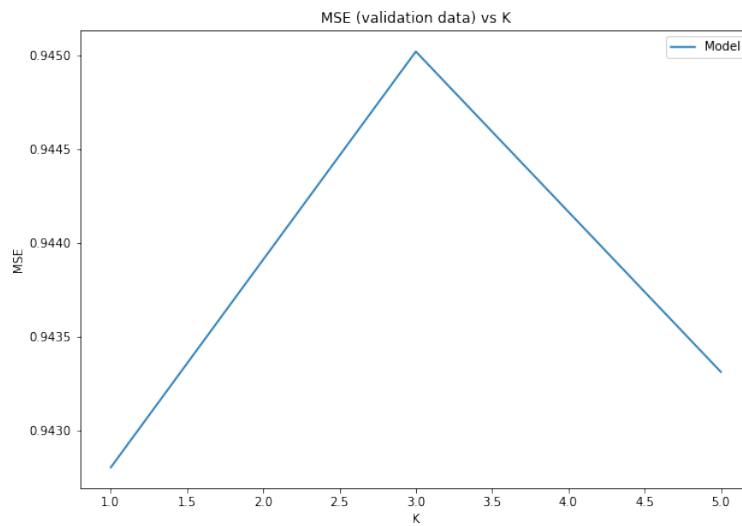
**Figure 5.23:** MSE performance for VIME across different values of the hyper-parameter  $p_m$



**Figure 5.24:** MSE performance for VIME across different values of the hyper-parameter  $\alpha$



**Figure 5.25:** MSE performance for VIME across different values of the hyper-parameter  $\beta$



**Figure 5.26:** MSE performance for VIME across different values of the hyper-parameter  $K$

## MSE between true and predicted values for each trait

Phenotype	GBLUP	MLP	VIME (only supervised)	VIME
CAR1	0.5771	0.6516	0.9161	0.8585
CAR2	0.6159	0.9510	0.9324	0.9003
CAR3	0.6698	1.0075	0.9531	0.8927
CAR4	0.1779	0.3369	0.6072	0.6096
CAR5	0.2063	0.5639	0.8131	0.7554
CAR6	0.8817	0.9496	0.9880	0.9946
CAR7	0.9985	1.0214	1.0144	1.0088
CAR8	0.9860	1.0406	1.0521	1.0263
CAR9	0.9633	1.0112	1.0361	1.0091
CAR10	0.9994	1.0179	1.0110	1.0123
CAR11	0.9942	1.0101	1.0062	1.0046
CAR12	0.9823	1.0227	1.0519	1.0254
CAR13	0.8957	0.9294	0.9882	0.9902
CAR14	0.8542	0.8849	0.9640	0.9615
CAR15	0.7192	1.1183	1.0171	0.9620
CAR16	0.8728	0.9788	1.0069	0.9979
CAR17	0.7175	0.7777	0.9058	0.9093
CAR18	0.8095	0.8443	0.9610	0.9533
CAR19	0.7023	0.7805	0.9440	0.9239
CAR20	0.5866	0.7393	0.9030	0.9131
CAR21	0.9563	0.9723	1.0009	0.9970
CAR22	0.8376	1.0371	1.0244	0.9964
CAR23	0.7551	0.7920	0.9484	0.9590
CAR24	0.9389	0.9687	0.9843	0.9905
CAR25	0.8730	1.0369	1.0227	0.9953
CAR26	0.8868	0.9234	1.0132	0.9997
CAR27	0.7351	0.7774	0.9355	0.9453
CAR28	0.9565	0.9922	1.0039	1.0016
CAR29	0.6909	0.7532	0.9117	0.9371
CAR30	0.9615	1.0065	1.0092	1.0001
CAR31	0.9281	0.9755	0.9919	0.9941
CAR32	0.9580	0.9827	1.0048	1.0008
CAR33	0.8738	0.8967	0.9761	0.9731
<b>Mean</b>	0.8049	0.9016	0.9666	0.9545
<b>Standard deviation</b>	0.2028	0.1618	0.0820	0.0836

Table 5.1: MSE between true and predicted values for each trait



## Slope of linear regression between true and predicted values for each trait

Phenotype	MLP	VIME (only supervised)	VIME
CAR1	0.4334	0.2063	0.1493
CAR2	0.4114	0.2169	0.1754
CAR3	0.3448	0.1728	0.1226
CAR4	0.7755	0.4658	0.3548
CAR5	0.7131	0.2640	0.2212
CAR6	0.1301	0.0467	0.0348
CAR7	0.0028	0.0020	0.0015
CAR8	0.0303	0.0117	0.0092
CAR9	0.0414	0.0266	0.0201
CAR10	0.0006	0.0017	-0.0011
CAR11	0.0121	0.0093	0.0052
CAR12	0.0319	0.0135	0.0100
CAR13	0.1068	0.0472	0.0319
CAR14	0.1336	0.0624	0.0483
CAR15	0.2457	0.1099	0.0846
CAR16	0.1056	0.0363	0.0303
CAR17	0.2269	0.1123	0.0854
CAR18	0.1684	0.0674	0.0536
CAR19	0.2553	0.0962	0.0757
CAR20	0.3682	0.1726	0.1305
CAR21	0.0393	0.0239	0.0156
CAR22	0.1427	0.0616	0.0433
CAR23	0.2103	0.0760	0.0509
CAR24	0.0550	0.0314	0.0209
CAR25	0.1156	0.0680	0.0550
CAR26	0.1171	0.0360	0.0290
CAR27	0.2428	0.1011	0.0799
CAR28	0.0724	0.0381	0.0280
CAR29	0.2696	0.1055	0.0702
CAR30	0.0431	0.0230	0.0188
CAR31	0.0642	0.0310	0.0224
CAR32	0.0396	0.0192	0.0148
CAR33	0.1226	0.0456	0.0341
<b>Mean</b>	0.1840	0.0849	0.0644
<b>Standard deviation</b>	0.1876	0.0954	0.0740

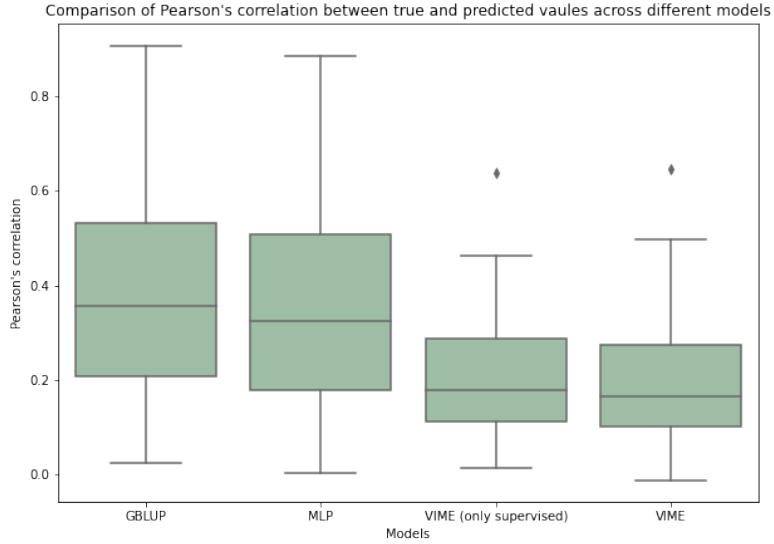
**Table 5.2:** Slope of linear regression between true and predicted values for each trait

**Pearson's correlation between true and predicted values for each trait**

<b>Phenotype</b>	<b>GBLUP</b>	<b>MLP</b>	<b>VIME (only supervised)</b>	<b>VIME</b>
CAR1	0.6503	0.6384	0.3991	0.4068
CAR2	0.6198	0.6059	0.4225	0.4227
CAR3	0.5747	0.5551	0.3848	0.3706
CAR4	0.9067	0.8850	0.6388	0.6452
CAR5	0.8909	0.8602	0.4618	0.4981
CAR6	0.3441	0.3277	0.1639	0.1488
CAR7	0.0405	0.0191	0.0162	0.0144
CAR8	0.1185	0.1098	0.0614	0.0590
CAR9	0.1918	0.1667	0.1043	0.1022
CAR10	0.0261	0.0044	0.0145	-0.0114
CAR11	0.0770	0.0655	0.0591	0.0433
CAR12	0.1333	0.1223	0.0725	0.0644
CAR13	0.3231	0.3040	0.1713	0.1484
CAR14	0.3820	0.3481	0.2113	0.2104
CAR15	0.5300	0.5078	0.2732	0.2561
CAR16	0.3567	0.3246	0.1307	0.1323
CAR17	0.5316	0.4904	0.3157	0.3043
CAR18	0.4365	0.4013	0.2191	0.2204
CAR19	0.5457	0.5107	0.2784	0.2832
CAR20	0.6430	0.6147	0.3880	0.3560
CAR21	0.2094	0.1782	0.1133	0.1002
CAR22	0.4031	0.3777	0.1947	0.1645
CAR23	0.4950	0.4560	0.2405	0.2063
CAR24	0.2474	0.2085	0.1445	0.1165
CAR25	0.3565	0.3255	0.2027	0.1941
CAR26	0.3365	0.3155	0.1383	0.1363
CAR27	0.5148	0.4859	0.2883	0.2749
CAR28	0.2089	0.2087	0.1442	0.1328
CAR29	0.5560	0.5203	0.3076	0.2540
CAR30	0.1964	0.1726	0.1090	0.1028
CAR31	0.2684	0.2323	0.1403	0.1194
CAR32	0.2051	0.1706	0.0931	0.0850
CAR33	0.3554	0.3254	0.1795	0.1676
<b>Mean</b>	0.3841	0.3588	0.2146	0.2039
<b>Standard deviation</b>	0.2217	0.2196	0.1414	0.1459

**Table 5.3:** Pearson's correlation between true and predicted values for each trait

## Comparison of Pearson's correlation between true and predicted values across different models



**Figure 5.27:** Box plot for comparison of Pearson's correlation between true and predicted values across different models

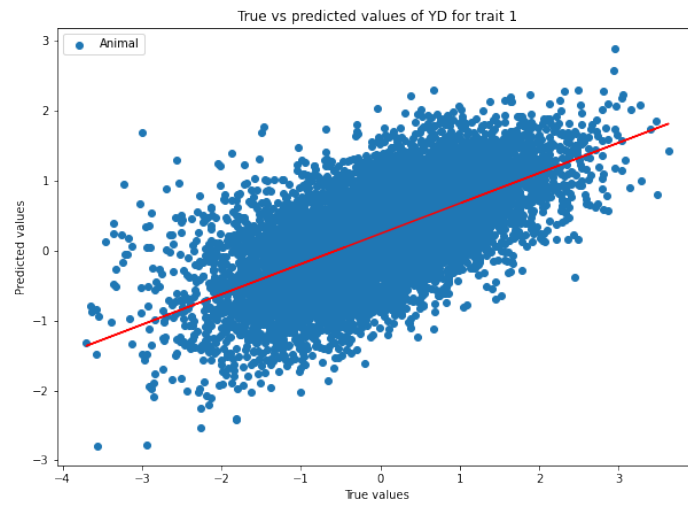
sum_sq	df	F	Pr(>F)
0.880141	3.0	8.465218	0.000036
4.436118	128.0	NaN	NaN

**Table 5.4:** ANOVA test for comparison of Pearson's correlation between true and predicted values across different models

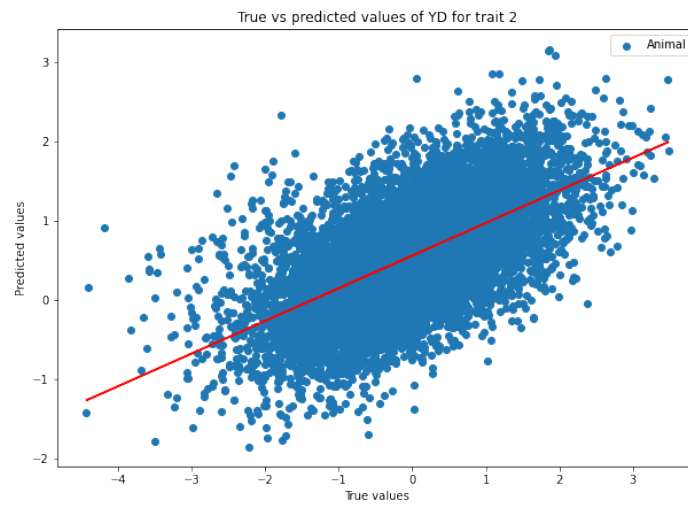
group1	group2	Diff	Lower	Upper	q-value	p-value
GBLUP	MLP	0.025342	-0.093967	0.144652	0.782003	0.900000
GBLUP	VIME (only supervised)	0.169473	0.050163	0.288782	5.229501	0.001807
GBLUP	VIME	0.180170	0.060860	0.299479	5.559582	0.001000
MLP	VIME (only supervised)	0.144130	0.024821	0.263440	4.447498	0.010986
MLP	VIME	0.154827	0.035518	0.274137	4.777579	0.005287
VIME (only supervised)	VIME	0.010697	-0.108613	0.130006	0.330082	0.900000

**Table 5.5:** Tukey's test for comparison of Pearson's correlation between true and predicted values across different models

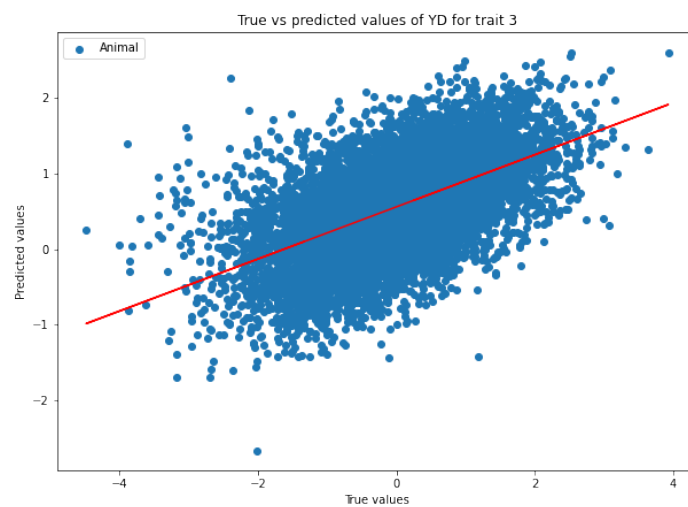
## MLP prediction by phenotype



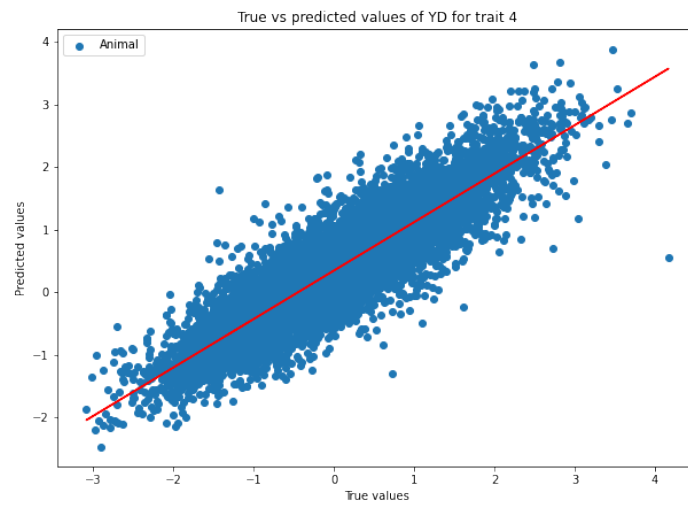
**Figure 5.28:** True vs predicted values of YD obtained by MLP model for trait 1



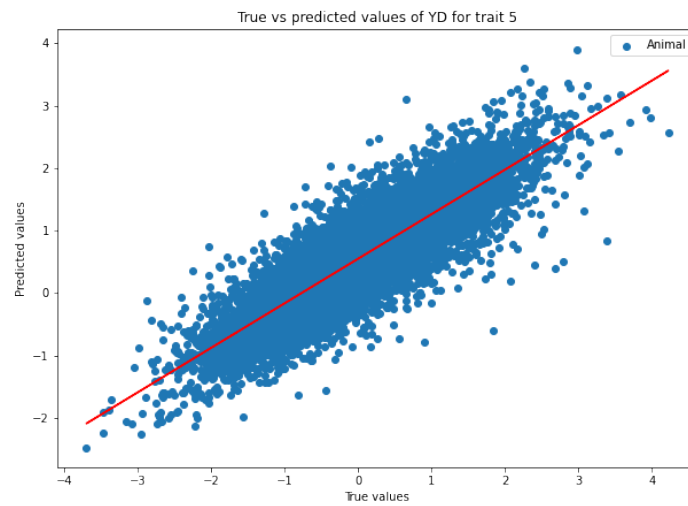
**Figure 5.29:** True vs predicted values of YD obtained by MLP model for trait 2



**Figure 5.30:** True vs predicted values of YD obtained by MLP model for trait 3

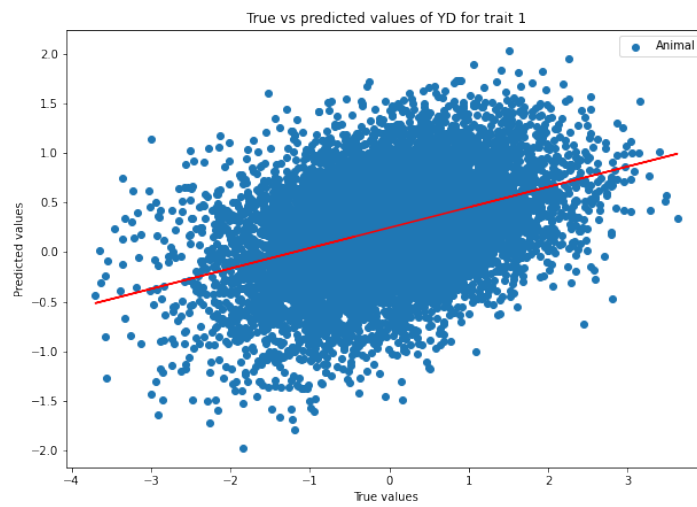


**Figure 5.31:** True vs predicted values of YD obtained by MLP model for trait 4

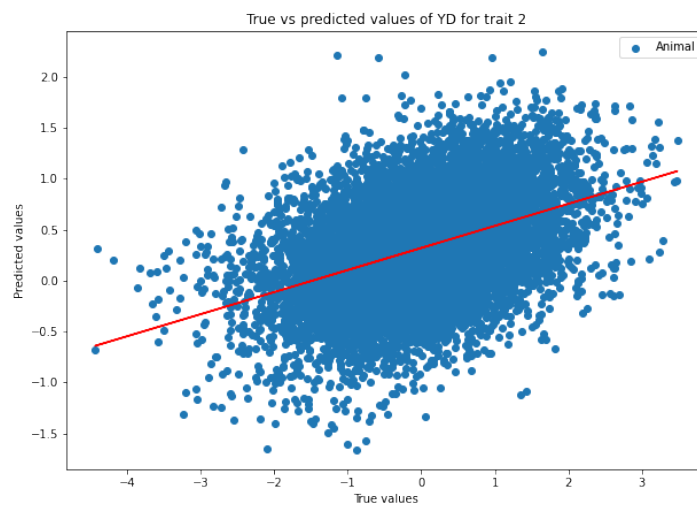


**Figure 5.32:** True vs predicted values of YD obtained by MLP model for trait 5

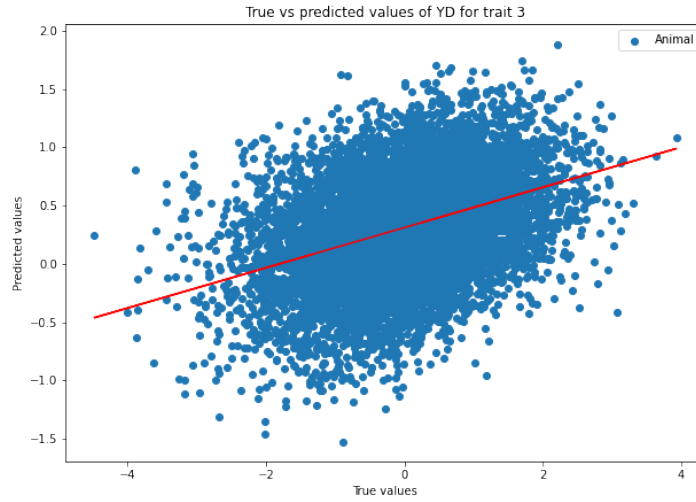
## VIME (only supervised) prediction by phenotype



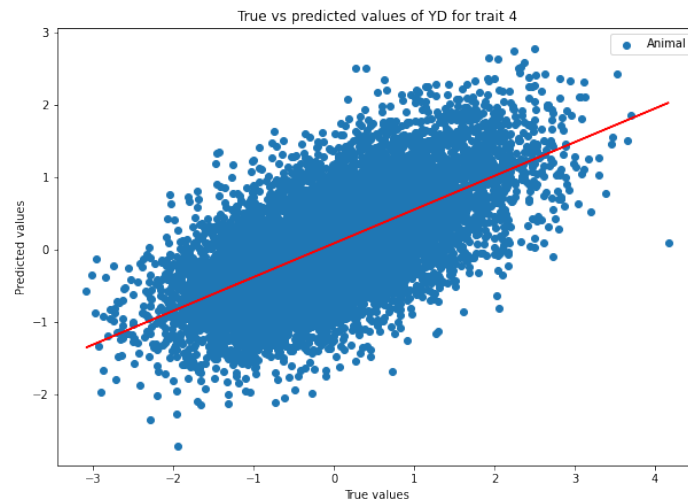
**Figure 5.33:** True vs predicted values of YD obtained by VIME (only supervised) model for trait 1



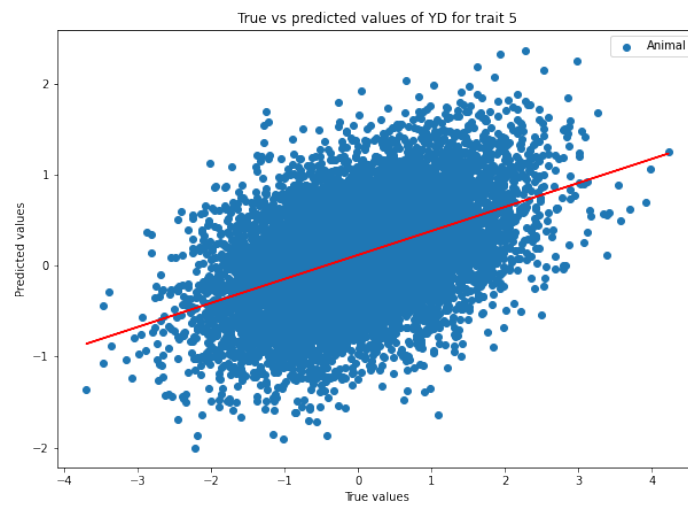
**Figure 5.34:** True vs predicted values of YD obtained by VIME (only supervised) model for trait 2



**Figure 5.35:** True vs predicted values of YD obtained by VIME (only supervised) model for trait 3

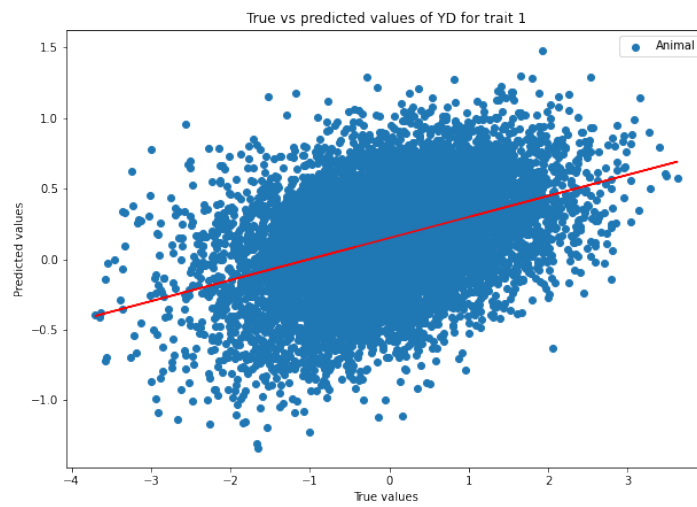


**Figure 5.36:** True vs predicted values of YD obtained by VIME (only supervised) model for trait 4

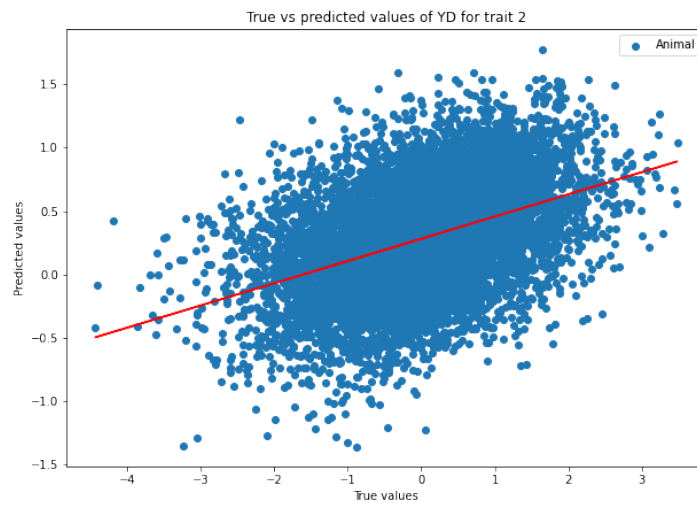


**Figure 5.37:** True vs predicted values of YD obtained by VIME (only supervised) model for trait 5

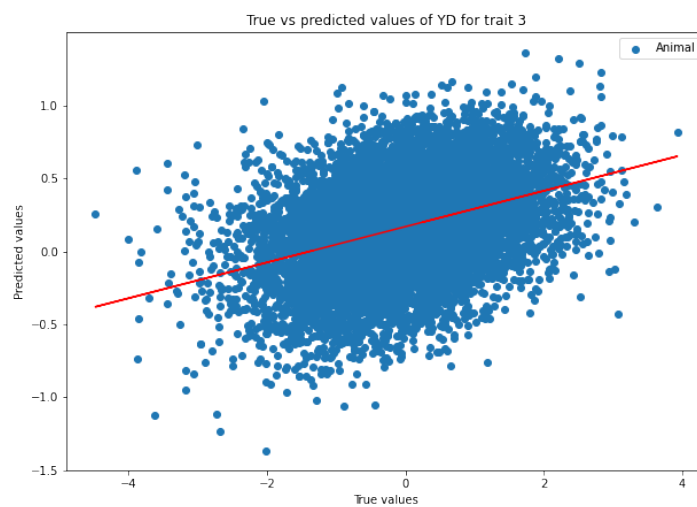
## VIME prediction by phenotype



**Figure 5.38:** True vs predicted values of YD obtained by VIME model for trait 1

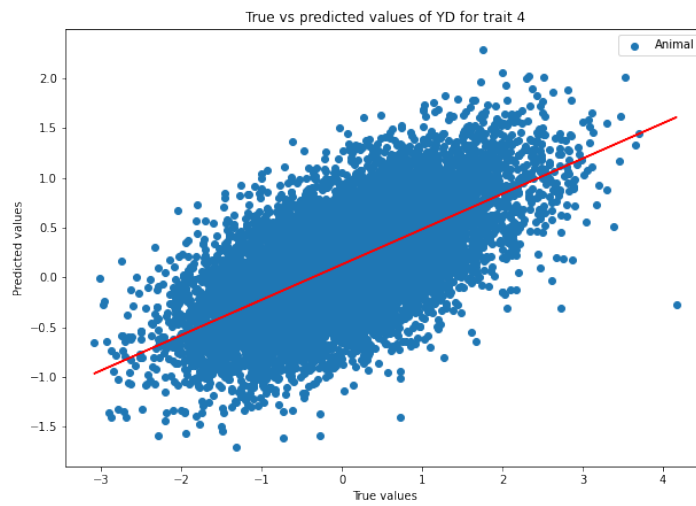


**Figure 5.39:** True vs predicted values of YD obtained by VIME model for trait 2

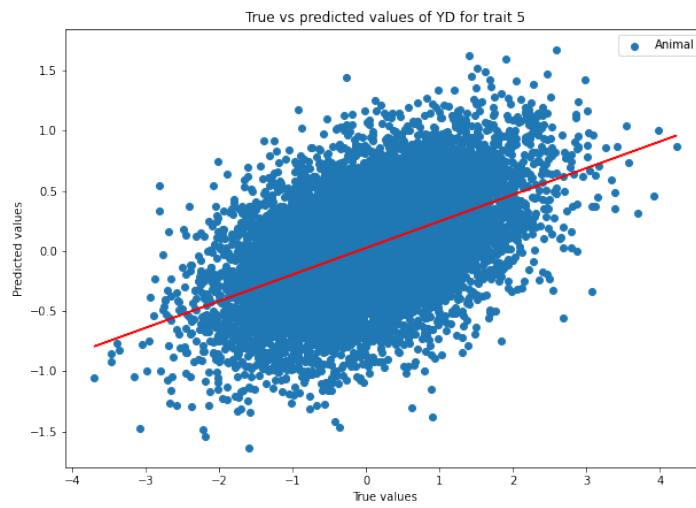


**Figure 5.40:** True vs predicted values of YD obtained by VIME model for trait 3





**Figure 5.41:** True vs predicted values of YD obtained by VIME model for trait 4



**Figure 5.42:** True vs predicted values of YD obtained by VIME model for trait 5

# Appendices

## Appendix A

# Glossary of genetics terminology

**allele** The different, alternative forms of a gene that can exist at a single locus (see dominance).

**Dominant:** An allele that expresses its phenotypic effect even when heterozygous with a recessive allele; if *A* is dominant over *a*, then *AA* (homo-) and *Aa* (heterozygot) have the same pheno(wild)type.

**Recessive:** An allele whose phenotypic effect is not expressed, a mutant (e.g.: *aa*)

**amino acid** A peptide; the basic building block of proteins (polypeptides + a C<sub>2</sub>O<sub>2</sub>H + H<sub>2</sub>N tail).

**DNA** (deoxyribonucleic acid) A double chain of linked nucleotides (having deoxyribose as their sugars); the fundamental substance of which genes are composed (see scan at end).

**gene** The fundamental physical and functional unit of heredity, which carries information from one generation to the next; a segment of DNA, composed of a transcribed region and a regulatory sequence that makes possible transcription .

**genome** The entire complement of genetic material in a chromosome set.

**genotype** The specific allelic composition of a cell-either of the entire or, more commonly, for a certain gene or set of genes; genetic characteristics (makeup) that determine the structure and function of an organism.

**heritability** It is a statistic used in the fields of breeding and genetics that estimates the degree of variation in a phenotypic trait in a population that is due to genetic variation between individuals in that population.

**heterozygote** (Gk. heteros, different) Has two different alleles of a gene; one trait can be visible (dominant) while the other can be hidden (recessive), or visible both (codominant or incomplete dominant).

**homozygote** (Gk. homo, same) Has two identical alleles of a gene either *AA* (dominant) or *aa* (recessive).

**locus** The particular physical location on the chromosome of which the gene for a given trait occurs.

**mRNA** (messenger RNA) An RNA molecule transcribed from the DNA of a gene, and from which a protein is translated by the action of ribosomes (constituting for 5% of total RNA)..

**pedigree** An ordered diagram of a family's relevant genetic features.

**phenotype** The physical appearance (makeup) of an organism controlled by its genes interacting with the environment; product of genotype.

**RNA** (ribonucleic acid) A single stranded nucleic acid similar to DNA but having ribose as its sugar and uracil rather than thymine as one of the bases.

**SNP** (single nucleotide polymorphism) A DNA sequence variation that occurs when a single nucleotide (adenine, thymine, cytosine, or guanine) in the genome sequence is altered and the particular alteration is present in at least 1% of the population.

# Bibliography

- [1] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin variable aléatoire Zidek, Anna Potapenko, Alex Bridgland, Clemens Meyer, Simon A. A. Kohl, Andrew J. Ballard, Andrew Cowie, Bernardino Romera-Paredes, Stanislav Nikolov, Rishub Jain, Jonas Adler, Trevor Back, Stig Petersen, David Reiman, Ellen Clancy, Michal Zielinski, Martin Steinegger, Michalina Pacholska, Tamas Berghammer, Sebastian Bodenstern, David Silver, Oriol Vinyals, Andrew W. Senior, Koray Kavukcuoglu, Pushmeet Kohli, and Demis Hassabis. Highly accurate protein structure prediction with AlphaFold. *Nature*, 596:583–589, 2021.
- [2] P. M. VanRaden. Efficient Methods to Compute Genomic Predictions. *Journal of Dairy Science*, 91(11):4414–4423, 2008.
- [3] D. Habier, R. L. Fernando, and J. C. M. Dekkers. The impact of genetic relationship information on genome-assisted breeding values. *Genetics*, 177(4):2389–2397, 2007.
- [4] Osvaal Antonio Montesinos-López, Abelardo Montesinos-López and José Crossa. *Multivariate Statistical Machine Learning Methods for Genomic Prediction*. Springer Cham, 2022.
- [5] Osvaal Antonio Montesinos-López, Abelardo Montesinos-López, Paulino Pérez-Rodríguez et al. A review of deep learning applications for genomic selection. *BMC Genomics*, 22(19), 2021.
- [6] Jinsung Yoon, Yao Zhang, James Jordon and Mihaela van der Schaar. VIME: Extending the Success of Self- and Semi-supervised Learning to Tabular Domain. *Advances in Neural Information Processing Systems*, 33:11033–11043, 2020.
- [7] Q. Zhang, J. Sidorenko, B. Couvy-Duchesne et al. Risk prediction of late-onset Alzheimer’s disease implies an oligogenic architecture. *Nature communications*, 11(4799), 2020.
- [8] Fahar Ibtisham, Li Zhang, Mei Xiao, Lilong An, Muhammad Ramzan, Aamir Nawab, Yi Zhao, Guanghui Li, and Yingmei Xu. Genomic selection and its application in animal breeding. *Thai Journal of Veterinary Medicine*, 47:301–310, 2017.
- [9] James Zou, Mikael Huss, Aubakar Abid, Pejman Mohammadi, Ali Torkamani, and Amalio Telenti. A primer on deep learning in genomics. *Nature Genetics*, 47:12–18, 2019.
- [10] Gökçen Eraslan, variable aléatoire Ziga Avsec, Gagneur, Julien Gagneur, and Fabian J. Theis. Deep learning: new computational modelling techniques for genomics. *Nature Reviews Genetics*, 47:389–403, 2019.
- [11] Arno van Hilten, Steven A. Kushner, Manfred Kayser, M. Arfan Ikram, Hieab H. H. Adams, Caroline C. W. Claver, Wiro J. Niessen, and Gennady V. Roshchupkin. GenNet framework: interpretable deep learning for predicting phenotypes from genetic data. *Communications Biology*, 4(1094):389–403, 2021.

- [12] Pierre Fumeron. M2 Internship Report: Application of Deep Learning to the prediction of sports performance in horses. *Laboratoire d'Informatique d'Avignon*, 2021.
- [13] Zachary R. McCaw, Thomas Colthurst, Taedong Yun, Nicholas A. Furlotte, Andrew Carroll, Babak Alipanahi, Cory Y. McLean and Farhad Hormozdiari. DeepNull models non-linear covariate effects to improve phenotypic prediction and association power. *Nature Communications*, 13(1):241, 2022.
- [14] JF Griffiths, AJ Griffiths, SR Wessler, RC Lewontin, WM Gelbart, DT Suzuki, and JH Miller. *An introduction to genetic analysis*. Macmillan, New York, 2005.
- [15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.
- [16] Kaiming He, Georgia Gkioxari, Piotr Dollár and Ross Girshick. Mask R-CNN. *IEEE International Conference on Computer Vision (ICCV)*, pages 2980–2988, 2017.
- [17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is All you Need. *Advances in Neural Information Processing Systems*, 30, 2017.