uc3m | Universidad Carlos III de Madrid

Master Degree in Statistics for Data Science

2020-2021

Master Thesis

"Application of Convolutional Neural Networks in the Multiple Testing Problem"

César Antonio Conejo Villalobos

Supervisor:

Stefano Cabras

Madrid. September 21, 2021

AVOID PLAGIARISM

The University uses the **Turnitin Feedback Studio** for the delivery of student work. This program compares the originality of the work delivered by each student with millions of electronic resources and detects those parts of the text that are copied and pasted. Plagiarizing in a TFM is considered a **Serious Misconduct**, and may result in permanent expulsion from the University.

SUMMARY

Multiple Hypothesis testing consists of a series of statistical procedures for solving hypothesis tests on high-dimensional data marginally. Several approaches have been developed for dealing with this statistical problem. Classical methods consist of defining and controlling a specific error rate. These methods usually rely on the p-values for collecting the evidence against the null hypothesis. Other alternatives have also been developed under a semi-supervised approach. In this case, we solve the large-scale testing using a null train sampling collected via endogenous or exogenous mechanisms. In this project, we combine both approaches. Employing simulations, we explore the possibility of handling cases where the user knows and controls the ground truth for solving multiple hypothesis testing problems in a supervised framework. Starting with calibrated p-values under the null hypothesis, we represent the p-values in terms of odds, converting them into lower bounds of Bayes Factors. Additionally, we take a step further, creating a matrix of the relative evidence among tests as the previously ordered minimum bounds quotients. This matrix representation is considered analogous to an image. With these ingredients, we train Convolutional Neural Networks to determine if this framework can detect the cases where the null hypothesis is rejected. We explore the ability of the CNNs to correctly classifying the hypothesis based on two primary examples. First, we study the efficiency of a diet applied to a female mice sample under several scenarios. Then, we explore the mean difference of two independent populations sampling from the normal distribution.

Keywords: Multiple Hypothesis Testing (MHT), p-value, Lower bound Bayes factors (LBBFs), Convolutional Neural Networks (CNNs), Multi-Label classification, Precision-Recall Curve, Area under Precision-Recall curve (AUPRC).

DEDICATION

I want to dedicate this work to my parents, which would have been extremely difficult to complete this project without their support.

I thank my father, Jorge, for motivating me to do my tasks and duties with excellence.

I thank my mother, Yanet, for sharing her teachings about perseverance.

Quiero dedicar este trabajo a mis padres. Sin su apoyo, hubiera sido extremadamente difícil completar este proyecto.

Agradezco a mi padre; Jorge, por motivarme a realizar mis tareas y deberes con excelencia.

Agradezco a mi madre; Yanet, por compartir sus enseñanzas sobre la perseverancia.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

In modern statistics, multiple testing focused on high-dimensional data has become an essential topic of study. Multiple hypotheses testing (MHT) corresponds to a collection of statistical procedures in which for a set of $m > 1$ null hypotheses, we want to study each statistical test individually. As a result, we obtain a partition of the m hypotheses into two sets for the cases where the null hypothesis is true and false of sizes m_0 and $m_1 = m - m_0$ respectively. The goal of MHT procedures is to discover the number of statistically significant features with the best accuracy while incurring a low proportion of false discoveries.

Typical applications of MHT methods come from biological and biomedical sciences. With the appearance of new technologies such as high-throughput systems and detectors, it is possible to obtain millions of samples of the physical activity for a specific organism. For example, microarrays allow researchers to detect gene expression[1] measurements of $m > 1$ levels per sample.

There are several approaches related to MHT. The first approach involves marginally applying the classical hypothesis test with some level of significance α. However, this method can result in several Type-I errors (False Positive, False Discovery) or Type-II errors (False Negative). Additionally, with the increasing number of the statistical test m, the number of errors occurring by chance also increases. Consequently, diverse techniques have been proposed to estimate and control an informative error rate in the setting of multiple comparisons. The typical errors rates controlled in MHT are based on two different approaches: the family-wise error rate (FWER) and the false discovery rate (FDR).

In the case of the FWER, the procedure ensures that a specific threshold bounds the probability of committing a single false-positive rejection. Although several techniques control the FWER, the traditional system under this approach is called Bonferroni's correction. However, these methods have the disadvantage of being considerably restrictive, especially in the cases where m is large. On the other hand, the FDR techniques control the rate at which the significant hypothesis is genuinely null. The philosophy of this approach consists of allowing a few false positives, as long as the majority of rejections are accurate. As a representative of this approach, we consider the Benjamini and Hochberg (BH) procedure.

Indistinctly of the three previous approaches, these MHT procedures use the p-value as the typical method of studying the evidence provided by the statistical tests from the data. Informally, the p-value represents a measure between 0 and 1 of the evidence against the null hypothesis. A typical scale of the evidence against H_0 is based on cut-offs for the p-value. Standard approaches consider a p-value of 0.01 as robust evidence against H_0

[1] Gene expression is the process in which the cells copy the DNA into the RNA.

and the range from 0.01 to 0.05 for solid evidence against the null hypothesis. Given the dependency of the p-value over the sample, the p-value is a random variable whose distribution is uniform if H_0 holds and concentrated around zero in the case that H_1 holds. Additionally, the p-values are calibrated when the theoretical sampling null distribution of the p-values is uniform.

Traditional MHT settings assume that the m p-values are calibrated based on the $U(0,1)$ distribution. However, several authors identify several disadvantages based on this assumption. For example, Cabras, 2016 declares that the settings where the p-values are calibrated are very few and apply only to simple statistical models. Irizarry and Love, 2021 advises that when we test many hypotheses simultaneously, even a tiny p-value cut-off can result in many false positives with high probability.

Consequently, given the difficulties of using the p-value, the statistical theory provides other mechanisms for studying the evidence provided by the data. Based on a Bayesian perspective, the Bayes Factors (BFs) is a likelihood ratio of the alternative hypothesis against the null hypothesis. Nevertheless, according to Cabras, 2016, fully defined and interpretable BFs require heavy computational techniques for being adjusted. Consequently, authors like Sellke et al., 2001 had proposed methods (with a low computational cost) for computing a lower bound on the odds of the null hypothesis against the alternative. Under this approach, starting with a set of calibrated p-values, we can obtain an infimum bound of their respective Bayes factors.

On the other hand, authors such as Mary and Roquain, 2021 warn about the MHT methods' dependence on the knowledge of the null distribution. As a result, they had proposed techniques based on a semi-supervised context. In this case, there is no necessary complete knowledge of the null distribution, but a sample from this null distribution is required. This sample from the null distribution is called by the authors the null training sample (NTS). In practical situations, Mary and Roquain, 2021 identifies three possible scenarios where it is possible to have a selection of the null distribution:

- Blackbox null sampling: A sampling machine can simulate according to the null distribution.
- The null sample is given: Previous experiments or expert criteria provide a fixed number of samples from the unknown null distribution.
- The Null sample is learned from data: An independent part of the data available in the experiment is used as NTS of the unknown null distribution.

During this project, we will explore an MHT procedure based on the previous ideas. Using the representation of the p-values as odds provided by Sellke et al., 2001, we use a sampling mechanism for learning from data. However, instead of a semi-supervised setting, we will explore this new procedure under a full supervised learning context. Specifically, we take a supervised approach based on multi-label classification. We intend to map

an input X to represent the evidence provided by a specific dataset to a binary response vector Y. The response vector will be assigned with a value of 1 when the alternative hypothesis is true and 0 when the null hypothesis is true.

In order to deal with our experiments, we will use Deep Learning tools, especially Convolutional Neural Networks (CNNs). CNNs are known to have remarkable performance in image classification and object detection. As a result, we consider an analogous image representation of the p-values used as input for the CNNs.

Like disclaimer, we do not intend to declare the used architectures in this project as the cutting-edge application of Deep Learning in the MHT field. Some authors like Ferrari Dacrema et al., 2020 warn about the risks of including specific architectural components in other applications domains outside Deep Learning. However, we found an interesting intersection between the ideas of Mary and Roquain, 2021, especially the cases where the null sample is given or learned from data, and the ideas of the machine learning method called **transfer learning**.

In summary, this project deals with two major topics. Chapter 2 will review the relevant concepts concerning the MHT procedures, starting with the classical hypothesis testing theory and finishing with the image representation of the p-values. This chapter will focus on detecting the possible difference between two independent groups using statistical tests. Next, chapter 3 reviews the main features of the CNNs, especially the two essential building blocks of a CNN architecture: Convolutional and pooling layers. We conclude this section with a simulated exercise applied to the efficiency of a diet based on a sample taken from a female mice population by examining the difference in the body weights of two groups. In chapter 4, applying the same architecture of the previous case, we study a simulation based on two normal populations. Finally, chapter 5 provides some relevant conclusions about this new methodology.

Concerning the software, the language used in this project is Python, especially the libraries Pandas, NumPy-SciPy, and scikit-learn. For training and calibrating the CNNs, we use the Keras API and TensorFlow. Additionally, we use the Google Colaboratory as notebooks for programming. Eventually, some of the architectures used in this project respond entirely to the restriction of the computational power provided by Google[2]. The code is available in this GitHub repository: Application_CNN_MHT_problems.

[2] 13 GB of RAM, extendable to 12 hours of GPU use. See the documentation.

2. MULTIPLE HYPOTHESIS TESTING

2.1. Hypothesis testing

Hypothesis testing is a statistical technique that detects features of some population (or populations) based on sample information. Typical applications of hypothesis testing are connected to the confirmation of certain conjectures that can be made about the parameters of the distributions or directly over the statistical models. When the researcher or statistician decides to use this statistical inference procedure, it is essential to consider two significant facts. The first issue is associated with the question: When to use or employ a hypothesis test? The second point is related to the question: Which hypothesis test to choose?

To answer the first interrogation, we can say that several authors[3] recommend using a hypothesis test only when the main objective of the analysis is to test a well-defined hypothesis. In other circumstances, others statistical techniques such as estimation and confidence intervals provide better instruments and decision criteria.

The second question requires some more elaboration. Choosing the accurate hypothesis test depends on basically five different types of factors. The first element is associated with the data and the level of measurement of the data under analysis. In this case, it is possible to identify two different types of variables: numerical and categorical. Numerical information is associated with continuous and discrete data. This type of data is usually summarized in terms of parameters, such as the mean or the median. On the other hand, categorical information is based on nominal and binary outcomes, and the data is mainly summarized in terms of proportions.

The second factor corresponds to the number of samples. Usually, hypothesis testing consists of one, two, or more groups under study. The third element consists of the purpose of the analysis. Typical applications of this statistical procedure consist of testing against a hypothesized value, comparing two statistics, the relationship between two variables, or predicting one variable in terms of another observed variable.

The fourth factor consists of the study design. The observed data can be obtained based on unpair groups in which data is collected from two or more independent groups. For instance, treatment and control groups work under this philosophy. In contrast, a pair group experiment collects data from one group before and after the phenomena under study. For example, consider the case where researchers study the efficiency of a new drug controlling high blood pressure. In this case, researchers measure the blood pressure before taking the medication. After a considerable break, the blood pressure of the participants is measured again. The firth element is connected with the assumption adopted

[3]For example, see (Wasserman, 2010, p. 150) and (Irizarry and Love, 2021, p. 63)

with the distribution of the data. In this case, normal and binomial distributions are the most typical cases. However, hypothesis testing is also possible under other non-normal distributions. Once we had defined the elements linked with the election of the appropriate type of test, we could select the most reasonable hypothesis according to the observed data and the main assumptions. Table 2.1 provides a summary of the most common techniques based on the previous factors.

	Categorical	Numerical	
One-Sample (One measure)	Test for Proportion		Test for a mean
Two-Sample	Difference of two proportions	Difference of two means (Independent samples)	
One-Sample (two measure)	Chi-sq test for independence	Regression analysis	Difference of two means (paired)

Table 2.1
Example of typical hypothesis test based on the type of data and number of samples.

Equally important than the previous questions, it is also essential to provide a brief mathematical background related to the hypothesis testing framework. First, we consider a random variable X with distribution belonging to the parametric family of distributions $\{F(\cdot;\theta) : \theta \in \Theta\}$ indexed by a parameter $\theta \in \Theta$ for whom we want to determine the validity of a hypothesis H_0 about the parameter θ against an alternative statement H_1. Therefore, we consider a partition of the parameter space Θ into two disjoint sets Θ_0 and Θ_1. The hypothesis to test has the following form:

$$H_0 : \theta \in \Theta_0 \text{ versus } H_1 : \theta \in \Theta_1$$

We can provide more details about the null and alternative statements based on definition 1. The central insight that we can obtain from this definition is the philosophy of considering the null hypothesis as the most conservative state.

Definition 1 (Null and alternative hypotheses)**.** *The null hypothesis (H_0) is the statement that is assumed true unless there is enough evidence to reject the statement. The opposite declaration is the alternative hypothesis (H_1).*

We call a **simple hypothesis** a hypothesis of the form $\theta = \theta_0$. Similarly, a hypothesis of the form $\theta > \theta_0$ or $\theta < \theta_0$. is called **composite hypothesis**.

Additionally, according to Molina-Peralta and García-Portugués, 2021, given a simple random sample $(x_1, ..., x_n)$ of X, it is possible to decide on admitting H_0 or H_1 based on a hypothesis test.

	Declared non-significant	Declared significant
H_0 True	Correct Decision (TN)	Type-I error: (FP)
H_0 False	Type-II error: (FN)	Correct decision: (TP)

Table 2.2
Possible outcomes of hypothesis testing.

Definition 2 (Test)**.** *A test of H_0 versus H_1 corresponds to a function $\phi : \mathbb{R}^n \to \{0, 1\}$ of the form:*

$$\phi(x_1, ..., x_n) = \begin{cases} 1 & \text{if } (x_1, ..., x_n) \in C \\ 0 & \text{if } (x_1, ..., x_n) \in C^c \end{cases}$$

In particular, the set C corresponds to the **rejection region** *and C^c is the* **acceptance region**. *As expected, C and C^c form a partition of the sample space $\mathbb{R}^n$.*

Thus, based on definition 2, the critical region ultimately determines a hypothesis test.

If we consider the type of experiment, we have two classes of tests. A hypothesis of the form:

$$H_0 : \theta = \theta_0 \text{ versus } H_1 : \theta \neq \theta_0$$

is called a **two-sided test** or **non-directional test**. According to Cohen, 1988, this is the most *natural* approach when we are comparing two or more populations. On the other hand, a test with any of the forms:

$$H_0 : \theta \leq \theta_0 \text{ versus } H_1 : \theta > \theta_0$$
$$H_0 : \theta \geq \theta_0 \text{ versus } H_1 : \theta < \theta_0$$

is considered as **one-sided test** or **directional test**.

Following the definition 1, we retain H_0 unless there is strong evidence to reject the null hypothesis. However, in the hypothesis testing process, we are conditioned to make two possible errors. Table 2.2 summarizes the four possible mutually exclusive outcomes of a test decision. In particular, there are two desired outcomes corresponding to the true positives (TP) and true negatives (TN). On the other hand, there are two undesired outcomes: false positives (FP) and false negatives (FN).

We can introduce some mathematical notation related to the previous errors:

- Type-I error (False Positive): Probability of rejecting the null hypothesis H_0 when we should not, given that H_0 is true.

$$\mathbb{P}(\text{Type-I error}) = \mathbb{P}(\text{Reject } H_0 \mid H_0 \text{ true}) = \mathbb{P}(X \in C | \ H_0) \leq \alpha$$

- Type-II error (False Negative): Probability of not reject the null hypothesis H_0 when we should, given that H_0 is false.

$$\mathbb{P}(\text{Type-II error}) = \mathbb{P}(\text{Do not reject } H_0 \mid H_1 \text{ true}) = \mathbb{P}(X \notin C \mid H_1) \leq \beta$$

The α and β values are generally small values used as bounds for controlling the probability of committing type-I and Type-II errors. These values are associated with the concepts of size and power of a hypothesis test.

Definition 3 (Power function)**.** *The power function of a test with rejection region C is defined by:*

$$\beta(\theta) = \mathbb{P}(X \in C \mid \theta)$$

Definition 4 (Size)**.** *The size of a test corresponds to the largest probability of rejecting H_0 when the null hypothesis is true. In other words:*

$$\alpha = \sup_{\theta \in \Theta_0} \beta(\theta)$$

A test is said to have a **significance level** *α if its size is less than or equal to α.*

Based on definition 2, the critical region C is determined as a subset of $\mathbb{R}^n$. However, we can compute it as a function of the observed sample and formulating the critical region as a subset of the range of the statistic. For example, a typical rejection region is of the form $C = \{x : T(x) > c\}$ where T is a test statistic and c is a critical value. More precisely:

Definition 5 (Test statistic)**.** *A test statistic of a hypothesis H_0 versus H_1 is a measurable function of the sample that under H_0 has a completely known distribution.*

Finally, we focus the rest of this section on the parameters associated with the reliability of the hypothesis test. The reliability consists of the precision or closeness of a sample value with which the researcher expects to approximate the relevant population value. Following the ideas of Cohen, 1988, several indirect factors can affect the trustworthiness of the test, such as the population value, shape of the population distribution, and the unit of measurement. However, four parameters can be associated directly with the reliability of a hypothesis test. Two of these parameters were previously mentioned: The significance level and the power. The following two parameters are associated with the degree to which the study phenomenon exists and the number of observed samples.

Definition 6 (Effect size)**.** *The effect size (ES) refers to the degree to which the phenomenon is present in the population.*

The effect size allows introducing the difference between being statistically significant and scientifically significant. When we reject H_0, we make the declaration that the test is statistically significant. However, although the result is declared statistically significant, the effect size can be small, resulting in a case where the declaration is not scientifically or practically significant.

Definition 7 (Sample size)**.** *The sample size corresponds to the number of observed samples from the population.*

To summarize, we have just defined the main concepts related to classical hypothesis testing and identifying the main parameters that affect the reliability or precision of the results during a hypothesis test:

1. Significance level.
2. Power.
3. Effect size.
4. Sample size.

The following section introduces the p-value, which is informally considered to measure the evidence against the null hypothesis H_0.

2.2. P-values

The p-value of a hypothesis test is a random variable related with the minor significance level α at which it is possible to reject the null hypothesis.

Definition 8 (P-value)**.** *Suppose that for every* $\alpha \in (0, 1)$ *we have a size* α *test with rejection region* C_α*. Then:*

$$\text{p-value} = \inf\{\alpha : T(x_1, ..., x_n) \in C_\alpha\}$$

Concerning the numerical value of the p-value, we can make the following asseverations:

1. If the evidence against H_0 is strong, the p-value will be small.
2. A Large p-value does not indicate substantial evidence in favor of H_0. In general, a large p-value can occur for two reasons:
   - H_0 is true.
   - H_0 is false, but the test has low power.
3. P-values do not represent the probability that the null hypothesis is true.

Given the stochastic nature of the p-values, theorem 1 reveals a critical property associated with this random variable:

Theorem 1 (P-value distribution)**.** *If the test statistic has a continuous distribution, then under* $H_0 : \theta = \theta_0$*, the p-value has distribution* $U(0, 1)$*. Therefore, if we reject* H_0 *when the p-value is less than* α*, then* $\mathbb{P}($ *Type-I error*$) = \alpha$

On the other hand, if H_0 is false, the distribution of the p-value will tend to concentrate closer to 0.

Finally, although there are several test statistics, we only focus on tests that consider two independent normal populations using small samples during this project. As a result, the test statistics used under the previous hypothesis is the unpaired t-test[4]. In the next section, we explore the cases where we conduct many hypothesis tests considering several features of the two populations.

2.3. Multiple Testing

Given the advances in electronic devices, areas like biostatistics and biomedical sciences had made possible to study and make inference in large-scale data. In this section of the thesis, we focus on statistical inference in the context of high-throughput measurements.

Large-scale or multiple hypothesis testing (MHT) refers to the situation in which we face m hypothesis test of the form:

$$H_{0i} \text{ versus } H_{1i}, \quad i = 1, ..., m$$

Generally, m is a large number reaching the hundreds, thousands, or even millions of data. The number of test m corresponds to the total number of **features** that are measured using high-throughput technologies. Additionally, the **samples** refers to the experimental units where the measurements are collected.

As opposed to the case of individual testing theory ($m = 1$), where it is only possible to commit one type of error, in the case of large-scale hypothesis testing, it is possible to make several type-I and Type-II errors during an experiment. Using the notation provided by Benjamini and Hochberg, 1995, table 2.3 summarizes all the possibilities of the m simultaneously hypothesis under a specified significance level α. In particular, from the total m test, we distinguish the following aspects:

- The number of features for which the null hypothesis is true corresponds to m_0. In this case, m_0 is associated with the *non-interesting* features.
- The number of features for which the null hypothesis is false is m_1. In particular, m_1 corresponds to the *interesting* features.

[4]For more details, see Appendix 6.1.

	Declared non-significant	Declared significant	Total
H_0 True	$U = m_0 - V$	V	m_0
H_0 False	$T = m_1 - S$	S	$m_1 = m - m_0$
	$m - R$	R	m

Table 2.3
Outcomes when testing m null hypothesis.

- R denotes the total number of hypotheses rejected; in other words, the total number of features we declare significant after applying a statistical procedure. Remember that the features are considered statistically significant when we reject the null hypothesis H_0. Additionally, $m - R$ is the total number of features that we believe non statistically significant.

- The number of features for which the null hypothesis is valid and the statistical procedure declares statistically significant is denoted by V. This number represents the total number of Type-I errors or false positives. Another popular name found in the literature for the value V is *false discovery*.

- The number of features for which the null hypothesis is false and the statistical procedure declares statistically significant is denoted by S. In particular, S represents the true positives or *true discoveries*.

- The number of features where the null hypothesis is false and is declared non-significant is denoted by T. In particular, $T = m_1 - S$ corresponds to the type-II errors or false negatives.

- $U = m_0 - V$ denotes the true negatives, in other words, the number of features where the null hypothesis is true and correctly declared as non-significant.

In general, we assume that m_0 is much greater than m_1. We can add the notation $p_0 = \frac{m_0}{m}$ as the proportion of true null hypothesis. Based on this notation, we assume values of p_0 not less than 90%. Our inference analysis goal consists of detecting as many cases for which the null hypothesis is false (S) without incorrectly detecting cases for which the null hypothesis is true (V).

When we make a large-scale hypothesis testing, only the value m is known. A-priori, we do not know the real proportion p_0. However, in our simulations, we set the values of m_0 and m_1 in order to establish the ground truth. On the other hand, R is an observable random variable. If each individual null hypothesis is tested separately at level α, then $R = R(\alpha)$ is increasing in α. Finally, V and S (also U and T) are unobservable random variables. Nevertheless, in our simulations, we know the values of these random values.

Applying the techniques of classical testing in the MHT domain can result in a considerable number of false positives. As a result, several *procedures* are defined in the context

of MHT, and the goal is to estimate and to control an informative *error rate* below a predefined value for the proposed procedure. For example, the classical hypothesis testing constitutes a particular procedure that we call *uncorrected hypothesis testing*-UHT. This procedure composes of two steps:

1. Compute the p-value of each feature.
2. Declare like significant all the features with p-value less than α.

Under the previous method, the strategy consists of controlling the **false positive rate** (FPR). The FPR constitutes the rate of genuinely null features that are declared significant.

In the next section, we explore additional procedures to estimate and control the errors rate in MHT problems.

2.4. Error Procedures

This segment explores two strategies used for controlling the error rates in table 2.3. One system constrains the probability of making one or more type-I errors. Another strategy controls the rate significant features are truly null. We start exploring the approach based on the likelihood of making Type-I errors.

2.4.1. Family Wise Error Rate

When the statistical analysis consists of a single null hypothesis H_0 with significance level α, according to theorem 1, the hypothesis test satisfies:

$$\alpha = \mathbb{P}(\text{Reject true } H_0) \tag{2.1}$$

In the case of a collection of m null hypothesis H_{0i} with $i = 1, ..., m$, the Family Wise Error Rate (FWER) is defined as the probability of making even one false rejection:

$$\alpha = \mathbb{P}(\text{Reject any true } H_{0i}) \tag{2.2}$$

There exist several methods for controlling the FWER. In this section, we consider the most popular way: The Bonferroni correction.

Definition 9 (The Bonferroni Method)**.** *Given the p-values $p_1, ..., p_m$, we reject the null hypothesis H_{0i} if:*

$$p_i < q_{FWER} = \frac{\alpha}{m} \text{ for } i = 1, ..., m$$

Theorem 2. *Using the Bonferroni method, the probability of falsely rejecting any null hypothesis is less than or equal to α, in other words, $\mathbb{P}(R) \leq \alpha$.*

According to (Efron and Hastie, 2016, p.275), methods based on controlling the FWER were initially designed for small-scale testing, for example, $m \leq 20$. Therefore, these methods are very conservative for large-scale testing, leading to a high rate of type-II errors. As a result, the second approach, based on controlling the false discovery rate, provides a more liberal criterion.

2.4.2. False Discovery Rate

In this case, we use the concept of Decision rule $\mathcal{D}$ provided by (Efron and Hastie, 2016, p.275). First, we define a decision rule $\mathcal{D}$ that rejects a total of R hypothesis with V type-I errors (False discovery). Then, we establish the **False-discovery proportion (FDP)** of $\mathcal{D}$ as:

$$\text{FDP}(\mathcal{D}) = \begin{cases} \frac{V}{R} & \text{if } R > 0 \\ 0 & \text{if } R = 0 \end{cases}$$

As expected, $\text{FDP}(\mathcal{D})$ is an unobservable random variable. However, under some scenarios, it is possible to control its expectation, called the **False Discovery Rate (FDR)**.

$$\text{FDR}(\mathcal{D}) = \mathbb{E}[\text{FDP}(\mathcal{D})] \tag{2.3}$$

One approach to control the FDR is given by Benjamini and Hochberg, 1995. In this case, we order the p-value from smallest to largest:

Definition 10 (The Benjamini-Hochberg (BH) method)**.** *Let $p_{(1)} < \cdots \cdot p_{(m)}$ denote the ordered p-values. We define:*

$$\mathcal{L}_i = \frac{i\alpha}{C_m m}, \text{ and } R = max\{i : p_{(i)} < \mathcal{L}_i\}$$

Where

$$C_m = \begin{cases} 1 & \text{if the p-values are independent} \\ \sum_{i=1}^{m} \frac{1}{i} & \text{otherwise} \end{cases}$$

Let $q_{FDR} = P_{(R)}$, we call q_{FDR} the BH rejection threshold. In this case, we reject all null hypothesis H_{0i} for which $p_{(i)} \leq q_{FDR}$.

Theorem 3. *If the Benjamini-Hochberg procedure is applied, then regardless of how many nulls hypotheses are valid and irrespective of the distribution of the p-values when the null hypothesis is false, we have that:*

$$FDR(\mathcal{D}) \leq \frac{m_0}{m}\alpha \leq \alpha$$

Figure 2.1 summarizes the errors procedures under uncorrected testing (rejection when $p_i < \alpha$, upper blue line, with $R = 4$), Bonferroni correction (rejection when $p_i < q_{\text{FWER}} = \alpha/m$, with $R = 0$) and BH procedure (rejection when $p_{(i)} < q_{\text{FDR}}$), where the threshold q_{FDR} is the most rightmost undercrossing of the upward sloping line. Under BH procedure, the number of rejections is $R = 2$.

Figure 2.1
Summary of uncorrected hypothesis test, Bonferroni correction and Benjamini-Hochberg correction.

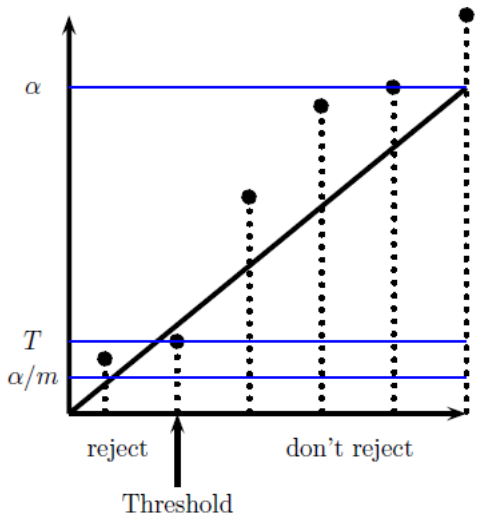


Image source: Wasserman (2010, p. 168)

2.5. Performance metrics

The situations presented in tables 2.2 and 2.3 are completely analogous to the confusion matrix used in binary classification problems. This section considers the multiple testing problem like a supervised machine learning model, specifically a binary classification problem. With this strategy, it is possible to compare the performance of the methods controlling the error rates in the large-scale testing framework. Additionally, we consider the features where the null hypothesis is false as the positive class.

In supervised learning classification, global accuracy, defined as the correct prediction ratio, is the most common metric used to measure the trained models' performance. However, we face two difficulties associated with this measure:

1. Given the low proportion of cases where the null hypothesis is false, we face an imbalanced dataset. The global accuracy can not measure the variable of interest appropriately or adequately consider the misclassification cost.

2. The significance level predefines the results obtained in the classification. As a result, changing the significance criterion will change the number of features declared

significant or non-significant.

One way of dealing with the first problem is to use a different metric called **balanced accuracy**. According to Brodersen et al., 2010, this measure corresponds to the average accuracy obtained on each class:

$$\text{balanced-accuracy} = \frac{1}{2}\left(\frac{TP}{TP+FN} + \frac{TN}{TN+FP}\right) \tag{2.4}$$

For equation (2.4), the range of the previous score is from 0 (worst) to 1 (best).

Another approach used with binary data is the Receiver Operating Characteristic (ROC) curve and the area under the ROC curve (AUROC). The ROC curve compares the true positive rate (TPR) and the false positive rate (FPR). In classification, the FPR is the ratio of negative instances that are incorrectly classified as positive. In Machine Learning literature is popular express the FPR as $FPR = 1 - \text{specificity} = 1 - TNR$, with **specificity** (True Negative Rate) computed as the ratio of negative instances that are correctly classified as negative. The ROC curve plots the FPR against the TPR for all possible thresholds. The AUROC is a useful way to compare different classifiers.

According to Aghaebrahimian and Cieliebak, 2019, the ROC is a probability curve and the AUROC measures separability. The area under the ROC curve specifies how much a model is capable of distinguishing between the classes[5]. The ROC and AUROC metrics provide two main advantages:

1. When we compute the ROC and AUROC curve for a specific predicted class, the value of AUROC and balanced accuracy are the same.

2. ROC curves allow us to determine the effect of the predefined significance level used for the hypothesis test.

However, given that the denominator of FPR considers the total number of true negatives, in the case of a skewed dataset, the ROC and AUROC can be too optimistic. As a result, another way used for evaluating the performance of a classifier is to look directly at the confusion matrix. In this case, we count the number of times instances of each class. For computing the confusion matrix, we must compare the predictions with their respective actual target. If we require a more concise metric, we can focus on the accuracy of the positive class predictions. This measure is given by the **precision** of the classifier, where precision $= \frac{TP}{TP+FP}$ with TP and FP as the true and false positive respectively. However, there are trivial ways of having a high precision metric. For example, consider the case of $m = 1\,000$ test where $m_1 = 100$. So, in case that one of the statistical procedures correctly declares as significantly one hypothesis, we have precision $= \frac{1}{1} = 100\%$. So,

[5]In our case, which observations come from the null or alternative hypothesis.

the precision metric does not consider the cases for the rest 99 features whose null hypothesis is false. In consequence, the precision is typically accompanied by the **recall**[6]. Recall consists of the ratio of positive instances that are correctly detected by the classifier, in other words: recall $= \frac{TP}{TP+FN}$, with FN referring the false negative. Given the inverse relationship between these two metrics, there exists a precision/recall tradeoff. As a result, based on their classification score and a specific decision threshold, the features are ranked. Features above the chosen threshold are considered positive. In particular, the higher the threshold, the higher the precision and the lower the recall. One way to visualize this tradeoff is obtained by plotting the precision and recall curve and the area below the precision-recall curve (AUPRC).

A perfect classifier will have $AUPRC = 1$. On the other hand, the baseline for this curve is the proportion of positive cases in our data. In the case of MHT problems, the baseline will be $p_1 = \frac{m_1}{m}$.

In summary, the precision-recall curve (and the AUPRC) is preferred in the cases where the positive class is rare or when we are more careful about the false positives than the false negatives. Otherwise, we can compare the models using the ROC curve and AUROC.

For this project, given the heavy imbalanced data, the focus of the performance on the positive class, and the precision/recall tradeoff decision based on the predefined threshold level, we consider the total AUPRC as the primary metric for comparing among all the proposed procedures. Additionally, the AUROC and the total number of rejections (R) are used only to reference the behavior of the different models.

The next step is to conclude the ways of measuring the performance of the models under the multi-label perspective. According to Koyejo et al., 2015, when we extend a set of binary metrics to a group of several labels, it is possible to construct the new metrics by averaging respect to the labels (instance-averaging) with respect to examples separately for each label (macro-averaging) or concerning both labels and examples (micro-averaging).

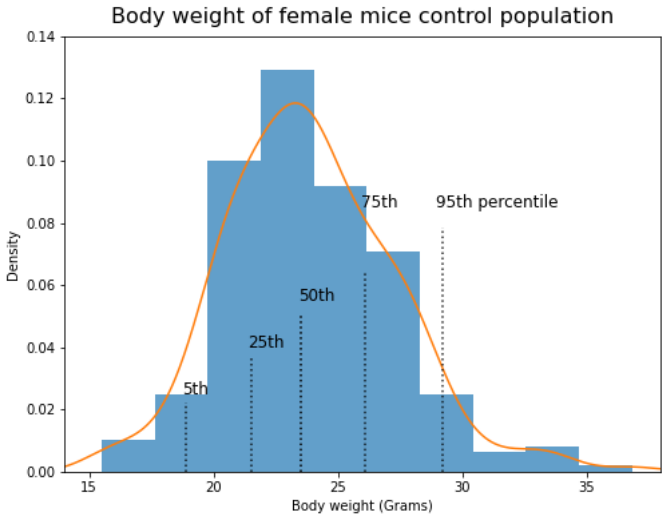
In our particular case, we focus on the performance of the models for the respective label and each example. The labels consist of the declared values with 0 (null hypothesis) and 1 (alternative hypothesis). The examples correspond to each particular MHT problem. As a result, we use micro-averaging for measuring the accuracy of the models in the multi-label setting. Additionally, based on the SK-Learn documentation, micro-average is preferred in multi-labels environments when we ignore the majority class. This scenario essentially occurs in cases of heavy imbalanced datasets.

[6] Also called sensitivity or true positive rate.

2.6. Simulation case 1: Female mice diet

To provide an example of all the previous concepts, we use the femaleControlsPopulation dataset from Irizarry and Love, 2021. The dataset contains the weight in grams of 225 female mice. This dataset corresponds to the *population* of mice from which we can sample our control and treatment sample units. Figure 2.2 shows the distribution of the female mice population. The range of body weights is from 15.51 to 38.84 grams. The sample mean of the weights corresponds to 23.89 grams with a standard deviation of 0.22 grams. First, we review some concepts related to power analysis considering classical hypothesis testing on one feature. Then, we analyze the hypothesis testing on high-dimensional data.

Figure 2.2
Distribution of the female mice body weight used as control population.



Simulation 1.1: Classical Hypothesis testing

In the classical hypothesis testing, we focus on power analysis, considering the power as a function of the significance level, the effect size, and the number of samples. Power analysis is part of research planning and lets us determine the required specifications of the experiment to reach a determined statistical power. Authors like (Irizarry and Love, 2021, p. 65) even consider this type of analysis as an ethical obligation, given that it is possible to determine the number of sample units exposed to the risk beforehand.

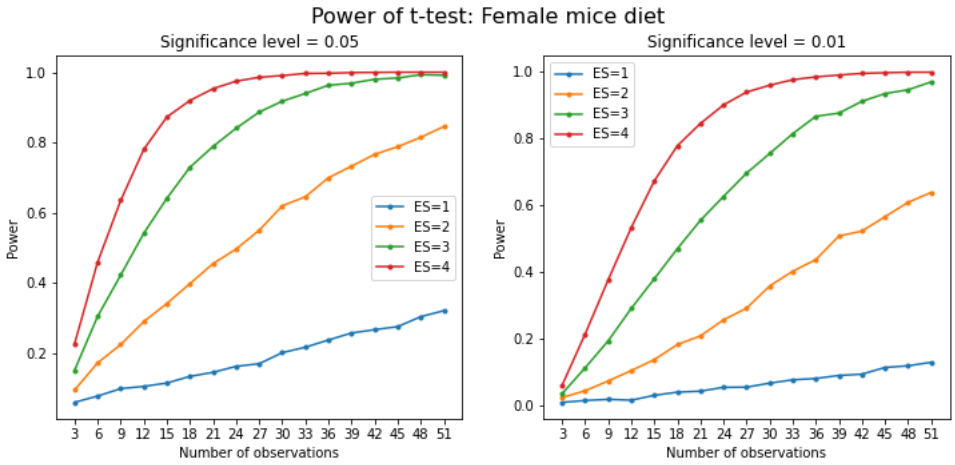
Figure 2.3 shows the simulation results for illustrating the concept of statistical power. Precisely, for predefined significance levels 0.05 and 0.01 and four effect sizes ranging from 1 to 4 grams, we compute the power in function of the number of observations from control and treatment groups and applying a t-test for comparing these two units. The power is calculated by iterating $B = 2\,000$ times and calculating the proportion of cases

where the null hypothesis is rejected.

Based on figure 2.3, we observe how after an effect size of 3 grams, independently of the significance level, we obtain relatively high power exposing the less quantity of mice. Furthermore, we notice how the statistical power has a positive relationship with the sample size. For a significance level of 0.05, after 12 sample units by group, we reach a statistical power superior to 0.5 for an ES of three grams. If we consider a significance criterion of 0.01, with the previous sample and effect size, the statistical power decrease to approximately 0.3.

Figure 2.3

Power plotted against sample size and fixed levels of ES and significance level



From the previous exercise, we obtain a crucial insight. We can observe how the effect size depends on the unit of measure (grams). However, authors like Cohen, 1988 provided metrics that allow the generalization of a universal effect size index. In particular, in the case of the difference between population means, we can standardize the difference by dividing the pooled within-population standard deviation. In summary, for two independent samples case A and B, we define d as the **ES index** of the t-test of means in standard units as:

$$d = \frac{|m_A - m_B|}{s} \tag{2.5}$$

with m_A and m_B as sample means expressed in the original units[7] and s is the pooled standard deviation of both samples. Based on expression (2.5) (Cohen, 1988, p. 40) provides conventional definitions of range of the ES index:

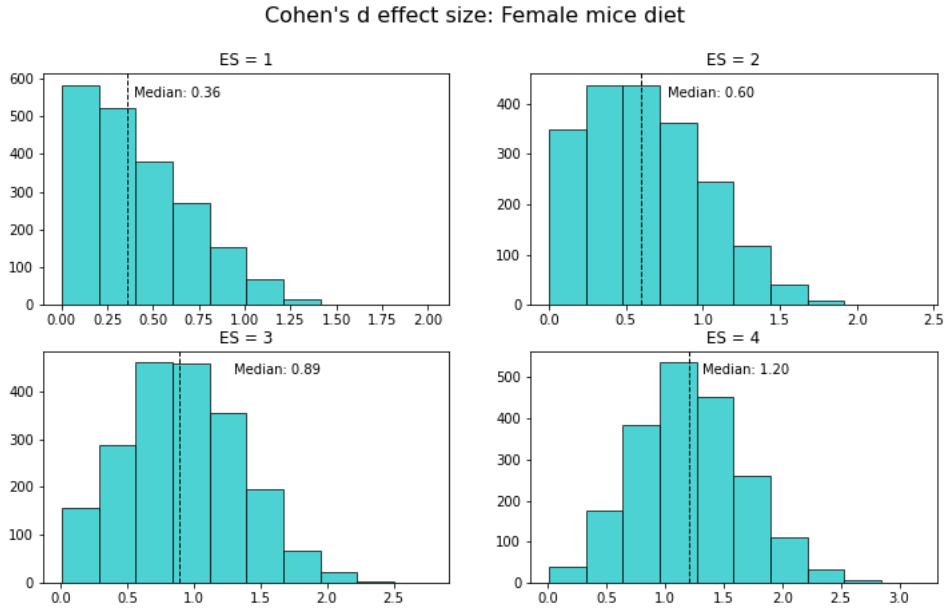
1. small: $d = 0.2$,
2. medium: $d = 0.5$,

[7]For the two-tailed test, the alternative hypothesis states only that $m_A \neq m_B$.

3. large: $d = 0.8$.

Considering the previous simulation, figure 2.4 reflects the distribution of the ES index when we sample $N = 12$ mice for each group, and the significance level is 0.05. In particular, for $ES = 1$ gram, we notice a concentration of ES index around zero with a median effect size index of 0.36. As a result, when $ES = 1$, we have a negligible effect size index. For $ES = 2$ grams, the distribution of ES index has a left-skewed form resulting in a range of small-medium effect sizes. The median, in this case, is equal to 0.6, resulting in a medium effect size index. Finally, we notice a more symmetric distribution associated with the ES index for effect sizes three and four grams. Starting with three grams, we can conclude that the effect size is considered relevant for differentiating the outcomes of diet between the treatment group and the control group.

Figure 2.4
Distribution of the effect size index.



Simulation 1.2: Multiple Hypothesis testing

Next, we consider an MHT problem where we perform tests for $m = 1\,000$ female mice diets, which 10% affect weight. Then, we design the following hypothesis test:

- H_{0i}: Diet i for $i = 1, ..., m$ does not affect weight.
- H_{1i}: Diet i for $i = 1, ..., m$ affects weight.

For the units whose null hypothesis is false, we consider an effect size over the weight has an average of three grams. For each test, we will use a sample size of $N = 12$ female mice from the population. Finally, the significance level is fixed to $\alpha = 0.05$.

	Not significant	Significant
True	860	40
False	48	52

	Not significant	Significant
True	900	0
False	98	2

	Not significant	Significant
True	900	0
False	94	6

Table 2.4

Summary results of MHT without correction, Bonferroni and BH procedure.

The goal will be to detect as many false null hypotheses as possible. To sum up, the null hypothesis is true for $m_0 = 900$ diets, corresponding to a proportion of true null hypothesis of $p_0 = \frac{m_0}{m} = 0.9$. The null hypothesis is false for $m_1 = 100$ features.

Figure 2.5 shows the histogram of the distribution of the p-values. According to theorem 1, we can consider this distribution as the mixture of the uniform distribution when the null hypothesis is true and the distribution of the concentrated p-values around zero for the cases where the alternative distribution is valid.

Figure 2.5

Histogram of p-values. Simulation considers m_1 diets having difference between the groups.

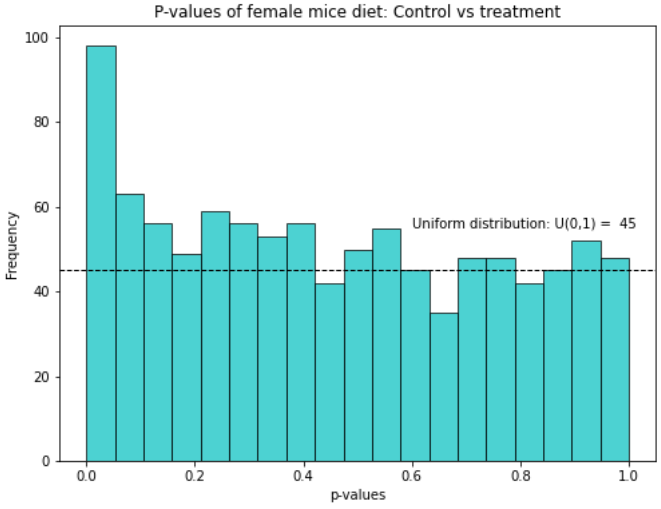


Table 2.4 shows the results of the simulated $m = 1\,000$ hypothesis test. In the hypothesis test without correction, the total rejections are $R = 92$ with a type-I error rate (FPR) of 0.04 and a type-II error rate of 0.48. The model's performance in terms of global accuracy is 0.91, and in terms of balanced accuracy is 0.74. Then, we have in the center of table 2.4 the summary table using the Bonferroni correction. In this case, we have only $R = 2$ rejections, representing a Type-I error 0 but with a costly Type-II error rate of 0.98. Global and balanced accuracy are 0.90 and 0.51, respectively. Finally, under the BH procedure, the rejections slightly increase to $R = 6$, preserving a type-I error of 0 and a Type-II error rate of 0.94. Global and balanced accuracy are 0.91 and 0.53, respectively.

	0.0001	0.001	0.005	0.01	0.025	0.05	0.1	0.2	0.3
Uncorrected	3	9	23	30	56	92	156	259	363
Bonferroni	0	0	0	1	2	2	3	5	6
BH	0	0	0	2	2	6	9	16	30

Table 2.5

Outcomes when testing m null hypothesis for specific significance levels.

The accuracy of the statistical analysis depends on the significance level used under each procedure. The left panel of figure 2.6 shows how the features are declared significant or not significant according to the control level threshold used by the BH procedure. The right panel is only a close-up of the features based on a control level of 0.05, where we can appreciate the six relevant features of table 2.4 under the BH procedure.

Figure 2.6

Distribution of the sorted p-values and two FDR control levels.

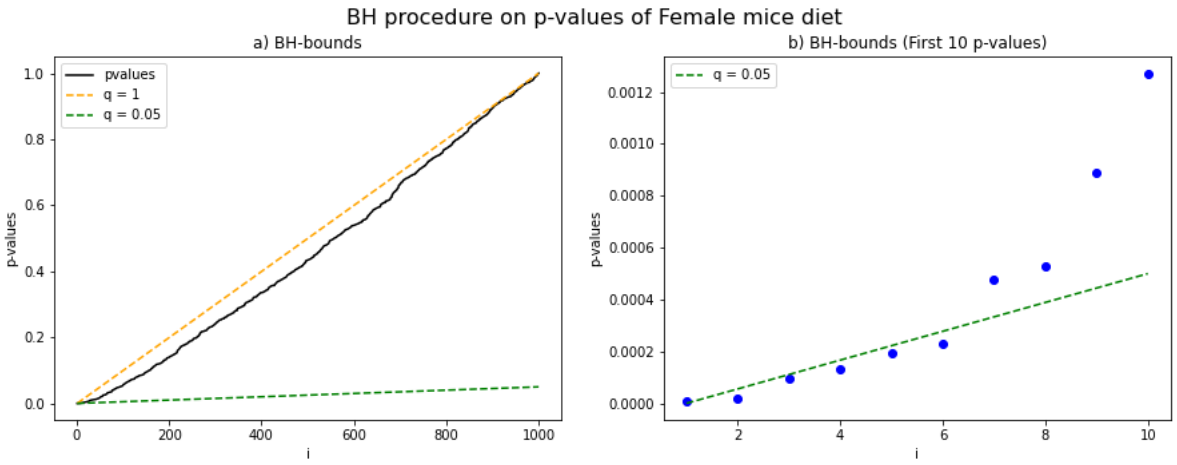


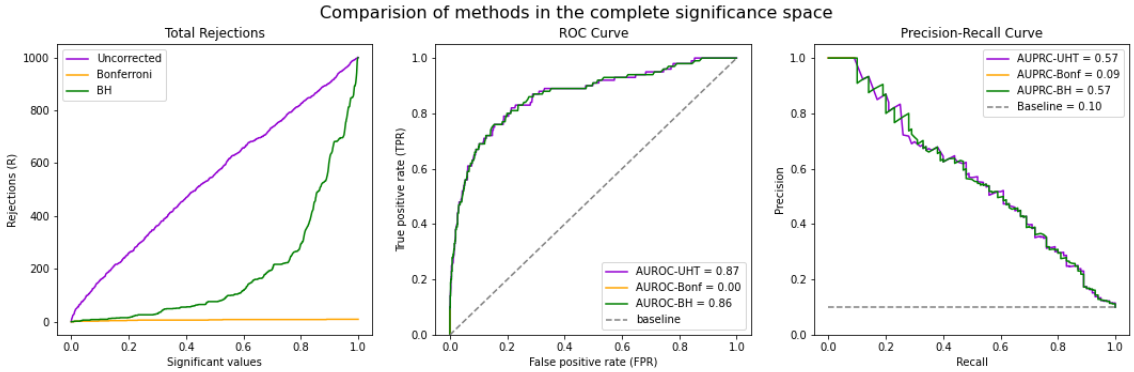
Table 2.5 shows the total number of rejections on the *significance space* ranging from 0.0001 to 0.3[8]. After a threshold of 0.1, hypothesis testing rejections without correction increase faster than Bonferroni and BH methods.

A complete overview of the performance of the models in the range from 0 to 1 is obtained in figure 2.7. In the left panel, we observe how the Bonferroni method has the lowest rejections compared to the other methods. In this situation, controlling the FWER assures a low statistical power. On the other hand, the number of rejections under BH explodes after a significance level of 0.6, reaching the total features rejected at a significance level of 1. Although the velocity of rejections is different for the Uncorrected hypothesis testing and the BH procedure, we notice how the ROC and Precision-Recall curve have similar behavior for both methods.

[8]The range from 0.0001 to 0.3 is related to the p-value calibration that we will see in section 2.7.

Figure 2.7

Performance metrics in the complete significance space from 0 to 1.



2.7. P-values in a Bayesian context

The previous analysis follows the traditional path of considering the p-values to measure the evidence against the null hypothesis model based on the input data. However, in some applications, the concept of p-value is often the victim of some misinterpretation or abuse[9].

Consequently, we introduce an alternative calibration of p-values in a Bayesian context. Under a Bayesian approach, we are interested in studying the expression $\mathbb{P}(H_0 \mid \text{Data})$. Unfortunately, it is impossible to derive $\mathbb{P}(H_0 \mid \text{Data})$ from the observed p-values. Nevertheless, it is possible to use the p-values for deriving an infimum bound of the previous conditional probability. Specifically, we compute a lower bound of the calibrated p-value that can be interpretable like an odds.

The ideas exposed in this section corresponds to the work of Sellke et al., 2001. In their paper, the authors offer two alternatives for calibrating the p-values. The first approach corresponds to a Bayesian perspective in which we obtain a lower bound of the odds provided by the data for H_0 to H_1. The second alternative considers the p-values as the conditional frequentist error probability. In both cases, the p-value calibrations are based in the context of two-side testing situations. Although the authors extend the second approach into a Bayesian framework, we focus only on the first alternative in this project.

One reason that justifies exploring new ways of measuring the evidence of a test is given by the possible contradictory results obtained if we base the discoveries of a study on p-values entirely. For example, based on the stated p-values in figure 2.5, we observe that from the 92 first p-values less than 0.05, 40 p-values (43%) corresponds to cases where the null hypothesis is true, but the premise is incorrectly rejected. If we reduce the significance level to 0.01, we only have 30 p-values less than this threshold, with five p-values (17%) being incorrect rejections.

[9]Sander et al., 2016 presents an extensive list of the most typical misunderstanding about P-value.

As a consequence, under this new interpretation, the p-value can be used to compute an inferior bound of the odds of the null hypothesis H_0 against the alternative hypothesis H_1. Following the Sellke et al., 2001 paper, their calibration assumes that the p-values have a uniform distribution under H_0 according to theorem 1. But, the innovative approach of Sellke et al., 2001 is based on considering alternative distributions for the p-value under the alternative hypothesis. In this way, under H_1, the density distribution of the p-value corresponds to $f(p|\xi)$, with ξ an unknown parameter.

In the scenario where appropriate test statistics offer large values of $T(X)$ as evidence for the alternative hypothesis, we have that under H_1, the density distribution of the p-values should be decreasing. For this reason, Sellke et al., 2001 proposes to use beta alternatives. Specifically, the recommendation of the authors consists in use a beta class distribution of the form Beta$(\xi, 1)$ with $0 \leq \xi \leq 1$. As a result, the density distribution under H_1 is given by the expression (2.6):

$$f(p|\xi) = \frac{p^{\xi-1}}{B(\xi,1)} = \frac{\Gamma(\xi+1)}{\Gamma(\xi)\Gamma(1)} p^{\xi-1} = \xi p^{\xi-1} \tag{2.6}$$

The simplified version of equation (2.6) is based on the properties of the gamma function $\Gamma(\xi+1) = \xi\Gamma(\xi)$ and $\Gamma(1) = 1$. Then, the odds (or Bayes factor) of H_0 to H_1 for a given prior density $\pi(\xi)$ under this alternative distribution is:

$$B_\pi(p) = \frac{f(p|1)}{\int_0^1 f(p|\xi)\pi(\xi)d\xi} \tag{2.7}$$

Finally, the lower bound of the Bayes Factor (2.7) is given by the equation (2.8):

$$LBBF(p) = \begin{cases} -ep\log(p) & \text{if } p \leq e^{-1} \\ 1 & \text{otherwise.} \end{cases} \tag{2.8}$$

Based on the lower bound of the Bayes factor (LBBF) for the p-value obtained in expression (2.8), table 2.6 shows the associated calibration of the p-values if we consider the significance levels used in table 2.5 and the significance level $1/e \approx 0.37$. In particular, for p-value = 0.05, the lower bound of the Bayes factor is LBBF(0.05) = 0.407. In terms of odds ratio, H_1 is just 2.5 times more likely to occur than H_0. If we reduce the significance level, we see how the odds against H_0 increase. However, this represents a considerable decrease in the power of the hypothesis test.

For an MHT problem with m features, we define the vector $\mathcal{B}^m = \{b_{(i)} = LBBF(p_{(i)}) \mid \forall i = 1, ..., m\}$ as the set of ordered lower bound Bayes factors related to the sorted p-values $p_{(i)}$ in ascending order. Returning to the collection of p-values in figure 2.5, we observe that 434 p-values are less than $1/e$. On the other hand, 566 p-values are above the previous threshold. Figure 2.8 reflects the relevant $m^* = 434$ features whose p-values is less

	0	1	2	3	4	5	6	7	8	9
p	0.0001	0.001	0.005	0.01	0.025	0.05	0.1	0.2	0.3	e^{-1}
LBBF(p)	0.003	0.019	0.072	0.125	0.251	0.407	0.626	0.875	0.982	1
Odds (H0 to H1)	399	53	14	8	4	2.5	1.6	1.14	1.02	1

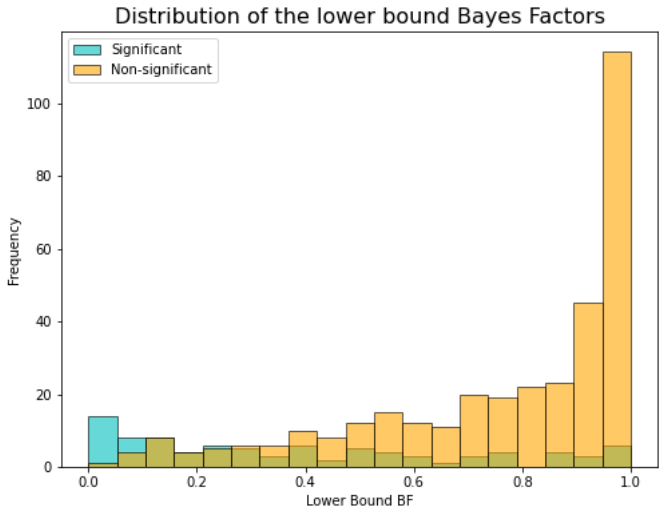
Table 2.6

Calibration p-value Bayesian context.

than e^{-1}. From the previous example, the distribution of the bounds is completely right-skewed, meaning that most of the features are roughly 1 to 1 for H_0 to H_1.

Figure 2.8

Distribution of the values of LBBF less than 1. In total, there are only 434 *features.*



In the next section, we present a new calibration of the p-values based on the previous method but adding some relationships among the p-values.

2.8. Proposal: P-values representation

We study the interaction of the lower bound Bayes factors of the m features by considering the quotient of each LBBF $b_{(i)}$ concerning the rest of the LBBFs in the vector $\mathcal{B}^m$. As a result, we will obtain a matrix B with size $m \times m$. In addition, we scale each entry down to the 0-1 range by dividing each entry by the maximum quotient of the matrix B.

Mathematically, we represent the calibrated p-values with the map $\Psi : \mathcal{B}^m \to \mathcal{M}^{m\times m}$ defined for each feature $i = 1, ..., m$ as $\Psi(b_{(i)}) = \frac{b_{(i)}}{b_{(j)}}\ \forall j = 1, ..., m$. As a result, we obtain

the matrix in expression (2.9) corresponding to the matrix of relative evidence among test.

$$B = \begin{bmatrix} \frac{b_{(1)}}{b_{(1)}} & \frac{b_{(2)}}{b_{(1)}} & \cdots & \frac{b_{(m)}}{b_{(1)}} \\ \frac{b_{(1)}}{b_{(2)}} & \frac{b_{(2)}}{b_{(2)}} & \cdots & \frac{b_{(m)}}{b_{(2)}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{b_{(1)}}{b_{(m)}} & \frac{b_{(2)}}{b_{(m)}} & \cdots & \frac{b_{(m)}}{b_{(m)}} \end{bmatrix} = \begin{bmatrix} b_{(1)(1)} & b_{(1)(2)} & \cdots & b_{(1)(m)} \\ b_{(2)(1)} & b_{(2)(2)} & \cdots & b_{(2)(m)} \\ \vdots & \vdots & \ddots & \vdots \\ b_{(m)(1)} & b_{(m)(2)} & \cdots & b_{(m)(m)} \end{bmatrix} \tag{2.9}$$

Then, we normalize each entry of matrix B in the range [0, 1] by dividing each matrix element by the maximum quotient $b_{\max} = max\{b_{(i)(j)}\}$ for $i, j = 1, ..., m$. We denote the scaled matrix of relative evidence among tests as $\bar{B}$ according to expression (2.10) with $\bar{b}_{(i)(j)} = \frac{b_{(i)(j)}}{b_{\max}}$.

$$\bar{B} = \begin{bmatrix} \bar{b}_{(1)(1)} & \bar{b}_{(1)(2)} & \cdots & \bar{b}_{(1)(m)} \\ \bar{b}_{(2)(1)} & \bar{b}_{(2)(2)} & \cdots & \bar{b}_{(2)(m)} \\ \vdots & \vdots & \ddots & \vdots \\ \bar{b}_{(m)(1)} & \bar{b}_{(m)(2)} & \cdots & \bar{b}_{(m)(m)} \end{bmatrix} \tag{2.10}$$

Based on the simulation case 1.2 of section 2.6, the left upper panel in figure 2.9 shows the representation of the p-values in grayscale. Given the considerable number of features, it is not easy to notice some insight. For this particular representation, 98% of the matrix entries have a value less than 0.005. In this situation, we notice how the highest values tend to stay in the uppermost right corner of matrix B. The right upper panel in figure 2.9 shows the representation in log scale[10]. Indeed, we notice the distribution of the lowest values in the left lowermost position of the matrix representation.

Finally, the low panel of figure 2.9 provides a reference of the distribution of the alternative hypothesis (black lines) and the null hypothesis (white blocks). As expected, after ordering the p-values, the interesting features[11] are concentrated in the left extreme of the vector. However, we notice some true alternative hypotheses over the rank of 400[12]. In the traditional MHT procedures, larger p-values[13] never cause in MTH procedures to reject H_0.

Based on the previous structure, the following chapter explores the possibility of using simulations instead of other observed information[14] to solve MHT problems. In particular, given the features and sample units, it is possible to simulate different scenarios varying the effect size and proportion of the true null hypothesis.

[10]The log scale is used only for the visual representation, not for evaluating the LBBF quotients or training the Deep Learning models in chapter 3.

[11]Interesting refers to the features whose null hypothesis is false.

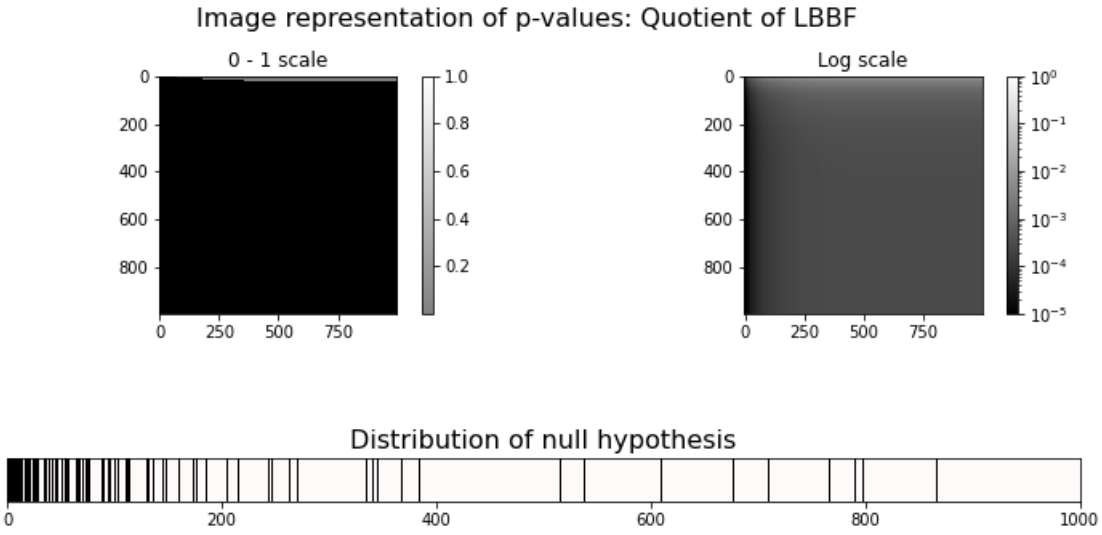
[12]Remember that the number of true alternative hypotheses is 100.

[13]Specifically, higher values than $1/e$.

[14]Using endogenous or exogenous information for sampling the null distribution was the approach employed by Mary and Roquain, 2021.

Figure 2.9

P-value representation of the MHT. In particular 98% *of the LBBF are less than 0.005.*



The simulation and posterior training of the models are based using a Deep Learning approach, essentially using Convolutional Neural Networks (CNNs). Under this point of view, we start to do the analogous of the p-values representation in the matrix (2.10) with an image and calling each matrix entry as a pixel.

CNNs typically have an accurate performance level when dealing with image objects. For this reason, it is crucial to determine the geometric structure in images that CNNs exploit. For image objects, one primary assumption of CNNs is based on the fact that data is compositional. In other words, pictures are formed of stationary patterns (essentially translation invariance) and exhibit feature locality (pixels that are more closer are more related). In our case, the matrix representation of the calibrated p-values as quotients of ordered LBBFs provides a topology relevant to the model's performance.

Finally, another reason for employing CNNs instead of other Artificial Neural Network approaches[15] is the curse of dimensionality problem. With a small number of features m, it is possible to use CNN and ANN architectures indistinctively. Nevertheless, when increasing the number of features, the number of parameters in ANN expands considerably, leading to misleading solutions or computational problems. On the other hand, CNNs is an architecture designed for high-dimensional learning problems.

[15] Such as Multilayer Perceptrons Networks.

3. CONVOLUTIONAL NEURAL NETWORKS

From the matrix representation obtained in section 2.8, we use Deep Learning (DL) frameworks to detect significant features and patterns in the scaled matrix of relative evidence among tests (2.10). We divide this section into three parts. First, we provide a brief introduction concerning the deep learning frameworks. This section focuses on Artificial Neural Networks (ANNs). The second part center on the study of the Convolutional Neural Networks (CNNs). We will review the main differences between the traditional ANNs and this particular architecture.

The idea of this section is not based on defining in great depth the philosophy of Deep Learning or explaining in profundity the concepts and processes related to this machine learning framework[16]. If the reader wants to go deep into the previous approach, the literature associated with Deep Learning is broadly abundant. For example, practical overviews of Deep Learning are available in the books of Géron, 2019 and Verdhan, 2021. Similarly, theoretical material is available in the paper of Roberts et al., 2021 and the book of Goodfellow et al., 2016.

On the contrary, the purpose of this section is to define the necessary concepts that allow us to answer the following points:

- To define the best neural network architecture for the specific problem. In other words, determine the depth and breadth of the Neural Network.
- To search the appropriate hyperparameters in order to obtain a suitable model and avoid fitting problems, specially overfitting.

Finally, the third part consists of two simulation exercises with $m = 100$ and $m = 1\,000$ features. In both practices, we continue using as the reference dataset the femaleControlsPopulation.

3.1. Artificial Neural Networks

3.1.1. Fundamentals of ANNs

This section will review the concepts properly of Deep Learning and traditional ANNs. A helpful way of study these concepts is by dig deep into the concepts of deep and learning. First, the *"deep"* in Deep Learning theory consists of successive layers of representation. As a result, the depth for a specific model relates to the number of layers in the ANN. In general, a primary artificial neural network architecture consists of three types of layers:

[16] It can take a complete thesis to reach that purpose.

1. Input Layer: It receives the input data. In the case of ANNs, the input is a 1-dimension object.

2. Hidden layer: Components of the neural networks involved in the processing, feature extraction, learning, and training of the neural network.

3. Output layer: Final piece of the neural network that makes a judgment based on the outputs of the preceding layers and generates the final result.

The layers are composed of artificial neurons, and the way these neurons are connected is crucial in the type of layers. In the case of the ANNs, the neurons in the layers are fully connected. Therefore, each neuron can deal with multiple inputs and outputs connections. The left panel of figure 3.1 provides an example where we consider a simple neural network with one input, hidden and output layer. The right panel of figure 3.1 shows how it is possible to deal with more complex models by adding several hidden layers. In particular, we notice how determining the precise number of hidden layers is the essence of neural network procedures.

Figure 3.1
Difference between a simple and Deep Learning Neural Network.
In both cases, the neurons are fully connected.

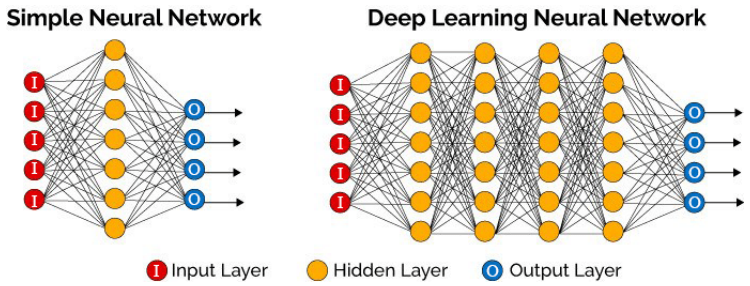


Image source: Williams et al. (2019, p. 4)

Second, it is possible to interpret the term *"learning"* as the process of adjusting the weights and the bias (intercepts) inside the network to improve the subsequent accuracy of the model. Improving the model's accuracy reduces the error term (difference between the actual value and predicted value) that is measured using a cost function. Figure 3.2 summarizes the interaction of the previous elements in the context of supervised learning for a model with one input layer receiving a dataset X, two hidden layers, and an output layer involved with the prediction of the target variable Y. After realizing an initial prediction, we can compare the accuracy of the output with the labeled actual values of the target variable. The tool used for measuring the accuracy of the model is the loss (or objective) function. The comparison of the predicted value and the true value in the objective function is the loss score. Finally, after a significant number of iterations, the neural network's purpose consists of decreasing the error rate. In order to achieve this

constant improvement, the loss score is used as a feedback signal to adjust the value of the weights via the optimizer. The backpropagation algorithm does this final process.

The training process consists in adjusting parameters for achieving the best accuracy. ANNs can learn some of these variables[17] using the raw data. However, other variables determining the neural network structure are impossible for the ANNs to learn by themselves and must be set before the training process. The last type of attribute refers to hyperparameters.

Figure 3.2

Principal components of a Neural Networks in a supervised learning context.

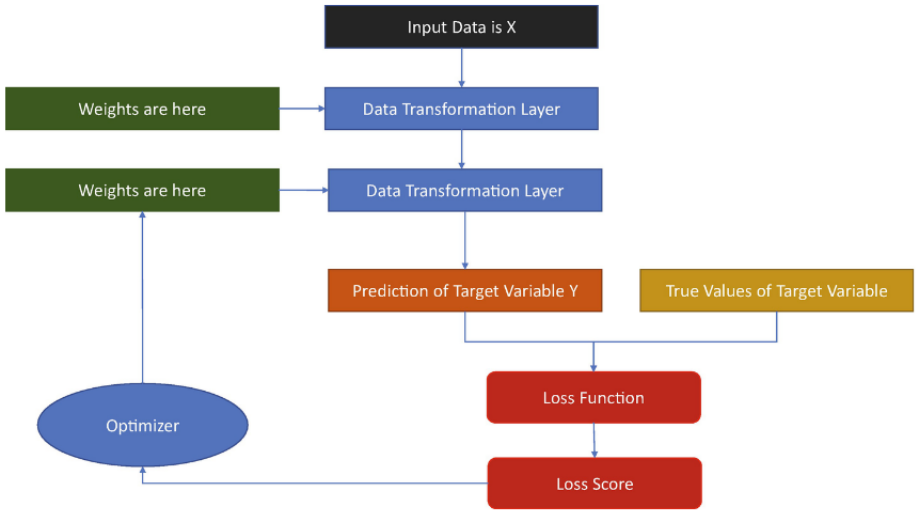


Image source: Verdhan (2021, p. 37)

3.1.2. Hyperparamerter tuning in Neural Networks

Given that it is possible to define any imaginable network architecture, several authors consider that the tremendous flexibility of Neural Networks is also one of the more relevant drawbacks[18]. Even if we use a simple ANN architecture, we need to define parameters such as the number of layers, neurons per layer, initial weights, etc. In consequence, several researchers consider that Deep Learning is more like Art than Science[19].

The list of hyperparameters in the next section does not represent the complete inventory of all the available options. It is a reference guide of the most relevant attributes, following the ideas of Géron, 2019. The first hyperparameter is related to the architecture depth of the neural network.

[17] For example, the weights and bias.
[18] See Géron, 2019.
[19] Instead of Science, some authors believe that DL is more related to engineering than Science.

Number of hidden Layers

In general, an ANN with a single hidden layer provided with enough neurons can theoretically model even the most complex functions. However, for complex problems, depth deep networks have a much higher parameter efficiency than simple networks. In other words, using the same amount of training data, depth networks using considerably fewer neurons than shallow approaches can reach a better performance.

If we focus on the input data, data have some structural properties that define the number of layers. For example, some datasets are structured hierarchically, and deep neural networks can utilize this structure. As a result, lower hidden layers model low-level structures. Intermediate hidden layers combine these low-level structures to model intermediate-level forms. The highest hidden layers (including the output layer) connect these intermediate structures for modeling the high-level features.

Additionally, the previous structure improves the ability to generalize to new datasets[20]. In other words, an ANN can use the low-level structures learned in other ANN trained with similar data and used for dealing with a similar task than our original ANN. Then, only our original ANN requires setting the specific high-level structures of the particular problem that the neural network tries to solve. As a result, it is common to reuse parts of pertained state-of-art networks that perform a similar task instead of training a model from scratch.

Finally, we must also consider the problems related to the case where many hidden layers are used. First, we increase the risk of making overfitting. Second, Given that the artificial neural networks are fully connected, deep neural networks considerably increase computation time when we train on big datasets.

Number of Neurons per Layer

Depending on the type of layer, we can define the correct number of neurons per layer into three categories. In the input layer, the number of neurons is entirely determined by the type of input. For example, in a matrix of size $m \times m$, the number of neurons in the input layer is just m^2 neurons. For the output layer, the number of neurons is predefined for the kind of task. For example, in the case of classification, the number of neurons in the last layer is just the number of classes. In our case, we want to classify each of the m features corresponds to the null or alternative hypothesis.

For the hidden layer, determining the number of hidden neurons is more complex. At this stage, we can identify three different alternatives concerning the size of the number of neurons.

1. Pyramidal: We set an initial number of neurons and then continue with fewer and

[20]Indeed, this is the principle behind some of the most relevant machine learning procedures at this moment: Transfer Learning

fewer neurons at each layer. This idea is based on the assumption that many low-level features can combine into far fewer high-level features.

2. Same number: In some cases, using the same number of neurons in all hidden layers works just as well or even better than the previous approach. This alternative also offers one advantage: it only represents one hyperparameter to tune instead of a specific number of neurons per layer.

3. Increasing number of neurons: In this case, we increase the numbers of neurons gradually until the networks start overfitting. This strategy is used with another approach called early stopping. We suspended the fitting process under the early stopping approach before the overfitting drastically biased the training process. We will give more details of the early-stopping when we discuss the number of iterations.

The next hyperparameters are related to the optimization process involved in the fitting process. In this case, the first election that we need to choose is: Which optimizer to use?

Optimizer

In general, Keras offers several optimizers that can be used for training the neural networks. Complete details are available on keras/optimizer documentation. In our case, we consider the Nadam optimizer given its high convergence speed and quality.

Batch Size

This hyperparameter has a significant impact on the model performance and training time. Large batch sizes allow the training algorithm to see more instances per second. However, we must be careful that large batch sizes often lead to training instabilities, especially at the beginning of training. The resulting model may not generalize as well as a model trained with small batch size. In our case, given the relatively low number of data, we use the default settings[21].

Learning rate

According to Géron, 2019, one recommendation for dealing with this variable is to start with a shallow scale. Then, we can plot the loss as a function of the learning rate. Next, we can reinitialize the model and train again using the appropriate learning rate.

Additionally, the applicable learning rate depends on the choose optimizer and the batch size. Finally, one crucial recommendation is that we always need to search for the correct learning rate each time we change the model's architecture or other hyperparameters. In our case, we use the default parameter.

[21] According to Keras model training documentation, the default parameter is 32.

Number of iterations

The number of total iterations of the model is called an **epoch**. In our neural networks, we consider 50 total iterations. Additionally, we add a callback technique called early-stopping. Adding this callback to the model will stop training when it measures no progress on the validation set for a consecutive number of epochs.

Initial weights

There are several options for the initial weights[22]. Based on the principle when H_0 holds, then the p-values are uniform, we consider the Heuniform as the initial weights for all the layers.

Activation function

Keras documentation provides a list of all the possible activation functions used in the model's architecture. In general, ReLU is the most common option used on the hidden layers.

In the case of the output layer, the activation function depends on the task. The framework used for solving the MHT problems is the Multi-Label classification problem. As a result, each feature is independent of other features. Consequently, the probability of each component corresponds to the Bernoulli distribution, which can be modeled using the sigmoid activation function.

3.1.3. Overfitting

Overfitting occurs when the model can learn the attributes and features accurately in the training set, but the accuracy drops on the validation and testing sets.

In general, the way of avoiding overfitting is training the data with more data. However, this can be a difficult task in some circumstances. Another more straightforward alternative is to reduce the complexity (depth) of the neural network.

There are other techniques that we can use to mitigate the problem of overfitting. Previously, we introduced the callback of Early stopping. Another method used for reducing the overfitting is to add a dropout layer. Dropout is another regularization method. During training, the output of some layers is randomly dropped out or neglected. With dropout, we will have a more noisy training process reaching a more robust solution.

Dropout is available in Keras, adding a new layer with the function dropout.

[22] See layers initializers. By default, Keras uses in the dense layers the weight glorot uniform.

3.2. Convolutional Neural Networks

The previous section shows how the neurons in ANNs consist mainly of a fully connected structure. However, this feature of ANNs can represent a problem for images with large dimensions. For example, (Géron, 2019, p. 595) provides an illustrative case in which ANNs can suffer from scalability problems. In the case of images with dimension 100×100 (For a total of 10 000 pixels), if the first hidden layer only has 1 000 neurons, we will have a total of $10\,000 \cdot 1\,000 = 10\,000\,000$ fully connections between the input and first hidden layer.

Given the high number of connections, we could reduce the number of neurons in the first hidden layer. Nevertheless, if we decrease the number of neurons, we will be very restrictive with the information transmitted to the next layer. Even with the overwhelming number of 1 000 neurons in the first hidden layer, the selected number only represents 1% of the input information. Of course, the problem can be worst if we consider more complex architectures.

In order to solve this potential problem, convolutional neural networks (CNNs) allow escalating this type of data. Specifically, CNNs solve the problem using partially connected layers and weight sharing. As a result, CNNs introduce two new building blocks: convolutional layers and pooling layers.

Additionally, there are other slight differences in the architectures of ANNs and CNNs:

1. In ANNs, every layer is composed of a long line of neurons. Given this restriction, it is necessary to flat the 2D input matrix into a 1D vector resulting in a long line of neurons. On the other hand, for CNNs, every layer is represented in a 3D tensor, with two indices referencing the spatial coordinates (Height and width of the matrix) and a third index used for indicating the number of channels. Typical applications of CNNs are based on classifying images under the scheme of 3 channels RGB. However, in our specific application, we only consider one channel[23]. This change in the input form considerably improves the match of neurons with the consecutive layers regarding their corresponding inputs.

2. Once the CNN has learned to recognize a pattern in one location, it can realize it in any other area. On the other hand, when an ANN has learned to recognize a pattern in one place, it can only remember it only in that particular location. We call this property as *local stationarity*, and it is inherited from the local invariance present in the images.

The following sections will explore the new characteristics associated with CNNs.

[23] Called gray-scale.

3.2.1. Convolutional layer

The idea of this section is to present the most relevant hyperparameters in calibrating spatial convolution over images. The function offered by Keras for dealing with images is Conv2D, and we will focus on the first four parameters: `filters`, `kernel_size`, `strides`, and `padding`.

There are a lot of theoretical aspects related to this kind of layer. Maybe, the most relevant issue constitutes the difference of the term convolution in the context of neural network and the rest of the mathematical literature. Sections 9.1 and 9.5 of Goodfellow et al., 2016 provide deep detail about the differences.

As we mentioned previously, the architecture of a CNN starts with an input image with size [height, width, channels]. For example, we begin with a size picture $[h, w, c]$. Then, we start *convolving* the image. In other words, we can move an area of size $[f_h, f_w]$ over each input image and cover each image entirely.

The $f_h \times f_w$ area moved over the entire image is named a **filter** or **Kernel**. This filter is composed of weights that are trained and updated during the modeling process. This filtering process is crucial in the procedure of detecting features in the image. For the cases on which a determined characteristic is present, the value obtained in the convolution will be a high number. On the other hand, if the feature is not present, the output will be lower. Another important parameter that we must set is how much the filter should move at each step. This parameter corresponds to the **stride** defined as the number of positions that we shift the filter in each moving with size $s_h \times s_w]$.

Finally, the output matrix after the pass of the filter over the entire input image is called **featured** or **activation map**. The feature map will have the convolutions of the filter over the whole picture. From the previous process, the dimension of the feature map is $[h - f_h + 1, w - f_w + 1]$. So, the feature map has fewer dimensions than the original spatial dimension of the original input $[h, w]$.

Specifically, we are losing the pixel on the border of the image. So, we can solve this problem by adding another parameter called **padding**. In padding, we add some pixels to an image that is getting processed. The most common practice for equaling the same height and width as the previous layer is adding p zeros to each side of the input boundaries. This approach is called zero padding. In the case of Keras, we have two options for this parameter:

1. `Same`: The output size of the activation map is equal to the number of input neurons divided by the stride. We are padding evenly with zeros to the boundaries left/up and right/down with $p_{start} = \left\lfloor \frac{s\lceil \frac{s}{I} \rceil - I + F - s}{2} \right\rfloor$ and $p_{end} = \left\lceil \frac{s\lceil \frac{s}{I} \rceil - I + F - s}{2} \right\rceil$ (With s for the stride, I for the input image and F for the filter size) zeros respectively.

One particular case occurs when $s = 1$. Under this setting, the layer's outputs will have identical spatial dimensions (width and height) as its inputs. This special case justifies the

name `same` in the parameter.

2. `Valid`: The convolutional layer does not use zero paddings ($p = p_{start} = p_{end} = 0$) and may ignore some rows and columns at the bottom and right of the input image (depending on the stride).

To sum up the previous configuration, we can set the following parameters associated with the convolutional layers and the dimension of the feature map:

1. Image size: [height, width, channels] = $[h, w, c]$.
2. filters: Number of filters = f_n.
3. Filter size: [height, width] = $[f_h, f_w]$.
4. Stride: $[s_h, s_w]$.
5. Padding: `same`, `valid`.

The output of the convolutional layer (the activation map) has dimension:

$$\left[\frac{h - f_h + p_{start} + p_{end}}{s_h} + 1, \quad \frac{w - f_w + p_{start} + p_{end}}{s_w} + 1, \quad f_n\right]$$

Figure 3.3 shows two possible examples of configurations of the convolutional layer. The input in both case is a image of dimension [h = 5, w = 7, c = 1] with padding equal to same. (zero padding)

In figure 3.3a, we set one filter of size $[f_h = 3, f_w = 3]$, stride $s = s_h = s_w = 1$ and padding $p = p_{start} = p_{end} = 1$. The dimensions of the feature map will be:

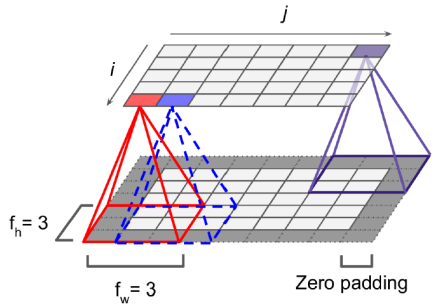
$$\left[\frac{h + 2p - f_h}{s} + 1 = 5, \quad \frac{w + 2p - f_w}{s} + 1 = 7, \quad f_n = 1\right]$$

For the figure 3.3b we apply a stride size $s = s_h = s_w = 2$, resulting in a feature map with dimension [3, 4, 1].

In deep convolutional layers, this computation can be considerably complex with a high computational cost. As a result, we can face the problem of scalability again. To solve this problem, we can use the pooling layer that we will see in the next section.

Figure 3.3
Convolutional layers

(a) *Connections bewteen layers and zero padding.*



(b) *Reducing dimensionality using a stride of 2 and zero padding.*

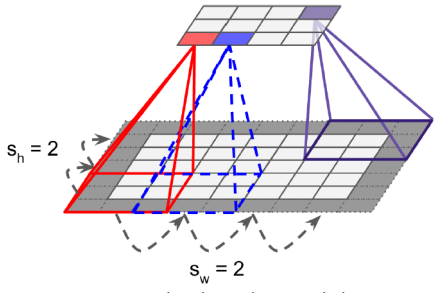


Image source: Géron (2019, p. 597)

3.2.2. Pooling Layer

Given the complexity that a deep CNN can reach, especially the number of filters of the associated feature maps that result from a convolution layer, we can decrease the dimension of this feature map using a pooling layer.

We can interpret a pooling layer like a downsampling procedure. From the feature map obtained after a convolutional layer, the pooling layer takes this input and creates an image with a lower resolution. So, for each feature map, we get a set of pooled features, where the size of the pooled filter is smaller than the feature map's size.

Similar to the convolutional layer, the relevant parameter of the pooling layer are its size (`pool_size`), the `stride`, and the `padding` type. However, the most pertinent differences concerning the convolutional layer are the absence of weights and the election of an aggregation function.

The maximum and average aggregation functions are the most common options. A different Keras function is required for the aggregation chosen, in our case, MaxPooling2D, and AveragePooling2D, respectively.

The left panel of figure 3.4 exhibits how the pooling layer operates on every input channel independently. As a result, the depth of the input and output from this layer is the same. The right panel of figure 3.4 presents a max layer with a 2 x 2 `pool_size` kernel, `stride` 2, and no padding. In this case, only the maximum input in each receptive field pass to the next layer. Additionally, the max-pooling layer drops the rest of the information. Moreover, because of the stride of 2, the output image has half the height and width of the input image.

Figure 3.4
Left: Downsampling property of pooling layers. Right: Max pool layer, of size 2×2 *and stride* 2.

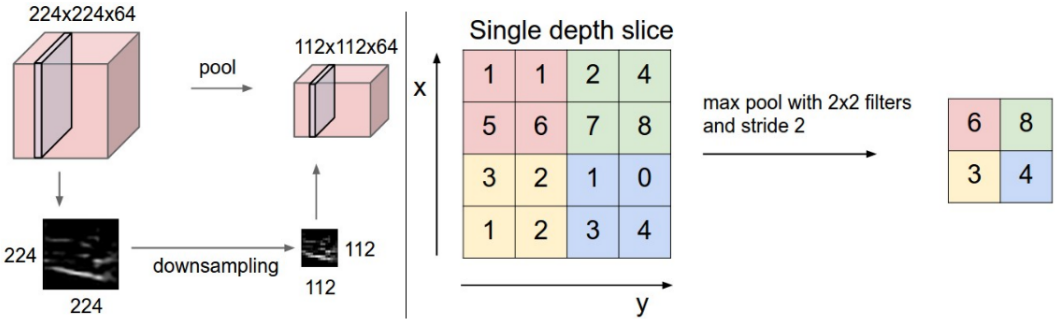


Image source: Gilon (2021, Convolutional Neural Networks (CNNs / ConvNets))

3.3. Simulation case 1: Female mice diet

In this section, we consider two Monte Carlo simulations scenarios using the female mice data set. We design two experiments considering a MHT problem with $m = 100$ and $m = 1\,000$ features. As in the case of simulation 1.2 of section 2.6, the null hypothesis H_{0i} consists that diet i does not affect weight. The alternative hypothesis declares that diet i is effective. Moreover, we rerun the tests with a sample size of $N = 12$. Finally, in both cases, we consider a range of several effect sizes and the true proportion of null hypothesis $p_0 = \frac{m_0}{m}$.

Simulation 1.3: m = 100 features

Figure 3.5 shows independent simulations of two MHT problems. In the left upper panel, we observe an input X_1 with size 100×100 from sampling 12 mice. The effect size, in this case, is a difference of 3 grams, and the alternative hypothesis is valid for $m_1 = 10$ cases. The right upper panel shows the distribution of the true response after ordering the p-values[24]. In this case, the smallest p-value is assigned to a feature where the null hypothesis is true, resulting in a potential false positive.

The lower panel of figure 3.5 reflects the X_2 representation of $m = 100$ features with a substantial effect size of 5 grams and a true proportion of the alternative hypothesis of 5%. Based on the image representations X_1 and X_2, we notice how considerable high effect sizes are related to a more separable display of the alternative and null hypothesis. The smaller the effect size, we see a more soft transition from the alternative to the null hypothesis.

[24]Consequently, ordering the LBBF values.

	Not significant	Significant
True	90	0
False	10	0

	Not significant	Significant
True	95	0
False	2	3

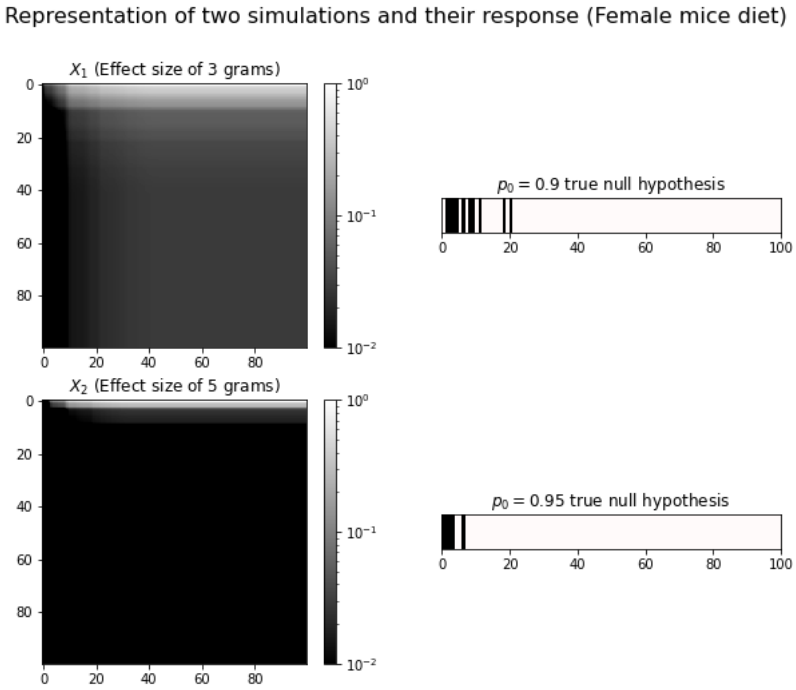
	Not significant	Significant
True	185	0
False	12	3

Table 3.1

Summary results of the two previous simulations. Right table contains the **micro** *performance of both simulations.*

Figure 3.5

Two independent simulations of $m = 100$ *features with different effect size and proportions of true null hypothesis.*



The left and central panel of table 3.1 shows the results after applying the BH procedure with a control level of $q_{\text{FDR}} = 0.05$. For representation X_1, the BH takes a conservative path and does not make any rejection. For X_2, the BH rejects correctly three of the five significant features. The right panel of table 3.1 shows the confusion matrix for quantifying the statistical declaration of all the classes jointly under a micro-averaging perspective. We compute the confusion matrix considering all the features and the MHT problems. Under this perspective, only three of fifteen hypotheses are true positives.

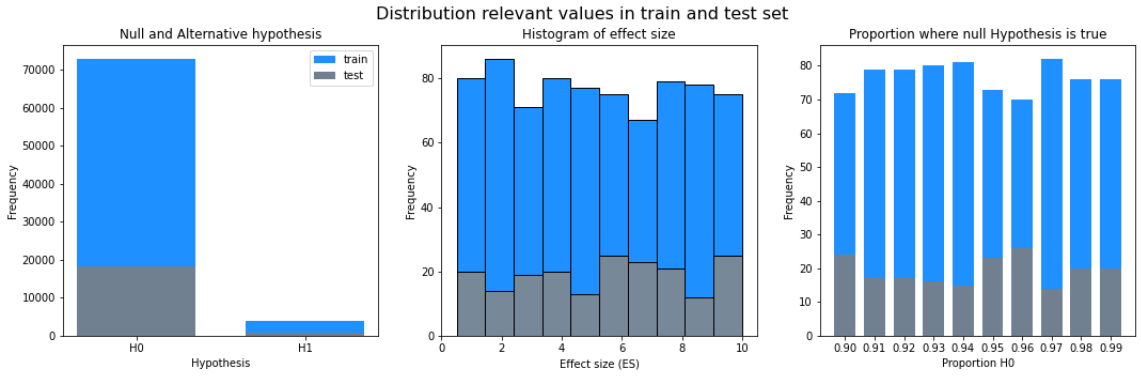
With the previous explanation, now we can extend the number of independent tests. Our simulation example considers 96 different effect sizes ranging from 0.5 to 10.0 grams with a difference of 0.1 grams between the effect sizes. The proportion of the true null hypothesis consists of 10 different values ranging from 0.9 to 0.99 with a distance of 0.1. As a result, we have a total of $96 * 10 = 960$ simulated scenarios.

Then, from the previous data, we split our observations into training and testing data. In particular, we consider 20% (192 observations) for testing. From the remaining 80% of the data (768 observations), we use 10% (77 registers) as validation data.

Figure 3.6 shows the distribution of the proportion of null and alternative hypotheses (left panel), the assignments of effect sizes (central panel), and the balance of true null hypotheses among the training and testing sets.

Figure 3.6

Distribution of the null hypothesis, Effect size and Proportion of H_0 among the train and test set.



From the left panel of figure 3.6, 91 104 observations (72 876 in training and 18 228 for testing) correspond to the genuine cases where the diet does not affect. On the other hand, 4 896 diets (3 924 in training and 972 in testing) are influential among the range of effect size and proportion discussed previously.

Given the relatively small number of features and observations, it is possible to calibrate an ANN model. The left panel of figure 3.7a shows a simple model with two hidden layers using 500 and 250 neurons and a dropout layer with a probability of 0.5. The output layer consists of 100 neurons, with one label for each feature. The total number of parameters is around 5 million parameters.

Then, figure 3.7b shows a simple architecture for a CNN model. In this case, the model starts with a convolutional layer where we apply two filters of size 2×2, stride = 1, and zero padding. Next, we use a maximum pooling layer of size two, stride equal to two, and zero padding. With this configuration, the number of parameters is reduced to half. Then, we use a similar structure of the ANN after the convolutional layer. Additionally, we add a dropout layer to reduce the risk of overfitting. The total number of parameters is around 2.6 million neurons.

Finally, for fitting both models, we use 50 epochs and the early-stopping feature. After the calibration process, the AUPRC for the ANN and CNN is 0.9 in the training set, resulting in a favorable model.

	0.0001	0.001	0.005	0.01	0.025	0.05	0.1	0.2	0.3
Uncorrected	372	518	827	877	1313	1363	2767	4923	6553
Bonferroni	104	226	331	372	434	475	518	760	785
BH	152	297	413	457	562	650	758	1019	1139
ANN	433	657	850	984	1288	1801	2754	4608	6430
CNN	427	639	837	981	1317	1797	2749	4608	6427

Table 3.2

Outcomes when testing $m = 100$ *null hypothesis for specific significance levels.*

Figure 3.7

Neural networks architectures for $m = 100$ *features.*

(a) *ANN architecture*

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten (Flatten)            (None, 10000)             0
_________________________________________________________________
dense (Dense)                (None, 500)               5000500
_________________________________________________________________
dense_1 (Dense)              (None, 250)               125250
_________________________________________________________________
dropout (Dropout)            (None, 250)               0
_________________________________________________________________
dense_2 (Dense)              (None, 100)               25100
=================================================================
Total params: 5,150,850
Trainable params: 5,150,850
Non-trainable params: 0
_________________________________________________________________
```

(b) *CNN architecture*

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 100, 100, 2)       20
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 50, 50, 2)         0
_________________________________________________________________
flatten (Flatten)            (None, 5000)              0
_________________________________________________________________
dense (Dense)                (None, 500)               2500500
_________________________________________________________________
dense_1 (Dense)              (None, 250)               125250
_________________________________________________________________
dropout (Dropout)            (None, 250)               0
_________________________________________________________________
dense_2 (Dense)              (None, 100)               25100
=================================================================
Total params: 2,650,870
Trainable params: 2,650,870
Non-trainable params: 0
_________________________________________________________________
```

Then, we proceed to compare the trained models on the test set with the traditional procedures. We start checking the total rejections and the type-I and type-II error rates for the thresholds of our significance space. Table 3.2 shows the cumulative number of significant calls for the various levels of α. In the traditional approach, the uncorrected method has the most cases of rejections. On the other hand, the correction procedures reflect the conservative number of rejections. Then, when we focus on the two Deep Learning approaches, we observe a similar behavior. Additionally, if we compare both DL frameworks with the uncorrected hypothesis test, we notice that at level 0.05, the rejections are even greater than the hypothesis testing without the correction.

The conservative behavior of the Bonferroni and BH corrections is more visible in table 3.3. Even for a significance level of 0.05, the FPR was equal to 0. On the other hand, the Type-I error rate for the uncorrected and deep learning approaches is close to the significance level α.

However, the strict control of the corrections methods over the Type-I error is paid with high values of Type-II errors. Table 3.4 shows how the Deep Learning methods considerably have the lowest values.

	0.0001	0.001	0.005	0.01	0.025	0.05	0.1	0.2	0.3
Uncorrected	0.00	0.00	0.01	0.01	0.03	0.03	0.11	0.22	0.31
Bonferroni	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.01
BH	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.02	0.02
ANN	0.00	0.00	0.00	0.01	0.02	0.05	0.10	0.20	0.30
CNN	0.00	0.00	0.00	0.01	0.02	0.05	0.10	0.20	0.30

Table 3.3

Simulation 3: Type-I error rate for $m = 100$ features.

	0.0001	0.001	0.005	0.01	0.025	0.05	0.1	0.2	0.3
Uncorrected	0.62	0.47	0.35	0.30	0.24	0.19	0.13	0.08	0.07
Bonferroni	0.89	0.77	0.66	0.62	0.55	0.51	0.47	0.42	0.39
BH	0.84	0.69	0.58	0.53	0.47	0.42	0.36	0.30	0.26
ANN	0.56	0.34	0.22	0.17	0.12	0.07	0.03	0.01	0.00
CNN	0.56	0.36	0.23	0.18	0.11	0.08	0.04	0.01	0.00

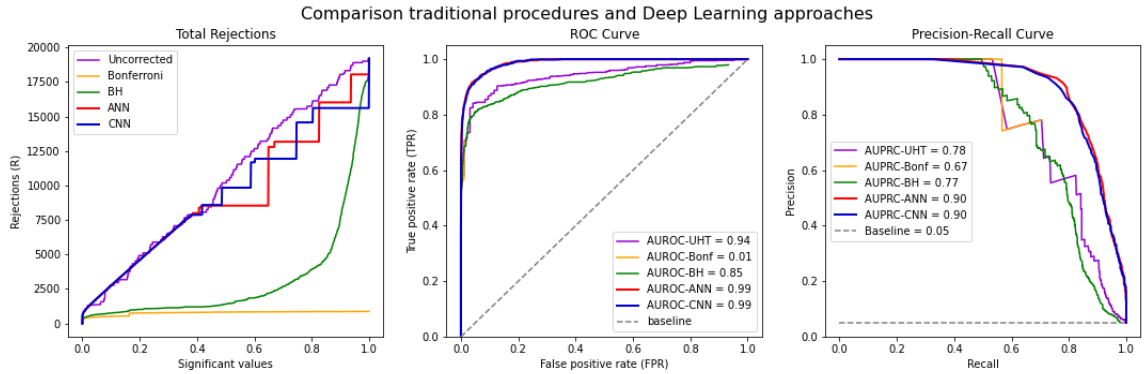
Table 3.4

Simulation 3: Type-II error rate for $m = 100$ features.

A complete overview of the model is available in figure 3.8. Based on the results of the rightmost panel, we notice how the Deep Learning methods based on the ANN and CNN architectures have the greatest values for the AUPRC[25]. Moreover, under the Deep Learning approaches, it is possible to simultaneously reach precision and recall values greater than 80%. In the second group of procedures, the uncorrected and the BH method also provide a good performance level. It is vital to notice that the BH provides a better recall around a precision level of 0.8 than uncorrected hypothesis testing but significantly lower than DL approaches. Finally, the Bonferroni correction provides an acceptable performance level. However, we reach a maximum recall level close to 70%.

Figure 3.8

Comparison traditional procedures and deep learning methods for $m = 100$ features.



[25] In particular, the value of the AUPRC in the test set is the same that the training dataset.

The left panel of figure 3.8 shows the total number of rejections from the 19 200 hypotheses available in the test set. Until the 0.4 significance level, the uncorrected and the Deep Learning methods follow a similar pattern. Then, the DL approaches increase, but the path is always lower than the UHT. Also, we notice a step behavior in both methods. In addition, for the cases of the correction procedure, the BH procedure follows the slower rejection pattern. Last, the Bonferroni method has the lowest power of all the forms, resulting in few rejections.

The central panel of figure 3.8 shows the ROC and AUROC. As mentioned in section 2.5, we observe almost a perfect metric value of 1 in ANN and CNN and a considerable high value for the hypothesis testing without correction.

In the appendix section 6.2, we show three visualizations that provide crucial insights about the previous CNN. Figure 6.1 shows the case where we do not consider the ordered lower bound Bayes factors, and their position is assigned randomly. In this case, the AUROC and AUPRC are considerably lower than the sorted version. This small simulation shows the relevance of the geometric structure in the accuracy of the model. Figure 6.2 shows three distinct image representations used as input for the previous CNN model. Suppose we only use as input of the CNN a diagonal matrix composed of the scaled LBBFs[26]. In that case, the AUPRC is considerably low than the value obtained in the proposed setting in expression (2.10).

On the other hand, if we consider only the quotients among the different LBBFs[27]; we obtain similar results in comparison to use the complete matrix. As a result, the LBBF quotients' relationship is crucial in the accuracy of the model. Furthermore, figure 6.3 shows the AUPRC for different effect sizes. Independently of the difference in grams between the groups, the CNN always has the best performance compared to traditional procedures.

Finally, If we simulate for twenty times, table 3.5 shows the average AUPRC performance (with the sample standard deviation) and the range of values for the area below the precision-recall curve. Under this experiment, the version of the CNN is considerable better than the other methodologies.

As we will review in the following simulation, increasing the number of features adds more complexity to the proposed models.

Simulation 1.4: m = 1 000 features

In this fourth simulation, we study a complex MHT considering $m = 1\,000$ features, a sample size of $N = 12$, an effect size ranging from 2 to 15 grams (with a step distance

[26]Remember that the originals LBBF were divided by the maximum component of the matrix representation.

[27]In other words, the main diagonal of input matrix is zero. We can extend this input even only to the upper triangular matrix of the quotients of the ordered LBBFs.

	Mean	SD	Min	Max
UHT	0.76	0.0145	0.64	0.88
Bonferroni	0.65	0.0131	0.55	0.78
BH	0.73	0.0125	0.61	0.85
CNN	0.84	0.0116	0.74	0.93

Table 3.5

Simulation 3: performance AUPRC for 20 simulations.

of 0.1), and three different proportions regarding the true null hypothesis of 0.9, 0.95 and 0.99. Given the high number of features, the total number of simulated observations is 390 independent problems, with 281 used for training, 31 for validation, and 78 for testing.

This stage aims to show how to deal with MTH problems with a considerable number of features. Figure 3.9a reflects an architecture with ANN. The depth of the neural network is four layers, with two hidden layers of 550 neurons and an output layer with 1 000 neurons. We also add a dropout layer. Although we are considering only 1 550 total neurons, the number of parameters in the neural network is around 300 million.

Figure 3.9

Neural networks architectures for $m = 1000$ features.

(a) *ANN architecture*

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten (Flatten)            (None, 1000000)           0
_________________________________________________________________
dense (Dense)                (None, 300)               300000300
_________________________________________________________________
dense_1 (Dense)              (None, 250)               75250
_________________________________________________________________
dropout (Dropout)            (None, 250)               0
_________________________________________________________________
dense_2 (Dense)              (None, 1000)              251000
=================================================================
Total params: 300,326,550
Trainable params: 300,326,550
Non-trainable params: 0
_________________________________________________________________
```

(b) *CNN architecture*

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 1000, 1000, 2)     20
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 500, 500, 2)       0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 500, 500, 4)       76
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 250, 250, 4)       0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 250, 250, 8)       296
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 125, 125, 8)       0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 125, 125, 16)      1168
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 63, 63, 16)        0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 63, 63, 32)        4640
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 63, 63, 32)        9248
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 32, 32, 32)        0
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 32, 32, 32)        9248
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 16, 16, 32)        0
_________________________________________________________________
flatten (Flatten)            (None, 8192)              0
_________________________________________________________________
dense (Dense)                (None, 300)               2457900
_________________________________________________________________
dense_1 (Dense)              (None, 250)               75250
_________________________________________________________________
dropout (Dropout)            (None, 250)               0
_________________________________________________________________
dense_2 (Dense)              (None, 1000)              251000
=================================================================
Total params: 2,808,846
Trainable params: 2,808,846
Non-trainable params: 0
_________________________________________________________________
```

Figure 3.9b shows a CNN created for dealing with this problem. The model architec-

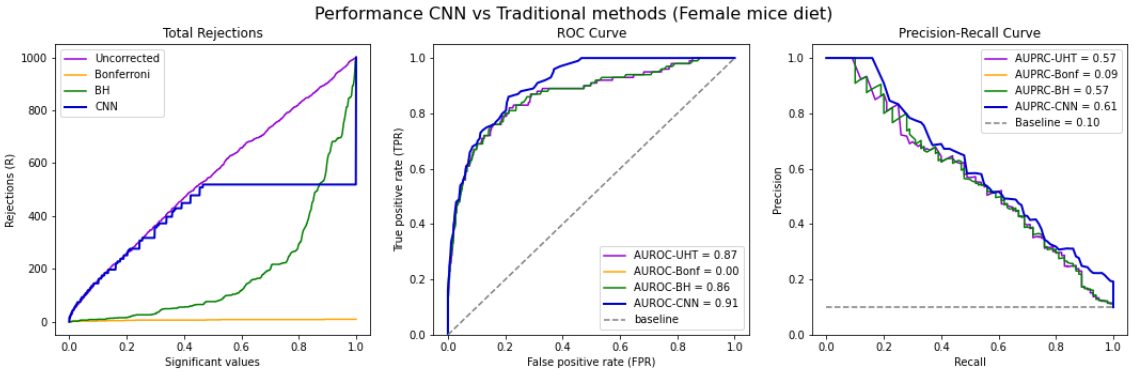
ture contains:

- Seven convolutional layers.
- Six pooling layers. The max-pooled is used for summarizing the outputs of the convolutional layers.
- Three fully connected layers, with a total of 1 550 neurons.

The size of this architecture can made overfitting a significant problem. We add a dropout layer as a regularization technique and early stopping. The number of parameters under CNN architectures reduces drastically to 2.8 million.

For measuring the performance of the previously trained model, we compare the model with the data provided in simulation 2 of section 2.6. Although we have a small training data set, figure 3.10 reflects that the performance of the CNN model over this particular evaluation is comparable with the results of the traditional methods. Consequently, we have a considerable margin for improving the accuracy of the CNN architecture, but we can reach a competitive model in contrast to the classical MHT methods.

Figure 3.10
Comparison traditional procedures and deep learning methods for $m = 1\,000$ features.



4. SIMULATION CASE 2: NORMAL POPULATIONS

This chapter provides another employment of the CNNs applied into MHT problems. On this occasion, we create a simulation following the ideas of example 4.1 in Cabras, 2016. We define $X_i \sim \mathcal{N}(\mu_i, \sigma_i^2)$, for $i = 1, ..., m$ being m independent normal populations with unknown variance σ_i^2. We want to test: $\{H_{0i} : \mu_i = 0$ versus $\mu_i \neq 0 \ \forall \sigma_i^2 > 0\}, i = 1, ..., m$. Under these experiments, the p-values are calibrated in the $U(0, 1)$. In order to observe the performance of a stable[28] CNN architecture, we use the CNN designed in figure 3.7b for simulation 1.3 of the previous chapter to a different data set. We consider a study of the following scenarios with $\sigma_i^2 = 1$ and 150 simulated normal populations under the following assumptions:

- Number of sample units: $N = 20$,
- Number of features: $m = 50, 100$ and 150,
- Proportions true null hypothesis: $p_0 = 0.9, 0.95$ and 0.99,
- Effect size: $\mu_i = \mu_A$ where $\mu_A = 0.5, 1$ and 2.

Before evaluating the CNN directly in the previous dataset, we train the CNN with another independent set of normal samples. Using the same number of sample units $N = 20$, we define three different CNNs models[29]. In each model, we consider 22 alternative means ranging from 0.3 to 2.4 (with a distance of 0.1 units among them) and ten proportions of true null hypothesis between 0.9 and 0.99. For the features $m = 50$ and $m = 100$ we simulate 200 different samples, resulting in 44 000 simulated data sets[30]. For $m = 150$, the number of simulations is 150 datasets[31].

Figure 4.1 shows the evaluation of the trained CNNs in the initially described dataset. We use it as a Benchmark for measuring the performance of CNN, comparing the results with the BH procedure. A priori, the behavior of the CNN is similar to the BH procedure.

However, we can extract two main insights. First, if we focus on the models with $\mu_A = 0.5$, the AUPRC for BH and CNNs is generally wrong. However, we notice that the CNN is equal to the proposed baseline $p_1 = m_1/m$. BH is always lower than the baseline. This behavior is the same independently of the number of features m. To put it another way, under the presence of low statistical power for detecting the alternative hypothesis, CNN better catches the cases where the null hypothesis is false than the BH procedure.

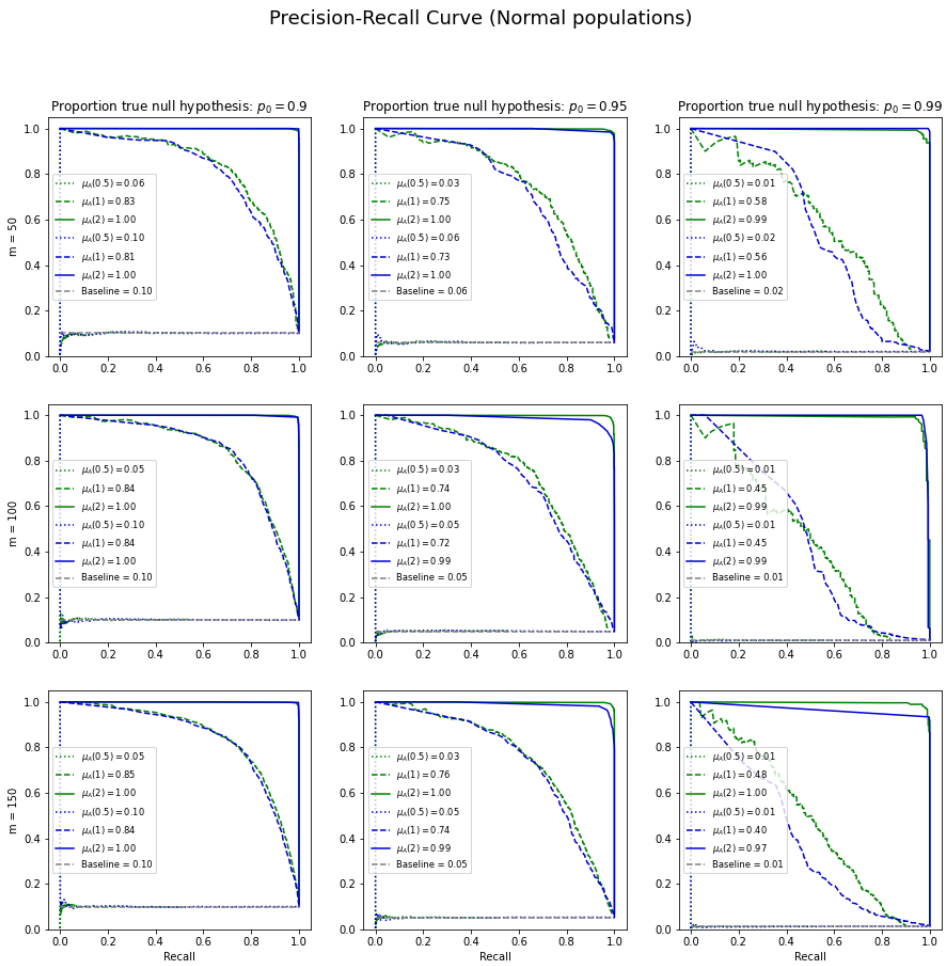
[28] By stable, we mean that we can apply the same architecture and parameters to several MHT problems.
[29] One for each value of $m = 50, 100, 150$.
[30] We have $22 * 10 = 220$ different scenarios. Then, we consider 200 times each scenario.
[31] We decrease the number of simulations due to restrictions in Colab. The total number of available datasets, in this case, is $22 * 10 * 150 = 33\,000$.

Lastly, for $\mu_A = 1$ and $\mu_A = 2$, we notice how the BH is better than the CNN alternative, but we do not find significant differences between the methodologies.

From the previous simulation, we extract several conclusions. First, there is still a margin for improving the CNN calibration. Second, this particular evaluation suggests that maybe it is necessary to design a new architecture depending on the number of features m and the specific dataset[32]. Third, although we have different datasets with the same number of elements ($m = 100$), the hyperparameters are not necessarily the same[33]. Fourth, similarly to simulation 1.4, we obtain a complete competitive method compared to the traditional approaches.

Figure 4.1

Precision-Recall curves after testing zero normal mean with unknown variance. Top scale: True proportion of null hypothesis p_0. Left scale, number of features $m = 50, 100, 150$. Green lines refers to the BH procedure and blue is used for the CNN method.



[32] Even in the cases where the number of features is relatively close.

[33] Remember that we are using the optimal hyperparameter founded in simulation 3 for the female mice diet.

5. CONCLUSION

During this project, we explore a new alternative for studying MHT problems implementing Convolutional Neural Networks. Starting with calibrated p-values and using the Sellke et al., 2001 representation, we create a square matrix with size equal to the m features for the p-values calibrated as the quotient of the ordered lower bound Bayers factors. Then, we normalize the quotients on the scale of 0-1 by dividing each pixel of the matrix by the maximum LBBF quotient. We use the word pixel given the analogy of the matrix representation with an image using one channel. This image representation contains in the diagonal the calibration of the p-value as odds of H_0 to H_1. Moreover, it also incorporates in the off-diagonal the interaction among the quotients. The scaled version of the LBBF quotients corresponds to the scaled matrix of relative evidence among the tests.

Considering the sorted LBBF and the relationship between the quotients, the idea is to use CNNs to solve MHT problems in a simulated context where the user knows the ground truth. Based on this approach, we generated two simulation cases. In the first case, we make simulations sampling the true control female mice population. In particular, with $N = 12$ sample units, we guarantee that we have enough statistical power in our experiments. In this first stage, we also split our analysis considering $m = 100$ and $m = 1\,000$ features. For the first scenario with one hundred features, we create a simple CNN model with one convolutional layer, one pooling layer, and two hidden layers. Next, we proceed with the hyperparameter tuning for solving this problem efficiently, resulting in 500 and 250 neurons in the hidden layers, respectively. Then, we include an output layer where the number of neurons depends on the number of features. Based on the 20 generated simulations, we notice how the CNN is highly efficient in detecting the significant features compared to the traditional procedures. In the second stage, we try to solve the same problem but considering one thousand features. This exercise aims to show how to scale the architecture of CNN with a high number of features. The complex architecture used in this case results from a considerably high number of parameters and computational restrictions. The comparison of the CNN model with the other procedures has no significant improvement in this particular evaluation.

In the second simulation exercise, we used the same architecture designed for the previous simulation but applied it to detect the differences between two normal populations. However, in this particular evaluation based on 50, 100, and 150 features, the results are very similar to the BH procedure. We obtain a crucial insight on this secondary simulation: The best hyperparameters obtained in one calibration are not necessarily the same for other MHT problems.

Although the first simulation gives satisfactory results of the possible application of CNNs into MHT problems, we must also be aware of this approach's potential drawbacks. The first problem is innate to the Deep Learning procedures, and it is related to

the loosing of inferential capabilities. Technically, the results obtained under the CNN can be considered as a black box for the researchers. The second obstacle is related to the computational problem. Typical applications of the MHT problems are related to millions of features. Given the computational restrictions, we could only explore as many $m = 1\,000$ features. The third difficulty is associated with the dependency of the p-values to a known null distribution. The representation provided by Sellke et al., 2001 works under the assumption that the p-values are calibrated under the uniform distribution.

However, the last two problems can be the source for future investigation. First, in order to deal with the computational problem, we can use sparse models. In particular, the matrix representation of relative evidence among tests defined in expression (2.10) considered twice the relationship between the quotients. Under sparse models, we could use the upper triangular matrix and explore if only one quotient correlation is enough for training the CNN. The right panel of figure 6.2 shows a possible opportunity of success based on the simulation exercise for the female mice diet with $m = 100$ features. In this particular evaluation, we observe only a slight decrease in the AUPRC of the complete representation compared to the upper matrix of the lower bound Bayes factors quotients.

The problem related to the assumption of calibrated p-values is solved by considering other approaches for representing the p-values. It is possible to extend the matrix of relative evidence among tests to cases where p-values are not necessarily uniform under the null hypothesis. For example, in Cabras and Castellanos, 2017, authors consider an empirical null distribution of p-values estimated directly from the data.

In other words, with the previous strategies, it will be possible to extend the application of CNNs to more general scenarios, including a large number of features and more general sampling null distributions.

6. APPENDIX

6.1. Appendix I: The t-test for means

This section assumes two independent random variables with distributions $X_1 \sim \mathcal{N}(\mu_1, \sigma_1^2)$ and $X_2 \sim \mathcal{N}(\mu_2, \sigma_2^2)$ with unknown means and variances. The exercise will be to test the equality of means by taking equal variance and a non-directional test. If we consider the estimator $\theta = \mu_1 - \mu_2$, the hypothesis test has the form:

$$H_0 : \theta = 0 \text{ versus } H_1 : \theta \neq 0 \tag{6.1}$$

The estimator of θ is simply the difference of the sample means $\hat{\theta} = \bar{X}_1 - \bar{X}_2 \sim \mathcal{N}\left(\mu_1 - \mu_2, \sigma^2\left(\frac{1}{n_1} + \frac{1}{n_2}\right)\right)$. In the case that we estimate σ^2 as the pooled variance:

$$S^2 = \frac{(n_1 - 1)S_1^2 + (n_2 - 1)S_2^2}{n_1 + n_2 - 2}$$

with S_i^2 $i = 1, 2$ corresponding to the sample variances for X_1 and X_2 respectively, we have the following test statistic:

$$T = \frac{\bar{X}_1 - \bar{X}_2}{S\sqrt{\frac{1}{n_1} + \frac{1}{n_2}}} \sim t_{n_1+n_2-2} \tag{6.2}$$

As a reference, a random variable T has t-distribution with k degrees of freedom has the probability density function:

$$f(t) = \frac{\Gamma\left(\frac{K+1}{2}\right)}{\sqrt{k\pi}\left(\frac{k}{2}\right)\left(1 + \frac{t^2}{k}\right)^{(k+1)/2}}, \quad t \in \mathbb{R} \tag{6.3}$$

Finally, under a significance level of α, the critical region for the hypothesis test in expression (6.1) is $C_\alpha = \{|T| > t_{n_1+n_2-2;\, \alpha/2}\}$.

6.2. Appendix II: Simulation 1.3. Female mice diet plots

Figure 6.1

Comparison of traditional procedures and deep learning methods for $m = 100$ features with random assignation of the LBBF.

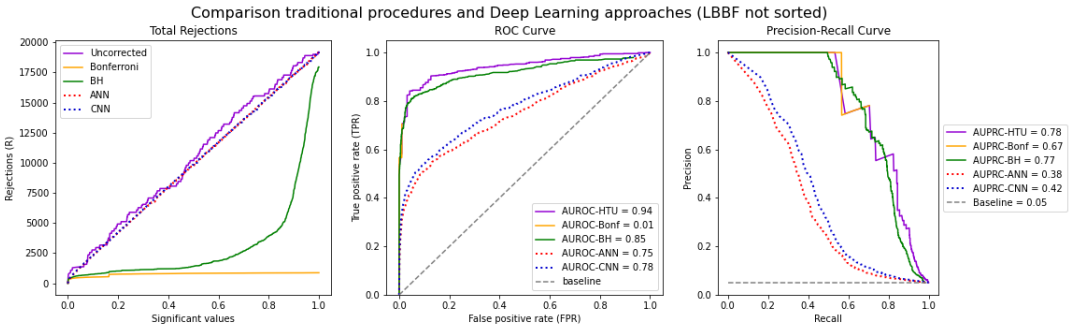


Figure 6.2

Precision-Recall curve and AUPRC from simulation 1.3 by components. Left: Main diagonal with scaled lower bound Bayes Factor. Center: Matrix interaction with only the quotients of the LBBF (main diagonal equal to zero). Right: Upper triangular matrix with the interactions form matrix in the central panel.

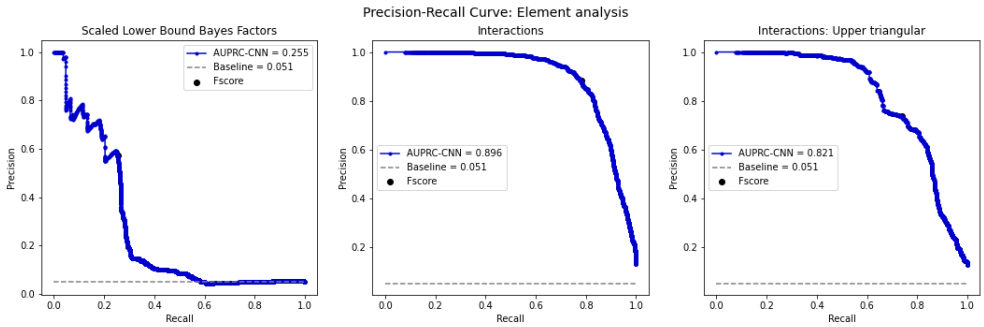
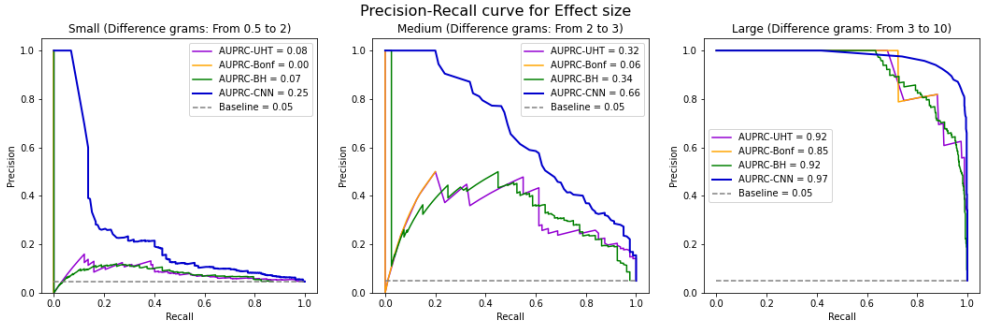


Figure 6.3

Comparison of traditional procedures and deep learning methods for $m = 100$ features with different effect size. Left: Small effect size, ranging from 0.5 to 2 grams. Center: Medium effect size ranging from 2 to 3 grams. Right: Large effect size ranging from 3 to 10 grams.



BIBLIOGRAPHY

Aghaebrahimian, A., & Cieliebak, M. (2019). Towards integration of statistical hypothesis tests into deep neural networks. *Arxiv*.

Benjamini, Y., & Hochberg, Y. (1995). Controlling the false discovery rate: A practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society*, *57*(1), 289–300.

Brodersen, K. H., Ong, C. S., Stephan, K. E., & Buhmann, J. M. (2010). The balanced accuracy and its posterior distribution. *20th International Conference on Pattern Recognition, IEEE*.

Cabras, S. (2016). A markov chain representation of the multiple testing problem. *Statistical methods in Medical Research*, *27*(2), 364–383.

Cabras, S., & Castellanos, M. E. (2017). P-value calibration in multiple hypothesis testing. *Statistics in Medicine*, *36*(18), 2875–86. https://doi.org/10.1002/sim.7330

Cohen, J. (1988). *Statistical power analysis for the behavioral sciences*. Routledge.

Efron, B., & Hastie, T. (2016). *Computer age statistical inference, student edition: Algorithms, evidence, and data science*. Cambridge University Press.

Ferrari Dacrema, M., Parroni, F., Cremonesi, P., & Jannach, D. (2020). Critically examining the claimed value of convolutions over user-item embedding maps for recommender systems. *Proceedings of the 29th ACM International Conference on Information and Knowledge Management*. https://doi.org/10.1145/3340531.3411901

Géron, A. (2019). *Hands-on machine learning with scikit-learn, keras, and tensorflow: Concepts, tools, and techniques to build intelligent systems*. O Reilly Media.

Gilon, Y. (2021). Cs231n convolutional neural networks for visual recognition.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning* [http://www.deeplearningbook.org]. MIT Press.

Irizarry, R., & Love, M. (2021). *Data analysis for the life sciences*. Leanpub.

Koyejo, O., Ravikumar, P., Natarajan, N., & Dhillon, I. S. (2015). Consistent multilabel classification. *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*, 3321–3329.

Mary, D., & Roquain, E. (2021). Semi-supervised multiple testing.

Molina-Peralta, I., & García-Portugués, E. (2021). *A first course on statistical inference. lecture notes*. Boowdown.

Roberts, D. A., Yaida, S., & Hanin, B. (2021). The principles of deep learning theory.

Sander, G., J., S. S., J., R. K., Charles, C. J. B. P., N., G. S., & G., A. D. (2016). Statistical tests, p values, confidence intervals, and power: A guide to misinterpretations. *European Journal of Epidemiology*, *31*(4), 337–350. https://doi.org/10.1007/s10654-016-0149-3

Sellke, T., Bayarri, M., & Berger, J. O. (2001). Calibration of p values for testing precise null hypothesis. *The American Statistician*, *55*(1), 62–71. https://www.jstor.org/stable/2685531

Verdhan, V. (2021). *Computer vision using deep learning: Neural network architectures with python and keras*. Apress.

Wasserman, L. (2010). *All of statistics: A concise course in statistical inference*. Springer.

Williams, V., Argyriou, V., & Shaw, P. Development of pptnet a neural network for the rapid prototyping of pulsed plasma thrusters. In: University of Viena, Austria. The 36th International Electric Propulsion Conference, 2019, September.