Haaga-Helia
University of Applied Sciences

Accuracy of different Machine Learning techniques at predicting students' study path selection

Adrien Ruegger

Bachelor's Thesis
Degree Programme in Business Information Technology
2021

Haaga-Helia
University of Applied Sciences

Abstract

23 April 2021

Author(s)
Adrien Ruegger

Degree programme
Degree Programme in Business Information Technology (BITe)

Report/thesis title
Accuracy of different Machine learning techniques at predicting students' study path selection

Number of pages and appendix pages
40 + 20

Taking the world by storm, machine learning has revolutionized how data is used and analyzed. Built on pre-existing algorithms, the sheer amount of data collected nowadays kickstarted this technology as a staple, as pattern recognition and other statistical studies require large sample size to yield relevant results.

The overarching goal of the thesis was to introduce Machine Learning in an intuitive and readily comprehensible way, as well as encourage schools to consider machine learning-driven personalized guidance as a future tool to improve educational offering. It teaches concepts such as artificial intelligence, statistical analysis and machine learning algorithms.

The study focuses on illustrating this knowledge through a comparison of different machine learning libraries, namely scikit-learn and TensorFlow, at predicting students' study path selection. The data used for this research comes from BITe students at Haaga-Helia in Finland and HES-SO ARC in Switzerland, as 97 of them responded to a questionnaire.

The questionnaire focused on study level, study path selection, demographic questions, as well as statements about BITe studies judged using a Likert scale.

Built exclusively using Python, four algorithms were tested using the dataset: logistic regression, support-vector machine, k-nearest neighbor and decision tree. On top of that, a deep learning neural network was also used to compete with the other algorithms.

As comparison was the main element of this research, each method was tested 500 times and the best, worst and mean of each technology were extracted. Each try had a randomized separation between the training set (75% of the data) and the testing set (25% of the data), but the proportion of study path was maintained.

The average accuracy was among 36% to 46%, while the best reached 67% to 83% and the worst 6% to 22%, highlighting the importance of cross-validation and preprocessing, especially with low sample sizes. It also called attention to the similarities and differences between the schools' students and mindset, backed by the survey results.

In line with the thesis' objectives, this project could be used to showcase how to improve machine learning results with deeper implementations and iterations. The idea itself could be developed further and implemented by schools to better guide students towards their ideal studies.

Keywords
Machine Learning, Deep Learning, Neural Networks, Study Path, scikit-learn, TensorFlow, Python, AI

Table of contents

Terms and abbreviations

AI	Artificial Intelligence
API	Application Programming Interface
BITe	Business Information Technology
GOFAI	Good Old-Fashioned AI
IDE	Integrated Development Environment
KNN	K-Nearest Neighbor
ML	Machine Learning
NN	Neural Network
PCA	Principal Component Analysis
SVM	Support-Vector Machine

1 Introduction

The new decade brought with itself numerous challenges as the coronavirus pandemic radically changed the way we communicate, work and interact. Education shifted to a remote system for most, allowing students to pursue their education without pauses. However, these unannounced circumstances proved to be hard to deal with for many as having to navigate through a major crisis while adapting to a new environment is no easy task(Sahu, 2020).

While not unique to the outbreak, schools now rely more than ever on their internet presence to encourage future alumni to enroll in their programs, as visits and workshops are less likely to happen. As such, it is critical to understand the link between a person's interests and their studies(Sellami et al., 2017) to be able to advertise interesting and tailored options to everyone and offer guided counselling on course selection.

Furthermore, student attrition is an ongoing issue in higher education all across the world(O'Keefe, 2013) and improving both the transparency of faculties and their features as well as strengthening the bond between institutions and students with better academic advising could improve students' perception and thus retention(Roberts & Styron, 2010).

One field of study that could support this vision is Machine Learning (ML). Taking the world by storm, ML has revolutionized how we use and analyze data. Built on pre-existing algorithms, it is the sheer amount of data collected nowadays that kickstarted this technology as a staple, as pattern recognition and other statistical studies require large sample size to yield relevant results(Marley, 2014). The use of ML to build recommendation tools has proven to be very successful as giant corporations use them, such as personalized feed on social media platforms or video suggestions on streaming platforms. However, different use cases call for different algorithms and methods to provide the best results in terms of efficiency and accuracy.

Consequently, this thesis explores the aforementioned themes through an initiation to ML and related studies via a comparison of different machine learning libraries at predicting the study path selection of students in the Degree Programme in Business Information Technology (BITe). It is the author's deep interest in new technologies and education that drives this research, as well as a deep care for those struggling with their career paths.

2 Research questions and Research methodology

2.1 Objectives of the project and research question

The overarching goal of the thesis is to introduce Machine Learning in an intuitive and readily comprehensible way, as well as encourage schools to consider ML-driven personalized guidance as a future tool to improve educational offering. For the thesis author, it serves as an introduction to the topic as well as personal training for writing academic papers and possibly course material, being interested in teaching later on.

The empirical research will focus on the following questions:

1.) Is there a significant margin in accuracy between the different Machine Learning libraries at predicting students' study path selection?

2.) What is the best accuracy the Machine Learning libraries can reach at predicting students' study path selection?

2.2 Scope of the thesis

This thesis focuses on the explanation of and research around Machine Learning. Ultimately, this thesis could serve as a stepping stone for both educational purposes and future products focused on helping students select their study path.

The thesis' scope does not include any sort of interface, application or website that can be fed with data to use with the algorithms. It will also not include an exhaustive number of algorithms and will focus on the most common and used instead.

2.3 Methodology

As a research thesis, the project follows a structure akin to those of academic papers to facilitate the understanding of the subject for beginners. Indeed, the thesis author's goal was to make the project a good entry point to modern machine learning programming for BITe students and possibly use it for teaching purposes. The thesis is conducted over a period of 4 months, from February to May.

3 Related Research

To better understand the mechanics behind our study, it is important to clarify the context and underlying technologies used in our project. Artificial Intelligence (AI), a broad field of study that encompasses Machine Learning is presented. Then the more specific themes of statistical analysis and datamining are brought up before dissecting the functioning of ML algorithms.

3.1 What is AI?

First coined in 1956, Artificial Intelligence is a field of computer science that focuses on the creation of systems that can work independently and intelligently, meaning that it can make decisions informed by parameters and variables to carry out tasks that supposedly require a human brain. Though the thought of computers or robots being able to surpass humans and take over the world is often used for fearmongering headlines or science-fiction movies, general AI, or AI that are capable of doing many different tasks, is nowhere near this sort of capabilities. The current developed AIs are what is called narrow AIs, they are only good and trained at doing one specific task where they might outperform humans, but they lack the ability to do many different tasks like a person can(Sajja, 2021).

3.1.1 Symbolic Learning

Beyond the distinction between narrow and general artificial intelligence, AI is divided into multiple subsections, starting with one of the two main groups: Symbolic Learning. Symbolic Learning, sometimes also called Good Old-Fashioned AI (GOFAI), uses symbols to represent real-world data and solve problems by using logic. It requires hard-coded rules, human knowledge, to understand how the symbols interact with each other. GOFAI also employs relations, either expressed as adjective to symbols to describe them, or verb to specify the dynamics between symbols(Philip, 2019).

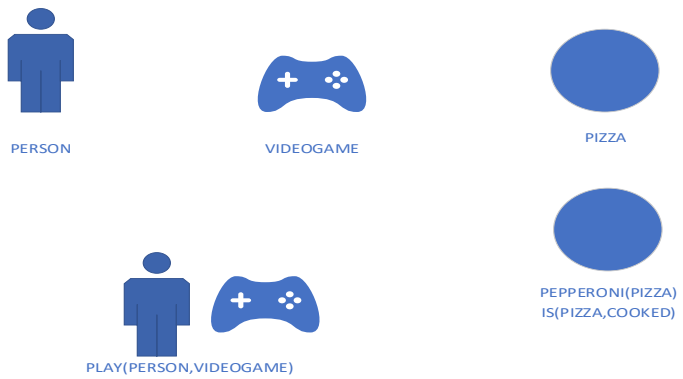


Figure 1. Example of Symbolic AI

The collection of symbols and relations a GOFAI uses like in Figure 1 is called a knowledge base and logic appears either as logical connectives (or, and, if/then, …). The rules are implemented as truth tables, which uses propositional logic to understand the implications of each equation. GOFAI also learns through inference, which compares propositions and to draw conclusions(Mira, 2008).

Symbolic Learning was particularly dominant prior to the 1990s but having hard-coded rules asked both for a lot of time and effort, as it demands people to first understand the relationships of the symbols before adding them to the knowledge base. Besides, GOFAI is particularly prone to preconceptions as the knowledge comes from a person which might have biases and assumptions. However, unlike the other main type of AI, Machine Learning, GOFAI does not require a lot of data, which is another reason why ML only became marketable recently with the rise of Big Data. Another one of its perks is that its logic is comparable to a glass box, whereas ML is more like a black box; because GOFAI uses logic and reason, while ML uses trial-and-error methods(Garnelo & Shanahan, 2019).

3.1.2 Machine Learning

Machine Learning is the other main type of AI, which saw a rise in popularity and usage with the rapid growth of data sets and interconnections that social media brought with themselves. As stated before, ML's accuracy depends on the quantity of information available as well as fast computing power, rendering it almost unusable until recently(Al-Jarrah et al., 2015). Contrary to GOFAI, ML are used to perform specific tasks but without any clear instructions. By feeding algorithms a large amount of data known as training data, ML-based AI tries to find recurring patterns and correlations between inputs, called features, and outputs, called labels(Tu, 2019).

As mentioned earlier, AI is split into many divisions and Machine Learning methods can be categorized into three groups which all use different algorithms: supervised learning, unsupervised learning, and reinforcement learning.

Supervised learning is used when the objectives are known, hence the learning being named task driven. Therefore, human intervention is essential for supervised learning, as it founds its knowledge on preexisting answers or targets.

There are two kinds of supervised ML: classification and regression. Classification is when data needs to be attributed to specific labels. The output is called nominal or discrete, as each value are distinct and separate. Regression on the opposite, is employed to predict unknown elements based on existing observations. In this case, the data is numerical and continuous(Kotsiantis, 2007).

For example, determining if a flower is part of a certain genus based on the size and number of its petals, its color and its fruit would be classification, as we assign a category to the flower. On the other hand, determining the size of a cat according to its race, age and parents' size would be regression, as the answer follows a continuous line established by the training data. Figure 2 represents how each model relates to the data.

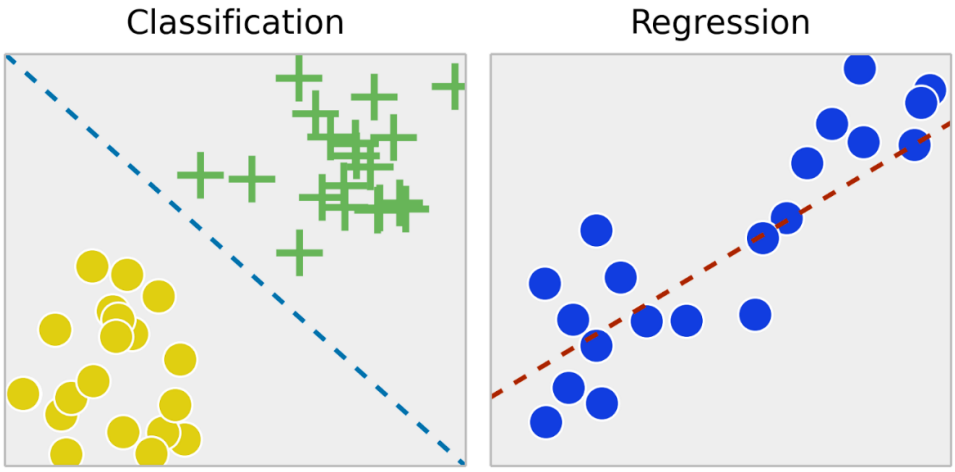


Figure 2. Visualization of Supervised ML

On the other hand, **unsupervised learning** draws on algorithms that do not need defined goals. Instead, they figure out patterns on their own using either clusters or associations. This type of learning is called data driven. This approach can lead to discoveries of hidden patterns that would otherwise be unidentifiable, for example in complex and multidimensional data models. It can also pick up anomalies and faults like a data point seemingly too different from the rest, or a value that seems abnormal(Schlegl et al., 2017).

Clustering classifies the data into groups, similarly to classification ML, but this time, classes are formed by the algorithm itself instead of being explicitly given. Comparatively, associations work by analyzing possible dependencies between items in a data set.

To put it simply, a clustering algorithm might be able to find new customer profiles for a supermarket, spotting recurring data like the products they bought, the time and day they do

their shopping and the average price of their basket. Meanwhile, an association algorithm would detect which products are often bought together, like bread and jam or pasta and sauce. This technology is particularly used in data mining, which will be introduced in the next chapter.

Finally, **reinforcement learning** is a type of ML that focuses on training AI to perform complicated tasks by using a reward system. Basically, an agent is put into an environment and is asked to perform a specific task or goal for which it will be rewarded. Starting with no knowledge, it will first effectuate random actions that may or may not be rewarding. However, through trial and error, it will slowly optimize itself, learning from each previous iteration(Buffet et al., 2020).

Reinforcement learning is especially sought-after in robotics and game development. A notable example that played a role in the resurgence of AI is DeepMind, which created a program called AlphaGo back in 2015 that was taught how to play Go and won against champions, a breakthrough at the time. They later developed other programs able to successfully play more complex games like *StarCraft*, a real-time strategy computer game, at pro level(Byford, 2016; Hoffman, 2019).

3.1.3 Neural Networks and Deep Learning

Deep Learning is a sub-field of ML that is inspired by the structure of the human brain. Using what is called an artificial neural network (NN), this type of AI uses multiple layers of neurons to recognize increasingly more complex features from an input(Cook, 2020).

The neurons of the first layer, called input layer, extract the smallest fragment possible of the incoming data, for example a pixel for an image. Then, it sends the information it has stored to the neurons of the next layer via channels. Each neuron of that layer assigns a value to the incoming channels, based on the relevance of the information for its task. Subsequent layers are called hidden layers, and all contain neurons with progressively more abstract tasks, from identifying contrast between pixels to recognizing eyes in a picture(Neapolitan & Neapolitan, 2018). Finally, after any number of hidden layers, the last layer, called output layer, contains a neuron for all the possible answers, which determine the odds of the final guess as a percentage. Figure 3 illustrates a NN model with 3 hidden layers, 3 input neurons and 2 output neurons.

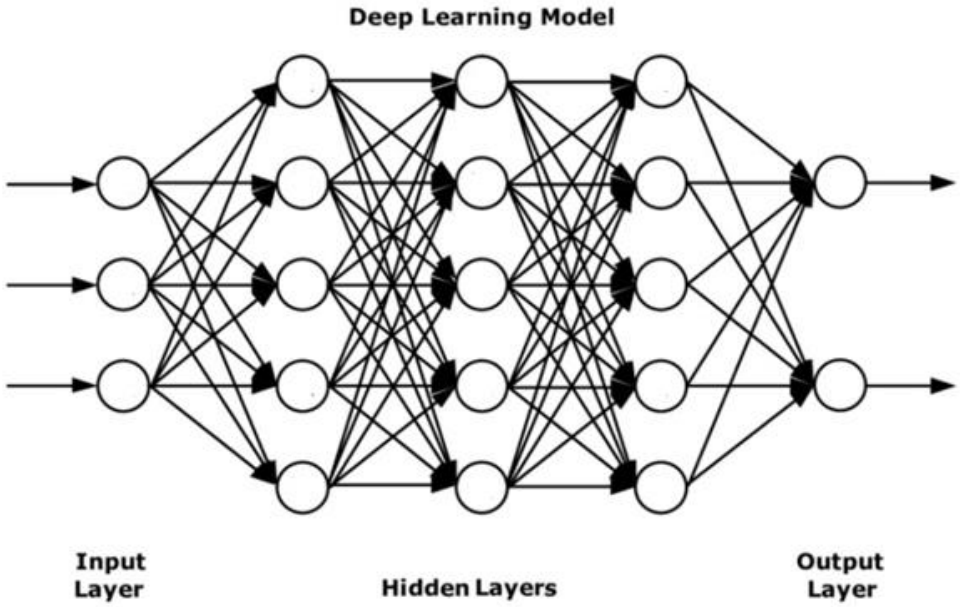


Figure 3. Example of a neural network structure

An important thing to consider with NNs, is that the number of hidden layers increases the complexity of the system, making the reasoning behind the end result more complicated to understand. This can be problematic when deep learning is used in medical or financial fields, as it is essential for patients or customers to understand the motives behind certain decisions, such as why a certain insurance or grant has been refused(Balasubramanian et al., 2018).

3.2 Statistical Analysis and Data mining

Statistics are the foundations which ML is built upon. Therefore, it is decisive to understand key concepts that are recurrent in ML algorithms. This chapter will also touch upon what data mining is and its applications in today's society.

3.2.1 Bias/variance Tradeoff

One of the most important statistical concepts that plagues machine learning is the bias/variance tradeoff. When using statistical analysis and ML, data is split into two separate sets, one for creating prediction models and teach ML algorithms called the training set and one for testing how well it handles new data called the testing set. The bias/variance tradeoff is a dilemma between how faithful the model is to its training data and the fidelity of the model when testing data is introduced to it(Neal et al., 2018).

Because of the separation of the data into subsets or simply the nature of the information, statistical models are incapable of replicating the perfect relationship between responses and variables. This is the **bias**. Note, that bias can also be inherent to the model used, as for example a linear regression will never be able to replicate a curved relationship, as it is a linear model. The amount of bias varies depending on how close to the testing data points the predictions are. **Variance** on the other hand, describes how vulnerable the model is to fluctuations in the training set(Belkin et al., 2019).

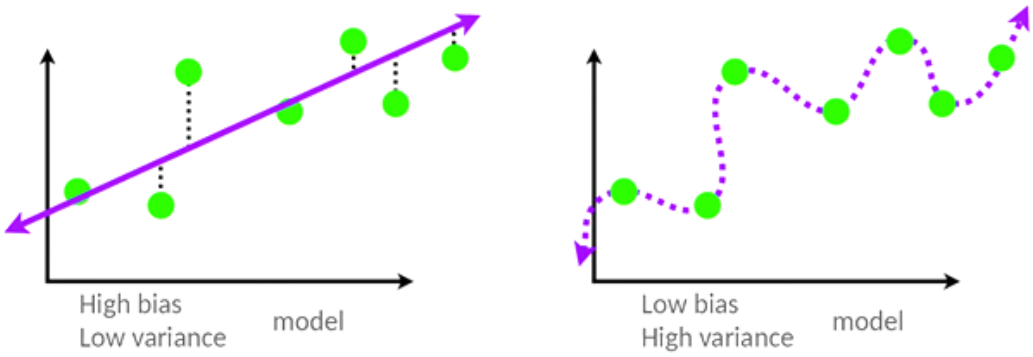


Figure 4. Comparison between two statistical models with training data set (green) to illustrate bias and variance (Starmer, 2020)

As shown on Figure 4, high variance models are technically more accurate than high bias models. However, Figure 5 highlights that too much variance is detrimental to the model as it ends up lowering the reliability of the model past the training phase. This phenomenon is called **overfitting**, because the model has become too dependent of its training set.

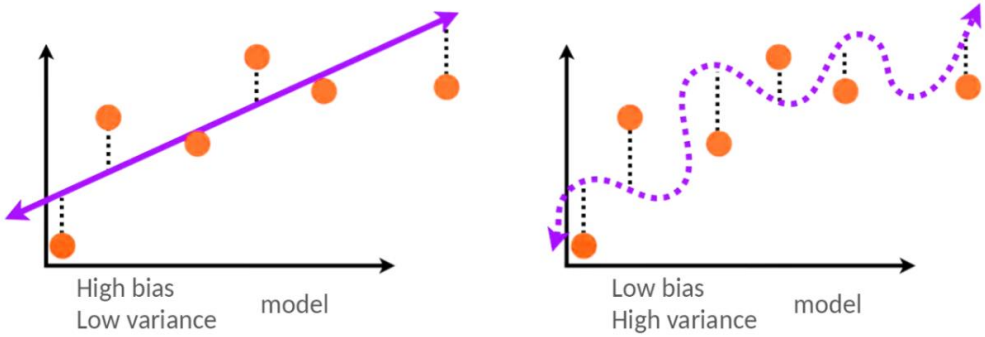


Figure 5. Same model comparison as Figure 4 but this time with testing data set (orange) (Starmer, 2020)

In conclusion, the ideal model aims to find the right compromise between variance and bias, as it needs to accurately represent all relevant relations in the dataset while also being able to produce consistent predictions across datasets(Yang et al., 2020).

3.2.2 Cross-validation

Data has to be separated into two sets to both train and test models, as using the same data for both tasks is a bad idea, since the algorithm might simply regurgitate the information that has been fed to it instead of making estimations(Berrar, 2018). However, simply dividing the data into two sets would be prone to an arbitrary choice, so the best way to ensure that this separation leads to a model with the optimum reliability is to use cross-validation.

The concept of cross-validation is to separate the data into multiple blocks and then test the accuracy of each different combinations to find the most accurate. The most common practice is the ten-fold cross validation, which separates the data into 10 blocks. A special case of cross validation is the “leave one out cross validation” which reduce the size of the testing dataset to an individual sample(Brownlee, 2019; Raschka, 2018).

Cross-validation is also used to find which method should be used for a problem, by comparing how well each algorithm perform. This also means that each algorithm will go through the training and testing data separation process, but while it can be time and resource consuming, cross-validation is very beneficial and guarantees that a model is fulfilling its goal to the best of its ability(Vanwinckelen & Blockeel, 2012).

3.2.3 Data Mining

A process that has seen a stark development recently is data mining. While often used as a blanket term for any form of information processing with huge amount of data or anything remotely close to AI or BI, data mining specifically refers to the discovery of hidden patterns or valuable structures from available data. It is crucial to understand that it does not concern the search of data like the name would suggest, but the search of knowledge within the data(Bramer, 2016). Analysis of the data is only a part of data mining, as other tasks such as data loading, data transformation, data modeling and data visualization are also part of the data mining process.

Compared to other data analysis methods, data mining uses ML to uncover information that cannot be found through traditional means like queries, giving an edge to those who employ this technology(Rahman, 2018).

3.3 ML algorithms

When using ML technologies, it is vital to grasp the importance of a well-chosen algorithms, as the algorithms dictate the perception and understanding we have of the data(Almustafa, 2020). As such, many algorithms have been created and tested in order to achieve the best accuracy for any given problem. This chapter presents the most commonly employed and available algorithms used in supervised and unsupervised learning approaches.

3.3.1 Predictions

Prediction algorithms are all used in supervised learning, whether for classification or regression. They use features and their relations to each other to define the key variable. Some are very similar from each other, as sometimes the only difference between classification and regression is the use of bounds to create categories. Some of the most popular techniques are linear regression, logistic regression, decision tree, support-vector machine (SVM) and k-nearest neighbor (KNN)(Brownlee J., 2019).

Linear regression is one of the most basic prediction models. It lies on the principle of independent and dependent variables, meaning that there is a clear relationship between the response and the features. Using training data as a basis, a linear equation is created to fit the observations as closely as possible by applying the least squares method, where the aim is to minimize the sum of the errors, which are the differences between the estimations and their projections on the line as shown on Figure 6.

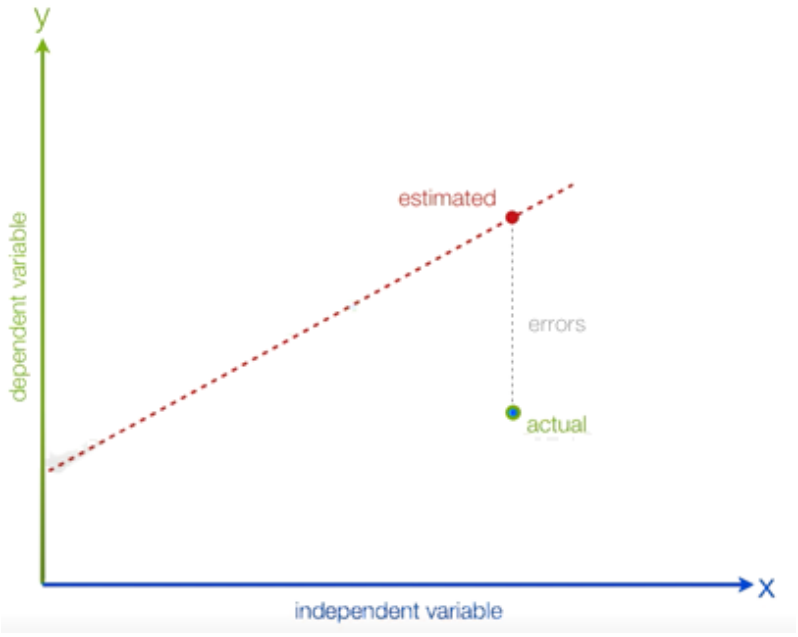


Figure 6. Visualization of linear regression terminology (Starmer, 2020)

Logistic regression is similar to linear regression but is used when the outcome variable is categorical. It is often used when the output is in binary form, such as “qualify” or “does not qualify”; this is called a binomial logistic regression. However, it is still possible to use logistic regression with more than two categories, for example with image classification of clothes; in this case, it is multinomial.

To predict data classes, logistic regression models use a logistic sigmoid function which establishes boundaries due to its asymptotic nature. Logistic regressions are often used when the amount of data is scarce, but also in large data sets as regression is quick and easy to implement(Connelly, 2020).

Decision tree is a classification model that functions using series of binary choices to categorize an input based on their path. A decision tree begins with a root node that points to internal nodes which can split as well, and this until the process is over when a leaf node is reached. Also, final classifications can be repeated, meaning that two different paths can point towards the same answers on different nodes. It is also possible that similar decision nodes appear in the tree with different values for example, as decision trees support both numeric and categorical classifications.

When building a decision tree, choosing the right root node is key(Bulac & Bulac, 2016). To determine which attribute should be the root node, impurity is calculated, which represent how decisive the trait is in the final outcome. The lesser the impurity, the more the variable is determining and the closer it should be to the root node, with the least impure characteristic being the root node itself.

Support-vector machine is a complex classification and regression method for supervised learning which uses thresholds called margins to separate data that belongs in the same group. To support outliers, SVM uses soft margins which allows data to be considered misclassified in order to create a model that has a better overall performance compared to a standard maximal margin classifier that simply split the space between the closest observation of each type at an equal distance. As soft margins and misclassifications influence bias and variance, cross-validation is used in SVM to determine the best soft margin classifier by checking the number of misclassifications and correctly classified data for each soft margin.

SVMs can support hugely varied data both in term of dimension and representation, because soft margin classifiers also called support vector classifiers adapt to the number of dimensions. The number of dimensions of the support vector classifier hyperplane is always 1 less than the dataset's, meaning that the support vector classifier is a point for a one-dimensional dataset, a line for a two-dimensional dataset, a plane for a three-dimensional dataset and so on. Furthermore, SVMs can transpose data to n-dimension using what is called a kernel function. This allows separation of data that would be otherwise impossible to divide using classifiers like presented in Figure 7.

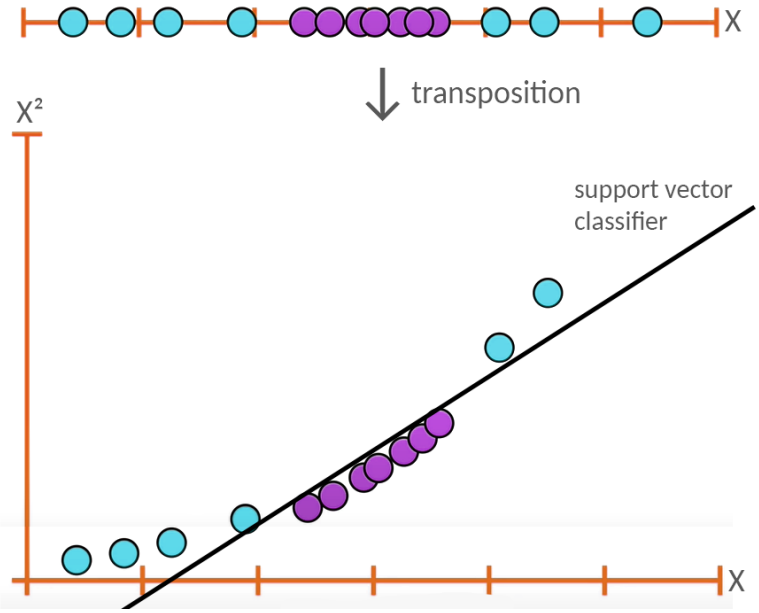


Figure 7.Example of solving classification problems with extra dimensions (Starmer, 2020)

K-Nearest Neighbor is a simple but powerful algorithm often seen used in most classification problems. KNN models assign a category to a new input by comparing it to the categories of the nearest data points on a graph(Larose & Larose, 2014). The number of data points used for the comparison, or nearest neighbors, is determined manually by the K parameter. For example, if K=3, the algorithm will only use the 3 closest points to determine which category the unknown fits the best, assigning it to the category represented the most among the nearest neighbors.

Finding the best value for K is invaluable but complicated, as there is no default setting that is suitable for all circumstances. Furthermore, low K values tend to be subject to outliers and noise, while high K values can completely overlook categories with low population. It is also important to avoid ties, as such results often end up being unclassified, so K values tend to be odd(Seidl, 2016).

3.3.2 Pattern recognition

Pattern recognition belongs to unsupervised learning, as the patterns are not yet known. These algorithms focus on closeness or similarities to find patterns or groups among the data. As this thesis focuses mainly on supervised learning, only one of the most used algorithms is explained, as it has both similitude and relevance with the topic and previous techniques(Mittal et al., 2019).

K-Means Clustering is a classification algorithm that draws similarities with KNN. Using a given K value to determine the number of clusters it has to find, K-means clustering picks a number of random data points equals to K and attribute all the other data points to groups based on the closest random point. Then, it calculates the mean of each group and reclassify data points using proximity again. If any cluster changes, the mean is assessed again, and classification goes on until there are no more changes. After that, the variations of all clusters are summed together and stored to be compared with the results given by other random data points. The goal is to find the random points that give the least amount of final variation to determine the best groupings(Sinaga & Yang, 2020).

As K-means clustering is used for unsupervised learning, it can be hard to find the best K value when using this algorithm, as the exact number of clusters necessary might not be known. A method used to find this variable is the elbow method like Figure 8. This showcases that the preferable K value is the one before an efficiency dip(SAPUTRA et al., 2020).

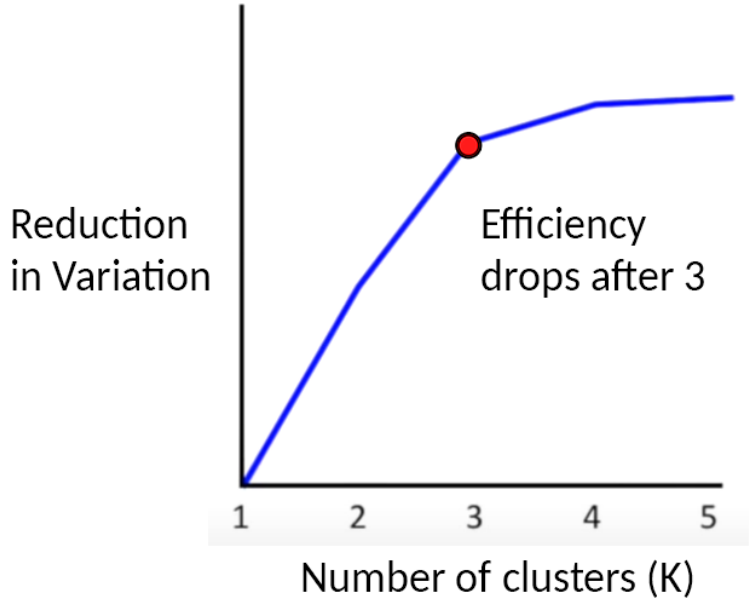


Figure 8. Illustration of an elbow plot used to determine K value

3.4 ML programming language

As ML relies heavily on algorithms, many programming languages can support this technology, making the creation of commercial ML tools possible for a lot of different scenarios. However, there are many factors that weigh in language selection. For example, in infrastructures and projects, using the same language for every application can make for a more robust environment and does not require as many skills, and therefore people, to maintain. Certain domains also favor a certain language, like video games who are often coded in C or C++. Yet, Python is currently considered the n°1 programming language for AI and ML (Raschka et al., 2020), but this could change as new technologies are being released, such as brainJS, a JavaScript library that brings NNs to an already very popular language.

3.4.1 Python

While first released in 1991, Python's popularity has been exponentially growing since it's 3.0 release in 2008. There are many reasons to this and the first one is its simplicity and readability. Indeed, Python is both beginner-friendly and very compact, able to do a lot of operations with as little code as possible, while also taking care of complex tasks as a high-level programming language (Kadriu et al., 2020). This amplifies a virtuous cycle that Python benefits from between users, libraries and fields of application, as its ease of use attracts more users and programmers, which develop open source tools thanks to the language's high extensibility design. As such, there are libraries available for almost all use cases with an enormous community behind them to get help from or interact with. It also benefits from being cross-platform, being able to be built and run on Windows, Mac and Linux operating systems.

Multiple machine learning and deep learning libraries are available on Python, the most noteworthy being TensorFlow, Keras, OpenCV and PyTorch.

3.4.2 Scikit-learn

Originally developed by David Cournapeau, scikit-learn is a library offering various kinds of ML algorithms such as SVMs, KNN and Decision Tree. It also provides multiples tools like dimensionality reduction, model selection and preprocessing to prepare and tune data for algorithms(Buitinck, 2013).

3.4.3 TensorFlow

Developed at Google, TensorFlow is a deep learning platform that provides application programming interfaces (APIs), allowing beginners and experts alike to create machine learning models. Free and open source, it is principally a Python extension, but models can be imported and deployed to other platforms, such as JavaScript and Edge devices (Android, iOS, Raspberry Pi). A first version was released on 2017 before leading to the release of TensorFlow 2.0 in 2019 to compete with the Facebook ML library PyTorch (Sayantini, 2019).

TensorFlow excels in abstraction, able to provide complex ML models through simplified interfaces, allowing the creation of graphs without having to worry about the implementation of algorithms and thus enabling developers to focus on the logic of the application (Tutorials Point, 2019).

One of TensorFlow's most fundamental aspect is the tensor. Tensors are generalizations of vectors and matrices to potentially higher dimensions, containing n-dimensional arrays of base datatypes. As demonstrated in Figure 9, they are defined by a rank, which is the number of dimensions a tensor has, and a shape, which represents the length of each axes of the tensor.

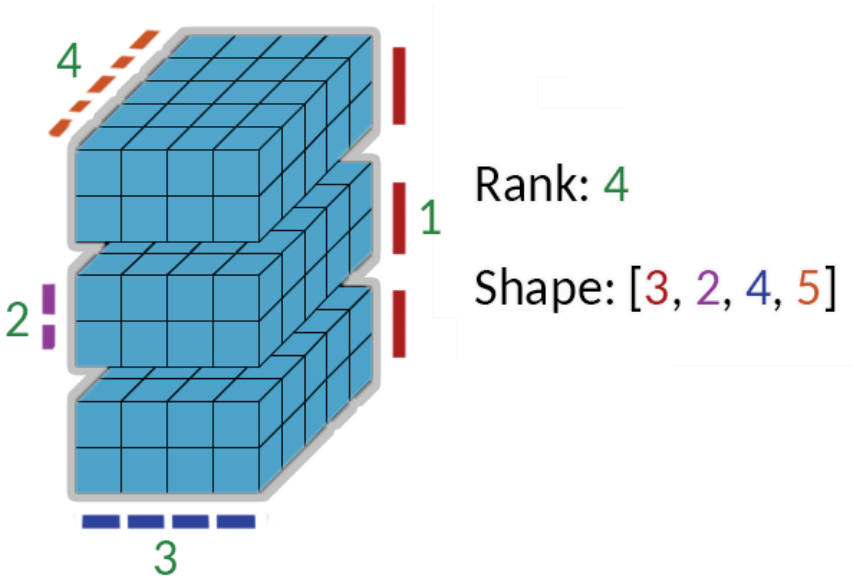


Figure 9. Representation of a rank-4 tensor

One of the most used tools of TensorFlow is **Keras**, a deep learning high-level API that is integrated with TensorFlow 2, offering approachable methods and functions to build robust machine learning applications, focused mostly on deep learning and neural networks.

4 Empirical study

This chapter focuses on the context and the preparation done for the study of the thesis. The project consists of a basic implementation of machine learning algorithms and neural network model to illustrate them and emphasize the importance of good practices.

4.1 Project background

The first inklings of a machine learning based thesis came to be in an email exchange between the thesis author and the thesis supervisor. The researcher initially proposed multiple subjects all dealing with various machine learning implementations such as an application for computer games recommendations or a small game using reinforcement learning, but ended up choosing Dr. Amir Dirin's suggestion, eager to work on a project linked to education and agreeing that the workload would better fit the time restriction of the thesis.

The topic itself is linked to previous research led by the thesis supervisor, focusing on trying out multiple ML libraries and using programming rather than data mining software. The data would also be gathered anew to better reflect the current world situation as the coronavirus pandemic severely affected students and education alike.

Soon after, a kickoff meeting was organized between both parties to discuss the project's structure as well as guidelines.

4.2 Project methodology

As working with Machine Learning requires good data, understanding the research and the metrics used is paramount to ensure that the results yielded by the experiment are both useful and viable. This chapter focuses on explaining what went into the experimentation before implementing the final design.

The first step was to create and conduct a questionnaire for our empirical study. The goal was to reach a sample size of at least a hundred BITe students, though the use of mock data could have been relevant in case that number wasn't met. Then research on related fields and the applicable methods of Machine Learning, as well as suitable algorithms were studied and documented in the thesis, adopting a zooming-in approach to introduce complicated concepts in an approachable way.

Lastly, the actual experiment, which entailed testing the accuracy of each machine learning technique over 500 tries, each time with a shuffled dataset, completed with reports and analysis of the results, such as examples, accuracies, to reflect upon the work that has been done.

4.3 Data gathering

Data gathering was among the first objectives of the project, as the whole process can take a long time to be completed.

4.3.1 Questionnaire

To collect information regarding study path, a questionnaire was made using the Google Forms tool. The design and questions of the survey were based both on previous works on the subject(Niemivirta, 2002) as well as another thesis Dr. Dirin worked on as an advisor(Saballe, 2019). A copy of the questionnaire can be found in Appendix 1.

The specialization paths, which were going to be the label of our AIs, were divided into four main ones: Software Development, Digital Service Design, Business and ICT, and ICT structure.

4.3.2 Data collection

Once the questionnaire was finished, it needed to be shared with as many students as possible, as a good sample size was crucial for the project to come to fruition. At first, only Haaga-Helia students were considered for the study, but then, the thesis advisor and the researcher agreed on collecting data from the HES-SO ARC University as well, the thesis author's school.

In order to officially reach BITe students at Haaga-Helia, a research permission form needed to be filled and submitted to Haaga-Helia's administration. A copy of that form can be found in Appendix 4.

When the data were extracted, there were a total of 25 respondents from the HE ARC and 72 respondents from Haaga-Helia. Appendix 2 and 3 provides an overview of the results of each school. Mock data generators such as Mockaroo were considered but ultimately not used, as enough respondents were found to conduct the experiment and the use of such tools could have altered the final result.

4.3.3 Data preparation

A preliminary step was to transfer the data from Google Forms to Google Sheets thanks to the automated tools provided by the services. This was chosen over exporting the data directly as csv as manipulating data in spreadsheets is much easier. Then, the first change to the data set was to remove the timestamp column that comes with Google Forms datasets, as it wasn't needed for the purpose of the machine learning experiment. The column names were also tweaked as they were primarily labelled as the survey's questions. Some data were also simplified such as the semesters were changed to numbers only. Then, the dataset was transformed into the csv format as it's one of the most common and lightweight formats with machine learning libraries.

4.4 Machine Learning

In order to use the multiple libraries needed for the project, they need to be installed in a Python virtual environment that will serve as the code interpreter. Virtual environments allow users to install libraries for specific uses rather than installing them in the main environment which is shared by all projects. As such, it is used to create tidy workplaces where libraries are only present if they are needed for the projects using the virtual environment. Figure 10 showcases the main steps to set up a virtual environment using Windows PowerShell.

```
# installs the virtual environment tool to the main
python -m pip install virtualenv
# creates a virtual environment named 'project'
python -m virtualenv project
# activates the virtual environment
projet\scripts\activate

# now libraries can be added to the environment by using pip

# installs tensorflow in the virtual environment
pip install tensorflow

# installs scikit-learn in the virtual environment
pip install scikit-learn

```

Figure 10. Steps to configure the Python virtual environment

The integrated development environment (IDE) used for the project is PyCharm(JetBrains, 2017). PyCharm, like most modern IDEs, allow the creation and configuration of projects through user-friendly interfaces. As such, the installation of the virtual environment and the packages could have also been done through PyCharm.

5 Implementation

This section covers the project development and the coding of the different algorithms as well as the neural network. These two methods are separated in two different PyCharm projects. The creation of a project on the IDE isn't documented as it is irrelevant to the experiment which can be done using any Python tool.

5.1 Algorithms

All algorithms are implemented following their basic implementation found in scikit-learn's documentation(Pedregosa, 2011).

5.1.1 Setup

The first thing to do when using libraries is to import them in the main file as shown in Figure 11. The line at the top is not an import but a future statement. Future statement changes how the interpreter functions and as such has to be the first line of the file. It allows the use of functionalities that are not yet implemented in the current Python module but will be in future version, allowing developers to get used to new features. In this project, it is used for multiple reasons such as divisions that return decimal numbers.

Beyond scikit-learn libraries, we also import numpy and pandas, which are two widespread data science libraries. Pandas(The pandas development team, 2020) is built on top of numpy and offers dataframes, which can be seen as similar to spreadsheets. The keyword *as* followed by a namespace allows the use of a shorter alias to avoid typing out the full name each time the module is utilized.

```
from __future__ import absolute_import, division, print_function, unicode_literals

from sklearn.linear_model import LogisticRegressionCV
from sklearn.svm import SVC
from sklearn import neighbors, datasets
from sklearn.model_selection import train_test_split
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.tree import DecisionTreeClassifier
import numpy as np
import pandas as pd
```

Figure 11. Future and imports for the algorithm project

5.1.2 Data Manipulation

In order to use the data which are stored in csv files in the project folder, they need to be loaded in the code. To do so, the method *pandas.read_csv()* is used, which transforms a csv file into a dataframe, as seen on Figure 12.

```
# Loading the two csv Data files
data = pd.read_csv("HH BITe Study Path Survey.csv")
data.update(pd.read_csv("HES-SO BITe Study Path Survey.csv"))

# Converting qualitative data into boolean columns
data = pd.get_dummies(data, columns=['Age', 'Gender', 'Origin'])
```

Figure 12. Loading and preparing the data for the algorithm project

Still on Figure 12, the next step is to transform most of our demographic columns, which are qualitative rather than quantitative, into data that can be interpreted by the algorithm. As simply turning each category variable into a number is not quite enough, since there is no gradual ranking between them, they are instead transformed into indicators. Indicators are Boolean variables created for each value of a categorical variable. For example, a column tracking the age category with either child, adult or senior would be transformed into three true/false indicators: age_child, age_adult and age_senior. This process is done to the columns age, gender and origins.

Then, the label and features were separated into two variables, named X and y for convenience purposes. The *pop()* function removes a column from a dataset and returns it. Only the values are extracted as scikit-learn mostly uses numpy arrays over pandas dataframes. Afterwards, the label is transformed into numeric values for standardization. This process is documented on Figure 13.

```
# Separating label from features
y = data.pop('Specialization').values
X = data.values

# Encoding label as numeric categories
label_encoder = LabelEncoder()
y = label_encoder.fit_transform(y)
X = StandardScaler().fit_transform(X)
```

Figure 13. Separating label and features for the algorithm project

5.1.3 Algorithm loop and pipelines

To illustrate that machine learning tools are not simply some sort of magical technology that can make sense of data without any effort, the algorithms are tested 500 times, using the same data but split differently every time. The number of times tested has been chosen as a compromise between speed and the ability to test as many data arrangement as possible. First, to be able to collect the data in this loop, an array has to be created for each algorithm, as shown on Figure 14.

```
# Declaring result arrays
LGRs = []
SVMs = []
KNNs = []
DTCs = []

# Looping for a 500 tries
for i in range(1, 501):
```

Figure 14. Result arrays declaration to store each loop's results for the algorithm project

Inside the loop, the data are shuffled by using the *train_test_split()* method, which separates both the label and the features into two sets, a training set and a testing set. The parameter *test_size* dictates the size of the test set, here equals to a fourth of the data. The *stratify* parameter, here switched on, keeps the proportion of label occurrences between both sets, to avoid ending up with certain data not showing up during training, thus being problematic during the testing phase, as the model would be unfamiliar with this output. Figure 15 sums up this process.

```
# Separating training data and testing data, Stratify keeps the proportion of data in both sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=1 / 4, stratify=y)
```

Figure 15. Splitting data into train and test samples using scikit-learn method for the algorithm project

As visible on Figure 16, the first algorithm tested is the logistic regression, which uses cross validation to speed the process by using results used in previous steps. The number of steps is determined by the *max_iter* parameter, but the optimization can stop earlier once it determines that it has reached its goal, and the number of folds used in the cross-validation process is determined by the *cv* parameter, here set to 3 as the default 5 is bigger than the

number of study paths we have. The *random_state* parameter shuffles the data but has no incidence on this particular instance.

```
# Pipeline for Logistic Regression with Cross Validation
clfLGR = make_pipeline(StandardScaler(), LogisticRegressionCV(random_state=i, max_iter=5000, cv=3))
clfLGR.fit(X_train, y_train.ravel())
LGRs.append(clfLGR.score(X_test, y_test))
```

Figure 16. Logistic regression pipeline

Pipelines allow multi steps processes to be done when the model is being trained. In these occurrences, it is used to standardize the data before using it with the algorithms as it is required for most machine learning estimators. The SVM model used in Figure 17 uses two parameters, *C* and *gamma*, influences the regularization and the Kernel coefficient respectively. 1 is the default value for *C* and 'auto' for *gamma* defines the coefficient as 1 divided by the number of features. Similar to the previous algorithms, the model is then trained by using the *fit()* function. The *ravel()* function is used on y_train to reverse its dimension. In a visual representation, it would become a column instead of a row. Then, the model is tested using *score()* which returns the accuracy of the test as a percentage and is stored in the array corresponding to the algorithm.

```
# Pipeline for Support-Vector Machine
clfSVM = make_pipeline(StandardScaler(), SVC(C=1, gamma='auto'))
clfSVM.fit(X_train, y_train.ravel())
SVMs.append(clfSVM.score(X_test, y_test))
```

Figure 17. SVM pipeline

As KNN models can greatly vary depending on the value chosen for K, this pipeline has been put into another loop to test the algorithm with K values from 1 to 20, represented by the j parameter in Figure 18.

```
# Loop to use K value from 1 to 20
for j in range(1, 21):
    # Pipeline for K-Nearest Neighbor
    clfKNN = make_pipeline(StandardScaler(), neighbors.KNeighborsClassifier(j))
    clfKNN.fit(X_train, y_train.ravel())
    KNNs.append(clfKNN.score(X_test, y_test))
```

Figure 18. KNN pipeline looped for K=1 to K=20

The decision tree model shown in Figure 19 is the simplest of all the algorithms, with no additional parameters, which shows how bare and simple a model can be when created using scikit-learn.

```
# Pipeline for Decision Tree
clfDTC = make_pipeline(StandardScaler(), DecisionTreeClassifier())
clfDTC.fit(X_train, y_train.ravel())
DTCs.append(clfDTC.score(X_test, y_test))
```

Figure 19. Decision tree pipeline

Finally, in Figure 20, once out of the loop, the results of each algorithm are printed in the console. Using the data of each array, the best, worst and mean are extracted using native *min()* and *max()* functions and numpy's *mean()* function.

```
# Printing the mean, best and worst accuracy results for each algorithm across all the tries
print('---- Logistic Regression accuracy \n Best: ' + "{:.0%}".format(max(LGRs))
      + '\n Worst: ' + "{:.0%}".format(min(LGRs))
      + '\n Mean: ' + "{:.0%}".format(np.mean(LGRs)))

print('---- Support-Vector Machine accuracy \n Best: ' + "{:.0%}".format(max(SVMs))
      + '\n Worst: ' + "{:.0%}".format(min(SVMs))
      + '\n Mean: ' + "{:.0%}".format(np.mean(SVMs)))

print('---- K-Nearest Neighbor \n Best: ' + "{:.0%}".format(max(KNNs))
      + '\n Worst: ' + "{:.0%}".format(min(KNNs))
      + '\n Mean: ' + "{:.0%}".format(np.mean(KNNs)))

print('---- Decision Tree accuracy \n Best: ' + "{:.0%}".format(max(DTCs))
      + '\n Worst: ' + "{:.0%}".format(min(DTCs))
      + '\n Mean: ' + "{:.0%}".format(np.mean(DTCs)))
```

Figure 20. Showing the results at the end of the algorithm experimentation

5.2 Neural Network

The deep learning project is built based on the Premade Estimators tutorial on TensorFlow's website (TensorFlow, 2021).

5.2.1 Setup

Similarly, to the algorithm project, future is used in the neural network project to use the most recent python functionalities, as shown on Figure 21. On top of TensorFlow, numpy and pandas, three modules from scikit-learn are reused, namely train_test_split, LabelEncoder and StandardScaler, as these functionalities are useful even with TensorFlow, especially for splitting the data and the preprocessing step.

```
from __future__ import absolute_import, division, print_function, unicode_literals

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
import tensorflow as tf
import numpy as np
import pandas as pd
```

Figure 21. Future and imports for the deep learning project

The second step when setting up the deep learning project is to create the input function for the neural network. This is defined early to avoid having to create the function later on, as the function itself is a parameter of the classifier used further in the code. As seen in Figure 22, the function creates a dataset from the tensors passed that serves as the input when training and testing. Notably, when training, the dataset is shuffled, and the function is repeated for a number of steps defined later on. The function returns a dataset with a number of entries equals to the *batch_size* parameter.

```
# defining a function for the neural network input
def input_fn(features, labels, training=True, batch_size=256):
    # Convert the inputs to a Dataset.
    dataset = tf.data.Dataset.from_tensor_slices((dict(features), labels))

    # Shuffle the data and repeat if using training mode.
    if training:
        dataset = dataset.shuffle(1000).repeat()

    return dataset.batch(batch_size)
```

Figure 22. Input function needed when using a TensorFlow neural network, defined earlier to avoid repeated inner functions

5.2.2 Data Manipulation

When it comes to data manipulation, most steps are similar to the scikit-learn project's steps. First the data are extracted from the csv files into pandas dataframes. Then, Boolean columns are made from the categorical columns, and label and features are separated into X and y.

The main difference, highlighted in Figure 23, is that the feature columns' names are saved. This is done in order to transform the arrays returned by the *train_test_split()* function used later on back into pandas dataframes, as TensorFlow works with dataframes rather than arrays. Then the data are encoded and standardized, and the results array is created to gather the data of the loop.

```
# Loading the two csv Data files
data = pd.read_csv("HH BITe Study Path Survey.csv")
data.update(pd.read_csv("HES-SO BITe Study Path Survey.csv"))

# Converting qualitative data into boolean columns
data = pd.get_dummies(data, columns=['Age', 'Gender', 'Origin'])

# Separating label from features
y = data.pop('Specialization').values
X = data.values

# Saving the column names for conversion later
X_columns = data.keys()

# Encoding label as numeric categories
label_encoder = LabelEncoder()
y = label_encoder.fit_transform(y)
X = StandardScaler().fit_transform(X)

results = []
```

Figure 23. Data preparation for the deep learning project, highlighted is the main difference with the algorithm project

Figure 24 showcases the preparation done inside the loop, where the data is split into training and testing sets, just like in the algorithm code. This time however, the arrays *X_train, X_test, y_train* and *y_test* are transformed back into dataframes by giving them the previously saved column names. An additional step necessary for TensorFlow's neural network to work properly is to remove any spaces in the columns' names. This is done by replacing blank spaces with a common substitute, underscores.

```
# Looping for a 500 tries
for i in range(1, 501):
    # Separating training data and testing data, Stratify keeps the proportion of data in both sets
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=1 / 4, stratify=y)

    # Transform the data into pandas dataframes for tensorflow compatibility
    X_train = pd.DataFrame(X_train, columns=X_columns)
    X_test = pd.DataFrame(X_test, columns=X_columns)
    y_train = pd.DataFrame(y_train, columns=['Specialization'])
    y_test = pd.DataFrame(y_test, columns=['Specialization'])

    # Remove blank spaces in column names to avoid errors later on
    X_train.columns = X_train.columns.str.replace(' ', '_')
    X_test.columns = X_test.columns.str.replace(' ', '_')
    y_train.columns = y_train.columns.str.replace(' ', '_')
    y_test.columns = y_test.columns.str.replace(' ', '_')
```

Figure 24. Data transformation to fit TensorFlow's use of dataframe

The final step of data manipulation is shown in Figure 25. It consists of creating an array with all the feature columns using a loop parsing through all the keys of the training dataset. To do this, as all columns now use numbers instead of text, the method *numeric_column()* is used, using the variable *key* of the loop as an attribute to find the data that it needs to add to the array.

```
# Feature columns describe how to use the input.
my_feature_columns = []
for key in X_train.keys():
    my_feature_columns.append(tf.feature_column.numeric_column(key=key))
```

Figure 25. Creating an array with the feature columns to use with the neural network

5.2.3 Neural network

Now, the NN has to be built. To do this, the TensorFlow premade estimator is used, which is a high-level complete model, perfect to start working with NNs. In Figure 26, 3 parameters are shown to be used when creating the classifier. First, the feature columns are passed to the classifier to give it an understanding of the coming data once training starts. Then, the *hidden_units* parameter, which dictates the middle layers of the NN by using an array representative of each node per layer. In this case, only two hidden layers are created, with 30 and 10 nodes, an arbitrary choice. It is important to note that more layers do not equate higher accuracy, and that it also muddles the impact of the inputs on the output. Finally, *n_classes* simply defines the number of categories the classifier will use. Since there are 4 different specialization paths in our survey, *n_classes* equals 4.

```
# Build a Neural network using 2 hidden layers.
classifier = tf.estimator.DNNClassifier(
    feature_columns=my_feature_columns,
    # Two hidden layers of 30 and 10 nodes respectively.
    hidden_units=[30, 10],
    # The model must choose between 4 classes/output.
    n_classes=4)
```

Figure 26. Creating the neural network for the deep learning project

Figure 27 shows how the classifier use the function that has been previously created to train the model. The keyword *lambda* allows to use the function *input_fn()* as a parameter without having to create the whole function inside the *classifier.train()* function which would be messy. The *steps* argument defines how many times the model trains itself using data from the input. This can be modified, but more is not always better, as the model can end up overfitted.

```
# Training the model
classifier.train(
    input_fn=lambda: input_fn(X_train, y_train, training=True),
    steps=4000)
# We include a lambda to avoid creating an inner function
```

Figure 27. Classifier training for the deep learning project

To test the model's accuracy, Figure 28 shows that the function used is *classifier.evaluate()*, which works similarly to the *classifier.train()* function, using the *input_fn()* function, this time without the training setting activated. The function returns a list of metrics that is stored in *eval_result*, then the accuracy is fetched from the list and added to the *results* array.

```
# Testing the model's accuracy
eval_result = classifier.evaluate(
    input_fn=lambda: input_fn(X_test, y_test, training=False))
results.append(eval_result.get('accuracy'))
```

Figure 28. Classifier testing for the deep learning project

Finally, once out of the loop, the results are printed using the same formatting as the algorithm project, showing the worst, the best and the mean accuracy of all tries, per Figure 29.

```
# Printing the mean, best and worst accuracy results for each algorithm across all the tries
print('---- Tensorflow Neural Network \n Best: ' + "{:.0%}".format(max(results))
      + '\n Worst: ' + "{:.0%}".format(min(results))
      + '\n Mean: ' + "{:.0%}".format(np.mean(results)))
```

Figure 29. Printing the result at the end of the deep learning project

6 Result

This chapter briefly goes over the outcome of the experiment, as details are brought up in the discussions chapter.

6.1 Algorithms

The results for the different algorithms shown on Figure 30 only took a few minutes of computing despite the different methods being looped multiple times each.

```
---- Logistic Regression accuracy
 Best: 72%
 Worst: 11%
 Mean: 43%
---- Support-Vector Machine accuracy
 Best: 72%
 Worst: 22%
 Mean: 46%
---- K-Nearest Neighbor
 Best: 83%
 Worst: 6%
 Mean: 38%
---- Decision Tree accuracy
 Best: 67%
 Worst: 11%
 Mean: 36%
```

Figure 30. Accuracy results of all the tested algorithms

6.2 Neural Network

In comparison, the deep learning project took multiple hours to output the results shown on Figure 31, as creating a neural network 500 times is much more time consuming than simpler algorithms.

```
---- Tensorflow Neural Network
 Best: 67%
 Worst: 11%
 Mean: 39%
```

Figure 31. Accuracy results of the neural network

7 Discussions

The carried-out research generated five ML models distributed over two libraries, scikit-learn and TensorFlow, which represented algorithmic classification and deep learning classification respectively. Focusing on shedding light on the accuracy of these methods, the experiment brings up interesting questions when compared with other research and leaves room for experimentation, improvement and finetuning. But despite its simplicity, the study holds an undeniable value thanks to its teaching purpose as well as originality and specificity. After all, the thesis is an introduction to machine learning rather than a guide to optimal ML configurations, and must retain its accessibility to people without prior knowledge of the subject.

7.1 Answer to research questions

1.) Is there a significant margin in accuracy between the different Machine Learning libraries at predicting students' study path selection?

Looking at the three values examined for each technique, a few observations can be made. The best results gravitate between 67% (TensorFlow NN & Decision Tree algorithm), 72% (SVM and Logistic Regression algorithms) and 83% (KNN algorithm). This shows a difference of 16% in best performance between highest and lowest ranking methods, which is quite significant, as it represents a 1 out of 6 guesses differences. In comparison, other research(Berhane, 2021) typically show a range of accuracies between 66% and 75% when using the same tools and data of similar complexity(Allibhai, 2018) which is encouraging. As for the worst, most techniques reach an accuracy of 11% with two outliers, KNN at 6% and SVM at 22%. The polarizing results of the KNN model can be explained by the loop for K values used in the project. Indeed, very high and very low K values will tend to miss more often, either because of outliers when K is too low, or because of barely populated labels when K is too high(Zhang et al., 2017).

When it comes to mean accuracy, the models show values between 36% and 46%. While above the accuracy of blind guesses at 25%, these low scores show the importance of data quantity, data selection and preprocessing. For example, a KNN model can see an improvement of 10% in accuracy by using scikit-learn preprocessing methods(Allibhai, 2018). Indeed, because of the small sample size given by the number of questionnaire respondents, each training phase used only about 72 samples to create a classifier, which makes them very sensible to small changes, thus the greatly varying accuracies(Moghaddam et al., 2020).

2.) What is the best accuracy the Machine Learning libraries can reach at predicting students' study path selection?

Looking strictly at the accuracy values of our tests, the best performance of the KNN model reached 83% of correct guesses, which is very encouraging for a basic implementation of a ML algorithm. However, with more time, effort and data, and a better understanding of the more complex functionalities offered by each library, there would be no doubt that the accuracy could get even higher. For example, if the data contained only Haaga-Helia respondents, perhaps the accuracy could be higher, as the education and psychology of students from different schools might be too different to be encompassed into a single model. As a comparison, research done using KNIME software to classify students' study paths between Digital Service Design and Software Development reached up to 94% accuracy when testing only Haaga-Helia students(Saballe, 2019). Overall, there are so many parameters that come into the creation of ML models that there is always a possibility to improve and finetune them(Gambella et al., 2021).

7.2 Reliability and validity

While the mean and lowest accuracy metrics found for both experiments can feel like they jeopardize the experiment, it is important to remember that each accuracy gathered during the study represents a fully operational model, meaning that an actual implementation would only use the most accurate model to fulfil its purpose. Indeed, the other measures are simply informational and serve the study, proving that data selection is a key process of ML, as it was the main variable between each try.

Either way, the thesis project itself could also be built upon to try and reach higher accuracies with more data and use of other functionalities. For example, using Keras functionalities or other ML libraries such as PyTorch could potentially yield better results(Sayantini, 2019).

8 Conclusion

Focusing principally on the study path selection of BITe students at Haaga-Helia and at HES-SO Arc, this thesis's objective was to introduce AI, ML and deep learning in an accessible way through explanations and applications, as well as demonstrating the capabilities of Python machine learning libraries, both contributing to the betterment of education. Furthermore, answers to the research questions were found through the experiment which supported the knowledge previously explained with empirical evidence.

Considering the unpredictability of the school environment due to the coronavirus pandemic, it was hard to tell what could threaten the project and its schedule. A lack of feedback from the questionnaire could have impacted the thesis negatively but would have still been salvageable by using mock data. It was also important that the thesis author worked diligently, as there wasn't much time until the end of the semester. Thankfully, the researcher had the occasion to focus and work diligently on the thesis, free from other obligations. Overall, this research was positively safe, and thanks to the thesis coordinator, no predicament barred the thesis author from achieving the goals set and learning valuable information throughout the project.

8.1 Future works

As this is a research thesis, there isn't any stakeholders, but the results of the project might interest schools in the future, especially Haaga-Helia and HES-SO ARC, as the survey was completed by students from these institutions, thus making the results more meaningful and tailored to them. An introductory course to AI, ML and NNs could also be created using the knowledge presented in the earlier chapters as this prominent technology is not yet part of most school curricula. An interesting project could be to create a frontend application which would allow students to answer the questionnaire on a website and get the result right away by using the trained model in the backend. Such a project could be used during Orientation Day at a school to give new students a suggestion on which courses are available to them based on their results.

Finally, eager to share his enthusiasm, the thesis author wants to encourage the ethical exploitation of ML technology in projects of all scopes, believing that this technology is at its core revolutionary and could benefit everyone if put in the right hands.

Table of Figures

References

Al-Jarrah, O. Y., Yoo, P. D., Muhaidat, S., Karagiannidis, G. K., & Taha, K. (2015). Efficient Machine Learning for Big Data: A Review. In *Big Data Research*. https://doi.org/10.1016/j.bdr.2015.04.001

Almustafa, K. M. (2020). Classification of epileptic seizure dataset using different machine learning algorithms. *Informatics in Medicine Unlocked*. https://doi.org/10.1016/j.imu.2020.100444

Balasubramanian, R., McElhaney, D., & Libarikian, A. (2018). Insurance 2030 – The impact of AI on the future of insurance. *Digital McKinsey & Company*.

Belkin, M., Hsu, D., Ma, S., & Mandal, S. (2019). Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proceedings of the National Academy of Sciences of the United States of America*. https://doi.org/10.1073/pnas.1903070116

Berrar, D. (2018). Cross-validation. In *Encyclopedia of Bioinformatics and Computational Biology: ABC of Bioinformatics*. https://doi.org/10.1016/B978-0-12-809633-8.20349-X

Bramer, M. (2016). *Introduction to Data Mining*. https://doi.org/10.1007/978-1-4471-7307-6_1

Brownlee, J. (2019). *A Gentle Introduction to k-fold Cross-Validation*. Machinelearningmastery.Com.

Brownlee J. (2019). Supervised and Unsupervised Machine Learning Algorithms. In *Machine Learning Mastery Pty. Ltd.*

Buffet, O., Pietquin, O., & Weng, P. (2020). Reinforcement Learning. In *arXiv*. https://doi.org/10.4249/scholarpedia.1448

Bulac, C., & Bulac, A. (2016). Decision Trees. In *Advanced Solutions in Power Systems: HVDC, FACTS, and AI Techniques*. https://doi.org/10.1002/9781119175391.ch18

Byford, S. (2016). *AlphaGo, a project of Google AI subsidiary DeepMind*. Quartz.

Connelly, L. (2020). Logistic regression. *MEDSURG Nursing*. https://doi.org/10.4324/9781351033909-32

Cook, T. R. (2020). Neural Networks. In *Advanced Studies in Theoretical and Applied Econometrics*. https://doi.org/10.1007/978-3-030-31150-6_6

Gambella, C., Ghaddar, B., & Naoum-Sawaya, J. (2021). Optimization problems for machine learning: A survey. In *European Journal of Operational Research*. https://doi.org/10.1016/j.ejor.2020.08.045

Garnelo, M., & Shanahan, M. (2019). Reconciling deep learning with symbolic artificial intelligence: representing objects and relations. In *Current Opinion in Behavioral Sciences*. https://doi.org/10.1016/j.cobeha.2018.12.010

Hoffman, T. (2019). DeepMind's new AI masters the online game StarCraft II. *Nature*. https://doi.org/10.1038/d41586-019-03343-4

Kadriu, A., Abazi-Bexheti, L., Abazi-Alili, H., & Ramadani, V. (2020). Investigating trends in learning programming using YouTube tutorials. *International Journal of Learning and Change*. https://doi.org/10.1504/IJLC.2020.106721

Kotsiantis, S. B. (2007). Supervised machine learning: A review of classification techniques. In

Informatica (Ljubljana). https://doi.org/10.31449/inf.v31i3.148

Larose, D. T., & Larose, C. D. (2014). k -Nearest Neighbor Algorithm . In *Discovering Knowledge in Data*. https://doi.org/10.1002/9781118874059.ch7

Marley, S. (2014). *The Importance and Effect of Sample Size*. Select Statistical Consultants.

Mira, J. M. (2008). Symbols versus connections: 50 years of artificial intelligence. *Neurocomputing*. https://doi.org/10.1016/j.neucom.2007.06.009

Mittal, K., Aggarwal, G., & Mahajan, P. (2019). Performance study of K-nearest neighbor classifier and K-means clustering for predicting the diagnostic accuracy. *International Journal of Information Technology (Singapore)*. https://doi.org/10.1007/s41870-018-0233-x

Moghaddam, D. D., Rahmati, O., Panahi, M., Tiefenbacher, J., Darabi, H., Haghizadeh, A., Haghighi, A. T., Nalivan, O. A., & Tien Bui, D. (2020). The effect of sample size on different machine learning models for groundwater potential mapping in mountain bedrock aquifers. *Catena*. https://doi.org/10.1016/j.catena.2019.104421

Neal, B., Mittal, S., Baratin, A., Tantia, V., Scicluna, M., Lacoste-Julien, S., & Mitliagkas, I. (2018). A modern take on the bias-variance tradeoff in neural networks. In *arXiv*.

Neapolitan, R. E., & Neapolitan, R. E. (2018). Neural Networks and Deep Learning. In *Artificial Intelligence*. https://doi.org/10.1201/b22400-15

O'Keefe, P. (2013). a Sense of Belonging: Improving Student Retention. *College Student Journal*.

Philip, B. (2019). How artificial intelligence works First wave : Symbolic artificial intelligence. *STOA | Panel for the Future of Science and Technology*.

Rahman, N. (2018). Data Mining Techniques and Applications. *International Journal of Strategic Information Technology and Applications*. https://doi.org/10.4018/ijsita.2018010104

Raschka, S. (2018). Model evaluation, model selection, and algorithm selection in machine learning. In *arXiv*.

Raschka, S., Patterson, J., & Nolet, C. (2020). Machine learning in python: Main developments and technology trends in data science, machine learning, and artificial intelligence. In *Information (Switzerland)*. https://doi.org/10.3390/info11040193

Roberts, J., & Styron, R. (2010). Student satisfaction and persistence: factors vital to student retention. *Research in Higher Education Journal*.

Sahu, P. (2020). Closure of Universities Due to Coronavirus Disease 2019 (COVID-19): Impact on Education and Mental Health of Students and Academic Staff. *Cureus*. https://doi.org/10.7759/cureus.7541

Sajja, P. S. (2021). Introduction to Artificial Intelligence. In *Studies in Computational Intelligence*. https://doi.org/10.1007/978-981-15-9589-9_1

SAPUTRA, D. M., SAPUTRA, D., & OSWARI, L. D. (2020). *Effect of Distance Metrics in Determining K-Value in K-Means Clustering Using Elbow and Silhouette Method*. https://doi.org/10.2991/aisr.k.200424.051

Sayantini. (2019). *Keras vs TensorFlow vs PyTorch | Deep Learning Frameworks | Edureka*. Edureka.

Schlegl, T., Seeböck, P., Waldstein, S. M., Schmidt-Erfurth, U., & Langs, G. (2017). Unsupervised

anomaly detection with generative adversarial networks to guide marker discovery. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. https://doi.org/10.1007/978-3-319-59050-9_12

Seidl, T. (2016). Nearest Neighbor Classification. In *Encyclopedia of Database Systems*. https://doi.org/10.1007/978-1-4899-7993-3_561-2

Sellami, A., El-Kassem, R. C., Al-Qassass, H. B., & Al-Rakeb, N. A. (2017). A path analysis of student interest in STEM, with specific reference to Qatari students. *Eurasia Journal of Mathematics, Science and Technology Education*. https://doi.org/10.12973/eurasia.2017.00999a

Sinaga, K. P., & Yang, M. S. (2020). Unsupervised K-means clustering algorithm. *IEEE Access*. https://doi.org/10.1109/ACCESS.2020.2988796

Tu, Y. (2019). Machine learning. In *EEG Signal Processing and Feature Extraction*. https://doi.org/10.1007/978-981-13-9113-2_15

Tutorials Point. (2019). TensorFlow Tutorial. In *Tutorials Point (I) Pvt. Ltd.*

Vanwinckelen, G., & Blockeel, H. (2012). On estimating model accuracy with repeated cross-validation. *21st Belgian-Dutch Conference on Machine Learning*.

Yang, Z., Yu, Y., You, C., Steinhardt, J., & Ma, Y. (2020). Rethinking bias-variance trade-off for generalization of neural networks. In *arXiv*.

Zhang, S., Li, X., Zong, M., Zhu, X., & Cheng, D. (2017). Learning k for kNN Classification. *ACM Transactions on Intelligent Systems and Technology*. https://doi.org/10.1145/2990508

Starmer, J. (2020). *Statquest with Josh Starmer.* https://www.youtube.com/c/joshstarmer/videos

TensorFlow (2021). *TensorFlow Core, Tutorials, Premade Estimators.* https://www.tensorflow.org/tutorials/estimator/premade

Buitinck et al. (2013). *API design for machine learning software: experiences from the scikit-learn project.* https://scikit-learn.org/0.18/_downloads/scikit-learn-docs.pdf

Niemivirta, M. (2002). *Motivation and Performance in Context: the Influence of Goal Orientations and Instructional Setting on Situational Appraisals and Task Performance. Psychologia - An International Journal Of Psychology In The Orient.* https://doi.org/10.2117/psysoc.2002.250

Allibhai, E. (2018). *Building a k-NN Model with Scikit-Learn. Available at:* https://towardsdatascience.com/building-a-k-nearest-neighbors-k-nn-model-with-scikit-learn-51209555453a

Berhane, F. (2021). *TensorFlow Tutorial. Available at:* https://datascience-enthusiast.com/DL/Tensorflow_Tutorial.html

JetBrains, (2017). *PyCharm. JetBrains. Available at:* https://www.jetbrains.com/pycharm/

Saballe, C. (2019). *Using Machine Learning Models to Predict the Study Path Selection of Business Information Technology Students.* https://www.theseus.fi/handle/10024/171738

Appendices

Appendix 1. Research Survey

BITe Study Path Survey

* Required

Part 1. Study Path

1. I am currently in my... *

Mark only one oval.

- 1st Semester
- 2nd Semester
- 3rd Semester
- 4th Semester
- 5th Semester
- 6th Semester
- 7th Semester
- 8th Semester and over

2. My main/preferred specialization path is... *

Mark only one oval.

- Software Development
- Digital Service Design
- Business and ICT
- ICT Infrastructure

Part 2. Study profile

Rate your agreement for each statement from a scale of 1 (strongly disagree) to 7 (strongly agree)

3. "I like to learn new languages" *

Mark only one oval.

	1	2	3	4	5	6	7	
Strongly disagree	◯	◯	◯	◯	◯	◯	◯	Strongly agree

4. "I am interested in technology." *

Mark only one oval.

	1	2	3	4	5	6	7	
Strongly disagree	◯	◯	◯	◯	◯	◯	◯	Strongly agree

5. "I want to understand how people use technology." *

Mark only one oval.

	1	2	3	4	5	6	7	
Strongly disagree	◯	◯	◯	◯	◯	◯	◯	Strongly agree

6. "It is important that I can be creative in my work." *

Mark only one oval.

	1	2	3	4	5	6	7	
Strongly disagree	◯	◯	◯	◯	◯	◯	◯	Strongly agree

7. "I solve problems specifically with the end goal in mind." *

Mark only one oval.

	1	2	3	4	5	6	7	
Strongly disagree	◯	◯	◯	◯	◯	◯	◯	Strongly agree

8. "An important goal for me is to do well in my studies." *

Mark only one oval.

	1	2	3	4	5	6	7	
Strongly disagree	◯	◯	◯	◯	◯	◯	◯	Strongly agree

9. "I enjoy working in an environment where there is always something new going on." *

Mark only one oval.

	1	2	3	4	5	6	7	
Strongly disagree	◯	◯	◯	◯	◯	◯	◯	Strongly agree

10. "I am interested in designing archetypes/prototypes." *

Mark only one oval.

	1	2	3	4	5	6	7	
Strongly disagree	◯	◯	◯	◯	◯	◯	◯	Strongly agree

11. "To acquire new knowledge is an important goal for me in school." *

Mark only one oval.

	1	2	3	4	5	6	7	
Strongly disagree	◯	◯	◯	◯	◯	◯	◯	Strongly agree

12. "I want to work with my hands." *

Mark only one oval.

	1	2	3	4	5	6	7	
Strongly disagree	◯	◯	◯	◯	◯	◯	◯	Strongly agree

13. "My goal is to succeed in school." *

Mark only one oval.

	1	2	3	4	5	6	7	
Strongly disagree	◯	◯	◯	◯	◯	◯	◯	Strongly agree

14. "I always keep myself with up-to-date information on new technological innovations." *

Mark only one oval.

	1	2	3	4	5	6	7	
Strongly disagree	◯	◯	◯	◯	◯	◯	◯	Strongly agree

15. "I would rather work with people than to work with machines." *

Mark only one oval.

	1	2	3	4	5	6	7	
Strongly disagree	◯	◯	◯	◯	◯	◯	◯	Strongly agree

16. "An important goal for me is to learn as much as possible." *

Mark only one oval.

	1	2	3	4	5	6	7	
Strongly disagree	◯	◯	◯	◯	◯	◯	◯	Strongly agree

17. "Career development and promotions are important for me." *

Mark only one oval.

	1	2	3	4	5	6	7	
Strongly disagree	◯	◯	◯	◯	◯	◯	◯	Strongly agree

18. "I would like to invent and develop new devices and applications." *

Mark only one oval.

	1	2	3	4	5	6	7	
Strongly disagree	◯	◯	◯	◯	◯	◯	◯	Strongly agree

19. "I enjoy coming up with new solutions to problems." *

Mark only one oval.

	1	2	3	4	5	6	7	
Strongly disagree	◯	◯	◯	◯	◯	◯	◯	Strongly agree

20. "Salary means a lot to me." *

Mark only one oval.

	1	2	3	4	5	6	7	
Strongly disagree	◯	◯	◯	◯	◯	◯	◯	Strongly agree

Part 3. Demographic details

21. Age *

Mark only one oval.

- ◯ 17 to 22 years old
- ◯ 23 to 28 years old
- ◯ 29 to 34 years old
- ◯ 35 years old and over

22. Gender *

Mark only one oval.

- ◯ Male
- ◯ Female
- ◯ Nonbinary
- ◯ Prefer not to say

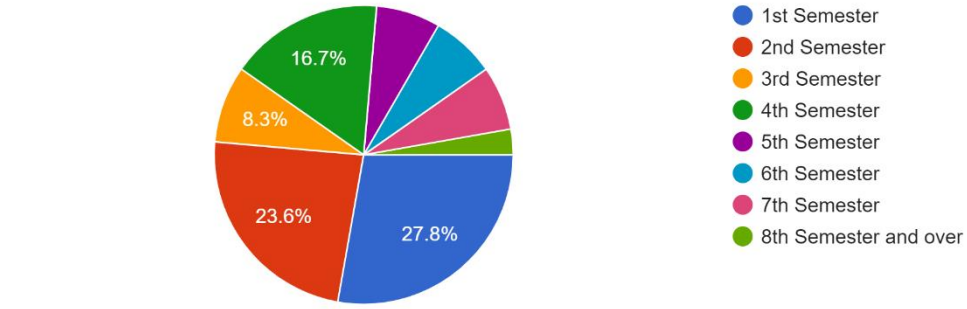
23. I am from... *

Mark only one oval.

- Finland
- Europe (other than Finland)
- Asia and Oceania
- Africa
- North America
- South America

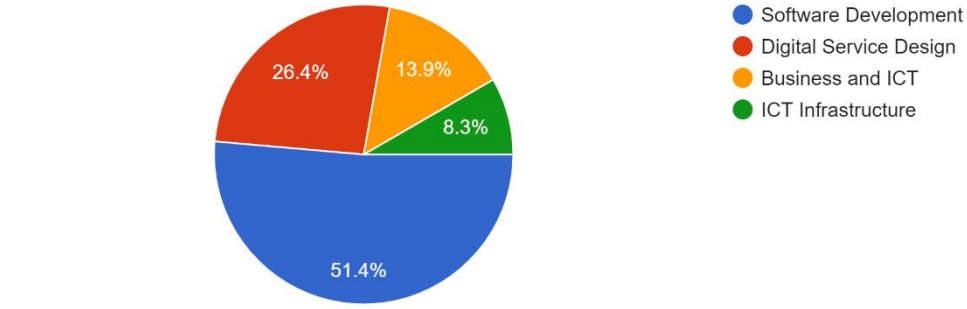
Appendix 2a. Haaga-Helia Respondents by Semester

I am currently in my...
72 responses

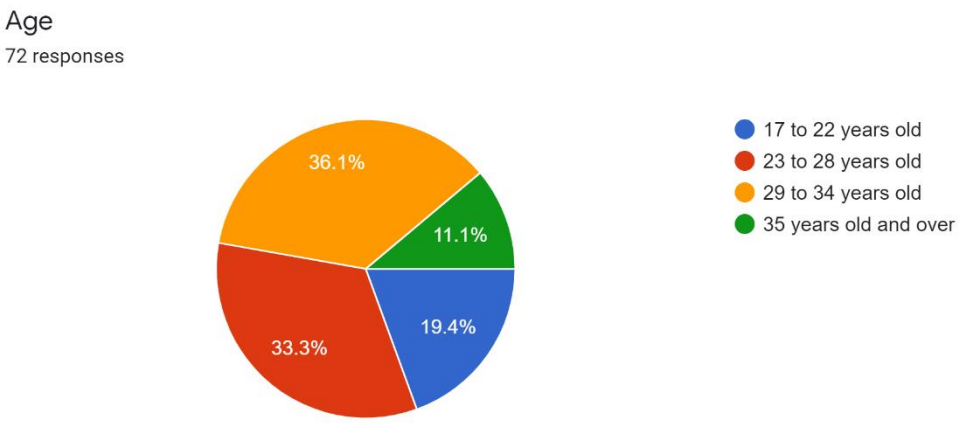


Appendix 2b. Haaga-Helia Respondents by Specialization

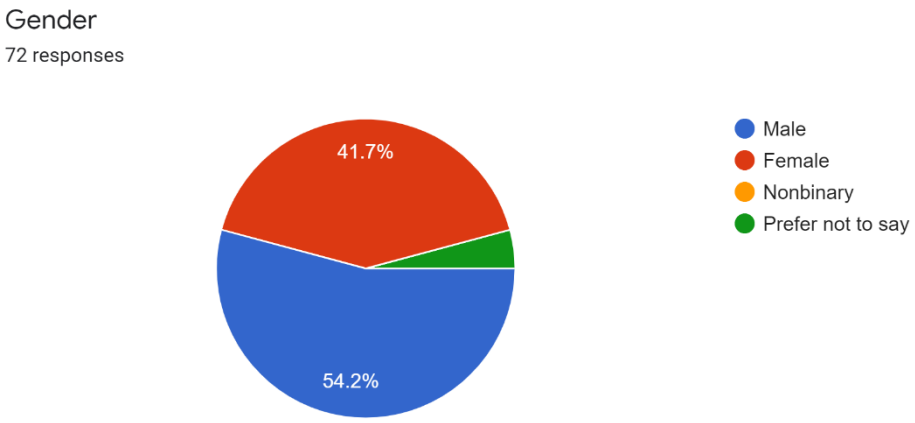
My main/preferred specialization path is...
72 responses



Appendix 2c. Haaga-Helia Respondents by Age



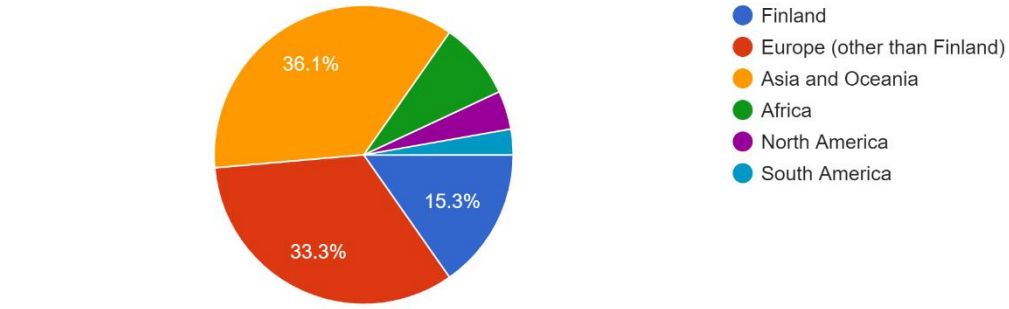
Appendix 2d. Haaga-Helia Respondents by Gender



Appendix 2e. Haaga-Helia Respondents by Area of Origin

I am from...

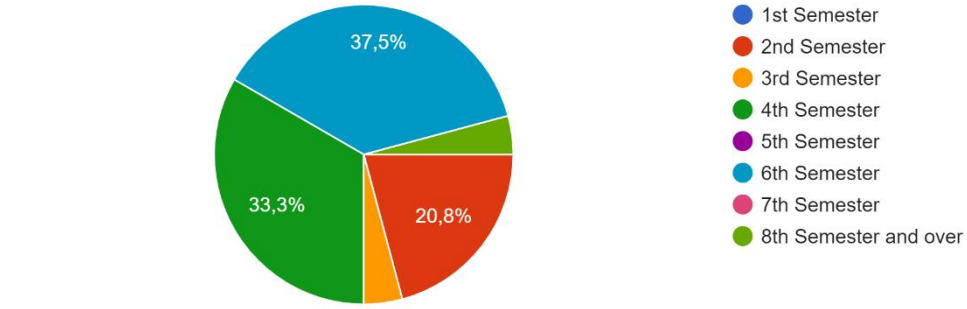
72 responses



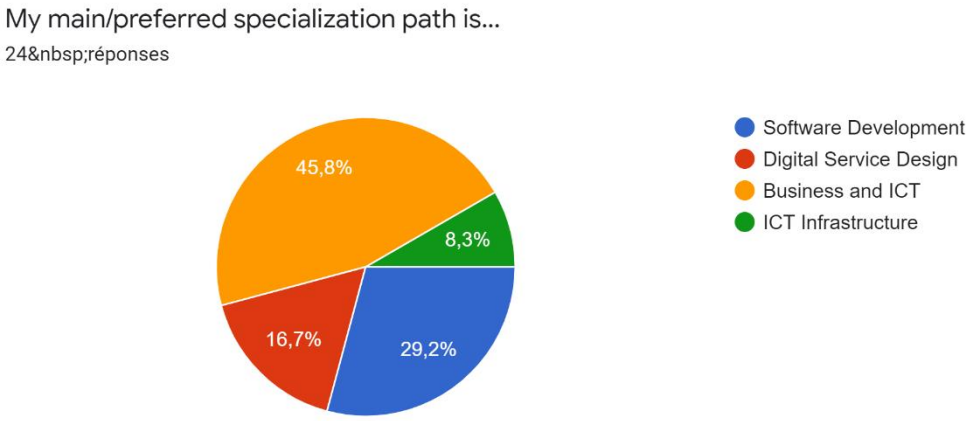
Appendix 3a. HES-SO HE ARC Respondents by Semester

I am currently in my...

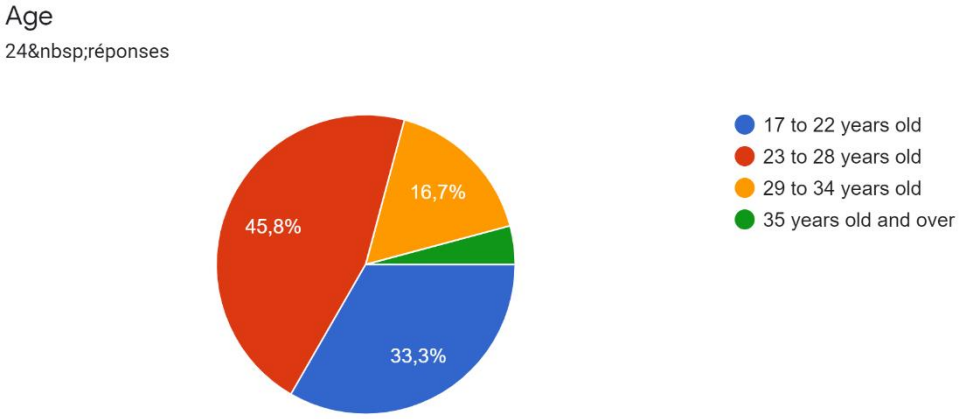
24 réponses



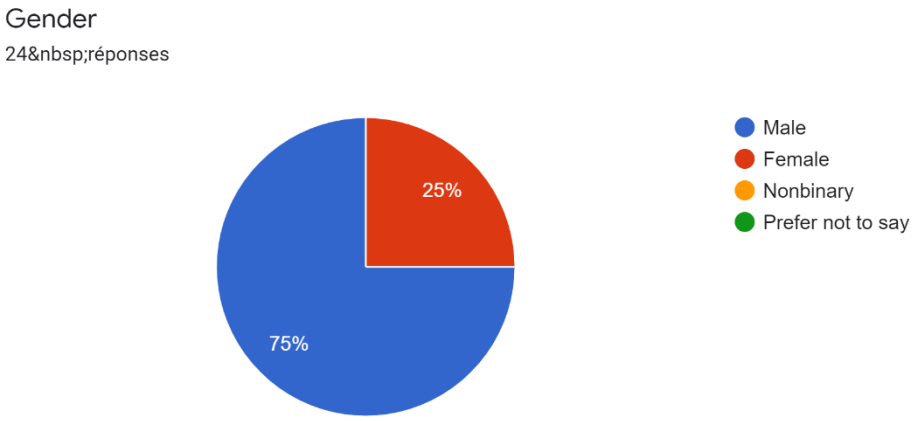
Appendix 3b. HES-SO HE ARC Respondents by Specialization



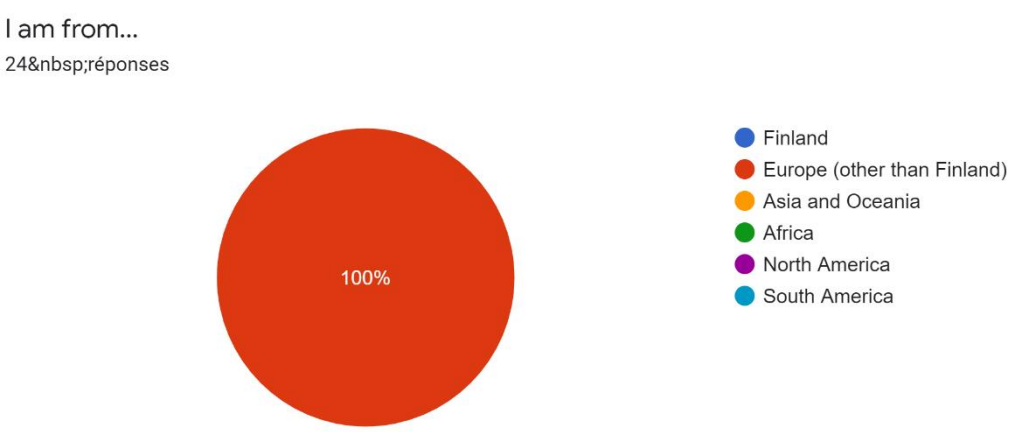
Appendix 3c. HES-SO HE ARC Respondents by Age



Appendix 3d. HES-SO HE ARC Respondents by Gender



Appendix 3e. HES-SO HE ARC Respondents by Area of Origin



Appendix 4. Research permission form

1	Name of Research Project / Thesis Comparing Machine Learning algorithms' accuracy at predicting students' study path
2	The name(s) of person(s) who conduct research Adrien Ruegger Degree Programme and campus Business Information Technology Double Degree Exchange Student in Pasila Campus
3	Supervisor of research (name, status/job, telephone number, e-mail address) Amir Dirin
4	Summary of research plan Research on study path and student preference
5	Client/sponsor N/A
6	Target group of research and sample size At least a 100 BITE students
7	Timetable of research February-March 2021
8	Description of research method Questionnaire

9	Date, signature and address of the applicant / student Date 18.02.2021 Name Adrien Ruegger Signature _____ Phone [redacted] E-mail [redacted] Address [redacted]
10	Date and signature of the supervisor Date [redacted] Name [redacted] Signature_____
11	Return the application to the following address by email: [redacted] or by mail: [redacted]
12	Decision ☐ Permission for research is given ☐ Permission for research is denied Date ___ . ___ 20 __ Application approved by (signature) _____ Name in block letters _____

Staff approving the application:

- Surveys to students: Satu Koivisto
- Surveys to Haaga-Helia staff: Teemu Kokko
- Surveys to alumni: Eva Loippo-Sännälä

Appendix 5. Scikit-learn algorithms code

```
from __future__ import absolute_import, division, print_function,
unicode_literals

from sklearn.linear_model import LogisticRegressionCV
from sklearn.svm import SVC
from sklearn import neighbors, datasets
from sklearn.model_selection import train_test_split
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.tree import DecisionTreeClassifier
import numpy as np
import pandas as pd

# Loading the two csv Data files
data = pd.read_csv("HH BITe Study Path Survey.csv")
data.update(pd.read_csv("HES-SO BITe Study Path Survey.csv"))

# Converting qualitative data into boolean columns
data = pd.get_dummies(data, columns=['Age', 'Gender', 'Origin'])

# Separating label from features
y = data.pop('Specialization').values
X = data.values

# Encoding label as numeric categories
label_encoder = LabelEncoder()
y = label_encoder.fit_transform(y)
X = StandardScaler().fit_transform(X)

# Declaring result arrays
LGRs = []
SVMs = []
KNNs = []
DTCs = []

# Looping for a 500 tries
for i in range(1, 501):

    # Separating training data and testing data, Stratify keeps the
proportion of data in both sets
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=1 /
4, stratify=y)

    # Pipeline for Logistic Regression with Cross Validation
    clfLGR = make_pipeline(StandardScaler(),
LogisticRegressionCV(random_state=i, max_iter=5000, cv=3))
    clfLGR.fit(X_train, y_train.ravel())
    LGRs.append(clfLGR.score(X_test, y_test))

    # Pipeline for Support-Vector Machine
    clfSVM = make_pipeline(StandardScaler(), SVC(C=1, gamma='auto'))
    clfSVM.fit(X_train, y_train.ravel())
    SVMs.append(clfSVM.score(X_test, y_test))
```

```
    # Loop to use K value from 1 to 20
    for j in range(1, 21):
        # Pipeline for K-Nearest Neighbor
        clfKNN = make_pipeline(StandardScaler(),
neighbors.KNeighborsClassifier(j))
        clfKNN.fit(X_train, y_train.ravel())
        KNNs.append(clfKNN.score(X_test, y_test))

    # Pipeline for Decision Tree
    clfDTC = make_pipeline(StandardScaler(), DecisionTreeClassifier())
    clfDTC.fit(X_train, y_train.ravel())
    DTCs.append(clfDTC.score(X_test, y_test))

# Printing best and worst accuracy results for each algorithm across all
the tries
print('---- Logistic Regression accuracy \n Best: ' +
"{:.0%}".format(max(LGRs))
      + '\n Worst: ' + "{:.0%}".format(min(LGRs))
      + '\n Mean: ' + "{:.0%}".format(np.mean(LGRs)))

print('---- Support-Vector Machine accuracy \n Best: ' +
"{:.0%}".format(max(SVMs))
      + '\n Worst: ' + "{:.0%}".format(min(SVMs))
      + '\n Mean: ' + "{:.0%}".format(np.mean(SVMs)))

print('---- K-Nearest Neighbor \n Best: ' + "{:.0%}".format(max(KNNs))
      + '\n Worst: ' + "{:.0%}".format(min(KNNs))
      + '\n Mean: ' + "{:.0%}".format(np.mean(KNNs)))

print('---- Decision Tree accuracy \n Best: ' + "{:.0%}".format(max(DTCs))
      + '\n Worst: ' + "{:.0%}".format(min(DTCs))
      + '\n Mean: ' + "{:.0%}".format(np.mean(DTCs)))
```

Appendix 5. TensorFlow deep learning code

```
from __future__ import absolute_import, division, print_function,
unicode_literals

from sklearn.linear_model import LogisticRegressionCV
from sklearn.svm import SVC
from sklearn import neighbors, datasets
from sklearn.model_selection import train_test_split
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.tree import DecisionTreeClassifier
import tensorflow as tf
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd


# defining a function for the neural network input
def input_fn(features, labels, training=True, batch_size=256):
    # Convert the inputs to a Dataset.
    dataset = tf.data.Dataset.from_tensor_slices((dict(features), labels))

    # Shuffle the data and repeat if using training mode.
    if training:
        dataset = dataset.shuffle(1000).repeat()

    return dataset.batch(batch_size)


# Loading the two csv Data files
data = pd.read_csv("HH BITe Study Path Survey.csv")
data.update(pd.read_csv("HES-SO BITe Study Path Survey.csv"))

# Converting qualitative data into boolean columns
data = pd.get_dummies(data, columns=['Age', 'Gender', 'Origin'])

# Separating label from features
y = data.pop('Specialization').values
X = data.values

# Saving the column names for conversion later
X_columns = data.keys()

# Encoding label as numeric categories
label_encoder = LabelEncoder()
y = label_encoder.fit_transform(y)
X = StandardScaler().fit_transform(X)

results = []

# Looping for a 500 tries
for i in range(1, 501):
    # Separating training data and testing data, Stratify keeps the
proportion of data in both sets
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=1 /
4, stratify=y)
```

```
    # Transform the data into pandas dataframes for tensorflow
compatibility
    X_train = pd.DataFrame(X_train, columns=X_columns)
    X_test = pd.DataFrame(X_test, columns=X_columns)
    y_train = pd.DataFrame(y_train, columns=['Specialization'])
    y_test = pd.DataFrame(y_test, columns=['Specialization'])

    # Remove blank spaces in column names to avoid errors later on
    X_train.columns = X_train.columns.str.replace(' ', '_')
    X_test.columns = X_test.columns.str.replace(' ', '_')
    y_train.columns = y_train.columns.str.replace(' ', '_')
    y_test.columns = y_test.columns.str.replace(' ', '_')

    # Feature columns describe how to use the input.
    my_feature_columns = []
    for key in X_train.keys():

my_feature_columns.append(tf.feature_column.numeric_column(key=key))
    print(my_feature_columns)

    # Build a Neural network using 2 hidden layers.
    classifier = tf.estimator.DNNClassifier(
        feature_columns=my_feature_columns,
        # Two hidden layers of 30 and 10 nodes respectively.
        hidden_units=[30, 10],
        # The model must choose between 4 classes/output.
        n_classes=4)

    # Training the model
    classifier.train(
        input_fn=lambda: input_fn(X_train, y_train, training=True),
        steps=4000)
    # We include a lambda to avoid creating an inner function

    # Testing the model's accuracy
    eval_result = classifier.evaluate(
        input_fn=lambda: input_fn(X_test, y_test, training=False))
    results.append(eval_result.get('accuracy'))

# Printing the mean, best and worst accuracy results for each algorithm
across all the tries
print('---- Tensorflow Neural Network \n Best: ' +
"{:.0%}".format(max(results))
      + '\n Worst: ' + "{:.0%}".format(min(results))
      + '\n Mean: ' + "{:.0%}".format(np.mean(results)))
```