# Final Report: G05

Mohamed Bouzid - 40221466 Bouzidmed1897@gmail.com

*Parmiss Shahinfard* - 40260687 Parmiss.sh98@gmail.com

Mohamadreza Azar Nasrabadi - 40290372 *mohamadreza.azar-nasrabadi.1@ens.etsmtl.ca*

Mary Acquah - 40259635 acquahmarypaz@gmail.com

*Abstract*

***The project focuses on the optimization and evaluation of different machine learning techniques for predicting customers' subscriptions to a term deposit in the banking industry. The project utilizes the "Bank Marketing" dataset, which contains customer information and details about marketing campaigns. The dataset is preprocessed by handling missing values, encoding categorical variables, and handling imbalanced classes by oversampling the data. This report explores three machine learning techniques: decision trees, semi-structured decision trees, and deep neural network (DNN). The performance of the models is evaluated using metrics such as accuracy, precision, recall, and F1 score. It was found that the semi-structured decision trees model outperformed both the optimized decision tree and DNN models in predicting customers' subscriptions to a term deposit. Therefore, the semi-supervised decision tree model can be considered the better technique for this particular task based on the experimental results.***

***Keywords—Marketing strategy, Decision Tree, Deep Neural Network, Classification***

## I. INTRODUCTION

In this study, we analyze a dataset focused on direct marketing campaigns carried out by a Portuguese banking institution through phone calls. The objective is to accurately predict whether clients will subscribe to a term deposit, allowing for tailored strategies that result in higher conversion rates and customer acquisition. To accomplish this prediction, three distinct machine learning techniques are employed to determine the most effective approach for forecasting customers' future subscriptions to a term deposit.

Associated challenges with respect to the problem include dealing with imbalanced data, handling missing values, finding a balance between accuracy and precision, managing outliers, and identifying suitable architectures for the models.

In the literature, these challenges have been addressed through various methods. For example, class imbalance in classification tasks leads to poor predictions for the minority class due to algorithms prioritizing the majority class. For example, A study conducted by Thabtah et al (2020) explores the relationship between class imbalance and classifier performance, highlighting the need to address this issue and its impact on applications like autism spectrum disorder screening.

In addition, to solve missing values, Poolsawad et al. (2012) proposed an innovative missing value imputation called Collateral Missing Value Estimation and dimensionality reduction techniques for clinical datasets. To handle missing values, various imputation methods such as mean imputation, expectation-maximization algorithm, k-nearest neighbor imputation, and artificial neural network imputation were applied.

In this report, our goal is to address these challenges by using oversampling for imbalanced data, replacing missing values with the most frequent values, and performing grid search to find optimal architectures. We explore three different models: decision trees with supervised learning, decision trees with semi-supervised learning, and neural networks with supervised learning. In the neural network implementation, we incorporate enhancement methods such as weight decay, batch normalization, and dropout.

Our results show approximately 80% accuracy and precision for supervised learning models and 90% accuracy and precision for semi-supervised learning models.

B) Related work

Spanhol et al. (2015) created the BreaKHis dataset for breast cancer histopathology images and proposed a classification method to distinguish between benign and malignant images. They emphasized the need for improvements in exploring magnification factors, combining classifiers, and reducing false positives. They utilized segmentation, texture representation, and four classifiers to evaluate features, showing potential accuracy enhancement through better descriptors or selection strategies. Additionally, Van Horn et al. (2018) introduced the iNaturalist dataset for species classification, highlighting the challenge of imbalanced datasets. They employed CNN models and found that current methods achieved only 67% top-one classification accuracy, indicating the dataset's difficulty.

## II. METHODOLOGIES

### *A. Dataset*

The dataset utilized in this study consists of direct marketing campaign data conducted by a Portuguese banking institution via phone calls. It was collected for research purposes and contains valuable information for predicting client subscriptions to term deposits. The dataset comprises 45,211 unique customer records with a total of 16 attributes. These attributes include a mix of numerical and categorical features, such as client information (age, job, education, housing loan, etc.), details about the marketing campaigns (day, month, duration, type), social and economic attributes, and other variables (campaign, previous contact, etc.). in figure 1 you can see the distribution of the votes among

different jobs, which gives insightful information about the dataset.

To prepare the dataset for model training, various preprocessing steps were undertaken. Firstly, missing values were addressed using the most frequent method. Since the missing values were initially represented as 'unknown' instead of NaN, these string values were replaced with NaN to ensure accurate processing. Furthermore, categorical variables were encoded using one-hot encoding, and numerical columns were appropriately scaled using the StandardScaler.

The dataset was divided into three sets: training, validation, and test, with the training set representing 80% of the data for model training. The remaining 20% was evenly split between the validation and test sets. This partitioning allowed for model optimization and assessing its generalization capability. To account for the dataset's imbalanced nature, where positive instances were significantly outnumbered, precision was chosen as the primary evaluation metric to prioritize accurate classification of positive instances, considering that accuracy alone could be misleading.

The random seed value used for reproducibility, set to 25. It ensures that the algorithm's results remain consistent across different runs.

To address overfitting and ensure model stability, a five-fold cross-validation approach was also employed to evaluate the model. The training set was divided into five subsets or

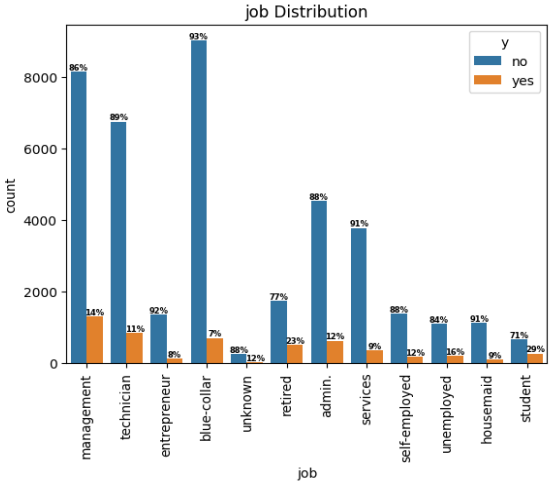


*Figure 1 - distribution of votes among different job categories*

folds, using each fold as a validation set once while training the model on the remaining four folds. This process was repeated five times, ensuring that each fold served as a validation set exactly once. By applying cross-fold validation, a robust estimate of model performance across different subsets of the data was obtained, offering reliable insights into its predictive capabilities.

### B. *Decision Tree*

The architecture of the decision tree model involves utilizing the scikit-learn library to implement the decision tree algorithm. The dataset is split into training and testing sets. The hyperparameters of the decision tree model, including the criterion (entropy) for information gain, maximum depth (6), and minimum samples split (4), are defined. This architecture was achieved in an extensive grid search during the optimization steps.

The decision tree classifier is created and trained using the training data. Predictions are then made on the test set, and the accuracy of the model is calculated. Additionally, performance metrics such as precision, recall, F1 score, and F-beta score are computed, along with a confusion matrix, to evaluate the model's performance. To enhance understanding, the decision tree is visualized using a plot, and a Graphviz dot file is exported for further visualization. Finally, cross-validation is performed on the entire dataset to assess the model's generalization ability, displaying cross-validation scores, mean accuracy, and standard deviation of accuracy.

### C. *Semi-supervised Decision Trees*

The implemented algorithm utilizes a semi-supervised decision tree architecture with specific hyperparameters. These hyperparameters include:

- **Labeled Ratio:** The percentage of labeled samples in the dataset, set to 0.2. This determines the proportion of data that has ground truth labels available for training.
- **Max Depth:** The maximum depth of the decision tree classifier, set to 4. This parameter controls the complexity and depth of the resulting decision tree.
- **Criterion:** The criterion used to measure the quality of a split in the decision tree, set to 'entropy'. This criterion evaluates the information gain based on the entropy of the target variable.
- **Confidence Threshold:** The confidence threshold for adding samples to the labeled dataset, set to 0.8. This threshold determines the level of certainty required for a prediction to be considered as high-confidence and included in the labeled dataset.
- **Iteration Limit:** The maximum number of iterations for predicting labels in the unlabeled data using the trained classifier, set to 5. This limit controls the number of iterative steps taken to incorporate unlabeled data into the training process.

The algorithm follows an iterative process to progressively improve its performance. Initially, an initial decision tree classifier is trained on the labeled dataset, and its performance is evaluated on a test set. The trained classifier is then used to predict labels for the unlabeled data, with a limit of 5 iterations. During each iteration, samples with predictions above the confidence threshold are added to the labeled dataset, and the classifier is retrained. This iterative process continues for the specified number of iterations.

After the iterations, the final classifier is used to predict labels for the remaining unlabeled data, and performance metrics are calculated on a combined test set. The model's generalization ability is evaluated through cross-validation.

### D. *Deep Neural Network (DNN)*

The provided code implements a neural network model specifically designed for binary classification tasks. The hyperparameters and settings were carefully chosen through a grid search process. The model architecture consists of two hidden layers, each with a hidden size of 64 and ReLU

activation functions. This choice is suitable for binary classification. The Cross Entropy Loss function is used to guide the model's parameter optimization by measuring the dissimilarity between predicted probabilities and true labels. This loss function is well-suited for training models to classify between two classes, as it maximizes the model's ability to differentiate between 'yes' and 'no' outcomes.

To prevent overfitting, a dropout rate of 0.1 is employed for regularization. This rate strikes a balance between reducing overfitting and preserving sufficient information for learning. Tuning the dropout rate aims to prevent overfitting while still capturing important patterns in the data.

During the grid search, learning rates of 0.1 and 0.01 were explored. The learning rate determines the step size at which the optimizer updates the model's parameters during training. A higher learning rate can accelerate convergence but risks overshooting the optimal solution, while a lower learning rate can improve stability but may lead to slower convergence. Trying different learning rates aims to find the optimal value that balances convergence speed and stability.

The model is trained using the Adam optimizer, which combines the benefits of AdaGrad and RMSProp optimizers. Adam adapts the learning rate for each parameter individually and is known for its efficiency and good generalization performance. By choosing Adam, the aim is to optimize both convergence speed and model performance.

The training process consisted of 10 epochs, and early stopping with a patience of 5 was implemented. Figure 2 illustrates the accuracy improvement throughout the epochs, indicating that after 10 epochs, the rate of accuracy increase becomes significantly slower. This observation suggests that the model's performance is unlikely to improve significantly beyond this point. Early stopping was employed to mitigate the risk of overfitting by stopping the training process if the model's performance did not show substantial improvement within the specified number of epochs. By utilizing early stopping, we ensured that the model achieved a balance between learning from the data and avoiding overfitting.

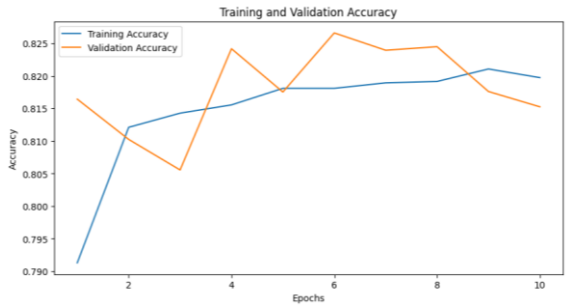


*Figure 2 - Training and Validation accuracy throughout epochs*

The computational complexity is estimated to be 2.23 seconds per epoch, with 9344.0 Floating Point Operations (FLOPS) involved.

Considering that we did not use a batch normalization layer, we evaluated its inclusion or exclusion during the grid search. By comparing the model's performance with and without batch normalization, we could determine its impact on convergence speed and model performance. This allowed us to select the configuration that achieves the best trade-off between convergence and stability.

### *E. Optimization Algorithm*

#### *1) Decision Trees*

First, a custom precision function is defined to facilitate precision calculation based on the confusion matrix. This function plays a crucial role in evaluating the classifier's performance.

To address the issue of class imbalance, the minority class samples are unsampled using the resampling technique, resulting in a balanced dataset. This step ensures that the classifier receives sufficient representation from both classes during training.

To mitigate the influence of varying feature scales, numerical features in the training and testing sets are standardized using the StandardScaler. This scaling process brings the features to a comparable range, promoting more reliable model training and evaluation. To optimize the decision tree classifier, a systematic hyperparameter tuning is performed. A grid search technique, accompanied by cross-validation with five folds, is employed. The hyperparameters considered include the maximum depth of the decision tree, the criterion for information gain (either 'gini' or 'entropy'), and the minimum number of samples required to split an internal node. The decision tree classifier serves as the base estimator for this process.

Upon completion of the grid search, the best hyperparameters are determined based on the achieved precision score, as measured by the custom precision function. These optimal hyperparameters are then utilized to instantiate a new decision tree classifier. The newly created classifier undergoes training using the training set, allowing it to learn the underlying patterns and relationships within the data.

The adopted approach in the provided code encompasses crucial steps in model validation and optimization. It addresses class imbalance, explores hyperparameter space, and utilizes rigorous evaluation metrics to ensure the classifier's performance is thoroughly assessed.

we can see that after the optimization steps are performed, Despite a slight decrease in accuracy, the new model outperforms the previous model in terms of precision, recall, and F1 score. With a precision of 0.834 compared to 0.625, the new model makes fewer false positive predictions, making it more reliable for identifying potential subscribers in the imbalanced bank marketing dataset. Additionally, the higher recall of 0.796 compared to 0.342 indicates that the new model captures positive class instances more effectively. Overall, the improved precision, recall, and F1 score demonstrate that the new model offers a more accurate and valuable solution for optimizing marketing strategies and identifying potential customers likely to subscribe.

#### *2) Deep Neural Network (DNN)*

To enhance the convergence speed, stability, and generalization capability of the Deep Neural Network (DNN) model, we employed various optimization techniques tailored to the characteristics of the bank marketing dataset. The optimization algorithm used for training the DNN model was the Adam optimizer, which combines the benefits of both Adaptive Gradient Algorithm (AdaGrad) and Root Mean Square Propagation (RMSProp), resulting in effective parameter updates during training.

The following optimization techniques were applied:

**Handling Imbalanced Data:** We employed the same technique used in Decision Trees (DTs) to handle imbalanced

data in the bank marketing dataset. This technique aims to address the issue of imbalanced class distribution and improve the model's ability to learn from minority class samples.

**Parameter Initialization:** Xavier uniform initialization was used to mitigate the vanishing or exploding gradient problem. By initializing the weights properly, we facilitate faster convergence and better performance by preventing gradients from becoming too small or too large.

**Mini-Batch Stochastic Gradient Descent (SGD):** To address memory efficiency and accelerate convergence, we utilized mini-batch SGD. This technique involves processing small subsets (mini-batches) of the training data at each iteration, reducing memory requirements and enabling efficient parallel processing.

**Early Stopping:** Early stopping was implemented with a patience of 5. If the validation loss did not improve for 5 consecutive epochs, training was stopped early. Early stopping prevents the model from overfitting to non-representative or noisy patterns, thus improving generalization.

**Grid Search for Hyperparameter Tuning:** We performed a grid search specifically for the hyperparameters related to the number of hidden layers, activation function, and batch normalization layer. By systematically exploring different combinations of these hyperparameters, we aimed to find the optimal configuration that balances model complexity, convergence speed, and stability.

The following hyperparameters were considered during the grid search:

A. **Batch Normalization Layer:** We evaluated the inclusion or exclusion of a batch normalization layer after each hidden layer. This allowed us to study its impact on convergence speed and model performance, selecting the configuration that achieved the best trade-off between convergence and stability.

B. **Learning Rate:** We explored two learning rates, 0.1 and 0.01, to determine the optimal value that balances convergence speed and stability. The learning rate determines the step size at which the optimizer updates the model's parameters during training.

C. **Number of Hidden Layers:** Different numbers of hidden layers (1, 2, 3, and 4) were explored using a grid search approach. This range allowed us to find the optimal architecture that balances model complexity and the capacity to capture relevant patterns in the dataset, while avoiding overfitting.

D. **Activation Function:** Two activation functions, ReLU and sigmoid, were considered using a grid search approach. ReLU introduces non-linearity and helps the model learn complex relationships, while sigmoid is suitable for producing a probability-like output between 0 and 1.

E. **Batch Size:** We experimented with different batch sizes during training. A smaller batch size introduces more noise into the gradient estimation but allows for more frequent updates, potentially leading to faster convergence. On the other hand, a larger batch size reduces the noise but updates the parameters less frequently.

The results of the grid search, along with the evaluation metrics, were recorded in the "grid_search_results" pandas DataFrame for analysis and comparison.

By employing these optimization techniques and conducting a comprehensive grid search, we successfully validated and optimized the DNN model, resulting in improved performance for the bank marketing task.

## III. Results

### *A. Experimental setup*

In the present study, the bank marketing dataset from the UCI Machine Learning Repository was utilized. Preprocessing techniques were employed to prepare the data for experimentation, encompassing procedures such as handling missing values and encoding categorical variables. Further details regarding these preprocessing steps have been elaborated upon earlier. To ensure consistency and comparability of model performance, the dataset was divided into train and test sets, with the test set encompassing 20% of the data. A fixed random state of 25 was employed to maintain uniform conditions across model evaluations. Regarding performance evaluation, four metrics were selected for analysis: precision, accuracy, recall, and F1-score.

The accuracy metric measures the proportion of correctly predicted values relative to the total number of samples in the dataset. While useful for balanced datasets, accuracy can yield misleading results when imbalances exist, favoring the majority class. To address this issue, emphasis was placed on achieving a balance between high precision for the class with a smaller number of samples. The F1-score, commonly employed for evaluating multi-class classification performance, is calculated as a combination of precision and recall. Precision quantifies the proportion of true positives relative to the sum of true positives and false positives, while recall measures the proportion of true positives relative to the sum of true positives and false negatives. The F1-score (with Beta = 1) is expressed as:

$$\text{F1-score} = 2 \times (\text{Precision} \times \text{Recall}) / (\text{Precision} + \text{Recall})$$

Cross-validation accuracy, another metric utilized in this study involved employing a 5-fold cross-validation approach to compute the mean accuracy. This technique involves partitioning the training set into five subsets, with 20% of the data allocated as the validation set in each iteration.

Furthermore, the training time of each model was measured to facilitate comparison of computational efficiency. Various experimental setups were implemented to optimize and validate the models utilized. For the decision tree model, an initial configuration with a maximum depth of 3 and an entropy criterion was adopted. The imbalanced nature of the dataset was addressed through upsampling of the minority class, which involved randomly duplicating samples from the minority class to match the majority class. Additionally, feature scaling was applied to the numerical features in the dataset.

Hyperparameter tuning was performed using GridsearchCV for the decision tree model. The hyperparameters considered included the maximum depth ('max_depth'), the criterion for node splitting ('criterion'), and the minimum number of samples required to split an internal node ('min_samples_split'). These hyperparameters are crucial in regulating the complexity and generalization capacity of the decision tree model. The 'max_depth' parameter determines the maximum depth of the tree, limiting the number of splits to prevent overfitting. The 'criterion' parameter specifies the quality measure employed for evaluating splits, offering options such as entropy and Gini impurity. The 'min_samples_split' parameter defines the minimum number of samples required to split an internal node, serving as a regularization mechanism to avoid overly specific splits that may lead to overfitting. Through exploration of various hyperparameter combinations, an optimal configuration was sought that achieved a balance between model complexity and generalization performance.

Moving on to the semi-supervised decision tree, the following hyperparameters were utilized: a labeled sample percentage of 20%, a maximum depth of 4, an entropy criterion, an 80% confidence interval, and 5 iterations. The initial Decision Tree Classifier was trained on the labeled training set, and performance metrics were computed for the labeled test set. Subsequently, semi-supervised learning was performed on unlabeled data using the trained model. This involved predicting labels for the unlabeled data, obtaining confidence scores for the predicted labels, and selecting samples with high confidence scores above the designated threshold. The high confidence samples were then added to the labeled data, while being removed from the unlabeled data. The decision tree model was subsequently retrained on the updated labeled data.

Regarding the supervised learning classification with deep learning model, an initial architecture comprising three fully connected layers was employed, with ReLU activation functions following the first two layers. The hidden size utilized in this initial attempt was set to 64. The Cross-entropy loss function and Adam optimizer with a learning rate of 0.01 were employed, and the model was trained for 20 epochs. To optimize the deep learning model, techniques discussed in the previous section were incorporated, taking into account the characteristics of the bank marketing dataset.

Grid search was utilized to identify the optimal configuration for the deep learning model. The number of hidden layers was explored, including values of 1, 2, 3, and 4, aiming to test various model complexities while avoiding overfitting. Two activation functions, ReLU and sigmoid, were considered based on the binary classification nature of the problem. Additionally, the presence of a batch normalization layer after each hidden layer was evaluated for its impact on convergence speed and model performance. Grid search was also employed to explore different learning rates, including 0.1 and 0.01, to examine varying convergence speeds. Learning rates lower than 0.01 were excluded to prevent excessive model training time. Finally, two batch sizes, 32 and 64, were explored, with these values commonly employed to ensure frequent updates for faster convergence, considering the dataset size.

By systematically exploring the various hyperparameters and architecture choices through grid search, the optimal configuration for the deep learning model was determined. This configuration entailed a hidden size of 64, three hidden layers, a dropout rate of 0.1, a learning rate of 0.01, a ReLU activation function, a cross-entropy loss function, the inclusion of batch normalization layers, and a batch size of 64.

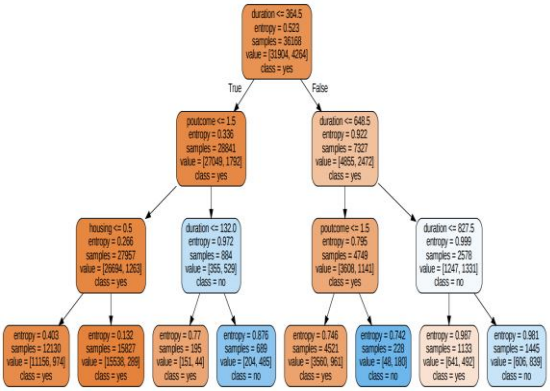


*Figure 3: Split of the initial attempt for DT*

## B. *Main Results*

The experiments conducted in the section above resulted in different results. Starting with the decision tree, the initial attempt did not result in good results. Even though the accuracy was 90%, the precision, the recall, and the F1-score were respectively equal to 62.45%, 34.24%, 44.23%. These results could be explained by the fact that the decision tree could not find a good split of the tree to distinct between predictions and is also due to the lack of optimization. In fact, by applying the optimization and running the model with the best hyperparameters found by the grid search, the results improved to 83,37%,79.59%, and 81.44% respectively for the precision, recall, and F1-score. The accuracy in the other hand, decreased to 81.85%. This could be explained by the fact that the dataset got upsampled, so the results are less misleading in this case.

Next, Semi-supervised learning classification with a decision tree was evaluated. The results outperformed the previous decision tree in terms of precision, recall, and F1 score. With a precision of 94.62% compared to 83,37%, the new model makes fewer false positive predictions, making it more reliable for identifying potential subscribers in the imbalanced bank marketing dataset. Additionally, the higher recall of 81.53% compared to 79.59%, indicates that the new model captures positive class instances more effectively. The accuracy and F1-score did improve as well to 94.88% and 87.59% respectively. Overall, the improved accuracy, precision, recall, and F1 score demonstrate that the new model offers a more accurate and valuable solution for optimizing marketing strategies and identifying potential customers likely to subscribe. Moving to the DL model, the initial setup did not result in good results with accuracy, precision, recall, and F1-score of 87.91%, 49.66%, 13.20%, and 20.85%. This could be explained by the fact that the

neural network is too simple to capture any patterns in the dataset.

The optimization described above resulted in accuracy, precision, recall, and F1-score of 83.04%, 82.28%, 84.26%, and 83.26%. For this model, we have also calculated the number of FLOPS which is equal 9344.0. The wall time for one epoch of training is 2.23 seconds. The different results of the tested models are summarized in the table 1. In terms of performance, the semi-supervised decision tree model stands out with the highest accuracy, precision, and cross-validation accuracy among the three models. It shows great potential for accurately predicting the outcomes of the bank marketing dataset.

*Table 1 - Comparing results of different approaches.*

| Model | Training Time | Accuracy | Precision | Recall | F1-score | Cross-validation accuracy |
|---|---|---|---|---|---|---|
| DT | 0.82 s | 82% | 83% | 80% | 81% | 71% |
| DT(semi-supervised) | 0.02 s | 95% | 95% | 82% | 88% | 93% |
| DNN | 33.76 | 84% | 83% | 86% | 84% | 83% |

### C. *Ablative Study*

In the final part of our project, we conducted an ablative study by tweaking different hyperparameters to see the effect of each one. For example, for the DT, the increasing of maximum depth from 3 to 6 improved the precision of the model from 62.45% to 83.37%. The addition of min-samples split di also have a good effect on the results. Next, for the NN, we visualized different number of hidden layers against the test accuracy. The results are shown in Figure 4 below.

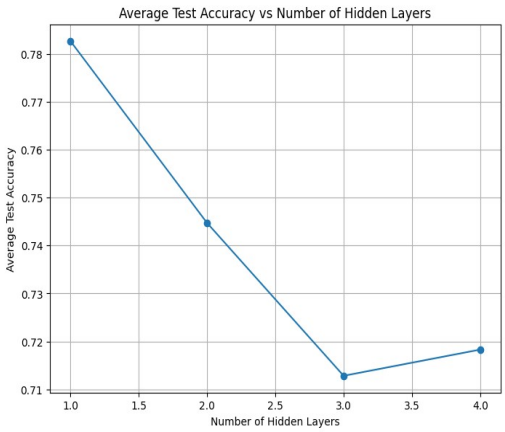


*Figure 4: Average test accuracy vs number of hidden layers*

We can see that adding more complexity and more hidden layers does not necessarily improve the accuracy of the model. Moreover, we conducted an ablative study for 4 other hyperparameters that are the batch_size, the activation function, the learning rate, and the presence of batch normalization layer. Results of the study are shown in Figure 5. The results show that increasing batch size to 64 slightly improves the precision of the model. The activation function that resulted in a better performance between ReLU and sigmoid is ReLU since sigmoid is more suitable for the output layer instead of hidden layers. Most importantly, we see that the batch normalization hugely improved the precision by almost 20% which indicated the importance of the normalization of the layers. Similarly, a smaller learning rate of 0.01 improved the results a lot rather than using 0.1. This indicates that smaller steps for gradient helps the model achieve better performance.

## IV. Conclusion

The bank marketing dataset is a good dataset to demonstrate the capability of semi-supervised learning and to compare three different methods. Our dataset consists of a direct marketing campaign conducted where the goal is to predict whether clients will subscribe to a term deposit (y= “yes”) or not (y= “no”). During the project, we have seen that it is important to correctly deal with imbalanced classes and to use appropriate preprocessing for accurate model performance. The first step was to apply EDA and different preprocessing steps to better understand the dataset. Next, three models (DT, Semi-supervised DT, and NN) were applied and their performances were evaluated. Moreover, an ablative study alongside the grid search was conducted and the three models were optimized accordingly. Out of the three models, Semi-supervised DT appeared to be the best-performing model. We can conclude that the goal that we set at the start of the project was reached successfully.

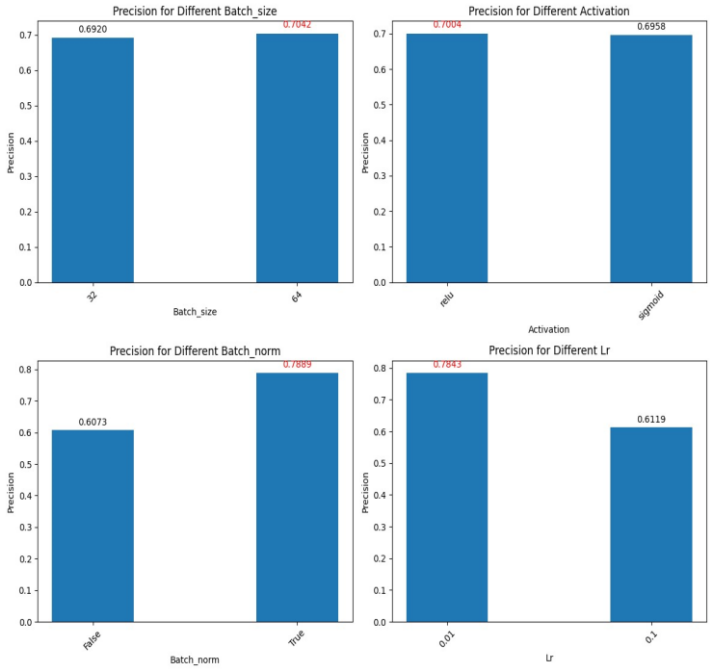


*Figure 5: Precision vs different hyperparameters*

## References


Thabtah, Fadi, et al. "Data imbalance in classification: Experimental evaluation." *Information Sciences* 513 (2020): 429-441.

Poolsawad, Nongnuch, et al. "Handling missing values in data mining-A case study of heart failure dataset." *2012 9th International Conference on Fuzzy Systems and Knowledge Discovery*. IEEE, 2012.

Spanhol, Fabio A., et al. "A dataset for breast cancer histopathological image classification." *Ieee transactions on biomedical engineering* 63.7 (2015): 1455-1462.

Van Horn, Grant, et al. "The inaturalist species classification and detection dataset." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018.