INF8245E - MACHINE LEARNING - FALL 2023 KAGGLE COMPETITION

TEAM : THEORETICALLY INCLINED MONKEYS

Pierre-Alexandre Aubé (2073036)
Department of Biomedical Engineering
Polytechnique Montreal
`pierre-alexandre.aube@polymtl.ca`

Mathéo Benoit-Paraschivoiu (2079549)
Department of Computer and Software Engineering
Polytechnique Montreal
`matheo.benoit-paraschivoiu@polymtl.ca`

Jean Siffert (1984533)
Department of Computer and Software Engineering
Polytechnique Montreal
`jean.siffert@polymtl.ca`

ABSTRACT

The following paper explores the creation of an image classification model, leveraging a dataset comprising 1024 distinct features and $25,515$ data points, with the goal of accurately classifying them among 100 labels. The report employs various classification algorithms and diverse preprocessing techniques to enhance model performance. The methodology, chosen preprocessing methods, algorithms experimented with, and the theoretical framework underpinning our approach are thoroughly discussed. Our developed model achieves a F1-Score of over 98% on the validation set, underscoring its efficacy in accurately categorizing abstract image features.

1 Preprocessing and feature design

The training dataset provided contains 25,515 data samples with corresponding labels. There are an additional 10,000 samples designated for testing. Each row represents a vector with 1,024 elements. The dataset encompasses a total of 100 unique labels.

Sample densities analysis

To gain a better understanding of our data, we carried out several analysis to determine its distribution. First, we performed Shapiro-Wilk tests on all features individually, to test the normality of the univariate distributions. For all features, the p-value of the test is below the 1% confidence level, so we reject the null hypothesis H_0 of normality.

We also performed a Henze Zirkler test to investigate the normality of the multivariate distribution, but at the 1% threshold the distribution does not appear Gaussian (Henze & Zirkler, 1990). We followed the same protocol on class-conditional densities and obtained similar results, the distributions are not Gaussian.

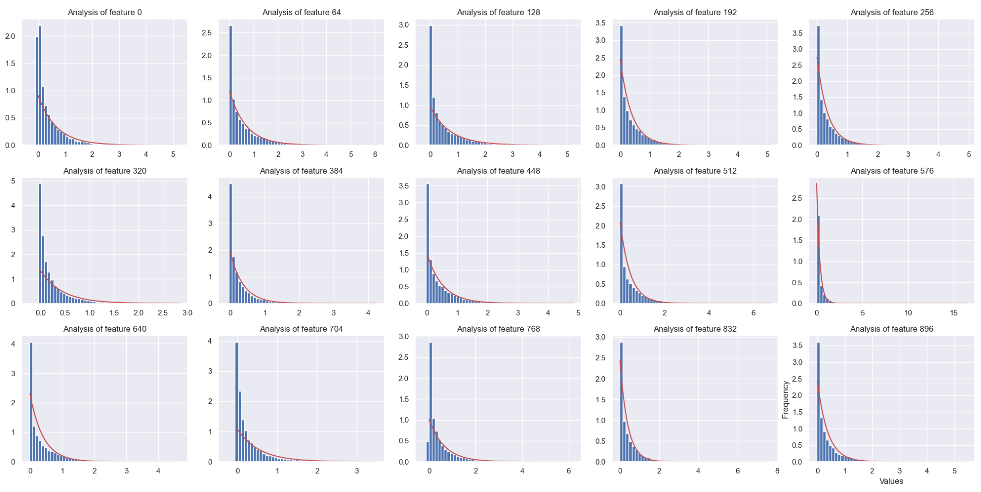

Figure 1: Training labels distribution

We then visualized certain features graphically. As we can see in Figure 1, the distributions are characterized by a large number of values close to zero and a fairly long tail on the right. We have tried to fit exponential laws (in red on the graph), which more or less approximate the densities of data. To realign the data and deal with the rightward skew, we will try a logarithmic or square-root transformation. The class-conditional distributions show similar distributions.

Training labels distribution & Classes imbalance

After analyzing the training data, we studied the training labels and noticed a disparity in the number of occurrences for each class. Like seen in Figure 2, there is a serious imbalance in the classes distributions, with the maximum and minimum occurrences varying between 398 and 2 respectively.

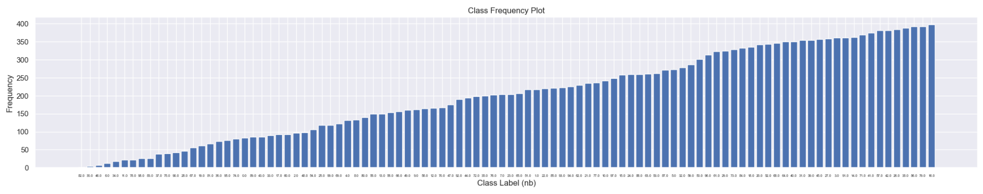

Figure 2: Training labels distribution

This will of course be a problem with our model, since it adds a positive bias for frequent classes and negatives bias for rare classes. The presence of imbalance in the class distribution could lead to poor generalization capabilities from the model. We decided to address this problem by trying different approaches of data augmentation.

Data augmentation - Over sampling using external library

We used the library `imbalanced-learn` (Lemaître et al., 2017) that performs simple randomized over-sampling with a "not majority" strategy, meaning that all classes will be augmented except the majority class. With this approach, we went from a dataset with 25515 samples to 39800 (398×100) where all classes are perfectly balanced.

Comparison of training and test distributions

We also compared the distributions of the average and the variance of each features for the training and test data to ensure that there were no major disparities between the two. Indeed, if there are significant differences between the two, our model will not be able to generalize correctly.

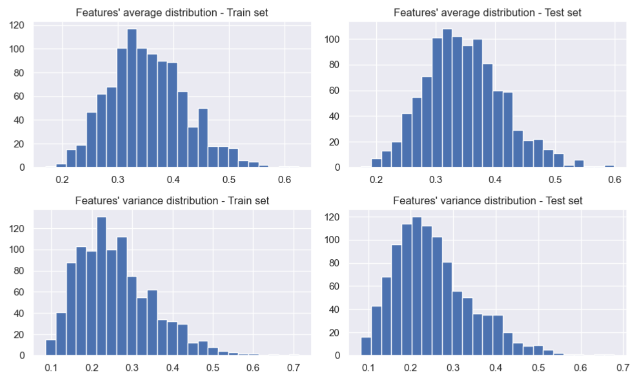

Figure 3: Histograms showing distribution of the average and variance of every feature

As we can see on Figure 3, there does not appear to be any significant differences between the two datasets concerning this point, which is good news.

Scaling of data

Before feeding our data into different algorithms, our preprocessing involved scaling the data. This is an important step because it standardizes every feature, ensuring that one feature isn't considered more important simply because it has a larger value or a different distribution. To achieve this, we employed two scaling techniques: standard scaling and min-max scaling. Standard scaling ensures that each feature's average is 0, and the standard deviation is 1, impacting its distribution. However, this might not be ideal in our case since our features do not have a Gaussian distribution. On the other hand, min-max scaling scales the data linearly between 0 and 1, preserving the initial distribution and may be better suited to our data. Following our analysis of each feature's distribution, we used two techniques to reduce the skewing of the data: log-transform and square root transform. All these data processing techniques were considered as hyperparameters of a given model during our final grid search.

Principal components analysis

We also thought of reducing the size of the data and extract important features using principal component analysis. This technique allows us to retain only those dimensions that capture the most variability (variance) in the data. We plan to try several variance thresholds (75%, 90%, 95%, 97.5%) and compare the results.

2 Algorithms

The current section gives an overview of the algorithms we have used to design our classification model. First of all, based on the observations made in the previous section, we can discard several options.

DISMISSED APPROACHES

We decided not to experiment with methods that assume a Gaussian distribution by class, such as `QDA`, `LDA` or `GNB`. Indeed, their inductive biases assume that class-conditional densities are Gaussian (Chandar, 2023), and this assumption is not valid on our data. We discard those models as the underlying theory does not align.

Due to our data existing in a 1024-dimensional hyperspace, we opted not to explore `KNN` solutions. This decision aligns with the curse of dimensionality theory as seen in class (Chandar, 2023).

EXPLORED MODELS

Based on our theoretical knowledge and data analysis, we decided to try out 4 classification algorithms.

LOGISTIC REGRESSION

Logistic regression models adhere to the discriminative paradigm. As demonstrated in class, these models enhance predictive performance over generative models, particularly when assumptions about the Gaussian distribution are poor, as is the case for us (Chandar, 2023). This is due to the fact that this model does not assume any distribution of the data as other models would. Additionally, in scenarios with numerous features, which once again is our case, logistic regression involves fewer parameters to train, contributing to its advantages. For multi-class problems, logistic regression fits $N - 1$ one-vs-rest classifiers where N is the number of classes and when predicting, the one that has the highest confidence is used to identify the class. It is important to note that a potential issue with this model is that it assumes that the classes are linearly separable, which might not hold in such a high dimensional space (Ching, 2020).

SUPPORT VECTOR MACHINES (SVM)

Given the effectiveness of **Support Vector Machines (SVMs)** in high-dimensional spaces, we explored this model, despite its general recommendation against large datasets due to extended training times. During training, SVM determines the decision boundaries between classes, but it also tries to maximize the margin between classes which might lead to better generalization on the test set when compared to logistic regression. Furthermore, this makes SVM less prone to overfitting, especially in high dimensions since the emphasis on the margin tends to simplify the decision boundary (Zhang, 2019).

NEURAL NETWORKS

In recent years, **neural networks** have been very useful for tasks such as image classification because they have the ability to modify the data's representation and extract complex patterns in the data. For this reason, we decided to explore this model further while keeping in mind the complexity associated with its different possible architectures (e.g. number of layers, number of nodes per layer, activation functions, etc.) making it hard to optimize.

RANDOM FORESTS

We decided to explore **random forests**, as they make no assumptions about data distribution. In addition, this algorithm is one of the ensemble methods, which have the advantage of reducing error and decreasing variance. Another key point is that they are highly efficient, which means we can train them quickly and experiment more easily (Louppe, 2015). Nevertheless, they can lead to overfitting, so we will have to be careful when choosing hyperparameters.

3 METHODOLOGY

With the aim of model selection, we set up a cross-validation process. We separated our training set into a training set and a validation set according to a standard 80%-20% split, ensuring similar labels distributions. Although we initially implemented k-folds validation, results mirrored those from a single validation set, leading us to adopt the simpler and more efficient approach.

INITIAL BASELINES

In order to have a baseline and be able to make comparisons before seeking to optimize our models, we trained models with each of the algorithms mentioned above using the default parameters in `sklearn`.

To establish a baseline for the neural network, we trained a simple three layer network with 1024 inputs, 512 nodes on the second layer and 100 outputs using ReLU as an activation function. However, at this point, we excluded Neural Networks (`NN`) from further consideration due to their complexity and our limited expertise in theory and experimentation. Fine-tuning the `NN` proved challenging, requiring intuition we lacked, leading us to the decision to remove it from the ongoing model selection process.

Model Tuning & Grid Searches

After establishing baseline benchmarks for comparison, our focus shifted towards enhancing the performance of our models. This optimization effort targeted two key aspects: refining preprocessing techniques and fine-tuning hyperparameters. We systematically explored various preprocessing methods and adjusted hyperparameters across each model, seeking the optimal combination to elevate overall performance.

First of all, at the preprocessing level, we tried standardization or scaling methods (standard normalization and minmax scaling) alongside data transformations (logarithm, square root or no transformation). Our objective was to determine the optimal preprocessing approach for enhancing model performance. To achieve this, we systematically applied each of these modifications to baseline models, retaining only those that gave the best performance. Following multiple iterations, we identified that the most effective data preprocessing involved a logarithmic transformation followed by scaling the data to a range between 0 and 1 using min-max scaling. Ultimately, we chose not to go forward with PCA dimensionality reduction, as its implementation consistently resulted in a deterioration of our model's performance.

We then analyzed the hyperparameters of the three algorithms (Logistic regression, Support Vectors Machine and Random Forest) that gave the best performance in order to optimize them. We referred to our courses (Chandar, 2023) and the `sklearn` documentation (Pedregosa et al., 2011) to establish the set of hyperparameters that could have the greatest impact for each algorithm. The selected hyperparameters will be shown in the next section. To evaluate the different combinations, we performed a grid search with 3-block cross-validation.

Ensemble models : Bagging & Stacking methods

At that point, we possessed models that perform well, but our goal was to enhance their performance. A notable challenge we faced was the disparity between the F1 scores of the validation and test sets (we will discuss this in further details later). To mitigate that variance and improve generalization, we implemented bagging. We therefore made a bagging model with our best model at this point on the test set, logistic regression.

Our next step involves stacking our best models to capitalize on the strengths of each, aiming to further enhance overall performance. This strategy enables us to leverage the diverse strengths of individual models and capture different elements more effectively. Based on our analysis of past models and their outcomes, we opted to assemble a stack comprising three distinct models: tuned logistic regression, tuned support vector classifier (SVC), and tuned random forest. This selection aimed to harness the diverse strengths and weaknesses inherent in each model.

Notable unsuccessful models

We will here discuss alternative models that we experimented with, but ultimately proved unsuccessful.

To perform data augmentation, we tried a manual approach for greater control and fine-tuning instead of using the library stated above. Employing a scientific methodology, we systematically isolated changes, focusing only on those that demonstrated improvement on our models. A minimum instances-per-class hyperparameter was established, and for classes falling below this threshold, data augmentation was applied to meet the minimum requirement. Our approach was based on the premise that reshaping each feature extracted from the initial image into a 32 x 32 format would preserve its meaning. We therefore tried out various symmetries, rotations and noise additions, retaining only those results that were convincing in comparison with a control model. Despite the theoretical merits of this method, it failed to improve performance, prompting us to explore other approaches.

We tried to do clustering by grouping observations together to create clusters. For instance, images of dogs or wolves share similarities and could be categorized into a common cluster. This "divide and conquer" strategy aimed to streamline the task by reducing the number of labels to predict and increasing the data per category/cluster. Regrettably, this approach did not yield a performance improvement. We attribute this to the insufficiently high similarity among images within a common cluster, resulting in more complexity rather than simplifying model choices.

We also aimed to leverage our knowledge of image-based observations to improve our performances but faced challenges reconstructing input images in a human-readable format. Indeed, the lack of insight into preprocessing steps made retroactively pinpointing lost data impractical. We also quickly experimented with a pre-trained image classification model, and its performance was remarkably poor ($< 5\%$) due to the competition data's non-conformity to a typical image structure used for the training of these models.

As we saw during the course, convolutional neural networks (CNN) are great models for image classification since they allow us to extract the images' features (Chandar, 2023). Knowing this and knowing that our features were extracted from images, we reshaped them into a 32×32 square image and fed them into a custom CNN. This did not give the level of performance we expected and our lack of knowledge on the subject prevented us from further exploring this option.

4 Results

Note: The subsequent results pertain to the weighted F1-score evaluated on both our validation set and the test set, which constitutes 10% of the data accessible to us through Kaggle.

We were impressed with the results of the default models without any data preprocessing where all models achieved over 0.90 on the validation set, although this performance was not reflected on the test set. These results are available in Table 1. At this point, the best model is logistic regression, which holds on the test set.

Table 1: Simple models performance on different datasets

Simple Models	**Validation F1-Score**	**Test F1-Score**
LR	0.94869	0.79543
SVC	0.93509	0.78653
RF	0.92550	0.68477
NN	0.91834	0.77616

For logistic regression, algorithm-specific hyperparameters had no impact and the best model is the default one. For the SVM model, we kept a Radial Basis Function (RBF) kernel in order to express non linear boundary decisions but we increased the regularization parameter `C` from 1.0 to 10.0. The regularization strength (L2 penalty) is inversely proportional to this parameter, we therefore reduced the regularization and the bias at the cost of a greater variance.

For random forest, we changed the criterion from the Gini index to the entropy and also restricted the number of features analyzed to make the splits by passing from the `sqrt` criterion to the `log2` criterion for the `max_features` parameter. Finally, we significantly increased the number of trees from 100 to 250.

The performance metrics of the fine-tuned models are outlined in Table 2. Notably, there is a significant enhancement in scores observed for the validation set, providing encouraging evidence that our modifications indeed contribute to the improvement of the model. These positive enhancements are also seen in the test set performance, although the degree of improvement is smaller.

Table 2: Tuned models performance on different datasets

Tuned Models	**Chosen Parameters**	**Validation F1-Score**	**Test F1-Score**
LR	max-iter: 100, penalty: l2, solver: lbfgs	0.98006	0.80538
SVC	C: 10.0, gamma: scale, kernel: rbf	0.98146	0.80283
RF	criterion: entropy, max-features: log2, n-estimators: 250	0.98108	0.78842

We tried different numbers of classifiers and the set that gave us the best results was to combine 25 logistic regressions with an F1-score of 0.97520 on the validation set and 0.80494 on the test set. However, our attempt at implementing stacking, disappointingly, did not enhance performance; instead, it led to a decrease in the model's effectiveness. These results are shown in Table 3.

Table 3: Ensemble methods models performance on different datasets

Ensemble Methods	**Validation F1-Score**	**Kaggle F1-Score**
Bagging	0.97477	0.80494
Stacking	0.92274	0.77168

Analysis

If we delve deeper into the numbers, three interesting findings emerge.

Firstly, a noticeable disparity exists between the validation and test sets, with discrepancies in scores reaching up to almost 20% in certain cases. Unfortunately, replicating the performance achieved on the validation set proved challenging on the test set. Nevertheless, we take pride in attaining an impressive score of 0.98146 on the validation set through data preprocessing and a finely-tuned Support Vector Classifier (SVC) model.

Upon further analysis of these results, it becomes apparent that our best-performing model on the validation set slightly differs from the one on the test set. Specifically, our tuned Support Vector Classifier (SVC) model exhibits a superior score on the validation set, while the tuned Logistic Regression (LR) model outperforms on the test set. This discrepancy may be attributed to the fact that the test F1-Score is calculated on only 10% of the data. Indeed, the tuned logistic regression model, tuned SVC model, and the bagging model closely align on the test set, likely due to the limited data upon which this metric is calculated.

Lastly, we observe similarities in scores among our three tuned models, with differences within 0.0015 of each other. However, the tuned random forest's performance on the test set is evidently inferior to that of the other models.

Overall, some results are unexpected. We are surprised by the success of logistic regression, suggesting that the data may be linearly separable to some extent in the 1024 dimensions. This observation is unexpected. Conversely, we are also shocked by the underwhelming results of the stacking method. We initially anticipated that this technique would enhance our performance, but contrary to our expectations, it did not. It is conceivable that stacking the Random Forest (RF) model contributed to these disappointing results, given that the tuned random forest classifier did not perform particularly well on the test set.

Note: Our best official submission on Kaggle is one issued from the predictions of the Bagging Model rather than the tuned logistic regression. While revisiting our models and predictions for the report, we introduced a seed during the rerun, leading to alterations in the outcomes of some models. Consequently, some of the results of the models exposed here are derived from late submissions and are not official.

5 Discussion

The methodology employed to achieve our results has several advantages. Firstly, we conducted comprehensive research on various algorithms applicable to the problem at hand. Utilizing the scientific method, we identified primary issues, formulated hypotheses, and systematically manipulated variables. Through iterative improvements, we aimed to derive the optimal model. While this approach yielded favourable results, it necessitated testing numerous algorithms and techniques. Although effective, this broad exploration prevented us from delving deeply into any specific algorithm. A more focused strategy, concentrating on a particular algorithm based on strong insights, might have led to an even better model. If we had a specific intuition about a model that we believed would perform exceptionally well, a more in-depth exploration of that particular model could have been a valuable strategy.

One of the main challenges we faced was the significant difference between performance on the validation set and the test set, as said earlier. Indeed, an improvement on the validation set did not always translate into an improvement on our test score. Hence, another possible improvement to our work would have been to spend more time understanding and analyzing the causes of this phenomenon and try to mitigate it.

In view of these significant differences, we think that it would have been interesting to try methods with a better generalization property, sacrificing performance on the validation set to improve performance on the test set. For example, algorithms like SVM that seek to maximize the margin between classes might have been able to perform better with more research. Furthermore, had we been able to reconstruct the images properly, a CNN could have been a great model as it is widely used for image classification.

6 Statement of contributions

We organized an initial meeting to discover the project together and establish a plan. We started by experimenting together, then when we all had a clear vision of the problem, we divided up the work.

After the first week's experimentation, we defined a working methodology to ensure common points of comparison. Each person was responsible for coding certain portions of the project and explaining their analyses in the report. We met regularly (2-3 times a week) to review the code, report together and share our progress. Therefore, we consider that every member of the team contributed equally to the success of this project.

We hereby state that all the work presented in this report is that of the authors.

REFERENCES

S. Chandar. Inf8245e-machine learning, 2023. URL `https://chandar-lab.github.io/INF8245E/schedule.html`.

C. Ching. Logistic regression theory for practitioners, 2020. URL `https://towardsdatascience.com/the-data-scientists-field-guide-to-logistic-regression-part-1-intuition-97084b11b`

N. Henze and B. Zirkler. A class of invariant consistent tests for multivariate normality. *Communications in Statistics - Theory and Methods*, 19(10):3595–3617, 1990. doi: 10.1080/03610929008830400. URL `https://doi.org/10.1080/03610929008830400`.

G. Lemaître, F. Nogueira, and C. K. Aridas. Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning. *Journal of Machine Learning Research*, 18(17):1–5, 2017. URL `http://jmlr.org/papers/v18/16-365.html`.

G. Louppe. Understanding random forests: From theory to practice, 2015.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Z. Zhang. Support vector machine explained, 2019. URL `https://towardsdatascience.com/support-vector-machine-explained-8bfef2f17e71`.