Linear and Logistic Regression

1st Sasha Denouvilliez-Pech
McGill University
sasha.denouvilliez-pech@mail.mcgill.ca

2nd Shidan Javaheri
McGill University
shidan.javaheri@mail.mcgill.ca

3rd Taz Scott-Talib
McGill University
taz.scott-talib@mail.mcgill.ca

***Abstract*—This project was centred around the introduction to common Machine Learning models. Its principal task was the implementation of two models: Linear Regression, and Logistic Regression with Gradient Descent. Our team implemented these models in Python using several core machine learning techniques: analytical linear regression, fully batched gradient descent, mini-batch stochastic gradient descent, Lasso regression for L1 regularization, and Adaptive Moment Estimation (ADAM). We performed linear regression on an eight-feature quantitative energy dataset, using our model to predict two output variables, and logistic regression on a six-feature categorical bankruptcy dataset, using this model to classify test instances with the correct label. We also ran several experiments on each implementation, allowing us to demonstrate that all our models were highly accurate, with mean squared error values less than 10 for all linear regression models, and an accuracy of 94 - 100% for the logistic regression models. Further analysis helped us determine the most salient features of each collection, as well as the most appropriate hyper-parameters for the various algorithms. In this paper, we will study the performance of each approach on two fixed datasets and analyze their strengths and potential shortcomings.**

***Index Terms*—Classification, Gradient Descent, Linear Regression, Logistic Regression, Machine Learning, Prediction, Regularization**

I. Introduction

Linear and Logistic Regression are amongst the most used models in the Machine Learning industry [1]. This project aimed to implement models belonging to these two classes in various ways, and to evaluate each of their performances on two datasets. The numerical data for linear regression, an energy dataset [2], consisted of 768 instances with 8 features and 2 outputs. The categorical data for logistic regression, a bankruptcy dataset [3], consisted of 249 instances of 7 features linked to 1 output. The solutions for the models were computed with multiple techniques, including closed-form solution for linear regression, fully batched and mini-batch stochastic gradient descent, Lasso regression, and Adaptive Moment Estimation. Performances of the models were evaluated using mean squared error for linear regression and cross-entropy loss for logistic regression. All linear regression models had a mean squared error of less than 10, and lositic regression models had an accuracy of 94 - 100%. Multiple findings gave insights into each dataset's most important features, as well as into the effect that training size, mini-batch size, and hyper-parameter values have on the cost of each model when it is run on unseen data.

II. Datasets and Data Pre-processing

A. Quantitative Energy Efficiency Dataset

The quantitative energy dataset [2] was used to train and then evaluate the performance of our linear regression models. Its 768 instances consist of 8 features, labelled $X_1 - X_8$ all of which are attributes of the building's dimensions and its glazing distribution. The dataset also has two outputs - Y_1, the heating rate of the building, and Y_2, its cooling rate. Fortunately, all instances contained no null values and were all floats or integers.

To understand the data, basic statistics were computed for each feature, such as the mean and standard deviation. The high range of means (0.23 (X_2) - 671.71 (X_7)) and standard deviations (0.11 (X_1) - 88.09 (X_2)) indicated the need to normalize the data. In addition, a correlation matrix between all features was used to indicate which features to select for our model.

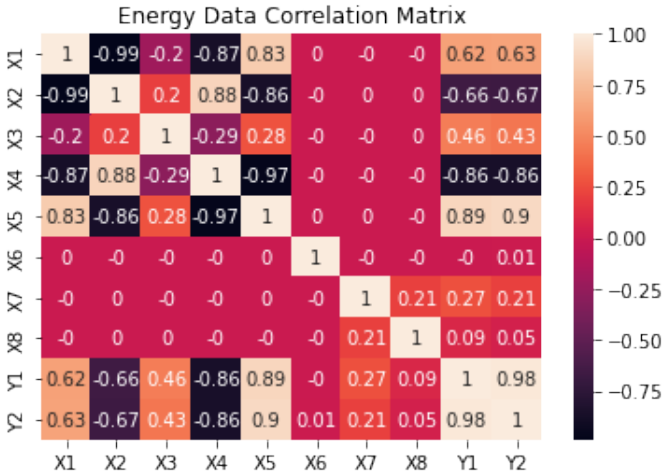

Fig. 1. **Correlation Matrix for Quantitative Energy Dataset**

With results from the covariance matrix, we can eliminate the least important of features that are highly correlated to prevent overfitting [4]. In addition, we can remove features that have no correlation with the output, since we are using linear models and according to Occam's Razor Bias, a

simpler model generalizes better [5]. As shown in Fig. 1, we can see that the pairs (X_1, X_2) and (X_4, X_5) are almost perfectly correlated, both with a correlation above 0.95, indicating that one of them an be eliminated. We chose to eliminate feature X_4, since it had a lower Z score than X_5, indicating that X_5 is more important [6]. Furthermore, when all normalized features were used in a Lasso regression model, X_5 had the highest weight for all values of λ , and along with X_4 was the only non-zero feature when $\lambda = 3$. This also indicates that X_5 is the most important feature in the dataset, a result we return to in Section III.

We can also identify features with no correlation to the output, and eliminate them since we are using linear models, and simpler models generalize better [5]. We eliminated both X_6 and X_8, which have correlations of less than or equal to 0.09 with both outputs. We decided to train and test our final models only using features X_1, X_3, X_5 and X_7 as a result. Although Occam's Razor Bias tells us that simpler models with generalize to unseen data better, we have to be aware of the ethical challenge a simple model poses. We may be oversimplifying a more complex situation and should keep that in mind when using this data to make decisions.

To further support these decisions, we compared performances of analytical linear regression on normalized data with and without feature selection, both times with an 80/20 test/train split. The model performed relatively well on normalized data, with average training and test costs of 11.07 and 11.38 respectively. However, these results were only after filtering out outliers and were vastly different depending on the random seed used. In addition, model would sometimes throw a singular matrix error, because the design matrix was not invertible, due to its columns being non-linearly independent. However, when using the normalized data with feature selection, average training and test costs are always below 10, with no outliers and lower standard deviation, and there were never any singular matrix errors. As a result, we used the normalized data with feature selection for the analysis of all of our models.

B. Qualitative Bankruptcy Dataset

The qualitative bankruptcy dataset [3] was used to train and evaluate the performance of our logistic regression models. Its 250 instances consist of 6 features, representing characteristic/risk factors related to bankruptcy. Each of these is represented by one of three values: *P* for positive, *A* for average and *N* for negative.

Since these features are categorical, we had to encode them into quantitative values in order to input them into a numerical model. We decided to use one-hot encoding via the ski-kit learn library [8]. This was a decision based on our data: since there were only six features, each with three possible values, it would not result in an input matrix that was too large. In addition, though a logical order exists between *P*, *A* and *N*, and thus label encoding is a viable option [7], we have no way of knowing where the average lies between the positive and negative values, and it might not be accurate to place it in the mathematical middle. With one-hot encoding, we ensured that each of the feature values was treated independently [7]. For similar reasons, we opted to not perform feature selection on the qualitative data.

The output feature, bankruptcy, only had two possible values (Bankruptcy or Non-Bankruptcy). This was very useful, as it meant we only required a single column of one-hot-encoded data. We chose the *B* (Bankruptcy) column, as it was more straightforward to interpret a value of 1 as Bankrupt and a value of 0 as Non-Bankrupt.

III. Results

We can analyse the results of all of our linear and logistic regression models. This will be done primarily using a train/test split of 80/20, where it is not a variable under analysis. Final costs are reported with the best hyper-parameters after observing how model performance changes for different values. Performance was measured with mean squared error for linear regression, and cross-entropy loss for logistic regression. Wherever possible, results will be reported as an average of 15 iterations of training and testing the model.

A. Performance Report

First and foremost, we can report the costs that each of these models give when they are tested on unseen data. The results in Table I demonstrate that all models have very similar and accurate performance, with low mean squared error $(9.69 - 9.78)$ and variance in their performance on different trials. These results used a learning rate of 0.1 and batch size of 32 for mini-batch gradient descent, and parameters of $\beta_1 = 0.5, \beta_2 = 0.5$ and learning rate of 0.1 for ADAM gradient descent. The learning rates were identified by plotting mean squared error vs. changing learning rate values (see Section III-E), and the values of β_1 and β_2 through trial and error. Results could have been improved with proper hyper-parameter tuning on a validation set, with cross validation.

TABLE I
Linear Regression Model Performance

Linear Reg. Models	**Cost (Mean Squared Error)**			
	Train. Avg.	***Train. STD***	***Test Avg.***	***Test STD***
Analytical	9.69	0.31	9.75	1.23
Fully Batched	9.71	0.30	9.78	1.27
Mini-Batch	9.69	0.31	9.75	1.23
ADAM	9.78	0.32	9.86	1.24

As seen in Table II, our logistic regression model using fully batched gradient descent showed an average cost of around 0.679 on the training data, with a fixed learning rate

of 0.1. The test data had a mean cost of 0.568, indicating that our model may be slightly underfitting since it performs slightly better on unknown data [5]. Given we are training with 200 total instances, there is sufficient data, but a larger set might yield a lower cross-entropy loss. Our results also showed that both the fully batched and mini-batch gradient descent models had very similar performances.

TABLE II
LOGISTIC REGRESSION PERFORMANCE

Logistic Reg. Models	**Cost Performances**			
	Train. Avg.	***Train. STD***	***Test Avg.***	***Test STD***
Fully Batched	0.679	0.005	0.568	0.020
Mini-Batch	0.677	0.005	0.567	0.027

B. Model Weights

As shown in Table III, all linear regression models compute very similar weights for each of the features we have selected. In addition, we can see that for both outputs, X_5, the overall height of the building, had the highest weight, further supporting our conclusion that it is the most important feature in our dataset [5].

TABLE III
LINEAR REGRESSION WEIGHTS

Linear Reg. Models	$\mathbf{w}_1$		$\mathbf{w}_3$		$\mathbf{w}_5$		$\mathbf{w}_7$		**Bias**	
	Y1	***Y2***	***Y1***	***Y2***	***Y1***	***Y2***	***Y1***	***Y2***	***Y1***	***Y2***
Analytical	-1.53	-2.09	1.52	0.80	9.81	10.02	2.72	1.97	22.29	24.57
Full Batch	-1.54	-2.09	1.48	0.76	9.78	10.03	2.70	1.95	22.25	24.56
Mini-Batch	-1.53	-2.09	1.52	0.80	9.81	10.02	2.72	1.97	22.29	24.57
ADAM	-1.42	-1.98	1.58	0.86	9.71	9.93	2.72	1.99	22.27	24.54

The logistic regression feature weights were extremely similar between the fully batched and mini-batch gradient descent algorithms, further confirming the performance similarities previously reported, so we have only displayed the former in Table IV. This table demonstrates that the Competitiveness feature, X_5, has the highest weights, and thus has the highest salience [5].

TABLE IV
LOGISTIC REGRESSION WEIGHTS

Fully Batched	**Negative**	**Average**	**Positive**
Industrial Risk	0.20	-0.84	-0.72
Management Risk	0.04	-0.71	-0.70
Financial Flexibility	1.88	-1.62	-1.63
Credibility	-1.69	-1.09	-1.97
Competitiveness	3.21	-1.70	-2.88
Operating Risk	-0.12	-0.18	-1.06
Bias	-1.37		

To graphically show the performance of our analytical linear regression model, we can plot the most important feature, overall building height, against the heating and cooling rates, and further simplify our model to be the equation $y = bias + w_5 \times X_5$. The results in Figure 2 show that the model works exactly as expected.

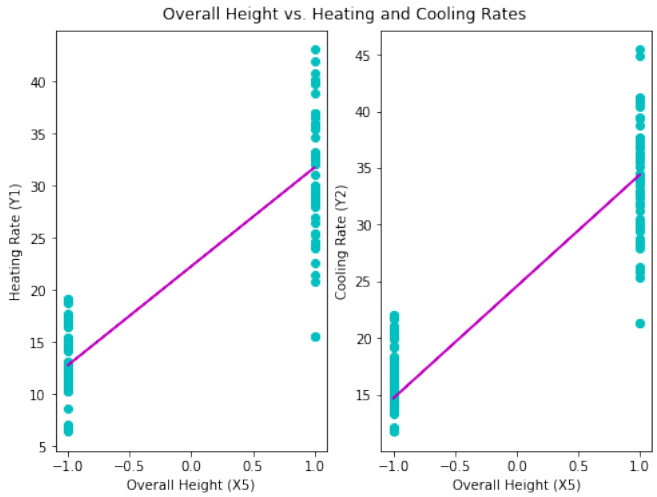

Fig. 2. **Overall Building Height vs. Heating and Cooling Rates**

Similarly, we can plot the most salient feature of the bankruptcy dataset, competitiveness (X_5), against the output probability. The graph in Figure 3 demonstrates our decision boundary correctly classifies all data points with negative competitiveness features as having a high probability of being bankrupt. This demonstrates a case where are model was 100% accurate. Over repeated trails, our model had an accuracy range of 94 - 100%. The confusion matrix demonstrating this can be found in our accompanying code.

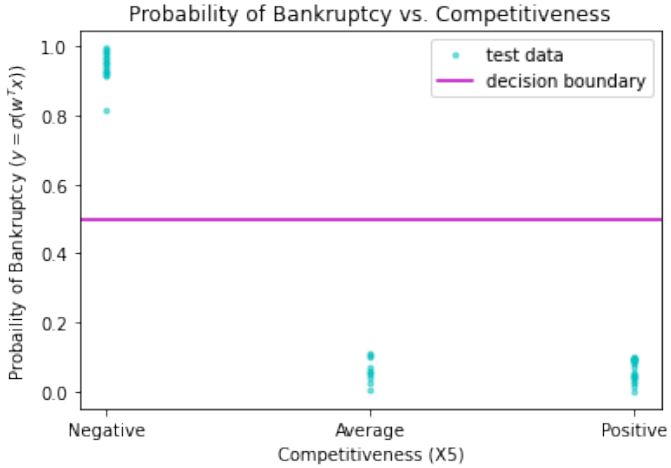

Fig. 3. **Probability of Bankruptcy vs. Competitiveness**

C. Growing subsets of the training data

Splitting the data for training and testing is an important metric when training models. Training the model must be sufficient to learn the distribution of the data correctly, but the purpose of the model is to generalize to unseen data.

We analysed the effect of training size on analytical linear regression and fully batched logistic regression.

For analytical linear regression, the cost as function of the training data size percentage is plotted for training and test costs in Figure 4 below. It shows that smaller training sizes increase the difference between the training and test costs, as expected. Furthermore, smaller training or testing sizes increase the variance of the costs associated with them since each subset is less representative of the whole dataset. Since there is not a major difference in costs based on training set size, this shows our model generalizes very well even when given only a small set of training data.

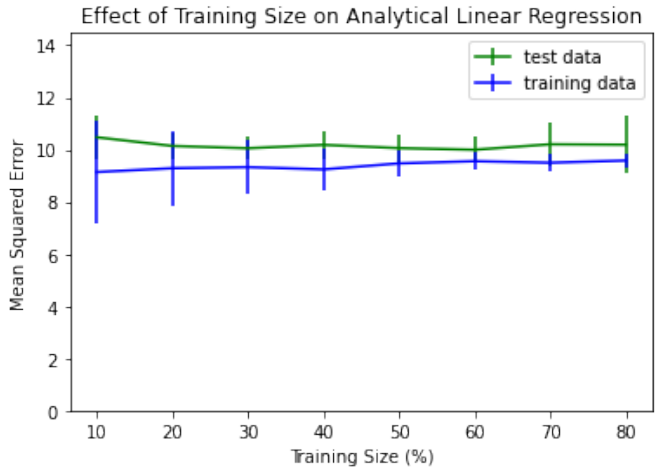

Fig. 4. **Effect of Increasing Training Size on Mean Squared Error for Analytic Linear Regression**

For fully batched logistic regression, both the test and training cross-entropy losses decreased with increases in the size of the training data, as show in Figure 5 below. This effect was more significant on the test data, as expected due to increased training instances and fewer test instances for which it could be wrong. However, each incorrect test instance has a greater impact on cost in a smaller dataset, which could explain the higher standard deviations on the test dataset curve as test size decreases.

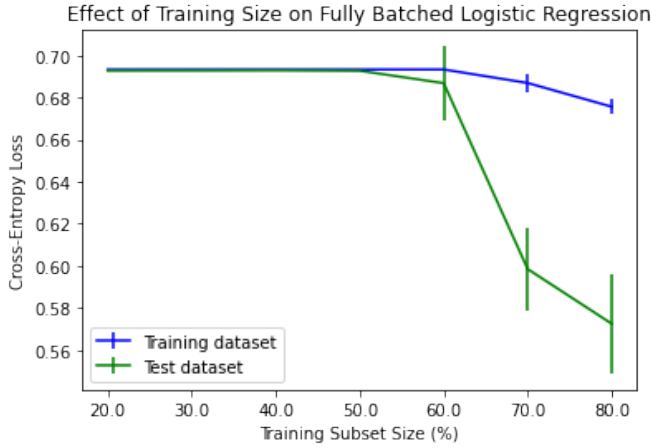

Fig. 5. **Effect of Increasing Training Size on Cross Entropy Loss for Logistic Regression**

D. Growing Mini-batch Sizes

The mini-batch size is the size of the subset of the dataset that is used to calculate the gradient. When the mean squared error is plotted against an increasing batch size, the performance stays more or less constant in both the linear and logistic regression models, despite a slight decrease in test costs initially for linear regression, and the test and training costs getting closer together. This shows that within the maximum number of iterations, both models are still able to converge to the minimum cost. However, as expected, the convergence speed in both models is proportional to batch size, with the number of epochs doubling as each batch size doubles as well. We realized that it was better to measure epochs than iterations, since with higher batch sizes, epochs are reached more frequently. This is demonstrated for logistic regression in Figure 6 below.

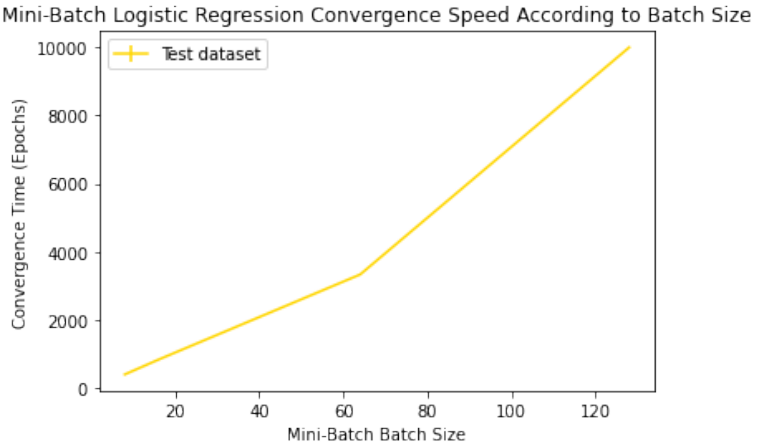

Fig. 6. **Effect of Increasing Batch Size on Convergence Time for Logistic Regression**

E. Different Learning Rates

Changing the learning rate as models are trained and tested has an interesting effect on their performance. For linear regression with gradient descent, we observed that learning rates within the range of 0.04 - 4 had very similar performance. The cost as a function of the learning rate has a vertical asymptote at 0 and diverges towards infinity at around 4.7, in line with the theory. Within this range, the learning rate would affect how fast the solution converges, but not what values it converges to. However, around 0, the initial weight values are not updated, and when the learning rate is too high, the weights diverge. From this, we could select our learning rate value to be 0.1 confidently. For ADAM gradient descent, we performed similar analysis. There were higher fluctuations in cost when the learning rate was greater than 1, but the model was much more resistant to its cost completely diverging with very high learning rates, despite large fluctuations occurring. This demonstrates another advantage using momentum and an

adaptive learning rate can have, and the results are shown in Figure 7 below. In the future we could perform proper hyper-parameter tuning to select the best value to use for our model, but we safely chose a value of 0.1.

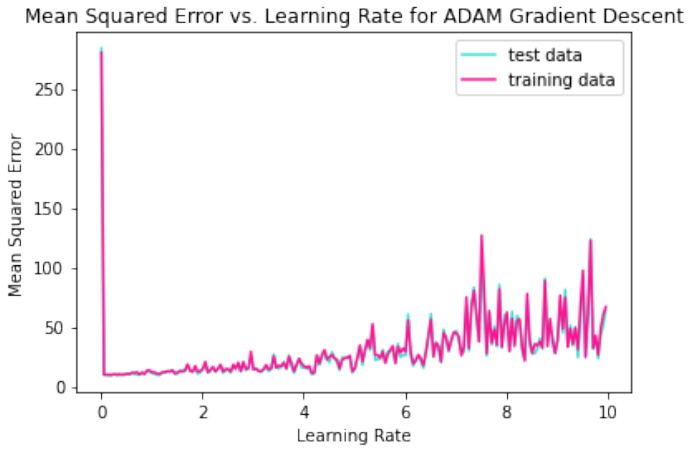

Fig. 7. **Effect of Increasing Learning Rate on ADAM Gradient Descent Performance**

For logistic regression the model also converges over a wide spread of different learning rates, with the expected asymptote at 0 and only a slight performance increase as we move towards 0.1. Our hypothesis was that this was due to our maximum number of iterations set at 10,000, and sure enough, the curve was much more noticeable with a limit of 100.

F. Analytical vs. Mini-Batch Linear Regression

Based on the results presented in Table I, we have demonstrated that Analytical and Mini-Batch solutions are practically equivalent in terms of their costs when they are run on normalized data with feature selection. The analytical linear regression is a closed-form solution whereas the mini-batch linear regression is a variation of the gradient descent, where the gradient is computed from subsets of fixed size. Since the linear regression is a linear combination of its inputs, the function is convex and the gradient descent should converge to the global minimum. Since the global minimum is the point that minimize the cost function, both Analytical and Mini-batch solutions should find it.

However, the analytical solution is computed directly with a matrix multiplication, and thus is much faster that the mini-batch model since our dataset is so small. Over an average of 15 runs, with a batch size of 32, the analytical model can be trained and predict outputs at least two orders of magnitude faster than the mini-batch gradient descent model. However, when taking into account the normalized data without feature selection, the Analytical solution proves to be less practical and unstable. It is very sensitive to overfitting, leading to high cost variance. Furthermore, if some features are linearly dependent or correlated, the matrix is singular and the solution does not exist. Gradient descent was not affected by these problems, and would be faster on a very large dataset, which explains why it is a more superior and robust model.

IV. Discussion & Conclusion

This first Mini Project introduced two of the most common models in ML, linear and logistic regression. Several models belonging to these two classes were trained and tested on two data sets, the quantitative energy dataset and qualitative bankruptcy dataset. Extensive data analysis and processing were completed, especially for linear regression. The data was plotted, statistics were computed, then it was normalized, and feature selection was performed with a covariance matrix and lasso regression. The models were trained using multiple solutions. For linear regression, the analytical solution and multiple variants of the gradient descent were computed. The most notable solution was the implementation of the Adaptive Moment Estimation (ADAM) algorithm. For logistic regression, fully batched and mini-batch versions were implemented, both using one-hot-encoding.

Through the performance evaluation of the models using the mean squared error for the linear regression and cross-entropy loss for logistic regression, all models were very accurate. All linear regression models had nearly identical costs, all lower than 10 with low standard deviations. The logistic regression model had an accuracy range of $94 - 100\%$ with both fully batched and mini-batch gradient descent. In addition, a few theoretical concepts were confirmed. In general, growing the training data size seems to reduce the difference between training and test costs, as well as the overall test cost. In general, smaller subsets relative to the data size tend to lead to high cost variance over multiple runs. For mini-batch gradient descent, the completion time is proportional to the batch size while the performance seems to stay constant. In gradient descent algorithms, the learning rate seems to be required to be in a range, within which performance is similar. However, above a certain value the model does not converge, although the ADAM algorithm is more resistant to this behavior.

Moving forward, our team would like to further investigate the theory behind feature selection. Choosing the right set of features is a major factor of the performance of the model, and improving this process would make future models stronger. Furthermore, it would be interesting to leverage proper hyper-parameter tuning using validation sets cross-validation to improve performance, particularly of the ADAM gradient descent algorithm, which has many hyper-parameters.

A. Statement of Contributions

All group members contributed equally to the project and to the report, and were familiar with all parts of code. Shidan and Sasha focused on all the Linear Regression models, and Taz focused on the Logisitc Regression models.

REFERENCES

[1] S. Ray, "A quick review of machine learning algorithms," in 2019 International conference on machine learning, big data, cloud and parallel computing (COMITCon), 2019: IEEE, pp. 35-39.

[2] "Energy efficiency Data Set," *UCI Machine Learning Repository*. [Online]. Available: https://archive.ics.uci.edu/ml/datasets/Energy+efficiency. [Accessed: 10-Feb-2023].

[3] "Qualitative_bankruptcy data set," *UCI Machine Learning Repository*. [Online]. Available: https://archive.ics.uci.edu/ml/datasets/Qualitative_Bankruptcy. [Accessed: 10-Feb-2023]

[4] M. A. Hall, "Correlation-based feature selection for machine learning," The University of Waikato, 1999.

[5] R. Rabani, "Applied Machine Learning," in McGill COmputer Science: COMP 551, 09-FEB-2023.

[6] R. Tang and X. Zhang, "CART decision tree combined with Boruta feature selection for medical data classification," in 2020 5th IEEE International Conference on Big Data Analytics (ICBDA), 2020: IEEE, pp. 80-84.

[7] Stephen Allwright, "One hot encoding vs label encoding," *Stephen Allwright*, 12-Jun-2022. [Online]. Available: https://stephenallwright.com/one-hot-encoding-vs-label-encoding/. [Accessed: 09-Feb-2023].

[8] "Sklearn.linear_model.logisticregression," *scikit*. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html. [Accessed: 09-Feb-2023].