Capability analysis of logistic regression in classification of students passing the Ontario secondary school literacy test in first attempt

Clayton Halim

Abstract Word Count: 152

Essay Word Count: 3643

Abstract

This paper analyses the effectiveness of logistic regression in a school dataset. A learning model was trained to determine whether or not a student will pass a standardized literacy test in Ontario on their first try. The dataset was formatted into a matrix with features engineered to clearly model a student's capability on tests. Different evaluation metrics such as precision, recall, and f-score were utilized and discussed on the learning model's performance. The regularization parameter lambda (λ), and prediction threshold were then varied to see the optimal value of both to achieve predictions with low bias and variance. The learning model achieved a peak f-score of 80% where the optimal value of lambda was 0 and a threshold of 50%. While oversampling aided the training on an imbalanced dataset, a larger dataset including data from schools with different literacy test performances would effectively allow the learning model to create more accurate predictions.

Contents

1 Introduction

The Ontario Secondary School Literary Test (OSSLT) is a province wide standardized test required to be passed for a student to graduate from high-school. The literary performance of students over from 2011 – 2015 has seemed to plateau at an 83% pass rate [1]. The EQAO, the Agency responsible, holds data on the students who have written their tests and utilizes it develop methods to increase performance. Machine learning has gained immense popularity over large companies such as Google, Microsoft, and Amazon, which provides the ability for a computer to extract knowledge from data without having to be explicitly programmed. Despite the fact that these companies have billions of training samples for their learning models [2], this paper looks at the capability for a learning algorithm with a small set of features and training sets from data on a school's own students to accurately classify whether or not they will pass the OSSLT on their first attempt.

Machine learning can be split into a variety of different sub-tasks, however the one used in classification is known as supervised learning. The general idea of this task is to have inputs (features), and labeled outputs used in the algorithm to "learn" how to identify a sample set known as training examples, given the same features. Some commonly used learning models used in supervised learning are neural networks and decision trees, but the one used for the classification in this paper is logistic regression. The algorithm is a binary classifier, indicating in this case, a probabilistic model to a pass or fail in the first try of writing the OSSLT. Parameters inputted into the algorithm are optimized using gradient descent to classify with the most minimal cost possible. An evaluation of classification accuracy is done from before the features were engineered to accommodate the unideal training set and after.

2 Student Dataset

2.1 Dataset Acquisition

The data stored by the school is separated into three files, containing a variety of information including birthdays and averages of students. The three data files were then processed, stripping away repeated data, leaving students ineligible for the OSSLT, and stitching the three files into one feature set. An example of what a feature might be is a person's age. This data is stored into a matrix *X*, with a size $m \times (n + 1)$, where *m* is equal to the number of training examples, and *n* is equal to the number of features. The plus 1 takes the bias term into consideration. The bias term simply serves the same purpose an intercept parameter would in the equation of a line, essentially to ensure that the estimator fits the data properly. The term x_1 will always take on the value of 1 because the bias term is only based on its parameter θ_1. Thus the parameter vector θ, will have a size of $(n + 1) \times 1$. These parameters aid in understanding the importance of each feature, allowing the learning algorithm to make decisions. The classification of the training examples is stored in a vector y which, within the vector, can only take on 2 values, where 0 represents a fail, 1 representing a pass. Note that the superscript in x (and y in Equation 3) is not an exponent, but rather an indicator of the ith (current) training example.

Equation 1. The training set matrix

$$X = \begin{bmatrix} x_1^{(1)} & \cdots & x_{n+1}^{(1)} \\ \vdots & \ddots & \vdots \\ x_1^{(m)} & \dots & x_{n+1}^{(m)} \end{bmatrix}$$

Equation 2. The parameters of features vector

$$\theta = \begin{bmatrix} \theta_1 \\ \vdots \\ \theta_{n+1} \end{bmatrix}$$

Equation 3. The classification of training set vector

$$y = \begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(m)} \end{bmatrix}$$

2.2 Dataset Imbalance and Evaluation Metrics

An imbalance in a dataset occurs when one or more classes in a dataset are underrepresented. In this case, only 78 out of the 948 training examples represent students who did not pass the OSSLT on their first attempt. With only about 8% of the dataset making up the minority class, the learning algorithm (later demonstrated in section 4.1) will have problems with classification. At first glance, the predicting accuracy of the algorithm will seem almost perfect achieving levels of around 98% accuracy, however the statistic is misleading. If the learning algorithm were designed to only "predict" the majority class of this training set, it would already achieve 92% accuracy. However since the objective is to identify those who are at risk of not passing the OSSLT at first attempt, that sort of algorithm would be particularly useless. To give better insight on the performance of the algorithm, a confusion matrix shown in Figure 1 is used.

Figure 1. Confusion Matrix

	Predicted Positive	**Predicted Negative**
Actual Positive	True Positive *(TP)*	False Negative *(FN)*
Actual Negative	False Positive *(FP)*	True Negative *(TN)*

A positive will mean a student did not pass the OSSLT on first attempt, while negative will mean they did pass. Additional evaluation metrics of the classifier can be extracted from the confusion matrix, precision and recall. These two metrics are defined as:

Equation 4. Precision

$$P = \frac{TP}{TP + FP}$$

Equation 5. Recall

$$R = \frac{TP}{TP + FN}$$

Precision looks at the amount of true positives the algorithm classifies against the total amount of positive predictions it made. Recall looks at the amount of true positives against the amount of actual positives. Thus, if the algorithm were to only predict that every student were to pass, it would have both a precision and recall of 0.

To conveniently look at the precision and recall of a classifier together, the two stats can be combined into what is known as the f_1-score or simply f-score. The f-score is the harmonic mean between precision and recall with a couple of useful properties: if either precision or recall is 0, then the f-score is 0, and if precision and recall are both 1 (perfect), then f-score is 1. Thus, the scale for the f-score is between 0 and 1, which is useful and will be used later in this paper for evaluation purposes.

2.3 Feature Engineering

A machine learning algorithm learns by assigning weights known as parameters to a respective feature. What features are used in the algorithm are extremely important as it affects prediction accuracy and how well it learns. The original raw data provided information on each student's database ID, birthday, grade, sex, homeroom, averages, community service hours, and many more. The idea of feature engineering is to transform the raw data into data that best represents the objective. The features selected for this problem were a student's sex, whether or

not English is their native language, their grade 9 English average, and their "education level" for English. In the province of Ontario, different levels of a course can be taken to suit a student's needs.

These levels include locally developed (courses that only emphasize on key basic skills and provide students with lots of support and flexibility), applied (courses that focus on essential concepts and emphasize on the practical applications), academic (courses that focus on essential concepts, explore related topics, and emphasize on both theoretical and practical applications), and pre-IB (courses developed to prepare students for the rigorous IB diploma programme) [3].

An example of how a feature was engineered from its raw form was determining if English was a student's native language. By inspecting the course code of the student's grade 9 English class, checking if it contained "ESL" indicates that the student took an "English as Second Language" course. If a student took multiple ESL courses in that year, a mean would be taken. Another choice that was made when engineering features was the choice to exclude the amount of community service hours a student has. This is because the raw dataset provided data on all students from a certain date. Thus if a student from grade 12 (last year of high school in Canada) has 420 hours completed, while a grade 10 student who has only 20 hours completed, the comparison is unjust as the grade 12 student has had more time to complete their hours. Therefore including community hours as a feature would only mislead the classifier.

The majority of features are handled as binary discrete, for example "is this student taking English as at an Academic level?", 1 for yes, 0 for no. Handling features like this is useful to separate importance when the classifier puts weights on each feature. In the dataset, out of the 78 people who did not pass the OSSLT on first attempt, only 2 were in the pre-IB programme. Thus the algorithm will understand the importance that it is unlikely for a student in the pre-IB

programme to fail the test. The only one feature with a continuous value is a student's grade 9 English average. The range of the feature is between 0 and 100, this becomes a problem for the learning algorithm as it becomes more difficult for it to compare this one feature with all the others as it is on a much larger scale [4]. To handle this problem, the feature was standardized through the use of z-scaling, meaning it was transformed such that it was given a mean of 0 and a standard deviation of 1.

Equation 6. Z-Scale

$$x_i = \frac{x_i - \bar{x}}{\sigma_x}$$

Each English average was subtracted by the mean of all English averages, and divided by the standard deviation. Feature engineering is critical to the performance of a learning algorithm and is always the first step.

3 Logistic Regression and Classification

3.1 The Hypothesis and Sigmoid Function

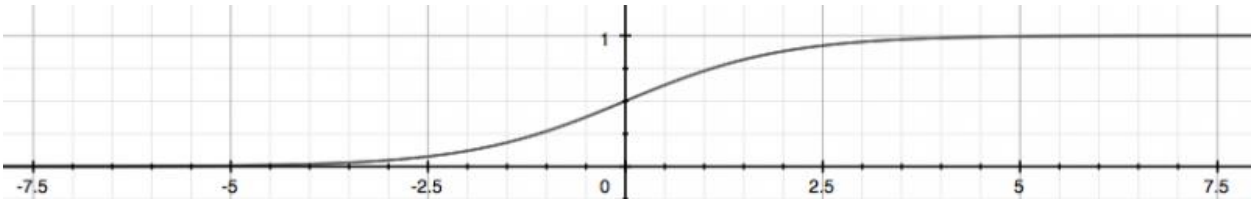
Rather than outputting a continuous value indicated by the "regression" in its name [5], logistic regression actually outputs a discrete value as a classification. While it can be used to classify more than two classes, it is out of the scope of the paper and will only be used for binary classification. To predict an output given an input, a hypothesis function is used, identified as $h_\theta(x)$. To obtain a model that only outputs between 0 and 1, the sigmoid (logistic) function which is able to accept any real number is used as the hypothesis.

Equation 7. The sigmoid function

$$h_\theta(x) = g(z) = \frac{1}{1 + e^{-z}}$$

$$Where\ z = \theta^T X$$

Figure 1. A graph of the sigmoid function [6]



The hypothesis will provide a probability for a certain output, thus:

Equation 8. Probability of a classification given by the hypothesis

$$h_\theta(x) = P(y = 1|x; \theta) = 1 - P(y = 0|x; \theta)$$

Typically, the decision boundary between one class and the other is at 0.5. However there is no set rule, the threshold can be changed to any value between 0 and 1 inclusive.

3.2 Cost Function and Gradient Descent

To penalize the logistic regression algorithm, when predicting incorrectly, a cost function with a convex shape when plotted is used to be minimized. The one used is represented by:

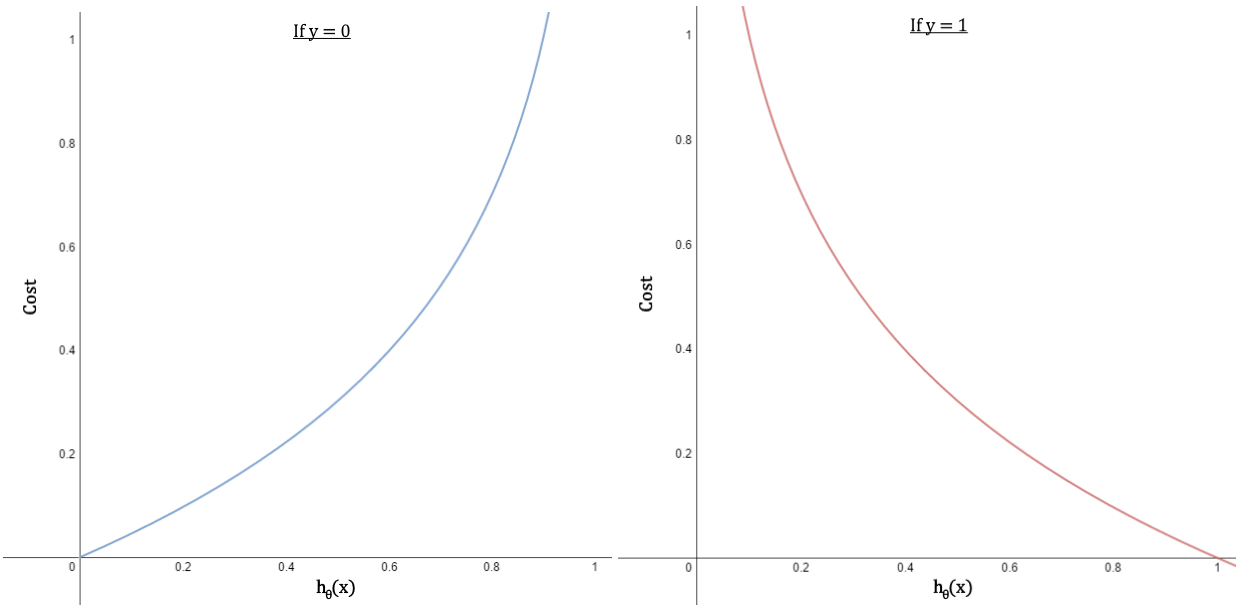
Equation 9. The cost function

$$J(\theta) = -\frac{1}{m}\sum_{i=0}^{m} Cost\left(h_\theta\left(x^{(i)}\right), y^{(i)}\right)$$

$$Where\ Cost(h_\theta(x), y) = -\log\left(h_\theta(x)\right) \quad if\ y = 1$$

$$Cost(h_\theta(x), y) = -\log\left(1 - h_\theta(x)\right) \quad if\ y = 0$$

Figure 2. Graphical representation of the cost function



The graphs in Figure 4, demonstrates the purpose of the cost function: *when y = 0, as $h_\theta(x)$ approaches 1 or when y = 1, as $h_\theta(x)$ approaches 0, cost approaches infinity*. The function in Equation 3, above can be condensed and put into vector form for simplicity. Since the value of *y* can be only 0 or 1, one of the two terms of the cost function will be zero and so the cost will be evaluated accordingly.

Equation 10. The cost function condensed

$$J(\theta) = -\frac{1}{m}\left(\log(g(X\theta))^T y + \log(1 - g(X\theta))^T(1 - y)\right) + \frac{\lambda}{2m}\theta^{1T}\theta$$

The last term in Equation 10 is an additional component not represented in Equation 9, this component is used for regularization. Overfitting is a problem machine learning, especially those with small data sets. That is a learning model might be close to or basically perfect, but if it were to predict on a different dataset, it would become severely inaccurate, meaning the learning

model did not generalize very well. What the regularization term (represented by λ) does, is inflate the parameters so when training, the learning algorithm will suppress certain features. The θ^1 in the equation represents the parameters but with the bias term holding the value of 0 since the bias term is not regularized. Regularization essentially allows the learning model curve to be smoothened out so that it may generalize well.

A relatively simple method known as gradient descent is utilized to identify the local minimum of a function. This method navigates through a function by taking the fastest steps towards its negative gradient. The gradient is found by taking the partial derivative of the cost function with respect to each parameter θ_j.The general form takes each parameter and assigns its new value by subtracting itself by the gradient multiplied by a learning rate, α. The learning rate is required to adjust the extent at which the parameter changes. A value too small, may require many iterations. A value too large may cause gradient descent to overshoot the local minimum and fail to converge.

Figure 3. Gradient descent form in logistic regression

Repeat

{

$$\theta_j := \theta_j - \alpha\left[\frac{1}{m}\left(\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}\right) + \frac{\lambda}{m}\theta_j\right]$$

}

The implementation in this paper utilizes a vector approach rather than an iterative approach as a vector approach is more computationally efficient by calculating all the values at once rather than one feature at a time.

Figure 4. Vector approach of gradient descent

Repeat

{

$$\theta := \theta - \frac{\alpha}{m} X^T (g(X\theta - \vec{y}) + \frac{\lambda}{m}\theta^1$$

}

4 Training the Learning Model

4.1 Train Test Split

A challenge that comes across in supervised learning is the actual training of the model. The problem is ensuring that the learning model has enough data to create accurate predictions and enough data to test on. A simple approach to solving this problem is the train test split method. As the name suggests, the dataset is randomly split in half into two categories: the training set, and the testing set. The pros of using this method allows for quick and easy evaluation of the learning model's performance. However the problem with this approach is that the split may not always be an even one when it comes to the different classes. This is especially a huge problem with imbalanced datasets such as this one. Not only does the learning problem have a minimal amount of data to learn from the minority class, it becomes even smaller when the training set is split. For additional comparison, a confusion matrix is included for a test without any modifications (ie. Feature engineering) in the dataset, and one with modifications.

Figure 5. Confusion matrix for logistic regression without any dataset modifications

	Predicted Failed	**Predicted Passed**
Actually Failed	0	24
Actually Passed	3	210

Figure 6. Confusion matrix for logistic regression with dataset modifications

	Predicted Failed	**Predicted Passed**
Actually Failed	184	32
Actually Passed	39	177

The first thing to note in the confusion matrix in Figure 6 is that the amount of data points is greater than the one in Figure 5. A simple version of an approach called oversampling was used to counter the dataset imbalance problem. Since the ratio of the majority class to the minority was about 11:1, the minority class was copied 11 times to attempt to mimic a balanced dataset. The importance of feature engineering is shown through this comparison as the learning model had an f-score of 0, while the second model had an f-score of 0.84. Fortunately, the amount of true positives and true negatives are much greater than the number of false positives and false negatives. Despite the data imbalanced being addressed, an additional problem arises where the number of students that were predicted to fail rose. This trade off changes as the threshold of which class logistic regression should predict.

While the ease of use is convenient for comparing certain aspects of the learning model, another drawback of the train test split method is that it may not be very consistent. Not only does performance vary due to the random shuffling of data, but how well the learning generalizes cannot be found through this method as well.

4.2 K-Fold Cross Validation

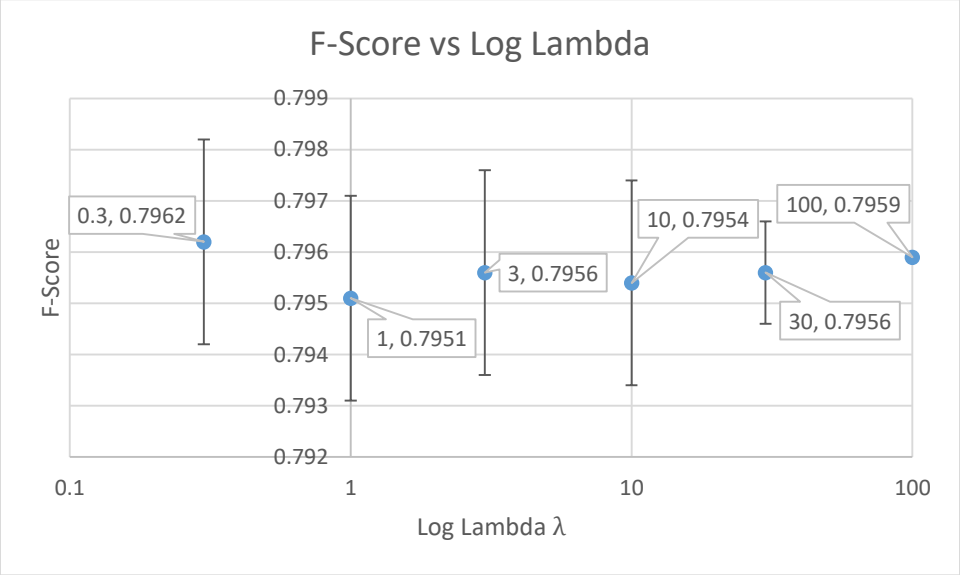
Two critical problems of the train test split method was the trade-off between using a part of the data as a training set or a testing set and now being able to determine how well a learning algorithm will generalize to any additional data that is inputted. K-fold cross validation addresses all these problems by splitting the dataset into k different “bins” that have been shuffled, rather than splitting the dataset into two. Now the learning algorithm iterates through each bin, identifying it as the test set, with every other bin being the training set. After each iteration, an evaluation is done on how well the learning model classified. In this case, the f-score was used due to it being an effective method to evaluate imbalanced datasets. In the end, the scores are averaged to evaluate how well the learning model will generalize to any dataset. This in a way allows the learning model to use the entire dataset as a training set and a test set at the same time. To get additional accuracy on the performance estimate, the k-fold cross validation method can be run several times to get an average along with its standard deviation since the dataset is shuffled each time. The drawback to k-fold cross validation is that it requires a greater amount of computing power, the learning model must learn through each fold. The value of k that was chosen was 10, Ron Kohavi [7] demonstrates experimentally that a ten-fold cross validation was the most effective even if computational power allows for additional folds. When running cross validation through the over sampled dataset, an f1-score of 0.81 ± 0.003 was outputted. It should be noted that no regularization ($\lambda = 0$) was used for the examples in this section and the previous (4.1).

5 Evaluation of the Learning Model

5.1 Performance with a Change in Lambda

The possibility of the risk that the learning model will overfit the dataset is addressed using the regularization term lambda (λ). Based on how k-fold cross validation describes how well a learning model will generalize to a new dataset, the best regularization parameter can be decided by iterating through different levels of lambda. The choices of lambda were chosen with the values of 0, 0.3, 1, 3, 10, 30, and 100 to inspect a wide range of values. The results from each regularization parameter in the learning algorithm are shown in Figure 7.

Figure 7.



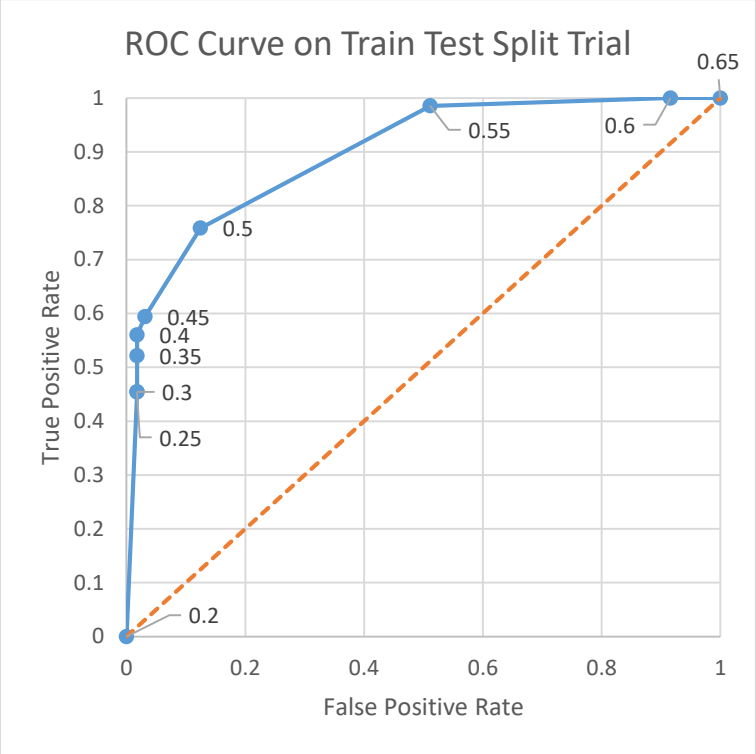
The above figure demonstrates that a change in the regularization parameter results in a negligible change in f-score. Not only is it a negligible change, but the error in the f-score also contains too much variation to extract any useful information in choosing the best value of lambda. Note that the graph's horizontal scale is logarithmic, thus the lambda with the value of 0

is not plotted. Despite that, with its value of 0.7948, the conclusions made remain the same. Since regularization puts additional costs on feature parameters to reduce its importance, regularization does not work very well with binary variables. Each of the binary features take on the same value (0 and 1), any regularization will bring down the each of those features by the same amount. Due to the fact that only one of the dataset's features are continuous with the rest being binary discrete, regularization would not help at all. Therefore, for the rest of each evaluation, the regularization parameter is set to 0.

5.2 Performance with a Change in Threshold

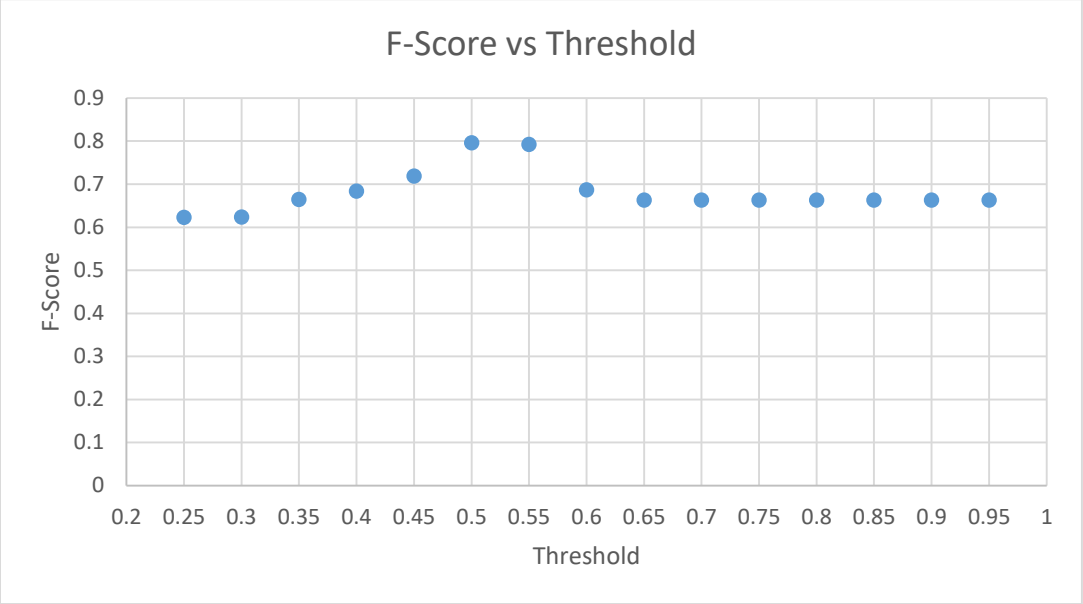
Logistic regression predicts the probability of a given input being a certain class, its decision on what to classify it as is known as the threshold. A higher threshold (ex. 90%), means that classifying the input as positive must require a great amount of confidence. This lowers the amount of false positives with the expense of possibly missing those who are actually positive (false negatives). This works the other way around as well and can be expressed using a receiver operating characteristic (ROC) curve (Figure 8).

Figure 8.



There are 2 points on the curve graph that every ROC has: (0, 0) meaning no positive predictions were made at all, and (1, 1) meaning every single data point has been predicted positive. The dashed diagonal line in the middle is called “random guess” or “line of no-discrimination”. It is ideal to have the classifier’s prediction performance above this line. A perfect classifier would have a point on (0, 1). The learning model seems to have an almost perfect true positive rate with a threshold of 55%, however at this point, the false positive rate reaches to about 50%. The f-score found through k-fold cross validation and threshold are plotted against each other as well to understand the overall performance.

Figure 9.



The best performance is achieved by a threshold of 50% with 55% trailing almost right behind. To choose which threshold to use for real world use depends on the objective of the classifier. It would be much worse if a cancer classifier only predicted positive on the ones it was truly confident for example. It would be better to receive positive results and then find out it was actually nothing than to get negative results and finding out that one has cancer when it is too late. In the same way, it would be worse for students who are likely to fail the OSSLT to be predicted that they would pass. Therefore despite having a smaller f-score, it is not wrong to choose a threshold of 55%.

5.3 Space and Time Complexity

Given that there are *n* features in the dataset, and *m* training examples, the amount of space logistic regression requires to run is *O(mn)*. The time complexity of logistic regression depends on the implementation. Since gradient descent operates through the number of iterations as a tunable parameter, the time needed to run a vector implementation of logistic regression is

$O(m)$, where m is the number of iterations set to run. However if an iterative implementation was used, meaning each parameter had to be individually be changed one at a time, the time complexity would be $O(mn)$ where n again is the number of features in the dataset.

5.4 Machine Learning with School Datasets

While the f-score of the learning model used could only peak at about 80%[1], it was a massive improvement from when it was essentially not usable at all. It is possible to have a relatively small dataset compared to large tech companies and extract useful and applicable knowledge. In addition, due to only being a student, it is impossible to make access all the data the school holds. Therefore some other valuable information such as attendance would possibly improve the learning model. The key to implementing effective machine learning algorithms is to clean up the data and manipulate it to be the best representation of the problem as possible.

6 Conclusion

Logistic regression is an effective learning model for classification. After feature engineering and over sampling the dataset, performance at detecting those who would not pass on the first try improved dramatically. It is possible to achieve a nearly perfect true positive detection with the expense of false positive detection reaching to a level of about 50%. To obtain a higher performance, simply obtaining more training examples would help. While it was concluded that the learning algorithm did not overfit the dataset, the data did only come from one school. The EQAO has classified schools based on their performance as either high achieving schools or low achieving schools [7]. The school the dataset is based is considered to be a high

[1] 80% and 0.80 can be used interchangeably

achieving school. Therefore in order to train a learning model applicable for all of Ontario, student data would have to include from a variety of schools across the province.

Acknowledgements

I would like to thank Andrew Ng for his machine learning course which provided the fundamental knowledge to go upon this research [10]. I would also like to thank the developers of Scikit-Learn [8] and Numpy [9] for their machine learning and scientific computing libraries used in the actual implementation of these algorithms.

References

[1] News, CBC. "Ontario Secondary School Students 82% Successful in 2015 Literacy Test." CBCnews. CBC/Radio Canada, 26 Aug. 2015. Web. 22 Feb. 2016.

[2] McMahan, H. Brendan, et al. "Ad click prediction: a view from the trenches." Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2013.

[3] "Secondary Program." Ontario Ministry of Education. Web. 28 Feb. 2016.

[4] James, Gareth. An Introduction to Statistical Learning. New York, NY: Springer, 2013. Print.

[5] Hosmer, David W., and Stanley Lemeshow. Applied Logistic Regression. New York: Wiley, 2000. Print.

[6] Neylon, Tyler. LogisticFunction. Digital image. Coursera. Coursera, 17 Feb. 2015. Web. 22 Feb. 2016. <https://share.coursera.org/wiki/index.php/File:Logistic_function.png>.

[7] Simon, Marielle, and Shelley Stagg Peterson. Characteristics of High- and Low-Achieving English-Language Schools. Rep. EQAO, Oct. 2012. Web. 22 Feb. 2016.

[8] Pedregosa, Fabian, et al. "Scikit-learn: Machine learning in Python." The Journal of Machine Learning Research 12 (2011): 2825-2830.

[9] Developers, NumPy. "NumPy." NumPy Numpy. Scipy Developers (2013).

[10] Ng, Andrew. "Stanford University Machine Learning (Course Handouts)." CS229 Machine Learning Autumn 2015. Stanford University. Web. 5 Oct. 2015.