K-nearest neighbours and Logistic regression classification comparison

Pablo Quinoa & Victor Ceballos Espinosa

Description and motivation

- Compare Logistic Regression and K-Nearest Neighbours algorithms on classifying adults in two categories: whether they earn more than $50K / year or not based on census data.
- We will try to find what type of applications each of the two models fit better for. Thus, try to answer the question: when should we use one or the other?

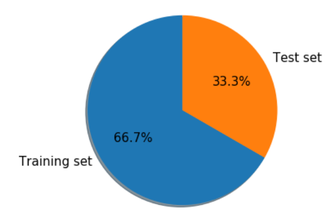
Hypothesis statement

- We thought that because of the different scales in the numeric variables, logistic regression was going to perform better (Srikanth, P. 2014.). After downscaling the data to the interval [0, 1] we expect to see a similar accuracy performance in both algorithms .
- We think that logistic regression should be much faster when classifying new samples than KNN, as this last will need to compute distances to all other points in the data set to classify a single point (Ashari et. al., 2013).

Initial analysis

- Dataset extracted from UCI Machine Learning Repository (https://archive.ics.uci.edu/ml/datasets/adult).
- 14 features (8 Categorical, 6 Numerical) and 1 label (two possible values).
- 76% of the training samples don't earn more than $50K / year and 24% earn more than $50K / year. To have the same representation of each class, we applied downsampling to the training data.
- The mean and standard deviation for each numerical feature before scaling down the values can be found in the table on the right.
- The numerical features were scaled down to the interval [0-1]. There are two reasons that explain why we decided to scale down the values. First of all, we can reduce the time that the models take to train. Secondly, the accuracy was increased.
- Categorical variables have been encoded using the method One-hot. To be more precise, we used OneHotEncoder class from Sklearn. Because of this, we ended up having 108 features in total.

	Mean		Standard Deviation	
Numerical feature	< £50 K / year	>= £50 K / year	< £50 K / year	>= £50 K / year
age	36,8	44,3	14,3	10,6
fnlwgt	190340,9	188005,0	106768,6	102247,3
education_num	9,6	11,6	2,4	2,4
capital_gain	148,8	4006,1	882,0	15131,3
capital_loss	53,1	195,0	318,4	588,6
hours_per_week	38,8	45,5	12,4	11,3



Models

Logistic regression

Logistic regression is a regression algorithm as its names indicates. The main difference with linear regression models is that it calculates the probability that a sample belongs to a class (logit). This logit is a sigmoid function that outputs a number between 0 and 1. Once this logit is calculated, it is possible to get the belonging class (Géron, 2017) using as a threshold 0.5.

Advantages

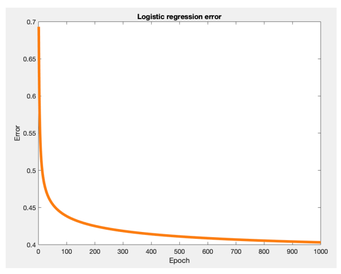
- It is not very computationally demanding to train.
- Variables don't need to be scaled down.
- Very understandable.
- Easy to implement.
- Fast prediction times.

Disadvantages

- Vulnerable to overfitting.
- Works better when unrelated features are removed. So it needs feature engineering.

Training

- To fit a predicted value between 0 and 1 we used the sigmoid function.
- To find the best weights or parameters for our logistic model we used gradient descent on the cost function.
- We trained the model through 1000 epochs and a batch size equal to the number of points in the dataset.
- Regularization was applied on the cost and gradient function to prevent overfitting.
- To fit a predicted value between 0 and 1 we used the sigmoid function.



Choice of parameters and experimental results

- In the case of the logistic regression model, hyperparameter tuning is not essential unlike other algorithms. The only hyperparameter we tuned was a lambda (λ) regularization factor. Such lambda adds a penalty to the cost function to prevent overfitting, so that our model generalizes better when applied to unseen data. We found out using five-fold cross-validation that regularization with a small lambda (0.01) was the perfect fit for our model, so it was the perfect bias-variance trade-off.
- We also calculated the error of the logistic regression for each epoch. As it can be seen in the plot on the right. The error drops dramatically fast before reaching half of the proposed number of epochs.

Lambda	Accuracy
0.01	81,36 %
0,1	81,25 %
0,5	81,29 %
1	81,18 %
10	81,12 %
100	79,22 %

Hyperparameter tuning on **training set** using five-fold cross-validation

K-Nearest Neighbours

K-Nearest Neighbours is a method to classify an object based on the majority class amongst its k-nearest neighbours (Ashari et. al., 2013). The way it works, is that given K (the number of neighbours to compare with), for each new element to classify, the distance to each sample in the training set is calculated. Finally, the majority label of the k closest elements is selected (Harrison, 2018).

Advantages

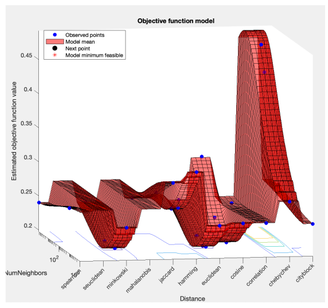
- It is a very simple algorithm
- Very easy to understand
- Easy to implement.
- No need to make further assumptions.

Disadvantages

- Computationally demanding.
- It is affected by the amount of samples in the training set. The more samples, the slower.
- Lazy training.
- Slow to classify. It may not be suitable for many real-time applications.

Training

- KNN does not have a formal training phase. However, a tree structure to divide the training data space is generated in order to accelerate the predictions.



Choice of parameters and experimental results

- To fulfill the hyperparameter tuning, an optimization technique called Bayesian Optimization was used. Such technique consists in a stochastic iterative process to find the best combination of hyperparameters applying five-fold cross-validation.
- The best model uses 41 neighbours and euclidean distance.
- We tested the same model given a downscaled version of the dataset and a non-downscaled one. Applying downscaling resulted in a higher accuracy of 77%. If downscaling is not applied, the accuracy drops to 68%.

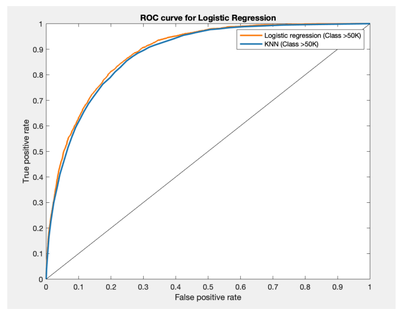
	Accuracy
Downscaling	77 %
No downscaling	68 %

Accuracy difference with and without applying downscaling on the data

To compare and evaluate both models, we have decided to use the following evaluation methods: Classification time. This is the time it takes to classify the entire test dataset. This will allow us to infer which one is more suitable for real time classification applications. In addition, we calculated precision, recall, F-score and accuracy on the testing set to compare both models performances and tradeoffs. Lastly, we plotted the ROC curve for each model. All measures have been calculated in the same environment.

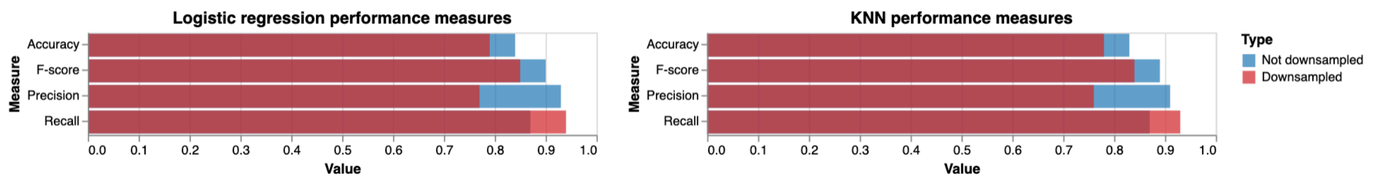
Analysis and critical evaluation

- The results obtained from both models are very close to each other. Most of the measures are very similar in both models as can be observed in the table in this section. Our results are reinforced by those of (Feng, J., et al., 2019) where 25 models were compared, and concluded that KNN and Logistic regression performed roughly equal.
- The main difference between both models is the time required to classify new instances. We compared the time to classify the entire test set (16281 data points) which can be observed in the same table. The time to classify these instances took three orders of magnitude more in KNN than in Logistic regression. With this result, we can conclude that for real time applications, KNN can be slow to fit its purpose. However, in (Hong, T., et al., 2016) clustering is used to group similar data points (e.g. Netflix users with a similar taste for movies), so that only calculating the distances against the other points of the cluster is required, hence radically reducing the classification time.
- It can be concluded that given a dataset that is constantly changing, KNN is a much better option than Logistic Regression. This is because KNN computes all calculations in real time, whereas Logistic Regression uses a trained model that would no longer be representative if the dataset is very dynamic.
- ROC curves almost overlap each other (AUC differs in 0.01).
- We also trained the models both with and without downsampling, and as the performance results suggest in the graph below in this section, we concluded that balancing the dataset before fitting the model is not the best solution as it biases the model and waste potentially useful data.



	Logistic regression	KNN
Accuracy	0,79	0,78
Precision	0,77	0,75
Recall	0,94	0,94
F-score	0,85	0,84
Classification time	14,21 ms	14737,55 ms
AUC	0,89	0,88

These measures have been calculated using the **test set**



Lessons learned and future work

- KNN does not generalize the training data until a classification is requested (lazy learning), but this can be very useful for when the training set is constantly updated.
- Unless our model has a low number of features or dimensions, KNN will be slow to classify compared to Logistic Regression. Thus, we found that there is no best algorithm, but rather an algorithm that suits better depending on each use case and the nature of the problem.
- Future work on KNN would be to explore the use of clustering to dramatically decrease the classification time required having as a reference the approach taken by (Feng, J., et al., 2019).
- Future work on Logistic Regression would be to examine how different basis functions alter the results of the model (e.g. tanh instead of the sigmoid).

Géron, A. (2017) Hands-on machine learning with Scikit-Learn and TensorFlow: concepts, tools, and techniques to build intelligent systems. First edition. Beijing ; Boston: O'Reilly Media.
Harrison, O. (2018). Machine Learning Basics with the K-Nearest Neighbors Algorithm. [online] Medium. Available at: https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761 [Accessed 8 Nov. 2019].
Ashari, A., Paryudi, I. and Min, A. (2013). Performance Comparison between Naïve Bayes, Decision Tree and k-Nearest Neighbor in Searching Alternative Design in an Energy Simulation Tool. International Journal of Advanced Computer Science and Applications, 4(11).
Srikanth, P. (2014). Naive Bayes or K-NN for classification?. [online] Medium. Available at: https://medium.com/data-science-analytics/naive-bayes-or-k-nn-for-classification-60a4d92e7bab [Accessed 9 Nov. 2019].
Fawcett, T. (2006). An introduction to ROC analysis. Pattern Recognition Letters, 27(8), pp.861-874.
Davoudi Kakhki, F., Freeman, S. and Mosher, G. (2019). Evaluating machine learning performance in predicting injury severity in agribusiness industries. Safety Science, 117, pp.257-262.
Feng, J., Wang, Y., Peng, J., Sun, M., Zeng, J. and Jiang, H. (2019). Comparison between logistic regression and machine learning algorithms on survival prediction of traumatic brain injuries. Journal of Critical Care, 54, pp.110-116.
Hong, T., Tsamis, D. (2016). Use of KNN for the Netflix Prize.