Project 1: Machine Learning 101

Group 65
Alhusain Abdalla, Rossen Vladimirov, Ege Yay

Abstract

In this project, we implemented two classification techniques – Logistic Regression and Naive Bayes – and compared their performance on four datasets. We cleaned the data, prepared it into the appropriate format and calculated meaningful statistics to help avoid errors in the algorithms and improve accuracy. Using NumPy and the Python standard library, we implemented both classifiers from scratch. For Logistic Regression, we found that gradient descent converges faster for larger learning rates. However, the highest accuracy was achieved with a learning rate which is neither too large nor too small. We also found that accuracy improves when the number of iterations increases. We implemented data normalization to speed up convergence and avoid numeric overflow when computing the logistic function. Both the Logistic Regression and Naive Bayes classifiers performed relatively well in terms of accuracy and were quick to train. In our tests, Logistic Regression achieved higher accuracy than Naive Bayes but was slower to train. We also showed that training the models on larger datasets leads to better accuracy. Finally, we demonstrated the importance of feature selection and developed a feature recommender – an automated approach for suggesting a good subset of features for training.

1. Introduction

1.1 Related Work

Discriminative classifiers such as the Logistic Regression classifier model the conditional probability p(y|X) and directly learn a map from the features X to the label y. On the other hand, generative classifiers like Naive Bayes model the joint probability p(X,y) of the features X and the label y. They use Bayes' rule to calculate the posterior probability p(y|X) and select the most likely label y as the prediction.

Ng and Jordan (2002) compared discriminative and generative classifiers as represented by Logistic Regression and Naive Bayes. They investigated the effect of the training set size on accuracy and challenged the popular belief that discriminative classifiers always outperform generative ones. They conducted 15 experiments and showed that Naive Bayes has better accuracy than Logistic Regression on smaller datasets, while Logistic Regression outperforms on larger ones. The researchers demonstrated that, while the Logistic Regression classifier has a lower asymptotic error than Naive Bayes, Naive Bayes reaches its higher asymptotic error faster.

1.2 Logistic Regression Implementation

The Logistic Regression implementation supports different learning rates and stopping criteria (tolerance and maximum number of iterations) for gradient descent. We investigated the effect of different learning rates for gradient descent on Logistic Regression performance (convergence speed and accuracy) and found that larger learning rates lead to faster convergence. However, the highest accuracy is achieved with a learning rate which is neither too large nor too small. We then investigated the accuracy of Logistic Regression as a function of the number of iterations of gradient descent and found that it improves when the number of iterations increases.

1.3 Naive Bayes Implementation

The Naive Bayes implementation is based on Bayes' rule and the assumption that the features are conditionally independent given the label. We implemented both Gaussian and Bernoulli likelihoods for features. We tested the Gaussian approach with and without the variance calculation. We found that the without-variance calculation leads to better performance as the standard deviation on small values (constructed by the product of probabilities) is not stable and the result does not change significantly. We also implemented the Bernoulli approach for binary features. We concluded that the Bernoulli approach is less accurate than the Gaussian one, and since most of our datasets are continuous, we used the Gaussian approach.

1.4 One-Hot Encoding

We preprocessed the data and performed one-hot encoding for categorical features. Although some algorithms can work directly with categorical data, others (e.g., Logistic Regression) require all input and output variables to be numeric. One-hot encoding is a solution for converting categorical variables into a numeric form. A categorical value is represented as a bit vector in which only a single bit is 1 and all others are 0.

1.5 Data Normalization

To achieve better performance with the Logistic Regression classifier, we implemented data normalization. We standardized each feature by subtracting its mean and dividing by its standard deviation. This rescales each feature to have mean 0 and standard deviation 1. This speeds up convergence and avoids numeric overflow when computing the logistic function.

1.6 K-Fold Cross-Validation (CV)

We implemented k-fold CV used in two scenarios. First, we use k-fold CV on the train/validation set to pick the best hyper-parameter values for Logistic Regression: learning rate, tolerance, and maximum number of iterations for gradient descent. We split the train/validation set into k-folds and each time take k−1 folds as a training set and 1 fold as a validation set. We pick the hyper-parameter values corresponding to the best average accuracy. Second, we use CV to assess each model's performance more accurately. Averaging the accuracy over all folds provides a better error estimate. We retrain the model on the full train/validation set and get the final accuracy on the test set.

1.7 Logistic Regression and Naive Bayes Performance

We demonstrated that the Logistic Regression and Naive Bayes models perform relatively well in terms of accuracy and are quick to train. The Logistic Regression classifier achieves higher accuracy than Naive Bayes on our datasets but takes longer to train. We investigated the effect of the size of the training dataset on accuracy. Training the models on larger datasets leads to better accuracy. Models learning from a larger number of examples generate better predictions.

1.8 Feature Recommender

We demonstrated the importance of feature selection to achieve good accuracy. To help with feature selection, we developed a feature recommender – an automated approach for suggesting a good subset of features for a classifier with high accuracy predictions. The feature recommender scans all features in a dataset and recommends features to be included in the subset of features suitable for training a classifier. The type of each feature (categorical, continuous, etc.) is provided as a parameter. For categorical features, the feature recommender counts the number of times a distinct value appears. If most (e.g., more than 80%) of the values in a categorical feature are the same, the feature is not recommended for inclusion in the subset of features for training. The threshold (80%, 90%, etc.) is configurable. For example, the feature recommender suggests excluding a feature from the Adult dataset which contains country names since 29,170 out of 32,561 values (89.6%) are the same (United-States). For continuous features, the recommender calculates the correlation between the feature and the target label. If the correlation coefficient is below a specified threshold (in absolute value), the feature is not recommended for inclusion.

2. Datasets

2.1 Datasets

We tested our implementations on four classification datasets downloaded from the UCI machine learning repository (Dua & Graff, 2019). Table 1 presents basic information regarding the datasets. Previous implementations and literature show that Logistic Regression and Naive Bayes classification perform differently according to the ratio between number of instances and number of features and the degree of correlation between features. Therefore, we ensured that our choice of datasets satisfied a contrast of these factors.

Table 1: *Basic information regarding our datasets*

Dataset	# Instances	# Features
Ionosphere	351	34
Adult	48842	14
Breast Cancer	569	10
Red Wine Quality	4898	12

2.2 Data Cleaning and Preparation

Before starting any actual machine learning coding or analyses, it is common that developers spend around two-thirds of their time cleaning and preparing their data (Jafferjee, 2019). This cleaning refers to dealing with useless features, missing data, cardinalities, outliers, and other inconsistencies in the data. Doing so, the data is ensured to be more representative and less probable to confuse the classifier (Jeatrakul, Wong & Fung, 2009). For all four datasets, we started by removing instances with missing data and features with irrelevant information or null values. For instance, all "ID" features were dropped. We also calculated the correlation matrix and set a threshold of >0.9 between any two features as an indicator to remove one of them (as it is redundant). However, this approach was found to drastically decrease the classification accuracy and was therefore cancelled.

To prepare the data, we converted all categorical features to binary features through one-hot encoding. For the Red Wine Quality dataset, we converted the labels from continuous values to binary by setting a threshold of ≥7 on the quality scale for a "good wine". Then, we split all datasets into their training and testing subsets using a dynamic parameter that decides the size of test data.

2.3 Statistics

Understanding our data is critical in preprocessing as well as analyzing the final results. For instance, we calculated the distribution of classes in each dataset, the correlation matrix between features, basic statistics of all features' columns (mean, standard deviation, highest 25%, etc.). In addition, we visualized some scatter plots and heatmaps between features to further understand their degrees of correlation. Figure 1 shows statistics applied to the breast cancer dataset.

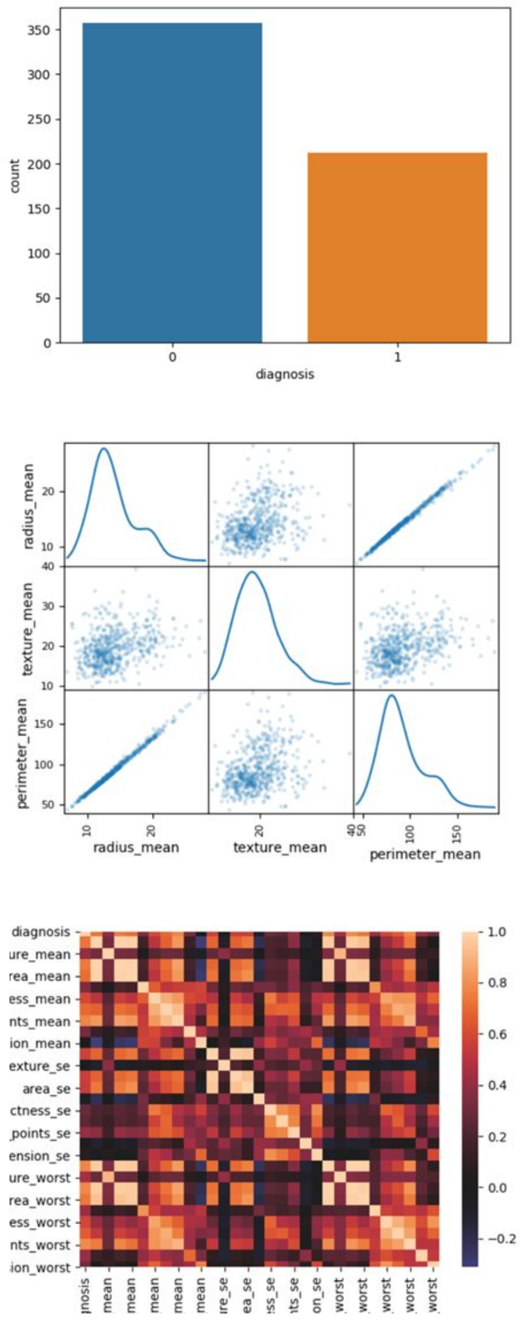


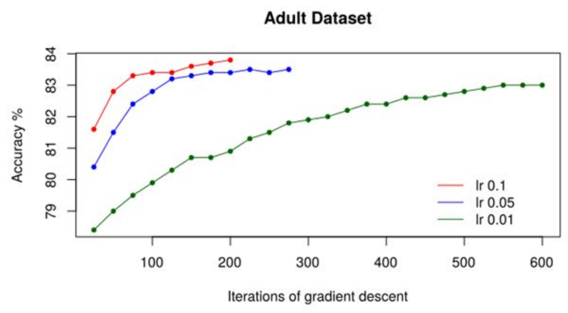
Figure 1: *Statistics applied to the Breast Cancer dataset.* ***A****. Distribution of positive vs. negative classes of the target variable.* ***B.*** *Scatter matrix of 3 features with density plots.* ***C.*** *Heatmap of the correlations between all features.*

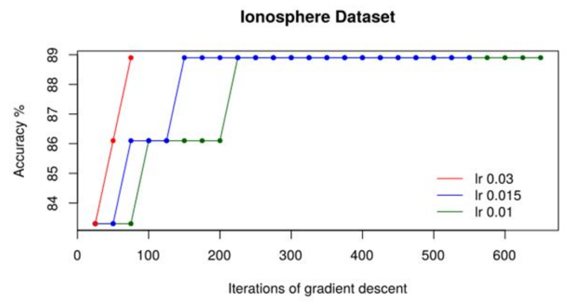
3. Results

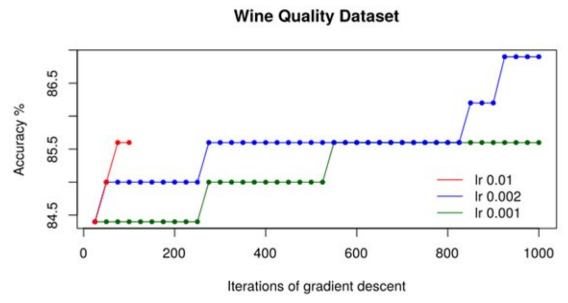
3.1 Effect of Learning Rate and Maximum Number of Iterations of Gradient Descent on Logistic Regression Performance

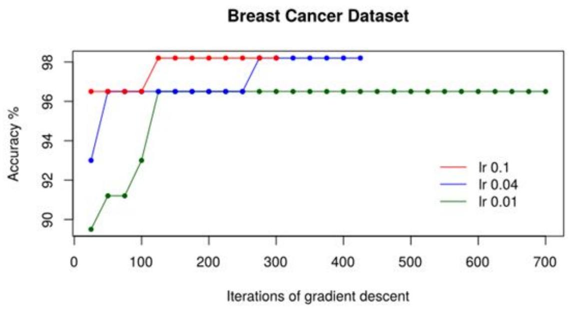
We set the learning rate for gradient descent and perform a 5-fold CV on the train/validation set to find the best values for other hyper-parameters.

We plot the accuracy on the train/validation set as a function of iterations of gradient descent. The gradient descent converges faster for larger learning rates. Logistic Regression achieves the highest accuracy with a learning rate which is neither too large nor too small. On the Wine Quality dataset, the accuracy achieved for learning rate 0.002 (86.9%) is higher than the accuracy for learning rates 0.01 and 0.001. The accuracy improves when the number of iterations increases.









3.2 Comparison of the Accuracy of the Logistic Regression and Naive Bayes Classifiers

We shuffle each dataset and divide it into a train/validation set (90%) and a test set (10%). First, we run 5-fold CV to pick the best hyper-parameter values for Logistic Regression and evaluate its accuracy on the train/validation set. The 5-fold CV algorithm runs with the following parameters: learning rate [0.1, 0.05, 0.01], max number of iterations [1000, 2000], tolerance: [0.01, 0.001, 0.0001]

Best hyper-parameter values for each dataset:

Dataset	**Best hyper-parameter values for Logistic Regression (5-fold CV)**		
	Learn. rate	**Max Iter.**	**Tolerance**
Adult	0.1	1000	0.0001
Ionosphere	0.05	1000	0.001
Wine Quality	0.1	1000	0.01
Breast Cancer	0.1	1000	0.0001

Second, we run 5-fold CV to evaluate the accuracy of the Naive Bayes classifier on the train/validation set. Third, we evaluate the final accuracy of each classifier on the test set.

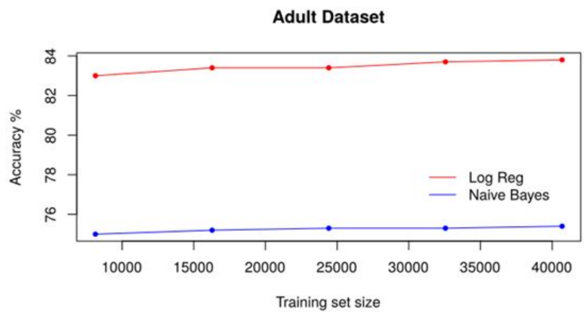
Accuracy of Logistic Regression and Naive Bayes:

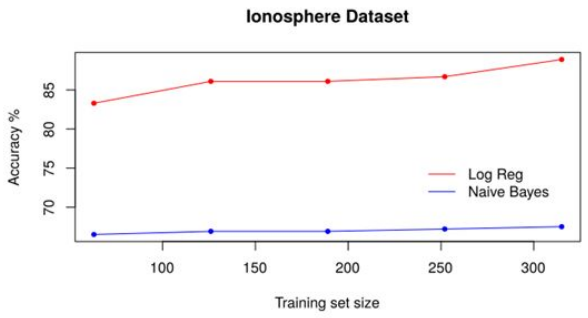
Dataset	**Accuracy %**			
	Logistic Regression		**Naive Bayes**	
	5-fold CV train/val	**test set**	**5-fold CV train/val**	**test set**
Adult	84.2	83.8	75.2	75.3
Ionosphere	87.6	88.9	63.8	66.7
Wine Quality	88.0	85.6	86.6	84.4
Breast Cancer	97.3	98.2	61.8	71.9

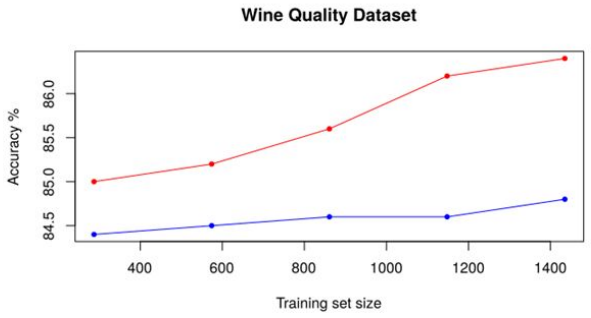
The Logistic Regression classifier achieves better accuracy than the Naive Bayes classifier on all datasets. The performance of both classifiers on the Wine Quality dataset is similar.

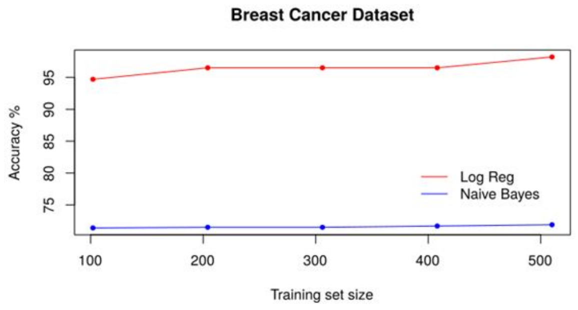
3.3 Accuracy of the Logistic Regression and Naive Bayes Classifiers as a Function of the Size of the Training Dataset

We shuffle each dataset and divide it into a training set (90%) and a test set (10%). We train each classifier with different training set sizes (20%, 40%, 60%, 80%, 100% of the original training set) and evaluate the accuracy on the test set. The accuracy of both Logistic Regression and Naive Bayes improves when the size of the training set increases.









4. Discussion and Conclusion

In this project, we implemented two classification techniques – Logistic Regression and Naive Bayes – and compared their performance on four datasets. Both classifiers were quick to train and achieved good accuracy with the Logistic Regression classifier constantly outperforming Naive Bayes. Both classifiers achieved better accuracy for larger training set sizes.

The gradient descent algorithm converged faster for larger learning rates. The highest accuracy was achieved with a learning rate that is neither too large nor too small. The accuracy improves when the number of iterations increases. Data normalization speeds up convergence and avoids numeric overflow when computing the logistic function. We also confirmed that k-fold CV is useful for finding the best hyper-parameter values for Logistic Regression as well as evaluating performance when comparing classifiers.

In the Naive Bayes classifier, the performance could be improved by restricting the floating-point stability for small values. Another improvement would be to implement the Bernoulli approach to generically calculate the posterior probabilities based on feature type. The Naive Bayes classifier is very fast and its performance can be improved by increasing the dataset size.

We highlighted the importance of data preparation and feature selection for achieving high accuracy. While all missing data were completely removed, other options (such as imputation) could be employed and result in an enhanced learning process. We developed a feature recommender – an automated approach for suggesting a good subset of features to train. As a future improvement, more criteria can be added to the recommender for categorical and continuous features. Also, the addition of new features through what is known as “feature engineering” in both of its approaches – transformations and aggregations – is another possible improvement in data preprocessing.

5. Statement of Contributions

Alhusain Abdalla implemented the loading, preprocessing, and analysis of the data. Rossen Vladimirov implemented the Logistic Regression classifier, k-fold cross-validation, accuracy evaluation, and feature recommender. Ege Yay implemented the Naive Bayes classifier.

References

1] Ng AY, Jordan MI. On discriminative vs. generative classifiers: A comparison of logistic regression and naive Bayes. In Advances in neural information processing systems 2002 (pp. 841-848).

2] Dua, D. and Graff, C. (2019). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

3] Jafferjee, A. (2019). Machine learning for data cleaning and unification.

4] Jeatrakul, P., Wong, K. W., & Fung, C. C. (2009). Using misclassification analysis for data cleaning. IWACIII 2009 - International Workshop on Advanced Computational Intelligence and Intelligent Informatics, (April 2014).