Classification Task Using Gaussian Naïve Bayes Algorithm

Why Naïve Bayes:

The classification algorithm used for the classification task of distinguishing between epithelial (where cancer cells live) and stromal (where immune cells and other normal cells live) regions is a method which uses the Bayes' Theorem as the basis, called the Naïve Bayes classifier. This algorithm is very robust which assumes that one feature present in a class is unrelated to the presence of any other feature in the same class (Ray, 2017). This classifier is easy to use and implement and is very scalable to the large datasets. Since we have multiple features and want to do a multi-class classification, this algorithm is very handy to use it performs well in this scenario. The number of data available with us is comparatively small and if we assume independence of features, then this classifier performs better than other classifiers like logistic and linear classifiers.

There are multiple variations of the Naïve Bayes classifier present, for example, the Gaussian Naïve Bayes, Multinomial Naïve Bayes, Bernoulli Naïve Bayes, etc. The Gaussian Naïve Bayes variation is used for the given classification task as we have used normal distribution and normalized each feature which is done easily by this variation of the classifier. The training time required for the dataset available in this classifier is also less using this classifier which is a known advantage of Naïve Bayes classifier.

The experiment performed:

The experiment performed using Naïve Bayes classification (Ray, 2017) is started by taking the input data sets for training and testing. The code uses pipelining (Pedregosa, 2011), normalization (Pedregosa, 2011) and principal component analysis (Pedregosa, 2011), and class balancing (Brendan Martin, n.d.) where different parameters are tunes for like the percentage of the variance to be maintained for the classification.

The training data is fit into the class balancing and the model is created using the balanced training data and then tested on the test data to come up with one output file containing classifications did which seem fairly accurate according to the Kaggle public scores (private scores pending at the time of writing up the report). Before choosing this classifier, several other classifiers were tried and tested on the training and test data, but their final output in terms of the normalization effect and the accuracy did not match up with the one achieved using Naïve Bayes algorithm. One more difference while experimenting the given data with different classifiers was that the number of mislabeled points in the whole dataset was the least in the chosen classifier as compared to the Multinomial variant, Bernoulli variant, Random Forest classifier, Linear Regression, Logistic Regression, and the decision trees. The parameter tuning during the principal component analysis to increase the performance of Naïve Bayes had a major impact on the final accuracy and the number of mislabeled points. The

variance to be maintained is given, based on which the optimum number of components are automatically chosen.

The performance of the Naïve Bayes algorithm is calculated on the total number of mislabeled points out of the total points present in the training set X and then going through the Y_test output to verify if each value in it matches the expected value in the set X or not. Then the total number of such points is calculated and its percentage is given as the accuracy of the algorithm. This same methodology is used as the basis for measuring the performance of all the algorithms evaluated during the evaluation phase, before choosing Naïve Bayes classifier. The training of the classifier is done using the sample files given on kaggle named X and y. The model training was also carried out using the standard way of splitting up the dataset into 30% and 70% data, but later on, commented. One notable observation during the observation was that the performance of the algorithm kept on improving as we decreased the percent of variance to be maintained of the dataset, but only up to the number about 81%, after which it started fluctuating again, before continuously decreasing. A near about similar performance was achieved for when the percentage of variance to be kept was 88, 85, and 79 during the pre-processing. Finally, if during the pre-processing of the dataset the percentage of variance to be maintained was not specified, the algorithm considers all of the features (112) and the performance and the accuracy achieved was not as high as earlier. This led to the conclusion that a better performance is achieved using only a subset of the features provided. The measure of performance throughout was considered as the total number of mislabeled points to the expected values and the score between X_train_res and y_train_res.

Experiment Explanation:

The classification algorithm carried out to classify the task uses various modules like numpy, imblearn (G. Lemaitre, n.d.), and sklearn (GaussianNB, StandardScalar, PCA, and make_pipeline) (Pedregosa, 2011). Data cleaning is applied using normalization of the dataset, where the mean of the distribution is zero and the variance of the distribution is 1.

Following this, the mean is subtracted from each value in the dataset and then divided by the standard deviation of the complete data. The principal component analysis is used to speed up the performance of the classifier. This is also used to specify the percentage of the variance that is to be retained before training the model and classification task.

The other possibility of using PCA is to specify the number of features that are to be chosen. This methodology is not followed using this algorithm and instead, the variance percentage retainment is specified. Based on this choice given, PCA then chooses the optimal number of features for retaining this variance.

The task of normalization, PCA and application of the Gaussian NaïveBayes is calculated on after the other where all three are applied using make_pipeline (Pedregosa, 2011) in which the final estimator's work is done using the Gaussian Naïve Bayes algorithm. The class balancing is done using the library from imblearn

(Brendan Martin, n.d.). This is necessary so that the number of negatives is somewhat equal to the number of positives. This is done using smote, which creates sample data to train the minor classes which are the positive ones. The model is then trained using the X_train_res and y_train_res training datasets. The prediction is done on the X_test_actual which yields an output of y_pred. This final output is then measured against accuracy by calculating the number of mislabeled points out of the total points (removed from code as only a print statement). The y_pred is a CSV file contains the header line and the points with their classifications. This is submitted to the kaggle competition which gives an accuracy of 0.82845.

Literature Review: Auto-Encoding Variational Bayes:

This paper proposes a way for training graphical models with unmeasurable posteriors and continuous latent variables. The reparametrization of latent variables is done so that a deterministic mapping can be devised. According to the authors, it enables the lower bound estimators to be different in comparison to the variational parameters. The previous work and existing models have been considered in brief and explanations have been provided. These include the Generative Stochastic Networks, Denoising Auto-Encoder theory and wake-sleep algorithm, which has a scope of improvements (Kingma, 2013).

The authors have a proposed an application of auto encoding variational Bayes where the period is a Gaussian centered on zero and the mean of the distribution is determined by a neural network output. According to the authors, this addresses the parameter estimation problem in a graphical model. The proposed method has a fast training to get the parameters for data generation, an approximation of the posterior for data representation and approximation of the marginal for the evaluation of the model for other activities and models. The authors have discussed various noise distributions, but have not compared them in detail.

The quality of supporting evidence provided by the authors seems fair and extensive. Multiple experiments are run on both Frey-faces dataset and the Most dataset (Kingma, 2013). The authors have compared the new lower bound with the wake-sleep algorithm, the new marginal likelihood with the likelihood of the wake-sleep algorithm. The visualization of 2D manifolds learned with the proposed solution and outlining of the samples got by sampling the dataset from the generative model is also shown by the authors in their study. However, they have not detailed much about the overfitting problem: when does it occur and how is it being handled by the new solution. The result of the experiments also reveals that the proposed model can improve the wake-sleep algorithm.

References

1. Brendan Martin, N. K., n.d. *LearnDataSci - Predicting Reddit News Sentiment with Naive Bayes and Other Text Classifiers.* [Online]
   Available at: https://www.learndatasci.com/tutorials/predicting-reddit-news-sentiment-naive-bayes-text-classifiers/
   [Accessed 26 November 2018].

2. G. Lemaitre, F. N. D. O. C. A., n.d. *imblearn.over_sampling.SMOTE.* [Online]
   Available at: https://imbalanced-learn.readthedocs.io/en/stable/generated/imblearn.over_sampling.SMOTE.html
   [Accessed 26 November 2018].

3. Kingma, D. P. W. M., 2013. Auto-Encoding Variational Bayes. *arXiv:1312.6114,* Issue arXiv:1312.6114.

4. Pedregosa, F. V. G. G. A. M. V. T. B. G. O. B. M. P. P. W. R. D. V. V. J. P. A. C. D. B. M. P. M. a. D. E., 2011. Scikit-learn: Machine Learning in Python. *JMLR 12,* pp. 2825-2830.

5. Ray, S., 2017. *6 Easy Steps to Learn Naive Bayes Algorithm.* [Online]
   Available at: https://www.analyticsvidhya.com/blog/2017/09/naive-bayes-explained/

6. Ray, S., 2017. *Essentials of Machine Learning Algorithms.* [Online]
   Available at: https://www.analyticsvidhya.com/blog/2017/09/common-machine-learning-algorithms/