DATA 602 @ UMBC

Homework Assignment 2

Note: Please work out the following problems in one single jupyter notebook in a coherent manner. For illustrations and questions which do not involve coding, use markdown cells.

1. **Unsatisfactory Performance of Logistic Regression:** Suppose you are implementing a simple logistic regression to solve a binary classification problem but you find unsatisfactory results.

   (a) What actions on your **dataset** would you consider to take in order to improve the performance of the model?

   (b) What would you consider to try for the **existing algorithm** to improve the performance of the model? **Hint:** Think about hyperparameters.

   (c) What other algorithms would you consider to try in this case? Why? Explain.

2. **Hybrid Naive Bayes:** In lecture 6, we introduced two types of naive Bayes classifiers, namely the classical naive Bayes classifier and the Gaussian naive Bayes classifier. As we saw, the former is applied when the features are categorical, whereas the latter is applied when features are continuous. With a simple modification, we can build a naive Bayes classifier when both categorical and continuous features are present. We refer to this classifier as the *hybrid naive Bayes classifier*. Suppose in a classification problem with K classes, we have d_1 categorical features, denoted by $Z = (z_1, z_2, \cdots, z_{d_1})$, and d_2 continuous features, denoted by $\vec{x} \in \mathbb{R}^{d_2}$. As in classical and Gaussian naive Bayes classifiers, the hybrid naive Bayes classifier assumes independence among features. This allows us to take the advantage of formula (8) in lecture 6, and predict the class of the target in the hybrid case by a similar formula

$$\hat{y} = \underset{i \in \{1,2\cdots,K\}}{\operatorname{argmax}} \Big(p(Z = Z^*|y = i)\, p(\vec{x} = \vec{x}^*|y = i)\, p(y = i) \Big) .$$

   In other words, in order to predict the class of the target variable, we can calculate the likelihoods of the categorical and continuous features separately, and multiply them due to their independence. That is the only change we need to impose for the hybrid naive Bayes classifier to work. A library, called Mixed Naive Bayes, has been written to implement naive Bayes classification to hybrid datasets. Visit the Github repository https://github.com/remykarem/mixed-naive-bayes to see how the mixed naive Bayes is implemented.

   (a) Build a mixed naive Bayes classifier for the Heart Disease Dataset used in lecture 6. In this implementation, consider all 13 features (Recall that in this dataset, 8 features are categorical and 5 features are continuous).

   (b) How do you evaluate the performance of the mixed naive Bayes classifier in part (a)? How do you compare the performance of the mixed naive Bayes classifier with the classical and Gaussian naive Bayes classifiers constructed in lecture 6?

(c) Is there a straightforward generalization of LDA/QDA classification algorithms to hybrid datasets? Why? Explain.

3. **LDA vs. QDA:** In this problem, you will compare the performances of the LDA and QDA classifiers on a given dataset. The goal is to find out what classifier (between LDA and QDA) is most appropriate to be applied in different cases. In this problem proceed as follows.

   (a) Download LDA-QDA-Toy-Data.csv and read the csv file into a dataframe. Regard x_1 and x_2 columns as continuous features, and the y column as a binary categorical target. Split the data into train and test with test_size=0.25.

   (b) Train LDA and QDA classifiers on the train subset established in part (a). Find the predictions of both classifiers for the test subset.

   (c) Calculate the relevant accuracy metrics by presenting the classification_report for both train and test subsets. Assess the two classifiers and contrast their performances.

   (d) Use plot_decision_regions from the plotting module of the mlxtend library to plot the decision regions for the two classifiers in the two dimensional feature space formed by x_1 and x_2 (revisit the "Binary Example" of lecture 4 to see how decision regions are plotted).

   (e) Based on your findings in parts (c) and (d) and your literature search, explain in what situations it would be more appropriate to use QDA classifier. In what situations is the LDA classifier the more appropriate choice of classifier?

4. **Random Forests vs. Gradient Boosting:** Contrast and compare random forest and gradient boosting classifiers from the point of view of

   (a) internal structures of the algorithms

   (b) being prone to overfitting

   (c) tuning hyperparameters

   (d) execution time to train

5. **Classification Analysis:** For this problem, use the csv file GateArrivalDelay-Classification.csv.

   In this problem, you will employ several different classification algorithms on the data published by FAA to predict whether a gate will be experiencing arrival delay at the ORD airport. The target variable in this problem is the Class which is a binary categorical variable ($y = 0$ corresponds to no delay, whereas $y = 1$ corresponds to an average of 15 minutes or more delay for a gate at ORD airport). Potential features are

   - % On-TimeGateArrivals
   - AverageBlockDelay

- AverageTaxiInDelay
- AverageAirborneDelay
- AverageAirportDepartureDelay
- AverageGateDepartureDelay

(a) Determine the nature of the classification problem. Is the current classification problem closer to a balanced classification or an unbalanced classification problem?

(b) Construct a random forest classifier for the target Class in the presence of all 6 features. Calculate the relative feature importance from the constructed forest and identify the two most significant features.

Note: For the rest of the problem, focus on the two features identified in part (b).

(c) Construct a logistic regression classifier for the classification problem. Calculate various relevant accuracy metrics, and assess the performance of your constructed classifier. What is the *AUC* of the corresponding ROC curve for this binary classifier? Plot decision regions in the feature space.

(d) Construct a support vector classifier to perform the classification task. Now use a couple of different kernels to construct support vector machines for the classification task at hand. In each case, calculate the accuracy metrics and *present your assessment of each classifier*. Define a range for the relevant hyperparameters and employ GridSearchCV to find the optimal choice of hyperparameters in the specified range. Rerun the classifiers with the optimal choices of hyperparameters and compare the performance of your classifier with the case where hyperparameters were not tuned. Plot the decision regions in the feature space.

(e) Construct a Gaussian naive Bayes classifier for the classification task at hand. Calculate the relevant accuracy metrics and plot the decision regions in the feature space. Investigate whether (and to what extent) the underlying assumptions of the naive Bayes algorithm are fulfilled. Explain the validity of the assumptions in this case, and explain whether you trust the results obtained from this classifier.

(f) Construct an LDA classifier and carefully assess its performance. Plot the decision regions in the feature space.

(g) Construct a decision tree classifier for the classification task at hand. Initially do NOT tune any of the hyperparameters of the tree. Does your tree suffer from overfitting? Use GridSearchCV and tune some of the hyperparameters of your tree to cure the high variance. Calculate the relevant accuracy metrics after tuning the hyperparameters of your tree. How successful were you in curing the high variance?

(h) In this part, you will apply some of the ensemble learning methods discussed in class. Proceed as follows:

i. Construct a random forest classifier (with the same two features), and try to tune the hyperparameters of the forest so that you'll find a more accurate model than the tree classifier constructed in part (g).

ii. Employ gradient boosting to construct a classifier for the same classification problem. Again try to come up with a more accurate classifier than those constructed earlier.

iii. One of the prominent ensemble learning methods is **stacking**. Recall, that in bagging and boosting methods, the weak learners (*i.e.* building blocks) are all models of the same type. However, in stacking, one can combine the results of several different types of weak learners to fortify the predictions. For this purpose, one trains a simple model to find best way of stacking/combining the results of different weak learners. This useful ensemble method is already built in scikit-learn. Visit scikit-learn page `https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.StackingClassifier.html` to learn about the implementation of the stacking method. Take three of your best classifiers you constructed in the previous parts and stack them through StackingClassifier (you'll feed your best classifiers through the estimators parameter). Employ a simple logistic regression (for the choice of final_estimator) to learn an optimal way of stacking your three best classifiers. How does stacking improve your results?

(i) In this problem, if your focus is on the $y = 1$ class (*i.e.* delayed gate arrivals), which classifier among all classifiers you constructed in previous parts gives you the *highest f1-score* for this class?

6. **Ensemble Methods Applied to Regression Problems:** Consider the same dataset (GateArrivalDelay.csv) in problem 4 of HW 1. Use one of the ensemble learning methods for regression (*e.g.* random forest regressor, gradient boosting regressor, extreme boosting regressor) to construct a model whose accuracy (R^2-score) beats the highest accuracy you could achieve in HW 1. Make sure that the model you construct in this problem is *not suffering from a severe high variance problem.*