IN4320-Machine Learning Final Project Report

Jihong, Ju
Student ID: 4518454

1 Introduction

The classification task is one of the most important problem in the Machine Learning area. Most problems in practice can be converted to a classification problem eventually. In this report, we will use a large dataset with high percentage of values missing in order to experiment various imputation and classification methods.

The structure of this report is as following: Section 2 demonstrates an statistical analysis of the dataset and the preprocessing procedures; Section 3 shows the experiments on missing value imputation methods and classification models, and discusses about the corresponding results obtained. The discussion about improvement and future works goes to the Section 4. The final section (Section 5) gives the conclusions draw from the analysis and experiments.

2 Preprocessing

The dataset of our problem contains thirteen different features including both continuous and categorical features, which are summarized in Table reftable:features. Even though the categorical features have numerical values in the dataset, they do not have an actual numerical meaning. In this project, all the categorical features are represented as binary vectors. By doing this, each category of the categorical features generate additional features with binary values 0 or 1. This method constructs a dataset of 104 features instead of the original 13 features.

It is noteworthy that after introducing binary vectors for the categorical features, this dataset becomes more sparse which requires the classification methods to handle with sparse matrix in an both effective and efficient way.

Another challenge that we confronted with is the missing values in the dataset. There are dozens of methods and techniques available in order to handling with missing values. Since these methods are based on different assumptions, it might be helpful if we could know the mechanism of how the values are missing before we apply any imputation methods.

Rubin [1] defined the three types of missing data: Missing Completely at Random, Missing at Random and Missing Not at Random as follows: Missing Completely at Random (MCAR) means that missingness does not depend on the neither observed values nor missing values of any variables in the data set,

Name	Type
age	continuous
work class	categorical
education	categorical
number of years of education	continuous
marital status	categorical
occupation	categorical
relationship	categorical
race	categorical
sex	categorical
income from investment sources	continuous
losses from investment sources	continuous
working hours per week	continuous
native country	categorical

Table 1: Thirteen original features and their types

whereas Missing at Random (MAR) means that the propensity of missing values are only correlated to the observed data and not the missing data. Missing Not at Random (MNAR), on the other hand, refers to whether values are missing is corresponding to its values.

Different missing mechanisms lead to different assumptions that can be made by the imputation methods. For example, Multiple imputation and Maximum Likelihood assume the data not to be MNAR, whereas List-wise deletion, requires the data are MCAR in order to avoid introducing bias in the results.

Because of the missing values of data, it is difficult to convince us in which mechanism data is missing. However, we can examine patterns in the observed data and estimate what is the most likely missing mechanism. We diagnosed the missing mechanism of the data in an alternative method: first create dummy variables where 1 indicates the value is missing and 0 indicates not missing. A Chi-square test is then run between this dummy variable and other variables pairwisely to see if the missingness of values is related to the values of observed data. The results gave that all the feature pairs fail to reject the null hypothesis (MCAR) which indicates that the values are likely to be missing completely at random. We might benefit from imputing missing values based on what we have observed. However, the chi-square test is just an rough estimator of the missing mechanism. The data is still possibly MAR or even MNAR.

Table 2 shows that occupation and work class have similar amounts of additional values missing compared with the other features. This is less likely to be just a coincidence sine apparently these two features have their intrinsic correlation. In other words, people who do some particular jobs may refuse to report their occupation as well as their work class. The only way to justify this hypothesis is to measure some of the missing data for these two features which is impractical in our case. Here we do not take the hypothesis of Missing Not at Random and assume that the data is missing completely at random

(MCAR). There is a risk of introducing bias by assuming MCAR, yet the bias introduced may be compensated by the gain of imputation. We will get back to this discussion in Section 3.1 with some experiments.

Feature	Number of values missing
age	9188
education	9148
income from investment sources	9089
losses from investment sources	9191
marital status	9163
native country	9843
number of years of education	9098
occupation	11302
race	9414
relationship	9373
sex	9024
work class	11429
working hours per week	9194

Table 2: Number of values missing for each feature

Apart from the type of the values, the scales of values also varies widely from feature to feature in the dataset. For example, age varies from 5.4 to 28.6 whilst income from investment sources varies from 0 to 31831. For classifiers based on distance in the features space like nearest neighbor classifier, the scale difference among features may cause bias to the objective function since scales of the feature do not intrinsically reflect the importance of features in this dataset. Therefore, normalization for independent features is important in these cases.

3 Experiments

3.1 Missing Values

Given the hypothesis of MCAR, we are able to apply imputation methods such as Data deletion, Random imputation, mean or mode imputation, etc. Apart from these parametric free methods, one can also implement Multiple Imputation with a regression (and/or classification) model, or maximize a likelihood to fit an underlying data distribution to predict the values missing. As we have mentioned in Section 2 and will further discuss in Section 3.2.1, it is difficult to find a appropriate data distribution model for the combination of numerical and categorical features. We will just implement a Multiple Imputation by Chained Equation (MICE) [2] but not the Maximum likelihood imputation in for this experiment. The pseudo code of MICE is given in Algorithm 1.

Data: data with missing values
Result: data with missing values filled by regression (and/or classification) model
Impute the missing values with mean or mode;
while *not converged* **do**
  for *each feature* **do**
    reset the missing values to NaN;
    fit the regression (and/or classification) model with the data not missing ;
    predict the values missing ;
    fill in the missing values with the prediction of regression (and/or classification) model ;
  end
end

Algorithm 1: Multiple imputation by Chained Equation (MICE)

The performance of missing values for different methods is simply evaluated by the performance of a benchmark classifier named XGBoost. It might be more starightforward to evaluate the filling values if we randomly drop available values and justify the values filled by imputation. However, the simulating method takes the assumption that the values are missing completely at random which is risky. Therefore, we made the decision to use indirect way of estimating the performance of imputation. The results of five imputation methods are shown in Figure 1a

The XGBoost default missing values handler has the best performance among the five imputation strategy mentioned above which is not surprised. The reason for that will be explained in Section 3b when we discuss about the XGBoost classifier. MICE with Gradient Boosting Regressor and Classifier, even though is not as good as XGBoost default, still shows a relatively high accuracy compared with the other three methods. Those three methods achieve similar accuracy except that the Mode imputation seems to be more robust than the other three methods. Since both XGBoost default and MICE achieved better performance than data deletion, it seems that most values are truly missing completely at random as we indicated in Section 2.

3.2 Classifiers

In attempt to make a good classification of the two classes, we have tried multiple different classifiers provided by the Python machine learning library scikit-learn: Linear Discriminant Analysis, Nearest Neirghbor, Multinomial Naive Bayes, Supported Vector Machine (SVM), Decision Tree, Random Forest Classifier, Adaboost Classifier and Gradient Boosting Classifier together with its alternative implementation XGBoost. The missing values were filled with the most common values for each feature which might be sub-optimal yet help avoid the influence of assumption taken by using the other imputation methods like Maximum Likelihood Imputation. Each classifier has been tuned using grid search

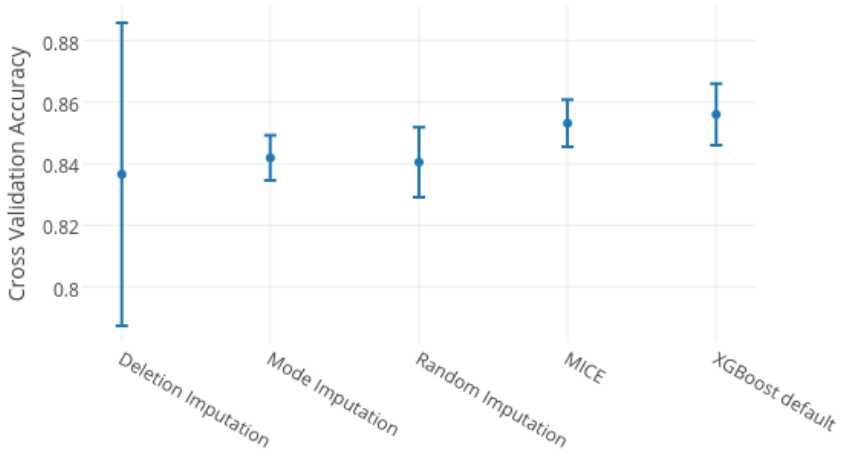


Figure 1: Cross Validation Accuracy for imputation methods

according to 10-fold cross validation error to optimize its performance such that the comparison is based on the optimized performance the classifier may achieve with the 104 features data inputs generated in Section 2.

The 10-fold cross validation accuracy with standard deviation for these classifiers is shown in Figure 2a. It turns out that SVM and ensemble methods have a better performance in general than the others. The rest of this section will discuss about the differentiation among these classifiers and the accordingly performance difference in our case.

3.2.1 Discriminant Analysis and Naive Bayes

Both Linear Discriminant Analysis (LDA) and Gaussian Naive Bayes (GNB) are based on the assumption of multivariate Gaussian distribution. The two major differences of the two classifiers are: (a) the two classes share the same covariance matrix in LDA but not in GNB; (b) GNB assumes the features are independent to each other, leading to diagonal covariance matrices for both classes.

LDA benefits from the assumption taken in (a) because the dataset is sparse such that they are, to some extent, linear separable. GNB, on the other hand, does not benefit from its assumption since the features are not completely independent. For example, education and number of years of education have apparent inherent correlation. Both LDA and GNB suffer from the underlying Gaussian distribution assumption. It can be seen from the learning curves in

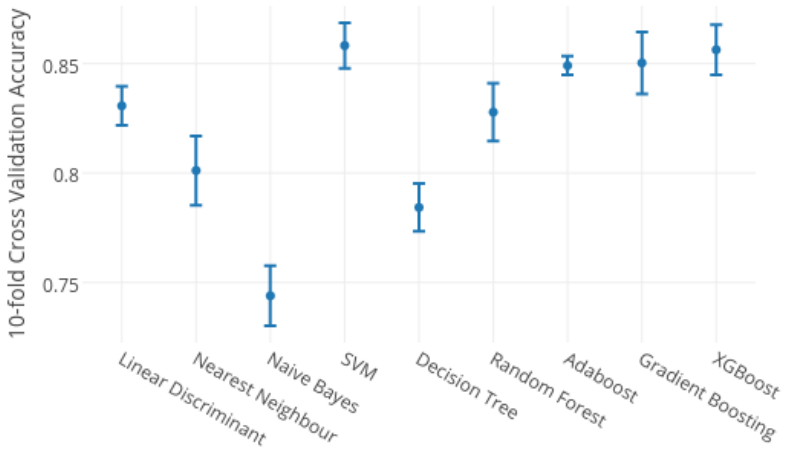
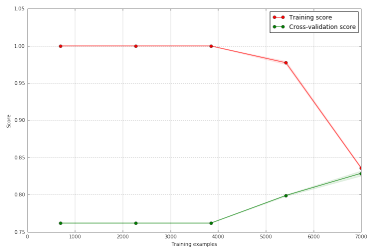


Figure 2: Cross Validation Accuracy for the selected classifiers

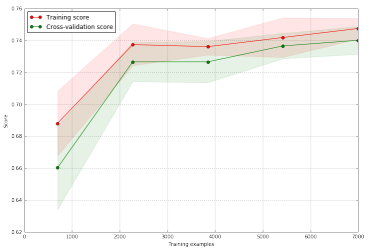
Figure 3a that both classifiers have higher bias than variance. We have also tried Quadratic Discriminant Analysis (QDA) which does not have the two restrictions mentioned above to the distribution assumption. It gives a 0.29 accuracy and a 0.13 standard deviation. The poor performance of QDA indicates the failure of Gaussian Distribution Assumption for the two classes in this problem. Actually, we have already known from Section 2 that our features contain both continuous and categorical data such that the two classes are not exactly Gaussian distributed. Instead, the binary representation may lead to a multivariate Bernoulli distribution of the categorical features. This explains the relatively poor performances of Discriminant Analysis and Naive Bayes methods being used. A combination of Gaussian distribution and Bernoulli distribution model, Bernoulli-Gaussian model [3], might be more appropriate to use the Bayesian model and Maximum A Posteriori (MAP) estimation in this case. However, the complexity of the combined distribution model requires a Gibbs sampler to generate distributed according to the posterior distribution of interest.

3.2.2 k-Nearest Neighbor (k-NN) and Support Vector Machine (SVM)

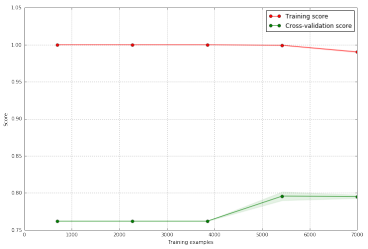
Different from Discriminant Analysis or Naive Bayes, kNN attempts to approximate the underlying distribution of the data in a non-parametric fashion. The underlying assumption of k-NN is that the two classes are distinguishable in the feature space. They cannot be highly overlapped, otherwise k-NN is likely to overfit. Over-fitting is indeed the problem of k-NN for classifying people who



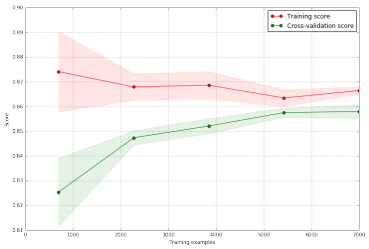
Linear Discriminant Analysis



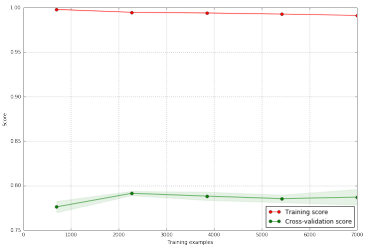
Gaussian Naive Bayes



Nearest Neighbor

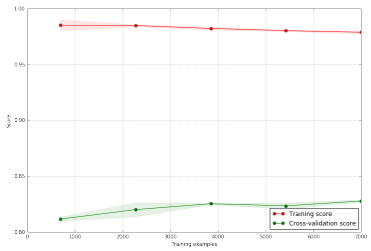


Support Vector Machine

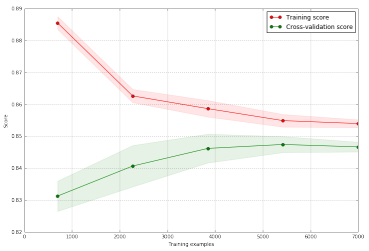


Decision Tree

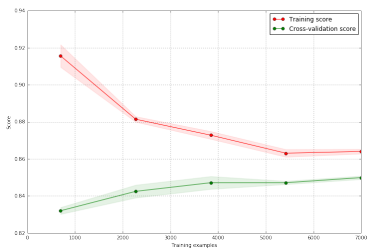
(a) Single Classifiers



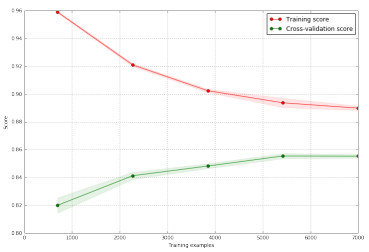
Random Forest



Adaboost



Gradient Boosting



XGBoost

(b) Ensemble Classifiers

Figure 3: Learning Curves for the single classifiers and the ensemble classifiers

earn over 40 thousands euros a year from who do not.

Instead of searching for decision boundary in the feature space itself, SVM assumes there exist a hyperplane separating the data points of different classes. By finding the hyperplane with the largest margin, SVM is able to separate the two classes more robustly, effectively avoiding overfitting. As a evidence, we observed a steady increase and steady decrease for the training accuracy and the cross validation accuracy respectively for SVM along with the number of samples increased in Figure 3a, whereas the two curves are far separated for Nearest Neighbor Classifier. The kernel used is the Radial Basis Function (RBF) kernel which appears to well capture the differentiation of the two classes. Moreover, SVM's success may be also ascribed to the sparsity of the dataset because it attaches more importance to the support vectors which sit on the edge of the margin.

3.2.3 Ensemble Classifiers

In order to improve generalizability/robustness over a single estimator, one usually applies ensemble methods to combine the predictions of several base estimators. The reason for this is that different base estimators may have different performance in certain regions of the dataset. The combination of various classifiers may combine the strength of each base classifier and drop the deleterious predictions such that the combined classifier is more generalizable and robust. The most commonly used ensemble methods can be divided into two groups: Bootstrap aggregating (a.k.a bagging) and boosting. In this section, we will try several different bagging methods and boosting methods respectively.

The base classifiers used in this section is the decision tree classifier. It is because the decision tree classifier does not require a data distribution model which is difficult to find one for our numerical-categorical combined features. Decision tree classifier is simple and fast so that iteratively running bagging or boosting do not introduce significant more time complexity. The problem of decision tree is also straight forward, it can be easily overfitted especially when the maximum depth and minimum population in leaf is not well tuned (Figure 3a). However, overfitting problem can be compensated by the ensemble methods as they are supposed to do, and the complexity of the tree can be tuned using grid search based on cross validation errors.

The first ensemble classifier is the Random Forest Classifier which samples with replacement when building each base classifiers and averages the prediction by each bass classifiers to give the final prediction. With Figure 3b and Figure 2a, we demonstrated that the Random Forest Classifier increases the overall cross validation accuracy but does not well solve the problem of overfitting since there is still a significant gap between the training score and cross validation score. Besides, the cross validation accuracy seems to has already reached the plateau despise of the increase of number of samples.

AdaBoost, short for Adaptive Boosting, iteratively learns weak classifiers while tweaking the weights of instances misclassified by the previous classifiers and build the final classifier by a weighted sum. AdaBoost has proven itself to

be less susceptible to the overfitting problem than other learning algorithms in some problems. [4] show that our case is one of those problems in Figure 3b. There is a dramatic decrease of training scores along with the increase of number of samples. The downside of AdaBoost rises from the adaptive "tweaking" of the classifiers, if samples are mislabeled or the inputs are noisy, the final classifier tends to be deceived by these noisy data and outliers. Remember that we filled in the missing values with the most common values for each features, inevitably introducing bias to our dataset. This may be one of the reasons that the performance of AdaBoost fails to keep improving when the number of estimators increases. Instead, too many number of iterations may introduce overfitting again. The final optimal number of estimators is a balance between high variance and high bias.

Gradient Boosting is another boosting method that estimates an approximation of $F(x)$ in the form of weighted sum of weak learners $h_i(x)$. It find the approximation F(x) by minimizing the expected value of an objective function $L(y, F(x))$ where y is the label and F(x) is the prediction. Unlike Gradient Descent, the gradient boosting algorithm does not change the parameter of the existing weak learners $F_{m-1}(x)$ in any way. Instead, it improves by adding another classifier $h_{i+1}(x)$ and incrementally expanded:

$$F_m(x) = F_{m-1}(x) + \arg\min_f \sum_{i=1}^{n} L(y_i, F_{m-1}(x_i) + f_m(x)) \tag{1}$$

where the existing classifier $F_{m-1}x = \sum_{i=1}^{m-1} \gamma_i h_i(x)$ and h and γ are the base estimator and its weight respectively; $L(\cdot)$ is the loss function which is usually mean square error for classification problem [**?**]

The new added base estimator can be fit to the pseudo-residuals, the derivative of the loss function with respect to the existing classifier $F_{m-1}(x)$, with the training set. The multiplier γ_m is then solved by minimizing the loss function above to construct the new base classifier $f_m(x)$ and update the existing classifier additively using Equation 1.

XGBoost [5] is a modified implementation of the Gradient Tree Boosting model. Apart from the efficiency improvement, one significant difference between XGBoost and the normal Gradient Boosting is the introduction of Regularization term which restrict the complexity of the tree in the objective Function. The objective function in XGBoost contain two parts: training loss and regularization. In XGBoost, the training loss function can be either mean square loss or logistic loss as the normal Gradient Boosting; the regularization term is defined by:

$$\Omega(f) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} \omega_j^2 w \tag{2}$$

here T is the number of decision tree leaves and ω_j is the vector of scores on leaves for the j-th tree. The regularization term protect the classification model from overfitting problem, leading to a more robust classifier. Figure 2a and Figure 3b demonstrate that XGBoost not only has a higher accuracy compared

with the other ensemble method, it also achieves a narrower standard deviation for the cross validation scores.

Another advantage of XGBoost is that it can handle the missing values by default as we have demonstrated in Section 3.1. The corresponding mechanism is simple: In each node of the new added base tree, a default direction for the missing values will be found by minimizing the objective function. This method relies on the assumption that the samples with missing values in the same node tend to be similar. As long as the number of missing values is not dominant in one node, this assumption is likely to hold when the data is missing at random (MAR) where the missingness of data is related to the observed data. This method can handle the case where multiple features have missing values because the missing values are treated as a special value in each node with the optimal default direction.

4 Discussion

Our dataset contains 38342 unlabeled samples in the test set. If we could make use of the patterns in the test set, we may be able to improve the performance. Since it is hard to find a parametric distribution model, we do not stick to the generative models of semi-supervised learning. Instead, we tried to implement a simple algorithm that iteratively includes samples from the test set with previous predictions as the labels to train the classifier until converged. The algorithm is shown in Algorithm 2

Data: training set with labels and test set without labels
Result: test set with labels
Fit the classifier with the training set;
Predict on test set;
Update the labels of samples in the test set using predictions;
while *not reach maximum iterations* **do**
  Random sampling from test set;
  Fit the classifier with the training set and sampled test set ;
  Predict on the test set ;
  for *samples in the test set* **do**
    if *the probability for prediction is higher than the probability of the current label* **then**
      Update the label using prediction
    end
  end
end

Algorithm 2: Transductive learning for XGBoost

In order to evaluate the gain from transductive learning using Algorithm 2, we divided the training set into a training subset and a test subset with ratio 1:4. The performance improvement turns out to be insignificant. This may be ascribed to the introduction previously misclassified samples to the learning process. A prediction with higher probability does not necessary lead to a

higher probability of being correct because the classifier may be kept by the misclassified data since the gradient boosting method is intrinsically sensitive to the noisy data and outliers.

Instead of using the unreliable samples from test set to generate different classifiers, we could use different classification models and ensemble their predictions. We have seen in Section 3.2.3 that the ensemble techniques improved the performance of the classifiers dramatically. The ensemble methods implemented in Section 3.2.3 are based on the weak classifier, decision tree. It is also possible to aggregate the predictions from various strong classifiers in order to further improve the performance of classification. The combining scheme can be either majority vote or weighted sum. The improvement is not guaranteed but as long as the classifiers are not similar, there exists an possibility of improving performance by aggregating multiple classifiers.

5 Conclusion

We have tried multiple imputation methods and classification methods on a large dataset with both numerical and categorical data. Multiple Imputation by Chained Equation (MICE) and the default missing value handling mechanism of XGBoost turns out to perform the best among all the five imputation methods within our experiment settings. SVM and Gradient Boosting methods were the best performed ones among selected classifiers with a mode imputation schema. The difficulty of finding a parametric data distribution model leads to relatively bad performance of Bayesian model for classification. Further improvement of the performance this model may arise from an combination of the Gaussian-Bernoulli distribution. A transductive learning method was proposed whereas it did not introduce any improvement of the performance.

References

[1] Donald B. Rubin. Inference and missing data. *Biometrika*, 63(3):581–592, December 1976.

[2] Melissa J. Azur, Elizabeth A. Stuart, Constantine Frangakis, and Philip J. Leaf. Multiple Imputation by Chained Equations: What is it and how does it work? *International journal of methods in psychiatric research*, 20(1):40–49, March 2011.

[3] C. Bazot, N. Dobigeon, J. Y. Tourneret, and A. O. Hero. A Bernoulli-Gaussian model for gene factor analysis. In *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5996–5999, May 2011.

[4] Yoav Freund and Robert E. Schapire. A desicion-theoretic generalization of on-line learning and an application to boosting. In Paul Vitányi, editor, *Computational Learning Theory*, number 904 in Lecture Notes in Com-

puter Science, pages 23–37. Springer Berlin Heidelberg, March 1995. DOI: 10.1007/3-540-59119-2_166.

[5] Tianqi Chen and Carlos Guestrin. XGBoost: A Scalable Tree Boosting System. *arXiv:1603.02754 [cs]*, March 2016. arXiv: 1603.02754.

6 Word Counts

Total words: 3465