Prediction of Adult income

Master of Data Analytics

## Project report CISC 6930: Data Mining

Fall 2017

Submitted by:

Professor & Advisor: Yijun Zhao

Fordham University

**Index**

|  |  |
| --- | --- |
| Content | Page |
| 1. [Overview](#Overview) | 3 |
| 1. [Problem Statement](#Problem_Statement) | 4 |
| 1. [Data Exploratoryanalysis](#Data_analysis) |  |
| 3.1 missing values  3.2 Unbalanced  3.3 Correlation  3.4 Feature types |  |
| 1. Ensemble Models 2. Trial Run |  |
| 5.1(Assumption) Drop Missing Values, Ignore Unbalance, Dummy Encoding  5.2 Results |  |
| 1. [Data Pre-processing](#Data_Preprocessing) |  |
| * 1. Data Encoding   2. [Imputation](#Data_imputation)   3. Bagging   4. Normalization (w Bagging Iteration)   5. Parameter Optimization   6. Feature Selection |  |
| Optimal Model |  |
| 1. [~~Training the model using data mining algorithms~~](#Algortithms) |  |
|  |  |
| 1. [~~Model selection (Ensemble Method)~~](#Model_selection) |  |
| 1. ~~Model application on testing set~~ |  |
| 1. ~~Conclusion~~ |  |
| 1. Reference documents links |  |
| 1. Reflection (Future Work) |  |

1. ***Overview***

**Project Description**

This Data Mining project uses the real-world dataset extracted from 1994 census bureau database to predict annual incomes of adults, given a set of attributes like employment details, demographic information etc.

The income level is classified in two classes – less than 50,000 and greater than or equal to 50,000.

The project uses data various data mining algorithms and are written in python and R.

**Summary of results**

1. ***Problem Statement***

Using the given dataset with attributes, the aim is to build a predictive model that determines income level for adults. Income levels are classified in two classes below $50K and above $50K annually (given in the dataset).

1. ***Data Exploratory Analysis***
2. Missing values:

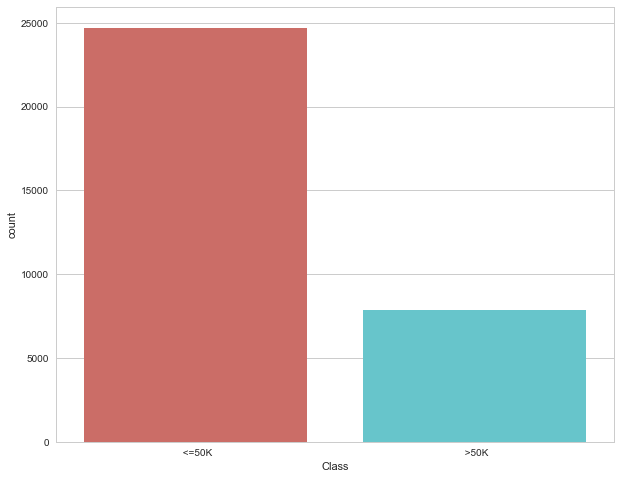
After a preliminary exploration of the census data, it is found that both training and testing data sets contain missing value. For training and test data alike, all of the missing values were found to be confined in three categorical values: *native\_country, workclass and occupation*.

7.45% of training data instances, that is 2399 rows, contained missing values whereas 7.5% of test data instances, that is 1221 rows, contained missing values.



1. Unbalanced data:

Training data set was unbalanced with a negative skew, where 77% of the instances were classified as negative and only 23% were classified as positive.

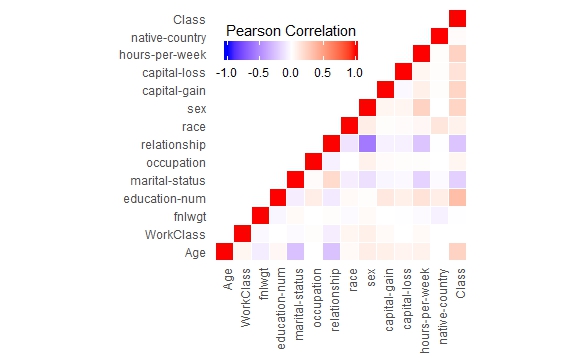


1. Types of variables:

There are three types of variables found in the data set

1. Continuous: fnlwgt
2. Categorical:
3. Ordinal: Education-num
4. Feature correlation

The plot shows the correlation between the features and class variables.



1. ***Data Pre-processing***
   1. ***Drop Missing values***
   3. Data cleaning and Imputation for missing values:

The training and testing datasets have data that is recorded as ‘?’. This needs to be managed so that it does not exert undue influence on the model. Two approaches are used for data cleaning and imputation

*A: Drop all the missing values*

In this approach, all missing values are dropped. The data is z-score normalized and ignored the unbalanced nature of the data in order to run preliminary tests and set a baseline for the classifier performance.

*B: Imputation*

To handle the missing training data, there are two categories of techniques exists, model based and non-model based approaches. Non-model based techniques includes mean imputation and hot-deck imputation. These techniques generally decrease the variance estimates in statistical procedures. Furthermore, these techniques also result in standard errors and bias in results. On the other hand, model based approaches includes data mining algorithm techniques to predict the missing values. (For ex - Regression model, decision tree, NB etc). This approach results decreasing the variance as well as bias.

For our project, we used three methods to impute the missing data which includes model based and non-model based techniques.

1. **Mode**

This is the example of a non-model based approaches. According to this approach, we fill the missing feature values with the most frequent data for the respective feature.

1. **Random-Forest Based Imputation**

This is another example of a model based approaches where multiple D-trees are built which contains information corresponding to attributes in the given dataset. This information is used to follow a given set of input attribute, depending on categorical nodes. After which, "Random forest" algorithm is used to generalize ensembles of D-trees through bagging which combines multiple random D-trees to aggregate the prediction for missing data.

For our implementation, we used a built-in package of R (missForest) for Random Forest based imputation. In missForest package, for each feature (Having missing values), it builds the random forest based on the given observations and then predicts the missing values. The algorithm continues to repeat these two steps until stopping criteria is met or the user specific maximum of iterations is reached.

According to default stopping criteria, after each iteration the difference between the previous and the new imputed data matrix is assessed for the continuous and categorical features. The default stopping criteria is defined such that the imputation process is stopped as soon as both differences have become larger once. In case of only one type of variable the computation stops as soon as the corresponding difference goes up for the first time. However, the imputation last performed where both differences went up is generally less accurate than the previous one. Therefore, whenever the computation stops due to the stopping criterion the before last imputation matrix is returned. In our case, we only had the missing values in categorical features, therefore second procedure applies.

Additionally, missForest algorithm provides the user with an estimate of the imputation error. This estimation is based on out-of-bag (OOB) error estimate of random forest.

<https://stat.ethz.ch/education/semesters/ss2013/ams/paper/missForest_1.2.pdf>

1. **KNN Based Imputation**

This is the example of a model based approaches. K nearest neighbor method is based on the given training observations, an aggregation of the k values of the nearest neighbors is used as the imputes values. (In this method, the aggregation type depends on the type of variable).

For our implementation, we used a built-in package of R (VIM) for KNN based imputation. In VIM package, the distance computation for defining the nearest neighbor is based on an extension of the grower’s distance algorithm, which can handle distance variable of type binary, categorical, ordered, continuous and semi-continuous. In our project, all the missing value features were categorical type, therefore we used grower’s distance function for to compute the distance between the observations.

Under this approach, there are two methods to impute the missing data using k nearest neighbor observations. First method of aggregation is to use the category with the most occurrences in the k nearest neighbors, if this results in a tie, a category from the tied categories will be randomly chosen. Second method of aggregation is to sample the category from the categories in the k nearest neighbors with probabilities equal to the occurrences in the k values.

For our implementation, we used the first method is aggregation to impute the missing values with the K value as "5". We also tried the imputation with different values of K as 3,5,7,10 and 15 while measuring the model performance using logistic regressions classifier with cross validation. However, there were no significant gain by increasing the K value. Therefore, we fix our KNN imputation model with K value as "5".

<https://cran.r-project.org/web/packages/VIM/VIM.pdf>

*C: Select the imputation approach*

The most successful imputation approach is selected and data is balanced with the Bagging Classifier Method.

* 1. Data Encoding:

As we found during our initial data analysis, there were few categorical features in the given dataset. There are many machine learning algorithms which can support categorical feature in computation without any manipulations but there are many more which do not support. Machine learning algorithm use for this project does not support the categorical feature directly and requires further manipulation in the data. Therefore, we had to figure out how to turn these categorical features into numerical features for algorithm processing.

There are many ways to encode the categorical features into numerical. As with many other aspects of data science, there is no best approach for categorical data encoding. Every approach has its trade-off and potential impact on analysis outcome. Therefore, we tried multiple ways to encode our data and measured its outcome by running logistic regression classifier with cross validation technique.

For our implementation, we used a built-in python package "**CategoryEncoders**", which provides different techniques to encode the categorical data - One-Hot, Ordinal, Binary, Backward Difference, BaseN, Hashing, Helmert, Leave-One-Out, Polynomial, Sum encoding and default dummy encoding. We then used the encoded data to run logistic classifier with cross validation to measure the encoding technique impact. Under mentioned is the output of our observation with random forest imputed dataset with z-score normalization.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Encoding Type** | **Dimensionality** | **accuracy** | **precision** | **recall** | **f1-score** |
| BackwardDifferenceEncoder | 89 | 81.70% | 28.13% | 83.42% | 42.08% |
| BaseNEncoder | 29 | 85.53% | 60.76% | 73.40% | 66.49% |
| BinaryEncoder | 29 | 85.41% | 60.66% | 73.00% | 66.26% |
| HashingEncoder | 14 | 84.31% | 48.62% | 76.36% | 59.41% |
| HelmertEncoder | 89 | 85.34% | 64.53% | 70.81% | 67.53% |
| LeaveOneOutEncoder | 13 | 85.36% | 56.21% | 75.57% | 64.47% |
| OneHotEncoder | 96 | 85.28% | 64.17% | 70.80% | 67.32% |
| OrdinalEncoder | 13 | 84.51% | 48.31% | 77.68% | 59.57% |
| PolynomialEncoder | 89 | 85.28% | 63.10% | 71.28% | 66.94% |
| SumEncoder | 89 | 85.42% | 64.30% | 71.19% | 67.57% |
| DummyEncoder | 89 | 85.51% | 64.12% | 71.58% | 67.65% |

According to the above observation, BaseNEncoder type encoding is giving us the highest performance. However, when running the same model on different versions of dataset, DummyEncoder results in consistent performer over other encoder types. Therefore, we choose Dummy Encoder to encode our categorical features.

1. ***Data Mining Algorithms***

**Random Forest**

It is a type of ensemble learning method, where a group of weak Decision Tree models combine to form a powerful model. It is a versatile machine learning method capable of performing both regression and classification tasks. The basic algorithm, a Decision Tree is run multiple times to classify the label, or each tree ‘votes’ for the classification. The forest chooses the classification having the most votes (over all the trees in the ‘forest’) and in case of regression, it averages the output produced by different trees.

**Naive Bayes**

Understandably, naive Bayes performed worse than the other models because of the conditional independence assumption.

**Logistic Regression**

Logistic regression does not consider the interactions between various features and hence its performance is less than the other models. The implementation of logistic regression without stepwise functionality did not provide with results with higher accuracy.

**Support Vector Machines**

SVMs, in general, work well with nonlinear data. We noticed nonlinearity in some of our variables like capital gain/loss etc. and hence we also used SVMs for our classification task.

**K Nearest Neighbor Classifier**

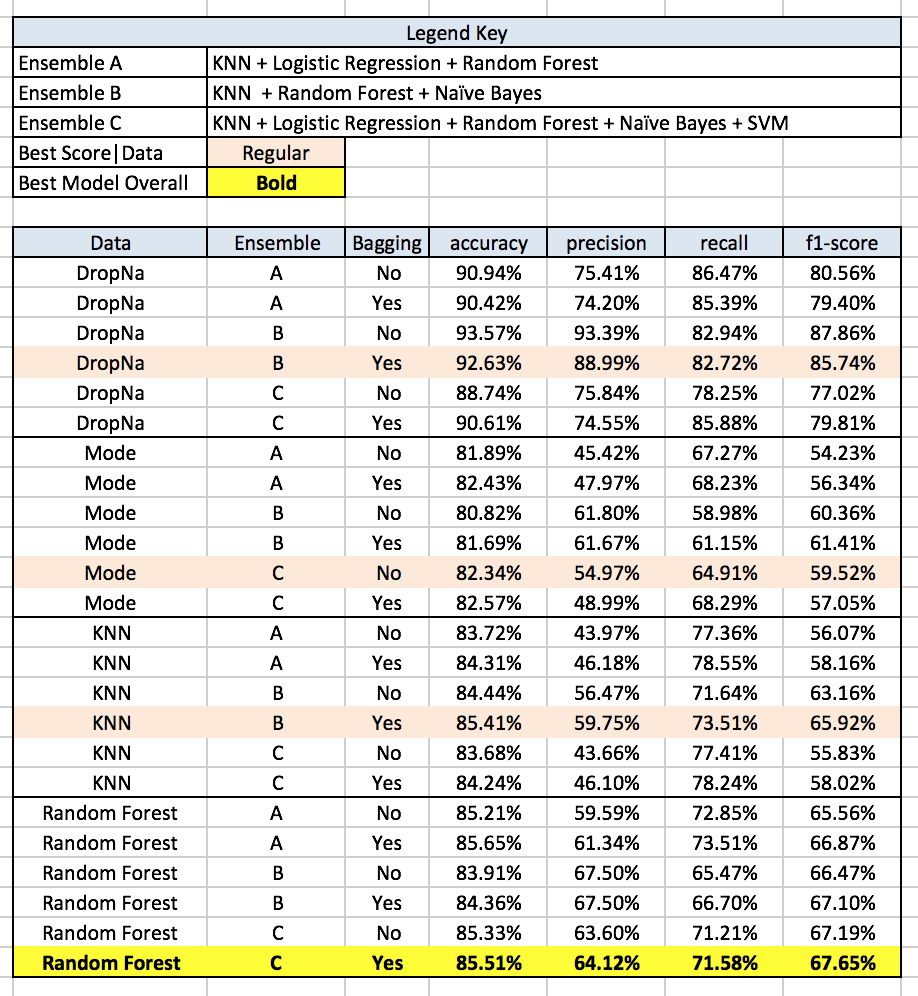
Using a KNN classifier for this task improved the performance significantly.

***Model selection (Ensemble Method)***

The Ensemble method is used to combine the multiple different algorithms. Due to the diversification and combined nature as compared to each other, the model gives a better prediction.

The key to creating a powerful ensemble model is diversity. An ensemble with two techniques that are very similar in nature performs poorly than a more diverse model.

So, to get the better accuracy, three ensemble models are created and the accuracies are compared to find the best model for the given data set.



*Bagging*