U.S. Census Income Classification Report

CISC 4631 - L01
Dr. Zhao
Fall 2019

By:

Shormie Faruque
Priska Mohunsingh
Zahin Roja
Kevin Wong

Link to Github Repository

Table of Contents

Link to Github Repository

Abstract

Overview and Project Description

A real world dataset from 1994 from the Census Bureau was used to predict the annual income of adults in the United States. The mission is to determine whether the income is greater than $50,000 or less than $50,000.

Problem Statement

Create a model to determine whether a person makes at least $50,000 a year. Use census-income.data.csv to train the classifiers and census-income.test.csv to test the classifiers. The algorithm should be based on the classification algorithms learned during the course. The algorithm can be a combination of methods and should incorporate one or more data mining techniques when the situation arises.

These techniques include (and are certainly not limited to):

- Handling imbalanced dataset
- Proper imputation methods for missing values
- Different treatment of various types of features: continuous, discrete, categorical, etc.

What Are We Obtaining With This Analysis?

The goal of this analysis is to examine a highly possible situation in an environment where organizations require donations. The purpose of an analysis as such will allow an organization or employee to better understand an individual's income, which will further allow them to make a

reasonable decision when pondering if they should pursue requesting an individual or how much should be requested (if they follow through with communicating a request).

End Goal

Our end goal for this task was to use a smart, accurate approach that will lead us to the highest possible accuracy. At the least, our expectation was to receive an 80% accuracy.

Overview of the Approaches Used

Before training our model, we had to preprocess both the training and test datasets. We used mode imputation to handle the missing values, converted the categorical variables into dummy variables to maintain numerical values throughout the dataset, used a bagging method to handle the imbalanced data, and used the Z-score method to normalize the dataset.

We built our model using the following classifiers: **Random Forest**, **Support Vector Machine (SVM)**, **Linear Regression**, **Naive Bayes**, and **KNN**. We used Pearson correlation ensemble to combine the classifiers and test the accuracy. We also computed Pearson correlation coefficients.

Brief Summary of the Results

The results show that when the test and train files are read by the train files, we have the highest accuracy of 88%. The equivalency of the test and train data to the train files was our control method, which was used to compare the different classification methods that we were using. With the different ensembles 1, 2 and 3, where the Pearson coefficient equation was used, SVM and Linear Regression had accuracy of 83.34%. Then, we have the Bagged Ensemble with mode imputed data where Linear Regression, KNN, and Random Forest had an accuracy of 82.71% which is lower than when we used SVM and Linear Regression together. By using the pearson

correlation coefficient, our goal was to achieve a higher accuracy because the imbalanced data is handled as specific areas are targeted through the Bagged Method and helps to reduce overfitting.

Our Hypothesis

We believed that education would have the greatest impact on a person's income. Generally in the U.S., the more education one has, the more money one makes, depending on his/her occupation. We also believed that the combination of Random Forest and SVM would result in the highest accuracy rates because both algorithms are known to perform well on their own.

Our Data Analysis

Variables

In order to conduct our analysis, our dataset consisted of fourteen different variables. These variables were predominantly continuous and categorical except for one, education_num, which is an ordinal variable.

Continuous Variables: These are numerical and have an infinite number of values between any two values.

- **age**: the age of the person (any integer greater than 0)
- **fnlwgt**: the final weight (also any integer greater than 0)
- **capital_gain**: capital gains for a person (any integer ≥ 0)
- **capital_loss:** capital loss for a person (any integer ≥ 0)
- **hours_per_week**: the number of hours a person has reported to have worked weekly

Categorical Variables: Contain a finite number of categories or distinct groups; might not have a logical order (i.e. gender, material type, payment method).

- **workclass:** employment status of person
- **education:** the highest level of education achieved by a person
- **marital_status:** marital status of a person
- **occupation:** occupation of a person
- **relationship:** one relationship attribute per person (i.e. wife)
- **race:** a person's race
- **sex:** the biological sex of a person (male, female)
- **native_country:** a person's country of origin

Ensemble Models

We built and tested ensemble classifier models comprised of multiple supervised learning models, namely **Random Forest**, **Support Vector Machine (SVM)**, **Linear Regression**, **Naive Bayes**, and **KNN**.

We determined the classification into those two > 50K and ≤ 50K by using categorical and continuous variables and classification methods (i.e. Random Forest and SVM.) We used Python for our data analysis.

We then passed Random Forest, Support Vector Machine (SVM), Linear Regression, Naive Bayes, and KNN through scikit-learn, as it appropriately and conveniently features various classification, clustering, and regression algorithms.

Data Pre-Processing

The preprocessing procedure included cleaning, concise formatting and restructuring of data. For this dataset in particular, there are some features that must be adjusted. The pre-processing

procedure is a vital task, as it will holistically improve the results and predictive power of our model.

Preprocessing Categorical Variables

Some features, such as ‘occupation’ or ‘race’ are categorical variables rather than continuous. However, in Machine learning, algorithms function solely with numerical values. Thus, it is a necessary step to translate categorical variables into continuous, numerical variables.

To complete this, we used one of the most common categorical transformation procedures, namely the ‘one-hot encoding’ procedure. We performed one-hot encoding, where a ‘dummy’ variable was created for each possible category of the categorical feature:

In order to “one-hot-encode” the dataset, we used the concatenation function upon the dummy variables. This function combines the dummy variables into one dataframe. We then dropped the categorical variables.

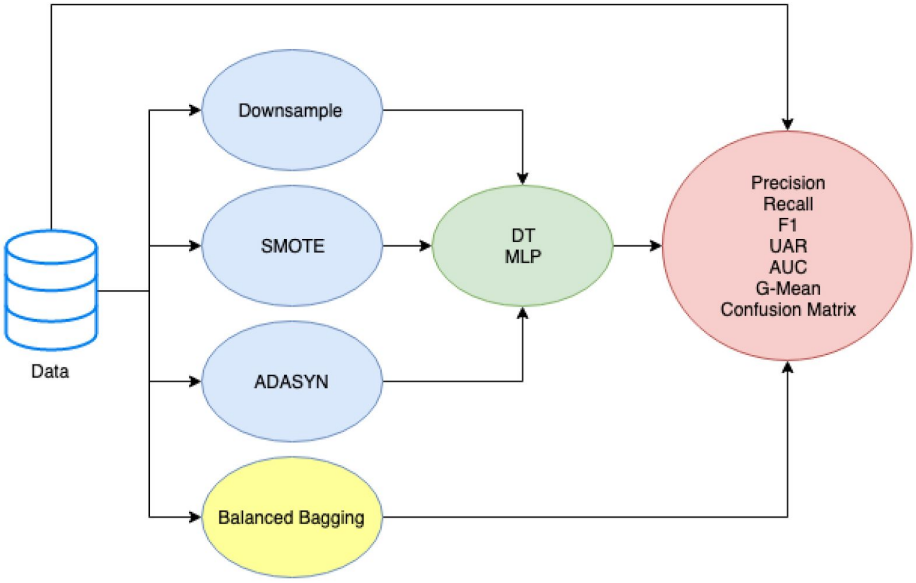
Preprocessing Missing Values

To deal with missing values, we replaced all question marks in the dataset with NaN values. The NaN values were removed once we converted the categorical variables into dummy variables. We were able to predict what the missing values were using mode imputation.

Dealing with Imbalanced Data

To balance our dataset, we used the bagging method. Bagging helps reduce overfitting and results with averaged data. To do the bagging, we called the Bagging classifier from the sklearn library in Python. We assigned n_estimators = 90 for Random Forest Classifier, n_neighbors =

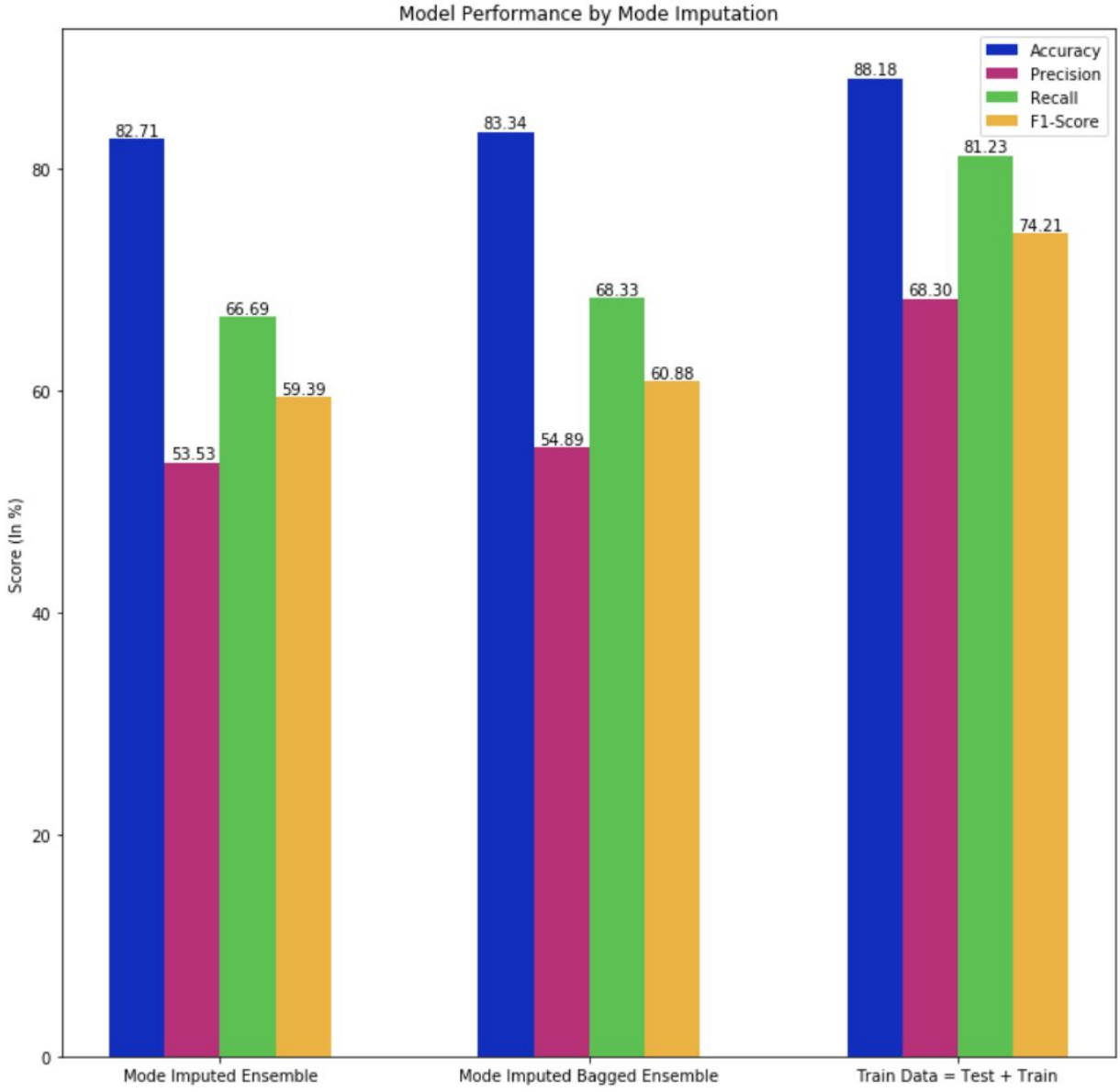
15 for KNN Neighbors, C = 10 for Linear Regression, and C = 10 and gamma = 0.1 for SVM. The classifiers were able to handle the imbalanced data without having to oversample or undersample manually without training. After implementing bagging and normalization, we iterated through neighbors for Random Forest and KNN to figure out the optimal number of neighbors for KNN and estimators for Random Forest. Handling the data without oversampling or undersampling enables us to save time and be efficient.



The image above shows a pipeline of data using various techniques for predictive modeling. Bagging is more efficient than the other techniques because it's not required to go through the DT MLP step to get to Precision, Recall, F1, Confusion Matrix, etc. If we used Downsample, SMOTE, ADASYN, it would take a longer runtime and could lead to a lower accuracy since there are more steps involved to get to the end goal in the pipeline. The pipeline ultimately shows why we decided to use Balanced Bagging over other methods to balance our imbalanced data.

Link to Github Repository

Unbalanced Data Bar Graph



The bar graphs visualize the performance, specifically the accuracy, precision, recall, and F1 score of each of the 3 models we analyzed based on mode imputation. The last bar graph, which represents the Test + Train displays the highest accuracy (~88%). We obtained a higher accuracy for this model mostly because the algorithms were running on the mode-imputed training data. In

comparison to the slight accuracy increase for the Bagged Ensemble, the precision, recall, and F1 score were relatively higher for the last model as well.

Normalizing the Dataset

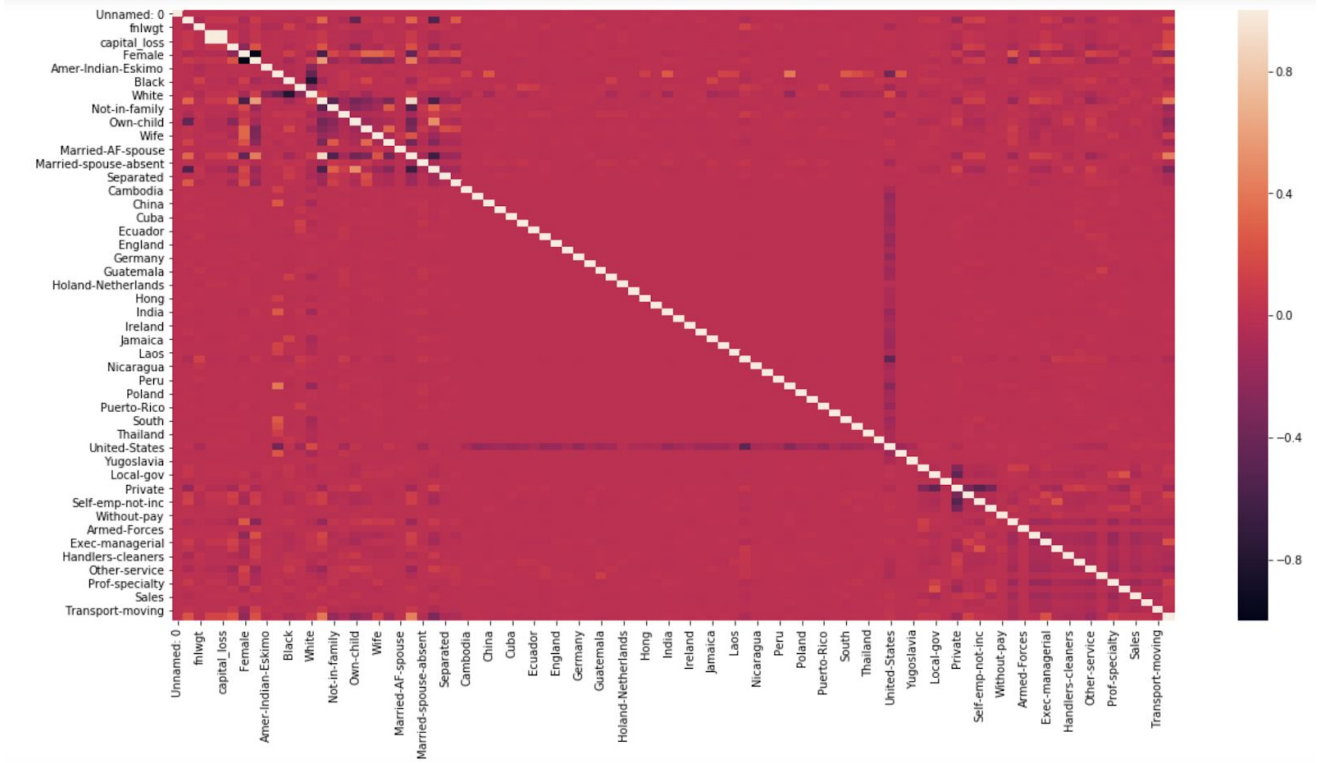
We must normalize the dataset to prevent some features with large values from skewing the classification results. We applied the Z-score method across all instances to make the data values more comparable to each other. To calculate Z-score, we used the mean and standard deviation functions that are part of Python's statistics library.

Results

Correlation Between Features and Variables

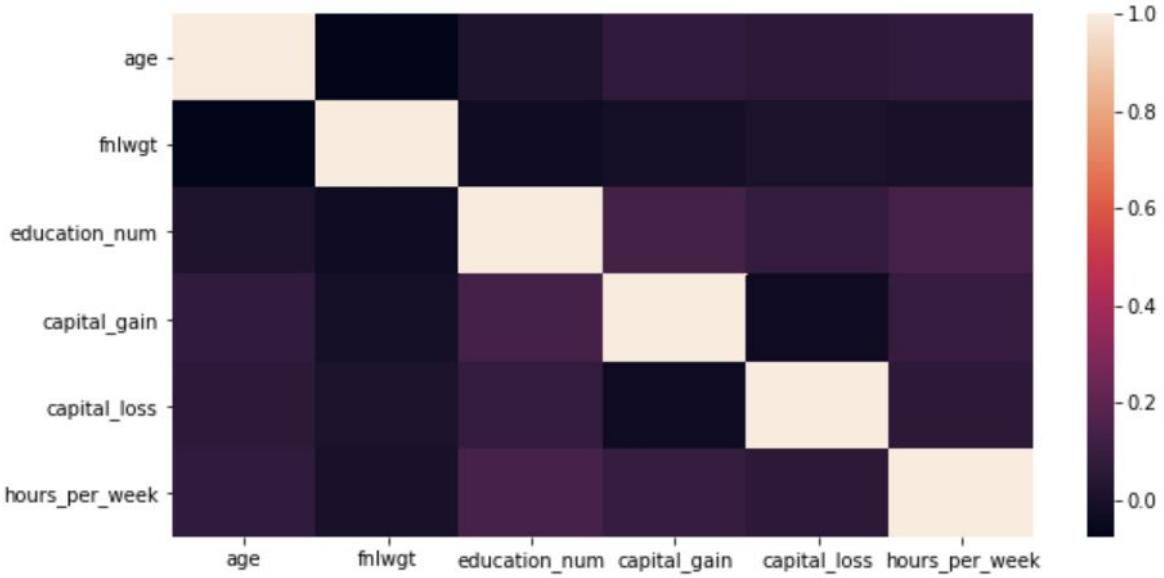
To visualize the correlation features and variables, we used Seaborn, which is a data visualization library grounded on matplotLib. To help us understand the data, we used a heat map, which uses color to display and communicate data behavior.

Link to Github Repository



The heat map above displays the combination of attributes ‘native_country’ and ‘occupation’ on the y-axis. On the x-axis, the categorical attributes ‘native_country’ and ‘occupation’ are also amalgamated to further convey the behavior between the chosen features and variables. This heat map shows pearson correlation coefficients, where there is a lower correlation as values move closer to -0.8. As values move closer towards +0.8, they have a more positive and stronger correlation between the given features. From the training balanced file, those who were white and female had a strong correlation, those who are white also had a missing/divorced spouse strong correlation. Private and self-employed also had a strong correlation. The native countries shown in the heat map also had strong correlation. Ultimately, the strong correlation with sex and native country implies that these two attributes may have also had an effect in determining whether an individual makes greater than or less than 50K leading to higher accuracy with the different algorithms.

Link to Github Repository



The heat map above depicts the pearson correlation coefficient and it measures the statistical correlation between two continuous variables, which is therefore a number between 0 and 1. Values that are closer to 0 signify that those variables have a lower correlation and values closer to +1 imply that those values have a stronger and more significant correlation on the goal.

Our end goal is determining if income > 50K or ≤ 50K. We can see that capital_gain and education_num have a lighter purple and so those variables have a stronger correlation than education_num and fnl_wgt which are a dark purple closer to 0. This means that an individual's education level is essential in determining whether or not they make over or under 50K. This makes sense because careers that require undergraduate degrees usually pay higher although there might be some outliers (i.e. Computer Science). Similarly, education_num is strongly correlated with capital_gain and hours_per_week.

This observation indicates that a greater number of hours are worked increases the capital in the company, and is attributed to individuals who have completed their education.

Accuracy Results

Based on our findings, we can conclude that when we combined Random Forest, Linear Regression, and KNN together, we achieved the highest accuracy of approximately 82% for the mode imputed data.

We also combined Random Forest, Linear Regression, KNN, SVM, and Naive Bayes however, we received a lower accuracy of approximately 81%.

When using Bagged Ensemble to combine Random Forest, Linear Regression, and KNN, we received an increase accuracy to approximately 83% for the mode-imputed data. By making the test and train data the same mode-imputed train data file, we obtained a higher accuracy of 88%, but this was because the algorithms were only running on the mode-imputed train file.

Pearson Coefficient Mode Ensemble

Ensemble 1

Random Forest and Logistic Regression were the least correlated algorithms at 59%.

a. RF and LR = 0.5930685983723457
b. LR and KNN = 0.6206311863239945
c. RF and KNN = 0.6704896273448653

Ensemble 2

Naive Bayes doesn't have a strong correlation with Random Forest and KNN although we expected Ensemble 2 to have a higher correlation because two different decision tree classification methods are being used.

a. RF and NB = 0.41173259165191306
b. NB and KNN = 0.4324767649063429
c. RF and KNN = 0.6704896273448653

Ensemble 3

There are four algorithms that we used in this ensemble and so we expected the highest accuracy to appear here.

a. SVM and NB = 0.46402116800760773
b. SVM + LR (Highest Accuracy) = 0.8438885603188111
c. SVM and KNN = 0.6653449646056273
d. SVM and RF = 0.6023234590583493

Conclusion

Results vs. Hypothesis

Based on our hypothesis, we predicted correctly that education had an effect on the income level that an individual was making. However, we were incorrect that Random Forest will produce the highest accuracy because SVM and LR resulted in the highest accuracy.

Next Steps

One of our next steps would be to use Random Forest to impute the missing values into a new Random Forest-imputed file because Random Forest by definition has multiple trees that can aggregate the data. Such steps would lead to an accuracy greater than 83%. Other steps we would take when imputing the data with Random Forest could include pre-imputing the data and growing the forest, updating the missing values using the proximity of the data and repeating this process to attain a higher accuracy. We could also decide to simultaneously impute the data as we grow the forest and repeat this process as well for the sake of getting a higher accuracy.

Likewise, when it comes to imputing data with SVM, we could follow a similar methodology and implement. Additionally, we also normalized our features to to test and train with the different algorithms and we could decide to test the files with the different algorithms without normalization which could possibly give us higher than 83% accuracy.

References

1. Lemon, C., Zelazo, C., & Mulakaluri, K. (n.d.). *Predicting if income exceeds $50,000 per year based on 1994 Us Census Data with Simple Classification Techniques*. Retrieved from http://cseweb.ucsd.edu/classes/sp15/cse190-c/reports/sp15/048.pdf.
2. MDPI. (24 July 2019). *Predictive Models for Imbalanced Data: A School Dropout Perspective.* Retrieved from https://www.mdpi.com/2227-7102/9/4/275.
3. Scikit-Learn. (n.d.). *Imputation of missing values.* Retrieved from https://scikit-learn.org/stable/modules/impute.html.

4. Seaborn. (n.d.). *seaborn.heatmap.* Retrieved from https://seaborn.pydata.org/generated/seaborn.heatmap.html.
5. Silicon Valley Data Science. (n.d.). *Learning from Imbalanced Classes.* https://www.svds.com/learning-imbalanced-classes/.
6. Statistics Solutions. (2019). *Pearson's Correlation Coefficient*. Retrieved from
7. https://www.statisticssolutions.com/pearsons-correlation-coefficient/
8. Towards Data Science. (n.d.). *Having an Imbalanced Dataset? Here Is How You Can Fix It.* Retrived from https://towardsdatascience.com/having-an-imbalanced-dataset-here-is-how-you-can-solve-it-1640568947eb.