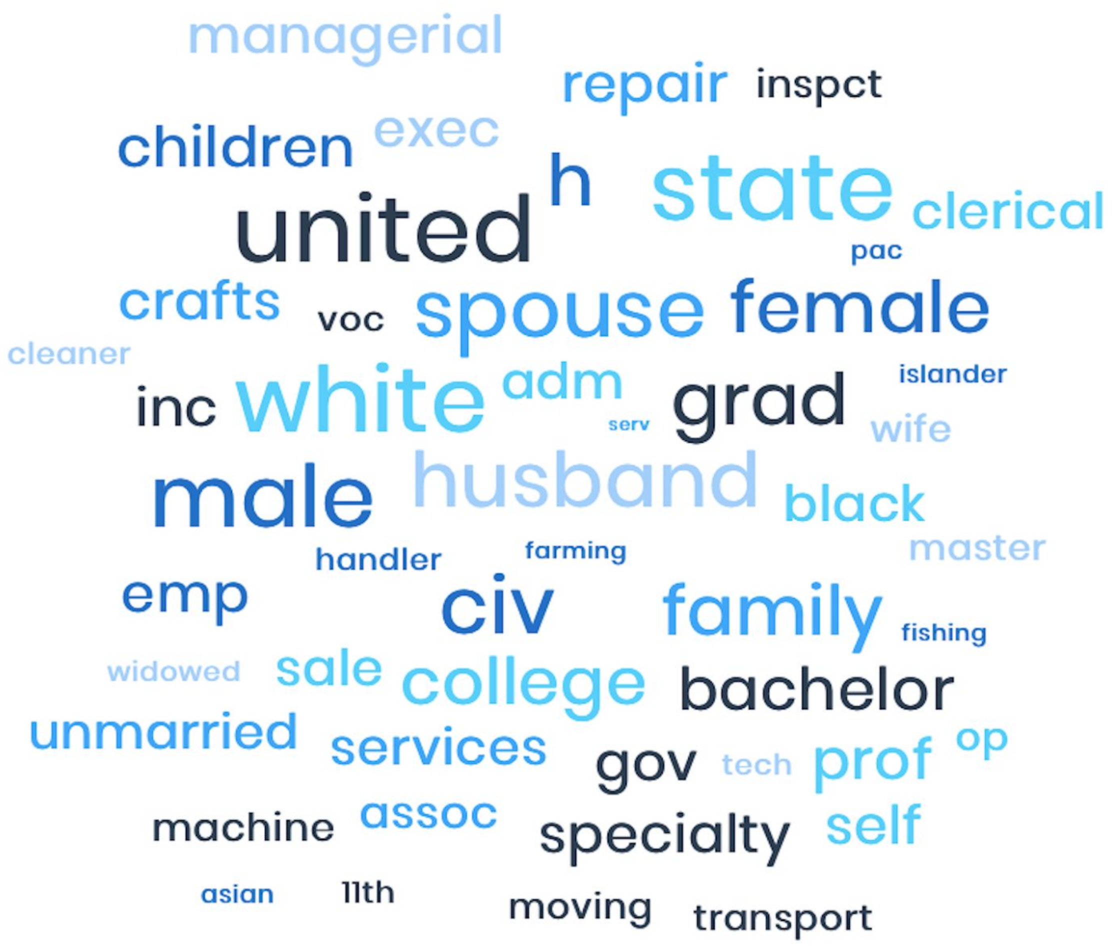
Data Mining Census Income Project: Predicting Greater or Less than $50,000 Annual Income Based on Census Data



Project Participants: Suzanne Zhen, John Caruso, Xiaozhe (Sam) Zhou, and Renjie Chen

Abstract

Use of Machine Learning algorithms is a reliable and accurate means of predicting income levels for people whose income classification is unknown. While one method may not be sufficient to achieve satisfactory accuracy, a collection of methods or ensemble of algorithms can provide decent accuracy in classifying income levels for people. Such classifications with sufficient accuracy can be made, provided that aggregate training data such as country of origin, sex, and age in addition to known income classification is available for analyzation from which models can be created to predict income classification. Various standard machine learning algorithms were evaluated for the efficacy in classifying test data. Of the algorithms tested, SVM (Support Vector Machine), KNN (K-Nearest Neighbors), and Random Forest were shown to be the most effective at classification while Naïve Bayes was shown to be the lowest performing method. In creating an ensemble classifier that can predict income level given the type of aforementioned training data, accuracy in excess of 80% was achieved for SVM, KNN, and Random Forest while Naïve Bayes resulted in accuracy below 60%

Introduction

Obtaining the final classification predictions for income levels first involved preprocessing to prepare the data followed by prediction of income levels via different types of machine learning algorithms that we learned about during the semester: SVM, Random Forests, KNN, and Naive Bayes.

KNN (K-Nearest Neighbors) is a supervised learning technique that attempts to make a classification prediction by measuring distance between the unknown or unclassified point and its already classified surrounding neighbors. Depending on the "K" used in the algorithm, the k nearest neighbors or those having the minimum distance, measured by Euclidean distance in this project, are used to classify the unknown point. Specifically, the mode is taken of the k nearest neighbors and that mode or most occurring classification of the neighbors becomes the predicted classification for the new point. This algorithm is based on the assumption that points of similar classification are close to one another and reside in clusters of similar data. Consequently, determining the nearest neighbors and predicting an unknown's classification based on that forms the basis for KNN.

SVM (Support Vector Machine) attempts to find a function to fit the hyperplane of separation between clusters of data. This makes the assumption, similar to KNN, that clusters of similar data reside in close proximity to each other such that determining an approximate function for the separation between clusters will allow for classification of an unknown point given input of that point's parameters. Particularly, linear SVM and polynomial provide a linear and polynomial fit respectively of that plane of separation while SVM using an RBF (Radial

Basis Function) kernel provides a higher order non-linear transformation. All three types of SVM were tested for their efficacy in this project as well.

Random Forests create an ensemble of decision trees that work together as majority vote classifiers. The number of decision trees to be created is decided upon at first then the testing dataset is mined for possible permutations of decision trees based on the attributes given. The final ensemble of trees decides the predicted classification of the unknown point.

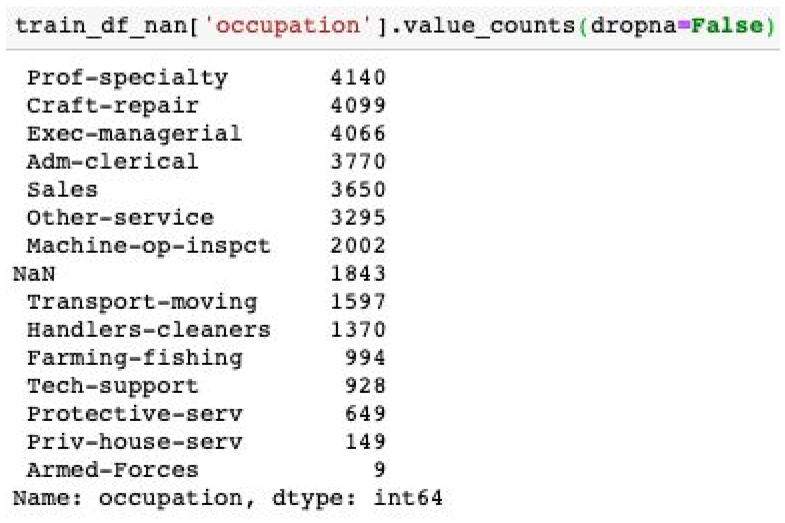
Naïve Bayes makes use of Bayes Rule which links and measures the probability of one event occurring based on its connection to another independent event. It entails the Naive Bayes assumption which measures probability of one classification or another in terms of independent features. Consequently, classification is predicted using Naïve Bayes by choosing the outcome that is most likely or probable.

Methods

The end-to-end process of actually classifying the test data took place over several steps. Overall, this process includes first preprocessing the training and test data, followed by construction of the actual classifiers, and then finally evaluation of the classification using standard metrics for measuring accuracy of ensemble classifiers such as precision, recall, f1-score, support, and accuracy for all the above. For the implementation of all machine learning algorithms, scikit-learn, a popular python library of packages was used to generate results.

Our training dataset contained 32,561 instances with a total number of 14 features while our testing dataset contained 16,281 instances with the same amount of features. The datasets consisted of both continuous and discrete features. As an initial preprocessing step, feature names were added to both datasets based on the descriptions in the “Census Income Names” file. The attributes within the target column “50K” were encoded as either “1” or “0” since the experiment follows a binary distribution, with value “1” being assigned to income greater than 50K and value “0” being assigned to income less than 50K.

Next, imputation was carried out for both the training and test data. The features with missing values in both datasets were: “Workclass”, “Occupation”, and “Native_Country”. The missing values initially represented by “?” were later transformed to “Nan” values instead. After this step, the missing values for “Workclass” and “Native_Country” were imputed with the mode of their respective columns. For “Occupation”, however, the missing values were imputed as “Other-service” since the mode “Prof-specialty did not have a distinctively high count compared to other occupations (see Figure 1).



```
train_df_nan['occupation'].value_counts(dropna=False)
```

```
 Prof-specialty       4140
 Craft-repair         4099
 Exec-managerial      4066
 Adm-clerical         3770
 Sales                3650
 Other-service        3295
 Machine-op-inspct    2002
NaN                   1843
 Transport-moving     1597
 Handlers-cleaners    1370
 Farming-fishing       994
 Tech-support          928
 Protective-serv       649
 Priv-house-serv       149
 Armed-Forces            9
Name: occupation, dtype: int64
```

Figure 1

After that, we observed an imbalance of classifications in our training data. Our training data has three times as many instances with income less than 50K as instances with income greater than 50K (see Figure 2). In order to avoid bias toward the majority class when implementing our algorithms, the minority fraction of data was oversampled by the appropriate multiple to create equal instances of high and low income within the training data. Consequently, we oversampled our training data to account for this bias.

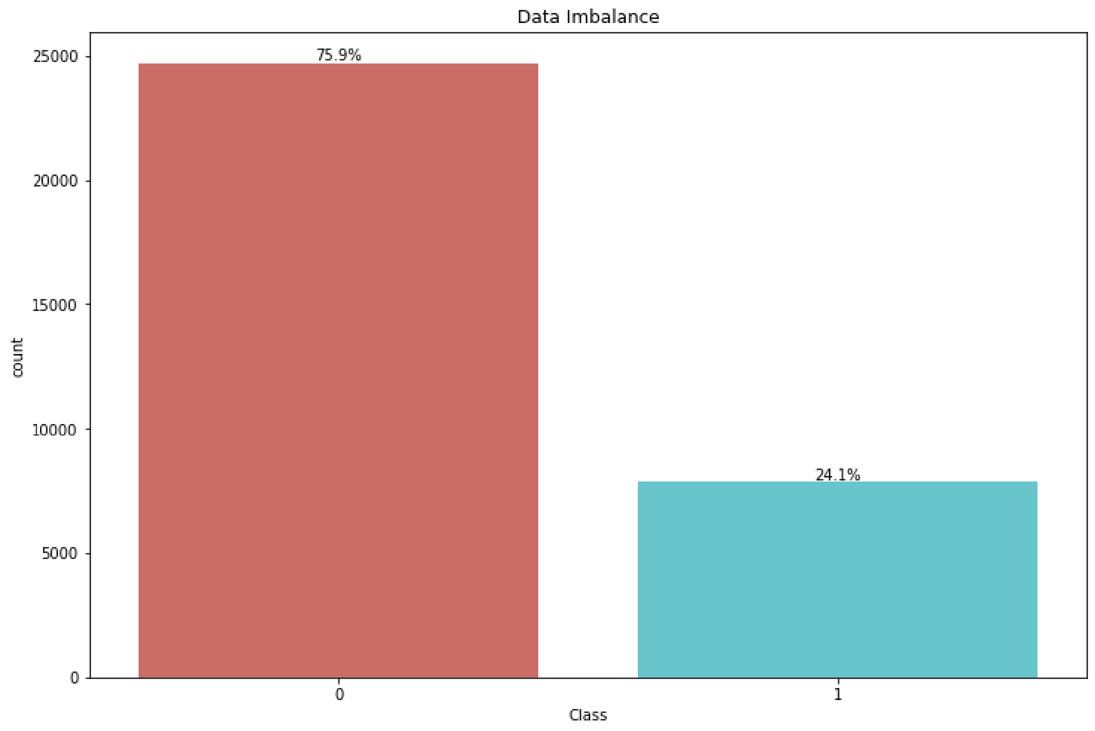


Figure2: Training Data Imbalance

Since linear algorithms will be implemented, features with discrete variables such as "Workclass", "Occupation", and "Sex" were separated from continuous variables for further processing. In order to transform discrete attributes into numeric variables, the method of "OneHotEncoding" was implemented. In this step, features with discrete attributes were split into multiple columns with respect to the number of attributes within the feature. Each attribute was transformed into a column with value of "1" if the attribute was present or value of "0" otherwise. The feature "Education" was dropped from analysis since a comparable continuous feature "education_num" was present in the data. The same encoding was applied to the testing data with one additional feature " Holand-Netherlands" (all encoded "0"'s) since this country was absent in the testing data.

Following discrete features encoding, continuous features including "Age", "Fnlwgt", "Education_num", "Capital_gain", "Capital_loss", and "Hours_per_week" were normalized. Specifically, these were normalized using the L2 norm so that all values were between the scale of "0" and "1". The encoded discrete features were then combined with the normalized continuous features to form our data frame for analysis.

Once pre-processing was complete, four different types of common machine learning algorithms, as discussed in the introduction, were used to classify the census data: SVM, Random Forest, Naïve Bayes, and KNN. For SVM, three different types of SVM were used: Linear SVM, RBF Kernel SVM, and Poly-Kernel SVM. Our results and a pertinent analysis of their meaningful conclusions are discussed below in the results section.

Results/Analysis

The goal of this project was to evaluate which machine learning algorithms, out of the ones chosen for testing, would produce the best results for predicting classification of income levels when combined into an ensemble classifier. In addition, we also make note of those machine learning algorithms that did not result in sufficient accuracy. Overall, Random Forest produced the highest accuracy while Naïve Bayes produced the lowest accuracy.

Before evaluating the test data with our models, we performed 5-fold cross-validation on our training data. Performing cross-validation gave us an initial evaluation of each model's accuracy and what to expect on the testing data (see figure 3).

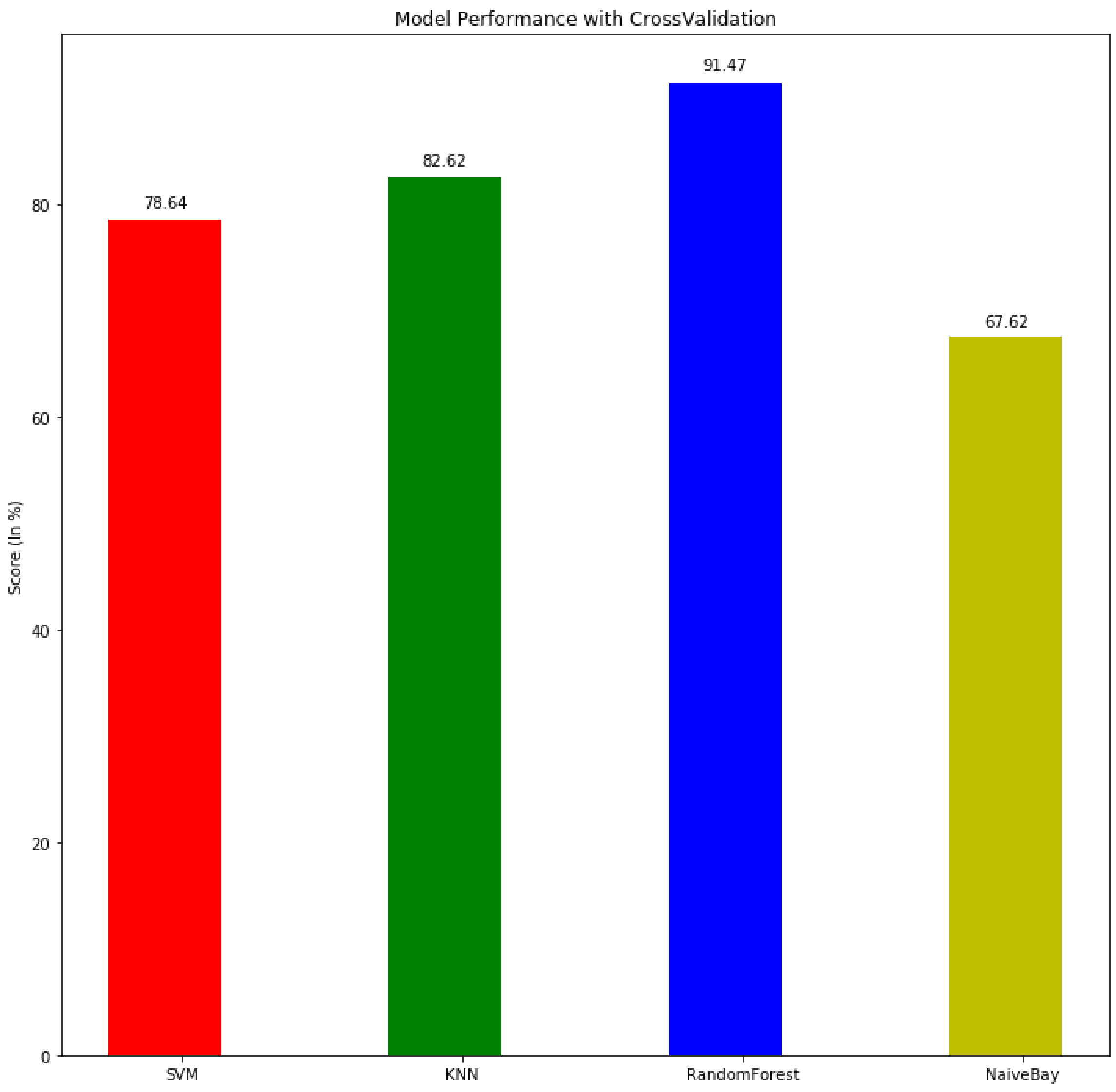


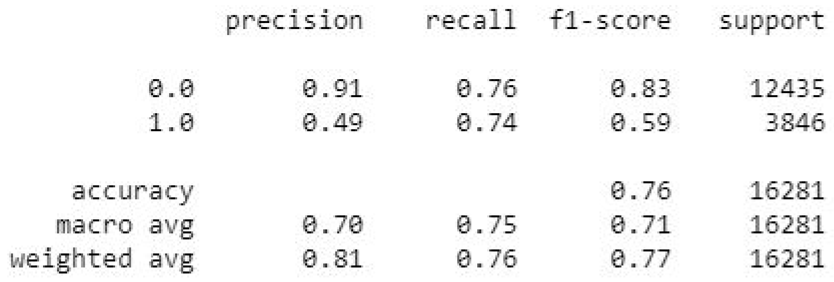
Figure 3: 5-fold CV Average Accuracies

To evaluate the different methods tested against the other, each was compared against the other within the ensemble in terms of accuracy, precision, recall, and f1-score for both a weighted average and a macro average. Accuracy is a standard metric that calculates the percentage of correctness of a given set of measured values to their true value. Precision is a measure of how well a positive outcome is predicted. Recall measures how many correctly classified values there are out of all possible points classified. This measure is also referred to as completeness of the data. F1 score is a measure of the test accuracy on a scale from 0 to 9. Specificity is a further measure of how well a negative outcome is predicted. For instance, if a classifier chooses 0 on the basis of the point not being 1 then specificity is a measure of how well that prediction comes about.

The comparisons were carried out in two different ways: Weighted average and macro average. The weighted average of a type of a classifier takes into account the metrics by class but not weighted using the data points while the macro average takes into account these same metrics but without the weighting

Another interesting thing that we noticed is the overall average accuracy for balanced data, using the method of oversampling, is lower than the accuracy we got from the raw data. However, some scores look better when using the balanced data. Here, we chose the result of the KNN classifier to further explain. Comparing these two confusion matrices, we can see for precision, a label 0 is high compared with the raw data. This is because after doing oversampling, the proportion of label 0 in the data is reduced from 75% to 50% therefore the classification of the FP for label 0 is low due to the lower proportion. Therefore, we think even though the overall performance for oversampling is worse, we can use this technique to accomplish some particular request. For example, here we can use our oversampling technique to predict the people with low income more accurately.

Figure 4: Data with oversampling

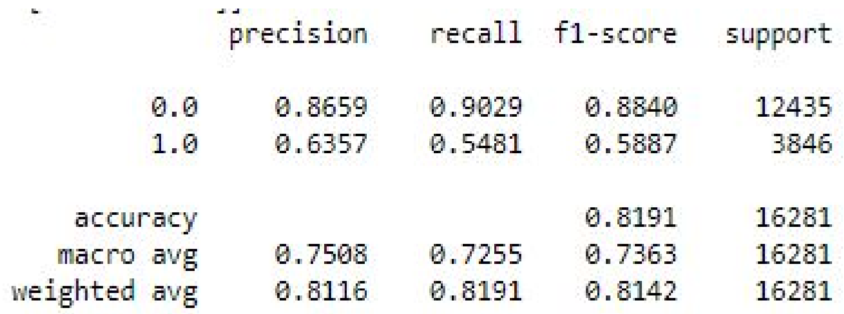


```
              precision    recall  f1-score   support

         0.0       0.91      0.76      0.83     12435
         1.0       0.49      0.74      0.59      3846

    accuracy                           0.76     16281
   macro avg       0.70      0.75      0.71     16281
weighted avg       0.81      0.76      0.77     16281
```

Figure 5: Raw data with Oversampling



```
              precision    recall  f1-score   support

         0.0     0.8659    0.9029    0.8840     12435
         1.0     0.6357    0.5481    0.5887      3846

    accuracy                         0.8191     16281
   macro avg     0.7508    0.7255    0.7363     16281
weighted avg     0.8116    0.8191    0.8142     16281
```

Figure 6: Model Performance with Weighted Average - Balanced Data

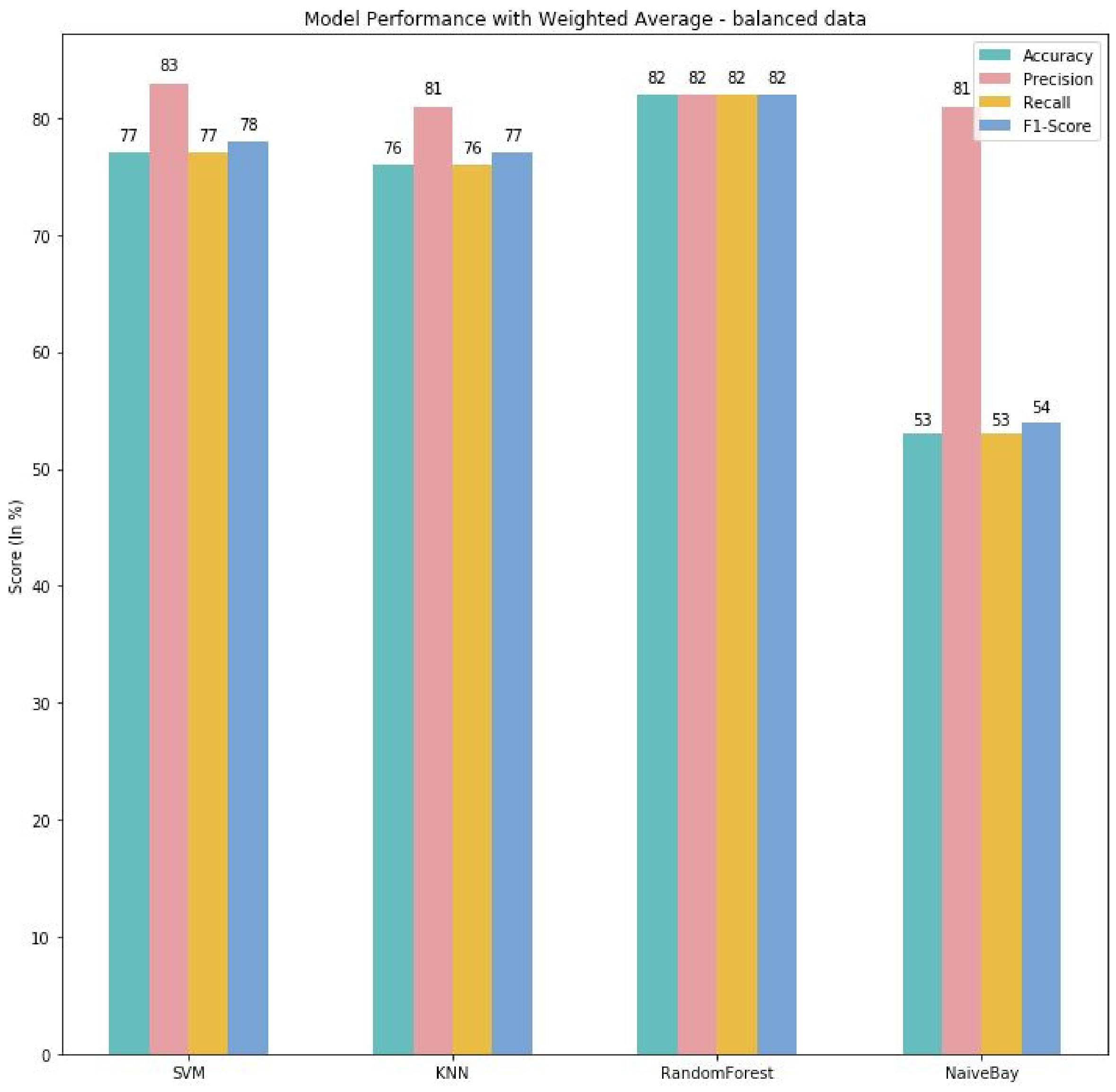
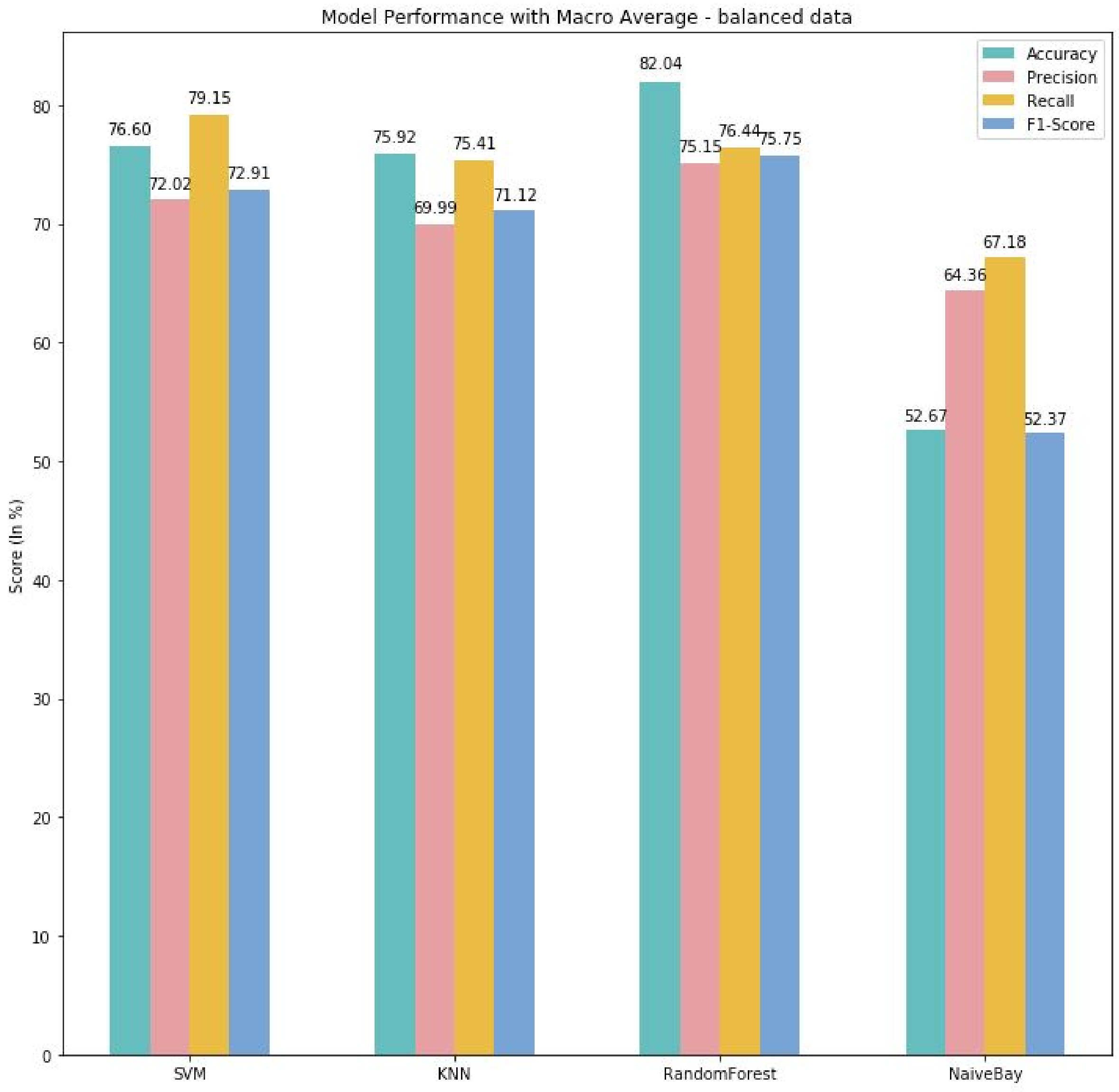


Figure 7: Model Performance with Macro Average - Balanced Data



Overall, both figure 6 and 7 show us that SVM, KNN, and Random Forest are much better methods, on the basis of accuracy, than Naive Bayes. Specifically, using macro average accuracy for each of these methods is 76%, 75.92%, 82.04%, and 52.67% respectively while for the weighted average it is 77%, 76%, 82%, and 53% respectively. With that said, we achieved much better classification with those first three methods.

In addition, our performance above in figure 6 and 7 shows that our data and preprocessing was sufficient to prove these findings. Particularly, precision metrics for each method in the weighted average used were in excess of 80% which tell us that the classifier classified positive outcomes of higher income well even though overall accuracy was substantially lower because lower accuracy was not measured well.

Of the methods that worked very well such as SVM, KNN, and Random Forest there are several reasons why they had high degrees of accuracy. First, in regard to SVM, this method works by finding the hyperplane around clusters of data. Because we were able to find accurate fits of approximations for these clusters, we can conclude that the data was dispersed in clusters such that a hyperplane could separate them. In other words, the data was not noisy.

KNN worked exceedingly well and we speculate that similar to the reasons that SVM worked well, the data was broken into similar clusters such that neighbors of differing classifications were farther apart and well dispersed such that euclidean distance was a good measure with which to define classification.

Random Forest also worked well because, again, the resulting ensemble classifier is based on the relationship or lowest entropy between attributes going from one level to another in its decision trees. Consequently, this also shows there was a clear relationship between attributes.

Finally, Naive Bayes did not work as well as the other classification algorithms because the Naive Bayes assumption here did not hold up for the data being used. Specifically, all the attributes were shown to be linked to one another through the analysis previously done talking about why the data was well classified by other algorithms. Consequently, making the assumption that all the attributes were separate and predicting classification based off that does not hold well, thus Naive Bayes does not accurately identify the income classifications.

Overall, we can see that our data was well separated from the start in terms of there being a clear pattern between attributes within respective features and income level classification. This was shown by the good accuracy of KNN, SVM, and Random Forest which are based off those linkages while Naive Bayes did not work well because it takes the probability of each feature into account separately.