CSIT 456 Project Report

Title: Income Identifier

Name: Tom Prokop

1. Description

1.1 Basic Information:

The purpose of this project is to use demographic data to see if it's indicative of income. Using data collected from the United States Census, we will be able to try to gain insights as to whether or not a machine can identify patterns correlating between demographic data and the income of that individual.

1.2 Project Objectives:

1. Build 4 binary classification models to try to discern the income of an individual based on demographic data.
2. Evaluate these four models to determine which model is the best.

1.3 Description of Data Set:

The dataset was taken online from the UCI repository. It was collected from the US census bureau in 1990 and contains 14 predictor variables along with one target variable. There are both categorical and numerical variables in the set of predictor variables, and the target variable is a binary variable. The target variable indicates whether the individual makes less than or equal to fifty thousand dollars per year or more than fifty thousand dollars per year.

2. Exploration of Data Analysis

2.1 Data Preparation

The data was already clean, but there was some preprocessing necessary before the implementation of the machine learning models. First, the features needed to be identified for any missing values. Instances in which there were NA values in some features were then removed from the dataset, which reduced the total instances from approximately 48,000 to 45,000. From there, the target variable is be converted into a Boolean integer of either "0" for "less than or equal to fifty thousand" or "1" for "greater than fifty thousand." In addition, the categorical variables needed to be encoded. Therefore, sklearn's OneHotEncoder was applied to all the categorical variables and joined to the dataset, and the original categorical variables were then dropped from the dataset. Finally, the target variable was moved to end of the dataset to make it easier to find through splicing, and feature variables and target variable were converted into the numpy arrays "X" and "y" respectively. These arrays were then split into the training and testing sets using sklearn's train_test_split.

2.2 Models Selection

This is a binary classification problem, and thus, four classification models have been used: KNN, Logistic Regression, Decision Tree, and Random Forest.

2.3 Models Implementation

Each model has been first initiated with no specific parameters and fitted to the training data. Following this, the tuning of hyperparameters began. Since it was so computationally intensive, Google Cloud AI was used. This allowed the models to be implemented on a cloud server with 128GB of RAM and 16 vCPUs and sped up the process some, but it still takes about 10 minutes to search for parameters for some models. Despite this fact, the process was relatively the same for all the models with the exception of the Random Forest model. For each model, a set of parameters was selected to iterate through using GridSearchCV. This was very computationally intensive, therefore the GridSearch implementation was broken up into two steps:

1. Apply to a broken range (i.e. range(10, 100, 10), or from 10 to 100 but only every 10).
2. Take the best parameters and narrow the GridSearch (i.e., the best parameter yielded is "30", so re-run the GridSearch with that parameter set to range(25,36).
3. Repeat Step 2 until the best parameter is found.

This method helped to break up the GridSearch into multiple processes to reduce the run-time while still providing a wide range of values for the parameters. For the Random Forest, however, a RandomizedSearch was applied prior to the GridSearch process. This is due to the fact that there were many parameters and a wide range that were intended to be tried, so the computational power was so intensive that it was decided to take a large sample of random parameters and pick the best one from this set to narrow the scope with GridSearch.

3. Evaluation

For the evaluation, there were three different metrics that were applied. The first metric applied was the ROC curve and its accompanied AUC score. The ROC curve for all the models is shown below in Figure 1, and the AUC scores are shown in Table 1.

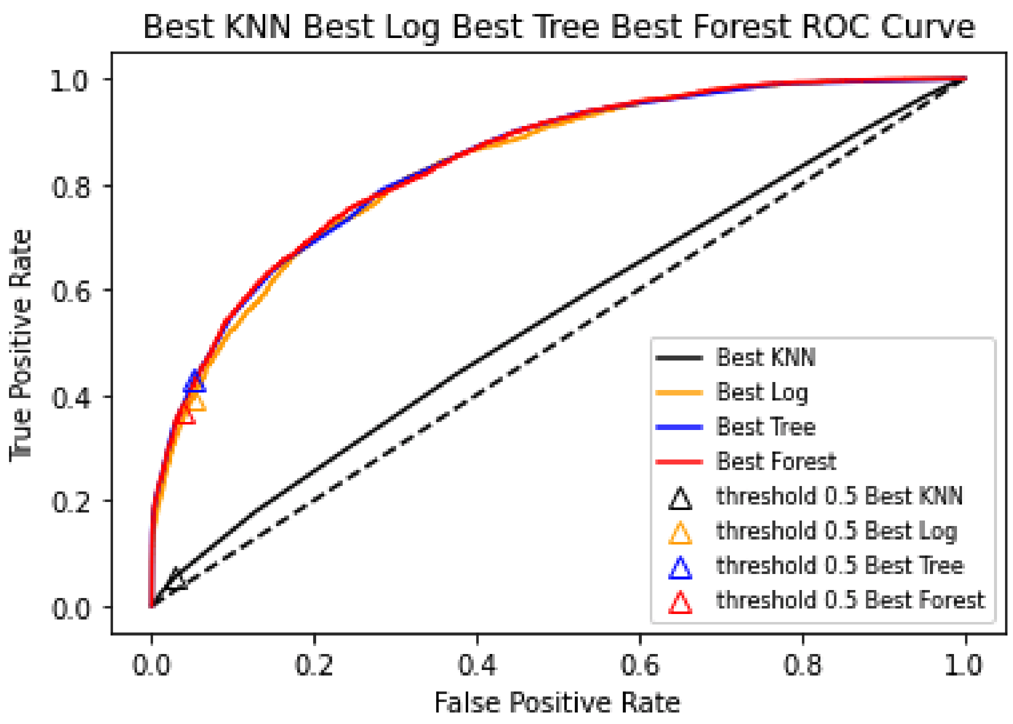
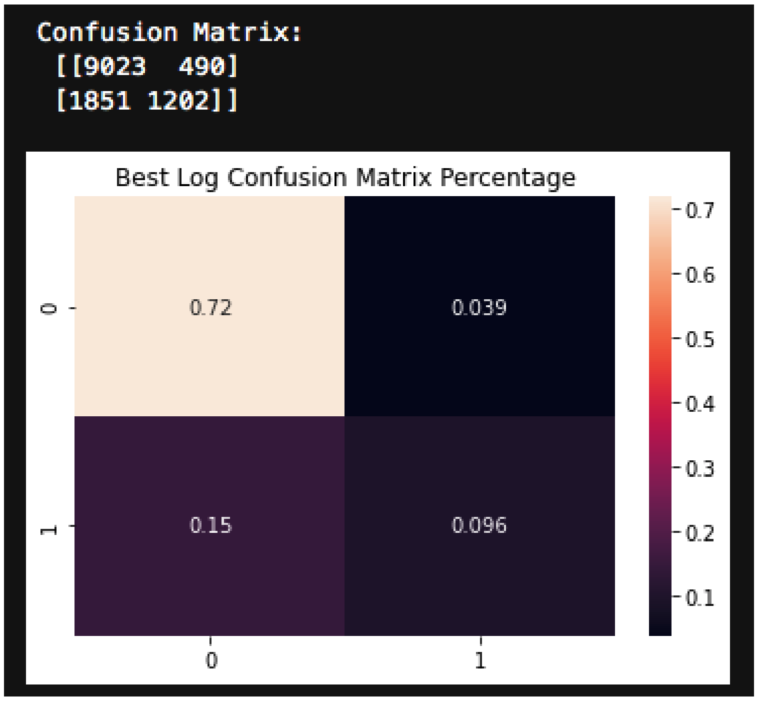


Figure 1

KNN	**0.5789353572708561**
LogReg	**0.8334390379789216**
Decision Tree	**0.8392621588490162**
Random Forest	**0.8408477113171009**

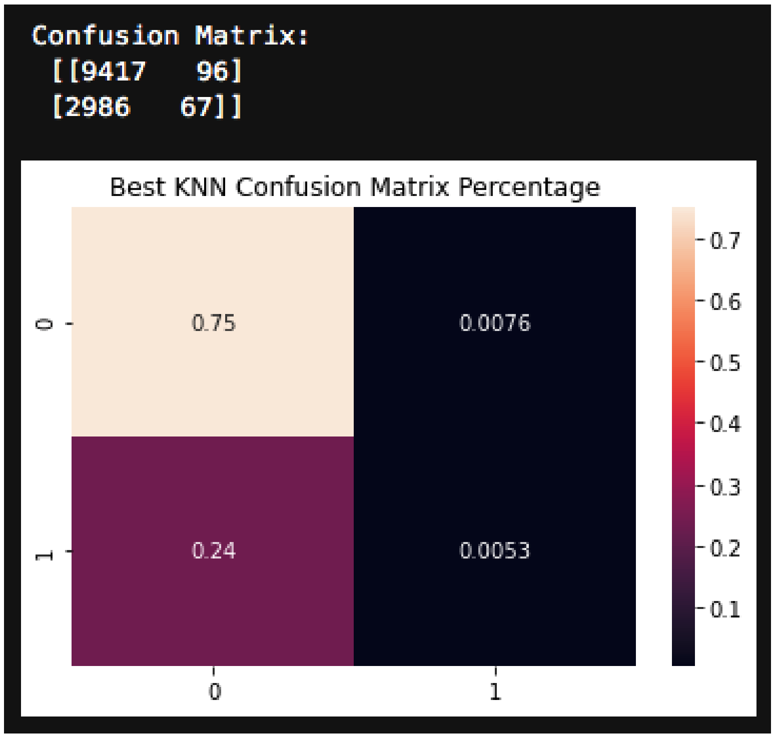
Table 1

In addition to the ROC curves, a confusion matrix for each model was calculated. The data from the confusion matrix was then visualized through a heatmap using seaborn in order to display the percentages of each quadrant. The correctly identified instances are in the diagonal (first and fourth quadrants) whereas the non-diagonal (second and third quadrants) are those that the model incorrectly classified. See Figures 2-5 for the confusion matrices below:



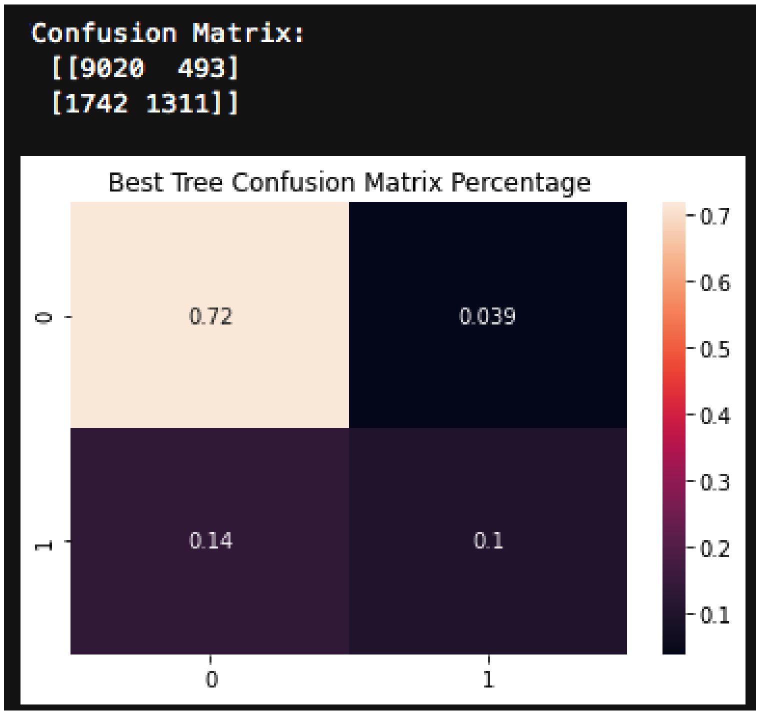
```
Confusion Matrix:
 [[9023  490]
 [1851 1202]]
```

Figure 2 – Best Log



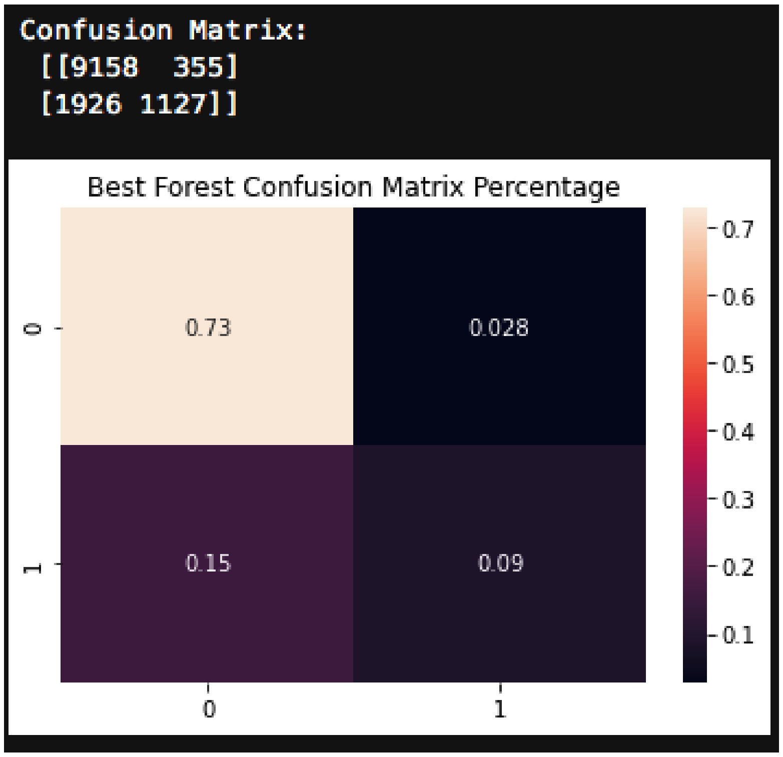
```
Confusion Matrix:
 [[9417   96]
 [2986   67]]
```

Figure 3 – Best KNN



```
Confusion Matrix:
 [[9020  493]
 [1742 1311]]
```

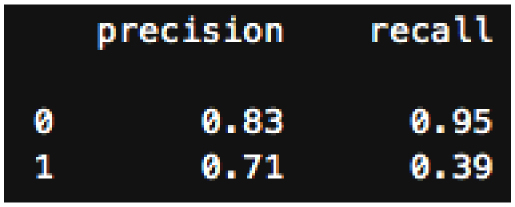
Figure 4 – Best Decision Tree



```
Confusion Matrix:
 [[9158  355]
 [1926 1127]]
```

Figure 5 -- Best Random Forest

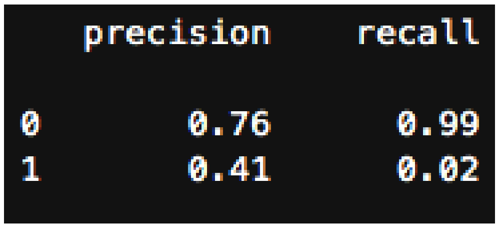
The third and final metrics used are precision and recall. See Figures 6-9 below:



```
   precision    recall

0       0.83      0.95
1       0.71      0.39
```

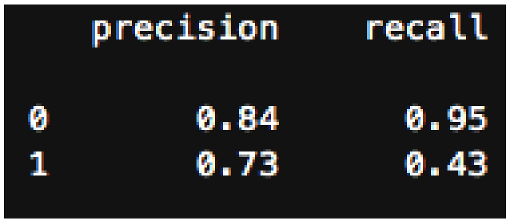
Figure 6 – Best Log



```
   precision    recall

0       0.76      0.99
1       0.41      0.02
```

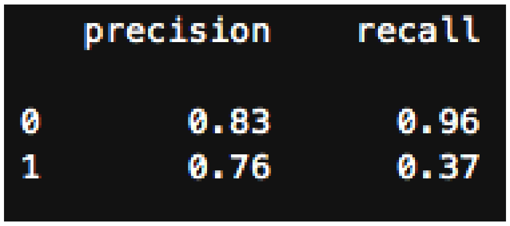
Figure 7 – Best KNN



```
   precision    recall

0       0.84      0.95
1       0.73      0.43
```

Figure 8 -- Best Decision Tree



```
   precision    recall

0       0.83      0.96
1       0.76      0.37
```

Figure 9 – Best Forest

4. Conclusion

Surprisingly, most of the models performed very similarly with one exception: the KNN. This model appeared to be an outlier. With an extremely low AUC score of 0.57. Furthermore, it also had the lowest precision among all the models. Looking at the confusion matrix in Figure 3, it is evident that the KNN almost never correctly predicted true negatives. In fact, it hardly predicted any false positives either. It’s possible that the dataset is not well suited for a KNN or that it is simply weaker in comparison to the other models for this type of classification problem. The other models, however, performed extremely similarly with AUC scores ranging from 0.833 to 0.840 with the Random Forest ranking the best. Beyond the AUC score, the confusion matrices, precision, and recall are almost identical. It’s possible that all these models perform similarly on this dataset, or, perhaps with more hyperparameter tuning, one model could pull ahead from the others and separate itself. Google Cloud API offers a three hundred dollar trial credit so perhaps in going forward with this project one could apply these models by renting a low cost supercomputer and adding more parameters to the GridSearch. All in all, these models provide better results than randomly guessing, so it seems that the original research question is answered: at least to some degree, demographic data is indicative of income.