University of Utah
School of Computing
CS6350 - Machine Learning

Final Competitive Project

Caleb Johnson

---

1 Dataset Overview

For this project, we are attempting to properly predict whether a person makes over $50,000 annually based on a number of features. These features include [age, working class, education, education number, marital status, occupation, relationship, race, sex, capital gain, and capital loss. Our output label is simply a binary classification for whether or not that individual made over $50,000.

We are provided both a training set and a testing test, with the training set having 20,000 data instances while the test set has 23,842 instances. The data is overwhelmingly clean, with few instances of missing or corrupted data.

2 Exploratory Data Analysis

To begin with, I wanted to take a look at some of the features, and get an idea of some of their behaviors before we do any preprocessing and feature engineering. We have 14 features, of which we have a mix of continuous and categorical variables.

One quick observation that help us is to look at the heatmap of the correlations between all features in our dataset. This will give us an idea of the co-occurence between variables that may affect our analysis. This heatmap can be seen in Figure 1.

3 Data Preprocessing and Feature Engineering

Our feature space contains a mixture of both categorical and continuous datatypes. For our use with sklearn models, we are going to need to encode our categorical variables into a more numerical input that the models can compare. To do this, I employed the LabelEncoder method from the preprocessing library within sklearn.

Additionally, our EDA gave us some insight into the correlation between features we have. For instance, relationship and sex demonstrated considerable correlation, which is understandable as relationship was predominately labeled as ”husband” or ”wife”. Removing these features would likely help reduce cooccurance. Additionally, and we will discuss this later in the paper, we can consider how engineering new features to replace current features could be beneficial. Currently, capital gain and capital loss appear to be similar

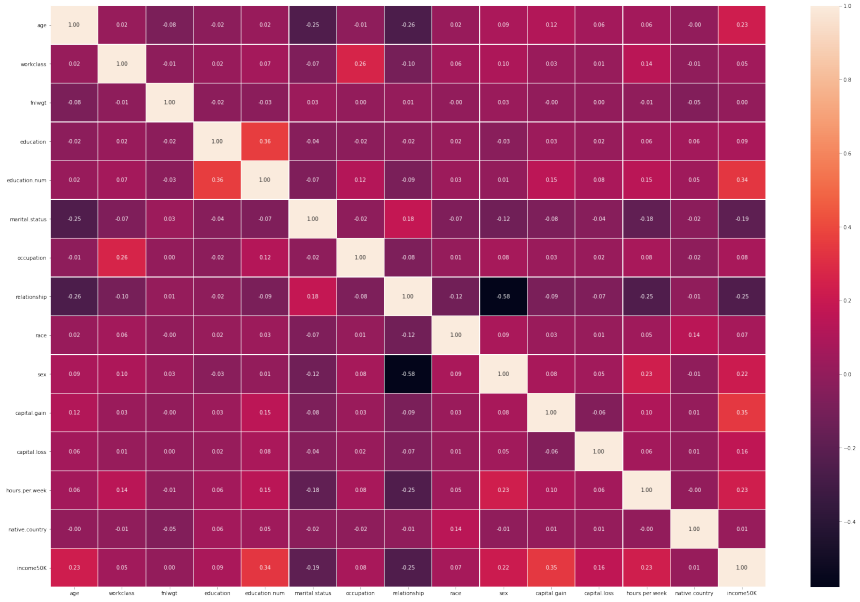


Figure 1: Feature Correlation Heatmap

continuous labels that could be combined. Education also appears to be much less influential that education number. We could consider removing this feature to maximize our classification.

Additionally, we had 1,848 rows that contained an unknown attribute, denoted as "?". Given our large sample size of 25,000, I decided to first try removing all of these rows from the dataset. This gave us a dataset now of just over 23,000 instances which seemed satisfactory for training purposes. When I ran the models below with this subset, I noticed that results seemed largely the same. I also tried replacing each "?" value with the mode response for this column. Overall this seemed to very slightly improve my results but not by a very large factor. I wasn't able to try nearest neighbor classifications, but I will discuss my ideas for that possibility in the future works section.

4 Model Cross-Validation

The first thing I attempted to do was to run a k-folds cross-validation on an assortment of models with no hyperparameter tuning. This method provides insight into the performance of a variety of machine learing models and gave me a baseline for which models appears to be handling this classifaction problem the best. The models that I selected to test were Naive Bayes, Logistic Regression, k-Nearest Neighbors, Random Forest, Decision Tree, and Support Vector. I am including the results from these different models in table 1.

5 Model 1: Logistic Regression

Simply based on my knowledge of logistic regression, and knowing its abilities as a binary classifier, it was the first model I tried to fit with the data. This was despite middling performance on our k-folds cross-validation, where it was outperformed by both Naive Bayes and Random Forest.

Similar to linear regression, logistic regression is a predictive analysis. It attempts to describe the relation-

k-Folds Cross-Validation Model Performance	
Model	Accuracy
Naive Bayes	0.8230
Logistic Regression	0.8117
kNN	0.7683
Random Forest	0.8589
Decision Tree	0.8063
Support Vector	0.7649

Table 1: Cross-Validation

ship between our dependent binary label (whether someone makes over 50K) and our independent variables (such as age and education level). The model applies a logistic function to linear model that allows us to achieve our binary classification. After implementing the model, we received very middling results. Testing on our split training set, we received accuracies still in the high 70%s. Finally, when I attempted to submit the results to Kaggle, we achieved an accuracy of only 0.7851.

In total, I question the Logistic Regression's utility when our submission outputs are probabilities compared to labels. I chose it as my first model, simply based on my inherent understanding of its underlying functions, despite not expecting to receive the greatest results.

6 Model 2: Random Forest

The second model that I chose to fit with the data was a random forest classifier. A random forest is a bagging ensemble model that leverages many unique decision trees. Each decision tree is build using a randomized subset of the feature space at each level. At these levels, the decision tree still splits on the feature that provides the most information gain, however the variability in feature space subsets allow for less overlap between trees in the ensemble.

6.1 Basic Model

I first attempted to fit the random forest model with default hyper parameter settings and without any feature engineering being performed. This was to provide me with a baseline for expectations, and receive cursory results to help me move forward with. When we scored our split training set, we received accuracy of around 86%, and our submitted probabilities received a score of 0.90329.

6.2 Hyperparameter Tuning and Feature Engineering

In an effort to improve our random forest model, I performed a grid search cross-validation over the hyperparameter input space. The model was run hundreds of times with different combinations of input values. After running for an extended period of time, the hyperparameter configuration that performed the best can be seen in Table 2.

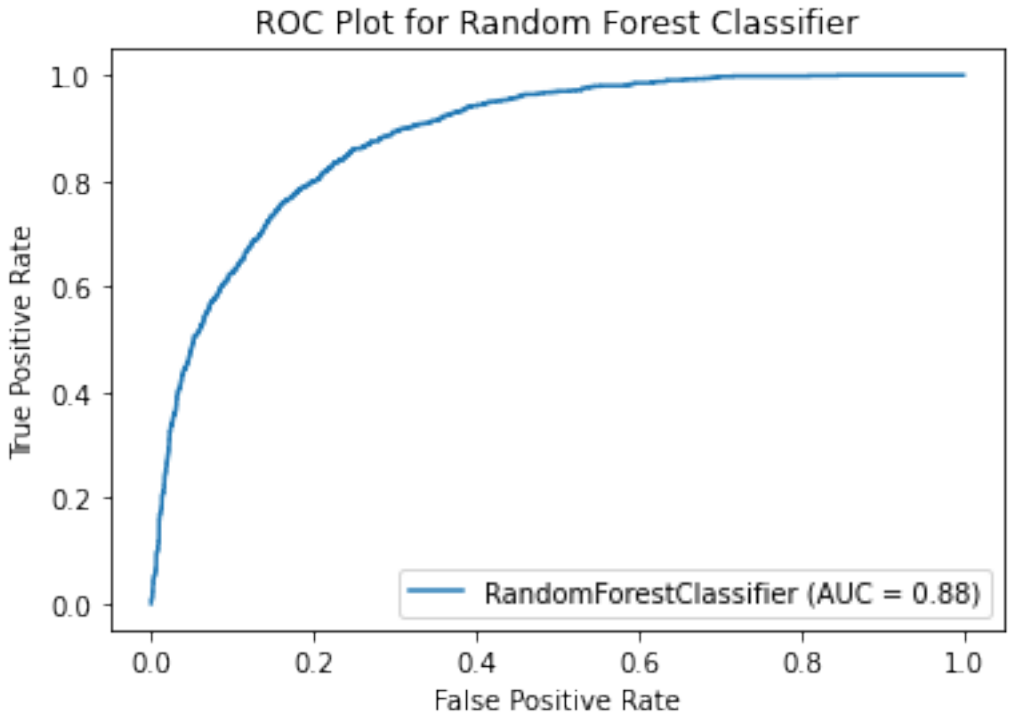
In addition, I decided to run the model on a reduced and modified feature space. After looking at the

Grid Search Cross-Validation Best Parameters	
Hyperparameter	Best Configuration
$n\_estimators$	2000
$min\_samples\_split$	2
$min\_samples\_leaf$	2
$max\_features$	”sqrt”
max_depth	20
$bootstrap$	True

Table 2: Cross-Validation

correlation heatmap and comparing their labeling, I decided that relationship and sex didn't provide much insight, especially when taking into account marital status. For instance, relationship was labeled predominately as ”husband” or ”wife”, both of which are handled well by both sex and marital status. Likewise, I decided to combine the capital gain and capital loss values into capital total. My first thought was to instead create a capital.net feature, but I decided that both capital gain and capital loss implied assets. Though their real-estate or investments may be losing value, I still felt that their possession implied a higher earning value.

Finally, when I ran the random forest classifier with the specified hyperparameter space and my engineered feature space, I received a score on Kaggle of just under 91.2%. This has been by far the best results that I have been able to achieve on my submission, though I do wonder why some of my models are falling behind. Below you can see the ROC plot for our random forest classifier, which had an area under the curve score of 0.88.



7 Model 3: Perceptron Variations

The third model that I tried to explore were different variations of the perceptron model that we designed in class. I got fairly comparable accuracies with both the standard and the max vote variations, of about 83% with my test data. Additionally, I tried fitting SKLearn's Multilayer Perceptron implementation and got similar results with minimal hyperparameter tuning.

8 Future Plans

I still have many different ideas and implementations that I want to explore. Firstly, I think I can do a lot better data processing to really maximize my results. Unknown datapoints have been one of the largest areas of interest for me throughout this project, and I have already tried two different techniques for handling them. First, I tried removing all row instances that contained unknown values, but this seemed to have a negligible effect in either direction for our analysis. Second, I tried to take the mode classification of the column that it appears in, which had slight effects but they didn't seem to be drastic. In the future, I would like to consider a nearest neighbor approach, where I find the K (defined perhaps as 5) most similar instances to the row with the unknown label, and then take the mode classification of this nearest neighbor set. I believe this would allow us to get a more nuanced value for these unknown instances, by looking at the most similar neighbors.

Additionally, I want to better understand why some of my more complex models are not performing better, such as SKLearn's Multilayer Perceptron and XGBoost. My testing scores when I run locally seem to be reasonable, however when I submit my predictions on Kaggle my scores seem to hover around the 75% mark, far below what I am expected them to be.

9 Code

All code can be found at the following GitHub repository. It was written in Python using Jupyter notebooks.
https://github.com/Calebdee/CS6350-FinalProject