Capstone Two Final Report

Introduction:

The problem for this capstone is: Can Bank of Antarctica use information about people collected in the census to predict if their income is likely above or below $50k a year in order to help decide whether they are eligible for a loan or not? My approach was to load a dataset of adult census data (adult.data) into a jupyter notebook and then create models of varying complexity to try and predict income from other information. I created four models: a logistic regression model, a decision tree model, a random forest classifier model and a gradient boosting classifier model. I then measured the performance of these models on test data using various metrics. According to accuracy and other metrics, the decision tree model, random forest classifier model and gradient boosting classifier model were all roughly tied in performance, while the logistic regression model performed markedly worse than these three models.

Problem Identification:

I chose this problem because I thought it was an interesting exercise in classification. Many problems in statistics focus on regression, but I thought that this problem was interesting because it instead focuses on classification. This problem deals with predicting income-and while income is often reported as a continuous variable, the census dataset only collected whether an individual's income was <=50K or >50K. Furthermore, I also appreciate another aspect of the problem I chose-the fact that this problem requires me to use a large amount and variety of the data processing/modeling skills that I've learned thus far. In order to properly and accurately predict income from other variables in this dataset-the data had to be cleaned properly, dummy variables had to be created for the categorical variables, the data had to be scaled properly, and multiple models using GridSearchCV for tuning had to be made.

The problem of predicting a bank loan applicant's income is very important for the Bank of Antarctica. The Bank of Antarctica considers an applicant's income to be an important variable in determining whether the applicant will pay back the loan. One reason income could be an important variable is because an applicant with higher income may have more discretionary income (not needed for survival costs) with which to quickly pay back the loan when required. The Bank of Antarctica decided to create a cutoff where they will grant loans to applicants with income >50K and deny loans to applicants with income <=50K. However, one problem is that the Bank of Antarctica does not always have direct data on an applicant's income and is not legally allowed to ask. Thus, instead, Bank of Antarctica commissions independent contractor to create a model with which they can accurately predict whether the

applicant is above or below the 50K cutoff based on other features about the applicant.

The Bank of Antarctica requires the model to be “good enough” at predicting whether an applicant earns less than or greater than 50K a year, otherwise the model isn’t useful. The Bank of Antarctica also would prefer but not require a model that is simpler, cheaper to implement, computationally inexpensive, and could be used in the cloud.

Data Wrangling:

At the beginning of this step, necessary libraries were imported, including pandas. Then, the correct working directory was set in order to make it easier to load in the dataset. Then, the dataset was loaded into the jupyter notebook using the pandas read_csv function. The dataset was inspected using the functions head(), info() and describe(). The sort_values() function was applied to several columns in order to see the spread of values in different columns. Describe() was also applied to one column. The isnull() function was used to attempt to find missing values in the dataset, and no missing values were found. The value_counts() function was applied to several columns in order to further see the different values in each column and their frequencies.

A new variable/column called ‘income_bool’ was created, which was essentially the same as the income column (which is the ‘label’ in this project) except with ‘>50K’ replaced with True and ‘<=50K’ replaced with False. This column makes it easier to perform mathematical operations that can be used to summarize/inspect the income of different groups. The groupby function was used sequentially several different times to create different datasets that grouped the original dataset by different categories. Some such categories/columns used for grouping were age, sex, country and education. Then, through tagging the .mean() function onto the end of the groupby() function, the proportion of each category that earns >50K could be assessed. It could be seen that ~0.31 of males in the entire project dataset earn >50K a year, while only ~0.11 of females earn >50K. Variation in income_bool could also be seen across countries, ages, education levels, etc.

Finally, histograms of columns in the dataset were created. These showed which values of each variable people tended to cluster around. These histograms made it easy to see the approximate mode of different variables in the dataset, and also the spread of values of each variable.

Exploratory Data Analysis:

At the start of this step, necessary libraries were imported (including pandas), a useful current working directory was established, and the dataset was read into the

jupyter notebook using pandas' read_csv. The function, .head(), was used to inspect some of the data, and all the columns were renamed with descriptive names for clarity. Since income was the column that I wanted to predict, and was thus one of the most important columns, I created a countplot of this column to get a sense of what its values were. Upon brief inspection, the countplot appeared to show that about ⅘ of people in the dataset earned <=50K and ⅕ earned >50K. There were no other values (e.g. missing data, etc) in the income column.

Since my prior knowledge suggests that age could be an important predictor of income, I printed some of the value_counts() of age to get a rough sense of some of that column's data. Using seaborn, I created a scatterplot of income (in terms of proportion earning >50K) vs age for all people in the dataset. This distribution appeared to roughly follow some sort of bell curve (for no obviously apparent reason. I would assume this bell curve is likely not a normal distribution) and proportion of people earning >50K peaked at around age 50 (at ~0.40). The proportion earning >50K gets steadily lower as one examines ages younger than 50 years, or older than 50 years. There were a couple outliers where ~3 age years between 70 and 90 appeared to earn a lot more than their surrounding ages, but this could possibly be due to a few individuals. It was seen from the earlier step of running value_counts() on the age column that there were very few people in some of the older years, and so it wouldn't take many people to drastically change the proportion earning >50K.

A barplot was also made of proportion earning >50K at each age year. This figure makes it easier to see the proportion earning >50K at each age year than the scatterplot did. A similar barplot was then made of capital gain vs age year. It can be seen from visually comparing this barplot to income vs age year that capital gain positively correlates with income. The fact that these barplots shared similar peaks and valleys at similar age years points to capital gain as a possible good predictor of income.

Barplots of income vs education and capital gain vs education were made, and suggests that education level strongly influences income and capital gains. A seaborn pairplot was made plotting variables in the dataset against each other and showing correlations (or lack thereof) between different variables. Such a plot can suggest which variables would be useful in predicting income and which would not. The plot can also suggest which variables are strongly correlated with one another. If two variables that strongly predict income are highly correlated with each other, then it may pay to only use one of these variables in order to speed up prediction time (and use fewer computational resources) and also because you may be double counting one underlying cause (e.g. if both variables reflect the same, single, underlying factor).

I proceeded to make a separate seaborn barplot for income vs each separate explanatory variable. Then, I created a histogram for each separate explanatory

variable. Performing the step of making each barplot showed visually what kind of relationship each explanatory variable had with income. The histograms showed the distribution of values of each explanatory variable.

Overall, I thought that I performed enough data exploration to get a satisfactory overview of the underlying data, and after this step, feel ready to proceed to preprocessing and modeling.

Preprocessing and Training Data Development:

The preprocessing of the data started with importing the necessary libraries, in this case pandas. Then pandas' read_csv function was used to load the census dataset (adult.data) saved to the hard drive into a pandas dataframe within the preprocessing jupyter notebook. The pandas .head() function was used to get a quick look at some of the data. The columns of the dataset were renamed to be more descriptive and clear. The dataset was once again examined using .head().

Then, the train_test_split function was imported and the X and y variables were created. X was created using all of the columns of the dataset except for income and y was created using only the income column. Then, dummy variables were created to hold the data of the categorical columns in X. This is necessary in order for machine learning algorithms to work correctly on the data. Then, the data was split into X_train, y_train, X_test and y_test and the test size was 0.2. Finally, a scaler was fit based on X_train and then applied to X_train and X_test. Since these preprocessing steps proceeded successfully, at the end of this notebook the data was now ready for modeling.

Modeling:

Four models were run-logistic regression, decision tree, random forest classifier and gradient boosting classifier. After each model was instantiated, its hyperparameters were roughly tuned with GridSearchCV and a parameter grid. Then, predictions from each model were made using the best parameters found by GridSearchCV, and predictions were compared to values in the test set via various scores. A single accuracy score was printed for each of the four models. In addition, a classification report and a confusion matrix was generated for each of the four models.

The model accuracies were as follows (rounded to three decimal places): Logistic Regression: 0.805 Decision Tree: 0.856 Random Forest Classifier: 0.859 Gradient Boosting Classifier: 0.865.

Thus, logistic regression had noticeably lower accuracy than the other three models. Overall, gradient boosting classifier had the highest accuracy. However, the three models, decision tree, random forest classifier, and gradient boosting classifier all

had very similar accuracies that were all within 0.01 of each other. And so, it is hard to say based on accuracy alone which of these three models will make the best predictions.

Final Results and Conclusion:

To get a full picture of a models performance, more than just an accuracy score must be examined. This is why I printed a full classification report and confusion matrix for each model. Choosing the best model may take into account multiple scores from the classification report as well as the runtime, complexity, and computational load associated with the model. The precision scores for each model for the class “earns >50K” and the class “earns <=50K” may also be relevant. If a certain business problem demands that a model have correct labeling for a very high percentage of a certain class, then looking at each model’s precision score for each class can be used to further distinguish which model should be used.

The three models of decision tree, random forest and gradient boost all have amazingly similar precision, recall and f1-scores across the board (incl. for both classes of >50K and <=50K), which makes them very hard to distinguish. Thus, because these three models all score so similarly across multiple different scoring methods, I think it is best to pick the decision tree. Occam’s Razor says always pick the most simple solution, and the decision tree is also likely less computationally expensive to run again than random forest or gradient boost. Lastly, decision tree is the easiest to understand intuitively out of those three models. Thus, based on work done so far, I would choose the decision tree model to implement in a business context.

One obvious idea for future research would be to try tuning all of these four models (but particularly the three with similar scores) using more different hyperparameters. Due to limited time (and slow runtime of tuning), only a limited set of hyperparameters were tried for each model, using GridSearchCV. However, a longer future investigation could try far more hyperparameters for each model. Doing so may make it clear which model makes the best predictions. It is possible that once far more hyperparameters are tried for each model, there will be more differences between different models’ final best scores.

Another idea for further research would be to perform more cross validation for each model by possibly cross validating each model across 5 folds. This also may allow for better hyperparameter tuning.

Yet another idea would be to try to find a larger version of the census dataset that was used: one that contains more entries. If such a dataset was found, training models on such a dataset could increase the accuracy/performance of the models.

In addition, the veracity of individuals' responses in the census dataset (including for the income column) could be further researched.

Lastly, more different kinds of models could be tried, including support vector machines and neural nets.

One recommendation for how a client could use my findings is to implement the decision tree (since I chose that to be the best model, all things considered). Since the Bank of Antarctica has education level, age, sex and other information on loan applicants (except for income information), it would be able to feed the 'features' of any given applicant into this decision tree model and the model could predict whether the applicant earns <=50K or >50K. If this model predicts loan eligibility (or the ability/likelihood to pay back a loan) better than a human/existing model, then it can save the bank money and increase the banks profitability by preventing the bank from granting as many loans that won't be paid back. It may also increase profitability if it recommends granting loans to individuals who wouldn't otherwise be granted loans (but are going to pay the loan back), and the bank earns interest from these repayments from these additional individuals.

This thus provides a quick (because the decision tree is less computationally expensive than other models) automated method to accept or reject a loan applicant. Furthermore, it is possible that removing human decision from the interaction lessens the bank's risk of liability. For example, it is possible that it is harder for an applicant to make a case that they were rejected unfairly or for illegal and discriminatory reasons if the decision is made by an algorithm (hopefully the algorithm would be tuned to avoid any form of discrimination). Avoiding lawsuits would save the bank money.

One other recommendation is that the client, the Bank of Antarctica, could consider implementing this decision tree in the cloud and creating an online tool (linked to this decision tree model) that allows customers to check whether they would be accepted or rejected for the loan before they decide to apply for the loan. Such a tool could be made quick and easy for customers to use because they would only have to answer a few short online questions about their own information in order to see if they would be accepted or rejected. This may not improve the banks profitability directly, but may increase customer/applicant satisfaction with the bank, which may improve the banks image/profitability in the long run.

One last recommendation is that if the decision tree model is implemented and found to continually make good predictions in the real world over an extended period of time, the Bank of Antarctica could consider selling this model (or a tweaked version) to other Banks or related businesses.