A POTPOURRI OF CLASSIFICATION METHODS ON THE ADULT DATASET

BRENDAN WHITAKER

ABSTRACT. We analyze the `Adult` dataset which contains a wide variety of demographic characteristics for working-age adults, along with a class we wish to predict: salary $> 50K$ or $\leq 50K$. The dataset was sourced from the UC Irvine Machine Learning Repository hosted by the Center for Machine Learning and Intelligent Systems. We first discuss our exploratory data analysis and transformations to prepare for classification. We then train five different classification models on our data and evaluate them on the provided test set to determine which is the preferred method, taking into account performance statistics such as accuracy and F-measure.

CONTENTS

1. EXPLORATORY DATA ANALYSIS

We give some preliminary information about the data. The `Adult` dataset contains metadata for 32561 individuals from ages 17-90. There are 14 axes of demographic information like `education`, `marital-status`, `race`, `sex`,.. etc, which can be utilized to predict the salary class. One of these is a `final-weight` real-valued variable which is supposed to be a "aggregate" measure of the rest of the data points which should correlate with salary, so people with similar demographics should have similar final weights, according to the documentation for the dataset. The publishers also note that this statement is only true for individuals residing in the same state as an artefact of the way the data was collected. We will examine later on how useful this axis is for capturing the variation in the data. The data was imported and analyzed using `python 2.7` in the Jupyter Notebook web application. The `NumPy`, `Matplotlib`, and `pandas` python libraries were imported in order to aid in data exploration, cleaning, and predictive modeling. the data was imported in a `pandas` dataframe, a structure similar to an Excel workbook.

Date: SP18.

We note here one of the first pre-processing issues encountered with this dataset: the lack of pre-existing column headers in the `.csv` file. In fact, the dataset was not even saved on the source website with a file extension, so it was unclear at first if it was even this type of file at all. But a cursory examination in a text editor revealed comma delimiters. But dealing with the lack of column headers proved to be quite an issue due to a bit of bad luck. In attempt to remedy this, a line was added to the top of the document with the column headers from the data description on the source website. A great deal of frustration revealed that the column headers are whitespace-sensitive, and that if spaces are left after the commas in the header as is common in prose, one must also include these spaces in any attempt to reference the data.

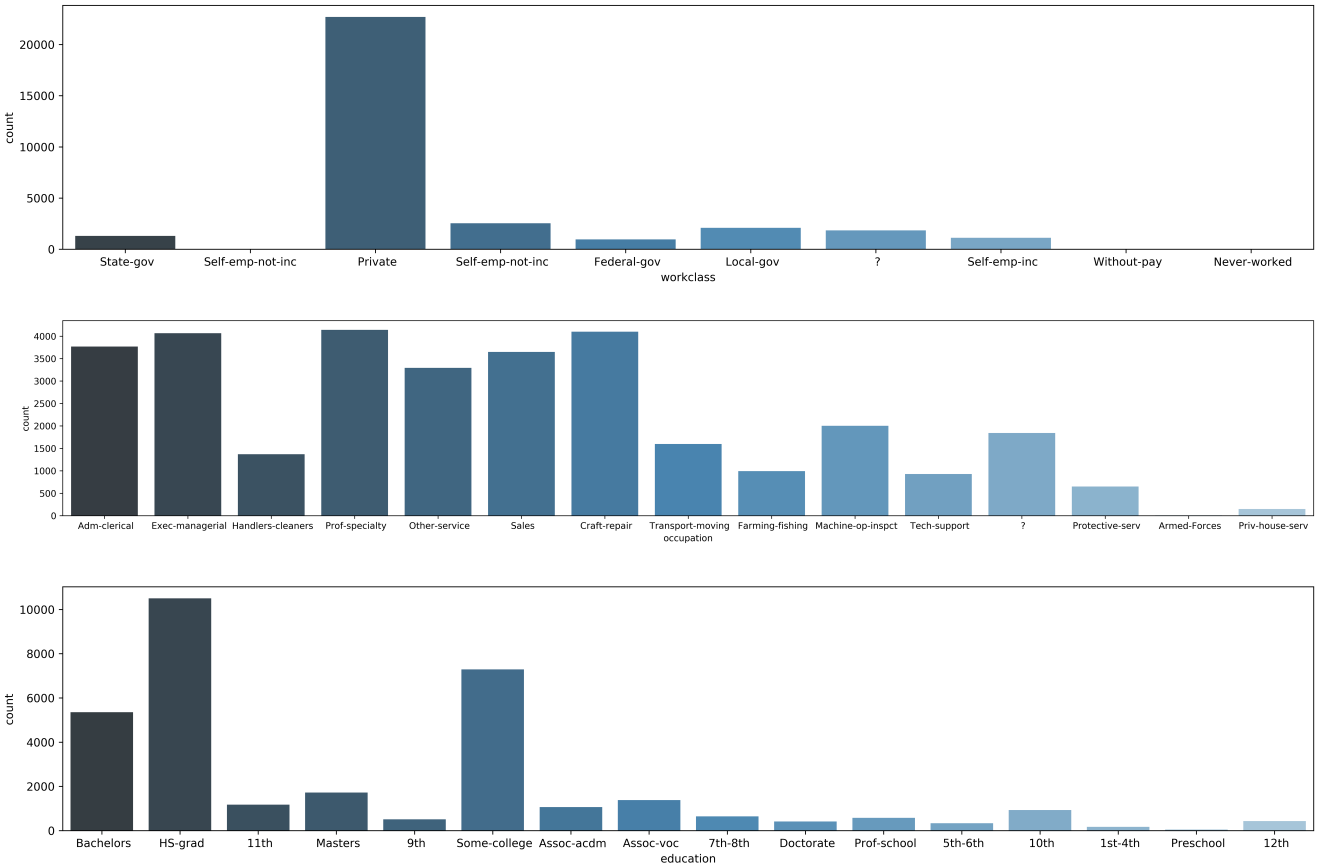

FIGURE 1. Categorical variable distributions continued.

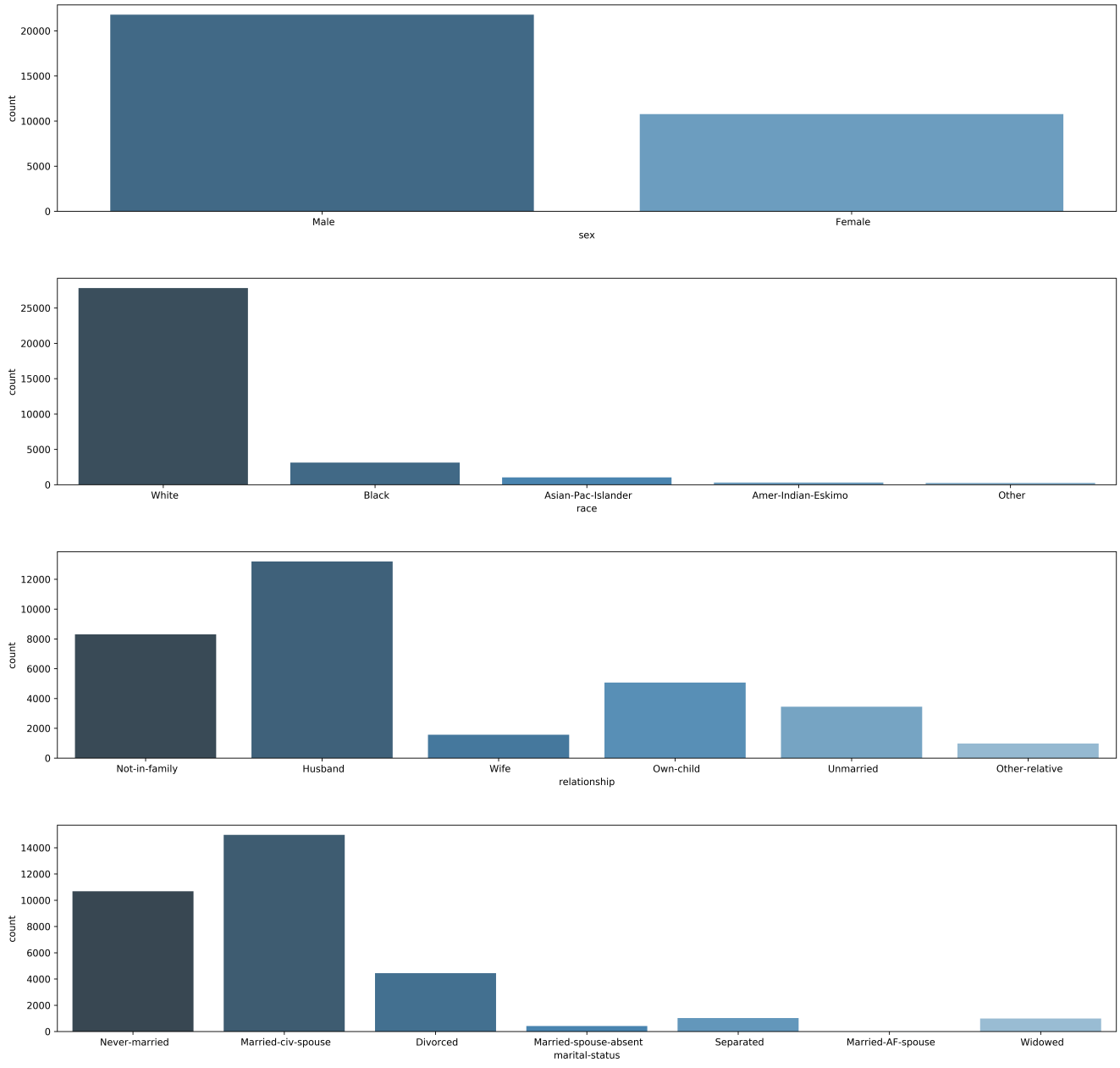

FIGURE 2. Categorical variable distributions.

We now give a more detailed summary of each of the variables, and perform any transformations before we begin training the models. We wish to omit any variables within which there is very little variation in the data, so

we use visual analysis of the frequency distributions given above in Figures 1 and 2. Although the chart has too many different categories to include it in this report, the `native-country` feature has `United States` for around 29,000 of its entries, thus we omit it since there is very little variation captured by it. We can see from the rest of the bar charts that most of the other variables capture a good deal of variation, though if we wanted to eliminate another, a good candidate might be `race`, since a vast majority of the adults are `white`. `workclass` would also be a good candidate since a large majority of those entries fall under `private`.

2. Data transformations

We choose not to omit any entries or attempt to reduce noise because these features of the dataset may well serve to be useful to the model and improve its performance on the test set. Deleting entries to solve issues like the missing/unidentified data in the `workclass` barchart may hurt the model more than it helps it, especially if the data on other axes of these removed entries were meaningful.

Since the model implementations only work with numeric data, we must first convert all the categorical data into numeric data. We will do this using a process known as label encoding. If there are k distinct labels for a given categorical variable in our data, we assign each variable a number 0 through $k-1$ and use this as our numeric data.

Below we see a table of simple summary statistics for the numerical variables from our set. Rather than looking at each of the pairwise correlations between these variables, we instead perform principal component analysis on it. We will reduce dimensionality as much as possible while preserving at least 95% of the variation in our numerical data. PCA is a good idea here since we have a variable `final-weight` which allegedly already represents several of the other variables present.

	age	final-weight	education-num	capital-gain	capital-loss	hours-per-week
count	32561.000000	3.256100e+04	32561.000000	32561.000000	32561.000000	32561.000000
mean	38.581647	1.897784e+05	10.080679	1077.648844	87.303830	40.437456
std	13.640433	1.055500e+05	2.572720	7385.292085	402.960219	12.347429
min	17.000000	1.228500e+04	1.000000	0.000000	0.000000	1.000000
25%	28.000000	1.178270e+05	9.000000	0.000000	0.000000	40.000000
50%	37.000000	1.783560e+05	10.000000	0.000000	0.000000	40.000000
75%	48.000000	2.370510e+05	12.000000	0.000000	0.000000	45.000000
max	90.000000	1.484705e+06	16.000000	99999.000000	4356.000000	99.000000

Table 1. Summary statistics for numerical variables.

We begin our PCA by creating a new DataFrame containing all of our numerical variables and the newly transformed categorical variables which are now real-valued thanks to our label-encoding. Thus we will be able to do PCA on all 13 remaining axes (we nixed the `native-country`). We will be using the `scikit-learn` library to standardize our feature values and perform the analysis. We note here that any transformations applied to the training data must also be applied in parallel to the test data so that we have sets with the same formatting when we go to test our models. So although we only fit PCA on the training data, we must apply it to both sets, and the same goes for our normalization. After applying PCA with 12 principal axes, we compute the percentage of variation of the original data preserves, which comes out to 96.58%. So we are above our threshold, and thus we've successfully reduced dimensionality by 1.

3. Model development

We note here our choice to use the training and test partition given to use by the source website instead of performing n-fold cross validation. We do this to avoid computation time issues since the dataset is quite large and the hardware at hand is sub-ideal. The five models we will be training are as follows:

(1) Decision Tree Classifier
(2) 3-layer Neural Network
(3) Support Vector Machine
(4) Ensemble Classifier (Decision Tree using Random Forest)
(5) Logistic Regression

We discuss the `scikit-learn` parameters used for each of the models. For the decision tree, we set a max depth of 3 and the minimum number node size to be 5 samples. As is intuitive, we found that a greater max depth yielded incrementally higher accuracy along with significantly longer training duration. Decreasing the min node size had a similar effect, and we conjecture that setting this to 1 or similarly low values for large datasets would make the model prone to overfitting. For the neural network, we used a 3-layer perceptron and with an arbitrary choice of 12 nodes per layer since we have 12 principal axes. We found that in testing numbers of layers ranging from 3-6, the training duration/computation time increased dramatically for larger numbers of layers. We found a similar effect for increasing the number of nodes per layer. However we settled on these relatively tame settings to make for a fair comparison between the five models. Significant increases in the nodes and layers gave us an increase in accuracy of less than 2 percent. For the SVM, we set the parameters $\gamma = 0.001$ and $C = 100$. We obtained a small < 4 percent increase in accuracy by using these settings over the default `scikit-learn` settings for the model. For the random forest, we set the number of trees to be 100 and the max number of features to be 3. Similar to the results with the neural net, we say a small increase in

accuracy when increaseing the number of trees and the max number of features per node, but this was offset by a significant increase in computation time, and this the reason for the parameters chosen. The implementation of the logistic regression model we used did not support parameter tuning, and so we used the defaults.

Our results are summarized below (refer to figure), where `DTREE` is the decision tree model, `NN` is the neural network, `SVM` is the support vector machine, `RF` is the random forest ensemble classifier, and `LOGREG` is logistic regression. Immediately standing out is the performance of the neural net, which had both the highest accuracy of all 5 models and the highest F-measure. However there is one more factor to consider, which is computation time. The neural net, along with the decision tree and the logistic regression model were all remarkably quick to train, however the support vector machine and the random forest were quite slow in comparison, each taking roughly 5 times longer to compute than the other three combined. For this reason, the clear winner is the neural network, since it had marginally higher accuracy than all the others, it's F-measure was highest and only approached by the random forest, and it was able to complete training in a relatively short time period. At least as far as the data we have collected in this experiment is concerned, there don't seem to be any significant drawbacks to choosing the neural network over any of the others, but we should note that it's accuracy gains over the other models are nothing to write home about: all of the models are in the 79-85% range, and they all have similarly close F-measures as well. One drawback to the methodology of the entire experiment is the slight loss in classifier accuracy incurred by using PCA. This, however, I think is often offset by quicker model training as a result of the lower dimensionality, especially with larger datasets with higher dimensionality, and when we are using complex models like the neural network or SVM.

```
number of principal components:  12
variation preserved in PCA:  0.965761836268
DTREE Accuracy is  79.8169645599
DTREE F-measure is  57.124217119
[[10806  1629]
 [ 1657  2189]]
             precision    recall  f1-score   support

          0       0.87      0.87      0.87     12435
          1       0.57      0.57      0.57      3846

avg / total       0.80      0.80      0.80     16281

NN Accuracy is  84.9333579018
NN F-measure is  63.599940644
[[11685   750]
 [ 1703  2143]]
             precision    recall  f1-score   support

          0       0.87      0.94      0.91     12435
          1       0.74      0.56      0.64      3846

avg / total       0.84      0.85      0.84     16281

SVM Accuracy is  83.6926478718
SVM F-measure is  55.7868442964
[[11951   484]
 [ 2171  1675]]
             precision    recall  f1-score   support

          0       0.85      0.96      0.90     12435
          1       0.78      0.44      0.56      3846

avg / total       0.83      0.84      0.82     16281

RF Accuracy is  84.2884343714
RF F-measure is  62.5695054141
[[11585   850]
 [ 1708  2138]]
             precision    recall  f1-score   support

          0       0.87      0.93      0.90     12435
          1       0.72      0.56      0.63      3846

avg / total       0.83      0.84      0.84     16281

LOGREG Accuracy is  82.4457957128
LOGREG F-measure is  54.184033344
[[11733   702]
 [ 2156  1690]]
             precision    recall  f1-score   support

          0       0.84      0.94      0.89     12435
          1       0.71      0.44      0.54      3846

avg / total       0.81      0.82      0.81     16281
```

FIGURE 3. Results of classification model testing.