Annalisa Wilde
DATA71200: Advanced Data Analysis
Final Project
May 22,2020

Modelling Who Works over the Age of 65

Introduction

There is a lot of evidence that people are working longer. Bureau of Labor Statistics data published in 2008 showed an increase in the workforce participation by people over the age of 65 had been increasing from 1977 to before the recession, with the biggest increases in women over the age of 65 and both sexes over the age of 75 (Bureau of Labor Statistics 2008). More recent data from after the recession shows that that trend has continued through at least 2016 (Pew Research Center 2016).

What I want to know is who is it that is working longer? Is everyone working longer, or are only certain groups working longer, depending on what kind of work they do? In my projects I explored the predictability of who is working, what are the factors that most influence who is working and if there are clear clusters around these categories. I wanted to know if I could predict who was retired or in full or part-time employment.

I chose this dataset because I was interested in the subject matter and because I wanted a project that was based on census data so that I could familiarize myself with the data structure itself. I wanted the practice of getting the data and getting it into a workable form in python. The learning associated with this puzzle was a major motivator for choosing this data.

Data Cleaning

The data for this analysis came from the IPUMS database for the US Census Bureau's Current Population Survey, specifically the income and occupation data from Annual Social and Economic Supplement (ASES) (Flood 2020). For this analysis I am using the yearly survey data from 2009 for people aged 65 and older. The total dataset was 21,402 people. The data included also immigration, race, ethnicity, and education metrics, as well as gender and marital status.

For my preliminary analysis I used python to convert the fixed-width ".dat" file to a data frame that I could turn into a csv file and to update the coded values to be human read-able. Almost all the categories were numerically encoded, which is probably good for machine learning but extremely hard to understand as a human wanting to explore the data. The code books were easiest to deal with by parsing them into data frames themselves and selecting the columns that I wanted. Next I looked at the various features themselves.

To start, my data had 90 variables from the survey, but initial analysis of the data showed that a number of those variables were null for the entire data set. I dropped all of those, plus some metadata columns including various IDs and dates for the survey and was left with 65 variables. Those variables included the following:

cpsid `"CPSID, household record"'
ownershp `"Ownership of dwelling"'

housret `"Return to home equity"'
proptax `"Annual property taxes"'
pubhous `"Living in public housing"'
rentsub `"Paying lower rent due to government subsidy"'
age `"Age"'
sex `"Sex"'
race `"Race"'
marst `"Marital status"'
yrimmig `"Year of immigration"'
citizen `"Citizenship status"'
hispan `"Hispanic origin"'
empstat `"Employment status"'
occ `"Occupation"'
ind `"Industry"'
classwkr `"Class of worker"'
uhrsworkt `"Hours usually worked per week at all jobs"'
uhrswork1 `"Hours usually worked per week at main job"'
wkstat `"Full or part time status"'
educ `"Educational attainment recode"'
wkswork1 `"Weeks worked last year"'
whyptly `"Reason for working part-time last year"'
ftotval `"Total family income"'
inctot `"Total personal income"'
incwage `"Wage and salary income"'
incbus `"Non-farm business income"'
incfarm `"Farm income"'
incss `"Social Security income"'
incwelfr `"Welfare (public assistance) income"'
incretir `"Retirement income"'
incssi `"Income from SSI"'
incdivid `"Income from dividends"'
incrent `"Income from rent"'
incalim `"Income from alimony"'
incother `"Income from other Source not specified"'
inclongj `"Earnings from longest job"'
oincbus `"Earnings from other work included business self-employment earnings"'
oincfarm `"Earnings from other work included farm self-employment earnings"'
oincwage `"Earnings from other work included wage and salary earnings"'
srcreti1 `"First source of retirement income"'
srcreti2 `"Second source of retirement income"'
increti1 `"Retirement income from first source"'
increti2 `"Retirement income from second source"'
incsurv1 `"Survivor benefits income from first source"'
incsurv2 `"Survivor benefits income from second source"'
incdisa1 `"Disability income from first source"'
incdisa2 `"Disability income from second source"'
srcearn `"Source of earnings from longest job"'
gotvdisa `"Received veterans' disability compensation"'
gotvothe `"Received other veterans' payments"'
gotvpens `"Received veterans' pension"'
gotvsurv `"Received veterans' survivor benefits"'
taxinc `"Taxable income amount"'
hourwage `"Hourly wage"'
union `"Union membership"'
earnweek `"Weekly earnings"'

To reduce the number of features in my dataset, I combined some of the features that represented the same kinds of income from different sources. The values combined and the output variables are below. I then used the incomes that weren't from wages to get a passive_inc value. It was the all_wages and all_business_farm values subtracts from the INCTOT value.

incwage, oincwage => all_wages
incbus, oincbus, incfarm, oincfamr => all_business_farm
incdisa1, incdisa2 => all_disability
incsurv1, incsurv2 => all_survivor

I also wanted a numeric value to measure "citizenship." For this I calculated "percent of life" that was percent of the life of the person that they had spent in the US. I calculated that by doing the year the person immigrated subtracted from 2009 divided by their age.

Because I was looking at employment, I wanted to keep information about if a person had multiple jobs even as I combined the income from them, so I created a binary flag "multi_jobs" that was set to true if the person had a value for oincwage. I also was interested in capturing the relationship between the a person's taxable income (TAXINC) vs their total household income (FTOTVAL) and the relationship between their total income (INCTOT) and their taxable income (TAXINC). I created the ratio_tax_inc that was TAXINC divided by FTOTVAL, and reported_inc_diff that was INCTOT minus TAXINC.

The value that I wanted for my "label" was split between two columns, EMPSTAT and WKSTAT. EMPSTAT had information about if the person was not in the labor force because they were retired or because they were unemployed. All working people were just "working". WKSTAT had the different levels of working: part-time, full-time, and "has job, has not worked in the last week." To get a single variable that described the employment and working status, I combined the EMPSTAT and WKSTAT variables into a variable called EMPWKSTAT that had the following values, shown with their distributions below. NILF means "not in the labor force."

```
NILF, retired                    15865
Full-time                         1756
Part-time                         1715
NILF, unable to work              1084
NILF, other                        547
Unemployed                         225
Has job, not at work last week     210
```

I then used the label encoder from the sklearn library to encode them into numeric values so that I could more easily create plots with them.

After visualizing, discussed more below, my final steps of cleaning were to one-hot encode the categorical data and, after doing a stratified split of the training and the test data, to scale the data and impute the value of 0 in any numeric features that still had null values. I also added default "NIU" (not in universe) values to the categorical features that still had null values.

For the supervised learning portion, I had a number of features that were only present if a person was working or highly correlated to working so they were dropped. These included: household income, total income, weeks worked, income from longest job, why part-time, taxable income, hourly wage, union membership, reported income difference, all wages, class of worker, hours worked, source of earnings, ratio of taxable income, multiple jobs, occupation, industry, and all income due to businesses and farms. I ended up not adding these back in for my project 2, but I would like to do more clustering of the data with these values added back, perhaps only looking at the data of those who are working.

Looking at the Data

After visualizing the data initially, a few things stood out. First, age seems to be binned at the top end of the spectrum, with people over the age of 80 all being put in one bucket and people over the age of 85 in another. This is ok as the older people are much less likely to be working. Second, the household income (FTOTVAL), hourly wage, income from retirement funds, your income from your longest job, and total income all follow a "normal curve for income" in this country, which is to say that they are significantly skewed to the right. Lastly, income for social security appears to follow a normal distribution, except for the large number of those with 0 who have yet to take their disbursements. This is probably by design but I don't know enough to say more. It also seems like they have "binned" anyone who gets more than $50,000 in social security, or this is the maximum yearly disbursement that you can get. Additionally, the visualizations confirmed what I already knew, which is that many of the features in my data are sparse, with the majority of individuals having a value of 0.

I also visualized the relationship between income due to social security and income due to retirement funds across the different values of EMPWKSTAT. For people who are retired, there is interestingly a relationship that looks like the area under a binomial curve. There also a surprising cluster of people who get a lot from social security but have little income from retirement funds. Part-time workers were more likely than full-time workers to have income due to retirement funds, but both groups had a lot of people who were getting social security payments but had no income due to retirement funds. The people who are not in the labor force who are "unable to work", and those that are "other" have similar relationships between retirement income and social security income. They are exceptional compared to other groups because the concentration of income to social security is mostly below $20,000. Other groups have a cluster stretching up to about $25,000.

Supervised Learning

The first model that I ran was a logistic regression on the EMPWKSTAT variable. Logistic regressions attempt to find a linear "decision boundary" between two groups, maximizing the probability that different groups will be on different sides of the boundary. The linear boundary is described by a function that includes an intercept and coefficients for each of the features which determine the degree to which they impact the slope of the line. Because my label variable has multiple classes in it, the model is using a "one-vs-rest" approach where it goes label class by label class and attempts to find the best decision boundary for each and then applies the data-points to each label class's model and the one that has the highest probability wins.

The initial accuracy of the logistic regression model with the default parameters was .76 on the training and .75 on the test. I then used cross validation in combination with a grid search to try different values of the hyperparameter to see if there was a better model that could be generated.

The hyperparameter for logistic regression is *C*, the inverse of regularization. Regularization is a limiting factor on the maximum value that a feature's coefficient and is meant to reduce the influence that outliers can have on the decision boundary during the model training. Because *C* is the inverse of regularization, small values of C mean more regularization.

My grid search result found that the highest value of C, 100, or the least amount of regularization, produced the best result. The default C is 1. However, that "best result" was the same as my initial result, with a prediction accuracy on the test set of .75. Running the model with the hyperparameter of 100 produced an f1-score of .86 for retired people and .29 for full-time workers. This is .01 higher than the f1-score for the default model. People "unable to work" had an f1-score of just .07 and for not-working "other" it was .01. Neither the default model or the model with a C of 100 could predict anyone who was part-time, unemployed, or had a job but had not worked in the last week.

Because I'm a nerd, I tried a series of different complex models on my dataset, including a Gaussian Naïve Bayes model, a simple decision tree model, a random forest classifier model, and a gradient boosting decision tree model. For sake of brevity, I will just discuss the model that had the best performance, which was the random forest model.

A random forest model creates a lot of small decision trees based on a random starting set of features. A decision tree makes a series of progressive splits to maximize the homogeneity of the groups of the target variable based on a given pool of features. The "splits" are different values of the predictive features that group the different target variables into the most heterogeneous groups. The resultant groups are called "leaves." In the random forest classifier, the pool of possible predictive features is kept fairly small but random for each tree. After all the trees are generated, the predictive values they have generated are then combined to produce the overall model.

The default random forest classifier model predicted less accurately than the tuned logistic regression, with .734 accuracy on the test data. Its f1-scores, however, were mixed. It predicted full-time workers and those who are unable to work better than the logistic regression with .33 and .13 respectively. The prediction of retired people was .85, .01 less than the tuned logistic regression. It also predicted some part-time people where the logistic regression had predicted none. The remaining classes were predicted (or rather not predicted) at the same rate as the logistic regression.

The hyperparameters that I used for tuning the random forest classifier models are the number of trees to build, the maximum features to include in each tree, and the minimum samples that could be in each leaf. The most performant random forest had 50 trees with a maximum of 20 features and a minimum of 5 samples per leaf.

The gains in classifying and accuracy are not large for any of these models. This is the "best" and it had a prediction accuracy of .76. The f1-score for full-time workers was .36, and the retirees were back up to the tuned logistic regression's value of .86. However, the "unable to work" prediction was less than logistic regression at .04, and the model stopped predicting part-time or non-working "other" people at all.

Given the relatively small changes in the prediction values, it is hard to say exactly what it is about my dataset that might generate those changes. In general, looking at the coefficients in the logistic regression (I updated my project two to include a graph of them), it seems like full-timers compared to retired people are frequently on different sides of 0, meaning that things that make it more likely that you are working full time also make it less likely that you are retired. Because of this relationship, the models are good at predicting those two categories. the features that most.

Looking at the coefficients for part-time however, they are "all over the map" relative to the other two categories. Sometimes the part-time coefficients are in the middle and sometimes they are more extreme than either of the other two, sometimes they are close to the full-time values and sometimes they are close to the retired values. There were no particular features that had a really large coefficient that influenced this class. This is different from the "unable to work" category which the models all were able to make some accurate predictions on, but not many. It had a number of relatively large coefficients compared to the retired category and I think this is what allowed it to stand out a bit more as a category. In both cases, however, the overall shape of the groups compared to the others meant that it remained difficult for all of the models to predict them.

I do think it is interesting that for the logistic regression less regularization of the variables was better. I think because there are so many categories in my target variable and they are very specific, the shapes of the decision boundaries that form around them need to have a pretty specific, tight-fitting decision boundary.

Principle Component Analysis

The principle component analysis (PCA) to capture 95% of the variance reduced my features (after one-hot encoding) from 100 to 26. This did not, however, improve the performance of the tuned random forest classifier. Overall accuracy on the pca test set was .005 below the regular test set. The f1-score for full-time as slightly less, for retirees was the same, but for "unable to work" was better, and there was an f1-score above zero for both the not working "other" category, as well as the "part-time" category.

PCA and Clustering

The PCA pre-processing made the clusters within my data more viewable on two dimensional graphs. However, the clusters were still not particularly distinct by the scoring metrics.

The k-Means cluster model starts by randomly picking a series of central points to group your data into a pre-determined number of clusters. It assigns all the points to the closest "center." Then it finds the mean of those data points, to which it moves the center. It then reassigns the data points based on the new "centers" and finds the new mean. It does this until it has reached a point of stasis where the centers don't move. For my data set, the "elbow" test of inertia was 12, which is a lot of clusters to begin with. Without PCA, there is no visible clustering of my data viewable on a two-dimensional graph. With PCA it becomes more clear, but there is still a lot of overlap and the adjusted Rand Index scores are very low for both. The silhouette score is slightly better for the PCA than the non-PCA-ed data but not only by .01, .13 versus .12.

For agglomerative clustering, the algorithm starts off with each individual sample and then combines them together in a way that minimizes the variance of the clusters. It keeps doing this iteratively until it reaches a specified number of clusters. For my dataset the specified number were 6, because those were the number of categories in my training data. PCA-ed data, again made the clusters much easier to visualize on the two-dimensional plane, but had very little impact on the “scoring” of the clusters. The clusters had the same density score via the silhouette score between the PCA-ed and non-PCA-ed data. For the adjusted Rand Index, the score was .01 for PCA-ed compared to .00 for non-PCA-ed. Which is abysmal and indicates that the clusters that do exist do not align at all with the labels in my target variable.

DBscan clustering follows the proximity of points by starting in a random place and then, based on an epsilon value, including points that are within a certain “distance” from the existing points in the cluster. It continues doing this until it runs out of points that are close enough to the existing cluster. There is a is also a minimum samples parameter that allows you to set the minimum number of samples that could be in a cluster. This clustering algorithm performed badly on both my non-PCA-ed and PCA-ed data, earning negative adjusted Rand Index and Silhouette scores, with the PCA-ed data being a bit more negative than the non-PCA-ed data.

Results

Across the three projects I learned a lot. I got to do the python data-parsing with census data. I learned, obviously, a lot of machine learning algorithms. A mistake I made in my project 1 that I had to fix in my project 2 was that I one-hot encoded my data after splitting my test and my training. This resulted in my training data having an extra categorical value that wasn’t present in my test data. Plus, I had an imputer that handled the null values for my categorical columns, but not for my numeric columns, so I had to add that in. I also didn’t originally get a list of all of my feature names anywhere so that I could meaningfully plot them. I wrote the extra steps to get my processed data back into a data frame before the final separation into test and train so that it would have those values.

I was surprised to see that the predictive capacity of all of the many models was so similar. I think it says a lot about the dataset that it basically has a predictive capacity of 75% across ever algorithm. I was also surprised to see in the feature importances’ for the decision tree model how much of a factor marital status seemed to play in who worked.

For fun, I also tried the supervised learning algorithms on a dichotomous “isWorking” variable. This was easier for the model to predict, with most of the algorithms getting between .83 and .85 accuracies. Interestingly again, however, there wasn’t a whole lot of variability between the algorithms, with the exception of the Gaussian Naïve Bayes default model which performed significantly worse, getting only an accuracy of .66. With tuning this improved to .83, but it did not predict anyone who wasn’t working.

If I had to do it over I would have changed the clustering project to focus on the set of seniors that were working and to include the variables related to work, like hourly wage and occupation to see what clusters might have come out of the subset who are working. I would also like to dig deeper into how to interpret the clusters. What are good ways to take the results of the agglomerative clustering? And what is the best way to use it to generate topologies of the groups that make up the cluster? That is something I would like to know how to do.

Notes

Bureau of Labor Statistics. 2008. “Older Workers: BLS Spotlight on Statistics.” https://www.bls.gov/spotlight/2008/older_workers/.

Pew Research Center. 2016. “More Older Americans Are Working than in Recent Years.” *Pew Research Center* (blog). https://www.pewresearch.org/fact-tank/2016/06/20/more-older-americans-are-working-and-working-more-than-they-used-to/.

Flood, Sarah, Miriam King, Renae Rodgers, Steven Ruggles and J. Robert Warren. 2020. Integrated Public Use Microdata Series, Current Population Survey: Version 7.0 [dataset]. Minneapolis, MN: IPUMS. https://doi.org/10.18128/D030.V7.0