Predicting Unemployment Status

Submission for the 2022 MEBDI Machine Learning Competition

Robert Winslow

June 2022

Abstract

I use a set of machine learning tools to predict next-year unemployment status for individuals in the Current Population Survey, using a subset of variables specified by the MEBDI competition judges. I use a Decision Tree algorithm, and two regularized linear regressions, and I improve the classification performance by combining these three classifiers using a hard-voting ensemble. This combined classifier is able to predict future unemployment in a holdout testing set with 70.95% accuracy and future non-unemployment (employment or not-in-labor-force) with 79.92% accuracy, for an average accuracy across classes of 75.44%.

The most important predictive features are a person's current work/employment status. Essentially, the best signal that someone will be unemployed next year is that they don't have a job this year.

1 Introduction

What factors predict future unemployment at the indvidual level? This was the question motivating the 2022 MEBDI Machine Learning Competition.

Judges provided participants with subset of variables describing several hundred thousand individuals from the IPUMS Current Population Survey dataset. (Sarah Flood and Westberry 2021) Each individual in the provided dataset was interviewed sometime between 2008 and 2015, with a followup interview one year later.

The variable to predict was whether a person indicated that they were unemployed during this followup interview, and the allowable predictors were a set of 23 variables from the initial interview, describing a person's occupation, demographics, education, and earnings.

Because only 5 percent of surveyed individuals were unemployed, an algorithm trained to maximize the *accuracy* of predictions would end up mostly ignoring the group that we are most interested in.[1] As such, algorithms were evaluated on their *balanced class-conditional accuracy*, described by the following goodness of fit formula:

$$GF \equiv \frac{\text{\# Correctly Predicted Unemployed}}{\text{\# Unemployed}} \cdot \frac{1}{2} + \frac{\text{\# Correctly Predicted Not Unemployed}}{\text{\# Not Unemployed}} \cdot \frac{1}{2}$$

In the case that two algorithms ended up with very similar scores (as did happen), the winner of the competition would be decided by the first term above, representing the accuracy of predictions made about individuals who were unemployed during their followup interview.

My algorithm, which used a combination of decision trees and regularized linear regressions, was able to achieve a balanced accuracy score of $GF \approx 75.44\%$. My algorithm's predictions on the testing subsample are summarized by the following confusion matrix:

Confusion Matrix for My Algorithm's Predictions

True Status	predicted NU	predicted U	Total
Not Unemployed	63144	15862	79006
(portion of row)	(0.7992)	(0.2008)	(1.0)
Unemployed	1278	3122	4400
(portion of row)	(0.2905)	(0.7095)	(1.0)

Of the 4400 individuals in the testing sample who reported being unemployed during their followup interview, my algorithm correctly predicted the status of 3122 of them, for a class-conditional accuracy of 70.95%. Similarly my algorithm correctly predicted the status of 79.92% of those individuals who weren't unemployed during the followup interview.

The rest of this note is as follows:

- In Section 2, I describe the data cleaning and feature engineering in more detail. Of particular note, I engineer a single binary variable which is able to achieve a score of $GF \approx 73\%$ when used as the only predictor.
- In Section 3, I describe the machine learning techniques I used for my algorithm, how I combined them, and discuss additional techniques which were not used in my final submission.
- Section 4 describes the results of my classification algorithm.
- Finally, the Appendix contains additional technical details, lists of variables, etc.

[1]You could achieve 95% accuracy simply by predicting that nobody will be unemployed.

2 Input Data

2.1 Data Source and Variables

The data for this competition comes from The IPUMS CPS database.(Sarah Flood and Westberry 2021)

The judges selected a sample from this data consisting of individuals who: - were interviewed sometime between 2008 and 2014 (inclusive) - had a followup interview the following year - and between 20 and 64 years old (inclusive) at the time of the initial interview.

A random subset of this sample was chosen and set aside as the testing set, while the rest of the data was designated as the training set. The training set, which consists of data from 333514 individuals, was sent to competitors at the beginning of the competition. The testing set, consisting of 83406 individuals, was withheld from competitors until after all code was submitted. Final scoring for the algorithms was done by evaluating each algorithm on the testing set.

The dependent variable to predict was a binary variable called `y_tp1`[2] in the training set. The value of `y_tp1` is set to 1 if an individual is unemployed during that second interview,[3] and 0 otherwise. A label of `y_tp1` means that the individual is *not unemployed*; they may be either employed or NILF (Not in the Labor Force).

A set of 23 independent variables were provided as predictors. The full list can be found in the appendix.

Finally, the training data included variables `cpsidp` and `cpsidp`, representing personal and household identification numbers. The majority of households (57%) had multiple individuals represented in the training data. In principle, one could use the value of `cpsidp` to link individuals together and use, say, a person's earnings as a predictor for their spouses' employment status. But I did not attempt to do this analysis.

2.2 Data Merging (Or Lack Thereof)

For the sake of having a level playing field, so that the results would depend solely on the algorithms used, the judges instructed competitors *not* to merge any data which was not included in the training set. As such, I provide two notes about how this data should thus be handled differently outside of a competition:

Firstly, earnings numbers are not adjusted for inflation. The allowable predictors include a variable labelled `earnweek`, representing the respondent's typical weekly earnings, as they were expressed to the interviewer. This means that numbers are in current-value US dollars, and the IPUMS documentation recommends that the data user adjust for inflation using the Consumer Price Index.

Secondly, IPUMS provides variables `hwtfinl` and `wtfinl` to be used in adjusting for sampling bias when preparing population-level statistical estimates. This is less of a concern for this particular application, where we're trying to look at individual-level characteristics. But one worry is that if a particular demographic is underrepresented in the training data, then a machine learning algorithm might fair poorly when trying to generalize its insights to people in that demographic.

2.3 Data Cleaning and Feature Engineering

Full details about how the input data was modified can be found in the appendix. But here are a few things of note:

[2]You can read "`y_tp1`" as "the value of y, our dependent variable, at time period *t plus one*.

[3]More specifically, this means that the individual's time *t+1* `empstat` was coded as categories 20,21, or 22 in the original data. They were 'Unemployed', 'Unemployed, experienced worker', or 'Unemployed, new worker'.

2.3.1 Data Cleaning

The IPUMS variable `earnweek` is numeric, and top-coded to 2885 dollars per week. Values of 9999.99 for this variable, however, represent individuals NIU (Not In Universe), which for this variable means people without a job or the self-employed. I imputed any employed people to median weekly earnings, and imputed all other missing earnings to zero.

The variables `ahrsworkt` and `uhrsworkt` were dealt with in a similar manner.

2.3.2 The most informative single variable.

One of the first things I did when exploring the data was check to see whether there were any simple heuristics with high performance. I evaluated each individual category as a 1-variable predictor[4], and found that far and away, the most informative was `empstat_At work`, which by itself achieves a balanced accuracy score of $GF \approx 72\%$. I then evaluated every combination of `empstat` to find the one which was most informative. The variable `empcombo` is set to 1 unless `empstat` is of 'Armed Forces', 'At work', or 'Has job, not at work last week'. A similar process was used to make `wkcombo` from the most informative combination of `wkstat` categories. The `empcombo` and `wkcombo` variables can each achieve $GF \approx 73\%$.

It takes quite a bit of work to gain small improvements over the simple heuristic of *"predict someone will be unemployed next year iff they don't currently have a job"*.

2.3.3 Other Feature Engineering

Categorical variables were converted to dummy variables. In some cases I engineered dummies with overlapping categories. For example, I included dummies for not just specific industries and occupations, but also for broad industry and occupation groups.

The `educ` variable indicates a person's highest level of education. I created nested "`educn`" dummies, representing whether a person has a given level of education *or higher*. That way I can, for example, have a coefficient relating to the effect of graduating high school, instead of just the effect of "*graduating high school and not going to college*" specifically.

Overall employment trends intuitively have a strong effect on an individual's job search prospects. Because we couldn't merge a time series for employment statistics into the data,[5] I calculated my own series for `y_tp1` as a function of (`year`,`month`) from the training data, and then smoothed this time series out to create a variable called `period_unrate`. I did something similar with age to create `byage_unrate`.

I also added variables for the logarithm of weekly earnings `logearnweek`, and for estimated hourly wages `earnhour_est`. The latter variable is calculated from typical weekly earnings and hours, both of which are based on self-reports instead of administrative data. As a result, the estimates for hourly earnings are very noisy and full of implausible values.

2.3.4 Feature Selection and scaling

Numeric variables were scaled to unit variance.

A variance threshold of 0.001 was applied to the dummies, which removed categories with a prevalence of less than approximately 300 people in the training data set. This left me with 632 predictors.

[4]What I actually did was just take each dummy variable column and its complement and compare it to the `y_tp1` column. But that's functionally equivalent.

[5]Even without the judge's restriction, this would be a terrible idea because of data leakage. Employment statistics time series are calculated from the CPS, and thus contain information about the testing set.

3 Machine Learning Algorithms

3.1 Decision Tree Classifiers

The first method I explored was the use of binary decision trees (Loh 2011). This kind of algorithm classifies individuals by asking a sequence of binary questions. At each stage, the algorithm picks a single variable (and a threshold if the variable isn't binary) which it uses to split the data into two subsets. The training process picks the variable and threshold which maximizes some measure of information gain.[6] This process can then be individually repeated for each subset, and in this way a tree is grown. For example, the first step of my final decision tree asks whether `empcombo` is equal to 0 or 1.

Decision trees have the advantage of being very transparent algorithms with easily visualizable decision functions. I felt they were a good first choice to explore the data, and some of my data processing decisions were made by examining decision trees trained on minimally cleaned data.

To prevent overfitting,[7] I set up a 5-fold cross-validation with a randomized parameter search to identify good regularization parameters. I left this parameter search running overnight, and chose regularization parameters based on the results the next morning. It turned out that there were a wide range of regularization parameters with similar performance. I suspect this is because the bases of the the generated trees were very 'stable' across different parameters, partially due to the fact that the first decision node is so informative.

I also explored the use of random forests, where many different decision trees are fit on subsets of the data set and their results are combined. This can boost the performance of decision trees by mitigating their instability. However, in this case, the gains in scoring were negligible, which I suspect is again related to the base of the tree being stable and informative.

Furthermore, while my decision trees had decent balanced accuracy scores, their accuracy for the class of to-be-unemployed people was disappointingly low, and I didn't anticipate that this would change much from further improvements to a decision-tree-based model. So rather than pursuing techniques such as boosting, I decided to try a completely different approach.

3.2 Regularized Linear Regressions

The next thing that came to mind was to use a linear model. As expected when dealing with so many colinear inputs, fitting a straightforward OLS results in unreasonably large coefficients.

LASSO is a modified version of OLS which adds a term to the minimand proportional to the L1 norm of the coefficient vector[8]. I turned the resulting LASSO regression into a binary prediction vector by setting a threshold at 0.05144, which is the portion of the training set for which `y_tp1` = 1.

I used grid search cross-validation to identify the optimal weight to placed on the LASSO's regularization term. And settled on a value of $\alpha = 7.5e - 5$. To save computation time, I started by evaluating a small number of potential αs, distributed logarithmically. I plotted the value of α vs the resulting average cross validation scores, eyeballed the region that seemed most promising, and then repeated the process with another grid of points focused on that region.

Ridge Regression is very similar to LASSO, except that the regularization term uses a Euclidean L2 norm instead of the L1 norm. Using the same method as for the LASSO regularization parameter, I

[6]The particular implementation I used, from the sklearn library for python,(Pedregosa et al. 2011) uses a metric called Gini Impurity to decide how best to split the data.

[7]An unrestricted tree of depth 19 would have more leaf nodes than there are individuals in the training sample. This would lead to an accuracy of 100% on the training set but terrible performance on the test set.

[8]In other words, the regression is penalized not just by the sum of squared errors, but also by the sum of the absolute values of the coefficients.

selected a regularization parameter of $\alpha = 1000$ for the Ridge Regression.[9]

Both regularized linear models were significant improvements over the decision tree, especially when it came to making accurate predictions about the to-be-unemployed.

3.3 Hard Voting Ensemble

The performance of classification algorithms can often be improved by combining the output of several classifiers in an ensemble classifier. The underlying idea is essentially the same as the notion of "the wisdom of the crowd".

For this competition, I used the simplest ensemble possible. I took the binary predictions of each of the three classifiers described above, and used a simple majority vote to determine the final prediction for each data point.

The resulting simple ensemble achieved higher GF scores than any of the three individual classifiers. The improvement was very small but consistent across cross-validation testing.

4 Results

4.1 Scores on the Testing Data Set

The following is a summary of the performance of each classifier and the ensemble. GFU (Goodness of Fit for Unemployed) is the class-conditional accuracy for to-be-unemployed individuals with `y_tp1` = 1.

Scores on Testing Data		
Classifier	GF	GFU
Decision Tree	746.26%	65.614%
LASSO	75.060%	70.591%
Ridge	75.365%	72.386%
Majority Vote	75.439%	70.955%

Confusion Matrices for each classifier can be found the Appendix.

4.2 Top Variables

The following table summarizes the most important features in the decision tree.[10]

	Variable Description	Gini Importance
1	Neither employed nor in the military. (`empcombo`)	0.77778
2	Works at a Private, for profit, firm. (`classwkr_`)	0.06022
3	"Iffy" full or part-time status.[11] (`wkcombo`)	0.03896
4	Married, spouse present (`marst_`)	0.0293
5	Unemployment rate during followup interview. (`period_unrate`)	0.02486
6	Construction Industry. (`indg_`)	0.01364
7	Has earned an associate's degree or higher. (`educn_`)	0.00758
8	Initial interview takes place during 2008. (`year_`)	0.00489
9	Full-time hours (35+), usually full-time. (`wkstat_`)	0.00314
10	Has earned a bachelor's degree or higher. (`educn_`)	0.0028

[9]This is a very heavy regularization, but that's not unreasonable when there are hundreds of variables, many of which are nearly perfectly correlated.

[10]As measured by how much binary decisions based on that variable reduce a metric called Gini impurity.

Largest Coefficients in LASSO regression:

	Variable Description	Coefficient
1	Unemployed, seeking full-time work. (`wkstat_`)	0.10027
2	Neither employed nor in the military. (`empcombo`)	0.09591
3	Hasn't worked in the last year. (`occ1990_NIU`)	0.04984
4	Armed Forces industry (`indg_`)	-0.04158
5	Personnel supply services industry. (`ind1990_`)	0.0322
6	Unemployed, experienced worker. (`empstat_`)	0.02771
7	NILF, other.[12] (`empstat_`)	0.02474
8	"Iffy" full or part-time status. (`wkcombo`)	0.02179
9	Grandchild of head of household. (`relate_`)	0.02136
10	Racial identity includes Black. (`racec_`)	0.01989
11	Highschool diploma or higher. (`educn_`)	-0.01555
12	Works at a Private, for profit, firm. (`classwkr_`)	0.01461
13	Construction Industry. (`indg_`)	0.01435
14	Married, spouse present. (`marst_`)	-0.01428
15	Child of head of household. (`relate_`)	0.01283

Largest Coefficients in Ridge regression:

	Variable Description	Coefficient
1	Unemployed, seeking full-time work. (`wkstat_`)	0.06739
2	Neither employed nor in the military. (`empcombo`)	0.05397
3	Unemployed, experienced worker (`empstat_`)	0.04557
4	Hasn't worked in the last year. (`occ1990_NIU`)	0.04044
5	Armed Forces industry. (`indg_`)	-0.03851
6	Armed Forces. (`empstat_`)	-0.03375
7	Personnel supply services industry. (`ind1990_`)	0.02756
8	Unemployed, seeking part-time work. (`wkstat_`)	-0.02652
9	NILF, other. (`empstat_`)	0.02245
10	Grandchild of head of household. (`relate_`)	0.02206
11	NIU, blank, or not in labor force. (`wkstat_`)	-0.02065
12	Self-employed (`class_selfemp`)	-0.01985
13	"Iffy" full or part-time status. (`wkcombo`)	0.01783
14	NILF, retired. (`empstat_`)	-0.01688
15	Not a wage or salary worker. (`paidhour_NIU`)	0.01586

As discussed in Section 2, a person's current work or employment status by itself is a powerful predictor of their employment status next year. It is thus no surprise that many of the coefficients with the largest magnitudes are related to these variables.

4.3 Speculation and Just-so Stories

Some stories in the results seem straightforward. The construction and personnel supply services industries have positive coefficients, I'd guess because these kinds of work are based on sporadic projects.

[11]`wkcombo` is the maximally informative combination of `wkstat` categories. To simplify, it indicates that a person is either NILF, unemployed, or employed but working unusual hours because of economic reasons. See the appendix for further details.

[12]Meaning that the person is Not In the Labor Force, but is able to work, and is not retired.

Meanwhile, being in the armed services *by definition* prevents someone from being unemployed for the duration of their enlistment, so only those near the end of their enlistment have even the possibility of being unemployed the following year.

That someone is living under their parents or grandparents is a predictor of unemployment. My guess is that rather than this living arrangement *causing* unemployment, this variable is capturing the effect of some other cause. Adults who find themselves in this living arrangement may be more likely to be struggling in ways not captured by survey questions about age, education, occupation etc. Or they may just be 'struggling to get off their feet' in some difficult to pinpoint way.

The negative coefficient for `Married, spouse present` is a bit of a mystery to me. My first guess was that it is related to people leaving the workforce to raise children. But looking at the initial interview's employment status, people who are married with their spouse present are *more* likely to be employed, and *less* likely to be Not In the Labor Force for 'other' reasons.

REFERENCES

Friedman, Jerome H. 2001. “Greedy Function Approximation: A Gradient Boosting Machine.” *Annals of Statistics*, 1189–1232.

Gal, Yarin, and Zoubin Ghahramani. 2016. “Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning.” In *International Conference on Machine Learning*, 1050–59. PMLR.

Klambauer, Günter, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. 2017. “Self-Normalizing Neural Networks.” *Advances in Neural Information Processing Systems* 30.

Loh, Wei-Yin. 2011. “Classification and Regression Trees.” *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 1 (1): 14–23.

Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, et al. 2011. “Scikit-Learn: Machine Learning in Python.” *Journal of Machine Learning Research* 12: 2825–30.

Sarah Flood, Renae Rodgers, Miriam King, and Michael Westberry. 2021. “Integrated Public Use Microdata Series, Current Population Survey: Version 9.0 [Dataset].” Minneapolis, MN: IPUMS. https://doi.org/10.18128/D030.V9.0.

Smith, Leslie N. 2018. “A Disciplined Approach to Neural Network Hyper-Parameters: Part 1–Learning Rate, Batch Size, Momentum, and Weight Decay.” *arXiv Preprint arXiv:1803.09820*.

5 Appendix

5.1 Confusion Matrices for Test Data Set

The following are the confusion matrices for the the algorithm evaluated on the testing data set.

The columns correspond to the predictions made by a classifier. The rows correspond to the true value of time t+1 unemployment status, with `y_tp1 = 1` meaning the person is Unemployed the year following the collection of input data about them, and `y_tp1 = 0` meaning the person is Not Unemployed the following year.

5.1.1 Individual Classifier Confusion Matrices

Decision Tree:

	predicted NU	predicted U	total
y_tp1 = 0 (NU)	66079	12927	79006
y_tp1 = 1 (U)	1513	2887	4400

Lasso:

	predicted NU	predicted U	total
y_tp1 = 0 (NU)	62833	16173	79006
y_tp1 = 1 (U)	1294	3106	4400

Ridge:

	predicted NU	predicted U	total
y_tp1 = 0 (NU)	61896	17110	79006
y_tp1 = 1 (U)	1215	3185	4400

5.1.2 Final Ensemble Confusion Matrix

Majority Vote:

	predicted NU	predicted U	total
y_tp1 = 0 (NU)	63144	15862	79006
y_tp1 = 1 (U)	1278	3122	4400

This means that of the 4400 individuals in the holdout test sample who are unemployed in the following year, the combined classifier correctly predicted that 3122 would be unemployed, but incorrectly predicted that 1278 of these people would *not* be unemployed.

Simlarily, for the 79006 individuals in the holdout test sample who are not unemployed in the following year, the combined classifier correctly predicts this status for 63144 of them.

5.1.3 Test Data Confusion Matrices, in Percentage Form.

The following tables convey the same information as above, but cells no longer display the count. Instead, they display each cell as a percentage of each row. Thus the top-left and bottom-right display the class-conditional accuracies.

Decision Tree:

	predicted NU	predicted U
y_tp1 = NU	83.638	16.362
y_tp1 = U	34.386	65.614

Lasso:

	predicted NU	predicted U
y_tp1 = NU	79.529	20.471
y_tp1 = U	29.409	70.591

Ridge:

	predicted NU	predicted U
y_tp1 = NU	78.343	21.657
y_tp1 = U	27.614	72.386

Majority Vote:

	predicted NU	predicted U
y_tp1 = NU	79.923	20.077
y_tp1 = U	29.045	70.955

5.2 Confusion Matrices for Training Data Set

The following are the confusion matrices for the the algorithm evaluated on the testing data set:

5.2.1 Individual Classifier Confusion Matrices

Decision Tree:

	predicted NU	predicted U	total
y_tp1 = 0 (NU)	263981	52377	316358
y_tp1 = 1 (U)	5834	11322	17156

Lasso:

	predicted NU	predicted U	total
y_tp1 = 0 (NU)	250415	65943	316358
y_tp1 = 1 (U)	4880	12276	17156

Ridge:

	predicted NU	predicted U	total
y_tp1 = 0 (NU)	246527	69831	316358
y_tp1 = 1 (U)	4603	12553	17156

5.2.2 Final Ensemble Confusion Matrix

Majority Vote:

	predicted NU	predicted U	total
y_tp1 = 0 (NU)	251607	64751	316358
y_tp1 = 1 (U)	4856	12300	17156

5.2.3 Training Data Confusion Matrices, in Percentage Form.

The following tables convey the same information as above, but cells no longer display the count. Instead, they display each cell as a percentage of each row. Thus the top-left and bottom-right display the class-conditional accuracies.

Decision Tree:

	predicted NU	predicted U
y_tp1 = NU	83.444	16.556
y_tp1 = U	34.006	65.994

Lasso:

	predicted NU	predicted U
y_tp1 = NU	79.156	20.844
y_tp1 = U	28.445	71.555

Ridge:

	predicted NU	predicted U
y_tp1 = NU	77.927	22.073
y_tp1 = U	26.83	73.17

Majority Vote:

	predicted NU	predicted U
y_tp1 = NU	79.532	20.468
y_tp1 = U	28.305	71.695

5.3 List of Variables Used for Analysis.

5.3.1 IPUMS CPS Variables

The training data included the following subset of variables from IPUMS CPS.

The Type column indicates how I use each variable in my estimators, with `N` being numeric variables, `C` being categorical variables which were translated into sets of dummy variables, and `N/C` being variables which I handled in both numerical and categorical forms.

Variable	Type	comment
month	C	Time of initial interview. EG 2009
year	C	Time of initial interview. EG 02 for February
empstat	C	Employment Status. Current time period.
earnweek	N	Dollar amount typically earned weekly at current job. Not adjusted for inflation.
statefip	C	State of Residence.
relate	C	Individual's relationship to head of household.
age	N/C	Person's age, in years.
sex	C	Sex.
race	C	Race. Includes a different category for every combination of identities.
marst	C	Marital status.
famsize	C	Number of family members in an individuals' household, including themself. Non-related roommates are not counted.
nchild	N/C	Number of person's children living with them, including adopted.
nativity	C	Is this person born in the US? What about their parents?
hispan	C	Hispanic origin.
labforce	C	Is this person in the labor force? Yes, no, or NIU (military).
occ1990	C	Occupational classification scheme.
ind1990	C	Industry that they work in or most recently worked in.
classwkr	C	Class of worker. Self-employed, etc. Describes most recent job if they are not currently employed.
uhrsworkt	N	Hours "usually" worked per week at all jobs.
ahrsworkt	N	Hours "actually" worked last week.
wkstat	C	Full or part time status, and why.
educ	N/C	Highest year of school or degree completed.
paidhour	C	Is this person paid hourly?

The following variables are also provided, but were not used as predictors.

Variable	comment
y_tp1	Variable to predict. Binary representation of a person's employment status (not unemployed or unemployed) the year after the other variables were collected.
cpsid	A households's identification Number in the CPS
cpsidp	A person's identification Number in the CPS

5.3.2 Engineered Variables

Variable	comment
racec	Six overlapping dummies for race.
educn	Nested overlapping dummies for educ. EG educ_bchlr=1 means the person has a bachelor's degree or higher.
education_ordinal	Numerical variable roughly equivalent to years of schooling
indg	Dummies for broad industry groups taken from the documentation page for ind1990
occg	Dummies for broad occupation groups taken from the documentation page for occ1990
empcombo	Binary variable representing most imformative combination of empstat categories.
wkcombo	Binary variable representing most imformative combination of wkstat categories.
class_selfemp	Combination of two categories from classwkr.
kids	Nested dummies for nchild.
nchildren_ordinal	nchild converted to integers.
agebin	Dummies for binned ranges of ages.
ageint	age variable converted to integers.
byage_unrate	Numerical variable representing smoothed mapping from age to average year+1 unemployment rate.
earnhour_est	Estimated hourly earnings from earnweek and hours worked. Very noisy.
logearnweek	Natural log of earnweek.
period_unrate	Numerical variable representing smoothed mapping from (year,month) to average year+1 unemployment rate.

5.4 Comparison to Baseline Logit with Threshold 0.5

The competition rules described a simple classifier that qualifying submissions would have to surpass:

> Deliver a GF measure for the test sample that is at least 0.03 (3 percentage points) higher than a classifier based on a logit regression that includes log(earnweek), age dummies, and all the remaining allowable predictors . . . without transformation or interaction terms. The classifier assigns $\hat{y}_{it+1}$ when the predicted probability is greater than 1/2.

I fit a model to the training data as described above and this is the resulting confusion matrix when making predictions about the testing set:

Confusion Matrix for Baseline Algorithm's Predictions

True Status	predicted NU	predicted U	Total
Not Unemployed	78824	182	79006
(portion of row)	(0.9977)	(0.0023)	(1.0)
Unemployed	4237	163	4400
(portion of row)	(0.9630)	(0.0370)	(1.0)

This algorithm only deviates from a prediction of "Not Unemployed" in a rare few cases. As a result, it correctly predicts the status of 99.77 percent of those who won't be unemployed, at the cost of only correctly predicting the status of 3.7 percent of those will be unemployed. This gives the baseline classifier described above a score of $GF \approx 51.74\%$

As a reminder, my algorithm achieve a score of $GF \approx 75.44\%$.

But also recall from section 2 that even very simple 1 variable models can achieve scores in excess of 72%. Measured in terms of absolute improvement to the GF score, most of my algorithm's gain over the baseline logistic regression comes from addressing the issue of imbalanced classes.

5.5 Decision Tree, Partially Visualized

The following is a visualization of the decision tree fit to the training data. Wherever it says `FURTHER DECISIONS TO BE MADE`, that indicates that a subtree has been removed to condense the visualization.

```
├ empcombo = 0
│ ├ classwkr_Private, for profit = 0
│ │ ├ wkcombo = 0
│ │ │ ├ marst_Married, spouse present = 0
│ │ │ │ ├ educn_assoct = 0
│ │ │ │ │ └ PREDICT NON-UNEMPLOYMENT
│ │ │ │ └ educn_assoct = 1
│ │ │ │   └ FURTHER DECISIONS TO BE MADE
│ │ │ └ marst_Married, spouse present = 1
│ │ │   ├ wkstat_Full-time hours (35+), usually full-time = 0
│ │ │   │ └ PREDICT NON-UNEMPLOYMENT
│ │ │   └ wkstat_Full-time hours (35+), usually full-time = 1
│ │ │     └ FURTHER DECISIONS TO BE MADE
│ │ └ wkcombo = 1
│ │   └ PREDICT NON-UNEMPLOYMENT
│ └ classwkr_Private, for profit = 1
│   ├ wkcombo = 0
│   │ ├ period_unrate <= 6.06
│   │ │ ├ marst_Married, spouse present = 0
│   │ │ │ ├ educn_bachlr = 0
│   │ │ │ │ └ FURTHER DECISIONS TO BE MADE
│   │ │ │ └ educn_bachlr = 1
│   │ │ │   └ PREDICT NON-UNEMPLOYMENT
│   │ │ └ marst_Married, spouse present = 1
│   │ │   ├ educn_assoct = 0
│   │ │   │ ├ indg_CONSTRUCTION = 0
│   │ │   │ │ └ FURTHER DECISIONS TO BE MADE
│   │ │   │ └ indg_CONSTRUCTION = 1
│   │ │   │   └ PREDICT NON-UNEMPLOYMENT
│   │ │   └ educn_assoct = 1
│   │ │     ├ indg_PROFESSIONAL AND RELATED SERVICES = 0
│   │ │     │ └ PREDICT NON-UNEMPLOYMENT
│   │ │     └ indg_PROFESSIONAL AND RELATED SERVICES = 1
│   │ │       └ FURTHER DECISIONS TO BE MADE
│   │ └ period_unrate > 6.06
│   │   ├ indg_CONSTRUCTION = 0
│   │   │ ├ marst_Married, spouse present = 0
│   │   │ │ ├ year_2008 = 0
│   │   │ │ │ └ PREDICT NON-UNEMPLOYMENT
│   │   │ │ └ year_2008 = 1
│   │   │ │   └ PREDICT UNEMPLOYMENT
│   │   │ └ marst_Married, spouse present = 1
│   │   │   └ FURTHER DECISIONS TO BE MADE
│   │   └ indg_CONSTRUCTION = 1
│   │     └ PREDICT UNEMPLOYMENT
│   └ wkcombo = 1
│     └ PREDICT UNEMPLOYMENT
└ empcombo = 1: PREDICT UNEMPLOYMENT
```

5.6 Data PreProcessing, In Detail

5.6.1 Adding New Features

5.6.1.1 Combinations of empstat and wkstat **empcombo** is a dummy variable which is set to 1 whenever **empstat** is one of:
`['Unemployed, experienced worker', 'Unemployed, new worker', 'NILF, unable to work', 'NILF, other', 'NILF, retired']`
and set to 0 when **empstat** is one of:
`['Armed Forces', 'At work', 'Has job, not at work last week']`

I originally conceived of this feature as a simple heuristic: People with jobs are very likely to still have jobs next year, probably much more likely than those without jobs. And it turns out that simply predicting unemployment iff **empstat** is neither `'Has job, not at work last week'` nor `'At work'`, yields a score of $GF \approx 0.726$ on the training set. That's pretty good! In fact, it's taken immense amounts of work to improve upon this simple rule-of-thumb.

After observing the performance of this heuristic, I performed a search of category combinations, evaluating each combined dummy variable as as a classifier on its lonesome. This led me to discover the **empcombo** described above. Simply using **empcombo** as the predictions yields a score of $GF \approx 0.727$ on the training set.

A similar analysis of **wkstat** category-combinations yielded the **wkcombo** variable, which by itself scores $GF \approx 0.733$ on the training set. **wkcombo** is set to 1 whenever **wkstat** is one of: `['Part-time hours, usually part-time for economic reasons', 'Part-time for economic reasons, usually full-time', 'Unemployed, seeking full-time work', 'NIU, blank, or not in labor force', 'Unemployed, seeking part-time work', 'Full-time hours, usually part-time for economic reasons', 'Not at work, usually part-time',]` And set to 0 otherwise.[13][14] While **wkcombo** has slightly higher informativeness than **empcombo**, it's more difficult to qualitatively describe. One way to editorialize **wkcombo** might be "economically vulnerable full or part-time status". A value of 1 indicates that that the person lacks a job, or is working unusual hours because of economic reasons.[15]

5.6.1.2 Smoothed time-series of unemployment rate. I combined the categorical features **year** and **month** to yield a combined **yearmonth** categorical feature, indicating the time period in which the survey took place.

I then calculated a time-series of unemployment rate from the data, finding the percentage of people for whom **y_tp1** = 1, conditional on the period, for each period. This time series was then smoothed out using a rolling average (see code for details), and the smoothed time series was used to map the categorical **yearmonth** to the numerical **period_unrate** feature.

A similar feature is constructed by plotting unemployment rate as a function of age, and then smoothing that, with the caveat that the rate for age 64 is set equal to that for age 63, because of a sparsity of data points.

5.6.1.3 Occupation and Industry Groups The IPUMS documentation for both **occ1990** and **ind1990** show how the categories for these features can be arranged into nested groups.

[13]**wkcombo** is set to 0 for any of: `Full-time hours (35+), usually full-time`, `Part-time for non-economic reasons, usually full-time`, `Not at work, usually full-time`, `Full-time hours, usually part-time for non-economic reasons`, or `Part-time hours, usually part-time for non-economic reasons`.

[14]There are several **wkstat** categories for which their inclusion or non-inclusion in **wkcombo** has negligible effect on the combo's informativeness. Namely, `'Full-time hours, usually part-time for economic reasons'`, `'Not at work, usually part-time'`, and `'Full-time hours, usually part-time for non-economic reasons'`

[15]However, "economic/non-economic" reasoning isn't given for employed people who are not at work, which makes this an incomplete description.

I parsed the html for these documentation pages, associated each industry and occupation category with the groups it is in, and then created new **occg__** and **indg__** dummy variables for each group.

There are hundreds of industry and occupation categories, many of which have prevalence which is so low that they aren't very useful as predictors. This seemed like a sensible way of agglomerating the categories. But even then, many of these groups still have low prevalence.

On its own the most informative of these new variables was `occg_MANAGERIAL AND PROFESSIONAL SPECIALTY OCCUPATIONS`, which predicts **y__tp1**=0. However, this feature was dropped out when I performed Lasso Regression, which hints to me that the reason this occupation group is informative by itself is because it is correlated with some other relevant factor, like education.

5.6.1.4 Nested Education dummies

The **educ** feature records a person's *highest* educational achievement.

This means that if we make a dummy variable for **educ** = `Bachelor's degree`, the coefficient for that dummy represents a comparison between (people with a bachelor's degree but no higher degree) and (people who either lack a bachelor's degree or have a graduate degree). That seems less sensible than comparing (people with a bachelors degree or higher) to (people without a bachelors degree).

I converted the **educ** feature to an ordinal variable, and then used that to construct a set of nested **educn__** dummies. Anyone with a bachelor's, master's, professional degree, or doctorate has **educn__bachlr** coded as 1.

5.6.1.5 Combined Race Categories

The **race** feature has many categories for combinations of different races. None of these combinations are very prevalent, so I simply amended the single-race dummies to include people who are mixed of that race, and include another dummy indicating whether someone identifies with multiple races.

For example, I coded a person with **race** = `White-Black` as **racec__white**, **racec__black**, and **racec__mixed**,

5.6.1.6 Several other created variables

Not all of these are used in every model. Many of these drop out with feature selection.

- **agebin** was created by binning **age** in increments of 5 years.
- Similar to the education categories, I converted **nchild** to an ordinal variable and then created nested categories. This wasn't as useful as the transformation on education, and I didn't bother to do the same with **famsize**.
- I tried to estimate hourly earnings in a feature I labelled **earnhour__est**. (These estimates are very noisy.) See the next subsection for details.

5.6.2 Cleaning Numeric Features

The following is a summary of what I did when processing the numerical features like earnings and hours.

For **ahrsworkt**:

1. Change any instance of `999` to `0`. This IPUMS variable uses 999 as a code for people "Not In Universe", which here is simply people who did not work last week.
2. Cap hours worked at 112. That's 16 hours per day. I think the person saying they have a 198 work week is confused or the data has been mis-entered.

For **uhrsworkt**:

1. Change any instance of `NIU` to `0`; the NIU set is people who lack a job (NILF, Unemployed, or Armed Forces). They therefore cannot have typical hours.
2. Change any instance of `Hours vary` to `np.nan`. They shouldn't just be set to zero; these people *are* working.
3. Impute these missing values to the median.
4. Cap hours worked at 112. That's 16 hours per day.

For **earnweek**:

1. Remove NIU values by changing any instance of `9999.99` to `np.nan`
2. Impute zero earnings for any group that couldn't possibly have earnings. This means an **empstat** which is neither `'Has job, not at work last week'` nor `'At work'`.[16]
    - Any remaining missing values represent people with jobs, but with unknown earnings data.
    - These people are mostly self-employed, and comprise about 10% of the training data set.

I created an estimated hourly earnings variable, called **earnhour_est**:

1. Create a crude estimate
    - For those with 0 earnings, earn hour is also zero.
    - For those with positive earnings and positive usual hrs worked, divide one by the other.
    - For anyone else, leave it as missing data.
        - These will be people with unknown earnings, people for whom 'Hours vary', and people who somehow earn money and have a job despite usually working 0 hours.
2. Deal with outliers. Cap hourly earnings to 200 dollars. Only a few of these very high earners have entries which look plausible.
3. Impute missing values to median.

I also include the natural logarithm of earnings, labelled as **logearnweek**. log(1 dollar) is 0. log(0.01) approx -5, and log(0)=-inf. But there really isn't much conceptual difference between making nothing and making 1 dollar a week, so I rounded all of those up to zero.

5.6.3 Scaling, Dummies, and Variance Thresholds

Numeric variables were scaled using sklearn's StandardScaler to unit variance without centering the data at zero.

Categorical variables, with some exceptions described above, were one-hot encoded to create dummy variables.

Then a variance threshold of 0.001 was applied to the features using sklearn's VarianceThreshold filter. When applied to the entire training set, this removes categories with a prevalence of less than approximately 300 people. Feature selection and regularization would nullify those features anyways, so this step just saves computation time.

Finally, there were a few categorical dummies which were excluded because they contained exactly the same information as another dummy. For example, `empstat_Armed Forces`,`labforce_NIU`, and `classwkr_Armed forces` have exactly the same vector of 1s in the training data, and so only the first of these was retained.

[16] A passing thought after the competition: I imputed 0 weekly earnings for anyone with **empstat**=`Armed Forces` but now am wondering if it would have more sensible to impute median earnings for this category. I expect the effect on final would be negligible.

5.7 Other Programming Notes

5.7.1 Software Packages Used

I used python version 3.9.6 on a Windows 10 machine for this competition.

In addition to python's built-in modules, I used several open-source python packages.

Module	Version	Sourcecode link	Used for...
scikit-learn	1.0.1	sklearn source	Data manipulation and machine learning algorithms.
pandas	1.3.4	pandas source	Data manipulation, analysis
numpy	1.21.4	numpy source	optimized mathematical operations
matplotlib	3.5.0	matplotlib source	drawing graphs

The exact version numbers I don't think should be too important. I tried not to use any functions that risk becoming deprecated.

However, for future projects, I recommend using scikit-learn version 1.1 or higher. This update makes several improvements to the consistency of of syntax across objects.

5.8 Comparison to Additional Models

5.8.1 Gradient Boosting

Gradient Boosted Decision Trees (GBDT) are an ensemble method wherein a sequence of small decision trees are trained on the data, with each successive tree trying to explain the residuals unexplained by the previous one (Friedman 2001).

After conducting a search of of hyperparameters, and using early stopping techniques to determine how many trees to use, I fit a GBDT model using a sequence of 140 trees of depth 2.

As expected, the resulting ensemble outperformed the single decision tree which I used in the competition, scoring $GF = 75.48\%$ on the testing set. However, the algorithm's accuracy in predicting the employment status of the to-be unemployed was still poorer than that of the regularized linear regression models.

The simple majority-vote ensemble has a similar GF score when the singular decision tree is replaced with the GBDT. But there are noticeable improvements for predicting the status of the to-be unemployed.

The GF score is improved when the ensemble is slightly adjusted. The GBDT, LASSO, and Ridge models all can output estimated probabilities instead of binary predictions. In the simple ensemble labelled "Average" in the table on the following page, the probabilities from each model are averaged, and a prediction of 1 is given if this average exceeds the threshold of 0.05144.

5.8.2 Neural Networks

During the competition, I trained a simple neural network on the data[17], but it performed worse than the decision tree. I anticipated that this could be rectified by tweaking the hyperparameters of the model, but doing so would be time consuming.

After the conclusion of the competition, I tested many different neural network structures and hyparameters.

In the table on the following page, I present the scores from a few networks that performed well on the validation set.

- The "Deep Neural Net" has 6 hidden layers of 100 neurons. I regularized the model using early stopping.
- The "Shallow Neural Net" has only 1 hidden layer of 500 neurons, and uses dropout for regularization.[18]
- Dropout also enables the use of a technique called Monte Carlo Dropout ("Shallow Neural Net MCD"), wherein a large set of predictions is made by the same model and then averaged, with some portion of the neurons 'ignored' when making each prediction (Gal and Ghahramani 2016). However, when I used this method, the improvements to GF score were small, while the ability to correctly predict the status of the too-be-unemployed in particular fell dramatically.

In each case, I started training with a low learning rate, gradually increased the learning rate until this no longer helped, and then gradually decreased the learning rate for the remained of testing. The technique of decreasing the learning rate near the end of training is called "simulated annealing", and the process of first increasing then decreasing the learning rate is called the "1cycle" technique (Smith 2018).

[17]I used the Keras python interface for the Tensorflow neural network library.

[18]In particular, it uses a variant called "alpha dropout". which is designed to work with SELU activation functions to self-normalize the network; both techniques are described in (Klambauer et al. 2017).

5.8.3 Final Summary of Results, Including Additional Models

The following table is a summary of the scores of the models I trained, both those trained before and after the competition. GFNU is the accuracy in predicting the employment status of those who were Not Unemployed during their followup interview; GFU is the accuracy in predicting the status of those Unemployed during the followup, and final metric GF, also called *balanced accuracy* is the average of the two.

	Mean CV Scores			Scores on Testing Set		
Model	GFNU	GFU	GF	GFNU	GFU	GF
Decision Tree	0.8180	0.6703	0.7442	0.8364	0.6561	0.7463
LASSO	0.7909	0.7145	0.7527	0.7953	0.7059	0.7506
Ridge	0.7781	0.7247	0.7514	0.7834	0.7239	0.7536
Majority Vote	0.7927	0.7146	0.7537	0.7992	0.7095	0.7544
Gradient Boost	0.8180	0.6915	0.7548	0.8205	0.6891	0.7548
Vote(Boost,LASSO,Ridge)	0.7917	0.7168	0.7542	0.7968	0.7118	0.7543
Average(Boost,LASSO,Ridge)	0.8031	0.7102	0.7567	0.8069	0.7070	0.7570
Deep Neural Net	0.8274	0.6784	0.7529	0.8197	0.6889	0.7543
Shallow Neural Net	0.7789	0.7199	0.7494	0.7729	0.7273	0.7501
Shallow Neural Net MCD				0.8698	0.6359	0.7529