Predicting People's Actions Using American Time Use Data

Bowen Bao

Executive Summary

The American Time Use Survey is a dataset monitoring time use in the US. The data measures how people of different demographics spent time throughout the day. ATUS data is very popular among economists and is historically used to determine labor inequality among people of different socioeconomic statuses and gender. However, very little research using American Time Use Data has been done on a predictive scale. Using the following models, I used 624 feature variables to predict the 18 activity categories.

1. Bernoulli
2. Gaussian
3. Nearest Centroid (Cross Validated)
4. Logistic Regression
5. Ridge Regression (Cross Validated)(Scaled)
6. Decision Tree (Cross Validated)
7. KNN (Cross Validated)
8. Neural Network (Cross Validated)(Scaled)
9. SVC

Decision Tree provided the best predictive power with a 0.60 accuracy score. The variables that were the most influential in prediction are time of day, location, and age of household members. Logistic Regression would have been a second-best model with a 0.40 accuracy score. However, this model was too computationally expensive for large datasets. Future work extending this project would be to expand the database to include the 15-year dataset extending from 2013 – 2018 and to do a separate analysis using the 4-code and 6-code activity lexicons. Although given the subtle differences in the higher specificity categories, it would be harder for the model to predict accurately.

Introduction

The American Time Use Survey is a dataset published by the Department of Labor Statistics and conducted by the US Census Bureau monitoring time use in the US. The data measures how people of different demographics spent time throughout the day. Individuals are randomly selected from a subset of households that have completed their interviews for the Current Population Survey (CPS). Respondents are interviewed one time about how they spent their time on the previous day, where they were, and whom they were with.

There are 8 datasets total and four that were used are:

1. Respondent file: information about ATUS respondents and their labor force and earnings.
2. Roster file: Information about household members and their children (under 18) as well as information such as age and sex.
3. Activity file: information about how ATUS respondents spent their diary day. This includes activity codes, activity start and stop times, and locations

4. Who file: codes that indicate who was present during each activity

The other 4 files were not used because they weren't relevant and if merged, the observations would have later been dropped because the entire row would have missing variables.

Very little research using American Time Use Data has been done on a predictive scale. ATUS data is very popular among economists and is historically used to determine labor inequality among people of different socioeconomic statuses and gender. This includes questions like whether men or women do more unpaid labor, housework, or child care to determine whether government intervention is necessary. This project is different from previous work in that it uses machine learning algorithms to make predictions. Given a person's demographics, location, time of day, and who they're with, can we predict what they're doing right now?

The data contains information about their demographics as well as categories about their actions coded by a certain lexicon.

ATUS 2018 Lexicon			
Major Categories	2nd-tier	3rd-tier	
02 Household Activities			
	01 Housework		
		01	Interior cleaning
		02	Laundry
		03	Sewing, repairing, & maintaining textiles
		04	Storing interior hh items, inc. food
		99	Housework, n.e.c.*
	02 Food & Drink Prep., Presentation, & Clean-up		
		01	Food and drink preparation
		02	Food presentation
		03	Kitchen and food clean-up
		99	Food & drink prep, presentation, & clean-up, n.e.c.*

[1]

Related Work

There is not much literature or previous work that has utilized this dataset in a predictive matter. Only one presentation has had related work. In an AAPOR 70th Annual Conference in 2015, Hariharan Arunachalam, Gregory Atkin, et al from University of Nebraska-Lincoln Department of Computer Science and Engineering presented their findings for I Know What You Did Next: Predicting Respondent's Next Activity Using Machine Learning[2]. Arunachalam and his team uses data from 2010 – 2013 using Tier 3, the most detailed encoding lexicon. Their method asks given an activity, what comes next and finds the attributes that affect the prediction. They use Markov Chain Models and Artificial Neural Networks to train and build a classifier to predict 5 possible next activities after an activity. The artificial neural network is better at finding unknown and hidden relationships. Using the Markov Chain Model, he predicts the top 5 possible next activities using 1) the entire dataset and 2) a sub dataset with similar demographics. The intuition for the latter is because daily activities and routines may be similar across people of

[1] https://www.bls.gov/tus/lexiconnoex2018.pdf
[2] http://www.aapor.org/AAPOR_Main/media/AnnualMeetingProceedings/2015/E2-2-Arunachalam.pdf

similar demographics. He trains the data on one year and tests it on the next year (i.e., train it on 2012 data and tests it on 2013 data). The results given is that the demographic based models did not consistently perform better than the non-demographic model. Using Artificial Neural Networks, the general conclusion is that while the common activity sequences (more generic lexicon codes ie "Traveling") are predicted correctly, the more unique activity sequences ("Travel related to personal care") are predicted inaccurately and that's because the unique activity sequences are relatively harder for the algorithms to learn while the common activity sequences do not occur enough to compensate. The ANN model currently only predicts one possible next activity. While the accuracy is low for a single classifier, the context shows that it will improve with 5 predictions. In the appendix, Arunachalam also shows his results using PCA. The data is the same as used in the Markov Chain Model where it utilizes first activity name, hour and minute of the end time of the first activity and the demographics and he lists the 14 most influential variables generated by PCA. The details of his methods are a little fuzzy since the paper is unavailable and only presentation slides are found.

Solution and Work

Since this problem is a multi-class classification problem, I applied models that extend off binary classifiers that have multiclass properties. Those include the following:

1. **Bernoulli**
2. **Gaussian**
3. **Nearest Centroid**
4. **Logistic Regression**
5. **Ridge Regression**
6. **Decision Tree**
7. **KNN**
8. **Neural Network**
9. **SVC**

I split the data into 70-20-10 as training-validation-test.

1. **Training**: I split the training data into 70-30 and take the 30 to get the test MSE. I tune the parameters within each function definition and apply the optimal parameters to get the model.
2. **Validation**: After getting the best parameters of each model type, I test the models on the validation data to see what the best model is.
3. **Test**: I retrain the optimal model on the Training + Validation data and I test it on the Test data.

Bernoulli

- Event occurrences are independent and it signifies success and failure.

Gaussian

- Returns the probability for each class

Nearest Centroid

- Assigns observations the label of the class whose mean is closest to the observation

- The parameter 'shrinkage', threshold for shrinking centroids to remove features is tuned.
- This is similar to k-means clustering where the data is partitioned into k distinct non-overlapping clusters
- In this case, we have 18 clusters, one for each activity category
- This method records the distance between the centroid of cluster A and the centroid of cluster B which can result in undesirable inversions

Logistic Regression

- Splits all the classes and performs log regression for the set in class 1/ not in class 1
- Repeats this for in class 2/ not in class 2 and so on
- This method takes the longest running time, substantially longer than the other functions for this reason.

Ridge Regression

- X is standardized so that each parameter has mean = 0 and a standard deviation = 1 so that effect would not be dominated by the most widely varying parameter.
- Alpha, the tradeoff between RSS and the shrinkage penalty, is tuned
- As alpha increases, flexibility decreases causing increased bias and decreased variance.
- Because this data has 624 x-variables which makes it a high dimensional problem, regularization or shrinkage plays a key role.

Neural Network

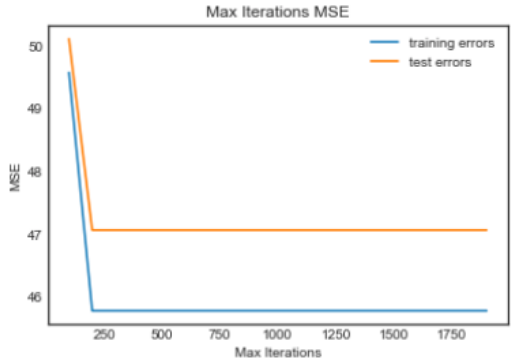
- Has multiclass perceptions which has n binary neurons in the output layer. The last layer of the neural network is a SoftMax function layer.
- X is also standardized so that each parameter has mean = 0 and a standard deviation = 1 for the same reason as above.
- When tuning the parameters for neural network, I wanted to test activation, max iteration, learning rate, alpha, and hidden layer. However, the computation time to run cross validation took too long so I tuned it manually. The results are below.

```
def Neural(X_train, Y_train, X_test, Y_test):
    # Y is dummy variable

    #Cross validation
    tuned_parameters = {'activation': ['identity', 'logistic', 'tanh', 'relu'],
    'max_iter': np.arange(1000, 10000, 10),
    'learning_rate_init': np.linspace(0.1, 1, 10),
    'alpha': 10.0 ** -np.arange(1, 10),
    'hidden_layer_sizes':np.arange(10, 50, 5),}
    cv = GridSearchCV(MLPClassifier(), tuned_parameters)
    cv.fit(X_train, Y_train)

    #Optimal parameters
    print('Best Params: ')
    print(cv.best_params_)
```

Original Tuned Parameters using Grid CV



Max iteration: Since max iteration didn't affect MSE past 250, I kept optimal max iteration at 1000 which is the default.

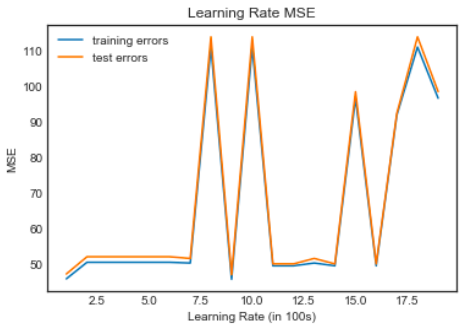


Figure 1

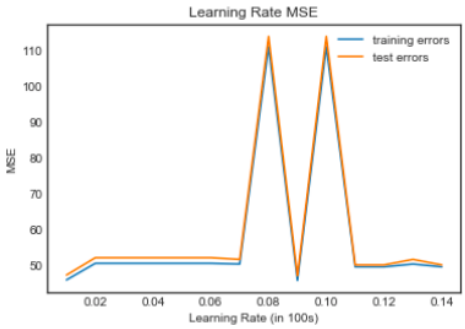


Figure 2: Close up version of Fig.1

Learning Rate: Optimal learning rate is at 0.09.

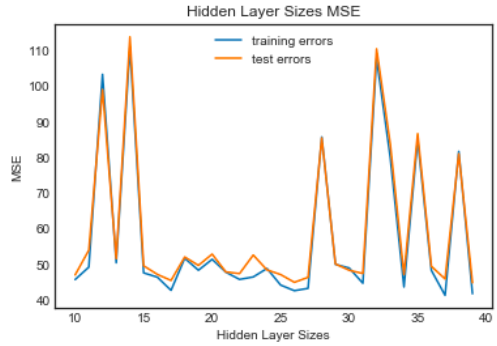


Figure 1

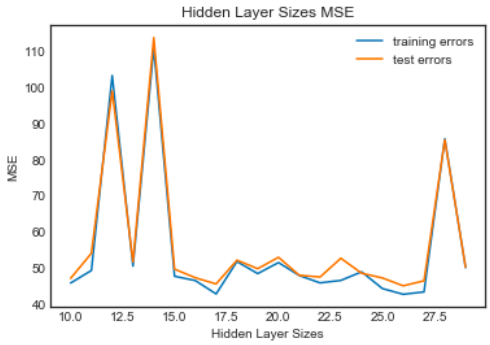
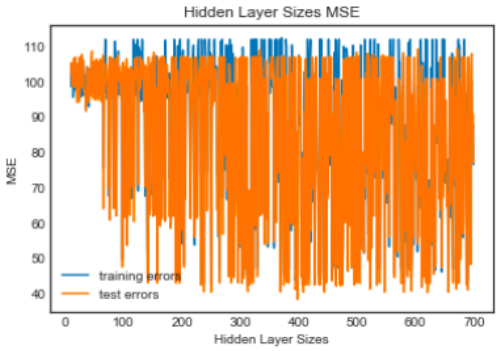
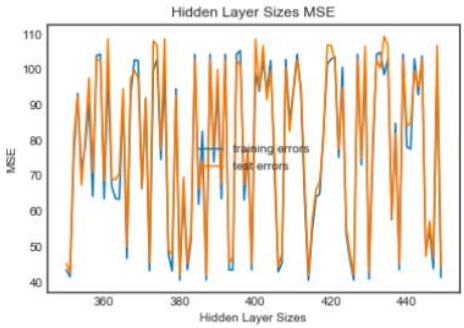
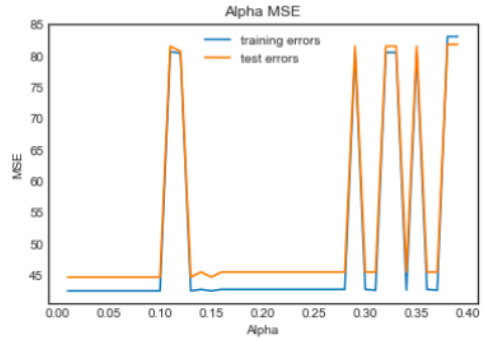


Figure 2: Close up version of Fig.1





Hidden Layer: Since it's better to have too many hidden units than too few, I took 380 as an optimal hidden layer size. One reason is because there are 624 features. I initially wanted to look at hidden layer sizes that are greater than 624. However, it didn't seem to make a difference and the lowest MSE occurs around 400.



Alpha: Since there is no effect of alpha = 0 to 0.1, I take the optimal alpha to be 0.05.

I use the model where our final parameters are: activation = logistic, max iteration = 1000, learning rate = 0.09, hidden layer = 17, alpha = 0.05.

Decision Tree

- Splits the training data based on the values of available features to produce generalization
- More closely mirror human decision making
- The 'number of features' is tuned from a range of 1 to 624 to account for all the available features.
- Each observation in a region is assigned to the most commonly occurring class in the region.
- Although in theory the predictive accuracy is not as good as other methods, in this case it was the best method by far in terms of predictive accuracy.

KNN

- The distance from a point is measured to every other training point and k smallest distances are identified
- The parameter 'number of neighbors' is tuned
- The performance for KNN is the worst when K=1 and will improve as K increases. In higher dimensions KNN is worst because the data points are spread out farther apart and most observations have no nearby neighbors.

SVC

- Maximizes the margin from separating hyperplanes
- SVC did not make it into the final analysis because the multi class option of the function uses 'crammer_singer' which is documented as "seldom used in practice as it rarely leads to better accuracy and is more expensive to compute"3. See below. Because of this, SVC was not used.

multi_class*{'ovr', 'crammer_singer'}, default='ovr'*

Determines the multi-class strategy if y contains more than two classes. "ovr" trains n_classes one-vs-rest classifiers, while "crammer_singer" optimizes a joint objective over all classes. While crammer_singer is interesting from a theoretical perspective as it is consistent, it is seldom used in practice as it rarely leads to better accuracy and is more expensive to compute. If "crammer_singer" is chosen, the options loss, penalty and dual will be ignored.

Results and Analysis

Because the computation time for Logistic Regression was too long, one variation of the analysis was run on 1000 observations. The results from the training set are below.

Model	Training MSE	Training Accuracy	Validation MSE	Validation Accuracy	Test MSE	Test Accuracy	Optimal Parameters
Bernoulli	68.01	0.27	70.91	0.26			
Gaussian	60.32	0.08	96.02	0.07			
Centroid	69.76	0.33	86.45	0.30			shrink_threshold': 7.5757
Ridge	65.40	0.04	98.33	0.05			
Decision Tree	46.47	0.52	62.22	0.49	42.08	0.50	alpha: 854.773
KNN	97.79	0.25	117.19	0.33			{'max_features': 1.0}
Neural Network	105.79	0.22	108.08	0.27			{'n_neighbors': 38}
Logistic	48.25	0.45	64.73	0.40			

Top 10 Most Important Features:

TEWHERE_12	21.063873	Car, truck, or motorcycle (driver)
TUACTDUR	7.401547	"Duration of activity in minutes (last activity not truncated at 4:00 a.m.)"
TUACTIVITY_N	6.142742	Activity line number"
TEWHERE_1	5.359539	Respondent's home or yard
TRTALONE	3.138173	Total nonwork-related time respondent spent alone (in minutes)
TEAGE	2.513610	Household member age
TRTALONE_WK	2.329074	Total work- and nonwork-related time respondent spent alone (in minutes)
TUACTDUR24	1.891785	Duration of activity in minutes (last activity truncated at 4:00 a.m.)
TEWHERE_14	1.602785	Walking
TUMONTH	1.412505	Month of diary day (month of day about which ATUS respondent was interviewed)"

Logistic Regression offered the second-best accuracy followed by Decision Tree.

3 https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html#sklearn.svm.LinearSVC

Not using Logistic Regression, I ran the analysis again on 20,000 observations. The entire dataset is >70,000 observations. Due to computational limitations and waiting time, the data was truncated.

Model	Training MSE	Training Accuracy	Validation MSE	Validation Accuracy	Test MSE	Test Accuracy	Optimal Parameters
Bernoulli	100.87	0.14	96.17	0.16			
Gaussian	83.67	0.03	78.36	0.02			
Centroid	70.82	0.18	70.36	0.18			{'shrink_threshold': 8.5858}
Ridge	73.63	0.10	75.24	0.10			alpha: 932.603
Decision Tree	62.46	0.58	60.28	0.57	53.65	0.60	{'max_features': 1.0}
KNN	97.88	0.34	95.84	0.34			{'n_neighbors': 27}
Neural Network	53.87	0.60	105.90	0.24			

Top 10 Most Important Features:		
TEWHERE_12	20.926444	Car, truck, or motorcycle (driver)
TUACTDUR	7.421104	"Duration of activity in minutes (last activity not truncated at 4:00 a.m.)"
TUACTIVITY_N	6.822602	Activity line number"
TEWHERE_1	4.502960	Respondent's home or yard
TUACTDUR24	3.350814	Duration of activity in minutes (last activity truncated at 4:00 a.m.)
TRTALONE	3.090374	Total nonwork-related time respondent spent alone (in minutes)
TRTALONE_WK	2.755452	Total work- and nonwork-related time respondent spent alone (in minutes)
TEAGE	2.202393	Household member age
TEWHERE_14	1.790457	Walking
TEWHERE_2	1.271366	Respondent's workplace

From above, we see that Decision Tree was the best predictor by far with the highest accuracy. Time of day, activity number, location, and amount of alone time, and household member age constituted the most important factors in determining the activity. This makes sense since time of day and location would most likely impact sleep or work while children's age would impact housework. The analysis conducted on the smaller dataset vs the larger didn't differ too much in terms of results.

What worked and didn't

Overall the predictive power was poorer than what I had hoped. Ideally, I had hoped the predictive power could be 80% accuracy. Training on a larger database did not significantly improve prediction.

Decision Tree may have been the best because it most closely resembles human decision making, which is the nature of choosing which activities to do.

One reason KNN might have performed poorly is because since this is a high dimensional problem, data points are spread out farther apart and most observations have no nearby neighbors.

For centroid, the optimal 'shrinkage' parameter is around 8 which can result in undesirable inversions. Since there are 18 categories, ideally there should be 18 centroids. However, this is not the case. The predictive power in this model was 18% accuracy.

For ridge, the alpha is around 900, which means there's a high tradeoff between RSS and the shrinkage penalty. The flexibility decreased causing increased bias. Since this was a high dimensional problem, this method pushed a lot of coefficients down to 0. Multicollinearity could

have also played a problem because since the nature of some of the questions can be repetitive, therefore some predictors can be a linear combination of the others. The predictive power is 10% accuracy.

For neural network, I suspect there might have been some overfitting. The predictive power on the training data was 60% accuracy whereas on validation data, it was 24%. This wasn't the case on the smaller 1000 observation dataset. The training accuracy there was 22% and the validation accuracy is 27%. Ideally, I wanted to address overfitting with early stopping iteration. However, I also needed to balance it with more hidden layers because of the high number of inputs.

Future Work

Due to computational limitations, the scale of the project was significantly cut down because of the inability to run a database of this scale. The original intent was to run the analysis on data over 15 years, between 2003 – 2018. However, that would have constituted ~1.3M observations and the computer was unable to merge the datasets, much less analyze. I used 2018 data, however, that also had issues to run analysis. I ended up using 20,000 observations in the 2018 data. So, the first suggested work would be to run the analysis on all of the 2018 data, then over the 15-year data.

The second part of the project that was cut was due to the tiers. There are three tiers of lexicons in this dataset: a two-code, a four-code, and a six-code. The level of specificity increases with higher codes. There are 18 two-code categories, 103 four-code categories, and 402 six-code categories. However, due to the computation power, creating predictions for the four-code and six-code lexicons would have been too time intensive. Therefore, the next part would be to include those into the analysis.

The third would be drop certain variables. When cleaning the data, we dropped observations that were missing variables. However, when going about merging the data, merging the Roster database would have created missing variables for people who were *with* the main participant at the activity but are not a part of the household. Merging the Respondent database would have created missing variables for people who were in the same household but not the main participant. Therefore, a third option would be to not merge those datasets because it inherently drops some activity observations.

Effort and Skills

Most of the time spent was trying to figure out what models to use and figuring out how to deal with computational issues. For computational issues, a lot of time was spent on waiting for program to run and realizing it won't compute and having to modify the code. Other times, it was figuring out if other computational resources are viable. This includes using Linux, RCC, vLab, Github. Although these are options, it wasn't feasible to copy the dataset over because it either exceeded max file upload or gave errors.

Figuring out what models to use also took up time. This is in part because this problem is a multi-class question and we haven't gone over these in class. The initial idea was to create a dummy variable for each categorical lexicon. For example, if a lexicon had 18 categories, then

we would have 18 dummy variables and run 18 functions for each dummy, get a vector of length 18, apply the SoftMax function to get a probability of each category which adds up to 1, and take the category of the one with the highest probability. I started coding this, however later realized that there are multiclass functions that includes the SoftMax function. The issue with doing this manually was that the test error would have had to be manually calculated so that it takes the minimum over all 18 variables. The multi-class functions do this automatically. Therefore, a third of the way through, I switched to using the multi-class functions. I consulted Scikit's library to see which functions we covered in class were inherently multiclass[4]. I also consulted Wikipedia's multiclass classification page to see which algorithms are commonly used for multiclass predictions[5].Because there was little research done before, I spent most of my time fine tuning my parameters.

Bibliography

"1.12. Multiclass and Multilabel Algorithms¶." *Scikit*, scikit-learn.org/stable/modules/multiclass.html.

Arunachalam, Hariharan. *I Know What You Did Next: Predicting Respondent's Next Activity Using Machine Learning*. University of Nebraska-Lincoln Department of Computer Science and Engineering, 15 May 2015.

"ATUS Data Files." *U.S. Bureau of Labor Statistics*, U.S. Bureau of Labor Statistics, www.bls.gov/tus/data.htm.

"Multiclass Classification." *Wikipedia*, Wikimedia Foundation, 15 May 2020, en.wikipedia.org/wiki/Multiclass_classification.

"Sklearn.svm.LinearSVC¶." *Scikit*, scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html#sklearn.svm.LinearSVC.

GitLab Link
https://mit.cs.uchicago.edu/mpcs53120-spr-20/bbao

[4] https://scikit-learn.org/stable/modules/multiclass.html

[5] https://en.wikipedia.org/wiki/Multiclass_classification