George Slota – DataMaster 5000

IST 557 Final Project

Abstract

As part of a competition hosted by the website Kaggle, a statistical model was developed for prediction of United States Census 2010 mailing return rates. A variety of supervised and unsupervised methods were investigated for both training and clustering of data. Observations of a high spatial dependence of the return rates resulted in success using K nearest neighbors-based predictors and clustering using hard state boundaries. A final submission to the competition using random forest and support vector machine training methods and state-based clustering resulted in a ranking of 27/244 with a model error of 2.98 (weighted average deviance from given to predicted return rates). Further work involved using a generalized boosted regression model trained with a high number of trees. This brought model error down to 2.86. With unlimited computation time, further improvements to this error value are possible.

1.0 Introduction

The purpose of this project was to develop a statistical model to predict the United States Census 2010 mailing return rates at the block level. The Census Bureau hosted this contest on Kaggle, a website which holds data science competitions, and intended to use the results of the competition for planning purposes for upcoming census and demographic surveys. The evaluation of results was based on the average error in prediction (known vs. predicted return rates for a block group) weighted by the population of the block group.

Effort on this project was split into two main parts. The first part consisted of handling and processing the input data. Multiple data sources were used, including both data provided in the competition as well as data from external sources. The data was combined, scaled, and examined for which predictors to include in the final models. In addition, a K nearest neighbors (KNN)-based method was introduced to create new spatial predictors. The second part of the project was training the actual model. A number of different models were tested and evaluated. In addition, a few clustering methods were also examined, including state and regional-based clustering as well as K means-based methods.

2.0 Method

As mentioned, the project methodology was split into two primary parts. Firstly input data was handled and manipulated. Secondly this data was used to train various models which were finally compared for efficacy. When it became apparent that the data and return rate were highly spatially dependent, clustering methods were also investigated.

2.1 Data

There were four primary data sources used during the competition. These sources included the original supplied data, coordinate data offered by YetiMan on the Kaggle forum [1], Census 2000 Return Rates also supplied on the forum [2], and external social data supplied on Angel by the instructor.

2.1.1 Competition Data

Data was supplied with statistics for each block group, the smallest geographical unit looked at by the census. Tract, county, and state are some the larger units, in increasing order. The data provided by the competition was split into training and testing sets, each containing data for about 126 and 85 thousand different block groups, respectively. The testing set was used for submission of predictions to the competition, and included all of the same predictors as the training set, only omitting the return rate. Most of the data was demographic (ages, populations, poverty rates, etc.) or geographic (land area, land type) in nature.

Two sources were used in compiling this given competition data. These included both census (CEN) and American Community Survey (ACS) results. The ACS data included a margin of error estimate for its given values. As a method for including this error estimate into the model was never found, these values were simply removed. An analysis of the CEN and ACS showed vast differences in some of their given values for the same data. As such, the predictive power of each data set individually, together, and averaged (for overlapping measurements) was investigated. No improvement in results was noted for removing the entirety of one set or the other or averaging them. Therefore, most models tested in this effort included both sets of data.

The data was also scaled where it was logically appropriate. In the data dictionary supplied by the contest, the columns for which scaling made sense were noted. The given columns were used as a basis for the scaling performed. An example of where scaling makes sense might be with population proportions, such as the number of a certain age group in the block group. A given number such as ‘100’ for the number of people aged 18-24 might have limited predictive ability relative to other block groups. However, a value scaled by population giving a proportion of ‘0.10’, or 10% being aged 18-24, can be directly compared to other similar values in other block groups.

In order to examine the possible benefits for scaling the data, models were tested using data that was either scaled or unscaled, as well as using both together. The unscaled data performed worst, followed by the scaled data, and then both together resulted in the lowest error. However, the computational costs of using the vastly increased data size did not seem to be worth the slight improvement over just using the scaled data. Most tested models therefore used only the scaled data.

In addition to the ACS margin of error data, a number of other predictors in the supplied dataset were removed. These included both the numerical and text labels for state/county/tract, as

well as columns that had a high proportion of missing data (TEA_Update_Leave_CEN_2010, BILQ_Mailout_count_CEN_2010), or columns that did not supply any data at all (Flag). In the final submission, additional columns were removed and replaced with imputed values from the supplied aggregated dataset, which will be discussed later.

2.1.2 Coordinate Data and K Nearest Neighbors

As mentioned, spatial data for each block group, given as the central coordinates, was compiled and supplied on the Kaggle Forum by YetiMan. This coordinate data included all block groups in both the training and testing sets. A visual inspection of return rates overlaid on a map of the country seemed to indicate a high spatial dependence of return rates of geophysical location, as is demonstrated in Figure 3, put together by Andrew L [3]. As such, a K nearest neighbor approach was attempted for both imputation of missing values as well as creating new predictors.

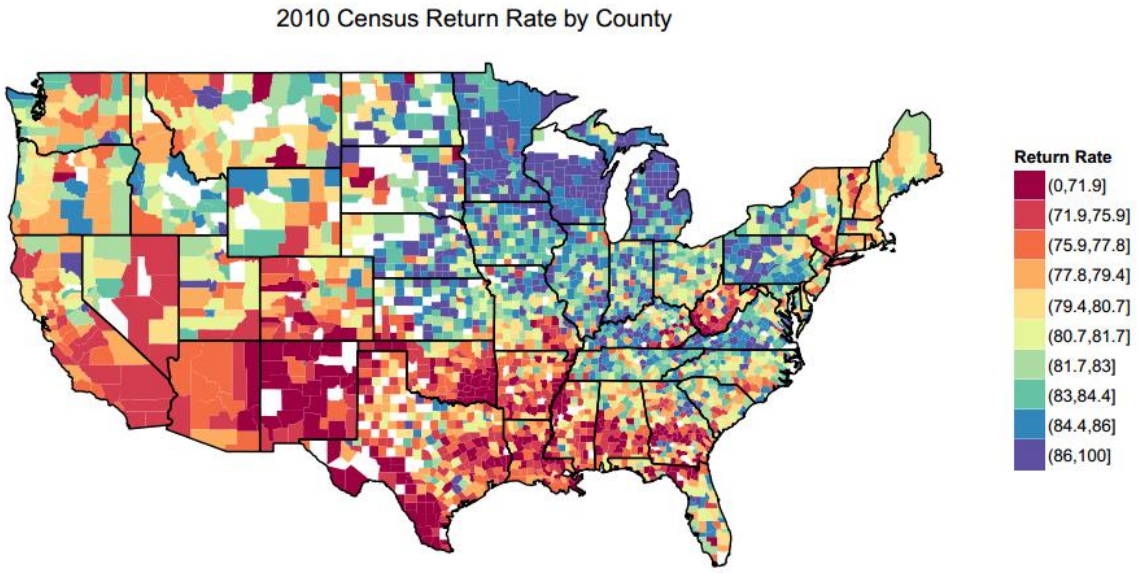

Figure 1 – Census 2010 Return Rate by County. Created by Andrew L.

Code was supplied on Angel by the instructor for creating new predictors based on the median of the K nearest block groups. This code was slightly modified to impute missing values for a number of predictors, which are given in Table 1. Using these imputed values showed a slight benefit over either zeroing out those values or taking the median or mean of the rest of the column. Spatial predictors for return rates were also created based on this approach by taking the median of the nearest K neighbor return rates. This predictor showed significant potential for reducing model error.

Predictors
Med_HHD_Inc_BG_ACS_06_10
Med_HHD_Inc_TR_ACS_06_10
Aggregate_HH_INC_ACS_06_10
Med_House_Val_BG_ACS_06_10
Med_house_val_tr_ACS_06_10
Aggr_House_Value_ACS_06_10
TEA_Mail_Out_Mail_Back_CEN_2010

Table 1 – Predictors Imputed Using KNN Median Method

The Angel code was more heavily modified to create a weighted KNN return rate predictor. The weightings were proportional to the inverse of the distance from the block group the predictor was being calculated for. Varying values of K were analyzed, including K=10,20,50,100,200,500. The value of K which resulted in the best overall prediction ability was K=200. The predictors calculated from this value were used in the Kaggle submission. The addition of this single KNN return value helped the error drop immensely. In fact, a linear model was trained just using the given training data and a single KNN value resulting in an error of about 3.25, comparable at the time to the best score achieved using all predictors and more complex training models.

It is now considered an oversight not to have used predictors for multiple values of K. As such, the final submission used multiple values of K, including K=5,15,50,100,200,500. The KNN code was modified once more in order to decrease computation time to allow for a higher proportion of the allowed time (12 hours) to be used for training the model. To do this, the total search space was substantially decreased by only looking at the K nearest in-state neighbors. Pseudocode for this method is given in Algorithm 1. Some less populous states had less than 500 total block groups. For instances when this occurred, the K value was simply taken as minimum of either the given K values or the number of block groups in the state. Using this state-based partitioning, computation time for all six K values decreased substantially, from over 4 hours to only 45 minutes.

An initial comparison was done for predictive ability between using only in-state neighbors and all neighbors that showed an actual slight improvement using the state-based method. This might be explained by the as-of-yet unexplained large return rate gradient that appears on certain states' borders. Looking at the map of return rates (Figure 1), this is most apparent on all of West Virginia's border, Tennessee's southern border, as well as Oklahoma's southeastern border, among multiple other examples. However, only limited testing was performed when comparing these observed results, and the differences were not substantial enough to make a confident conclusion. As was mentioned, the primary motivation for pursuing this method was to decrease computation time.

Algorithm 1 – Weighted K Nearest In-State Neighbors

- Begin
- For each state:
    - Extract state data from full set
    - For each BG in the state data:
        - Calculate distances for every other BG relative to the current
            - (Only use training BGs for this current train/test split)
        - Sort the distances in increasing order
        - For each K:
            - Extract only first K distances (or min(K, length(distances))
            - Take inverse distances (dist_inv = 1 / distances)
            - Get sum of inverse distances (dist_sum = sum(dist_inv))
            - Get weights for each K BG (weights = dist_inv/dist_sum)
            - Calc new predictor (np = sum(RR[dist_indexes]*weights)
            - Add np to table holding all new predictors
- Return table of new predictors
- End

2.1.3 Census 2000 Return Rates

Return rate data at the levels of state, county, and tract were found for the 2000 census and were posted onto the Kaggle forum. These values were then matched to every block group and tested for prediction. A small benefit to using these values was noted. For the final submission, only the tract and county levels were used, as those were the only ones supplied in the final given aggregated data set.

2.1.4 External Social Data

Another set of external data came from the instructor on Angel. This data included predictors based on social factors, including poverty, obesity, etc. On their own, this data showed poor predictive performance. However combined with other data in an existing model, a slight benefit was noted. These predictors were also used in the final submission.

2.2 Models

A number of different training models and methods were investigated during the course of this project. These models included a variety of linear models, neural networks, support vector machines, random forests, and a generalized boosted regression model.

2.2.1 Linear

Linear models were initially investigated for a number of reasons. At the beginning of work on this project, classwork involving linear models coincided, which resulted in code being readily available. In addition, linear models were very quick to handle the large datasets, which helped tremendously while doing the initial investigations/scaling/etc. of the data. A minimum error of about 3.6 was achieved during these initial efforts. As the project progressed, it became apparent that linear models did not have the needed complexity to accurately predict the return rates to a substantial degree. Therefore more advanced models were sought.

2.2.2 Support Vector Machine

Support vector machines (SVMs) were the next class of models looked at during the project. The R package ‘e1071’ was used for this. Using a non-linear kernel, a substantial decrease in model error was observed. However, calculation on the full dataset took a considerable amount of time. Often, computation would not be able to complete within the 24 hour time limit imposed on Penn State's Lion-XC computing cluster.

In order to get around the long computation times, prediction was done using multiple smaller SVMs that were trained with only a subset of the training data. Each model was then used for prediction on the full test set, with the results being averaged as sort of a naive stacking. A parametric study was also completed on some of the adjustable parameters within the SVM model. Of particular importance were the cost and gamma values. It was found that these values shared some dependence, and an ideal ratio between them was (cost * 10) = (1 / gamma) regardless of magnitude. Different kernels were also investigated, with the default of 'radial' appearing to produce the best results.

Relatively good success, and a weighted error of about 3.2, was achieved by averaging 20 models each trained with 20k random datapoints. However, a different model was sought that would be able to train using the whole set, as it was anticipated that this would improve the error.

2.2.3 Neural Networks

Neural networks were the next model investigated. The primary motivation was that with neural networks, training times increase proportionally to an increasing number of training points, as compared to SVMs, which increase exponentially. Software packages in R were investigated for usage. However, due to a lack of customizability of parameters and missing training features, a previously self-built neural network trainer was utilized. A range of network structures and parameters were investigated. However, no improvement was observed when compared to using SVMs and training times were still fairly substantial. The best Kaggle submission using neural networks resulted in a weighted error of about 3.3.

2.2.4 Random Forest

Training times with SMVs and NNs were still pretty considerable. Based on the success that other teams within the class had while using random forests, this model was the next one attempted. Training could be completed for the entire dataset in a number of hours. However, the error was not lower than that achieved using multiple trained SVMs. Random forests were successfully used in conjunction with clustering, as will be explained later.

2.2.5 Generalized Boosted Regression Models

After the Kaggle contests had ended, the leaders submitted their models and code to the discussion forum. Overwhelmingly, a majority of the successful competitors had used generalized boosted regression models from the 'gbm' R package [4,5,6]. Use of this model showed instant success, with the lowest error values observed yet out of all of the tested models.

Parameters were chosen based off of Paul Mineiro's posted solution [4]. These parameters are given in Table 2. Him and his teammate, maternaj, finished 5th in the competition with an error of 2.62. Based on the postings on the forum by Paul and others, it seemed as though increasing the number of trees used for training, even past 20k, slowly decreased the testing error in the model. For the final submission, a limited value of 5000 trees was selected for two reasons. One reason was that it consistently resulted in errors below 2.9, giving full credit for that grading portion of the project. The second reason was that it kept the training time beneath the 12 hour time limit for training times. Training times for this model were about 5 hours on (hammer.rcc.psu.edu) and about 10 hours on (lionxc.aset.psu.edu) for each of the final data splits.

Parameter	Value
distribution	"laplace"
n.trees	5000
shrinkage	0.025
interaction.depth	16
bag.fraction	0.5
n.minobsinnode	10
keep.data	TRUE

Table 2 – Paul Mineiro's gbm Parameters

2.3 Clustering

In addition to investigating multiple training models, further exploiting the spatial dependence of the return rate was attempted through clustering. Clustering was initially tried through various hard coded boundaries based on state borders. This was noted to result in low errors, as well as being quick and easy to program. Other unsupervised clustering methods were

attempted additionally, but long computation times and memory limitations resulted in only K means being used.

2.3.1 State-Based Clustering

Clustering using boundaries that were hard-coded based on state borders was attempted after it was noted as successful in a presentation by the team of the Stragglers. Training and testing data for each state individually showed a substantial improvement over previous methods. Different boundaries were also investigated, including creating boundaries based on cultural differences (i.e. Deep South, New England, Pacific NW, etc.), as well as boundaries which tried to take into account the geographic dependence of return rate as was demonstrated in Figure 1. However, neither of these groupings showed any improvement over training each state individually.

It was apparent by looking at the results that some states consistently got high testing/validation errors. Focus was then placed on attempts to improve the 'bad states' by training them in conjunction with neighboring 'good states'. This did result in lower errors for these certain states. However, in a vast majority of cases, combining the training resulted in a higher increase in error of the good state than decrease in error of the bad state, which resulted in a higher error overall. Finally, it was determined to simply train all of the bad states, given in Table 3, together. This showed the best, although it was not substantial, improvement over training each of these states individually.

Combined States
Alaska
District of Columbia
West Virginia
Wyoming
Rhode Island
New Mexico

Table 3 – The states combined during Training. All other states were trained individually.

2.3.2 K Means Clustering

Based on the success observed with state-based clustering, unsupervised clustering was next attempted. Initially this was attempted using only the coordinate data for each block group using various clustering methods. However, memory limitations and computation time limited the only available clustering algorithm to K means. Values of K between 5 and 200 were analyzed. This gave uninteresting results overall. Next, it was noted that states which have a low variation in return rates tended to result in the lowest training errors when using state-based clustering. Therefore, the next attempted method was to include influence from the calculated KNN return rate predictors when determining clusters.

While including the return rate, the two coordinate data points were centered and normalized with each other, under the assumption that (1 degree latitude == 1 degree longitude). The KNN return rate was then centered and normalized, as well as weighted. Weighting was done because it was unknown how much influence the KNN values were to be given relative to coordinates when determining cluster assignments. A wide range of K values and weightings were tested. It was found that a weighting of about (0.1) and K value around 10 produced the best results when training with random forest. Average error using a random training/validation split approached a value of 2.8. However, when applying this technique to the test data, the submission error was unable to get below 3.05. This was likely attributable to the fact that clustering with the training/validation was more accurate, as the KNN return rate values included data from both sets while a training/testing split only used data from the training set.

3.0 Results

The best submission to the Kaggle competition included state-based clustering with training being done using both random forest and support vector machine models. The input data included the external and KNN return rate values as previously mentioned, with the scaling and data removal done as well. Initially, predictions were done with the random forest and support vector machines separately. However, taking the average of both models seemingly helped push down the final error a little bit. The error at the end of the competition was 2.98, which resulted in a ranking of 27/244.

The final submission for this project was done using the normal input data handled as before, minus the KNN imputation step. Imputed values for most of the columns that this was previously done were taken from the aggregated data file on Angel. For the other predictors, the NA values were simply set to zero. External demographic data (obesity, etc) was used from the previously supplied file. KNN return rate values were calculated based on in-state neighbors for values of K=5,15,50,100,200,500. The model used was a gbm trained with 5000 trees and the other parameters as given in Table 2. The final average error value as well as the errors for each i=1..5 iteration are given in Table 4. As can be seen, using the gbm dropped the error considerably when compared to previous efforts.

Training Iteration	**Weighted Mean Error**
1	2.873811
2	2.870458
3	2.870604
4	2.873811
5	2.828994
Average	2.863536

Table 4 – Final Model Errors for each of the Five Training/Testing Splits

Further work was done to investigate just how low of an error could be achieved without direct concern for processing times. Therefore, a gbm model was further trained using only the

single split of i=1. Figure 2 is a plot of the weighted error taken at intervals of 1000 trees. From this figure, it appears only about an additional 1% error reduction was achieved after 15k trees compared to using 5k. It is likely that the additional data and predictors created and used by the competition leaders had a higher influence on their score than simply using a lot of trees with the gbm model.

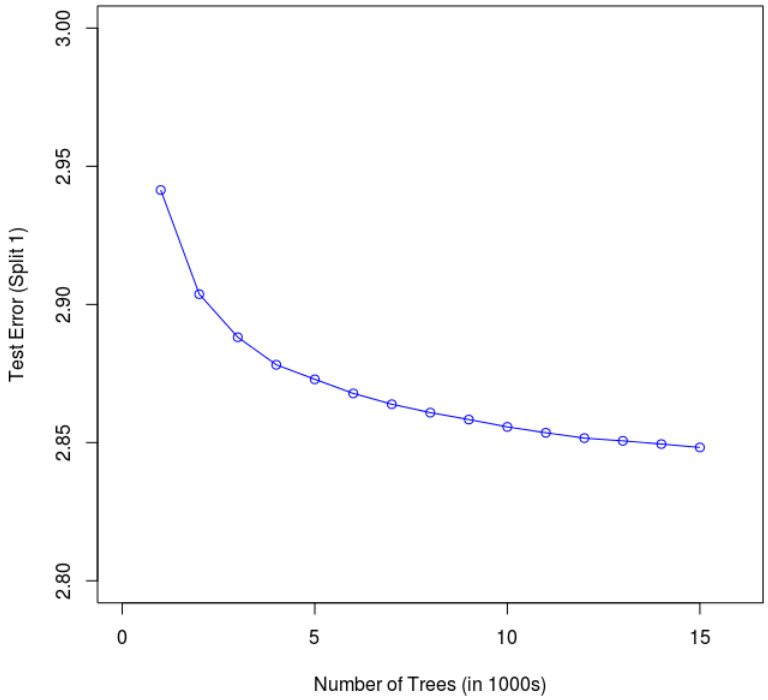

Figure 2 – Model Error as a Function of Number of Trees

4.0 Discussion

There were a number of valuable lessons learned while doing this project in terms of both the Census competition as well as data mining in general. Properly trimming and scaling the data was the first important step. Actual knowledge of the meaning behind the data is noted to be very important in terms of modeling it and using it for predictions. In addition, the external data sources added additional predictive ability to the models. Using all available data sources is undoubtedly important.

Based on the success of clustering by state as well as the substantial predictive ability of the KNN return rate predictors, the spatial dependence of the return rate cannot be understated. Interesting results, such as the large return rate gradients across state boundaries, demonstrate that this spatial dependence can be both hard and soft. This further demonstrates the importance of having intimate knowledge of the reasons behind why the data looks and behaves as it does.

It was also discovered that, in addition to exploring all available data sources, multiple different training models should be attempted. Different properties of the data are likely best exploited through fitting with different model types. In the instance of the data used for this

project,the generalized boosted regression model run with a certain parameter set seemed to indicate the best fit for this data. However, due to the long computation times needed for this model, there are likely other instances where the tradeoff between accuracy/time needed becomes more important. In those instances especially, using other models would likely be necessary.

5.0 Future Work

Using the knowledge gained during this project, there are a number of areas for possible improvement. Looking at forum posts by B Yang and OldMilwaukee [6], there might be considerable interaction between certain inputs in the data. Based on their posts, interaction between owner/renter ratios as well as the population of certain age groups might be important. Figure 3, made by kubqr1 [7], demonstrates a way in which these variables interact. It appears that the proportion of renters to owners as well as the proportion of certain age groups within the block group have a high effect on the overall return rate. These interactions were not fully investigated during this project and thus were not included in the final model.

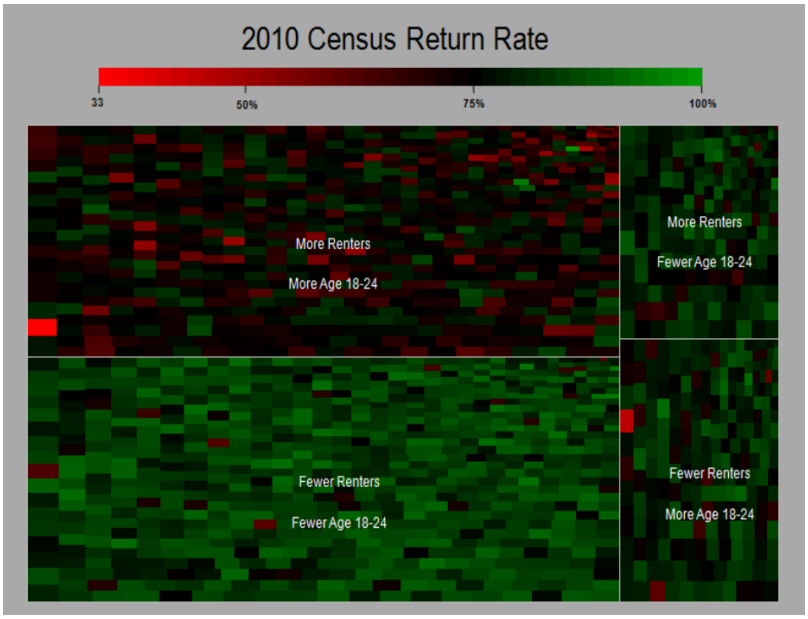

Figure 3 - Return Rates by Percent 18-24 and Percent Renters. Created by kubqr1.

More complex models could also be investigated. Possible methods might include more advanced clustering or model stacking and boosting methods. Further exploiting the spatial dependence of return rates might be applicable with this further work.

6.0 References

1. YetiMan, Kaggle Census Competition Forums,<http://www.kaggle.com/c/us-census-challenge/forums/t/2511/external-data-deadline-for-new-data-sources-is-passed?page=8>. Retrieved 17 Dec 2012.
2. Maternaj, Kaggle Census Competition Forums, <http://www.kaggle.com/c/us-census-challenge/forums/t/2511/external-data-deadline-for-new-data-sources-is-passed?page=9>. Retrieved 17 Dec 2012.
3. Andrew L, 2010 Return Rate by County Map, Kaggle Visualization Competition, <http://www.kaggle.com/c/us-census-challenge/prospector#223>. Retrieved 17 Dec 2012.
4. Paul Miniero, For Peer Review, Kaggle Census Competition Forum, <http://www.kaggle.com/c/us-census-challenge/forums/t/3065/for-peer-review>. Retrieved 17 Dec 2012.
5. YetiMan, For peer review, as per the rules, Kaggle Census Competition Forum, <http://www.kaggle.com/c/us-census-challenge/forums/t/3051/for-peer-review-as-per-the-rules>. Retrieved 17 Dec 2012.
6. OldMilwaukee, Solutions to the Challenge, Kaggle Census Competition Forum, <http://www.kaggle.com/c/us-census-challenge/forums/t/3045/solutions-to-the-challenge>. Retrieved 17 Dec 2012.
7. kubqr1, Return Rates by Percent 18-24 and Percent Renters, <http://www.kaggle.com/c/us-census-challenge/prospector#226>. Retrieved 17 Dec 2012.