Predicting City Consumption

Jim Caine, Matt Robison, Jonny Swift

June, 2015

DePaul University

CSC 672

1 Introduction

The supply and consumption of renewable energy is expected to rise significantly in the near future, the rate at which it does dependent on a number of factors such as the price of natural gas in the future, the implementation of policies aimed at reducing greenhouse gas emissions, and advances in renewable energy technology. However, before increasing reliability on sustainable energy from renewable sources, individual cities first must understand the dynamics of their current electrical energy consumption and have a good idea of how much energy is needed where and when.

The fictional Power City, USA is looking to forecast hourly electrical energy consumption for their entire city, focusing on 8 key city sectors: residential, office, K-12 schools, grocery, food service, lodging, retail, and health care. Energy consumption is dependent on a number of variables including weather, date, time, type of day, population, and sector, and because of all these heavily influencing factors, the consumption of energy is extremely volatile and changes not only on a daily basis, but also an hourly basis. In this report, predictive models are built to forecast the hourly electrical energy consumption for Power City for any sector and hour of the day.

Feature extraction and a normalization process is applied to the dataset to smooth the data and explain more variance in the data. Additional features are then derived from the dataset including the square foot per person for each sector, the change in continuous variables over time, the rolling mean for continuous variables, and bins for the categorical variables.

Two types of algorithms are applied to the dataset to predict consumption: a linear regression model and a decision tree model. The decision tree model performed better than the linear model, most likely because the decision tree model accounted for interactions between the variables within the model. Several techniques were applied to the model for improved performance, such as careful feature selection, feature transformations, fitting the model on different segments of the data (sectors), and using the MARS algorithm to naturally introduce splines and interactions into the model. However, the linear model still underperformed when compared to the decision tree model. The model is still improved by applying several ensemble techniques using the decision tree model as the base estimator. In particular, AdaBoost using deep trees (as opposed to decision stumps) is seen to improve the model by improving decision boundaries with large errors. Finally, a stacking model improves accuracy even further using the predictions from the random forest and AdaBoost models as input. This final stacking model achieves a mean average error of 1.12 x 10^{-4} KW/SQFT per hour and sector, or 0.0215 KW/SQFT (or 4.82%) over all sectors for a full calendar day, meaning our final predictive model was very accurate in forecasting electrical energy consumption in Power City.

2 Data Preprocessing

Five different datasets are provided describing electricity consumption per hour and sector combination along with various attributes that are thought to explain electricity consumption. The five disparate datasets are merged into a single dataset for analysis. First, weather attributes that are given per hour of the day are joined to the consumption data by month, hour, and day. Similarly, the solar angle is joined to the dataset on month, hour, and day. Next, attributes describing the day of the week are joined to the dataset by month, day, and hour. Finally, the square foot of each sector is found. To do so, the population for different age segments of each sector is given for multiple tract IDs and are summed together to find the population of Power City per age segment. This is multiplied by the square footage per person in each age segment per sector. The square footage per sector is then joined to the main dataset by sector.

The season and hour feature are binned to smooth the data and reduce the dimensionality when using bins instead of dummy variables. The first three months are mapped to Winter, and increments of 3 months are mapped to the appropriate season to create the season bins. The hours are binned into increments of 4 to create the hour bins, where the first four hours are binned, the second four are binned, and so on. The weekdays and holidays features are binned to create two additional flag variables that indicate whether the day of the week is a weekend and whether the day of the week is a holiday.

Interaction terms are also added to the dataset. The exhaustive list of combinations between the variables is very large, so three different approaches were taken: exhaustive PCA, adding interaction terms intuitively, and allowing the interaction terms to be added during modeling via MARS. In exhaustive PCA, all possible combinations of terms that do not result in a zero-variance vector are added to a dataset and are transformed into principal components. As shown in Figure 1, no single principal component explained most of the variance in the dataset, however, after about 25 components, the incremental variance explained flattens out. Overall, the principal components were found to provide little if any improvement in the model, so they are removed from the dataset. Initially interaction terms were created by running a linear regression model, determining the significant variables by their p-values, creating second order interaction terms, adding them to the regression model, and then fitting the new model to the data. This proved to be unwieldy, as there were 96 variables in the data set, including the original terms and some transformed variables. As a solution, a MARS model was created to automatically create a polynomial regression model.

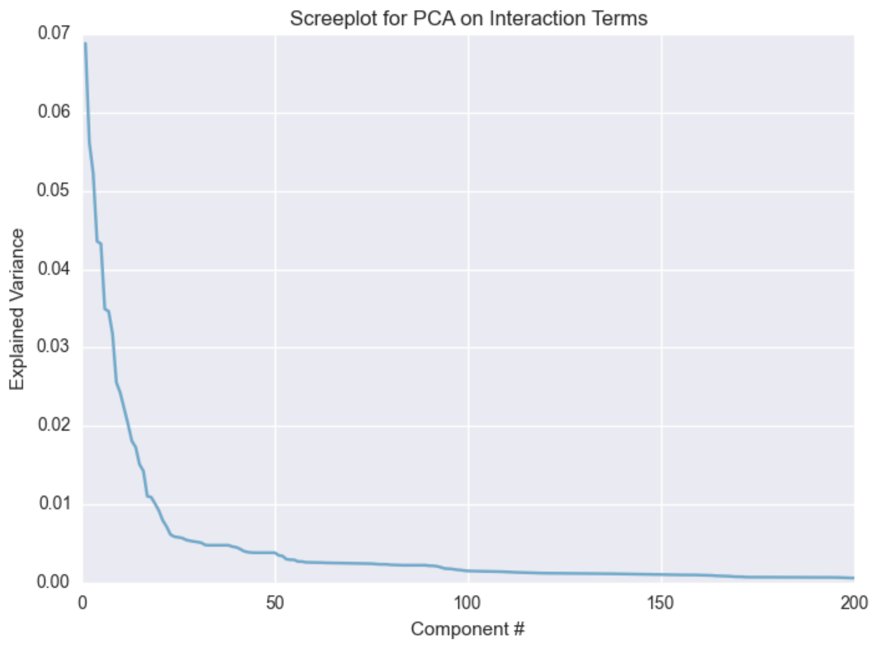


Figure 1: A screeplot of the interaction terms shows indicates that the majority of the variance can be explained by 25 components.

It is thought that the change in features could affect consumption patterns. For example, consumption might be less on Friday because it is closer to the weekend, or dramatic decreases in temperature could send people running for their thermostats. To account for this, hour over hour change and rolling mean variables are created. The hour over hour change features simply represent the value of the feature one-hour prior subtracted by the value of the feature that particular hour. Similarly, the rolling mean feature finds the average of the feature over the past full day.

Finally, the data is transformed into standard spreadsheet format by normalizing the continuous variables using a z-score normalization and representing the categorical variables by dummy variables. A full description of the final schema is presented in Appendix A. The entire feature set is a mass over-representation of the data, and therefore, is not used in its entirety when modeling. The feature selection process for each model is described in detail in the Models section.

3 Data Exploration

Figure 2 shows the frequency of binned consumption over all sectors as well as over each individual sector. From the figure, it is seen that consumption is not normal. It can also be considered multi-modal, as there are many peaks and values as the mean consumption varies. This is partially explained by sector. The food service sector has large consumption overall, and the peaks for larger values of consumption are seen to be heavily influenced by the food service sector. Each individual sector is also seen to be multimodal, and therefore, other features must be responsible for forming these clusters.

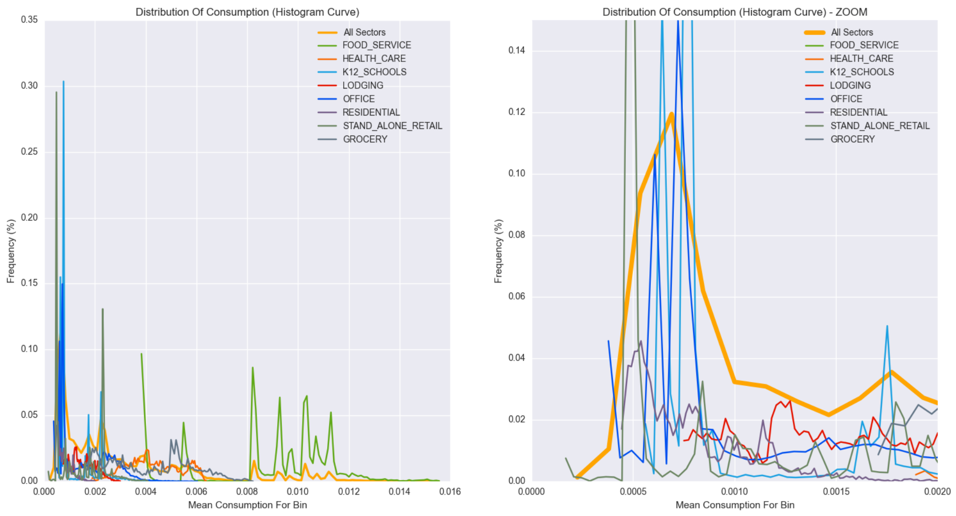


Figure 2: Distribution of consumption for sectors show non-normal behaviors.

The boxplot in Figure 3 of Consumption by Sector show the varying ranges in the distribution of consumption between each sector and the difference in the size of consumption between the sectors. Noticeably, Food Service is the sector with the greatest consumption of energy and the most variance in its consumption patterns. It exhibits the same pattern, of energy use broken into 3 distinct periods, in both this graph and the Month, Day and Hour chart below. Not unexpectedly, it was one of the most difficult sectors to model.

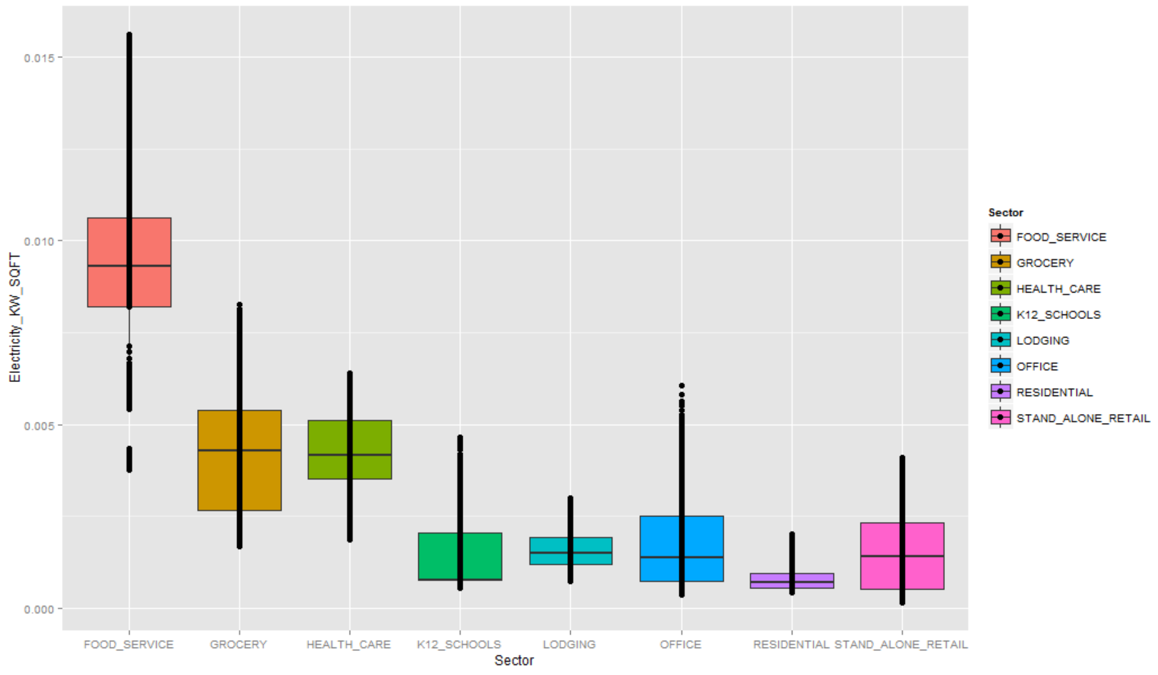


Figure 3: A box plot shows the magnitude, variance, and outliers for the consumption for each sector.

Figure 4 shows consumption over the calendar year. Each data point represents the average consumption over a single calendar day. The all sectors category is also averaged over all sectors. Three trends in the dataset are identified in Figure X. First, consumption is dependent on the time of the year – consumption is higher and more volatile in the summer months than the winter months. Second, the time of the year effects each sector differently. For example, the lodging increases consumption earlier than the food service sector. Finally, the ups and downs throughout the entire calendar year indicates that consumption is dependent on the day of the week..

Histograms of the continuous variables can be seen in Appendix 1. The temperature and solar elevation features are both bimodal, likely being influenced by the winter and summer seasons. The remaining weather variables appear to follow "norma-ish" distributions, each having a fat tail to some degree. Because these features follow this type of pattern, a z-score normalization is favored throughout model building and is seen to result in models that outperform models with min-max transformed features.

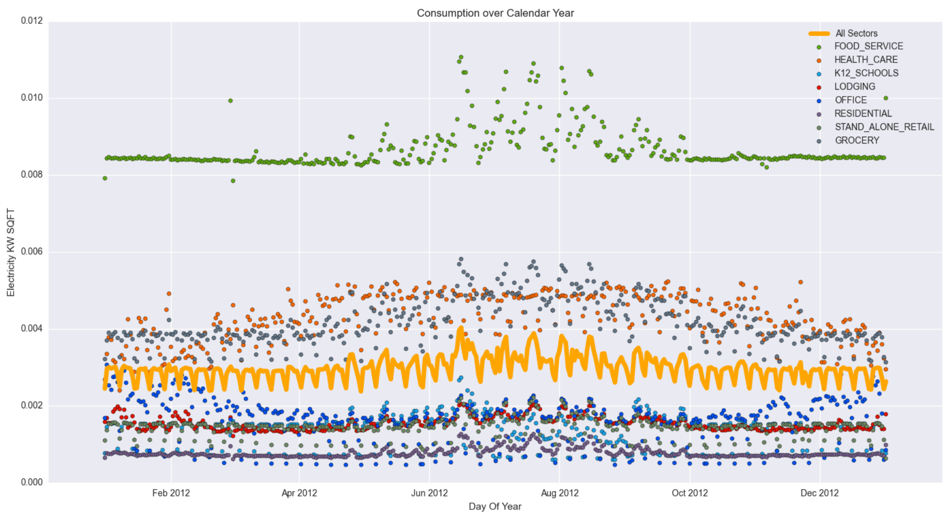


Figure 4: Consumption patterns increase and become more volatile in the summer.

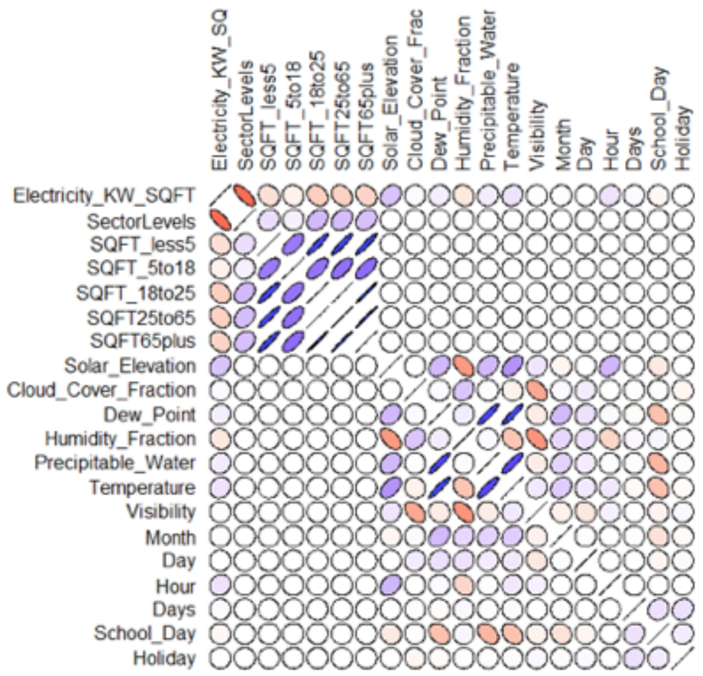


Figure 5: Correlation matrix indicates relationships between all variables in the original dataset.

The correlation matrix shows that the Square Foot per person variables, as well as Temperature, Humidity, and Dew Point are highly correlated. This isn't surprising as they are based on the same underlying data and are used to create another term, i.e. Temperature and Dew Point equal Humidity. Sector Levels are negatively correlated with the energy consumption variable. This arises because each sector isn't broken out separately, and one sector's usage means the others have none, throwing the correlations off. Further models will break the sectors into eight separate models.

The chart of energy consumption over All sectors shows higher usage across all sectors in the Summer months, 6, 7 and 8, and an oddly fractured pattern of usage overall.

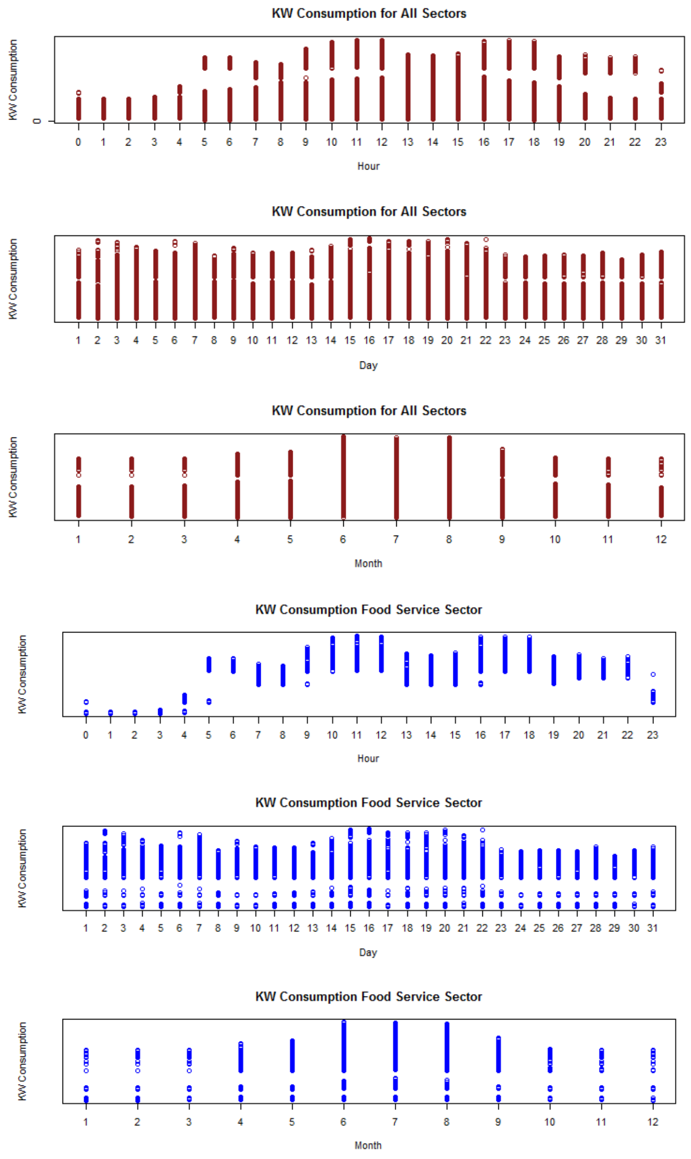


Figure 6

4 Methodology & Models

Six total models are created to predict the consumption for any observation. 20% of the dataset is partitioned into a testing set to ensure the model generalizes well to unseen data. The primary metrics for evaluation are R2 and Mean Absolute Error (MAE). MAE is chosen because minimizing MAE best achieves the goal of predicting consumption as well as possible. In other words, a prediction that is twice the magnitude of another prediction should be penalized only twice as much. Each model is tested X times to obtain a 95% confidence interval to ensure evaluation stability

The model is created over all sectors and includes sector dummy variables within the feature set, meaning that the main goal is to minimize consumption over all sectors. Therefore, our model will focus more heavily on sectors with high consumption. The performance for each sector is still recorded for these models, and an experiment of creating eight different models is also explored..

Six different prediction algorithms corresponding to the six models are applied to the dataset. The accuracy of the models is shown in the Analysis section along with a comparison and analysis of the different models.

4.1 Multiple Linear Regression

The first model we were interested in examining was a multiple linear regression model in order to see if the data was linearly related and could be modeled using a linear predictor function. Linear regression is often used to create predictive models to forecast dependent variable values, and after exploring the data, we could see that quite a few independent variables were linearly correlated with energy consumption. Using regression analysis, our goal was to create an equation that we could use to accurately calculate and forecast the hourly electrical energy consumption in Power City, based off of other influential factors.

After running multiple linear regression on our merged consumption dataset with a 80% training/20% testing split and performing backward stepwise variable selection, the model based on the testing set was whittled down to 32 significant variables and achieved an R2 of 86.74% on the testing dataset, indicating the model fit the data quite well. It achieved a MAE value of 7.57e-04, which was 25.13% of the average energy consumption in Power City. After analyzing the residual plots (seen in Appendix AAA1), we could see that there was definitely some heteroscedasticity in the data, as well as some distinct clusters that were likely showing the behavioral differences between different sectors.

Because some sectors had much higher energy consumption levels, higher population levels, and/or abnormal consumption patterns, we decided to split up the 8 sectors into separate datasets and to create a separate regression model for each sector in order to see if the model metrics could be improved (see model comparison chart in Appendix BBB. Although the R2 values were lower for every individual sector model (except for Grocery) when compared to the overall merged model, the MAE values improved a decent amount for all sectors except for Food Service (average error rate of the sectors is around 17.5%, while the overall model was 25.13%). Therefore, since the MAE and error rate were minimized, splitting up the data into individual sectors proved to be the best route for linear regression modeling.

In order to combat some of the heteroscedasticity in the data, we experimented by incorporating some more variable transformations and interaction terms into the data. For all 8 individual sector models as well as the overall merged consumption model, we experimented by performing log transformations on the dependent variable and by intuitively testing different combinations of interaction terms up to the fourth order for significant factors. Taking the log of the dependent variable proved to not make much of a difference in model performance, however by incorporating significant interaction terms, we were able to significantly improve the metrics and performance of all of the models. The overall merged consumption model achieved an R2 of %, a MAE of , and an error rate of % on the testing set (improved from R2 of 86.74%, MAE of 7.57e-04, and error rate of 25.13%), and all of the metrics of the individual sector models also improved significantly, as seen in the model comparison table in Appendix BBB.

Therefore, when it comes to linear regression modeling, the best method entails incorporating significant interaction terms into 8 individual sector models (although this was not the case with all of the modeling methods we tried). Also, even though these methods greatly improved the predictive accuracy of the model, there was still some heteroscedasticity in the data and room for improvement in our metrics and predictions, so we looked at implementing the MARS modeling method.

4.2 MARS

We chose to implement the MARS method because from the descriptive statistics and regression plots, (see Appendix AAA) it was apparent that a linear regression model wouldn't fit the data perfectly. We needed to create numerous interaction variables and develop a polynomial regression model, and the MARS method automated that process.

The "earth" R package builds polynomial regression models using the techniques in Friedman's papers, "Multivariate Adaptive Regression Splines" and "Fast MARS". The term "MARS" is trademarked and thus not used in the name of the package. A backronym for "earth" is "Enhanced Adaptive Regression Through Hinges".

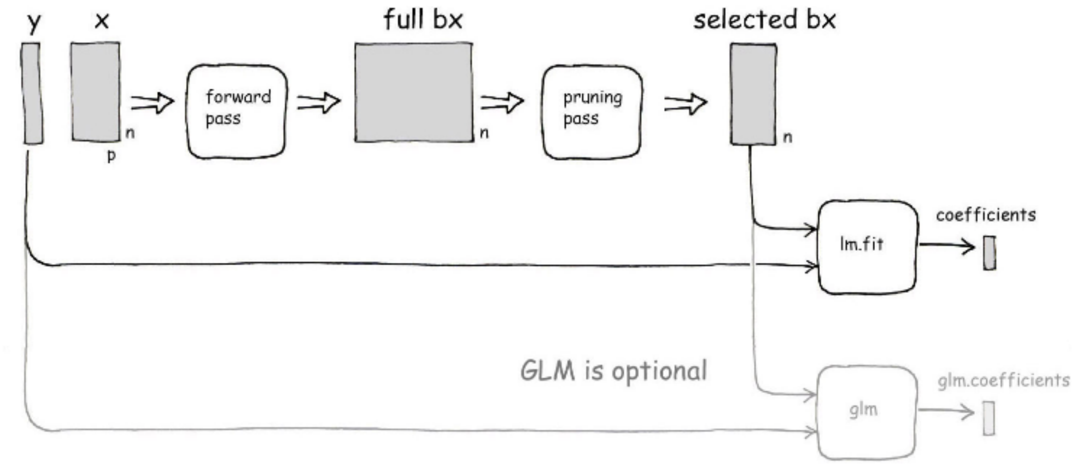


Figure 7

The process, as shown in Figure 7 above, uses the dependent variable, y, and some independent variables, x_1 through xn to create a polynomial regression function by first taking a forward pass through the independent variables to create a matrix of terms, then a pruning pass to create and cross-validate models based on subsets of the terms; choosing the best model based on a the best statistics, and finally fitting that model to the distribution of the data.

The forward pass adds terms in pairs until the first of the following conditions is met: the maximum number of terms has been reached; adding a term changes R2 by less than 0.001; R2 reached a value of 0.999 or more; GR2 is less than -10 (the model is overparameterized); or no new term increases R2.

The pruning pass, or backward pass, determines the subset of terms with the lowest Generalized Cross-Validation (GVC) score, and updates the matrix of terms to reflect that. Finally, the process calls lm.fit on the data using the dependent variable and the final subset of terms created in the pruning pass.

There are numerous arguments that can be used in ‘earth” models, for this model I used:

Parameter	Description
Linpreds	Specifies the predictors that should enter linearly, instead of in hinge functions. In such, we can control which term is included in the forward pass of the “earth” process. In its current implementation, the GCV penalty for predictors that enter linearly is the same as that for predictors with knots.
Pmethod	Determines the pruning method used; forward, backward or exhaustive. Backward is the fastest computationally, so that is the method I used.
Varmod.method	Determines the type of model to use to determine the variance in the data, in this case I chose “earth”.
Ncross	Because the process randomly splits the data into nfolds each time, varying CV results are obtained. By setting the ncross argument to >1, usually 10 to 20, the “earth” model averages the RSS of each cross-validation pass to create a GVC.
Nfold	Determines the size of the split the process uses in its cross-validations. Usually 5 or 10, for the standard 80/20 or 90/10 split.
degree	Determines the maximum degree of interaction between the terms.
Measures in “earth”	GCV is a form of the RSS penalized by the effective number of model parameters and divided by the number of observations. Generalized R-Square (GRsq) normalizes the GCV in the same way that the R-square normalizes the RSS.

Table 1

The MARS process does automatic variable selection (meaning it includes important variables in the model and excludes unimportant ones). However, bear in mind that variable selection is not a “clean” process and there is usually some arbitrariness in the selection, especially in the presence of multicollinearity. This means that if two variables have some collinearity, then the pruning pass may select one over the other, even if the selected variable isn’t the best variable.

To combat this I chose to limit the independent variables in the model to: Hour (0-23), Weekdays (Sunday – Saturday), Month (1-12), Holidays (New Years – No Holiday), and Rolling 1-day means of the z-scores for the Weather variables (Solar Elevation – Visibility).

Given more time, further investigation into the other parameters of the MARS modeling method could produce a better model than the one shown in the model comparison table, but with little hope of exceeding the final model we chose.

4.3 Decision Tree

As a contrast to the linear models, a decision tree model generates a set of rules based on the values of the features in the dataset. Because the decision tree is grown downward, interactions amongst the features should naturally be accounted for. This particular problem is a prediction problem, so the algorithm applied to the dataset is the CART algorithm. This algorithm finds the best feature and value to create a rule to split the data observations into two separate nodes. This continues as the tree grows in depth until a stopping criterion is reached. The prediction for the final node is then a linear regression prediction based on all of the observations that are in the node.

Feature selection is not as important when building the decision tree in comparison to a linear model because the model will only be built on features that provide the most information gain, and thus, ignore irrelevant features. **All features from the dataset are used in creating the decision model except the rolling mean and derivative features. The decision to remove these two sets of variables was made as this creates a simpler model while the difference in performance is negligible.**

Three main parameters are varied to optimize the model: the maximum depth of the tree, the minimum number of samples in a leaf node to split, and the maximum number of features in the dataset to use in the model. The performance for the varied parameters is shown in Appendix 2A.

4.4 Random Forest

Random forests were introduced in 2001 by Leo Breiman [2]. The main idea in the random forest algorithm is to train multiple diverse estimators to get an improved prediction when all are combined using a weighted average. This is similar to how humans behave, where we seek out multiple opinions before we are satisfied. The important component in the model is diversity – with no diversity, the prediction would be the same as a single decision tree. The random forest algorithms achieves diversity by allowing only a random subset of features to split a node.

In a random forest, it is important to have a large set of features so that is still a large number of features to train a model after a random sample of the entire feature set is taken. In addition, it can be advantageous to have an over-representative dataset when performing random forest, as the different trees obtain diversity through selecting a random subset of features but still have a complete dataset to fit the model.

The random forest model is built using the optimized parameters from the decision tree. Two other parameters are varied to optimize the model: the number of models in the forest and the number of features that are allowed to be used when determining the best split. The performance for varied parameters is shown in Appendix 2B.

4.5 AdaBoost

Similar to the random forest, the AdaBoost algorithm creates a series of base estimators. The primary difference between AdaBoost and random forest is in how diversity is achieved between the estimators. In random forest, diversity is obtained by randomly selecting features for each estimator. In AdaBoost, diversity is obtained by randomly sampling the dataset with replacement. More specifically, for each "boosting iteration" in AdaBoost, a random sample from the dataset with observations with larger errors being more likely to being sampled.

The base estimator for the AdaBoost algorithm can be any prediction model. AdaBoost is often performed using "tree stumps", or very shallow decision trees. As long as the accuracy of the model is greater than 50%, the boosting algorithm should be able to improve it as more models are created. For this particular dataset, however, tree stumps were inadequate as base estimators for AdaBoost. Instead, a decision tree with a maximum depth of 8 is used.

Two parameters are varied to optimize the model: the number of base estimators and the learning rate. The number of base estimators is simply the number of decision trees contained in the model. The learning rate indicates how influential models created in later boosting rounds will be. The lower the learning rate, the more influential models created in later boosting rounds will be. The performance for these varied parameters is shown in Appendix 2C.

4.6 Stacking

Unlike random forest and AdaBoost, stacking only creates one final model. Similar to random forest and AdaBoost, stacking incorporates other models within the final model. Stacking creates several models and uses the predictions from the models as additional features in the dataset. After a base estimator is trained, the train predictions are computed and appended to the training frame, and test predictions are computed and appended to the test frame. The actual values are left out of both the ultimate train and test set, however, are used to compute the train predictions in the first place.

The stacking model is created using a decision tree, random forest, and AdaBoost models as base estimators.

Analysis

Table 2 shows the results for each of the six models created. AdaBoost is the best performing model in terms of minimizing error. The performance for the models over the various sectors are shown in Appendix 3. AdaBoost is also the best performing model for the individual sectors. **Turn this into a paragraph talking about appendix 3 and the sectors. The only problem is that my model is fit towards the entire dataset, where I think your results are for the models that were fit to that specific sector. Perhaps, we can talk about how AdaBoost still is the better performer and then in the conclusion section, we can say that splitting the sectors up is a next step.**

Model	Train MAE	Test MAE	Train R^2	Test R^2	Mean	Error
Linear Regression					2.9×10^{-3}	
MARS					2.9×10^{-3}	
Decision Tree	1.81×10^{-4}	2.10×10^{-4}	99.13%	98.39%	2.9×10^{-3}	**7.02%**
Random Forest	1.63×10^{-4}	1.82×10^{-4}	99.36%	99.08%	2.9×10^{-3}	**6.05%**
AdaBoost	1.12×10^{-4}	1.44×10^{-4}	99.74%	99.31%	2.9×10^{-3}	**4.82%**
Stacking	1.12×10^{-4}	1.47×10^{-4}	99.73%	99.36%	2.9×10^{-3}	**4.96%**

Table 2: AdaBoost minimizes the error out of all six models.

Linear Regression Results

The final linear regression model for all sectors combined with interaction terms introduced had an R^2 of % and a MAE of for the training set and an R^2 of %, MAE of , and Error Rate of % for the testing set. Although these metrics are pretty good and greatly improved after incorporating interaction terms into the model, as we can see from the residual plots in Appendix AAA2, there is still some heteroscedasticity in the data and much room for improvement in the prediction errors.

MARS Results

The final MARS model for all sectors had an R^2 of 94.6% and an MAE of 4.74e-04 for the training set and an R^2 of 94.59%, MAE of 5.11e-04, and an Error Rate of 17.16% for the testing set. Although the metrics of the model are good, a look at the q-q plot in Appendix AAA3 shows that it has difficulty fitting the data at both the highest and lowest ends of consumption. Obvious trends in the data seem to imply that some further transformations of the variables might need to be applied.

Deeper trees outperform shallow trees. Figure 8 shows a plot of the residuals vs. actual consumption for varying levels of maximum depth. When the model is only able to make one split, the predictions are clustered into two distinct segments. As the tree is allowed to split further, the model is able to find distinctions between the different observations, thus allowing more varied and better predictions. As the tree grows to a maximum depth of seven, the predictions and residuals become stable, indicating that there is no benefit to growing the tree any deeper than a maximum depth of eight.

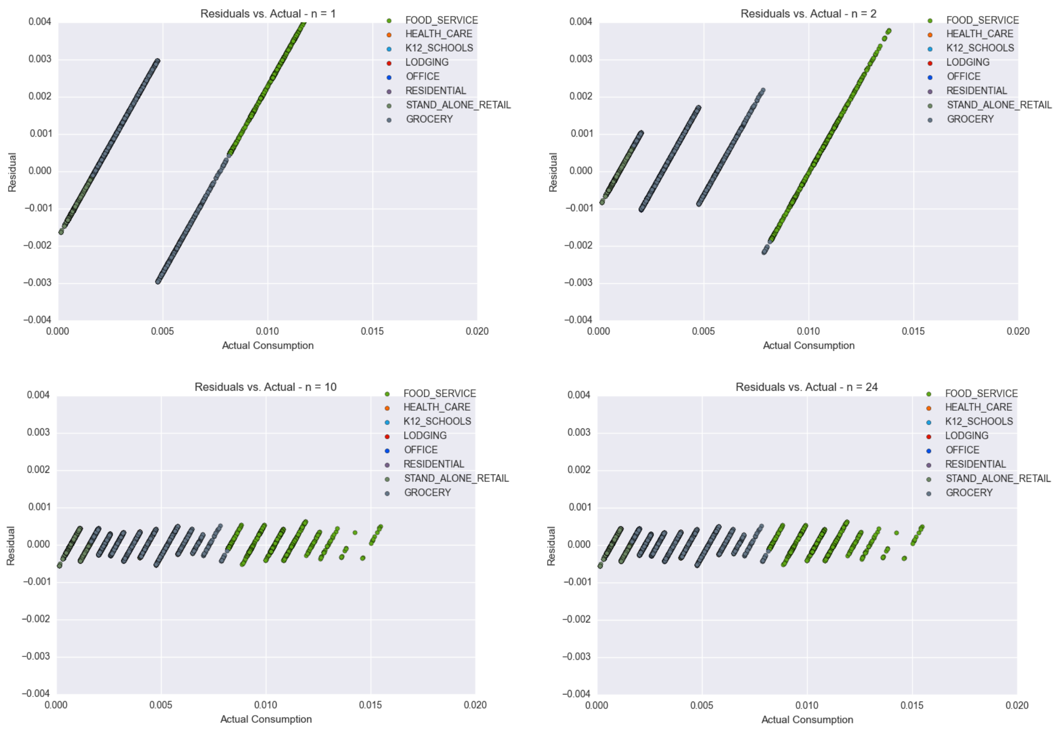


Figure 8: Shallow trees do not allow for the predictions to vary as there are minimal leaf nodes.

Random forest performs better when the number of features considered for each tree in the forest is small. When a random forest considers a larger amount of features, the algorithm performs more like the base estimator (a decision tree in this instance) because diversity is not created, resulting in the trees in the forest to give similar predictions.

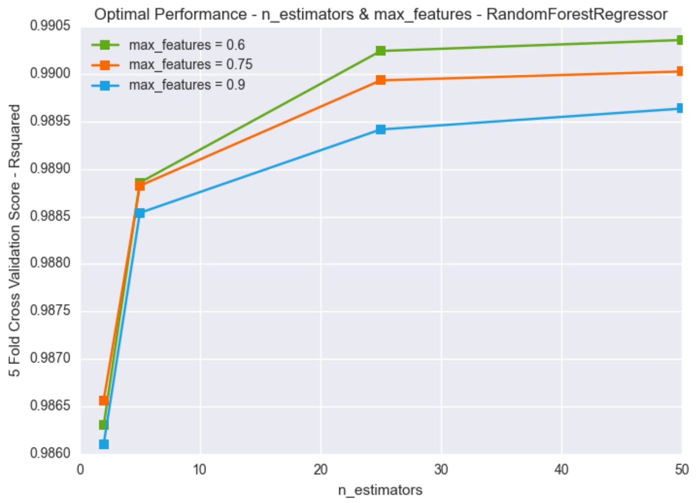


Figure 9: Considering less features for each tree in random forest is essential for diversity and improved accuracy in the model.

Random forest increases decision tree accuracy. Figure 10 shows the random forest model over varied number of estimators. The multiple diverse estimators in the random forest create additional decision boundaries. This is shown in the plots by the relative flattening out of the line clusters.

AdaBoost does a better job than random forest at correcting errors. Figure X shows the AdaBoost model over varied number of estimators. The residuals after only one estimator are identical to a decision tree because the first estimator created in the AdaBoost model is a decision tree. As the number of estimators is increased, the residuals start to improve and move away from the natural line clusters that were created in the decision tree. This is likely due to the diversity that is created by resampling the data. It is also interesting to see that the larger residuals improve faster than the smaller residuals. This is likely an affect of oversampling the observations with the largest residuals at each iteration – each iterative model focuses on training these observations.

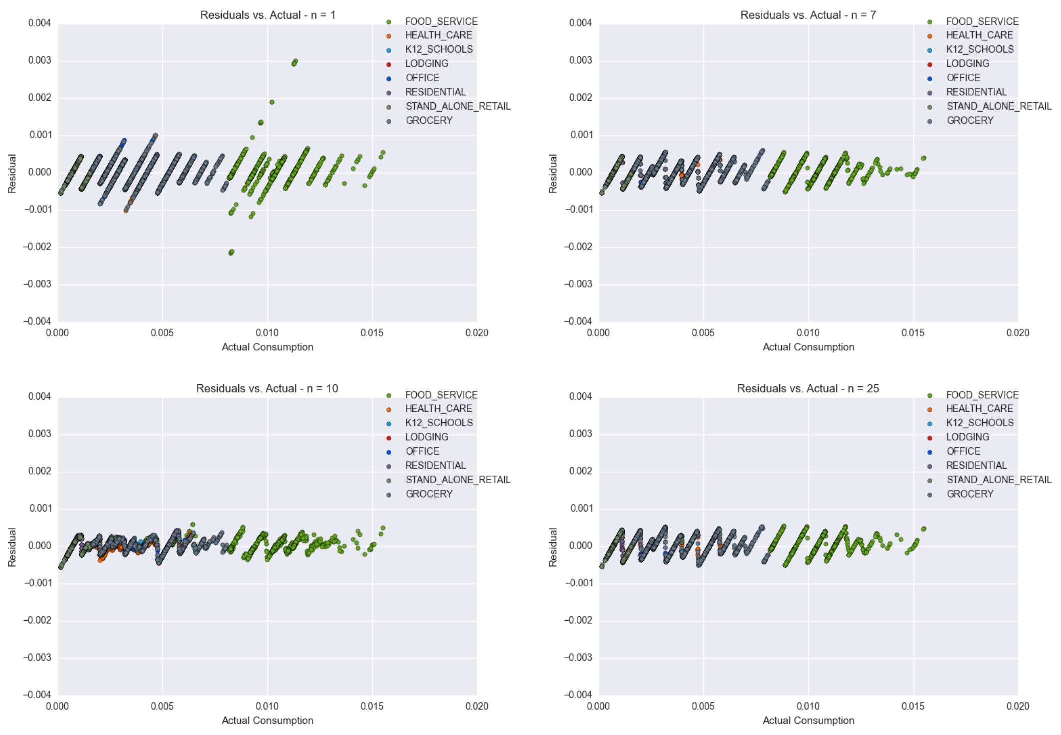


Figure 10: Additional estimators for the random forest model improve performance, but improvement not stable.

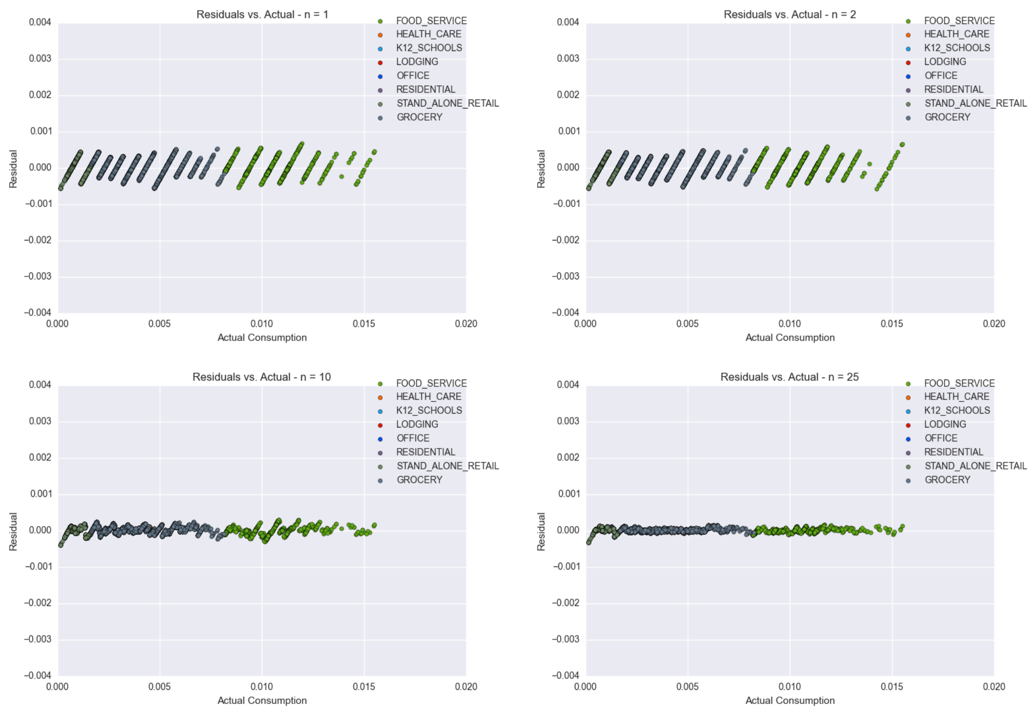


Figure 11: AdaBoost corrects the observations with the highest magnitude of error without affecting other observations

Stacking does not improve upon AdaBoost. Three different ultimate estimators are applied to the stacked dataset: a decision tree, random forest, and AdaBoost estimator. Table 3 shows that the decision tree ultimate estimator far underperforms random forest and AdaBoost, with random forest having slightly better accuracy. The decision tree improves slightly upon the decision tree accuracy without stacking, while random forest and AdaBoost actually performance worse in terms of error.

Ultimate Estimator	Train MAE	Test MAE	Train R2	Test R2	Mean	Error
Decision Tree	1.85×10^{-4}	2.07×10^{-4}	99.32%	98.77%	2.9×10^{-3}	**7.00%**
Random Forest	1.76×10^{-4}	1.97×10^{-4}	99.37%	98.99%	2.9×10^{-3}	**6.66%**
AdaBoost	1.12×10^{-4}	1.47×10^{-4}	99.73%	99.36%	2.9×10^{-3}	**4.96%**

Table 3: Both random forest and AdaBoost models are worse when using a stacked dataset.

The feature importance for the AdaBoost technique on the stacked dataframe is shown in Table 4. The only features considered important in the dataset are the predictions that are created by the base estimators. The decision tree contributes marginally to the prediction, while the AdaBoost attribute contributes towards 96% of the prediction. The main idea in stacking is for the model to learn when to use what model, in this particular instance, AdaBoost outperforms the other models nearly all the time. Thus, stacking does not improve over

Feature	Gini Index
Decision Tree Predictions	0.00009
Random Forest Predictions	0.04
AdaBoost Predictions	0.96
All Other Features	0.00000

Table 4: Stacking uses not other features besides the predictions from random forest and AdaBoost.

Conclusion

In this analysis, it is shown that the decision tree model outperforms the linear model. Further, the performance of the decision tree is improved by both random forest and AdaBoost. Most notably, AdaBoost improves accuracy better than random forest as oversampling the observations with the highest errors caused those observations to be improved without affecting the other observations while creating more diverse estimators as shown by the progress of the residual plot over incremental boosting rounds in AdaBoost.

The final model chosen is the stacked random forest model using predictions from decision tree, random forest, and AdaBoost estimators as additional parameters. The model provides a mean average error of 0.000048 KW/SQFT of electricity per hour and sector, or .0092 KW/SQFT of electricity for the entire city per day. This stacked model achieves most of its predictability from the AdaBoost predictions feature.

The accuracy for our model is very high, nearly predicting the consumption exactly. One explanation for this is that the train set was loaded with data throughout the entire year, thus including data before and after the test observation. Therefore, the model should be applied to the following year's data in a time series fashion to further validate the results.

It may be that the chosen model's accuracy comes from its algorithm which has learned to mirror the algorithm used to create the synthetic data used in the project. It would be interesting to attempt to fit another, similar dataset to see if the model is as accurate at predicting those values.

Given more time, further investigation into the other parameters of the MARS model could provide a better model through that process but with little hope of exceeding the final model we chose.

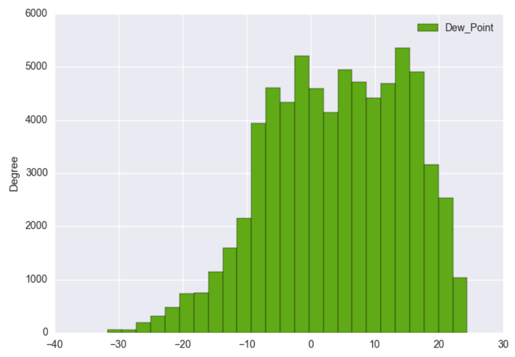
Appendix A: Schema Of Final Dataset

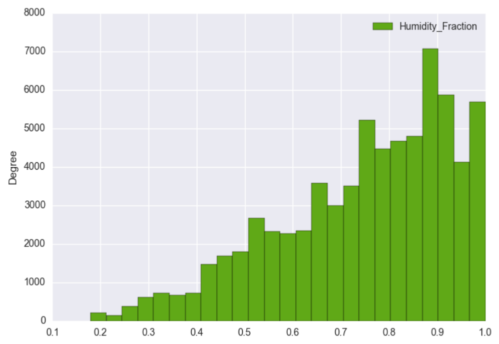
Attribute / Attribute Set	Number of Attributes	Type	Description
Consumption	1	Continuous	Response variable
Hour dummies	23	Dummies	Flag indicating whether the observation occurs in a given hour
Weather		Continuous	

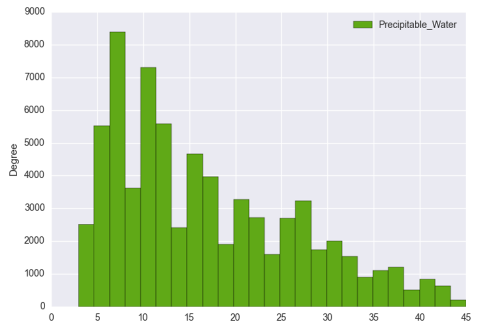
References

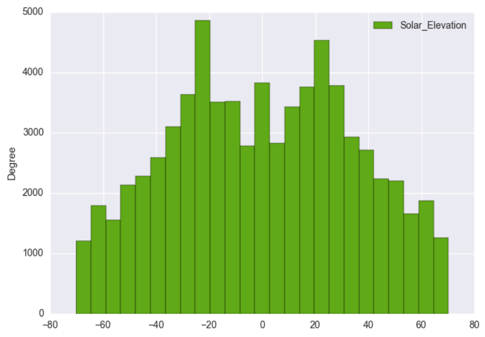
[1] http://www.milbo.org/doc/earth-notes.pdf
[2] Breiman – random forests
[3]
[4]

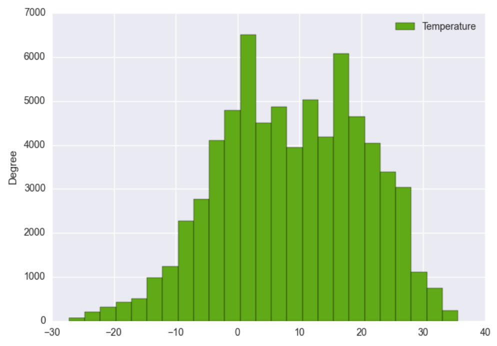
Append X: Histogram of Continuous Variables

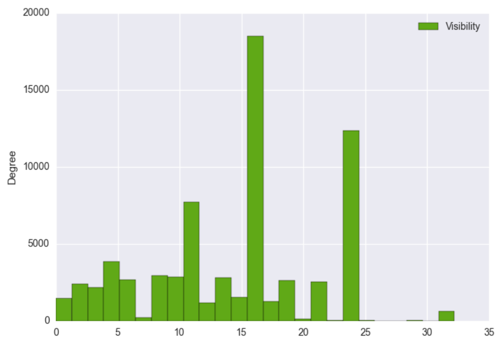












Appendix 2A: Optimization Of Decision Tree

max_depth	max_features	min_samples_leaf	validation_score
5	0.5	1	0.8895
5	0.5	50	0.8952
5	0.5	500	0.8537
5	0.8	1	0.9151
5	0.8	50	0.9115
5	0.8	500	0.8955
5	auto	1	0.9194
5	auto	50	0.9198
5	auto	500	0.9038
10	0.5	1	0.9625
10	0.5	50	0.9581
10	0.5	500	0.9122
10	0.8	1	0.9718
10	0.8	50	0.9642
10	0.8	500	0.9180
10	auto	1	0.9732
10	auto	50	0.9682
10	auto	500	0.9234
20	0.5	1	0.9831
20	0.5	50	0.9742
20	0.5	500	0.9106
20	0.8	1	0.9846
20	0.8	50	0.9766
20	0.8	500	0.9206
20	auto	1	0.9851
20	auto	50	0.9788
20	auto	500	0.9236
50	0.5	1	0.9836
50	0.5	50	0.9754
50	0.5	500	0.9084
50	0.8	1	0.9845
50	0.8	50	0.9775
50	0.8	500	0.9226
50	auto	1	0.9851
50	auto	50	0.9788
50	auto	500	0.9236

Appendix 2B: Optimization Of Random Forest

max_features	n_estimators	validation_score
0.6	2	0.9863
0.6	5	0.9888
0.6	25	0.9902
0.6	50	0.9903
0.75	2	0.9865
0.75	5	0.9888
0.75	25	0.9899
0.75	50	0.9900
0.9	2	0.9861
0.9	5	0.9885
0.9	25	0.9894
0.9	50	0.9896

Appendix 2C: Optimization Of AdaBoost

learning_rate	n_estimators	validation_score
0.5	2	0.9838
0.5	5	0.9879
0.5	25	0.9916
0.5	50	0.9921
2	2	0.9807
2	5	0.9897
2	25	0.9925
2	50	0.9926
3	2	0.9591
3	5	0.9809
3	25	0.9638
3	50	0.9651
5	2	-3.0720
5	5	0.7495
5	25	0.6384
5	50	0.5556

Appendix 3: Results For All Models Over All Sectors

Model	Sector	Test R^2	Test MAE	Test Error
Linear Regression	All			
	Food Service			
	Grocery			
	Health Care			
	K-12 Schools			
	Lodging			
	Office			
	Residential			
	Retail			
MARS	All			
	Food Service			
	Grocery			
	Health Care			
	K-12 Schools			
	Lodging			
	Office			
	Residential			
	Retail			
Decision Tree	**All**	**98.39%**	**2.08 x 10^{-4}**	**7.02%**
	Food Service	95.41%	2.17 x 10^{-4}	2.52%
	Grocery	92.67%	2.09 x 10^{-4}	4.89%
	Health Care	66.42%	2.15 x 10^{-4}	14.02%
	K-12 Schools	81.68%	2.49 x 10^{-4}	15.53%
	Lodging	35.48%	1.83 x 10^{-4}	23.84%
	Office	92.66%	1.86 x 10^{-4}	12.97%
	Residential	95.34%	2.41 x 10^{-4}	5.76%
	Retail	84.69%	1.64 x 10^{-4}	12.49%
Random Forest	**All**	**99.08%**	**1.63 x 10^{-4}**	**5.41%**
	Food Service	98.07%	1.37 x 10^{-4}	1.58%
	Grocery	97.44%	1.68 x 10^{-4}	4.00%
	Health Care	91.16%	1.20 x 10^{-4}	9.01%
	K-12 Schools	76.20%	1.84 x 10^{-4}	11.95%
	Lodging	18.16%	2.06 x 10^{-4}	26.27%
	Office	94.07%	1.66 x 10^{-4}	11.48%
	Residential	94.41%	1.60 x 10^{-4}	3.70%
	Retail	91.48%	1.65 x 10^{-4}	10.10%
AdaBoost	**All**	**99.32%**	**1.44 x 10^{-4}**	**4.83%**
	Food Service	98.11%	1.61 x 10^{-4}	3.84%
	Grocery	92.69%	1.16 x 10^{-4}	8.81%
	Health Care	95.85%	1.36 x 10^{-4}	8.46%
	K-12 Schools	93.51%	1.53 x 10^{-4}	19.48%
	Lodging	96.14%	1.54 x 10^{-4}	10.73%
	Office	97.79%	1.44 x 10^{-4}	1.67%
	Residential	96.78%	1.45 x 10^{-4}	3.37%
	Retail	88.17%	1.43 x 10^{-4}	9.37%
Stacking	**All**	**99.19%**	**1.43 x 10^{-4}**	**4.08%**
	Food Service	96.97%	1.43 x 10^{-4}	1.19%
	Grocery	98.15%	1.43 x 10^{-4}	2.83%
	Health Care	94.76%	1.43 x 10^{-4}	6.87%
	K-12 Schools	55.53%	1.43 x 10^{-4}	21.08%

	Lodging	97.10%	1.22×10^{-4}	2.88%
	Office	89.22%	1.32×10^{-4}	8.50%
	Residential	95.68%	1.40×10^{-4}	9.57%
	Retail	92.07%	0.84×10^{-4}	6.29%

Appendix X-A: Decision Tree Plots Over Max Depth

Appendix X-B: Random Forest Plots Over Varying Number of Estimators

Appendix X-C: AdaBoost Plots Over Varying Number of Estimators