Problem Statement

The Energy Information Administration (EIA) has a plethora of hourly electricity demand data for different regions in the lower 48 states. Energy companies rely on accurate forecasts to predict upcoming demand. The goal of this project is to train a machine learning model on weather/electricity data to generate a more nuanced energy-demand forecast. The client for this type of project would be any energy company looking to better understand how the demand for their electricity varies hour to hour. An improved forecast helps energy companies take next steps to fully capitalize on and perhaps better brace for large spikes in electricity demand. Additionally an improved predictive model will shed insight on the features that most heavily affect electricity demand.

Data Cleaning and Wrangling

My project consists of primarily two data sets: (1) electricity data from the EIA, i.e., electricity demand, and (2) hourly, local climatological data (LCD) downloaded from the National Oceanic and Atmospheric Administration (NOAA) website. To limit the scope of the project, I retrieved both data sets from the city of Los Angeles (specifically, Los Angeles Department of Water and Power data from EIA and the Los Angeles Downtown LCD data). Electricity data were retrieved using EIA's API and then unpacked into a pandas `DataFrame`. Since LCD data are not available via NOAA's API, I manually downloaded LCD data from the NOAA website as a CSV file which I imported to a pandas `DataFrame`. Electricity data contain hourly entries from July 2015 to present. To align the data correctly, I downloaded the LCD data from July 2015 to September 2018 and later cut the electricity data to fit the date range.

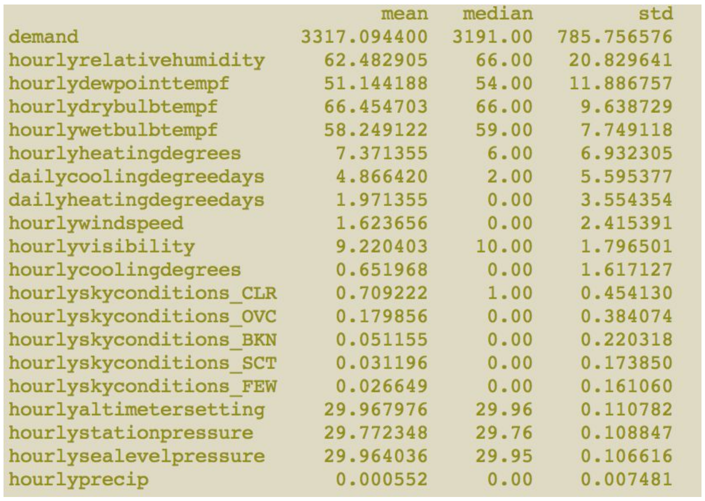
The electricity data required some cleaning--there were very few null values in the set, however there were a very small number of outliers where data was either entered in as zero or a large negative number. Removing outliers cut only ~.01% of the data. LCD data on the other hand required more cleansing. Throughout the cleaning of the LCD data, I frequently referred to the LCD documentation for guidance on how to treat the features. Firstly, all features (except daily cooling/heating degrees) that did not record hourly entries, i.e., monthly and daily records, were dropped. I used `bfill` to fill the the hourly entries of a specific day with that day's heating/cooling degrees. Heating/cooling degree days are the number of degrees below/above 65 degrees Fahrenheit (wet bulb temperature) (negative values are set to zero) respectively and represents the amounting heating/cooling necessary for that temperature. In this way these features are a strong proxy for electricity demand; we specifically expect CDD to be a strong proxy for electricity demand in LA because LA's primary energy consumption is cooling during the summer months. Additionally I added *hourly* heating and cooling degrees to get higher granularity on a metric which is relevant to energy consumption. I dropped a number of hourly columns with excessive null entries compared to the size of the data set and those with low predictive power. I dropped hourly pressure tendency and change because there are excess null values and because measures of pressure were already present in the data set, hourly wind direction and gust speed for the same reasons, and hourly present weather type because of excess null values and the dispersity of possible values, indicating low predictive power.

One of the main challenges in cleaning the LCD data was that there were in some cases multiple entries for the same hour. I wanted to have just one entry per hour such that I could eventually align LCD data with the hourly entries in the electricity data. I wrote a function that goes through the weather data and for each instance where there are multiple hourly entries, I keep the hourly entry closest to the hour and drop the remaining rows. For instance, if there are entries at 11:05, 11:36, and 11:56, I kept the entry corresponding to 11:05 and dropped the other rows. I performed the cleaning this way because the data closest to the hour is most representative for that hour. I then renamed the date such that the minute column is set to 00 to once again align with the electricity data. Either way, the values for multiple, per-hour entries are very similar, so the choice of which entry to keep doesn't make a huge difference.

The hourly sky condition feature required additional cleaning. The hourly sky condition feature contains information about each cloud layer (up to three) using six different categories to describe the cloud coverage. As stated in the documentation, the full state of the sky can be best determined the the category of the last (i.e., highest) layer. With this in mind, I replaced the sky condition feature with the condition of the upper most layer to make the feature consistent. This feature also included some information about the cloud layer height which I excluded because these data were only present for a small number of entries. There were also a number of features of mixed float and string types. After looking at their unique values, I saw that some values were string and floats arbitrarily. I simply converted strings to floats to make the features consistent. Finally, I convert our categorical value to numeric using Pandas' `get_dummies` function to be used for analyzing correlations among and performing learning.

Next we deal with null values and outliers. To determine whether to fill null values with median of `ffill`, I plotted histograms, violin plots, and printed stats for each feature. These plots can be found here. Features where the median was close the mean suggests few outliers and that the median is an appropriate statistic to fill null entries. Conversely, null values in features with outliers present were filled using `ffill`, assuming that previous values have some predictive value for subsequent values since we are dealing with time-stream data. For outliers, because there was not a single feature where the presence of outliers significantly shifted the median from the mean and because the data is such high granularity, outliers were kept without further cleaning. Lastly, we cut both data sets to fit the same date range as mentioned above, and combine the electricity and weather data into a single `DataFrame` on date index. A table of summary statistics for LA can be seen in **Figure 1**.

After cleaning, we are left with ~27K data points from July 2015 to September 2018 to perform statistical analysis and predictive modeling. The code for this process can be found here.

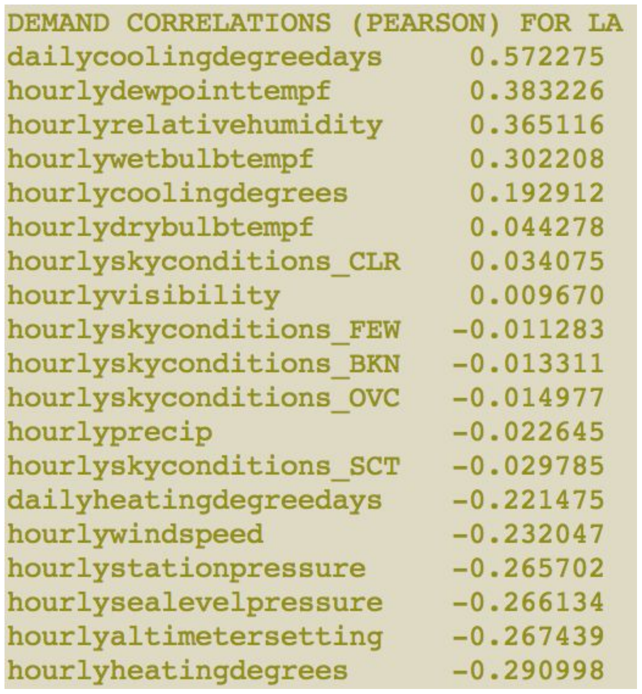


```
                               mean   median         std
demand                  3317.094400  3191.00  785.756576
hourlyrelativehumidity    62.482905    66.00   20.829641
hourlydewpointtempf       51.144188    54.00   11.886757
hourlydrybulbtempf        66.454703    66.00    9.638729
hourlywetbulbtempf        58.249122    59.00    7.749118
hourlyheatingdegrees       7.371355     6.00    6.932305
dailycoolingdegreedays     4.866420     2.00    5.595377
dailyheatingdegreedays     1.971355     0.00    3.554354
hourlywindspeed            1.623656     0.00    2.415391
hourlyvisibility           9.220403    10.00    1.796501
hourlycoolingdegrees       0.651968     0.00    1.617127
hourlyskyconditions_CLR    0.709222     1.00    0.454130
hourlyskyconditions_OVC    0.179856     0.00    0.384074
hourlyskyconditions_BKN    0.051155     0.00    0.220318
hourlyskyconditions_SCT    0.031196     0.00    0.173850
hourlyskyconditions_FEW    0.026649     0.00    0.161060
hourlyaltimetersetting    29.967976    29.96    0.110782
hourlystationpressure     29.772348    29.76    0.108847
hourlysealevelpressure    29.964036    29.95    0.106616
hourlyprecip               0.000552     0.00    0.007481
```

Figure 1. Summary statistics for all the features in the data set for LA. As discussed, since the median is not too far away from the mean for almost all of these features, filling null values with either the median of `ffill` is an appropriate procedure.

Exploratory Data Analysis

Here we perform some initial exploratory data analysis (EDA) for the weather/electricity data sets. One of the first questions we ask is what is the relationship between the independent variables (the weather features) and the dependent variable (electricity demand measured in MWh). Pandas has a convenient method (the .corr() method) for finding the Pearson correlation coefficients between all the different variables. The Pearson correlation coefficient “is a measure of the linear correlation between two variables” that varies from -1 to 1, indicating negative or positive correlations respectively. **Figure 2** shows the sorted Pearson correlation coefficients. Since electricity usage in LA is dominated by cooling in the summer months as shown in **Figure 3**, we see that the CDD feature has a strong positive correlation to electricity demand.



```
DEMAND CORRELATIONS (PEARSON) FOR LA
dailycoolingdegreedays        0.572275
hourlydewpointtempf           0.383226
hourlyrelativehumidity        0.365116
hourlywetbulbtempf            0.302208
hourlycoolingdegrees          0.192912
hourlydrybulbtempf            0.044278
hourlyskyconditions_CLR       0.034075
hourlyvisibility              0.009670
hourlyskyconditions_FEW      -0.011283
hourlyskyconditions_BKN      -0.013311
hourlyskyconditions_OVC      -0.014977
hourlyprecip                 -0.022645
hourlyskyconditions_SCT      -0.029785
dailyheatingdegreedays       -0.221475
hourlywindspeed              -0.232047
hourlystationpressure        -0.265702
hourlysealevelpressure       -0.266134
hourlyaltimetersetting       -0.267439
hourlyheatingdegrees         -0.290998
```

Figure 2. Pearson correlation coefficients

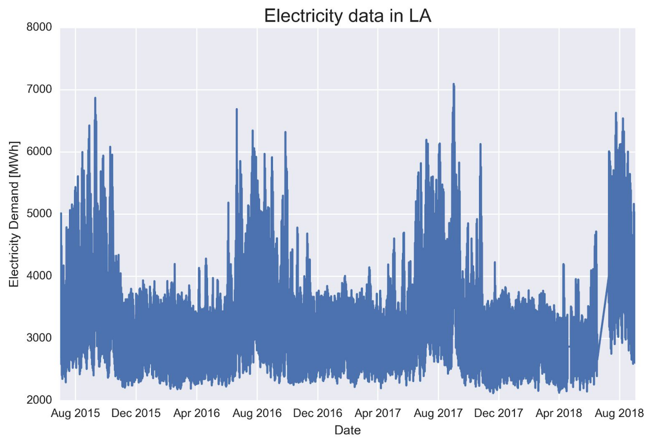
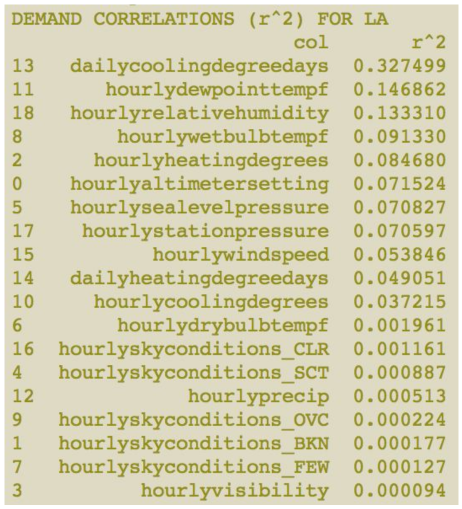


Figure 3. Electricity demand versus time for both LA. Demand peaks in the summer season in LA, thus explaining the strong correlation between demand and CDD.

Additionally, **Figure 4** shows computed r^2 values. r^2 signifies “the proportion of the variance in the dependent variable (i.e., electricity demand) that is predictable from the independent variables” (i.e., weather features). Once again, CDD is a significant feature for LA that captured over 30% of the variance in electricity demand.



```
DEMAND CORRELATIONS (r^2) FOR LA
                          col       r^2
13   dailycoolingdegreedays  0.327499
11      hourlydewpointtempf  0.146862
18   hourlyrelativehumidity  0.133310
8        hourlywetbulbtempf  0.091330
2      hourlyheatingdegrees  0.084680
0    hourlyaltimetersetting  0.071524
5    hourlysealevelpressure  0.070827
17    hourlystationpressure  0.070597
15          hourlywindspeed  0.053846
14   dailyheatingdegreedays  0.049051
10     hourlycoolingdegrees  0.037215
6        hourlydrybulbtempf  0.001961
16  hourlyskyconditions_CLR  0.001161
4   hourlyskyconditions_SCT  0.000887
12             hourlyprecip  0.000513
9   hourlyskyconditions_OVC  0.000224
1   hourlyskyconditions_BKN  0.000177
7   hourlyskyconditions_FEW  0.000127
3          hourlyvisibility  0.000094
```

Figure 4. r^2 values for LA.

In our data set, we have three features for pressure (hourly sea level, station pressure, and hourly altimeter setting) and three for temperature (hourly dew point, wet, and dry bulb temperature). Having multiple features for essentially the same quantity raises concerns over collinearity. **Figure 5** shows a scatter matrix for the different pressure features. As predicted, these features show high collinearity and as a result we keep the feature with the highest r^2 as referenced in **Figure 4**.

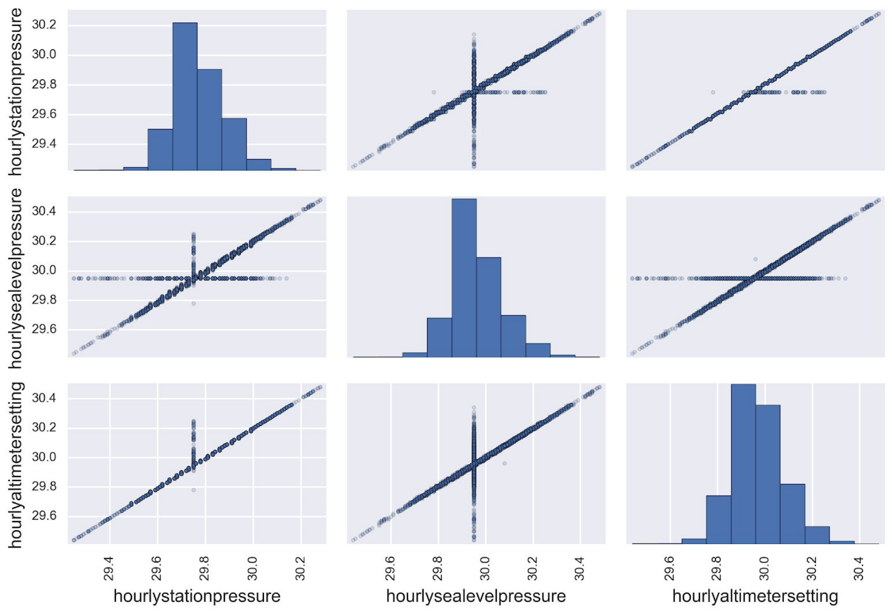


Figure 5. Scatter matrices for pressure features.

Collinearity is obvious in the scatter plots and the histograms also show very similar values.

Figure 6 repeats the above analysis but with the three temperature features. Collinearity is not as obvious but still strong. Since we derive hourly CDD and HDD from wet bulb temperature, we drop that column. Dry bulb temp has a low r^2 value, so we also drop that column, leaving us with dew point temperature as the main temperature feature.

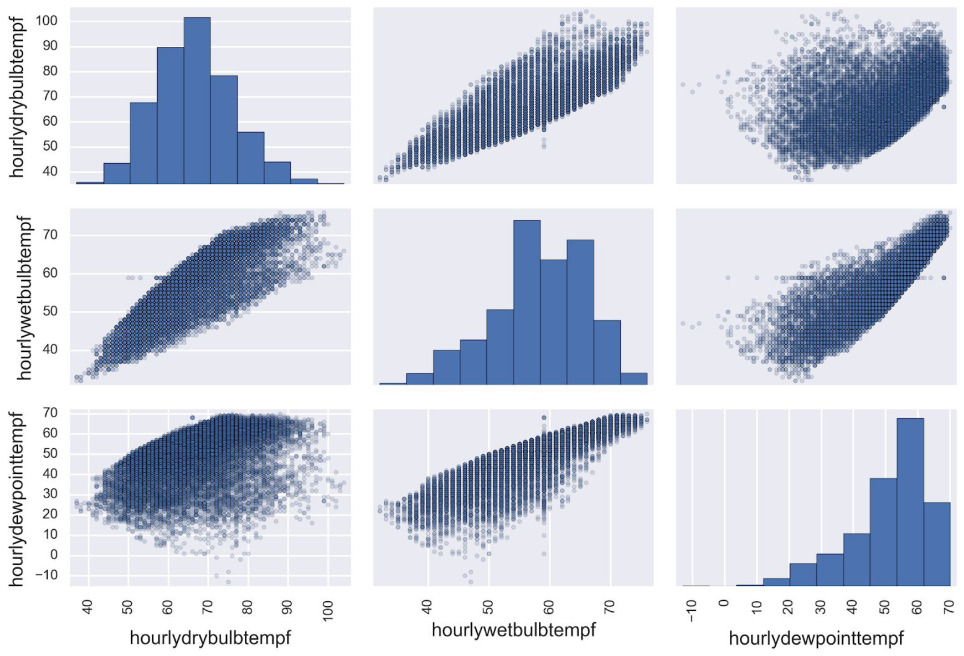
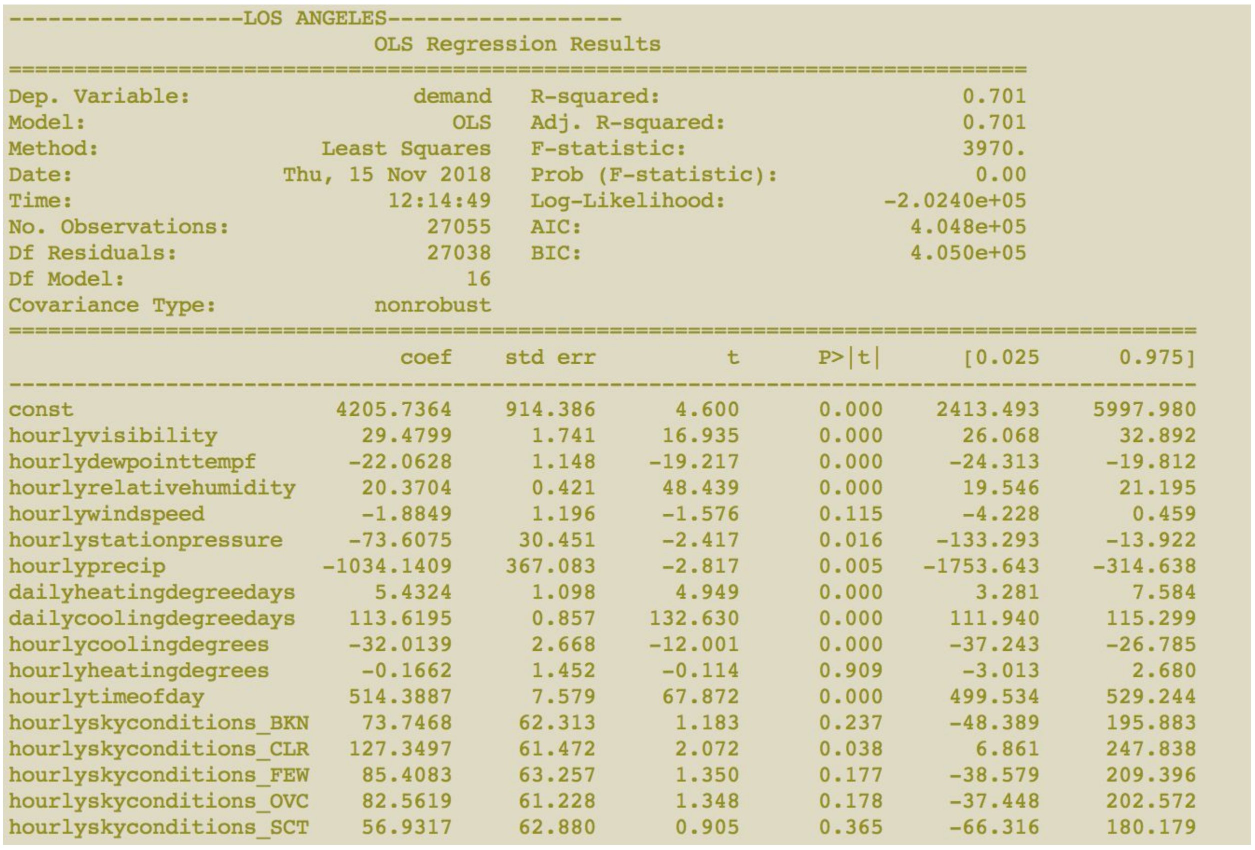


Figure 6. Scatter matrices for temperature features. Collinearity is seen between different temperature measurements.

Since people behave different in the daytime versus the nighttime regardless of how hot or cold it is that day, an ‘hourlytimeofday’ column was added, representing if the current time is day or night (zero between 6AM and 6PM, one otherwise). The temperature data is important for identifying heating and cooling peaks, but the regression can start leaning on the temperature as a proxy for daytime. This feature increased our multivariate regression r^2 considerably (from 0.686 to 0.701), signifying its importance.

Finally we perform a multiple ordinary least squares (OLS) regression on all the features versus electricity demand. **Figure 6** shows the results. High p-value features confirm our intuitions--that heating is not important for LA. Sky condition, i.e., the cloud coverage, and wind speed are intuitively not important features for predicting electricity demand. These less-useful features are indicated by high p-values and any feature with a p-value greater than 0.1 was dropped from the data set.



```
------------------LOS ANGELES------------------
                            OLS Regression Results
==============================================================================
Dep. Variable:                 demand   R-squared:                       0.701
Model:                            OLS   Adj. R-squared:                  0.701
Method:                 Least Squares   F-statistic:                     3970.
Date:                Thu, 15 Nov 2018   Prob (F-statistic):               0.00
Time:                        12:14:49   Log-Likelihood:            -2.0240e+05
No. Observations:               27055   AIC:                         4.048e+05
Df Residuals:                   27038   BIC:                         4.050e+05
Df Model:                          16
Covariance Type:            nonrobust
=============================================================================================
                              coef    std err          t      P>|t|      [0.025      0.975]
---------------------------------------------------------------------------------------------
const                    4205.7364    914.386      4.600      0.000    2413.493    5997.980
hourlyvisibility           29.4799      1.741     16.935      0.000      26.068      32.892
hourlydewpointtempf       -22.0628      1.148    -19.217      0.000     -24.313     -19.812
hourlyrelativehumidity     20.3704      0.421     48.439      0.000      19.546      21.195
hourlywindspeed            -1.8849      1.196     -1.576      0.115      -4.228       0.459
hourlystationpressure     -73.6075     30.451     -2.417      0.016    -133.293     -13.922
hourlyprecip            -1034.1409    367.083     -2.817      0.005   -1753.643    -314.638
dailyheatingdegreedays      5.4324      1.098      4.949      0.000       3.281       7.584
dailycoolingdegreedays    113.6195      0.857    132.630      0.000     111.940     115.299
hourlycoolingdegrees      -32.0139      2.668    -12.001      0.000     -37.243     -26.785
hourlyheatingdegrees       -0.1662      1.452     -0.114      0.909      -3.013       2.680
hourlytimeofday           514.3887      7.579     67.872      0.000     499.534     529.244
hourlyskyconditions_BKN    73.7468     62.313      1.183      0.237     -48.389     195.883
hourlyskyconditions_CLR   127.3497     61.472      2.072      0.038       6.861     247.838
hourlyskyconditions_FEW    85.4083     63.257      1.350      0.177     -38.579     209.396
hourlyskyconditions_OVC    82.5619     61.228      1.348      0.178     -37.448     202.572
hourlyskyconditions_SCT    56.9317     62.880      0.905      0.365     -66.316     180.179
```

Figure 7. Multivariate OLS regression for all the features + constant. Shown are a list of coefficients with errors, t-statistics, confidence intervals.

Machine Learning Modeling

Now we prepare our dataset for machine learning modeling. Our first step is to transform our time series data into a form suitable for supervised learning. Drawing heavily from this blog post, we transform our dataset such that we use *all* of the features (i.e., electricity demand plus all the weather features) from the *previous* timestep to predict electricity demand *at the current timestep.* Since electricity demand the hour before seems like a relevant proxy for the electricity demand at the current hour, this procedure seems intuitive. After reframing our data in this format, we split our data into a training and testing set and begin evaluating our machine learning models. To make the analysis consistent with our later utilization of recurrent neural networks and long short-term memory, we designate the first year of data as the training set and the rest of the data as the testing set (~32% training and ~68% testing). This split avoids overfitting and a more traditional 80/20 split on training and testing yields virtually the same model performance. The general process for modeling is: 1.) use a min-max scaler on all the features to make them more comparable, 2.) fit to the training set, and 3.) compute the relevant metrics on the testing set and compare all of the models at the end. In our analysis we perform no hyperparameter tuning for the models (default settings of sklearn) and still achieve quite remarkable results.

Our first model is simple OLS linear regression. OLS linear regression "chooses the parameters of a linear function of a set of explanatory variables by the principle of least squares: minimizing the sum of the squares of the differences between the observed dependent variable (values of the variable being predicted) in the given dataset and those predicted by the linear function." Put another way, OLS linear regression in our setting posits that electricity demand at the current timestep is determined by a linear combination of weather features and electricity demand at the previous timestep. To find the coefficients of the linear combination, OLS minimizes the sum of the square differences between our prediction for electricity demand (that uses electricity demand and weather data) and the true value of electricity demand. It's the simplest machine learning model and we performed OLS on our dataset prior to machine learning modeling, however the dataset was not framed as a supervised learning problem as described above. Using linear regression, **we achieve an r^2 of 0.962 and an root-mean-squared error (RMSE) of 150.1 MWh.**

Next we use a decision tree regressor as the model. Decision trees make predictions by continually creating splits on each of the features such that information gain is maximized. Splitting can cease according to a variety of criteria such as a maximum depth or node purity threshold. Using a decision tree, **we achieve an r^2 of 0.931 and an root-mean-squared error (RMSE) of 208.5 MWh.**

Next we use a k-nearest neighbors (k-NN) algorithm for our regression problem. k-NN makes predictions by "finding a predefined number of training samples closest in distance to the new point and to predict a label" from those distances. Using a k-NN, **we achieve an r^2 of 0.906 and an root-mean-squared error (RMSE) of 243.4 MWh.**

Now we use a random forest regressor. Random forest is an ensemble method, meaning it utilized a collection of predictors to make a strong prediction and minimize variance. Random forest fits a number of decision trees on a bootstrap subsample of the data; additionally, for each tree, each split is made using a randomized subset of features. Because of this randomness, the variance of the estimator goes down and we avoid many of the overfitting problems of the decision tree. Using a random forest regressor, **we achieve an r^2 of 0.963 and an root-mean-squared error (RMSE) of 153.7 MWh.**

Next we use a gradient boosting regressor. The gradient boosting regressor is another ensemble method where predictions are made sequentially (i.e., current predictions depend on previous ones). Gradient boosting "repetitively leverages patterns in the residuals [of the fit] to strengthen a model with weak predictors and improve performance." Using gradient boosting regressor, **we achieve an r^2 of 0.967 and an root-mean-squared error (RMSE) of 145.5 MWh.**

Next we use the bagging ensemble regressor. This regressor "fits base regressors each on random subsets of the original dataset and then aggregate their individual predictions (either by voting or by averaging) to form a final prediction." Whereas the gradient boosting regressor used a sequential technique to combine predictors, bagging combined many independent predictors in parallel to make a final prediction. Using gradient boosting regressor, **we achieve an r^2 of 0.967 and an root-mean-squared error (RMSE) of 145.5 MWh.**

Finally we use the AdaBoost regression model. AdaBoost is an ensemble method that works "by fitting a regressor on the original dataset and then fitting additional copies of the regressor on the same dataset but where the weights of instances are adjusted according to the error of the current prediction." Specifically, the weights of the instances are adjusted using an exponential scaling. Using an AdaBoost regressor, **we achieve an r^2 of 0.949 and an root-mean-squared error (RMSE) of 177.7 MWh.** The creation, training, and performance of the above models is documented in this IPython notebook.

Before summarizing all the results, we use one last predictive model but it is different than the techniques described above. Specifically, we use recurrent neural networks (RNNs) and long short-term memory (LSTM). A detailed walkthrough on the basics of RNNs and LSTM along with the modeling procedure can be found in this IPython notebook. Using RNNs with LSTM, **we achieve an r^2 of 0.961 and an root-mean-squared error (RMSE) of 156.8 MWh.** Additionally, we show the loss (in mean-absolute error) versus epoch for training the LSTM in **Figure 8.** The testing loss remains a bit higher than the training loss, indicating that we are not overfitting.

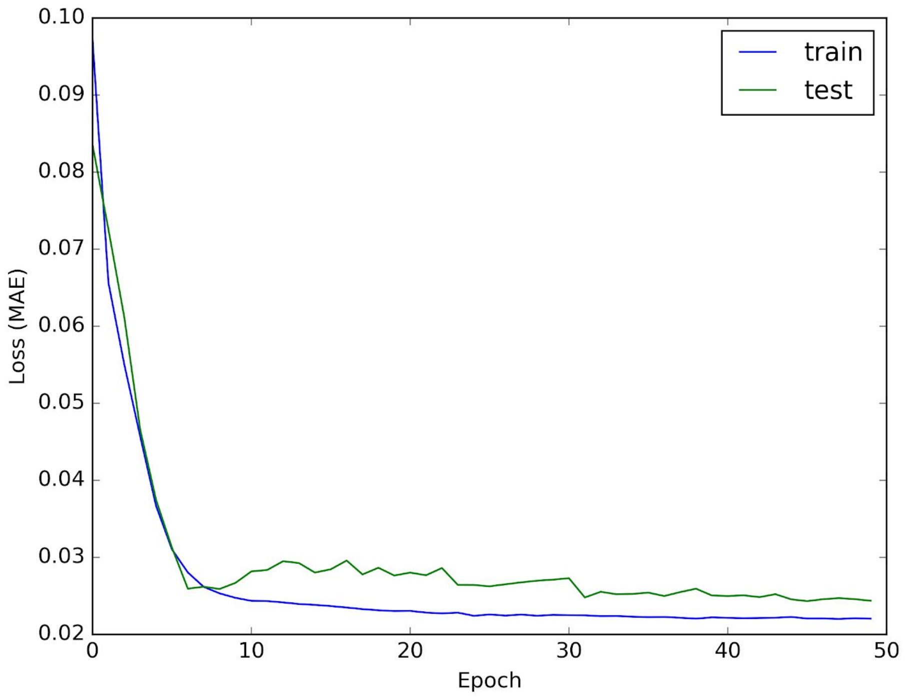


Figure 8. Loss measured in mean-absolute error (MAE) versus epoch for both training and testing of our LSTM network. Testing error remains slightly above training, indicating that we are not overfitting. The LSTM network was optimized using the Adam implementation of stochastic gradient descent with MAE loss.

EIA also provides a day-ahead electricity demand forecast that we would like to compare to our models. EIA uses various methods to forecast demand. The forecast depends heavily on weather forecasts and the day of the week. The procedure to measure the performance of EIA's forecast is found in the above IPython notebook. With EIA's forecast, **we compute an r^2 of 0.917 and an root-mean-squared error (RMSE) of 227.9 MWh, indicating that out of the box machine learning models indeed perform better than EIA's model.**

Results are summarized in **Table 1** sorted by r^2. In general, ensemble machine learning models perform better than others, although simple linear regression performs very well. The training and testing errors for both all models remain within the percent level of each other, indicating we are not substantially overfitting the data. Since cross-validation methods become trickier when dealing with correlated, timestream data, we leave out cross-validation in our analysis to keep all of our models comparable and consistent. We find that the gradient boosting model has the best performance, although the top five models are all similar in performance (differences of <1% in performance between them).

Model Name	**Root Mean Squared Error (RMSE) [MWh]**	r^2
Gradient Boosting	**145.5**	**0.9665**
Linear Regression	152.8	0.9630
Random Forest	153.7	0.9626
Bagging	153.9	0.9625
RNNs w/ LSTM	156.8	0.9610
AdaBoost	177.7	0.9499
Decision Tree	208.5	0.9312
EIA’s day-ahead forecast	227.9	0.9174
k-NN	243.4	0.9062

Table 1. Table of results of machine learning models. The best performing model (gradient boosting) is highlighted.

With our best-performing model chosen (gradient boosting), we now want to see which features are most important in predicting electricity demand. This is shown in **Figure 9.** Unsurprisingly, demand at the current previous hour is the most important feature for predicting demand at the current hour, but other proxies like heating degree days are also a strong predictor as we expected.

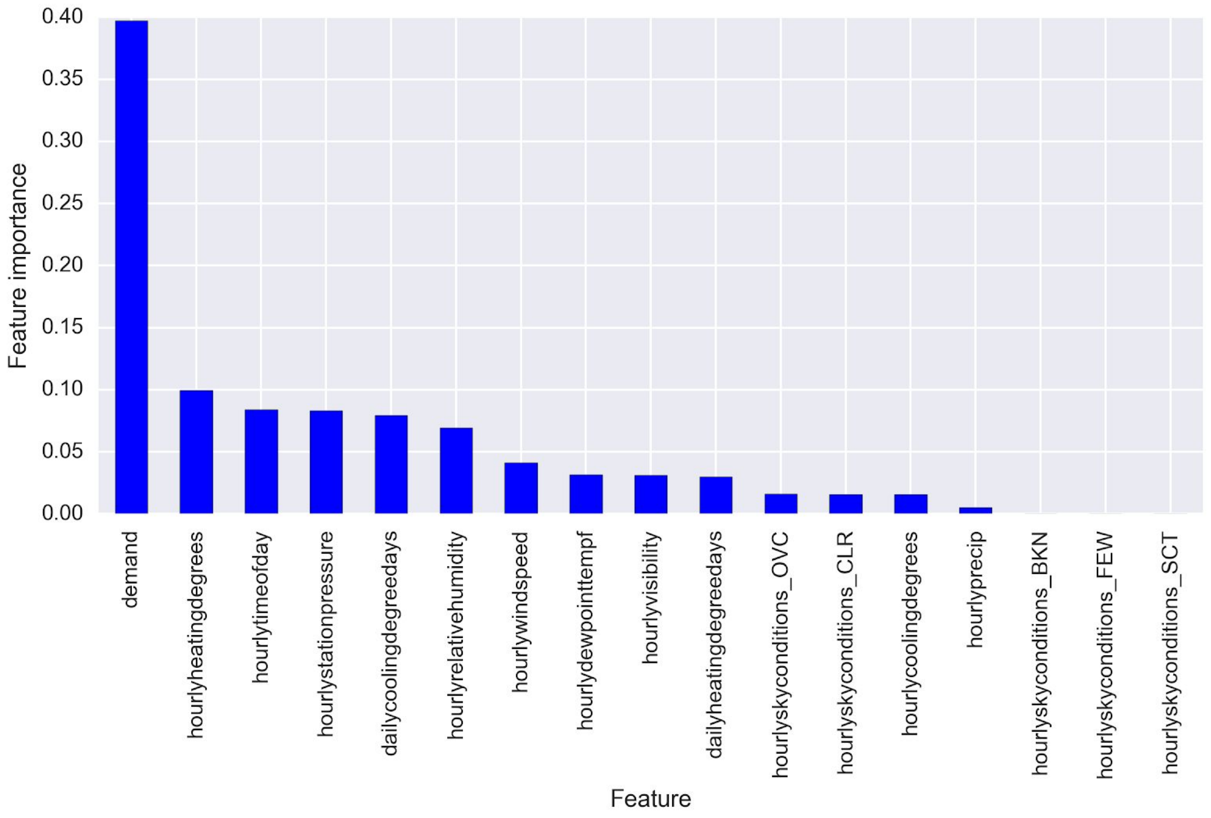


Figure 9. Feature importance for each feature in the gradient boosting model. Demand at the previous time step is four times more important than heating degree days, the next most important feature. Reframing our supervised learning problem in this way greatly improved the predictive power of the models.

Conclusion

We show that outperforming EIA's day-ahead electricity demand forecast is actually quite easy with machine learning. Remarkably all but one of our models (k-NN) has better performance than EIA's model. Relatively simple data scraping, cleaning, and machine-learning modeling from public data sources results in much more accurate modeling than traditional methods. **We report an overall ~5% increase in performance over EIA's model;** these models could then directly be used by energy companies to better brace for high-demand spikes in electricity and ultimately increase profits from more accurate predictions. Additional hyperparameter tuning could be performed on the models to increase performance, although the increase would most likely be marginal at best and since out models already outperform the EIA's, we conclude the report without tuning.