MIE465 | Analytics in Action | Final Report

Ontario Global Adjustment Demand Day Prediction

April 13th 2018

Trevor Waite	1000362528
Joshua Paras	1001309675
Mustafa Bengali	1002486785
Laith Barghouthi	1001250576

Abstract

Ontarian electricity users were split into Class A and B in 2012. Class A consumers (industrial and commercial) are those with electricity demands exceeding 1 MW. Global Adjustment (GA) is a component added to the monthly electricity bill to cover the costs of maintaining and introducing new electricity infrastructure in Ontario. The GA component for each class A customer is calculated based on their contribution percentage to the top 5 demand hours (Demand Days) every year. This is a form of economic incentive to reduce stress on the grid during high demand scenarios and can make up over 50% of a users monthly electricity bill. The goal of this project is provide a prediction model to a Class A commercial customer in Toronto (i.e First Canadian Place Tower, TD Centre etc.) that indicates when these Demand Days will occur. This will allow the customer to reduce consumption or switch to off grid power on these days therefore eliminating their GA contribution and substantially reducing their total annual energy costs.

Our model requires the use of Ontario's historical electricity demand data and weather temperature data. The demand data was taken from the provincial electricity system operators website (IESO). Weather temperature data was obtained from Kaggle. IESO consumption data was transformed by log and high-order differencing in order to remove stationarity. Historical weather data was transformed into Cooling Degree Days (CDD) and Heating Degree Days (HDD) sub datasets. These transformed sets were used to build 5 SARIMA models. From these models, the day ahead forecasts of demand min, mean, max, CDD and HDD were determined and inputted into a logistic regression model which classified whether or not the the following day was a top $\boldsymbol{\lambda}$ demand day, where $\boldsymbol{\lambda}$ was a tweakable parameter.

In order to benchmark our final results against a real business case, the team reached out to Cadillac Fairview (CF), a commercial real-estate company in downtown Toronto. CF's RBC Centre, Simcoe Place and the Ritz-Carlton currently uses a 3rd party consultant to predict demand days and instruct them when to switch to off grid power (diesel generators). In 2017, they were instructed to switch off grid a total of 32 times for a 4 hour time frame. In these 35 days, 4 days were top 5 peak demand days. CF reported a substantial amount of savings (dollar figure confidential) including the cost of the 3rd party consultant. Our SARIMA/Logistic regression combination model currently captures 4 of the top 5 demand days in 2017 if instructed to shut off for 17 days ($\boldsymbol{\lambda}$ = 17) for a 4 hour time frame. All 5 demand days can be encapsulated if this threshold is increased to 35 days ($\boldsymbol{\lambda}$ = 35) for the same 4 hour time frame. In conclusion, if our model was to be further developed into an online decision making tool for building operation managers, commercial office towers with off-grid power solutions could potentially cut their monthly electricity bill in half.

1.0 Introduction and Problem Statement

Monthly electricity bills in Ontario are derived from the consumer's hourly usage multiplied by the **HOEP,** (Hourly Ontario Energy Price) a dynamic price that changes with demand. In the early 2000s [1], **IESO** introduced **Global Adjustment** (GA), into the electricity bill. GA is essentially a way to help cover the costs of maintaining existing as well as implementing new electricity infrastructure. In 2011, Ontarian electricity users were split into two categories [2], Class A and Class B. Consumers that have a demand of more than 1MW have the option to be classified as Class A customers. Class A customers pay GA based on the percentage of electricity they use during the top 5 demand hours during a 12 month base period. These 5 demand hours have to be on different days (i.e if the top 2 hours in the base period happen to be on the same day, only the higher demand is counted), and therefore, a more common way to refer to these days is **Demand Days.**

The goal of this project is provide a prediction model to a Class A commercial customer in Toronto (i.e First Canadian Place Tower, TD Centre etc.) that indicates when these **Demand Days** will occur. GA for Class A users can make up over 50% of their monthly electricity bill [3] and therefore the benefit of this type of prediction would be that it would allow for these users to reduce consumption, shut off operations or switch to an off grid alternative on these days effectively reducing their energy costs dramatically.

2.0 Data Gathering and Analysis

2.1 Data Sources

In order to produce our predictive model, we needed Ontario electricity demand data as well as historical weather data. Demand data was obtained from the IESO power data directory [4]. This data contained hourly demand MW per day for various geographical zones across Ontario from 2002 to present day. These zones were summed to get the total demand across Ontario. Weather data was obtained from Kaggle[5] which included hourly temperature and humidity data from 2013 to November 2017 from various cities across North America including Toronto. The Toronto column was kept and all other cities were removed.

2.2 Exploratory Data Analysis

This section will walk through observations and correlations we have uncovered within our datasets. Through better understanding of data, we can more effectively construct and interpret our future models. All of the following graphs and analyses were produced in Python using the following libraries: Pandas, Matplot, SKlearn and Statsmodels.

- The highest peak demand over the year tends to be during summer and winter as shown in Figures 1 and 4.
- Figure 2 outlines the Top 5 Demand days hour of occurrence. All the peak hours tend to occur between 12-8pm. Knowing this, a shutdown strategy that determines a demand day would not need to last the whole day, but rather, this evening time frame.

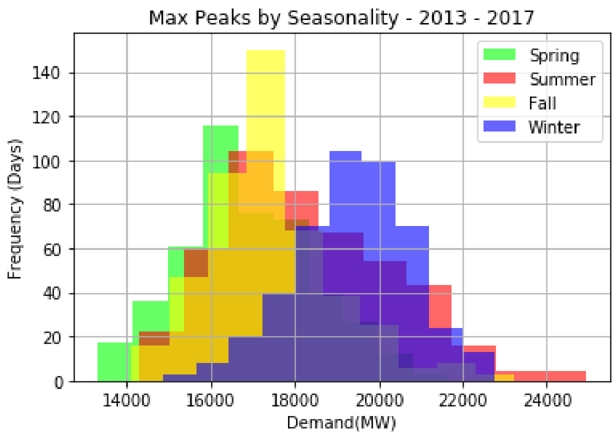


Figure 1: Peak Demand by Season

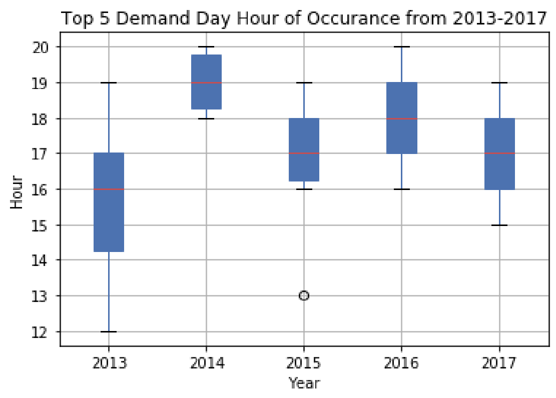


Figure 2: Hourly Range of Peak Demand

- As expected, there is a strong correlation between temperature and demand. Figure 3 depicts spikes in demand that occur during temperature extremes. Figure 4 depicts the correlation between demand and temperature, being negatively correlated for colder temperatures (R^2 = -0.74 below 14°C) and positively correlated for warmer temperatures (R^2 = 0.67 above 14°C).

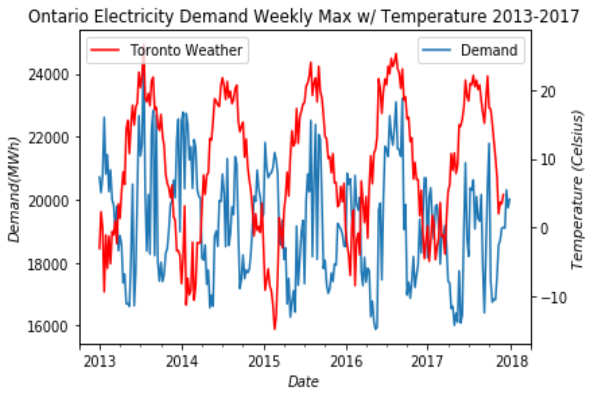


Figure 3: Demand & Temperature

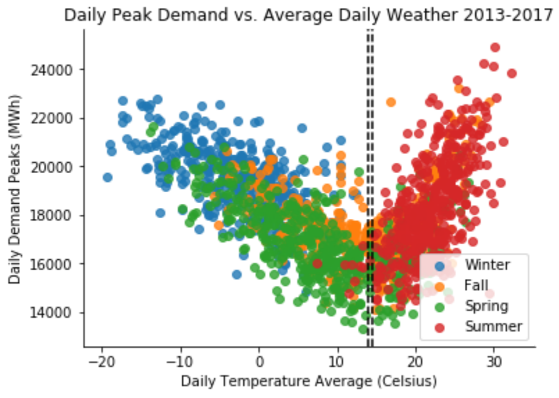


Figure 4: Demand Peak and Temperature

- Due to the strong relationship between weather and demand, the weather data was broken down into Heating Degree Days and Cooling Degree Days (Appendix A Weather Temperature to HDD/CDD code)
- HDD/CDD indicate how many degrees the temperature is above the temperature sensitive usage. In our case this balance point temperature is 14°C.
- The load profile of a demand day greatly differs from that of a regular day. The typical day, as seen in Appendix Figure 1, displays a small ramp up around 7-8am, that sustains at the level reached or gradually increases until 5-8pm where we see a daily peak. Evaluating the top five demand day profiles (Appendix Figure 2), we see similar behaviour except for the difference in magnitude (peak, min and average) along with the fact they tend to ramp up faster, and continue ramping up until a peak is met around 2-4pm creating a more round, hill-like shape.

3.0 Methodology

Our prediction model methodology will use the results of a day-ahead forecast model as the inputs to a classification model. The first model will predict future values of daily features, while the second model will determine which of those future days will be a Demand Day.

3.1 Prediction System

All data undergoes a form of transformation followed by isolating it into training and testing sets and finally inputting it in a model. The detailed system and process is explained below and presented in Figure 8. Consumption data is first collected and transformed using both log and seasonal differencing. A training set is then inputted into a SARIMA (Seasonal Autoregressive Integrated Moving Average) model. Temperature data is extracted from historical weather data, which is then transformed to Heating Degree Days and Cooling Degree Days (HDD & CDD). A training set is then inputted into another SARIMA model. A logistic regression model is trained with the historical demand and CDD/HDD data. Forecasted data from the SARIMA models is then inputted into a classification regression model, which ultimately outputs whether the following day is a demand day (1) or not (0).

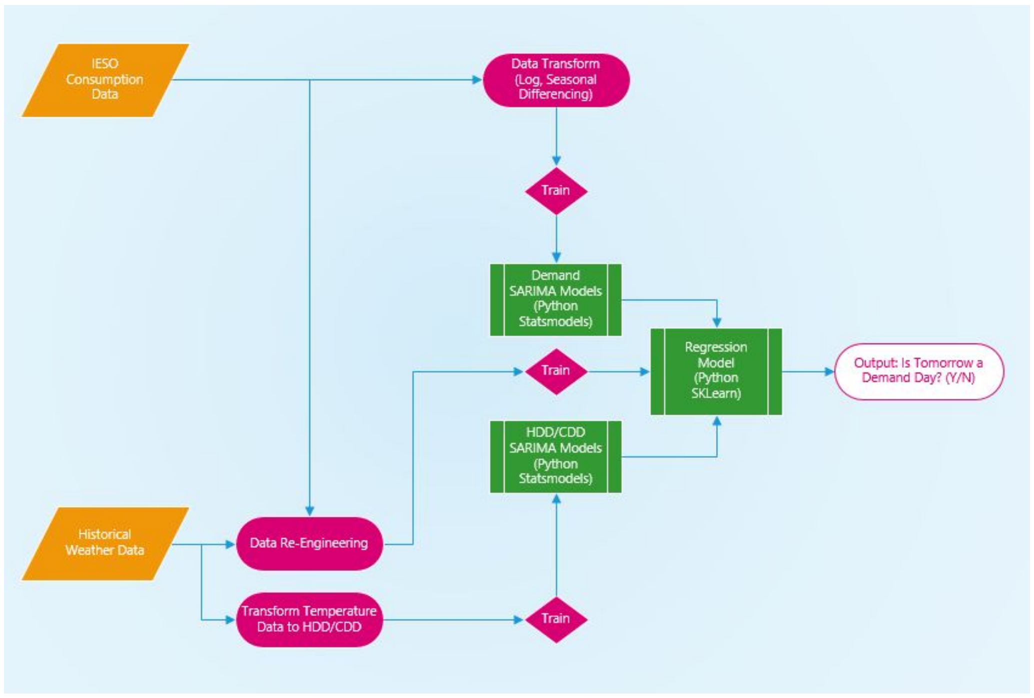

IESO Consumption Data
Data Transform (Log, Seasonal Differencing)
Train
Demand SARIMA Models (Python Statsmodels)
Train
Regression Model (Python SKLearn)
Output: Is Tomorrow a Demand Day? (Y/N)
HDD/CDD SARIMA Models (Python Statsmodels)
Historical Weather Data
Data Re-Engineering
Transform Temperature Data to HDD/CDD
Train

Figure 8: Prediction System Flow Chart

3.2 SARIMA Forecasting Model

A SARIMA (Seasonal Autoregressive Integrated Moving Average) model is a popular time-series forecasting tool used in many industries such as finance, energy and economics. In the SARIMA model, the estimated value of a variable is simply its linear combination of the past values (referred to as lags) and the past errors. The AR part of ARIMA indicates that the changing variable of interest is regressed on its own lag. The MA portion indicates that the regression error is actually a linear combination of the error terms whose values occurred at the same time in the past. The I illustrates that the data has been replaced with the difference between their values and the previous values (differencing). The general form of the non seasonal ARIMA is as follows:

$$\nabla^d Y_t = \mu + \phi_1 \nabla^d Y_{t-1} + \phi_2 \nabla^d Y_{t-2} + \cdots + \phi_p \nabla^d Y_{t-p} + U_t + \theta_1 U_{t-1} + \theta_2 U_{t-2} + \cdots + \theta_q Y_{t-q} \quad (1)$$

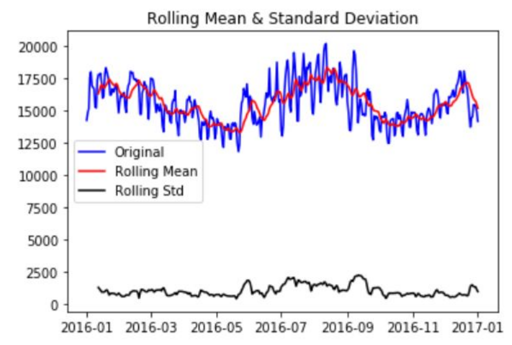
Or generally,

$$\phi(L)(1-L)^d Y_t = c + \theta(L)U_t \quad (2)$$

Where the difference operator is ∇, Φ and $\boldsymbol{\theta}$ are the coefficients of the autoregressive component and moving average respectively. Y_t and U_t are the actual values and white noise at time t, respectively. The numbers p,d,q determine the order of the ARIMA model where p is the autoregressive, d is the degree of differencing and q is the order of the moving average. The SARIMA is very similar to the ARIMA however is incorporates seasonal parameters, important considering the seasonality of our energy data. The multiplicative seasonal ARIMA model is represented by ARIMA(p,d,q)x$(P,D,Q)_k$ where the P,D,Q represent the seasonal AR, I and MA terms and k represents the number of seasons. This can be written as the following:

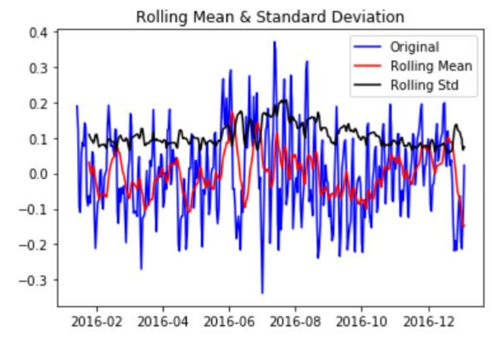
$$\phi_p(L)\Phi_P(L^k)(1-L)^d(1-L^k)^D Y_t = \theta_q(L)\Theta_Q(L^k)U_t \quad (3)$$

An important assumption of the SARIMA model, is that data is stationary meaning that it has a constant mean, variance and autocorrelation structure over time. Initial testing using the popular stationarity test Dickey-Fuller (See Appendix B for python code) suggested that our original demand data was not stationary as indicated in Figure 9. The mean and variance were not constant and the auto and partial correlated feature had correlations past 3 lags. In order to remediate this, high order differencing (order 12) was applied to remove autocorrelation and log transform was applied to reduce mean and variability. The Dickey-Fuller test was rerun which rejected the null hypothesis ($p < 0.1$) indicating that the data was now stationary and ready for SARIMA modeling (Figure 10).



```
Results of Dickey-Fuller Test:
Test Statistic                  -1.814023
p-value                          0.373522
#Lags Used                      17.000000
Number of Observations Used    349.000000
Critical Value (1%)             -3.449227
Critical Value (5%)             -2.869857
Critical Value (10%)            -2.571201
```

Figure 9: Test of Original Data



```
Results of Dickey-Fuller Test:
Test Statistic                  -2.638034
p-value                          0.085388
#Lags Used                      17.000000
Number of Observations Used    339.000000
Critical Value (1%)             -3.449788
Critical Value (5%)             -2.870104
Critical Value (10%)            -2.571332
```

Figure 10: Test of Transformed Data

In order to reduce overfitting, the transformed data was split into 365 training and testing sets for five separate SARIMA models consisting of the min, mean and max of demand as well as the HDD and CDD of weather data (See Appendix C and D for python code). A python function was built which took 2013-2016 data to train a model, then test the day ahead model forecast against actual January 1st 2017. This process would reiterate, retraining including January 1st 2017 and testing against January 2nd. The reason that small day ahead prediction were favour over longer 2,3,4 and 5 day forecasts were due to the nature of the SARIMA model. Week ahead forecasts were experimented with using 52 testing and training sets, however, this strategy resulted in extremely poor performance as SARIMA is only strong when predicting small spans into the future.

3.3 Classification Model

3.3.1 Data Re-engineering

As with the SARIMA model, the hourly weather data was converted into daily HDD and CDD, and the demand data was converted into daily maximum, mean, and minimum demand. Using this historical data, a new binary data column was created indicating whether that day is a top demand day or not (1 or 0). Several sets were created each predicting a different number of top days (varying $\boldsymbol{\lambda}$) in order to train the models.

3.3.2 CART/Random Forest

An initial model was built in MATLAB using Classification and Regression Trees(CART). Two sets of demand data were used: one for summer and one for winter. The data was randomly split into testing and training sets, and over 100 runs both sets of data yielded a prediction accuracy of approximately 77% (Around 77% of the predicted top demand days were actually top demand days). This number does not take in to account the number of incorrect demand days predicted by the model, which was high due to CART's tendency for overfitting. This method was not used as part of our final model as the results were not nearly as accurate as the logistic regression model. Therefore, due to time constraints, excessive effort was not spent exploring the model and fine tuning it to obtain better results. The code used is given in the Appendix E for reference.

3.3.3 Logistic Regression

A logistic regression model was chosen as our classifier due to the binary nature of the problem. Python's SciKit learn library was used for its LogisticRegression, Cross_validation (10 folds), and Confusion_matrix functions. The forecasted data from the SARIMA model was put into this model to make several sets, predicting a range of top days from 2 to 56 in order to compare it against the true historical 5 demand days of 2017 using the confusion matrix. The proportion of True Positives (the number of historical demand days predicted) were used to measure model success as accuracy would be skewed since the vast majority of days will not be historical demand days. Screenshots of the Python code that does this can be found in Appendix F.

4.0 Experimental Results

4.1 SARIMA Results

In order to find the optimal parameters for the SARIMA, the SciPy optimize brute function was utilized (See Appendix G for python code) in order to do a grid search over all possible p, d, q, P, D, Q parameter combinations from 0 to 1 in order to minimize the average MAPE (Mean Absolute Percentage Error) and RMSE (Root Mean Squared Error) across all 334 folds (weather data only available to November 31st for 2017), focusing on performance in summer months. This was done for a seasonality value k of 4 (Spring, Summer, Fall, Winter) and 12 (Monthly), however, a k of 12 greatly surpassed 4 in terms of model error. The best performing parameters and their corresponding average MAPE and RMSE across all folds are illustrated below as well as some example actual vs. forecasted plots.

Table 1: SARIMA Results

Model	Optimal SARIMA Parameters	Average MAPE	Average RMSE
CDD	ARIMA $(1,1,0)x(0,0,0)_{12}$	2.15%	0.74°C
HDD	ARIMA $(1,1,0)x(0,0,0)_{12}$	2.03%	1.55°C
DemMax	ARIMA $(1,1,1)x(0,1,1)_{12}$	3.8%	818.02 MW
DemMean	ARIMA $(1,0,1)x(0,1,1)_{12}$	4.6%	713.64 MW
DemMin	ARIMA $(1,0,1)x(0,1,1)_{12}$	3.6%	489.39 MW

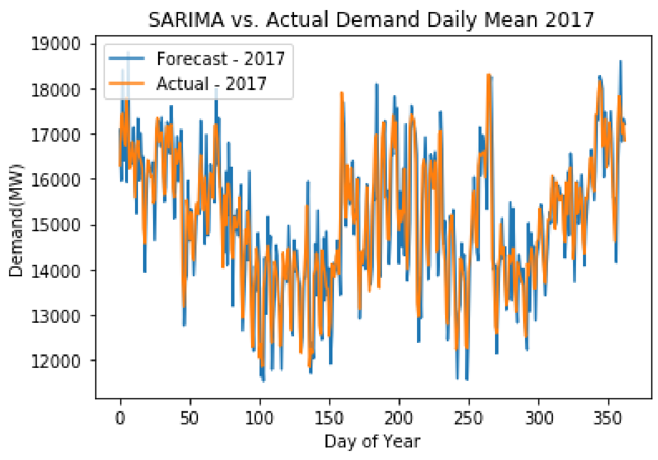


Figure 11: Mean SARIMA vs. Actual

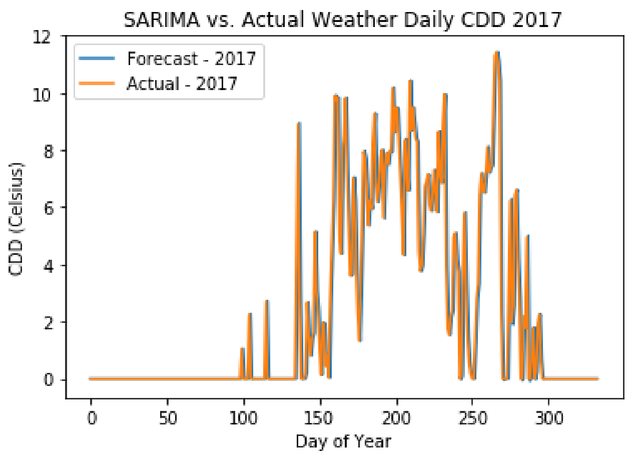


Figure 12: CDD SARIMA vs. Actual

4.2 Historical Logistic Regression Results

A baseline was created by using the historical 2017 data as the model input. At 12 days predicted there was a True Positive proportion of 80%, and at 26 days 100% demand day coverage (Figure 13). Table 2 shows the results for the model at 12 days predicted. All of the p values obtained from a chi squared test are below 0.05, so they are all statistically significant. The coefficients for CDD and HDD are noticeably larger than the ones for demand, this is due to the actual values of demand being much larger than the values for CDD and HDD.

Table 2: Historical regression results at 12 days predicted

Independent Variable	Coefficient	P-value
Intercept	-0.0793726	-
CDD	1.37195	5.31069e-39
HDD	1.06157	0.0041704

DemMax	0.000963764	0
DemMean	0.00347216	0
DemMin	-0.00709305	2.66316e-135

4.3 Total Results

Overall, the results were very exciting. It achieved 80% coverage when 17 days were predicted, and 100% coverage at 35 predicted (Figure 13). This means that when the model was asked to predict the 35 top demand days, it got them all correct. The following table contains the model coefficients and p values at 17 days. As all coefficient P-values are below 0.05, they are all statistically significant

Table 3: Forecasted regression results at 17 days predicted

Independent Variable	**Coefficient**	**P-value**
Intercept	-0.0390244	-
CDD	0.63987	4.15989e-41
HDD	0.371126	0.0408934
DemMax	0.000162216	0
DemMean	0.000752686	0
DemMin	-0.00174632	0

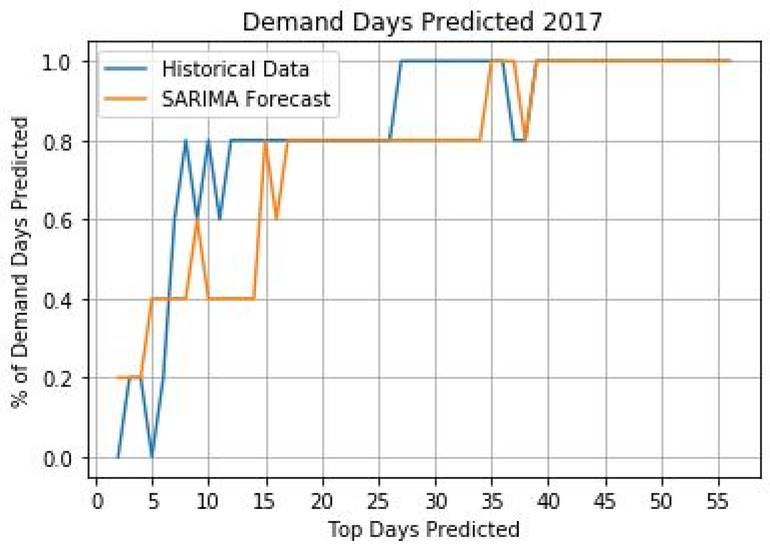


Figure 13: Overall System Performance vs. Historical Data Performance

5.0 Conclusions

The main goal of this project was to build a model in which demand days could be reasonably predicted such that a Class A commercial office building could reduce consumption and save on Global Adjustment costs. Using a systematic approach where day-ahead forecasted demand (SARIMA) and weather values were fed into a logistic classification model we were able to achieve outstanding results. The current model predicts 4 out of 5 top demand days for 17 days predicted and 5 of 5 for 35 days predicted. In order to benchmark and put these results into perspective, the team reached out to an industry professional. Cadillac Fairview's Operations Manager for RBC Centre, Simcoe Place and The Ritz Carlton reported in 2017 his energy consultants model able to predict 4 of 5 demand days for 32 days. Our model does quite well in comparison and if further tuned could be extremely competitive in the market.

6.0 Future Work

While the SARIMA models performed really well, another model shown to perform comparatively well if not outperform SARIMA is an artificial neural network (ANN). Several academic documents ventured into comparing the performances of SARIMA and ANN models and only found that ANN outperformed SARIMA by a small margin. Though this might be a useful and advantageous approach for the future, it might not have a significant impact on our findings. [6]

Another option would be forecasting demand days using Facebook Prophet which works well with a large number of outliers and holidays or big events during a year. It proves to be easier to use in the sense that it detects seasonality in data without the need for further configuration as with a SARIMA model. [7]

For the future, we believe our model can serve people in the industry to help them optimize their shutoff or energy reduction schedules. For that reason, developing a web decision making tool for building operation managers is a plausible business proposition. As more users sign up for the service and share their individual energy consumption data, this could be fed back into the prediction model therefore increasing overall accuracy.

7.0 References

[1] "Understanding Global Adjustment", IESO, 2018.

[2] "Global Adjustment in Ontario", Sygration, 2018.

[3] B. Hill, "What is the Global Adjustment fee? The mysterious cost Ontario hydro customers must pay", Global News, 2018. [Online]. Available: https://globalnews.ca/news/2839995/what-is-the-global-adjustment-fee-the-mysterious-cost-ontario-hydro-customers-must-pay/. [Accessed: 09- Mar- 2018].

[4] "Data Directory - Zonal Demands, 2003-2017", IESO, 2018. [Online]. Available: http://www.ieso.ca/en/power-data/data-directory. [Accessed: 09- Mar- 2018].

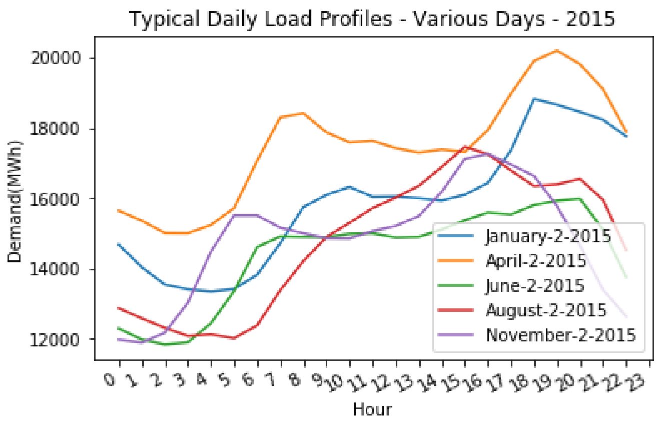
[5] "Historical Hourly Weather Data 2012-2017", Kaggle.com, 2018. [Online]. Available: https://www.kaggle.com/selfishgene/historical-hourly-weather-data/data. [Accessed: 09- Mar- 2018].

[6] A. Camara, W. Feixing and L. Xiuqin, "Energy Consumption Forecasting Using Seasonal ARIMA with Artificial Neural Networks Models", International Journal of Business and Management, vol. 11, no. 5, p. 231, 2016.

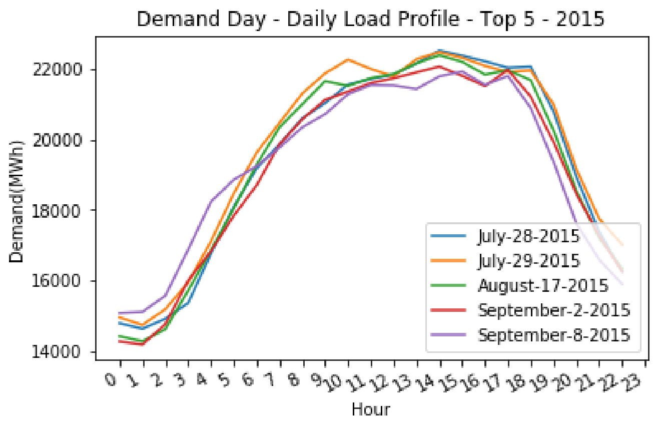
[7] Is Prophet Really Better than ARIMA for Forecasting Time Series Data?", learn data science, 2018. [Online]. Available: https://blog.exploratory.io/is-prophet-better-than-arima-for-forecasting-time-series-fa9ae08a5851. [Accessed: 12- Apr- 2018].

8.0 Appendix

HOEP → Hourly Ontario Energy Price: The price the consumer pays for electricity at any given hour(varies from hour to hour)
IESO → Independent Electricity Systems Operator: They control the energy marketplace in Ontario, are in charge of selling electricity at wholesale prices directly to class A consumers.
Global Adjustment → HOEP is only one part of the total commodity cost for electricity. The global adjustment (GA) is the component that covers the cost of building new electricity infrastructure in the province, maintaining existing resources, as well as providing conservation and demand management programs.
Demand Days → The 5 days in which the peak demand hours are reached, if a demand day has more than one peak hour, the highest demand is used



Appendix Figure 1: Typical Daily Load Profiles



Appendix Figure 2: Peak Demand Day Profiles

A: Temperature Data to HDD/CDD Code

```
#weather normalize data

#importanting weather data
temp = pd.read_csv("temperature.csv", usecols = [0,26], parse_dates=["datetime"], index_col="datetime")
#convert to degrees celsius
temp = temp - 273.15


#calcualte HDD/CDD
talpha = 14 #temp where correlation between temperature and demand inverts
tbeta = 14

temp['CDD'] = (((temp['Toronto'].resample('D').max()+temp['Toronto'].resample('D').min())/2-talpha))
temp['HDD'] = ((tbeta-(temp['Toronto'].resample('D').max()+temp['Toronto'].resample('D').min())/2))
temp.CDD[temp.CDD < 0] = 0
temp.HDD[temp.HDD < 0] = 0

print(temp['2012-10-02':].resample('D').mean())
```

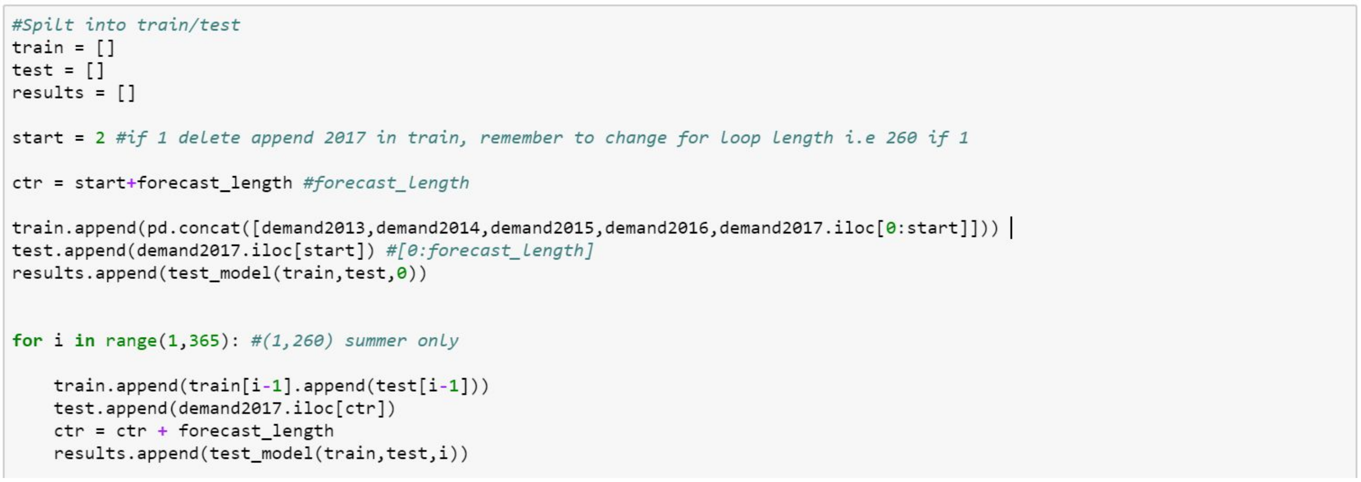
B: Dickey-Fuller Test Code

```python
from statsmodels.tsa.stattools import adfuller
def test_stationarity(timeseries):

    #Determing rolling statistics
    rolmean = pd.rolling_mean(timeseries, window=12)
    rolstd = pd.rolling_std(timeseries, window=12)

    #Plot rolling statistics:
    orig = plt.plot(timeseries, color='blue',label='Original')
    mean = plt.plot(rolmean, color='red', label='Rolling Mean')
    std = plt.plot(rolstd, color='black', label = 'Rolling Std')
    plt.legend(loc='best')
    plt.title('Rolling Mean & Standard Deviation')
    plt.show(block=False)

    #Perform Dickey-Fuller test:
    print ('Results of Dickey-Fuller Test:')
    dftest = adfuller(timeseries, autolag='AIC')
    dfoutput = pd.Series(dftest[0:4], index=['Test Statistic','p-value','#Lags Used','Number of Observations Used'])
    for key,value in dftest[4].items():
        dfoutput['Critical Value (%s)'%key] = value
    print (dfoutput)
```

C: SARIMA Transformation and Model Fitting Code (Demand Max)

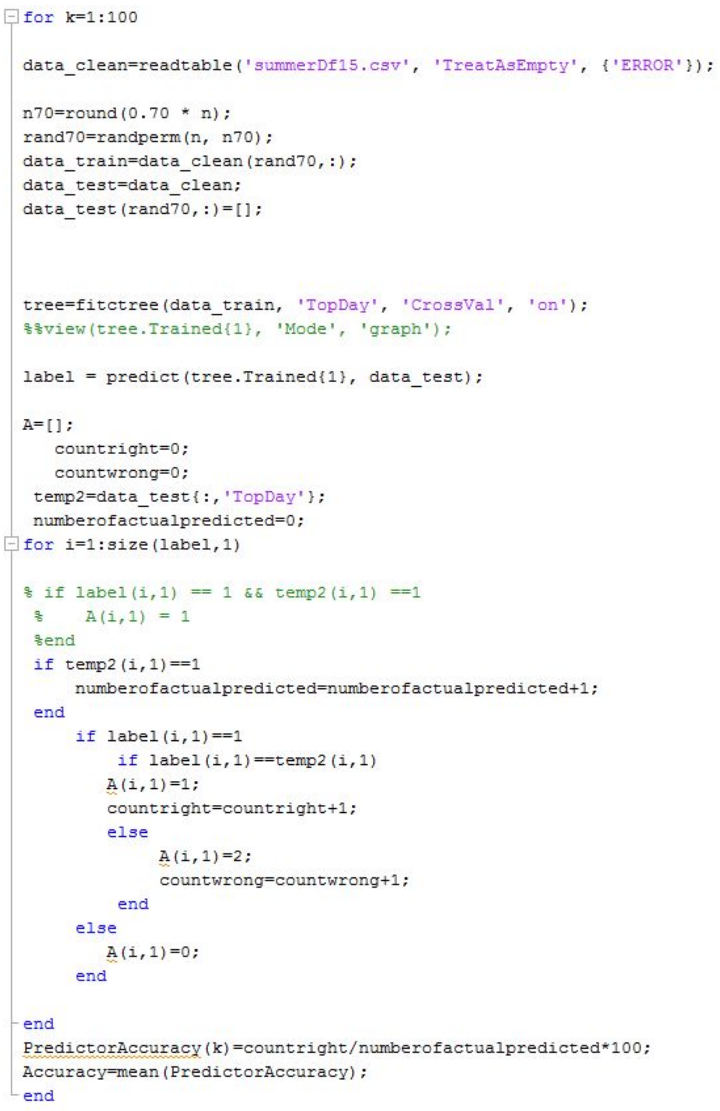
```python
#build models and validate
def test_model (train, test, i):

    #convert by log
    dTot_train_log = np.log(train[i]) # change

    #seasonal differencing
    seasons = 12
    dTot_train_log_ewma = dTot_train_log - dTot_train_log.shift(seasons)
    |
    #fit model
    mod = sm.tsa.statespace.SARIMAX(dTot_train_log_ewma.resample('D'),
                                    order=(1, 1, 1),
                                    seasonal_order=(0, 1, 1, seasons),
                                    enforce_stationarity=False,
                                    enforce_invertibility=False)
    results = mod.fit()


    #forecast
    forecast_log_diff = pd.Series(results.get_forecast(steps = forecast_length).predicted_mean, copy = True)
    forecast_log_diff.index  = test[i].index #.resample('D').mean()

    #Remove differencing
    forecast_log =  forecast_log_diff + dTot_train_log.shift(seasons).iloc[len(train[i])-1] #try mean?

    #Remove log
    forecast =  pd.Series(np.exp(forecast_log), index=forecast_log.index)

    return forecast
```

D: SARIMA Train/Test Code



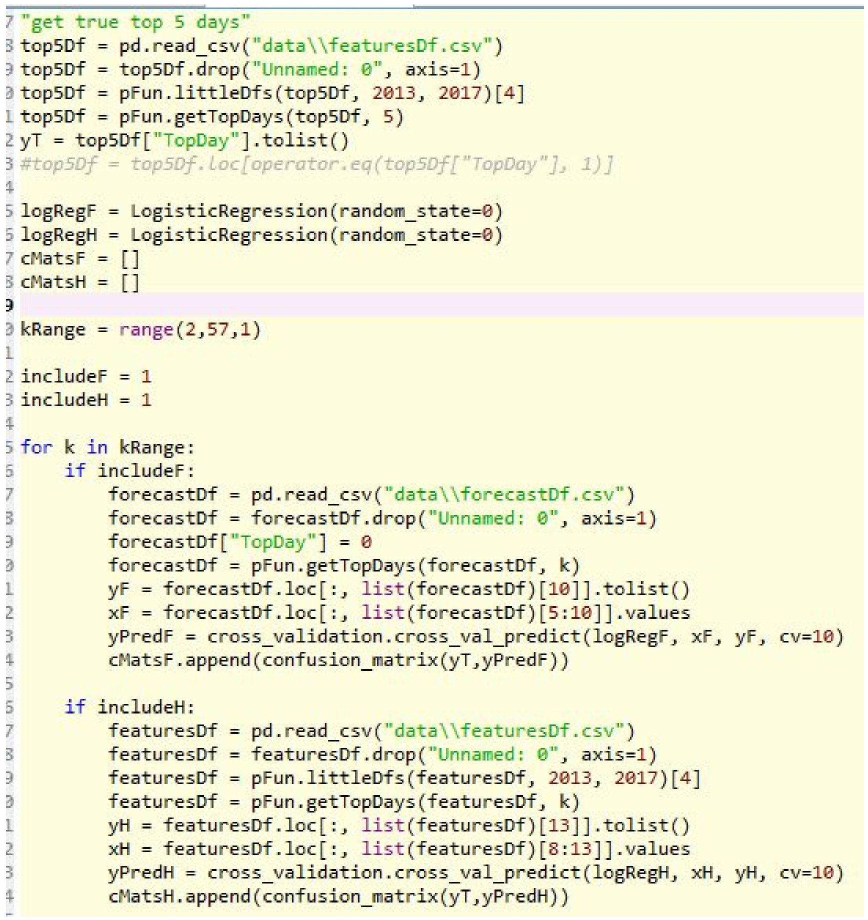
```
#Spilt into train/test
train = []
test = []
results = []

start = 2 #if 1 delete append 2017 in train, remember to change for loop length i.e 260 if 1

ctr = start+forecast_length #forecast_length

train.append(pd.concat([demand2013,demand2014,demand2015,demand2016,demand2017.iloc[0:start]]))
test.append(demand2017.iloc[start]) #[0:forecast_length]
results.append(test_model(train,test,0))


for i in range(1,365): #(1,260) summer only

    train.append(train[i-1].append(test[i-1]))
    test.append(demand2017.iloc[ctr])
    ctr = ctr + forecast_length
    results.append(test_model(train,test,i))
```

E: CART Code (Summer)



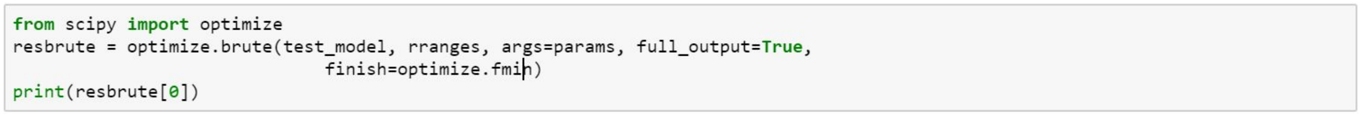
```
for k=1:100

data_clean=readtable('summerDf15.csv', 'TreatAsEmpty', {'ERROR'});

n70=round(0.70 * n);
rand70=randperm(n, n70);
data_train=data_clean(rand70,:);
data_test=data_clean;
data_test(rand70,:)=[];



tree=fitctree(data_train, 'TopDay', 'CrossVal', 'on');
%%view(tree.Trained{1}, 'Mode', 'graph');

label = predict(tree.Trained{1}, data_test);

A=[];
   countright=0;
   countwrong=0;
 temp2=data_test{:,'TopDay'};
 numberofactualpredicted=0;
for i=1:size(label,1)

% if label(i,1) == 1 && temp2(i,1) ==1
 %    A(i,1) = 1
 %end
 if temp2(i,1)==1
     numberofactualpredicted=numberofactualpredicted+1;
 end
     if label(i,1)==1
         if label(i,1)==temp2(i,1)
        A(i,1)=1;
        countright=countright+1;
        else
             A(i,1)=2;
             countwrong=countwrong+1;
         end
     else
        A(i,1)=0;
     end

end
PredictorAccuracy(k)=countright/numberofactualpredicted*100;
Accuracy=mean(PredictorAccuracy);
end
```

F: Logistic Regression Code



```
"get true top 5 days"
top5Df = pd.read_csv("data\\featuresDf.csv")
top5Df = top5Df.drop("Unnamed: 0", axis=1)
top5Df = pFun.littleDfs(top5Df, 2013, 2017)[4]
top5Df = pFun.getTopDays(top5Df, 5)
yT = top5Df["TopDay"].tolist()
#top5Df = top5Df.loc[operator.eq(top5Df["TopDay"], 1)]

logRegF = LogisticRegression(random_state=0)
logRegH = LogisticRegression(random_state=0)
cMatsF = []
cMatsH = []

kRange = range(2,57,1)

includeF = 1
includeH = 1

for k in kRange:
    if includeF:
        forecastDf = pd.read_csv("data\\forecastDf.csv")
        forecastDf = forecastDf.drop("Unnamed: 0", axis=1)
        forecastDf["TopDay"] = 0
        forecastDf = pFun.getTopDays(forecastDf, k)
        yF = forecastDf.loc[:, list(forecastDf)[10]].tolist()
        xF = forecastDf.loc[:, list(forecastDf)[5:10]].values
        yPredF = cross_validation.cross_val_predict(logRegF, xF, yF, cv=10)
        cMatsF.append(confusion_matrix(yT,yPredF))

    if includeH:
        featuresDf = pd.read_csv("data\\featuresDf.csv")
        featuresDf = featuresDf.drop("Unnamed: 0", axis=1)
        featuresDf = pFun.littleDfs(featuresDf, 2013, 2017)[4]
        featuresDf = pFun.getTopDays(featuresDf, k)
        yH = featuresDf.loc[:, list(featuresDf)[13]].tolist()
        xH = featuresDf.loc[:, list(featuresDf)[8:13]].values
        yPredH = cross_validation.cross_val_predict(logRegH, xH, yH, cv=10)
        cMatsH.append(confusion_matrix(yT,yPredH))
```

G: SARIMA Parameter Optimization Code (SciPy Brute)



```
from scipy import optimize
resbrute = optimize.brute(test_model, rranges, args=params, full_output=True,
                          finish=optimize.fmin)
print(resbrute[0])
```