Appliances energy Prediction

Abhinav Tiwari
Tiwari.abh@husky.neu.edu
CSYE 7245, Spring 2018, Northeastern University

Abstract

This project presents and discusses data-driven predictive models for the energy use of appliances. Data used include measurements of temperature and humidity sensors from a wireless network, weather from a nearby airport station and recorded energy use of lighting fixtures. The paper discusses data filtering to remove non-predictive parameters and feature ranking. Four statistical models were trained with repeated cross validation and evaluated in a testing set: (a) support vector machine with radial kernel, (b) random forest and (c)Xtreme gradient boosting. The best model(XgBoost) was able to explain 55% of the variance (R2) in testing set when using all the predictors. From the wireless network, the data from the kitchen, laundry and living room were ranked the highest in importance for the energy prediction. The prediction models with only the weather data, selected the atmospheric pressure (which is correlated to wind speed) as the most relevant weather data variable in the prediction. Therefore, atmospheric pressure may be important to include in energy prediction models and for building performance modeling.

Introduction

The understanding of the appliances energy use in buildings has been the subject of numerous research studies, since appliances represent a significant portion (between 20 and 30% of the electrical energy demand. For instance, in a study in the UK for domestic buildings, appliances, such as televisions and consumer electronics operating in standby were attributed to 10.2% increase in the electricity consumption. Regression models for energy use can help to understand the relationships between different variables and to quantify their impact. Thus, prediction models of electrical energy consumption in buildings can be useful for a number of applications: to determine adequate sizing of photovoltaics and energy storage to diminish power flow into the grid , to detect abnormal energy use patterns , to be part of an energy management system for load control to model predictive control applications where the loads are needed , demand side management (DSM) and demand side response(DSR) and as an input for building performance simulation analysis . Corresponding. As indicated in , the electricity consumption in domestic buildings is explained by two main factors: the type and number of electrical appliances and the use of the appliances by the occupants. Naturally, both factors are interrelated. The domestic appliances use by the occupants would leave traceable signals in the indoor environment near the vicinity of the appliance, for example: the temperature, humidity, vibrations, light and noise. The occupancy level of the building in different locations could also help to determine the use of the appliances. In this work, the prediction was carried out using different data sources and environmental parameters (indoor and outdoor conditions). Specifically, data from a nearby airport weather station, temperature and humidity in different rooms in the house from a wireless sensor network and one sub-metered electrical energy consumption (lights) have been used to predict the energy use by appliances. Four regression models have been tested, namely (a) support vector machine with radial basis function kernel (SVM-radial), (b) random forest (RF) and (c) Xtreme gradient boosting machines(XgBoost) with different combinations of predictors. The present work mostly deals with the problem of aggregate appliances energy use prediction rather than the topic of modeling of appliances energy loads. Because of that, the literature review focuses on this topic.

DATA:

Time series Energy Efficiency of appliance

- https://archive.ics.uci.edu/ml/datasets/Appliances+energy+prediction

```
Variable
Description
              ====================

              date time year-month-day hour :minute: second
              Appliances, energy use in Wh
              lights, energy use of light fixtures in the house in Wh
              T1, Temperature in kitchen area, in Celsius
              RH_1, Humidity in kitchen area, in %
              T2, Temperature in living room area, in Celsius
              RH_2, Humidity in living room area, in %
              T3, Temperature in laundry room area
              RH_3, Humidity in laundry room area, in %
              T4, Temperature in office room, in Celsius
              RH_4, Humidity in office room, in %
```

T5, Temperature in bathroom, in Celsius
RH_5, Humidity in bathroom, in %
T6, Temperature outside the building (north side), in Celsius
RH_6, Humidity outside the building (north side), in %
T7, Temperature in ironing room , in Celsius
RH_7, Humidity in ironing room, in %
T8, Temperature in teenager room 2, in Celsius
RH_8, Humidity in teenager room 2, in %
T9, Temperature in parent’s room, in Celsius
RH_9, Humidity in parent’s room, in %
To, Temperature outside (from Chièvres weather station), in Celsius
Pressure (from Chièvres weather station), in mm Hg
RH_out, Humidity outside (from Chièvres weather station), in %
Windspeed (from Chièvres weather station), in m/s
Visibility (from Chièvres weather station), in km
Tdewpoint (from Chièvres weather station), °C
rv1, Random variable 1, nondimensional
rv2, Rnadom variable 2, nondimensional

Where indicated, data from the nearest airport weather station (Chièvres Airport, Belgium) was downloaded from a public data set from Reliable Prognosis, rp5.ru. Permission was obtained from Reliable Prognosis for the distribution of the 4 months of data.

Code with Documentation:

Checking whether time-series is stationary or not

In [37]:
```
# Using graphical and Dickey-Fuller test to check whether data is stationary or not
from statsmodels.tsa.stattools import adfuller
def test_stationarity(timeseries):

    #Determing rolling statistics
    rolmean = pd.rolling_mean(timeseries, window=120)
    rolstd = pd.rolling_std(timeseries, window=120)

    #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)
    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:\Users\Abhinav\Anaconda3\lib\site-packages\statsmodels\compat\pandas.py:56: FutureWarning: The pandas.core.datetools module i
s deprecated and will be removed in a future version. Please use the pandas.tseries module instead.
  from pandas.core import datetools
```

```
Results of Dickey-Fuller Test:
Test Statistic                   -21.616378
p-value                            0.000000
#Lags Used                        11.000000
Number of Observations Used    19723.000000
Critical Value (1%)               -3.430682
Critical Value (5%)               -2.861687
Critical Value (10%)              -2.566848
dtype: float64
```

The ‘Test Statistic’ is less than the ‘Critical Value’, we can reject the null hypothesis and say that the series is stationary.

Feature scoring using F-score and removing features with p-value less than 0.05

In [184]:

```
# Arranging the dataframe in order of fscore
final_data = test.sort_values(by='fscore',ascending=False)
final_data
```

Out[184]:

	cols	fscore	pval
25	Hour	973.160402	1.394061e-208
0	lights	799.076292	2.305108e-172
21	RH_out	468.470558	1.077516e-102
3	T2	288.664211	2.784947e-64
11	T6	276.909655	9.333867e-62
19	T_out	195.934202	2.624854e-44
16	RH_8	176.061071	5.211566e-40
22	Windspeed	150.924181	1.456471e-34
2	RH_1	147.140998	9.639431e-34
5	T3	143.812455	5.086416e-33
12	RH_6	137.474160	1.209481e-31
4	RH_2	72.408319	1.873022e-17
14	RH_7	61.284663	5.187296e-15
1	T1	60.854665	6.449169e-15
27	day_0	58.369971	2.270739e-14
18	RH_9	52.398435	4.697109e-13
28	day_1	35.905110	2.107611e-09
7	T4	32.069964	1.507881e-08
15	T8	30.949192	2.683103e-08
6	RH_3	26.024294	2.402540e-07

In [185]:

```
#Removing all the columns which have p value less than 0.05
X_pval = final_data.loc[(final_data['pval'] <= 0.05), ['cols','fscore', 'pval']]
X_pval
```

Out[185]:

	cols	fscore	pval
25	Hour	973.160402	1.394061e-208
0	lights	799.076292	2.305108e-172
21	RH_out	468.470558	1.077516e-102
3	T2	288.664211	2.784947e-64
11	T6	276.909655	9.333867e-62
19	T_out	195.934202	2.624854e-44
16	RH_8	176.061071	5.211566e-40
22	Windspeed	150.924181	1.456471e-34
2	RH_1	147.140998	9.639431e-34
5	T3	143.812455	5.086416e-33
12	RH_6	137.474160	1.209481e-31
4	RH_2	72.408319	1.873022e-17

SVM using Radial Kernel with parameter tuning

In [40]:
```
from sklearn import svm, grid_search
def svr_param_selection(X, y, nfolds):
    Cs = [0.001, 0.01, 0.1, 1, 10]
    gammas = [0.001, 0.01, 0.1, 1]
    param_grid = {'C': Cs, 'gamma' : gammas}
    grid_search = GridSearchCV(svm.SVR(kernel='rbf'), param_grid, cv=nfolds)
    grid_search.fit(X, y)
    grid_search.best_params_
    return grid_search.best_params_
```

In [44]:
```
svr_param_selection(X_svm, y, 2)
```

Out[44]:
```
{'C': 10, 'gamma': 1}
```

In [49]:
```

X_train3, X_test3, y_train3, y_test3 = train_test_split(X_svm, y, test_size=0.2, random_state=2)
clf = svm.SVR(C=10,  gamma= 1, kernel='rbf')
clf.fit(X_train3, y_train3)
svr_pred = clf.predict(X_test3)
svr_score  = sqrt(mean_squared_error(y_test3, svr_pred))
svr_score
```

Out[49]:
```
90.97070745804282
```

SVM using radial kernel for grid searched value of c = 10 and gamma = 1 gave RMSE value of 90.97

Random Forest Regressor

```
from sklearn.ensemble import RandomForestRegressor
from sklearn.cross_validation import cross_val_score
from sklearn.cross_validation import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X_final, y, test_size=0.2, random_state=2)
regr = RandomForestRegressor(max_depth=2, random_state=0)
regr.fit(X_train, y_train)
```

```
Out[48]: RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=2,
                    max_features='auto', max_leaf_nodes=None,
                    min_impurity_decrease=0.0, min_impurity_split=None,
                    min_samples_leaf=1, min_samples_split=2,
                    min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=1,
                    oob_score=False, random_state=0, verbose=0, warm_start=False)
```

```
In [217]: regr.get_params
```

```
Out[217]: <bound method BaseEstimator.get_params of RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=2,
                    max_features='auto', max_leaf_nodes=None,
                    min_impurity_decrease=0.0, min_impurity_split=None,
                    min_samples_leaf=1, min_samples_split=2,
                    min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=1,
                    oob_score=False, random_state=0, verbose=0, warm_start=False)>
```

```
In [218]: predicted_rf = regr.predict(X_test)
```

```
In [219]: from sklearn.metrics import mean_squared_error
          from math import sqrt

          #getting rmse value of the prediction from random forest regressor
          rf_score  = sqrt(mean_squared_error(y_test, predicted_rf))
          rf_score
```

```
Out[219]: 89.95362252615679
```

Random Forest Regressor gave RMSE value of 89.95

Xgboost Model using default values

In [220]:
```
#Importing xgboost and getting xgbregressor
import xgboost as xgb
from xgboost import XGBRegressor
```

In [221]:
```
#Training the X_final features using default values using xgbRegressor
xgb = XGBRegressor(n_estimators=100, learning_rate=0.08, gamma=0, subsample=0.75, colsample_bytree=1, max_depth=7)
X_train1, X_test1, y_train1, y_test1 = train_test_split(X_final, y, test_size=0.2, random_state=1)
model = xgb.fit(X_train1,y_train1)
```

In [222]:
```
predicted_xgb = xgb.predict(X_test1)
```

In [224]:
```
#Getting the rmse of the predicted value
xgb_score  = sqrt(mean_squared_error(y_test1, predicted_xgb))
xgb_score
```

Out[224]:
```
77.18413144404134
```

XgBoost RMSE value is 77.18

RMSE value of xgboost is better than random forest

Tuning for Xgboost

```
# setting grid values for learning rate, estimators and max-depth
learning_rate = [0.0001, 0.001, 0.01, 0.1, 0.2, 0.3]
n_estimators = [50, 100, 150, 200]
max_depth = [2, 4, 6, 8]

#Putting the parameters in a dictionary for grid search
param_grid = dict(learning_rate=learning_rate, max_depth=max_depth, n_estimators=n_estimators)

#Using kfold cross validation technique where k=10
kfold = StratifiedKFold(n_splits=10, shuffle=True, random_state=7)
grid_search = GridSearchCV(model1, param_grid, n_jobs=-1, cv=kfold)
grid_result = grid_search.fit(X, y)
```

```
C:\Users\Abhinav\Anaconda3\lib\site-packages\sklearn\model_selection\_split.py:605: Warning: The least populated class in y has only 1 members, which is too few. The minimum number of members in any class cannot be less than n_splits=10.
  % (min_groups, self.n_splits)), Warning)
```

In [152]:
```
# summarize results and print the best values for the regressor found using grid search
print("Best: %f using %s" % (grid_result.best_score_, grid_result.best_params_))
```

```
Best: 0.547848 using {'learning_rate': 0.2, 'max_depth': 8, 'n_estimators': 200}
```

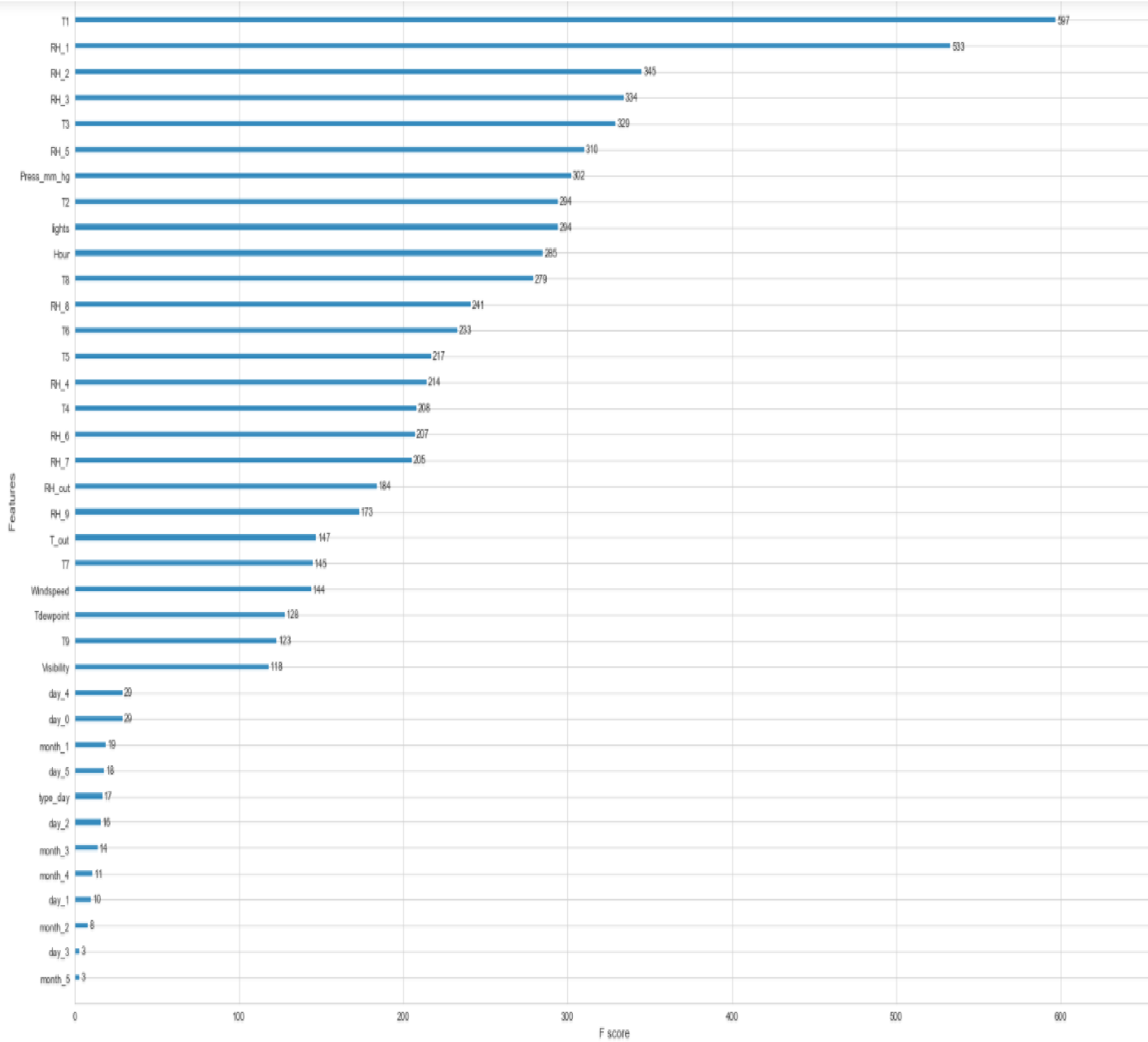
The best value for eta is 0.2, max_depth is 8 and estimators is 200, where 0.547 of variance of test set is explained.

In [201]:
```
xgb1 = XGBRegressor(n_estimators=200, learning_rate= 0.2, gamma=0, subsample=0.75, colsample_bytree=1, max_depth=8)
X_train2, X_test2, y_train2, y_test2 = train_test_split(X_final, y, test_size=0.2, random_state=2)
model1 = xgb1.fit(X_train2,y_train2)
cv_xgb = xgb.predict(X_test2)
xgb_score1  = sqrt(mean_squared_error(y_test2, cv_xgb))
xgb_score1
```

Out[201]: 56.56277246063585

Tuned XgBoost RMSE is 56.56

Feature Importance using Xgboost



The above graph shows that weather data, selected the atmospheric pressure (which is correlated to wind speed) as the most-relevant weather data variable in the prediction. Therefore, atmospheric pressure may be important to include in energy prediction models and for building performance modeling

Results:

- The dataset which was time-series was a stationary data.
- XgBoost gave the best result for this data set with a testing RMSE value of 56.6
- From the feature importance graph, it can be concluded that Pressure is the most important parameter when taking outside atmosphere in consideration for this dataset (also because it is highly correlated to windspeed).

Discussion

The statistical data analysis has shown thought-provoking results in both the exploratory analysis and in prediction models. The pairwise plots are useful because they shed light on the different relationships between parameters that could be hidden in final predictive models. The GBM and RF models improve the RSME and R2 of predictions compared to the SVM-radial. The weather data from the nearby weather station was shown to increase the prediction accuracy in the GBM models. The GBM models with only weather data ranked the pressure as the most important weather variable, followed by the outdoor temperature, dew point temperature, outdoor relative humidity, wind speed and visibility. The possible explanation for why the pressure has a strong prediction power may be related to its influence on the wind speed and higher rainfall probability which could potentially increase the occupancy of the house. Research found that atmospheric pressure is highly correlated with the cooling degree minutes (CDM) and heating degree minutes (HDM). Also, pressure has direct effects on air humidity ratio, density and enthalpy. Data from a wireless sensor network that measures humidity and temperature has been proven to increase the prediction accuracy. The data analysis showed that data from the kitchen, laundry room, living room and bathrooms had the most important contributions. Data from the other rooms also helps in the prediction. The prediction of appliances consumption with data from the wireless network indicates that it can help to locate where in a building the main appliances' energy consumption contribution are found. When using all the predictors the light consumption was ranked highly.

References

- Data driven prediction models of energy use of appliances in a low-energy house. Luis M. Candanedo, Véronique Feldheim, Dominique Deramaix. Energy and Buildings, Volume 140, 1 April 2017, Pages 81-97, ISSN 0378-7788, http://dx.doi.org/10.1016/j.enbuild.2017.01.083.
- https://www.analyticsvidhya.com/blog/tag/time-series-analysis/
- https://archive.ics.uci.edu/ml/datasets/Appliances+energy+prediction
- http://xgboost.readthedocs.io/en/latest/python/python_api.html