Week 6, 11-10-2019

Intelligent Buildings

7LY5M0_Yiwen Shen_Assignment 1

Problem 1: General questions

(1) What is an intelligent building and how does it differ from a building automation system? In your opinion what developments are needed to speed the future implementation of smart buildings?

A building with intelligent automation systems are not only able to consume energy and electricity effectively but also utilise sensors or actuators to collect data from the internal and external sources. (Internal including user data and energy consumption; external sources including weather forecast etc.) The collected data can be analysed by computer systems, optimised by the machine learning algorithms and thus providing insights to satisfy the purpose of improving the building performance, bring sustainabilities into people's lives.

In the recent development of Intelligent building automation systems, designers and engineers have gained interest in **User Experience** aspect (Wen, J. T. (2018). From an Industrial designer perspective, user experience design lies in the intersection of system design and aesthetics interaction (Ross, P. R., & Wensveen, S. A. 2010.). Aesthetic qualities exist not only in the form of building design but also in the interactions between occupants and building. Good interaction design in the context of commercial buildings will offer occupants good experience, maintain their good physical condition, they will also more likely to increase their level of user commitment to the building and thus improve their productivity (Verhaart, J., Li, R., & Zeiler, W. 2018).

For example, user involvement in the design of personal warming/ cooling system is essential (Verhaart, J., Li, R., & Zeiler, W. 2018). Occupants would like to require more thermal comfort in a building, however, thermal comfort is a very difficult aspect to address since it can be defined individually different based on the gender, age, the physical condition of different occupants, as well as humidity and air quality in the building. In order to create an environment with thermal comfort that satisfies the majority of occupants, designers need to study users as well as their behaviours and perceptions.

On the other hand, building managers would like to get more easy access to the data sources, where they can also observe and monitor building performance (Wen, J. T. (2018). For instance, in the control room, where buildings receive data from different distributed sensors, the obtained data should always be visualized clearly, allowing building managers to control building without extensive training. In my opinion, UI (user interface) design from the monitoring system should be designed simple yet functional to facilitate building managers to perform tasks.

(2) Explain the differences between extrapolation and interpolation. What are their pitfalls for Predictive modelling?

Extrapolation and interpolation are both methods used to predict or estimate hypothetical values for a variable. However, there are some differences between them:

- **Extrapolation** is a method that estimates a value that is beyond the original observation range (Extrapolation. 2019). Which means the value of a variable stands in between known observations. However, by using the extrapolation method, it usually has a higher risk to produce meaningless results that can not deliver higher accuracy in a predictive model.

- **Interpolation** is a method that estimates a value that stays within the range of a set of known data points (Interpolation. 2019). The value of a variable stands between known data points. However, the data points sometimes have likely very similar position or characteristics, For example, in a small amount of datasheet, when using 'ffill' method to perform interpolation function in Jupyter, the new predicted data will use the same value from the previous data point, the result would be very close to known data points, This can also affect the accuracy in a predictive model.

(3) Why is it important to check if a data is stationary or not? Which methods could be used to check the stationarity of a data set?
Stationary is a vital characteristic of time series data. Therefore, stationarity is an important aspect that needs to be addressed when performing data analysis or machine learning (Duke's Fuqua,2019). The goal of machine learning is to generate predictions beyond the training set of the model, stationaried data set is easy to perform prediction. For instance, a stationaried data set allows data analysts to simply predict its statistical properties will be the same in the future as algorithm models have been performed similarly in the past.

We can identify obvious stationary and non-stationary time series by observing line plot. Trends and seasonality (Duke's Fuqua,2019) can be found on stationary data sets. There are also many methods that can be used to check stationarity in the data set. Summary statistics and Statistics tests are two useful methods.

(4) Project Haystack is an initiative to reduce efforts in data preparation. What other initiatives can you think of that will reduce the workload in the data analytics process?
Altair knowledge hub (Data Governance Tools,2019) also utilizes machine learning algorithms that recommend data analysts relevant data sets and preparation steps to reduce unnecessary workload. Last but not least, Altair reduces thresholds for newcomers when learning about data preparation by designing an intuitive and collaborative user interface, it facilitates bridge the gaps between different stakeholders, designers, researchers and data analysts is able to collaborate the same data set remotely at the same time.

Alteryx Designer (Alteryx Designer,2019) is another data preparation cloud platform that aims to reduce the workload in data preparation, integration and analysis. Alteryx designers feature 'drag + drop' visual programming in an intuitive user interface to prep, blend and analyse data. Users can also perform advanced analytics such as predictive, statistical and spatial analytics in the same workflow.

We can conclude that the trend of performing data analysis seems to be more code-free or code-friendly by utilizing an intuitive user interface design on a cloud platform. As data analytics becomes a popular trend, the threshold for data analytics is getting lower and lower, and data analytics platforms that provide a good user experience take data analytics to a new level.

Problem 2: General questions

(1) The *pd.to_datetime* is used to convert UnixDateTime to DateTimeIndex, to make it clear for data analysis, the UnixDateTime is removed and a new index column named "TimeIndex" is added (see figure 1).

```
In [30]: # converting Unix date to DateTimeIndex
df['TimeIndex'] = pd.to_datetime(df['UnixDateTime'],unit = 's')
df.drop('UnixDateTime', axis =1, inplace=True)
df.set_index('TimeIndex')
```

Out[30]:

TimeIndex	Aggregate [W]	Fridge [W]	Freezer [W]	Washer Dryer [W]	Washing Machine [W]	Toaster [W]	Computer [W]	Television Site [W]	Microwave [W]	Kettle [W]
2013-11-01 22:13:18	358.0	0	0	0	0	0	17	138	2	0
2013-11-01 22:13:31	357.0	0	0	0	0	0	17	138	2	0
2013-11-01 22:13:46	358.0	0	0	0	0	0	17	138	2	0
2013-11-01 22:13:59	357.0	0	0	0	0	0	16	138	2	0
2013-11-01 22:14:14	NaN	0	0	0	0	0	17	139	2	0
2013-11-01 22:14:16	358.0	0	0	0	0	0	17	139	2	0

Figure 1. Data Frame

- *pd.isnull(df)* is used to identify the location of the NaN value.
- *pd.isnull(df).sum()* is used to calculate missing values. In the given datasheet, it seems there are only missing values in the Aggregate, to calculate the ratio of missing data, the following operation is performed:

```
ratio_missing_data = pd.isnull(df['Aggregate [W]']).sum()/len(df['Aggregate [W]'])
```

The ratio of missing data is **0.19999997061285135**

(2) To choose which method will be suitable to fill in the missing values. First of all, the missing values and aggregate values are being plotted. The blue dots indicate the missing values (see figure 2).

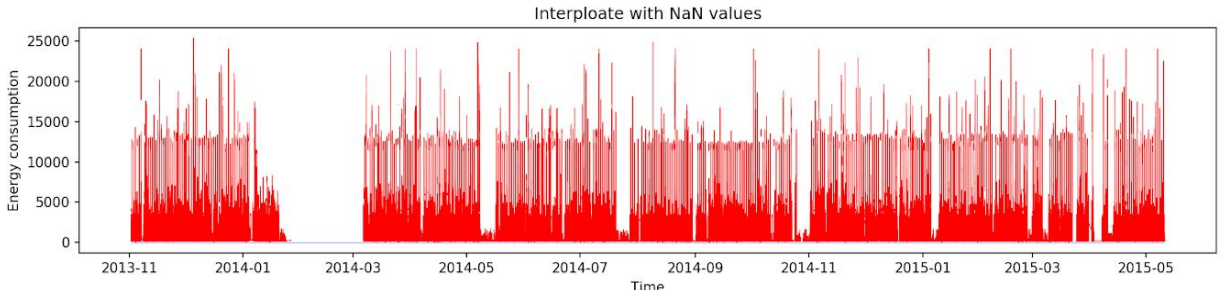


Figure 2. Interpolate NaN values

Three interpolate functions are used to fill in the missing values.

- **Method 1: df1 = df.fillna(0)**, this method will replace all NaN elements with 0s. As we can see from the following figure,the blue dot is covered by red line (see figure 3). However, by using this method will make the mean values look horrible,it will also make the prediction less precious.

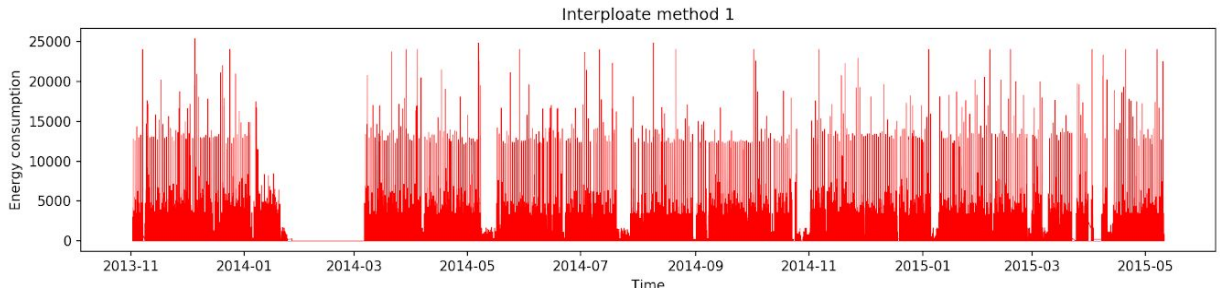


Figure 3. Method 1

- **Method 2: df2 = df.fillna(method = 'ffill')**, this method will replace all NaN elements forward or backward (see figure 4).

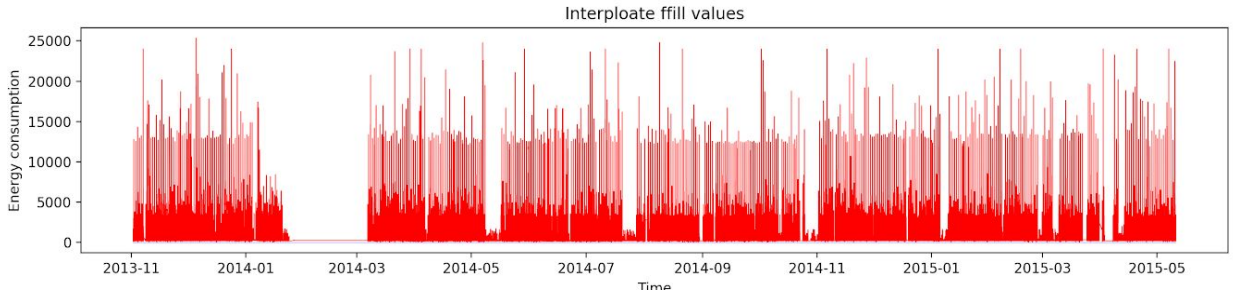


Figure 4. Method 2

- **Method 3: df3 = df.fillna(df['Aggregate'].mean())**, this method will replace all NaN elements with mean value of aggregate. Comparing the missing values and replaced values that fill with mean value of aggregate seems deliver higher accuracy for prediction, therefore method 3 is being used (see figure 5).

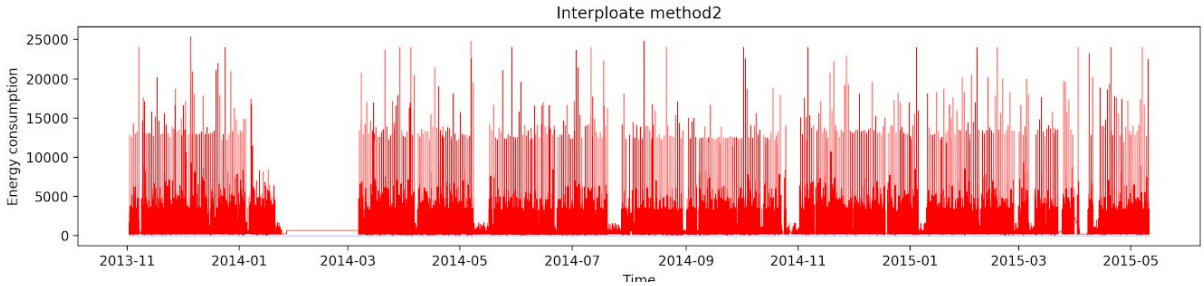


Figure 5. Method 3

(3) Outlier detection

- To find outliers, the ***EllipticEnvelope*** library was used in this data set. The dataframe 3 (missing values were filled with mean value of aggregate) was used. The parameter of outliner fraction is 0.001. The following algorithm was applied to find the amount of outliers:

```
n_outliers = int(outliers_fraction * n_samples)
n_inliers = n_samples - n_outliers
```

- Resample the dataframe into an array of tuples and the timedate index object to ints before training the model .
- Execute algorithm to identify the outliers, and then check the number of outliers from the prediction match to the amount of outliers, there is one outliner missing, but the result is very close to what we predict.

```
# Check if number of outliers found is corr
y_pred[y_pred == -1].sum()
```

```
-6806
```

```
n_outliers
```

```
6805
```

Figure 6. Outliers check

The plot shows the amount of outliers (see figure 7).

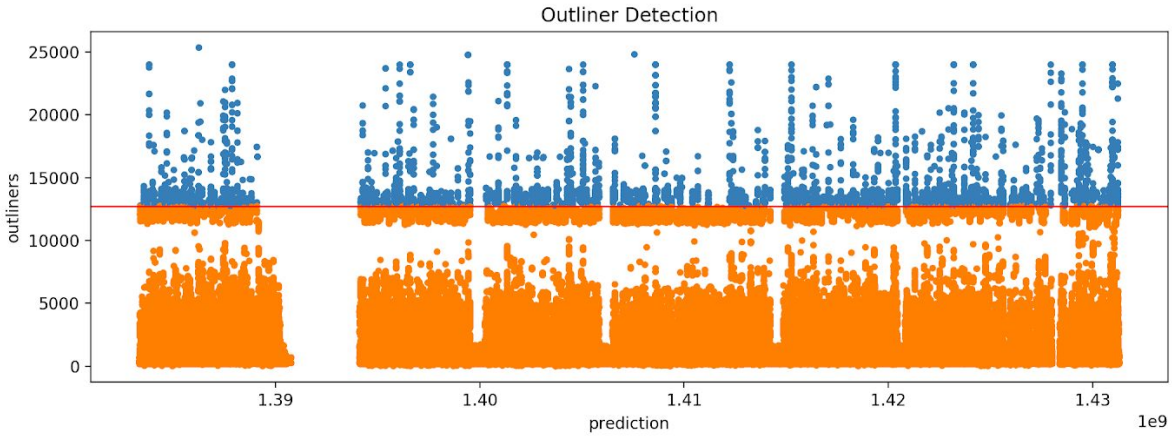


Figure 7. Outliers

(4) Warning function

The warning function is executed by defining a function named 'warning'. Inside of the warning function, a if-statement is used when the consumption of power reaches 200kw. The exact date and time will be printed and stored in a new Excel file named 'new dataframe' (see appendix x). In the exercise only the first 10 warnings were printed to demonstrate the warning function was executed correctly. By observing the sum of appliances, we could conclude that the cause of the power surge was due to **none** of the appliances.

(5) After resampling the aggregate power as daily, weekly and monthly by using sum and mean resampling function,we can see a significant difference by comparing both plots.

- In the resample sum value chart, the sum values of daily and weekly aggregate have been relatively stable, although there are significant drops in 2014-01 and 2014 -11. In contrast, the sum of monthly aggregate suffered a severe decline. Particularly from 2014-01 to 2014-03, It is hard to find relationships between these sum values.

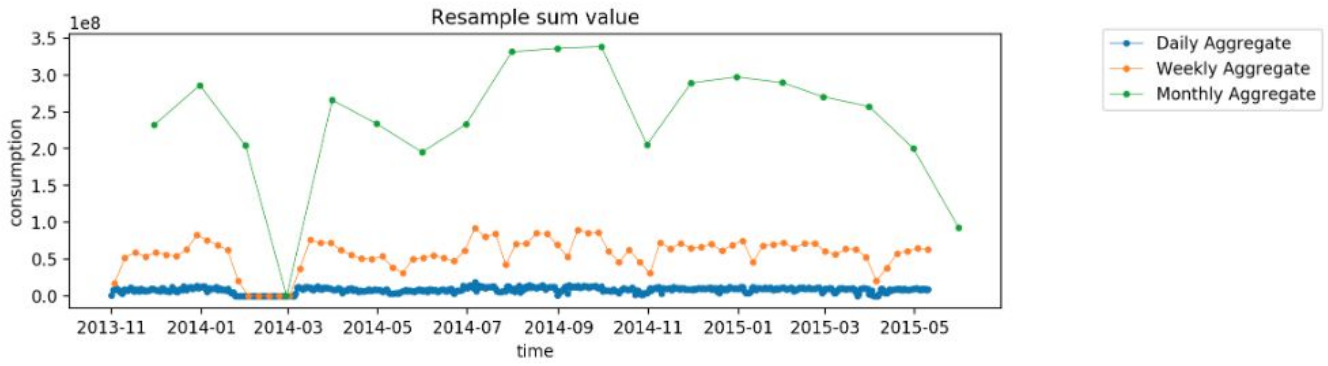


Figure 8. Resample sum value

- In the resample mean value chart, the mean values of weekly aggregate and monthly aggregate have been relatively stable, they have similar amplitude and fluctuation. In contrast, The mean values of daily life have been more dynamic.

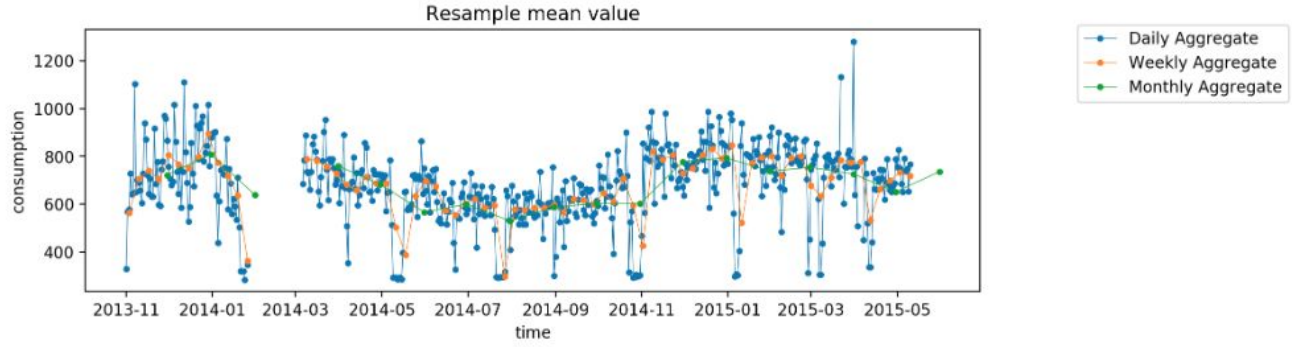


Figure 9. Resample mean values

Problem 3: Extracting trends from observations

(1) Heatmap

To create a heatmap shows the fan speed with hours and days on respectively on Y-axis and X-axis, the ***seaborn library*** was utilized. By using pandas date-time index function, the data sets were able to be split into different periods.

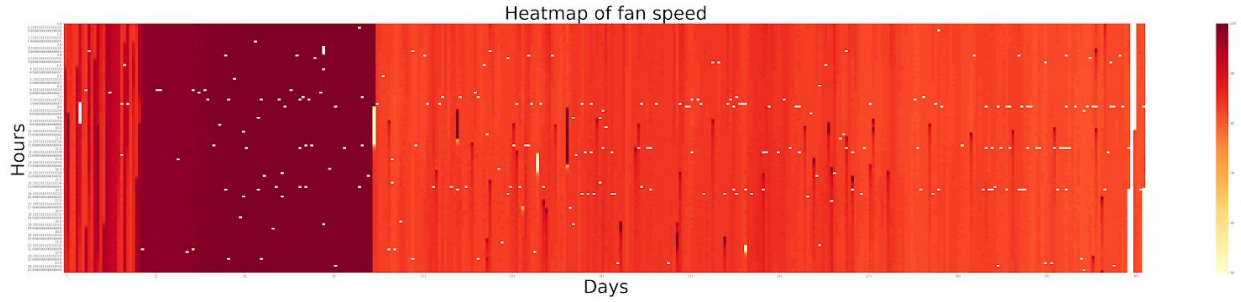


Figure 10. Heatmap

(2-1) Irregular control patterns:

- The white dots and spaces indicate missing data (**NaN** values), which means data on that moment of the day were not recorded. For instance, on day 361, data were not recorded in the morning but indeed recorded at night (figure 11 & 12).

```
In [305]: df.loc[df['day_exact']== 361]
```

Out[305]:

System Time	Fan control	Supply Air Temp	Ambient Air Temp	year	month	day	hour	hour_exact	day_exact
2015-12-27 00:00:00	NaN	NaN	NaN	2015	12	27	0	0.000000	361
2015-12-27 00:05:00	NaN	NaN	NaN	2015	12	27	0	0.083333	361
2015-12-27 00:10:00	NaN	NaN	NaN	2015	12	27	0	0.166667	361
2015-12-27 00:15:00	NaN	NaN	NaN	2015	12	27	0	0.250000	361
...	...	...	...	...	...	...	...	...	...
2015-12-27 21:30:00	82.0	20.2000	11.9875	2015	12	27	21	21.500000	361
2015-12-27 21:35:00	82.0	20.2000	12.2000	2015	12	27	21	21.583333	361
2015-12-27 21:40:00	82.0	20.2000	12.4500	2015	12	27	21	21.666667	361

Figure 11 & 12. NaN values

- In the first month (day 1 to 25), the performance of the fan does not look normal, it is difficult to make predictions from the information given by the heat map.

(2-2)Operation Schedule:

- From 26th of February to 15th of April (day 26 to 105) the speed of fans reached the highest peak (around 100). In the rest of the year (from 16th of April to 31st of December) the speed of fan maintains around 80.
- The fan increases its speed most likely between 8:00 am-9:00 am, 1:00 pm -2:00 pm and 8:00pm - 9:00 pm.

(3) Scatterplot:

To create a scatter plot, the ***matplotlib*** was used, supply air temperature on Y-axis and ambient air temperature is on X-axis (see figure 13).

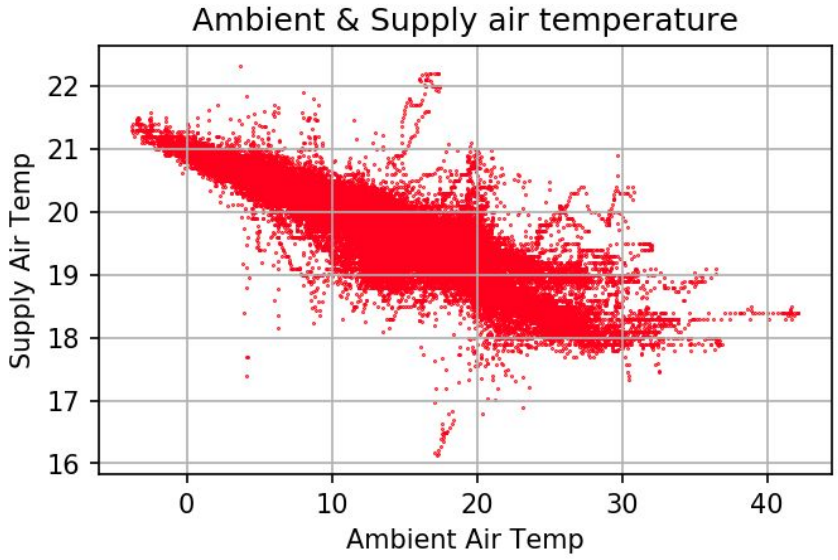


Figure 13. Scatter Plot

(4) Irregular control patterns:
In general, this is a negative correlation and several outliers can be found in the scatter plot. The supply air temperature is sometimes higher or low compared to the rest of the data. For example. some irregular behaviors occur when the ambient air temperature is around 15 degrees.

Control settings
This is a negative correlation, which means the relationship between two variables (x = supply air temperature, y = ambient air temperature) whereby they move in opposite directions. If variables x and y are negatively correlated, as x increases in value, y will decrease; similarly, if x decreases in value, y will increase. Thus, the control settings can be: as the supply air temperature is increased, the ambient air temperature will decrease, similarly, if supply air temperature decreases in value, ambient air temperature will increase.

(5) ACF & PACF
To create ACF and PACF plot, it is necessary to fill the missing data, therefore, the '***ffill***' method was used to fill the missing data. Specifically, the ***Statsmodel library*** (Statsmodels.2019) was used to create ACF and PACF plots.

- **ACF** measures the correlation between the present value of the series with its past values. ACF contains components like trend and seasonality. This allows data analysts to estimate predictions and find patterns in the data. In the plot (figure x), we could find out there is a trend in the data, the autocorrelations tend to have positive values that decrease as the lags increase.

- **PACF** measures the correlation between a variable and a lag of itself that is not explained by correlations at all lower-order lags. In the plot, the partial autocorrelations significant spike only at lag 1, 2,3,4, especially the value at lag 3 is negative. The partial autocorrelations tend to have both positive and negative values. It is impossible to find a global trend is PACF.

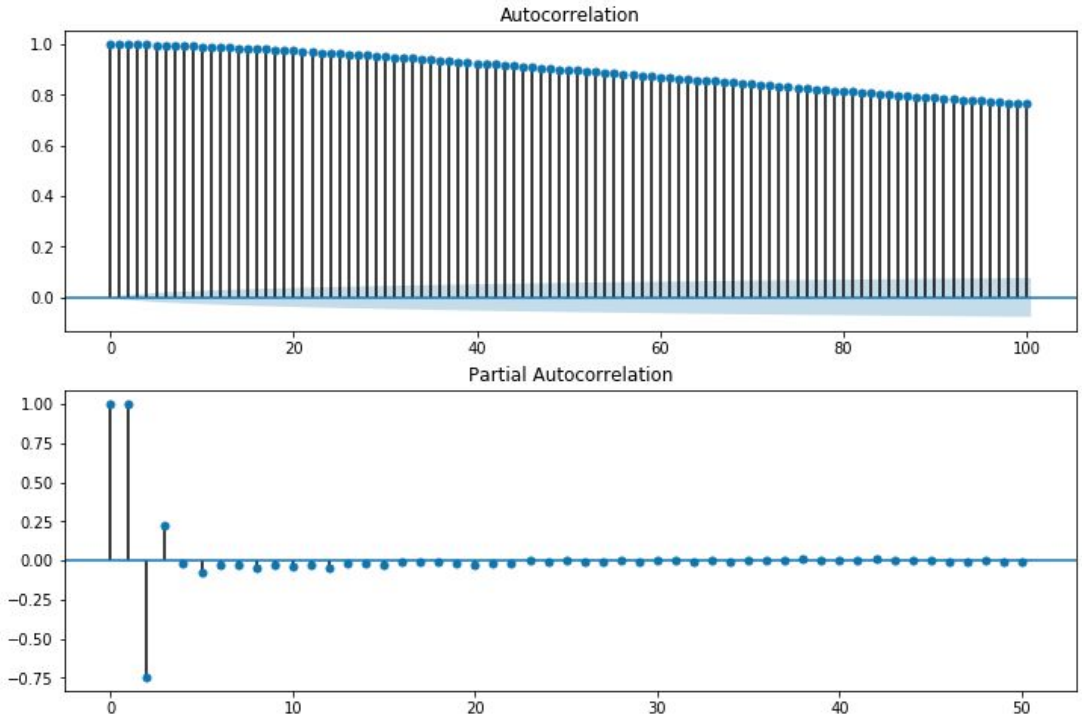


Figure 14. ACF & PACF

(6) Linear Regression

The **linear** regression model was used to fit the regression model and **MSE (mean square error)** metric was used to evaluate the quality of the fitting.

First of all, it is important to assess the performance of the model, therefore, the k-folds cross validation (k = 3) was used to access its performance. As we can see the score of the model is around 0.79, which indicates that the performance of the model is relatively decent.

```
model = LinearRegression()
scores = []
kfold = KFold(n_splits=3, shuffle=True)
for i, (train, test) in enumerate(kfold.split(X, Y)):
 model.fit(X.iloc[train,:], Y.iloc[train,:])
 scores.append(model.score(X.iloc[test,:], Y.iloc[test,:]))
print(scores)
```

```
[0.7941475425982178, 0.7899141336785844, 0.7934601199424676]
```

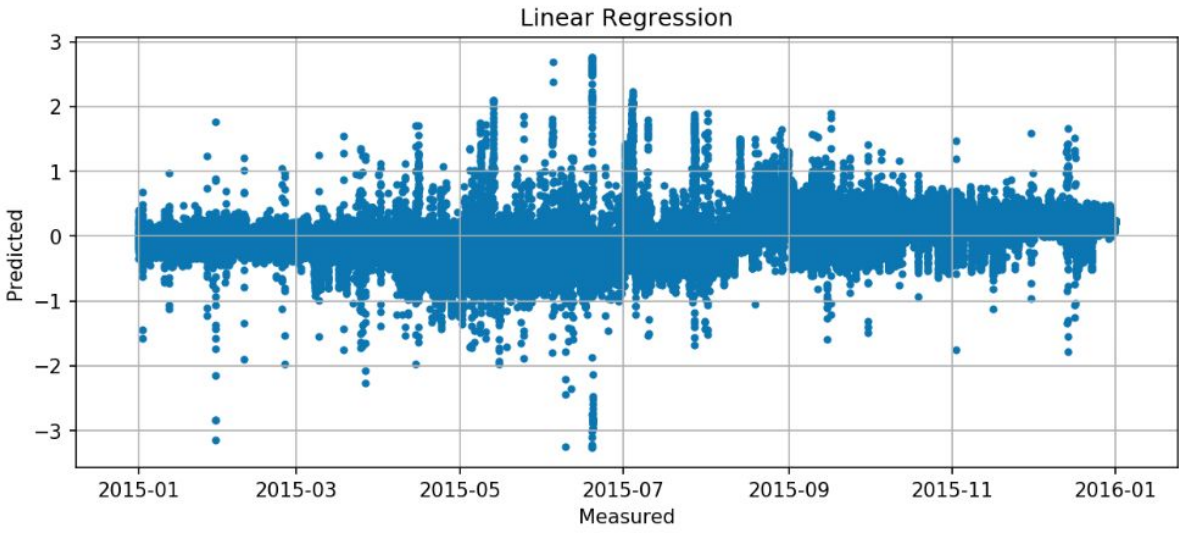
```
plt.figure(figsize = (10,4), dpi = 150)
plt.scatter(X.index,Y-predictions, marker='.')
mean_square_error_metric = sum((Y-predictions)['Supply Air Temp']**2)/len(Y)

plt.title('Linear Regression')
plt.xlabel('Measured')
plt.ylabel('Predicted')
plt.savefig('Interploate fillin values')

plt.grid(True)
plt.show()
print(mean_square_error_metric)
```

Figure 15. Scatter Plot

MSE is the mean of the squared difference between estimate and the data. This is an identical calculation to the calculation of the variance of a statistic, where the estimate is the mean (Mean squared error 2019). Ideally the MSE should be 0, however, in reality it is impossible to achieve 0 in a model, therefore, smaller MSE generally indicates a better estimate. In this model, the MSE is 0.091, this indicates that we have obtained a score.




```
0.09146927211112549
```

Figure 16. Scatter Plot

Reference

1. Wen, J. T. (2018). *Intelligent building control systems: a survey of modern building control and sensing strategies*. Cham, Switzerland: Springer.
2. Ross, P. R., & Wensveen, S. A. (2010). *Designing aesthetics of behavior in interaction: Using aesthetic experience as a mechanism for design*. International Journal of Design, 4(2), 3-13.
3. Verhaart, J., Li, R., & Zeiler, W. (2018). *User interaction patterns of a personal cooling system: A measurement study*. Science and Technology for the Built Environment, 24(1), 57-72.
4. Extrapolation. (2019, October 9). Retrieved from https://en.wikipedia.org/wiki/Extrapolation.
5. Interpolation. (2019, October 9). Retrieved from https://en.wikipedia.org/wiki/Interpolation.
6. Duke's Fuqua . (n.d.). Retrieved from https://www.fuqua.duke.edu/.
7. Knowledge Hub. (2019). Retrieved from https://www.datawatch.com/in-action/knowledge-hub/.
8. Alteryx Designer . (2019). Retrieved from https://www.alteryx.com/
9. Means squared error.(2019.) Retrieve from https://en.wikipedia.org/wiki/Mean_squared_error.
10. Stasmodels.(2019). Retrieved from https://www.statsmodels.org/stable/index.html.

Appendix

1. Warning function excel

Intelligent Buildings

7LY5M0_Yiwen Shen_Assignment 2

1. Introduction

Creating a comfortable indoor environment is one of the essential functions of buildings as it affects occupant satisfaction [2,7], well-being [8] and productivity [9]. This brings design challenges in particularly HVAC systems that not only provide thermal comfort to satisfy the needs of most occupants but also to reduce energy consumption to improve the building performance [3,4]. However, the most commonly used methods (e.g. PID control and on/off control) to manage cooling, heating and ventilation systems of the entire building are becoming inefficient.

At the same time, with the growth in big data in the design of HVAC systems, engineers and designers are facing more complex tasks or problems involving a large amount of data as well as various environmental factors (airspeed, humidity, air temperature) without any existing formulas or equations to evaluate individual thermal comfort. Additionally, due to the nature of data that keeps updating, this requires companies to invest a lot of time and specialists to perform intensive data analysis if building managers/owners want to improve the building performance.

To overcome these drawbacks mentioned above, several studies [2,3] have been conducted into personal conditioning systems by utilizing machine learning algorithms, which are able to offer cooling and heating locally to occupants. For example, in the work of [2], an individual conditioning system is installed around the desk to mimic an office room in the climate chamber, at the Faculty of Built Environment, Eindhoven University of Technology. By having such a system would create opportunities for users to adapt to the local climate around their workplace to make themselves comfortable. Furthermore, users can also feedback their desires and needs regarding heating or cooling as input data to systems to indicate their preferred thermal conditions.

Personal data in terms of environmental factors (e.g. air temperature, airspeed, humidity) and personal factors (e.g. head, feet, mean skin temperature) were collected that serves to create a predictive personal comfort model. In this report, we implemented these personal data in the MATLAB environment by using supervised machine learning algorithms.

2. Goal

This assignment aims to develop a predictive comfort model by utilizing machine learning algorithms. The obtained data is recorded by using several embedded temperature sensors on different body parts. The exercise will be executed in MATLAB by using Classification Learner App [1], it is an app in the Statistics and Machine Learning Toolbox that allows training models to classify data by using supervised machine learning. Several algorithms are included in this app but mainly two categories of algorithms (SVM and Ensemble) were applied in the assignment. In order to achieve a predictive model that scores the highest accuracy, parameter fine-tuning was iteratively performed.

Input parameters are time, mean skin temperature and local skin temperature, the resulting response represent the thermal comfort/sensation vote by the individual user. These votes were classified into 3 classes (-1, 0, 1) to be addressed as a classification problem.

- -1 indicates users need heating
- 0 indicates the thermal condition was perceived comfortable
- 1 indicates that users need cooling.

10-fold cross-validation was used to estimate the performance of the model on the following indicators: **1)** Accuracy, **2)** Average AUC and **3)** True Positive Rate.

3 Method

There is no single machine learning algorithm that works for every problem, and identifying the suitable algorithm is always a process of trial and error [1]. This assignment aims to develop a predictive model that delivers the highest accuracy, this requires iteratively training and evaluating classification models. The training procedure will be executed as listed below:

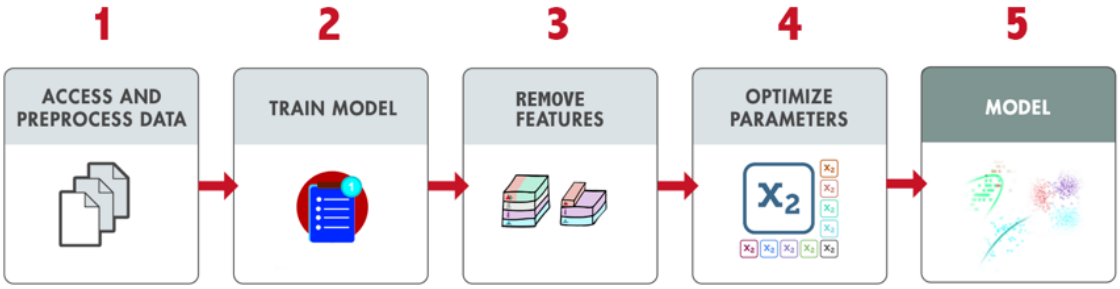


Figure 1. Training procedure to achieve satisfactory performance

1. **Access and preprocess data:** Data analysis & visualization in Jupyter Notebook
2. **Train models:**
    a. Test 1: SVM (one vs one) and Ensemble algorithms with all features
    b. Test 2: SVM (one vs all) with all features.
3. **Sensitivity analysis**
    a. Test 3: Remove features one by one in SVM (one vs one) and Ensemble algorithms
4. **Optimize parameters:**
    a. Test 4: Parameter fine-tuning in SVM and Ensemble algorithms
5. **Selection of the best model.**

In this exercise, two categories of algorithms are being used: SVM and Ensemble algorithms. Executing different algorithms can acquire different accuracy, it is useful to distinguish each algorithm's characteristic in terms of its flexibility and prediction speed. Higher flexibility usually can result in a better accuracy rate and fast prediction speed can be beneficial to eliminate time spent in training models. Table 1 lists the difference in accuracies of different algorithms:

Learner Type	Learning Algorithms	Flexibility	Prediction Speed
SVM	Linear SVM	Low	Binary: Fast / Multiclass: Medium
	Quadratic SVM	Medium	Binary: Fast / Multiclass: Slow
	Cubic SVM	Medium	Binary: Fast / Multiclass: Slow
	Fine Gaussian SVM	High	Binary: Fast / Multiclass: Slow
	Medium Gaussian SVM	Medium	Binary: Fast / Multiclass: Slow
	Coarse Gaussian SVM	Low	Binary: Fast / Multiclass: Slow
ENSEMBLE	Boosted Trees	Medium to High	Fast
	Bagged Tree	High	Medium
	Subspace Discriminant	Medium	Medium
	Subspace KNN	Medium	Medium
	RUSBoosted Trees	Medium	Fast

Table 1. Algorithms used for performing machine learning and their corresponding flexibility and prediction speed [6].

4 Results

4.1 Access and preprocess data
Data analysis & visualization in Jupyter Notebook

The first step in performing any machine learning project is to observe the given data. The excel file (Shen.xlsx) was imported into Jupyter Notebook to perform data analysis and visualization. When it comes to analyzing data with a large number of datasets, it usually takes a lot of time to manually process the data in excel file, therefore, using Juypter Notebook not only saves time and improves work efficiency, but also offers high flexibility and accuracy.

Table 2 shows that amount of the records in each class. Regarding to the calculation that was performed in Juypter [Appendix 1], 455 records were obtained in this dataset, which can be considered as a small size of dataset. The pie chart and line chart were both created by utilizes Matplotlib library in Juypter. According to the pie chart, it is evident that the distribution of each class is unbalanced:

- Class 0 takes up more than half of the dataset, which accounts for 50,3% of the total
- Class 1 only accounts for approximately one third of the total (29.5%) and
- Class -1 only accounts for 20,2% of the total

In summary, based on the observation of the pie chart, the accuracy may not be high enough due to the size of the dataset and the unbalanced distribution of the classes.

Classifier	Records
-1	92
0	229
1	134
Total	455

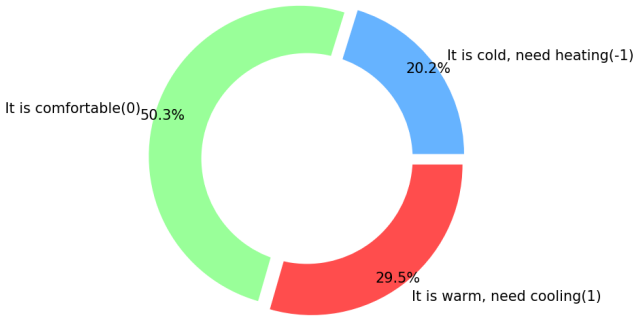


Table 2 & Figure 2. Number of recorded classifiers and total their percentage in the dataset "Shen.xlsx'

The line chart below shows the temperature changes obtained from different body parts. Hands, feet, mean skin and backL are represented by red, blue, brown and green color respectively. According to figure2, the temperature of the mean skin tends to be relatively stable over time, the temperature change on the backL is the most dynamic, and the temperature changes of the hands and feet are most similar.

This line chart gives an intuitive observation of the temperature changes at different body parts. The purpose to visualize all the features serves to observe the effect of removing each feature on the accuracy of the overall model.

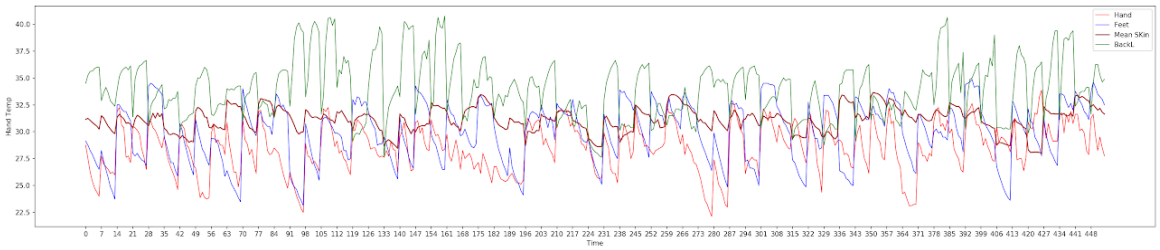


Figure 2. Line chart of each parameters from dataset.

4.2 Training model

Test 1: SVM (one vs one) and Ensemble algorithms with all features.

In the first test, all the SVM algorithms and Ensemble algorithms are executed at the same time in MATLAB by following the default parameter settings. Table 3 shows the summary of the results. The color code method is used to quickly find the significant value of indicators:

- Green cells represent the highest-scoring metrics of all algorithms.
- The blue cell represents the average value of all indicators, the red text indicates the low value of the TPR score, and the green text represents the high value.

Algorithms		Accuracy	-1			0			1			AUC (Average)
			FPR	TPR	AUC	FPR	TPR	AUC	FPR	TPR	AUC	
SVM	Linear	50.3%	100%	0	55%	0	100%	51%	100%	0	48%	51.33%
	Quadratic	53%	97%	3%	66%	9%	91%	57%	78%	22%	63%	62%
	Cubic	56%	67%	33%	74%	28%	72%	65%	54%	46%	75%	71.33%
	Fine Gaussian	53.2%	95%	5%	66%	10%	90%	63%	76%	24%	69%	66%
	Medium Gaussian	56.3%	97%	3%	69%	4%	96%	61%	75%	25%	69%	66.33%
	Coarse Gaussian	50.3%	100%	0	57%	0	100%	55%	100&	0	60%	57.33%
Average		53,2%	92,6%	7,3%	64,5%	8,5%	91,8%	58,6%	80,5%	19,5%	64%	64,4%
Ensemble	Boosted	53.6%	85%	15%	67%	16%	84%	62%	72%	28%	64%	64.33%
	Bagged	54.7%	63%	37%	74%	28%	72%	66%	62%	38%	67%	63%
	Subspace Discriminant	50.1%	100%	0	58%	2%	98%	55%	97%	3%	50%	54.33%
	Sunspace KNN	47.3%	79%	21%	62%	33%	67%	54%	69%	31%	56%	57.33%
	RUSBoosted Trees	49.2%	43%	57%	71%	53%	47%	65%	52%	48%	68%	68%
Average		50,98%	74%	26%	66,4	26,4%	73,6%	60,4%	70,4%	29,6%	61%	61,4%

Table 3. Results of SVM and Ensemble algorithms with all features.

Inferences:

- TPR for class -1 and 1 result relatively better by applying Ensemble algorithms.
- TPR for class 0 achieves highest average score comparing with 1 and -1, additionally, SVM algorithms perform better than Ensemble algorithms according to the average accuracy and TPR.
- TPR for class -1 results poorly (between 0 to 33%) by using SVM algorithms and relatively better (between 0 to 57%) by using Ensemble algorithms. The cause of this phenomenon is likely to be the unbalanced distribution of class -1, which only accounts for 20,2% of the total dataset. Thus, -1 has the lowest prediction rate.
- TPR for class 1 results slightly better but still looks poor, the cause of this phenomenon can also be the unbalanced distribution of class 1, which only accounts 29,5% of the total dataset.
- TPR for class 0 results highest (the closer to 100%, the better), however, in theory, the value of TPR cannot reach to 100%, otherwise, there may be a great possibility of overfitting.
- AUC for class -1.0 and 1 is always between 58% to 65%, which can be considered at the level of poor or fail regarding to [5].
- The best algorithm is Medium Gaussian which scores 56,3%, after that is Cubic and Bagged Trees that score 56% and 54,7% respectively. Thus, correlation between

algorithm flexibility and accuracy can be seen. Algorithm with a higher flexibility delivers higher accuracy. Prediction speed does not seem to have any correlation with accuracy.

Test 2: SVM (one vs all) with all features.

In the first test, all the SVM algorithms were executed by following the default settings (one-vs-one). It may be worth trying all the SVM algorithms with different multiclass method which is one-vs-all to see if the accuracy can be improved. Table 4 shows the summary of the results. The same color code method from previous test was also applied in this observation.

Algorithms		Accuracy	-1			0			1			AUC (Average)
			FPR	TPR	AUC	FPR	TPR	AUC	FPR	TPR	AUC	
SVM (one vs one)	Linear	50.3%	100%	0	55%	0	100%	51%	100%	0	48%	51.33%
	Quadratic	53%	97%	3%	66%	9%	91%	57%	78%	22%	63%	62%
	Cubic	56%	67%	33%	74%	28%	72%	65%	54%	46%	75%	71.33%
	Fine Gaussian	53.2%	95%	5%	66%	10%	90%	63%	76%	24%	69%	66%
	Medium Gaussian	56.3%	97%	3%	69%	4%	96%	61%	75%	25%	69%	66.33%
	Coarse Gaussian	50.3%	100%	0	57%	0	100%	55%	100&	0	60%	57.33%
Average		53.2%	92.6%	7.3%	64.5%	8.5%	91.8%	58.6%	80.5%	19.5%	64%	64.4%
SVM (one vs all)	Linear	48.6%	98%	2%	57%	10%	90%	60%	94%	6%	53%	56.66%
	Quadratic	53.4%	100%	0	59%	16%	84%	64%	73%	27%	59%	60.67%
	Cubic	55.8%	82%	18%	65%	8%	92%	65%	67%	33%	68%	66%
	Fine Gaussian	57.8%	91%	9%	64%	8%	92%	65%	34%	66%	64%	64.3%
	Medium Gaussian	57.8%	91%	9%	64%	3%	97%	49%	34%	66%	64%	59%
	Coarse Gaussian	51.0%	100%	0	56%	10%	90%	60%	93%	7%	55%	57%
Average		54%	93.67%	6.5%	60.83%	9.16%	90.8%	52.3%	65.8%	34.16%	60.5%	60.6%

Table 4. Results of SVM with different multiclass method

Inferences:

- The average accuracy is slightly improved (+ 0.8%), with the largest increase occurring in Fine Gaussian with an accuracy increases 4.6%.
- The Average TPR for class -1 and 0 are decreased -0.8 % and -1% respectively, in contrast, the value of average TPR for class 1 increases significantly by 14,66%.
- The Average AUC for all the algorithms is decreased.
- Although the average accuracy is slightly improved, TPR for class-1 and 1 still perform poorly in prediction by using the new classification method. The reason may still be the unbalanced distribution of classes in dataset.

4.3 Sensitivity Analysis.

Test 3: Remove features one by one in SVM (one vs one) and ensemble algorithms

Sensitivity analysis is executed in the third test, this serves to find out how an individual feature (input parameter) influences the accuracy of a learning model. Regarding to the data visualization and analysis that was performed in the section 4.1, it is evident that each input parameter performed differently that may influence the accuracy of the model. The experiment is done by removing each feature (parameter) one by one and analyzing the difference in the resulting accuracy. Color code method is applied in this experiment to provide a quick visual interpretation.

- Blue cells represent the average value of all the indicators, red text indicates the increased value and green text indicates the decreased value.

Algorithms		Accuracy without different features and each difference value						
		Accuracy	w/o time	w/o Hand	w/o Feet	w/o Mean skin	w/o BackL	Average
SVM	Linear	50.3%	50.3% (0)	50.3% (0)	50.3% (0)	50.3% (0)	50.3% (0)	50.3%
	Quadratic	53%	53.6% (+0.6%)	52.7% (-0.3%)	52.1% (-0.9%)	54.5% (+1.5%)	53.8% (0.8%)	53.28%
	Cubic (slow)	56%	57.4% (+1.4%)	56.5% (+0.5%)	53% (-3%)	53% (-3%)	56.9% (0.9%)	55.46%
	Fine Gaussian	53.2%	57.1% (+3.9%)	52.7% (-0.5%)	50.3% (-2.9%)	47.9% (+3.4%)	54.3% (-1.1%)	52.58%
	Medium Gaussian	56.3%	56.0% (-0.3%)	55.4% (-0.9%)	53.8% (-2.5%)	54.5% (-1.8 %)	54.1% (-2.2%)	55.01%
	Coarse Gaussian	51.0%	50.3% (-0.7%)	50.3% (-0.7%)	50.3% (-0.7%)	50.3% (-0.7%)	50.3% (-0.7%)	50.41%
	Average	53.3%	54.11% (+0.8%)	52.9% (-0.3%)	51.6% (-1.7%)	51.7% (-1.6%)	53.28% (-0.02%)	52.84%
Ensemble	Boosted	53.6%	53.2% (-0.4%)	51.2% (-2.4%)	54.3% (+0.7%)	52.3% (-1.3%)	53.4% (-0.2%)	53%
	Bagged	54.7%	53.2% (-1.5%)	52.3% (-2.5%)	51.9% (-2.8%)	50.1% (-4.6%)	53.4% (-1.3%)	52.75%
	Subspace Discriminant	50.1%	50.1% (0)	50.3% (+0.2%)	50.3% (+0.2%)	50.3% (+0.2%)	49.9% (+0.2%)	50.16%
	Subspace KNN	47.3%	48.1% (+0.8%)	47.0% (-0.3%)	46.4% (-0.9%)	48.6% (+1.3%)	49.0% (+1.7%)	47.43%
	RUSBoosted Trees	49.2%	51.6% (+2.4%)	47.0% (-2.2%)	49.2% (0)	43.1% (-6.1%)	48.8% (-0.4%)	48.15%
	Average	51%	51.2% (+0.2%)	49.6% (-0.4%)	50.2% (-0.8%)	50.4% (-0.6%)	50.9% (-0.1%)	50.3%

Table 5. Results for sensitivity study with features selected for prediction.

Inferences:

- The average accuracy of all the algorithms decreased except without the Time feature. Thus, the Time feature seems to be a removable feature when performing training, the rest features (Hand, Feet, BackL and Mean skin) are recommended to be kept in prediction.
- Algorithms perform their worst prediction without the Feet feature. The average accuracy in SVM and Ensemble algorithms decreased by -1.7% and -0,8% respectively, therefore, Feet feature should be the one that needs to be kept in prediction.
- According to the previous visualization (section 4.1), although the curvature of the Hand feature and Feet feature tend to be consistent, the maintain of the Feet feature is obviously more important than the Hand feature in the prediction.
- Excluding any features does not seem to affect accuracy by using the SVM Linear algorithm.
- Almost all the accuracy increased in Subspace Discriminate prediction, but the average accuracy within this algorithm still lower than Bagged Trees and Boosted Trees.
- In almost all the cases, the algorithms perform their best with all the features included, therefore, for further analysis that all the features will be kept for prediction.

4.4 Optimize parameters

Test 4: Parameter fine-tuning in SVM and Ensemble algorithms.

The purpose of this test aims to improve the accuracy of the algorithms by fine-tuning the parameter values to the individual algorithm. Two parameters will be manually tuned in SVM algorithms: 1) Box Constraint and 2) Kernel Scale. Three parameters will be manually tuned in Ensemble algorithms: 1) Maximum number of splits, 2) Number of learners and 3) Learning rate. To eliminate time spent, only the best performing algorithms from previous test will be selected for fine tuning. Which are listed below:

- SVM: Cubic, Fine Gaussian and Medium Gaussian
- Ensemble: Boosted Trees and Bagged Trees

Figure 3 shows the result of SVM tuning with 5 different parameter box constraint values and 4 different kernel scale values. According to the bar chart that Fine Gaussians and Medium Gaussians both achieved its highest accuracy at 58,5% with the value of following parameters:

- Highest accuracy of Fine Gaussians = **58.5%** (k= 1, box constraint = 1)
- Highest accuracy of Medium Gaussians = **58,5%** (k = 1, box constraint = 1)

In general, the parameters fine-tuning of box constraint and kernel scale did not bring any significant improvement on the accuracy of the model prediction. In the first experiment, the obtained highest accuracy is 56.3% performed by Fine Gaussians, the increment of accuracy only increased 2.2%, which is not significant in this test.

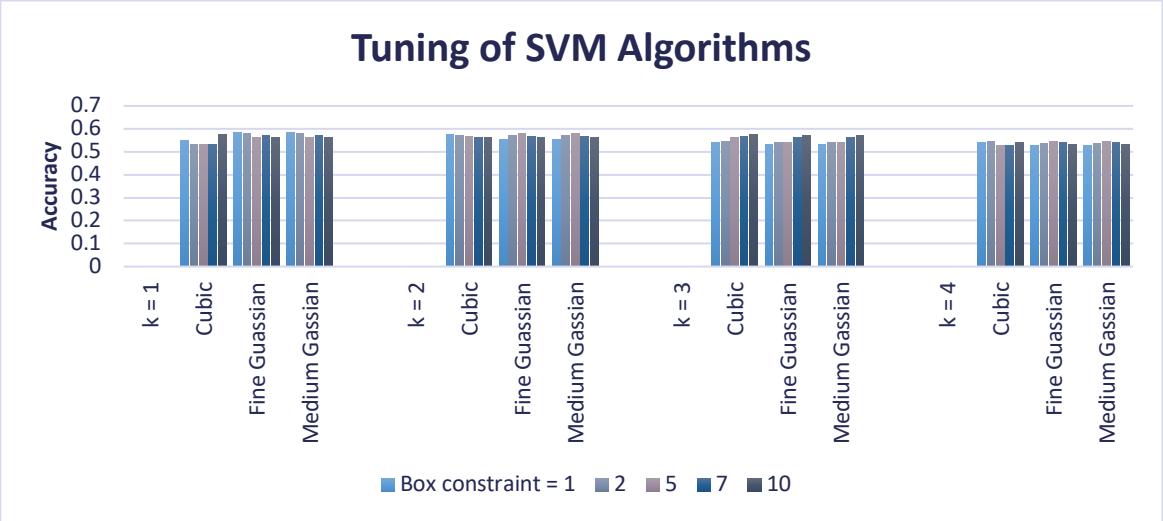


Figure 3. Accuracy in SVM algorithms with box constraint and Kernel scale.

Figure 4 shows the result of Bagged Trees tuning with 5 different values for number of learners and maximum number of splits. There is no clear improvement or trend to be found by increasing the number of learners or number of splits based on visual observation.

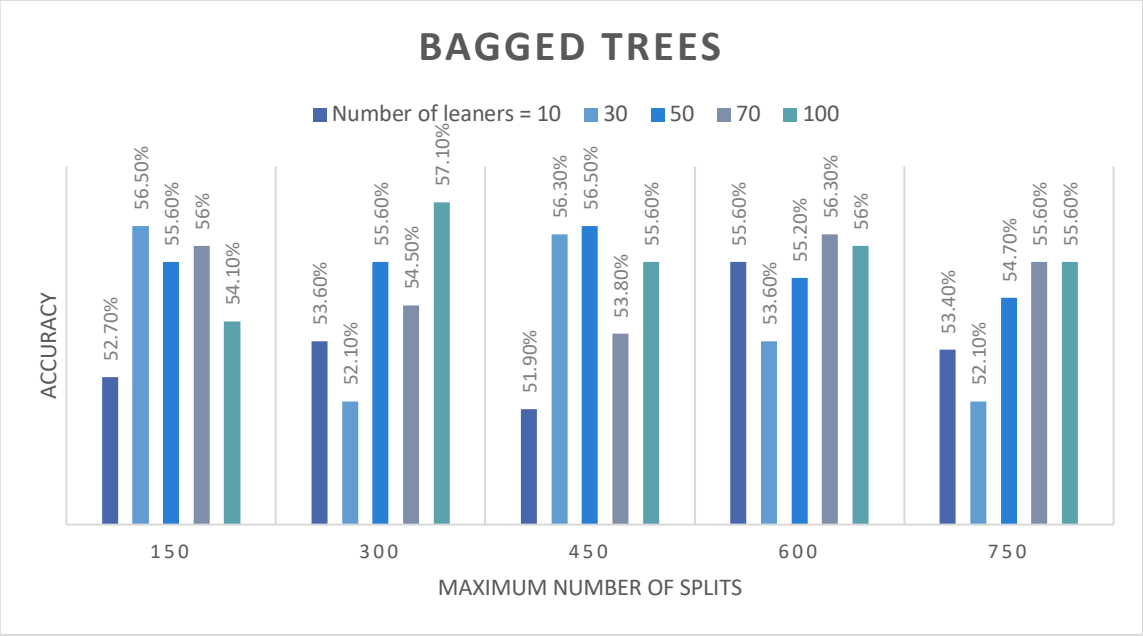


Figure 4. Tuning of Bagged Trees

Regarding to the chart (figure4) that Bagged Trees achieved its highest accuracy at 57,1 % with following value of parameters:

- Highest accuracy of Bagged Trees **= 57,1%** (Maximum number of splits = 300, number of learners = 100)

However, comparing with the parameter values of the first training with default settings in test 1 (Maximum number of splits = 454, number of learners = 30) the accuracy is only increased **2,4%.** It is not confident to conclude that by decreasing number of leaners or increase the maximum number of splits would yields a higher accuracy.

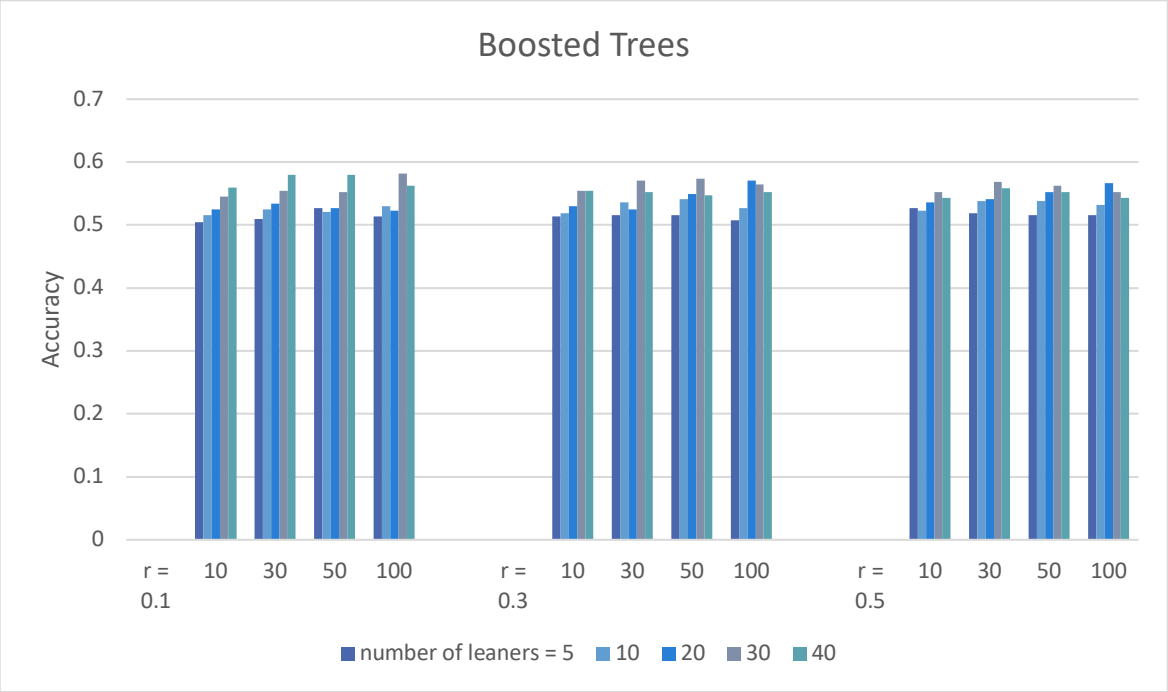


Figure 5. Tuning of boosted trees ensemble

Figure 5 shows the result of fine-tuning of Boosted Trees algorithms with 3, 4 and 5 different values for learning rate, number of splits and number of learners respectively. According to the accuracy in the first test, the highest accuracy model is 53,6 % with following parameter value:

- Number of leaners: 30
- Maximum number of splits: 20
- Learning rate: 0,1

It is noticeable that by increasing the number of splits and learners, the accuracy goes up slightly, however, learning rate does not seem to have any effect on the prediction. The Boosted Trees in test 4 achieved its highest accuracy at **58%** with following parameter value:

- Number of leaners: 40
- Maximum number of splits: 30 or 50
- Learning rate: 0,1

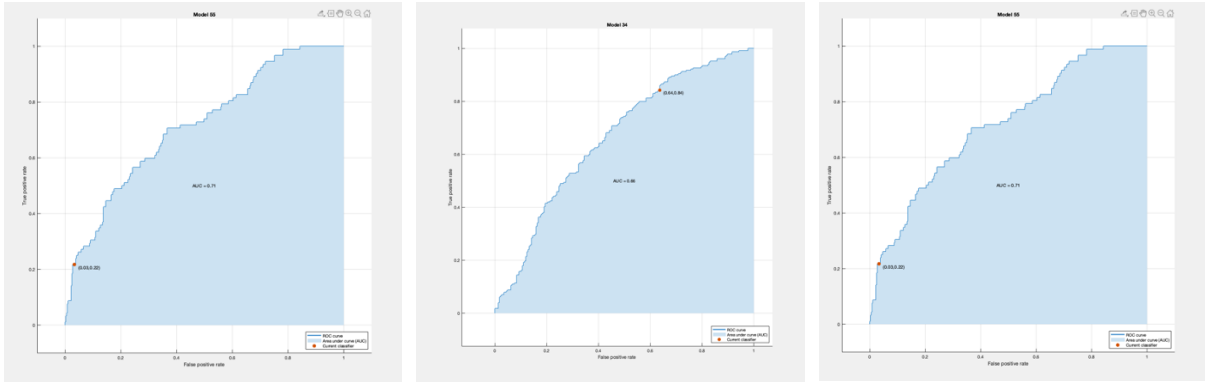
The accuracy of Boosted Trees after fine-tuning only increases 4,4%. The increment is relatively higher that bagged trees. This accuracy is achieved by increasing the number of learners by 10 and number of splits by 10 or 30.

4.5. Selection of the best algorithms

In this assignment, the model with highest accuracy was obtained in test 4 after parameter fine-tuning by using Fine Gaussians SVM and Medium Gaussians SVM algorithms. Figure 6 shows the Confusion Matrix and Roc Curve of Fine Gaussians SVM (same results as Medium Gaussians SVM algorithms). Figure 7 shows ROC curve for different classes.

The TPR value for class -1, 0 and 1 is 22%, 84% and 40% and AUC for class -1,0 and 1 is 71%, 66% and 71%.

Figure 6. Tuning of boosted trees ensemble



(a) Class -1 (b)Class 0 (c)Class 1

Figure 7. ROC curve for different classes

5. Conclusion

Intelligent buildings are more likely to be designed by integrating smart HVAC systems with large datasets generated by not only people, but also by computers and smart devices. This phenomenon of big data seems to continue its rapid growth. Machine learning contributes automated solutions that it can learn from data, and importantly, using data to answer questions or make effective predictions. Machine learning is no longer considered as a novel element in the offerings from IT or installation companies. It has been growing as one of the most expected functions or features in their products or services. In this assignment, supervised machine learning was applied in the field of designing personal conditioning systems. Different types of data were collected via smart sensors, data analysis & visualization were performed in Juypter Notebook and model training including parameter fine-tuning was performed in MATLAB.

Throughout the entire assignment, a sequence of performing machine learning with using the different algorithms (SVM and Ensembled algorithms) was formalized and executed. The models which score highest accuracy are Fine Gaussians SVM and Medium Gaussians SVM, this entire model training process raises the awareness of how difficult and time consuming to

achieve a suitable algorithm with high accuracy. One of the key takeaways is that in order to achieve a higher accuracy, the distribution of each class needs to be as balance as possible.

All in all, this course brought useful insights on thinking further about how machine learning can be applied and integrated in building services or HVAC system designs, which can assist designers or engineers in reaching new heights that never could have achieved before. I have learned a lot from this course, not only the theoretical frameworks of performing data analysis and visualization in both Jupyter and MATLAB, but also gain practical experience to quickly apply these methods to new machine learning projects.

Reference:

1. MathWorks, Statistics and Machine Learning Toolbox- User's Guide R2018a, 2018.
2. K. Katić, R. Li, J. Verhaart, W. Zeiler, Neural network based predictive control of personalized heating systems, Energy Build. 174 (2018). doi:10.1016/j.enbuild.2018.06.033.
3. J. Kim, Y. Zhou, S. Schiavon, P. Raftery, G. Brager, Personal comfort models: Predicting individuals' thermal preference using occupant heating and cooling behavior and machine learning, Build. Environ. 129 (2018) 96–106. doi:10.1016/j.buildenv.2017.12.011.
4. C. Dai, H. Zhang, E. Arens, Z. Lian, Machine learning approaches to predict thermal demands using skin temperatures: Steady-state conditions, Build. Environ. 114 (2017) 1–10. doi:10.1016/j.buildenv.2016.12.005.
5. Tape, T. (n.d.). AUC . Retrieved from http://gim.unmc.edu/dxtests/roc3.htm.
6. choose a classfier. (2019, November 1). Retrieved from https://nl.mathworks.com/help/stats/choose-a-classifier.html#bunt0n0-1.
7. A. Wagner, E. Gossauer, C. Moosmann, T. Gropp, R. Leonhart, Thermal comfort andworkplace occupant satisfaction - results offield studies in German low energyoffice buildings, Energy Build. 39 (2007) 758–769,http://dx.doi.org/10.1016/j.enbuild.2007.02.013.
8. W.J. Fisk, A.H. Rosenfeld, Estimates of improved productivity and health frombetter indoor environments, Indoor Air 7 (1997) 158–172,http://dx.doi.org/10.1111/j.1600-0668.1997.t01-1-00002.x.
9. K.W. Tham, H.C. Willem, Room air temperature affects occupants' physiology,perceptions and mental alertness, Bu