Article

Automating indoor weather adjustments

Aman Kumar Soni (11715307)

1. School of Computer Science Engineering,Lovely Professional University,Punjab 144411, India; aman.11715307@lpu.in(A.K.S.)

Abstract: In this paper, an approach to solving classification problems by using decision trees, Support Vector Machine, neural networks system is elaborated. The field of Information theory is used to select a set of important attributes that can be used to classify tuples. Data mining topics will be discussed and a good dataset of Occupancy Detection Dataset will be used to create a neural network and churn this system's performance. Accuracy of occupancy forecasts using data from light, temperature, humidity and CO2 sensors were analyzed with different types of calculations using an open-source system. closed during the stay. Generally, the best accuracy (from 95 to 99%) is obtained from LDA (Linear Discriminant Analysis), CART (Classified And Regression Trees) and RF (Random Forest). The results indicate that the right choice of features together with the correct classification model can have a significant impact on the accuracy of the prediction. Timestamp details are included in the models, and they usually increase the accuracy of the findings. Interestingly, using only one predictor (temperature) the LDA model was able to estimate the location with 85% accuracy and 83% in the two test sets.

Keywords: deep learning; LDA (Linear Discriminant Analysis), CART (Classified And Regression Trees) and RF (Random Forest), Decision Trees, Support Vector Machine, Neural Network

.

1. Introduction:

The true cut-off for residential acquisition recently estimated energy savings by 30% to 42%. Experimental studies reported that energy savings were 37% in and between 29-80% when residual data were used as input for HVAC control algorithms. Nowadays, the availability of sensor nodes is getting bigger and more complex, and the computing power of flexible systems determines to stay more a promising way to use low power with proper HVAC control and building lighting systems. Other residency programs include safety and the willingness to change the behavior of the residents. A system that can accurately detect the presence of residents without the use of a camera is very interesting due to privacy issues.

This study used data recorded from light, temperature, humidity, CO2 sensors, as a means of locating and digital camera for establishing a residential model for the monitoring model. The integration of these sensors is already found in many structures. Models trained and tested in this work are RF (Random Forest), GBM (Gradient Boosting Equipment), LDA (Linear Discriminant Analysis), and CART (Classified And Regression T Tree). These types of statistical classifications are already used in the open-source program Python. To our knowledge, the use and performance of RF, GBM and LDA models have not been reported in the scientific literature of human discovery.

This work builds on some research that showed that better occupancy detection could be achieved using high-resolution and accurate monitoring equipment. It also addresses model selection and entries by examining different feature combinations that have not been reported yet. Also, for the first time, the estimation time has been taken and included in the classification models. The task at hand is primarily about the acquisition of human residence time and not the task of developing a new mathematical learning algorithm. As a result, this section focuses more on the literature related to occupancy and sensory data.

2. Techniques and Approaches:

1. Decision Tree:

Decision Trees (DTs) are a high-quality unreadable learning technique used for classification and regression. The goal is to create a model that predicts the number of a target variable by learning simple decision rules derived from the filter the model or we can say derived from the data structure

So, the overall decision trees display a fusion of environmental symptoms. Each path from tree roots to a leaf corresponds to a combination of fitness tests, and the tree itself experiences these interactions (Mitchell, 1997, p.53). Until each leaf area is covered by non-sampling individuals as much as possible

General Form:

Choose a leafy area with a flawless sample. Replace that leaf blade with a test node that separates the measured sample that is presented in small empty sheets, according to the entropy calculation.

Specific Form:

Examine the attributes to add at the next level of the tree using an entropy calculation. Choose the attribute that minimizes the entropy.

2. Implementation Of Decision Tree in Python:

In Python, scikit-learn is a widely used library for implementing machine learning algorithms, Decision Tree is also available in the scikit-learn library and follows the same structure (Import library, object creation, fitting model and prediction). Let's look at the below code this has been performed on theOccupancy Detection chosen for this project:

```
#Import Library

import numpy as np
import matplotlib.pyplot as plt
from sklearn.tree import DecisionTreeClassifier
from sklearn.cross_validation import train_test_split

#Assumed you have, X (predictor) and Y (target) for training data set and
X_test(predictor) of test_dataset
# Create Decision Tree classification object

decisionTree    =    DecisionTreeClassifier(max_depth=3)
#Train the model using the training sets and check score
decisionTree.fit(X_train, y_train)
#Predicting           the           score
predictions=decisionTree.predict(X_test
)
print ("Accuracy of Decision Tree",accuracy_score(y_test,predictions))
```

Some advantages of decision trees are:

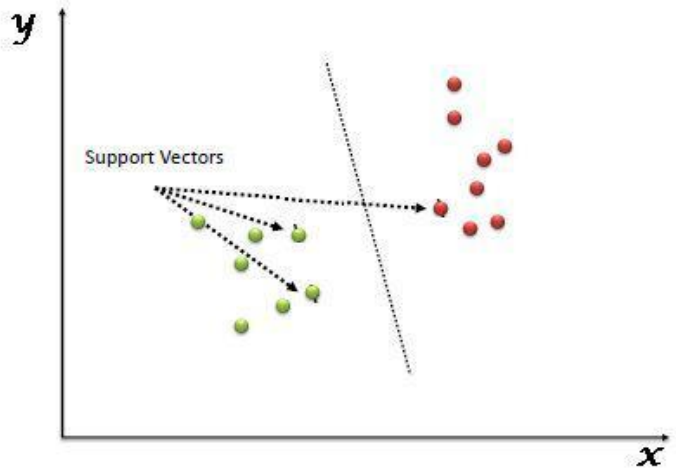
Easy to understand and translate. Trees can be seen. Requires minimal data correction. Some techniques require data editing, a dummy variable needs to be created and empty values removed. Indication in spite of that this module does not rest with the missing values. The cost of using a tree (e.g. forecast data) is logarithmic on the number of data points used to train the tree. Able to manage mathematical and phase/categorical information.

Other techniques are usually specialized in analyzing databases with a single variant. You can handle many output problems. It is possible to validate the model using a statistical test. That makes it possible to respond to the model's reliability. It works well even if its assumptions violate the original model in which the data were generated.

Some disadvantages of decision trees are:

Decision Tree learner can create sophisticated trees that do not properly process data. This is called excessive. Techniques such as pruning (not yet supported), setting the minimum number of samples needed for leaf area or establishing a complete tree depth are required to avoid this problem. Decision trees can be scalable because small variations in the data may result in the production of a completely different tree. This problem is reduced by using decision trees within the ensemble. The problem of learning the right decision tree is known to be NP-perfect under several aspects of doing good even in simple terms. As a result, effective tree learning algorithms are based on heuristic algorithms such as the greedy algorithm in which direct decisions are made at each location. Such algorithms can ensure the return of a global decision tree. Decision Tree can create discriminating trees when other classes rule. It is therefore recommended to estimate the dataset before the fit and decision tree.

3. Support Vector Machine:



The "Support Vector Machine" (SVM) is a supervised learning algorithm that can be used for both splits or undo challenges. However, it is widely used in classification problems without speculation in the data distribution. In this algorithm, we plot each data element as a point in the n-dimensional space (where the value of n is the number of elements you have) for the value of each element that is the sum of a particular combination. Subsequently, we perform the classification by finding a hyperplane that divides the two phases correctly (see image above).
Support Vectors simply include individual observations. The vector support machine is a border that better separates these two sections (hyperplane/line).

Working of Support Vector Machine:

We got a brief idea about the process of dividing these two phases into a larger planet. Let's understand how to find the right hyperplane?

A support vector machine creates a hyperplane or set of hyperplanes in an elevated or infinite space, which can be used for division, retrieval or other tasks. In fact, good classification is achieved by the hyperplane with the highest distance at the training data points closest to any class (called functional margin), because it is usually larger until it measures the generalization error of the classifier.

Implementation Of Support Vector Machine in Python:

In language Python scikit-learn is a widely used library for implementing machine learning algorithms, SVM is also available in the scikit-learn library and follows the same structure (Import library, object creation, fitting model and prediction). Let’s look at the below code this has been performed on the Occupancy Detection chosen for this project:

```
#Import Library
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cross_validation import train_test_split
from sklearn.svm import SVC
#Assumed you have, X (predictor) and Y (target) for training data set and
X_test(predictor) of test_dataset
# Create SVM classification object
clfs=SVC(kernel='rbf',gamma=2, C=1)
# there is various option associated with it, like changing kernel, gamma and
C value. Will discuss more # about it in next section.Train the model using
the training sets and check score
clfs.fit(X_train, y_train)
#Prediction Of Score
y_pred=clfs.predict(X_test)
zxc=accuracy_score(y_test,y_pred)
print('Accuracy of SVM Classifier', zxc)
```

We tried to plot a graph for SVM .The code snippet for the same is :

```
X.shape[1]
x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1 xx,
yy = np.meshgrid(np.arange(x_min, x_max, h),
                      np.arange(y_min, y_max, h))

plt.subplot(1, 1, 1)
plt.subplots_adjust(wspace=0.4, hspace=0.4)
Z = clfs.predict(np.c_[xx.ravel(), yy.ravel()])

# Put the result into a color plot
Z = Z.reshape(xx.shape)
plt.contourf(xx, yy, Z, cmap=plt.cm.coolwarm, alpha=0.8)
# Plot also the training points
plt.scatter(X[:, 0], X[:, 1], c=y, cmap=plt.cm.coolwarm)
plt.xlabel('Sepal length')
plt.ylabel('Sepal width')
plt.xlim(xx.min(), xx.max())
plt.ylim(yy.min(), yy.max())
plt.xticks(())
plt.yticks(())
plt.title('SVM')
plt.show()
```

The advantages of support vector machines are:

It works well in higher dimensional spaces. It is effective in cases where the sample size is larger than the number of samples.

It uses a set of training points for the decision function (called support vectors), so it also works memory. Varies: Different Kernel functions can be specified for the decision function. Standard kernels are provided, but it is also possible to specify custom configurations.

The disadvantages of support vector machines include:

If the number of features is greater than the number of samples, the method may give incorrect performance.

SVMs do not provide exact estimates of the probability, this calculation uses two strong 5-factor returns (see scores and opportunities, below).

4. Neural Networks:

Neural networks have been used successfully in a variety of supervised and unsupervised learning applications. Neural-network methods are not used for mining-data operations, however, because they often produce incomprehensible models and require a considerable amount of training time. In this article, we outline the neural-network learning capabilities they are capable of to produce understandable models, and that does not require excessive training time.Nural networks are a machine learning tool that attempts to mimic the learning pattern of natural neological networks. Biological neural networks connect neurons and dendrites that receive input, which, based on this input, transmits the output signal through the axon to another neuron. We will try to simulate this process using Artificial Neural Networks (ANN), which we will simply look at as neural networks from now on. The process of building a neural network begins with a very basic form, a single perceptron.

Anatomy Of Neural Network:

1. The Neural Network map is a set of input locations in a set of output locations.

2. The number of input nodes and output nodes varies.

3. The network itself is associated with a competitive number of locations with conflicting topology.

The neural network used in the paper is used to extract features by requiring the network to learn how to retrieve input data from output locations using a different number of hidden locations. The network can be trained to optimize input maps to the corresponding output values by providing a training set. The network is regularly tested and engineered to produce the right result. The output generation by the neural network is accomplished by the □ring values from the locations. Installation is transferred to an input layer that can activate the internal layers, which then activates the output flow, and finally results.

Neural Network follows the topology of Back Propagated Network in which inputs are put through a 'hidden layer 'before the Output Layer.All the nodes are connected between the layers.

The supervised training of Back Propagated Network includes :
1. Desired output of training inputs.
2. Error= Difference Between Desired and actual Output.
3. Change Weight for more accuracy.
4. Changing the output layer by propagating back to previous layer.
5. Hidden Layers and Number of neurons in the network.

5. Training the model:

Now is the time to train our model. SciKit Learn makes this much easier, by using calibration features. In this case we will import our estimator (Multi-Layer Perceptron Classifier model) from the neural_network library of SciKit-Read.

```
from sklearn.neural_network import MLPClassifier
```

Next we build a model example, there are many parameters to choose from that you can define and customize here, we will simply define the hidden_size_file. In this parameter you pass through the discovery involving the number of neurons you want in each part, where the nth entry in the triangle represents the number of neurons are in the nth layer of the MLP model.

```
mlp = MLPClassifier(hidden_layer_sizes=(6,6,6))
```

Now that the model has made that we can fit the training data to our model, remember that this data has already been processed and scaled:

```
mlp.fit(X_train,y_train)
```

6. Predictions and Evaluation:

Now that we have a model it is time to use it to get predictions. We can do this easily with the predict() method off of our fitted model:

```
predictions = mlp.predict(X_test)
```

Now we can use scikit learn's built in metrics such that a classification report and confusion matrix to evaluate how well our model performing:

```
cnf_matrix =confusion_matrix(y_test,predictions)
print ("Accuracy of MLP Neural network",accuracy_score(y_test,predictions))
print ("")
print(classification_report(y_test,predictions))
```

To plot the confusion Matrix, a function was made to plot the array with visualization:

```
def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):
    """
    This function prints and plots the confusion matrix.
    Normalization can be applied by setting `normalize=True`.
    """
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)
```

7. Information Theory:

Information theory measure information in bits.

Information gain=(Entropy of distribution before the split)–(entropy of distribution after it)

Information gain is the amount of information that's gained by knowing the value of the attribute, which is the entropy of the distribution before the split minus the entropy of the distribution after it. The largest information gain is equivalent to the smallest entropy.
The entropy (very common in Information Theory) characterizes the (im)purity of an arbitrary collection of examples.

Entropy Calculations:

If we have a set with k different values in it, we can calculate the entropy as follows:
Entropy(p1,p2,…,pn)=−p1log(p1)−p2log(p2)− −pnlog(pn)
If all instances in a group were known to be all in the same class, then the information value of being told the class of a particular instance is Zero

If instances are evenly split between classes, then the information value of being told the class of a particular instance is Maximized.

Information theory measures the value of information using "***entropy***", which is measured in "***bits***".

For example, if we assume in our Occupancy Dataset and take a single attribute Temperature,let's take the dataset between 19 and 21 then let's find its Entropy and Information Gain :

Child Entropy*= -12083/12568 [log(12083/12568)/log 2] - 485/12568 [log(485/12568)/log2]*

=0.235789942

Now, let's find the weight of occupancy dataset who has temperature between 19 and 21 as follows :

Weight of each value*= -P(Customers with Occupancy Temperature between 19 and 21) * [log P(Customers with Occupancy Temperature between 19 and 21)/log 2]*

*= -12568/20560 * [log (12569/20560)/log 2)*

= 0..434041958

Weighted Entropy *= P(Customers with Occupancy Temperature between 19 and 21) * Child Entropy*
*= 12568/20560 **

0.235789942

=0.144141641

After all the calculations

Information Gain *=Entropy of all the data - Entropy Of Child Expected Data .*
Here, We found Entropy of all the data was and Conditional Entropy of Temperature set between 19 and 21 is 0.530942508

So,Information gain for Credit History =0.779836704-0.530942508 =0.248894.

8. Approach:

We followed the approach of **Cross-Industry Standard Process** for Data Mining (CRISP-DM) used commonly by data mining experts for Occupancy Detection Dataset.

Business Understanding:

To determine how several natural conditions can affect the occupancy of an office room. If the Occupancy is equal to one then occupied and if it is equal to 0 then No Occupancy status. Analyzing which attribute contributes to the office room. Importing Data from .txt files into Jupyter notebook and concatenating to increase dataset rows. Especially for Neural network where increased no. of training dataset is required to train the machine in regards to supervised learning.

Data Understanding:

The dataset consists of 20560 readings with attributes like Temperature, Date, Relative Humidity, Light, CO2 , Humidity Ratio and Occupancy.

Attribute:-
date *time year-month-day hour:minute:second*

***Temperature**, in Celsius*
***Relative Humidity**, %*
***Light**, in Lux*
***CO2**, in ppm*
***Humidity Ratio**, Derived quantity from temperature and relative humidity, in kgwater-vapor/kg-air*
***Occupancy**, 0 or 1, 0 for not occupied, 1 for occupied status*

The dataset was created from readings and findings found for 9 days between 2nd and 18th Feb of 2015. So day is not a good variable to determine occupancy status. When there is no daylight almost always the occupancy is 0.

Data Prepration:

The dataset would be subjected to a random state of 0.1 to split the data. The training dataset would be 18504 rows and the test dataset would be 2056 rows. Any abnormalities in the data have been cleaned out. Out of 20 attributes Date, the timestamp is not very useful in regards to predicting occupancy status, therefore this attribute would be removed from further classification. No abnormalities were found in other attributes. Pair plots were drawn between attributes to find patterns. The predicted values are binary and not continuous.

Out[1]:

	Temperature	Humidity	Light	CO2	HumidityRatio	Occupancy
1	23.18	27.2720	426.0	721.25	0.004793	1
2	23.15	27.2675	429.5	714.00	0.004783	1
3	23.15	27.2450	426.0	713.50	0.004779	1
4	23.15	27.2000	426.0	708.25	0.004772	1
5	23.10	27.2000	426.0	704.50	0.004757	1

Data would look like the above table for most part of the modelling.

Modeling:

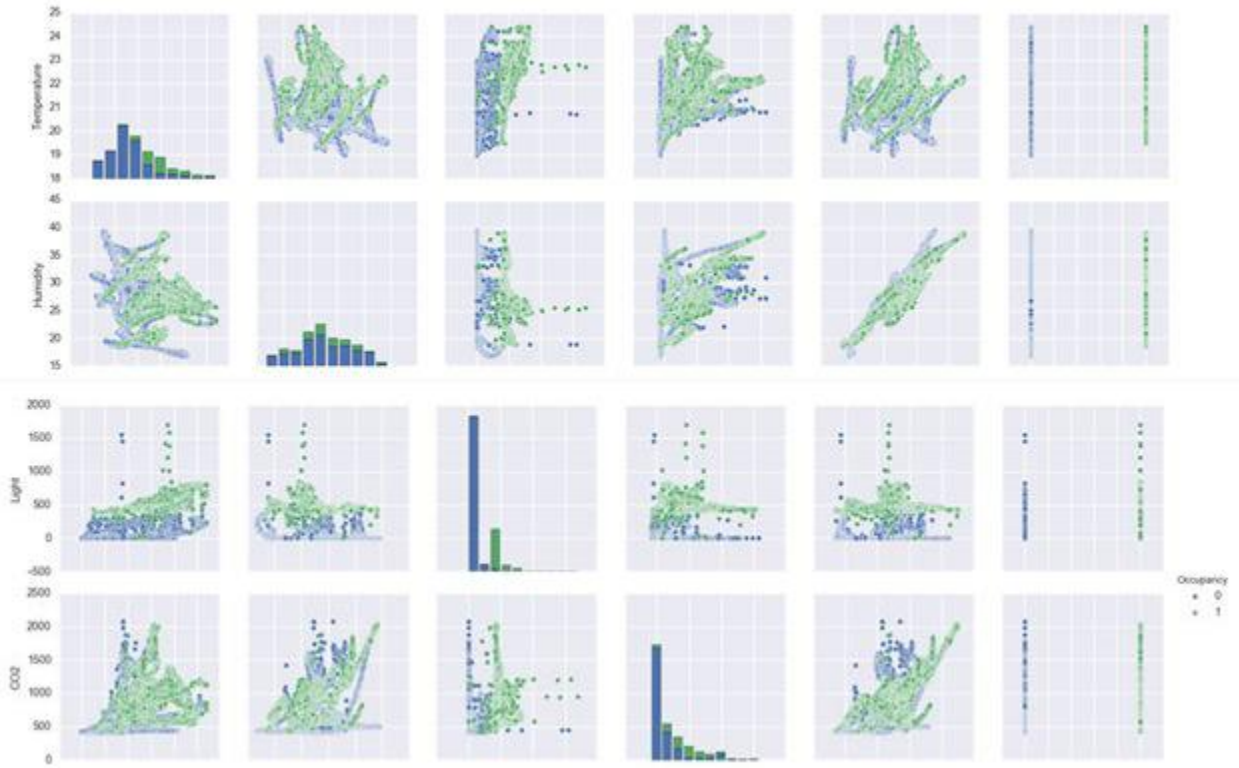
Support Viveors builds a hyperplane or a set of hyperplanes in a high or infinite space, which can be used for categorization, grouping or other functions. It applies to both separate and continuous phases and outputs. In this approach, we divide a population or sample into two or more sets (or subordinate sets) based on the most important partition/divider in the input variables, Neural networks are machine learning algorithms that attempt to mimic the learning pattern of natural networks.In supervised mining operations such as classification, it is common to use error rates as quality measures for data mining types. Therefore, we typically separate the dataset into train and

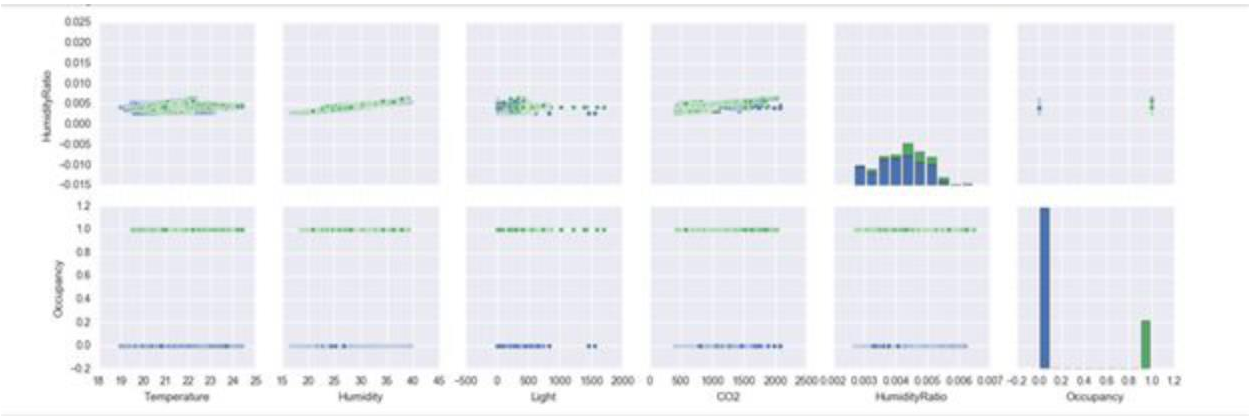
test sets, build the model on the train set, and estimate its quality on the separate test set. The original Datasets were converted to Data training sets, testing sets. A primary component of the plan is determining how to divide the available dataset into training, test datasets.

Run the modeling tool like decision trees using scikit on the prepared dataset to create one or more models.For Decision tree classification, a decision tree figure was generated. The max depth was kept at 3, even 4 was found favorable anything else made the tree complicated. ROC was drawn for the Neural Network classifier. SVM plot was drawn for the SVM classifier. Finally, the Confusion Matrix was drawn for all 3.

Testing and Evaluation:

Hidden layers were kept at 6 to compare with 6 tuples (6 attributes) and so as not to complicate the model.





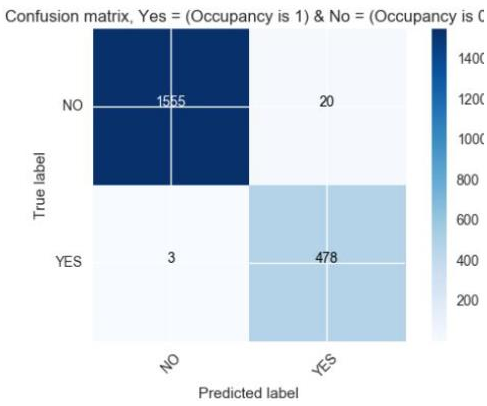


Fig a1. Confusion Matrix - DT

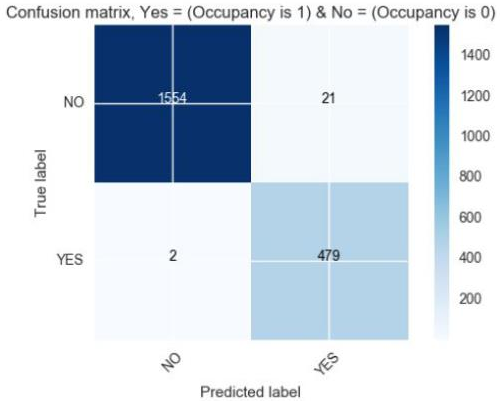


Fig a2. Confusion Matrix - NN

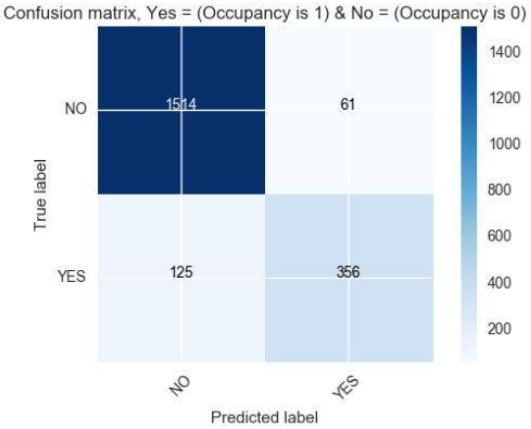


Fig a3. Confusion Matrix - SVM

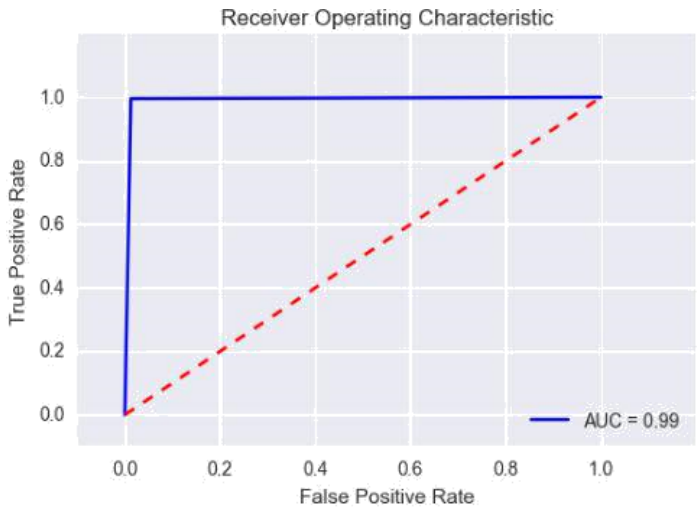


Fig b. ROC

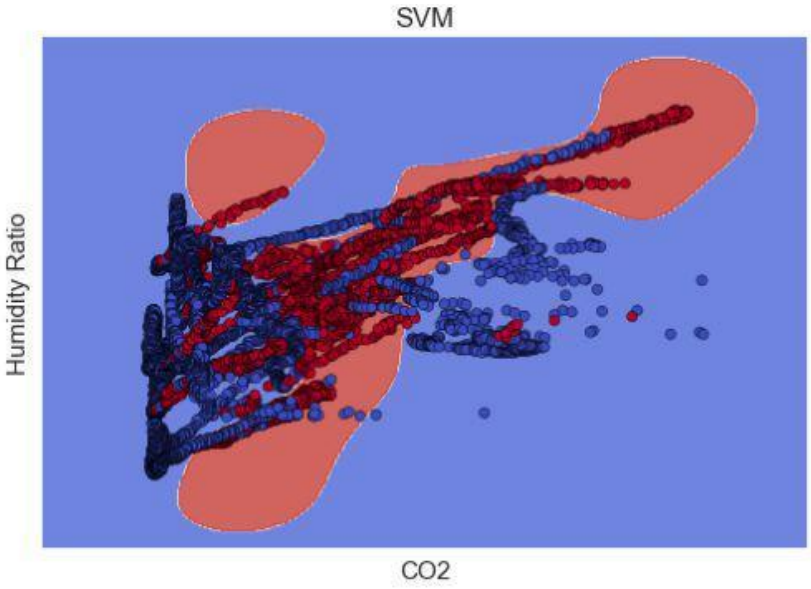


Fig c. SVM

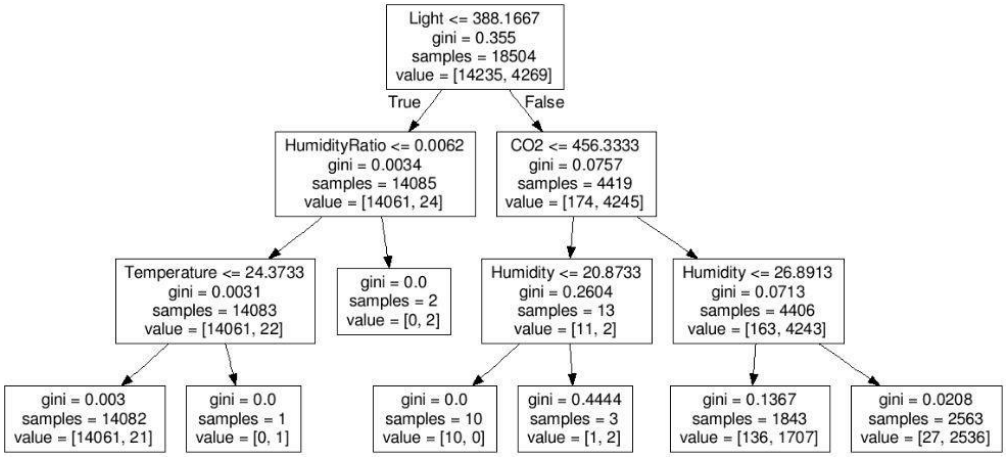


Fig d. Decision tree

Accuracy of Decision Tree = 0.988813229572.

Accuracy of MLP Neural network= 0.988326848249

	precision	recall	f1-score	support
0	1.00	0.99	0.99	1575
1	0.96	0.99	0.98	481
avg / total	0.99	0.99	0.99	2056

Accuracy of SVM Classifier= 0.90953307393

Deployment:

The results show highly positive results for Decision tree and Neural Network both having close to 99% accuracy score. Hence highly reliable to predict occupancy status for the office rooms. Supporting the Neural network's claim is the ROC that gives an outstanding area under the curve. And SVM classifier, with accuracy close to 91%, produced a lot of outliers as seen in Fig C. Due to this inconsistencies SVM may not be the best choice for the dataset. Probably collecting more data over a year would produce better results.

Result and Description of Test:

The experiments were conducted using the Occupancy Detection Dataset with the help of Decision Tree, Support Vector Machine, Neural network training. The light attribute seems to be the most influential as it tops the node in the Decision tree (Fig d.).

The last column on the classification is Ground-truth occupancy. Thus the prediction is to be done to classify each pattern good or bad.

No.	Attribute	G	G'
1.	Temperature	0.248894	0.210766
2.	Humidity	0.011974	0.006426

3.	Light	0.090242	0.0307446
4.	CO2	0.014051	0.134037
5.	Humidity Ratio	0.252705	0.261335
Average		**0.307562**	**0.12866172**

Table 1.5 Attribute gains of the Occupancy data set.

Accuracy of 99% achieved for Neural Network is very encouraging, this classifier along with Decision tree would be a great fit for the Occupancy Detection dataset.

3. Conclusion:

This work has shown that it is possible to achieve high accuracy in population determination of DT, CART and LDA types. The Decision tree trained model of temp, CO2 and W showed the worst performance in the test sets. This is most likely due to the presence of multiple variables included in the model. However, high accuracy was obtained while using only two predictors (temperature and light, light and CO2 and light and humidity, light and humidity measurements) and LDA models. In the first part of the data analysis, it can be inferred that these compounds exhibit a good separation gap between the occupancy condition.This research has also shown that most are a good practice to include information related to the time of day and week when you build models for classification. The increase in accuracy has been shown. Combining time information with heat proved to produce accuracy between 86-96% from 65-86% without it.

The study also found that using a light sensor would have higher accuracy in the CART models compared and reported much higher. And the CART temperature model had 67-87% accuracy compared to 55-65%. The suspected reasons for this difference are the high sensor accuracy, alignment and/or location of the sensors.

Although not covered in this work, the data suggest that for well-differentiated feature actors, the unsupervised separator model k-means can correctly detect the state of the habitat. Future work could also focus on predicting the number of people living there. One way to improve the accuracy of the settlement would be to use a stochastic model.

References:-

[1] V.L. Erickson, M.Á. Carreira-Perpiñán, A.E. Cerpa, OBSERVE: Occupancy-based system for efficient reduction of HVAC energy, in: Information Processing in Sensor Networks (IPSN), 2011 10th International Conference on, IEEE, 2011, pp. 258-269.

[2] V.L. Erickson, M.Á. Carreira-Perpiñán, A.E. Cerpa, Occupancy Modeling and Prediction for Building Energy Management, ACM Transactions on Sensor Networks (TOSN), 10 (3) (2014) 42.

[3] B. Dong, B. Andrews, Sensor-based occupancy behavioral pattern recognition for energy and comfort management in intelligent buildings, in: Proceedings of building simulation, 2009.

[4] J. Brooks, S. Goyal, R. Subramany, Y. Lin, T. Middelkoop, L. Arpan, L. Carloni, P. Barooah, An experimental investigation of occupancy-based energy-efficient control of commercial building indoor climate, in: Decision and Control (CDC), 2014 IEEE 53rd Annual Conference on, IEEE, 2014, pp. 56805685.

[5] J. Brooks, S. Kumar, S. Goyal, R. Subramany, P. Barooah, Energy-efficient control of under-actuated HVAC zones in commercial buildings, Energy and Buildings, 93 (2015) 160-168.

[6] I. Richardson, M. Thomson, D. Infield, A high-resolution domestic building occupancy model for energy demand simulations, Energy and Buildings, 40 (8) (2008) 1560-1566.

[7] S. Meyn, A. Surana, Y. Lin, S.M. Oggianu, S. Narayanan, T.A. Frewen, A sensor-utility-network method for estimation of occupancy in buildings, in: Decision and Control, 2009 held jointly with the 2009 28th Chinese Control Conference. CDC/CCC 2009. Proceedings of the 48th IEEE Conference on, IEEE, 2009, pp. 1494-1500.

[8] V.L. Erickson, Y. Lin, A. Kamthe, R. Brahme, A. Surana, A.E. Cerpa, M.D. Sohn, S. Narayanan, Energy efficient building environment control strategies using real-time occupancy measurements, in: Proceedings of the First ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Buildings, ACM, 2009, pp. 19-24

[9] C. Liao, P. Barooah, An integrated approach to occupancy modeling and estimation in commercial buildings, in: American Control Conference (ACC), 2010, IEEE, 2010, pp. 3130-3135.

[10] C. Liao, Y. Lin, P. Barooah, Agent-based and graphical modelling of building occupancy, Journal of Building Performance Simulation, 5 (1) (2011) 5-25.

[11]http://scikitlearn.org/stable/auto_examples/model_selection/plot_confusion_matrix.html#sphx-glr-auto-examples-model-selection-plot-confusion-matrix-py

[12] R.H. Dodier, G.P. Henze, D.K. Tiller, X. Guo, Building occupancy detection through sensor belief networks, Energy and Buildings, 38 (9) (2006) 1033-1043.

[13] K.P. Lam, M. Höynck, B. Dong, B. Andrews, Y.-S. Chiou, R. Zhang, D. Benitez, J. Choi, Occupancy detection through an extensive environmental sensor network in an open-plan office building, IBPSA Building Simulation, 145 (2009) 1452-1459.

[14] https://biesnecker.com/2015/03/27/visualizing-a-scikit-learn-decision-tree/

[15] Y. Agarwal, B. Balaji, R. Gupta, J. Lyles, M. Wei, T. Weng, Occupancy-driven energy management for smart building automation, in: Proceedings of the 2nd ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Building, ACM, 2010, pp. 1-6.