Applied Computational Intelligence Project

1st Shane Davey
Software Engineering
Lakehead University
Thunder Bay, Canada
scdavey@lakeheadu.ca

***Abstract*—This paper covers the process of designing and implementing multiple machine learning models that will be able to predict the number of people that are in a given room at any given time, each model will be compared by its performance and the best model will be selected as the final model for the problem.**

***Index Terms*—machine learning, supervised machine learning, artificial intelligence, linear regression, k-nearest neighbor regressor, decision tree regressor**

I. Introduction

A. Background

In this project we will be using a data set that was recorded for the purpose of predicting the number of people in a given room, this data set is publicly available for other people to use as well. The idea is to record numerous conditions within the room through the use of many sensors, then with each entry the number of people in the room is also recorded. Then this is repeated many many times until there is enough data for a machine learning model to actually be trained to predict the number of people when given only the conditions in the room.

B. Purpose

The purpose of trying to estimate the number of residents that are currently inside of a given room could be useful information for multiple reasons. Certain businesses like hotels or landlords that own apartment buildings to rent out rooms for other people to live in may benefit from this type of system. They may have some sort of resident limit or pricing based on number of residents per room, some people may be inclined to lie about how many people are actually staying in a given room and with this system they would be able to know for certain just how many people are actually living in that room. On top of that with the pandemic and its regulations it is very common that certain places will have to monitor the amount of people in a given space which this technology could also be used for.

C. Hypothesis

I believe that by the end of this process of training and testing the various modes we will be able to produce one that is highly accurate and is able to make correct predictions about the number of people. I think this because of the quality of the recorded data set, it covers a wide range of metrics that can be used for the predictions. By using this many sensors it will create more than enough variable that even if some do not have a great correlation to the number of residents surly others will. Then when we include all of them, the highly correlated ones and some of the not so much so ones the predictions will be just that much more accurate.

II. Literary Review

A. Machine Learning

Machine learning is a type of data analysis that has become a very common and popular tool in the industry. This method is capable of building a model that can make predictions of an output when given a set of inputs. The model is capable of doing so by using a large set of data in or to train it to make these predictions, the more accurate the predictions it can make the better the model is. This field is a branch of artificial intelligence as the model is using a set of training data in order to ”learn” how to make better and better predictions.

B. Supervised Machine Learning

Supervised machine learning is a sub category of machine learning where the data that is provided during the training phase includes the target value as well as the input values. There is also unsupervised machine learning where the variable that is to be predicted is not included in the training data. Supervised machine learning is important for this project as the three models that will be trained will all be supervised machine learning models.

C. Linear Regression

Linear regression is a very common and very simple type of supervised machine learning model, the goal is for the model to find a line of best fit between the input and output variables of the training data as shown in fig. 1. This line represents the linear relation ship between the independant and dependant variables and is then used to predict an output when given an input. There are also two types of linear regression including simple linear regression and multiple linear regression, this refers to the number of input variables. If there is only one input x for each output y that is a simple linear regression model,

$$y = bx + c$$

(1) where y is the output variable, x is the input variable, b is the slope and c is the intercept. However if there are more than one input for each output then it is considered multiple linear regression, using the same equation as equation (1) but for n number of inputs x.

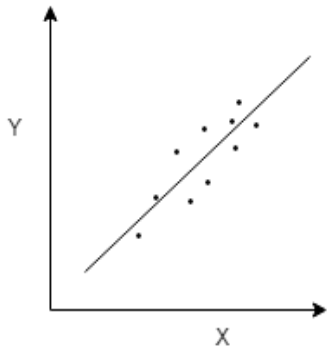

Fig. 1. Line of Best Fit.

D. Decision Tree Regressor

Decision trees can be used for both classification or regression in this case we will be using a decision tree regressor. Each decision tree will contain nodes, branches and leaves and will use a set of multiple binary rules as decision in order to make a prediction. The first node in the tree is the root node and it represents the entire set of data before the tree begins splitting it up. There are also decision nodes that represent a split in the tree, one decision node will branch of into multiple sub-nodes. The nodes that do not split up into multiple sub-nodes are called leaf nodes. A decision tree starts with the entire data set at the root node then as the data moves along through the tree it is split up by the decision nodes, after making many of these decisions and further splitting up the data the model will be confident enough to make a decision which will be a leaf node. An example of a decision tree can be seen in fig. 2.

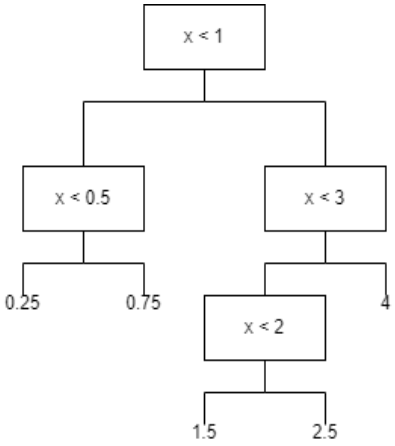

Fig. 2. Decision Tree Regressor.

E. K-Nearest Neighbor Regressor

The k-nearest neighbor (KNN) is another machine learning model that can be used for either classification or regression but for its purpose in this project will be used for regression. This model is simple however very effective, this model is able to make predictions by using feature similarity. The model will use the training points as a set of references, when a new point in introduced it will be associated with its nearest neighbors the neighbors closest on different sides of the new point can then be used to determine the value of the point in-between them. An example of a test point being added to a trained model can be seen in fig. 3 in this figure assume a value of k = 3, the 3 points within the circle will be used to determine the value of the new point.

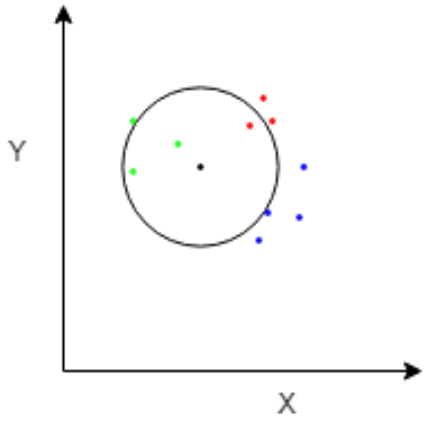

Fig. 3. KNN Regressor Making a Prediction.

F. Evaluation Metrics

The evaluation metrics for a machine learning model are what is used in order to determine the overall performance of a trained model. Once the model has been trained they are given a set of input variables that make up the data test set, the model will then make a prediction for each of the entries. How accurately and consistently the model is able to make predictions will determine how effective it is, the main metrics used to determine this are the goodness of fit (r-squared), mean squared error (MSE) and mean absolute error (MAE). First off r-squared is a measurement used for regression models that shows a percentage of how well the predicted data matches the actual data. MSE measures the average squared difference between the estimated data and the actual data and is calculated as,

$$MSE = (1/n)\sum_{i=1}^{n}(Y_i - Y_i')^2$$

(2) where n is the number of predictions generated from n data points, Y is the vector of observed values and Y' is the vector of predicted values. Finally we have MAE which measures a how close the predictions are to their actual outcomes and is given as,

$$MAE = (\sum_{i=1}^{n}(|Y_i - X_i|))/n$$

(3) where Y is the prediction, X is the actual value and n is again the number of data points.

III. Method

A. Variables

For this system there will need to be eight main variables in order to track and record all of the valuable data properly. First of all the data samples were taken using 5 different types of sensors that were placed in the environment, they consisted of, temperature, light, sound, CO2 and digital passive infrared (PIR). The readings for each of these sensors are going to need their own variable to track each of these measurements, 5 variables in total. Next we have another two variables that will be related to documenting and organizing the readings, every time a reading is done the entry will need to be recorded and indexed with a date/time and also a room number. Finally the last variable we need will be the one to track the amount of people that are in the room after all of the calculations, this will also be recorded with the date/time and room number.

B. Attributes

These variables will have some different attributes depending on the type of data being recorded or by how they will be used within the system. For example each of the sensors will record the variables in different ways the temperature will be degrees Celsius making it a double, light will be recorded in lux as an integer, sound will be in volts read from the amplifier output and will be a double, CO2 is recorded in parts per million (PPM) and will be an integer, and PIR will be a boolean detecting if there was movement. On top of just the sensors we have a few other variables like the date/time whether that is one variable or two it will be recorded as a date/time variable, next we have an integer to record the room number, and finally we have another integer to record how many people are in the room. The correlation between each of the variables can be seen in the heat map below in fig. 4 while the attribute description is in table. 1.

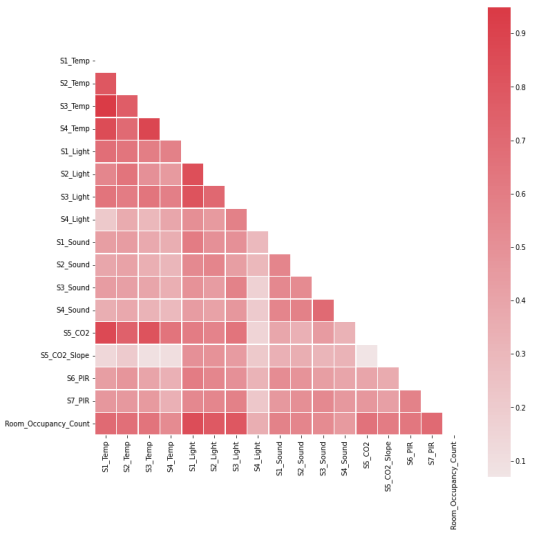

Fig. 4. Correlation between variables using heat map.

Attribute Name	Description
S Temp	Room temperature (in Celsius)
S Light	Amount of light (in lux)
S Sound	Loudness in the room (in DB)
S CO2	Amount of CO2 in the air (in PPM)
S PIR	Detects if there is movement (boolean)
Room Occupancy Count	Number of people in the room (int)

Table. 1. Attribute description of the dataset.

C. Samples

This data set contains 10129 samples which is perfect as there will be plenty of data to work with when training the model and is over the recommended minimum amount of 10000 for this project in particular. These samples were uploaded as of 2021-01-20 making it fairly recent, on top of that this type of project is a regression as the idea is to determine the amount of people in a particular space not just if there is people in the space or not.

IV. Data Curation and Conditioning

This part of the process is for getting the data reading and making sure that it is in the best state possible for use within the machine learning algorithms. So far from the repository we have gotten only the raw data that they have collected from their tests so there will need to be some changes made in preparation.

A. Reading in the Data

The first thing that we want to do is read in the data from the repository so that we can begin the preprocessing methods, first downloading the CSV file and then bringing it into the script and saving it as a dataframe. a trimmed example of the data table can be seen below in table. 2.

B. Checking Data

Now that we have the access the data the next thing that we want to do is to make sure that all the data is complete and nothing is missing or incorrect, the code for this can be seen in Appendix A. After running the code we get the following output ”All data is complete” so we know that we can move on to the next step.

C. Removing Unnecessary Data

Since all the data is good and ready to go we can start by removing the date column as it is not really useful in terms of predicting room occupancy however time will be important as the time of day between different days may give a good indication of how many people may be home at any given time. To do this all we have to do is a simple command to drop that column of the data table.

D. Converting Data Types

Now we still have one important column that is not numeric and that’s the time of day so that will need to be converted next. Converting all of the variables to numeric is very important as it makes working with all of the data much easier and a single action can easily be performed over the entire data

Date	Time	S1Temp	S1Light	S1Sound	S5CO2	S5CO2Slope	S6PIR	RoomOccupancyCount
2017/12/22	10:49:41	24.94	121	0.08	390	0.769231	0	1
2017/12/22	10:50:12	24.94	121	0.93	390	0.646154	0	1
2017/12/22	10:50:42	25.00	121	0.43	390	0.519231	0	1

Table. 2. Original Data From Repository.

frame such as manipulation, which we will be using later, since each category is of the same data type.

E. Removing Outliers

Now we have all of the important data in the right format we can now remove the outliers to clean up the data a little bit more. This step is important because if we leave the outliers in the data then the results can be skewed and reduce the accuracy and the consistency of the final model. The code for this operation can be seen in Appendix B.

After running the script we can see that all of the entries that were more than 3 standard deviations away have been removed and the table was reduced down to 8516 rows with more consistant data.

F. Normalizing Data

Now that the outliers are gone we can normalize the data, the purpose of this is to consolidate the ranges of all the columns to that they all follow a common scale but the differences in values are still maintained, the only one that will not be changed will be the room occupancy as that is the variable that the model will be trying to predict. The script used to normalize the data can been seen in Appendix C and an example of the normalized table can be seen in Table. 3.

The dataframe in its current state is now ready to be used in our machine learning model and based on the changes made it should perform much better than it would have before as well as be much easier to work with in terms of memory.

G. Linear Regression

The first model that will be used is the linear regression model, we can use the effectiveness of this model to help determine the effectiveness of the other models that will be tested afterwards. The first step in training any of the models is to split the dataframe into two parts, first the independant variables, everything except room occupancy count, and then the dependant variable which is room occupancy count. Once we have out variables the next thing to do is to split the data up into two parts the training data that will be used to train the model and the testing data that will be used to test how well the model can make predictions. It is important not to train the model with data that will be used for testing as it will inflate the result. Now that we have all of our data ready we can begin by creating a new linear regression model and fitting the training data to it. Once the model is training its time to give the model the independant variables of the test set and have it start making its predictions. After the predictions are made we use the dependant variables from the test set and compare them to the predictions made by the model. We then get the results of this in the form of our three performance metrics r-squared, MSE and MAE. The table of the results is shown in Table. 4 while the script for implementing the model is shown in Appendix C. All of the models follow a similar sequence when setting up, this process can be seen in the diagram of Fig. 5. There will also be a sanity check performed to verify the results of our metrics, we will tabulate the predictions with the actual results to make sure that each one is valid.

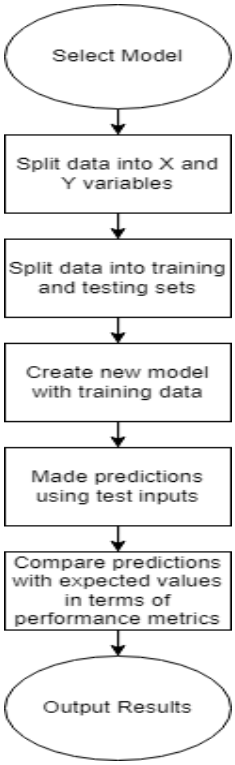

Fig. 5. Implementing Machine Learning Models.

H. Decision Tree Regressor

The next algorithm that we will be implementing is the decision tree regressor, this model will be implemented in the same way as the other models as shown in Fig. 5. however instead we will use DecisionTreeRegressor() to create the model. This model will be fitted with the same split of data and will be evaluated with the same metrics as before. The results of the model can be seen in Table. 5.

Decision tree regression is different from linear regression in that it has hyper parameters that can be used to tune the model in order to make it function more effectively. In hyperparameter tuning you give a list of variables for each of the hyperparameters, the script will then test the model with every combination of parameters and then output which ones

Time	S1Temp	S1Light	S1Sound	S5CO2	S5CO2Slope	S6PIR	RoomOccupancyCount
0.451202	0.000329	0.001441	0.000041	0.004554	0.000049	0.00004	1
0.451560	0.000329	0.001441	0.000051	0.004554	0.000048	0.00004	1
0.451908	0.000330	0.001441	0.000045	0.004554	0.000046	0.00004	1

Table. 3. Data After Normalization.

r-squared	MSE	MAE
0.87058871	0.01560613	0.05517887

Table. 4. Linear Regression Prediction Performance.

r-squared	MSE	MAE
0.96494550	0.00422733	0.00140911

Table. 5. Decision Tree Regression Prediction Performance.

will be the best for that situation. An example of this process is shown in Appendix D.

The hyperparameters we will be using for tuning this model include, splitter, max depth, minimum leaf samples, minimum weight leaf fraction, maximum features, and maximum leaf nodes. After running the script we get the result with the best parameters to use then perform the same implementation sequence as in fig. 4 but with the given parameters. The new performance of the model after tuning can be seen in Table. 6 while the summary of hyperparameters can be seen in table 7. Finally we perform the same sanity check to verify that the results make sense based on the given results.

r-squared	MSE	MAE
0.332200155	0.080532167	0.086920298

Table. 6. Tuned DTR Prediction Performance.

Attribute Name	Description
Max depth	3
Max features	auto
Max leaf nodes	none
Min leaf samples	1
Min leaf weight fraction	0.1
Splitter	best

Table. 7. Summary of Best Hyperparameter Settings.

I. K-Nearest Neighbor Regressor

The final algorithm that will be tested is the k-nearest neighbor model, just like the last two models this one will use the same data but be created with KNeighborsRegressor(). The results of the original model can be seen in Table. 8.

r-squared	MSE	MAE
0.97912310	0.00251761	0.0018788163

Table. 8. KNN Prediction Performance.

Again we want to tune the model in order to get the best performance possible with the model, the process will be the same as tuning the previous model however we will be using different hyperparameters specific to KNN. These hyperparameters include leaf size, number of neighbors and p, the results after tuning the model will be shown in Table. 9, while the summary of hyperparameters can be seen in table 10.

r-squared	MSE	MAE
0.9795905	0.0024612	0.0017848

Table. 9. Tuned KNN Prediction Performance.

Attribute Name	Description
Leaf size	10
Number of neighbors	5
p	1

Table. 10. Summary of Best Hyperparameter Settings.

Lastly all we need to do is perform the same sanity check as the previous algorithms to double check that the results we got before and after tuning are correct.

V. Finalizing the Best Model

The comparison of the performance for each of the models including before and after tuning can be seen in table. 11. First we have the linear regression model followed by DCT before and after tuning then finally KNN before and after turning respectively.

r-squared	MSE	MAE
0.8705887	0.0156061	0.0551788
0.9649455	0.0042273	0.0014091
0.3322001	0.0805321	0.0869203
0.9791231	0.0025176	0.0018788
0.9795905	0.0024612	0.0017848

Table. 11. Tuned Model Prediction Performance.

VI. Conclusion/Discussion

In conclusion based on the performance of each algorithm we can see that the tuned KNN was the best for r-squared and MSE, but the original DCT model was the best for MAE. There is also in interesting result between the DCT before and after tuning as the hyperparameters that we supposed to be the best made the model perform much worse across all metrics. All in all over all we also know that the Tuned KNN model outperformed the other 4 models being the best in two metrics and coming in a close second for the third one.

These results are not surprising as KNN models are known for being able to make very accurate predictions when trained correctly even compared to the other models used where the DCT model typically performs better when it comes to classification problems. Also the pipeline that produced the best results in this case for the KNN was a leaf size = 10, n neighbors = 5 and p = 1. Another way to compare the models is based on the execution time for each prediction to compare how quickly the models can make a prediction as in certain circumstances this may be important. Using the script shown in Appendix E we were able to calculate the execution time per prediction for each of the models the results are shown in table. 12.

Linear Regression	Decision Tree	K-Nearest Neighbor
3.3553363081e-06	2.0737468295e-06	2.091695965e-04

Table. 12. Comparing Execution Time Between Models.

Based on the table we can see that the tuned KNN model was far superior in terms or execution time as well furthering it as the best choice among the models.

The training strategy for these models was to use a split of test size = 0.25 in order get more out of the training portion to help make the model even more accurate. We didn't want to go any lower than that however for the test size since you also want plenty of data to test with so the sample size of predictions will be much larger. As for the hyperparameter part the goal was to try a good range of variables and not have it be to limited as we may miss a better solution. The main issue with this is the time complexity of testing that many combinations at first it was taking many hours to complete the execution. This was little unreasonable as many times it would time out before completion so the total range of the variables was kept the same but instead we tested less variables within the range. All of these methods were used in conjunction for the regularization of the model in order to get the absolute best and most consistent performance possible.

VII. Future Work

As for future work in this project I think it would be great to be able to actually put the final model to the real test. This would entail quite a bit of set as you would have to recreate the environment in which the original data was actually recorded which would take quite a few sensors. After setting up these sensors we could read in their data in real time as test data, then using the same model we could make real time predictions about how many people are in the room. This data could then be recorded and compared with the accuracy of the the test data that was already performed in order to see how well the model would preform in a real world situation.

VIII. Acknowledgment

Thank you to Adarch Pal Singh and Dr. Sachin Chaudhari for providing the data set used for this project on the machine learning repository, without it this would not have been possible. Also professor Akilan for his feed back throughout the implementation of this project that only help to further enhance the effectiveness of the model as well as the quality of the report.

IX. References

Data set: Adarsh Pal Singh, Vivek Jain, Sachin Chaudhari, Frank Alexander Kraemer, Stefan Werner and Vishal Garg, Machine Learning-Based Occupancy Estimation Using Multivariate Sensor Nodes, in 2018 IEEE Globecom Workshops (GC Wkshps), 2018

X. Appendix

A.

```
if␣np.where(pd.isnull(room_df))␣==␣NullArrayProxy:␣#check␣for␣empty␣entries
␣␣print␣("Some␣data␣may␣be␣missing")␣#if␣there␣is␣data␣missing␣notify␣user
else:␣#if␣all␣the␣data␣is␣there
␣␣print␣("All␣data␣is␣complete")␣#notify␣user
␣␣
```

B.

```
␣zScores␣=␣np.abs(zscore(room_df))␣#get␣the␣z␣scores␣off␣all␣the␣data␣in␣the␣dataframe
goodData␣=␣(zScores␣<␣3).all(axis=1)␣#filter␣out␣all␣of␣the␣outliers
refined_df␣=␣room_df[goodData]␣#remove␣all␣of␣the␣outliers␣from␣the␣dataframe
refined_df␣=␣refined_df.reset_index()␣#reset␣the␣index␣to␣avoid␣gaps
refined_df␣#view␣refined␣data␣frame
```

C.

```
#Linear␣regression
X␣=␣normalized_df.drop('Room_Occupancy_Count',␣axis=1)␣#create␣independant␣variables
Y␣=␣normalized_df['Room_Occupancy_Count']␣#create␣dependant␣variable
X_train,␣X_test,␣Y_train,␣Y_test␣=␣train_test_split(X,␣Y,␣random_state␣=␣0)␣#split␣data
LRM␣=␣linear_model.LinearRegression().fit(X_train,␣Y_train)␣#create␣LR␣model
prediction␣=␣LRM.predict(X_test)␣#make␣a␣prediction␣with␣LMR␣and␣testing␣data
test_MAE␣=␣mean_absolute_error(Y_test,␣prediction)␣#calculate␣the␣MAE
test_MSE␣=␣mean_squared_error(Y_test,␣prediction)␣#calculate␣the␣MSE
test_r2␣=␣r2_score(Y_test,␣prediction)␣#calculate␣goodness␣of␣fit

print(test_MAE)␣#return␣MAE
print(test_MSE)␣#return␣MSE
print(test_r2)␣#return␣goodness␣of␣fit
```

D.

```
#Decision␣Treehyper␣perameter␣tuning
from␣sklearn.model_selection␣import␣GridSearchCV
#create␣all␣the␣hyperparameters␣that␣will␣be␣tested
parameters={"splitter":["best","random"],
␣␣␣␣␣␣␣␣␣␣␣"max_depth"␣:␣[1,3,5,7,9,11,12],
␣␣␣␣␣␣␣␣␣␣␣"min_samples_leaf":[1,2,3,4,5,6,7,8,9,10],
␣␣␣␣␣␣␣␣␣␣␣"min_weight_fraction_leaf":[0.1,0.2,0.3,0.4,0.5],
␣␣␣␣␣␣␣␣␣␣␣"max_features":["auto","log2","sqrt",None],
␣␣␣␣␣␣␣␣␣␣␣"max_leaf_nodes":[None,10,20,30,40,50]␣}

tuning_model=GridSearchCV(DTM,␣param_grid=parameters,cv=10,verbose=3)␣#create␣the␣tuning␣model

tuning_model.fit(X_train,Y_train)
```

E.

```
start␣=␣time.process_time()
result␣=␣LRM.predict(X_test)
end␣=␣time.process_time()
total␣=␣end␣-␣start
per_prediction␣=␣total/2129
print(per_prediction)
```