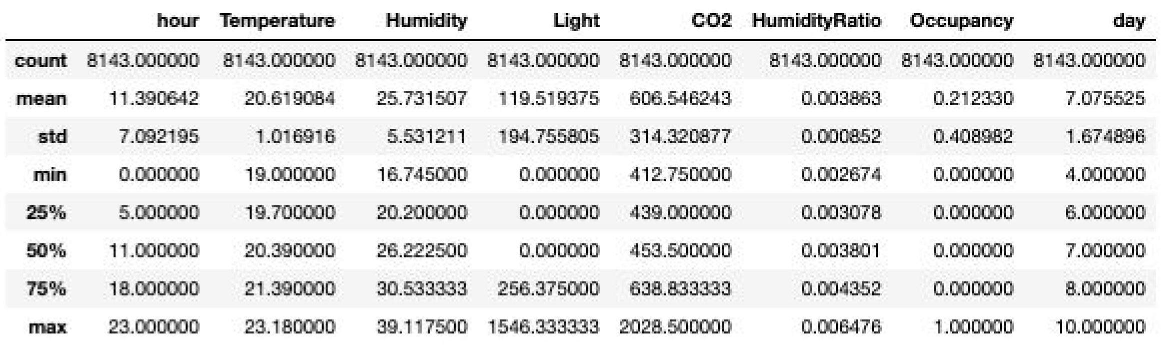
Mark Mersnik
ID: 010455185

EE 257 Final Project Report: Occupancy Detection

Abstract: For our final project, we were asked to utilize a dataset from the UCI machine learning repository in order to test the skills we have learned in this course. The dataset that will be discussed in this report is titled “Occupancy Detection”. It contains data about a room's environment and whether an occupant is present. The input parameters are various characteristics of the room and the output is whether an occupant is present. In order to predict whether an occupant is present, I fit five classification models. The models are logistic regression, linear/quadratic discriminant analysis (LDA/QDA), k-nearest neighbor (KNN), and random forest. Every one of these models proved to be very accurate at predicting occupancy, boasting a mean-squared error rate of less than 1.25% for each.

Data Set Description: The data from the “Occupancy Detection” dataset consists of 6 input parameters and a single output with 8143 data points each. The input variables are date, temperature, humidity, light, CO2, and humidity ratio. Each of the inputs are a floating point number, aside from the date parameter. The date was an object type which I had to convert to datetime64 type in order to analyze each piece (e.g. year, day, hour, etc.). However, when I did extract each part of the date, I found that all of the data was taken over the course of 7 days. This means that only the day and hour part of the date were useful for my models. As for my output variable, since this is a classification problem, it has an integer value of either 0 or 1, depending on if an occupant is present (0 for no occupant and 1 for an occupant). None of my data points were found to be missing so my dataset could be used as is. As for the attributes of my data, a table is shown in figure 1 describing each.



	hour	Temperature	Humidity	Light	CO2	HumidityRatio	Occupancy	day
count	8143.000000	8143.000000	8143.000000	8143.000000	8143.000000	8143.000000	8143.000000	8143.000000
mean	11.390642	20.619084	25.731507	119.519375	606.546243	0.003863	0.212330	7.075525
std	7.092195	1.016916	5.531211	194.755805	314.320877	0.000852	0.408982	1.674896
min	0.000000	19.000000	16.745000	0.000000	412.750000	0.002674	0.000000	4.000000
25%	5.000000	19.700000	20.200000	0.000000	439.000000	0.003078	0.000000	6.000000
50%	11.000000	20.390000	26.222500	0.000000	453.500000	0.003801	0.000000	7.000000
75%	18.000000	21.390000	30.533333	256.375000	638.833333	0.004352	0.000000	8.000000
max	23.000000	23.180000	39.117500	1546.333333	2028.500000	0.006476	1.000000	10.000000

Figure 1: The attributes of the “Occupancy Detection” dataset.

Data Set Visualization:

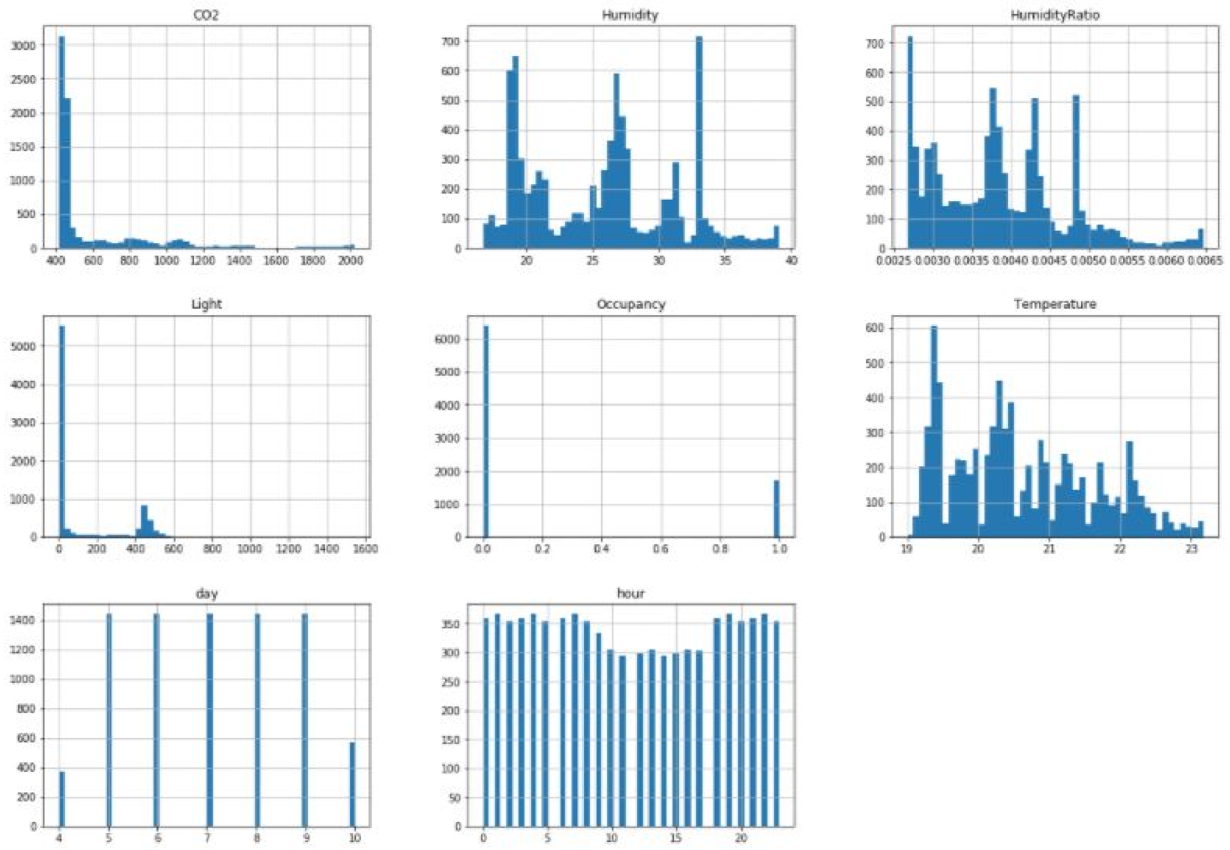


Figure 2: Histograms for each of the variables.

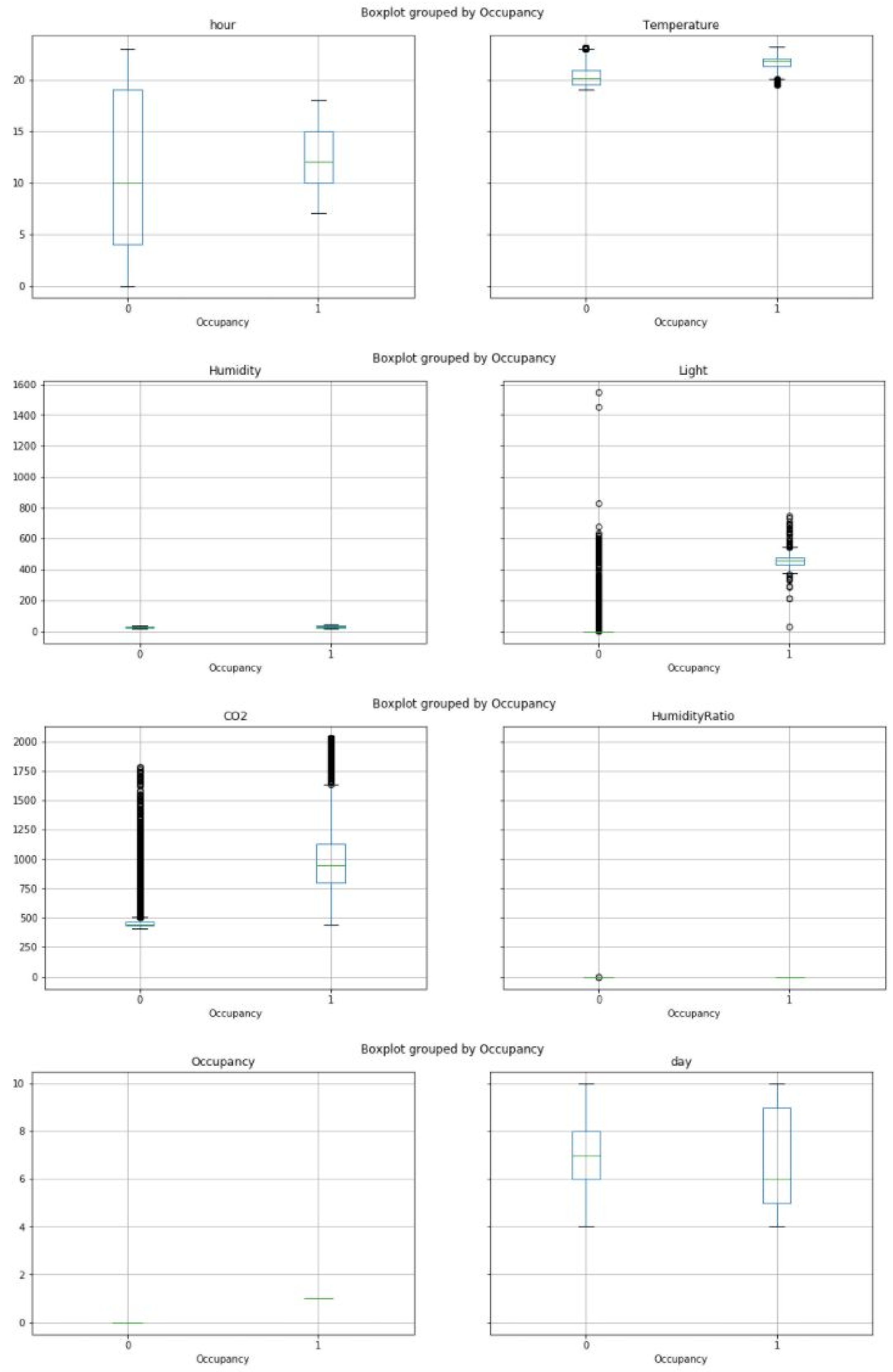


Figure 3: Boxplots for each of the variables, each grouped by Occupancy output variable.

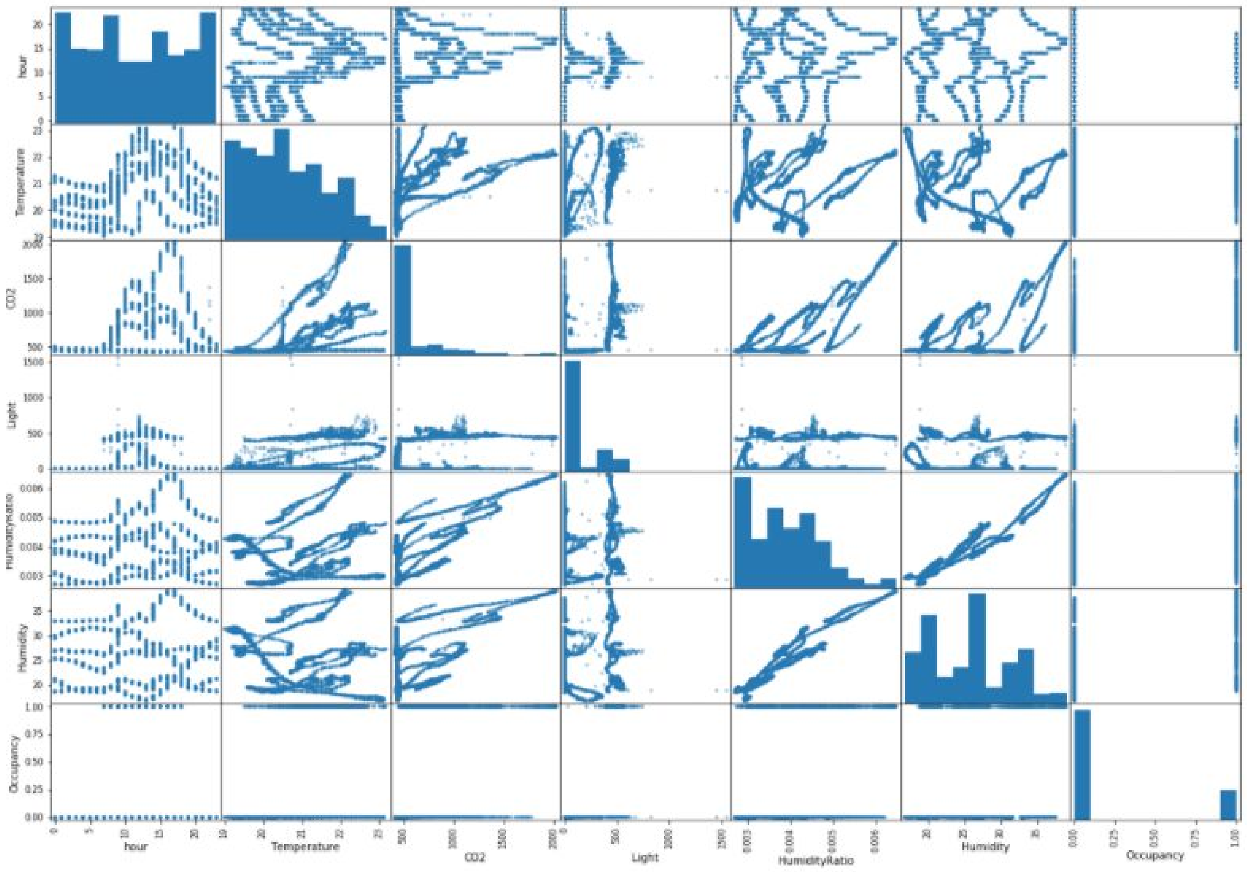
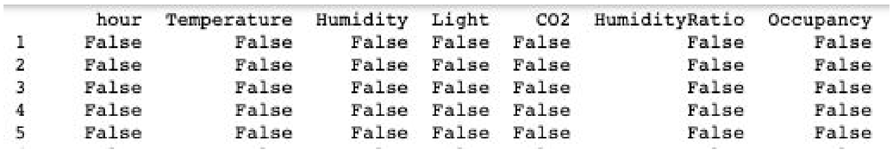


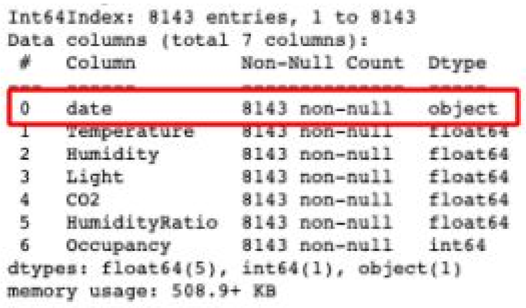
Figure 4: Scatter plot matrix containing each of the variables.

<u>Data Set Cleaning:</u> When it came time to clean the data, this dataset made the job fairly simple. The dataset contains no missing values that had to be adjusted or replaced, analyzed in figure 5. As for the data types, the only feature that needed adjusting was the date variable which I converted from an object (shown in figure 6) to the datetime64 data type (shown in figure 7) in order to extract the day and hour variables (shown in figure 8). Aside from that, the dataset did not need any other cleaning. No other transformations or scaling was performed on the dataset.



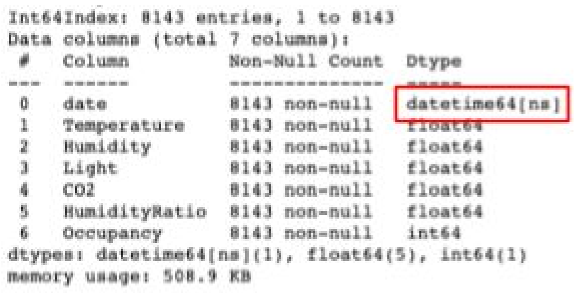
```
    hour  Temperature  Humidity  Light    CO2  HumidityRatio  Occupancy
1  False        False     False  False  False          False      False
2  False        False     False  False  False          False      False
3  False        False     False  False  False          False      False
4  False        False     False  False  False          False      False
5  False        False     False  False  False          False      False
```

Figure 5: Using the pandas.DataFrame.isna() function to check for missing values.



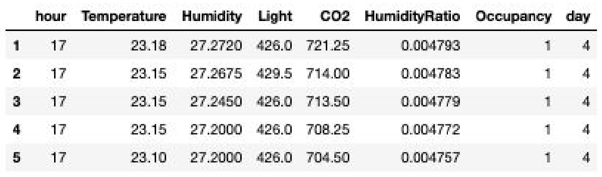
```
Int64Index: 8143 entries, 1 to 8143
Data columns (total 7 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   date           8143 non-null   object
 1   Temperature    8143 non-null   float64
 2   Humidity       8143 non-null   float64
 3   Light          8143 non-null   float64
 4   CO2            8143 non-null   float64
 5   HumidityRatio  8143 non-null   float64
 6   Occupancy      8143 non-null   int64
dtypes: float64(5), int64(1), object(1)
memory usage: 508.9+ KB
```

Figure 6: The list of the data type of each of the dataset features.



```
Int64Index: 8143 entries, 1 to 8143
Data columns (total 7 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   date           8143 non-null   datetime64[ns]
 1   Temperature    8143 non-null   float64
 2   Humidity       8143 non-null   float64
 3   Light          8143 non-null   float64
 4   CO2            8143 non-null   float64
 5   HumidityRatio  8143 non-null   float64
 6   Occupancy      8143 non-null   int64
dtypes: datetime64[ns](1), float64(5), int64(1)
memory usage: 508.9 KB
```

Figure 7: Data type list showing the conversion of date feature from object to datetime64.



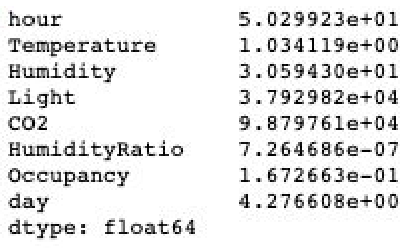
	hour	Temperature	Humidity	Light	CO2	HumidityRatio	Occupancy	day
1	17	23.18	27.2720	426.0	721.25	0.004793	1	4
2	17	23.15	27.2675	429.5	714.00	0.004783	1	4
3	17	23.15	27.2450	426.0	713.50	0.004779	1	4
4	17	23.15	27.2000	426.0	708.25	0.004772	1	4
5	17	23.10	27.2000	426.0	704.50	0.004757	1	4

Figure 8: The first 5 rows of our data showing the hour and day features extracted from date.

Related Work: The original use of this data is very similar to the way it is being used in this project and is very well documented in an article called, “Accurate occupancy detection of an office room from light,temperature, humidity and CO2 measurements using statistical learning models” (1). Two researchers by the names of Luis M. Candanedo and Veronique Feldheim at the University of Mons in Belgium, used sensors to collect various data from their office. The data included the date, temperature, humidity, light, CO2, humidity ratio, and whether the room was occupied. This last parameter, occupancy, was the variable that they were interested in predicting. They did so in order to test multiple classification models, with varying input variables, on their ability to predict the output accurately.

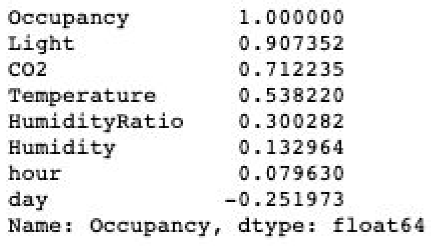
The classification models that they used in their predictions were linear discriminant analysis (LDA), random forest (RF), gradient boosting machine (GBM), and classification and regression trees (CART), each of which were implemented in R. They then varied the inputs variables that were used in each of the models to see how it would affect the accuracy. In the end, they found that each of their models performed very well when all of the initial variables were used as well as “week status” (WS) and “seconds from midnight” (NSM). Here, week status is either 0 or 1 signifying weekend(0) or weekday(1). They found the best accuracy (above 95%) is obtained from LDA, RF and CART.

<u>Feature Extraction:</u> When choosing which features to keep and which to eliminate, the first step I took was calculating the variance of each of the variables (shown in figure 9). I then calculated the correlation between each of the input variables with the output variable “Occupancy” (shown in figure 10). All of the variables showed usefulness, correlation greater than 0, in predicting the output aside from the “day” variable. As you can see in the correlation between day and occupancy is negative. Due to this, I decided to remove the day variable all together.



```
hour              5.029923e+01
Temperature       1.034119e+00
Humidity          3.059430e+01
Light             3.792982e+04
CO2               9.879761e+04
HumidityRatio     7.264686e-07
Occupancy         1.672663e-01
day               4.276608e+00
dtype: float64
```

Figure 9: The calculation for the variance of each of the variables.



```
Occupancy         1.000000
Light             0.907352
CO2               0.712235
Temperature       0.538220
HumidityRatio     0.300282
Humidity          0.132964
hour              0.079630
day              -0.251973
Name: Occupancy, dtype: float64
```

Figure 10: Correlation calculation that resulted in the removal of the ‘day’ variable.

<u>Model Development:</u> In predicting the output variable, I decided to work with five different models. The models that I chose were logistic regression, k-nearest neighbor, linear discriminant analysis, quadratic discriminant analysis, and random forest. All of these models were implemented using the scikit-learn python libraries. I used the same train and test sets for each of the models which I acquired using the train_test_split function from the sci-kit learn model_selection library. The dataset was split with 80% for the training set and 20% for the test set. The train_test_split function was used in order to ensure that an equal representation ratio of the output variable was represented in each set. In the end, both sets had about 79% of the data representing no occupancy and 21% representing occupancy. In order to guarantee the same sets are generated each time, I used random_state=42 for the train_test_split function. Lastly, I split the input variables from the output variables in order to create my four sets X_train, X_test, y_train, y_test. The y sets only contain the output variable "Occupancy" observations while the X sets contain all other variable observations.

Once I had my training and test sets, I began training each of my models using these same sets. For the logistic regression model, I simply fitted the model to my training data using the fit() method. After that, I plugged in my test inputs and generated an array of predictions based on those inputs.

The procedure for the k-nearest neighbor model was very similar to logistic regression except that I had to decide on a k value. The k-values that have been found to result in the best performance are k=5 and k=10. This is due to the fact that they keep bias and variance well balanced. For this specific dataset, I found k=5 to generate the best results.

For the LDA and QDA models, the procedure again involved using the training and test sets to fit these models. The only difference between the two was that LDA required me to specify which solver I wanted to use. Since we primarily covered the least squares approach, this is the solver I specified. As for QDA, the model was kept to the default settings that are set in the sci-kit learn library.

The last model that I fit was random forest. For this model, I kept most of the default setting but changed the maximum features that the model can choose from. The default settings have the model choose from the entire set of predictors each time a split is made. In order to tailor this to the strategies we learned in class, I changed the settings to only be able to choose from the square root of the amount of predictors. These predictors are chosen by random from the entire set. Due to this randomness characteristic of this model, I also set a random state in order to ensure that my procedure could be accurately replicated. The random state value I chose was 0.

<u>**Fine-tune your models & Feature Set:**</u>

The results from each of my models, after their first runs, was initially very good. Each of my models scored above 98% accuracy without any adjustments being made. The feature set was divided up into training and testing using the train_test_split function which optimizes the split for the user. I inputted that I wanted a 80-20 split between training and testing. Each of the sets were split in such a way that the ratio of occupancy-no occupancy was equal in the training and test sets. This created optimal training and test sets to ensure that my model would encounter enough of each possible outcome. This makes for more accurate model development and testing.

The fine-tuning of my models was fairly simple since the accuracy was very high even after just the first run. Also, since the dataset is not really big, I did not have to worry about improving computational performance. This meant that I could design each model to fit the dataset as best it could. This is most clearly seen in the random forest model where no restrictions were set on the max depth of each of the decision trees that are generated. I left it on default which generates each decision tree in a way where all of the leaves are pure, meaning the tree splits until no leaf contains more than one of the output classes. The only other model that required fine-tuning is KNN. I had to test a few different values for K to find the value that produces the most accurate results. The results showed that the best value is k=5.

Performance: In order to measure the performance of each of the five models, I calculated the mean-squared-error (MSE), R^2 error, and the accuracy levels based on the confusion matrix. The accuracy levels based on the confusion matrix were calculated for when an occupant is present, when no occupant is present, and for the full result. The results showed that all of the models were very accurate. To my surprise, this was the case from the very beginning, before any fine-tuning was done. None of my models showed an accuracy level of less than 98.8%. The best performing model was found to be random forest which showed an accuracy level of 99.69% and an R^2 error of 0.98. This is due to how thorough this model's fitting process is, especially since no limit was set for the max depth of each of the decision trees. The second best model was found to be KNN which had an accuracy of 98.96% and an R^2 error of 0.94. Looking at the last three models, QDA and logistic regression were found to have identical performance with both having accuracy levels of 98.83% and an R^2 errors of 0.93 while LDA was just a little behind, coming in last with an accuracy level of 98.77% and an R^2 error of 0.926. However, a surprising characteristic of the last three models (QDA, logistic regression, and LDA) was that all three of them had 100% accuracy when it came to predicting when no occupant was present, something none of the top performing models were able to do. This goes to show that no one model is the best, each has their strengths. The resulting MSE, R^2 error, and accuracy levels based on the confusion matrix are shown below in figures 11-15.

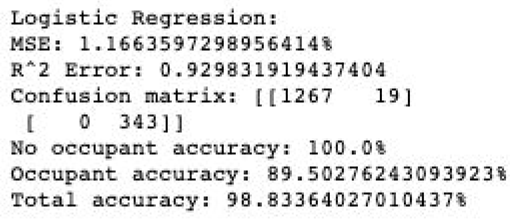

```
Logistic Regression:
MSE: 1.1663597298956414%
R^2 Error: 0.929831919437404
Confusion matrix: [[1267   19]
 [   0  343]]
No occupant accuracy: 100.0%
Occupant accuracy: 89.50276243093923%
Total accuracy: 98.83364027010437%
```

Figure 11: Performance results for the Logistic Regression model.

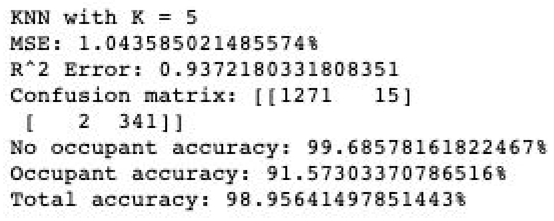
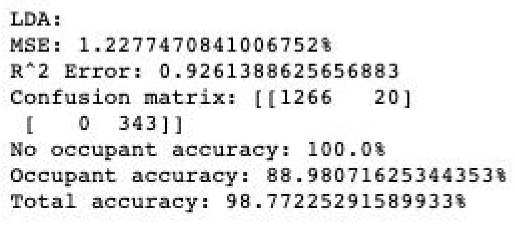
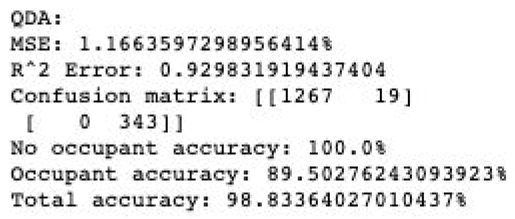

```
KNN with K = 5
MSE: 1.043585021485574%
R^2 Error: 0.9372180331808351
Confusion matrix: [[1271   15]
 [   2  341]]
No occupant accuracy: 99.68578161822467%
Occupant accuracy: 91.57303370786516%
Total accuracy: 98.95641497851443%
```

Figure 12: Performance results for the K-Nearest Neighbor model.



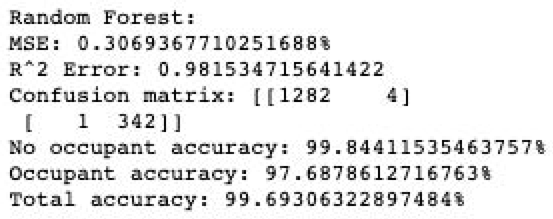
```
LDA:
MSE: 1.2277470841006752%
R^2 Error: 0.9261388625656883
Confusion matrix: [[1266   20]
 [   0  343]]
No occupant accuracy: 100.0%
Occupant accuracy: 88.98071625344353%
Total accuracy: 98.77225291589933%
```

Figure 13: Performance results for the Linear Discriminant Analysis (LDA) model.



```
QDA:
MSE: 1.1663597298956414%
R^2 Error: 0.929831919437404
Confusion matrix: [[1267   19]
 [   0  343]]
No occupant accuracy: 100.0%
Occupant accuracy: 89.50276243093923%
Total accuracy: 98.83364027010437%
```

Figure 11: Performance results for the Quadratic Discriminant Analysis (QDA) model.



```
Random Forest:
MSE: 0.3069367710251688%
R^2 Error: 0.981534715641422
Confusion matrix: [[1282    4]
 [   1  342]]
No occupant accuracy: 99.84411535463757%
Occupant accuracy: 97.6878612716763%
Total accuracy: 99.69306322897484%
```

Figure 11: Performance results for the Random Forest model.

References:

1) L. M. Candanedo and V. Feldheim, "Accurate occupancy detection of an office room from light, temperature, humidity and CO 2 measurements using statistical learning models," *Energy and Buildings*, vol. 112, pp. 28–39, Jan. 2016.
2) G. James, D. Witten, T. Hastie, and R. Tibshirani, *An introduction to statistical learning: with applications in R*. New York, NY: Springer, 2017.