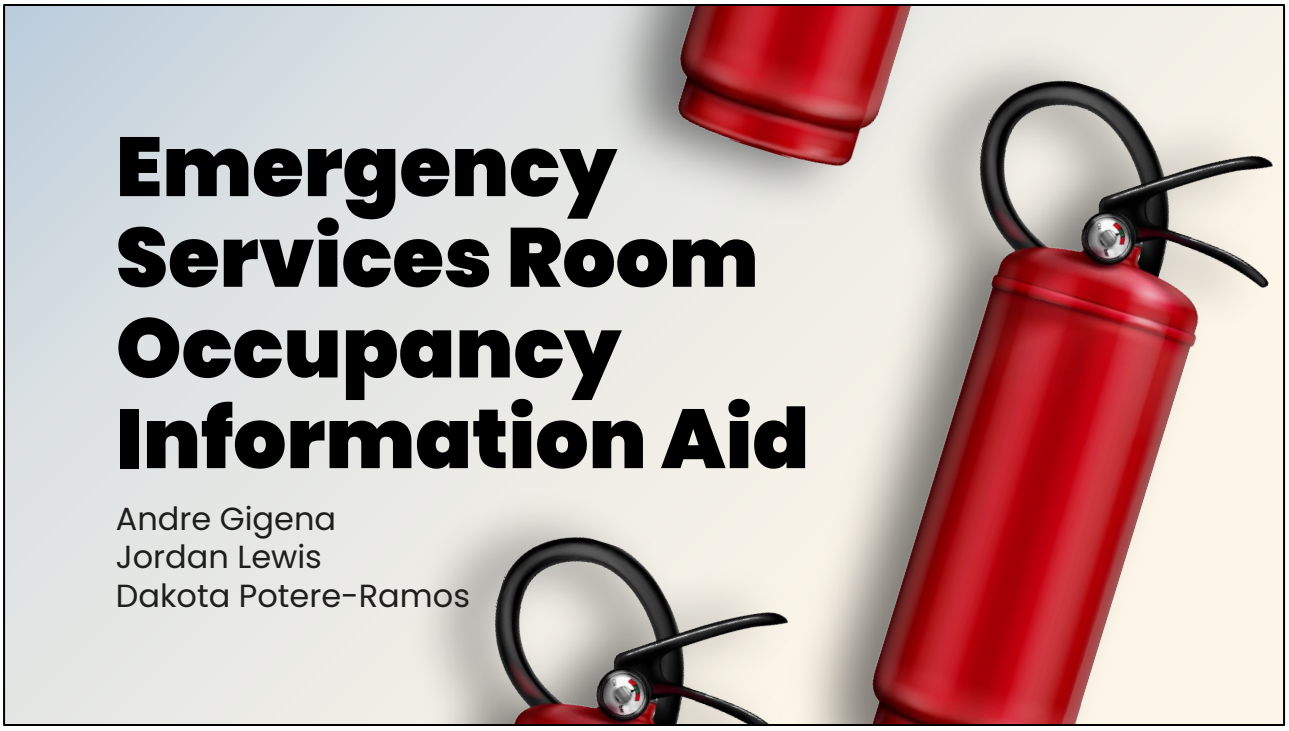


Andre (10 seconds):

Today, Jordan, Dakota, and I will be sharing with you the results of our machine learning project to predict room occupancy for use by emergency services.

Research Question

Can non-intrusive sensors and machine learning algorithms be used to inform emergency services of estimated room occupancy?

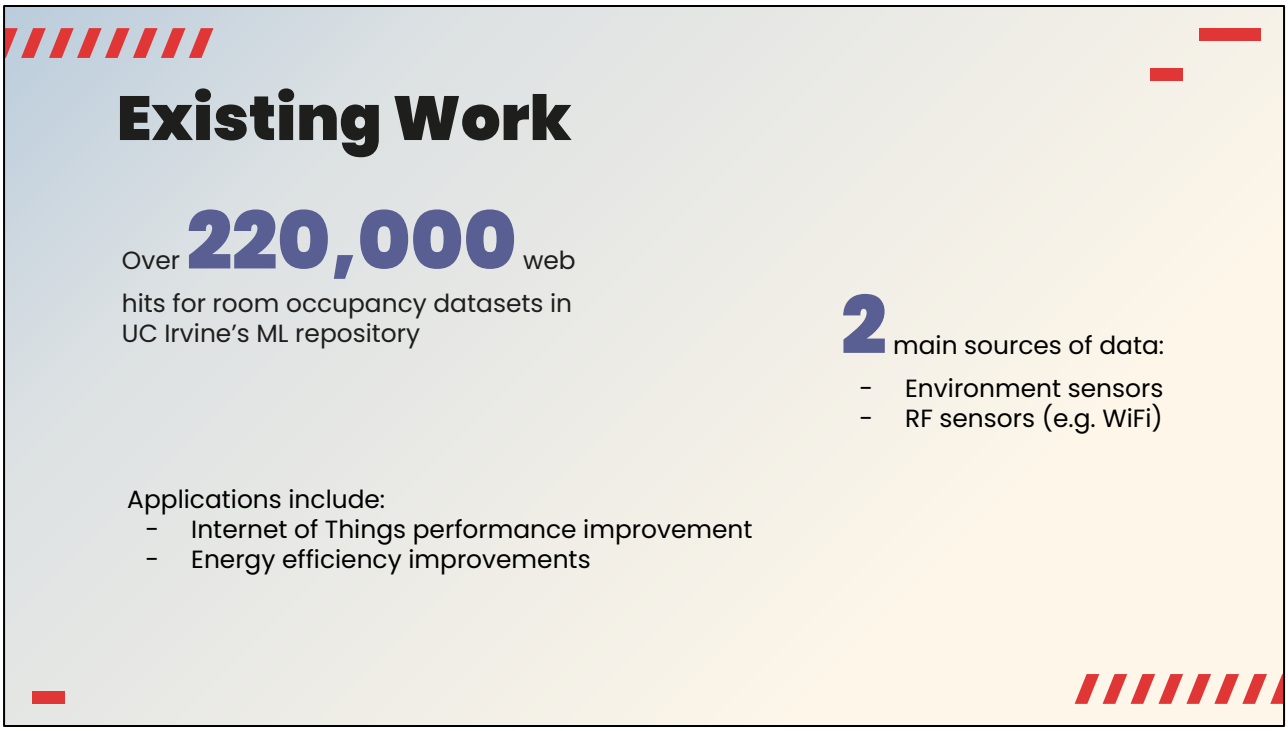
“We risk a lot to save a lot”

Andre (45 seconds):

The question we’re trying to answer is “Can non-intrusive sensors and machine learning be used to inform emergency services of estimated room occupancy?” This project serves as a proof-of-concept for the data processing and machine learning algorithms that could eventually be included in something like the fire control panel in commercial buildings to provide emergency services with additional information when responding to calls. In talking with a friend of mine who is a firefighter, he liked the idea of an additional information source like this, but underscored the importance of providing accurate information. A quote he kept repeating was, “We risk a lot to save a lot,” which really drove home the value of life in emergency situations and the need for our models to perform well.

Sources:

- Firefighter image: https://pixabay.com/photos/firemen-firefighter-fire-flames-78110/



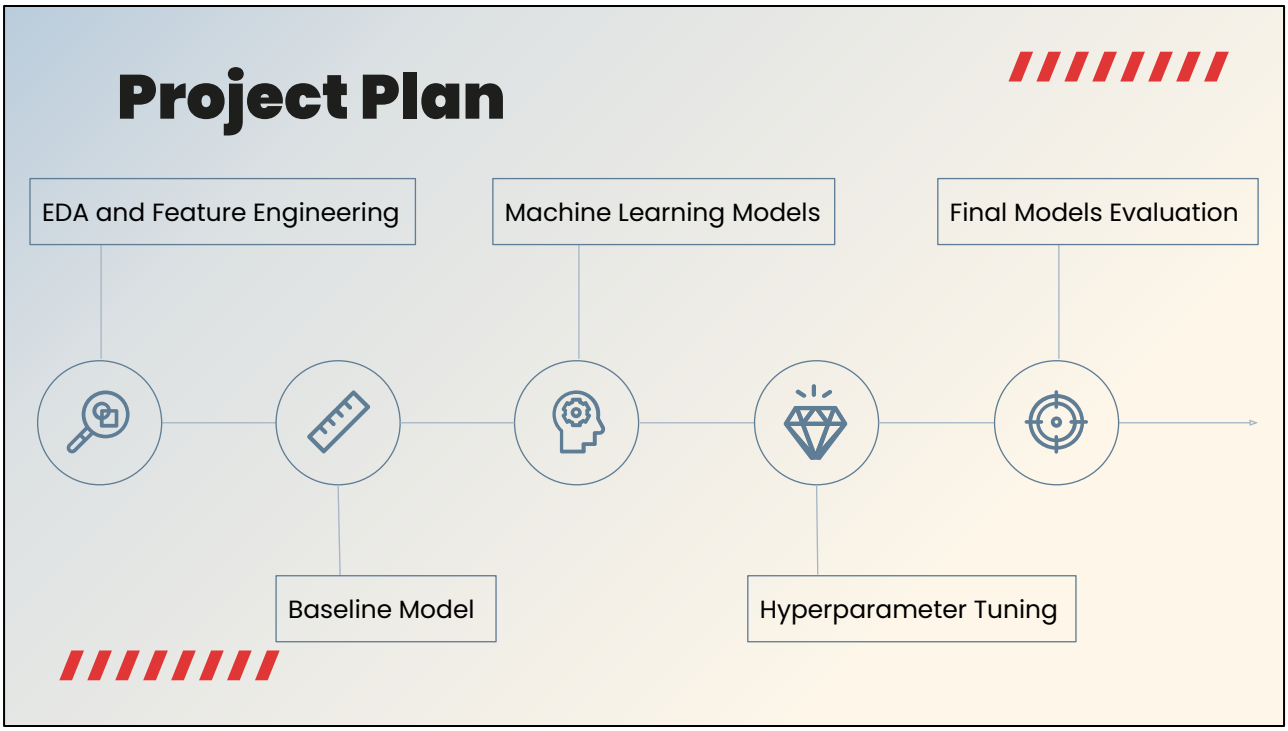
Andre (50 seconds):

Room occupancy prediction seems to be a popular machine learning experiment with over 220,000 web hits for room occupancy datasets in UC Irvine's machine learning repository alone. Previous work in this area relies on two main sources of data including data from environment sensors (think temperature or humidity) or data from radio frequency sensors (think WiFi). Despite large numbers of machine learning experiments, actual applications of room occupancy machine learning algorithms were a little more difficult to find. One application we found cited in a paper was Internet of Things (IoT) performance improvement such as improved smart home automated lighting features. Another application we found cited was improved energy efficiency by varying air conditioning based on room occupancy.

Sources:

- Web hits statistic: https://archive.ics.uci.edu/ml/datasets/Room+Occupancy+Estimation, https://archive.ics.uci.edu/ml/datasets/Occupancy+Detection+
- Applications:
  - Adarsh Pal Singh, 'Machine Learning for IoT Applications: Sensor Data Analytics and Data Reduction Techniques', Masters Thesis, [Web Link], 2020.
  - Wei Wang, Jiayu Chen, Tianzhen Hong, Occupancy prediction through machine learning and data fusion of environmental sensing and Wi-Fi sensing in buildings, Automation in Construction, Volume 94, 2018,

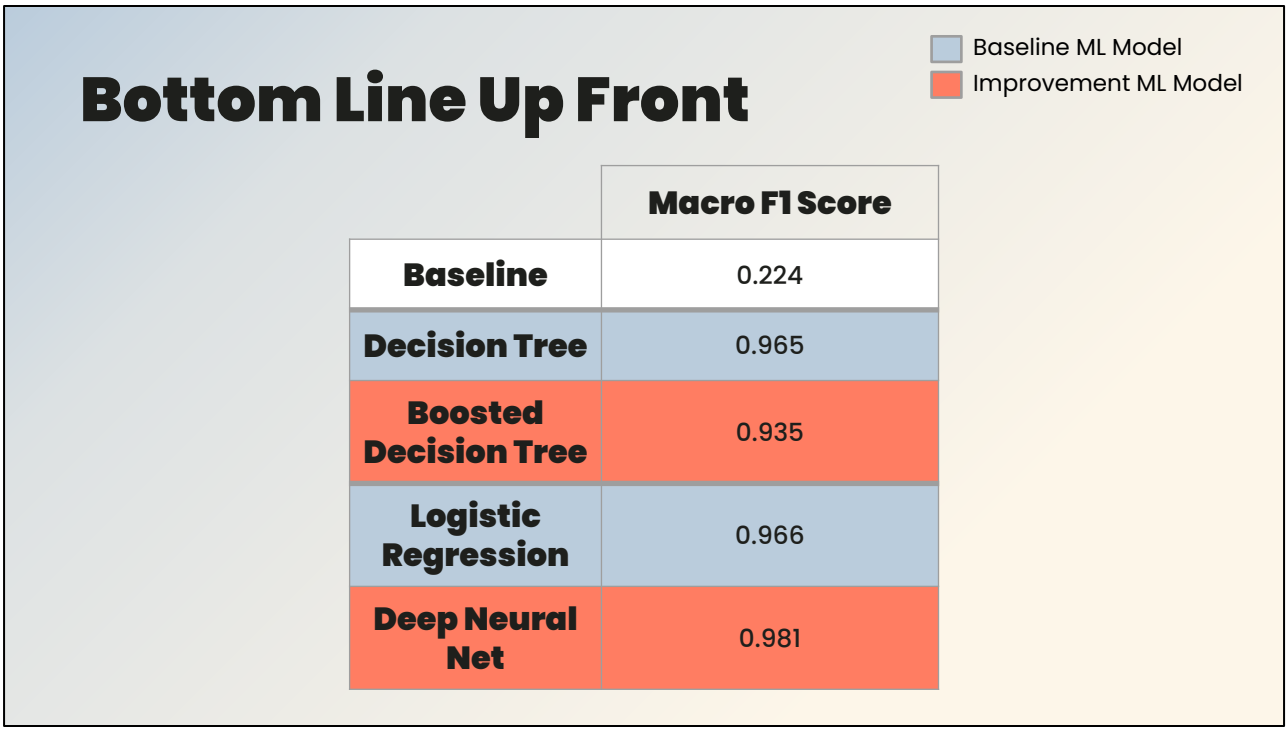
- Pages 233-243, ISSN 0926-5805, https://doi.org/10.1016/j.autcon.2018.07.007.



Dakota (55 seconds)

Here we have outlined our overall plan to approach this research question:

- We begin by performing exploratory data analysis to familiarize ourselves with the data, identify nuances and shortcomings, derive new features, and prepare the data for use in machine learning models
- Next we create a baseline model as a benchmark to compare how well our machine learning models are performing. For this project the baseline model predicts the room is always unoccupied
- We then create the first iterations of our machine learning models. For this project we will explore decision trees, gradient boosted decision trees, logistic regression and deep neural networks.
- With the machine learning models created and trained we will then tune the hyperparameters of each model to optimize how well they generalize to new unseen data
- Once our hyperparameter tuning is complete we will then evaluate our models and see how well they can predict the unseen test data.
- We will utilize macro averaged F1-score as the metric to evaluate our models' performance. We'll talk more about why we chose this metric later.



Dakota (45 seconds)

Alright, before we dive into the data and specifics of each model here is a quick summary of our final results. We believe providing this information up front will help steer your attention and aid in deriving insights from the models.

Moving from least complex at the top to most complex at the bottom we see a general trend of improved performance as we add complexity, surprisingly the gradient boosted decision tree did not follow this trend. All of our models performed well with f1-scores above .9.

Room Occupancy Estimation Data Set

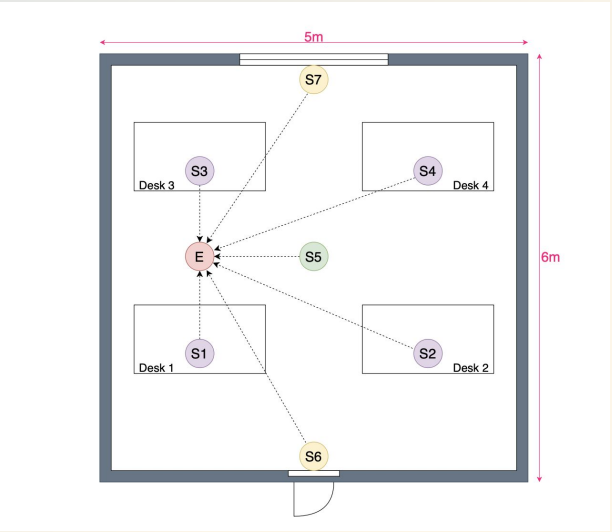
Source: UC Irvine's ML Repository

Size of Dataset: 10,129 data points

Dataset features are measurements from various sensors including:

- 4 Temperature (℃)
- 4 Light (Lux)
- 4 Sound (Volts)
- 1 CO_2 (PPM)
- 1 CO_2 Slope (dPPM/dt)
- 2 PIR motion (Binary)

Target: room occupancy ranging from 0-3 people



Dakota (70 seconds)

Our dataset comes from a study in UC Irvine's Machine Learning Repository that was conducted in 2018 by Singh et al. at the International Institute of Information and Technology in Hyderabad . Sensors were placed in a single room and the values and occupancy numbers were recorded over time.

There are 10,129 instances with 16 features each. Many of these features come from multiple sensors of the same type. Each instance also records the number of persons in the room which ranges between 0 and 3.

The non-intrusive sensors used in this study are temperature, light, sound, CO2 and IR motion detection.

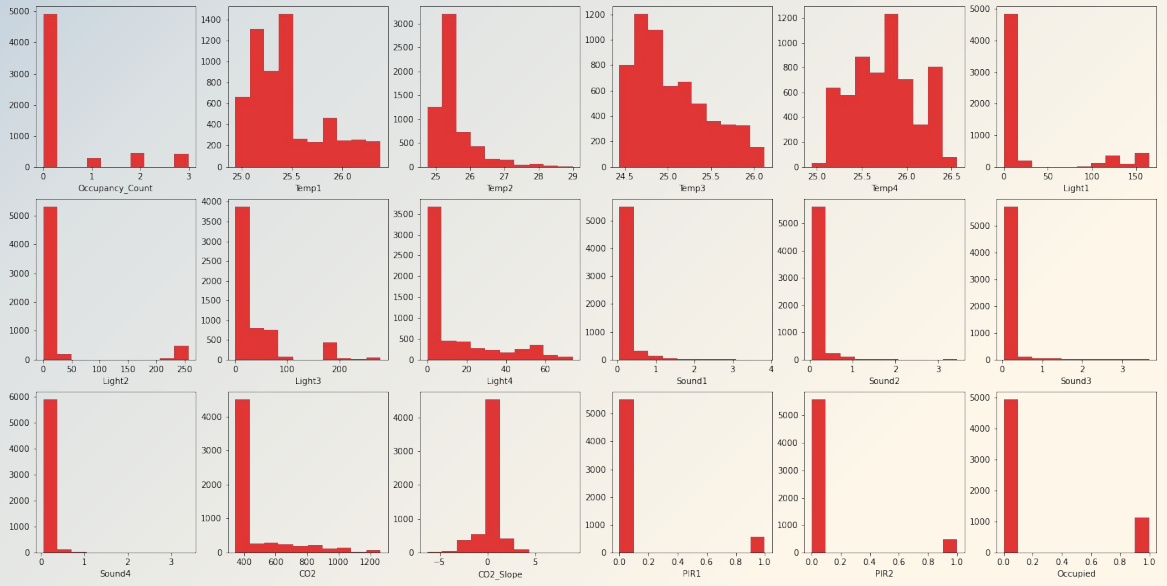
We explored a number of different ideas to derive new features from the existing dataset such as time derivatives of temperature, products of two features, and polynomial transformations of the features. However, we ultimately discarded these derived features as they did not add meaningful information to the models. We did create a binary feature to indicate whether a room was occupied or not which was used in the data preprocessing stage.

Referencing the image on the right, the temperature, light, and sound sensors were placed at nodes S1, S2, S3, and S4, above the desks.
The CO2 sensor was placed at the S5 node in the center of the room

The IR sensors were placed at nodes S6 and S7 which are above a door and window respectively

https://doi.org/10.1109/GLOCOMW.2018.8644432

Feature Distributions



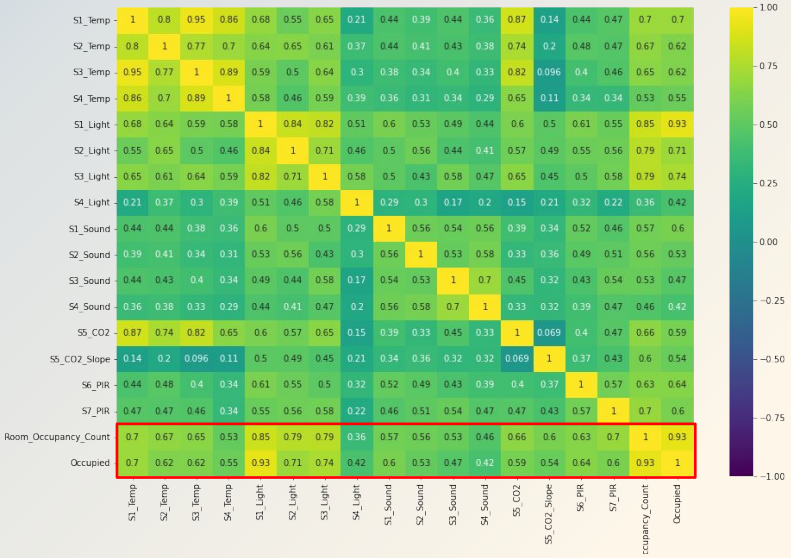
Jordan (25 Seconds)
Here is distribution of the various features and outputs of our data.

Overall the sensors have very narrow and consistent ranges for values.

Many of the sensors display heavily imbalanced data towards the lower ranges, corresponding to the large number of 0 occupancy instances.

This imbalance in the labels for our outputs is seen in the top left and bottom right graphs which show the distribution of the number of occupants and a binary representation of occupied versus unoccupied, respectively.

Feature Correlation



Jordan (30 Seconds)

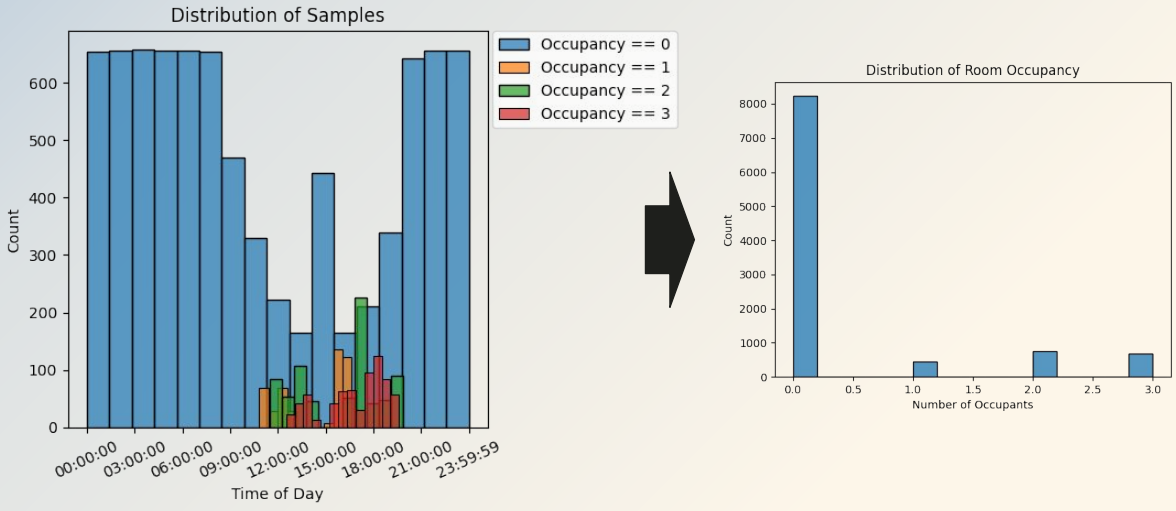
This correlation plot shows the correlation between the variables with more yellow colors being more strongly positively correlated.

The correlation for the output can be seen in the lower two rows of the graph, highlighted in red here.

The first of those rows shows the correlation with the number of occupants while the last row shows the correlation between the binary output of whether a room is occupied or not.

The temperature and light sensors had the strongest correlation with room occupancy and are seen further to the left on the matrix.

Class Imbalance

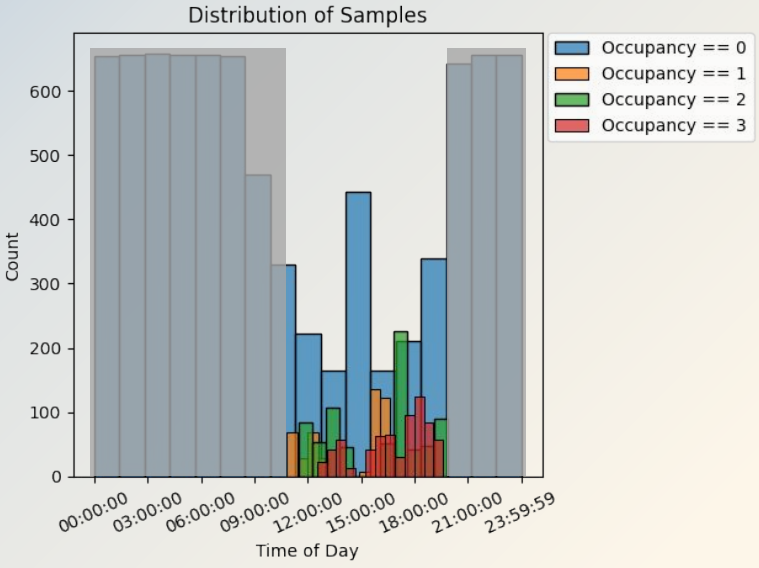


Jordan (20 Seconds)

Coming back to the imbalance in the output labels of our data, the plot on the left shows the distribution of values throughout different times of the day and the plot at right shows the overall distribution between the labels.

It is clear that the overwhelming majority of the data is for unoccupied room times, but also that all the occupied room labels occur during a specific time window during the day.

Class Imbalance Filtering



Filtering applied exclusively to training dataset

Jordan (5 Seconds)

So we downsampled our training data to this window of the day to have a less skewed distribution of labels.

Downsampling Results

Jordan (25 Seconds)

These are the results of downsampling our data and applying a 60/20/20 split.

The top row of graphs shows the distribution by count and the lower row shows the distribution as a percentage of the total.

From left to right we have the original data, the training data, the validation data, and finally the test data.

You can see that the training data being downsampled had a significant impact on the distribution of the outputs compared to the validation and test data which reflect the original data distribution.

Approach

Baseline ML Model
Improvement ML Model

Baseline Model

Baseline Model	Simple constant model which always predicts zero occupants

Decision Trees

Decision Tree	Simple non-linear model that could partition the data well
Gradient Boosted Trees	Ensemble method will help minimize the errors of baseline

Neural Networks

Logistic Regression	Exploration into whether the data is linearly separable
Deep Neural Network	Non-linear model that could capture nuances more effectively

Andre (70 seconds):

Our baseline, non-machine-learning model is to predict the majority class of zero occupants. As improvements to this baseline model, there are two classes of machine learning algorithms we used: decision trees and neural networks.

For the decision tree class of models, our baseline model is a decision tree. We suspected there are certain thresholds in the sensor data based on room occupancy which a decision tree would effectively identify. However, we didn't expect the decision boundary to be clear due to small differences in sensor data between a non-occupied room and a just-recently-occupied room. Thus, we included gradient boosted trees with the hopes of an ensemble method yielding better results.

For the neural network class of models, our baseline model is a logistic regression (single layer neural network). The thought was that a logistic regression model could help us identify whether the data was linearly separable or not between classes. Based on our suspicion that the data was not linearly separable, we included a deep neural network to improve predictive performance by capturing the non-linearities and nuances we suspected are present in the data.

F1 Score

Hybrid metric useful for unbalanced classes

Provides a balance between precision and recall

Desire to maximize true positives while minimizing false positives and false negatives.

Macro averaged: calculates metric for each class then averages

$$\text{F1 Score} = \frac{TP}{TP + \frac{1}{2}(FP + FN)}$$

$$\text{Macro F1 Score} = \frac{\sum_{i=1}^{n} \text{F1 Score}_i}{n}$$

Dakota (45 seconds)

To evaluate our models' performance we will utilize the macro averaged f1 score.

The f1 score is a hybrid metric which is especially useful when dealing with class imbalances as it provides a harmonic mean between precision and recall. Given our data is heavily imbalanced toward the unoccupied class we chose f1 score as a robust metric to evaluate our models.

Echoing back the quote "We risk a lot to save a lot" we desire to maximize our models correct predictions of room occupancy (the true positives) and minimize the models incorrect predictions (the false positives and false negatives) in a multi class setting.

The macro averaged f1 score is calculated by averaging the f1 score of each class. This provides a robust score that weights all classes equally.

Decision Tree

0.965

Macro-averaged f1 score

Hyperparameter	Values
Max Tree Depth	1, 2, 3, **4**, 5, 6, 7, 8
Criterion	Entropy, **Gini Impurity**

- Hyperparameter tuning performed with non-standardized validation data set
- Final model trained on combined non-standardized training and validation data set, evaluated on test set
- Number of samples per leaf and per split was decided by sklearn

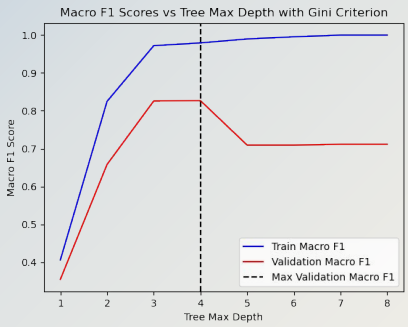
Dakota (30 seconds):

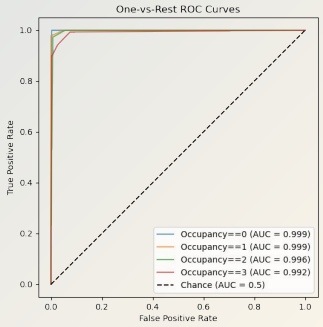
Ok, moving on to our experiments and hyperparameter tuning. Generally speaking we used an exhaustive approach for tuning hyperparameters. For all possible combinations of the values seen in this table, a model was built and evaluated. The results were captured and the combination which maximized macro-averaged f1 scores on the validation data was chosen as optimal. Of the 16 models analyzed, a max tree depth of 4 with gini impurity criterion produced the optimal results.

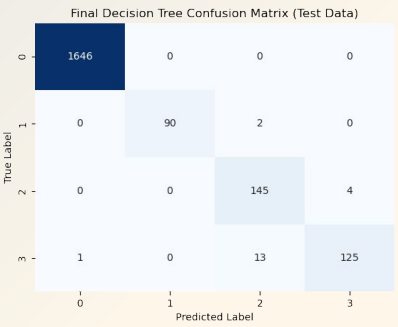
Decision Tree

0.965

Macro-averaged f1 score







Dakota (25 seconds):

The leftmost figure visualizes the relationship between max tree depth and the macro f1 score with the optimal tree depth indicated by the dashed vertical line. The middle figure displays the ROC curve for the final decision tree model evaluated on the test data, the model almost perfectly predicts occupancy 0 through 2 but struggles slightly with occupancy 3. The confusion matrix seen on the right confirms this behavior.

Gradient Boosted Decision Trees

Hyperparameter	Values
Learning Rate	0.01, 0.05, 0.1, **0.2**, 0.3
No. Estimators	50, **100**, 150, 200
Subsample	**0.1**, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0
Criterion	**Friedman MSE**, Squared Error
Max Tree Depth	1, 2, **3**, 4

0.935

Macro-averaged f1 score

- Hyperparameter tuning performed with non-standardized validation data set
- Final model trained on combined non-standardized training and validation data set, evaluated on test set
- Number of samples per leaf and per split was decided by sklearn

Dakota (35 seconds):

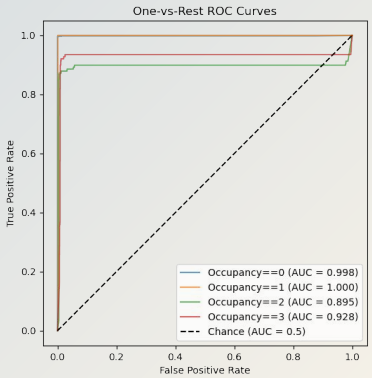
In a similar fashion as the decision trees, the gradient boosted models were also tuned with an exhaustive approach. For all possible combinations of the values seen in this table, a gradient boosted model was built and evaluated. The results were captured and the combination which maximized macro-averaged f1 scores on the validation data was chosen as optimal. Of the 1600 models analyzed, a learning rate of 0.2, with 100 estimators, a subsample of 0.1, friedman mse criterion, and max tree depth of 3 produced the optimal results.

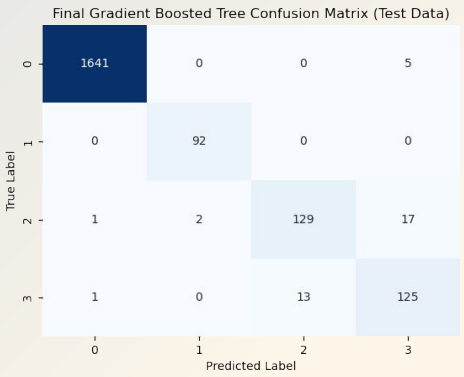
Note:
Subsample = fraction of samples used for fitting individual base learners (weak trees). <1.0 = stochastic gradient boosting.

Gradient Boosted Decision Trees

0.935

Macro-averaged f1 score





Dakota (20 seconds):

The left figure displays the ROC curve for the final gradient boosted tree model evaluated on the test data. Surprisingly this model did not perform as well as the final decision tree model, the model struggled to correctly identify 2 and 3 room occupants. The confusion matrix seen on the right confirms this behavior.

Logistic Regression

0.966

Macro-averaged f1 score

Hyperparameter	Values
Optimizer	**Adam**, SGD
Learning Rate	0.001, **0.01**, 0.1
Loss Function	Kullback-Leibler Divergence, **Categorical Crossentropy**
Batch Size	16, **32**, 64

- Hyperparameter experimentation utilized standardized training and validation datasets
- Final model tuning utilized combined and standardized training/validation with loss evaluated on the testing dataset
- Number of epochs was decided by TensorFlow's EarlyStopping callback

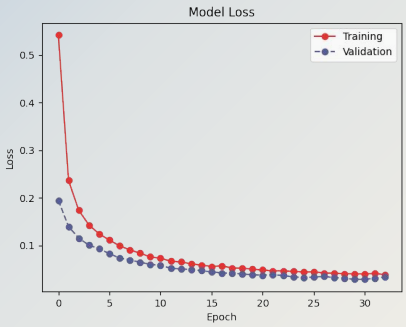
Andre (45 seconds):

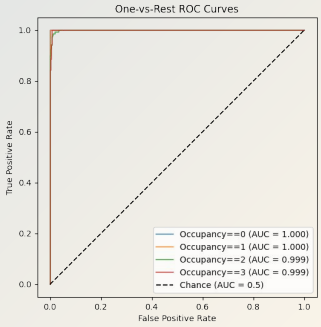
Moving onto the neural network class of models – the logistic regression model achieved a macro-averaged f1 score of 0.966. This was accomplished by running models with a full-factorial combination of the various hyperparameters listed in the table on the left. With each combination, the model's macro f1 score on validation data was recorded. Ultimately, 36 different models were evaluated and the optimal parameter configuration (highlighted in blue) was identified as yielding the best results. A final version of the model was trained using a combined and standardized training and validation dataset with performance evaluated on the testing dataset. The number of epochs was decided by TensorFlow's EarlyStopping callback.

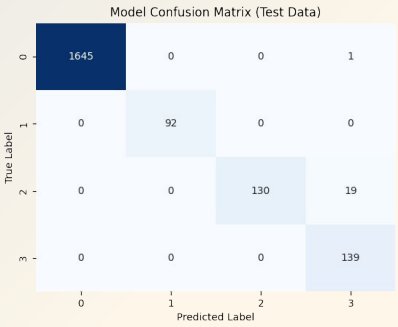
Logistic Regression

0.966

Macro-averaged f1 score







Andre (40 seconds):

On the left, you can see the training and validation loss for the final version of the model. Again, the training curve comes from the combined training and validation data and the validation curve is from the test data. We suspect the validation curve is lower than the training curve due to the high number of “No occupant” samples in the test data and the fact that the logistic regression is really good at predicting the class of no occupants (as can be seen in the confusion matrix on the right). The ROC curve in the middle shows that the logistic regression has effectively distinguished between classes and gets relatively close to perfect classifier performance for many of the classes of occupancy.

Deep Neural Net

Hyperparameter	Values
Optimizer	**Adam**, SGD
Learning Rate	**0.001**, 0.01, 0.1
Loss Function	**Categorical Crossentropy**
Input Layer Size	**16**
Number of Hidden Layers	1, **2**, 3
Hidden Layer Sizes	4, **8**, 32, 64, 128 4, 8, **32**, 64, 128
Dropout Layer	Included, **Not Included**

0.981

Macro-averaged f1 score

- Similar process was used to logistic regression hyperparameter experimentation & tuning in regards to training and test sets
- After one iteration with each combination, top performing scores were re-run with multiple iterations to rank performance by average score
- Number of epochs was decided by TensorFlow's EarlyStopping callback

Jordan (40 seconds):

The deep neural net model achieved a macro-averaged f1 score of 0.981.

Similar to the logistic regression this was accomplished by a full combination of the hyperparameters listed in the table on the left.

This resulted in thousands of models being run and the macro f1 score being recorded for each.

To better understand the differences in so many models, the models with the highest macro f1 score were re-run over multiple iterations and the results averaged

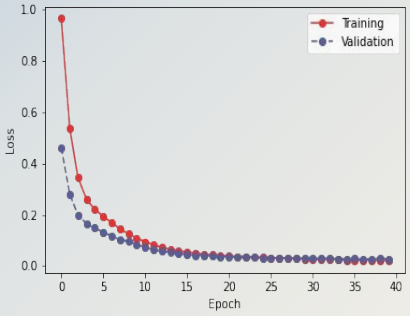
The resulting hyperparameters with the highest performing score are highlighted in blue.

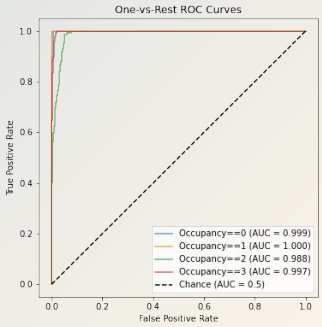
Again, similar to the logistic regression process, the final model was trained using the combined training and validation dataset with performance evaluated on the testing dataset and the number of epochs decided by TensorFlow's EarlyStopping callback.

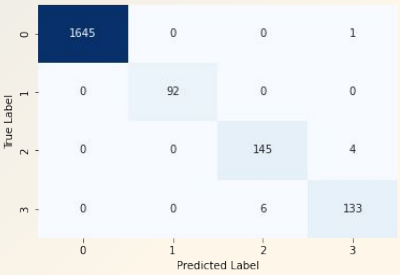
Deep Neural Net

0.981

Macro-averaged f1 score







Jordan (40 seconds):

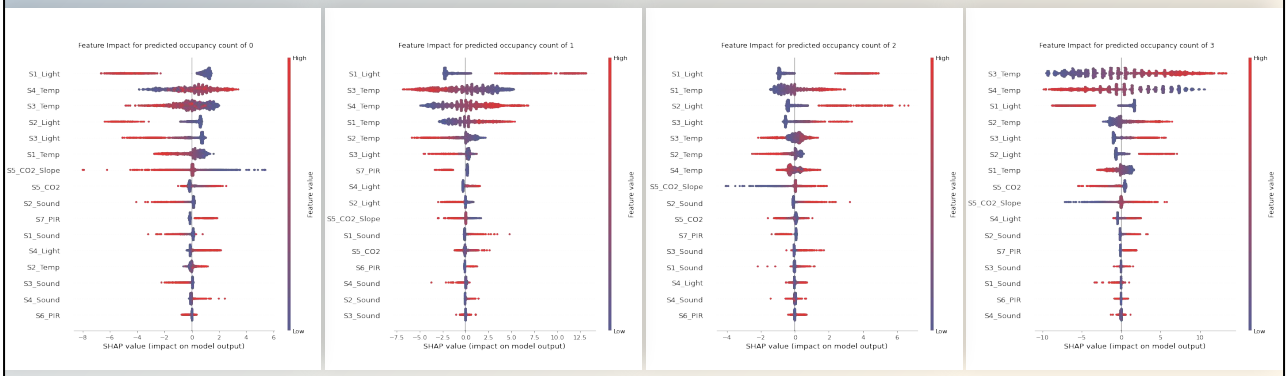
On the left, you can see the training and validation loss for the final version of the model, which had higher initial losses than the logistic regression model, but has a similar shape overall

The ROC curve in the middle also has some notable differences from the performance of the logistic regression model.

The neural net had a little extra trouble in distinguishing between some examples of a room occupied by 2 or 3 people leading to that slower ascending green line representing the 2 person occupancy label.

The confusion matrix at right shows that a handful of rooms with 2 were actually predicted as 3 and that a handful of rooms with 3 were actually predicted as 2.

Deep Neural Net



Jordan (1 minute 15 seconds)

These charts were produced using the Shap package and provide an explanation on the outcome of the different classes as a result of the feature values.

Each plot is for a different occupancy level starting with 0 on the left.

[CLICK]

Each of the dots in this plot represents the values of one sample and the red colors represent high values for that feature and the blue dots represent low values for that feature.

Dots further to the left have a negative impact on predicting that class and dots on the right have a positive impact.

The features are listed by the absolute value of their impact.

So for the 0 class we see that the S1 Light sensor is listed first and that the high values had a strong negative impact while low values were closer to neutral.

[CLICK]

In contrast as we look at occupied rooms we see that the high light values, like here

for room occupancy of 1, lead to a positive prediction impact.

[CLICK]

Again for room occupancies of 2 the S1 Light has the most impact

[CLICK]

While for room occupancies of 3 the temperature sensors have the most impact and the S1 light actually has negative impact.

A final point to note is that the top 5 features for each label are similar and have a wide range of impact. Whereas many of the features listed lower have a neutral impact.

Conclusion

- **Looking Forward:**
  - Conduct further studies to determine best performing location and number of sensors
- **Summary:**
  - The models used reliably predict room occupancy given the features in the dataset

	Macro F1 Score
Baseline	0.224
Decision Tree	0.965
Boosted Decision Tree	0.935
Logistic Regression	0.966
Deep Neural Net	0.981

Jordan (35 Seconds)

On that note, our advice for further studies is to conduct research on to what sensors and which locations have the best performance for prediction.

It would be cumbersome to add so many different sensors to every room, but perhaps a smaller number of sensors in strategic locations can still yield powerful results.

In conclusion, looking at our models, we were able to reliably predict the room occupancy given the features in this data set.

The macro F1 score for each model is shown again in the table at right.

The deep neural net had the best performance, but the decision tree and logistic regression still had very high performing results.

Thank you!
Q/A?

Link to GitHub Repository

https://github.com/dakotaramos/207_final_project

Individual Contributions

- Andre
  - Contributed to the “Initial EDA and Feature Engineering” section of the notebook
  - Wrote various utility functions in the “Model Pipeline” section of the notebook
  - Implemented the entirety of the “Logistic Regression” section of the notebook
  - Significant contributions to first half of slides and the “Logistic Regression” section
- Jordan
  - Contributed to the “Initial EDA and Feature Engineering” section of the notebook
  - Experimented with each of the preliminary models
  - Implemented the entirety of the “Neural Net” section of the notebook
  - Contributions to final presentation slides
- Dakota
  - Performed preliminary research on model types, sklearn and tensorflow frameworks, evaluation metrics, optimizers, feature engineering, and upsampling / downsampling techniques
  - Contributed to the “Initial EDA and Feature Engineering” section of the notebook
  - Wrote helper functions to build decision trees and gradient boosted decision trees
  - Implemented the entirety of the “Baseline Model”, “Decision Trees”, and “Gradient Boosted Decision Trees” sections of the notebook
  - Contributions to final presentation slides

We all agree that we each contributed equally to this project in terms of time, effort, and contributions to deliverables

Last slide