Revit Fixtures with Machine Learning (December 2019)

Aguirre, Daniel
Dagui001@fiu.edu
Capstone Report

Abstract—**The MEPT engineering design field has been largely overlooked by the data science industry. The design process has been fairly consistent since its hand drafting days and has little to no automation involved. This project's goal is to demonstrate that a machine learning application would be efficient and accurate enough for the industry to apply it to their design process. The metric to measure the significance of this project was a precision score of 95%. Through a machine learning algorithm this project predicts whether a smoke detector is required in each room of a project building, and its subsequent layout in the designated room using Revit and Dynamo. The data needed to train the machine learning algorithm consist of previous Revit projects, that are mined using PyRevit and stored in SQL. The parameters mined are tested for their correlation and dependency to predict whether a smoke detector is needed. The applicable parameters are then transformed to a usable format specific to the machine learning algorithm. The train data is then fit to the predicted model and the predictions are outputted to Revit's Dynamo. Dynamo is used to lay out the predicted smoke detectors in their designated rooms. Once the project is completed and constructed, the data is mined from Revit and dumped into the master database in SQL for future training. The algorithms that met the specified metric were TensorFlow's Keras, KNN, Random Forest, and SVM, while Naive Bayes and Logistic Regression fell short. The final process from mining new Revit project data to laying out the predicted smoke detectors back in Revit takes a few minutes in contrast to hours or days when done manually. The future is vast for this field and promising.**

Index Terms- Data Science, MEPT, Machine Learning, Revit

I. Introduction

Mechanical, Electrical, Plumbing, and Technology engineering design (MEPT), is an essential portion of the construction process for any type of structure. MEPT design fees can vary greatly depending on the type of project and location; market standard lies between 1-4% of the construction cost of the structure [3]. This can result in fees from as low as a couple of thousands of dollars to millions of dollars. The man-hour gross cost required to produce the design is currently not cost effective and the field is becoming more competitive every year. MEPT design has been fairly consistent for as long as construction plans have been used. The process consists of an engineering team using their education, experience, and code requirements to produce the design. Just like in any other industry, the process becomes similar to a factory line. With the introduction of codes and standards, most projects are very similar to one another with only small changes in orientation, scale, and layouts. The issue is that at the end of the project, the large majority of the design data and information is stored in the heads of the design team and as paper/electronic copies of the project plans. Every time you begin a new project, the new design team consults previous projects and experienced colleagues. The industry is becoming saturated with unused data and this trend will continue in the next years.

The introduction of Data Science will provide the ability to extract saturated data to expedite the design process, enhance decision making, provide a consistent product along all project phases [15] and ultimately, eliminate the need to redesign the same project over and over again. The goal to be accomplished is to use Data Science and more specifically machine learning to design the layout of smoke detectors using data from previous projects. A smoke detector is an electrical/low voltage fixture that is very common and is needed in almost all projects. The machine learning applications will be able to save engineering companies from a couple hundred dollars in man-hours for a small renovation job to even thousands for larger new construction projects. In addition to man-hours and due to the ability to produce a more consistent product, the engineering company will also be able to save money in change orders and litigation.

The process will consist of mining previous Revit projects using Python and creating a database on SQL. The data will then be preprocessed using statistical methods in R to isolate the most applicable parameters and transform them into a usable format. The processed data will then be applied to the most effective machine learning algorithms using Scikit-Learn and TensorFlow to produce a prediction model. The new project data will be taken out of Revit and applied to the machine learning prediction model. This produces the predicted parameter that is injected back into the Revit project using Dynamo. A prediction accuracy of greater than 95% is necessary for this proof of concept to be determined a success. The 95% accuracy shall vindicate and validate the time saved with the automatization of the model predicting and laying out the smoke detectors in Revit in lieu of the engineering team making the decision and manually laying out the smoke detectors. Once the project is completed, the project data is mined and added to the SQL database to begin the cycle again. With every project, the prediction models will continue to learn, evolve, and adapt to the engineering team's design criteria.

II. Data Set

The data set for this machine learning application consists of data mined from previous projects that the engineering company has completed using Revit. Currently, the data is not publicly available and there are no online substitutions suitable for this project. Therefore, the data used for this project was composed of a survey that was distributed to professional engineers and designers with experience in reference to the layout requirements of smoke detectors. The survey consisted of five different scenarios, each with a different room parameter. Those parameters are possible criteria to determine the presence or absence of a smoke detector in any specific room. The survey produced 330 scenarios. The scenarios consisted of the following parameters:

1. Building with or without a sprinkler system
2. Building Occupation (Residential, Medical, Office)
3. Room name
4. Room area

The survey was built on google forms, with a google sheets page producing the four scenarios and collecting the submittals. The building with or without a fire sprinkler system was randomly selected. The building occupation was randomly selected with the following options: office, medical, or residential building. For each building occupancy, a room was randomly selected from a room name bank. The room area was randomly selected from a range of 100-1000 square feet in intervals of 100.

III. Features and Processing

A. Analysis and Testing

Visualization of the data Figures 1 and 2 show that there is not a clear association between room area and the presence of smoke detectors. Data shows that there is no association between building occupancy and the presence of smoke detector as shown in Fig. 3.

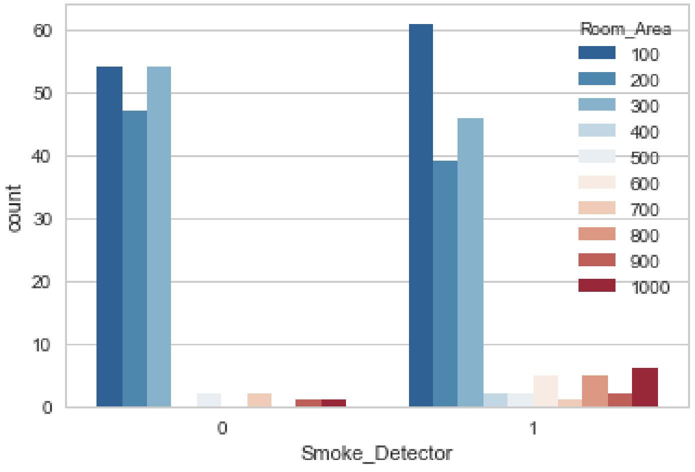

Fig. 1. Count of Smoke Detector/No Smoke Detectors per Room Area.

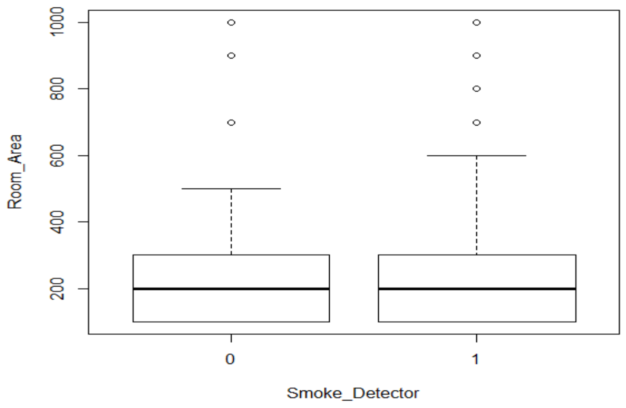

Fig. 2. Box Plot of Smoke Detector/No Smoke Detectors per Room Area.

However, building occupancy determines the categorization of the room which is a feature necessary as a predictor. In contrast, Fig. 4 shows a clear visual association between presence of fire sprinkler and the necessity for a smoke detector. To determine whether the observed dependence is due to random fluctuations, the Chi-Square test for independence was performed on the dataset using R. Due to the random sampling method, this approach was appropriate since the variables are categorical and there were at least five observations in each cell of the contingency table [13]. The null hypothesis in the tests states that there was no association between the presence of smoke detector and each one of the other variables. Then, the alternative hypothesis states that there was an association between the variables.

TABLE I

Chi-Square Test
X vs. Smoke Detector-Significance Level =.01

Classification X	λ^2	**df**	**p-value**	**Compare p with α**
Fire Sprinkler	74.162	1	2.2EE-16	p<**α**
Room Area	17.865	9	0.03677	p>**α**
RA of Corridor	10	8	0.265	p>**α**
RA of Office	10.572	9	0.3062	p>**α**
Room Area	17.865	9	0.03677	p>**α**
RA of Gym	1.7746	2	0.4118	p>**α**
RA of Bedroom	1.0	2	0.6065	p>**α**

The significance level was chosen to be 0.01.
The result p-values for each test were:

Fire sprinkler vs. SD……………	p=2.2EE-16<<0.01
Room area vs. SD……………...	p=0.037>0.01
Room area of Gym vs. SD……...	p=0.412>0.01
Room area of Corridor vs. SD….	p=0.265>0.01

Room area of Bedroom vs. SD… p=0.600>0.01

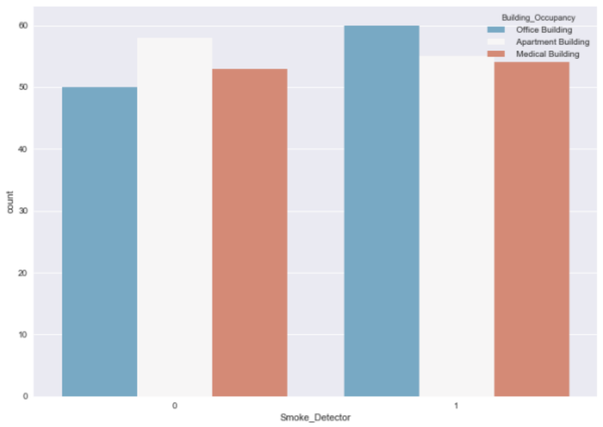

Fig. 3. Count of Smoke Detector/No Smoke Detectors per Building Occupancy.

Since the p-value for the fire sprinkler system test was less than the significance level, we rejected the null hypothesis, thus concluding that there was a relationship between the presence of a fire sprinkler and a smoke detector. For the rest of the tests we fail to reject the null hypothesis, thus concluding that there was no association between the variables.

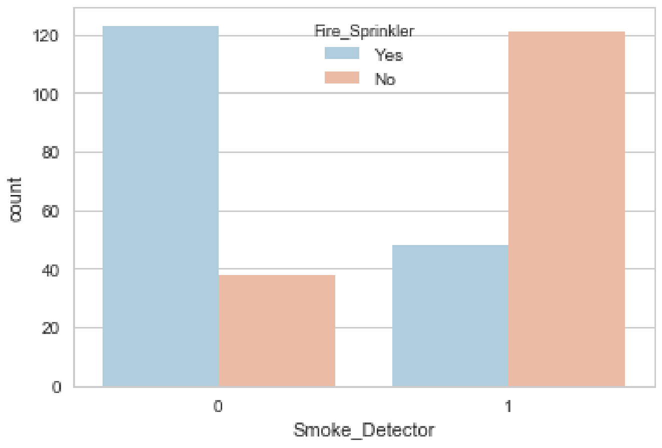

Fig. 4. Number of Smoke Detector/No Smoke Detector per Building that have Fire Sprinklers or do not have Fire Sprinklers.

B. Revit

Revit is a building information modeling software [5] used by many in the MEPT field. Revit is largely replacing AutoCAD as the primary modeling software used to produce construction documents. The major difference between Revit and AutoCAD is what makes this project possible. For instance, while AutoCAD is more of a drafting tool used to visualize the design, Revit is a database constructed to visualize the design. In addition to visualizing the design in three dimensions, you can also create a schedule or report regarding anything that you can imagine. Everything in Revit is parameterized and structured. The data that is needed to be mined is neatly structured in every project. When you start a new MEPT project, the architect provides you with a Revit model loaded with all the data needed to run it through the prediction models.

Revit is more like a blank canvas and there are multiple ways of doing the same thing. The process to mine the data out of Revit was done by using PyRevit. PyRevit is an IronPython script library for Revit [29]. Using PyRevit, a ribbon extension button was added that runs a python script that collects all the data needed to create a database regarding smoke detectors. This data is saved to a JSON file. The JSON file is then dumped into a SQL dataset. SQL (standard query language), is used to manage large amounts of data. The largest advantage of SQL is the ability to control what data gets used without deleting the unused data. The data that gets mined from Revit is usually numerous and often redundant. Once decided on what data was dependent for the prediction model, a CSV file output of the data with the specific parameters that were specifically needed could be requested. This dataset will grow with every completed project, and ultimately, fuel the accuracy of the prediction model.

C. Data Preprocessing

The data processing for some of the algorithms was required to transform the nominal attributes into binary attributes by using multiple transformation methods. First, Pandas get_dummies was used for the separation of the string values. A data frame was returned with all the possible values after splitting every string. If the text value in the original data frame at the same index contains the string (Column name/ Split values), then the value at that position is a 1, otherwise it is a 0 [9].

The next step in the preprocessing for certain models was to apply Standard Scalar from Scikit-Learn. This transformation was needed because many algorithms were limited with data non normally distributed. The numeric attributes were normalized and standardized to have zero mean and a standard deviation equal to one. The formula for z (1) below was used to calculate the standard score of each column and the columns were transformed from 1s and 0s to a normally distributed dataset. The data in the column was centered and scaled independently [24].

$$z = \frac{x-\mu}{s} \qquad (1)$$

μ is the mean of the training samples and s is the standard deviation of the training samples.

Scikit-Learn's train test split was used for the preprocessing of all the datasets except for the TensorFlow algorithm. The process was used to split the dataset into training data and test data. A dedicated data set should be used to test the model because reusing a data set to test the model after using it for training could give inaccurate results [12]. For each algorithm, the ratio of testing to training data was calculated. This ratio was dependent on the type of algorithm, with some requiring a higher number of training data. However, because of the limited dataset, a standard ratio of 30% test data versus 70% training data was applied.

Every time the code is run, the split is done randomly, and a different set of trained and tested data is produced. Splitting the data once creates results that are not replicable. This problem is solved through Scikit-learn that produces random state input with the function. The random state input is the seed used to generate the split data [25]. As long as the same random state input is used, the test train split will be replicable. This process also provides the ability to train and fit a large variety of the same data set. This will provide a more accurate precision/error rate.

The preprocessing for the TensorFlow's Kera's model can be performed using Scikit-Learn's library to transform the data, but with TensorFlow's library the model can be exported with the preprocessing included [16]. Through this method, the raw data could be directly included in the model, which is a great advantage. The data compiled was mixed data types, so the numerical fields were sorted and separated to avoid some of the overhead. The separated continuous data was then normalized and when the preprocessing was completed, the data types were combined. The preprocessing was done to both the training data and test data that were split in the beginning of the process, unlike the rest of the models.

IV. Models and Strategies

The goal for this project was to use machine learning to create the most efficient and accurate model to predict which room requires a smoke detector. This was performed by utilizing the different parameters in the dataset which were determined to be dependent.

All of the algorithms that were tested during this project were supervised learners since they had input and output data to train and test [6]. The idea is to implement this process to the industry and automatize the design development process of smoke detectors using Revit. During the remainder of the design process the layout of those smoke detectors can change. The designers/engineers/architect/owner can change the design by: wanting a smoke detector in a room that did not need one, not wanting one in a room that was showing one but did not need one, or they can find a situation where it is required and one was not shown. Once the project has been completed, this feedback will make it to the master database and machine learning model. This specific feedback is what makes this project a supervised machine learning project.

A. KNN

The KNN algorithm does not assume the value of a parameter for the purpose of analysis and is known as a lazy learning algorithm which means there is no training phase. The algorithm works by separating the data points into classes to predict the classification of a new point [7]. Among the pros of the KNN are that it is very simple to implement and explain. The algorithm is versatile in its application with classification or regression. The cons of KNN are that it is not memory efficient and in turn, the prediction phase is significantly slower when dealing with bigger datasets. It also requires a higher degree of preprocessing of the data to exclude independent data that does not play a significant role in the prediction phase and could even negatively affect the data by skewing the prediction.
The dataset was transformed using get_dummies and train_test_split as explained in features and processes. The metric used in the algorithm was Minkowski distance (see equation (2) below), the leaf size was 30, and the weight was set at uniform.

$$\left(\sum_{i=1}^{n}|x_i - y_i|^p\right)^{1/p} \qquad (2)$$

B. Naïve Bayes

Naïve Bayes classifiers is not just one type of machine learning algorithm but a group of classifiers that share a similar principle. The main principle of Bayes' Theorem is the update of the conditional probabilities by using available data. The principle can be expressed in the equation (3). The dataset used for the Naïve Bayes algorithms was transformed using get_dummies, standard scalar, and train_test_split as explained in features and processes.

The concepts needed when working with a Naïve Bayes classifier is evidence, a priori, and posteriori. Evidence is the probability of event B, Priori is the probability of A, and posteriori is the probability of A given B [2]. With the smoke detector dataset, B represents the building having fire sprinklers, a room named bedroom, and the building occupancy is a residential building. The priori is the probability of there being a smoke detector, and the posteriori is the probability that the room requires a smoke detector given the evidence.

The Gaussian Naïve Bayes algorithm assumes that the continuous data is normally distributed and the parameters (standard deviation and mean) of the evidence are estimated.

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \qquad (3)$$

Bernoulli Naïve Bayes penalizes the non-occurrence of a feature i that is an indicator for class y as shown in equation (4). [18]

$$P(x_i|y) = P(i|y)x_i + \big(1 - P(i|y)\big)(1 - x_i) \qquad (4)$$

C. Logistic Regression

The logistic regression algorithm uses a logistic function as seen in equation (5) to predict the probability of a model with a binary outcome. The binary outcome is a qualitative response (1/0, Yes/No, True/False) to some other variable that could be categorical or numerical and it analyzes the qualitative responses to the quantitative predictor variables. [23]. The dataset used for the Logistic Regression algorithms was transformed using get_dummies, standard scalar and train_test_split as explained in features and processes.

$$p_i = P(Y_i = 1|x) = \frac{e^{\beta_0+\beta_1 x}}{1+e^{\beta_0+\beta_1 x}} \qquad (5)$$

The parameters 0 and 1 of this model are estimated using a maximum likelihood method. The parameters are related to odds ratio and can be interpreted as the increase in odds of occurrence. [4]

D. Support Vector Machine (SVM)

Support Vector Machine is a discriminative classifier used for regression, classification, and outlier detection. It is an effective method for high dimensional spaces [19] which is highly applicable to our processed dataset. It is based on a set of labeled training data in the algorithm decision making. SVM is a method that with a lower computational power a considerable accuracy level is produced, with the disadvantage of not giving the probability estimates. The dataset used for the SVM algorithms was transformed using get_dummies, standard scalar and train_test_split as explained in features and processes. The algorithm maximizes the distance between the data points of different classes which increases the confidence in the classification of future data points [1].

The two parameters that had the greatest influence on the SVM algorithm accuracy were C and Gamma. The Gamma parameter is the inverse of the radius of influence and is applied to handle non-linear classifications. High values of gamma mean a close influence of samples selected by the model as support vectors and low values means ‘far’ influence [19]. The C parameter does a tradeoff of correct classification to maximize the margin of the decision’s function. The higher the C the lower the bias becomes and the higher the variance.

E. TensorFlow’s Kera

TensorFlow is Google’s new machine learning framework. It streamlines the process of preprocessing the data and then training the model using that data with Kera’s support for TensorFlow-specific functionality. TensorFlow's Kera is the implementation of the Kera’s API specification which is superior to build and train models[11]. In most problems the sequential API works by creating models’ layer-by-layer. The predictions are provided with ease and future data can be implemented into the model with ease as well. TensorFlow has a Python front end, while its back end is completely C++. TensorFlow 2.0 is graph based computation with data that flows from nodes called tensors. These tensors are multi-dimensional data arrays [27].

TensorFlow uses a different form of preprocessing from the rest of the python-based algorithms, although the output was similar to the dataset produced by Scikit-Learn’s preprocessing. Different preprocessing layers depend on the data you need to work with. The activation layers used were Relu and Sigmoid. The loss was set at binary corossentropy, the optimizer at Adam, and the metrics to accuracy. The epoch was set to three, because the dataset was fairly consistent. The epoch could be set at a higher setting depending on the variety of future datasets.

F. Random Forest

Random Forest is an algorithm that at its core has different types of decision trees working together. The trees all give a classification, and the Random Forest algorithm counts each classification as a vote. The classification with the most votes receives the final classification [20]. The preprocessing of the data is important with Random Forest, because Gini decision tree is the default type of decision tree used. The dataset used for the Random Forest algorithm was transformed using get_dummies, standard scalar and train_test_split as explained in features and processes. The Gini index in Gini decision tree and binary classification problems will be zero if all the scenarios in the columns are positive or all are negative. If the distribution of the columns is normalized, then the Gini index can be maximized [21] using the equation (6) below.

$$gini(D) = 1 - \sum_{j=1}^{n} p_j^2 \qquad (6)$$

Pj = relative frequency of class j in D.

G. Dynamo

Dynamo is a programming interface that is offered and installed with Revit. Dynamo is open source and designed to work with a workflow programming interface [10] that is user friendly for those not familiar with programming. Dynamo was used in the project to read the output of the projection of the machine learning models and translate the data to actual smoke detectors on the plan. The native interface allows for python scripts to also be utilized.

V. Results and Analysis

The algorithms applied to the data produced predictions in the layout of smoke detectors with high levels of accuracy. The results prove that the application of machine learning will be able to thrive with this type of data. To evaluate the performance of the model accuracy, precision, recall, and F1-score metrics were calculated.

TABLE II [14]

	Predicted Class		
Actual Class		Class=Yes	Class=No
	Class=Yes	True Positives (TP) - Smoke Detector was correctly predicted	False Negative (FN) - Failed to predict Smoke Detector
	Class=No	False Positive (FP) – Incorrectly predicted Smoke Detector	True Negative (TN) - Predicted the absence of Smoke Detector

$$Precision = \frac{TP}{TP+FP} \qquad (7)$$

$$Recall = \frac{TP}{TP+FN} \qquad (8)$$

$$F1 - Score = \frac{2*Recall*Precision}{Recall+Precision} \qquad (9)$$

In addition to the initial train and fit of the prediction model, the train test fit function was used to produce 150 different train and test data sets trained and fit to a prediction model. The prediction model’s classification report scores were recorded and the minimum, maximum, and average are shown to provide a more accurate precision score.

A.KNN

The KNN algorithm using the Minkowski metric and five nearest neighbors resulted in a 98% accuracy rate with a randomly split train and test split (see Table III). The five nearest neighbors were calculated to provide the minimal error rate with the least number of nearest neighbors. The same error rate would have been achieved with seven nearest neighbors as shown in Fig 5.

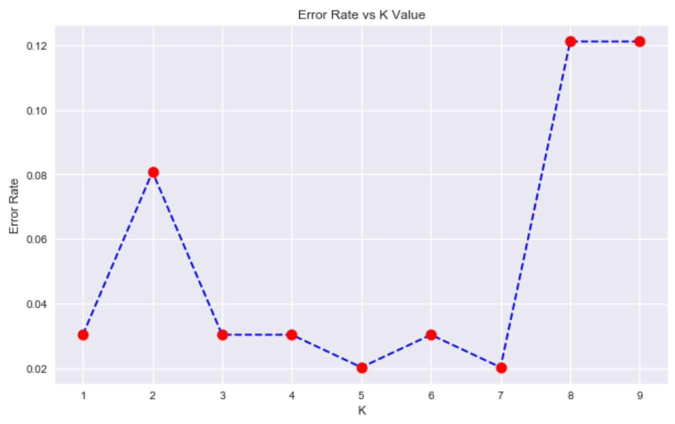

Fig. 5. Count of Nearest neighbors (K) and the associated Error Rate.

TABLE III

KNN Classification Report				
	Precision	Recall	F1-Score	Support
No Smoke Detector	1.00	0.96	0.98	46
Yes Smoke Detector	0.96	1.00	0.98	53
Avg/Total	0.98	0.98	0.98	99

The results from the sampling using the 150 random test and train split data sets exceeded the minimum threshold specified to vindicate the project. The average for the precision resulted in 96% (see table IV) which was lower than the 98% (see table III) calculated in the individual classification report. Even though the minimum of the precision score lies at 85%, the overall majority of the precision scores are largely distributed between 94% and 100%, as evident in Fig. 6.

TABLE IV

KNN Classification Report of 150 Random States				
	Precision	Recall	F1-Score	Support
Minimum	0.85	0.85	0.87	39
Maximum	1.00	1.00	1.00	60
Mean/Avg	0.96	0.96	0.96	49.5

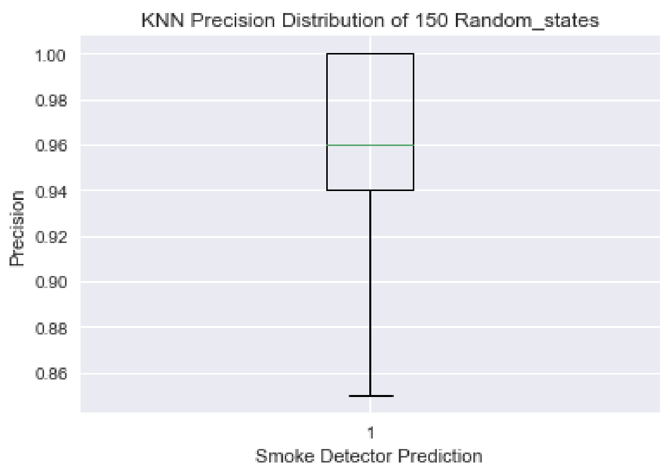

Fig. 6. Boxplot visualizing the distribution of the precision scores for KNN when running the model with 150 different random states.

B. *Random Forest*

As previously stated, Random Forest is a group of decision trees that assigns votes for classification, with the category with the most votes being the final decision [26]. The Random Forest criterion used was a Gini Index Decision Tree, so the individual Gini Index Decision Tree algorithm was also applied to the data as a reference. The overall precision of Random Forest in Table VI had an increase of two percent when compared to the Decision Tree in Table V; this increase is equivalent to a building with 500 rooms having ten rooms miscategorized.

TABLE V

Decision Tree Classification Report				
	Precision	Recall	F1-Score	Support
No Smoke Detector	0.92	0.98	0.95	46
Yes Smoke Detector	0.98	0.92	0.95	53
avg/total	0.95	0.95	0.95	99

TABLE VI

Random Forest Classification Report				
	Precision	Recall	F1-Score	Support
No Smoke Detector	0.96	0.98	0.97	46
Yes Smoke Detector	0.98	0.98	0.97	53
avg/total	0.97	0.97	0.97	99

The results from the sampling using the 150 random test and train split data sets exceeded the minimum threshold specified to vindicate the project. The average for the precision resulted in 96% (see table VII) which was lower than the 97% (see table VI) calculated in the individual classification report. Fig. 7 shows that the overall majority of the precision scores are distributed between 93% and 98%.

TABLE VII

Random Forest Classification Report of 150 Random States

	Precision	Recall	F1-Score	Support
Minimum	0.82	0.81	0.88	39
Maximum	1.00	1.00	1.00	60
Mean/Avg	0.96	0.96	0.96	49.5

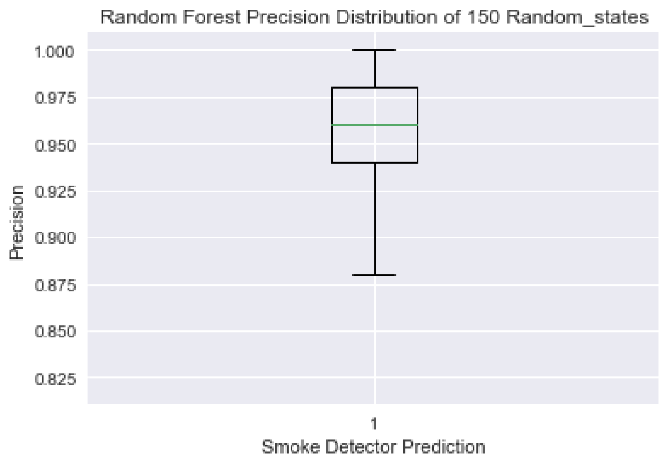

Fig. 7. Boxplot visualizing the distribution of the precision scores for Random Forest when running the model with 150 different random states.

C. *Naïve Bayes*

Naïve Bayes includes three types of algorithms: Gaussian, Multinomial, and Bernoulli. The highest accuracy was obtained through applying the Bernoulli Naïve Bayes algorithm with an accuracy of 94% (see Table X), while Gaussian (see Table VII) and Multinomial (seeTable VIII) fell short with sub 90% which is required to remain as a contender. Gaussian fared rather poorly with a 73% accuracy level which automatically disqualifies its use from future testing with larger datasets. The Multinomial resulted in 84% which will not be applicable with a medium sized data set like the one used for this project but can be shelved for testing with a larger dataset.

TABLE VII

Gaussian Naïve Bayes Classification Report

	Precision	Recall	F1-Score	Support
No Smoke Detector	0.72	0.60	0.66	43
Yes Smoke Detector	0.73	0.82	0.77	56
avg/total	0.73	0.73	0.72	99

TABLE IX

Multinomial Naïve Bayes Classification Report

	Precision	Recall	F1-Score	Support
No Smoke Detector	0.85	0.77	0.80	43
Yes Smoke Detector	0.83	0.89	0.86	56
avg/total	0.84	0.84	0.84	99

TABLE X

Bernoulli Naïve Bayes Classification Report

	Precision	Recall	F1-Score	Support
No Smoke Detector	0.93	0.93	0.93	43
Yes Smoke Detector	0.95	0.95	0.95	56
avg/total	0.94	0.94	0.94	99

The results from the sampling using the 150 random test and train split data sets were right under the minimum threshold specified to vindicate the project. The average for the precision resulted in 93% (see table XI) which was lower than the 94% (see table X) calculated in the individual classification report. Fig. 8 shows that the overall majority of the precision scores are largely distributed between 89% and 96%.

TABLE XI

Naïve Bayes Classification Report of 150 Random States

	Precision	Recall	F1-Score	Support
Minimum	0.78	0.79	0.82	39
Maximum	1.00	1.00	0.99	60
Mean/Avg	0.93	0.92	0.92	49.5

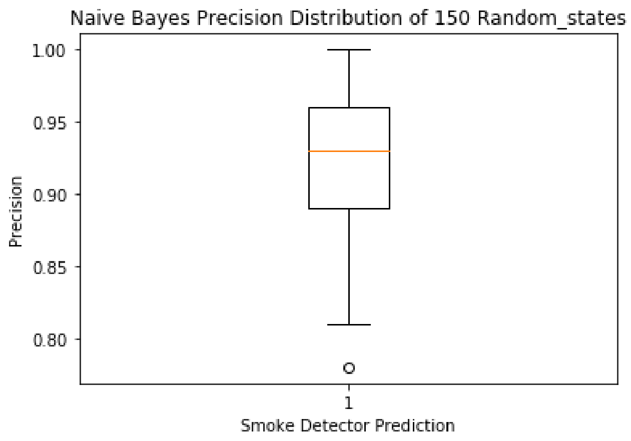

Fig. 8. Boxplot visualizing the distribution of the precision scores for Naïve Bayes when running the model with 150 different random states.

D. *Regression*

The Regression algorithm had an overall accuracy of 91% (see Table XII), which was the lowest of all the algorithms tested. The R_2 score was calculated to be around 94%, which is the fraction of the variance in the response variable explained by the variation of the manipulated variable. The higher the R_2 the better the model fits the data. A future project will include retesting the Regression algorithm with a larger dataset, which can produce a higher R_2 score and accuracy level.

TABLE XII

Regression Classification Report

R_2 Score = 0.9437

	Precision	Recall	F1-Score	Support
No Smoke Detector	0.87	0.93	0.90	44
Yes Smoke Detector	0.94	0.89	0.92	55
avg/total	0.91	0.91	0.91	99

In the case of Regression, sampling with 150 random test and train split data sets yield a slightly lower result than the minimum threshold specified to vindicate the project. The average for the precision was 89% (see table XIV) which was lower than the 91% (see table XIII) calculated in the individual classification report. Fig. 9 shows that the overall majority of the precision scores are distributed between 86% and 92%.

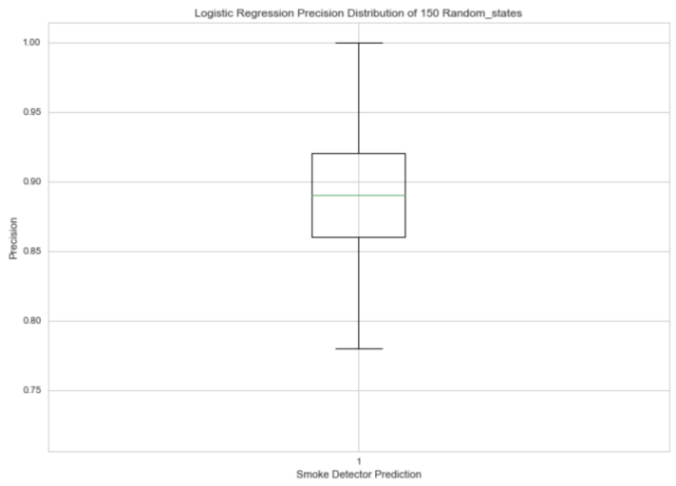

Fig. 9. Boxplot visualizing the distribution of the precision scores for Logistic Regression when running the model with 150 different random states.

TABLE XIV

Logistic Regression Classification Report of 150 Random States

	Precision	Recall	F1-Score	Support
Minimum	0.72	0.76	0.77	39
Maximum	1.00	1.00	0.95	60
Mean/Avg	0.89	0.89	0.89	49.5

E. *SVM*

The function GridSearchCV was used with a verbose=3 to calculate the highest C (C=0.1,1,10,100,1000) and Gamma (Gamma=1,0.1,0.01,0.001,0.0001) parameters. The function calculated that the highest score was obtained when C=1000 and Gamma=0.01. This provided an increase of 2% from the standard parameter's accuracy rate in the SVM model from 93% to 95% as shown in Table XV.

TABLE XV

SVM Classification Report

	Precision	Recall	F1-Score	Support
No Smoke Detector	0.93	0.95	0.94	44
Yes Smoke Detector	0.96	0.95	0.95	55
avg/total	0.95	0.95	0.95	99

In SVM, the results from the sampling with the 150 random test and train split data sets exceeded the minimum threshold specified to vindicate the project. The average for the precision was 96% (see Table VXI), which was higher than the 95% (see Table VX) calculated in the individual classification report. Fig. 10 shows that the overall majority of the precision scores are distributed between 93% and 98%.

TABLE XVI

SVM Classification Report of 150 Random States

	Precision	Recall	F1-Score	Support
Minimum	0.83	0.79	0.86	39
Maximum	1.00	1.00	0.95	60
Mean/Avg	0.96	0.95	0.95	49.5

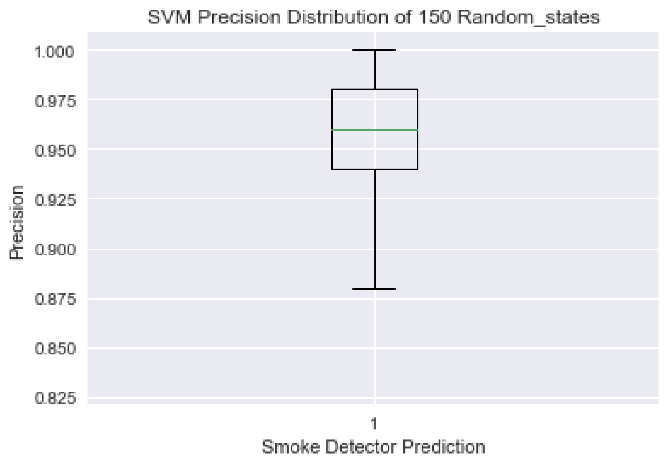

Fig. 10. Boxplot visualizing the distribution of the precision scores for SVM when running the model with 150 different random states.

F. *TensorFlow's Keras*

The parameter applied was the number of epoch that the model needed to perform. It was evident that only three epoch were needed to maximize the accuracy and minimize the loss. An epoch is one pass through the dataset. The test accuracy was greater than the threshold of 95% required to vindicate the automatization of the design of smoke detectors.

TABLE XVII

TensorFlow's Kera's Prediction Output	
Test loss	0.100097664091
Test Accuracy	0.969230771065
Smoke Detector: 99.96% | Actual outcome: Smoke Detector Smoke Detector: 0.00% | Actual outcome: No Smoke Detector Smoke Detector: 0.05% | Actual outcome: No Smoke Detector Smoke Detector: 0.00% | Actual outcome: No Smoke Detector Smoke Detector: 0.06% | Actual outcome: No Smoke Detector Smoke Detector: 0.04% | Actual outcome: No Smoke Detector Smoke Detector: 100.00% | Actual outcome: Smoke Detector Smoke Detector: 85.90% | Actual outcome: Smoke Detector Smoke Detector: 99.99% | Actual outcome: Smoke Detector Smoke Detector: 100.00% | Actual outcome: Smoke Detector Smoke Detector: 85.90% | Actual outcome: Smoke Detector Smoke Detector: 0.07% | Actual outcome: No Smoke Detector	

G. *Revit/Dynamo*

The predictions were outputted from the models and directly read into Revit through Dynamo (see Fig. 13). Dynamo was used to match the rooms categorized as to have a smoke detector with the rooms in the Revit model (see Fig. 11). When the rooms were identified, a smoke detector was located within the room at the ceiling height (see Fig. 12).

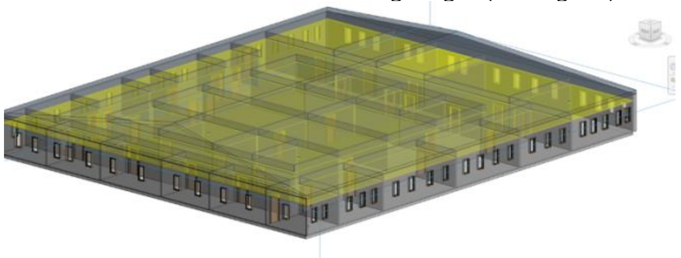

Fig. 11. Test Building used for smoke detector layout.

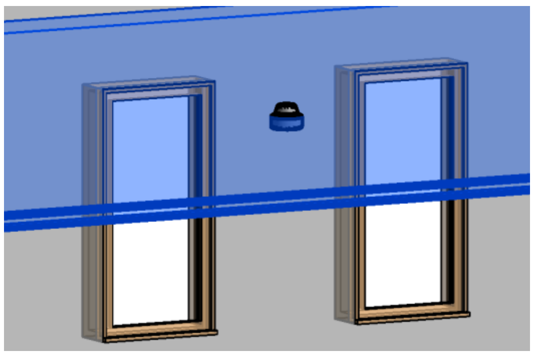

Fig. 12. Smoke Detector placed in the ceiling of a room in Revit.

Fig. 13. Dynamo block programming to read from file and lay the smoke detectors in the rooms specified using the prediction model.

VI. Conclusion

Throughout this project, the goal was to identify the most efficient and precise algorithm to predict if a room requires a smoke detector. The metric that was established to support the automatization of the design and layout of the fixture within Revit was an accuracy of 95% or greater. In Table XVIII the mean precision scores are provided. The algorithms that met the metric were Random Forest, KNN, SVM, and TensorFlow's Keras. Logistic Regression and Naïve Bayes fell short of the minimum performance metric. Logistic Regression does not work well with a large quantity of categorical parameters like the one applied in this project. The algorithm can be given a second chance with a larger dataset once it is implemented in an industrial setting. The Naïve Bayes algorithm's kryptonite is a category in the test data that is missing from the training data. The prediction for the missing data will be set at 0 which is called Zero Frequency [8]. The solution that would alleviate the issue in future applications would be the use of Laplace estimation and larger datasets that are not missing any categories.

As expected with any machine learning application, the preprocessing section of the project was intense. The dataset was mixed with categorical, numerical, and continuous data. Transforming the data into a working algorithm to optimize its ability to produce predictions was like walking a tight rope.

Ideally, an existing database of previous projects would have been preferred, however, none are publicly available at the time the project was conducted. If given a larger database, the accuracy in the layout of smoke detectors would have been significantly higher with the TensorFlow's Kera's algorithm, and at least a 1% to 3% increase in accuracy with the rest of the algorithms.

Although Random Forest tied with KNN and TensorFlow's Keras with a 96% accuracy, the use of this algorithm should be minimized. The runtime needed to produce the prediction was significantly longer than KNN and not as consistent and versatile as TensorFlow's Keras. The most promising algorithm is TF Keras, because with every new submission of data, it can evolve and learn the most efficiently. The interaction with Revit and Dynamo were also satisfactory. Having the ability to visualize the output of the prediction models using their interactive programming was very impressive and accurate.

TABLE XVIII

Algorithm	Mean Precision	Mean Recall	Mean F1-Score	Support
Decision Tree	0.96	0.96	0.96	49.5
Log. Regression	0.89	0.89	0.89	49.5
Naïve Bayes	0.93	0.92	0.92	49.5
SVM	0.96	0.95	0.95	49.5
TF's Keras	0.97	-	-	-
KNN	.96	.96	.96	49.5

Finally, Revit streamlined the process by being able to natively use Python to mine the data in the preprocessing, and it's use of Python with Dynamo on the post. The data mined from Revit will always be consistent and would only require minor adjustments from project to project. This would mean that from the moment the new project data is mined to the moment the fixtures are laid out in Revit could take essentially a few minutes. In a large scale, this could represent an entire work week of man-hours needed for an engineer to lay out all the smoke detectors of a building. This will result in a company saving thousands of dollars. Once it is applied across the board with other fixture types, it will translate to hundreds of thousands of dollars, a faster project turnaround, and a more consistent product.

VII. Future Endeavors

With a successful proof of concept, the future is limitless. The project shows that automation of the design process of MEPT construction is not only possible, but inevitable. The next step includes extending the application to all technology and electrical based fixtures that typically have one per room. This will be followed by placing electrical and technology fixtures that have more than one in each room and are laid out in specific locations. Once this is mastered, we can move on to mechanical fixtures, prediction of heat loads, and CFM requirements per room. The design of duct and piping will inevitably follow, although it may be an order of magnitude more challenging [17]. I look forward to conquering these challenges in the near future.

References

[1] 1.4. Support Vector Machines. (n.d.). Retrieved from https://scikit-learn.org/stable/modules/svm.html#svm-mathematical-formulation.

[2] 1.9.Naïve Bayes. (n.d.). Retrieved from https://scikit-learn.org/stable/modules/Naïve_bayes.html.

[3] Admin, W. (2013, February 4). Typical MEP Design Fees Archives. Retrieved from https://www.engineeringdesignresources.com/tag/typical-mep-design-fees/.

[4] Analytics Vidhya Content Team, & Analytics Vidhya Content. (2019, September 3). Simple Introduction to Logistic Regression in R. Retrieved from https://www.analyticsvidhya.com/blog/2015/11/beginners-guide-on-logistic-regression-in-r/.

[5] BIM Software Features: Revit 2019. (n.d.). Retrieved from https://www.autodesk.com/products/revit/features.

[6] Bishop, Christopher M. (2016). Pattern Recognition And Machine Learning. Place of publication not identified: Springer-Verlag, New York. .

[7] Bronshtein, A. (2019, May 6). A Quick Introduction to K-Nearest Neighbors Algorithm. Retrieved from https://blog.usejournal.com/a-quick-introduction-to-k-nearest-neighbors-algorithm-62214cea29c7.

[8] Chauhan, G. (2018, October 8). All about Naive Bayes. Retrieved from https://towardsdatascience.com/all-about-naive-bayes-8e13cef044cf.

[9] Day, K. D. in, Kartikaybhutani, & Day, D. in. (2019, August 23). Python: Pandas Series.str.get_dummies(). Retrieved from https://www.geeksforgeeks.org/python-pandas-series-str-get_dummies/.

[10] Dynamo. (n.d.).Retrieved from https://knowledge.autodesk.com/support/revit-products/getting-started/caas/CloudHelp/cloudhelp/2019/ENU/Revit-Customize/files/GUID-F45641B0-830B-4FF8-A75C-693846E3513B-htm.html.

[11] Getting started with the Keras Sequential model. (n.d.). Retrieved from https://keras.io/getting-started/sequential-model-guide/.

[12] Grus, J. (2019). Data science from scratch: first principles with Python. Sebastopol, CA: OReilly Media.

[13] Hastie, T., Friedman, J., & Tisbshirani, R. (2017). The Elements of statistical learning: data mining, inference, and prediction. New York: Springer.

[14] Joshi, R., Ahmad, F., Joseph, Luna, A., Wayne, Arjun, … Berg, M. (2016, November 11). Accuracy, Precision, Recall & F1 Score: Interpretation of Performance Measures. Retrieved from https://blog.exsilio.com/all/accuracy-precision-recall-f1-score-interpretation-of-performance-measures/.

[15] Leskovec, J., Rajaraman, A., & Ullman, J. D. (2016). Mining of massive datasets. Delhi: Cambridge University Press.

[16] Load CSV data : TensorFlow Core. (n.d.). Retrieved from https://www.tensorflow.org/tutorials/load_data/csv.

[17] Mata, A. M. A. image of A. M. A. M. A. (2017). AI MEP: Beyond Dynamo, Machine Learning to predict HVAC performance. Retrieved from https://www.autodesk.com/autodesk-university/class/AI-MEP-Beyond-Dynamo-Machine-Learning-predict-HVAC-performance-2017.

[18] Naïve Bayes Classifiers. (2019, January 14). Retrieved from https://www.geeksforgeeks.org/Naïve-bayes-classifiers/.

[19] Patel, S. (2017, May 4). Chapter 2 : SVM (Support Vector Machine) - Theory. Retrieved from https://medium.com/machine-learning-101/chapter-2-svm-support-vector-machine-theory-f0812effc72.

[20] Random Forests Classifiers in Python. (n.d.). Retrieved from https://www.datacamp.com/community/tutorials/random-forests-classifier-python.

[21] Random Forests Leo Breiman and Adele Cutler. (n.d.). Retrieved from https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm.

[22] RBF SVM parameters. (n.d.). Retrieved from https://scikit-learn.org/stable/auto_examples/svm/plot_rbf_parameters.html.

[23] Scheaffer, R. L., McClave, J. T., & Mulekar, M. S. (2011). Probability and statistics for engineers. Boston, MA: Brooks/Cole.

[24] Scikit-Learn.preprocessing.StandardScaler. (n.d.). Retrieved from https://scikit-learn.org/stable/modules/generated/Scikit-Learn.preprocessing.StandardScaler.html.

[25] Sklearn.model_selection.train_test_split. (n.d.). Retrieved from https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html.

[26] Tamhane, A. C., & Dunlop, D. D. (2006). Statistics and Data Analysis. Beijing: Higher Education Press.

[27] Tan, P.-N., Steinbach, M., & Kumar, V. (2006). Introduction to data mining. Harlow: Pearson.

[28] TensorFlow Core. (n.d.). Retrieved from https://www.tensorflow.org/tutorials/.

[29] The pyRevit IronPython Script Library. (n.d.). Retrieved from https://thebuildingcoder.typepad.com/blog/2016/04/the-pyrevit-ironpython-script-library.html.