INSTITUTE FOR ADVANCED COMPUTING AND SOFTWARE DEVELOPMENT

AKURDI, PUNE

Documentation On

"Prediction of Room Occupancy based on Environmental Factors"

PG-DBDA FEB 2020

Submitted By:

Group No: 5

Avinashkumar Mishra - 1307

Abhishek Chaudhari – 1311

Project Guide:

Mr. Akshay Tilekar (External Guide)

Mrs. Tejal Mehetre (Internal Guide)

Mr. Rahul Pund (Internal Guide)

Mr. Manish (Internal Guide)

Mr. Prashant Karhale
Centre Coordinator

INDEX
Table of Contents

Acknowledgement

First and Foremost we thank to **almighty God** for giving us the support, strength, positive spirit and talent to do this project.

“The satisfaction and euphoria that accompany the successful completion of any task would be incomplete without the mention of the people who made it possible”.

“This work is the result of inspiration, motivation, knowledge, interest, support, guidance, cooperation and efforts by many people at different levels. We are indebted to all of them”.

We would also like to take this opportunity to acknowledge the valuable contributions made by our **family members** by supporting and motivating us in every walk of life.

We are thankful to **Mr. Prashant Karhale,** Centre Co-ordinator IACSD CDAC, Akurdi Pune for providing the opportunity, infrastructure and facilities for entire work.

We would like to express our great appreciation to our project Guide **Mr. Akshay Tilekar,** Internal Project Guides **Mrs. Tejal, Mr. Rahul, and Mr. Manish** for their valuable and constructive suggestions during the planning and development of this project.

We also thank all staff members from the IACSD who in some way or other

Helped us in completion of this project.

We cannot conclude our acknowledgement without expressing our thanks to our friends who helped us directly or indirectly during the course of this project.

Feedback for improving the contents of the report would be more than welcome.

CHAPTER 1

INTRODUCTION

If you ever had to advertise a house or apartment for rent, it would be useful and profitable to know in advance as to how likely someone is probably going to 'occupy' that room.

What makes someone buy a room or a house? Various factors like how hot or cool the room is, or how well the room is lit affects that decision. Usually our brains do this analysis when we search rooms but why not "science" it up a bit and let Machine Learning take care of it? We shall use Python to model this.

This predictions might help Heating, Ventilating and Air Conditioning. For instance, we are using sensors like thermostats to get information about the environment and with that info our system decides to heat or not situation.

If no occupants are in the room, then utilization of energy resources will waste our wealth, in prior if we are getting the prediction of room occupied or not we can adjust the energy resources in our convenience which will save our wealth and the energy resources also.

E.g. If the room is vacant then we can switch off the AC, lights, fans and if the room is occupied we will adjust the AC based on occupants in the environment , which also helpful to save the money.

Occupancy detection in buildings estimated to save energy in the order of 30 to 42%, detecting the presence of the occupants without using a camera is very interesting due to privacy concerns.

We'll use a famous dataset called 'Occupancy Detection', and like it's not so cool name, it has features such as Temperature, Humidity, Light and other …..Stuff. Good thing about Machine Learning is we don't truly have to 'understand' the meaning of every feature in detail as long as we have a rough idea as to how it affects our response variable (the thing we predict).

1.1 Purpose

The objective of the project is to predict whether the room is occupied or not based on the Environmental factors in that space for eg. Temperature,CO2, Humidity, Light etc. Clearly the problem tell us that classification model has to be build so that it can predict the classes

accurately with the help of Machine learning models like Logistic Regression, SVM,KNN Classifier, Random Forest, Ridge Classifier etc.

This is Standard Supervised Classification task. A classification problem where we have to predict whether the room is occupied or not

1.2 Scope

Initial functional requirement will be:-

- Understanding the data to get the proper Insights.
- Selecting the algorithm meeting requirement.
- Hyper tuning the parameters of the algorithm.
- Testing the algorithm with test sets.
- Analyse the results and if necessary do the changes in algorithm.

1.2.1 Initial non Functional requirement will be:

Major observation from the data

1. The occupancy i.e. dependent variable has negative class i.e. 0 having 15810 Data points whereas the positive class have 4750 observations.

2. The Humidity ratio is derived from Humidity and Temperature column.

3. The Light and CO2 variables have high values ranging from 400-2000.

4. During office hours i.e. from 7 am to 18.00 pm there have maximum chances for the Room to be occupied.

1.3 Machine learning life cycle:-

Machine learning has given the computer systems the abilities to automatically learn without being explicitly programmed. But how does a machine learning system work? So, it can be described using the life cycle of machine learning. Machine learning life cycle is a cyclic process to build an efficient machine learning project. The main purpose of the life cycle is to find a solution to the problem or project.

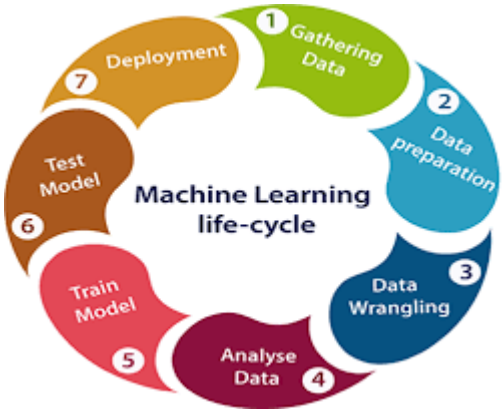


Fig.1 Machine learning lifecycle.

1.3.1 Machine learning Life Cycle Model

Gathering Data:

Data Gathering is the first step of the machine learning life cycle. The goal of this step is to identify and obtain all data-related problems.

In this step, we need to identify the different data sources, as data can be collected from various sources such as files, database, internet, or mobile devices. It is one of the most important steps of the life cycle. The quantity and quality of the collected data will determine the efficiency of the output. The more will be the data, the more accurate will be the prediction.

This step includes the below tasks:

Identify various data sources

Collect data

Integrate the data obtained from different sources

By performing the above task, we get a coherent set of data, also called as a dataset. It will be used in further steps.

Data Preparation:-

a. Data exploration:

It is used to understand the nature of data that we have to work with. We need to understand the characteristics, format, and quality of data. A better understanding of data leads to an effective outcome. In this, we find Correlations, general trends, and outliers.

b. Data pre-processing:

Now the next step is pre-processing of data for its analysis.

Data Wrangling:

Data wrangling is the process of cleaning and converting raw data into a useable format. It is the process of cleaning the data, selecting the variable to use, and transforming the data in a proper format to make it more suitable for analysis in the next step. It is one of the most important steps of the complete process. Cleaning of data is required to address the quality issues.

It is not necessary that data we have collected is always of our use as some of the data may not be useful. In real-world applications, collected data may have various issues, including:

Missing Values

Duplicate data

Invalid data

Noise

So, we use various filtering techniques to clean the data.

It is mandatory to detect and remove the above issues because it can negatively affect the quality of the outcome.

Data Analysis:

Now the cleaned and prepared data is passed on to the analysis step. This step involves:

Selection of analytical techniques

Building models

Review the result

The aim of this step is to build a machine learning model to analyze the data using various analytical techniques and review the outcome. It starts with the determination of the type of the problems, where we select the machine learning techniques such as Classification, Regression, Cluster analysis, Association, etc. then build the model using prepared data, and evaluate the model.

Hence, in this step, we take the data and use machine learning algorithms to build the model.

Train Model:

Now the next step is to train the model, in this step we train our model to improve its performance for better outcome of the problem.

We use datasets to train the model using various machine learning algorithms. Training a model is required so that it can understand the various patterns, rules, and, features.

Test Model:-

Once our machine learning model has been trained on a given dataset, then we test the model. In this step, we check for the accuracy of our model by providing a test dataset to it.

Testing the model determines the percentage accuracy of the model as per the requirement of project or problem.

Deployment:-

The last step of machine learning life cycle is deployment, where we deploy the model in the real-world system.

If the above-prepared model is producing an accurate result as per our requirement with acceptable speed, then we deploy the model in the real system. But before deploying the project, we will check whether it is improving its performance using available data or not. The deployment phase is similar to making the final report for a project.

1.4 Literature Review

1 https://www.trueoccupancy.com/occupancy-sensor-guide

a. What are occupancy sensors?

An occupancy sensor is an electronic device that detects the presence of a person. There are various occupancy sensing technologies, but the most common are infrared, microwave, ultrasonic, and video image processing. Occupancy sensors are usually connected to a building's Internet of Things (IoT) network and feed data back to alarms or building automation systems for lighting control, ventilation, and zone access.

b. How do occupancy sensors work?

Occupancy sensors work by detecting motion and changes in their environment; each sensing technology does this in a different way, for example:

Basic PIR (passive infrared) sensors detect changes in temperature (body heat).

Ultrasonic sensors emit high-frequency sound waves, outside of human hearing range, and use the Doppler effect of returning sounds waves to detect people.

Time of flight infrared sensors use a similar principle as ultrasonic sensors; however, instead of sound, they use infrared light. By emitting invisible infrared light, AI on board these sensors learns their surroundings and detect changes when people pass by.

These sensors are typically installed in strategic locations to pick up motion in high-traffic or isolated areas. Designed to be discreet, they are typically ceiling-mounted devices that detect individuals and groups entering or exiting a zone – building, floor, room, etc.

The data collected is then transmitted to a cloud-based platform where AI algorithms can calculate real-time occupancy levels and space utilisation, firing off pre-determined commands to the connected automated devices such as lighting or HVAC systems.

CHAPTER 2

OVERALL DESCRIPTION

Our aim from the project is to make use of pandas, matplotlib, & seaborn libraries from python to extract the meaningful insights from the data and classifier models like Random forest, SVM, Logistic Regression, KNN, Ridge Classifier & scikit-learn libraries for machine learning.

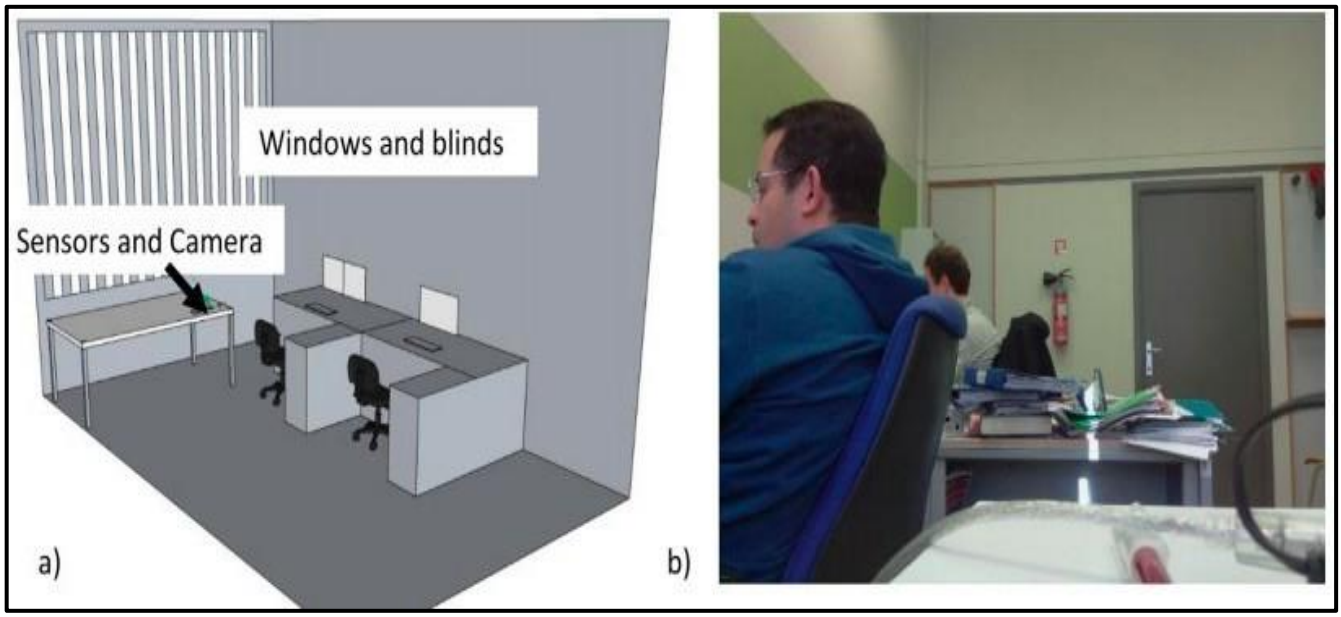
Secondly, to learn how to hyper tune the parameters using grid search for every machine learning model.

And in the end, to predict whether the room is occupied or not with that machine learning algorithm which gives maximum accuracy.

Attributes in the dataset

- Date and timestamp.
- Temperature in Celsius.
- Relative humidity as a percentage.
- Light measured in lux.
- Carbon dioxide measured in parts per million.
- Humidity ratio, derived from temperature and relative humidity measured in kilograms of water vapour per kilogram of air.
- Occupancy as either 1 for occupied or 0 for not occupied.

2.1 Data Collection and Setup:



An office room […] was monitored for the following variables: temperature, humidity, light and CO2 levels. A microcontroller was employed to acquire the data. A ZigBee radio was connected to it and was used to transmit the information to a recording station. A digital camera was used to determine if the room was occupied or not. The camera time stamped pictures every minute and these were studied manually to label the data.

2.2 Functional requirement specification:

Use case: - Prediction of room occupancy based on Environmental factor

Brief Description:

Understanding the problem statement is the first and foremost step. This would help you give an intuition of what you will face ahead of time. Let us see the problem statement .It is a classification problem where we have to predict whether the room is occupied or not. In a classification problem, we have to predict discrete values based on a given set of independent variable(s). Classification can be of two types:

Binary Classification: In this classification we have to predict either of the two given classes. For example: classifying the gender as male or female, predicting the result as win or lose, etc. Multiclass Classification: Here we have to classify the data into three or more classes. For example: classifying a movie's genre as comedy, action or romantic, classify fruits as oranges, apples, or pears, etc.

CHAPTER 3

Data Processing and Data Visualisation

Data Processing:-

The sample data is as below:-

Index	date	emperatur	Humidity	Light	CO2	HumidityRatic	Occupancy
0	2015-02-04 17:51:00	23.18	27.272	426	721.25	0.00479299	1
1	2015-02-04 17:51:59	23.15	27.2675	429.5	714	0.00478344	1
2	2015-02-04 17:53:00	23.15	27.245	426	713.5	0.00477946	1
3	2015-02-04 17:54:00	23.15	27.2	426	708.25	0.00477151	1
4	2015-02-04 17:55:00	23.1	27.2	426	704.5	0.00475699	1
5	2015-02-04 17:55:59	23.1	27.2	419	701	0.00475699	1
6	2015-02-04 17:57:00	23.1	27.2	419	701.667	0.00475699	1
7	2015-02-04 17:57:59	23.1	27.2	419	699	0.00475699	1
8	2015-02-04 17:58:59	23.1	27.2	419	689.333	0.00475699	1
9	2015-02-04 18:00:00	23.075	27.175	419	688	0.00474535	1

Fig.2:- Snapshot of the data

Here we can see that the date variable is in string format, so while loading the data we have to take care of the date variable, it should be in python date time object. Further we can see that the occupancy is measure at different time stamp. **So it is interesting to find out at which hours the room is occupied mostly.**

Descriptive Statistics:

Index	emperatur	Humidity	Light	CO2	HumidityRatio	Occupancy
count	20560	20560	20560	20560	20560	20560
mean	20.9062	27.6559	130.757	690.553	0.00422831	0.231031
std	1.05531	4.98215	210.431	311.201	0.000767869	0.421503
min	19	16.745	0	412.75	0.00267413	0
25%	20.2	24.5	0	460	0.00371902	0
50%	20.7	27.29	0	565.417	0.00429189	0
75%	21.525	31.29	301	804.667	0.00483177	0
max	24.4083	39.5	1697.25	2076.5	0.00647601	1

Fig3. Descriptive statistics

We can see that the light variable has 0 observation also, that means may be during night period lights are off .We can see the value of CO2 are very high as compared to others ,The humidity ration has low values , this is due to the variable is derived from ratio of Humidity and temperature.

3.1 Data Visualisation:-

a. Correlation Heat map:-

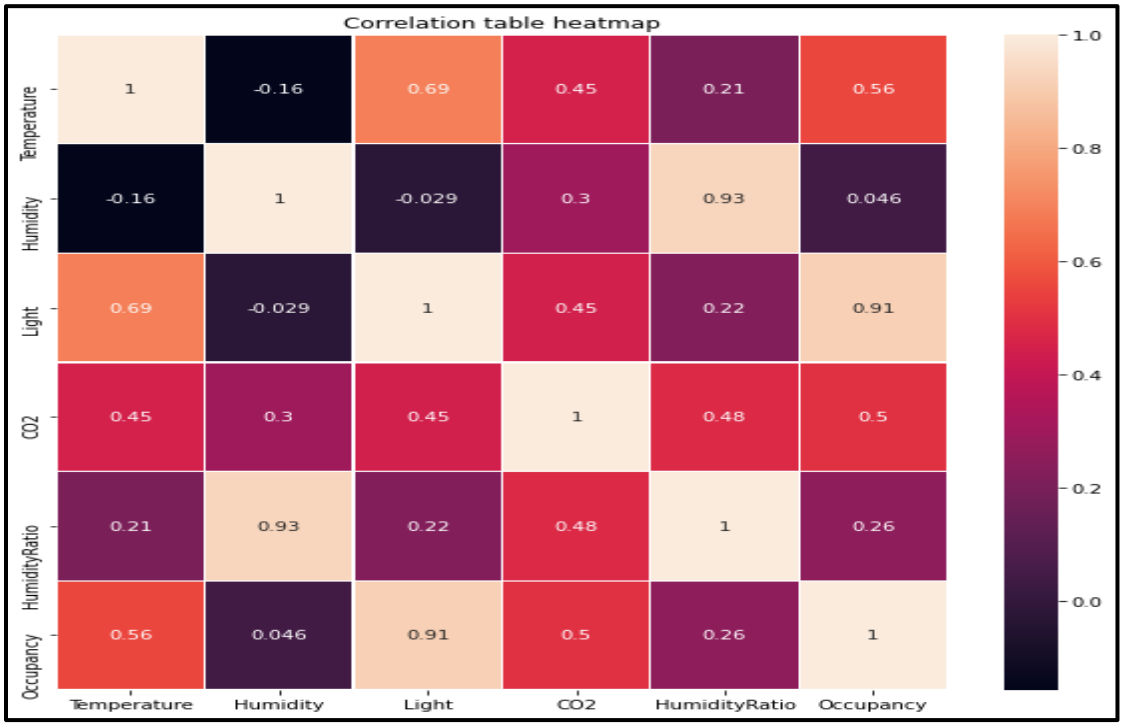


Fig.4:- Correlation heatmap

The Light variable is the highest correlation with occupancy i.e. 0.91

b. Distribution plot:-

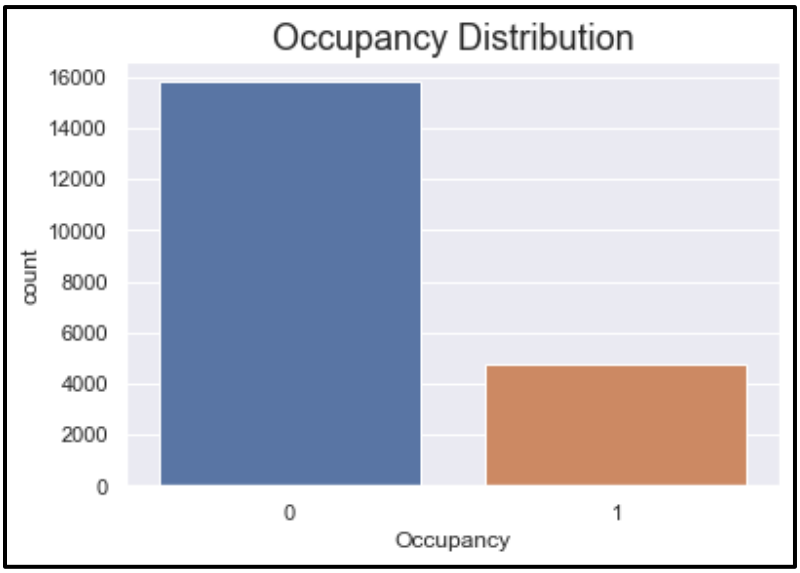


Fig 5. Distribution plot

We can see that the not occupied class has more variables and the data is showing positively skewed pattern.

We can check whether the time stamp shows any significant pattern or not.

c. Time vs Density Histogram:-

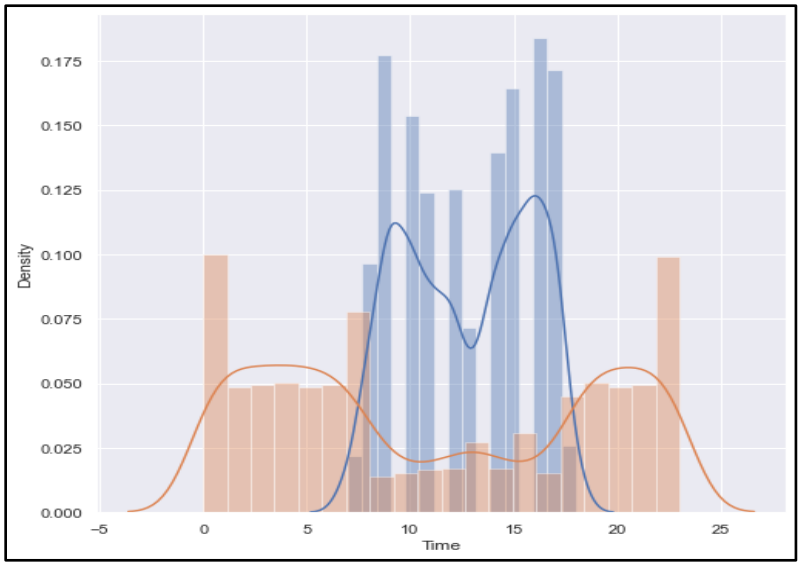


Fig 6. Density plot

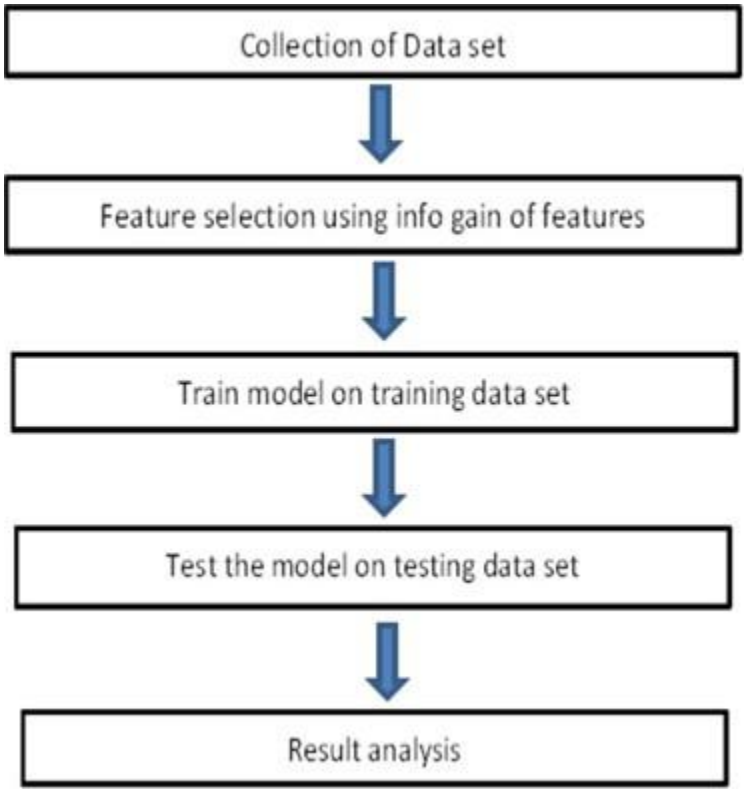
Between 07:00 and 18:00 there are occupants in the environment or not. But the time come to non-working hours, then we can absolutely say that there is no occupant.

Chapter 4:-

System Design

4.1 Flowchart of the System:

The flowchart of the algorithm



The above flowchart describes the working flow of the project. We first built the UI of the of occupancy prediction in python. Then research was done to select the best algorithm from the list of algorithms. The algorithm was later implemented in python and deployed.

Chapter 5

Model Building

5.1 Algorithm research and Selection:-

The machine learning classification models have been used for prediction of Occupancy. The brief details of each model is described below.

5.1.1 Logistic Regression:-

Logistic regression is named for the function used at the core of the method, the logistic function.

The logistic function, also called the sigmoid function was developed by statisticians to describe properties of population growth in ecology, rising quickly and maxing out at the carrying capacity of the environment. It's an S-shaped curve that can take any real-valued number and map it into a value between 0 and 1, but never exactly at those limits.

$$1 / (1 + e^{-value})$$

Where e is the base of the natural logarithms (Euler's number or the EXP() function in your spreadsheet) and value is the actual numerical value that you want to transform. Below is a plot of the numbers between -5 and 5 transformed into the range 0 and 1 using the logistic function.

Logistic regression uses an equation as the representation, very much like linear regression. Input values (x) are combined linearly using weights or coefficient values (referred to as the Greek capital letter Beta) to predict an output value (y). A key difference from linear regression is that the output value being modelled is a binary values (0 or 1) rather than a numeric value.

Below is an example logistic regression equation:

$$y = e^{(b0 + b1*x)} / (1 + e^{(b0 + b1*x)})$$

Where y is the predicted output, b0 is the bias or intercept term and b1 is the coefficient for the single input value (x). Each column in your input data has an associated b coefficient (a constant real value) that must be learned from your training data.

The actual representation of the model that you would store in memory or in a file are the coefficients in the equation (the beta value or b's).

Pros:-

1. Logistic regression is easier to implement, interpret, and very efficient to train.
2. It makes no assumptions about distributions of classes in feature space.

Cons:-

1. If the number of observations is lesser than the number of features, Logistic Regression should not be used, otherwise, it may lead to overfitting.
2. It constructs linear boundaries.

```
##Logistic Regression
from sklearn.metrics import accuracy_score
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve
logit_model = LogisticRegression()
# fit the model on the training set
logit_model.fit(X_train, y_train)
# predict the test set
yhat = logit_model.predict(X_test)
yhat1=logit_model.predict(X_train)
# evaluate model skill
score = accuracy_score(y_test, yhat)
score1=accuracy_score(y_train,yhat1)
print(score)
print(score1)
```

```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import RepeatedStratifiedKFold
from sklearn.model_selection import GridSearchCV
# define models and parameters
solvers = ['newton-cg', 'lbfgs', 'liblinear']
penalty = ['l2']
c_values = [100, 10, 1.0, 0.1, 0.01]
grid = dict(solver=solvers,penalty=penalty,C=c_values)
model_params={
    'logistic':{
        'model':LogisticRegression(),
        'params':grid
        }

    }

scores = []
for model_name, mp in model_params.items():
    clf =  GridSearchCV(mp['model'], mp['params'], cv=10, return_train_score=False)
    clf.fit(X_test,y_test)
    scores.append({
        'model': model_name,
        'best_score': clf.best_score_,
        'best_params': clf.best_params_
    })

logistic_df = pd.DataFrame(scores,columns=['model','best_score','best_params'])
logistic_df
```

Output:- Logistic regression score :-0.989 , prams:-{'C': 0.01, 'penalty': 'l2', 'solver': 'lbfgs'}

5.1.2 Ridge Classifier:-

The Ridge Classifier, based on Ridge regression method, converts the label data into [-1, 1] and solves the problem with regression method. The highest value in prediction is accepted as a target class and for multiclass data multi-output regression is applied.

Advantages:-

Avoids overfitting a model.

They does not require unbiased estimators.

They add just enough bias to make the estimates reasonably reliable approximations to true population values.

They still perform well in cases of a large multivariate data with the number of predictors (p) larger than the number of observations (n).

The ridge estimator is preferably good at improving the least-squares estimate when there is multicollinearity.

Disadvantages

They include all the predictors in the final model.

They are unable to perform feature selection.

They shrink the coefficients towards zero.

They trade the variance for bias.

```
model = RidgeClassifier()
# fit the model on the training set
model.fit(X_train, y_train)
# predict the test set
yhat = model.predict(X_test)
yhat1=model.predict(X_train)
# evaluate model skill
score = accuracy_score(y_test, yhat)
```

```
from sklearn.linear_model import RidgeClassifier
####HyperParameter tuning for Ridge Classifier

alpha = [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0]
Ridge_grid = dict(alpha=alpha)
Ridge_params={
    'Ridge':{
        'model':RidgeClassifier(),
        'Ridge_params':Ridge_grid
        }

    }

Ridge_scores = []
for model_name, mp in Ridge_params.items():
    clf =  GridSearchCV(mp['model'], mp['Ridge_params'], cv=10, return_train_score=False)
    clf.fit(X_test,y_test)
    Ridge_scores.append({
        'model': model_name,
        'best_score': clf.best_score_,
        'best_params': clf.best_params_
    })
Ridge_df = pd.DataFrame(Ridge_scores,columns=['model','best_score','best_params'])
Ridge_df
```

Ridge Classifier: - scores:-0.987, params :{'alpha': 0.1}

5.1.3 KNN algorithm:-

K-Nearest Neighbours In pattern recognition, the k-nearest neighbour's algorithm (k-NN) is a non-parametric method used for classification and regression. In both cases, the input

consists of the k closest training examples in the feature space. The output depends on whether k-NN is used for classification or regression. In k-NN classification, the output is a class membership. An object is classified by a majority vote of its neighbours, with the object being assigned to the class most common among its k nearest neighbours (k is a positive integer, typically small). If k = 1, then the object is simply assigned to the class of that single nearest neighbour.

```
from sklearn.metrics import confusion_matrix
knn_model = KNeighborsClassifier(n_neighbors=9,metric='manhattan',weights='distance')
knn_model.fit(X_train, y_train)
y_pred = knn_model.predict(X_test)
y_pred1=knn_model.predict(X_train)
print(accuracy_score(y_test, y_pred))
print(accuracy_score(y_train, y_pred1))
```

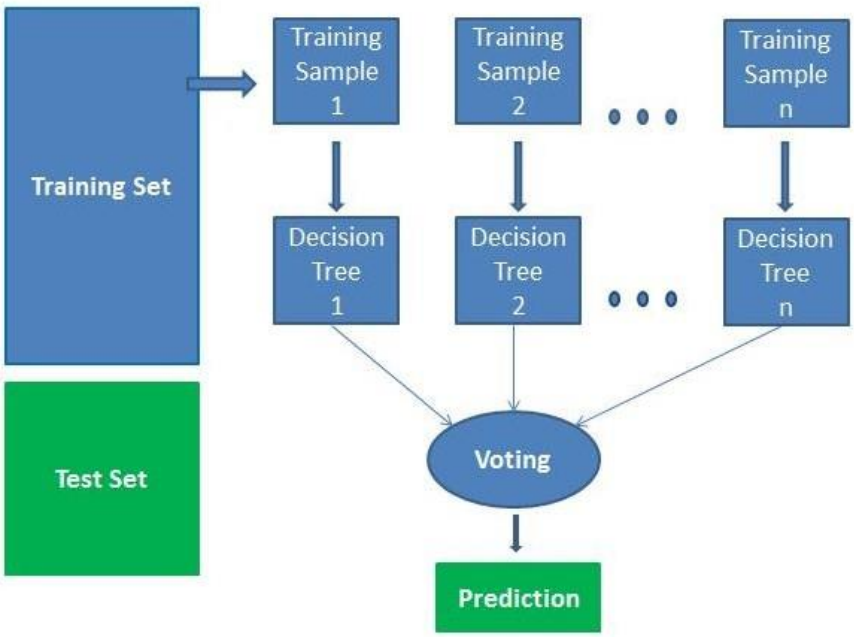
```
from sklearn.metrics import f1_score
from sklearn.neighbors import KNeighborsClassifier
## KNN with Manhattan and Euclidean
# parameter-tuning for knn
n_neighbors =range(1,200,2)
weights = ['uniform', 'distance']
metric= ['euclidean', 'manhattan']
Knn_grid = dict(n_neighbors=n_neighbors,weights=weights,metric=metric)

KNN_params={
    'K_Neighbours':{
        'model':KNeighborsClassifier(),
        'params':Knn_grid
        }

    }

Knn_scores = []

for model_name, mp in KNN_params.items():
    clf =  GridSearchCV(mp['model'], mp['params'], cv=10, return_train_score=False)
    clf.fit(X_test,y_test)
    Knn_scores.append({
        'model': model_name,
        'best_score': clf.best_score_,
        'best_params': clf.best_params_
    })
    Knn_df = pd.DataFrame(Knn_scores,columns=['model','best_score','best_params'])
Knn_df
```

Kneighbours: - scores:-0.99, params :-{'metric': 'manhattan', 'n_neighbors': 9, 'weights': 'distance'}

5.1.4 Random Forest:-

Random forests is a supervised learning algorithm. It can be used both for classification and regression. It is also the most flexible and easy to use algorithm. A forest is comprised of trees. It is said that the more trees it has, the more robust a forest is. Random forests creates decision trees on randomly selected data samples, gets prediction from each tree and selects the best solution by means of voting. It also provides a pretty good indicator of the feature importance.

Random forests has a variety of applications, such as recommendation engines, image classification and feature selection. It can be used to classify loyal loan applicants, identify fraudulent activity and predict diseases. It lies at the base of the Boruta algorithm, which selects important features in a dataset.



Advantages:

•Random forests is considered as a highly accurate and robust method because of the number of decision trees participating in the process.

•It does not suffer from the overfitting problem. The main reason is that it takes the average of all the predictions, which cancels out the biases.

•The algorithm can be used in both classification and regression problems.

Disadvantages:

•Random forests is slow in generating predictions because it has multiple decision trees. Whenever it makes a prediction, all the trees in the forest

Have to make a prediction for the same given input and then perform voting on it. This whole process is time-consuming.

•The model is difficult to interpret compared to a decision tree, where you can easily make a decision by following the path in the tree.

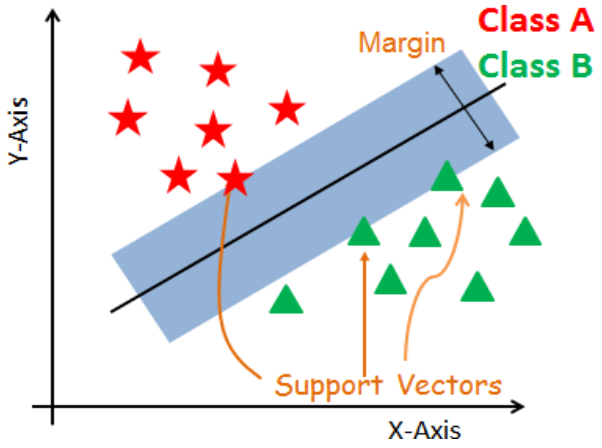
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, accuracy_score
##Hyper Parameter Tuning for Random Forest
n_estimators = [10, 100, 1000]
max_features = ['sqrt', 'log2']
# define grid search
rfgrid = dict(n_estimators=n_estimators,max_features=max_features)

rfparams={
    'RandomForest':{
        'model':RandomForestClassifier(random_state=2020,oob_score=True),
        'params':rfgrid
        }

    }

rfscores = []

for model_name, mp in rfparams.items():
    clf =  GridSearchCV(mp['model'], mp['params'], cv=10, return_train_score=False)
    clf.fit(X_test,y_test)
    rfscores.append({
        'model': model_name,
        'best_score': clf.best_score_,
        'best_params': clf.best_params_
    })

rf_df = pd.DataFrame(rfscores,columns=['model','best_score','best_params'])
rf_df
```

random_forest:- score:-0.992, params:-{'max_features': 'sqrt', 'n_estimators': 100}

5.1.5:- Support Vector Machine

Support Vector Machines is considered to be a classification approach, it but can be employed in both types of classification and regression problems. It can easily handle multiple continuous and categorical variables. SVM constructs a hyperplane in multidimensional space to separate different classes. SVM generates optimal hyperplane in an iterative manner, which is used to minimize an error. The core idea of SVM is to find a maximum marginal hyperplane (MMH) that best divides the dataset into classes.

Support Vectors



Support vectors are the data points, which are closest to the hyperplane. These points will define the separating line better by calculating margins. These points are more relevant to the construction of the classifier.

Hyperplane

A hyperplane is a decision plane which separates between a set of objects having different class memberships.

Margin

A margin is a gap between the two lines on the closest class points. This is calculated as the perpendicular distance from the line to support vectors or closest

Points. If the margin is larger in between the classes, then it is considered a good margin, a smaller margin is a bad margins.

```
from sklearn.svm import SVC
kernel = ['poly', 'rbf', 'sigmoid']
C = [50, 10, 1.0, 0.1, 0.01]
gamma = ['scale']
# define grid search
SVM_grid = dict(kernel=kernel,C=C,gamma=gamma)

SVM_params={
    'SVM':{
        'model':SVC(),
        'params':SVM_grid
        }

    }
SVM_scores = []
for model_name, mp in SVM_params.items():
    clf =  GridSearchCV(mp['model'], mp['params'], cv=10, return_train_score=False)
    clf.fit(X_test,y_test)
    SVM_scores.append({
        'model': model_name,
        'best_score': clf.best_score_,
        'best_params': clf.best_params_
    })

SVM_df = pd.DataFrame(SVM_scores,columns=['model','best_score','best_params'])
SVM_df
```

Svm: - scores:-0.989, params :-{'C': 50, 'gamma': 'scale', 'kernel': 'rbf'}

5.1.6:- Bagged Decision Tree:-

Bagging (Bootstrap Aggregation) is used when our goal is to reduce the variance of a decision tree. Here idea is to create several subsets of data from training sample chosen randomly with replacement. Now, each collection of subset data is used to train their decision trees. As a result, we end up with an ensemble of different models. Average of all the predictions from different trees are used which is more robust than a single decision tree.

```
from sklearn.ensemble import BaggingClassifier
n_estimators = [10, 100, 1000]
# define grid search
bagging_grid = dict(n_estimators=n_estimators)

baggingparams={
    'BaggingClassifier':{
        'model':BaggingClassifier(random_state=2020),
        'params':bagging_grid
        }

    }

bg_scores = []

for model_name, mp in baggingparams.items():
    clf =  GridSearchCV(mp['model'], mp['params'], cv=10, return_train_score=False)
    clf.fit(X_test,y_test)
    bg_scores.append({
        'model': model_name,
        'best_score': clf.best_score_,
        'best_params': clf.best_params_
    })

bg_df = pd.DataFrame(bg_scores,columns=['model','best_score','best_params'])
bg_df
```

Bagging Classifier:-score:-0.991, params:-{'n_estimators': 100}

Chapter 6

Results

1. From All the models which we have fitted Random forest is the best model with best accuracy around 99%.

2. The accuracy of each and every algorithms are high , so overfitting criteria are also check to diagnose the problem but, model perform in same manner for training and testing dataset.

3. The dependent variable is look unbalanced so, up sampled and down sampled techniques are also applied, but it doesn't create a significance difference to the models.

4. The reason behind accuracy may be, data is not so complex and not capturing so much variations.

5. This is also a good signs we can see, if sensors have min errors than the prediction will be much stronger.

Chapter 7

Future Scope

The scope of Machine Learning is not limited to only predictions, but with the help of this project we can upgrade this in our home also with the help of some IOT techniques. Currently in industry motion sensors are used to detect the occupancy but if these Machine learning model get embedded in the sensors, then it may be best option for motion sensor techniques, as it uses environmental factors for detecting the presence, so any bad things like robbery and all are also be detected if the room is closed with no occupants.

CHAPTER 8

Conclusion

From a proper analysis of positive points and constraints on the component, it can be safely concluded that the product is a highly efficient component. This application is working properly and meeting to all sensors requirements for predicting the occupancy. This component can be easily plugged in many other systems.

There have been numbers cases of computer glitches, errors in content and most important weight of features is fixed in automated prediction system, So in the near future the so –called software could be made more secure, reliable and dynamic weight adjustment .In near future this module of prediction can be integrate with the module of automated processing system. The system is trained on old training dataset in future software can be made such that new testing date should also take part in training data after some fix time.

References:-

1. Accurate occupancy detection of an office room from light, temperature, humidity and CO2 measurements using statistical learning models. Luis M. Candanedo, Veronique Feldheim. Energy and Buildings. Volume 112, 15 January 2016, Pages 28-39.

2. https://www.trueoccupancy.com/occupancy-sensor-guide