**Occupancy detection data**

**Engineering Clinic - U18INI5600**

Report by

**Meghavarshine (18BIT010)**

**Mirudula laxmi T (18BIT205)**

**Priyadharshini (18BIT009)**

**Kumaraguru College of Technology**

# Abstract

Experimental data used for binary classification (room occupancy) from Temperature,Humidity,Light and CO2. Ground-truth occupancy was obtained from time stamped pictures that were taken every minute.

***Keywords****: Linear discriminant analysis, Classification and Regression Trees, Random forests, energy conservation in buildings, occupancy detection, GBM models. Caret, ggplot2.*

# 

# 

# Introduction

An occupancy detection is an indoor motion detecting device used to detect the presence of a person to automatically control lights or temperature or ventilation systems.The evolution of IoT devices and wireless sensors today has brought a great impact in data collection, thus producing vast amounts of raw data sets that are rich in information and sometimes contain far too many attributes or features for learning models to handle, therefore creates high dimensionality data.

Making the right or best selection about which features to choose to contribute the most to the prediction variable or target output is important in building an effective and efficient predictive model. The importance of performing feature selection before modelling any data set can contribute towards improving modelling accuracy, less opportunity to make decisions based on noisy data and faster computational time to train models .

# Introduction to the problem statement

This project is trying to solve whether it is possible to design a sensor system that keeps track of occupancy in an area and use this information as input to context-aware systems and make raw data sets that are rich in information and contain attributes. This problem is interesting because the occupancy information collected, can in turn be used for a variety of smart services. These services can aid in environmental control, energy management, increasing work efficiency, and security applications.

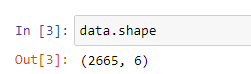
# Dataset Details and Exploration

Dataset was downloaded from<https://www.kaggle.com/datasets>

Dataset : Room Occupancy

Dataset type : Classification

Dataset link :<https://www.kaggle.com/sachinsharma1123/room-occupancy>



This Dataset contains 2665 rows and 6 columns with features ,

* Temperature
* Humidity
* Light
* CO2
* Humidity ratio
* And the target variable Occupancy.

Target Variable:

* 1-if there are chances of room occupancy.
* 0-No chances of room occupancy

## Algorithms chosen

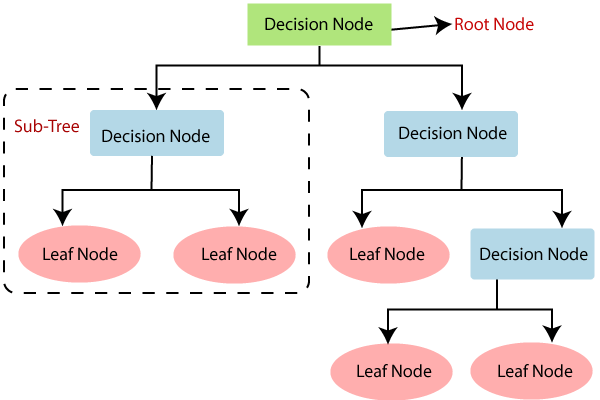
The best feature selection method mainly on selecting subsets of features and classification algorithms that are robust to predict room occupancy based on a benchmark data set. The objective is to find the best features by comparing the performance of various feature selection techniques namely information gain, correlation and wrapper subset algorithms with supervised classifiers such as neural network, decision tree and K-nearest neighbours. In comparing classification model performance, it is reported that the highest accuracies, ranging from 95% to 99% of the occupancy room prediction have been obtained from training classification models.

**Machine Learning Algorithms**

* **Decision tree**

Decision Tree is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome.

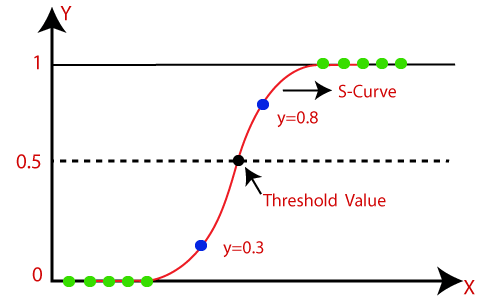
In a Decision tree, there are two nodes, which are the Decision Node and Leaf Node. Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches. The decisions or the test are performed on the basis of features of the given dataset. It is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions. It is called a decision tree because, similar to a tree, it starts with the root node, which expands on further branches and constructs a tree-like structure.



* **Logistic regression**

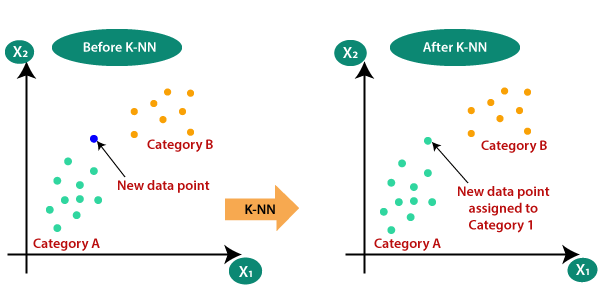
Logistic regression is one of the most popular Machine Learning algorithms, which comes under the Supervised Learning technique. It is used for predicting the categorical dependent variable using a given set of independent variables. Logistic regression predicts the output of a categorical dependent variable.

Therefore, the outcome must be a categorical or discrete value. It can be either Yes or No, 0 or 1, true or False, etc. but instead of giving the exact value as 0 and 1, it gives the probabilistic values which lie between 0 and 1.



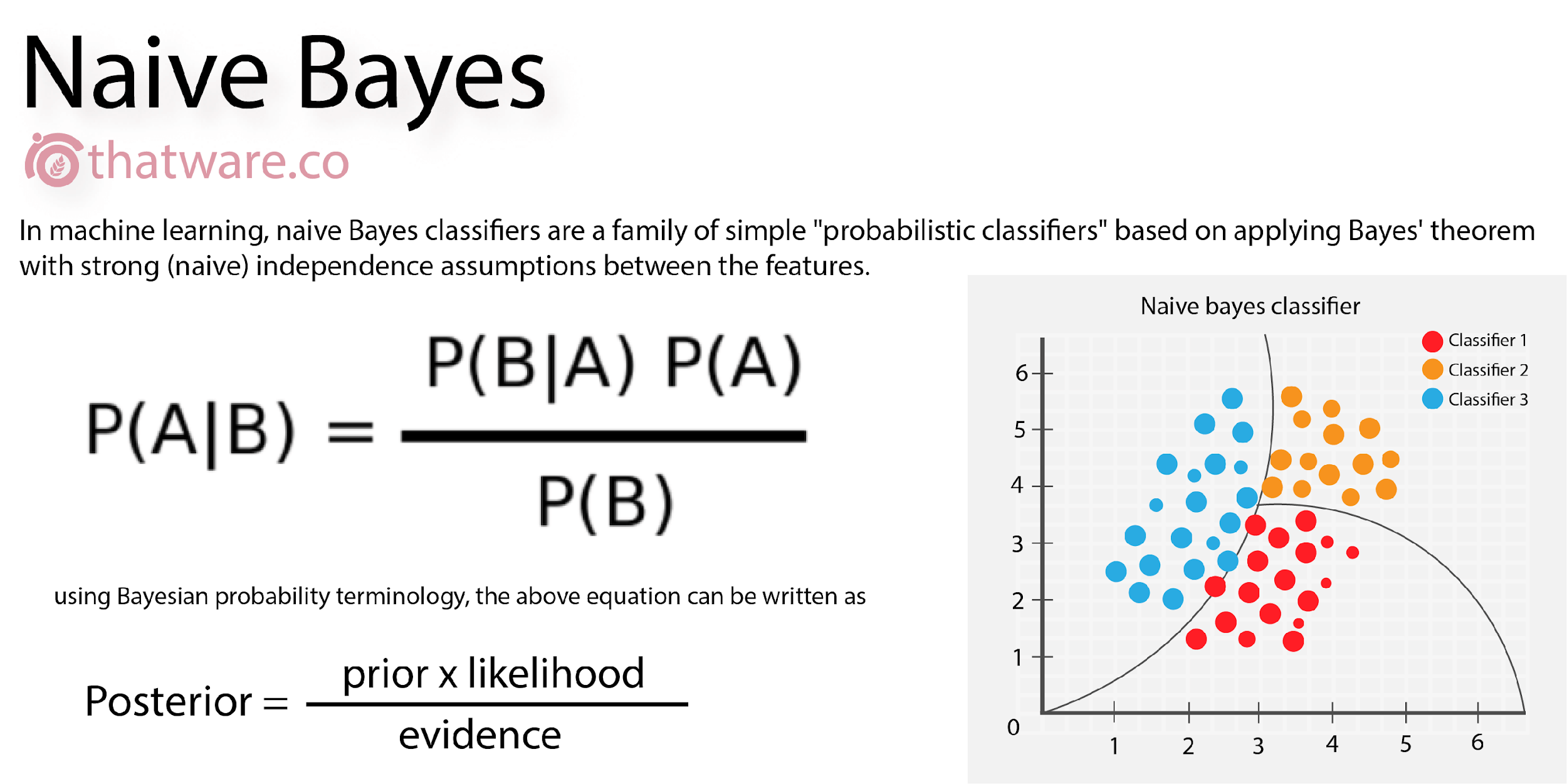
* **K nearest neighbor classifier**

K-Nearest Neighbor is one of the simplest Machine Learning algorithms based on Supervised Learning technique. K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories. K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm. K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems.



* **Naïve bayes**

Naïve Bayes algorithm is a supervised learning algorithm, which is based on Bayes theorem and used for solving classification problems. It is mainly used in text classification that includes a high-dimensional training dataset. Naïve Bayes Classifier is one of the simple and most effective Classification algorithms which helps in building the fast machine learning models that can make quick predictions. It is a probabilistic classifier, which means it predicts on the basis of the probability of an object.



**Feature Selection Techniques**

Feature selection is the process of reducing the number of input variables when developing a predictive model to improve the performance of the model.

* **Univariate technique**

Univariate analysis is the technique of comparing and analyzing the dependency of a single predictor and a response variable. ... Univariate Analysis is thought to be one of the simplest forms of data analysis as it doesn't deal with causes or relationships, like a regression would.

* **Recursive feature elimination technique**

RFE is popular because it is easy to configure and use and because it is effective at selecting those features (columns) in a training dataset that are more or most relevant in predicting the target variable.

* **Principal component analysis**

Principal Component Analysis (PCA) is an unsupervised, non-parametric statistical technique primarily used for dimensionality reduction in machine learning. PCA can also be used to filter noisy datasets, such as image compression. The first principal component expresses the most amount of variance.

## Data preprocessing

Data preprocessing includes data cleaning, data normalization, data transformation, and data reduction as part of the knowledge discovery process. When creating a machine learning project, it is not always a case that we come across clean and formatted data. And while doing any operation with data, it is mandatory to clean it and put it in a formatted way. So, for this, we use data preprocessing tasks.

* **Rescaling technique**

Your preprocessed data may contain attributes with a mixture of scales for various quantities such as dollars, kilograms and sales volume. Many machine learning methods expect or are more effective if the data attributes have the same scale.

* **Standardization technique**

Standardization refers to shifting the distribution of each attribute to have a mean of zero and a standard deviation of one (unit variance). It is useful to standardize attributes for a model that relies on the distribution of attributes such as Gaussian processes.

## Implementation & Results

Step 1: import all the required libraries.

Step 2: extract the chosen csv file.

Step 3: separate input and output components.

Step 4: apply rescaling preprocessing technique.

**Load the data using numpy and pandas packages**

import numpy as np

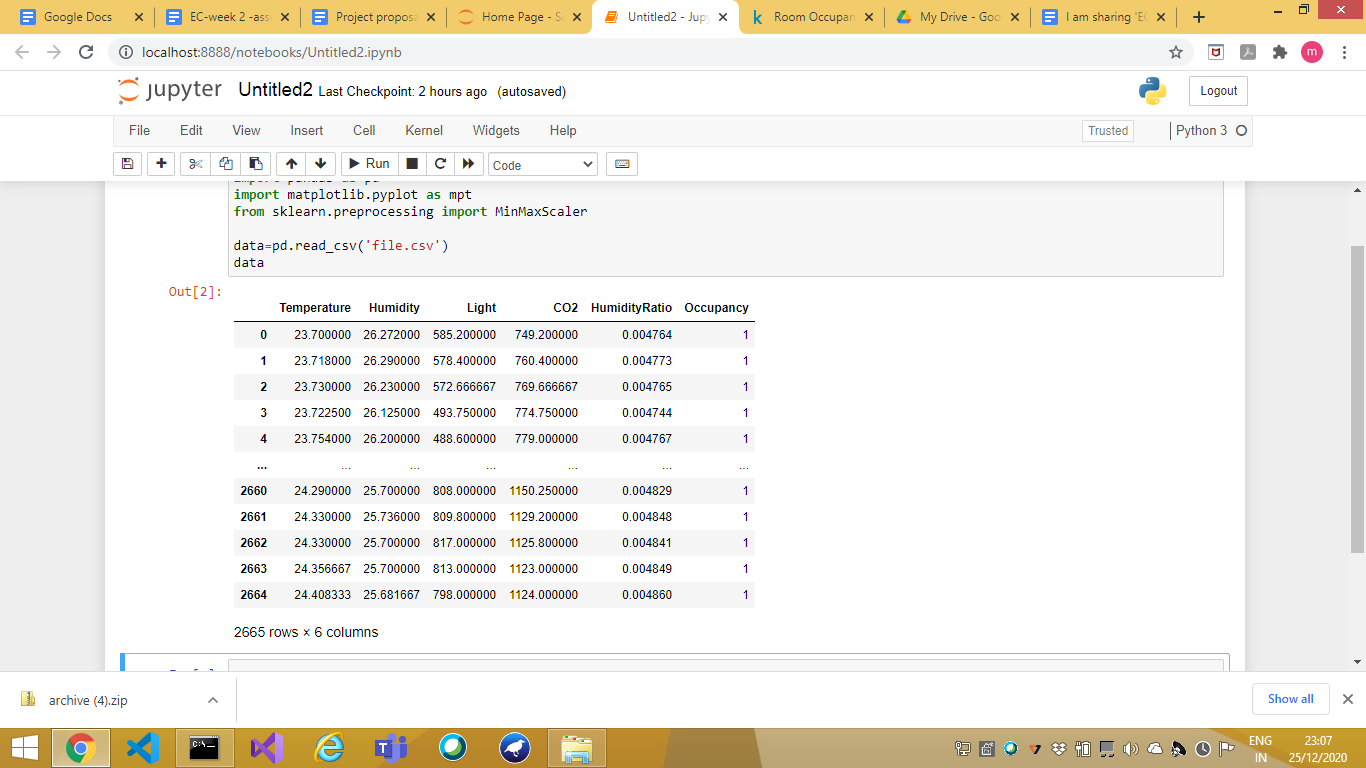
import pandas as pd

import matplotlib.pyplot as mpt

data=pd.read\_csv('file.csv')

data

**Output**

****

**Data set details**

Experimental data used for binary classification (room occupancy) from Temperature,Humidity,Light and CO2.

This dataset contains 6 attributes in which 5 are feature and 1 is target variable

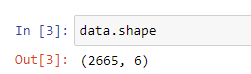
* Temperature
* Humidity
* Light
* Carbon dioxide(CO2)
* Humidity ratio
* Target Variable-Occupancy
* 1- if there is a chance of room occupancy.
* 0- No chances of room occupancy

**Dataset type:** Classification dataset

**Missing values:** none

|  |
| --- |
| **With this data we can predict whether the room is occupied or not** |

**Dataset Shape**

****

1. Apply preprocessing techniques
2. Compare the results of different algorithms by repeatedly doing the following:

* Apply a particular automatic feature selection technique
* Use a specific train and test split
* Apply classification / regression algorithm and display the confusion matrix for the same

|  |
| --- |
| **Algorithms - Decision tree, Logistic regression, K nearest neighbors, Naïve Bayes** |

* **PreProcessing technique** - **Rescaling**

|  |
| --- |
| **from numpy import set\_printoptions from sklearn.preprocessing import MinMaxScaler array=data.values X=array[:,0:5] Y=array[:,4] scaler = MinMaxScaler(feature\_range=(0,1)) rescaledX = scaler.fit\_transform(X) set\_printoptions(precision=3) print(rescaledX[0:5,:])** |

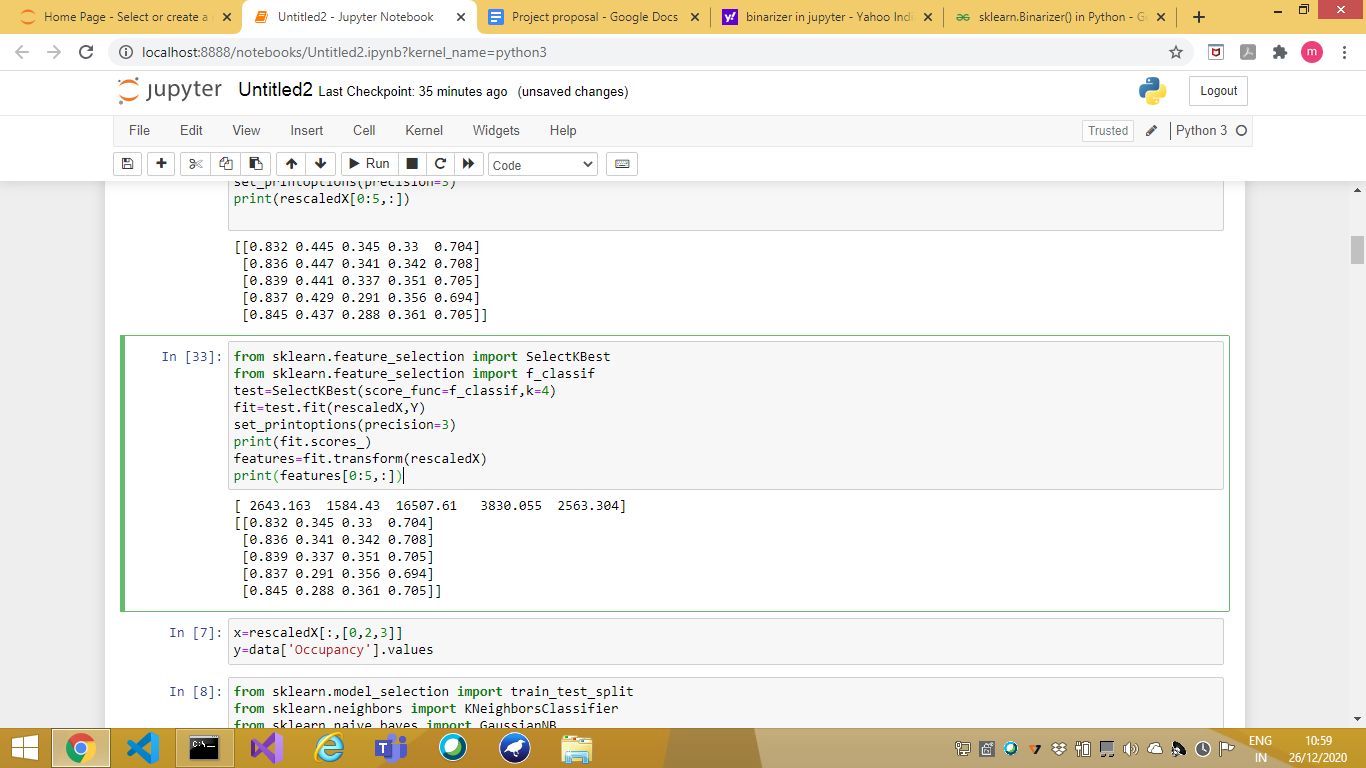
**Output**

|  |
| --- |
| [[0.832 0.445 0.345 0.33 0.704]  [0.836 0.447 0.341 0.342 0.708]  [0.839 0.441 0.337 0.351 0.705]  [0.837 0.429 0.291 0.356 0.694]  [0.845 0.437 0.288 0.361 0.705]] |

* **Feature selection technique** - **univariate selection**

|  |
| --- |
| **from sklearn.feature\_selection import SelectKBest from sklearn.feature\_selection import f\_classif test=SelectKBest(score\_func=f\_classif,k=4) fit=test.fit(rescaledX,Y) set\_printoptions(precision=3) print(fit.scores\_) features=fit.transform(rescaledX) print(features[0:5,:])** |

**Output**

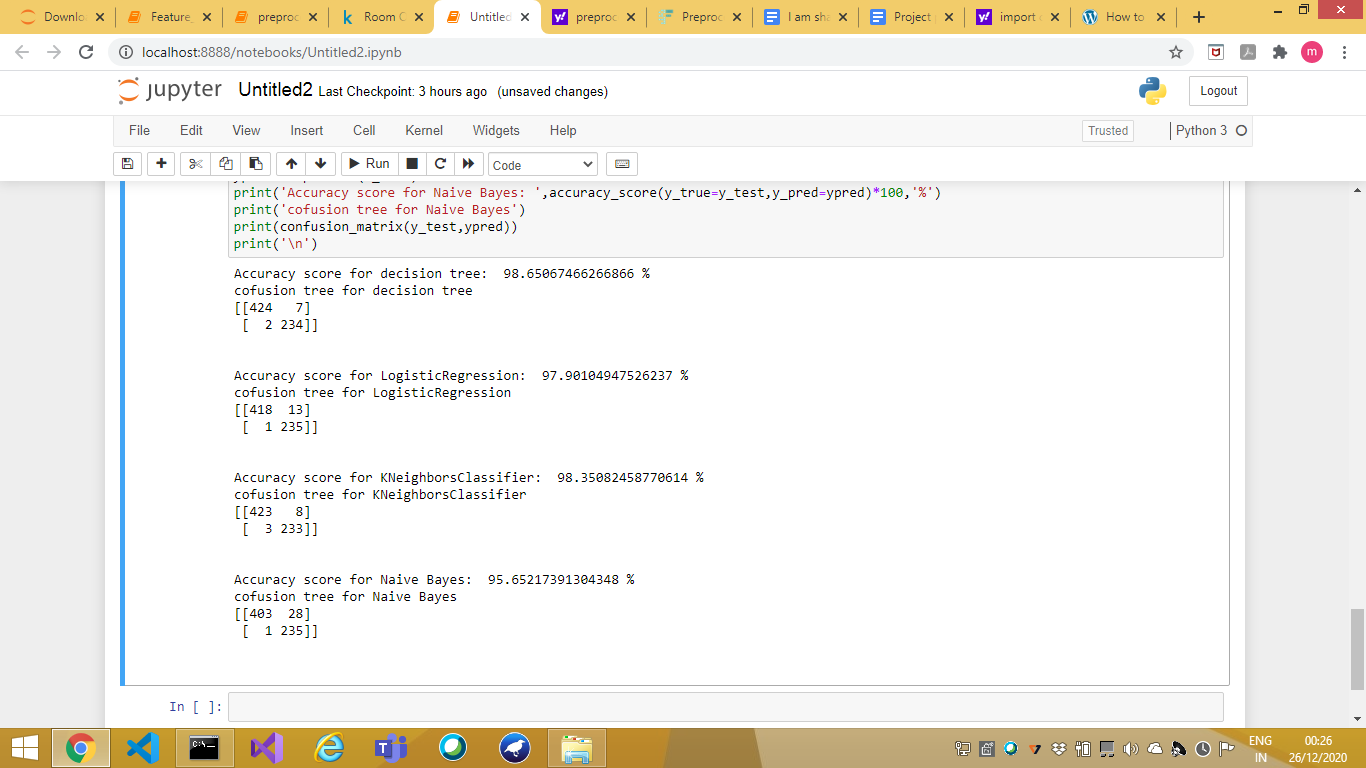
****

|  |
| --- |
| **x=rescaledX[:,[0,2,3]] y=data['Occupancy'].values** |

**Accuracy and confusion matrix various algorithm**

|  |
| --- |
| **from sklearn.model\_selection import train\_test\_split from sklearn.neighbors import KNeighborsClassifier from sklearn.naive\_bayes import GaussianNB from sklearn.tree import DecisionTreeClassifier, plot\_tree from sklearn.linear\_model import LogisticRegression from sklearn.metrics import accuracy\_score from sklearn.metrics import confusion\_matrix  x\_train, x\_test, y\_train, y\_test = train\_test\_split(X, y, random\_state=0,test\_size=0.25) clf=DecisionTreeClassifier(criterion='entropy') clf.fit(x\_train,y\_train) ypred=clf.predict(x\_test) print('Accuracy score for decision tree: ',accuracy\_score(y\_true=y\_test,y\_pred=ypred)\*100,'%') print('cofusion tree for decision tree') print(confusion\_matrix(y\_test,ypred)) print('\n') lr=LogisticRegression() lr.fit(x\_train,y\_train) ypred=lr.predict(x\_test) print('Accuracy score for LogisticRegression: ',accuracy\_score(y\_true=y\_test,y\_pred=ypred)\*100,'%') print('cofusion tree for LogisticRegression') print(confusion\_matrix(y\_test,ypred)) print('\n') kn=KNeighborsClassifier(n\_neighbors=5) kn.fit(x\_train,y\_train) ypred=kn.predict(x\_test) print('Accuracy score for KNeighborsClassifier: ',accuracy\_score(y\_true=y\_test,y\_pred=ypred)\*100,'%') print('cofusion tree for KNeighborsClassifier') print(confusion\_matrix(y\_test,ypred)) print('\n') nb=GaussianNB() nb.fit(x\_train,y\_train) ypred=nb.predict(x\_test) print('Accuracy score for Naive Bayes: ',accuracy\_score(y\_true=y\_test,y\_pred=ypred)\*100,'%') print('cofusion tree for Naive Bayes') print(confusion\_matrix(y\_test,ypred)) print('\n')** |

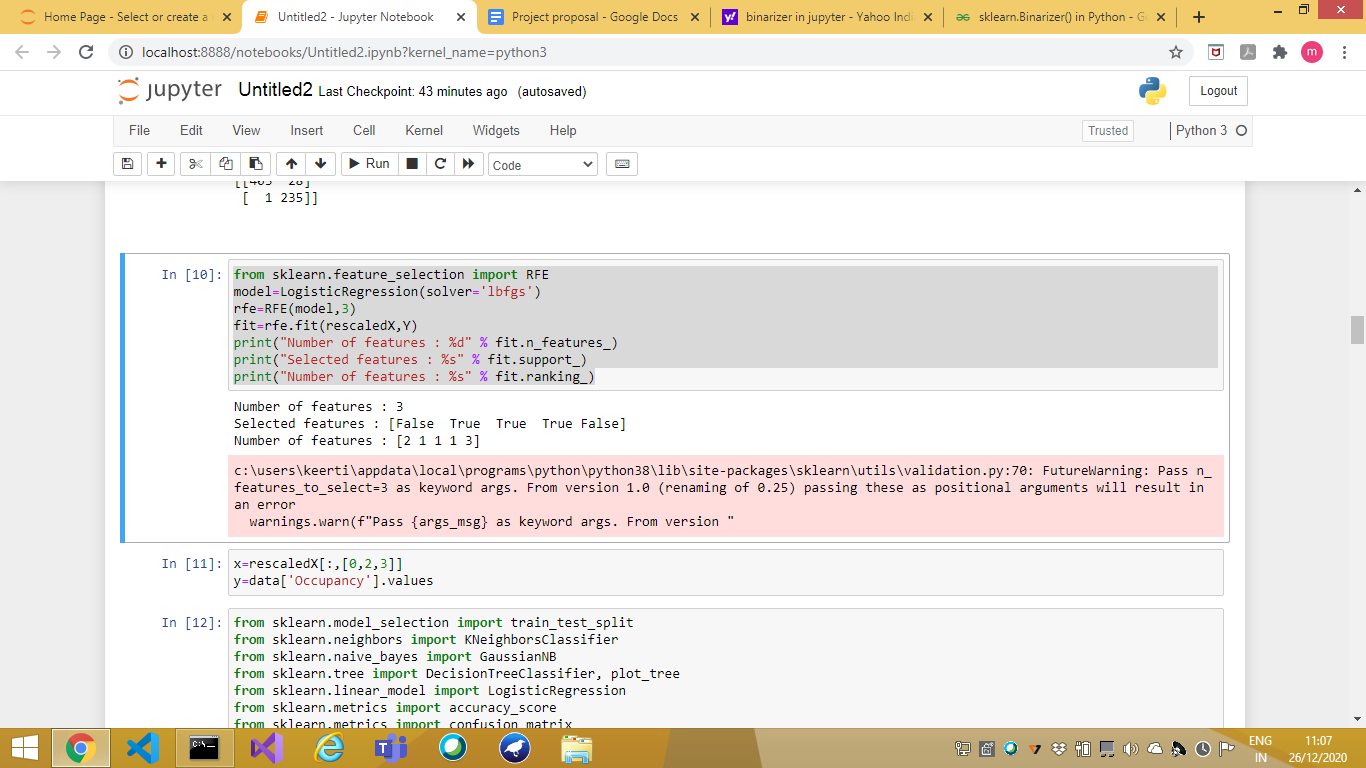
**Output**

****

* **Feature selection technique** - **Recursive feature elimination**

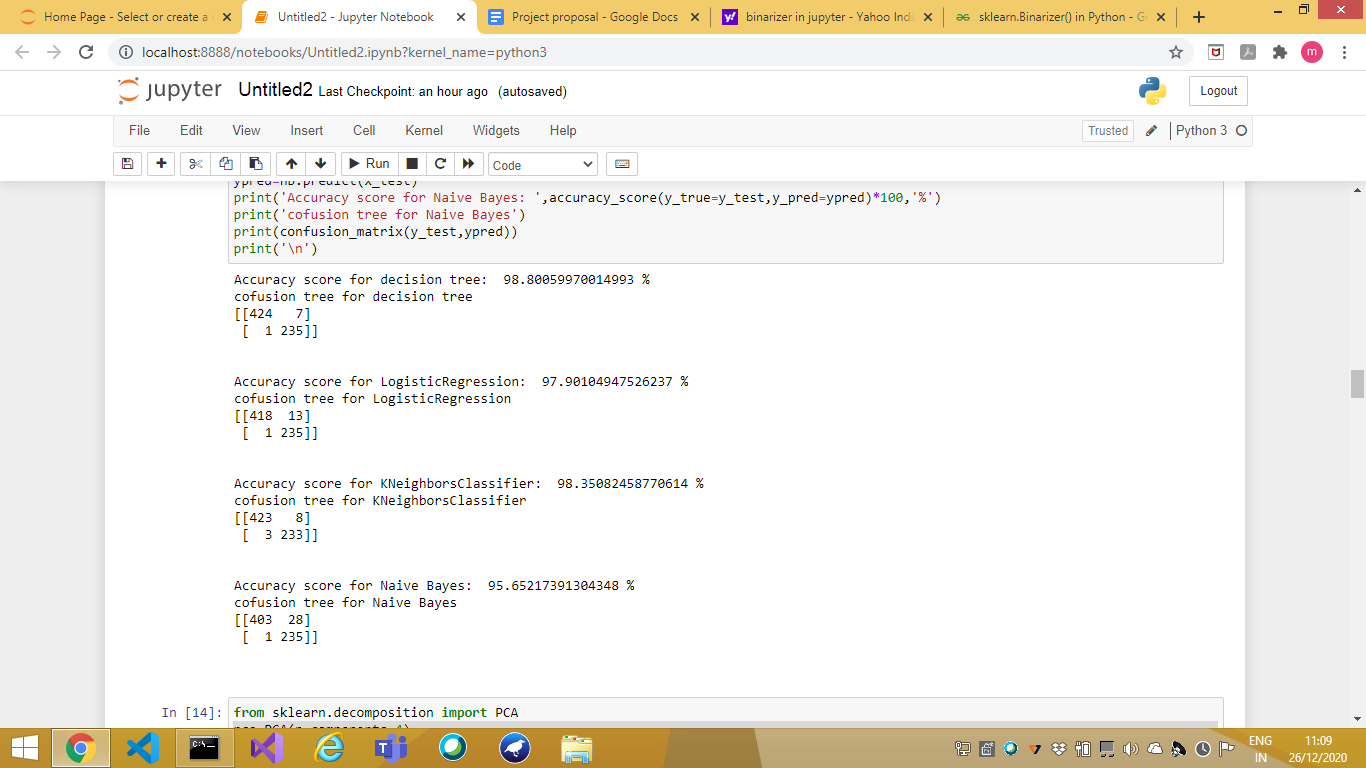
|  |
| --- |
| **from sklearn.feature\_selection import RFE model=LogisticRegression(solver='lbfgs') rfe=RFE(model,3) fit=rfe.fit(rescaledX,Y) print("Number of features : %d" % fit.n\_features\_) print("Selected features : %s" % fit.support\_) print("Number of features : %s" % fit.ranking\_)** |

**Output**

****

|  |
| --- |
| **x=rescaledX[:,[0,2,3]] y=data['Occupancy'].values** |

**Accuracy and confusion matrix various algorithm**

****

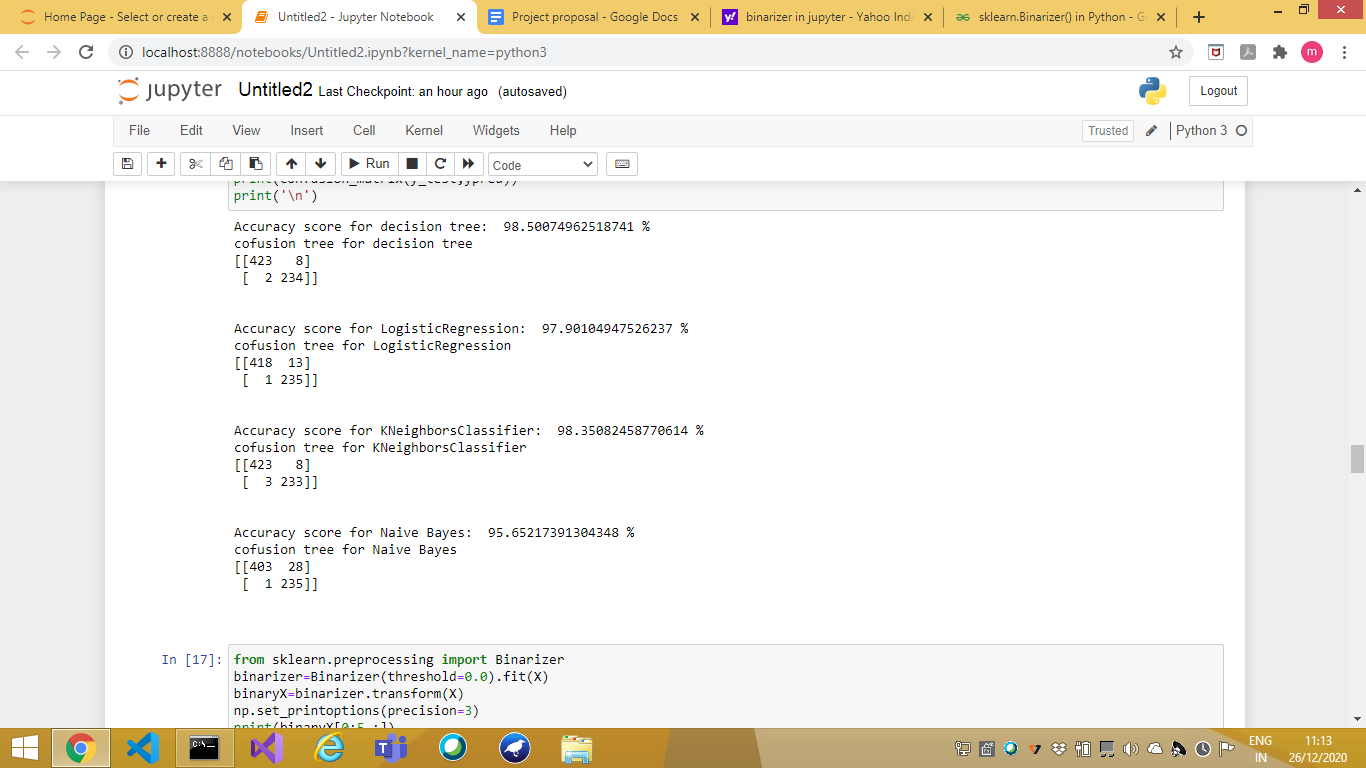
* **Feature selection Technique** - **Principal component analysis**

|  |
| --- |
| **from sklearn.decomposition import PCA pca=PCA(n\_components=4) fit=pca.fit(rescaledX) print("Explained Variance : %s" % fit.explained\_variance\_ratio\_) print(fit.components\_)** |

# 

|  |
| --- |
| **x=rescaledX[:,[0,1,2]] f=data['Occupancy'].values** |

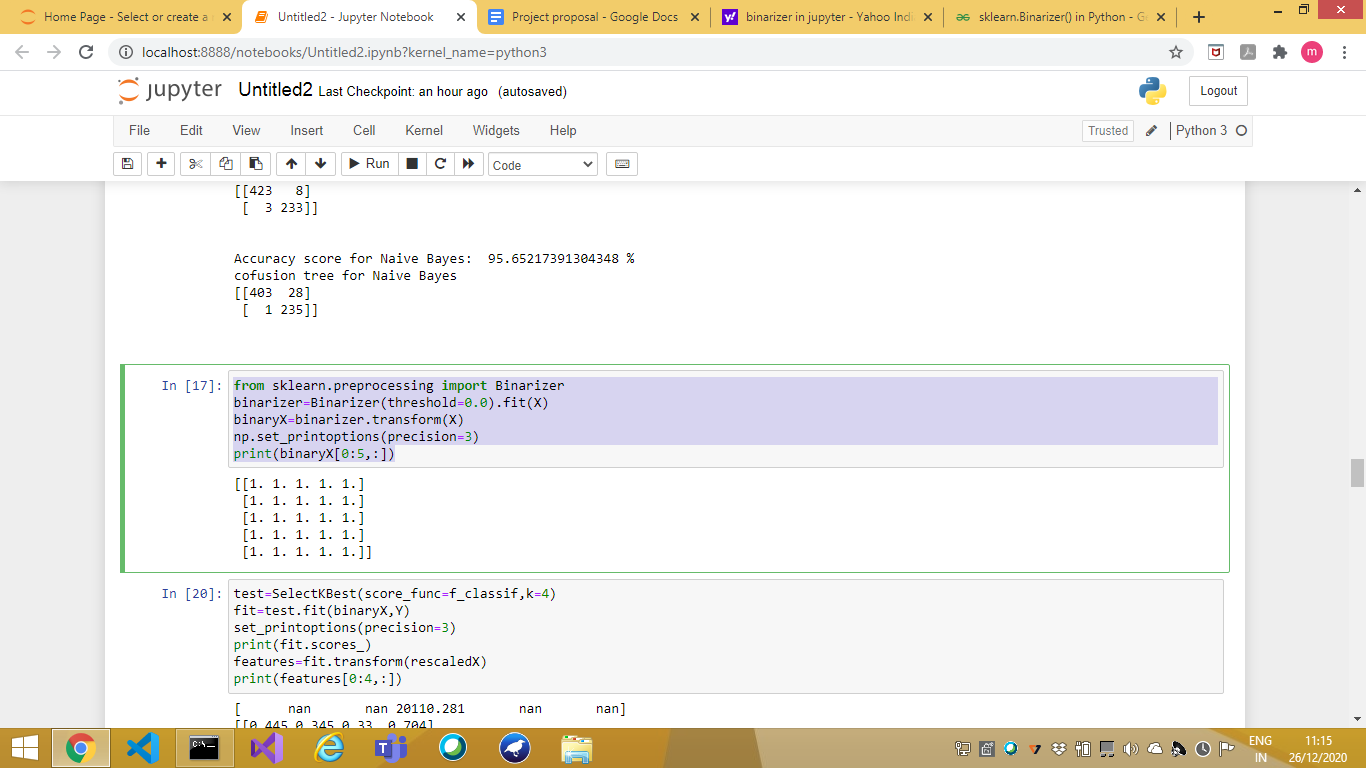
**Accuracy and confusion matrix various algorithm**



* **Preprocessing Technique - Binarization**

|  |
| --- |
| **from sklearn.preprocessing import Binarizer binarizer=Binarizer(threshold=0.0).fit(X) binaryX=binarizer.transform(X) np.set\_printoptions(precision=3) print(binaryX[0:5,:])** |

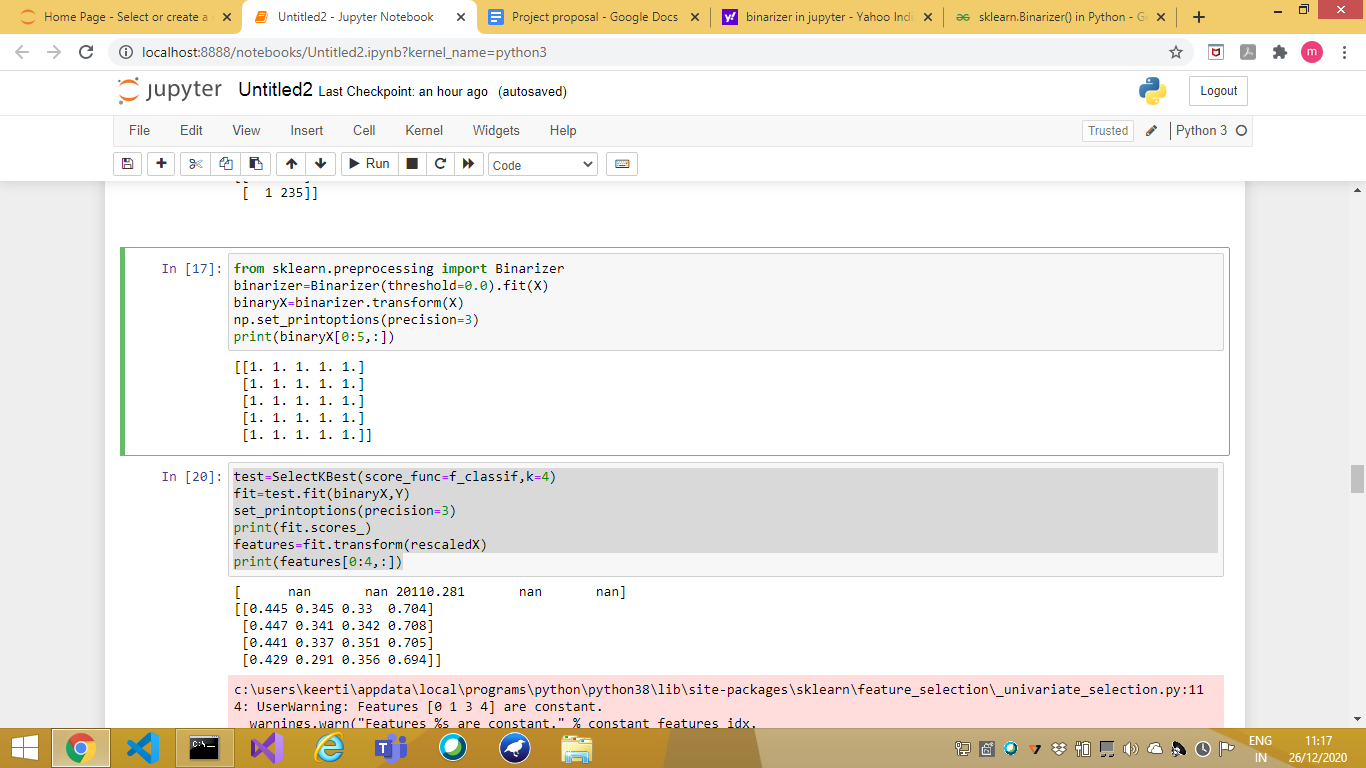
**Output**

****

* **Feature Selection Technique** - **Univariate**

|  |
| --- |
| **test=SelectKBest(score\_func=f\_classif,k=4) fit=test.fit(binaryX,Y) set\_printoptions(precision=3) print(fit.scores\_) features=fit.transform(rescaledX) print(features[0:4,:])** |

**Output**

****

|  |
| --- |
| **x=binaryX[:,[0,3]] y=data['Occupancy'].values** |

**Accuracy and confusion matrix various algorithm**

# 

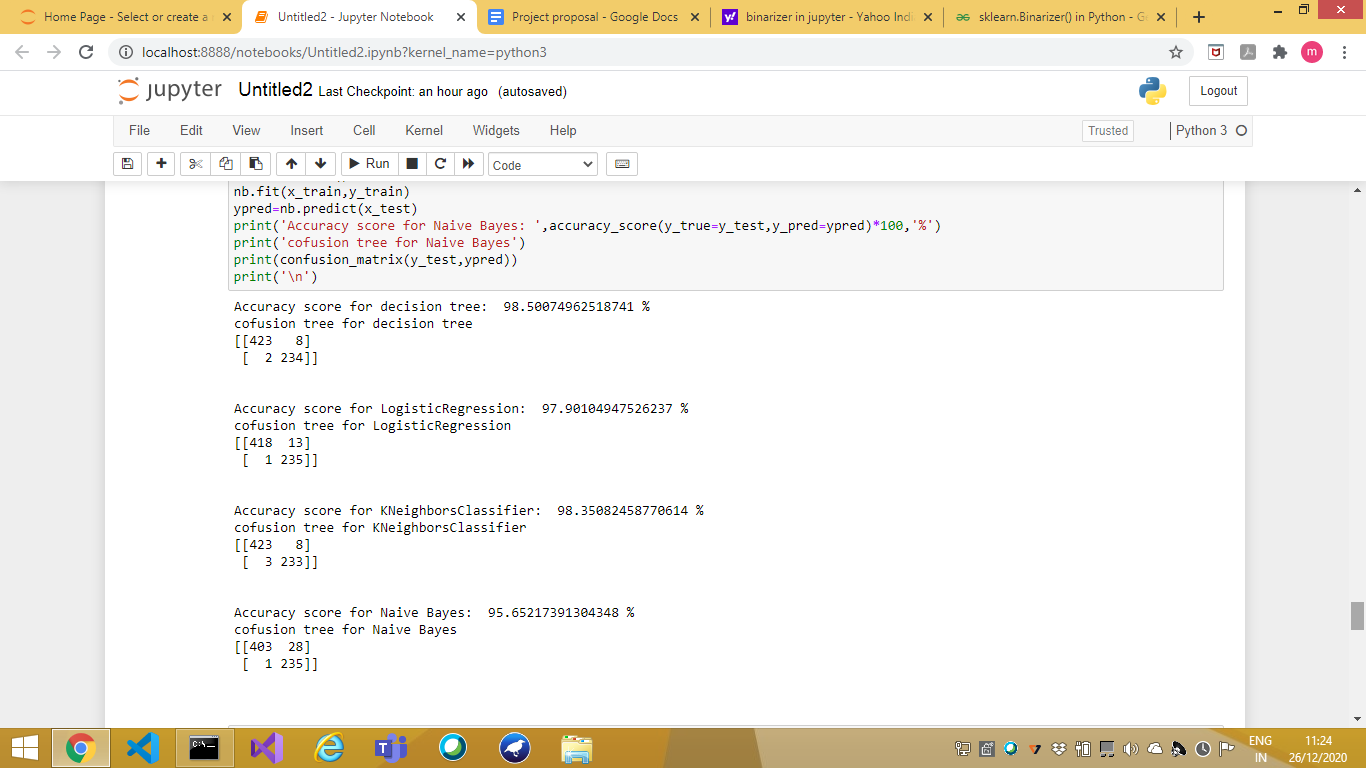
* **Feature selection: Recursive Feature elimination**

|  |
| --- |
| **model=LogisticRegression(solver='lbfgs') rfe=RFE(model,3) fit=rfe.fit(binaryX,Y) print("Number of features : %d" % fit.n\_features\_) print("Selected features : %s" % fit.support\_) print("Number of features : %s" % fit.ranking\_)** |

# 

|  |
| --- |
| **x=binaryX[:,[0,2,3]] y=data['Occupancy'].values** |

**Accuracy and confusion matrix various algorithm**

****

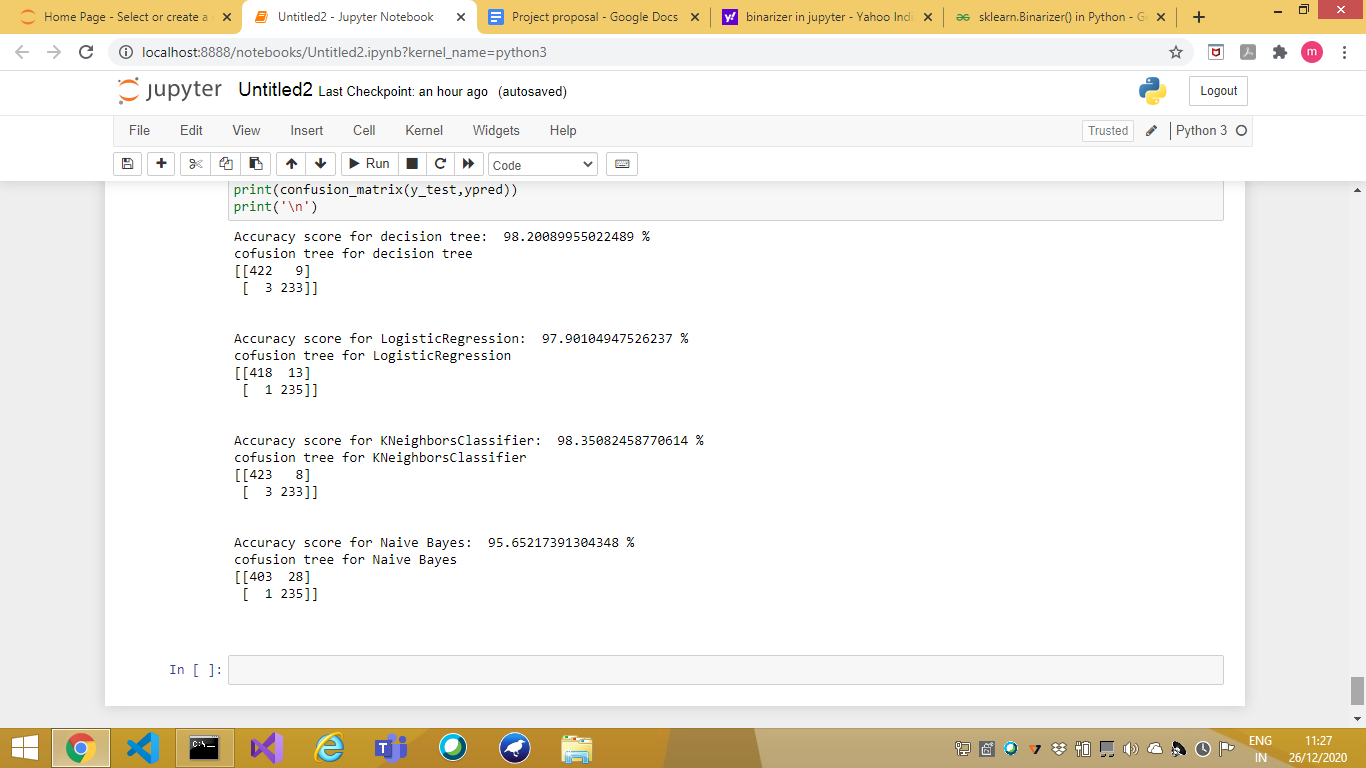
* **Feature selection Technique** : **Principal component analysis**

|  |
| --- |
| **pca=PCA(n\_components=4) fit=pca.fit(binaryX) print("Explained Variance : %s" % fit.explained\_variance\_ratio\_) print(fit.components\_)** |

# 

|  |
| --- |
| **x=binaryX[:,[0,1]] y=data['Occupancy'].values** |

**Accuracy and confusion matrix various algorithm**

****

**Draw graphs for the comparison results and write inferences for the same**

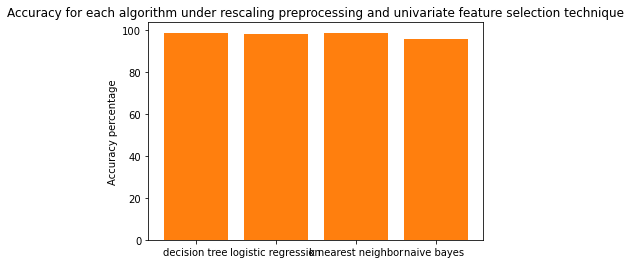
**GRAPHICAL REPRESENTATION**

**Preprocessing Technique - Rescaling**

* Feature selection technique - univariate

|  |
| --- |
| **ty=('decision tree', 'logistic regression','k nearest neighbor', 'naive bayes ') ac=[98.65,97.90,98.35,95.65] y\_pos=np.arange(len(ty)) plt.bar(y\_pos,ac,align='center', alpha=1, color='C11') plt.xticks(y\_pos,ty) plt.ylabel('Accuracy percentage') plt.title('Accuracy for each algorithm under rescaling preprocessing and univariate feature selection technique') plt.show()** |

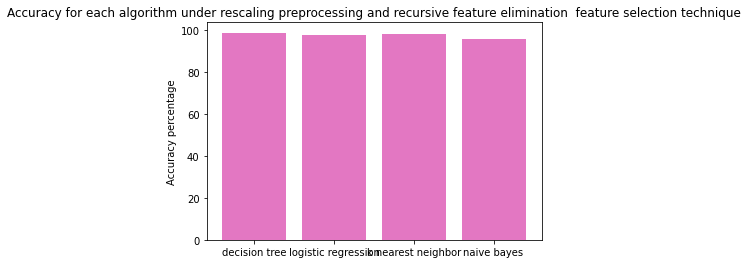
PLOT:

****

* **Feature selection technique - recursive feature elimination**

|  |
| --- |
| **ty=('decision tree', 'logistic regression','k nearest neighbor', 'naive bayes ') ac=[98.80,97.90,98.35,95.65] y\_pos=np.arange(len(ty)) plt.bar(y\_pos,ac,align='center', alpha=1, color='C6') plt.xticks(y\_pos,ty) plt.ylabel('Accuracy percentage') plt.title('Accuracy for each algorithm under rescaling preprocessing and recursive feature elimination feature selection technique') plt.show()** |

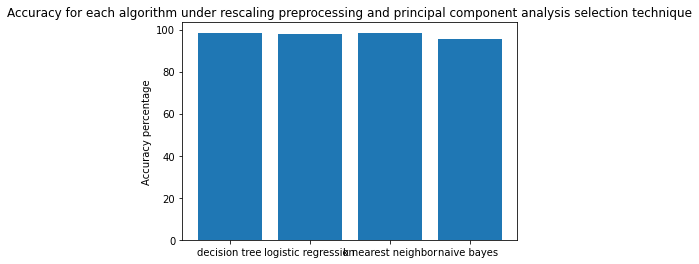
PLOT

****

* **Feature selection technique - univariate principal component analysis**

|  |
| --- |
| **ty=('decision tree', 'logistic regression','k nearest neighbor', 'naive bayes ') ac=[98.50,97.90,98.35,95.65] y\_pos=np.arange(len(ty)) plt.bar(y\_pos,ac,align='center', alpha=1, color='C10') plt.xticks(y\_pos,ty) plt.ylabel('Accuracy percentage') plt.title('Accuracy for each algorithm under rescaling preprocessing and principal component analysis selection technique') plt.show()** |

PLOT

****

**Preprocessing Technique - Binarization**

* **Feature selection technique - univariate**

|  |
| --- |
| **ty=('decision tree', 'logistic regression','k nearest neighbor', 'naive bayes ') ac=[98.65,97.90,98.35,95.65] y\_pos=np.arange(len(ty)) plt.bar(y\_pos,ac,align='center', alpha=1, color='C11') plt.xticks(y\_pos,ty) plt.ylabel('Accuracy percentage') plt.title('Accuracy for each algorithm under binarization preprocessing and univariate feature selection technique') plt.show()** |

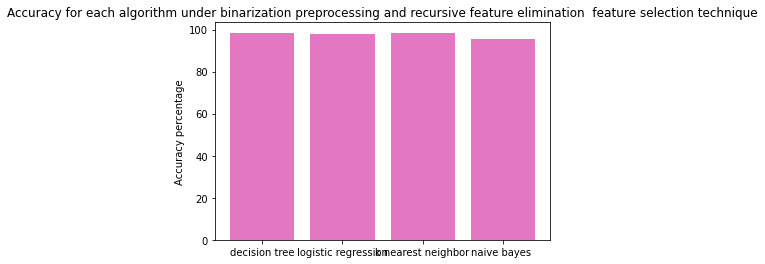
PLOT:

****

* **Feature selection technique - recursive feature elimination**

|  |
| --- |
| **ty=('decision tree', 'logistic regression','k nearest neighbor', 'naive bayes ') ac=[98.50,97.90,98.35,95.65] y\_pos=np.arange(len(ty)) plt.bar(y\_pos,ac,align='center', alpha=1, color='C6') plt.xticks(y\_pos,ty) plt.ylabel('Accuracy percentage') plt.title('Accuracy for each algorithm under binarization preprocessing and recursive feature elimination feature selection technique') plt.show()** |

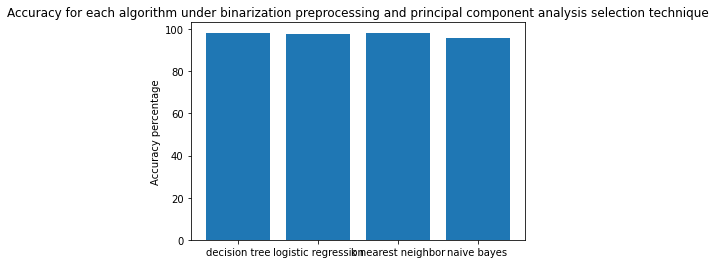
PLOT:

****

* **Feature selection technique - univariate principal component analysis**

|  |
| --- |
| **ty=('decision tree', 'logistic regression','k nearest neighbor', 'naive bayes ') ac=[98.20,97.90,98.35,95.65] y\_pos=np.arange(len(ty)) plt.bar(y\_pos,ac,align='center', alpha=1, color='C10') plt.xticks(y\_pos,ty) plt.ylabel('Accuracy percentage') plt.title('Accuracy for each algorithm under binarization preprocessing and principal component analysis selection technique') plt.show()** |

PLOT:

****

**INFERENCE**

In Accuracy order of algorithm used:

|  |
| --- |
| **Decision tree, Logistic regression, K nearest neighbors, Naïve Bayes** |

|  |  |  |  |
| --- | --- | --- | --- |
| **Technique** | **Feature Selection** | **Training : Test** | **Accuracy** |
| **Rescaling** | **univariate** | **0.75 : 0.25** | |  |  |  |  | | --- | --- | --- | --- | | **98.65** | **97.90** | **98.35** | **95.65** | |
| **Rescaling** | **Recursive feature elimination** | **0.75 : 0.25** | |  |  |  |  | | --- | --- | --- | --- | | **98.80** | **97.90** | **98.35** | **95.65** | |
| **Rescaling** | **Principal component analysis** | **0.75 : 0.25** | |  |  |  |  | | --- | --- | --- | --- | | **98.50** | **97.90** | **98.35** | **95.65** | |
| **Binarization** | **univariate** | **0.75 : 0.25** | |  |  |  |  | | --- | --- | --- | --- | | **98.65** | **97.90** | **98.35** | **95.65** | |
| **Binarization** | **Recursive feature elimination** | **0.75 : 0.25** | |  |  |  |  | | --- | --- | --- | --- | | **98.50** | **97.90** | **98.35** | **95.65** | |
| **Binarization** | **Principal component analysis** | **0.75 : 0.25** | |  |  |  |  | | --- | --- | --- | --- | | **98.20** | **97.90** | **98.35** | **95.65** | |

**Result**

By referring to the above table we can say that Decision tree algorithm works well on all the preprocessing technique and feature selections

**For,**

Preprocessing Technique - Rescaling

Feature selection technique- recursive feature elimination

Accuracy - 98.80%

This is the best working combo for our dataset

## Conclusion / Summary

In our report, we have used various preprocessing, feature selection techniques and machine learning algorithms. This method of data analysis has automated analytical model building. It turns out to be an idea that a system can learn from data, identify patterns and make decisions with minimal human intervention. Machine learning, if used in various fields, can obtain tremendous advantages in the present and future world. It can be concluded that in order to solve the classification problem of predicting room occupancy and data set available, WSE features selection algorithm was found to identify the most appropriate features. These are Date, Humidity, Light and CO2, out of the 8 features available. Using the chosen feature set, classification comparison results demonstrated the effectiveness of IBk classifier compared to MLP and LMT, in which there are statistically significant differences between the performances of the three classification algorithms.