Principal Component Analysis and Classification of Room Occupancy using Machine Learning

Student name: Mahmood Dhabi Student ID: 40185647
GitHub link: https://github.com/MahDabi1/Dhabi_INSE_6220_Project/tree/main

Abstract— Principal Component Analysis (PCA) is an unsupervised machine learning algorithm which is helpful to overcome dilemmas like the curse of dimensionality. It has the ability to transform large datasets with correlated variables into smaller uncorrelated ones while preserving the information they used to hold. In this report, PCA is applied on an experimental dataset to classify between samples with room occupancy binary classification. Three different classification models, i.e., Logistic Regression (LR), K-Nearest Neighbor (KNN), and Quadratic Discriminant Analysis (QDA) are applied on the original dataset and on the transformed one (after applying PCA) to classify the occupancy status of given data of rooms. Afterwards, each model is tuned with the ideal hyperparameters in order to get better performance metrics and the performance of each model is measured using F1 score, confusion matrix and receiver operating characteristic (ROC) curves. By using PyCaret library, LR has shown superior performance over other mentioned algorithms before applying PCA on the data but was put down by QDA after applying PCA. Finally, an experiment is conducted to interpret the models using the explainable AI Shapley value. For this purpose, the Extra Trees (ET) classifier model was used. Overall, all the algorithms successfully determined the two classes of room occupancy status and the numbers proved that.

I. INTRODUCTION

Occupancy detection is a main building block of commercial and residential building automation systems [1]. The increasing focus on designing human living environments that are responsive to the people that inhabit them, raises the demand for energy efficient and cost-effective sensing devices that are capable of counting the number of occupants in a room [2].

In this report, at first Principal component analysis (PCA) is applied on the room occupancy dataset with the aim of dimensionality reduction. Afterwards, three popular classification algorithms, logistic regression (LR), K-nearest neighbor (K-NN) and Quadratic Discriminant analysis (QDA) are applied on the original dataset and PCA transformed dataset. The purpose is to determine whether the room is occupied or not. Finally, with the quest of interpreting classification models explainable AI (artificial intelligence) Shapley values are used. It should be noted that the presented results from the classification algorithms in this report, all of them represent the results obtained after applying PCA. More clearly, in this report the results are used from the transformed dataset. The classification results of the original dataset can be found in the Google Colab notebook. The rest of the report is organized as follows: Section II describes the PCA methodology, Section III gives an overview of the three classification algorithms, Section IV provides the room occupancy dataset description, Section V discusses about PCA results, Section VI provides a extensive analysis of the classification results, Section VII provides a discussion on the explainable AI Shapley values. Finally, in section VIII, the conclusion is drawn.

II. PRINCIPAL COMPONENT ANALYSIS

The majority of real-world datasets are highly dimensional. These kinds of datasets are very expensive to process and store, and they are occasionally impossible to visualize. By condensing a huge set of variables into a smaller one that retains the majority of the information from the original dataset, feature reduction techniques like PCA help to reduce the dimensionality of large datasets. PCA is a mathematical algorithm that reduces the dimensionality of the data while retaining most of the variation in the data set1. It accomplishes this reduction by identifying directions, called principal components, along which the variation in the data is maximal. By using a few components, each sample can be represented by relatively few numbers instead of by values for thousands of variables. Samples can then be plotted, making it possible to visually assess similarities and differences between samples and determine whether samples can be grouped. [3]. It accomplishes this goal by condensing the data into fewer dimensions that serve as feature summaries.

A. PCA algorithm

PCA can be applied to a data matrix X with dimension n × p using the followings steps [4]:

1) Standardization: The main goal of this step is to standardize the initial variable so that they all contribute equally to the analysis. First, compute the mean vector $\bar{x}$ of each column of the data set. The mean vector is a p dimensional vector can be expressed by:

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i. \quad (1)$$

The data is standardized by subtracting the mean of each column from each item in the data matrix. The final centered data matrix (Y) can be expressed as follows:

$$Y = HX, \quad (2)$$

where H represents the centering matrix.

2) Covariance matrix computation: The aim of this step is to determine the relationship among the variables. Sometimes variables are so closely related that they contain redundant information. To identify these correlations, the covariance matrix is computed. The p × p covariance matrix is computed as follows:

$$S = \frac{1}{n-1} Y^T Y. \quad (3)$$

3) Eigen decomposition: Using the eigen decomposition the eigenvalues and eigenvectors of S can be computed. Eigenvectors represent the direction of each principle component (PC) whereas eigenvalues represent the variance captured by each PC. Eigen decomposition can be computed using the following equation:

$$S = A\Lambda A^T, \quad (4)$$

where A means the p × p orthogonal matrix of eigenvectors and Λ is the diagonal matrix of eigenvalues.

4) Principal components: It computes the transformed matrix Z that is the size of n×p. The rows of Z represent the observations and columns of Z represent the PCs. The number of PCs is equal to the dimensions of the original data matrix. The equation of Z can be given by:

$$Z = YA. \quad (5)$$

III. MACHINE LEARNING-BASED CLASSIFICATION ALGORITHMS

A. Logistic Regression (LR)

Linear regression is one of the simplest and most common machine learning algorithms. It is a mathematical approach used to perform predictive analysis. Linear regression allows continuous/real or mathematical variables projections. Sir Francis Galton first suggested the concept of linear regression in 1894. Linear regression is a mathematical test used for evaluating and quantifying the relationship between the considered variables [5]. LR labels the samples as 1 or 0. For example: if the value of the sample is 0.49 or below, it labels the sample as 0. On the other hand, if the value of the sample is 0.5 or above, the LR classifies the sample as 1. This decision is determined by a function, which is called: logistic function. The logistic function can be expressed by

$$S(z) = \frac{1}{1+e^{-z}}, \quad (6)$$

Here, the output of S(z) is between 0 and 1 and z represents the input to the function. The main advantage of using LR is that it is very easy to implement and can handle the binary classification problem of room occupancy classification such as if the room status is occupied or not. However, LR is prone to overfitting in high dimensional datasets. It is also sensitive to outliers that means if any sample is much deviated from the expected range, then the classification may provide incorrect results. However, this problem can be overcome with proper feature scaling.

B. K- nearest neighbor (K-NN)

The k-Nearest Neighbor is one of the simplest Machine Learning algorithms. Besides its simplicity, k-Nearest Neighbor is a widely used technique, being successfully applied in a large number of domains. Most of the recent interest in the k-Nearest Neighbor search is due to the increasing availability of data represented in new data types such as free text, images, audio, video and so on [6]. Its efficiency as

being a lazy learning method without pre modeling prohibits it from being applied to areas where dynamic classification is needed for large repository. [7]. As a lazy-learner, in the training phase, K-NN only stores the data and performs the computation during the classification process. K-NN is one the simplest classification algorithms, where an object is classified by a majority vote of its neighbors, with the object being assigned to the most common class amongst its k numbers of nearest neighbors. K-NN uses all labeled training instances as a model of the target function. To classify a sample, at first K-NN selects the number of neighbors, calculates the Euclidean distance of k number of neighbors, takes the K nearest neighbors as per the calculated Euclidean distance. Finally, K-NN assigns the new sample to the class which has the maximum number of samples.

C. Quadratic Discriminant Analysis (QDA)

A standard approach to supervised classification problems is quadratic discriminant analysis (QDA), which models the likelihood of each class as a Gaussian distribution, then uses the posterior distributions to estimate the class for a given test point [8]. For each class of observations, a unique covariance matrix is estimated in QDA. When it is known in advance that specific classes have discernible covariance, QDA performs well.s. Mathematically, the covariance matrix Σy is required to be estimated separately for each class y, y = 1, 2, ..., K. The QDA function is defined as:

$$\delta_y(x) = -\frac{1}{2}\log|\Sigma_k| - \frac{1}{2}(x-\mu_k)^T\Sigma_k^{-1}(x-\mu_k) + \log_{\pi_k}, \quad (7)$$

where x represents the test instance, Σy represents the covariance matrix of class y, μk is the mean vector of class y and πk is the prior probability of class y. The classification rule for QDA is to find the class y which maximizes the quadratic discriminant function. Specifically, the classification rule for QDA can be described as below:

$$G(x) = arg\max_k \delta_k(x). \quad (8)$$

Since the number of QDA parameters is quadratic, QDA should be used with care when the feature space is large.

IV. DATA SET DESCRIPTION

The dataset “Occupancy Detection Data Set UCI” is obtained from Kaggle. This dataset gives information related to room occupancy classification. Dataset contains 7 columns; Occupancy is the class variable which is affected by the other 6 columns. Here the aim is to reduce the number of the columns to the best according to the accuracy metric. The features describe the room occupancy status, the features are: Date time year-month-day hour: minute: second, Temperature, in Celsius, Relative Humidity, %, Light, in Lux, CO2, in ppm, Humidity Ratio, Derived quantity from temperature and relative humidity, in kgwater-vapor/kg-air and Occupancy, 0 or 1, 0 for not occupied, 1 for occupied status (Date is dropped and so are the duplicates)

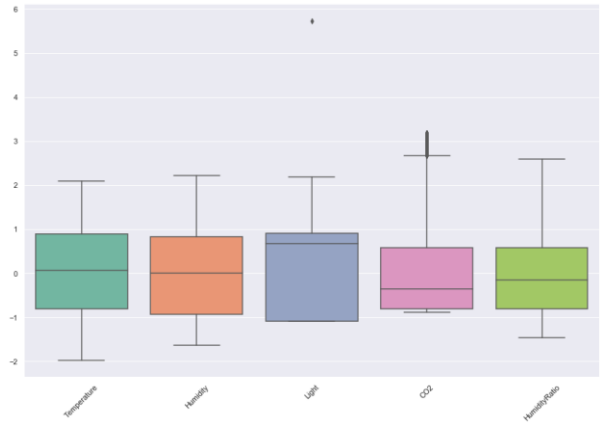


Fig. 1: Box Plot

Utilizing the box and whisker plots and their five number summaries on the dataset, the distributions, central values, and variability of the features were measured. Fig. 1 illustrates the box plot of the features of the room occupancy dataset. It can be observed from Fig. 1 that most of the features follow approximately normal distribution. However, outliers exist in some features.

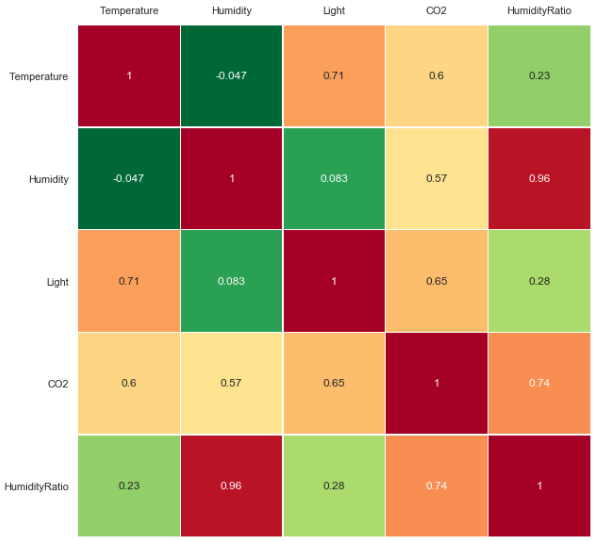


Fig. 2 Correlation matrix

Fig. 2 shows the correlation matrix for the normalized features of the dataset. The features with large positive numbers are Light, CO2, and Humidity Ratio. This evidence implies that these three features are highly correlated. Other two features, i.e. oldpeak and slope show less correlation with the other features in the dataset. The pairplot in Fig. 3 supports this observation. The highly correlated features contain a higher number of cells with regularly increasing lines. On the contrary, Light and Humidity display less apparent correlation.

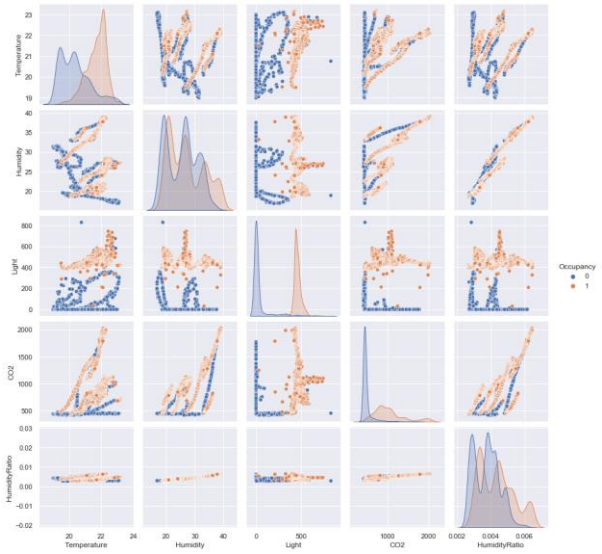


Fig. 3 Pairplot

V. PCA RESULTS

PCA is applied on the room occupancy dataset. Implementation of PCA can be done in two ways: (1) developing PCA from scratch using standard Python libraries such as numpy, (2) using popular and well documented PCA libraries. In the Google Colab notebook implementation of both methods is provided. Even though results obtained from both ways are similar, usage of the PCA library brings more flexibility to the user and a lot can be done by writing only a single line of code. In this report, the figures and plots are shown from the implementation using the PCA library. By applying the PCA steps, the feature set of 5 can be reduced to r numbers of features where r < 5. The original n × p dataset is reduced using the eigenvector matrix A. Each column of the eigenvector matrix A is represented by a PC. Each PC captures an amount of data that determines the dimension (r). The obtained eigenvector matrix (A) for the room occupancy dataset is as follows:

$$A = \begin{bmatrix} 0.358 & 0.547 & -0.673 & 0.295 & -0.180 \\ 0.409 & -0.559 & 0.058 & 0.264 & -0.669 \\ 0.397 & 0.481 & 0.725 & 0.292 & 0.019 \\ 0.545 & 0.068 & -0.006 & -0.834 & -0.054 \\ 0.500 & -0.391 & -0.135 & 0.250 & 0.719 \end{bmatrix}$$

and the corresponding eigenvalues are:

$$\lambda = \begin{bmatrix} 2.974 \\ 1.580 \\ 0.290 \\ 0.157 \\ 0.001 \end{bmatrix}$$

Fig 4 and pareto plot in Fig. 5 demonstrate the scree plot and pareto plot of the PCs. The scree plot and pareto plot display the amount of variance explained by each principal component. The percentage of variance experienced by j-th PC can be evaluated using the following equation:

$$j = \frac{\lambda_j}{\sum_j^p \lambda_j} \times 100, j = 1, 2, ..., p, \quad (9)$$

Where λj represents the eigenvalue and the amount of variance of the j-the PC.

Figs 4 and 5 plot the number of PCs vs the explained variance. It can be observed from both figures that the variance of first three PC's contribute to 96.8% of the amount of variance of the original dataset; but the first two contribute to 91.06% and that can give a great representation of the dataset
i.e., the first PC holds 59.46% of variance, the second PC holds 31.6% of variance, and the third PC holds 5.79% of variance. The scree plot presents that the elbow is located on the third PC. These observations imply that the dimension of the featureset can be reduced to eight (r = 3) but also the first two PC's did great on the variance.

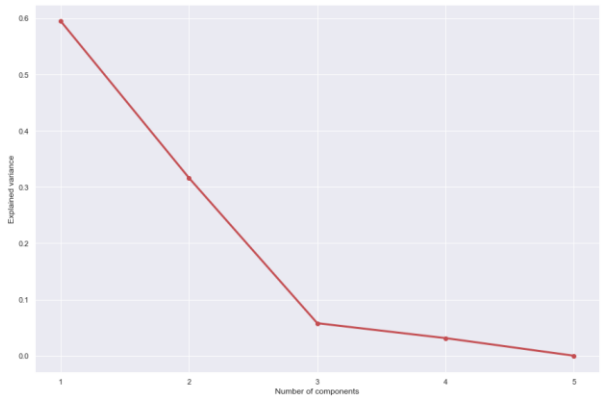


Fig. 4: Scree Plot

The first principal component Z1 is given by:
Z1 = 0.3579X1 + 0.4086X2 + 0.3967X3 + 0.5451X4 + 0.5004X5
It can be observed from the first PC that X4 (CO2) and X5 (Humidity Ratio) contribute the most to the first PC. However, none of the features have a negligible contribution to the first PC.

The second principal component Z2 is given by:
Z2 =0.5474X1 - 0.5585X2 + 0.4808X3 + 0.0679X4 - 0.3907X5
From the second PC Z2 it can be seen that X1 (Temperature), X2 (Humidity) have the highest contribution in the second PC while X4 (CO2) has a negligible contribution so it can be ignored.
Z2 =0.5474X1 - 0.5585X2 + 0.4808X3 - 0.3907X5

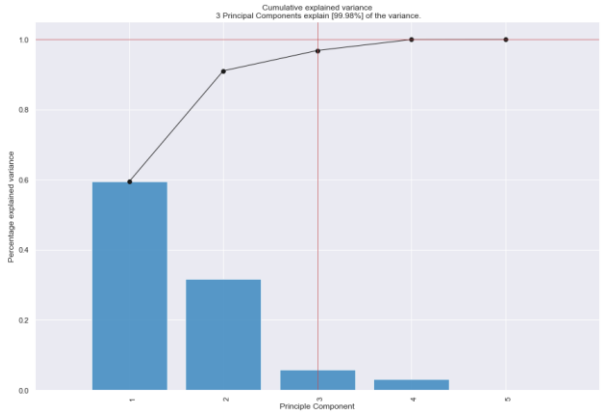


Fig. 5: Pareto Plot

Fig. 6 represents the PC coefficient plot. It visually represents the amount of contribution each feature has on the first two PCs. The figure supports the previous calculation of PCs, and it can be clearly seen from the figure that CO2 and Humidity Ratio have the highest contributions in the first PC. On the other hand, Humidity, Temperature and Light have the greatest contribution to the second PC.

The Biplot in Fig. 7 displays a different visual representation of the first two PCs. The axes of biplot represent the first two PCs. The rows of the eigenvector matrix are shown as a vector. Each of the observations in the dataset is drawn as a dot on the plot. The vectors for features namely, CO2 and Humidity Ratio show very small angles with the first PC. This evidence supports the analysis of the PC coefficient plot of Fig. 6. It implies that these three features have a large contribution to the first PC, while CO2 has a small contribution to the second PC.

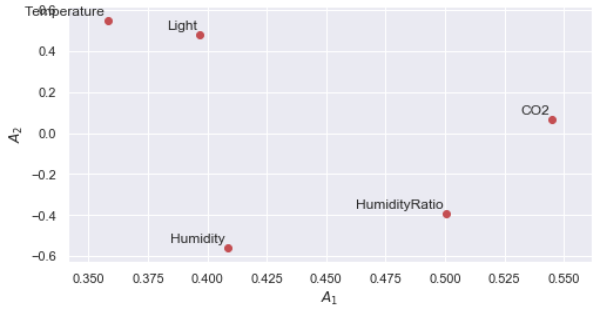


Fig. 6: PC coefficient plot

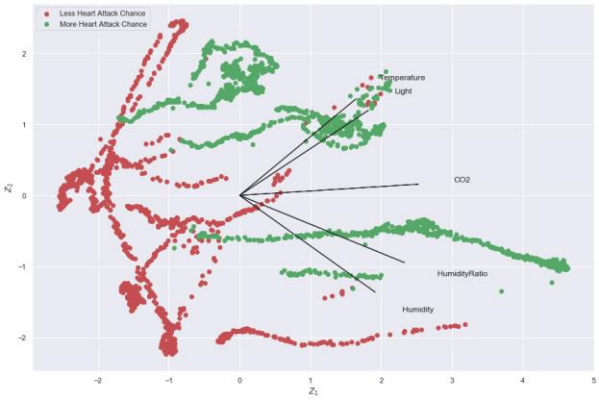


Fig. 7: Biplot

VI. CLASSIFICATION RESULTS

In this section, the performance of three popular classification algorithms on the room occupancy dataset is discussed. To observe the effects of PCA on the room occupancy dataset, the classification algorithms are applied on the original dataset as well as the PCA applied dataset with three PCA components. The classification is performed using PyCaret library of Python. The original dataset is split into a train and test set with the proportion of 70% and 30%, respectively. For the sake of reproducibility, the session id is set with 123 and provided 4 workers for the sake of the parallelism. Using PyCaret, it is possible to create a performance comparison table among all available classification algorithms on the target dataset and find the best model with the highest accuracy. It can be observed from Fig. 8 that, before applying PCA, the best three classification models with the highest accuracies on the room occupancy dataset are Random Forest Classifier (rf), Extra Trees Classifier (et), and Ada Boost Classifier (ada).

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
rf	Random Forest Classifier	0.9936	0.9986	0.9963	0.9909	0.9936	0.9871	0.9872	0.1280
et	Extra Trees Classifier	0.9931	0.9992	0.9945	0.9917	0.9931	0.9862	0.9862	0.0890
ada	Ada Boost Classifier	0.9922	0.9974	0.9935	0.9908	0.9922	0.9844	0.9844	0.0550
xgboost	Extreme Gradient Boosting	0.9922	0.9991	0.9926	0.9917	0.9922	0.9844	0.9844	0.0620
lightgbm	Light Gradient Boosting Machine	0.9917	0.9974	0.9917	0.9918	0.9917	0.9834	0.9835	0.1120
gbc	Gradient Boosting Classifier	0.9913	0.9981	0.9908	0.9917	0.9912	0.9825	0.9826	0.0950
dt	Decision Tree Classifier	0.9908	0.9908	0.9889	0.9926	0.9907	0.9816	0.9816	0.0070
knn	K Neighbors Classifier	0.9904	0.9921	0.9991	0.9820	0.9904	0.9807	0.9809	0.0110
qda	Quadratic Discriminant Analysis	0.9894	0.9922	0.9972	0.9819	0.9895	0.9789	0.9790	0.0080
lr	Logistic Regression	0.9862	0.9923	0.9908	0.9818	0.9862	0.9724	0.9725	0.5390
nb	Naive Bayes	0.9816	0.9916	0.9972	0.9670	0.9819	0.9632	0.9638	0.0060
ridge	Ridge Classifier	0.9784	0.0000	0.9981	0.9602	0.9788	0.9568	0.9576	0.0070
lda	Linear Discriminant Analysis	0.9770	0.9901	0.9981	0.9576	0.9774	0.9540	0.9550	0.0060
svm	SVM - Linear Kernel	0.9669	0.0000	0.9926	0.9454	0.9679	0.9339	0.9361	0.0080
dummy	Dummy Classifier	0.5023	0.5000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0050

Fig. 8: Comparison among classification models before applying PCA

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
xgboost	Extreme Gradient Boosting	0.9848	0.9978	0.9889	0.9809	0.9848	0.9697	0.9698	0.0630
rf	Random Forest Classifier	0.9844	0.9967	0.9917	0.9775	0.9845	0.9688	0.9690	0.1160
lightgbm	Light Gradient Boosting Machine	0.9844	0.9981	0.9861	0.9827	0.9843	0.9688	0.9690	0.0390
et	Extra Trees Classifier	0.9830	0.9972	0.9898	0.9765	0.9831	0.9660	0.9662	0.0960
knn	K Neighbors Classifier	0.9825	0.9957	0.9926	0.9731	0.9827	0.9651	0.9654	0.0120
dt	Decision Tree Classifier	0.9802	0.9802	0.9788	0.9817	0.9801	0.9605	0.9607	0.0060
gbc	Gradient Boosting Classifier	0.9802	0.9962	0.9898	0.9712	0.9804	0.9605	0.9608	0.0750
ada	Ada Boost Classifier	0.9697	0.9938	0.9760	0.9638	0.9697	0.9393	0.9397	0.0490
ridge	Ridge Classifier	0.9278	0.0000	0.9455	0.9130	0.9288	0.8557	0.8566	0.0060
lda	Linear Discriminant Analysis	0.9278	0.9747	0.9455	0.9130	0.9288	0.8557	0.8566	0.0060
svm	SVM - Linear Kernel	0.9269	0.0000	0.9492	0.9091	0.9284	0.8539	0.8555	0.0070
lr	Logistic Regression	0.9237	0.9759	0.9344	0.9146	0.9242	0.8474	0.8479	0.0110
qda	Quadratic Discriminant Analysis	0.9237	0.9749	0.9307	0.9177	0.9239	0.8474	0.8479	0.0060
nb	Naive Bayes	0.9205	0.9720	0.9307	0.9122	0.9210	0.8410	0.8417	0.0060
dummy	Dummy Classifier	0.5023	0.5000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0080

Fig. 9: Comparison among classification models after applying PCA

```
tuned_lr_pca = tune_model(lr_pca)
```
✓ 0.7s

	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	0.8000	0.8687	0.7273	0.8889	0.8000	0.6040	0.6162
1	0.8421	0.9222	0.7000	1.0000	0.8235	0.6885	0.7246
2	0.8947	0.9444	0.9000	0.9000	0.9000	0.7889	0.7889
3	0.8421	0.8778	0.8000	0.8889	0.8421	0.6851	0.6889
4	0.8947	0.9556	0.9000	0.9000	0.9000	0.7889	0.7889
5	0.7895	0.9091	0.8182	0.8182	0.8182	0.5682	0.5682
6	0.7895	0.9091	0.7273	0.8889	0.8000	0.5824	0.5955
7	0.8947	0.9659	0.9091	0.9091	0.9091	0.7841	0.7841
8	0.8421	0.8750	0.8182	0.9000	0.8571	0.6816	0.6854
9	0.8421	0.8295	1.0000	0.7857	0.8800	0.6587	0.7008
Mean	0.8432	0.9057	0.8300	0.8880	0.8530	0.6830	0.6941
SD	0.0395	0.0410	0.0917	0.0536	0.0402	0.0793	0.0767

Fig. 10: LR metrics score after hyperparameter tuning.

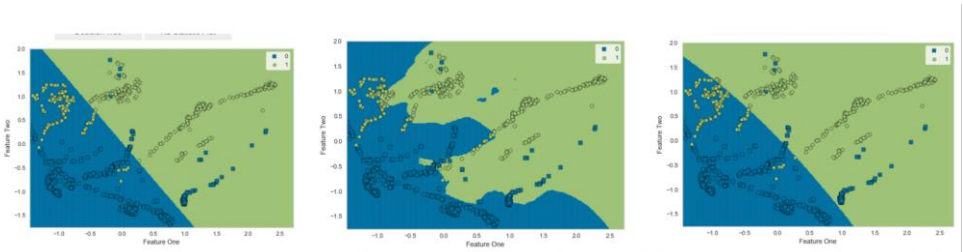
On the other hand, Fig. 9 demonstrates the comparison among the classification models after applying PCA. Here, the best three models which give the highest accuracy on the transformed dataset are Extreme Gradient Boosting, Random Forest Classifier, and Light Gradient Boosting Machine. Hence, these three algorithms are taken for the evaluation purposes during the rest of the experiment. The original and transformed dataset are trained, tuned, and evaluated using these three algorithms. Both experiments (classification algorithms applied on original dataset and transformed dataset) can be found in Google Colab

notebook. However, in this report, we only focus on the results obtained after applying PCA (transformed dataset) by using LR (Logistic Regression), KNN (K-Nearest Neighbor) and QDA (Quadratic Discriminant Analysis) only.

To improve the performance of a model, hyperparameters tuning plays an important role. Hyperparameter tuning with PyCaret involves three steps; create a model, tune it and evaluate its performance. First, a classification model per algorithm is produced. Then the tune model function is used for tuning the model with ideal hyperparameters. This function automatically tunes the model with effective hyperparameters on a pre-defined search space and scores it uses stratified K-fold cross validation. By default, PyCaret applies 10-fold stratified K-fold validation on the three algorithms. Moreover, the LR model is tuned with L2 penalty, a regularization technique to prevent overfitting problems. For KNN, the number of k nearest members are tuned and the reg param for regularization is tuned for QDA. It can be observed from Fig. 10 that tuned LR model metrics are better than the base model metrics (before hyperparameter tuning).

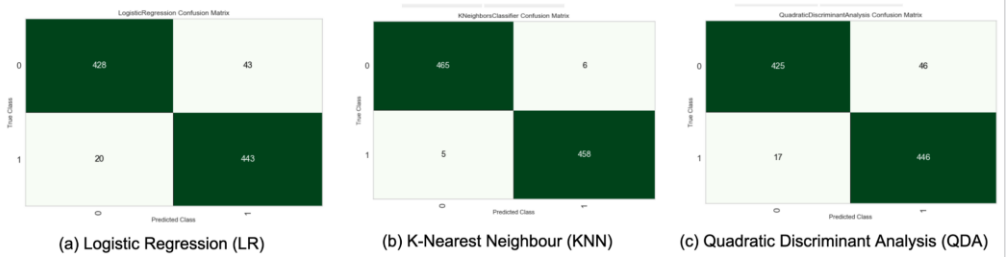
Fig. 11 illustrates the decision boundaries formed by the model on the transformed dataset. A decision boundary is a hyperplane that separates data points into specific classes and the algorithm switches from one class to another. The x-axis of the figures corresponds to the first PC and y axis corresponds to the second PC. The square shaped dots represent the observations for class 0 (room not occupied status) and class 1 (room occupied status) is represented by the round shaped dots. The figure displays the differences among the decision boundaries that are formed by the algorithms. It is clearly visible from the figure that KNN has the best decision boundary than LR and QDA. The decision boundary of KNN can separate the data instances of both classes more accurately and the F1 score of it shows the difference.

Since the room occupancy dataset is a binary classification problem, precision and recall can evaluate the performance of each class individually. Precision and recall are two measurements which together are used to evaluate the performance of classification. ‘‘Precision’’ is usually defined as the degree to which a score obtained by a person on one occasion is repeated on a second occasion (i.e., test–retest reliability in more traditional terms); or a score given by one rater is matched by that given by a second rater (i.e., interrater reliability) [9]. The obtained results from precision and recall are presented using the confusion matrices Fig. 12. A confusion matrix (also known as a contingency table or an error matrix) is a table layout that allows visualization of the performance of a supervised learning algorithm [10]. It breaks down for each class and uses count numbers to show correct and erroneous predictions. Fig. 12 shows the confusion matrix tables for the three algorithms which were applied on transformed dataset. The confusion matrices for the original dataset can be found in the Google Colab notebook. In the figure, the horizontal axis represents the class prediction, and the vertical axis represents the true label. First, the comparison model on Fig. 9 shows that KNN outperforms other compared models: LR and QDA. This observation is also supported by the confusion matrices of Fig.12. LR misclassified the lowest numbers of instances. For example, LR misclassified 43 instances from class 0 (Room not occupied status) as class 1 (Room occupied status) and 20 instances of class 1 (Room occupied status) are misclassified as class 0 (Room not occupied status). On the other hand, K-NN misclassified 6 instances of class 0 (Room not occupied status) and 5 instances of class 1 (Room occupied status) whereas QDA misclassified 46 instances of class 0 (Room not occupied status) and 17 instances of class 1 (Room occupied status).



(a) Logistic Regression (LR) (b) K-Nearest Neighbour (KNN) (c) Quadratic Discriminant Analysis (QDA)

Fig. 11: Decision Boundaries of the three algorithms applied on transformed dataset



(a) Logistic Regression (LR) (b) K-Nearest Neighbour (KNN) (c) Quadratic Discriminant Analysis (QDA)

Fig. 12: Confusion matrices of the three classification algorithms applied on transformed dataset

Another measurement of the performance evaluation is F1- score. The F1-score combines the precision and recall of a classifier into a single metric by taking their harmonic mean [9]. It is a great metric to compare the results among the classifiers. For example: classifier A has a higher recall, and classifier B has higher precision. In this situation, the F1-score helps to determine the better classifier. The function of F1- score can be defined as below:

$$F1 - score = 2 \times \frac{precision \times recall}{precision + recall}. \quad (13)$$

It can be observed from Fig. 8 and Fig. 9 that the F1-score of all compared algorithms has improved after applying PCA. This observation implies the fact that dimension reduction weakens the dependencies among the features. All these observations are evidence of the benefits of applying PCA and hyperparameters tuning.

As the final analysis step, the receiver operating characteristic (ROC) curve for the LR algorithm is shown in Fig. 13. The ROC curve is a graph illustrating the performance of the classifier model at all classification thresholds. This curve plots two parameters: true positive rate and false positive rate. These parameters are key components in the construction of the confusion matrix. Therefore, the ROC curve and the confusion matrix are closely related and can be thought of as different visual representations of the same measurement. You can find ROC curves for K-NN and QDA in the Google Colab manual. The ROC curve of the LR in Figure 13 reflects the results of the confounding matrix. It plots the false-positive rate (x-axis) versus the true-positive rate (y-axis) for several different candidate threshold values between 0.0 and 1.0. It also displays macro and micro average curve graphs. The other graphs of KNN and QDA have no significant differences from LR and can be found in Google Colab notebook. This result shows that all of the three algorithms can be used for this classification task.

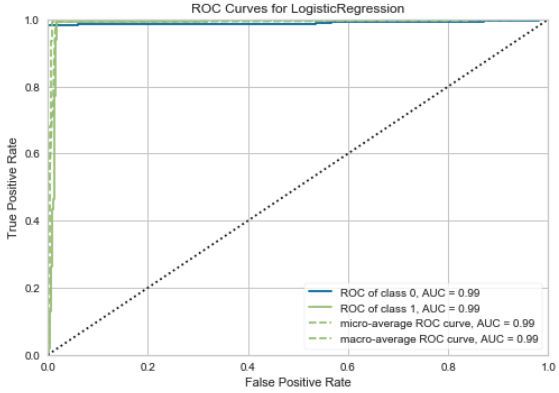


Fig. 13: ROC curve LR

VII. EXPLAINABLE AI WITH SHAPLEY VALUES

Model interpretability is an important metric in the context of ML. There are several ways of enhancing the explainability of a model and feature importance is one of them. Feature importance helps to estimate the contribution of each feature in the prediction process. Hence to get an overview of the most important features on the PCs, we use the SHAP

values by importing the open source "shap" library of Python.

SHAP (Shapley Additive Explanations) is a method that was first introduced by Lundberg and Lee [11] in 2016. SHAP explains individual predictions based on the optimal Shapley values using the concept of game theory. Specifically, SHAP can explain the prediction of an instance x by computing the contribution of each feature to the prediction [12]. Borrowing the concept of game theory, each feature of a dataset acts as players in a coalition. A player can be an individual feature value, e.g., for tabular data. or a group of feature values. Shapley values describe how to adequately distribute the prediction among the feature set. The SHAP library of Python is still in its development stage, and it only supports tree-based models (i.e; decision tree, random forest, extra trees classifier) for binary classification problems. Since the room occupancy chance dataset is a binary classification problem, shap analysis cannot be performed on LR, KNN, and QDA. Therefore, for the shap analysis, I have chosen the fourth best model of the transformed dataset, which is "Extra trees classifier (ET)". Similar to other models, at first, an ET model is created and tuned with ideal hyperparameters. Then the tuned model is passed to the shap library for producing the interpretation plots. For our case, each PC acts as a player in the coalition. Fig. 14 displays the summary plot of SHAP values. The summary plot combines feature importance with feature effects. Each point on the summary plot is a Shapley value for a PC and an instance. The y-axis represents the PCs and Shapley values are positioned on the x-axis. More specifically, component_1 represents the first PC, component_2 represents the second PC. We can observe some jittered overlapping points in the direction of the y-axis which indicates the distribution of the Shapley values per PC. All the PCs are ordered according to their importance. This observation supports the pareto plot and scree plot which indicates that the first PC holds the most feature variance. The red color indicates high PC value and blue color indicates low PC value. To interpret the summary plot, we can say that a low level of PC value has a high and positive impact on the room occupancy status. Similarly, a high level of PC value has a low and negative impact on the room occupancy status prediction. More clearly, PCs are negatively correlated with the target variable.

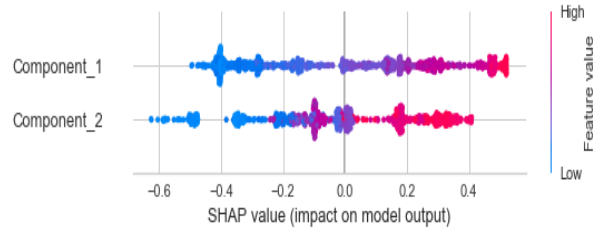


Fig. 14: Summary plot

Fig. 15: Force plot for a single observation

Fig. 15 represents the force plot for a single observation. It should be noted that this plot can only be made for one observation. This plot displays the features each contributing to push the model output from the base value. The base value means the value that would be predicted if no features are known for the current output [11]. In other words, it is the mean prediction of the test set. Here, the base value is 0.5. In the plot, the bold 0.00 is the model's score for this observation. Higher scores lead the model to predict 1 and lower scores lead the model to predict 0. The blue color on the second PC (Component_2) indicates that it is pushing the prediction to be lower. This particular observation is classified as class 0 (Room not occupied status) because it is pushed mostly towards the left. However, this plot is only an output for this specific observation. It does not describe the predicted output of the entire model. Fig.16 displays the combined force plot of all PCs. This plot is a combination of all individual force plots with 90-degree rotation and are stacked horizontally. In this plot, the y-axis is the x-axis of the individual force plot. There are 934 data points in the transformed test set, hence the x-axis has 934 observations. This combined force plot shows the influence of each PC on the current prediction. Values in the blue color are considered to have a positive influence on the prediction whereas values in the red color have a negative influence on the prediction.

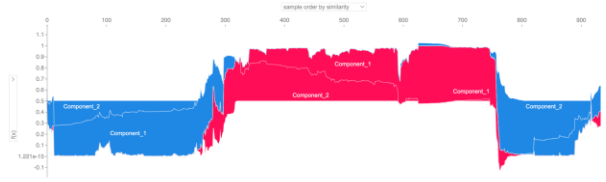


Fig. 16: Combined force plot

VIII. CONCLUSION

In conclusion, PCA and three popular classification algorithms are applied on the room occupancy dataset. The room occupancy dataset holds information on attributes of personal information about each patient to decide if the room status was occupied or not. First, PCA is applied on the original dataset. The first two PCs apprehend 91.06% variance of the data. Hence, the feature set can be reduced to 2 from 5 (after dropping date). Extensive experiments are conducted on the first two PC's and different plots are generated to validate the obtained results from different perspectives. To move forward, three classification algorithms, LR, K-NN and QDA, are applied on the original dataset as well as transformed dataset with first two components.

The optimal hyperparameter values are chosen for each method, and performance is assessed by contrasting confusion matrices, ROC curves, and F1-scores. It has been noted that each algorithm's performance metrics score has dramatically increased following hyperparameter adjustment. For the original dataset, the RF, ET, and ADA algorithms performed the best. It's interesting to note that after applying PCA, XGBOOST, RF, and LightGBM performed best and displayed the best performance metrics. Lastly, a number of interpretation charts are created using explainable AI shapley variables to improve the model's interpretability. In conclusion, all three systems can accurately predict the occupancy status of a room (occupied or not).

REFERENCES

[1] Wilhelm Kleiminger, Christian Beckel, Thorsten Staake, Silvia Santini "Occupancy Detection from Electricity Consumption Data" BuildSys'13: Proceedings of the 5th ACM Workshop on Embedded Systems for Energy-Efficient BuildingsNovember 2013 Pages 1–8

[2] Yordan P. Raykov , Emre Ozer , Ganesh Dasika , Alexis Boukouvalas , Max A. Little . "Predicting room occupancy with a single passive infrared (PIR) sensor through behavior extraction" UbiComp '16: Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous ComputingSeptember 2016 Pages 1016–1027

[3] Markus Ringnér "What is principal component analysis?" Nature Biotechnology volume 26, pages303–304 (2008)

[4] A. B. Hamza, Advanced Statistical Approaches to Quality. Unpublished.

[5] Dastan Hussen Maulud , Adnan Mohsin Abdulazeez "A Review on Linear Regression Comprehensive in Machine Learning" Maulud et al. / Journal of Applied Science and Technology Trends Vol. 01, No. 04, pp. 140 –147, (2020)

[6] Gustavo E.A.P.A. Batista , Diego Furtado Silva "How k-Nearest Neighbor Parameters Affect its Performance?" Caixa Postal 668 – 13560-970 – S˜ao Carlos – SP – Brasil

[7] Gongde Guo , Hui Wang , David Bell , Yaxin Bi , and Kieran Greer "KNN Model-Based Approach in Classification" Part of the Lecture Notes in Computer Science book series (LNCS,volume 2888)

[8] Santosh Srivastava, Maya R. Gupta, Bela´ A. Frigyik "Bayesian Quadratic Discriminant Analysis" Journal of Machine Learning Research 8 (2007) 1277-1305

[9] David L. Streinera,*, Geoffrey R. Normanb "''Precision'' and ''Accuracy'': Two Terms That Are Neither" Journal of Clinical Epidemiology 59 (2006) 327–330

[10] V.Mohan Patro , Manas Ranjan Patra "Augmenting Weighted Average with Confusion Matrix to Enhance Classification Accuracy" TRANSACTIONS ON MACHINE LEARNING AND ARTIFICIAL INTELLIGENCE VOLUME 2, ISSUE 4 ISSN: 2054 - 7390

[11] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," Advances in neural information processing systems, vol. 30, 2017.

[12] C. Molnar, Interpretable Machine Learning, 2nd ed., 2022. [Online]. Available: https://christophm.github.io/interpretable-ml-book