Principal Component Analysis and Boston Housing Price Prediction Using Machine Learning

Alireza Lorestani 40253734
GitHub Link: github.com/alirezalorestani23/INSE6220

***Abstract*—The Boston Housing dataset is a popular dataset in machine learning used to predict the value of houses in Boston. In this research project, Principal Component Analysis (PCA) is used to reduce dimensionality in the Boston Housing dataset. PCA is able to reduce the number of correlated variables in the dataset while maintaining most of the vital information. Four classification algorithms, namely Decision Tree, KNN, Logistic Regression, and LDA are applied to both the original dataset and the transformed dataset (after applying PCA) to predict the house prices. The models are then fine-tuned with the best hyperparameters and their performance is evaluated using F1 score, confusion matrix, and receiver operating characteristic (ROC) curves. The decision boundaries of LDA model is also visualized to show it fit to the dataset. The results show that LDA, among those four, outperforms the other models. Finally, the Shapley values of the Random Forest classifier model are used to interpret the model's predictions. The models successfully predict housing prices, achieving an F1 score close to 1.**

***Index Terms*—Principal component analysis, classification, Decision Tree, KNN, Logistic Regression, Random Forest**

I. Introduction

Housing prices are a critical aspect of the real estate market and affect the livelihoods of millions of people around the world. Accurately predicting housing prices is essential for making informed decisions, whether for buying or selling a property, assessing market trends, or forecasting economic indicators. This is where machine learning and data analysis techniques, such as principal component analysis (PCA) and various classification algorithms, can play a significant role in providing insights and making accurate predictions based on historical data [1]–[3].

The Boston Housing dataset is a widely used dataset in machine learning, which contains information about various factors that affect the housing prices in Boston [4]. In this report, we aim to apply Principal Component Analysis (PCA) and machine learning techniques on the Boston Housing dataset to predict housing prices. The first step of the analysis involves applying PCA to reduce the dimensionality of the dataset, which can help in improving the performance of the machine learning algorithms. Four popular classification algorithms, namely Decision Tree, KNN, and Logistic Regression are applied to the original dataset, and LDA is applied to the PCA-transformed dataset. The objective is to determine the factors that significantly impact housing prices and to build a model that can accurately predict housing prices. Finally, the performance of the models is evaluated using various metrics. [1]–[3].

The rest of the report is organized as follows: Section II describes the methodology of PCA, Section III provides an overview of the machine learning algorithms used in this study, Section IV describes the Boston Housing dataset, Section V presents the PCA results, Section VI analyzes the performance of the machine learning models, and Section VII concludes the report.

II. PRINCIPAL COMPONENT ANALYSIS

Principal Component Analysis (PCA) is a technique used for reducing the complexity of high-dimensional datasets while preserving trends and patterns. Most real-world datasets contain high dimensionality, which makes processing and storing them expensive and sometimes impossible to visualize. PCA can help to reduce the dimensionality of large datasets by transforming a large set of variables into a smaller one that still contains most of the information of the original dataset [5], [6].

PCA accomplishes its goal by condensing the data into fewer dimensions that serve as feature summaries. This is achieved by following a set of steps that include standardization, covariance matrix computation, eigen decomposition, and principal components computation. PCA helps to identify relationships among the variables in the dataset and extract the most important features, which can be used to make accurate predictions [5], [6].

A. PCA algorithm

PCA can be applied to a data matrix X with dimension n×p using the following steps [7]:

1) **Standardization:** The main goal of this step is to standardize the initial variables so that they all contribute equally to the analysis. At first, the mean vector $\bar{x}$ of each column of the dataset is computed. 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: $Y = HX$, 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. In order to identify these correlations, the covariance matrix is computed. The $p \times p$ covariance matrix is computed as follows:

$S = \frac{1}{n-1} Y^T Y$.

3) **Eigen decomposition:** Using the eigen decomposition, the eigenvalues and eigenvectors of S can be computed. Eigenvectors represent the direction of each principal component (PC), whereas eigenvalues represent the variance captured by each PC. Eigen decomposition can be computed using the following equation: $S = A\Lambda A^T$, where A is the $p \times p$ orthogonal matrix of eigenvectors and Λ is the diagonal matrix of eigenvalues.

4) **Principal components:** It computes the transformed matrix Z that is size of $n \times p$. The rows of Z represent the observations and columns of Z represent the PCs. The number of PCs is equal to the dimension of the original data matrix. The equation for Z can be given by: $Z = YA$ [8]

III. MACHINE LEARNING-BASED CLASSIFICATION ALGORITHMS

A. Decision Tree

Decision Tree is a supervised learning algorithm that can be used for both classification and regression tasks [9]. It creates a tree-like model of decisions and their possible consequences. Each internal node of the tree represents a test on a feature, each branch represents the outcome of the test, and each leaf node represents a class label or a numeric value. The goal of the decision tree is to create a model that predicts the value of a target variable based on several input variables. Decision Tree can handle both categorical and numerical data, and it is easy to interpret and visualize the model. One of the most popular decision tree algorithms is the Classification and Regression Tree (CART) algorithm, which uses the Gini impurity or entropy to measure the quality of a split [10]. Formula: The Gini impurity measure is given by:

$$I_G(t) = \sum_{i=1}^{c} p(i|t)(1 - p(i|t)) \tag{1}$$

where c is the number of classes, $p(i|t)$ is the proportion of the samples that belong to class i at $node_t$, and $I_G(t)$ is the Gini impurity at $node_t$.

B. K-Nearest Neighbors (K-NN)

K-NN is a non-parametric lazy learning algorithm that can be used for both classification and regression tasks [10]. It works by finding the k-nearest neighbors of a test instance in the feature space and then assigning the test instance to the most common class or the average of the k-nearest neighbors. The choice of k is a hyperparameter that needs to be tuned for optimal performance. K-NN can handle both categorical and numerical data, and it is easy to implement and understand. However, K-NN is sensitive to the curse of dimensionality, and it can be computationally expensive for large datasets. Formula: The distance between two instances x and z is given by the Euclidean distance:

$$d(x, z) = \sqrt{\sum_{i=1}^{n} (x_i - z_i)^2} \tag{2}$$

where n is the number of features.

C. Logistic Regression (LR)

LR is a statistical method that tries to create the best-fitting model in order to establish a relationship between the class and features [11]. It is used for binary classification problems, such as determining if a tumor is benign or malignant. LR works by labeling the samples as 1 or 0, based on a logistic function. The logistic function can be expressed by the equation:

$$S(z) = \frac{1}{1 + e^{-z}} \tag{3}$$

where the output of $S(z)$ is between 0 and 1 and z represents the input to the function. 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.

The classification function for LR can be defined as:

$$f(x) = \begin{cases} 1, & \text{if } S(w^T x + b) \geq 0.5 \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

where x is the input sample, w is the weight vector, b is the bias term, and S is the logistic function.

D. Random Forest

Random Forest is an ensemble method that combines multiple decision trees to improve accuracy and reduce overfitting. It randomly selects a subset of features for each decision tree and builds the trees on different subsets of the data. The final classification result is based on the majority vote of the individual trees.

The classification function for Random Forest can be defined as:

$$f(x) = mode f_1(x), f_2(x), ..., f_T(x) \tag{5}$$

where x is the test sample, T is the total number of trees, and $f_i(x)$ is the classification function of the i^{th} decision tree.

E. LDA

LDA, or Linear Discriminant Analysis, is a supervised learning algorithm used for classification problems. The goal of LDA is to find a linear combination of features that best separates the classes in the data. The algorithm maximizes the ratio of the between-class variance to the within-class variance. This can be achieved by computing the scatter matrices of the data and finding the projection that maximizes the between-class separation and minimizes the within-class scatter.

IV. DATASET DESCRIPTION

The Boston Housing dataset is a widely used dataset in machine learning and statistics. It contains information collected by the U.S. Census Service concerning housing in the area of Boston, Massachusetts. The dataset includes 506 instances, with 13 attributes including per capita crime rate by town $CRIM$, the proportion of residential land zoned for lots over 25,000 sq. ft ZN, and nitric oxide concentration NOX among others.

For this analysis, we have selected six attributes from the dataset: $INDUS$, NOX, DIS, RAD, and TAX. These attributes were chosen because they have been shown to significantly impact the median value of owner-occupied homes $MEDV$, which is categorized in three different labels, in Boston.

- $INDUS$ represents the proportion of non-retail business acres per town. This variable has a negative impact on the value of homes as an increase in industrial development may lead to increased pollution, traffic, and noise, which can decrease the desirability of the location and ultimately lower housing prices.
- NOX represents the concentration of nitric oxides in parts per 10 million. This variable also has a negative impact on home values, as higher concentrations of NOX are associated with air pollution, which can have adverse health effects and reduce the overall desirability of a location.
- DIS represents the weighted distances to five Boston employment centers. This variable positively impacts home values as shorter distances to employment centers can make a location more attractive to potential buyers.
- RAD represents the accessibility to radial highways. This variable has a mixed impact on home values. While higher accessibility to highways can make a location more attractive for commuters, it can also lead to increased traffic and noise levels.
- TAX represents the full-value property-tax rate per \$10,000. This variable has a negative impact on home values, as higher property taxes can lead to increased costs for homeowners and reduce affordability.
- $MEDV$ represents the categorization of the median value of owner-occupied homes into three categories: affordable (A), mid-range (M), and luxury (L). This is the target variable that we aim to predict using the other attributes.

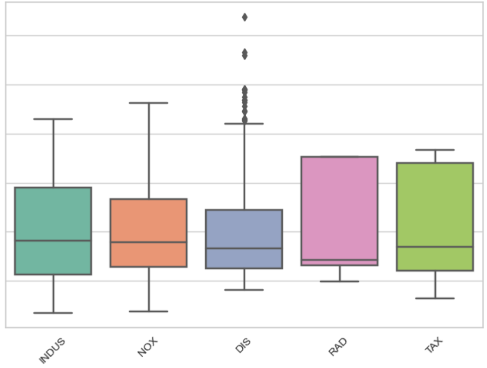


Fig. 1. Box Plot

The boxplot in 1 displays the distribution of selected features in the Boston Housing dataset. The $INDUS$ and NOX features have the largest interquartile ranges, indicating that their values are spread out over a wider range. The TAX feature has the smallest interquartile range, indicating that its values are more concentrated around the median. The median value for the DIS feature is higher compared to the other features, suggesting that houses with shorter distances to employment centers tend to have higher values. Additionally, the RAD feature has a relatively large number of outliers, indicating that some areas with higher accessibility to highways have lower housing values.

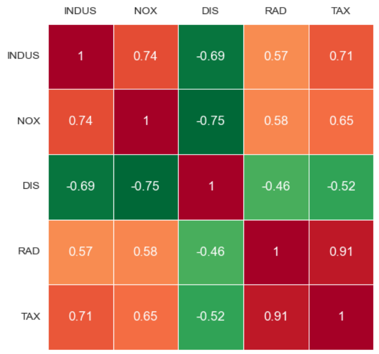


Fig. 2. Correlation Matrix

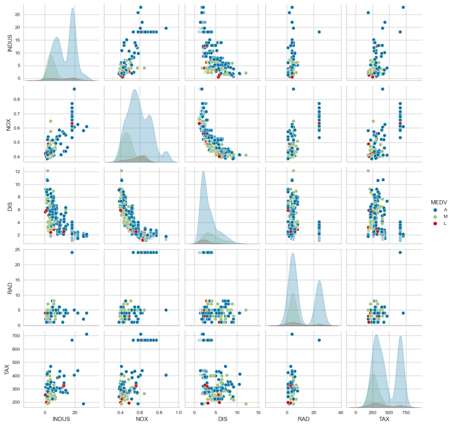


Fig. 3. Pair Plot

The correlation matrix 2 shows the pairwise correlations between the selected features INDUS, NOX, DIS, RAD, and TAX of the Boston Housing dataset. The heatmap indicates the strength and direction of these correlations using color-coded values. The strongest positive correlation is observed between TAX and RAD, which indicates that the accessibility to radial highways RAD and property taxes TAX are positively related. On the other hand, DIS and NOX have a strong negative correlation, suggesting that the weighted distances to employment centers DIS and the concentration of nitric oxides NOX are negatively related. Overall, the heatmap suggests that the selected features have both positive and negative correlations with each other, which can provide valuable insights for predicting the median value of owner-occupied homes MEDV. Figure 3 also supports the observations.

V. PCA RESULTS

PCA (Principal Component Analysis) has been used on the Boston Housing dataset with 5 features. The implementation of PCA can be done in two ways: (1) by developing PCA from scratch using standard Python libraries such as numpy, and (2) by using popular and well-documented PCA libraries. In this report, both methods have been implemented and are accessible on GitHub, but the figures and plots are shown from the implementation using the PCA library, as it brings more flexibility to the user with less code. Applying the PCA steps reduces the feature set to a smaller set of r features where r is less than 5. The original dataset of n × p is reduced using the eigenvector matrix A, where each column of the eigenvector matrix A represents a Principal Component (PC). Each PC captures an amount of data that determines the dimension (r). The obtained eigenvector matrix (A) for Boston Housing dataset is as follows:

$$A = \begin{bmatrix} -0.45 & -0.21 & 0.79 & -0.21 & -0.26 \\ -0.45 & -0.31 & -0.12 & 0.82 & 0.01 \\ 0.41 & 0.54 & 0.50 & 0.51 & -0.071 \\ -0.43 & 0.58 & -0.30 & -0.05 & -0.61 \\ -0.46 & 0.46 & 0.07 & -0.08 & 0.74 \end{bmatrix} \tag{6}$$

and the corresponding eigenvalues are:

$$\lambda = \begin{bmatrix} 3.64 \\ 0.76 \\ 0.29 \\ 0.23 \\ 0.06 \end{bmatrix} \tag{7}$$

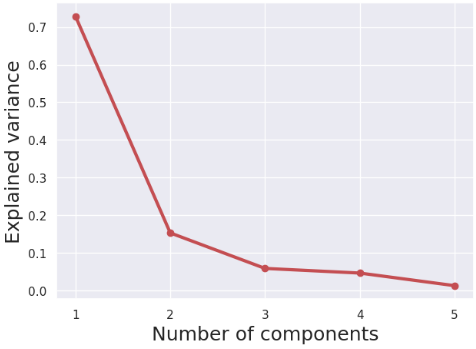


Fig. 4. Scree Plot

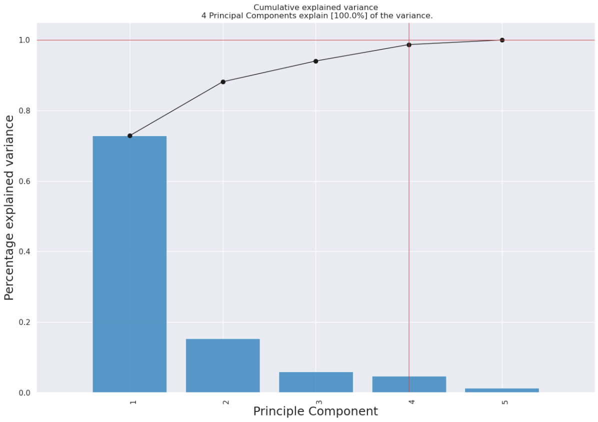


Fig. 5. Pareto Plot

Fig. 4 and the 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 the j-th PC can be evaluated using the following equation:

$$j = \frac{\lambda_j}{\sum_{i=1}^{p} \lambda_i} \times 100, \quad j = 1, 2, \ldots, p, \tag{8}$$

where λ_j represents the eigenvalue and the amount of variance of the j-th PC.

Fig. 4 and 5 plot the number of PCs vs the explained variance. It can be observed from both figures that the variance of the first two PCs contributes to 88.1% of the amount of variance of the original dataset; i.e., the first PC holds 72.8% of the variance ($l_1 = 72.8\%$) and the second PC holds 15.3% of the variance ($l_2 = 15.3\%$). The scree plot presents that the elbow is located on the second PC. These two observations imply that the dimension of the features can be reduced to two ($r = 2$).

The first principal component Z_1 is given by:

$$Z1 = -0.45x_1 - 0.45x_2 + 0.41x_3 - 0.43x_4 - 0.46x_5 \tag{9}$$

It can be observed from the first PC that all features contribute the same to the first PC. However, none of the features have a negligible contribution to the first PC.

The second principal component Z_2 is given by:

$$Z1 = -0.21x_1 - 0.31x_2 + 0.54x_3 + 0.58x_4 + 0.46x_5 \tag{10}$$

Same as the first PC none of the features have a negligible contribution. However, It can be seen that X_3 (DIS) and X_4 (RAD) have the highest contribution in the second PC.

Fig. 6 represents the PC coefficient plot. It visually represents the amount of contribution each feature has on the first two PCs and it clearly supports the previous calculation of PCs. The Biplot in Fig. 7 displays a different visual representation of the first two PCs. The axes of the 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 plot also shows the relationship

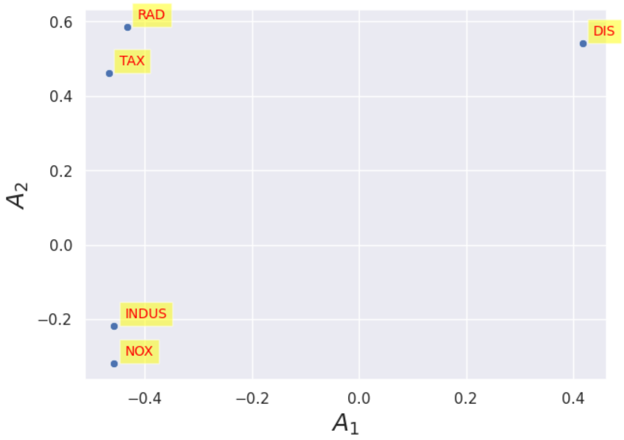


Fig. 6. Coefficient Plot

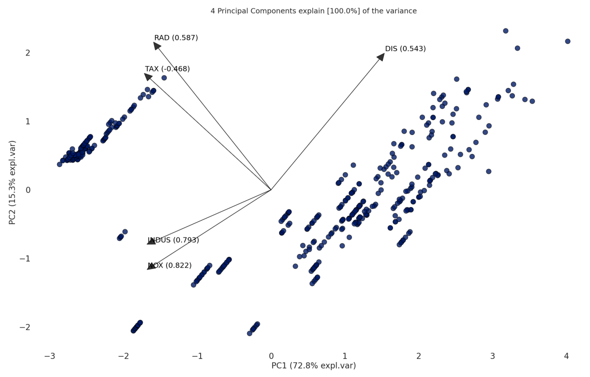
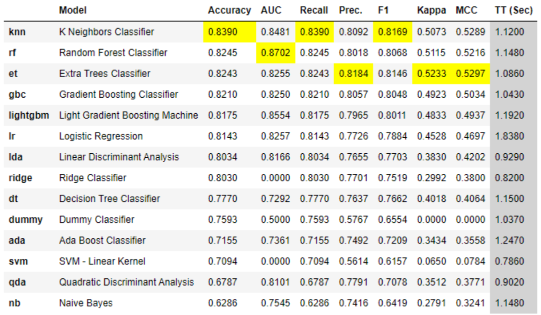


Fig. 7. Biplot

between the observations and the features. The location of each observation relative to the feature vectors shows the correlation between the observation and the feature. If an observation is closer to a feature vector, it means that the observation has a higher value for that feature. Conversely, if an observation is farther from a feature vector, it means that the observation has a lower value for that feature.

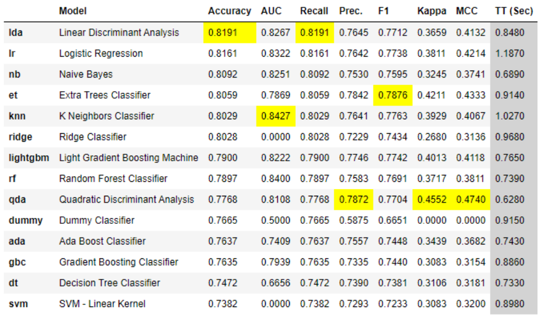
VI. CLASSIFICATION RESULTS

In this section, we discuss the performance of four commonly used classification algorithms on the Boston Housing dataset. To ensure reproducibility, we split the original dataset into 70% training and 30% testing data and set the session id to 123. We used the PyCaret library in Python to create a performance comparison table among all available classification algorithms and find the best model with the highest accuracy. It can be seen in Fig. 8 prior to applying PCA, the K Neighbors Classifier (KNN), Random Forest Classifier (rf), and Extra Trees Classifier (et) achieved the highest accuracies. Based on Fig. 9 after applying PCA, the Linear Discriminant Analysis, Logistic Regression, and Naive Bayes algorithms had the highest accuracy on the transformed dataset. Therefore, the LDA was used for evaluation purposes in the rest of the experiment. The original and transformed datasets were trained, tuned, and evaluated using this algorithm. The tuning of hyper-parameters plays a crucial role in improving the performance of a model. PyCaret's tune_model() function automatically tunes the model with effective hyper-parameters on a pre-defined search space and scores it using stratified K-fold cross-validation. By default, PyCaret applies 10-fold stratified K-fold validation on the three algorithms. The experiments and results can be found on GitHub.



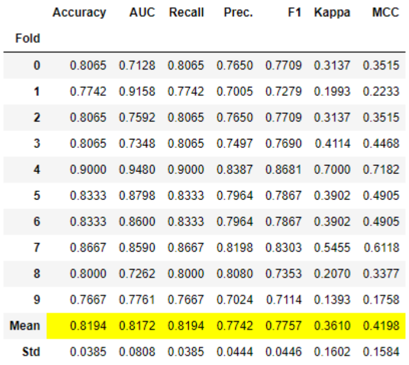
	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
knn	K Neighbors Classifier	0.8390	0.8481	0.8390	0.8092	0.8169	0.5073	0.5289	1.1200
rf	Random Forest Classifier	0.8245	0.8702	0.8245	0.8018	0.8068	0.5115	0.5216	1.1480
et	Extra Trees Classifier	0.8243	0.8255	0.8243	0.8184	0.8146	0.5233	0.5297	1.0860
gbc	Gradient Boosting Classifier	0.8210	0.8250	0.8210	0.8057	0.8048	0.4923	0.5034	1.0430
lightgbm	Light Gradient Boosting Machine	0.8175	0.8554	0.8175	0.7965	0.8011	0.4833	0.4937	1.1920
lr	Logistic Regression	0.8143	0.8257	0.8143	0.7726	0.7884	0.4528	0.4697	1.8380
lda	Linear Discriminant Analysis	0.8034	0.8166	0.8034	0.7655	0.7703	0.3830	0.4202	0.9290
ridge	Ridge Classifier	0.8030	0.0000	0.8030	0.7701	0.7519	0.2992	0.3800	0.8200
dt	Decision Tree Classifier	0.7770	0.7292	0.7770	0.7637	0.7662	0.4018	0.4064	1.1500
dummy	Dummy Classifier	0.7593	0.5000	0.7593	0.5767	0.6554	0.0000	0.0000	1.0370
ada	Ada Boost Classifier	0.7155	0.7361	0.7155	0.7492	0.7209	0.3434	0.3558	1.2470
svm	SVM - Linear Kernel	0.7094	0.0000	0.7094	0.5614	0.6157	0.0650	0.0784	0.7860
qda	Quadratic Discriminant Analysis	0.6787	0.8101	0.6787	0.7791	0.7078	0.3512	0.3771	0.9020
nb	Naive Bayes	0.6286	0.7545	0.6286	0.7416	0.6419	0.2791	0.3241	1.1480

Fig. 8. Comparison among classification models before applying PCA



	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
lda	Linear Discriminant Analysis	0.8191	0.8267	0.8191	0.7645	0.7712	0.3659	0.4132	0.8480
lr	Logistic Regression	0.8161	0.8322	0.8161	0.7642	0.7738	0.3811	0.4214	1.1870
nb	Naive Bayes	0.8092	0.8251	0.8092	0.7530	0.7595	0.3245	0.3741	0.6890
et	Extra Trees Classifier	0.8059	0.7869	0.8059	0.7842	0.7876	0.4211	0.4333	0.9140
knn	K Neighbors Classifier	0.8029	0.8427	0.8029	0.7641	0.7763	0.3929	0.4067	1.0270
ridge	Ridge Classifier	0.8028	0.0000	0.8028	0.7229	0.7434	0.2680	0.3136	0.9680
lightgbm	Light Gradient Boosting Machine	0.7900	0.8222	0.7900	0.7746	0.7742	0.4013	0.4118	0.7650
rf	Random Forest Classifier	0.7897	0.8400	0.7897	0.7583	0.7691	0.3717	0.3811	0.7390
qda	Quadratic Discriminant Analysis	0.7768	0.8108	0.7768	0.7872	0.7704	0.4552	0.4740	0.6280
dummy	Dummy Classifier	0.7665	0.5000	0.7665	0.5875	0.6651	0.0000	0.0000	0.9150
ada	Ada Boost Classifier	0.7637	0.7409	0.7637	0.7557	0.7448	0.3439	0.3682	0.7430
gbc	Gradient Boosting Classifier	0.7635	0.7939	0.7635	0.7335	0.7440	0.3083	0.3154	0.8860
dt	Decision Tree Classifier	0.7472	0.6656	0.7472	0.7390	0.7381	0.3106	0.3181	0.7330
svm	SVM - Linear Kernel	0.7382	0.0000	0.7382	0.7293	0.7233	0.3083	0.3200	0.8980

Fig. 9. Comparison among classification models after applying PCA



	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
Fold							
0	0.8065	0.7128	0.8065	0.7650	0.7709	0.3137	0.3515
1	0.7742	0.9158	0.7742	0.7005	0.7279	0.1993	0.2233
2	0.8065	0.7592	0.8065	0.7650	0.7709	0.3137	0.3515
3	0.8065	0.7348	0.8065	0.7497	0.7690	0.4114	0.4468
4	0.9000	0.9480	0.9000	0.8387	0.8681	0.7000	0.7182
5	0.8333	0.8798	0.8333	0.7964	0.7867	0.3902	0.4905
6	0.8333	0.8600	0.8333	0.7964	0.7867	0.3902	0.4905
7	0.8667	0.8590	0.8667	0.8198	0.8303	0.5455	0.6118
8	0.8000	0.7262	0.8000	0.8080	0.7353	0.2070	0.3377
9	0.7667	0.7761	0.7667	0.7024	0.7114	0.1393	0.1758
Mean	0.8194	0.8172	0.8194	0.7742	0.7757	0.3610	0.4198
Std	0.0385	0.0808	0.0385	0.0444	0.0446	0.1602	0.1584

Fig. 10. LDA metrics score after hyperparameter tuning

Fig. 10 illustrates the metrics scores for the LDA algorithm. The metrics include accuracy, precision, recall, and F1-score. Accuracy is the proportion of correctly classified instances out of all the instances. It is defined as:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{11}$$

where TP is the number of true positives (correctly predicted positives), TN is the number of true negatives (correctly predicted negatives), FP is the number of false positives (incorrectly predicted positives), and FN is the number of false negatives (incorrectly predicted negatives).

Precision is the proportion of true positives out of all predicted positives. It is defined as:

$$Precision = \frac{TP}{TP + FP} \tag{12}$$

Recall is the proportion of true positives out of all actual positives. It is defined as:

$$Recall = \frac{TP}{TP + FN} \tag{13}$$

F1-score is the harmonic mean of precision and recall. It is defined as:

$$F1 - score = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{14}$$

High accuracy means that the model is able to correctly classify a large proportion of the instances. High precision means that the model is able to accurately predict the positive class. High recall means that the model is able to correctly identify a large proportion of the actual positives. A high F1-score means that the model has a good balance between precision and recall.

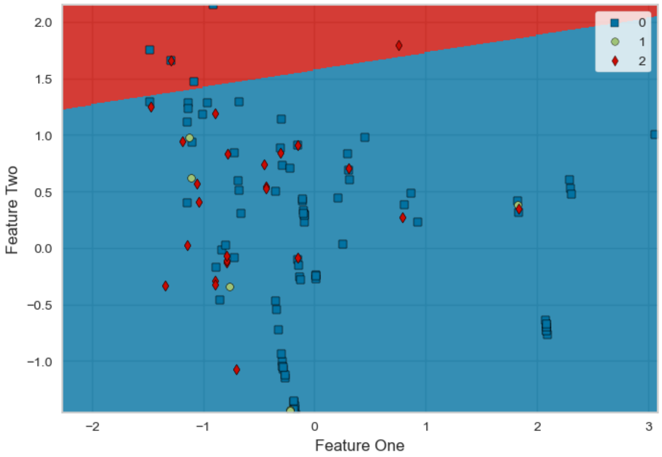


Fig. 11. Decision Boundaries of the LDA algorithm applied on transformed dataset

Figure 11 displays the decision boundaries created by the model after transforming the dataset. The decision boundary is a plane that divides the data points into separate classes, and the algorithm switches from one class to another. The first principal component (PC) is represented on the x-axis, and the second principal component is represented on the y-axis. The square-shaped dots in the figure represent the observations for class A, while class M is represented by the round-shaped dots. The diamond-shaped points represent class H.

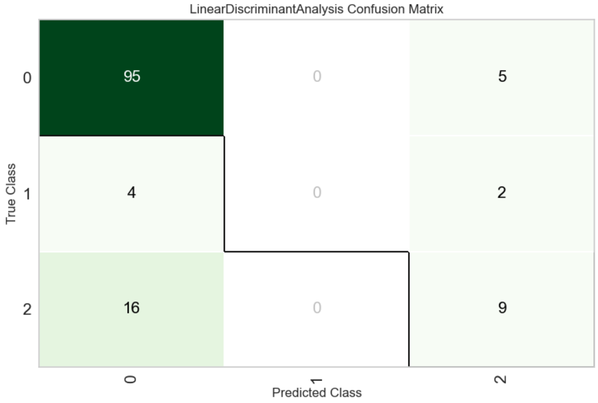


Fig. 12. Confusion matrices of the LDA classification algorithm applied on transformed dataset

Fig. 12 provides a visual representation of the confusion matrix for the LDA model on the transformed Boston housing dataset. The matrix is a square table that has the true class labels on the vertical axis and the predicted class labels on the horizontal axis. The matrix is divided into four quadrants representing the four possible outcomes: true positive (TP), false positive (FP), true negative (TN), and false negative (FN).

The values in the cells of the confusion matrix represent the number of instances that fall into each of the four possible outcomes. The diagonal cells of the matrix correspond to the number of correctly classified instances (TP and TN), while the off-diagonal cells correspond to the number of misclassified instances (FP and FN). The overall accuracy of the LDA model can be calculated from the values in the confusion matrix by dividing the sum of the diagonal values by the total number of observations.

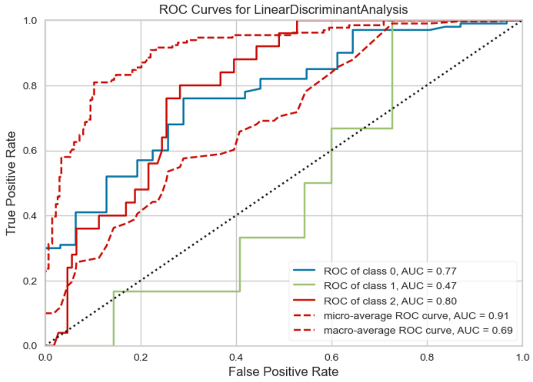


Fig. 13. ROC curve LDA

The final analysis step for the LDA algorithm involves the creation of a receiver operating characteristic (ROC) curve, which is shown in Figure 13. A ROC curve is a graphical representation of the performance of a classification model at all possible classification thresholds. This curve plots two parameters, the True Positive Rate (TPR) and the False Positive Rate (FPR), which are calculated based on the number of true positive and false positive classifications made by the model. The ROC curve and the confusion matrix are closely related and can be considered as different visual representations of the same measurement.

VII. Explainable AI with Shapely Values

In the field of machine learning, it is important to have models that are interpretable, meaning that it is possible to understand how the model is making its predictions. One way to increase interpretability is to analyze feature importance, which helps to determine how much each feature contributes to the model's predictions. To do this, we can use SHAP (Shapley Additive Explanations), a method that explains individual predictions by calculating the contribution of each feature using game theory. In SHAP, each feature acts as a player in a coalition, and Shapley values are used to distribute the prediction among the features.

The open-source "shap" library of Python can be used to implement SHAP. However, the library only supports tree-based models for binary classification problems.

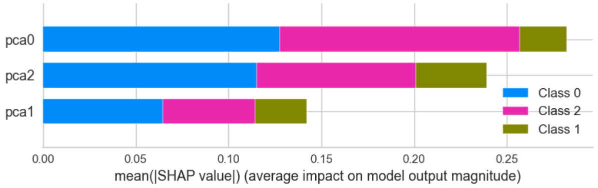


Fig. 14. Summary plot

Fig. 14 shows the summary plot of SHAP values, which combines feature importance with feature effects. The y-axis represents the PCs, and Shapley values are positioned on the x-axis. Component_1 represents the first PC, component_2 represents the second PC, and component_3 represents the third PC. The PCs are ordered according to their importance. The red color indicates high PC value and blue color indicates low PC value. We can interpret the summary plot to mean that a low level of PC value has a high and positive impact on Boston Housing price prediction, while a high level of PC value has a low and negative impact on Boston Housing price prediction. PCs are negatively correlated with the target variable. Fig. 15 shows the force plot for a single observation, which displays the features that contribute to the model's output. The base value is the mean prediction of the test set, and the bold 0.00 is the model's score for this observation.

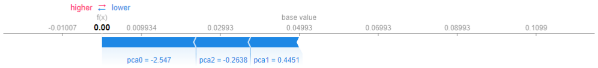


Fig. 15. Force plot for a single observation

Fig. 16 shows the combined force plot of all PCs, which displays the influence of each PC on the current prediction. Values in the blue color are considered to have a positive influence on the prediction, while values in the red color have a negative influence.

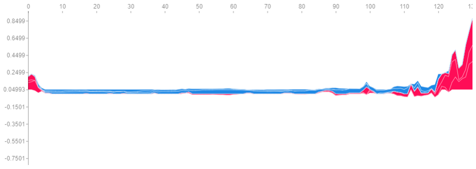


Fig. 16. Combined force plot

VIII. Conclusion

To summarize, the Boston Housing dataset was used to apply PCA and four popular classification algorithms to identify the price. PCA was first applied to the dataset, resulting in the reduction of the feature set from 5 to 2, and experiments were conducted on the first two principal components. LR, K-NN, and Dt algorithms were then applied to the original dataset and LDA was applied to the transformed dataset. The KNN, RF, and ET, algorithms performed best on the original dataset, while LDA, LR, and NB performed the highest after applying PCA. Finally, several interpretation plots using explainable AI shapley values were produced to increase the interpretability of the model. Overall, the algorithm was successful in determining the price of Boston houses.

References

[1] hossain2020predicting
[2] kang2021prediction
[3] sun2020housing
[4] harrison1978hedonic
[5] Abdi, Hervé, and Lynne J. Williams. "Principal component analysis." Wiley interdisciplinary reviews: computational statistics 2.4 (2010): 433-459.
[6] Jolliffe, Ian T. Principal component analysis. John Wiley and Sons, 2005.
[7] A. Dasgupta, "Principal Component Analysis," in Fundamentals of Data Analytics, Springer, 2019, pp. 47-62.
[8] J. Abdi, "Principal Component Analysis," in Encyclopedia of Measurement and Statistics, N. Salkind, Ed. Sage, 2007, pp. 1–9.
[9] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, "Classification and Regression Trees", Wadsworth International Group, 1984.
[10] T. Hastie, R. Tibshirani, and J. Friedman, "The Elements of Statistical Learning: Data Mining, Inference, and Prediction", Springer, 2009.
[11] Hosmer Jr, D. W., Lemeshow, S., and Sturdivant, R. X. (2013). Applied logistic regression. John Wiley and Sons.