Principal Component Analysis and Car Price Prediction Using Machine Learning

Nasim Fani 40254291
https://github.com/nasim-fani/6220-final

***Abstract*—The Principal Component Analysis (PCA) is a widely used method to reduce the dimensionality of large datasets while maintaining most of the relevant information. By transforming the original set of correlated variables into a smaller set of uncorrelated variables, known as principal components, PCA allows for easier analysis of complex datasets. In this report, we apply PCA to an automobile price prediction dataset to classify cars into three groups based on their price. We use four different classification algorithms - logistic regression, k-nearest neighbour, decision tree, and random forest - on the original dataset and the transformed dataset (after PCA) to identify the pricing class of the cars. We then tune each model with ideal hyperparameters to improve performance and measure each algorithm's performance using F1 score, confusion matrix, and ROC curves. We also show the decision boundaries for each model to demonstrate their fit on the dataset. Our analysis shows that Decision Tree Classifier outperforms the other available machine learning models. Finally, we conduct an experiment to interpret the model using Shapley values, an explainable AI technique, using random forest classifier model. Overall, our algorithms successfully classify the cars into their appropriate price classes, achieving an F1 score of nearly 1.**

***Index Terms*—Principal component analysis, binary classification, logistic regression, K-nearest neighbour, decision tree**

I. Introduction

The automobile industry is one of the most important and rapidly growing industries in the world, with the global automotive market expected to reach over $90 million by 2026. [1] In this industry, pricing is a critical factor for both manufacturers and consumers, as it directly impacts the profitability of manufacturers and the affordability of vehicles for consumers. Accurately predicting automobile prices is therefore an important problem to solve.

Machine learning has emerged as a powerful tool for solving a variety of problems, including automobile price detection. One popular technique used in machine learning is Principal Component Analysis (PCA), which can be used to reduce the dimensionality of large datasets while retaining most of the information. By applying PCA to automobile price datasets, it is possible to identify the most important factors that affect automobile prices and use this information to accurately predict prices.

In this report, we explore the use of PCA in automobile price detection using machine learning. We apply PCA to the Automobile dataset and compare the performance of different machine learning algorithms, including logistic regression, k-nearest neighbor, decision tree, and random forest, on the original and transformed datasets. We tune the hyperparameters of each model to obtain better performance metrics, such as F1 score, confusion matrix, and receiver operating characteristic (ROC) curves. Additionally, we show the decision boundaries for each model to visualize the fitting of the models on the dataset.

Overall, our report aims to demonstrate the usefulness of PCA and machine learning algorithms in accurately predicting automobile prices, thereby providing valuable insights to manufacturers and consumers in the automobile industry.

In this report, at first Principle component analysis (PCA) is applied to the Automobile dataset with the aim to dimensionality reduction. Afterward, four popular classification algorithms, Logistic Regression (LR), K-nearest neighbor (K-NN), Decision Tree (DT), and Random Forest are applied to the original dataset and PCA transformed dataset. The purpose is to determine whether a car is low-priced, medium-priced, or high-priced. Finally, to interpret the classification models using explainable AI, Shapley values are utilized. It is important to note that the classification results presented in this report were obtained using the transformed dataset after applying PCA. The results of the original dataset can be found in the Google Colab notebook.

The rest of the report is structured in the following manner: Section II elaborates on the methodology of PCA. Section III gives an outline of the three classification algorithms. Section IV presents the description of the automobile dataset. Section V discusses the results obtained after applying PCA. Section VI provides a comprehensive analysis of the classification results. Section VII talks about the discussion on the use of explainable AI Shapley values. Lastly, in Section VIII, a conclusion is drawn.

II. Principal Component Analysis

PCA is a mathematical technique used to simplify complex data by reducing the number of variables without losing too much information. [2] It works by identifying patterns and correlations between variables and then combining them to form new variables called principal components. These principal components represent the most important features of the original data and are ranked by their ability to explain the variability in the data. By using PCA, we can reduce the dimensionality of large datasets while retaining the most important information, which can help us to visualize and understand the data better, as well as improve the performance of machine learning algorithms.

Identify applicable funding agency here. If none, delete this.

A. PCA algorithm

PCA is an algorithm that follows a set of steps to transform a dataset into a set of principal components. The general steps of the PCA algorithm are as follows: [3]

1) **Standardize the data:** This involves centering the data by subtracting the mean of each column from each observation

$$Y = x - \bar{x}, \tag{1}$$

and the $\bar{x}$ is the mean of each column which can be calculated using this formula:

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i. \tag{2}$$

So, the standard data can be expressed as:

$$Y = HX, \tag{3}$$

2) **Calculate the covariance matrix:** This involves calculating the covariance between each pair of variables in the standardized data. The aim of this step is to understand how the variables of the input data set are varying from the mean with respect to each other, or in other words, to see if there is any relationship between them.

$$\text{cov}(X) = \frac{1}{n-1}Y^T Y \tag{4}$$

3) **Compute the eigenvectors and eigenvalues of the covariance matrix:** The eigenvectors are the directions in which the data varies the most, while the eigenvalues represent the magnitude of the variation in each direction. Let A be a square matrix (in our case the covariance matrix), υ a vector, and λ a scalar that satisfies $A\nu = \lambda\nu$, then λ is called the eigenvalue associated with eigenvector υ of A.

$$A\nu - \lambda\nu = 0; \quad (A - \lambda I)\nu = 0 \tag{5}$$

Since we already know υ is a non-zero vector, the only way this equation can be equal to zero is if:

$$\det(A - \lambda I) = 0 \tag{6}$$

Eigen decomposition can be computed using the following equation:

$$S = A\Lambda A^T \tag{7}$$

4) **Select the principal components:** The principal components are the eigenvectors that correspond to the largest eigenvalues. The number of principal components selected is typically determined by the amount of variance in the data that they explain.
5) **Transform the data:** The data is transformed into the new coordinate system defined by the principal components, where each observation is represented as a linear combination of the principal components.

$$Z = YA \tag{8}$$

III. Machine Learning-based Classification Algorithms

A. Logistic Regression Model

This type of statistical model is often used for classification and predictive analytics. Logistic Regression is used when the dependent variable(target) is categorical and transforms its output using the logistic sigmoid function to return a probability value which can then be mapped to two or more discrete classes. In machine learning, we use sigmoid to map predictions to probabilities: [4]

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{9}$$

The current prediction function returns a probability score between 0 and 1. In order to map this to a discrete class, we select a threshold value or tipping point above which we will classify values into class 1, and below which we classify values into class 2.

$$\begin{aligned} p &\geqslant 0.5, class = 1 \\ p &< 0.5, class = 0 \end{aligned} \tag{10}$$

For example, if our threshold was .5 and our prediction function returned .7, we would classify this observation as positive.

B. K Nearest Neighbors (KNN)

The k-nearest neighbors algorithm, also known as KNN or k-NN, is a non-parametric, supervised learning classifier, which uses proximity to make classifications or predictions about the grouping of an individual data point. [5] In KNN classification, the algorithm first stores all of the available labeled data as its training set. When it is given a new, unlabeled input data point, it calculates the distance between the new point and all of the training data points. It then selects the k closest data points (i.e., k nearest neighbors) to the new point based on their distances.

Once the k nearest neighbors are identified, KNN assigns the new data point to the class that is most common among the k nearest neighbors. In other words, it chooses the majority class of the k nearest neighbors to be the class of the new data point.

The choice of the parameter k, which represents the number of neighbors to consider, is critical to the performance of the KNN algorithm. A small k value will make the model more sensitive to noise, while a large k value may result in the misclassification of the minority class. The optimal k value depends on the specific problem and data set being analyzed and is usually determined through cross-validation.

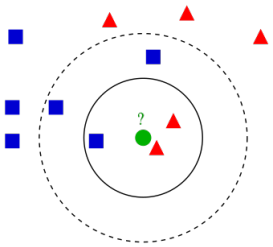


Fig. 1. Example of k-NN classification. The test sample (green dot) should be classified either to blue squares or to red triangles. If k = 3 (solid line circle) it is assigned to the red triangles because there are 2 triangles and only 1 square inside the inner circle. If k = 5 (dashed line circle) it is assigned to the blue squares (3 squares vs. 2 triangles inside the outer circle). [6]

C. Decision Trees (DT)

Decision Tree is a Supervised Machine Learning Algorithm that uses a set of rules to make decisions, similar to how humans make decisions. The goal of using a Decision Tree is to create a training model that can use to predict the class or value of the target variable by learning simple decision rules inferred from prior data(training data).

In Decision Trees, for predicting a class label for a record we start from the root of the tree. We compare the values of the root attribute with the record's attribute. On the basis of comparison, we follow the branch corresponding to that value and jump to the next node.

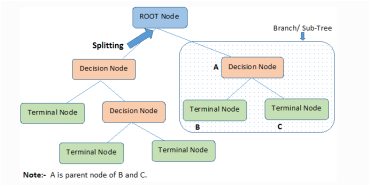


Fig. 2. Terminology related to Decision Trees

IV. Data set description

The automobile data set is obtained from UCI and it is an abstract from 1985 Ward's Automotive Yearbook. The original data set consists of 26 attributes(columns) and it includes 205 entries for each of these attributes. The features describe the car's characteristics. In this project, for the car price prediction purpose, five columns out of 26 have been selected, including highway-mpg, engine-size, horsepower, curb-weight, and bore. Also, the price attribute is the label. The price attribute has been categorized into three sections:

- Low-priced; below $18500
- Medium-priced; between $18500 and $32000
- High-priced; higher than $32000

There are also 13 duplicate rows and some of the instances contain Nan values which are indicated by the "?" sign.

The distribution, central values, and variability of the features in the Automobile dataset were measured using box and whisker plots and their corresponding five-number summaries. The resulting figure 3 displays the box plot of the features.

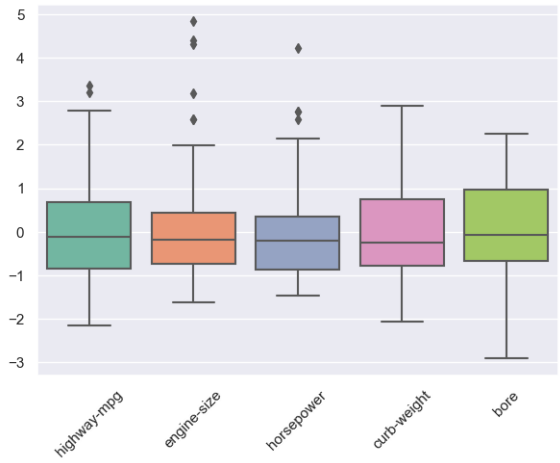


Fig. 3. Box-plot

It can be observed from Fig. 3 that most of the features follow an approximately normal distribution. However, outliers exist in all features except for curb-wight and bore and the outliers are on the left.

Fig. 4. Correlation Matrix

The correlation matrix (Fig. 4) shows the pairwise correlations between features. Looking at the off-diagonal elements, we see that there is a strong negative correlation between highway-mpg and horsepower, meaning that cars with higher horsepower tend to have lower fuel efficiency on the highway. There is also a strong positive correlation between engine-size and horsepower. Also, Curb-weight is positively correlated with engine-size and horsepower. Finally, there is a weak positive correlation between bore and horsepower (0.56), which suggests that cars with larger cylinder bores may have more horsepower. The pairplot in Fig. 5 confirms this observation, showing that highly correlated features have a larger number of cells with consistently increasing lines.

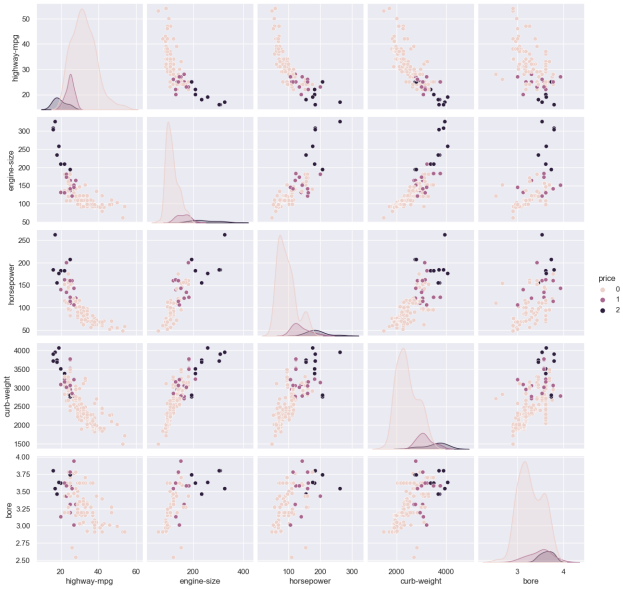


Fig. 5. Pair Plot

V. PCA Results

PCA can be implemented without using a library by using basic linear algebra operations as the general steps were explained in section II. Also, there are several Python libraries that provide PCA implementation, including scikit-learn, TensorFlow, Pandas, etc. Each library has its own advantages and disadvantages, depending on the specific requirements of the task at hand. Using a library can save time and effort in implementing PCA, while providing robust and efficient results. So, in this project, sklearn library has been used. By performing the necessary steps of PCA, it is possible to reduce the original feature set of 5 variables to a smaller set of r variables where r is less than 5. This is accomplished through the use of an eigenvector matrix A, which is capable of reducing the n × p dataset. Each of the eigenvectors in matrix A is associated with a principal component (PC) that is able to capture a specific amount of data and determine the dimension (r) of the dataset. The resulting eigenvector matrix for the Automobile dataset is provided below.

$$A = \begin{bmatrix} -0.456 & -0.124 & 0.770 & 0.129 & -0.405 \\ 0.462 & 0.258 & 0.619 & 0.0111 & 0.579 \\ 0.461 & 0.292 & -0.0415 & 0.693 & -0.467 \\ 0.470 & 0.068 & 0.127 & -0.690 & -0.528 \\ 0.377 & -0.909 & 0.066 & 0.156 & 0.029 \end{bmatrix} \tag{11}$$

and the corresponding eigenvalues are:

$$\lambda = \begin{bmatrix} 3.915 \\ 0.531 \\ 0.273 \\ 0.223 \\ 0.083 \end{bmatrix} \tag{12}$$

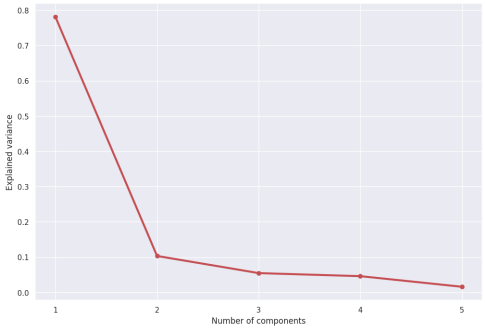


Fig. 6. PCA Explained Variance

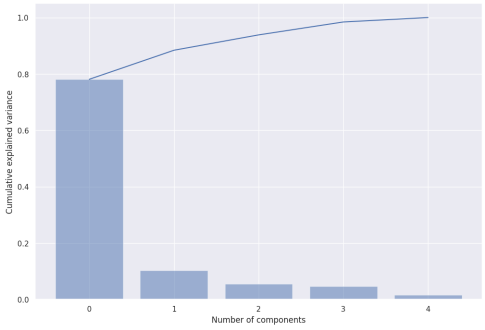


Fig. 7. PCA Cumulative Explained Variance

Figure 6 and Figure 7 illustrate the scree plot and pareto plot for the principal components. These plots depict the amount of variance explained by each principal component. We can determine the percentage of variance explained by the j_{th} principal component using the following equation:

$$explained - variance_j = \frac{\lambda_j}{\sum_j^p \lambda_j} \times 100, j = 0, 1, 2, ... \tag{13}$$

λ_j represents the eigenvalue and the amount of variance of the j^{th} PC.

Both figures show that the combined variance of the first two principal components accounts for 88.5% of the variance in the original dataset. Specifically, the first principal component holds 77.9% of the variance 77.9% and the second principal component holds 10.6% of the variance 10.6%. The scree plot reveals that the elbow point is located on the second principal component. These two observations suggest that the dimensionality of the feature set can be reduced to two ($r = 2$). The first principal component Z1 is given by:

$$\begin{aligned} Z_1 = -0.456X_1 + 0.462X_2 + 0.461X_3 \\ +0.470X_4 + 0.377X_5 \end{aligned} \tag{14}$$

It can be observed that X_1, X_2, X_3, and X_4 contribute the most in the first PC. However, none of the features have a negligible contribution to the first PC.

The second principal component Z2 is given by:

$$\begin{aligned} Z_2 = -0.124X_1 + 0.258X_2 + 0.292X_3 \\ +0.068X_4 - 0.909X_5 \end{aligned} \tag{15}$$

From the second PC, it can be seen that X_5 has the highest contribution in the second PC. The contributions for X_4 is very small and hence negligible. Therefore, Z2 can be rewritten as follows:

$$Z_2 = -0.124X_1 + 0.258X_2 + 0.292X_3 - 0.909X_5 \quad (16)$$

Fig. 8 displays the PC coefficient plot, which visually represents the level of contribution that each feature has on the first two PCs. This figure confirms the earlier calculation of PCs and clearly illustrates that"engine-size", "curb-weight", and "horsepower" have a high contribution to the first principal component, as they have loadings that are located closer to the positive end of the x-axis. On the other hand, "highway-mpg" and "bore" have relatively low contributions to the first principal component, as their loadings are located closer to the negative end of the x-axis. For the second principal component, it appears that "highway-mpg" and "bore" have higher contributions, as their loadings are located closer to the positive end of the y-axis. "Engine-size" and "curb-weight" have a lower contribution to the second principal component, as their loadings are located closer to the negative end of the y-axis. "Horsepower" has a moderate contribution to the second principal component, as its loading is located somewhat in the middle of the y-axis. This Fig. shows that variables like "engine-size", "horsepower", and "curb-weight" are highly positively correlated with each other and are located near each other in the top right quadrant of the plot. "highway-mpg" is located in the top left quadrant of the plot, indicating that it is negatively correlated with the other variables.

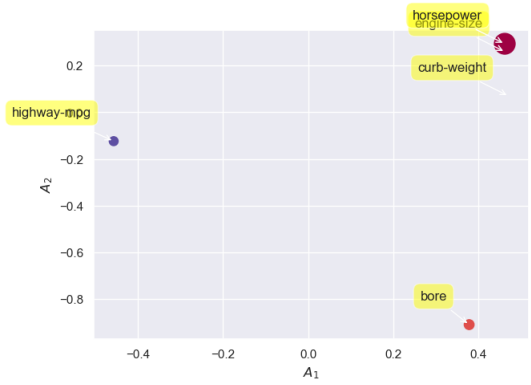


Fig. 8. Coefficient Plot

Fig. 9 shows a biplot that presents a different visual representation of the first two PCs.The vectors for each feature are displayed on the biplot, with the direction and length of the vector indicating the correlation and contribution of the feature to the PCs. In this biplot, the vectors for engine-size, horsepower, and curb-weight have a small angle with the first PC and a large angle with the second PC, indicating that these features contribute strongly to the first PC and less to the second PC. The vectors for highway-mpg and bore, on the other hand, have a larger angle with the first PC and a smaller angle with the second PC, indicating that they are more related to the second PC than the first. The biplot also shows that engine-size, horsepower, and curb-weight are positively correlated with each other, as they are facing in the same direction on the biplot, while highway-mpg and bore are negatively correlated with each other, as they are pointing in opposite directions.

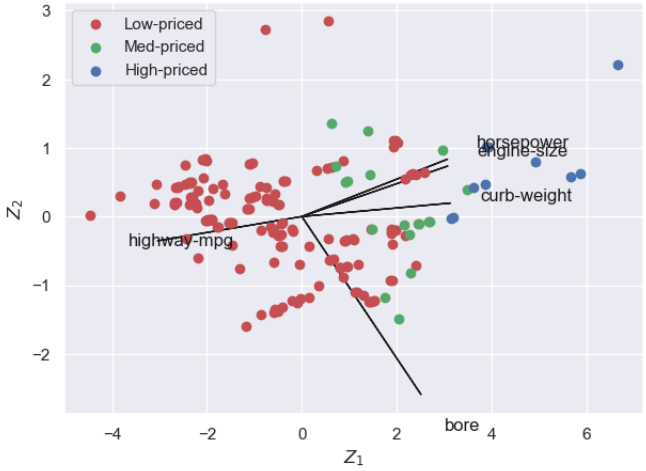


Fig. 9. Coefficient Plot

VI. Classification Results

This section presents a discussion on the performance of four commonly used classification algorithms (Linear Regression, K-NN, Decision Tree, Random Forest) on the Automobile dataset. To examine the impact of PCA on the dataset, the best model, Decision Tree is applied to both the original dataset and the PCA-transformed dataset with PCA components. The PyCaret library in Python is used to carry out the classification. The original dataset is divided into a seen set (70%) and a test set (30%). To ensure reproducibility, the session ID is set to 123.

By utilizing PyCaret, it is achievable to produce a table that compares the performance of all classification algorithms accessible for the specified dataset and to determine the model with the utmost accuracy. As displayed in Figure 10, prior to implementing PCA, Random Forest, Ada boost, and Logestic Regression are the top three classification models in terms of accuracy on the Automobile dataset. Figure 11 illustrates a comparison among the classification models after PCA has been applied. The top three models with the highest accuracy on the transformed dataset are shown to be Decision Tree, Extra Tree, and Gradient Boosting.

For the rest of this report, we will focus on the following models as our potential options. It is important to note that these choices were made for the sake of illustration and do not necessarily indicate that they are the best-performing or most suitable for this specific dataset.

- Decision Tree Classifier (dt)
- K Neighbors Classifier (knn)
- Logistic Regression (lr)

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
rf	Random Forest Classifier	0.9205	0.6708	0.7386	0.7075	0.7206	0.5648	0.5880	1.0260
ada	Ada Boost Classifier	0.9121	0.6289	0.7303	0.7140	0.7205	0.6098	0.6217	0.9430
lr	Logistic Regression	0.9023	0.6720	0.6295	0.6027	0.6109	0.4291	0.4503	2.2430
knn	K Neighbors Classifier	0.8947	0.5911	0.6220	0.5575	0.5869	0.3254	0.3492	1.0210
ridge	Ridge Classifier	0.8947	0.0000	0.6220	0.5575	0.5869	0.3254	0.3492	1.1440
gbc	Gradient Boosting Classifier	0.8947	0.6237	0.7220	0.6984	0.7079	0.5054	0.5236	1.0450
et	Extra Trees Classifier	0.8947	0.6524	0.7220	0.6984	0.7079	0.5054	0.5236	0.9550
dt	Decision Tree Classifier	0.8848	0.5759	0.7212	0.7058	0.7114	0.5090	0.5201	0.9110
lda	Linear Discriminant Analysis	0.8758	0.6654	0.7121	0.6498	0.6671	0.4527	0.4800	0.9700
lightgbm	Light Gradient Boosting Machine	0.8682	0.6630	0.6955	0.6951	0.6913	0.4667	0.4869	1.2200
dummy	Dummy Classifier	0.8515	0.3500	0.5788	0.4786	0.5239	0.0000	0.0000	0.8820
qda	Quadratic Discriminant Analysis	0.8417	0.6168	0.6871	0.6290	0.6431	0.3443	0.3659	0.8290
nb	Naive Bayes	0.8235	0.6795	0.6962	0.6568	0.6480	0.4384	0.4817	0.9970
svm	SVM - Linear Kernel	0.7015	0.0000	0.4288	0.3474	0.3786	0.0273	0.0296	0.9380

Fig. 10. Comparison among classification models before applying PCA

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
dt	Decision Tree Classifier	0.9115	0.6167	0.7115	0.6981	0.7027	0.6442	0.6659	2.2790
et	Extra Trees Classifier	0.9051	0.6612	0.7385	0.7500	0.7342	0.5979	0.6361	2.5050
gbc	Gradient Boosting Classifier	0.8968	0.6301	0.7385	0.7000	0.7175	0.5754	0.6086	2.5340
rf	Random Forest Classifier	0.8885	0.6545	0.6885	0.6500	0.6675	0.4934	0.5292	2.4390
lightgbm	Light Gradient Boosting Machine	0.8885	0.6641	0.6885	0.6532	0.6686	0.5308	0.5593	2.2320
knn	K Neighbors Classifier	0.8821	0.6808	0.7154	0.7544	0.7237	0.5482	0.5774	2.3230
lr	Logistic Regression	0.8808	0.6526	0.6308	0.6026	0.6161	0.4358	0.4584	2.5860
ridge	Ridge Classifier	0.8808	0.0000	0.6308	0.5695	0.5983	0.3194	0.3461	2.3210
svm	SVM - Linear Kernel	0.8801	0.0000	0.6885	0.6498	0.6665	0.4634	0.4886	2.3100
qda	Quadratic Discriminant Analysis	0.8506	0.6494	0.6923	0.7487	0.7078	0.4548	0.4877	2.4570
dummy	Dummy Classifier	0.8423	0.3500	0.5923	0.5012	0.5429	0.0000	0.0000	2.4420
nb	Naive Bayes	0.8340	0.6346	0.6423	0.6923	0.6544	0.3636	0.3906	2.3240
lda	Linear Discriminant Analysis	0.8340	0.6532	0.6423	0.6819	0.6523	0.3546	0.3718	2.1690
ada	Ada Boost Classifier	0.7308	0.5859	0.6308	0.6387	0.6068	0.3151	0.3684	2.1860

Fig. 11. Comparison among classification models after applying PCA

- Random Forest (rf)

These four candidate models were trained, tuned and evaluated on both the original and transformed datasets. However, for the purpose of this report, only the results obtained after applying PCA (i.e., transformed dataset) will be discussed. The experiments on both datasets can be found in the Google Colab notebook. Hyperparameter tuning is an essential process for improving the performance of a model. PyCaret provides a three-step process for hyperparameter tuning, which involves creating a model, tuning it, and evaluating its performance. Firstly, a classification model is created for each algorithm. Then, the tune_model() function is utilized to fine-tune the model with optimal hyperparameters. This function performs an automatic search for the best hyperparameters within a predefined search space and evaluates the model using stratified K-fold cross-validation. the output of this function is a scoring grid with CV scores by fold of the best-selected model based on optimized parameters. [7]By default, PyCaret applies 10-fold stratified K-fold validation for the three algorithms. It can be observed from the Fig. 12 that tuned KNN model metrics are better than the base model metrics (before hyperparameter tuning).

Fig. 13 illustrates the ROC curve for the four classification methods mentioned above. The ROC curve is a commonly used method to evaluate the performance of a binary classification model. [8] It plots the True Positive Rate (TPR) against the False Positive Rate (FPR) at various classification thresholds. The area under the ROC curve (AUC) is also used as a performance metric, where a value of 1 indicates a perfect classification and a value of 0.5 indicates random guessing.

Fold	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	0.9231	1.0000	0.9231	0.8846	0.8974	0.7174	0.7335
1	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
2	0.8462	0.7468	0.8462	0.7756	0.8094	0.2778	0.3010
3	0.9231	1.0000	0.9231	0.8526	0.8863	0.6389	0.6922
4	0.8462	0.9744	0.8462	0.7756	0.8094	0.2778	0.3010
5	0.8462	0.7468	0.8462	0.7756	0.8094	0.2778	0.3010
6	0.9231	0.7051	0.9231	0.8526	0.8863	0.6389	0.6922
7	0.9167	0.0000	0.0000	0.0000	0.0000	0.7143	0.7319
8	0.9167	0.0000	0.5000	1.0000	0.6667	0.6250	0.6742
9	0.8333	0.0000	0.5000	0.5000	0.5000	0.4000	0.4000
Mean	0.8974	0.6173	0.7308	0.7417	0.7265	0.5568	0.5827
Std	0.0502	0.4185	0.2939	0.2811	0.2754	0.2288	0.2287

Fig. 12. Tuned KNN Score Grid

In Fig 13(a), the AUC appears to be around 0.80, which suggests that the classifier is relatively good at distinguishing between positive and negative classes. The ROC curve also provides a way to determine the optimal classification threshold for the model. The point on the curve closest to the upper left corner represents the optimal trade-off between TPR and FPR. In this case, the optimal threshold seems to be around 0.4, where the TPR is about 0.75 and the FPR is about 0.25.

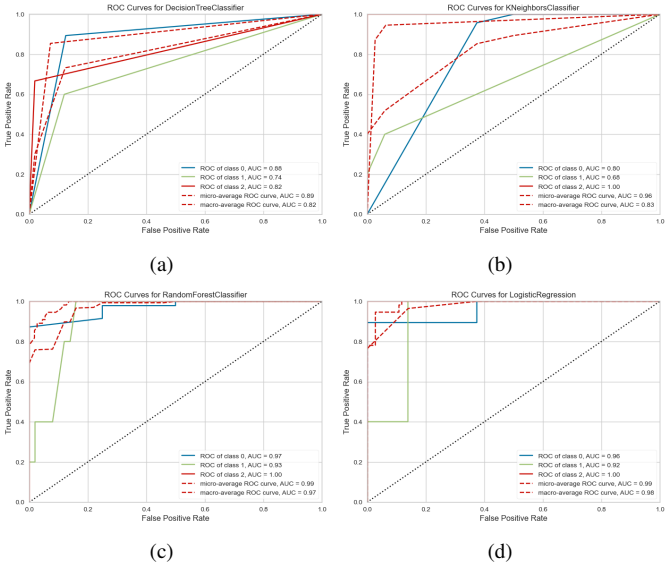
The (b) curve in this Fig is related to KNN classification and it shows that the model has good performance since it is close to the top left corner of the plot. The area under the curve (AUC) is also relatively high, which further indicates good performance. The AUC appears to be close to 0.9, which suggests that the model has good discriminative power.

In the (c) curve, the AUC is 0.85, which indicates that the classifier has a reasonably good ability to distinguish between positive and negative classes. The curve is sloping upwards toward the top-left corner of the plot, which is indicative of a good classifier. At an FPR of 0.2, the TPR is approximately 0.75, meaning that the classifier can correctly identify 75% of the positive cases while keeping the false positive rate at 20%. Overall, this ROC curve suggests that the classifier has a moderate to good performance, but further analysis is needed to determine the suitability of the model for a specific application.

Regarding the (d) curve, the model appears to have relatively high performance, as the curve is close to the top-left corner of the plot. The area under the curve (AUC) is also a commonly used metric for evaluating classification performance, with a higher AUC indicating better performance.

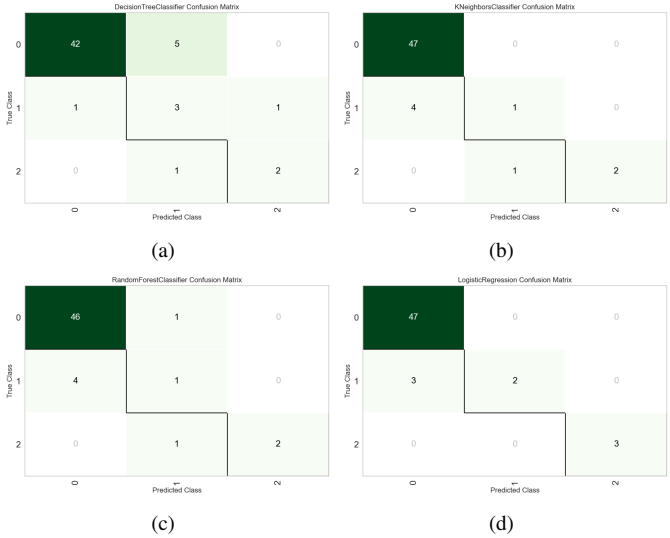
Fig. 14 a confusion matrix is a table that summarizes the performance of a binary classification model by comparing the predicted class labels against the true class labels. The matrix contains four elements: true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN). These elements can be used to calculate various performance metrics such as accuracy, precision, recall, and F1 score.

- Precision measures the proportion of positive predictions that are actually correct (true positives) out of all the



(a) (b) (c) (d)

Fig. 13. (a) ROC - Decision Tree (b) ROC - K-nearest Neighbour (c) ROC - Random Forest (d) ROC - Linear Regression



(a) (b) (c) (d)

Fig. 14. (a) Confusion Matrix - Decision Tree (b) Confusion Matrix - K-nearest Neighbour (c) Confusion Matrix - Random Forest (d) Confusion Matrix - Linear Regression

positive predictions (true positives and false positives).

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \tag{17}$$

- Recall is another metric that can be calculated from the confusion matrix. Recall measures the proportion of positive samples that the model has correctly identified (true positives) out of all the actual positive samples (true positives and false negatives).

$$Recall = \frac{TP}{TP + FN} \tag{18}$$

- Finally, we can also calculate the F1 score, which is the harmonic mean of precision and recall. The F1 score gives a single measure of the model's performance that balances precision and recall.

$$F_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \tag{19}$$

VII. Explainable AI with Shapely Values

Ensuring model interpretability is a crucial factor in the realm of machine learning. Various methods can be employed to enhance the explainability of a model, and one such method is feature importance. Feature importance enables the estimation of the significance of each feature in the prediction process. Therefore, to gain an understanding of the most significant features on the PCs, the SHAP (SHapley Additive exPlanations) values are utilized by importing the "shap" library, which is an open-source Python library.

SHAP (SHapley Additive exPlanations) is a popular library in Python used to explain the output of machine learning models. The library uses the Shapley value, a concept from cooperative game theory, to compute the contribution of each feature to the prediction of the model. SHAP offers a unified approach to interpret models, which works with different types of models, such as tree-based models, linear models, and neural networks, among others.

One of the main reasons to use the SHAP library is to gain insights into how the model is making its predictions. This is important because it allows us to understand the underlying patterns in the data that the model has learned, and to check whether these patterns align with our prior knowledge about the problem domain. Additionally, interpreting the model's output can help us identify cases where the model is making incorrect or biased predictions, which can be important for addressing ethical concerns and improving the model's performance. SHAP also provides several visualizations that help to understand the model's predictions. These visualizations can be used to explore the relationship between individual features and the model output, to identify interactions between features, and to compare the impact of different features on the model output. These visualizations make it easy to communicate the results of the analysis to non-experts and stakeholders, making it a useful tool for decision-making processes.

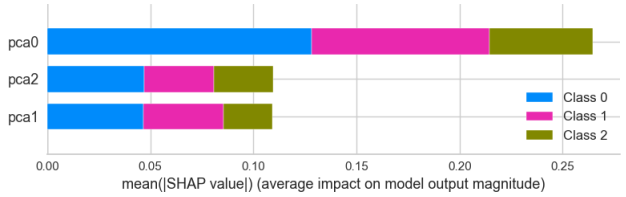


Fig. 15. SHAP Summary plot

Fig. 15 is SHAP summary plot for the tuned random Forest classifier. The SHAP summary plot provides an overview of the most significant features that influence the target variable. The x-axis represents the SHAP values, which indicate the ex-

tent to which a feature contributes to increasing or decreasing the target variable. The color of the dots represents the feature value, with red indicating high values and blue indicating low values. It is clear that PC_0 has the highest SHAP value while that of PC_1 and PC_2 are almost the same. This means that changes in PC_0 have the most substantial impact on the target variable.

Fig. 16. Force plot for a single observation

The force plot shown in Fig. 16 is specific to a single observation and illustrates how individual features contribute to the model output. It is important to note that this type of plot can only be generated for a single observation, not for multiple instances. The plot indicates which features are responsible for pushing the model output away from the base value, which is the predicted value in the absence of any known features for the given observation. In this case, the force plot is generated for a single observation, with a base value of 0.5. The bold 0.074 represents the model's score for this observation. If the score is higher, the model is more likely to predict 1, while if it is lower, the model is more likely to predict 0. the blue color indicates that it pushes the prediction to be lower.

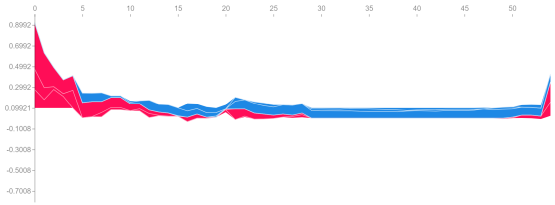


Fig. 17. Combined force plot of all PCs.

Figure 17 exhibits a collective force plot of all principal components (PCs). This plot is formed by stacking all individual force plots horizontally and rotating them by 90 degrees. The y-axis represents the x-axis of each individual force plot. The x-axis has 55 observations. The combined force plot illustrates the impact of each PC on the present prediction. Values in blue indicate a positive influence on the prediction, while values in red have a negative influence.

VIII. Conclusion

In this study, the Atumobile dataset was thoroughly analyzed using principal component analysis (PCA) and four classification algorithms, namely logistic regression (LR), K-nearest neighbor (K-NN), Desicion Tree (DT), and Random Forest (RF). The dataset holds information about car attributes that can be used to classify them into three categories; low-priced (0), medium-priced (1) and high-priced (2). To start the analysis, PCA was applied to the original dataset, and the first two principal components were able to capture 88.5% of the variance, resulting in a feature set reduction from 5 to 2. To validate the results from different perspectives, various experiments were conducted on the transformed data, and different plots were generated.

After tuning the hyperparameters of each algorithm, it was observed that the performance metrics of each algorithm had improved. The RF, AB, and LR algorithms performed best on the original dataset, while DT, ET, and GBC showed the best performance metrics after applying PCA. Moreover, to increase the interpretability of the model, several interpretation plots were generated using the shapley values.

In conclusion, the study successfully demonstrated that all four algorithms were effective in determining price classes for cars. The analysis showed that PCA could effectively reduce the feature set while maintaining the accuracy of the model. The generated plots provided insights into the individual features' influence on the prediction, enabling clinicians to interpret the results better. Overall, this study could potentially aid in improving the accuracy of predicting cars' price, thus leading to better treatment outcomes.

References

[1] www.abiresearch.com/blogs/2023/02/09/2023-automotive-industry/
[2] Jolliffe, I.T. (2002). Principal Component Analysis. Springer-Verlag New York.
[3] H. Abdi and L. J. Williams, "Principal component analysis," Wiley interdisciplinary reviews: computational statistics, vol. 2, no. 4, pp. 433–459, 2010.
[4] ml-cheatsheet.readthedocs.io/en/latest/logistic_regression.html
[5] en.wikipedia.org/wiki/K-nearest_neighbors_algorithm
[6] www.kdnuggets.com/2020/01/decision-tree-algorithm-explained.html
[7] pycaret.readthedocs.io/en/latest/api/classification.html
[8] Insights from a ROC Analysis", Mitchell et al., 1982, Biometrics journal