Statistical Learning project

Ettore Oteri - 34054A

DSE

Abstract

The following report aims to analyze the real estate market in the city of Milan, using a dataset containing variables that affect the price: number of rooms, square meters of living space, number of bathrooms, and condominium fees. The original dataset is continuously updated and available on Kaggle. It was created by web-scraping the house-announcements website Immobiliare using Beautiful Soup. It contains both cleaned and raw data. The actual goal of this report is to predict house prices using the available data. This topic has become quite popular in recent years due to the recent and dramatic rise in house prices and rents. Therefore, I found it interesting to conduct an analysis to understand which features of a house contribute to increasing its value, besides the square footage, of course.
After extensive research on the main websites that provide datasets and finding the right one, I decided to apply both unsupervised and supervised learning techniques to the data. The techniques used are PCA and decision trees, which will be discussed in more detail later.
In tandem with the analysis, I have included the *theoretical background (written in italics)* necessary to understand all the steps.
As an appendix, I have attached all the R code and the dataset.

Data Cleaning

The first necessary step to start the analysis is data cleaning. After installing various packages and loading the required libraries, I load the dataset into the software. I then begin with the cleaning process, ensuring that all values are numeric and that there are no missing values. The file originally contained 2,130 observations, which were reduced to 1,635 after data cleaning. I chose to omit the rows with missing values because, after trying a different approach—replacing the missing values with their respective means—I noticed a drop in performance of the decision tree prediction model, likely due to higher bias.

Unsupervised Learning

I can now finally start with the PCA.
Principal Component Analysis (PCA) is a statistical technique used to reduce the dimensionality of a dataset while retaining as much information as possible. By reducing dimensionality, it simplifies data analysis and reduces the risk of overfitting in predictive models. Each principal component is a linear combination of the original variables. The goal of PCA is to identify the components that explain most of the variance in the dataset.

After calculating the means and standard deviations for the variables…

```
> descriptive
                              M     sigma
price                636573.37 551340.08
rooms                     3.08      1.04
m2                      109.74     58.09
bathrooms                 1.58      0.68
condominium_expenses    233.97    159.43
```

…and after calculating the correlation between the variables (*correlation measures the degree of linear dependence between two variables*), I visualize the correlation matrix to get an initial sense of how the features relate to each other.

```
> round(rho,3)
                   price rooms    m2 bathrooms condominium_expenses
price              1.000 0.608 0.789     0.620                0.517
rooms              0.608 1.000 0.810     0.704                0.552
m2                 0.789 0.810 1.000     0.764                0.555
bathrooms          0.620 0.704 0.764     1.000                0.476
condominium_expenses 0.517 0.552 0.555     0.476                1.000
```

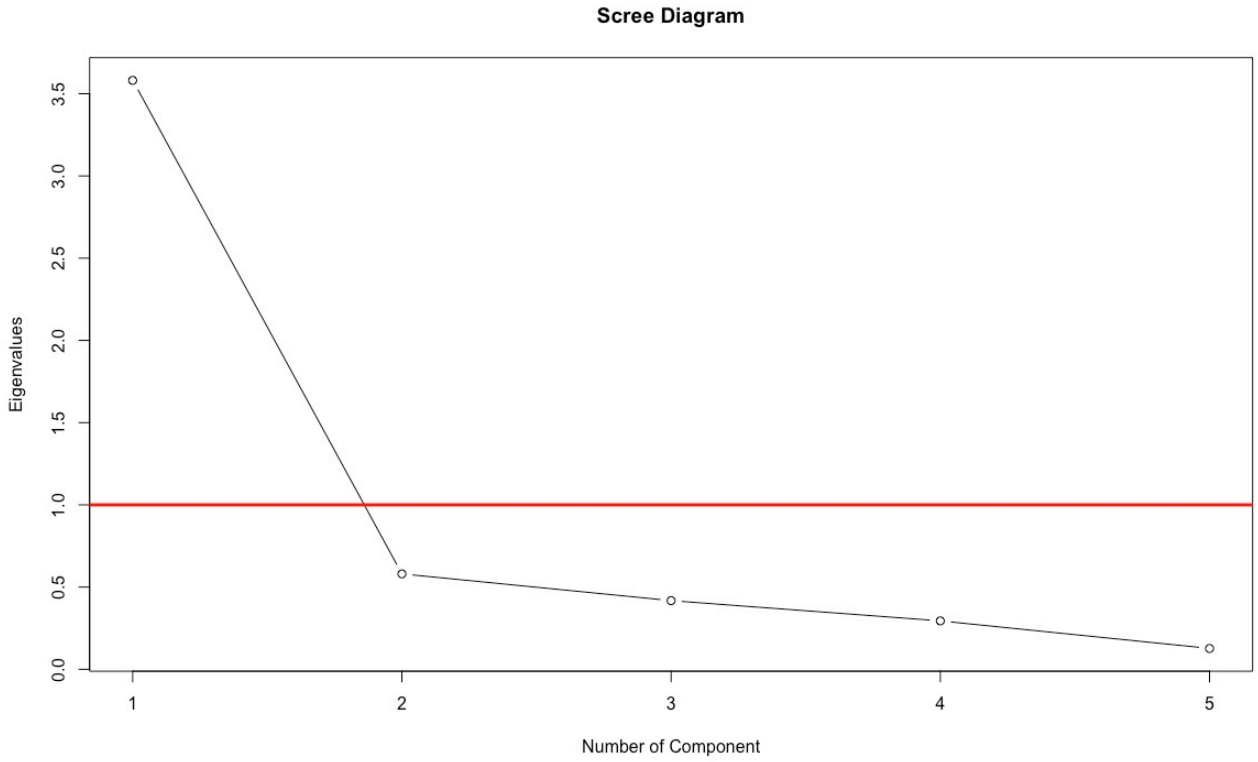
Next, I calculate the eigenvectors and eigenvalues derived from the rho matrix and the variance explained by each principal component. Finally, I visualize everything, along with the cumulative variance, in a table.

```
> tab
     eigenvelues % variance % cum variance
[1,]       3.581     71.621         71.621
[2,]       0.580     11.605         83.226
[3,]       0.418      8.352         91.578
[4,]       0.294      5.888         97.466
[5,]       0.127      2.534        100.000
```

Now I can move to the scree diagram.

The scree plot shows the distribution of eigenvalues associated with the principal components resulting from the PCA analysis. This graph is used to determine how many principal components to retain in order to adequately represent the data, balancing model simplification with information retention. The graph shows the eigenvalues in decreasing order with respect to the number of components, helping to identify the point where additional components no longer contribute significantly to the explained variance. This point is called the "elbow point."

We plot it:



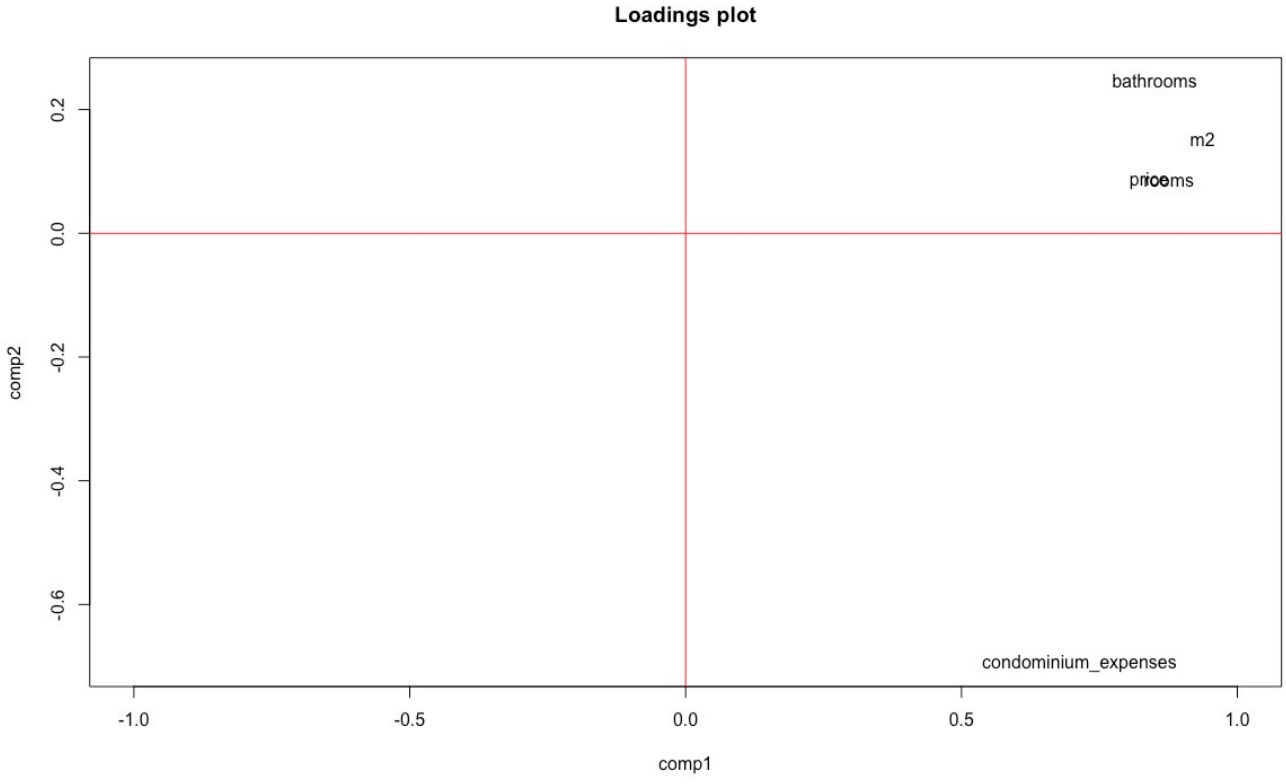
The y-axis shows the eigenvalues corresponding to each component. A high eigenvalue indicates that the component explains a significant portion of the variance in the data.
The x-axis indicates the component number. Each point represents a principal component (PC1, PC2, etc.).
The red horizontal line represents the eigenvalue threshold of 1.
The first principal component (PC1) has an eigenvalue greater than 3.5, which indicates that this component explains a significant portion of the variance in the dataset.
PC2 has an eigenvalue slightly higher than 1, meaning that it contributes meaningfully to the total variance, albeit to a lesser extent than the first component.
In contrast, PC3, PC4, and PC5 have eigenvalues lower than 1, which suggests that they do not add much variance compared to what each original variable individually explains.

The sharp initial drop in the line in the graph indicates that the first two components explain most of the variance. Afterward, the eigenvalues decrease drastically and flatten out, suggesting that subsequent components have a reduced contribution.
According to the scree plot, only PC1 and PC2 should be retained because they are the only ones that contribute significantly to the explanation of variance in the dataset, eliminating those that explain little and can be considered "noise."

I calculate the principal component coefficients (by multiplying the eigenvectors by the square root of their corresponding eigenvalues) and the communality, *i.e., the proportion of variance for each variable explained by the two principal components* (indicating how well the variables are described by the principal components).

```
> comp
                      Comp1  Comp2 communality
price                 0.840  0.085    0.712825
rooms                 0.876  0.083    0.774265
m2                    0.937  0.153    0.901378
bathrooms             0.850  0.246    0.783016
condominium_expenses 0.714 -0.695    0.992821
```

I visualize the loadings plot:

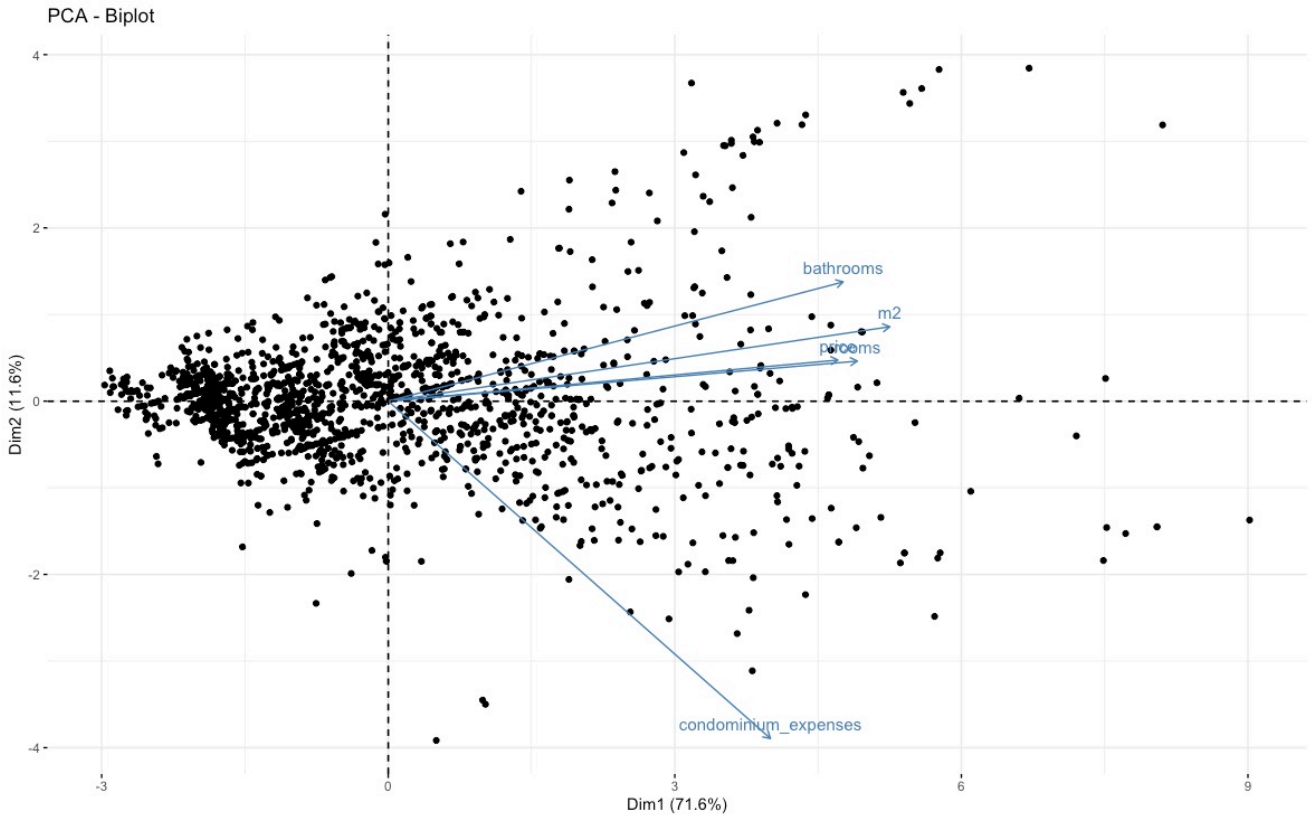


This graph represents the Loadings Plot of the first two principal components obtained from the PCA applied to the dataset I am analyzing. *This type of plot shows how each original variable contributes to the new principal components. The variables are represented by points in the two-dimensional space of the principal components, where:*

• the X-axis (comp1) represents the loadings of the first principal component (PC1),
• the Y-axis (comp2) represents the loadings of the second principal component (PC2).

The loadings determine the extent to which each original variable contributes to its respective principal component. Variables with a high loading on a component strongly influence that component.
The position of the variables in the plot indicates how much each one contributes to the two principal components.
The condominium expenses variable has a strong positive loading on the first principal component (PC1), with a value around +1. This means that the first principal component is strongly influenced by this variable.
The three variables—bathrooms, m2 (square meters), and rooms—are grouped very close to each other and show weaker loadings on both components. They are positioned on the right side of the graph with low loadings on both PC1 and PC2, which suggests that they do not contribute significantly to either component. This plot is very useful for identifying hidden patterns in the data.

Now I visualize the Biplot:



This graph is a biplot resulting from a principal component analysis. It shows both the observations (black dots) and the variables (blue arrows) in the space of the first two principal components.

• The first principal component (X-axis) explains 71.6% of the total variance. This indicates that much of the variability in the original data can be reduced to this axis.
• The second principal component (Y-axis) explains 11.6% of the variance. Although less relevant than the first component, it is still useful for describing a significant portion of the variability in the data.
Each arrow represents a variable in the dataset. The length of each arrow indicates how much that variable contributes to the variance explained by the principal components. The longer the arrow, the greater the contribution of that variable.
The direction of the arrows indicates the correlation between the variables and the principal components: variables like bathrooms, rooms, and m2 appear to be positively correlated with the first component (Dim1).
The condominium_expenses variable points in the opposite direction, indicating a negative relationship with the first component.
The price variable seems aligned with the first principal component, suggesting that price has a positive correlation with other variables (such as square meters and rooms).
The black dots represent the observations (properties in Milan). *Their position in the graph shows how each observation is distributed relative to the principal components.*
Most of the points are concentrated near the origin, suggesting that many observations do not deviate significantly along the first two principal components.
Some points further from the origin may represent properties with more extreme characteristics compared to the rest of the dataset.
The proximity between the arrows for rooms, bathrooms, and m2 suggests a strong correlation between these variables, which is intuitive: a larger number of rooms is likely associated with more bathrooms and larger square footage.

Supervised Learning

After completing this detailed analysis using the unsupervised learning technique called PCA (the results of which will be useful later), we move on to the supervised learning part. After various trials, I decided that the most suitable model for my analysis was decision trees: *decision trees are a supervised machine learning method used for both classification and regression problems. The goal of a decision tree is to create a model that predicts the value of the target variable based on several predictor variables.*

The main components of decision trees are:

• Decision nodes, which represent questions or conditions based on the predictor variables. Each node splits the dataset into two or more branches.
• Branches represent the answers to the question posed at the node (true/false or multiple categories).
• Leaves are the terminal nodes of the tree and represent the predicted class or value.

But how does a decision tree work?
At each level, the dataset is split based on the predictor variable that creates the best separation between the classes. The tree continues to split the dataset until it reaches a certain stopping criterion, which could be: maximum tree depth, a minimum number of observations per node, or a threshold for purity in the terminal nodes.

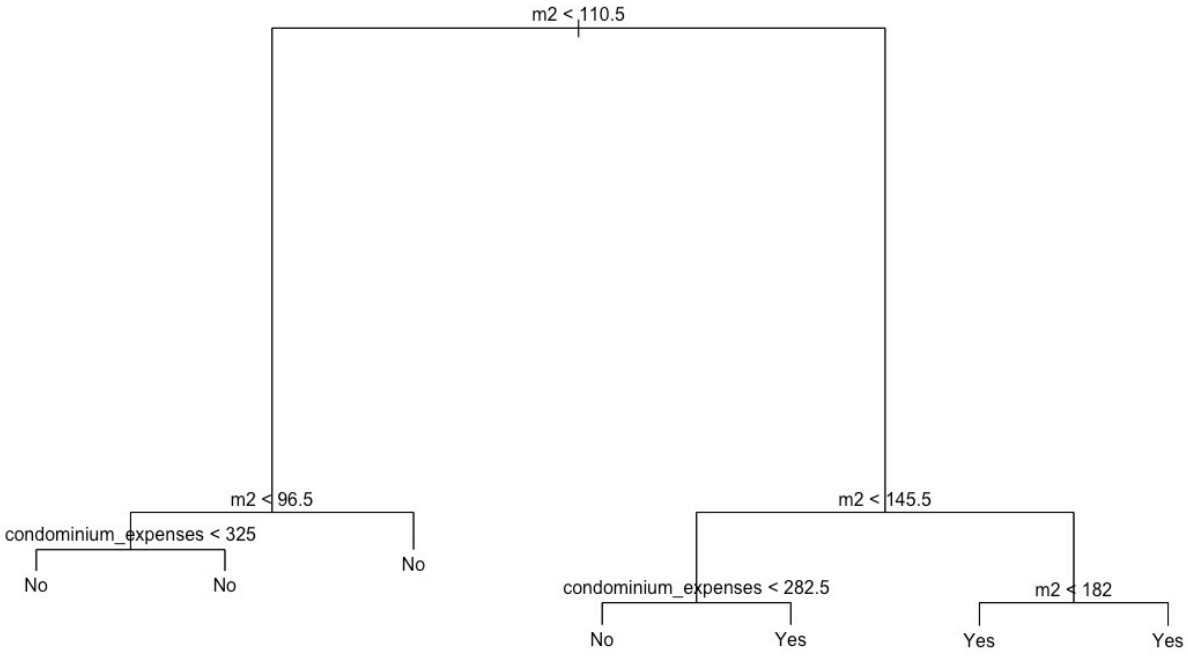
Pros of Decision Trees:

• Simplicity and interpretability: Decision trees are easy to understand and interpret, even for those with no experience in advanced machine learning techniques.
• No need for data preprocessing: There is no need to scale data or handle categorical variables separately, as decision trees work well with both numeric and categorical variables.

Cons:

• Overfitting: If not pruned, trees can become too complex and overly fit the training data, losing generalization ability.
• Stability: Decision trees can be unstable, meaning small changes in the data can lead to very different tree structures.

After setting the price threshold between expensive and non-expensive houses at €700,000, I trained and subsequently tested a predictive tree model on the data, which used the available variables to output whether or not each house was expensive.
Here is the result:



The tree represents a series of decisions based on predictive variables that lead to a binary classification of “Yes/No.”

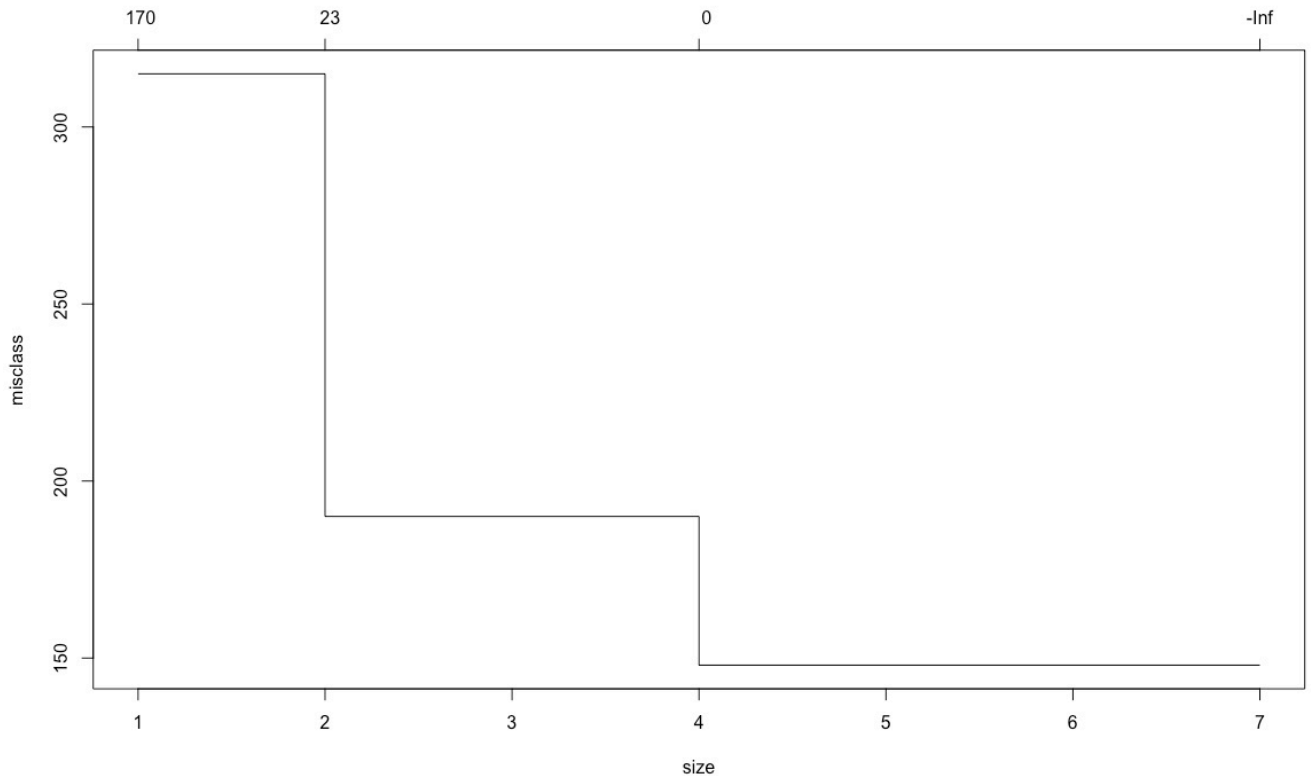
• Square meters (m²) seem to be the most important variable in influencing the decision. The main threshold is 110.5 m²: if the area is smaller, the tree generally tends to classify the properties as "No," indicating they are not high-value properties.
• Condominium expenses are used in some nodes to further refine the decision, but mainly for medium to large properties (i.e., above 96.5 or 145.5 m²).

After testing the model on the test set, I print the confusion matrix and calculate the success rate to assess its predictive reliability.

```
> # confusion matrix
>
> tree.pred=predict(tree_Houses_in_Milan2,Houses_in_Milan2[-train,],type="class")
> with(Houses_in_Milan2[-train,],table(tree.pred,price))
         price
tree.pred  No Yes
      No  273  37
      Yes  31  94
>
> # I compute the success rate
>
> confusion_matrix <- table(tree.pred, Houses_in_Milan2[-train,]$price)
> success_rate <- sum(diag(confusion_matrix)) / sum(confusion_matrix)
> print(success_rate)
[1] 0.8436782
```

The success rate is quite high, making the model reliable as it is approximately 84%. This means that the model was able to correctly predict the output 84 times out of 100 using records never seen before and not used in the training set.

I try to prune the tree to see if I can optimize the model and reduce overfitting while maintaining high accuracy.
As a first step, I visualize the CP Plot:



This graph is a complexity pruning plot (*also known as CP plot, from "cost complexity pruning"), which is used to evaluate the optimal number of nodes in the decision tree before proceeding with pruning.*

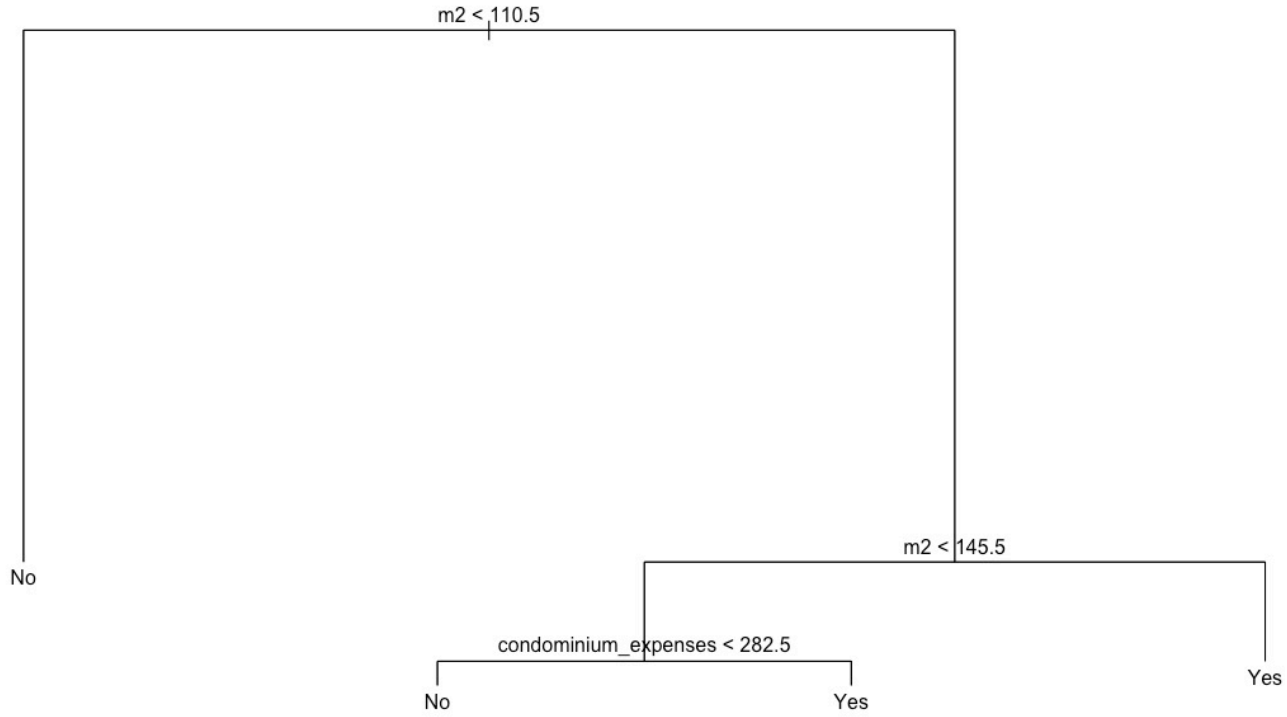
• The x-axis shows the size of the tree (the number of terminal nodes), while the y-axis indicates the misclassification rate (error rate in classification).

The graph shows how the error rate changes with the size of the tree: in general, a tree that is too large (complex) can overfit the training data, while a tree that is too small may not adequately capture the patterns present.

We notice that, initially, the tree has a very high error rate. At the point with size = 2, the error rate drops drastically (to about 23 errors). This suggests that a certain level of complexity is necessary to improve the tree's accuracy.
After the point size = 4, the error rate stabilizes at 0, suggesting that the tree has reached a very high level of fit or that there may be overfitting to the training data.
Finally, when the tree becomes complex and very large (size > 5), we see that the error rate becomes -Inf. This is indicative of a potential numerical issue or extreme overfitting, where the tree is memorizing the data instead of generalizing.

My final decision, after several tests, was to maintain 3 final nodes, as all other combinations led to lower success rates—2 caused evident underfitting, and from 4 onwards, the model started overfitting.

Here's the pruned tree:



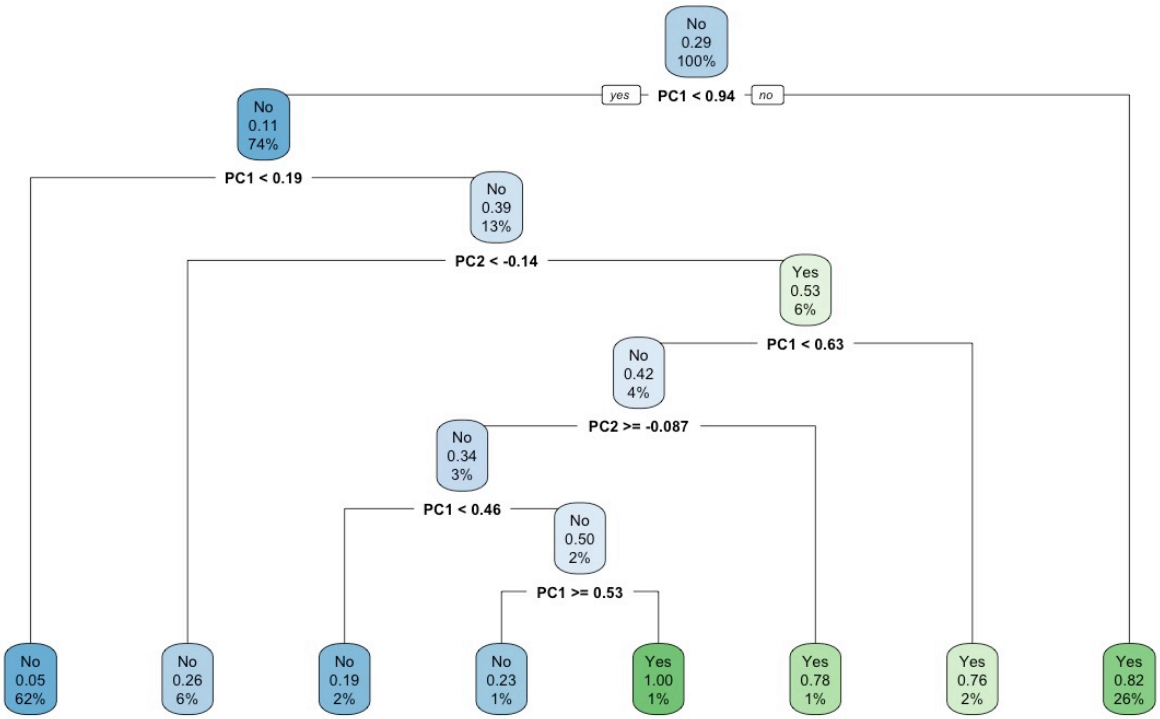
The new pruned tree has a smaller number of nodes and leaves. This is the result of the pruning process, which removes branches considered non-essential or that do not significantly improve the tree's predictive ability.
The variables that remain in this tree are only square meters and condominium expenses. This makes the tree more interpretable and less complex, though it may result in a slight loss of accuracy in exchange for greater generalization.

I then proceed to test the new model:

```
> # confusion matrix
>
> tree.pred=predict(prune.Houses_in_Milan2,Houses_in_Milan2[-train,],type="class")
> with(Houses_in_Milan2[-train,],table(tree.pred,price))
         price
tree.pred  No Yes
      No  273  37
      Yes  31  94
>
> # I compute the success rate
>
> confusion_matrix <- table(tree.pred, Houses_in_Milan2[-train,]$price)
> success_rate <- sum(diag(confusion_matrix)) / sum(confusion_matrix)
> print(success_rate)
[1] 0.8436782
```

The success rate remained exactly the same, so this technique worked perfectly, maintaining reliability while reducing complexity and the risk of overfitting on the training data.

Now I try a different approach: I train a decision tree on the results of the PCA and test its performance.



The PC1 seems to play a predominant role in the data division, as it is used for almost all the main decision nodes. This confirms what we have already noted, that the first principal component (PC1) explains much of the variability in the data and is a good indicator of the target classification.

Meanwhile, PC2 is used in subsequent nodes to further refine the decisions made based on PC1. This indicates that the second principal component is less important than PC1, but is still useful for separating more complex observations.

The percentages indicated next to each node represent the proportion of total observations that reach that node. For example, at the root, 100% of the observations are considered, but at the first bifurcation (PC1 < 0.94), only 74% follow the path to the left. These percentages can be useful in understanding how the dataset is split and how much each division contributes to the final classification.

Now, we just need to test the model and assess its accuracy:

```
> print(confusion_matrix)

tree_pred  No Yes
      No  270  37
      Yes  34  94
>
> # I compute the success rate
> success_rate <- sum(diag(confusion_matrix)) / sum(confusion_matrix)
> print(success_rate)
[1] 0.8367816
```

The model's accuracy (still approximately 84% as in the previous models) using the principal components derived from PCA is only slightly less precise than the model using all variables. The fact that the accuracy is lower is normal since PCA reduces the information in the data, even though it tries to retain most of the variance. However, the fact that the accuracy decreased so little is a sign that this procedure was carried out correctly and effectively retained most of the variability, while greatly simplifying the model's interpretability by using only 2 variables instead of the original 5.

Conclusions

In light of this work, I feel confident in listing the following most relevant and noteworthy findings:

1. Square meters as the main determinant of price: from the decision tree, it is clear that the most important variable for classifying house prices is the square footage (m^2), as expected before starting the analysis.
2. Condominium expenses as a secondary discriminating factor: this indicates that, in addition to the size of the property, fixed costs related to condominium expenses significantly influence market value.
3. Effective dimensionality reduction: the PCA showed that most of the variance in the data can be explained by the first two principal components. This allowed us to reduce the model's complexity without a significant loss of information.
4. Efficiency of the pruned tree: pruning the decision tree resulted in a simpler and more interpretable model, while maintaining the same predictive accuracy. The pruned tree reduced the number of nodes, eliminating less relevant variables and improving the model's robustness by limiting the risk of overfitting and keeping a clear decision structure.

Clearly, this is just a starting point, as a more in-depth market analysis would require a larger dataset and consideration of many more variables. However, I am very satisfied with the result obtained, both in terms of descriptive analysis of the phenomenon and the level of accuracy achieved by the models.