Analysis and prediction models for California's Housing Price

Andrea Frattini 959213

Contents

List of Figures

Abstract

The aim of this paper is to analyze the "California's Housing Price" Kaggle's dataset in order to develop suitable predictive models, the definition of data clusters and any a-priori relationships between variables. In particular, for achieving these purposes, in the first part the following Supervised Learning techniques will be used:

- Forward Stepwise
- Ridge Regression
- LASSO Model
- Partial Least Squares (PLS)

and then a comparison between the mean square errors provided by these models will be performed. Afterwards, in the second part, other two Unsupervised Learning techniques will be implemented:

- Principal Component Analysis
- K-means Clustering

for visually evaluating the behavior of the data. In the end, it is provided a conclusion of the analysis performed.

1 Theoretical Concepts

1.1 Forward Stepwise Model Selection

This method works in the following way:

1. It starts with a model that contains no variables (called *Null Model*).

2. Then, at each step, it adds the most significant variable one after the other.

3. Finally it stops when a pre-specified stopping rule is reached or when all the variables have been included in the model.

There are different ways in which the model can determine the most significant variable at each step:

- The one that has the smallest p-value.
- The one that provides the highest increase in R2.
- The one that provides the highest drop in model RSS (Residuals Sum of Squares) compared to other predictors under consideration.
- The one that is mostly correlated with the response variable, if you are evaluating which are the best variables for a regression model.

In this paper, the last criterion is going to be used. Mathematically, the Forward Selection algorithm is as follows:

1. Let M_0 denote the null model
2. for $k = 0$, ... ,p - 1 (where p is the number of predictors):
   (a) Consider all p-k models that augment the predictors in M_k with one additional predictor.
   (b) Choose the best among these pk models, and call it M_{k+1}. Here best is defined as having smallest RSS or highest R2.
3. Select a single best model from among M_0, . . . , M_p using cross-validated prediction error, Cp (AIC), BIC, or adjusted R^2.

The model containing all of the predictors will always have the smallest RSS and the largest R2, since these quantities are related to the training error and, then, we want a model with low test error. For this reason the RSS and R^2 are not suitable for selecting the best model among a collection of models with different numbers of predictors. In this paper the Mallow's Cp is going to be used first, while cross-validation will be used secondly. Let us recall these two concepts.

1.1.1 Mallow's Cp

Using this type of error we will select the number of variables that provide the model with the lowest Cp's value. The formula is:

$$\text{Cp} = \frac{1}{n}(\text{RSS} + 2\text{d}\sigma^2)$$

where d is the total number of parameters used and σ^2 is an estimate of the variance of the error ϵ associated with each response measurement.

1.1.2 Cross-Validation For Forward Model Selection

Each of the procedures returns a sequence of models M_k indexed by model size k = 0,1,2,... . Our job here is to select $\hat{k}$. Once selected, we will return model $M_{\hat{k}}$ and we will compute the cross-validation error for each model M_k under consideration, and then select the k for which the resulting estimated test error is the smallest. This procedure has an advantage relative to AIC, BIC, Cp, and adjusted R^2, in that it provides a direct estimate of the test error, and doesn't require an estimate of the error variance σ^2.

1.2 Ridge Regression

There are many way other than classical regression through which is possible to handle the bias-variance trade-off. Among these methods, the *regularization* or *shrinkage* methods are very useful for the best predictor selection and improvement of estimates. The aim is to make the estimates of βs as smaller as possible or to set them to zero. One of those regularization methods is the so called *Ridge Regression*, that has the following formula:

$$\hat{\beta}_r(\lambda) = \underset{||\beta||^2 \leq \lambda}{\arg\min} \, ||Y - X\beta||^2 \text{ for any } \lambda > 0$$

Ridge regression uses the minimizer of a penalized squared error loss function to estimate the regression coefficients, where "penalized" means that coefficients are forced to go towards zero through a tuning parameter, λ. Coefficient estimates $\hat{\beta} = (\hat{\beta}_0, \hat{\beta}_1, \ldots, \hat{\beta}_p)$ are given by:

$$\hat{\beta}_r(\lambda) = \underset{||\beta||^2 \leq \lambda}{\arg\max} \, ||Y - X\beta||^2 + \lambda\beta^1 D\beta \qquad (1)$$

where typically D is a diagonal matrix with 0 in the [1, 1] position and ones on the rest of the diagonal. The last term is the penalty and, in particular, $\beta^1 D\beta$ is the sum from j=1 to p of the ${\beta_j}^2$ so that the intercept is not considered and then we can express the quantity to minimize as:

$$\sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{p}\beta_{ij}x_{ij})^2 + \lambda\sum_{j=1}^{p}{\beta_j}^2 = \text{RSS} + \lambda\sum_{j=1}^{p}{\beta_j}^2$$

Furthermore, by taking centered values of x_{ij} (i.e. zero mean values), we can also estimate the intercept through the expression:

$$\hat{\beta}_0 = \bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$$

In this way we have to estimate p coefficients $\beta_1, \ldots, \beta_p$ in the ridge regression and therefore equation (1) becomes:

$$\hat{\beta} = \arg\min_{\beta} ||Y - X\beta||^2 + \lambda\beta^1\beta$$

1.2.1 Remarks

For a complete comprehension of what we have just said, it's important to make some remarks about the *shrinkage penalty* $\lambda\sum_{j=1}^{p}{\beta_j}^2$. It depends on the tuning parameter λ:

- When $\lambda = 0$ the shrinkage penalty term has no effect at all and we get is the linear model.
- When the shrinkage penalty is small, then $\beta_1, \ldots, \beta_p$ are close to zero and the linear regression is prevalent.
- As λ grows the shrinkage effect tends to grow too and $\beta_1, \ldots, \beta_p$ will approach zero closer and closer.

This makes most sense when covariates have been standardized, so that the β_js are 'equally penalized'.
Another important remark is that ridge regression model depends also on the scale of the predictors and then it's fundamental to standardize the predictors before performing it.

1.2.2 Ridge Regression and Collinearity

Let X_1, X_2 be two standardized, strongly positively collinear predictors and let β_1, β_2 be their respectively slopes. Then fits of the form

$$(\beta_1+\gamma)X_1+(\beta_2-\gamma)X_2 = \beta_1X_1+\gamma X_1+\beta_2X_2-\gamma X_2 = \text{EY} + \gamma(X_1 - X_2)$$

have similar MSE as γ varies since the difference between X_1 and X_2 is small when they are strongly positively correlated, e.i. the OLS cannot easily distinguish among this fits. For large values of γ the ridge regression prefers fits that minimize the quantity $(\beta_1 + \gamma)^2 + (\beta_2 - \gamma)^2$, which is minimized when $\gamma = (\beta_1 - \beta_2)/2$. Ridge Regression prefers coefficient estimates for which strongly positively correlated covariates have similar estimated effects and therefore, when there is multicollinearity, it adds a little bit of bias reducing variance at the same time (while, instead, least squares estimates are unbiased but their variance could be large).

1.3 Least Absolute Shrinkage and Selection Operator (LASSO) Model

In regression, when the number of predictors p is large compared to the sample size, there are a lot of problems in applying simple methods (e.g. OLS), such as:

- Multicollinearity: some predictors can be highly correlated and sometimes, when $p > n$, the least squares estimator could not give a solution.
- Overfitting: a model with high p may be sub-optimal when the true model is sparse

One possible solution for such problems, among the others, is a regularization method called *LASSO*.
A disadvantage of the ridge regression is that it maintains all the input variables in the result, making the interpretation more difficult. The question arises spontaneously: can we do variable selection and shrinkage estimation simultaneously? Yes we can, and one methods to do so in the LASSO model: it estimates β by minimizing:

$$\sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij})^2 + \lambda\sum_{j=1}^{p}|\beta_j| = \text{RSS} + \lambda\sum_{j=1}^{p}|\beta_j|$$

where the difference with the ridge lies in the penalty part: it could seem a little difference but, actually, it makes qualitative gaps practically as well as theoretically. Indeed, the nature of this penalty term causes some coefficients to be shrunken exactly to zero providing sometimes, as results, sparse predictive models (e.i. models where some coefficients are exactly zero). In this way, LASSO is able to performe also variable selection.
In a situation where most of the supposed real coefficients could be zero, shrinkage methods are attractive, and, among these methods, LASSO is probably the most popular and used.

1.3.1 LASSO's Problems

The main problem of LASSO is to manually find the solution: standard numerical optimization methods cannot be applied directly since we need to optimize a non-differentiable objective function and the presence of the absolute value in the loss function implies that the solutions are nonlinear in the Y_i, and there is no closed form expression as in ridge regression. Fortunately, there are softwares that make the computation for us using one of the following alternative formulas:

1. $\hat{\beta_L} = \arg\min_{\beta} [\sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij})^2 + \lambda \sum_{j=1}^{p} |\beta_j|] =$
   $= \arg\min_{\beta} [\text{RSS} + \lambda \sum_{j=1}^{p} |\beta_j|]$

2. $\hat{\beta_L} = \arg\min_{\beta} [\sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij})^2$ subject to $\sum_{j=1}^{p} |\beta_j| \leq s$

where for every value of λ, a corresponding value s can be found such that the two equations share the same solutions: s can be thought as a *budget*.
Note: λ and s go in opposite directions.
In conclusion, the smaller the value of the tuning parameter s (or the larger the penalty parameter λ), the more shrinkage we will have for the OLS estimates. Shrinkage reduces the variance of the estimates, at the expense of (likely) introducing bias. It is often the case that the reduction in variance (tuned at some level s) is well worth the price of bias, so that the trade-off in overall accuracy is favorable.

1.4 Partial Least Squares (PLS)

Partial Least Squares tries to find the direction that best represents the predictors in a *supervised way*. The main difference with the Principal Component Regression (PCR, an unsupervised method which has the same aim of PLS) is that PLS uses the response variable to identify new components that approximate the original predictors well, but also that are related to the response. PLS constructs a set of linear combinations of the original predictors for regression, but it uses y (in addition to X) for this construction. It is assumed that the response variables are centered and the predictors standardized. The procedure begins by computing the univariate regression coefficients $\hat{\phi}_j$ from p simple linear regression for y regressed on each X_j. The regression coefficients from simple linear regression models are proportional to the linear correlation coefficient between the response y and the predictor X_j, where j = 1,2,...,p.
These coefficients $\hat{\phi_{11}}, \hat{\phi_{12}}, ..., \hat{\phi_{1j}}, ..., \hat{\phi_{1p}}$ are projections that can be interpreted as the 'loadings' of the following first 'component' (using PCA's terminology): $z_1 = \sum_{j=1}^{p} \hat{\phi_{1j}} x_{ij}$, which is the *first partial least squares direction* corresponding to the first component in PCA. In doing that, PLS places the highest weight on the variables that are more strongly related to the response. Then, to identify the *second PLS direction* we first adjust each variable x_j for z_1, by regressing it on z_1 and taking residuals (we can think about this residuals as the remaining information not explained by the first PLS direction). We then compute z_2 (using the orthogonalization of x_1, x_2, . . . , x_p) in exactly the same way as z_1 was computed based on the original data (this iterative approach can be repeated m times to identify multiple PLS components z_1, . . . , z_m). In the end, we use least squares to fit a linear model to predict Y using z_1, . . . , z_m.
Note: the inputs x_j in the linear combination z_1 are weighted by their relationship with the response y.

1.5 Principal Component Analysis (PCA)

The aim of the PCA is to produce a low-dimensional representation of a dataset finding a sequence of linear combinations of the variables that have maximal variance and are mutually uncorrelated. The *First Principal Component* of a set of features X_1, X_2, . . . , X_p is the normalized linear combination of the variables $Z_1 = \phi_{11}X_1 + \phi_{21}X_2 + ... + \phi_{p1}X_p$, that has

the largest variance. When we say *normalized* we mean that the sum of the squared ϕ_js equals 1, where $\phi_{11}, ..., \phi_{p1}$ are the so called *loadings* of the first principal component which, all together, make up the principal component loading vector: $\phi_1 = (\phi_{11} \quad \phi_{21} \quad ... \quad \phi_{p1})^T$.

1.5.1 Computations of PCA

Suppose we have a n x p data set X. We are only interested in the variance and then we assume that all variables in X are standardized (each of them has mean zero). We then look for the linear combination of the sample features:

$$z_{i1} = \phi_{11}x_{i1} + \phi_{21}x_{i2} + ... + \phi_{p1}x_{ip} \qquad (1)$$

for $i = 1, ..., n$, that has the largest variance, subject to the usual constraint. Since each of the x_{ij} has mean zero, then so does z_{i1} and, therefore, the sample variance of the z_{i1} can be written as $\frac{1}{n}\sum_{i=1}^{n} z_{i1}^2$. Substituting equation (1) into the first principal component loading vector we can solve the following optimization problem:

$$\underset{\phi_{11},...,\phi p1}{\arg\max} \frac{1}{n}\sum_{i=1}^{n}(\sum_{j=1}^{p}\phi_{j1}x_{ij})^2 \text{ s.t. } \sum_{j=1}^{p}\phi_{j1}^2=1$$

and from here on we will refer to Z_1 as the first principal component, with realized values $z_{11}, ..., z_{n1}$.
The loading vector defines a direction in feature space along which the data vary the most. If we project the n data points $x_1, ..., x_n$ onto this direction, the projected values are the principal component scores $z_{11}, ..., z_{n1}$ themselves.
For the subsequent principal component, e.i. the second one, we take the linear combination of $X_1, ..., X_p$ that has maximal variance among all linear combinations that are uncorrelated with Z_1. The second principal component scores $z_{12}, z_{22}, ..., z_{n2}$ take the form: $z_{i2} = \phi_{12}x_{i1} + \phi_{22}x_{i2} + ... + \phi_{p2}x_{ip}$, where ϕ_2 is second principal component loading vector and therefore it turns out that constraining Z_2 to be uncorrelated with Z_1 is equivalent to constrain the direction ϕ_1 to be orthogonal (perpendicular) to the direction ϕ_2, and so on.
Note: scaling the variables is strongly recommended for getting and understandable *biplot*.

1.5.2 Proportion of Variance Explained (PVE)

To understand the strength of each component, we are interested in knowing the proportion of variance explained (PVE) by each one. The *total variance* present in a data set is defined as:

$$\sum_{j=1}^{p} VAR(X_j) = \sum_{j=1}^{p} \frac{1}{n} \sum_{i=1}^{n} x_{ij}^2$$

and the variance explained by the mth principal component is :

$$\text{VAR}(Z_m) = \frac{1}{n} \sum_{i=1}^{n} z_{im}^2$$

Therefore, the PVE of the mth principal component is given by the quantity:

$$\frac{\sum_{i=1}^{n} z_{im}^2}{\sum_{j=1}^{p} \sum_{i=1}^{n} x_{ij}^2} \qquad \text{which is between 0 and 1.}$$

The PVEs sum to one and display them in a plot is very useful in order to understand the behavior. In the end we have to say that, for deciding how many principal components to use, we should use a *scree plot* and look for a kink (or elbow).

1.6 K-means Clustering

K-means Clustering is very simple to understand even if it has several disadvantages. The problem is how to assign semantics to each cluster and thus measure the "performance" of the algorithm: in many cases, a good way to proceed is to visualize the clusters. Obviously, if the data has high-dimensional features, as in many cases, the visualization is not that easy. In this case, we can reduce the dimensionality of our data using, for example, PCA and therefore plotting the data in a 2D plot and then visualize the clusters. However, what we see in this graph is a 2D projection of our data, so it may not be very accurate, but it can still give us an idea of how our clusters are distributed.

1.6.1 Implementation of K-means Algorithm

K-means clustering is based on the idea of a centroid. The centroid of a set of data points $S = x_1, x_2, \ldots, x_n$ can be calculated as $\text{C} = \frac{\sum_{i=1}^{n} x_i}{|S|}$ where $|S|$ is the number of points in the data set. The centroid is essentially the geometric centre of the data set. The algorithm works in the following way:

1. Initializing Centroids: we must know a priori how many clusters (and then how many centroids) we are looking for (i.e. the k value). It is common practice to randomly initialize the centroids, e.i. randomly select k data points from your data set and call them the centroids for the k clusters.

2. Assigning Data Points to a Cluster: it assigns every data point to a cluster by computing the distances between the data point in question and each centroid. It then assigns that data point to the cluster belonging to the closest centroid. It repeats this procedure for every data point, thus assigning each data point to a cluster.

3. Recompute the Centroids: Once each data point has been assigned to a cluster, the centroid of each cluster needs to be computed. This can be done using the centroid formula shown above. Then, the algorithm iterates between step 2 and step 3 until convergence occurs (e.i. when the centroids stop changing).

1.6.2 Disadvantages of k-Means Clustering

They are:

- The final clusters are highly dependent upon the initialization choices of the initial centroids. Since the centroids are often randomly initialized, this could lead to different results each time that the algorithm is run.
- The algorithm is quite computationally intensive.
- We must know the k value a priori.

---

2 Introduction to the dataset

First, we need to understand how the "California's Housing Price" dataset is structured. This file has been downloaded from Kaggle (https://www.kaggle.com/camnugent/california-housing-prices) and it contains data concerning the following characteristics of California's houses:

- Longitude: A measure of how far west a house is; a higher value is farther west.
- Latitude: A measure of how far north a house is; a higher value is farther north.
- HousingMedianAge: Median age of a house within a block; a lower number is a newer building.
- TotalRooms: Total number of rooms within a block.
- TotalBedrooms: Total number of bedrooms within a block.
- Population: Total number of people residing within a block.
- Households: Total number of households, a group of people residing within a home unit, for a block.
- MedianIncome: Median income for households within a block of houses (measured in tens of thousands of US Dollars).
- MedianHouseValue: Median house value for households within a block (measured in US Dollars).
- OceanProximity: Location of the house w.r.t ocean/sea.

The dataset in composed by 20640 observations, each of which with the 10 features described above. Let's start the analysis with some preliminaries inspections of the data.

---

3 Pre-Processing Phase

3.1 Inspecting the dataset

Before starting to analyze the dataset, it's crucial to see how it is structured, looking for missing values or possible outliers. The following figure represents the output of the command which returns the original dataset:

```
  longitude         latitude     housing_median_age  total_rooms    total_bedrooms     population
Min.   :-124.3   Min.   :32.54   Min.   : 1.00      Min.   :    2   Min.   :   1.0   Min.   :    3
1st Qu.:-121.8   1st Qu.:33.93   1st Qu.:18.00      1st Qu.: 1448   1st Qu.: 296.0   1st Qu.:  787
Median :-118.5   Median :34.26   Median :29.00      Median : 2127   Median : 435.0   Median : 1166
Mean   :-119.6   Mean   :35.63   Mean   :28.64      Mean   : 2636   Mean   : 537.9   Mean   : 1425
3rd Qu.:-118.0   3rd Qu.:37.71   3rd Qu.:37.00      3rd Qu.: 3148   3rd Qu.: 647.0   3rd Qu.: 1725
Max.   :-114.3   Max.   :41.95   Max.   :52.00      Max.   :39320   Max.   :6445.0   Max.   :35682
                                                                   NA's   :207
  households     median_income     median_house_value   ocean_proximity
Min.   :   1.0   Min.   : 0.4999   Min.   : 14999     <1H OCEAN :9136
1st Qu.: 280.0   1st Qu.: 2.5634   1st Qu.:119600     INLAND    :6551
Median : 409.0   Median : 3.5348   Median :179700     ISLAND    :   5
Mean   : 499.5   Mean   : 3.8707   Mean   :206856     NEAR BAY  :2290
3rd Qu.: 605.0   3rd Qu.: 4.7432   3rd Qu.:264725     NEAR OCEAN:2658
Max.   :6082.0   Max.   :15.0001   Max.   :500001
```

Figure 3.1: Structure of the dataset

The figure 3.1 shows that the feature TotalBedrooms contains 207 NA (i.e. null values) and that there are some other features which have a maximum value very far from the 3rd quartile, such as TotalRooms, TotalBedrooms, Population, Households, MedianIncome and MedianHouseValue, which means that later it will be necessary to check for outliers. But, for the moment, let's focus on the last feature, i.e. OceanProximity: it is a categorical variable and one of its values (ISLAND) appears only 5 time over 20640 observations.
So, after having understood the main characteristic of the dataset, the first things to do are to delete the observations containing null values (we can do that because the NA are only around the 1% of the dataset, and then this operation won't affect the future results) and then to convert the variable OceanProximity from a categorical to a binary one, using the "one hot encoding" technique, which allows to automatically create new dummy features (one for each subcategory of the original feature) and put a 1 into a cell if the characteristic express by the name of the current dummy variable is possessed by the original corresponding observation, and 0 otherwise. Then, we delete the column containing the variable OceanProximityISLAND since, as said, it contains only 5 values and then it would be useless for our analysis.
Another important aspect to highlight is that most of the variables have different measurement units and then it's crucial to standardize all of them in order to eliminate the unit of measure and to center their mean to zero.
After having performed all the aforementioned operations, the new dataset looks like in the following figure:

```
  longitude          latitude       housing_median_age  total_rooms      total_bedrooms      population
Min.   :-2.3854   Min.   :-1.4479   Min.   :-2.19453   Min.   :-1.2056   Min.   :-1.2741   Min.   :-1.2548
1st Qu.:-1.1127   1st Qu.:-0.7973   1st Qu.:-0.84445   1st Qu.:-0.5430   1st Qu.:-0.5740   1st Qu.:-0.5630
Median : 0.5394   Median :-0.6428   Median : 0.02914   Median :-0.2332   Median :-0.2441   Median :-0.2285
Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.00000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000
3rd Qu.: 0.7790   3rd Qu.: 0.9768   3rd Qu.: 0.66447   3rd Qu.: 0.2318   3rd Qu.: 0.2590   3rd Qu.: 0.2621
Max.   : 2.6256   Max.   : 2.9568   Max.   : 1.85572   Max.   :16.7867   Max.   :14.0184   Max.   :30.2301
  households      median_income     median_house_value ocean_proximity_<1H OCEAN ocean_proximity_INLAND
Min.   :-1.3038   Min.   :-1.7750   Min.   :-1.6621   Min.   :0.0000            Min.   :0.0000
1st Qu.:-0.5740   1st Qu.:-0.6884   1st Qu.:-0.7568   1st Qu.:0.0000            1st Qu.:0.0000
Median :-0.2366   Median :-0.1762   Median :-0.2353   Median :0.0000            Median :0.0000
Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000   Mean   :0.4421            Mean   :0.3179
3rd Qu.: 0.2735   3rd Qu.: 0.4596   3rd Qu.: 0.5010   3rd Qu.:1.0000            3rd Qu.:1.0000
Max.   :14.6026   Max.   : 5.8595   Max.   : 2.5394   Max.   :1.0000            Max.   :1.0000
ocean_proximity_NEAR BAY ocean_proximity_NEAR OCEAN
Min.   :0.0000           Min.   :0.0000
1st Qu.:0.0000           1st Qu.:0.0000
Median :0.0000           Median :0.0000
Mean   :0.1111           Mean   :0.1286
3rd Qu.:0.0000           3rd Qu.:0.0000
Max.   :1.0000           Max.   :1.0000
```

Figure 3.2: Cleaned up, standardized dataset

3.2 Outliers

Due to the great distance between the maximum value and the 3rd quartile of some variables, it's likely to expect to have some outliers. For checking their presence, the most common technique is to look at the boxplot of the standardized dataset.

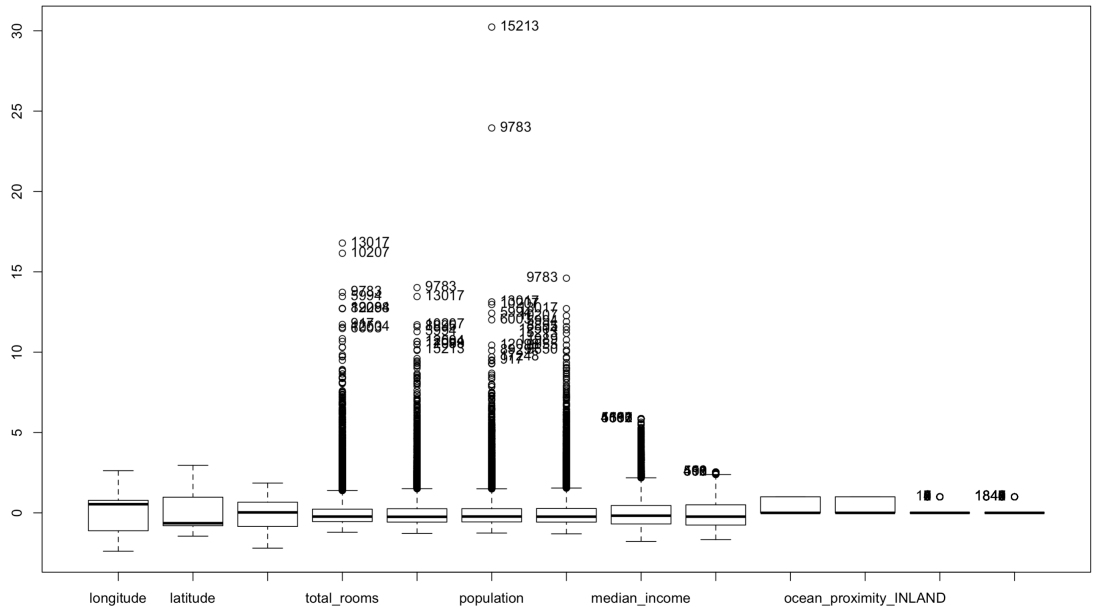

Figure 3.3: Boxplot of the standardized dataset

It's rather easy to see from Fig.3.3 that from the feature TotalRooms until MedianIncome there are a lot of outliers and some of them are very far from the mean of the corresponding variable. Given that those outliers are likely to affect the results of the analysis, the best way is to drop them. In particular, to find data points far from the mean, at each feature has been applied the usual formula for defining outliers, i.e. a point x is NOT considered an outliers if $x > Q1 - 1.5 * IQR$ or if $x < Q3 + 1.5 * IQR$, where Q1 = first quartile, Q3 = third quartile and IQR = interquartile range, namely Q3 - Q1. In this way, it was easy to identify and eliminate the outliers, ending up with the following boxplot.

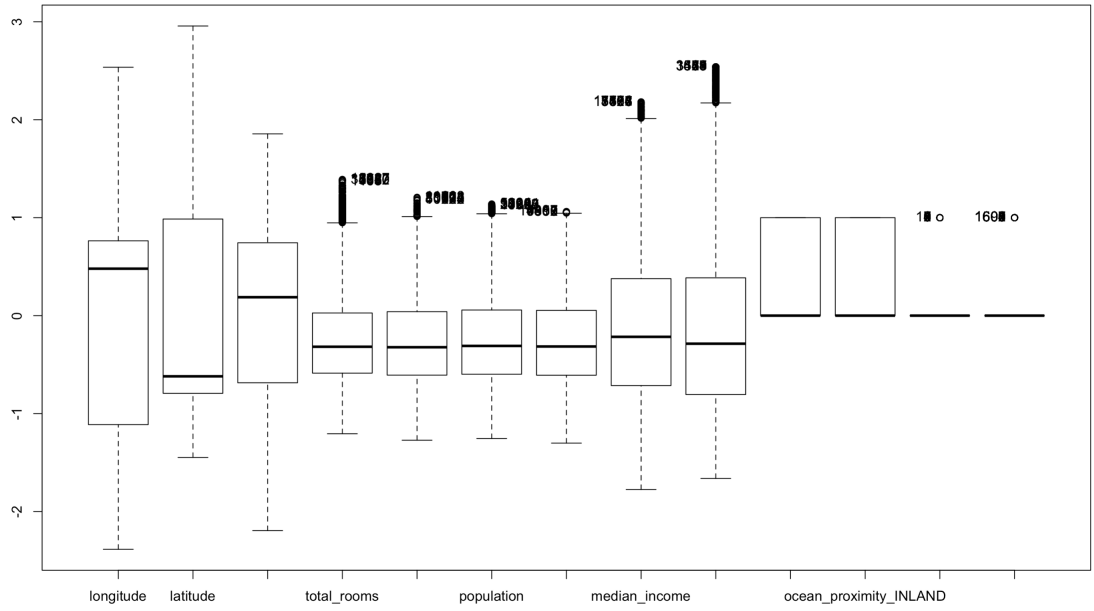

Figure 3.4: Boxplot without outliers

The most important thing to notice comparing Fig.3.3 and Fig.3.4 is that the vertical axis has been reduced from a 0-30 range to a 0-3 range, maintaining only those outliers that are likely to not affect too much the computations. Now the dataset has been restructured in a proper way and then it's possible to go over with the inspection of the features.

3.3 Collinearity

It's also important to inspect the relationship between all the variables because when some features are highly correlated with each other the OLS will provide unstable estimates of the coefficients. So, if in this dataset there will be high correlated features, then it would be necessary to carefully think about which would be the right models for analyzing our data.

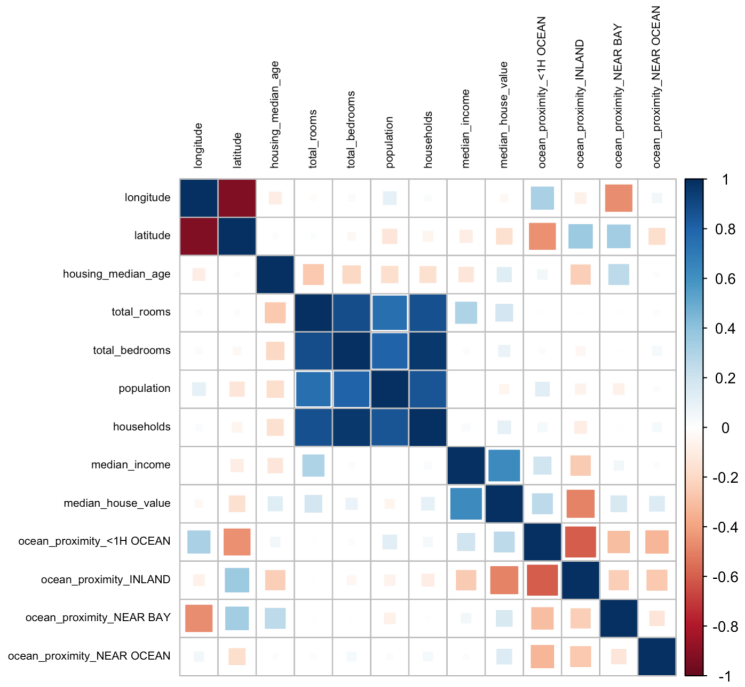

Figure 3.5: Correlation plot

As shown by the figure 3.5, which represents the correlation plot of the dataset, our data are affected by multicollinearity (a lot of variables are strongly correlated with each other). Hence, a good model for analyzing them could be the Ridge Regression, since it helps to overcome the problem of highly correlated variables by shrinking the estimated weights toward zero and decreasing the whole variance. Another possible solution could be to use the LASSO Regression: in contrast with the Ridge, which minimizing the weight of some feature reduces their contribution to the model, the Lasso model performs an actual 'variable selection' bringing the not selected to

zero and generating a sparse model (with some nonzero features).
For a more complete and deeper analysis, in this paper we will use also the Forward Stepwise Model Selection technique, in order to have a further level of understanding and comparison.

4 Supervised Learning Analysis

4.1 Forward Stepwise Model Selection

In our situation, this technique creates nested models following the last scenario described the subsection "1.1 Forward Stepwise Model Selection" of the section "Theoretical Concepts" for selecting the most significant variable to add at each step, as shown in the following figure (which is a sample of the entire work done by the model).

```
Selection Algorithm: forward
          longitude latitude housing_median_age total_rooms total_bedrooms population households median_income
1  ( 1 )  " "       " "      " "                " "         " "            " "        " "        "*"
2  ( 1 )  " "       " "      " "                " "         " "            " "        " "        "*"
3  ( 1 )  " "       " "      "*"                " "         " "            " "        " "        "*"
4  ( 1 )  " "       " "      "*"                " "         "*"            " "        " "        "*"
5  ( 1 )  " "       " "      "*"                " "         "*"            "*"        " "        "*"
6  ( 1 )  " "       " "      "*"                "*"         "*"            "*"        " "        "*"
7  ( 1 )  " "       " "      "*"                "*"         "*"            "*"        "*"        "*"
8  ( 1 )  " "       " "      "*"                "*"         "*"            "*"        "*"        "*"
9  ( 1 )  "*"       " "      "*"                "*"         "*"            "*"        "*"        "*"
10  ( 1 ) "*"       "*"      "*"                "*"         "*"            "*"        "*"        "*"
11  ( 1 ) "*"       "*"      "*"                "*"         "*"            "*"        "*"        "*"
12  ( 1 ) "*"       "*"      "*"                "*"         "*"            "*"        "*"        "*"
```

Figure 4.1: Variable selection step by step

Then, in order to see which is the number of variable that guarantees the lowest Mallows' Cp, we plot the result of the Forward Stepwise against the Cp:

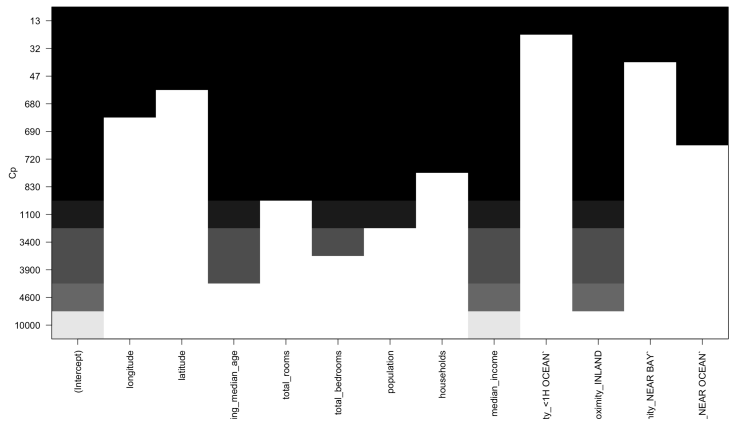

Figure 4.2: Mallow's Cp for any variable

From the graph in Fig.4.2 we can see that the model providing the lowest Cp (e.i. the lowest estimated test error) is the one containing all the variable but the response one (MedianHouseValue).

4.1.1 Forward Stepwise Model Selection using K-Fold Cross-Validation

Since we have 12 variables in the dataset, we know that there are 12 models to evaluate through k-fold cross-validation and this is a crucial information given that, in order to proceed, we first need to construct a N x P matrix where N is the number of folds (10 in this case) and P is the number of variables to analyze. This matrix will contain, column by column, the Mean Square Error (MSE from here on) of each one of the k iteration associated with the model with a number of variables corresponding to the index of the current column. For example, the first column will contain the MSEs of all the k iteration of the model including only the first feature (the most correlated with the response variable); the second column will contain the MSEs of all the k iteration of the model including the first two most correlated variables, and so on until the MSEs model including all the 12 variables are stored. Finally, we take the column-by-column average and we plot the result.

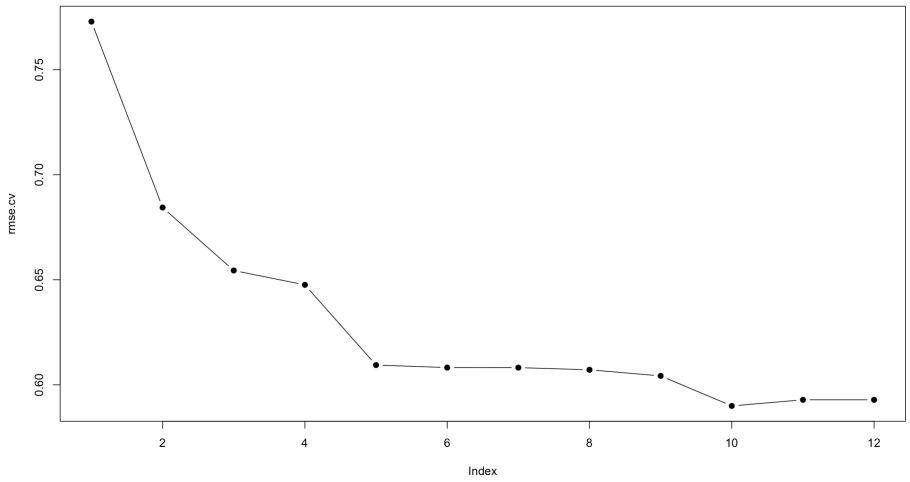

Figure 4.3: MSE for each model

Obviously, for obtaining the above graph the dataset has been previously divided into a training set (70% of the entire dataset) on which the model has been trained, and a test set (the remaining 30% of the dataset) on which predictions have been calculated, in order to compute the MSEs. The figure 4.3 shows that the best model seems to be the one with only 10 variables, which yield to the following value of MSE:

```
$Value
[1] 0.5899694

$Index
[1] 10
```

Figure 4.4: MSE value provides by the model using 10 variables

4.2 Ride Regression with Cross-Validation

As explained before, due to the presence of multicollinearity, the Ridge Regression should be a good predictive model in our case. So, a cross-validated ridge regression has been performed over the data, in order to understand how the MSE behaves varying the value of the logarithm of Lambda.

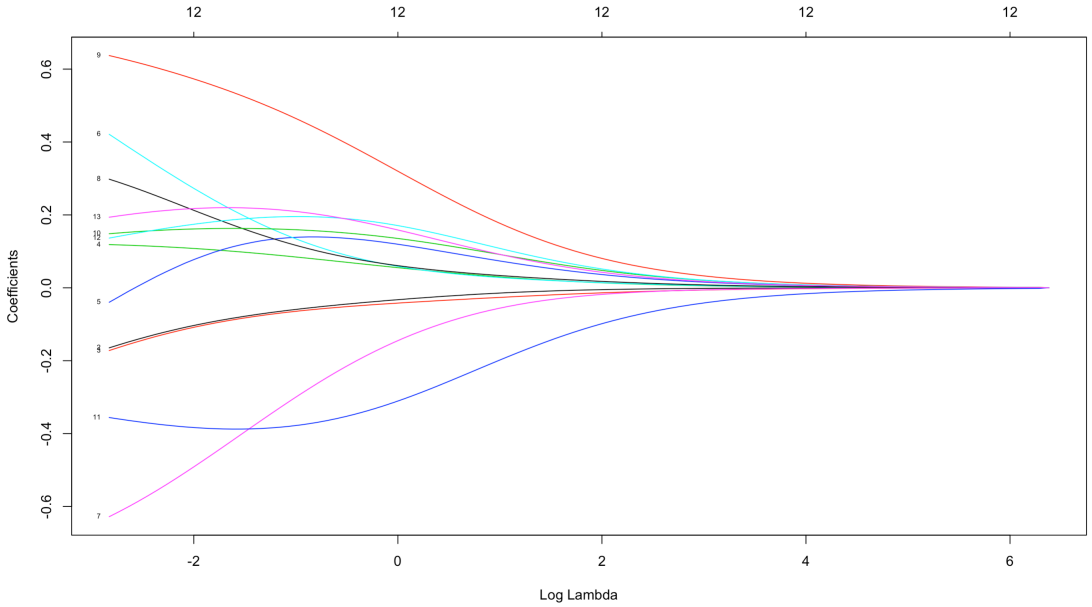

Figure 4.5: Variables' behavior over different values of logLambda

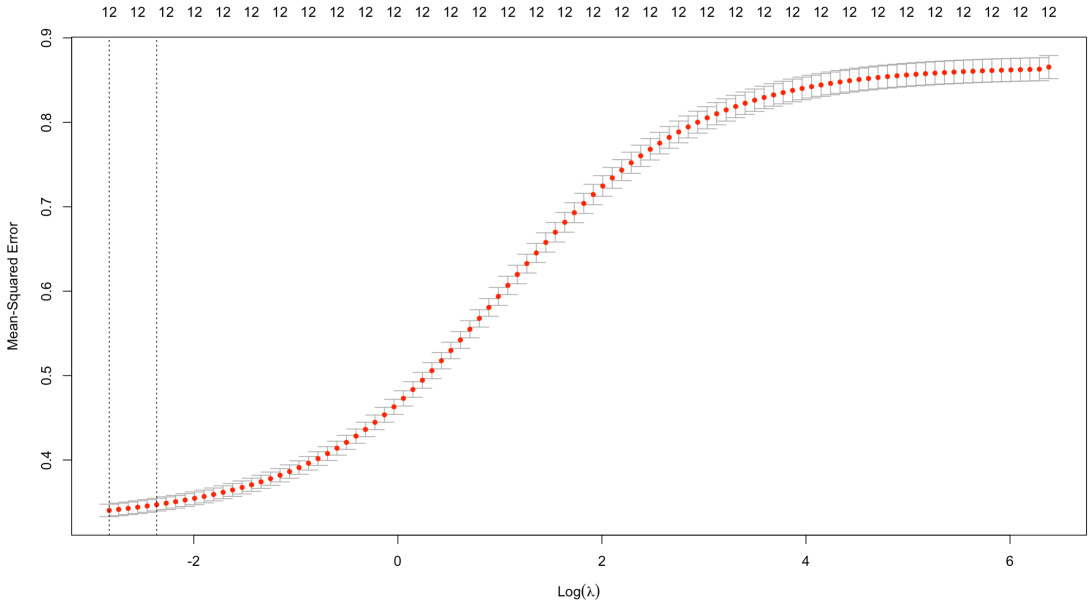

Figure 4.6: MSE behavior over different values of logLambda

Analyzing first the figure 4.6 we can see that the logLambda values providing the minimum MSE and the minimum variance are, respectively, the most-left vertical dotted line and the subsequent dotted vertical line, which are very close with each other. Now, for understanding what happens to each variable in the situation corresponding to the aforementioned logLambda value, the reader should imagine lengthening the two dotted lines over the figure 4.5: in both the situation, the lines intersect the variables when their values are different from zero (as expected, since the ridge bring the value of less useful variables close to zero, but not exactly zero). In particular, we are interested in the MSE value associated to the logLambda's value which provides the minimum variance: in this case, as pointed out before, the value of the 12 variables is different from zero and then, all of them are included in the model. The coefficients (estimated over the logLambda's value that provides the minimum variance) and the corresponding MSE are shown in the following figures.

```
$`Value of log(lambda) providing minimum MSE`
[1] -2.829524

$`Value of log(lambda) providing minimum variance`
[1] -2.364355

$`MSE associated to the lambda which provides the lowest variance`
[1] 0.3471329
```

Figure 4.7: logLambda and corresponding MSE values

```
14 x 1 sparse Matrix of class "dgCMatrix"
                                        s1
(Intercept)                   0.003745722
(Intercept)                   .
longitude                    -0.127104295
latitude                     -0.132990028
housing_median_age            0.113541640
total_rooms                   0.031361956
total_bedrooms                0.335767426
population                   -0.556272487
households                    0.252074665
median_income                 0.603680028
`ocean_proximity_<1H OCEAN`   0.157308390
ocean_proximity_INLAND       -0.373827542
`ocean_proximity_NEAR BAY`    0.159228852
`ocean_proximity_NEAR OCEAN`  0.209865944
```

Figure 4.8: Coefficients of Ridge Regression

The figure 4.8 shows that the variables Longitude, Latitude, Population, OceanProximityINLAND are negatively correlated with the value of the houses: this is quite reasonable since the more the houses are in the inland, the farther they are from the ocean and therefore the cheaper they are; moreover, the higher is the population's density the cheaper houses are, because they will probably be apartments in buildings or skyscrapers (and therefore smaller and without gardens) which clearly reduces the cost of them.
The other variables, instead, are positively correlated and even in this case the reason seems to be simple to understand, so let's analyze some feature:

- TotalRooms and TotalBedrooms: obviously, the higher the number of the rooms or bedrooms, the higher the value of the house (and probably the greater the square footage of the house).

- Households: probably, the higher the number of people residing in that neighborhood, the better will be that zone.

- MedianIncome: the higher the median income for households within a block of houses, the richer will be the neighborhood and therefore the better it will be for living.

- OceanProximity: the closer you get to the ocean, clearly the more the houses cost rises and consequently the estimated coefficient also rises.

Similar arguments could be made for the remaining variables.

4.3 LASSO Model with Cross-Validation

Now we can run the LASSO Model with cross-validation in order to understand if even in this case all the variables are selected in the best model, and whether the MSE associated to the logLambda's value that provides the minimum variance changes or not. With the help of the following graphs we will make the same motif as the one done for the Ridge Regression in the previous section.

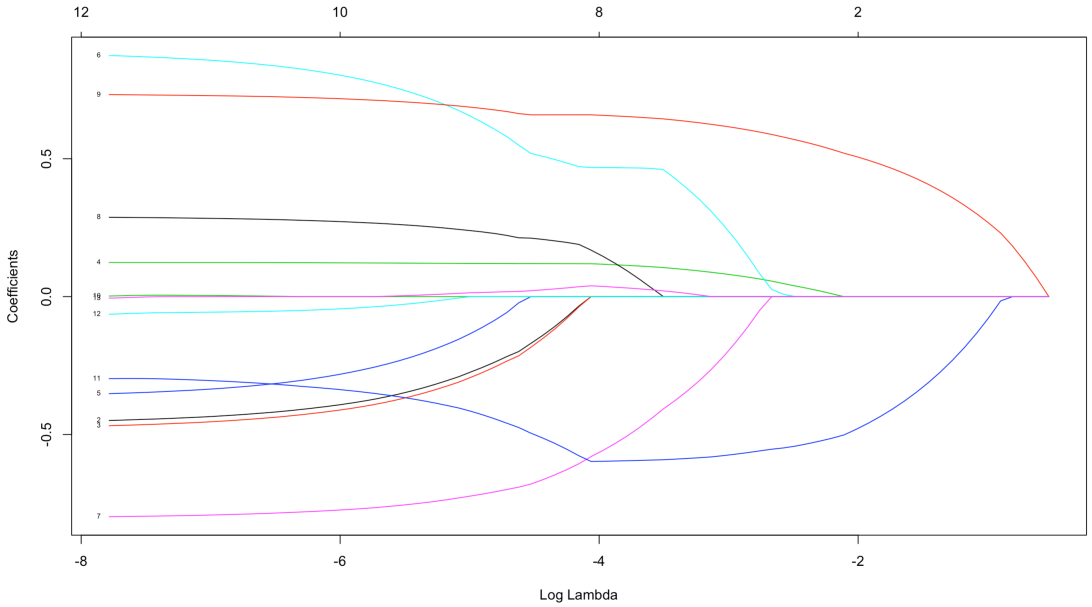

Figure 4.9: Variables' behavior over different values of logLambda

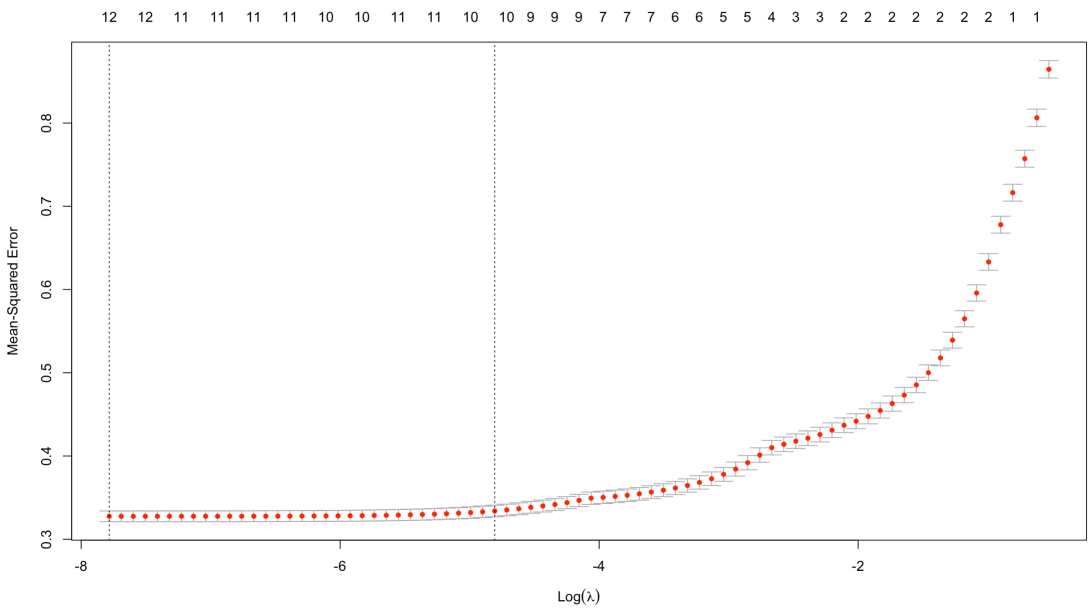

Figure 4.10: MSE behavior over different values of logLambda

Looking first at the figure 4.10 we can establish that the values of logLambda which provide the minimum MSE and the minimum variance are more distant than in the Ridge, but this time if we imagine again lengthening the most-right dotted line over the figure 4.9 we can see that some variables have already gone to zero. To be sure of that, let's see the logLambda values, the associated MSE and the estimated coefficients.

```
$`Value of log(lambda) providing minimum MSE`
[1] -7.783571

$`Value of log(lambda) providing minimum variance`
[1] -4.806491

$`MSE associated to the lambda which provides the lowest variance`
[1] 0.3340428
```

Figure 4.11: logLambda and corresponding MSE values

```
14 x 1 sparse Matrix of class "dgCMatrix"
                                       s1
(Intercept)                  0.14331873
(Intercept)                  .
longitude                   -0.23774430
latitude                    -0.25534101
housing_median_age           0.12003904
total_rooms                 -0.08569346
total_bedrooms               0.60694257
population                  -0.70820634
households                   0.22896443
median_income                0.67704435
`ocean_proximity_<1H OCEAN`  .
ocean_proximity_INLAND      -0.44319458
`ocean_proximity_NEAR BAY`   .
`ocean_proximity_NEAR OCEAN` 0.01644999
```

Figure 4.12: Coefficients of Lasso model

Analyzing the figure 4.12 we notice that the most important difference, with respect the corresponding figure for the Ridge, is that here two variables have been gone to zero (i.e. OceanProximity1HOCEAN and OceanProximityNEARBAY) and that now the feature TotalRooms is negatively correlated to the response variable. Moreover, the MSE provided by the Lasso model is

a little smaller than the one provided by the Ridge Regression and, then, it could be interesting to perform another kind of analysis, applying a method suitable for our case and compare all these models for having a clearer picture of the whole situation.

4.4 Partial Least Squares Regression (PLSR)

PLSR is a technique very useful when you have few observations relative to the number of predictors, or your predictors are highly associated with each other (like in our case), making a standard regression analysis problematic. Applying the Partial Least Squares Regression to the training set (which contains the 70% of the entire dataset) using MedianHouseValue as response variable, we get the following results:

```
Data:   X dimension: 12147 12
        Y dimension: 12147 1
Fit method: kernelpls
Number of components considered: 12

VALIDATION: RMSEP
Cross-validated using 10 random segments.
       (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps  9 comps
CV          0.9306   0.6613   0.6203   0.6056   0.5963   0.5862   0.5798   0.5751   0.5735   0.5732
adjCV       0.9306   0.6613   0.6203   0.6056   0.5962   0.5862   0.5798   0.5751   0.5735   0.5732
       10 comps  11 comps  12 comps
CV       0.5713    0.5713    0.5709
adjCV    0.5713    0.5713    0.5708

TRAINING: % variance explained
                    1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps  9 comps
X                     17.01    40.63    56.82    70.99    78.34    82.37    85.21    91.35    99.41
median_house_value    49.55    55.61    57.70    59.01    60.39    61.26    61.90    62.11    62.15
                    10 comps  11 comps  12 comps
X                      99.77     99.99    100.00
median_house_value     62.41     62.42     62.48
```

Figure 4.13: Summary of the PLS

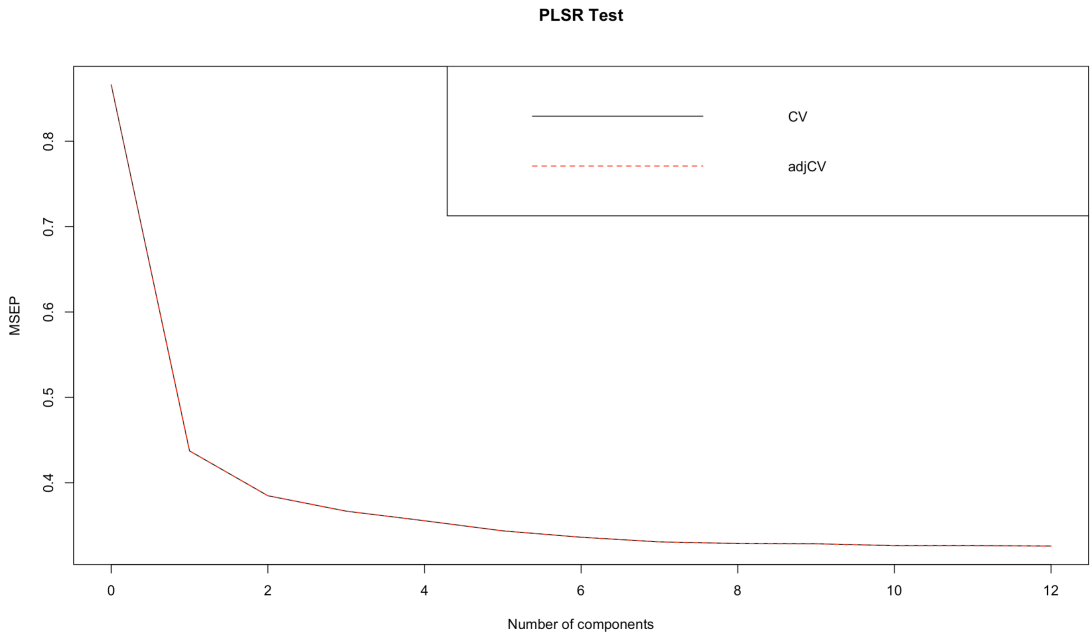

Figure 4.14: Plot of the CV's error of PLS

From the above figure 4.13 we can see that the CV error goes down until the 5th component while, after that, it stabilizes around a value of 0,57 with centesimal decreasing at each added component. So, we will take into account the first 5 components: with them we are able to explain the 78.34% of the overall variance and, moreover, we can also predict the 60.39% of the variance of MedienaHouseValue which is quite a good percentage. Thus, we have reduced the number of components from 12 to 5, using less than the half of the initial predictors but anyway being able to make good prediction. Finally, is also important to notice that the MSE associated with this model has a value equal to 0.6389637.

5 Choosing The Best Supervised Model

5.1 Comparison

So far we have evaluated four different models, e.i. Forward Stepwise Model Selection, Ridge Regression, Lasso Model and Partial Least Squares Regres-

sion, each of which has given us different results and different errors. So now the question is: which of them should we pick? Which is the "best" model? For answering to this question, we need to have a look at the following graph:

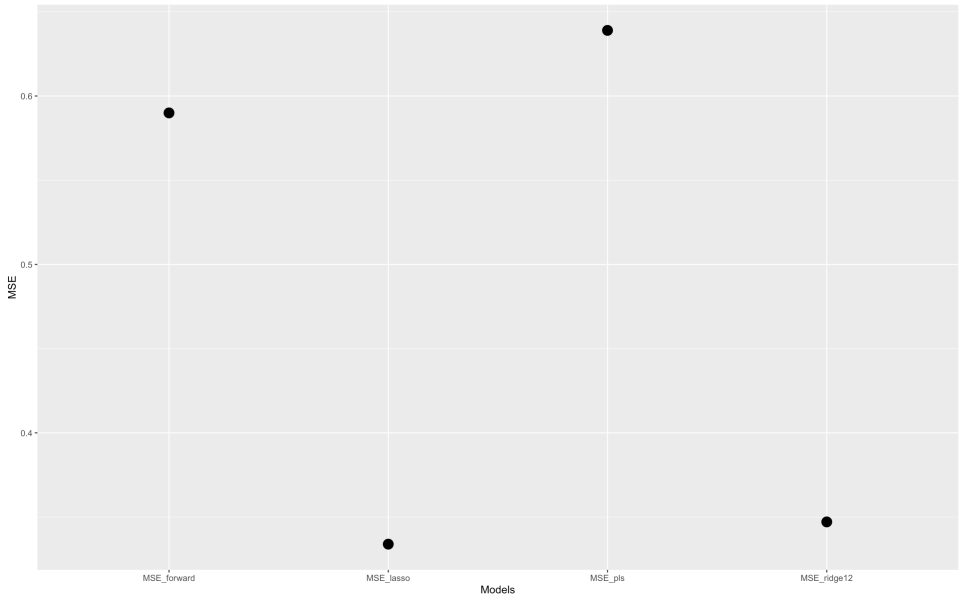

Figure 5.1: Mean Square Errors of the evaluated models

According with the figure 5.1, in which there is the comparison between the MSEs of the four different models, it seems that:

- Partial Least Squares Regression: it's the model that provides the highest MSE (e.i. 0.6389637) and then, it's the worst model among all.
- Forward Stepwise Model Selection: it provided an error equal to 0,5899694 and so, it's the third best model.
- Ridge Regression: it is the second best model with an MSE that is very close with the best one, and it's equal to 0,3471329.
- LASSO Model: it is the best model among all, because it's the one providing the lowest MSE which has a value of 0,3340428.

But, can we celebrate the LASSO model as the best one only looking at the MSEs?

6 Unsupervised Learning Analysis

This chapter will provide some unsupervised analyses, which will give us a brief and non-exhaustive view over the data. The main idea is that this kind of analysis should be performed BEFORE the evaluation and/or implementation of a supervised learning model, since the unsupervised one wants only to give a generic idea of how the data are distributed and if they are somehow correlated.

6.1 Principal Component Analysis

Since we are practicing an *Unsupervised* method like PCA, the response variable MedianHouseValue must be removed before proceeding. Secondly we need to turn all the variables into numerical one, otherwise we could have distorted results.

```
Classes 'data.table' and 'data.frame':  17353 obs. of  12 variables:
 $ longitude                 : num  -1.33 -1.34 -1.34 -1.34 -1.34 ...
 $ latitude                  : num  1.04 1.04 1.04 1.04 1.03 ...
 $ housing_median_age        : num  1.86 1.86 1.86 1.86 1.86 ...
 $ total_rooms               : num  -0.5352 -0.6235 -0.462 -0.7859 -0.0464 ...
 $ total_bedrooms            : num  -0.826 -0.719 -0.612 -0.771 -0.116 ...
 $ population                : num  -0.82 -0.765 -0.759 -0.893 -0.292 ...
 $ households                : num  -0.8434 -0.7335 -0.6289 -0.8016 0.0381 ...
 $ median_income             : num  1.7829 0.9329 -0.0131 0.0872 -0.1117 ...
 $ ocean_proximity_<1H OCEAN : num  0 0 0 0 0 0 0 0 0 0 ...
 $ ocean_proximity_INLAND    : num  0 0 0 0 0 0 0 0 0 0 ...
 $ ocean_proximity_NEAR BAY  : num  1 1 1 1 1 1 1 1 1 1 ...
 $ ocean_proximity_NEAR OCEAN: num  0 0 0 0 0 0 0 0 0 0 ...
```

Figure 6.1: Type of dataset's variables

Now that we have reached a situation like the one described above the fig. 6.1, we can apply the PCA analysis to our data, getting the following result.

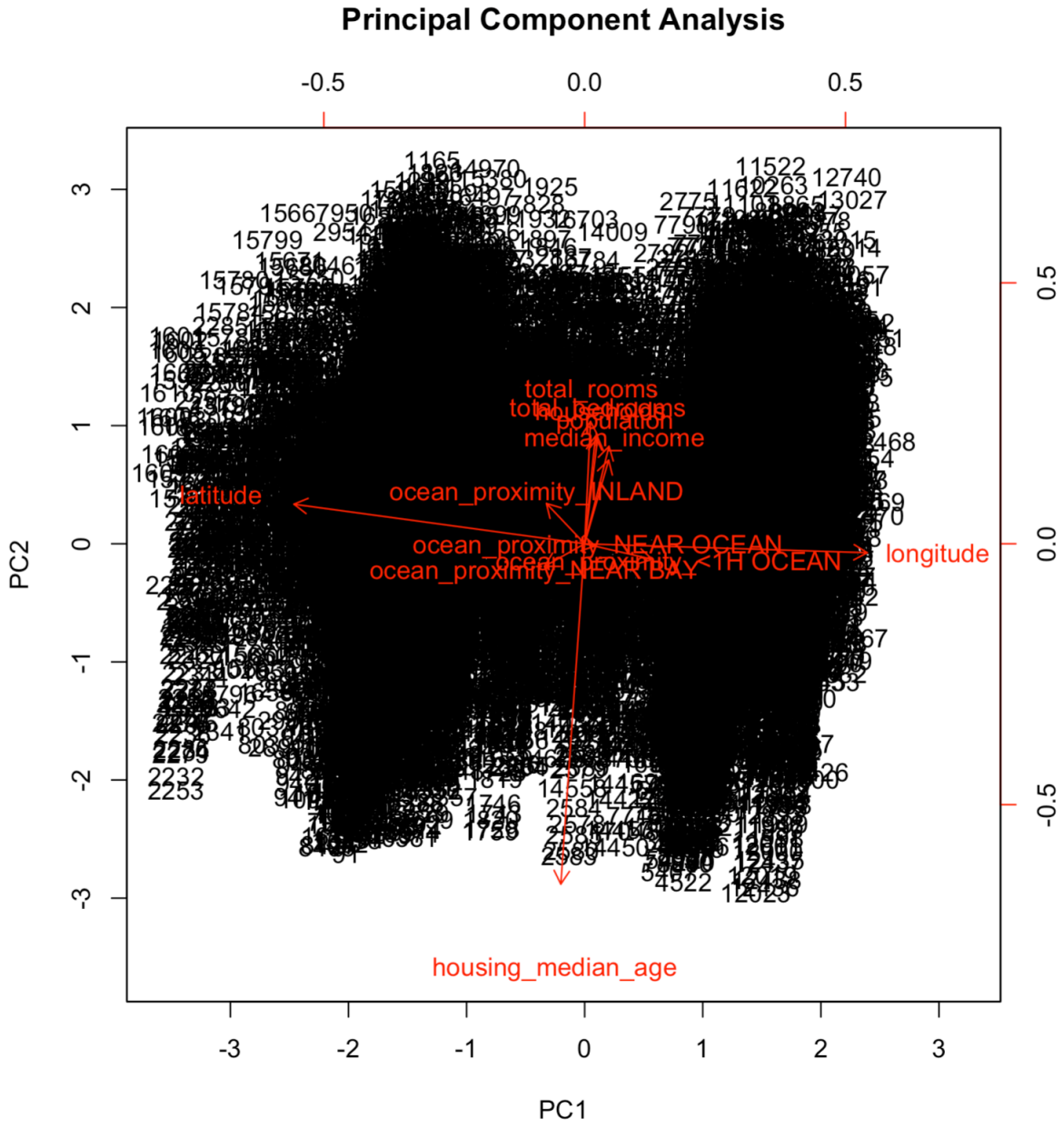

Figure 6.2: PCA analysis

This biplot shows the observations as points in a plane formed by the first two principal components (PC1 on the horizontal axes and PC2 on the

vertical one). In addition to the observations, the plot shows the original variables as vectors (red arrows). They begin at the origin [0,0] and extend to coordinates given by the loading vector. These vectors can be interpreted in three ways:

- The *orientation* (direction) of the vector, with respect to the principal component space, in particular, its angle with the principal component axes: the more parallel to a principal component axis is a vector, the more it contributes only to that PC.
- The *length* in the space: the longer the vector, the more variability of this variable is represented by the two displayed principal components; short vectors thus don't contribute too much.
- The *angles* between vectors of different variables show their correlation in this space: small angles represent high positive correlation, right angles represent lack of correlation, opposite angles represent high negative correlation.

Then, observing the graph 6.2 we can infer that the first principal component (PC1) is mostly dominated by Longitude and Latitude which are strongly negatively correlated with each other, when instead the second principal component (PC2) is mostly dominated by HousingMedianAge which is negatively correlated with a group of variables, e.i. TotalRooms, TotalBedrooms, Population, Households, MedianIncome which, moreover, are highly positively correlated with each other. Another observation is that the features that mostly contribute to PC1 and PC2 are nearly uncorrelated with each other, since they form almost right angles with the variables associated with the other component. For example, HousingMedianAge is almost perpendicular to both Latitude and Longitude, as the other group of variables aforementioned. For what concerns the group of variables representing the position of a house with respect to the ocean, we can see that the length of their representative vectors is small and, then, they don't contribute so much to the variability of the dataset.
We also aim to find the components which explain the maximum variance: this is because, we want to retain as much information as possible using these components. So, higher is the explained variance, higher will be the information contained in those components. To compute the proportion of variance explained by each component, we simply divide the variance by the sum of the total variance and it yields to:

```
[1] 0.393044240 0.215596125 0.139031786 0.126432424 0.053370637 0.036120191 0.017572415
[8] 0.009996057 0.004619427 0.003086275
```

Figure 6.3: Proportion of variance explained by the first 10 components

Figure 6.3 shows us that first principal component explains 39.3% of the overall variance while the second the 21.56%. So, how do we decide how many components should we select for modeling stage? The answer to this question is provided by a scree plot. A scree plot is used to access components or factors which explains the most of the variability in the data. It represents values in descending order.

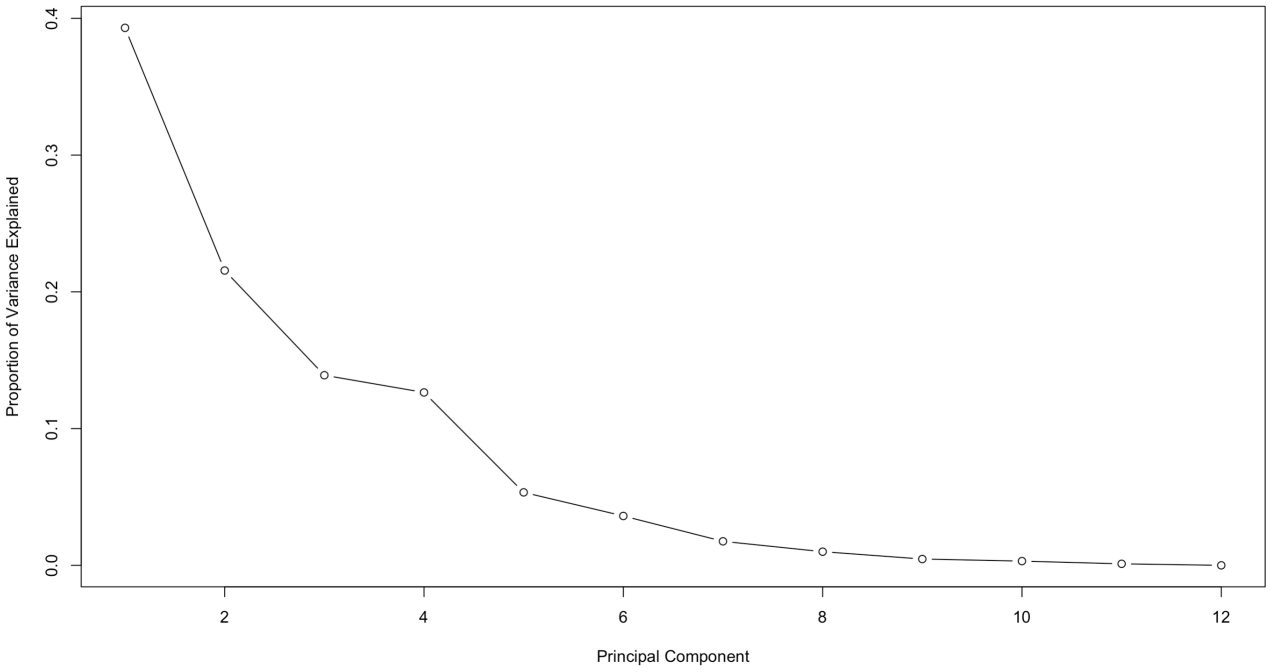

Figure 6.4: Plot of the proportion of variance explained by components

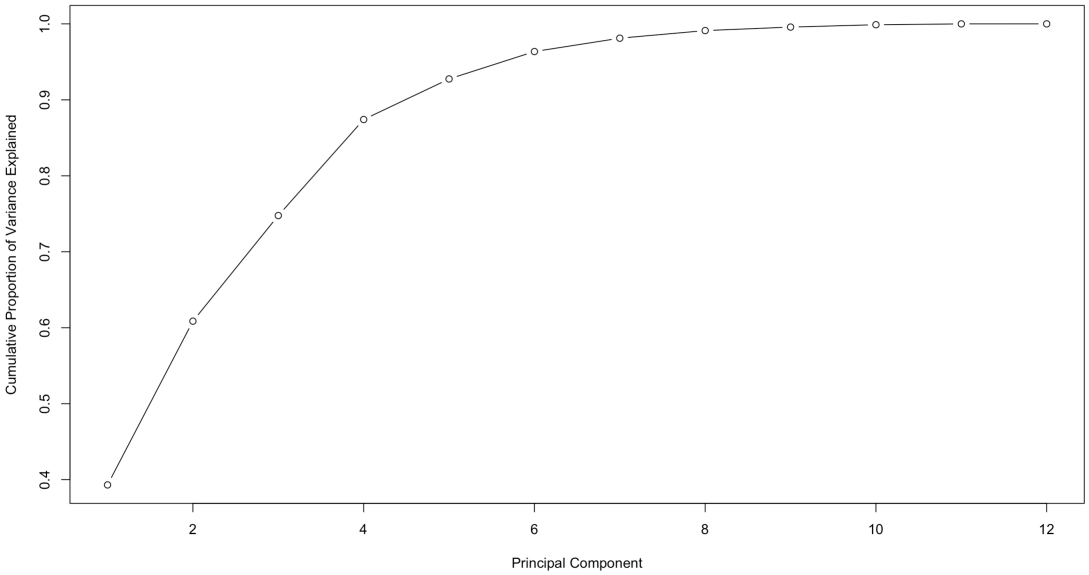

Figure 6.5: Plot of the cumulative proportion of variance explained by components

So, from the two above plots we can assess that with the first 2 components we should be able to explain more than the 65% of the variance (which is a good proportion) and that from the 6th component (included) on, the increase in the variance explained adding a new component is very small, since with the first 6 components nearly the 95% of the variance is already explained and then, following this reason, it does not make so much sense to take more than 5 components. In conclusion, as said in the subsection "1.5.2 Proportion of Variance Explained (PVE)", for deciding how many components we should take, we need to look for a kink in fig. 6.4, which in this case could be either when there are 3 components or 5, with which we would be able to explain, respectively, around 75% and 90% of the variance. Obviously, the lower the number of components chosen, the simpler will be the model, but the more similar to the original one it will be.

6.2 K-means Clustering

In the subsection 1.6.2 we have already said that one of the disadvantages of the K-means algorithm is the fact that we must now the number of clusters K a priori: this is a problem that we need to overrule somehow. In order to that,

a Within group Sum of Squares (WSS) plot has been implemented. This is a kind of plot where we have on the horizontal axis the optimal number of the k clusters while on the vertical axis we have the WSS which helps us to measure the within clusters variation. Then, if:

- the WSS is high, then also the variation within a cluster is high. in In this case, the extreme scenario is when k=1, where we have the highest *within* group variation and the lowest *between* group variation.
- the WSS is low, the variation within a cluster is low. In this case, the extreme scenario is when k=n (n is the number of observations), where we have the lowest *within* group variation and the highest *between* group variation.

Clearly, our aim is to find an optimal balance between reducing the *within* group variation and increasing the *between* group variation by choosing an optimal number of clusters k. To do that, we need to look for a kink in the WSS plot:

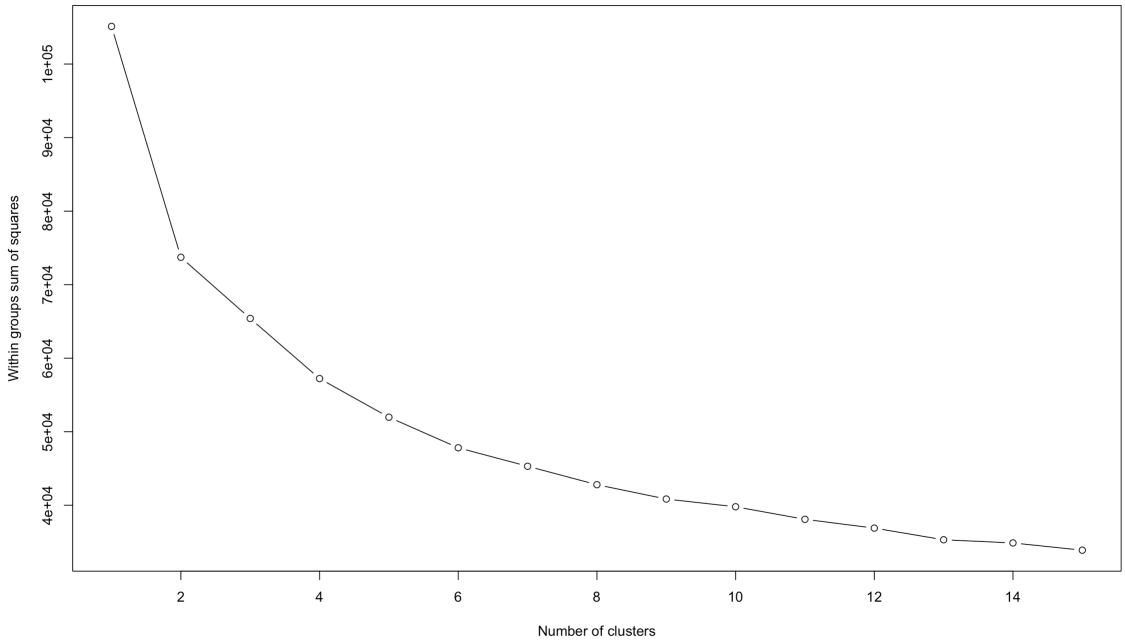

Figure 6.6: Plot of the Within group Sum of Squares

Analyzing the figure 6.6 it's rather easy to see that the elbow to which we are looking for is in correspondence of k=2 and, then, we will perform a 2-means cluster analysis. Recalling that our dataset is multi-dimensional, we need to perform the k-means after having reduced the dimensionality through a PCA. The result is:

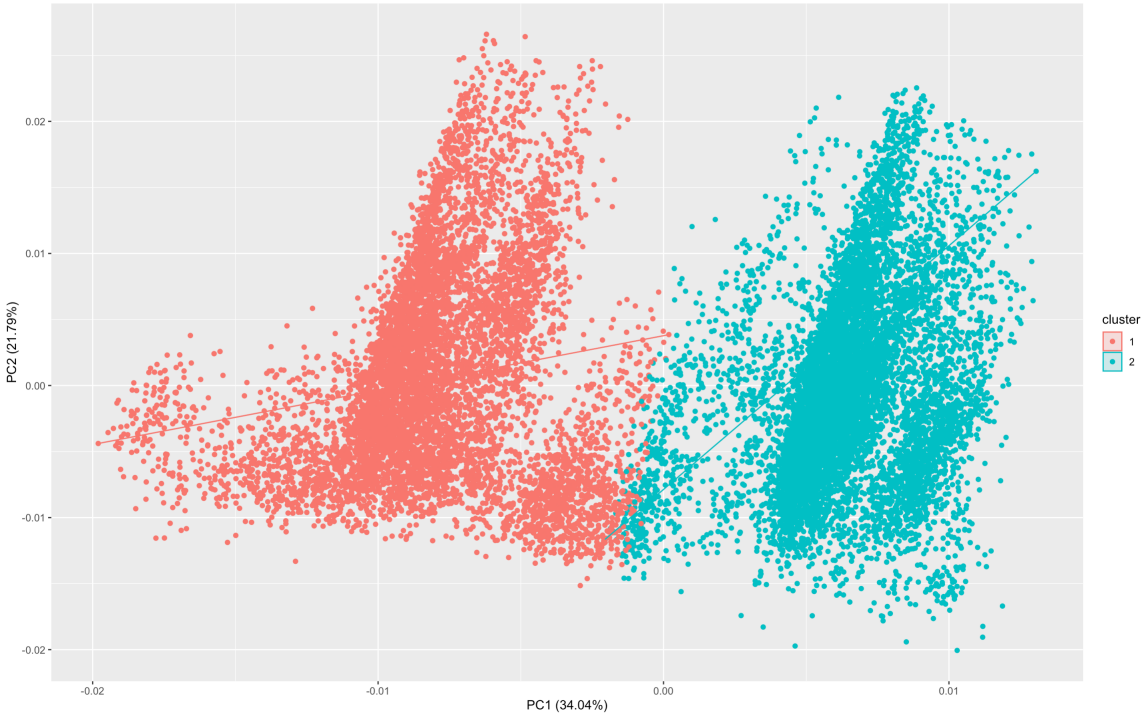

Figure 6.7: 2-Means Analysis

The two clusters appear to share a similar shape regarding the distribution of points in each group. Also, we can see that in the middle of the plot there is a small overlap of the two groups Let us have also a look at the k-means centers:

```
   longitude  latitude housing_median_age total_rooms total_bedrooms population households median_income median_house_value
1  0.7869791 -0.815257         0.11371733  -0.2616483     -0.2475483 -0.1997753 -0.2406125   -0.07343538         0.01041159
2 -1.0254673  1.075350         0.08566134  -0.2466658     -0.2814031 -0.3160699 -0.2869560   -0.17435922        -0.21194494
  ocean_proximity_<1H OCEAN ocean_proximity_INLAND ocean_proximity_NEAR BAY ocean_proximity_NEAR OCEAN
1                 0.6460417              0.1957292                0.0000000                0.15770833
2                 0.1665162              0.4901329                0.2499678                0.09338321
```

Figure 6.8: 2-Means Centers

From figure 6.8 we can know the average value of each feature in each of the two clusters (e.g., the first row corresponds to the average values of each variable in the first cluster). In particular, the centers for the variables TotalRooms, TotalBedrooms and Households are very similar in both the clusters, so the overlapping points could arise from these features and, therefore, the two clusters are not "distinct by nature".

7 Conclusion

Throughout this paper we have seen applied different techniques of two main methods: *Supervised* and *Unsupervised* methods. In a real-world situation, the *Unsupervised* one should be applied first, to make ourselves a general idea of the distribution of the data points and of the eventual relationships existing between them. Through PCA we have been able to understand some relationships (in terms of correlation) among the variables, plotting them into a 2-dimensional space without loosing too much informations. On the other hand, using K-means clustering we have seen that the data points can be grouped in two non-distincted clusters, which suggests us that some variables could be redundant, such as for example TotalRooms and TotalBedrooms (it's not difficult to think that probably the total number of bedrooms is an information already contained into the TotalRooms variable). Then, we have also seen four different supervised learning models, we have evaluated and implemented them over our data in order to understand which would be the best predictive one, but before declaring the Lasso Model as the best one, it's important to make few considerations comparing it with the Ridge Regression, given that their MSEs are very close.
The Ridge needs the all 12 variables to be properly implemented and then, for making predictions, it needs as input a file containing al lot of specifics about a house in California: the latitude, the longitude, the median age of a house within a block, the total number of rooms within a block, the total number of bedrooms within a block, the total number of people residing within a block, and so on. These are a lot of informations and it's not unreasonable thinking that to provide all of them is unusual or al least unlikely, but at the same time, they are necessary for a complete and correct prediction using this model. Of course, it would be better if we had a model that needs less informations for making a prediction at least as good as the one provided by the Ridge. Fortunately, this is exactly the case of the LASSO model: as seen

in the figure 4.12 the it needs only 10 variables for making a prediction with an associated MSE even lower than the one provided by the Ridge. With LASSO, we don't need to specify the OcenProximityLess1Hour and OcenProximityNearBay, which means that we only need to know if a house is in inland or barely near the ocean.
In conclusion, LASSO is the best model for predicting the value of an house in California for two essential reasons:

1. The MSE provided is the lowest one with respect to all the other models that have been implemented.

2. It needs a less strict description of the houses, e.i. it uses less variables and then it's needed to provide less features about the house (which clearly makes life easier in a concrete real-world situation).

8 Appendix

The R codes through which the analyses have been performed are provided in the following link (GitHub):
https://github.com/AndreaFrattini/Statistical-Learning/blob/main/House_Cal.R