Predicting sales of insurance policies

Pierre-Louis Guillot

February 2020

Chapter 1

Exploration

This chapter is dedicated to exploring the data set and explaining some of the observation that were made before trying to build a model. The first section will be dedicated to analysing the raw data set, the second one will be dedicated to transformations that were made to the data set in order to build our models, and the third section will describe how models should be evaluated.

1.1 Analysis of the data set

The data set consists of $n = 50000$ observations of $p = 12$ predictors all labeled with either 0 or 1 depending on if the potential policy holder bought the policy or not. The purpose of this work is to create a machine learning model able to predict, for new customers outside of the data set, whether or not they will buy the policy. The model's prediction can only rely on the value of predictors for the corresponding unlabeled customers.
The fact that the number of observations is very large and the number of predictors relatively small should be kept in mind. It suggests that more complex models – prone to overfitting on data sets with fewer observations and/or more predictors – might work well.
One should also consider missing values. As reported in table 1.1, there are around 1% of missing values for each predictor. However there is a total of 5180 observation (10.36% of n) for which at least one of the predictor has a missing value. This means that a significant number of observations are incomplete and missing values should be dealt with properly, as discussed in further sections. This should not impact the overall quality of the data set as there are still many (44820) observations with complete information.

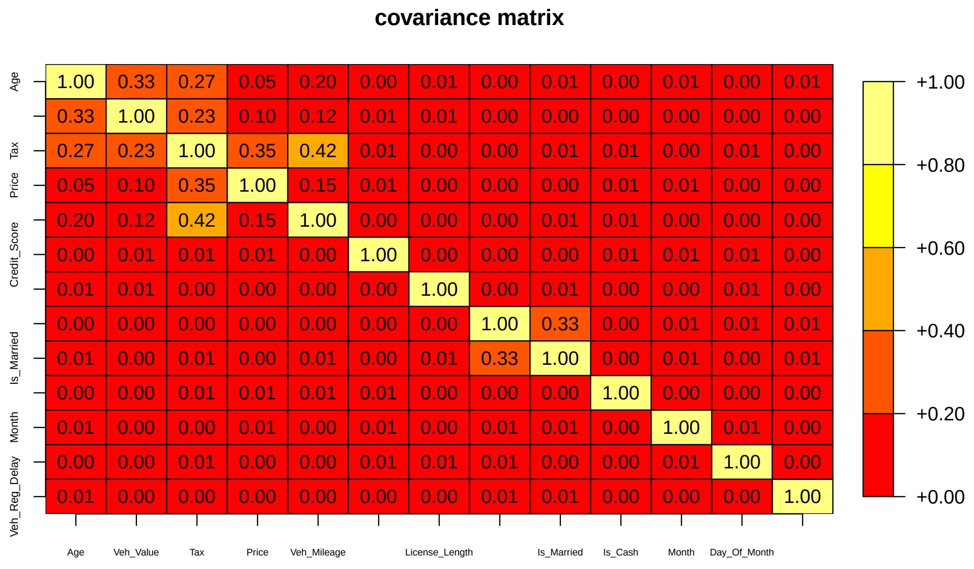
We should also mention that the proportion of observations for which the corresponding policy was sold is quite high. It accounts for 69.8% of the data set. It is worth wondering if this proportion is representative of what we can expect on data sets the classifier is going to be used on. Maybe cases for which the policy was not sold tend

Age	Vehicle Value	Tax	Price	Vehicle Mileage	Credit Score
0.966%	1.050%	1.034%	1.030%	1.018%	0.994%

License Length	Date	Marital Status	Payment Type	Vehicle Reg Year
0.996%	0.994%	0.966%	0.968%	0.976%

Table 1.1: Percentage of missing values for each predictor

to not be registered in the database, depending on the moment of the process at which policies are registered. This could be problematic as proportion of data sold are used in certain models such as discriminant analysis. An exaggeratedly high proportion of observation labeled as sold may give a too high probability that any given observation is predicted as leading to a sale. However, without additional information, we assumed that the proportion of policy sold is representative of the data sets on which the predictors will need to perform.
After turning the data into numerical values (cf further sections), we looked at the correlation matrix of the data set, as reported in figure 1.1.



1.00	0.33	0.27	0.05	0.20	0.00	0.01	0.00	0.01	0.00	0.01	0.00	0.01
0.33	1.00	0.23	0.10	0.12	0.01	0.01	0.00	0.00	0.00	0.00	0.00	0.00
0.27	0.23	1.00	0.35	0.42	0.01	0.00	0.00	0.01	0.01	0.00	0.01	0.00
0.05	0.10	0.35	1.00	0.15	0.01	0.00	0.00	0.00	0.01	0.01	0.00	0.00
0.20	0.12	0.42	0.15	1.00	0.00	0.00	0.00	0.01	0.01	0.00	0.00	0.00
0.00	0.01	0.01	0.01	0.00	1.00	0.00	0.00	0.00	0.01	0.01	0.01	0.00
0.01	0.01	0.00	0.00	0.00	0.00	1.00	0.00	0.01	0.00	0.00	0.01	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.33	0.00	0.01	0.01	0.01
0.01	0.00	0.01	0.00	0.01	0.00	0.01	0.33	1.00	0.00	0.01	0.00	0.01
0.00	0.00	0.01	0.01	0.01	0.01	0.00	0.00	0.00	1.00	0.00	0.00	0.00
0.01	0.00	0.00	0.01	0.00	0.01	0.00	0.01	0.01	0.00	1.00	0.01	0.00
0.00	0.00	0.01	0.00	0.00	0.01	0.01	0.01	0.00	0.00	0.01	1.00	0.00
0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.01	0.00	0.00	0.00	1.00

Figure 1.1: Correlation matrix of the data

It looks like the different predictors aren't that correlated except for a few instances. We also looked at the proportion of explained variance for each consecutive linear subspace generated by pca, as reported in figure 1.2. There seem to be **very little redundancy in the data**, at least as far as axes of minimal variance go. The curve seems to be very close to the identity curve, which means there is no small set of linear combination of predictors that can account for most of the variance in the data set.

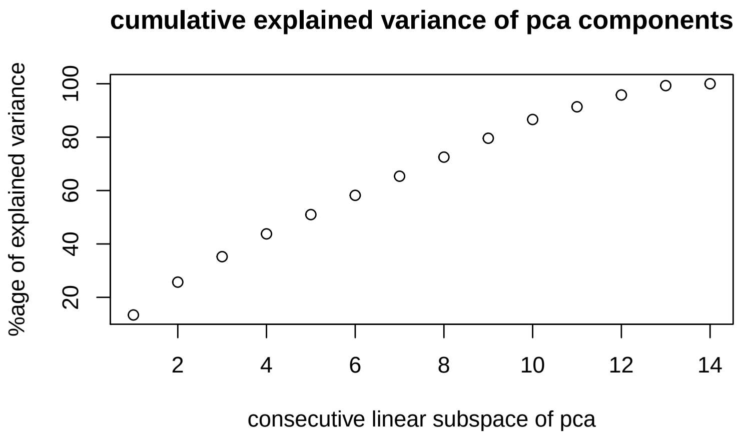


Figure 1.2: Cumulative variance of pca's principal components

1.2 Analysis and transformation of predictors

Model buildings techniques often rely on numerical values of predictors, which require transformations of the data set. Moreover, we would like to analyse the shape of the data and see if we can find any interesting pattern. Several predictors are numerical and do not need to be to be transformed, such as *Age*, *Vehicle Value*, *Tax*, *Price*, *Vehicle Mileage* or *License Length*. Their marginal distributions within each of the two class (*sold*, and *not sold*) were plotted as represented respectively in figures 1.3, 1.4, 1.5, 1.6, 1.7, 1.8. We observe that the predictors seem to be smoothly distributed within each class. More precisely, the shapes of their density very much looks like the shape of a Gaussian curve. One could infer that, considering only those predictors, the data set follows a gaussian mixture model in which the values of predictors within a class are normally distributed. This hypothesis will be very important for methods such as discriminant analysis and we will keep that in mind when building a model.

Moreover, we observe that the mean of the distributions of *Price* and *Age* differ significantly from one class to another : they might play an important role in knowing whether or not a policy is sold. Within-class marginal distributions of *Tax*, *Vehicle Mileage* and *Vehicle Value* seem to have similar mean but different variance, which can help discriminating as well. *License Length* has a similar marginal distribution within both classes but it does not necessarily imply that it is useless for discriminating. Those marginal distributions are merely partial representations of the interaction between a predictor and the response variable.

Credit Score is a special case of numerical predictor. As represented in figures 1.9 and 1.10, it seems to be set to 9999 for exactly 1% of the observation, and normally distributed around 500 for the remaining 99%.

9999 seems to be an arbitrary large value. A definitive statement would require to look

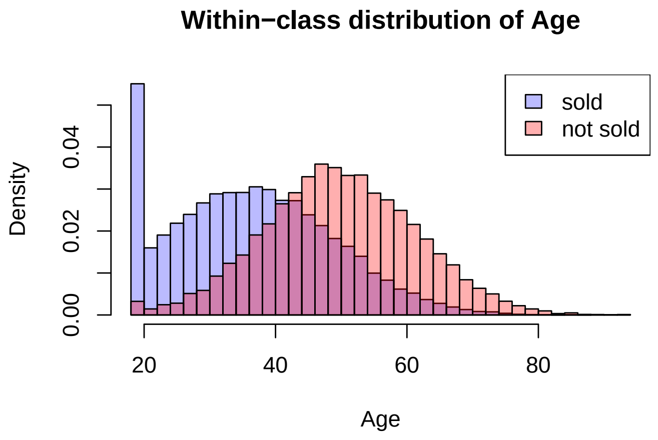


Figure 1.3: Within-class distribution of age

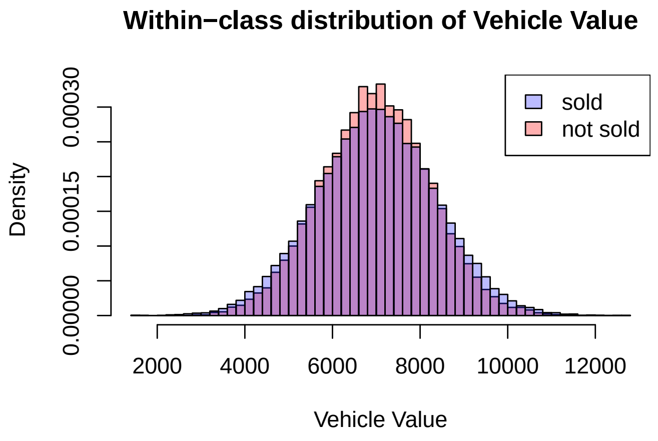


Figure 1.4: Within-class distribution of Vehicle Value

at how the credit scores are actually computed but our hypothesis is that is the number itself is arbitrary and represents an infinitely high value. In that context, the value of 9999 would not make much sense in itself, which could cause problem for models such as linear models. With the assumption that 9999 is arbitrary, should the difference between an observation with 9999 credit score and an observation with credit score 999 be exactly 18 times as important as the difference between observations with 999 and 499 credit scores ? Probably not. We tried to account for this in our models but none of the steps in that direction really worked. This is probably because credit scores equal to 9999 are very few and barely have an impact on the building of models.

Marital status and *Payment Type* are unordered categorical predictors. A way to transform them into numerical variables is to use *dummy variables*. It works well when

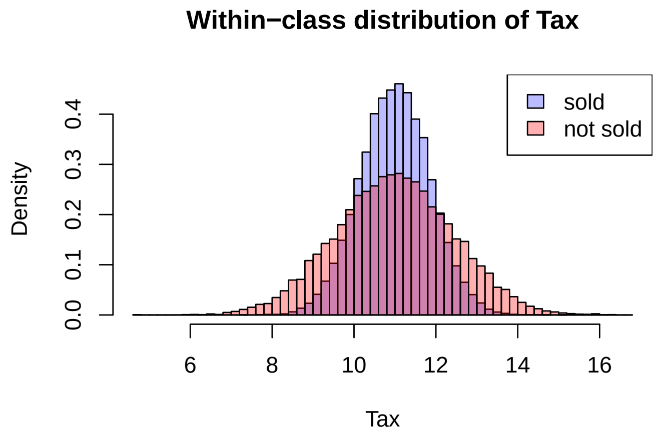


Figure 1.5: Within-class distribution of Tax

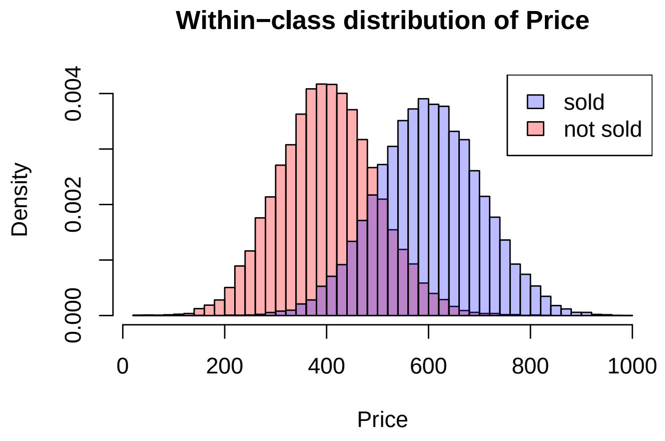


Figure 1.6: Within-class distribution of Price

there are few classes – which is the case here with respectively 3 and 2 classes.

Date and *Vehicle Registration Year* are two temporal predictors that need to be discussed. One of the most important thing to consider is that **every observation in the data set correspond to a policy that starts in 2016**. This is probably the most important observation in the data. We assume that the model we are trying to build will be used to predict data for policies that don't necessarily start in 2016. A common way of turning a date into a numerical value is to represent it as the number of days since an earlier date. However this might be problematic for generalizing outside of 2016. Let's consider the case of simpler models. They might detect patterns showing that sales decrease at the end of 2016, and deduce that the sales will be lower in 2017. Is this a good deduction, or are the variation of sales in 2016 due to seasonal variations

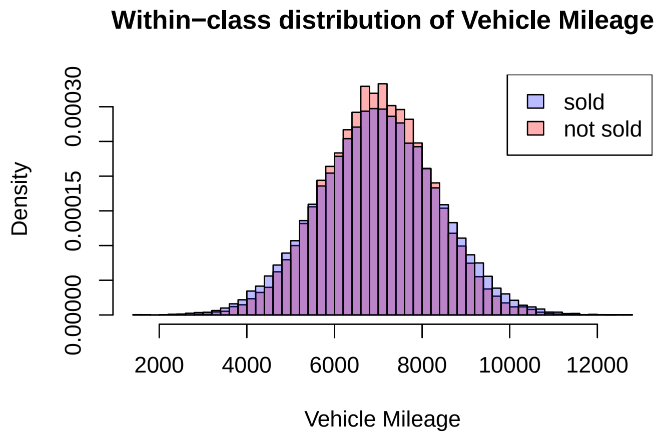


Figure 1.7: Within-class distribution of Vehicle Mileage

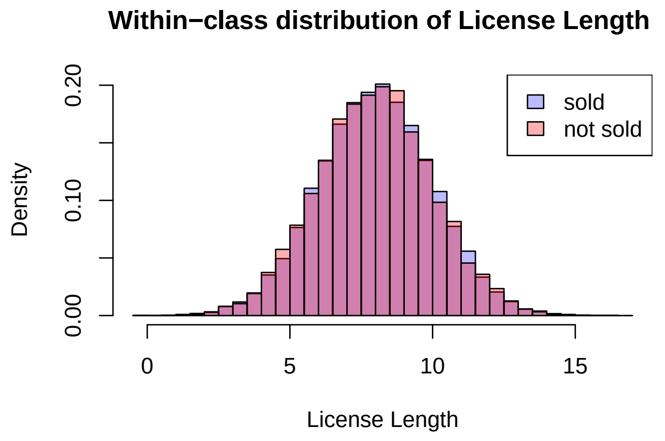


Figure 1.8: Within-class distribution of License Length

(lower sales in the end of the year in that case) rather that trends that extend throughout the years ? The hypothesis here is that former is a more important factor. But what about more complex models ? They would do even worse, considering that complex models tend to have unpredictable results outside of the boundaries of the data they are trained on. Dates outside of 2016 will be out of the interval on which the complex model was trained and it is pretty much bound to generalize poorly. Our solution to both problem is to rather consider an integer in $\{0, 1, \dots, 365\}$ corresponding to the number of day since the first of January. An issue with this encoding is that it will be hard for smoother models to capture the day of the month of the corresponding date. . We tried to account for that and transform the data so that it can include the day of the month but it didn't have any significant impact, probably because the date of start

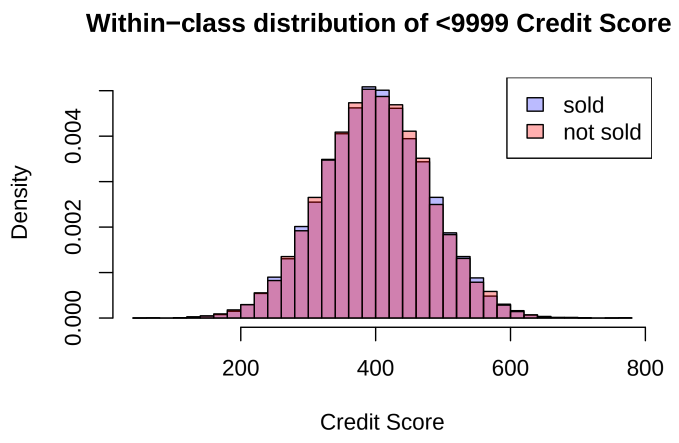


Figure 1.9: Within-class distribution of Credit Scores ¡ 9999

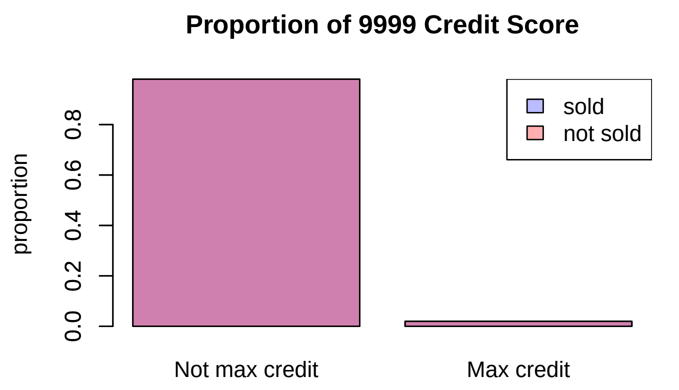


Figure 1.10: Within-class proportion of Credit Scores equal to 9999

isn't the most impactful factor for whether or not a policy is sold.
NOTE : Transforming the date into an integer in $\{0, 1, \dots, 365\}$ only matters when the training is over and our model is applied to new instances outside of 2016. It should not impact results such a classification accuracy, since all the observations we are going to test our model with also are in 2016.
A similar reasoning can be made for *Vehicle Registration Year*. It can be decomposed as the difference between the year at which the policy starts and the delay between the registration date and the start of the policy. In a similar fashion, because all the policies were sold in 2016, our machine learning model may interpret interaction between how much time passes between the registration and the policy as trends that only depend on the year. Another way to formulate this distinction is : is the fact that there are proportionally a little more sold than unsold policies for vehicle registered in 2012 due to the year itself or the fact that those vehicles happen to be 4 years old ? Our hypothesis is that the age of the vehicle is a much more important factor. Therefore for generalization purposes in other years than 2016, the variable will be transformed into *Vehicle Registration Delay.*

1.3 Dealing with missing values

1.4 Model evaluation

The most straightforward metric for measuring a classifier's fitness of the data is the miss-classification rate. It will be use thoroughly to evaluate whether or not a model is good at properly discriminating observations it has not been trained on. However, depending on the usage of the classifier, false negative rate might also be worth considering. Assuming the classifier is used to detect if a policy is worth being presented to a customer a not, missing a sale might be more regretful than contacting a client for no reason and we might want to have a conservative approach of the problem. In that case, the best thing to do would be to find a model with both low miss-classification rate and low false negative rate. However, without further information on the usage of the classifier, we used the miss-classification rate as an indicator of performance for classifiers.
We also used cross-validation, a rigorous method for sampling the data set and measuring the fitness of a model. Cross-validation is an efficient way of measuring the generalization power of a method when the data set on which it is made is similar to data it will be subject to in the future. However, each observation the models will be tested on corresponds to a policy starting on the year 2016. Very complex models might overfit by recognizing patterns that only ever occurs during that year, and it would not be captured by cross-validation. This could lead to us choosing a model that has low cross-validation error but that doesn't generalize well to other years. For that reason, we might consider choosing a model with a slightly higher cross-validated miss-classification rate if it is simpler, has less parameters and is less prone to overfitting.

Chapter 2

Model building

This section will be dedicated to explaining the building of our machine learning model. In the first two sections we will describe the different techniques we tried to use, how we tried to improve them, and how successful they were. In the third and final section, we will chose the most fit classifier and interpret the results.

2.1 Discriminant analysis

Discriminant analysis are a set of techniques for classification that rely on **modeling the distribution** of the data. Common discriminant analysis techniques assume that the data is a mixture of *gaussian distributions*, in the sense that each class is normally distributed. Once the distribution is restricted to the family of Gaussian distributions, its parameters can be estimated by the value that maximizes the corresponding likelihood function. The idea behind discriminant analysis classification is to compare the density of the estimated distribution according to each label and taking the label whose corresponding density is the highest.
The assumption that predictors are normally distributed seems pretty fair as it matches the observations we've made when exploring the data.

2.1.1 LDA, QDA, Naive Bayes

Each of the two classes are normal distributions and the parameters of the joint distribution are Σ_{Sold} and $\Sigma_{\text{Not Sold}}$ their covariance matrices, μ_{Sold} and $\mu_{\text{Not Sold}}$ their mean and π_0 and π_1 the probability of an observation belonging to each class. These parameters are estimated by taking their maximum-likelihood estimate (respectively the empirical covariance matrices, empirical mean and proportion of observations within the class). *LDA*, *QDA* and *Naive Bayes* are three common discriminant analysis techniques that simply rely on different assumption on the distribution. QDA does not make any additional assumptions, LDA assumes that there is only one covariance matrix and $\Sigma_{\text{Not Sold}} = \Sigma_{\text{Sold}}$, and naive bayes assume that there is only one covariance matrix and that is is diagonal. Cross validated miss-classification rate according to these three techniques are represented in table 2.1.

naive bayes	LDA	QDA
7.166%	2.204%	1.187%

Table 2.1: Discriminant analysis methods miss-classification rates

As mentionned in our exploration in the data, the class of an observation seems to have an impact on the variance for some of the predictor's marginal distribution. It is not surprising that QDA performs better than LDA in that context, as only the former considers a difference of covariance between the two classes. Moreover, we also mentionned that n is very large and p quite small, and that more complex models with more parameters might tend to perform better because overfitting is less of a problem. QDA also is a more complex extension of LDA with more parameters to estimate.
As there seem to be very little correlation between predictors (1.1), we could expect naive bayes to perform better than LDA. However, even though this correlation is small it seems to be significant and with such a large number of observation it is not surprising that LDA performs better than its simpler version.

2.1.2 Other attemps at discriminant analysis

Because there seem to be little correlation between predictors, we tried to implement a mix of naive bayes and QDA we called *naive QDA*. The assumption here is that each class had its own covariance matrix but that they are diagonal. The corresponding classifier has a similar cross-validated missclassification rate than naive bayes (**8.04%**). It performs much worse than QDA, probably for the same reasons that naive bayes performs worse than LDA : the correlations are small but they exist, and the more complex a model the better. However it also performs worse than naive Bayes. Our only explanation is that since the required hypothesis are wrong for both classifiers, their behaviour is unpredictable.

We also tried to tackle the issue of 9999 credit scores by training two classifiers depending on if the value of credit score is maxed out or not, so that the assumption of normality of the distribution always hold. It performed a bit worse (**2.22%** for LDA and **1.49%** for QDA), probably because the classifiers were trained on smaller samples. There are probably much better ways of dealing with credit scores but any steps we made in that direction seems to have no impact of the missclassification rate, and credit score seems like it has little impact on the sale of a policy anyway.

2.2 Other methods

2.2.1 Logistic Regression

Contrary to discriminant analysis that models the joint distribution of the data, Logistic regression is a **discriminative** algorithm that only models the posterior distribution of the label according to all predictors. The assumption here is that it this distribution depends on a linear combination of predictors put into a logistic function. More precisely, it tries to estimate a parameter β such that :

$$\mathbb{P}(\text{Sale} = 1 | X = x) = \frac{1}{1 + \exp(<\beta, x>)}$$

With X the vector of predictor values. It is a pretty simple and smooth model as there aren't many parameters to estimate and it is a function of a linear combination of values of predictors.
The cross-validated miss-classification rate of the corresponding classifier is **1.84%**. It performs better than LDA even though it only requires to estimate p parameters (as opposed to $\mathcal{O}(p^2)$), and much better than naive bayes that has a similar number of parameter ($2p$). It is common for logistic regression to perform better than discriminant analysis when the hypothesis of the latter are not met. However, QDA performs better than logistic regression, either because the hypothesis are quite true or because the model is more complex and makes more use of the great number of observations.

2.2.2 KNN

KNN is a simple non-parametric technique that classifies an observation as the most common label among its K Nearest Neighboors. The value of K is an hyperparameter that needs to be tuned by cross-validation, as reported in figure 2.1
$K = 5$ seems to be the best value of K, and its cross-validation miss-classification rate is around **3.41%**. It performs worse than LDA, QDA and logistic regression. This might be because their assumptions matches the data set well or because KNN is a pretty simple technique that might not make use of the great number of observations in the data set.
NOTE : Further investigations could be made for KNN with other ways of defining distance than the euclidean distance.

2.2.3 SVM

Without going too much into details, SVM classification models are much more complex than all the previously mentioned models. They rely on finding an hyperplane that **optimally** separates the data into two classes, in the sense that its distance to observations is maximal. The simple linear version of SVM only has one hyperparameter : the proportional cost of miss-classified observations as opposed to correctly classified observations when computing the distance. However, more complex versions of SVM can be

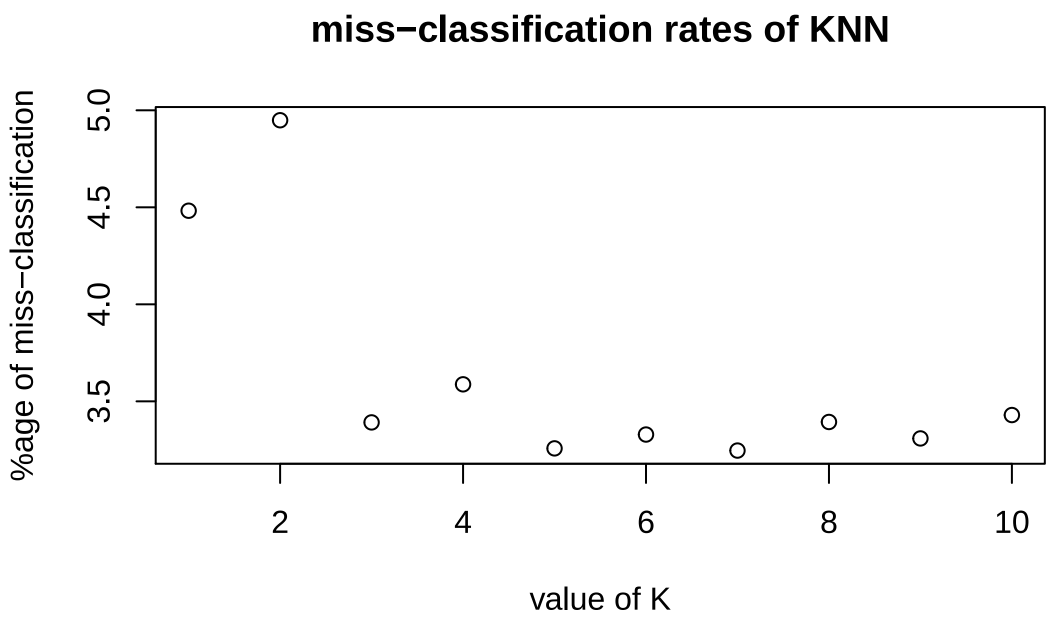


Figure 2.1: Tuning of KNN according to K

defined by using alternative versions of the dot product. We considered three categories of SVM classifiers and for each of them we tried to find the best hyperparameters.

Linear kernel

Using cross-validated miss-classification rate, we tuned the only hyperparameter of linear kernel svm classification, as reported in figure 2.2.

The optimal cost is around 1 with **1.66%** miss-classification rate. Even though it still performs worse than QDA, it is a pretty good result, which is promising since linear svm classification is one of the simplest form of svm classification.

Polynomial kernel

Polynomial kernel svm classification replaces the value of the dot product with $(\gamma < x, y > +\text{coef}_0)^d$. The tuning of γ, d, coef_0 and cost the 4 hyperparameters was pretty chaotic and will not be explained in details here. The best set of hyperparameters we found after a bit of exploration were $\gamma = 1$, $d = 2$, $\text{coef}_0 = 1$, and cost $= 1$ with a miss-classification rate of 1.30%. Even though it is a very promising result, it still doesn't outperform QDA.

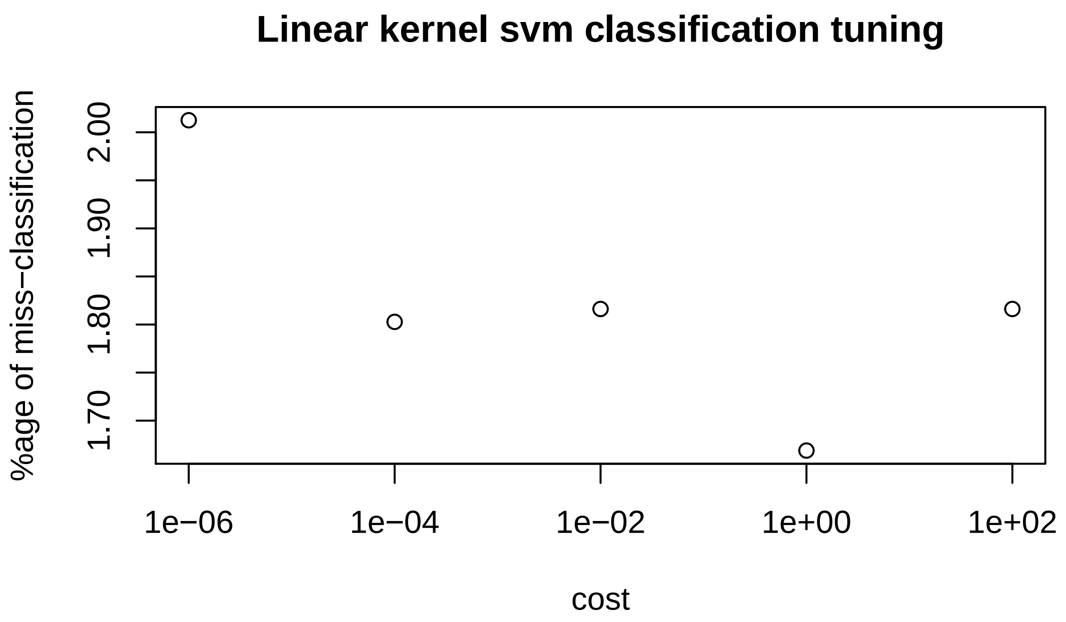


Figure 2.2: Tuning of cost for linear svm classification

RBF kernel

RBF kernel svm classification replaces the value of the dot product with $\exp(-\gamma||x-y||^2)$. It only has two parameters to tune : γ and cost.
In order to find the best combination, we performed a grid search and for each combination trained the classifier on a very small subset of the data set (for computational reasons). One of the plots that were produced during this grid search is reported in figure 2.3 but there are many more in the plot folder of the files you will find attached.

The best found combination of hyperparameters is $\gamma = 0.01$ and cost $= 100$. It has a cross-validated miss-classification rate of 1.33% when trained on the whole data set. It still does not outperform QDA.
However, at the end of the tuning process we realized that transformations were made to the data so that it is more easily learned by smoother models such as discriminant analysis. Such transformations are useless and ever hurtful for less predictable models such as SVM classifications. We tried inputting into SVM classification other methods for pre-processing the data, for instance by directly putting the number of days since the beginning of the year rather than the day in the month and the number of the month. SVM is the only method for which such a pre processing turned out to be efficient. Using this preprocessing with RBF kernel and the previously mentionned hyperparamer, we could lower the miss-classification rate to 1.164% and very slightly outperform QDA.

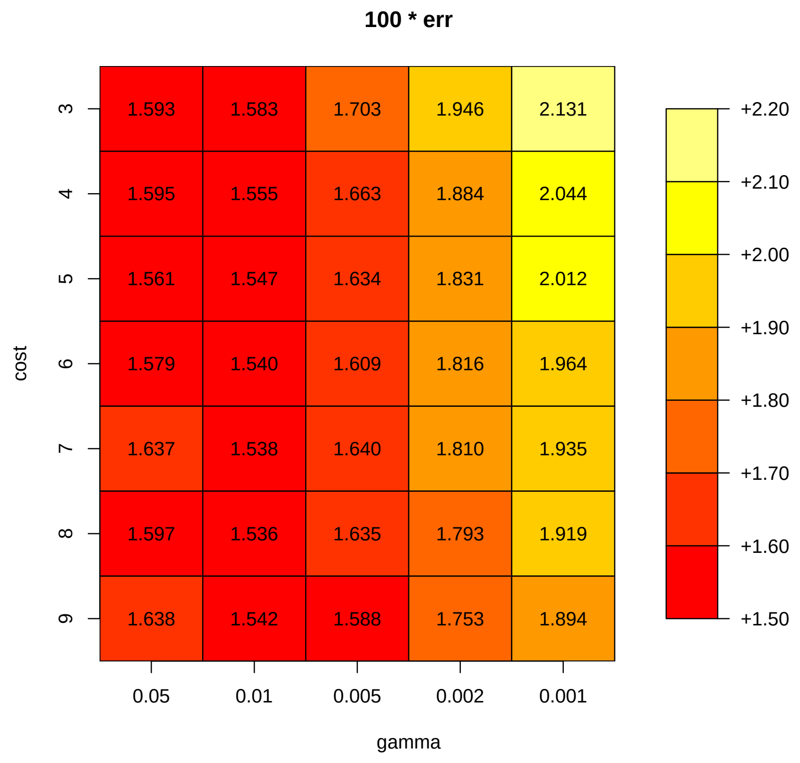


Figure 2.3: Tuning of cost and γ for rbf kernel svm classification

This needs to be investigated. However, tuning svm hyperparameters is very computationally costly and typically requires either a lot of computational power or waiting a long time. My laptop doesn't have a graphic card and I didn't have enough time to tune hyperparameters further if I wanted to explore other methods. Most of the tuning was made by training the classifier on only 1% of the data (which explains why errors in figure 2.3 are all higher than 1.3%). I also didn't have much time to explore higher degrees for polynomial kernel which took a very long time to compute. In the end I believe svm classifier might perform well with less preprocessing of the data and more precise tuning and they should be investigated.

2.2.4 Trees

Classification tree

We also tried using classification trees. Tree-based classifier have the advantage of being really easily to interpret and they could be very useful to give insights to the decision makers. This is not the most important method here. We will not explain the building of our tree-based classifier in details and just give some interesting results. Trees are created by first finding for each set the pair of predictor and value that best minimizes an impurity measure of the data, and repeating the process in order to grow a larger and larger tree up to a certain point. A subtree is then selected by optimizing a criterion called *cost-complexity* that minimizes the sum between the number of subsets in which the space of predictor is divided multiplied by an hyperparameter and the impurity measure. This hyperparameter was tuned by cross validation. The corresponding classifier has a miss-classification rate of 5.85%, which is poorer than most other methods but still good enough for it to be meaningful. Its result when trained on the whole data set is displayed in figure 3.2

Random forest

We also tried using random forest, a method that grows many trees by training them on bootstrapped training sets and only allowing the growth of the tree to be made within a randomly chosen subset of predictor at each step. A given observation will be classified according to a majority vote of the many trees in the random forest. This methods has a miss-classification rate of 1.98% which is pretty good but clearly not the best.

2.3 Conclusion

A summary of all the miss-classification rates can be found in table 2.2.

QDA	1.19%
LDA	2.20%
Naive Bayes	7.17%
KNN	3.41%
Logistic Regression	1.84%
Linear SVM	1.66%
Polynomial SVM	1.30%
RBF SVM	1.33%
RBF SVM (less preprocessing)	1.16%
Tree	5.85%
Random Forest	1.98%

Table 2.2: Cross-validated miss-classification rates for tested classifiers

There seems to several classifier with great miss-classification rates, the best two being Quadaric Discriminant Analysis (QDA) and the version of RBF SVM classification with a different preprocessing of the data. The fact that QDA works so well when compared to more sophisticated methods such as SVM or more robust methods such as Logistic Regression is a very strong hint that the hypothesis required for it to work are true. It seems like the within-class distribution **are** normally distributed, which is coherent with our first observations.

Considering what was previously mentioned in section 1.4, because too complex model may recognize pattern that only exist in 2016, we might want to go for a model with a slightly higher cross-validation miss-classification rate if it is less prone to overfitting. This would mean chosing **QDA** over our RBF SVM classifier as the final model for predicting future sales. Note that a more conservative alternative would be to chose Logistic Regression, which does have a greater miss-classification rate but works spectacularly well considering the number of parameter it uses. It uses half as many as Naive bayes but with 4 times less miss-classification. It also estimates around 20 times less scalar parameters than QDA and should be much less prone to overfitting. However there is so much in the data that hints at the within-class distribution being gaussian that QDA has to be our final choice.

Chapter 3

Insights

Using some of the more straightforward machine learning models and our *a priori* observations, there are some insights in the data that can be made in order to help decision makers understand how likely a policy is to be sold or not. There are mainly two really obvious insights that should be given : the relative importance of predictors in the sale of a policy, and a decision tree that can help understand how to reason with potential policies.

3.1 Relative importance of predictors

It was clear from our *a priori* observations that some criteria (*Age*, *Tax*, *Price*, *Vehicle Mileage* and *Vehicle Value*) would be more important than others in discriminating between sold and unsold policies, just by looking at how different their within-class marginal distributions were. Moreover, LDA gives us a way to **quantify** the relative importance of different criteria. LDA stands for Linear Discriminant Analysis and is called that way because with the assumption that the within-class covariance matrix are equal, classifying an observation is made by comparing a linear combination of predictors to a given threshold. By looking at the coefficient of the predictor in the linear combination, one can understand the relative importance of criteria when compared to one another for determining whether or not their LDA prediction will be *Sold* or *Not Sold*. As LDA has a sucess rate of around 98%, it is also a pretty good indicator of the relative importance of criterion for **determining whether the corresponding policy will be sold or not**. Please note that for this metric to be scale invariant the values of predictors will need to be scaled first so that for instance a 50% increase in Age is not necessarily less important than a 50% increase in Price that are all typically higher. The computed relative importance of criteria are plotted in figure 3.1.
One can notice that large differences of importance of criteria inferred in our first observations were true. Price seems to be the most important predictor. The sign of the corresponding weights helps us understand whether an increase in each predictors implies that a sale is more likely or less likely. An increase in Price seems to **increase**

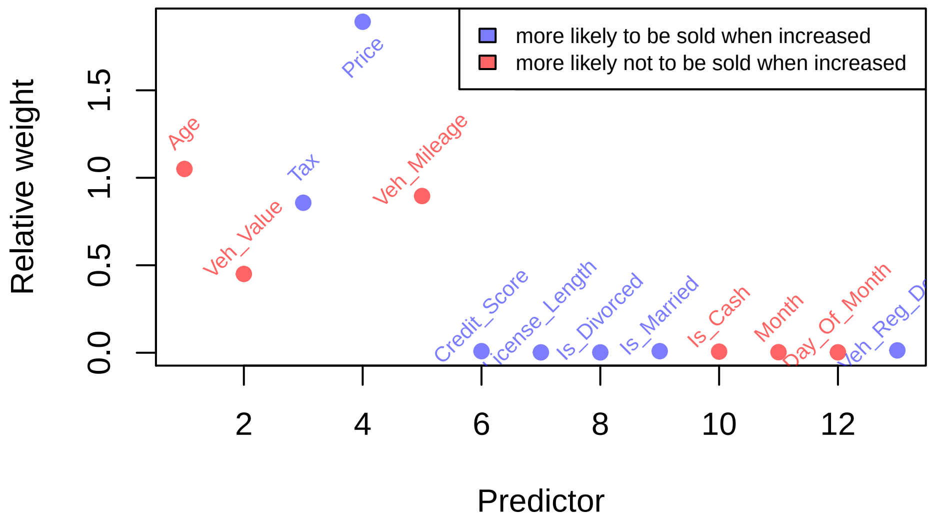


Figure 3.1: Relative importance of predictors in 97.8% success linear classification

the probability of a sale which is pretty surprising (but could be predicted from its within-class marginal distributions in reported in figure 1.6). One interpretation of this phenomenon would be that there is some kind of cognitive bias making the user believe more expensive policies are better. However, one must keep in mind that insurance policy prices vary from one customer to another. Another explanation is that higher priced policies are more competitive than lower priced policies with regard to the market. In that scenario, if a customer is *risky* and pays a high price for insurance he is likely to buy ours because it's cheaper than most, but if it's a safe customer with typically low policy prices he isn't because ours is more expensive than most.

3.2 Decision tree

The tree represented in figure 3.2 provides a very straightforward algorithm that can determine with around 95% accuracy whether or not a policy will be sold. It is a great way for a decision maker to understand how to discriminate between a policy that will be sold and a policy that won't. I believe it is in itself simple to understand, efficient and insightful, maybe even more so than the relative importance in criteria determined by LDA. I also think they are quite complementary : it is easier to understand the decision tree than what a ”relative importance” means, but the latter gives a global vision of the predictors and is usefull to determine quickly whether or not x criterion is important.

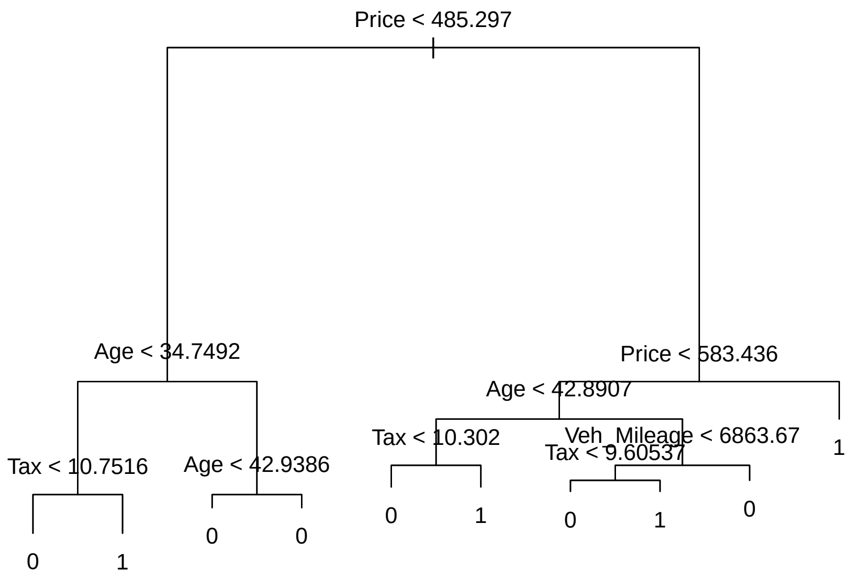


Figure 3.2: 94% success-rate decision tree