University of Padova

Department of Mathematics "Tullio Levi-Civita"

Master Thesis in Data Science

Modelling of annual turnover of "Società di persone" and "Ditte individuali": a case study based on SIAM data set

Supervisor
Prof. Mariangela Guidolin
University of Padova

Co-supervisor
Dr. Giovanni Tessiore
Cerved Group

Master Candidate
Giorgio Dilda

Student ID
1231418

Academic Year

2021-2022

I would like to thank my family, my friends and Giulia.
They all played a key role in making this important achievement possible.

Abstract

In this thesis I present the development and the results of the project I carried out during my internship in Cerved Group. The focus of the project was to build a model able to estimate the turnover of small Italian companies that are not obliged by law to draft a financial report. The fact that this indicator is not available for these companies poses a serious problem when trying to automate tasks when they are involved. To estimate the model I was able to exploit the SIAM data set in order to retrieve the target variable: this is a database in which banks put financial report proxies of their client falling in these categories. The model had to satisfy some criteria of interpretability for business reasons and so I propose here some variants of linear models and a complex XGBoost model made interpretable via SHAP values. All the new models proposed significantly outperform the old model that is in production today with the XGBoost model achieving slightly higher performance. The proposed models have been judged very positively by the business side and one of them will be implemented in production after further diagnostic analysis.

Contents

Listing of figures

Listing of tables

Listing of acronyms

SVR	Support vector regressor
SVM	Support vector machine
SP	Società di persone (Company juridical form in Italian law)
DI	Ditta individuale (Company juridical form in Italian law)
BM	Balda-Monforte encoding (one of the possible encodings of business sectors)
SHAP	SHapley Additive exPlanations
ML	Machine Learning
SIRC	Sistema Informatico Rischio di Credito (Credit Risk IT System)
KPI	Key Performance Indicator
MSE	Mean Squared Error
SVD	Singular value Decomposition
GAM	General Additive Model

1 Framework

1.1 CeBi

This work has been carried out during my internship and working experience in CeBi. CeBi, which stands for Centrale dei Bilanci (Central of financial reports), was founded in 1983 at the behest of Banca d'Italia and other banking institutes with the purpose to centralize, standardize and store information coming from financial reports. Since then CeBi has grown a lot, in particular in the last decade it became part of Cerved Group.
In this period, in parallel to more traditional reports and analysis, CeBi developed a new line of products oriented to quantitative risk assessment. The amount and quality of proprietary data, the deep dominion knowledge and cooperation with other entities of Cerved Group made it market leader of its sector in Italy. Today CeBi is split into four different teams:

Financial Analysis and Data

This is the team that used to be the core of the company. It manages financial report information and takes care of extracting information from them.

Risk Advanced Analytics and Innovation

This is the team that develops new core models. These range in a wide variety of tasks: credit risk assessment for single companies, real estate price estimation, churn models, bank account history based financial risk are just some of the examples.

Anaytics Consulting

This is the team that deals with large companies that require some kind of customization of core products, for example integrating data the company is collecting or doing some fine tuning on a data set provided by the company itself.

Qualitative Risk Analysis and Forecasting

This team produces sectorial qualitative and qualitative analysis .

Nowadays most of CeBi's revenues still come from business relationships with banks (64%) but the consulting is growing quickly (21%), the remaining comes from the sale of proprietary software (15%).

1.2 The Problem

The task i worked on is, in the context of the Risk Advanced Analytics and Innovation team, the updating of a model that predicts annual turnover of small companies. In particular it refers to those that fall under the juridical forms of "ditta individuale" (DI) and "Società di persone" (SP) which are usually used by small companies and freelancers. The peculiarity of these two juridical forms is that for the italian law they are not required to publish a financial report and so in practice they don't. This of course is a problem for a company that historically has based its products on this specific type of data since these companies even if small are quite numerous. There are is lot of information that can be extracted from a financial report but there is one KPI in particular that plays a key role in almost every model: annual turnover. For this reason in CeBi it has been estimated a model that, with the few available data provides a reasonable estimate of this KPI to be used as a input proxy for all other delivered products. There is however a big obstacle which is to retrieve the target data in order to train the model but CeBi is able to overcome this thanks to the SIAM data base.

SIAM data base

The SIAM is an agreement among CeBi and 6 italian banks that started in 2005, its goal is to collect data related to DI and SP. Each bank that is part of this convention sends financial statement proxies of DI and SP that have a bank account with them to CeBi that manages the central data base. Each member of the agreement is given, according to its dimension, a minimum amount of financial statements proxies to provide to the central data base in order to be able to access the whole information. The main goal of this agreement for the banks is, by a credit perspective, to retrieve financial information of prospect clients but as a side effect CeBi has the right to exploit those data for statistical purposes: those information cannot be used as predictors for any model but can collectively be used in order to estimate models or to generate aggregate statistics.

2
Methods

In this chapter I am going to briefly cover the main tools i used during my research. The theoretical approach for all these models, with the exception of SHAP, has been taken by [1].

2.1 Linear Regression

Linear regression models are by far one of most old and yet diffused families of models. The progenitor of this entire family is the simple least squares linear regression; this is a very simple, robust, flexible and interpretable model that can be applied to a wide variety of problems. It is usually not as good as more complex models as for example grandient boosting models when used as out-of-the-box tool but it can often reach similar performances when appropriately tuned. The framework of this model is the following: we have a matrix of data $X_{m\times n}$ in which the i-th row represents an observation of our data set with its n features and the j-th column represents the m values the i-th feature assumes in our data; and a vector of length m which represents the target variable of our problem. The goal of the model is to find a set of coefficients β such that we can predict the output of a given observation x as $y = \beta_0 + \sum_{j=1}^{n} x_j\beta_j$ where j runs along the features. This could be rewritten using vectorial notation as $y = \beta_0 + x \cdot \beta$ where $x = \left(x_1, \cdots, x_n\right)$ and $\beta = \begin{pmatrix} \beta_1 \\ \vdots \\ \beta_n \end{pmatrix}$ Here the term β_0 is usually referred to as intercept but it can easily be incorporated in the summation y adding to our data matrix X a new column with a constant value across all rows, the value assigned is not important in this configuration of the problem (since it will only scale the β_0 coefficient) but will have some relevance when we will introduce regularizations. The criterion with which we determine the β is very simple: we want to reduce the sum (or mean) of the squared errors our model commits when predicting over our training set. We can express this in math as:

$$\beta_{best} = \underset{\beta}{\operatorname{argmin}} \, MSE(X, y, \beta) = \underset{\beta}{\operatorname{argmin}} \sum_{i=1}^{m} (x_i \cdot \beta - y_i)^2 = \underset{\beta}{\operatorname{argmin}} \, (X\beta - y)(X\beta - y)^\top \tag{2.1}$$

This is a well known problem for which we can compute the exact solution setting the first derivative with respect to β equal to zero and imposing that the Hessian is positive definite:

$$\frac{\partial MSE}{\partial \beta} = -2X^\top (y - X\beta) \tag{2.2}$$

$$\frac{\partial^2 MSE}{\partial \beta \partial \beta^\top} = 2X^\top X \tag{2.3}$$

the second condition is always satisfied when $X^\top X$ is full rank (since $X^\top X/m$ is the convariance matrix of the features in our data set this means that we do not have perfectly correlated features) while from the first we can retrieve:

$$\beta_{best} = (X^\top X)^{-1} X^\top y \tag{2.4}$$

From a geometrical point of view we are interpolating the cloud of m points of our data with a hyperplane passing by the origin in the $\mathbb{R}^{n+2}$ space (if we choose not to incorporate β_0 in the X matrix by adding a constant column then we are defining a hyperplane in the $\mathbb{R}^{n+1}$ space but we are removing the constraint that it has to pass by the origin).

2.1.1 Ridge Regression

Many variants of this base model exist, we are going to use a couple of the more known. The first one is the Ridge regularization, it consists in a regularization related to the size of the coefficients that is introduced in the objective functions of the problem:

$$\beta_{Ridge} = \underset{\beta}{\operatorname{argmin}} \sum_{i=1}^{m} (x_i \cdot \beta - y_i)^2 + \lambda \sum_{i=1}^{m} \beta^2 = (X^\top X + \lambda \mathbb{I})^{-1} X^\top y \tag{2.5}$$

The original Ridge regularization would not include the intercept in the penalization term, which would make the computation of the optimal solution a bit harder. There are two ways to work around this problem: the first is centering the target variable in order to have $\beta_0 = 0$; the second is a trick that consists of incorporating the intercept in the data matrix as suggested above, but taking care to use a very high constant term, this will result in $\beta_0^2 \approx 0$ making it de facto not regularized. In one way or the other we can get rid of the unpenalized intercept term and get to the formulation shown in (3). This regularization contributes to solve a couple of well known problems realted to collinearity in our data. The first is related to the computation of the solution: in the original linear regression model we have to compute the inverse of $X^\top X$ which is not possible in case X has not full column rank; with the Ridge penalization however we need to invert $(X^\top X + \lambda \mathbb{I})$ which is possible even in case X has not full column rank. If we apply SVD to our dataset,

$$X = UDV^\top \tag{2.6}$$

where $X, U \in \mathbb{M}_{m\times n}$, $D, V \in \mathbb{M}_{n\times n}$, U, V are orthogonal and D is diagonal with $d_1 \geq d_2 \geq \cdots \geq d_n \geq 0$. The first of the new coordinates, which is a linear combination of the original features defined by the first row of V is going to be the direction that catches the most variance in our data, going on with the other elements of this new basis we find the components that capture each time the greatest amount of residual variance. When we try to write down the predictions over our training set of our Ridge regression with this notation we obtain the following:

$$X\beta_{Ridge} = (X^\top X + \lambda \mathbb{I})^{-1} X^\top y = UD(D^2 + \lambda \mathbb{I})^{-}1DU^\top y = \sum_{j=1}^{n} u_j \frac{d_j^2}{d_j^2 + \lambda} u_j^\top y \tag{2.7}$$

Since λ is positive $\frac{d_j^2}{d_j^2+\lambda} \leq 1$; this means that we have a shrinking of the contribution of components that capture lower variance in our data set. This means that if our features are all uncorrelated we just obtain a scaling of the original linear regression coefficients by a factor: $\beta_{Ridge} = \frac{\beta_{best}}{1+\lambda}$. Incidentally the quantity $\sum_{j=1}^{n} \frac{d_j^2}{d_j^2+\lambda}$ represents the effective degrees of freedom of this type of regression giving a measure of the strength of the regularization.

2.1.2 Lasso Regression

Lasso regression is very similar to Ridge regression but as we will see some minor changes in the definition of the penalty term result in a completely different behaviour of the resulting model. In this case instead of penalizing the square of the coefficients we penalize their absolute values obtaining:

$$\beta_{Lasso} = \underset{\beta}{\operatorname{argmin}} \sum_{i=1}^{m} (x_i \cdot \beta - y_i)^2 + \lambda \sum_{i=1}^{m} |\beta_i| \tag{2.8}$$

Both Lasso and Ridge problems can be rewritten replacing the penalty term with a hard constraint on β like shown below, it has been demonstrated that a one to one relation between λ and t exists in both problems:

$$\beta_{Lasso} = \underset{\beta\, s.t. \sum_{i=1}^{n} |\beta_i| \leq t}{\operatorname{argmin}} \sum_{i=1}^{m} (x_i \cdot \beta - y_i)^2 \tag{2.9}$$

$$\beta_{Ridge} = \underset{\beta\, s.t. \sum_{i=1}^{n} \beta_i^2 \leq t}{\operatorname{argmin}} \sum_{i=1}^{m} (x_i \cdot \beta - y_i)^2 \tag{2.10}$$

In this form it becomes easier to see how the different penalties promote different behaviours of the coefficients. Once fixed a certain t for both regressions we have two different dominions: a l_2 ball for the Ridge (which is a circle in 2 dimension and a sphere in 3) and a l_1 ball for the Lasso (which is a diamond in 2 dimensions and a octahedron in 3 dimensions). The solution of the problem will be the point of each dominion that is on the lowest possible contour of the error function: while the l_2 ball is smooth, the l_1 ball presents edges and corners and if the solution lies on one of these it means that on or more coefficients will be set to zero. The main advantage in applying Lasso penalty to a linear regression problem is that it will apply an automatic feature selection: increasing λ, and therefore decreasing t, β_{Lasso} will have more and more coefficients set to 0. This can be considered a sort of automatic and continuous feature selection (continuous varying λ). It is worth to point out that while selecting variables Lasso is also increasing the bias of the model (because even the survived coefficients will be shrunk in some measure), for this reason it has been formulated a variant of the original lasso to overcome this problem called relaxed Lasso. The procedure is pretty straight forward: a Lasso regression is applied to the original problem and then an simple linear regression is applied to the subset of features $\{i | \beta_{Lasso\, i} \neq 0\}$.

2.1.3 Elastic Net

Elastic net is simply a fusion of the two previous penalties, it can be written in different forms but we will use the following notation:

$$\beta_{ElasticNet} = \underset{\beta}{\operatorname{argmin}} \sum_{i=1}^{m} (x_i \cdot \beta - y_i)^2 + \lambda(\alpha \sum_{i=1}^{m} |\beta_i| + (1-\alpha) \sum_{i=1}^{m} \beta_i^2) \tag{2.11}$$

In this form we have two hyper parameters: λ regulates the strength of the regularization while α regulates the proportions of l_1 and l_2 penalties. Usually α is chosen in order to shift the great majority of the penalty on the l_1 term; this is due to the fact that while Ridge resolves collinearity related problems even with small penalty coefficients and does not provide significant advantages increasing it beyond this point, Lasso's penalty coefficient is tightly bound to how much we sparsity we desire in our coefficients and has a heavier impact on the resulting model.

2.2 Support Vector Regressor

SVR is a transposition of the SVM concept into a regression problem. The SVM is a special kind of linear model applied in binary classification problems that has the property to have its solution defined only by a small subset of observations close to the hyperplane that separates the two populations. This feature makes the model robust to outliers and therefore widely applied in many different fields. The SVR brings this to the regression world just by changing the quadratic loss function we saw earlier with another one. The framework of the problem is very similar: we have a observation matrix X, a response variable array y and a set of coefficients we want to estimate β. We define the error of a prediction of our model as

$$r = y - \hat{y} = y - \beta \cdot x$$

Once we have defined the error of a single prediction we moce on to define the loss function of the SVR model:

$$V_\varepsilon(r) = \begin{cases} 0 \; if \, |r| < \varepsilon \\ |r| - \varepsilon \; otherwise \end{cases} . \tag{2.12}$$

Note that this function depends on a hyper parameter ε that we have to choose in advance.

The behaviour of this function is pretty straight forward: if the error in the prediciton is lower than ε then that observation won't contribute to the total loss; on the other hand if an observation has a very high error its contribution to the gradient of the loss function will be the same of an observation closer to it (unlike in the simple linear regression where due to the quadratic loss function its weight would be higher) making this loss function more robust to outliers.

2.3 Gradient Boosting

Tree gradient boosting is a widely applied complex machine learning model. It is an ensemble model that exploits a number of weak learners, regression trees in this specific case, in order to achieve high performance.

2.3.1 Regression Tree

Regression trees are a very simple model. The idea is to divide the space of samples in different regions each corresponding to an output value. This can be expressed as:

$$\hat{y} = T(x, \Omega) = \sum_{j=1}^{J} \lambda_j I(x \in R_j) \ where \ \Omega = \{(\lambda_j, R_J) forj = 1, \cdots, J\} \tag{2.13}$$

where I is a function that takes value 1 if its argument is verified and 0 otherwise,J is the number of different regions identified by the tree, R_j is the j-th region, x is the sample whose target variable value we are trying to estimate and $\hat{y}$ is the predicted value. As we can see the computation of the output itself is pretty straight forward, but how do we estimate the optimal set of regions and their corresponding values? The easy part are the values $_j$ corresponding to each region, they simply correspond to the value that minimizes the loss over the set of training samples belonging to the corresponding region; in our specific case since the loss will be MSE the optimal value would be the mean of the response variable of the samples belonging to the corresponding region.

$$\lambda_j = \frac{\sum_{i=1}^{m} y_i I(x_i \in R_j)}{\sum_{i=1}^{m} I(x_i \in R_j)} \tag{2.14}$$

On the other hand it computationally infeasible to find the optimal set of regions to split the sample space and for this reason usually greedy algorithms are applied. The most common strategy consists in splitting along one variable j at a time selecting an optimal split point s such that the loss function is minimized. If we identify the two regions R_1 and R_2 defined by the first split as $R_1(j,s) = \{X|X_j \leq s\}$ and $R_2(j,s) = \{X|X_j > s\}$ the optimal solution for j and s is going to be

$$j, s = \underset{j,s}{\text{argmin}} \min_{c_1} \sum_{x_i \in R_1(j,s)} (y_i - c1)^2 + \min_{c_2} \sum_{x_i \in R_2(j,s)} (y_i - c2)^2 \tag{2.15}$$

the two terms $c1$ and $c2$ can easily be computed as the mean of the y_i belonging to their region. This procedure is then repeated separately on each of the 2 new regions iterating until some stopping criterion is satisfied. Stopping criteria may vary form a cross-validated (or validation set) increase in loss function, reaching a fixed number of iterations or reaching a minimum amount of observations in a single region.

2.3.2 Tree Gradient Boosting

Trees however present significant limitations, if not carefully tuned they easily fall in heavy bias or variance problems and are not able to extrapolate at all. Moreover since they are built sequentially with a greedy algorithm they may not catch some patterns present in the data. To fix some of these weaknesses gradient boosting has been introduced. To build a gradient boosting model we start by estimating a single tree on our training data set. The hyper parameters of this tree, such as its stopping criterion, will have to be set in advance; since this tree will just be a part o a bigger model it is usually better to use hyper parameters that lead to a low variance tree (such as low maximum depth and high minimum population per leaf). We define the loss function of our problem as:

$$L(y_i, y^) = (y_i - y^)2$$

And therefore we can define the first tree as:

$$\Omega_1 = \underset{\Omega}{\text{argmin}} \sum_i = 1^m L(y_i, T(x_i, \Omega)) = \underset{\Omega}{\text{argmin}} \sum_{j=1}^{J} \sum_{x_i \in R_J} L(y_i, \lambda_j) \tag{2.16}$$

After building the tree we proceed building another one changing the target variable from y to $y - T(x, \Omega_1)$ and therefore we will define the second tree as:

$$\Omega_2 = \underset{\Omega}{\text{argmin}} \sum_i = 1^m L(y_i - T(x_i, \Omega_1), T(x_i, \Omega)) \tag{2.17}$$

From here we can easily write an iterative rule to train the Z-th tree as

$$\Omega_Z = \underset{\Omega}{\text{argmin}} \sum_{i} = 1^m L(y_i - \sum_{z=1}^{Z-1} T(x_i, \Omega_z), T(x_i, \Omega)) \tag{2.18}$$

The prediction of the model is simply going to be the sum of the predictions of the estimated trees:

$$\hat{y} = \sum_{z=1}^{Z} T(x, \Omega_z) \tag{2.19}$$

This model can grow a lot in complexity increasing the number of trees and therefore we need some ways to avoid overfitting. The first solution is, as mentioned above, to tune the hyperparameters of the underlying trees. In particular the maximum depth hyper parameter has a special meaning: it represents the maximum number of features interacting with each other that our model can catch. If we set maximum depth to 1 then our model will behave very similarly to a GAM model while setting it to i will allow to have interactions among at most i features at the same time. Other ways to deal with overfitting are to decrease the number of trees included in the model, and introducing a learning rate. The learning rate α scales the target variable for each tree by a factor, making the progress of the model slower (in terms of trees) on the training set but more robust.

$$\Omega_Z = \underset{\Omega}{\text{argmin}} \sum_{i} = 1^m L(\alpha(y_i - \sum_{z=1}^{Z-1} T(x_i, \Omega_z)), T(x_i, \Omega)) \tag{2.20}$$

2.4 SHAP

The problem of interpretability is a well known topic in machine learning models. It basically comes down to the fact that more complex and therefore usually more performing and adaptive models are usually black boxes. This means that a human, given all the model's parameter and hyper parameters needed, can not understand what roles the different input features played in determining the outcome of a given observation. For example let's think about a simple multi layer neural network, this is a well known and standard model that has the property to be able to spot even complex patterns in the dataset without much tuning effort from the data scientist side. The model itself is pretty straightforward and someone with some basics in matricial computation could easily learn to compute the output of the trained model with little effort given the matrices that connect the different layers and the kernel function of the nodes. The problem arises when trying to address the role each feature played in the process since the final form of the result is very complex and each feature interacts with all the others in different combinations.

Interpretability of machine learning models itself has been a very hot research topic in the last years and a few algorithms with this aim have been published up to now. In this project I used one of them, SHAP, to try to make a gradient boosting model to a certain degree interpretable. SHAP is a method first published in 2017 based on a game theoretical approach. In the SHAP framework every feature included in the model represents a player of a cooperative game. The cooperative game is the minimization of the loss function of the problem we are dealing with. After the normal estimation of a model we would know just the outcome of the game (the loss function minimum value reached) after all players have done their move, making it hard to tell which one had a significant impact on the outcome and which did not. The idea is to measure partial outcome after each player has done its move; to do so we need to estimate many models each trained on a subset of the set of features originally used for the model we want to analyse.

In the example in figure 2.1 we have a hypothetical model that includes 3 features (A, B and C). As it can be seen from the figure we start training a model with the same characteristics of the one we want to explain on a empty data set: this of course will be a dummy model and the output value will always be the expected value of the target in the training data set. From here we add a feature obtaining 3 different models: one using A, one using B and one using C. We can see that now the output values vary depending on the variable that is used in the model, and we have a first catch of how the three variables influence the final value. We proceed training new models using sets of 2 features, generating the second layer of models in the figure. In the last layer we have the full model, exploiting the full set of features. The gaps in the result between a model in the i-th layer and a model in the (i+1)-th layer are called marginal contributions of the new feature that has been added. To compute the overall impact of a given variable all we have to do is a weighted mean of its marginal contributions. The weights are obtained following two simple rules: Weights of marginal contributions of the same row (with the same cardinality of the used feature set) should be equal and the sum of the weights of each row should be equal. This can be summarized in a single formula that gives the weight corresponding to the i-th layer on a set with I total features:

$$w_i = \frac{1}{I\binom{I}{i}} \tag{2.21}$$

If we call $M_s(x)$ the output of the model trained with the subset s of features on the observation x and we call $P(S)$ the set of parts of a set S we can write the following general formula for the

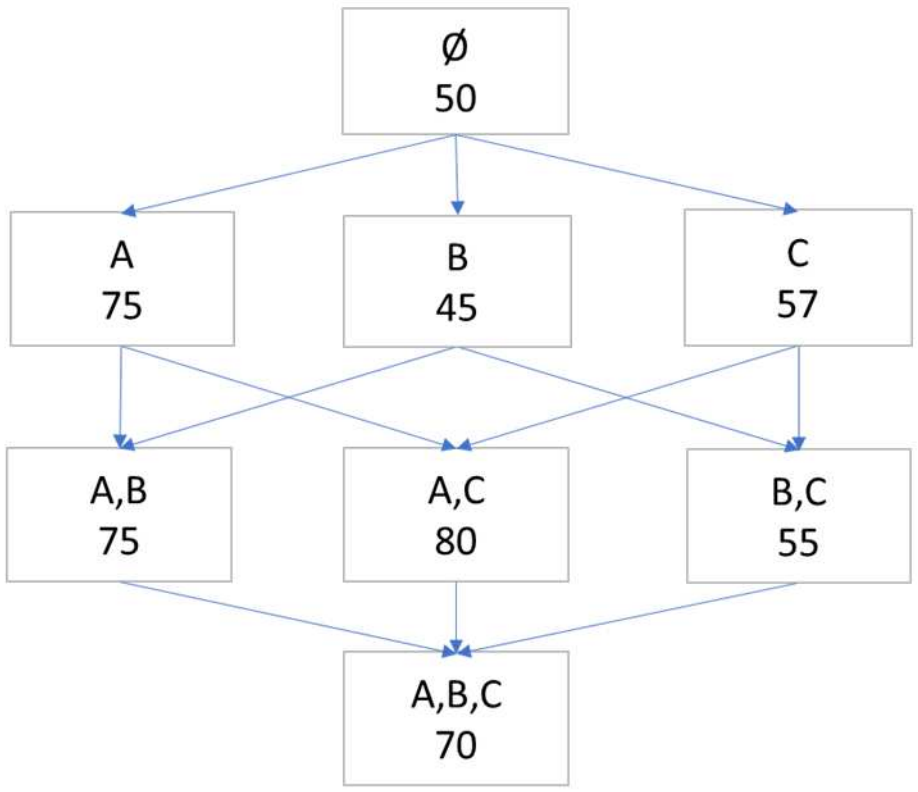


Figure 2.1: Toy example of SHAP on a 3 features dataset

SHAP value associated with feature A over observation x:

$$SHAP_A(x) = \sum_{s \in \{s \in P(S) s.t.\ A ins\}} w_s(M_s(x) - M_{s\ \{A\}}$$

In our arbitrary example shown above the shap value of variable A with respect to the observation x would be computed as:

$$SHAP_A(x) = \frac{1}{3}(M_A(x) - M(x)) + \frac{1}{6}(M_{AB}(x) - M_B(x)) + \frac{1}{6}(M_{AC}(x) - M_C(x)) + \frac{1}{3}(M_{ABC}(x) - M_{BC}(x)) \tag{2.22}$$

This kind of approach offers some interesting advantages:

- **It is model agnostic** this means we can apply shap virtually to any model as long as we are able to build alternative versions of it using only a subset of features

- **The SHAP values are additive** meaning $M_S = M + \sum_{x \in S} SHAP_x$ when S is the full set of features.

This means we can explain how much each feature contributed to the gap from the expected value in an additive way, which means that we can retrieve for example the total contribution of a one hot encoded categorical variable just by adding the SHAP values of its dummy features. There is however comes to the cost of a big flaw: the number of models we need to estimate in order to follow this procedure quickly explodes. With simple combinatory computation we can get to the conclusion that the number of models to be trained in order to carry out this procedure on a dataset with N features is 2^N, which becomes unfeasible increasing N. For this reason the library [2] I used for this purpose does not actually use the original algorithm but exploits some approximations in order to make the computation possible. We can use the SHAP values in two ways:

- **Locally** given an observation we can assess the weight of each variable in its final result (these are simply $SHAP_A(x_i)$)
- **Globally** given a large population we can assess which variables are in general more impactful on the outcome (these can be computed as $\frac{1}{N} \sum_{i=1}^{N} SHAP_A(x_i)$)

We will see both these kinds of analysis in the next chapters.

3
Dataset

As anticipated in the introduction the data comes from the SIAM, a database fuelled by banks with financial report proxies of their own clients belonging to the categories of DI and SP. The fraction of this repository I used for this project's purpose ranges between 2015 and 2019 and includes 235919 reports from 81670 different companies. As we can see in figure 3.1 the quantity of information funnelled into this project is consistently decreasing over time with 2019's volume being almost a third of 2015's.

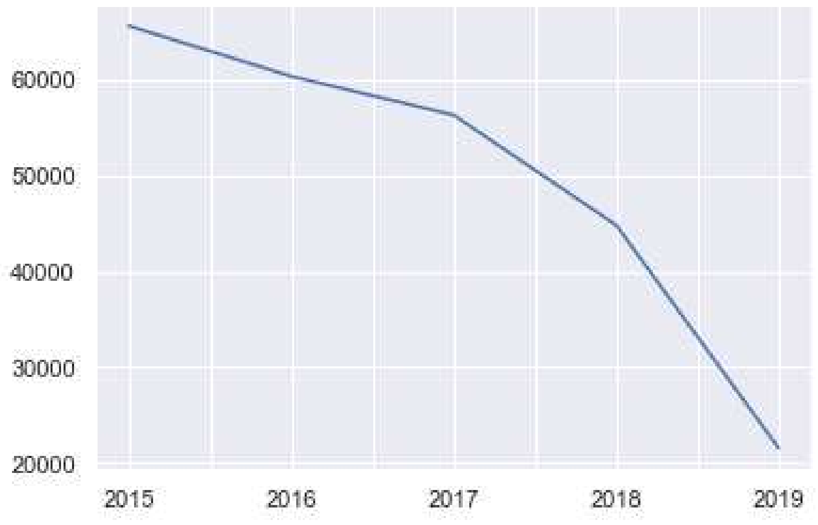


Figure 3.1: SIAM population over the last years

This poses a problem for the future because if this trend does not revert it will not be possible in some years a new estimation of the model exploiting this powerful tool to retrieve the target variable. In the SIAM data set there are of course available a lot of features for every single report but since typically these would not be available when the need to estimate the turnover arises we only took from there the annual turnover and registry information of the different companies. To keep the data set clean form anomalies we decided to remove observations whose annual turnover was lower than 5000 euros. We will see now some of these features in detail.

3.1 Features overview

Turnover

The turnover is the target variable of this project. The first thing to consider was that not all the report refer to a 12 months period, there are a minority that are referring to different spans of time. We chose to remove observations whose covered period was lower than 6 months and normalized the turnover of remaining observations to a 12 months. Another problem was that this feature was very skewed so instead of taking it as is we applied a logarithmic transformation to it (there was no need to add a constant to guarantee the existence of the logarithm since we already applied a filter that removed turnover values lower than 5000).

Geographical zone

This is a categorical variable that identifies the macro area of Italy the company operates in. There are 5 different areas that have been formally defined by ISTAT, which is the national statistics institute, that present significant differences in the economic tissue. This difference is due both to morphological and historical reasons and it can be spotted both in the population of the observations falling in each area (figure 3.2) and in the distribution of the turnovers per area (figure 3.3).

Region

The region is another categorical feature that gives more specific information with respect to the geographical zone. There are 20 regions in Italy, each corresponding to only one of the five geographical zones. The boundaries of these regions have administrative purposes but still they provide some information in term of economic characteristics of the territory. This can be easily spotted looking at the turnover distribution by region in figure 3.4.

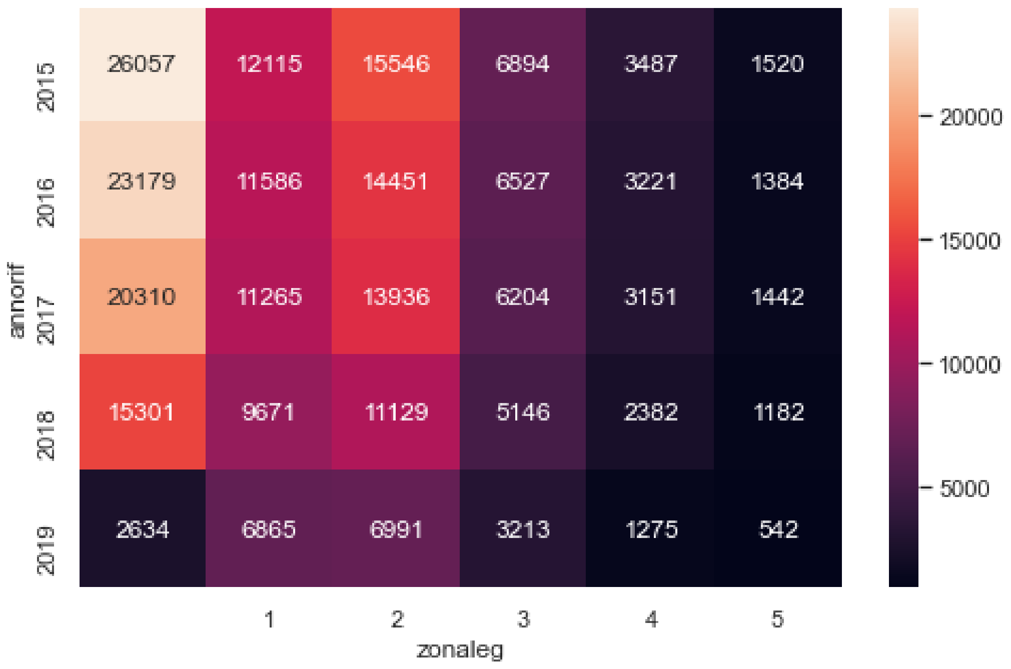


Figure 3.2: Distribution of SIAM population across years and geographical zones

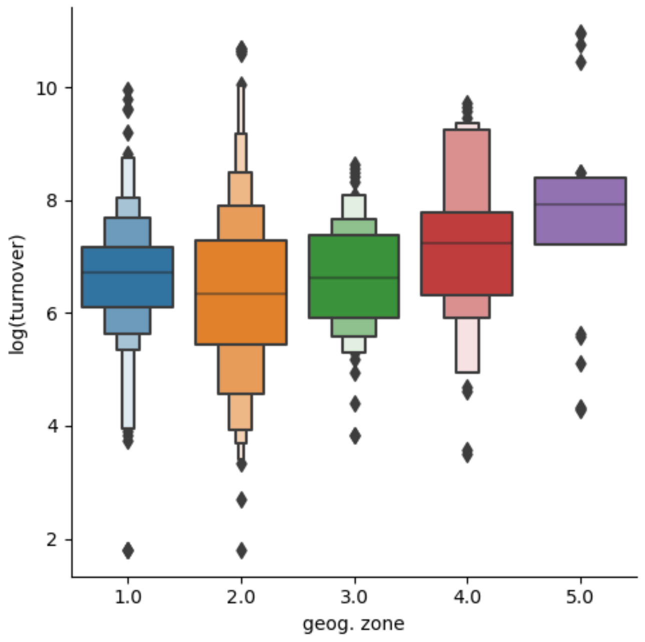


Figure 3.3: Distribution of turnover in different geographical zones

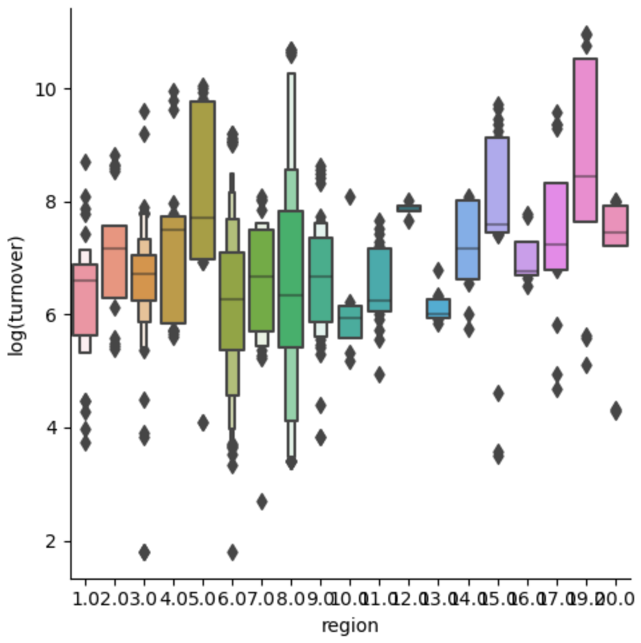


Figure 3.4: Distribution of turnover in different regions

Number of employees

This is by far one of the most important features in the data set in terms of predictive power. A common practice is to transform this kind of variables, that are known to have a monotonic relationship with the target, using a isotonic regression. An isotonic regression model is trained on the training set using as endogenous variable the feature we want to transform, in this case the number of employees, and as exogenous variable the target variable, in this case the logarithm of the turnover. The variable used in the models will be the transformation via this isotonic regression of the original variable. This kind of procedure is particularly useful when using simple models, such as linear regressions, involving non-linear relationships between a feature and the target. This can be done by many means depending on the variable and our domain knowledge (spline and loess are just some examples) but in our case we chose the isotonic regression.

Local units

This is a continuous variable that records the number of structures at the disposal of the company.

Property Score

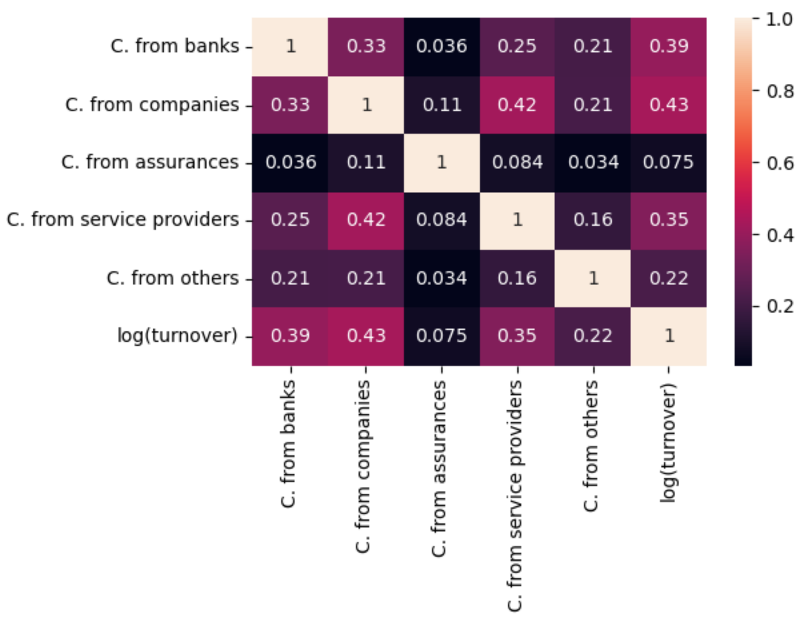
This is a score that has been developed for another project but was included in this project too. This score was trained to predict financial risk of a company using only data related to their properties and can therefore be used as another proxy of the total value of a subject's properties.

Age

A continuous variable that registers time since the legal foundation of the subject in years.

Consultations

in this set of variables it is stored the number of times the subject has been the object of a query in one of Cerved's business information services. This information is split by the kind of client that looked for the subject: banks, assurances, service providers, companies and others. This feature has been included in the data set over the assumption that a greater number of connections with other entities is correlated with a larger business of the subjects. As we can see from figure 3.5 the correlation with the turnover heavily depends on the type of entity that performed the search. Consultations from companies for instance are heavily correlated, probably because they are closely correlated to the size of the business bubble of the subject.



	C. from banks	C. from companies	C. from assurances	C. from service providers	C. from others	log(turnover)
C. from banks	1	0.33	0.036	0.25	0.21	0.39
C. from companies	0.33	1	0.11	0.42	0.21	0.43
C. from assurances	0.036	0.11	1	0.084	0.034	0.075
C. from service providers	0.25	0.42	0.084	1	0.16	0.35
C. from others	0.21	0.21	0.034	0.16	1	0.22
log(turnover)	0.39	0.43	0.075	0.35	0.22	1

Figure 3.5: Pearson correlation among the number of consultations divided by entity and the turnover

Business sector encoding

The last piece of the dataset identifies the sector in which the company is operating, there are three variables encoding this aspect:

- **ATECO 2007** This is the one that is valid for the Italian legal system since 2007.
- **SIRC** this encoding has been created by CEBI in order to carry out sector-wise studies before the ATECO 2007 encoding was released and is nowadays still used in many proprietary models and reports.
- **Balda-Monforte** The Balda Monforte encoding is the last one being created. It is an aggregation of the ATECO encoding based on the Italian business environment, developed exploiting both the financial reports data and sector experts knowledge available to CEBI, for statistical purposes.

Of these 3 possibilities the ATECO has been discarded first since its purpose is more administrative than statistical. BM encoding would be the most curated but happens to have a very high cardinality (369 unique values), on the other hand SIRC encoding has a far lower cardinality but is a bit older and so probably less up to date than the previous one. As we can see from figures 3.6 and 3.7 both encodings split the population in significantly different families. During the modeling part we will try to exploit both these encodings to establish which of them is the better choice.

3.2 Train-Test splitting

We paid particular care in splitting the data set in order to obtain meaningful models and a robust estimation of their performances. In this aspect of the problem the main point we focused on were two.

Stratification

We chose to stratify the population by turnover and SIRC sector. This is important in order to obtain two balanced sets that are able to properly train and evaluate the models we are going to build uniformly over the samples distribution.

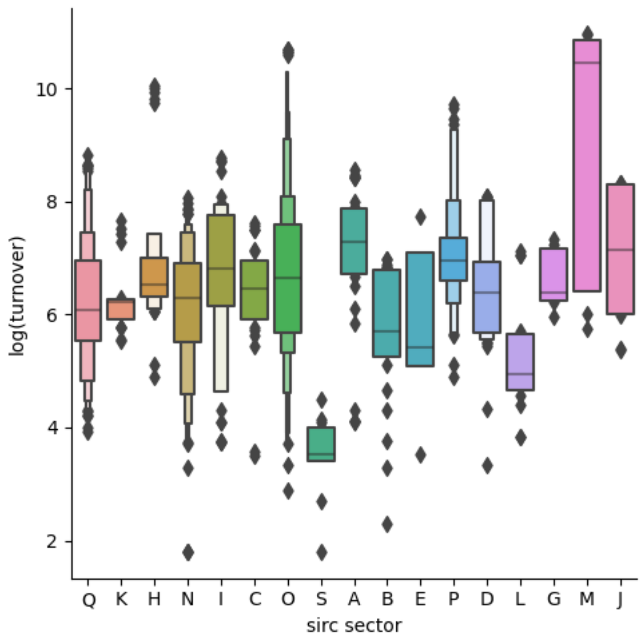


Figure 3.6: Distribution of turnover by SIRC sector

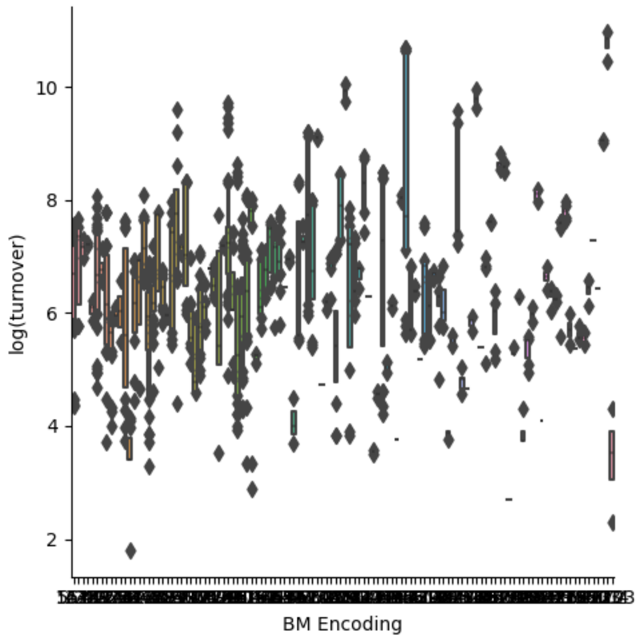


Figure 3.7: Distribution of turnover by BM category

Separation of subjects

Since an observation in our data set is uniquely identified by a subject id and a year of reference we chose to impose that a single subject id could only be present in either the training set or the test set. Since two observations related to the same subject in different years are highly correlated if we did not impose this kind of separation there could be the chance to not properly detect overfitting in one of our models even when evaluating it on the test set. If for example one of our models was heavily overfitting to the point that it closely behaves like a 1-NN model, it could still reach high performance on the test set as long as at least an observation from the same subject referring to a different year was present in the training set.

4 Models

4.1 Linear modelling

As a first attempt to model the phenomenon I tried to estimate some very basic and interpretable models belonging to the linear models family.

Implementation of sectorial differences

When trying to work with this kind of models on such a heterogeneous phenomenon the first problem is choosing how to express to the model that subjects working in different sectors should be modelled differently. For instance, for a company working in the agriculture field the number of employees may be a very indicative parameter but for a gas station it may not be so important or it may have a completely different weight on the final result. The simplest way would be to one hot encode the variable representing the sector in which the company is working and throw it into the model as a categorical variable. However this would be a very optimistic way of dealing with the phenomenon: since we are predicting the logarithm of the turnover this would correspond to introducing a different factor for each sector to be applied to the final result of the model. A different way to model it is by a segmented model. This means that once we have chosen a simple base model we estimate a different version of it for each possible value of the feature representing the business sector. During the prediction phase the model will read the business sector of the subject and use the appropriate model to make the prediction. Since I could not find an off-the-shelf implementation of this kind of model I implemented my own python3 version of it that I share in the snippet below. This implementation allows the parameterization of the base model and follows the scikit-learn [3] API syntax making it very easy to use even in a plug-and-play perspective. The only notable difference is that the user is asked to explicitly provide the scoring function when calling the score method since I could not find a way to automatically retrieve the default from the base model.

```python
\label{code:segmented
}
from sklearn.base import clone
import pandas as pd
from tqdm import tqdm
import numpy as np

class SegmentedModel():
    def __init__(self, base_model):
        self.base_model = base_model
        self.models = {}

    def fit(self, X, y, segment_feature):
        self.segment_feature = segment_feature
        values = X[segment_feature].unique()
        segment_column = X[segment_feature]
        self.output_dtype = y.dtype
        for v in tqdm(values):
            self.models[v] =
            ↪  clone(self.base_model).fit(X.loc[segment_column == v,
            ↪  :].drop(segment_feature, axis=1),y[segment_column == v])
        self.variable_names = X.drop(segment_feature, axis=1).columns

    def predict(self, X, segment_feature=None):
        if segment_feature == None:
            segment_feature = self.segment_feature
        if segment_feature not in X.columns:
            raise ValueError(segment_feature + ' variable not found in
            ↪  dataset')
        segment_column = X[segment_feature]
        diff =
        ↪  set(segment_column.unique()).difference(self.models.keys())
        if len(diff) > 0:
            raise ValueError(
                'The following values of segment variable were not
                ↪  present in training dataset : ' + str(diff))
        res = np.empty(X.shape[0], dtype=self.output_dtype)
        for v in segment_column.unique():
            res[segment_column == v] =
            ↪  self.models[v].predict(X.loc[segment_column == v,
            ↪  :].drop(segment_feature, axis=1))
        return res

    def coef(self):
        all_coef = {}
        for x in self.models.keys():
            all_coef[x] = self.models[x].coef_
        all_coef = pd.DataFrame(all_coef).transpose()
        all_coef.columns = self.variable_names
        return all_coef

    def score(self, X, y, metric, segment_feature=None):
        if segment_feature == None:
            segment_feature = self.segment_feature
        y_pred = self.predict(X, segment_feature=segment_feature)
        return metric(y, y_pred)
```

Of course the first approach is much more coarse but implies a lower computational effort and lower degrees of freedom of the final model; the segmentation approach on the other hand is able to better catch the variety of behaviours in different sectors but produces a model with much higher complexity and presents some pitfalls. Namely if we had a data set composed of n features plus a categorical feature representing the business sector assuming p unique values and we chose to use a simple linear regression with intercept the first solution would generate a model with $n + (p-1) + 1$ degrees of freedom, the $(p-1)$ comes from the fact that we would drop one of the p values including it in the intercept in order to avoid collinearities, while the latter $p(n+1)$, in this case we can not drop on of the p values since we have to fit a model for each of them.

Choice of the segmentation variable

Now the problem comes down to what variable we want to use to represent the sector in which the company is working and as anticipated in the previous chapter, we have two main choices: SIRC encoding and Balda-Monforte encoding. The latter one is more recent and much more curated with respect to the first one, the problem is that Balda-Monforte's granularity is very fine meaning that a lot of its categories are empty or almost empty thus making it impossible to estimate a meaningful model for each one of them. This could be a major problem also in the perspective of a shrinking database in which the less populated categories are just going to grow in number as time goes by. I figured out a way to take the benefits from both encoding at once in exchange for an extra computational effort during the estimation of the model. The idea is to fit a model for each SIRC sector (they are all widely populated) and one for each Balda-Monforte sector with a population above a given threshold; during the prediction phase the model will check whether the instance's Balda Monforte sector has a model linked to it, if that's the case that model will be used for the prediction of that specific instance otherwise the corresponding SIRC model will be used as a proxy. From now on we will refer to this new segmentation as "zoom segmentation" since its main property is to zoom in or out depending on the population of the category of the observation. While this seems quite cumbersome it can be implemented using the SegmentedModel class above quite easily.

Choice of base model

Once we have chose the segmentation variable and how to encode it in the model we have to choose the model. Given that the former model used for this task was a linear one I chose linear models in order both to have a fair comparison and to satisfy the project's requirement of robustness:

- **Elastic net** an enhanced version of the linear regression that includes a combination of two different penalties

- **Linear SVR** a model that extends the concept of traditional SVM to regression problems

While the idea behind this two models is different the final model is a linear one in both cases making them both valid candidates.

We will compare here the performances of models that represent different combinations of the three choices we presented above.

4.1.1 Elastic net with SIRC segmentation

The first model is an elastic net model segmented along the SIRC sector. I chose however not to drop the BM feature but instead to one hot encode it and use it in the model. Of course this kind of encoding adds a lot of degrees of freedom to the model since BM has high cardinality but since the model is segmented only a handful of them will actually be active in each segment and furthermore the l_1 penalty included in the elastic net will help to keep the coefficients related to the less meaningful at zero. In this context we could even scale their binary flags by a constant c in order to tune the strength of the regularization ($0 < c < 1$ for stronger regularization and $c > 1$ for weaker regularization) however in this specific case I chose to leave them as binary flag with 0/1 values. The chosen penalty coefficients for this model have been, following the notation used in the Methods chapter, $\lambda = 0.0005$ and $\alpha = 0.9$. This model consists of 20 underlying linear models each with 427 coefficients (for a grand total of 8560 degrees of freedom counting the intercepts) and reaches a performance of $R^2 = 0.656$ on the test set and $R^2 = 0.659$ on the training set. In figure 4.1 we can see a scatter plot showing the model predictions versus their true value on the training set.

4.1.2 Elastic net with zoom segmentation

This model is very similar to the previous one with the exception that it is segmented along the zoom segmentation. Of course in this case we do not include SIRC sector variable since it would assume only one value in each submodel and would therefore be useless. This model has slightly different hyper parameters with respect to the previous one using as a base model an elastic net with $\lambda = 0.2$ and $\alpha = 0.01$, it has 16473 degrees of freedom 47% of which are brought to zero from the l_1 penalty. The performances are quite similar with $R^2 = 0.665$ on the train set and $R^2 = 0.665$ on test set and even the predicted versus target plot in figure 4.2 is very similar.

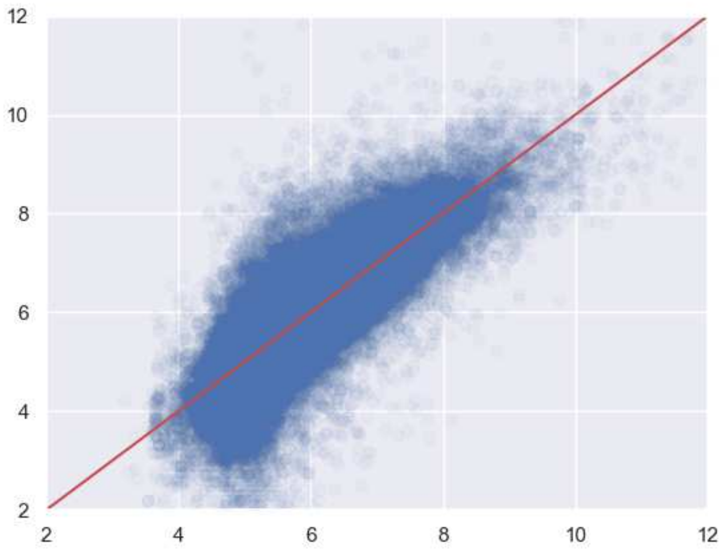


Figure 4.1: Performance of Elastic Net (SIRC) on the test set, predicted values are on the x axis and true values on the y axis

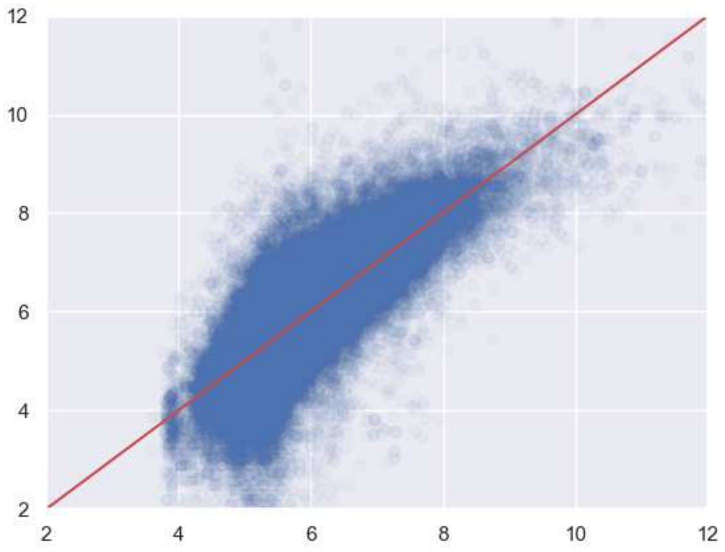


Figure 4.2: Performance of Elastic Net (zoom) on the test set, predicted values are on the x axis and true values on the y axis

4.1.3 SVR with SIRC segmentation

For the SVR model only the SIRC segmentation variant has been estimated. This is due to the fact that since the strength of this model is in its robustness to outliers using a high cardinality segmentation (and therefore feeding each SVR submodel with smaller data sets) would be counterproductive. I actually tried to estimate some SVR with zoom segmentation but the performance dropped dramatically. The data set used to estimate this model is identical to one used for the elastic net with SIRC segmentation. The results are a bit lower reaching $R^2 = 0.612$ on the training set and $R^2 = 0.610$ on the test set. In figure 4.3 we can see its dispersion plot.

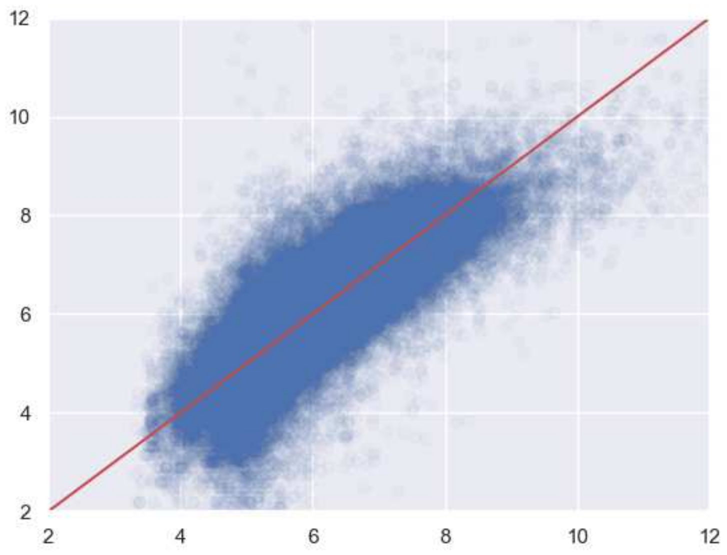


Figure 4.3: Performance of SVR (SIRC) on the test set, predicted values are on the x axis and true values on the y axis

4.2 Complex Modelling

Even if the former model was a linear one and all the requirements asked by the business side asked for a simple linear model I chose to estimate at least one complex model for this problem. The reasons for this choice are multiple. The first is to have a benchmark to evaluate the performances of simpler models. A complex model, such as tree based gradient boosting or neural networks, even with all its limitations in terms of interpretability should be able to extract the great majority of knowledge from the data set, allowing us to assess how much information we are sacrificing in the name of a simpler and interpretable model. This is in my opinion a very important KPI since could lead to a change of policy from the business side in the case of a considerable difference in performance. In the second place, as we saw in the Methods chapter, in the last years there has been an intensive research in the field of interpretability of complex ML models and this was a good opportunity to showcase to the business side the new opportunities that this kind of solution can bring to table.

4.3 Gradient Boosting

For the gradient boosting model (implemented using [4]) the preparation of the data set is a bit different from the linear models. Namely we are not segmenting so we use both SIRC and BM varibles as one hot encoded features, the algorithm will then choose which one to split on in each iteration. We are not using the transformed version of the number of employees and local units because there would be no point since the ordinal relation is all that matters, and for the same reason there is no need for scaling in this case. In this case there where 4 hyper parameters to tune that heavily interact with each other so to make things easier I decided to estimate them via cross-validated random grid search (see [5] for further details) The results of this process are shown in figures 4.4,4.6 and 4.5 with the selected values highlighted in red.

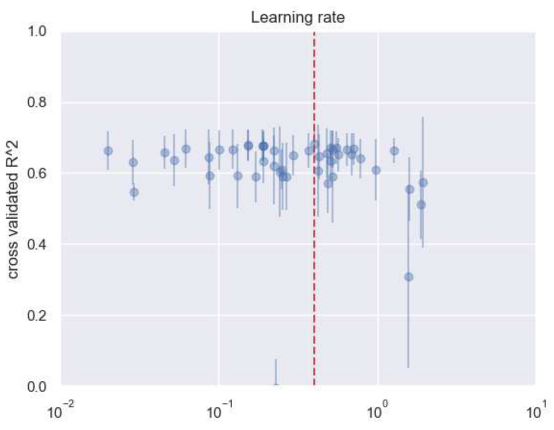


Figure 4.4: Random search results for the learning rate hyperparameter

In the heatmap in figure 4.7 we can see the interaction between the number of estimators and the learning rate; it is clearly visible a better performing region with an elongated shape.

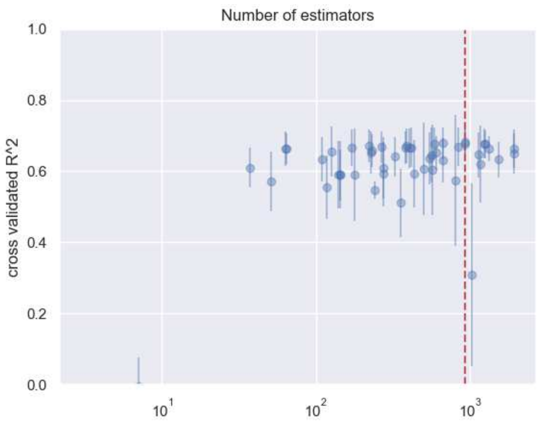


Figure 4.5: Random search results for the number of trees hyper parameter

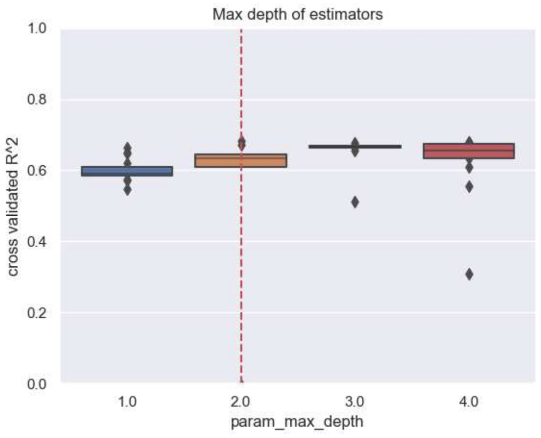


Figure 4.6: Random search results for the maximum depth hyper parameter

The shape of this region highlights the fact that to keep high performance when increasing the number of estimators we need to decrease the learning rate, which is a well known behaviour of ensemble models but this maps also gives quantitative information about what range of values of learning rate is appropriate for a given number of estimators in this specific case. The performance reached by this model is of course higher with respect to the simpler models we saw earlier: $R^2 = 0.706$ on the test set and $R^2 = 0.725$ on the train set. As is the model however does not meet the interpretability requirement that is asked for this task. For this reason in order to make it a valid candidate I applied the SHAP algorithm to it to make it more interpretable. I show below the two types of analysis we can carry out with SHAP values.

The results shown in figure 4.8 are the SHAP values relative to a single observation and they

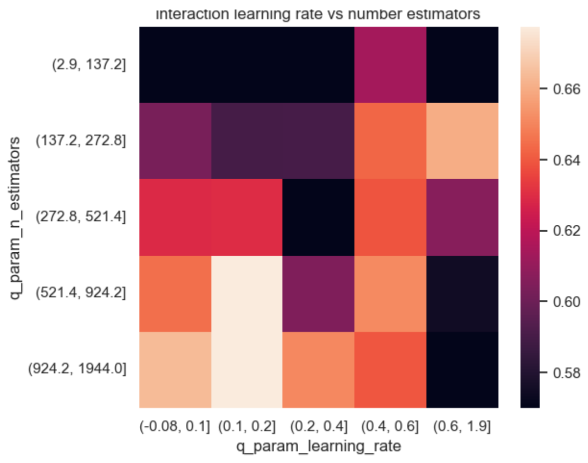


Figure 4.7: Interaction among learning rate and number of estimators hyper parameters

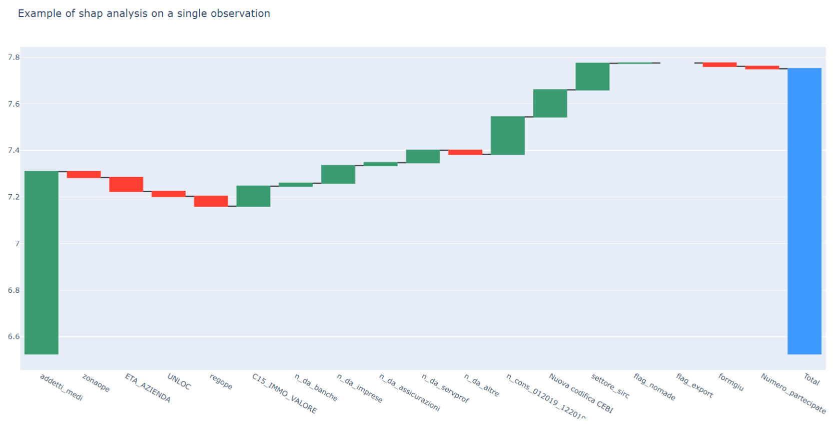


Figure 4.8: SHAP values for a single observation

show, starting from the expected value how much and in which sense each feature contributed to the final result. Of course this can be carried out potentially for every observation for interpretability; the problem is that, while in a linear model the weight and verse of the contribution of each feature can be established just by looking at the model and knowing the feature scale, in this case we have to execute the SHAP algorithm which heavily depends on feature interactions, making it a sort of "reactive" explainability.

Figure 4.9 shows the result of a feature importance measure we can retrieve from SHAP values. This is obtained by applying the algorithm the a wide population of samples and the for each feature compting the mean of absolute values of the SHAP values related to it. The absolute value is necessary because the expected value of the SHAP values is 0 since they represent a deviation from the expected value. Similarly to Gini importance this kind of analysis does

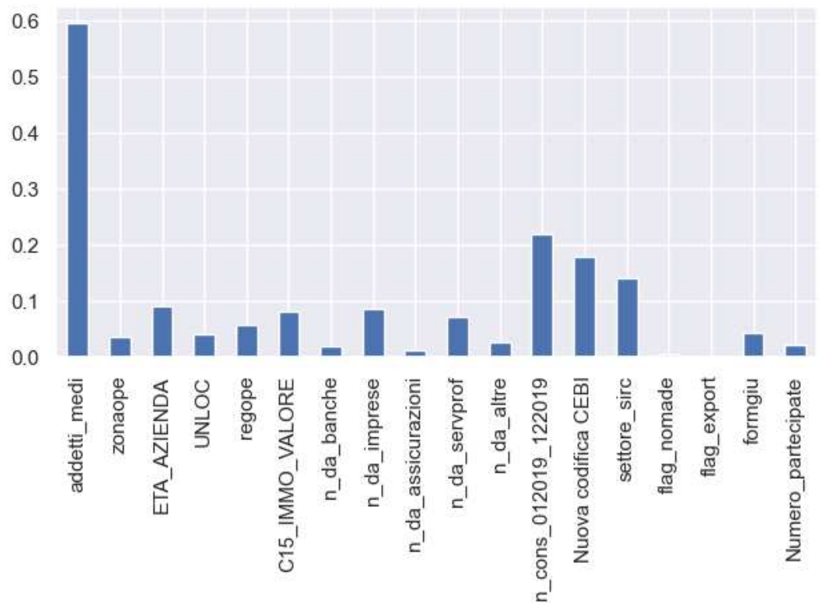


Figure 4.9: Feature importance with SHAP values

not provide information about the verse of the impact but only a general weight of the feature in the outcome of the model. For instance we could interpret the value 0.6 associated to the number of employees as "on average the information about the number of employees changes the outcome by 0.6". Note that since the these values are additive we can sum those related to the same domain of information in order to have give an importance to the whole area; namely we could sum up the values of region and geographical zone in order to get the importance of geolocalization of a company in this specific problem. Analyzing figure 4.9 we can assess that, as expected, the number of employees is the most important variable, followed by the business sector (given by the sum of SIRC and BM) and consultations data. There are a couple of interesting considerations to point out here. The first is that the BM encoding obtains a higher weight with respect to SIRC; this means that with the actual population of the dataset we are still able to exploit BM encoding better than the SIRC one even if this may change in the future due to the shrinking of the data base population. The second is the astonishing importance of consultations: the diffusion of Cerved's B2B tools in the Italian economic tissue is so deep that just gathering the data about searches gives approximately about the same amount of information that knowing the business sector of the company we are analyzing.

4.3.1 Geographical Analysis

As a side analysis I thought it would be interesting to see how different sectors' turnover varies on the Italian territory. Let's first frame the problem and how I tried to answer it. To carry out this kind of analysis I exploited the trained elastic net model with the SIRC segmentation. The model includes both the operative geographical zone and the geographical region in which the company operates as one hot encoded variables. Zones are five large geographical areas each including two or more of the twenty Italian regions; each zone should have, for historical and geological reasons, a different economic tissue that could lead to different performance of similar companies. The best way to look at this data is form a regional perspective, for this reason I summed the coefficients related to each regional dummy to those related to the corresponding zone dummy in order to have the complete geographical information in a single parameter.

The questions we would like to answer with this analysis are two:

- What are the sectors for which the geographical location of the company has a greater impact on their turnover?
- Do geographical zones actually define area of different performance for similar companies?

The first question is pretty straightforward: it is enough to see for which sectors the geographical dummies' coefficients have the highest variance. It is useful to remember that this model has a L1 regularization component meaning that dummies related to regions with a low population will most likely have a coefficient equal to zero. The SIRC three segments with the highest geographical dependencies according to this analysis are: means of transport related business, electronics and IT and fuel, energy and utilities. The less influenced instead are: distribution, non-financial services, information communication and entertainment. It would be useful now to take a closer look at how these geographical coefficients are distributed on the territory both to gain some extra insight on the phenomenon we are modelling and to try to give an answer to the second question.

First it is useful to remind that the subjects of this study are very small companies and that these geographical coefficients refer to the difference in turnover given that two hypothetical companies share the same set of the other parameters. There are a couple of observations worth to be pointed out on these examples. The first is that there seems to be some clusters in some of the distributions of the coefficients but there is not a clear correspondence with the geographical zones. For instance the metal processing sector (4.14) and the electrical engineering and IT (4.11) one have higher turnover in the most industrialized zones, the construction sector

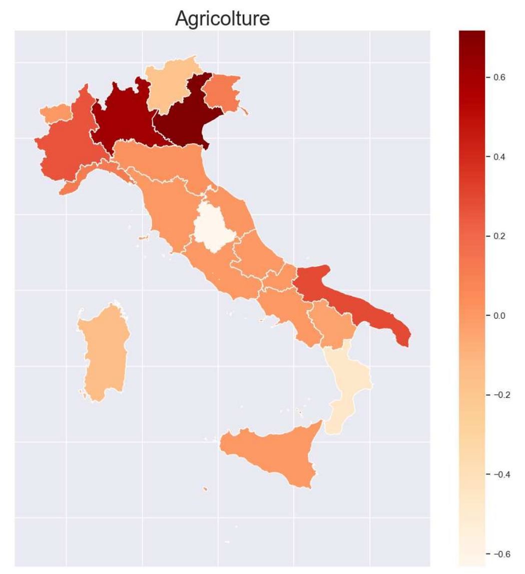


Figure 4.10: Geographical coefficients for agricolture SIRC sector

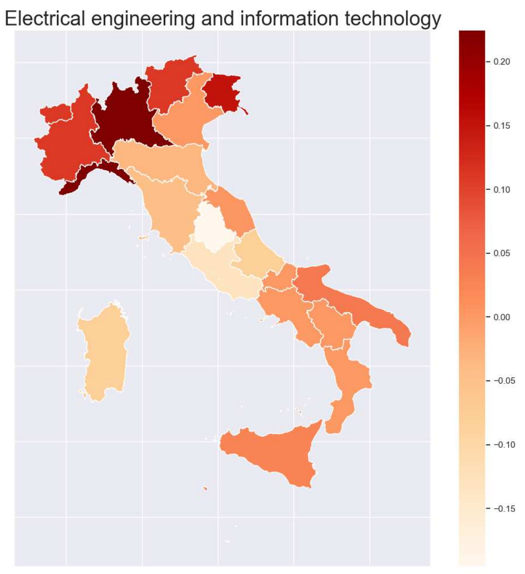


Figure 4.11: Geographical coefficients for electrical engineering and information technology SIRC sector

(4.12)on the other hand seems to have increased turnover in the south with the exception of Lombardia in the north but fuel,energy and utility sector (4.13) does not follow any apparent pattern. The second one is an observation about the interpretation of these coefficients. Let's take a look to the map related to the agriculture sector (4.10): we can clearly see that there are four regions (Piemonte, Lombardia, Veneto and Puglia) associated with higher coefficients

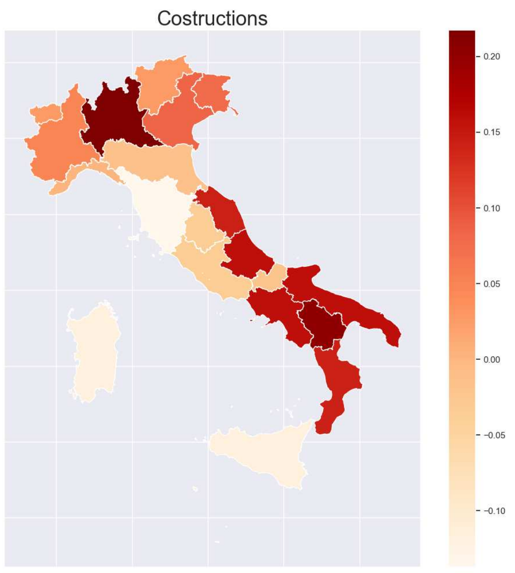


Figure 4.12: Geographical coefficients for costructions SIRC sector

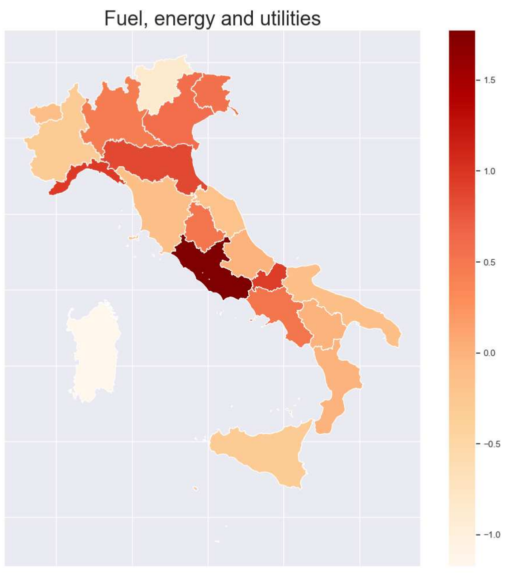


Figure 4.13: Geographical coefficients for fuel, energy and utilities SIRC sector

with respect to the rest; the reason of that is probably that these regions are know for the production of more expensive goods, such as olives for oil in Puglia and prized grapes in Veneto and Piemonte. The real estate sector om the other hand has a spike of turnover in Sardinia, characterized by elite tourism. This suggests a reflection about the interpretation of these coefficients: the difference in turnover of companies in different regions may not be due to different charac-

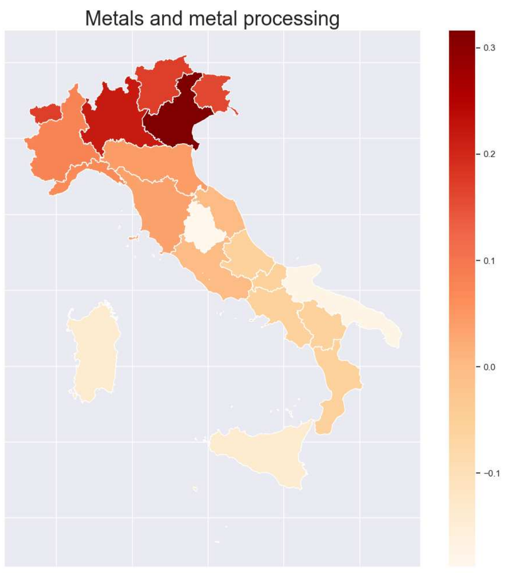


Figure 4.14: Geographical coefficients for metals and metal processing SIRC sector

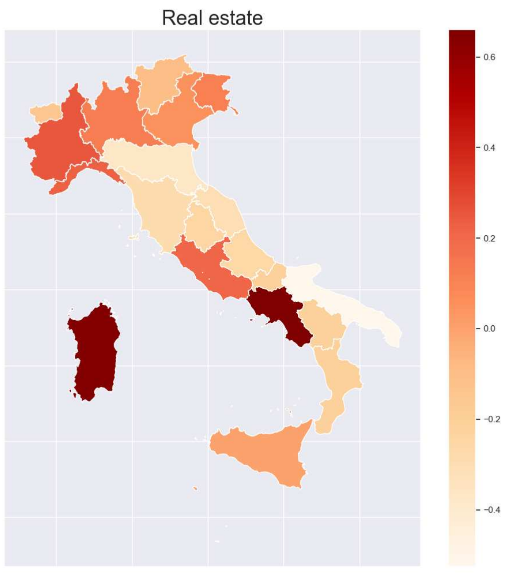


Figure 4.15: Geographical coefficients for real estate SIRC sector

teristics of the market, instead it could be explained by an alternative model of business (such as to cultivate expensive grapes instead of crops or selling luxury mansions instead of ordinary houses) that may, or may not, be transplanted in other regions.

5
Results

In this chapter we will compare together the results of the models we estimated and the model that is currently in production. We will start from a quick comparison of the predictions of these models on the test set.

As we can easily spot from figure 5.1 all of the models I estimated during my work on this project share a problem: they tend to predict very low turnover subjects, around 2-3 log(turnover), as 5. We can spot this phenomenon looking at the small appendices present in the last row of plots at around 5 on the x axis. This phenomenon is present, even if less evident, in the gradient boosting model and the legacy model too. I was not able to explain this kind of behaviour but since it is shared even with the legacy model it seems some problem related to the world of this kind of companies that would deserve to be deepened. Another interesting insight that we can retrieve from this kind of visualization is the relationships between the models. For example it is easy to spot a high correlation between the SVR (SIRC) and elastic net (SIRC) models. We can however make the comparison easier by looking at the person correlation values in figure 5.2. With this perspective we can assess for example that the elastic net (SIRC) model's prediction are closer to the SVR with the same segmentation than to the elastic net with the zoom segmentation. Another expected but still worth to be pointed out result is that the legacy model has a weak relationship with all the others. This is imputable to a few reasons:

- **Different samples** of course the base that was used to estimate the model was completely different with respect to the one I used.

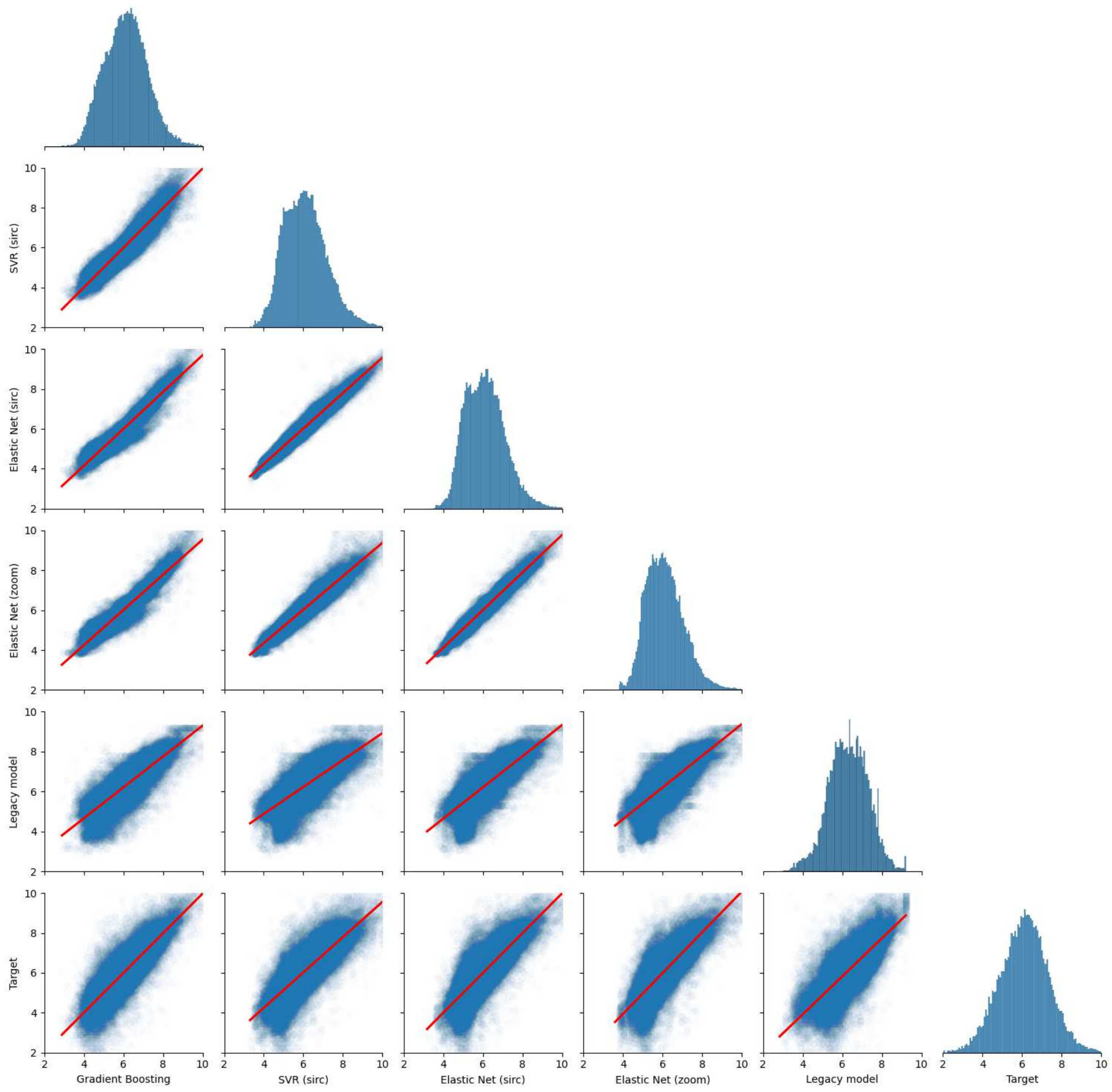


Figure 5.1: Comparison of the prediction of the different models on the test set

- **Overrides** the old model has build-in some overrides to explicitly prevent extreme output (this can be spotted in its scatter plots especially towards high values).
- **Different features included** the old model exploited a different set of features, including registry information about the company owner that were removed due to new policies on ML model.

Set	Elastic Net (Zoom)	Elastic Net (SIRC)	SVR (SIRC)	Gradient Boosting	Legacy
Train	0.665	0.659	0.612	0.725	0.552
Test	0.665	0.656	0.610	0.706	0.553

Table 5.1: R square comparison.

- **No segmentation** the old model was not segmented along the business sector.

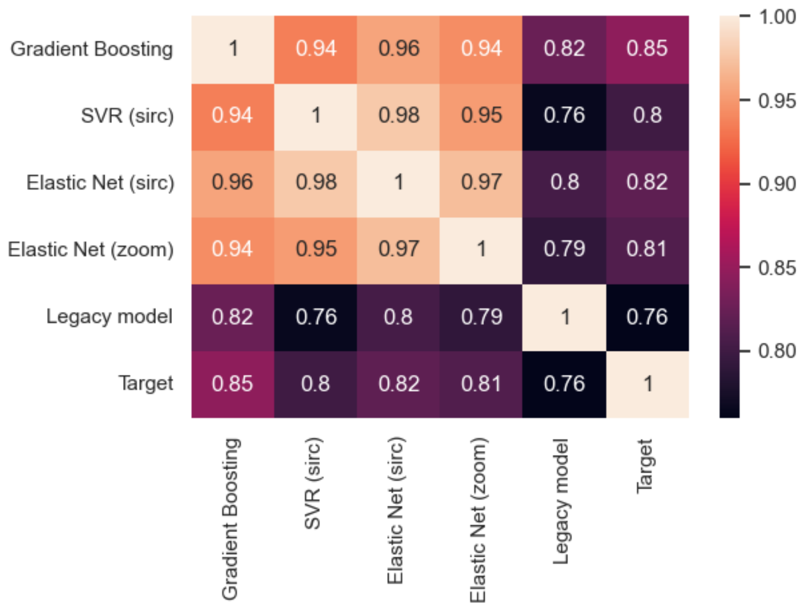


Figure 5.2: Comparison of the prediction of the different models on the test set via Pearson correlation

As we can see from the table 5.1, elastic net models reach better performances with respect to SVR, gradient boosting outperforms elastic net models with a solid margin while the legacy model falls significantly behind in terms of performance.

6
Conclusion

In conclusion this scouting was considered successful by my supervisors. The enhancement of performance, the refining of the feature set and the general improvement in terms of technology of the models were considered sufficient to bring the ideas born in this project in production. It has not been decided yet whether an elastic net model or the gradient boosting one will be chosen to replace the legacy model. Before the chosen model will actually be integrated in production there will be however some additional analysis to carry out in order to guarantee the stability of the model and its robustness. For what concerns further improvement opportunities I would suggest as main "hot topics" a further investigation on the low turnover companies phenomenon, and the further investigation and integration of data coming from Cerved's business information market. This is particularly relevant if we think that the sole number of consultations reached in the gradient boosting model a relevance (computed with SHAP) similar to the one achieved by the business sector variables.

References

[1] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*, ser. Springer Series in Statistics. New York, NY, USA: Springer New York Inc., 2001.

[2] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Curran Associates, Inc., 2017, pp. 4765–4774. [Online]. Available: http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf

[3] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[4] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '16. New York, NY, USA: ACM, 2016, pp. 785–794. [Online]. Available: http://doi.acm.org/10.1145/2939672.2939785

[5] A. Géron, *Hands-on machine learning with Scikit-Learn and TensorFlow : concepts, tools, and techniques to build intelligent systems*. Sebastopol, CA: O'Reilly Media, 2017.

Acknowledgments

I would like to thank both professor Guidolin and Dr. Tessiore that followed me on this project respectively from the university and the company side. Together with the others colleagues in CeBi they helped me, despite a considerable number of adversities, to conclude this parenthesis of my life. I would also like to thank CeBi for believing in me as a professional and allowing me to work in a stimulating environment on challenging projects.