Bachelor Thesis

Enhancing Business Strategy Through Machine Learning Techniques in The Telecommunication Industry

Ivan Al Khayat
Artificial Intelligence

Academic Year 2023/2024

Bachelor in Artificial Intelligence
Pavia 2025

Bachelor in Artificial Intelligence

Ivan Al Khayat

Enhancing Business Strategy Through Machine Learning Techniques in The Telecommunication Industry

Supervisor: **Professor Matteo Zignani**

Abstract

This thesis aims to describe how artificial intelligence can be employed to develop a better selling strategy for a telecommunication service already on the market, deployed by an Internet Service Provider (ISP) company across the Italian national territory. More specifically, it is in the context of an MVP (minimum viable product), an essential step before scaling a solution to a production environment.
The analyses start from the collection of public data from ISTAT and company's internal data, which are combined to make predictions about sales in Italian municipalities, and consequently improve the efficacy of the sales campaign.
The adopted model is a random forest regressor and the statistical unit is each Italian municipality. The covariates of the model are the characteristics of the cities. The target variable is the run rate, a metric used in finance to predict future revenues. The run rate is not flawless because it assumes that current performance levels will remain constant over time, ignoring potential fluctuations and changes. Since the magnitude of the root mean squared error (RMSE) of the model is comparable to the magnitude of the target variable, the performance of the random forest regressor is unsatisfactory, in both training and test sets, indicating the model is underfitting. This scenario can occur when the data quality is poor or the dataset is not large enough to effectively train the model.
Lastly, explainability plays a key role in understanding how a model made a certain prediction. The impact of each given feature, both locally and globally, is assessed by presenting SHAP values in different types of graphs.
Future developments can include wider hyperparameter tuning or even considering more robust models such as gradient boosting regressors, collecting and including more representative data (for example about users traffic) to provide a model that can be deployed in an operational environment.

Keywords

Machine Learning, Ensemble Methods, Regression

Contents

Chapter 1
Introduction

The context of this thesis is an internship in a consulting company for an Internet Service Provider client. The client has developed a new internet connection technology and aims at maximising sales by making use of targeted advertisement campaign. To achieve this goal they wanted to exploit artificial intelligence in order to increase their revenues from the service in question. The proposed AI model is a RandomForest regressor with as inputs the time series of the Italian municipality characteristics on monthly basis and as output the number of new customers that bought the new Internet connection technology for that month. The creation of each record involved both public and company-owned data that were joined together. The predictions of the model were deemed not good enough because of the unreasonable output values obtained in inference. Actually public data were not always up-to-date, and the client did not provide us enough features that could have helped the model to achieve better performances. Furthermore, the hardware for model training was inadequate for complex machine learning tasks.
The thesis is structured according to the chronological phases of the project. Finally, the model results are shown and discussed.

1.1 Problem Formulation

The functional goal of this project is to help the client increase the sales of a service by focusing their marketing campaigns on specific areas. To do this, a regression problem is being formulated, namely, predicting the run rate of the company's product, which is given as a continuous target variable.

1.2 Outline

The next chapters of the thesis are organised as follows:

Chapter 2 Data Understanding and Exploratory Data Analysis: the next chapter goes in depth into a very important phase in any data science project. The purpose is to understand the kind of given data as well as their meaning. Data are plotted with different kinds of graphs so to better understand their distributions.

Chapter 3 Model Creation And Training: this chapter describes the details of the machine learning algorithm chosen for the proposed tasks. It starts with some general introduction about how a decision tree works and how to combine multiple of them to get better predictions, technique called ensemble learning. Finally the chosen evaluation metrics is mathematically described and some some comments on the obtained results are made.

Chapter 4 Explainability: this part shows why it is important to have an explainable model. Explainability is treated using shapeley values, a concept derived from game theory. The mathematical equation modelling the strength of each feature contribution to the final outcome is described. Finally it is clarified the differences between local and global explainability.

Chapter 5 Conclusions: in the conclusions chapter there are some considerations including further works as well as some criticisms about the whole project. It starts with a quick review of project goals. The work environment is then described to give the context where the team had to work in.

Chapter 2
Data Understanding and Exploratory Data Analysis

Data understanding is the process whereby consultants and clients agree upon the data model. A data model organizes data structures and standardizes their relations.
With the aim of predicting new potential customers across the Italian municipalities, two families of variables have been considered: endogenous and exogenous.
Exogenous variables, describing the 'environmental' context, are taken from the Italian Institute of Statistics (ISTAT), whereas endogenous variables, describing the company's trends, are provided by the client.
Finally all pieces of information are joined together to obtain a single table to feed an algorithm; hopefully being lightweight, efficient, and explainable as much as possible for predicting a target variable. To optimize model performances, a nice-to-have factor is getting enough historical data to ensure having relevant information to feed the model.

As a second step, there is the exploratory data analysis (EDA), to mainly check distributions and correlations, later detailed more in-depth. Both stages are crucial since data quality is key to determining the performance of the predictive model. The final purpose is to forecast the run rate of a service.

2.1 Target variable overview:

In the finance sector, the run rate (the target variable of this project) is a metric used to estimate future performances based on current activity. This indicator is particularly useful when a company needs to forecast revenues for upcoming months, especially if it has less than a year of historical data available, or when significant changes have occurred in the business strategy (e.g., updating product prices, restyling, rebranding, adopting new technologies, etc...).

In this case, by **run rate**, it means the number of new potential customers that will purchase the service. Since the service is priced with a flat fee for each new customer, then a direct relationship can be established between new customers and revenue. What is typically done is to extrapolate one month of revenue data and project it onto future months. This projection is often achieved using autoregressive models or similar forecasting techniques to gain insights into potential sales trends. However, the run rate metric is not flawless. Seasonality is a factor contributing most to the run rate variability. For example, consider the scenario of a children's toys company using sales figures from the Christmas period as a basis to predict the following year. In such a case, there is a high risk of experiencing an overly optimistic sales forecast. [5]

2.2 Exogenous variables:

Relevant data can be retrieved from the ISTAT periodic publications, as they provide valuable information about public entities such as municipalities, provinces, regions, and population-related information with regard to education level, average income, and family members.
In early times ISTAT published a census every decade up to the year 2011, but from 2018 onwards part of publications, more specifically the `Censimento permanente della popolazione e delle abitazioni`, became yearly.

How does the census happen? It occurs in two modalities: list survey (`rilevazione di lista`) and areal (`rilevazione areale`) survey. In both cases, personal privacy is guaranteed by statistical secrecy, Italian law and GDPR.
To provide an example, in the 2021 census, the list survey consisted of interrogating some families among a list of 2400 selected municipalities. 1143 cities are always involved annually, while the others are involved on rotation. The number of families extracted was approximately 950,000. The final goal is to gather information about family members' habits and their living places. Surveyed people must be residents of the municipality. The surveyed families are notified with an official letter including all the needed instructions on how to fill out the electronic survey.
The `rilevazione areale` survey, instead, consists of sampling families from some locations referenced in the 'Registro Statistico dei Luoghi'. The main purpose of this method is to obtain the number of resident people in each municipality. It involved nearly 2800 selected municipalities annually from 2018 to year 2021, among which almost 1100 municipalities are regularly involved and the others are sampled rotationally [4].

2.3 Endogenous variables:

Endogenous variables are data owned by the client and, since it is an ISP their detail is strictly confidential, thus not disclosable.
Why and how is this kind of data stored? In general, ISP companies collect data concerning end-user traffic and customer-related data. ISPs retain such information for both legal and other purposes, always complying with current regulations.
The measure of 17 January 2008 named 'Security In Telephone And Internet Traffic Data' published by the Italian authority 'Garante per la protezione dei dati personali' , more precisely the article 7.3, states "Traffic data that are retained exclusively for the purpose of detecting and suppressing criminal offences must be processed with the help of IT systems that are physically different from those used to manage traffic data for other purposes. This applies to both processing and storage components" [10]. The other purposes mentioned in the measure concern scopes such as billing, marketing and fraud prevention; this is where artificial intelligence is increasingly employed to optimize business performances through statistical and machine learning algorithms.

2.4 Exploratory Data Analysis:

The role of Exploratory Data Analysis — EDA — is to retrieve the main characteristics of data (eg. getting mean and standard deviation of distributions, median values etc..) and it also involves data visualization techniques. In the exploratory data analysis stage, data scientists can comprehend the goodness of retrieved tables (eg. well-formatted dates, no duplicate primary keys, mandatory fields without missing values etc..).

Why is data integrity important? The datum must be reliable for scientists, who must be able to perform subsequent computations based on it, in order to avoid inconsistent operations such as divisions by zero. In case of data inconsistencies, some data cleaning operations must be performed and possibly agreed with the client (eg. data imputing,

rows and features removal, formatting, etc...).
The main technical tools used to perform EDA are `pandas`, `numpy` and `seaborn` Python modules, updated at their latest release. These modules are completely open source.

Once data integrity is verified, the next step is to perform some feature engineering by merging tables, also called DataFrame in `pandas`, through "join" operations. Join operations are very common in databases, they are aimed at constructing a single table with data coming from multiple ones based on a related column or set of columns. This operation in waterfall allows retrieval of data that span across several tables in a relational database system.

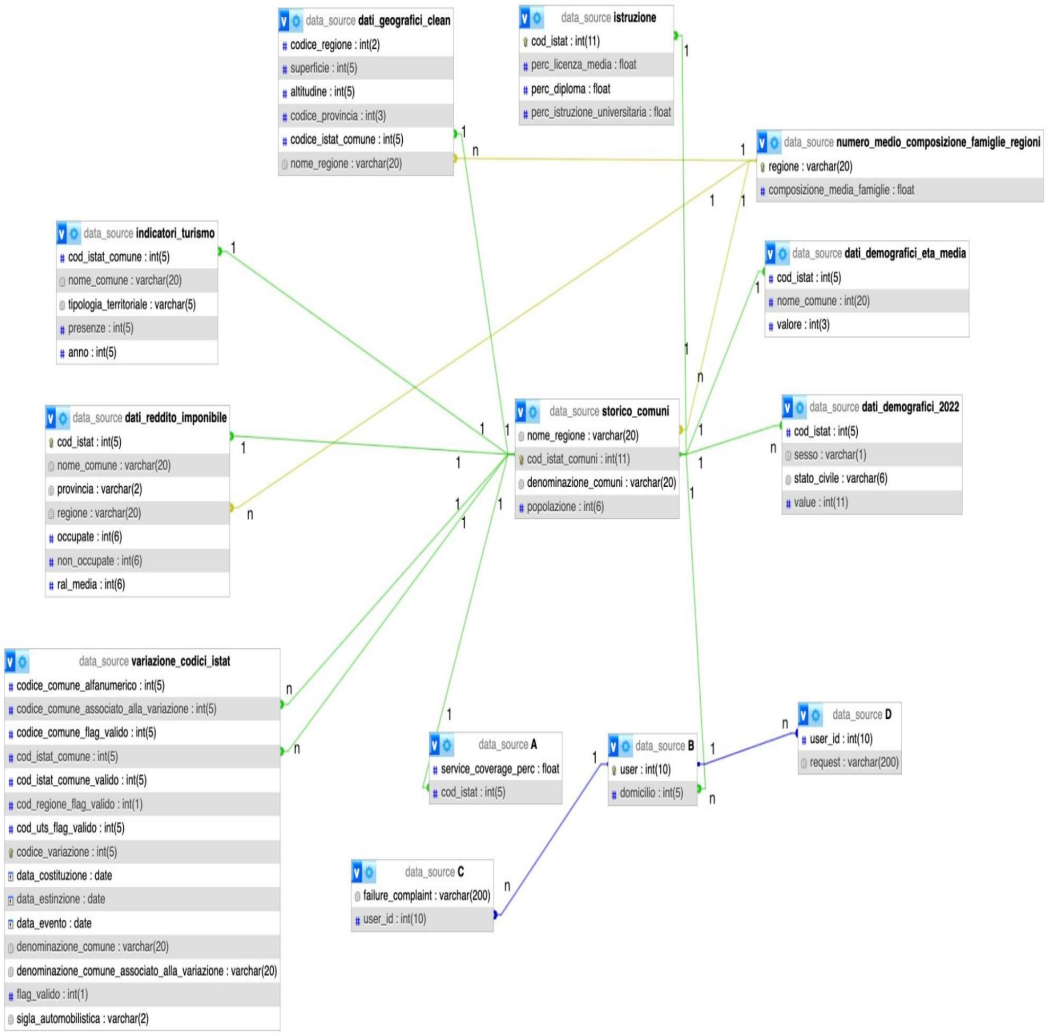

Figure 2.1: The above ER diagram is the result of data cleaning and feature engineering of both ISTAT and internal data.

In the setting of this project the statistical unit is each municipality, represented by the table "storico_comuni" in the Entity-relations (ER) diagram; its primary key is the field "cod_istat_comune" which is a unique code identifying a municipality. Since each municipality can undergo changes such as suppression or union with another town, there is the table "variazione_codice_istat" to keep track of these changes. In relation to the municipalities there are data about average income (table: dati_reddito_imponibile), annual tourism (table: indicatori_turismo), territorial morphology (table: dati_geografici_clean),

education (table: istruzione), age (table: dati_demografici_eta_media) and marital status (table: dati_demografici_2022).

Although previously mentioned tables could be part of “storico_comuni“ (it would be the best logical grouping), they are separate elements because the ISTAT website allows to download such information in the form of separate tables, leaving analysts to do further jobs such as merging or cleaning. Each municipality has an associated region and the region registry data is found in “numero_medio_composizione_famiglia_regione“ table. The average number of family members is at a regional level and this datum has been incorporated in each municipality instance. Additional data come from internal tables named A, B, C, and D (aliased names and fields because of stipulated NDAs) containing data about service coverage, customers, complaints, and requests respectively. Relevant columns have been added to each municipality.

A further step is to understand the distributions of the covariates and to find correlations among them; this way it is possible to detect patterns, outliers and possibly insights. To verify the distribution of the covariates the most common methods are histograms, QuantileQuantile plots, cumulative distribution functions, scatterplots, and boxplots. Hereafter is a description of each:

- Histogram: this graph shows straightforwardly the distribution of observations grouped in bins to count the occurrences within each.

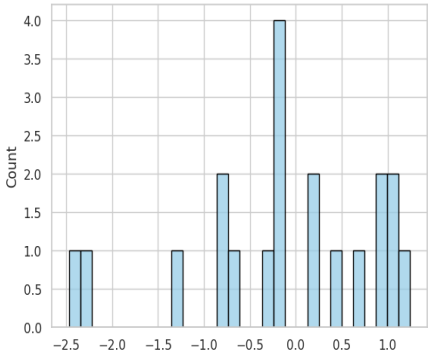

Figure 2.2: histogram plot

- QuantileQuantile plots: abbreviated as QQ-plots, it is a graphical representation to compare the distribution of a variable with a common one (such as normal, uniform, Poisson, etc..) or to compare the distribution of variables. In case the observations lay in the main diagonal, the red line, it means that the data follow a theoretical distribution.

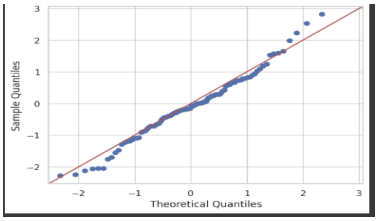

Figure 2.3: QQ plot

- Cumulative distribution functions: This graph shows the probability $P(X \leqslant x)$ of a feature, taken as a random variable, that the observations of a variable are less or equal to a specified value.

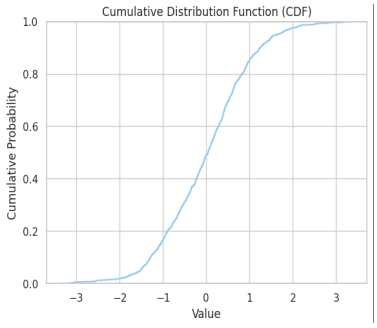

Figure 2.4: cumulative distribution function plot

- Scatterplots: This graph involves a dependent and independent variable, very helpful in visualizing relationships between variables such as non-linearities.

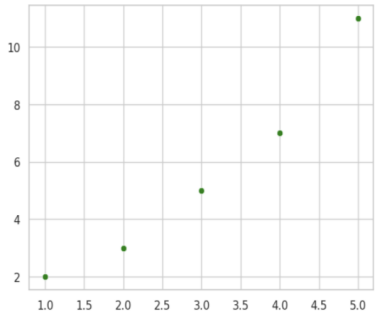

Figure 2.5: scatterplot

- Box Plots: also known as Box and Whiskers plot, it focuses on showing the distribution of a random variable made of a box (box extending from quartile 1 to 3 containing the median, remembering that each quartile has the same number of data points), the whiskers (showing the min and the max value, as long as the are between the fences) and the fences, where values outside of them are potentially outliers. Fences are located at a distance of 1.5 the size of the box (interquartile range) from the quartiles themselves.

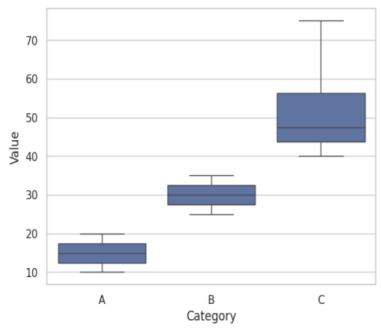

Figure 2.6: boxplot plot

2.5 Correlation analysis:

In the exploratory data analysis stage, it is often useful to see correlations between variables. It is well known that a model with highly correlated features will perform poorly. That is basically due to the fact that, when that is the case, most of the columns give the same information as any other, thus not adding anything new to the dataset; moreover, it would also cause problems of interpretability in linear models due to collinearity (we would not know which covariate is responsible for the change in the target).
A first naive collinearity check is to assess whether the rank of the matrix with covariates is equal to the number of columns, if so, we are sure that features are linearly independent.
However this check is weak because in the case of 2 columns having all value pairs linearly dependent except for a single couple, then the columns are formally linearly independent even though the almost totality of values are linearly related.
A more robust approach is the use of the Pearson correlation coefficient. It is designed to measure the linear correlation between 2 covariates. It takes values in the range [-1, 1], with the extrema meaning respectively very strong negative and very strong positive correlation, implying high predictability.
If instead this coefficient is zero or almost zero, then there is not a linear relationship between the covariates. Hereafter the formula for the computation of the Pearson correlation coefficient:

$$\rho_{X,Y} = \frac{\text{cov}(X,Y)}{\sigma_X \sigma_Y}$$

Where:

- $\rho_{X,Y}$ is the Pearson correlation coefficient.
- $\text{cov}(X,Y) = \frac{1}{N}\sum_{i=1}^{N}(x_i - \mu_X)(y_i - \mu_Y)$ is the covariance between variables X and Y.
- $\sigma_X = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu_X)^2}$ is the standard deviation of X.
- $\sigma_Y = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \mu_Y)^2}$ is the standard deviation of Y.
- μ_X and μ_Y are the means of the X and Y populations, respectively.
- N is the number of data points in the population.

A more advanced correlation measure is the Spearman correlation coefficient. This measure is non-parametric meaning that there are no a priori assumptions on the population, thus being more flexible. It is good to resort to it to detect monotonic relationships between variables.
Spearman correlation coefficient assumes values in the range $[-1, 1]$. With values near -1, it means that there is a monotonically decreasing relationship, and for +1 there is instead a monotonically increasing relationship. When it is said that Spearman correlation is "rank-based", it means that it measures the strength and direction of the relationship between two variables based on the ranks of the data values rather than their actual numerical values. The formulation of the Spearman correlation coefficient is the following:

$$\rho_{X,Y} = 1 - \frac{6\sum d_i^2}{N(N^2-1)}$$

Where:

- $\rho_{X,Y}$ is the Spearman correlation coefficient.

- $d_i = R(x_i) - R(y_i)$ is the difference between the ranks of corresponding values x_i and y_i.
- $R(x_i)$ and $R(y_i)$ are the ranks of x_i and y_i, respectively.
- N is the number of data points in the population.

The Spearman correlation coefficient is also used to check correlations between covariates and the target variable to see if there is any feature that can be assumed as a proxy to have a rough estimation of how the target will look like. In this project, the Spearman correlation coefficient is the most suited since we want to discover not only linear relationships; justified by the fact that most of the data are exogenous and it is usually hard to see direct correlations with the kind of target variable we have established that is sales-related.

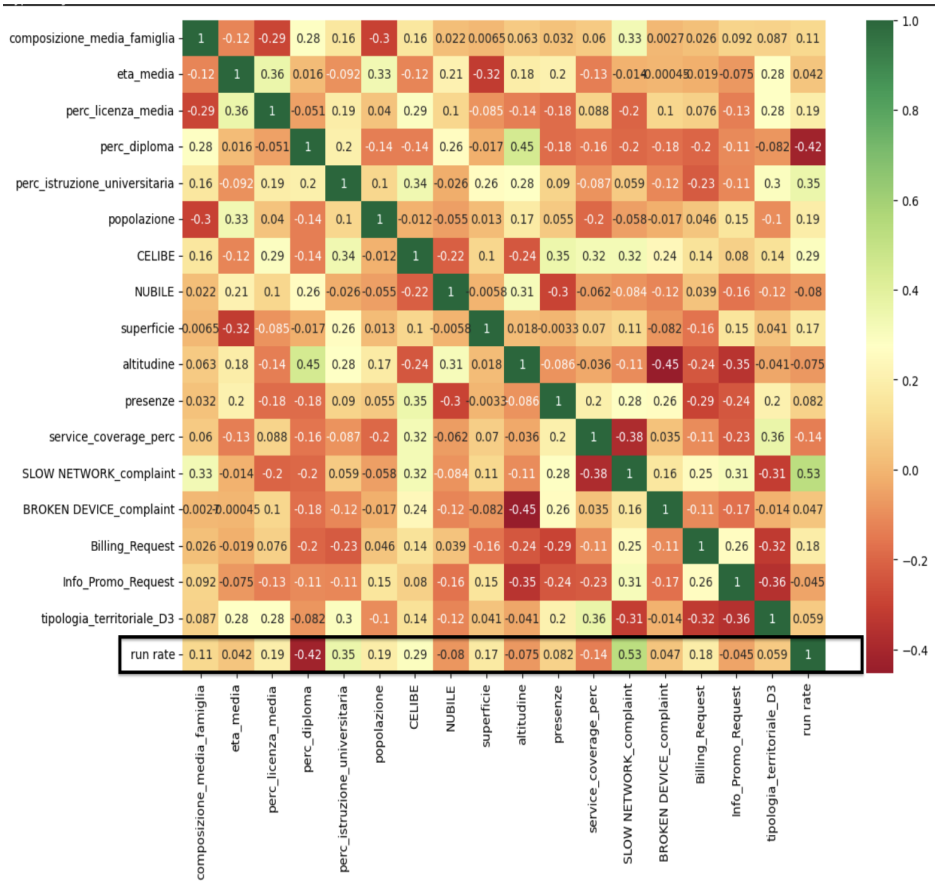

Figure 2.7: correlation matrix between features and target.

The boxed row of the correlation matrix shows a green tile. It contains the correlation value between the "SLOW_NETWORK_complaint" and the run rate. It is worth noting that the correlation coefficient ranges from -1 to +1. The fact that there is a 0.53 value for the feature mentioned above indicates that in general, the run rate grows as the number of slow network complaints. This can be explained as follows: if a municipality experiences a high volume of complaints about their internet connection, these issues may be addressed by switching to a more advanced service, for example, the one treated in this thesis.

Chapter 3
Model Creation And Training:

3.1 Feature scaling:

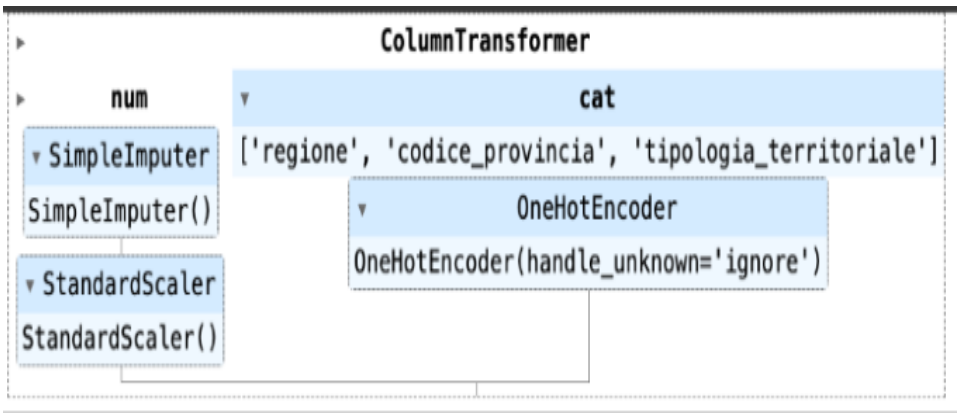

Figure 3.1: pipeline for transforming features that will feed the random forest regressor model.

In this project, it has been used the pipelining mechanism provided by `scikit-learn`, as shown above. It is a comfortable way to design machine learning models that may include data preprocessing, model selection, and hyperparameter tuning steps. More specifically, in our case, numerical and categorical features undergo separate preprocessing steps. Hereafter the meaning of each numerical feature given to the model:

- composizione_media_famiglia: mean number of members in a family at a regional level and assigned to each municipality belonging to the same region;
- eta_media: residents' average age of a municipality;
- perc_licenza_media: percentage of resident people holding at most an EQF 2 level;
- perc_diploma: percentage of resident people holding at most an EQF 4 level;
- perc_istruzione_universitaria: percentage of resident people holding an EQF greater or equal to 6th level;
- popolazione: number of resident people in a municipality for at least 6 years;
- CELIBE: number of male people not married in a municipality;
- NUBILE: number of female people not married in a municipality;
- superficie: municipality surface area in squared km;

- altitudine: altitude, distance above sea level in meters;
- presenze: numer of yearly stays in a municipality.
- service_coverage_perc: part of a municipality covered by the service (aliased field)
- SLOW NETWORK_complaint: this feature is derived from table C by manually clustering the field "failure_complaint"(aliased) which contains a free text area, so to take the most frequent cases. The 2 most frequent clusters identified were SLOW NETWORK (aliased) and BROKEN DEVICE (aliased); so for each municipality, the count of those kinds of complaints will be model features.
- BROKEN DEVICE_complaint: same as above;
- Billing_Request: similarly to what is done above, the most frequent types of requests are identified by manually clustering contents of the field "request" (aliased) from table D. The 2 main types of requests have been identified, and to feed the model, they will become features where the values for each row correspond to the number of requests opened in a municipality of category Info_Promo (aliased) and Billing (aliased).
- Info_Promo_Request: same as above.

It is remarked the fact that all municipality (row) identifiers have been removed because they bring no inferential information since they are all different.

As a first step on the pipeline, there is the imputation; it is a process whereby missing values are filled according to specific criteria that can be either a default value or something data dependent, such as mean, median, or most frequent datum based on the feature distribution.

When there are features expressed in different scales, for example, magnitudes like 10^2 versus 10^8 , many machine learning models would train slowly due to the strong computational effort. Additionally, in the case of linear models, the presence of features with high magnitudes compared to those with low magnitudes could negatively affect the model's performance, as the high magnitude values may dominate the model's learning process and overshadow the contributions of features with lower magnitudes. A way to solve this problem is to perform feature scaling. The main feature scaling approaches present in the scikit-learn module are min-max scaling and standardization.

Min-max scaling works by bringing the observed values of a feature in the range [0,1]. To perform such operation the following formula is used:

$$X' = \frac{(X - X_{\text{min}})}{X_{\text{max}} - X_{\text{min}}}$$

Where:

- X' is the scaled value.
- X is the original value.
- X_{min} is the minimum value of the feature X.
- X_{max} is the maximum value of the feature X.

However, min-max scaling is well suited when it is known the space where data can range (for example students' grades), and unfortunately, that is not our case.

The other way to perform feature scaling is by standardizing features, meaning that the values will be rescaled and "centered" at zero. More precisely, the feature will be rescaled in such a way that the mean of the distribution is 0 and the standard deviation is 1, like in a standard Gaussian distribution. If in the presence of a multivariate setting, then each feature will be treated independently.

$$z = \frac{x - \mu}{\sigma} \tag{3.1}$$

where:

- x is the original feature value.
- μ is the mean of the feature.
- σ is the standard deviation of the feature.
- z is the standardized value.

As for the categorical variables they undergo different processing stages. Categorical variables concerning our problem are:

- regione: the region where the municipality is located;
- codice_provincia: province code belonging to the province owning the municipality;
- tipologia_territoriale: land morphology (eg. hill, mountain, seaside, etc..)

For these variables the only preprocessing step is OneHotEncoding. It consists of taking each value of every categorical variable and transforming it into a dummy variable to feed the model. This way the record will have value 1 for the column corresponding to the actual sample evaluation and 0 everywhere else. The drawback of this method is that the feature matrix will become sparser and sparser, leading to model inefficiencies (in such a case it is a good choice to consider Principal component analysis). Moreover, this method was chosen because there cannot be any importance order between territories a priori. If one would want to instill ordering in categorical variables, then the way of doing such encoding would be OrdinalEncoding. That is the case of ratings expressed in natural language words, which creates a mapping between values and real numbers not creating further columns for the model.

3.2 Model Creation: Random Forest

3.2.1 Decision Tree:

The random forest is a machine learning algorithm used to perform both regression and classification tasks. In the case of classification, the scikit-learn implementation is the RandomForestClassifer and for regression it is the `RandomForestRegressor`; the latter will be used in this project. How does the random forest algorithm work?

In the Random forest algorithm there are several decision trees, which are in turn machine learning algorithms with both regression (of our interest) and classification variants. Let us explore how the decision tree regressor works. Suppose that one wants to make a prediction for a municipality where there have been more than 1.5 complaints about the slow network (i.e. field "SLOW NETWORK_complaint" > 1.5). The algorithm traverses the tree starting from the root and finally reaches the leaf node that predicts the value 9.772 for the run rate. That predicted value is the average among the 4 training instances

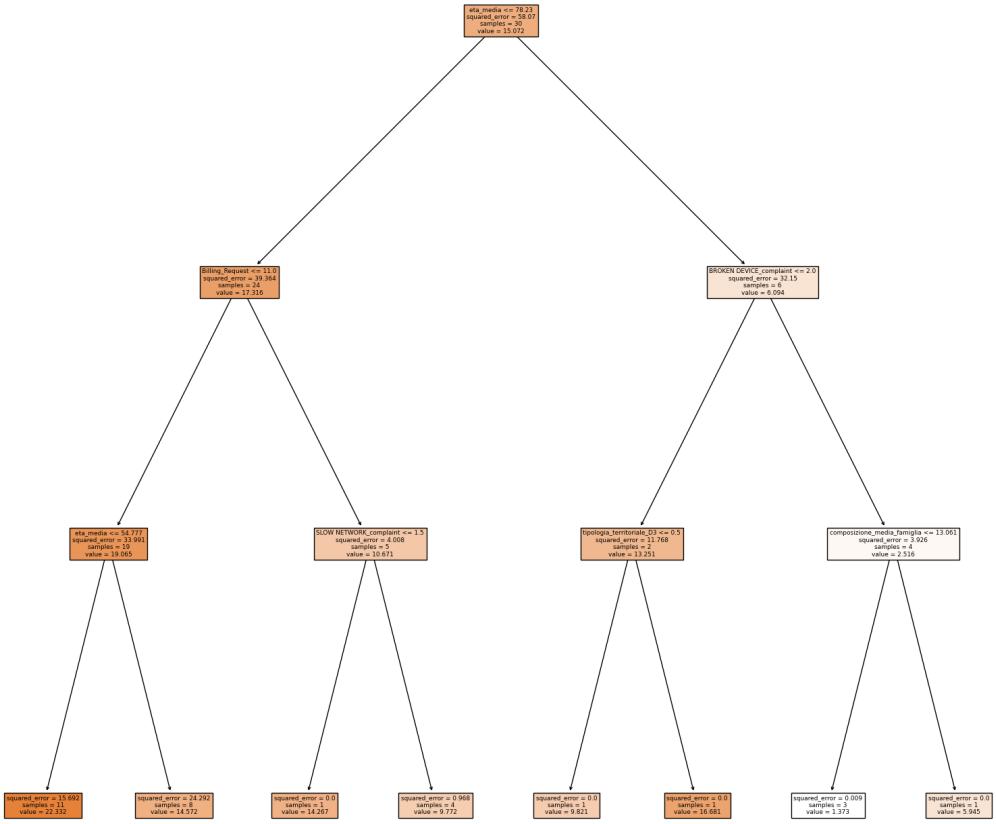

Figure 3.2: Decision tree splits, if any condition is true go to left child, otherwise go to right child. [11]

referring to the leaf node. This prediction is associated with a mean squared error= 0.968, calculated with 4 samples.

To better visualize on a cartesian plane how a decision tree is categorizing or making regression, the regression can be based on only one feature for simplicity. In the following figure, one can see what the splitting looks like, with four splits based only on the covariate "composizione_media_famiglia":

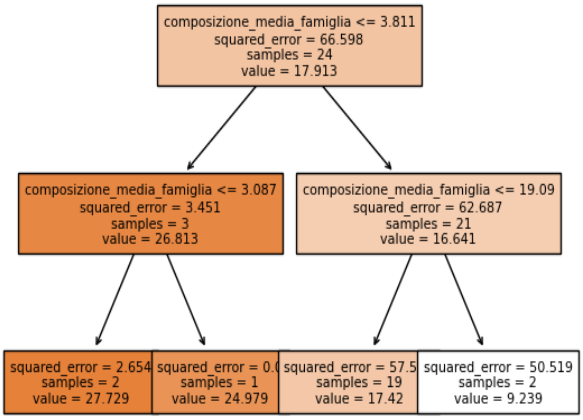

Figure 3.3: Decision tree splits simplified.

It is shown that with a tree made of three levels (i.e. depth equal to two), there are four leaf nodes, coinciding with the number of levels composing the step function created by the decision tree regressor.

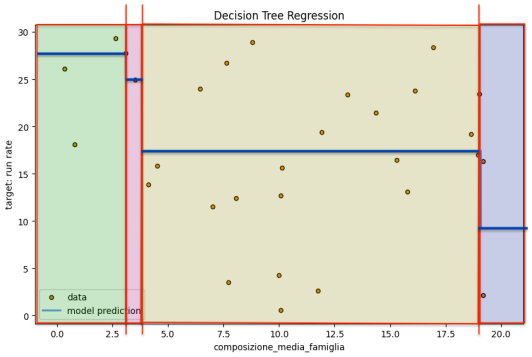

Figure 3.4: Decision tree splits with step function of four levels.

Each level of the step function shown above, in blue, is the algorithm prediction and it has been calculated by averaging the target variable for samples belonging to the same region. In general, there are $2^{(N-1)}$ (with N = number of levels of the tree, as in all binary trees), which will be the total number of different predictions that the decision tree regressor can make.

3.2.2 Ensemble learning:

Up to now, we have spoken about a single classifier or regressor algorithm, but what about having more than one to make predictions? This question encompasses the concept of ensemble learning. It is like asking a difficult question to many people (10, 100, 100 or even more) versus asking the difficult question to a single person, only. More often than not the most accurate answer will be generated by the group of people.

The same happens in machine learning, where the answers of many classifiers or regressors can be put together to answer the problem. This technique seems quite naive, but actually is very powerful, and nowadays many of the best-performing models are based

on an ensemble of classifiers or regressors. The thing to keep in mind is that ensemble learning works best when classifiers or regressors are independent of one another.

Some of the most common ensemble learning methods include voting regressors, bagging and boosting. The *Voting Regressor* specifically aggregates the predictions of each base regressor and then predicts the final output based on the average of these predictions. The output of a voting regressor can be calculated as follows:

$$\hat{y} = \frac{1}{N} \sum_{i=1}^{N} \hat{y}_i$$

where:

- $\hat{y}$ is the final predicted value from the Voting Regressor.
- N is the number of individual regression models.
- $\hat{y}_i$ is the predicted value from the i-th regression model.

Moreover, the weight of each regressor can be modified; this can be helpful when there are regressors that are typically more accurate than others within the ensemble [12]. For example, there can be three different regressors (e.g. one support vector machine regressor, one decision tree regressor, and one perceptron regressor), all trained in the entire training set, and the output of the voting will be their average.

Another powerful ensemble method is *bagging*, (shortening of bootstrap aggregation). This method consists of creating bootstrap samples, that are samples with replacement, from the training set, and on each of them, a single classifier or regressor (also called weak learner) is trained. If instead there is no replacement on the samples, then the technique is called "pasting" [2]. There can be for example three different regressors that are trained on different samples and finally, the outcome of the regression is the average of the three predictors. The most famous bagging technique is the *random forest*, which will be later treated in depth in the next section. It is important to remark the different training processes between voting and bagging methods. In the following figures 3.5 and 3.6 here is depicted the training diagram.

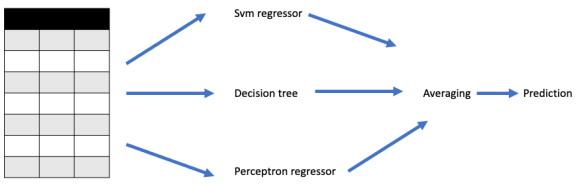

Figure 3.5: voting regressor with three weak learners.

Actually the most powerful ensemble learning technique is *boosting*. Instead of training different regressors on different data (that could be done in parallel), in boosting several

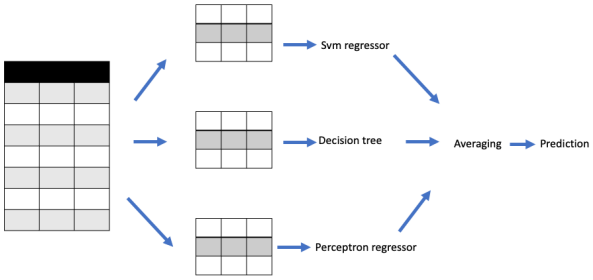

Figure 3.6: Bagging regressor with three weak learners. Notice the bootstrap samples in between the training set and each regressor.

regressors are trained sequentially in the following way: the first regressor trains and makes mistakes, then the second regressor is trained on the mistakes made by the first one; then, there can be a third regressor that is trained on the mistakes made by the second one and so on and so forth. Examples of boosting models include *AdaBoost*, *GradientBoosting*, *XGBoost*.

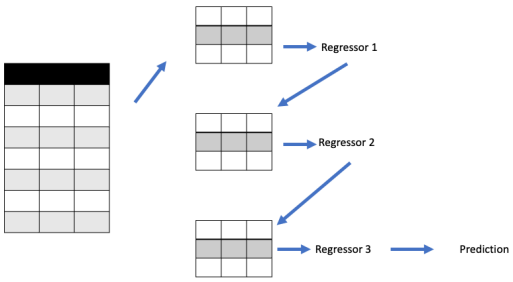

Figure 3.7: Boosting diagram mechanism.

3.2.3 Random Forest:

A random forest is an ensemble of decision trees and is very commonly used in both classification and regression. In scikit-learn it is implemented in the `RandomForestRegressor` class so that the programmer doesn't have to instantiate first a `BaggingRegressor` and the many weak learners. The problem with single decision trees is that they are very prone to overfitting, so averaging predictions of many of them can handle this problem. Hereafter is the algorithm of the random forest:

Algorithm 1 Random Forest Regressor algorithm for training and prediction. This algorithm trains multiple decision trees on different bootstrap samples of the training data and averages their predictions for regression tasks.

Require: Training dataset $\mathcal{D} = \{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$, Number of trees B, Number of features to sample at each split m
Ensure: Final prediction $\hat{y}$ for a new instance x

Initialize an empty set of trees $\{h_b\}_{b=1}^{B}$
for all $b = 1$ to B **do**
  Generate a bootstrap sample $\mathcal{D}_b$ by sampling N instances with replacement from $\mathcal{D}$
  Train a decision tree regressor $h_b(x)$ on $\mathcal{D}_b$
  for all nodes in h_b **do**
    if node is not a leaf **then**
      Select the best split among a random subset of m features
    end if
  end for
end for
return $\hat{y} = \frac{1}{B} \sum_{b=1}^{B} h_b(x)$

Terminology

- $\mathcal{D}$ is the original training dataset with N instances.
- $\mathcal{D}_b$ is the b-th bootstrap sample of the training dataset.
- $h_b(x)$ is the b-th decision tree regressor trained on $\mathcal{D}_b$.
- B is the total number of decision trees in the forest.
- M is the total number of features in the dataset.
- m is the number of randomly selected features to consider for the best split at each node.
- $\hat{y}$ is the final predicted value obtained by averaging the predictions of all B trees.

3.2.4 Random forest regressor parameters

As already mentioned, the standard random forest regressor implementation available in the scikit-learn is in the class `RandomForestRegressor`, as part of the package `sklearn.ensemble`. The available parameters are here listed [13]:

- n_estimators (int, default=100): The number of decision trees in the forest.
- criterion ("squared_error", "absolute_error", "poisson", default="squared_error"): The function to measure the quality of a split.
- max_depth (int, default=None): The maximum depth of the tree.
- min_samples_split (int or float, default=2): The minimum number of samples required to split an internal node.
- min_samples_leaf (int or float, default=1): The minimum number of samples required to be at a leaf node.
- min_weight_fraction_leaf (float, default=0.0): The minimum weighted fraction of the sum total of weights required to be at a leaf node.

- max_features ("auto", "sqrt", "log2" or int or float, default="auto"): The number of features to consider when looking for the best split.
- max_leaf_nodes (int, default=None): Grow a tree with max_leaf_nodes in best-first fashion.
- min_impurity_decrease (float, default=0.0): A node will be split if this split induces a decrease of the impurity greater than or equal to this value.
- bootstrap (bool, default=True): Whether bootstrap samples are used when building trees.
- oob_score (bool, default=False): Whether to use out-of-bag samples to estimate the generalization score.
- n_jobs (int, default=None): The number of jobs to run in parallel. None means 1.
- random_state (int, RandomState instance or None, default=None): Controls the randomness of the estimator.
- verbose (int, default=0): Controls the verbosity when fitting and predicting.
- warm_start (bool, default=False): When set to True, reuse the solution of the previous call to fit and add more estimators to the ensemble.
- ccp_alpha (non-negative float, default=0.0): Complexity parameter used for Minimal Cost-Complexity Pruning.
- max_samples (int or float, default=None): If bootstrap is True, the number of samples to draw from X to train each base estimator.

3.2.5 Model Selection

As already mentioned, the chosen machine learning model is the `Random Forest Regressor`, since it proved to be more powerful than linear models in many contexts and it is not a completely black box model, unlike the deep learning ones. Model selection refers to a phase where many machine learning algorithms are proposed and tuned to finally choose the best performing one. Conceptually it consists of an outer and an inner loop. In the outer loop each algorithm is tested (eg. SVRegressor, RandomForest Regressor, XGBoost-Regressor, etc..). In the inner loop there is the testing of each specified configuration of the hyperparameters of the algorithm (eg. number of estimators in the Random Forest). Unfortunately, due to time and resource availability reasons, for this project very limited parameter tuning has been performed. The only tuned hyperparameters are n_estimators, max_depth, max_features and max_leaf_nodes using a grid search technique.
What is meant by grid search? It is a way to find the best hyperparameters (i.e. parameters that are not learnt by the model but are set manually, the lower their number the better). This method consists of creating a list of values for each hyperparameter and then tries all the possible combinations of them.

This methodology is opposed to the randomised search, where for each hyperparameter a linear space can be specified, so that the search will sample a random point to try the configuration. The training was performed locally, on low-end laptops (nothing on cloud were allowed to be executed to minimize data-leak risks) and the training samples are almost 170 thousands (each Italian municipality, repeated each month for 2 years considered).

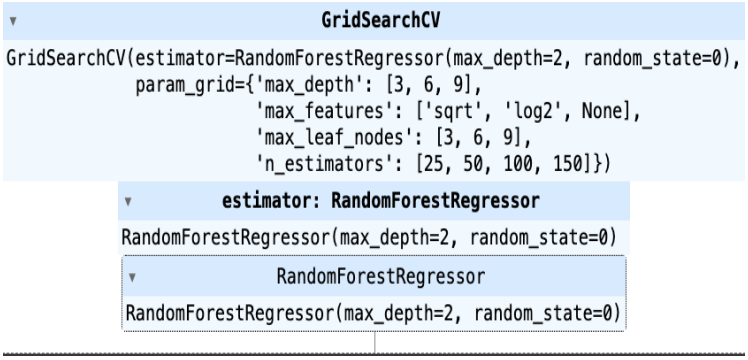

Figure 3.8: Grid search made with scikit-learn.

```
RandomForestRegressor(
    max_depth=3,
    max_features='sqrt',
    max_leaf_nodes=3,
    n_estimators=150,
    random_state=0
)
```

turned out to be the best model.

3.2.6 Feature importance

For the random forest, in scikit- learn, there are two main ways of computing feature importance that are impurity-based (that is biased towards features with a high number of different values) and feature permutation-based [14].
Permutation importance technique is well suited for nonlinear estimators. It works by shuffling the values of each feature and it computes the performance degradation of the model after the shuffling [15]. It depends on the selected scoring function.

The algorithm starts by considering the training set and the already fitted model. It then computes the root mean squared error (RMSE) on the fitted model. For each feature, there are k repetitions where the column is randomly shuffled (keeping the others intact), so a corrupted version of the training set is generated, and on that version, a new RMSE is computed. Finally, the importance on the j-th feature is computed as:

$$i_j = s - \frac{1}{k}\sum_{k=1}^{K} s_{kj} \tag{3.2}$$

where [15]:

- s: the performance on the original dataset.
- k: The number of permutations or the total number of iterations.
- s_{kj}: The performance metric for the k-th permutation of feature j.
- K: The total number of permutations.

Regarding our case, the following image depicts relevant features of our model: it is shown that the three most important features are 'SLOW NETWORK_complaint','eta_media', and 'perc_diploma'. The sum of all feature importances adds up to 1.

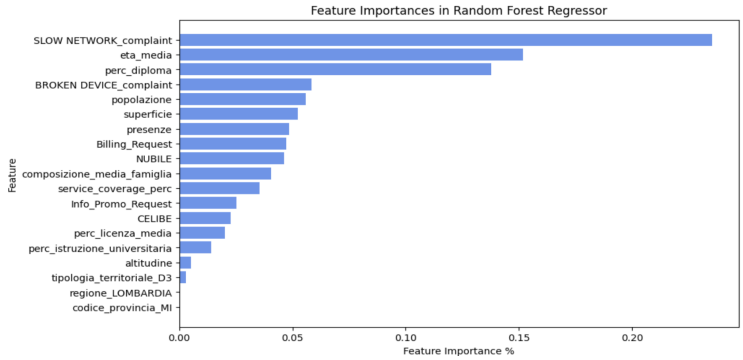

Figure 3.9: Importance of each feature in determining the output of the model.

3.3 Model evaluation

There are many performance metrics to evaluate regression models; some of the most important are R^2, root mean squared error (RMSE), and mean absolute error (MAE). In the following section, the RMSE is treated in detail, since it is the evaluation metric used in this project.

3.3.1 Root Mean Squared Error

This evaluation metric is actually one of the most used in sectors such as finance (e.g. stock market prediction), energy (e.g. forecast of energy demand), and climate science (e.g. for models able to predict precipitation level). One characteristic that makes this metric so popular is that it can heavily penalise high-magnitude errors in optimization problems when large deviations from actual values are not desired, as opposed for example to mean absolute error.

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{3.3}$$

- n: total number of observations or data points.
- y_i: actual value of the i-th observation.
- $\hat{y}_i$: predicted value of the i-th observation.

One could think that the lower the RMSE is the better is the model. However if a model can produce zero RMSE in the training set, that does not necessarily mean that it is perfect, rather, it is more likely that the model is overfitting that set (in such a case regularization techniques could be considered as a remedy). In this case, it has been observed that the model had high RMSE in both the training and test phases, being a symptom of underfitting. In the reference dataset, the run rate ranges between 0 and 1000, and the RMSE is 120, that not acceptable for the proposed task. The units of the RMSE are the same as the target variable, in this case, the potential customers.

Chapter 4
Explainability

The client strongly requested a tool to know how the model makes predictions, i.e. how to get explainable outcomes. Why explainability matters? Explainability is aimed at increasing trust in AI systems. Suppose to have a situation where a model is already deployed in an operational environment but its predictions are not accurate enough. In such a case one needs a debugging tool to understand how the model is predicting values, for example by understanding which features contributed most to the unwanted outcomes. Technically speaking, it is important to remark that the feature importance implemented by the random forest algorithm is useful for global predictions in general, but what about a specific prediction? It doesn't even tell what is the feature contribution to increase or decrease the predicted value. In this chapter, we discuss the Python package called SHAP (SHapley Additive exPlanations)[1]. It provides functionalities such as local and global explainability, feature interactions, and many more.

4.1 Mathematical background

One of the most interesting information data that SHAP can provide is feature contributions, also called SHAP or Shapley values.
The following formula, inspired by game theory, shows how SHAP computes internally the contribution of a feature i[6][7]:

$$\phi_i = \sum_{S \subseteq F \setminus \{i\}} \frac{|S|!\,(|F| - |S| - 1)!}{|F|!} \left[f_{S\cup\{i\}}(x_{S\cup\{i\}}) - f_S(x_S) \right] \tag{4.1}$$

where:

- ϕ_i: it is the contribution of the i-th feature of a sample in determining the increase or decrease of the model's output with respect to the average prediction computed on all the dataset;
- F: The set of all features;
- $|F|$: The cardinality (i.e. number of elements) of set F, that is the total number of features and if put together with "!" symbol indicates its factorial;
- $\sum_{S \subseteq N \setminus \{i\}}$: summation over the permutations of total features in every subset of features S without the i-th feature (remember that in permutations, the position of elements matter);
- S: A subset of F that does not include the i-th feature;

- $|S|!$: factorial of the cardinality of set S, being the number of ways to arrange the subset S of features;
- $f_S(x_S)$: prediction values of a model that considers only the features present in the subset S;
- $f_{S\cup\{i\}}(x_{S\cup\{i\}})$: prediction values of a model that considers features in subset S including the i-th feature;
- x_S:represents the values of the input features in the set S.

semantically:

- $\frac{|S|!\,(|F|-|S|-1)!}{|F|!}$: this term acts as a weight. The numerator is number of subsets that S can form. The denominator is the number of ways that the total features can be arranged. This weight gives the probability that i-th feature makes a contribution a dataset with $|S|$ features, given that the complete dataset has $|F|$ features.
- $\left[f_{S\cup\{i\}}(x_{S\cup\{i\}}) - f_S(x_S)\right]$: Marginal contribution of the i-th feature [9].

The SHAP package implements the `TreeExplainer` class, suited for ensemble models such as random forest. Explanations can be visualised with plots with their own properties that will be discussed in the next sections.

4.2 Local Interpretability

Local interpretability focuses on understanding individual predictions. It explains why a model made a specific prediction for a single instance (i.e. a single row of data). To visualize the explanation of a single row, SHAP provides specific plots such as waterfall and force plots. Hereafter their description.

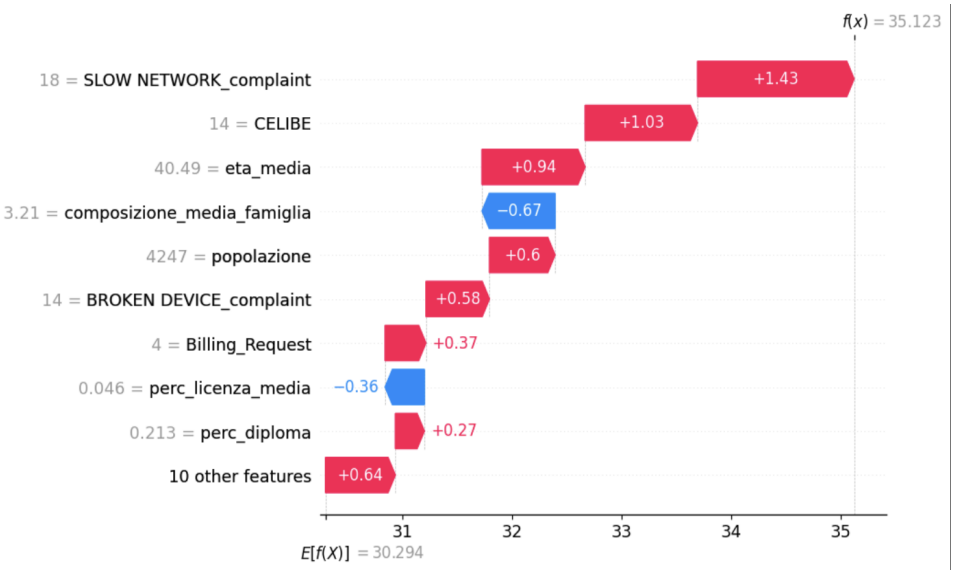

Figure 4.1: Shapley values in a waterfall plot.

The above figure shows a waterfall plot aimed at explaining a single prediction. In this case the random forest regressor predicted a value of 35.123, that is depicted as $f(x)$. The other important value is $E[f(x)]$, i.e. the average prediction, of the trained model, made on the test set. On the left side there are the feature values of the sample being considered. This plot shows clearly what is the contribution of each feature, for example the value 18 of the feature "SLOW NETWORK_complaint" has contributed to and increase of 1.43 on the run rate. Likewise the feature value 3.21 of “composizione_media_famiglia“ contributed to a decrease of 0.67 on the run rate.

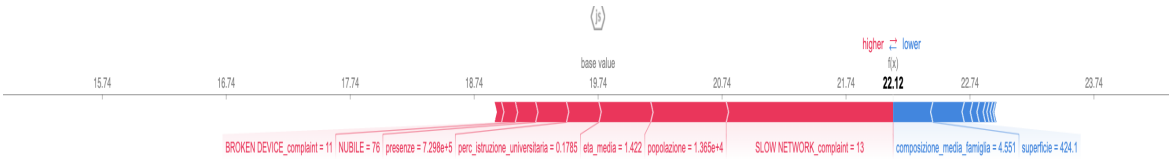

Figure 4.2: Shapley values in a force plot.

Another way of visualising shapley values is by using a force plot (shown in Figure 4.2). It can be thought as a compact version of the waterfall plot. However this visualization could be less immediate to interpret. In this case there is an average prediction (displayed as base value) of 19.74 and the result for the current sample is 22.12. It is also shown the feature contributions colored in red (if positive) and in blue (if negative).

4.3 Global Interpretability

It is important to remark that shap values are in the same units as the target variable, thus providing a more accurate explanation compared to random forest feature importance. SHAP provides a summary plot functionality that visualizes feature impacts in two key variants: the feature importance plot and the beeswarm plot.

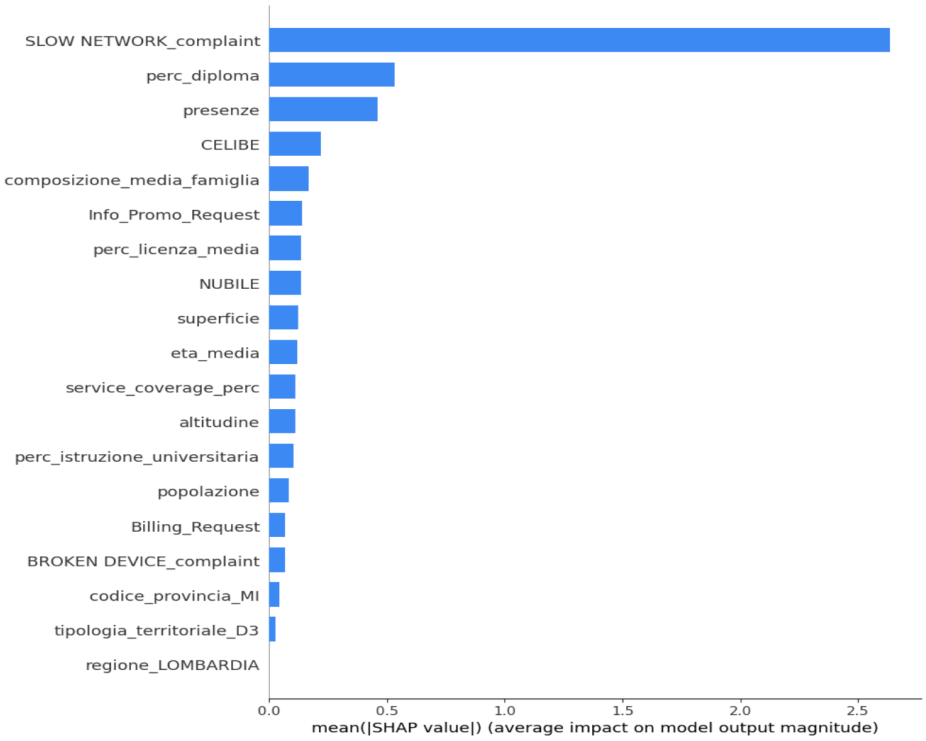

Figure 4.3: Shapley values of feature importance in a bar plot.

This feature importance graph is rather simple to interpret: the larger the bar, the more important the feature. It is basically the average of the importance in modulus of each feature computed on each instance. By looking at this bar plot one can state that, on average, the feature "SLOW NETWORK_complaint" has an impact of more that 2.5 users on the run rate. Since SHAP values coming from the `TreeExplainer` are model dependent (they reflect how the current model assigns importances), global interpretability results were delivered to the client to justify how the model makes predictions. Below the mathematical formula used to compute each bar I_J [8]:

$$I_j = \frac{1}{N} \sum_{i=1}^{N} |\phi_i^{(j)}| \tag{4.2}$$

where:

- N is the total number of instances.
- $\phi_i^{(j)}$ is the SHAP value for feature j in instance i.

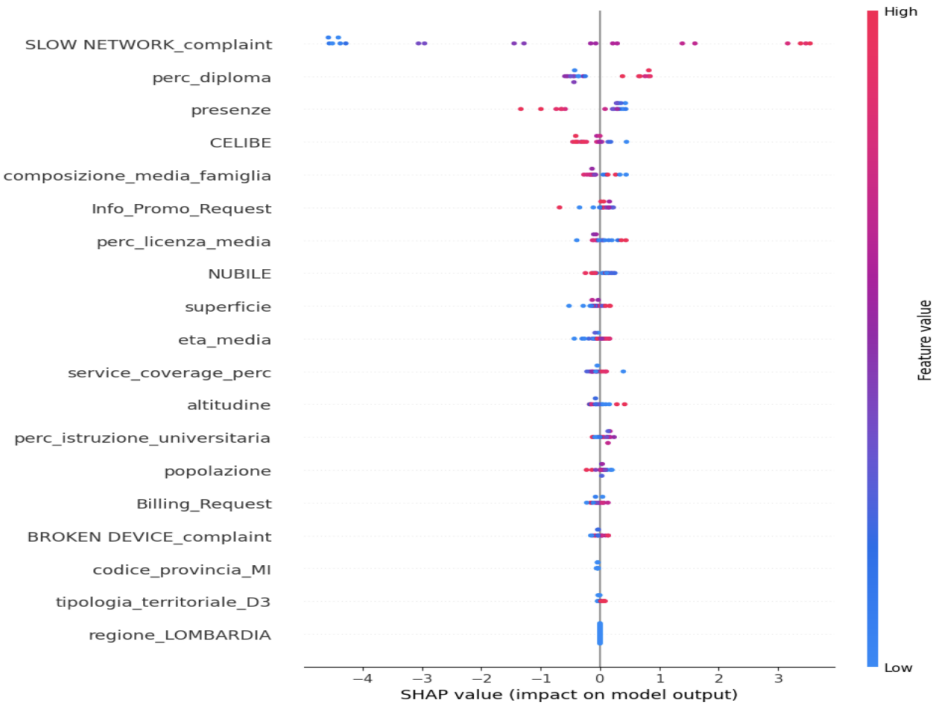

Figure 4.4: Shapley values in a beeswarm plot.

The above graph is a beeswarm plot that is aimed to show the impact of features for the overall model. It is remaked that the explanation is performed on the test set. On the left side there are all the features listed. Each point of the plot represents the shapley value of a specific feature for a specific instance and its color represents the magnitude of the feature value. If there are many datapoints with the same SHAP values for a certain feature those points will undergo an offset so to avoid overlaps in the plot.
It is thus clear that this kind of global explainability is intrinsecally based on many local explainabilities.
In this case one can state that high values of "SLOW NETWOK_complaint" have a great importance on the outcome because there are many red dots with SHAP value > 3 for the above mentioned feature.

Chapter 5
Conclusions

Hereafter a discussion about project goals, methods, problems occurred, and further possible developments that this project could be subject to.

5.1 Review of Project Goals

The goal of this project was to increase sales of a telecommunication service by focusing on remunerative municipalities. To achieve this, a machine learning algorithm has been designed and a target variable was established, the run rate. The following steps were accomplished: collection of public and company-owned data, data cleaning and feature engineering, correlation analysis, training and evaluation of a random forest regression algorithm, and explanation of the outputs of the model. Unfortunately, the model was deemed not precise enough due to its large root mean squared error, so in the end a descriptive statistical analysis about the interesting correlation between network complaints and run rate was finally delivered to the client.

5.2 Context

This project was accomplished in a consultancy environment, and I worked as a data scientist intern. Unfortunately, because of my intern role, the client decided to not grant me any license to access either their hardware or their data. Actually, the data presented in this thesis are mocked up. The only hardware resource I was allowed to use was the consultant company laptop and I was kept up to date with the project's progress only through videoconference software, where colleagues had the chance to screenshare their content with me. Under no circumstances was any customer data permitted to be manipulated on my given laptop, where no additional software, apart from the preinstalled ones (videoconference and virtual private network [VPN] client), could be installed and very limited cloud resources were allowed. Since treated data are confidential it was not possible to share them among laptops (also because the USB ports of all computers prevented any memory-endowed device to be read). Moreover, the majority of the thesis content had to be written during the internship period. After that period, I would no longer be authorized to ask any project-related questions to prevent data leaks. Another major problem that I encountered was related to the model results. Such an issue was treated with the client, and together with the team, it was agreed to change the approach; from a regression problem, it shifted to a classification problem using logistic regression, with the new target being the probability that a private customer would buy the service. This change of paradigm happened nearly the end of my internship when already more than half of the thesis was written based on the original approach. Another technical problem was that the

client did not specify any target variable, so the consultants had to engineer a custom one, that is the run rate. Very poor results were achieved in both approaches (regression and classification) because, according to the consultants, the features were scarcely correlated with the target. The consultants specified the need for more internal data, for example, end-user traffic data, that could be more correlated with the target, but actually the client did not demonstrate proactiveness in this sense. Regarding public data from ISTAT, some features had to be filled with 2011 data values [3], the most updated. The demographic scenario in Italy changed considerably from back then (e.g. merged municipalities, suppressed provinces, migrations etc...). It is important to remark once again that data quality remains key for a good model to make satisfactory predictions. Despite all the above limitations, consultants were still able to deliver at least a descriptive analysis that could be used as a basis for further developments.

5.3 Future Works

For this project future works could include the exploration of other features, perhaps with the analyses of internet traffic data, and exploit more powerful machine learning models like XGBoost. In case of availability of dedicated ML training hardware, a more intensive hyperparameter tuning could have been performed. If the model predictions are satisfactory enough, then it could have been implemented as a web service (for example using the Flask Python web framework) or as a Python module in a production environment, so the client can use it whenever needed. The model would then need to be monitored, usually by putting a human in the loop, to evaluate predictions over time, so in case of performance degradation over time it can be retrained on new data.

References

[1] SHAP developers. https://shap.readthedocs.io/en/latest/. Accessed: 2024-08-24. (Cited on page 23)

[2] Aurelien Geron. *Hands-On Machine Learning with Scikit-Learn and TensorFlow*. O'Reilly, 2021. (Cited on page 17)

[3] ISTAT. http://dati-censimentopopolazione.istat.it/Index.aspx?DataSetCode=DICA_GRADOISTR1. Accessed: 2024-08-24. (Cited on page 29)

[4] ISTAT. Censimenti permanenti popolazione e abitazioni. https://www.istat.it/it/files//2018/09/FAQ-censimento-permanente-popolazione-Istat-2021.pdf. Accessed: 2024-08-24. (Cited on page 6)

[5] Will Kenton. Run Rate: Definition, How It Works, and Risks With Using It run rate. https://www.investopedia.com/terms/r/runrate.asp. Accessed: 2024-08-24. (Cited on page 5)

[6] Scott Lundberg and Su-In Lee. A unified approach to interpreting model predictions, 2017. (Cited on page 23)

[7] Christoph Molnar. https://christophm.github.io/interpretable-ml-book/shapley.html#the-shapley-value. Accessed: 2024-08-24. (Cited on page 23)

[8] Christoph Molnar. https://christophm.github.io/interpretable-ml-book/shap.html#shap-feature-importance. Accessed: 2024-08-24. (Cited on page 26)

[9] Conor O'Sullivan. https://towardsdatascience.com/from-shapley-to-shap-understanding-the-math-e7155414213b. Accessed: 2024-08-24. (Cited on page 24)

[10] GARANTE PER LA PROTEZIONE DEI DATI PERSONALI. Sicurezza dei dati di traffico telefonico e telematico - 17 gennaio 2008 [1482111]. https://www.garanteprivacy.it/home/docweb/-/docweb-display/docweb/1482111. Accessed: 2024-08-24. (Cited on page 6)

[11] scikit-learn developers. https://scikit-learn.org/stable/auto_examples/tree/plot_unveil_tree_structure.html#sphx-glr-auto-examples-tree-plot-unveil-tree-structure-py. Accessed: 2024-08-24. (Cited on page 15)

[12] scikit-learn developers. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.VotingRegressor.html. Accessed: 2024-08-24. (Cited on page 17)

[13] scikit-learn developers. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html. Accessed: 2024-08-24. (Cited on page 19)

[14] scikit-learn developers. `https://scikit-learn.org/stable/auto_examples/ensemble/plot_forest_importances.html`. Accessed: 2024-08-24. (Cited on page 21)

[15] scikit-learn developers. `https://scikit-learn.org/stable/modules/permutation_importance.html#permutation-importance`. Accessed: 2024-08-24. (Cited on page 21)