Household income prediction with ridge regression

Teresa Tanzi, mat. 925574
teresa.tanzi@studenti.unimi.it

F943T - Algorithms for Massive Datasets
F943X - Statistical Methods for Machine Learning
Università degli Studi di Milano

October 2, 2020

Abstract

We report an implementation from scratch of the ridge regression algorithm with square loss. Our aim is to predict the income of American households in 2013. We used the 2013 American Survey dataset. Due to the huge dimension of the dataset, we exploited the distributed framework of Apache Spark and we implemented the ridge regression with gradient descent. The grid search and the holdout validation have been performed in order to find the best hyperparameters combination. We find that the corresponding model has RMSE$_{test}$ = 73027 $.

I declare that this material, which I now submit for assessment, is entirely my own work and has not been taken from the work of others, save and to the extent that such work has been cited and acknowledged within the text of my work. I understand that plagiarism, collusion, and copying are grave and serious offences in the university and accept the penalties that would be imposed should I engage in plagiarism, collusion or copying. This assignment, or any part of it, has not been previously submitted by me or any other person for assessment on this or any other course of study.

1. Introduction

Machine learning is widely used to deal with different kind of problems in different disciplines. In particular, supervised learning allows us to investigate in an automatic way a relationship between some input features and the corresponding target output, in order to predict the target for new data. If the output values we aim to predict are continuous numbers, we are dealing with a regression problem. One of the simplest regression algorithm is the linear regression, which assumes a linear relationship between the quantities. The use of ridge regression allows us to overcame the intrinsic risk of overfitting. In this document we present the design, the implementation and the evaluation of the ridge regression algorithm, in order to learn a linear relationship between some features relative to the lifestyle of American households and their income. We use the 2013 American Community Survey dataset provided by Kaggle [10], which contains a huge amount of data. It is managed exploiting the distributed framework Apache Spark for large scale data processing [7]. The minimization of the empirical risk in the training of the ridge regression model is performed through the gradient descent (GD) algorithm, executed computing the gradient in a distributed way, via Map-Reduce [3].

This document is organized as as follows. In Section 2 we present the dataset, while the preprocessing operations are discussed in Section 3. In Section 4 we illustrate the ridge regression algorithm and report how it scales up with data size. The experiment is described in Section 5. Finally, the results are discussed in Section 6.

2. Dataset

The 2013 American Community Survey dataset [10] contains information about 3.5 million American households in 2013, specifically on who they are and how they live. The dataset is composed by two different surveys:

- Housing: containing information relative to the housing units, such as the year when the home was built, the presence of different kind of facilities, the language spoken by the household, etc. …, for a total of 231 columns and 1.48 M observations.
- Population: containing information relative to the people, such as age, education, employment, etc. …, for a total of 283 columns and 3.13 M observations.

The total dimension of the dataset is approximately 4 GB, making it hard to manage in its entirety by the main memory of a standard machine. Detailed information on the specific files are reported in Table 1. It is noteworthy that each survey content is resumed in two files.

File	Type	# Observations	# Columns	Size
`ss13husa.csv`	Housing	756 k	231	587.47 MB
`ss13husb.csv`	Housing	720 k	231	559.43 MB
`ss13pusa.csv`	Population	1.61 M	283	1.42 GB
`ss13pusb.csv`	Population	1.52 M	283	1.33 GB

Table 1: Information about the dataset files.

The task is to find a relationship connecting the household income (column HINCP in the housing subset) with the remaining features (other columns of the same subset). This relationship will allow us to make predictions of household income on the basis of other features. In this work, for sake of simplicity, only the housing subset has been considered, leaving to future woks the possibility to join the two subsets on the feature SERIALNO (serial number). In Table 2 the first ten columns of the used dataset are reported as example.

RT	SERIALNO	DIVISION	PUMA	REGION	ST	ADJHSG	ADJINC	WGTP	NP
H	84	6	02600	3	01	1000000	1007549	00000	01
H	154	6	02500	3	01	1000000	1007549	00051	04
H	156	6	01700	3	01	1000000	1007549	00449	01
H	160	6	02200	3	01	1000000	1007549	00016	03
H	231	6	02400	3	01	1000000	1007549	00052	01

Table 2: First 10 columns (over 231 total columns) of household dataset. The full table with detailed information is available in [10].

3. Preprocessing and data exploration

The two housing files are loaded in the project as Spark DataFrames, which are distributed collections of data organized into named columns. They are then merged by the union operation. Spark provides a variety of functions and methods useful to perform data preprocessing on huge DataFrames. The preprocessing phase includes the following steps.

Subdivision of the dataset. The dataset is divided in training set, validation set and test set with ratio 6:2:2, since we will evaluate the performance of the generated models with the holdout method. This step avoids data leakage [2], which raises when information about test or validation set is revealed to the model, leading to overfitting.

Drop constant categorical features. The dataset is composed by numerical features, except for the categorical feature RT (record type), which assumes the constant value ‘H’ for all the data points in the housing files. As this feature is not informative it has been dropped from the dataset.

Drop other income features. Since the purpose of this project is to predict the HINCP (household income) values, FINCP (family income) has been removed, as required by the specifications of the project.

Drop data points with null label. A significant problem is that our dataset contains a lot of missing values. First, we have removed all the data points presenting a missing value in HINCP feature, the label to be predicted, because those would not be helpful for training and for testing the predictor.

Drop features with more than 60% of null values. Each feature is then analyzed in order to understand the fraction of missing values. As shown in Figure 1, a small number of features are null for nearly the totality of data points in the training set. For these, replacing the null values with a proxy value may introduce a distortion in the data. Thus, all the features with more than the 60% of missing values are removed, following [9]. The value of the threshold may be considered as an hyperparameter to be properly tuned. We removed 10 columns out of 229, corresponding to the 4.37%.

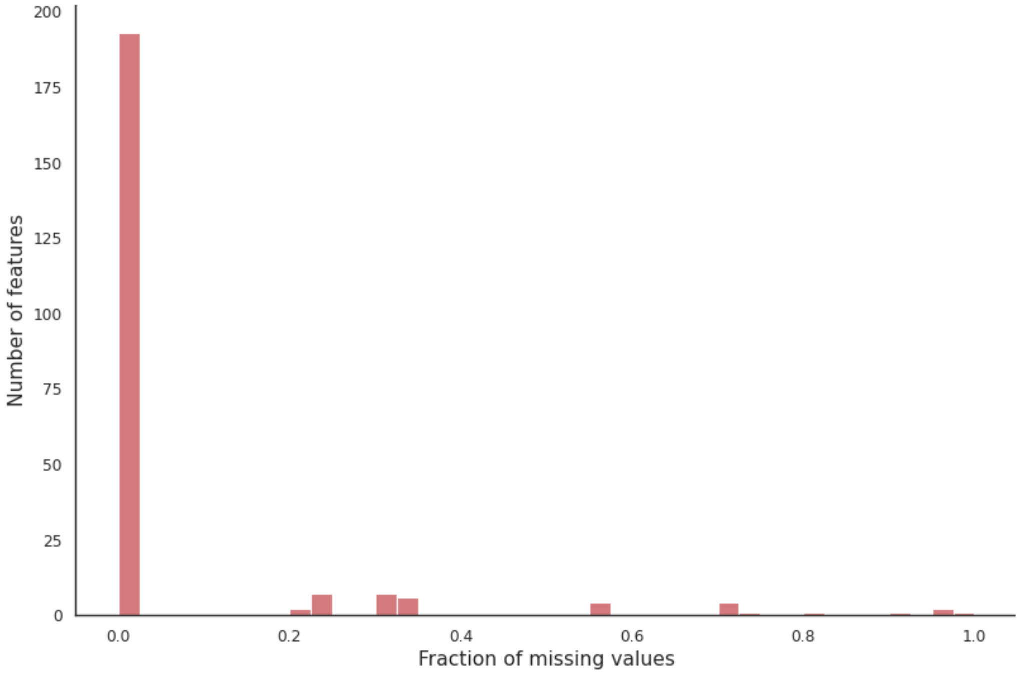

Figure 1: Distribution of missing values in features.

Impute missing values. The remaining null values are managed by an imputation technique, i.e. they are replaced with a proxy value according to a certain rule. We replace all the null values with the average of non null values of the specific feature. This could also be done using the median or the mode instead. This kind of proxy has the effect of reducing the variance in the feature. There exists more complex imputation techniques, based on probability and noise introduction [5].

Find features with high correlation. Different machine learning algorithms require different preprocessing on the data. Linear regression suffers from multicollinearity, which occurs when independent variables in a regression model are correlated [11]. This problem is reduced in ridge regression, as explained in [4], making unnecessary to find and drop features with high correlation. However, we analyze the correlation between features for data exploration purposes. In Figure 2 the correlation matrix is reported.

An example of a couple of features with high positive correlation coefficient is:

- FES (family type and employment status): e.g. married-couple family, husband and wife in labor force.

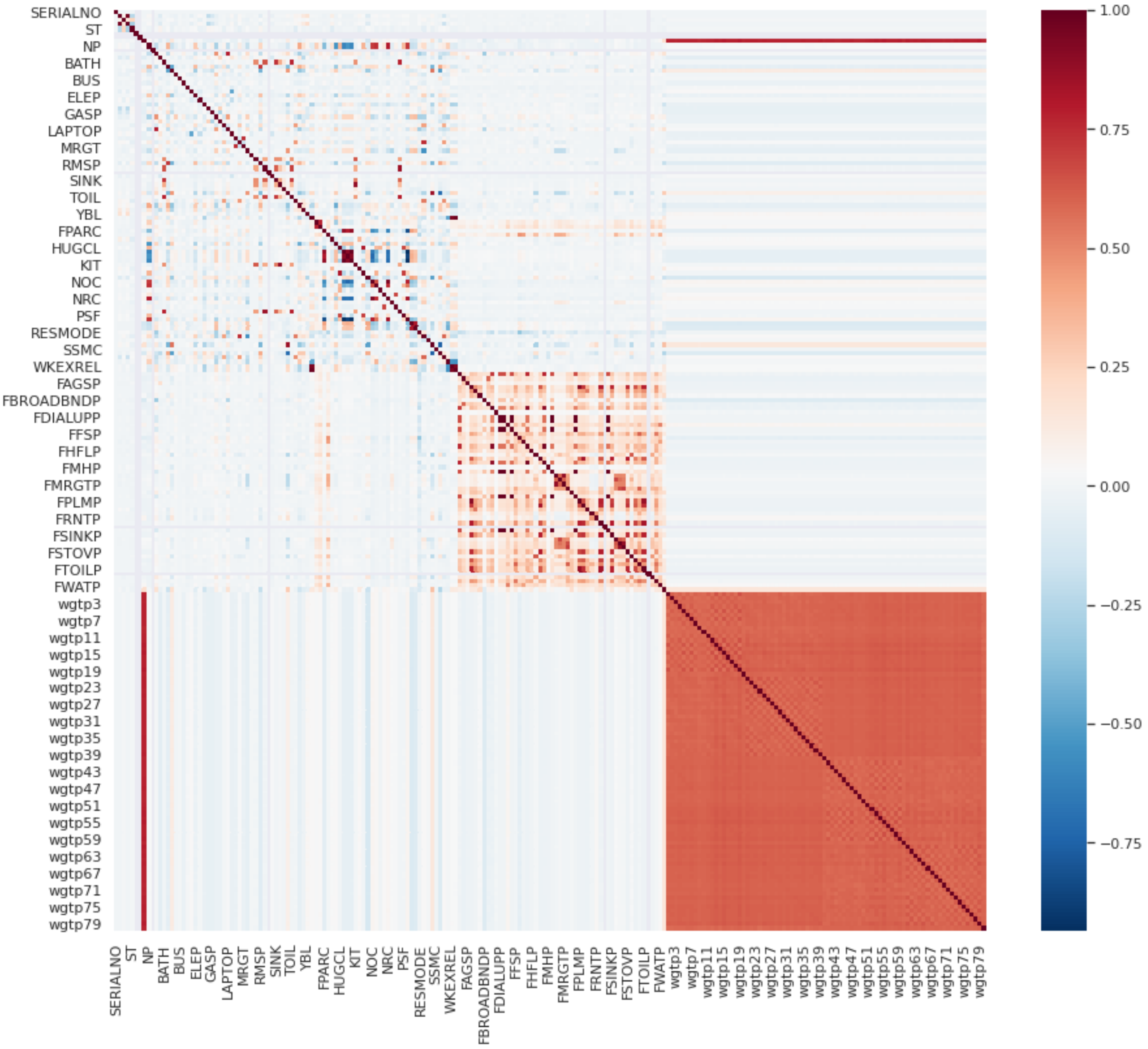

Figure 2: Correlation matrix between features. It is built by computing the Pearson correlation coefficient for each pair of features. Red to blue cells represent cases from high positive to high negative correlation.

- WORKSTAT (work status of householder or spouse in family households): e.g. husband and wife both in labor force, both employed or in armed forces.

While an example of a couple of features with high negative correlation coefficient is:

- HUPAC (presence and age of children): e.g. with children 6 to 17 years only.
- R18 (presence of persons under 18 years in household): e.g. one or more persons under 18 in household.

One way to deal with correlation is to apply PCA (Principal Component Analysis) algorithm, see below. Together with ridge regression, PCA solves the problem.

Find outliers. Ridge regression, as other linear methods, is sensitive to outliers. It is possible to find, remove or correct them in several ways [8]. We use the IQR (interquartile range) score. We denote with q_1 and q_3 the first and the third quartiles, respectively (corresponding to quantiles of order 0.25 and 0.75). The interquartile range is computed as $iqr = q_3 - q_1$. We define outliers of a feature all the values x such that:

$$(x > q_3 + 1.5 \times iqr) \vee (x < q_1 - 1.5 \times iqr)$$

Due to the large number of features and outliers in each feature, a lot of potential outliers has been found. Dropping or modifying all these data points implies a significant distortion of the original data. Figure 3 shows how the training set size is reduced as long as we remove the outliers for each feature, yielding to a total data loss of 95%. Thus, the outliers treatment has not been performed. Another possible solution is to relax the threshold used to detect outliers, tuning properly the value of the threshold.

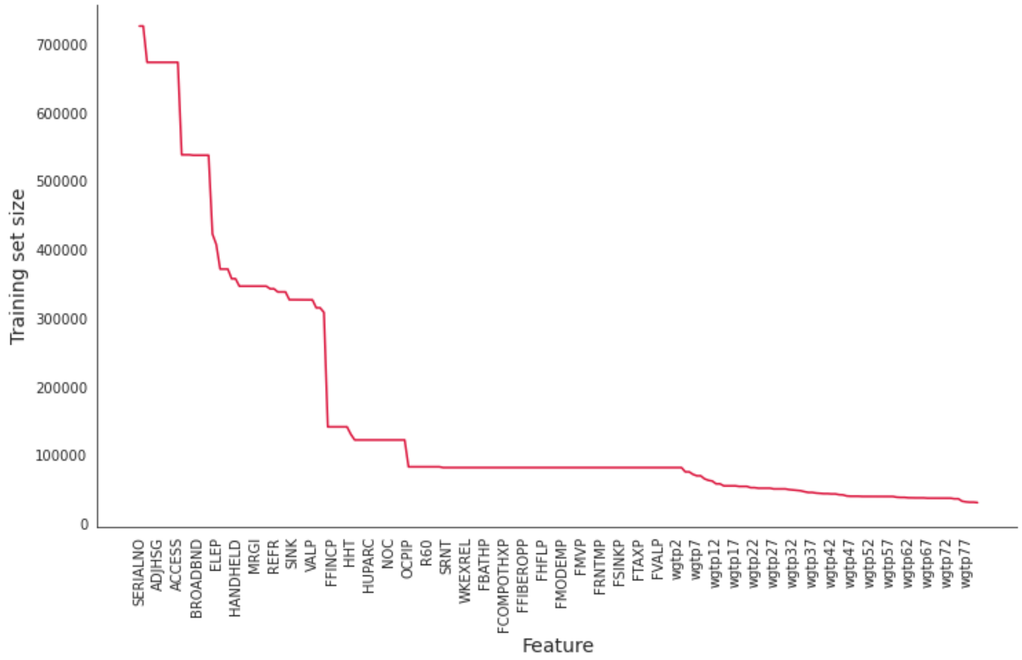

Figure 3: Decrease of training set size while outliers for each feature are removed.

Scale features. The dataset contains features with different ranges of values and unit of measure. Since we use algorithms exploiting the concept of variance and Euclidean distance, such as PCA and GD (see Subsection 4.3) we need to standardize all the features to be compared [1]. Among the different scaling techniques, in this project we apply the standardization, which transforms data to having mean = 0 and standard deviation = 1. The standard score z of a value x is computed as:

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

where μ and σ are the the original mean standard deviation of the feature.

Apply PCA. Due to the significant number of features, the application of some technique of dimensionality reduction is considered. We have applied PCA in order to map data points on a new feature space of smaller dimension. The new features are called principal components, sorted by variance. The first principal component contains as much of the variance in the data as possible, and the following ones account for remaining variance as much as they possibly can. The dimensionality reduction is performed by selecting the first n principal components and dropping the remaining ones. The number n of desired PCA features has to be chosen beforehand and should be properly tuned. We choose $n = 30$, reducing our data to 13.7% of the starting size. As shown in Figure 4, 30 PCA features correspond to 63% of the original variance.

For sake of visualization, 1000 data points of the training set have been plotted in Figure 5 as a function of combinations of the first four PCA features. It is apparent that the distribution of the labels is in some way related to the value of those features, especially the 3rd and the 4th.

Add feature with constant value 1. A new feature with constant value 1 is added to each observation in order to include a constant term in the linear relation among the features, see Subsection 4.1.

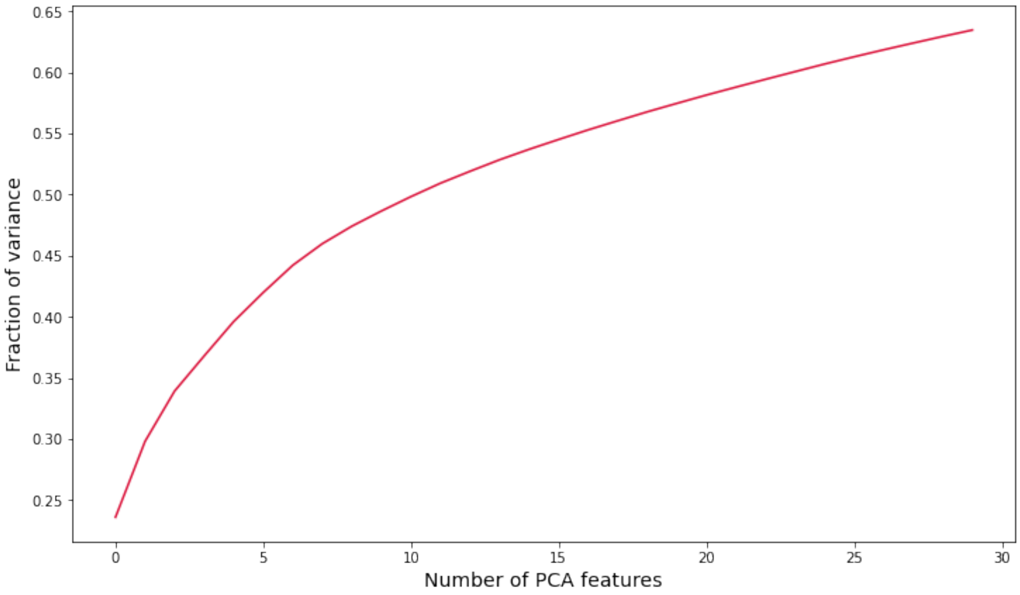

Figure 4: Fraction of global variance retained by the first 0 to 30 PCA features.

Cast into LabeledPoint Finally, the three DataFrames are transformed into RDDs (Resilient Distributed Dataset) of LabeledPoints, ready to be used to train the model. A Spark RDD is an abstraction of a set of data representing a partitioned collection of elements that can be operated on in parallel. In our case, the elements are objects of the class LabeledPoint and represent data points defined by their feature vector and their label.

4. Algorithm

Ridge regression [6] is a regularization method applied to linear regression in order to mitigate its instability problem and to prevent overfitting. Overfitting occurs when the proposed predictor learns too much on the training data and cannot generalize from unobserved data.

4.1. Linear regression

Let $\{(\mathbf{x}^{(1)}, y^{(1)}), \dots, (\mathbf{x}^{(n)}, y^{(n)})\}$ be a training set, where $(\mathbf{x}^{(j)}, y^{(j)}) \in \mathbb{R}^d \times \mathbb{R}$ for $j = 1, \dots, n$. Under the assumption of a linear relationship between the features $\mathbf{x}^{(j)}$ of the data point and its label $y^{(j)}$, linear regression search for such relationship. This allows us to infer the predicted label $\hat{y}^{(j)}$:

$$\hat{y}^{(j)} = \mathbf{w} \cdot \mathbf{x}^{(j)}$$

where $\mathbf{w} \in \mathbb{R}^d$ is the vector of the coefficients[1] that minimizes the empirical risk with respect to the square loss function $\ell(\mathbf{w}, (\mathbf{x}^{(j)}, y^{(j)})) = (\mathbf{w} \cdot \mathbf{x}^{(j)} - y^{(j)})^2$. Let X be the design matrix $n \times d$ that has a row for each observation $\mathbf{x}^{(j)}$ and let $\mathbf{y}$ be the vector containing all the n corresponding to true labels. Thus, the optimal vector $\mathbf{w}^*$ is chosen as:

$$\mathbf{w}^* = \underset{\mathbf{w} \in \mathbb{R}^d}{\operatorname{argmin}} \sum_{j=1}^{n} (\mathbf{w} \cdot \mathbf{x}^{(j)} - y^{(j)})^2 = \underset{\mathbf{w} \in \mathbb{R}^d}{\operatorname{argmin}} ||X \cdot \mathbf{w} - \mathbf{y}||^2$$

[1] Without loss of generality, we can ignore the constant term w_0 of the linear combination. We can embed w_0 in the weight vector $\mathbf{w}$ by adding a constant term 1 to the feature vector $\mathbf{x}$, thus: $\mathbf{w} \cdot \mathbf{x} + w_0 = (w_0, \mathbf{w}) \cdot (1, \mathbf{x}) = w_0 + w_1 x_1 + w_d x_d$

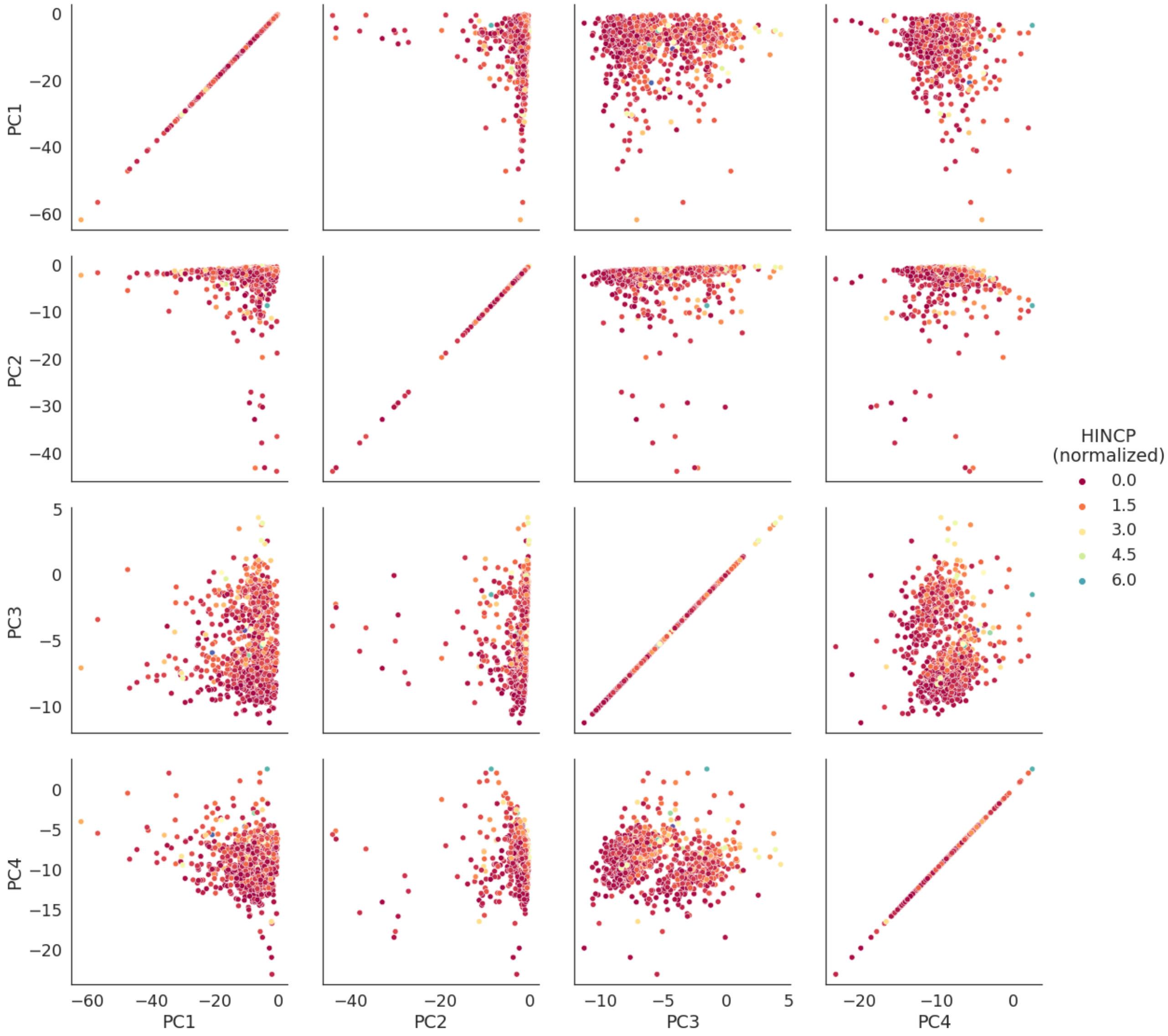

Figure 5: Training data distribution over combinations of the first 4 PCA features. The points are colored according to the value of the label.

4.2. Ridge regression

As previously said, linear regression is unstable: the final solution may significantly vary when dataset is perturbed. In order to mitigate the instability, we add a regularization term $\lambda||\mathbf{w}||^2$ to the function to be optimized:

$$\mathbf{w}^* = \underset{\mathbf{w}\in\mathbb{R}^d}{\operatorname{argmin}} \left(||X \cdot \mathbf{w} - \mathbf{y}||^2 + \lambda||\mathbf{w}||^2\right)$$

where $\lambda > 0$ is the regularization parameter. It has to be properly tuned according to the bias we want to add to the learning process, in order to keep small the components of the weight vector and control the instability.

4.3. Gradient descent

Since the objective function of ridge regression is convex, we can find its optimum analytically, setting its gradient equal to zero. This process has a time complexity $O(d^3 + nd^2)$ and a space complexity $O(d^2 + nd)$, making it prohibitive on large scale data. To solve the problem, we can exploit GD, which is an iterative algorithm searching for the minimum of the function. It takes steps proportional to the negative of the gradient of the object function at the current point with constant of proportionality depending on a value ξ. We call ξ learning rate.

Since GD is an iterative process, there are multiple possibilities to set the exit criterion. In our case, we define beforehand the number T of iterations to be performed. The training procedure of ridge regression with GD is shown in Algorithm 1. In the implementation of this algorithm, we choose $\mathbf{w}_1 = \mathbf{0}$ for line 1.

Algorithm 1: Ridge regression with gradient descent.

```
Input: training set $\{(\mathbf{x}^{(1)}, y^{(1)}), \dots, (\mathbf{x}^{(n)}, y^{(n)})\}$
       regularization factor $\lambda$
       number of iterations $T$
       learning rate $\xi$
Output: weight vector $\mathbf{w}_{T+1}$
1 choose $\mathbf{w}_1 \in \mathbb{R}^d$
2 for $t = 1, \dots, T$ do
3     $\xi_t \leftarrow \frac{\xi}{n\sqrt{t}}$
4     $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \xi_t \left[\sum_{j=1}^{n} (\mathbf{w}_t \cdot \mathbf{x}^{(j)} - y^{(j)})\mathbf{x}^{(j)} + \lambda \mathbf{w}_t)\right]$
5 end
6 return $\mathbf{w}_{T+1}$
```

The sum in line 4 of Algorithm 1 is too big for a single node, but it can be computed exploiting the programming model Map-Reduce [3], supported by the Spark distributed framework. Each node compute a single element of the sum, with a time complexity $O(d)$ and a space complexity $O(d)$. The final sum is computed through a reduce operation.

5. Experiment

The execution is composed by different steps detailed in the following paragraphs.

Dataset loading and preprocessing. The dataset is automatically loaded in the project as a Spark DataFrame. As described in Section 3, the preprocessing steps are applied, obtaining three Spark RDDs of LabeledPoints, containing the test set, the training set and the validation set, respectively. Information about the three datasets is reported in Table 3. The splitting of the dataset is performed with the Spark function `randomSplit()` taking as input a seed. We chose as seed the number 12345. If the seed is changed, the number of data points in each subdivision slightly changes.

Dataset	# Data points	# Features
Training set	726531	30
Test set	242105	30
Validation set	242628	30

Table 3: Information about the datases after the preprocessing phase.

Hyperparameters tuning and model selection. As discussed in Section 4, ridge regression with GD needs the following hyperparameters to be chosen:

- Regularization parameter $\lambda \in \mathbb{R}_+$. It controls the amount of bias added to the model in order to make it more stable and avoid overfitting.
- Number of iterations $n \in \mathbb{N}_+$ performed by the GD procedure to find the minimum of the objective function.

- Learning rate $\xi \in \mathbb{R}_+$. It determines the step size while moving toward a minimum of a loss function.

To find the best combination of those hyperparameters, a grid search is performed with the following values:

- $\lambda = 1, 10^{-5}, 10^{-10}$
- $n = 500, 1000, 1500$
- $\xi = 0.001, 0.0001, 10^{-5}$

For each combination (λ, n, ξ) the corresponding model has been evaluated through the holdout method, computing the RMSE (root mean squared error) on the validation set. Note that the holdout estimate of error rate is likely to be misleading if we happen to get an unfortunate split. This limitation can be overcome with some other evaluation methods, like cross-validation, but at the expense of higher computational cost.

In Figure 6 we show how RMSE$_{val}$ vary among the different combinations of the hyperparameters. Due to the limited computing resources and the large amount of data, to obtain results in a reasonable amount of time the number n of iterations in GD is necessarily quite small. This makes models with high learning rate ξ preferable, since they move faster towards the minimum of the objective function, despite they lead to more inaccurate results. Indeed, among the tested n, see Figure 6, a larger value better approximates the minimum. Unexpectedly, the value of the regularization parameter λ seems to have no effect, although we observe a slightly lower RMSE$_{val}$ for lower λ, in digits less significant than those reported in Figure 6. Anyhow, a wider range for λ should be tested in order to better understand its contribute to the learning process.

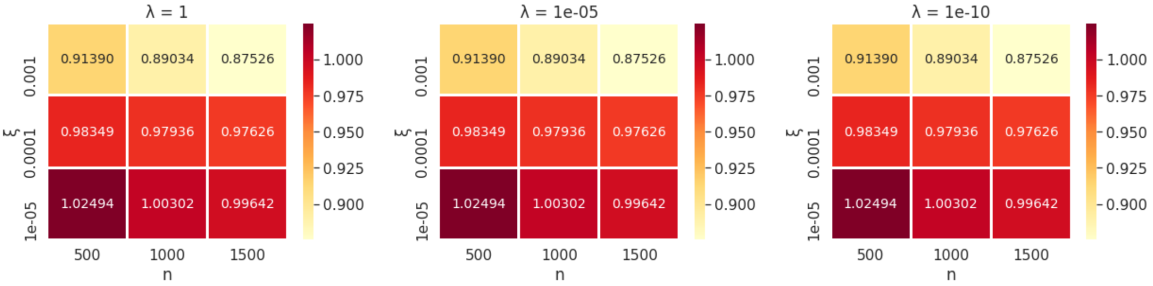

Figure 6: Validation RMSE for all combinations of hyperparameters. Lighter colors represent lower values of RMSE$_{val}$.

The model with lower RMSE$_{val}$, relative to hyperparameters $\lambda = 10^{-10}$, $n = 1500$ and $\xi = 0.001$, has been chosen as the best predictor. The corresponding RMSEs for the training set and the validation set are 0.88280 and 0.87526, respectively.

Model evaluation. We evaluate the best model on the test set, containing data not used neither for training nor feature selection. We obtain RMSE$_{test}$ $= 0.88071$, corresponding to an average error of 73027 \$. Figure 7 shows how the best model behaves on a random subsample of 1000 data points taken from the test set. The points concentrate around the bisector of the one-to-one plot, indicating the correct behaviour of the model. Points with largest spread represent outliers of the true labels, as we can see observing the partial distribution on the vertical axis.

6. Summary and Conclusions

We have built a regressor to predict the household income from the 2013 American Community Survey dataset. First of all we cleaned the data. Since the dimension of the dataset is quite large, the PCA algorithm has been used in order to reduce the amount of features involved. Then, we implemented ridge regression with square loss, exploiting GD and the distributed framework provided by Spark. The best hyperparameters combination are selected by a grid search on a small set of values. We validated each trained model via the holdout technique and choose the model with lower RMSE$_{val}$.

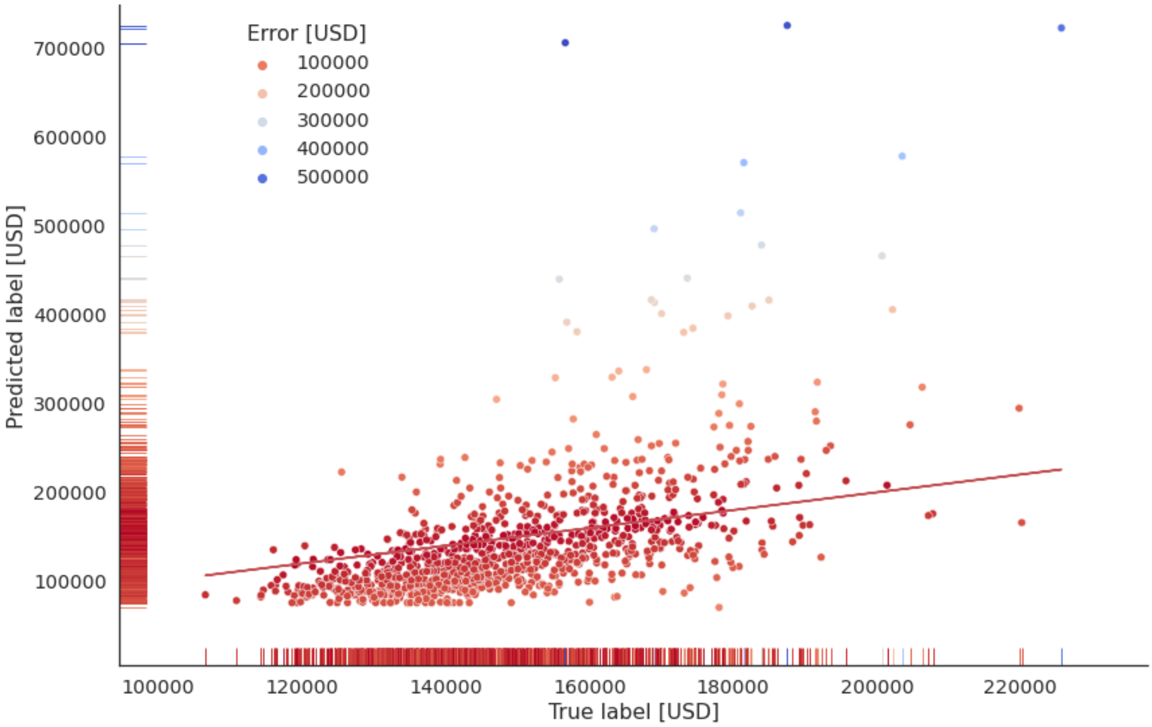

Figure 7: Comparison between the true labels and the predicted labels on a random sample of 1000 data points. Points with a more vibrant red color correspond to correctly predicted labels, while points with blue color correspond to very wrong predictions. Those points represent the outliers, as we can see observing the partial distribution of true labels on the y-axis.

We identified some critical issues: the huge amount of data required a long time to be analyzed on a standard machine. This prevented us to make a finer analysis on the dataset and on the hyperparameters. Despite of this our result yields a working predictor. We find a value of RMSE$_{test}$= 73027 \$, similar to RMSE$_{train}$ = 73190 \$ and RMSE$_{val}$ = 72565 \$. This could be a good sign of the absence of overfitting.

We propose to execute a future experiment using a more powerful hardware, also exploiting a cloud-based computing service, in order to add the following improvements and execute them in a reasonable time:

- Join together the two surveys composing the whole dataset (housing and population) in order to get more features that could help to build the predictor.
- Make a finer grid-search, since we tested a small set of values for each hyperparameter because of the computational limitations.
- Execute the experiment with different values of the threshold percentage of null values for a feature to be dropped and the number of PCA features to keep.
- Execute the experiment using different imputation strategies and the scaling techniques.
- Test the behavior of the algorithm with different procedures of outliers detection and management, with various thresholds and on different attributes of the dataset.
- Use cross-validation in order to give a more robust estimate of the tested models' performances.

References

[1] S. Asaithambi. *Why, How and When to Scale your Features.* In: Medium. Dec, 2017. URL: https://medium.com/greyatom/why-how-and-when-to-scale-your-features-4b30ab09db5e, accessed in Sept, 2020.

[2] J. Brownlee. *How to Avoid Data Leakage When Performing Data Preparation.* In: Machine Learning Mastery. Jun, 2020. URL: `https://machinelearningmastery.com/data-preparation-without-data-leakage`, accessed in Sept, 2020.

[3] J. Deanand, S. Ghemawat. *MapReduce: Simplified Data Processing on Large Clusters.* In: 6th Symposium on Operating Systems Design and Implementation, 2004.

[4] A. M. El-Habil, K. I. A. Almghari. *Remedy of multicollinearity using Ridge regression.* In: Journal of Al Azhar University-Gaza, pp. 119-134, 2011.

[5] A. Gelman. *Missing-data imputation.* In: Data Analysis Using Regression and Multilevel/Hierarchical Models, pp. 529-543, 2006. Cambridge University Press.

[6] A. E. Hoerl, R. W. Kennard. *Ridge Regression: Biased Estimation for Nonorthogonal Problems.* In: Technometrics, pp. 55-67, 1970.

[7] S. Salloum, R. Dautov, X. Chen, P. X. Peng, J. Z. Huang. *Big data analytics on Apache Spark.* In: International Journal of Data Science and Analytics, pp. 145–164, 2016.

[8] N. Sharma. *Ways to Detect and Remove the Outliers.* In: Towards Data Science. May, 2018. URL: `https://towardsdatascience.com/ways-to-detect-and-remove-the-outliers-404d16608dba`, accessed in Sept, 2020.

[9] A. Swalin. *How to Handle Missing Data.* In: Towards Data Science. Jan, 2018. URL: `https://towardsdatascience.com/how-to-handle-missing-data-8646b18db0d4`, accessed in Jun, 2020.

[10] US Census Bureau. *2013 American Community Survey, Version 2.* May, 2017. Retrieved in June, 2020 from `https://www.kaggle.com/census/2013-american-community-survey`.

[11] S. Wu. *Multicollinearity in Regression.* In: Towards Data Science. May, 2019. URL: `https://towardsdatascience.com/multi-collinearity-in-regression-fe7a2c1467ea`, accessed in Sept, 2020.