SAINEY MANGA
MASTER'S IN DATA SCIENCE AND ECONOMICS, UNIVERSITY OF MILAN

MACHINE LEARNING PROJECT 2019/2020

Author's Note

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

Course Coordinator: PROF. NICOLO CESA-BIANCHI

1. ABSTRACT:

This project aims at predicting the housing prices by implementing ridge regression algorithm for regression from scratch without the use of popular libraries in python. After implementing the ridge regression model, we then evaluated our model with cross validation (K= 10) and studied the dependence of the cross-validated risk estimate on the parameter alpha of ridge regression. Finally, we used Principal Component Analysis (PCA) as a dimensionality reduction tool to improve the risk estimate of our prediction.

2. TOPIC OVERVIEW:

Linear Regression: Linear Regression is a supervised machine learning algorithm where the predicted output is continuous and has a constant slope. In linear regression our predictors are linear functions h: $\mathbb{R}^d \rightarrow \mathbb{R}$ each parameterized by a vector $w \in \mathbb{R}^d$ of real coefficients. That is, $h(x) = w^\top x$. Given a training set $(x1, y_1), \ldots, (x_m, y_m) \in \mathbb{R}^d \times \mathbb{R}$, the linear regression predictor is ERM with respect to the square loss,

$\hat{w} = \underset{w \in \mathbb{R}^d}{argmin} \sum_{t=1}^{m} (w^\top x_t - y_t)^2$, where $\hat{w} = \underset{w \in \mathbb{R}^d}{argmin} ||Sw - y||^2$

Ridge Regression: Ridge Regression is a model tuning method that is used to analyze any data that suffers from multicollinearity. From the linear regression formula ($\hat{w}$), we can instead use the regularized form, also known as Ridge Regression,

$$\hat{w}_\alpha = \underset{w \in \mathbb{R}^d}{argmin} ||Sw - y||^2 - \alpha||w||^2$$

where $\alpha > 0$ is the regularization parameter. When $\alpha \rightarrow 0$ we recover the standard linear regression solution. When $\alpha \rightarrow \infty$, the solution $\hat{w}_\alpha$ becomes the zero vector. This shows that α can be used to control the bias of the algorithm.

Cross Validation: To tune our parameters, that is, to find the best value of the parameters, we use a fraction of the training set as validation set. We run the algorithm only on the validation set and we pick the predictor that has the smallest validation error. It is a technique for evaluating a machine learning model and testing its performance. It helps to compare and select an appropriate model for the specific predictive modeling problem. There are a lot of different techniques that

may be used to **cross-validate** a model, but we applied the **K-Fold** cross validation in this project. K-Fold cross validation introduces a way of splitting the dataset which helps to overcome the "test only once bottleneck".

Principal Component Analysis (PCA): PCA is a dimensionality-reduction method that is often used to reduce the dimensionality of large data sets, by transforming a large set of variables into a smaller one that still contains most of the information in the large set. In PCA, we find the compression matrix $\mathcal{W}$ and the recovering matrix $\boldsymbol{u}$ so that the total squared distance between the original and recovered vectors is minimal; namely, we aim at solving the problem:

$$\underset{\mathcal{W} \in \mathbb{R}^{n,\,d}, \mathcal{U} \in \mathbb{R}^{n,\,d}}{argmin} \sum_{t=1}^{m} ||\mathcal{X}_i - \boldsymbol{u}\mathcal{W}\mathcal{X}_i||^2 2$$

Squared Loss: Squared Loss is a loss function that can be used in the learning setting in which we are predicting a real-valued variable $\mathcal{Y}$ given an input variable $\mathcal{X}$. That is, we are given the following scenario: let h be a hypothesis (i.e., a statistical model). Let S = {($\mathcal{X}$1, $\mathcal{Y}$1), ($\mathcal{X}$2, $\mathcal{Y}$2), . . ., ($\mathcal{X}$n, $\mathcal{Y}$ n)} be our training data where xi ∈ $\mathcal{X}$ are the instances ($\mathcal{X}$ is the space of possible instances) and yi ∈ $\mathbb{R}$ is a numeric value corresponding to each instance. In this setting, the squared loss for a given item in our training data, ($\mathcal{Y}$, $\mathcal{X}$), is given by:

$$\ell_{\text{squared}}(\mathcal{X}, \mathcal{Y}, h) = \left(\mathcal{Y} - h(\mathcal{X})\right)^2$$

Empirical Risk Minimization: A learning algorithm receives as input a training set S, sampled from an unknown distribution **D** and labeled by some target function ***f*,** and should output a predictor hs: $\mathcal{X} \rightarrow \mathcal{Y}$ (the subscript S emphasizes the fact that the output predictor depends on S). The goal of the algorithm is to find hs that minimizes the error with respect to the unknown **D** and ***f*.** Since the training sample is the snapshot of the world that is available to the learner, it makes sense to search for a solution that works well on that data. This learning paradigm – coming up with a predictor h that minimizes $\boldsymbol{L}s(h)$ – is called *Empirical Risk Minimization* or ERM for short.

3. DATA DESCRIPTION AND PREPROCESSING:

The dataset "California Housing prices" contains information from the California 1990 housing census. It consists of ten (10) variables namely:

1. Longitude: A measure of how far west a house is; a higher value is farther west
2. Latitude: A measure of how far north a house is; a higher value is farther north
3. Housing_Median_Age: Median age of a house within a block; a lower number is a newer building
4. Total_Rooms: Total number of rooms within a block
5. Total_Bedrooms: Total number of bedrooms within a block
6. Population: Total number of people residing within a block
7. Households: Total number of households, a group of people residing within a home unit, for a block
8. Median_Income: Median income for households within a block of houses (measured in tens of thousands of US Dollars)
9. Median_House_Value: Median house value for households within a block (measured in US Dollars)
10. Ocean_Proximity: Location of the house w.r.t ocean/sea

	longitude	**latitude**	**HMA**	**TR**	**TB**	**Pop**	**Househ.**	**MI**	**MHV**
Count	20640.00	20640.00	20640.00	20640.00	20433.00	20640.00	20640.00	20640.00	20640.00
Mean	-119.57	35.63	28.64	2635.76	537.87	1425.48	499.54	3.87	206855.82
Std	2.00	2.14	12.59	2181.62	421.39	1132.46	382.33	1.90	115395.62
Min	-124.35	32.54	1.00	2.00	1.00	3.00	1.00	0.50	14999.00
25%	-121.80	33.93	18.00	1447.75	296.00	787.00	280.00	2.56	119600.00
50%	-118.49	34.26	29.00	2127.00	435.00	1166.00	409.00	3.53	179700.00
75%	-118.01	37.71	37.00	3148.00	647.00	1725.00	605.00	4.74	264725.00
Max	-114.31	41.95	52.00	39320.00	6445.00	35682.00	6082.00	15.00	500001.00

Table 1: Described Data

From table 1, we can learn so many facts about our data that will guide us in our choice of data manipulation methods. The Count tuple indicates that our data has a total of 20640 rows with

some missing values in the attribute Total_bedroom because it is the only attribute with a total number of 20433 rows. It is an easy task to check for missing values in python even though the table shows what we want. The attribute Ocean_proximity has been left out in the data description table because it is a categorical variable, and the described function only deals with numerical variables. Another issue we can deduce from our table is that the attributes in the table have different scales, we shall discuss further in the subsequent subsections.

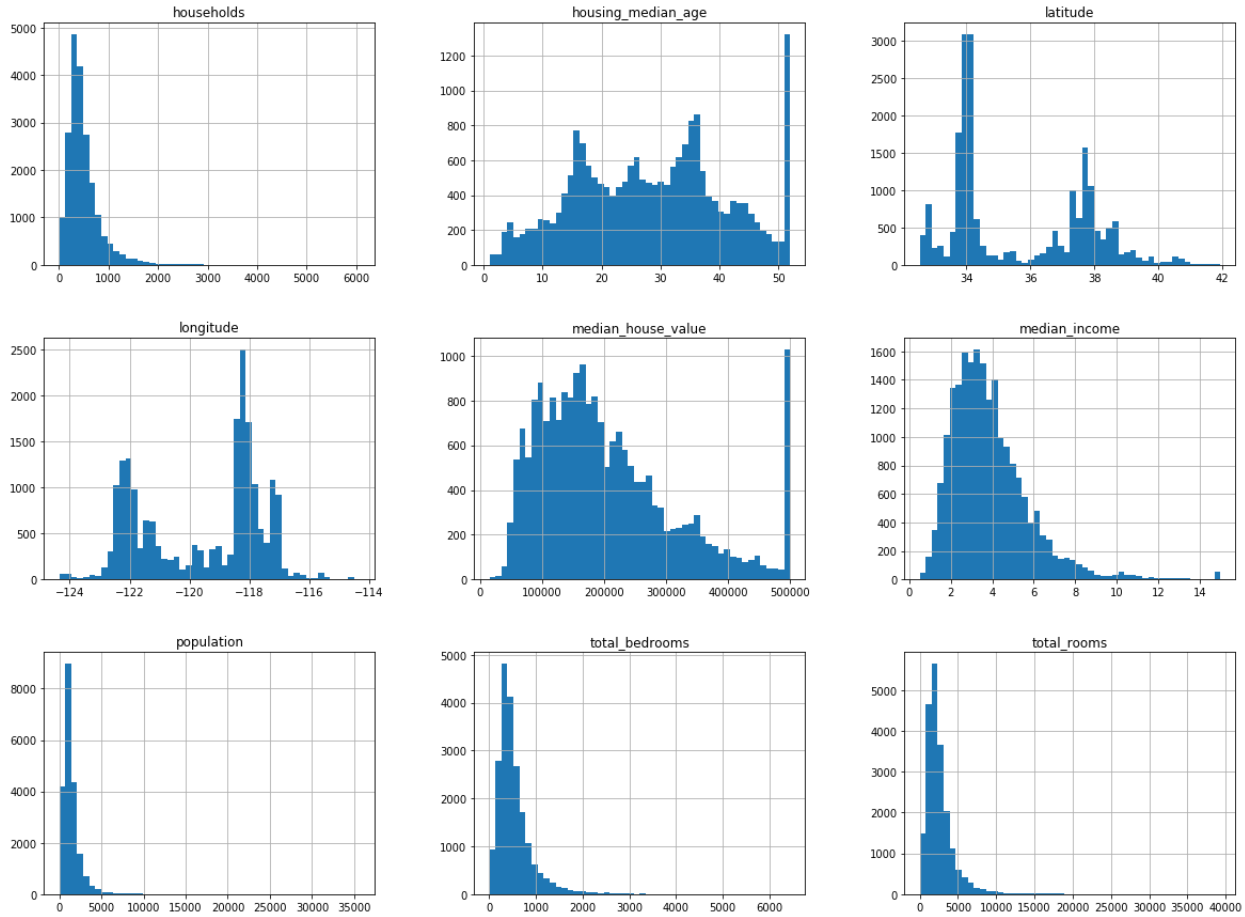

Figure 1: *Histograms showing Data distribution*

To have a better understanding of the nature of data we are dealing with, we plotted the histograms above to see the distribution of the data. We can say that all our attributes are skewed, and that attributes Median_income, Housing_median_age and Meadian_house_value were capped. The latter variable may cause issues for our algorithm in terms of precise predictability since it is the label variable. At this point we can choose to fix the skewedness of our data, but this makes it

difficult during scaling our data as the methods yield almost similar results. So far, we have seen that the Ocean_proximity attribute is still left out in our analysis. That is because it is a categorical variable, and we shall discuss about it exclusively before preparing our model.

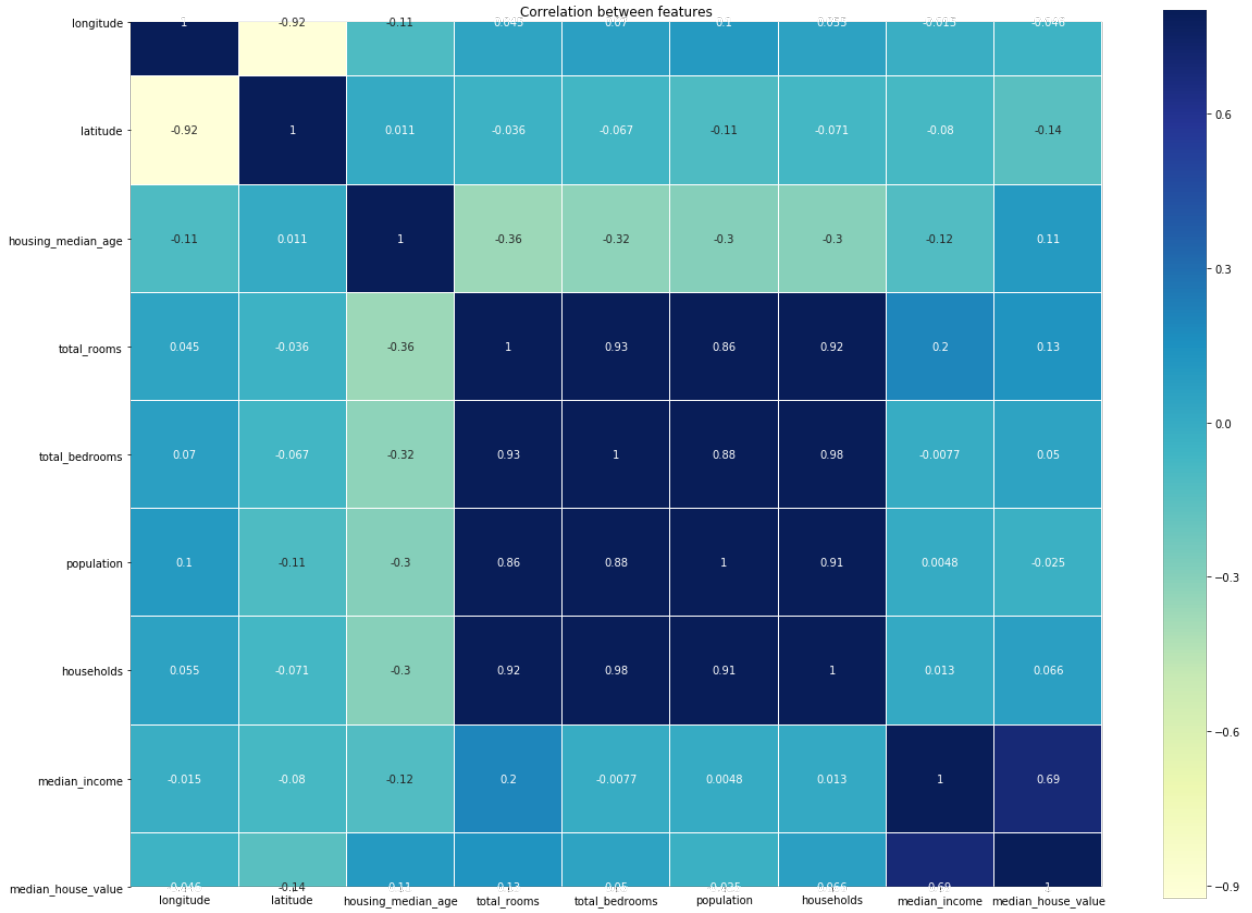

Figure 2: *Correlation Plot*

From Figure 2, we can see that multicollinearity is present amongst our variables that will represent our predictors. For instance, Longitude and Latitude, Total_rooms and Total_bedrooms, Population, and households, all have a correlation above 80%. This raised the question as to whether we should worry about multicollinearity? One idea is to drop some of those variables that are correlated with others that much, but we know that these variables are also so important in determining our target variable. These issues remind us that one of the powers of Ridge Regression is that it can deal with multicollinearity, so we decided not to drop any variables as they will affect our model accuracy.

The issues we must handle before we finally train our model are missing values, categorical variable, and feature scaling.

- **Missing Values:** Our data has only one variable with missing values i.e., Total_bedrooms. It has a total number of 207 missing values which we should choose how to deal with. One easy but naive option is to drop all the tuples with missing values and use the remaining data for our analysis. This option is not a good idea as 207 tuples from our data is a lot of information and we cannot loss just that. Another option is to drop the very attribute with missing values, that also is not appropriate for our purpose as we know the number of Total_bedrooms a house has will indeed affect its price. We choose to fill in the missing values by applying the "median value method". This is done by first calculating the median of the attribute "Total_bedrooms" and then filling the median to the tuples of the said variable that have missing values. We choose this choice because it maintains our original dataset with very negligible modification and avoiding any loss of information.
- **Categorical Variable:** Another issue that requires handling is the categorical variable "Ocean_proximity". Here too, a naïve approach will be to exclude the variable from our predictors, but it is a good idea to see the nature of the very categories and how important they can be in predicting our target variable:

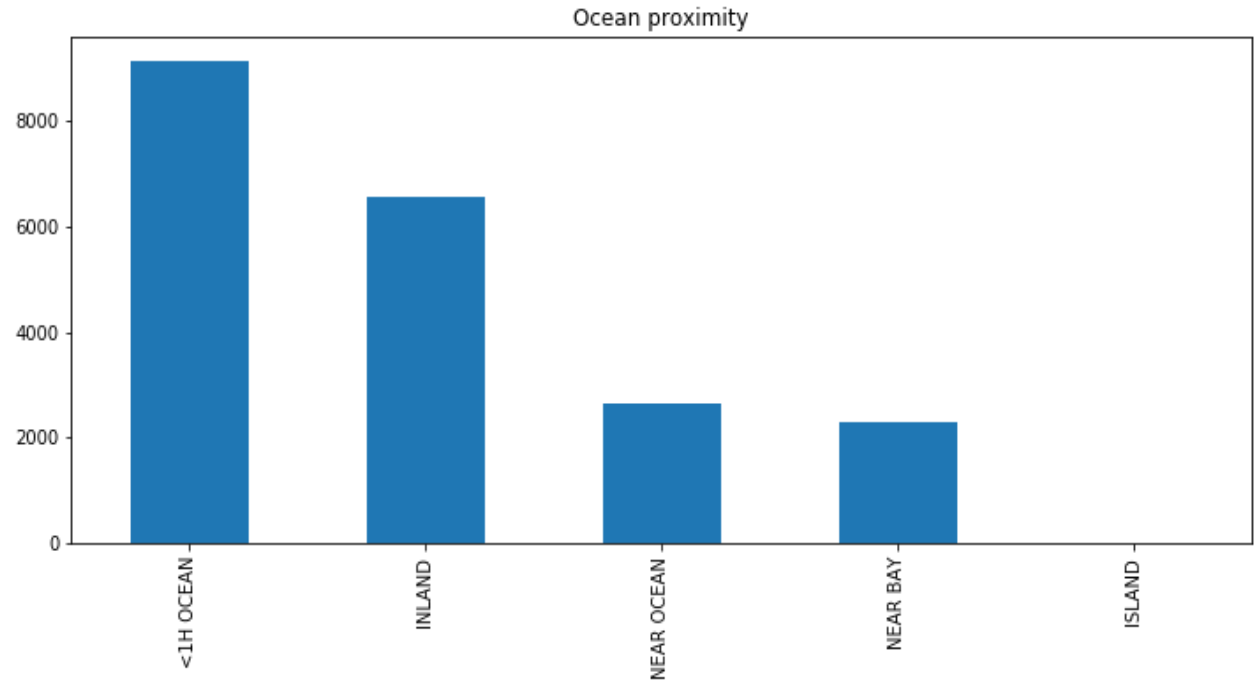

Figure 3: *Categories of Ocean_proximity*

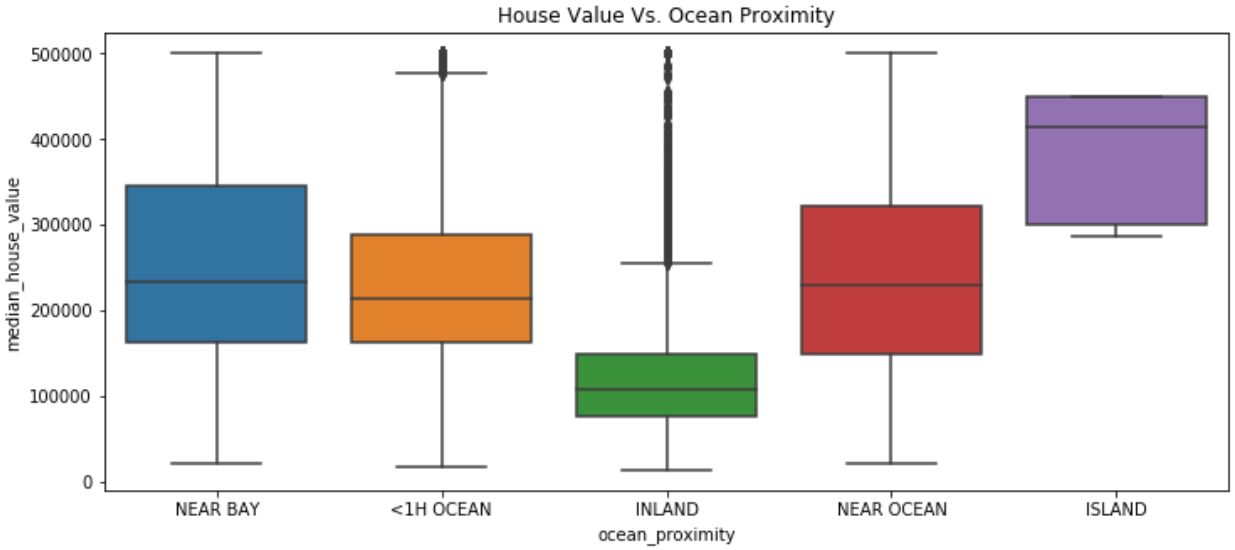

Figure 4: *Categories Vs. Median_house_value*

From Figure 3, the category with the lowest is "Island" with only 5 tuples, but the remaining categories have a significant number of rows. However, we could see from figure 4 that all the categories have an impact on the Median_house_value. Houses in the "Island" are expected to have higher prices than any other proximity. Of course, as expected, houses near "Inland" are expected to have lower prices. Now that we know Ocean_proximity has 5 categories, it can guide our choice in fixing this problem. "Label encoding" simply fixes discrete values on each of the categories (e.g., 0, 1, 2, 3, 4), but the problem of this method takes categories 0 and 1 to be more related than 0 and 4 even though it might not be the case. Since we only have 5 categories, which is a relatively small number, it will be a good idea to adopt "OneHot Encoding". The approach fixes dummies on our categories. In our case, a 1 is taken to be houses "Near Bay" and 0 otherwise. The disadvantage of "OneHot Encoding" though, is that it cannot work in the case where we have many categories to deal with, so we are pretty much safe here.

- **Feature Scaling:** The final step in cleaning our data is to scale it down as we saw before that our data attributes are differently scaled. There are several methods we can implore to achieve this goal. Some of the common methods are Normalization (also known as Min_max method) and Standardization. Before we decide on which method to apply, we need to check for possible outliers in our data as each of the two methods mentioned above are affected by outliers. For

instance, the Normalization method is more affected by outliers than the Standardization method. So, we must deal with outliers first, then decide which method to adopt.

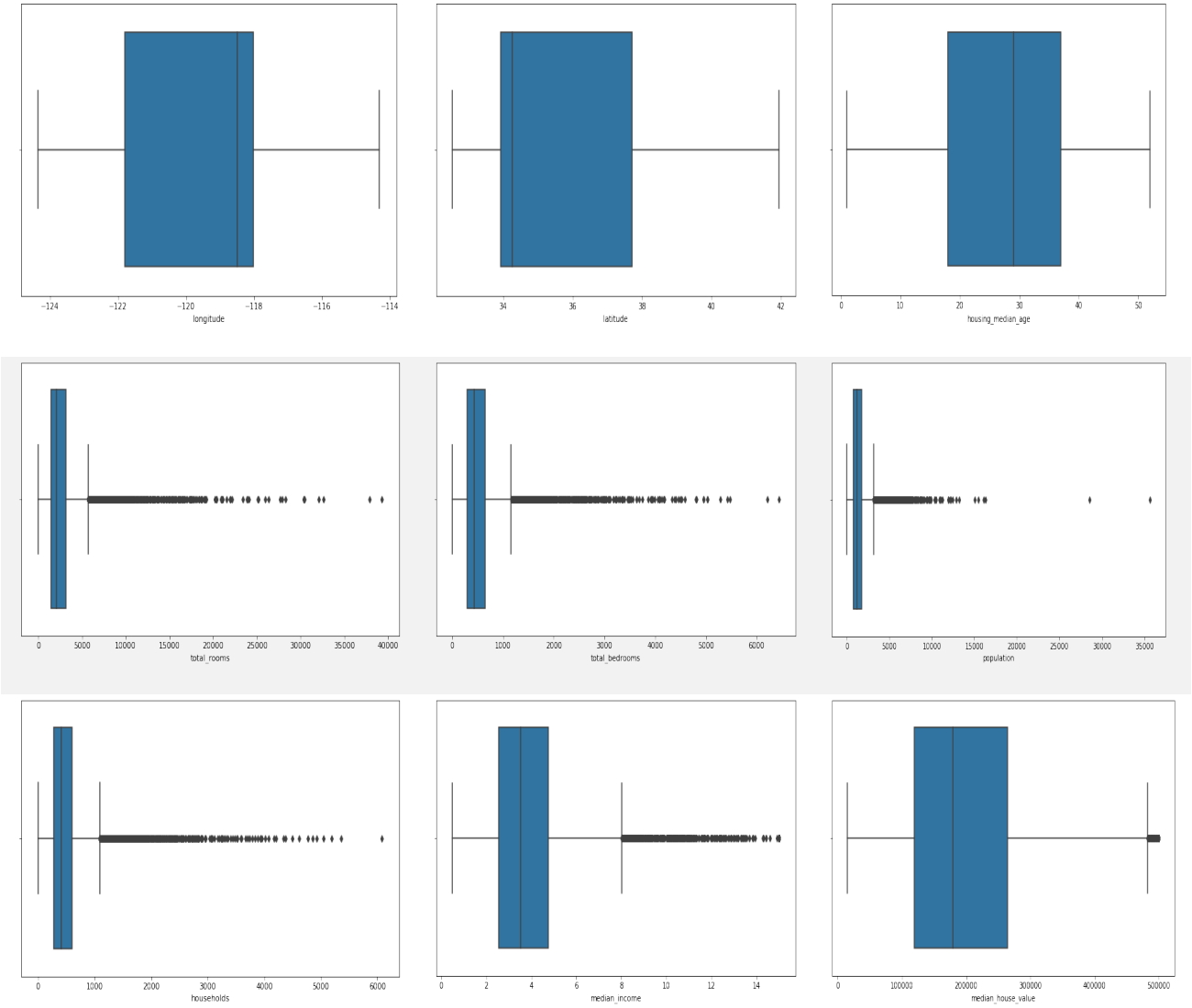

Figure 5: *Outliers*

Figure 5 shows that only three variables are free of outliers, but the remaining six (6) have outliers. After identifying all the variable with outliers, we calculated a fraction of data in each variable that contains those outliers. This method gives us an output of each outlier variable with both an upper bound and lower bound. At this point, we go ahead and cut these portions of data from our dataset so that it can be free of outliers. After cutting out the outliers, our dataset has finally reduced to 17609 from the initial 20640 instances. Finally, we decided to choose the Normalization

(Min_max) method to scale our data as discussed earlier. This indeed, will be our final step of data preparation.

Min-Max Scaler (Normalization) rescales the data to a predefined range, typically 0–1, using the formula:

$$\mathcal{X}_{\text{norm}} = \frac{\mathcal{X} - \mathcal{X}_{\min}}{\mathcal{X}_{\max} - \mathcal{X}_{\min}}$$

The formula means that the minimum is subtracted from the value and divided by the maximum minus the minimum value.

HMA	TR	TB	Pop.	Housh.	MI	MHV	<1H OCEAN	INLAND	ISLAND	NEAR BAY	NEAR OCEAN
1.00	0.26	0.16	0.16	0.16	0.90	0.72	0.0	0.0	0.0	1.0	0.0
1.00	0.22	0.20	0.18	0.20	0.68	0.70	0.0	0.0	0.0	1.0	0.0
1.00	0.29	0.24	0.18	0.24	0.45	0.70	0.0	0.0	0.0	1.0	0.0
1.00	0.16	0.18	0.13	0.18	0.47	0.55	0.0	0.0	0.0	1.0	0.0
1.00	0.45	0.42	0.35	0.47	0.42	0.61	0.0	0.0	0.0	1.0	0.0
1.00	0.54	0.59	0.37	0.59	0.35	0.48	0.0	0.0	0.0	1.0	0.0
0.80	0.45	0.57	0.38	0.54	0.21	0.45	0.0	0.0	0.0	1.0	0.0
1.00	0.62	0.61	0.49	0.65	0.42	0.53	0.0	0.0	0.0	1.0	0.0
1.00	0.39	0.37	0.29	0.37	0.36	0.57	0.0	0.0	0.0	1.0	0.0
1.00	0.62	0.65	0.48	0.67	0.37	0.49	0.0	0.0	0.0	1.0	0.0

Table 2: Normalized Data

We could not show all the variables in Table 2 because of limited space. Now we see that our data has been uniformly scaled down to a range of 0-1, just like we wanted. We could also see that the attribute "Ocean_proximity" has already been encoded using OneHot Encoding as explained before. Finally, our data is as the way we wanted and ready for training.

4. EXPERIMENTAL RESULTS

- **Ridge Regression Algorithm**: We split 80% of our dataset into training set and 20% into Test set using a random state. Our target variable $\mathcal{Y}$ represents Housing_median_value, and our predictor $\mathcal{X}$ represents the remaining variables in our dataset. The regularization parameter α is set between 0.1 and 200 with a vectorized value of 6000. We fitted our ridge regressor on both our

$\mathcal{X}$-train and $\mathcal{Y}$-train using the alpha intervals, then we predicted on both the train set and test set to obtain the training error and the test error respectively.

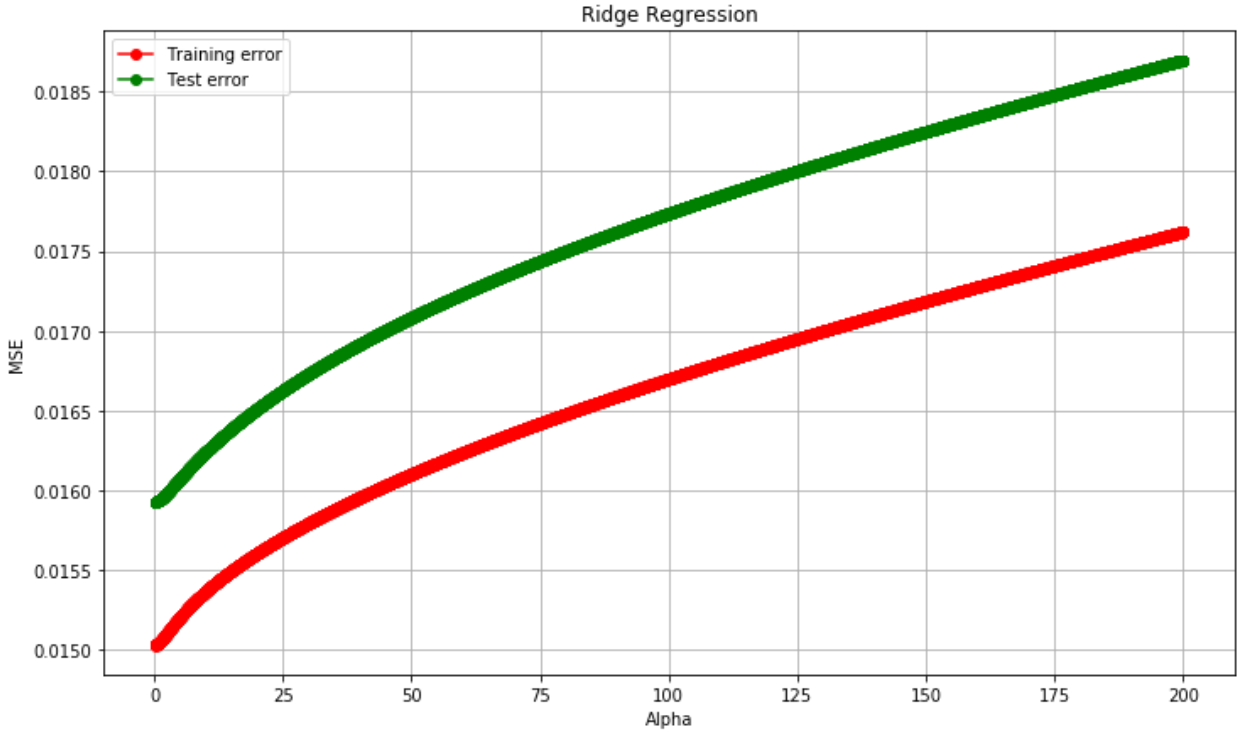

Figure 6: *Ridge Regression*

We see that the distance of our errors was not large at very low values of Alpha, but suddenly widens as Alpha increases. This shows that overfitting does not occur at lower values of alpha, but biasness progressively increases with higher values of Alpha, and this is accompanied with higher MSE. At any given Alpha, training error is lower than the test error of our model.

- **The dependence of the cross-validated risk on Alpha**: We first applied the K-Fold cross-validation with K=10 as the number of splits and Alpha=0.1 to measure the cross-validated risk estimate. On implementing our cross-validation function, the same routine of splitting and fitting the training and test data was applied as the initial stage. The cross validated risk estimate is 0.015, more detail can be seen in the cross-validated diagram on the Jupiter notebook. We then proceeded to study the dependence of the cross-validated risk on Alpha where we maintained the range of Alpha to be from 0.1 to 200 and vectorized value of 6000 as usual. For each of alpha on the range, we measured the cross validated estimate against the dataset and set it as training and test error.

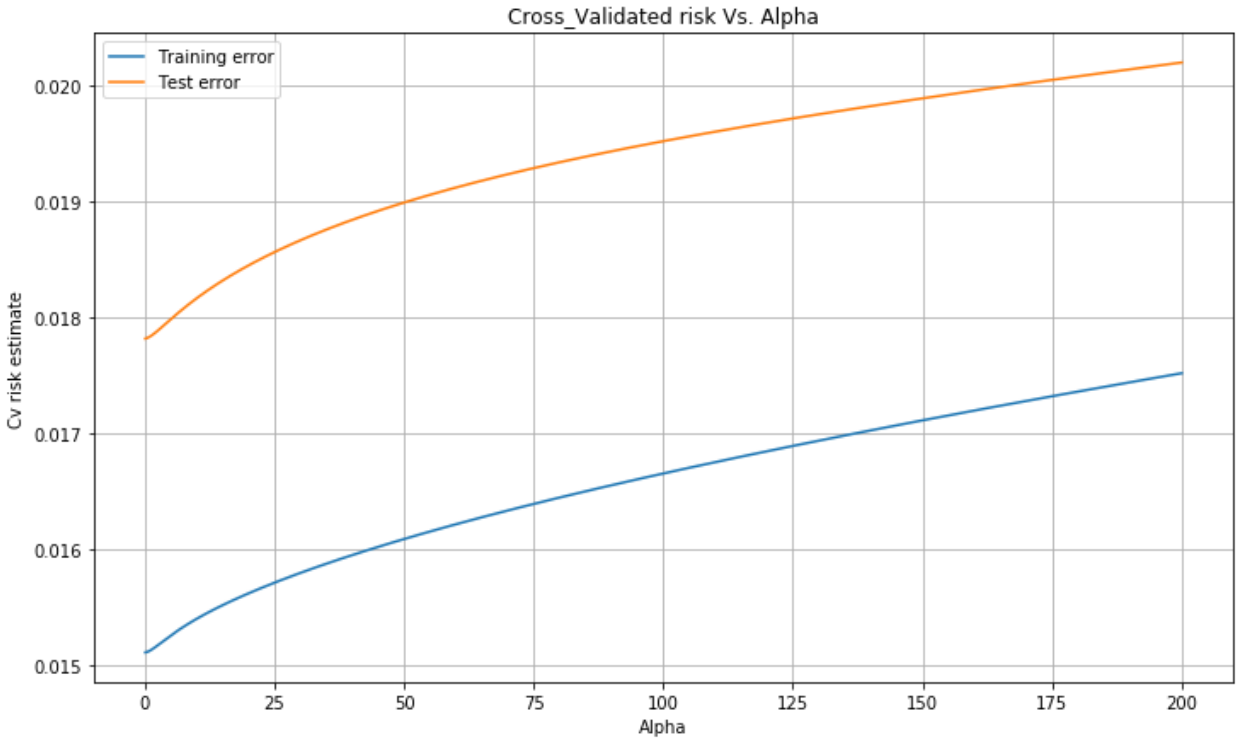

Figure 7: *Dependence of cv risk on Alpha*

From Figure 7, we can deduce that that there is a positive linear relationship between the cross-validated risk estimate and the parameter Alpha. Cross-validated risk estimates increase as the values of parameter Alpha increases. The distance between the errors is wide with training error lower than that of the test error.

- **Trying PCA:** we tried applying PCA to improve the cross-validated risk estimate of our prediction. On the PCA-prediction function, we set up our parameters as the usual Alpha and the traditional train and test set of our target variable and predictors. Then we applied the prediction function on the cross-validated function to return the training and test scores. Setting Alpha as 0.1 and number of splits as 10, we implemented the cross-validated function. The risk estimate produce is 0.016 which is not any better than that of our ridge regression algorithm. Now we implemented the learning curve on our dataset to see the effect of the training size on the risk estimate. The number of components seems to perform better or tend to be more meaningful when we set it between 5 and 6, so we finally choose number of components equal to 6.

We can see from Figure 8 that the lowest values of training size have lowest risk estimate, relatively higher values have fluctuating risk estimate. In a nut cell, risk estimates increase as training size increase then started to fall at some intervals of the training sizes and later increase at

the latter intervals of the training sizes. This suggests that there is no linear relationship between the two.

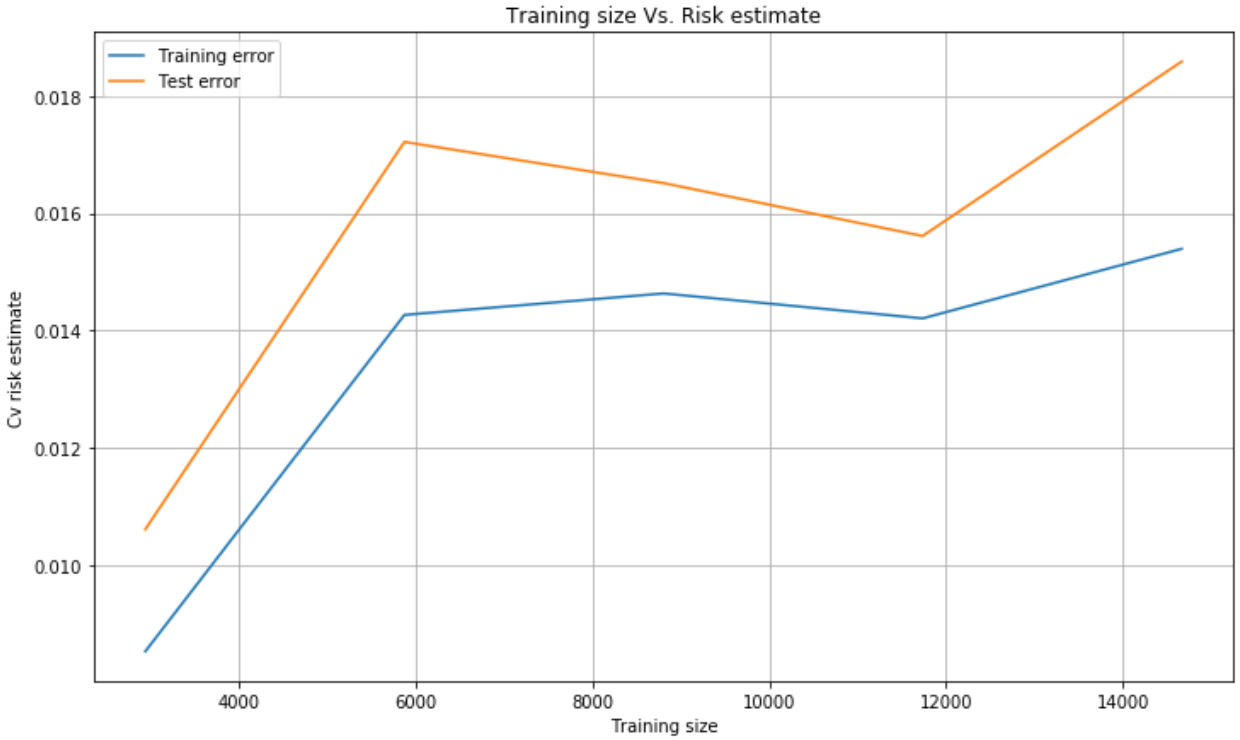

Figure 8: Dependence of risk estimate on training size

5. CONLUSION

The algorithms implemented in this project were Ridge Regression, then we studied the dependence of the cross-validated risk on the Alpha parameter, and finally implemented PCA with the hope of improving the risk estimate.

However, as we have seen the figures of the risk estimates, the Ridge Regression algorithm has a better risk estimate than that of the PCA.

The cross-validated risk estimate increase as the Alpha parameter increased, which indicates a positive linear relationship between the two. Therefore, we say the risk estimate depend on Alpha.

Reference:

(1) Nicolo Cesa-Bianchi, Lecture notes, Linear prediction, Machine Learning (http://cesa-bianchi.di.unimi.it/MSA/Notes/linear.pdf)

(2) Samantha Jackson, Feature Transformation (thttps://medium.com/@sjacks/feature-transformation-21282d1a3215)

(3) Shalev-Shwartz, Ben David, understanding machine learning, theory and algorithm

(4) Aurelien Geron, Hands on machine learning with Scikit-Learn and TensorFlow: Concepts, tools, and techniques to build intelligent systems.

(5) Principal Component Regression vs. Ridge Regression, Nirpy Research (https://nirpyresearch.com/pcr-vs-ridge-regression-nir-data-python/)

(6) Gareth James, Daniela Witten, Trevor Hastie and Robert Tibshirani, Introduction to Statistical Learning with Applications in R.