Machine Learning Engineer Nanodegree

House Prices: Advanced Regression Techniques Using Ensemble Method

German Rezzonico

January 18, 2016

Abstract

An ensemble averaging model, based on real estate data, will be used to make predictions about a home's monetary value. After an exhaustive feature engineering, multiple machine learning regression algorithms will be trained and tuned, to obtain a better predictive performance. A model like the one described, could be invaluable for someone like a real estate agent or for one of the many companies operating in the multi billion dollar real estate industry.

Keywords

Python, regression techniques, feature engineering, grid search, ensemble

1. Definition

1.1. Domain Background

Machine learning is the subfield of computer science that "gives computers the ability to learn without being explicitly programmed" (Arthur Samuel, 1959)[1]. There are different learning styles in machine learning: supervised machine learning ("task of inferring a function from labeled training data")[2] and unsupervised machine learning (task of inferring a function to describe hidden structure from unlabeled data)[3]. Ensemble methods use multiple learning algorithms to obtain better predictive performance than could be obtained from any of these constituent

[1] Phil Simon (March 18, 2013). Too Big to Ignore: The Business Case for Big Data. Wiley. p. 89. ISBN 978-1-118-63817-0.
[2] Mehryar Mohri, Afshin Rostamizadeh, Ameet Talwalkar (2012) *Foundations of Machine Learning*, The MIT Press ISBN 9780262018258.
[3] https://en.wikipedia.org/wiki/Unsupervised_learning

learning algorithms alone.[4] Ensemble averaging take the mean of individual model predictions and often reduces overfit.

A real estate is “property consisting of land and the buildings on it”[5]. A real estate appraisal, property valuation or land valuation is the process of developing an opinion of value for real property (usually market value)… and every property is unique...”[6]. “The real estate industry is a big business generating billions of dollars in revenue annually. In 2016 there were approximately 210,000 companies operating in the residential brokerage and management field, which generated $200 billion in revenue; there were 35,000 companies operating in the commercial brokerage and management field, generating $35 billion in revenue”.[7] Is it possible to train a model, based on real state data, to be used to make predictions about a home’s monetary value? Such model could be invaluable for someone like a real estate agent or for one of the many companies operating in the real state industry.

The Boston Housing Data Set (Harrison and Rubinfeld 1978)[8] have been publicly available for some years and many research works have been published. On the other hand, the Ames Housing dataset[9] compiled by Dean De Cock for use in data science education, is a modernized and expanded version of the often cited Boston Housing dataset. And a competition on Kaggle[10] , the House Prices: Advanced Regression Techniques[11], using the Kaggle version[12] of the Ames dataset presents an incredible opportunity in using machine learning ensemble methods to predict the house prices.

1.2. Problem Statement

In this problem we will predict, through supervised machine learning methods more specifically through regression techniques, housing prices using the Kaggle version of the Ames dataset. To find a solution to this problem, at a first attempt the first two and the first ten most correlated features[13] will be used to train several regressors[14]. Once the regressors are trained the solution to this problem can be achieved, that is the predictions on the house pricing can be made and, according to these predictions, the RSMLE score[15] of the regressors and of the ensemble can be calculated to see if a specific value of RSMLE have been reach by comparing these values

---

[4] Opitz, D.; Maclin, R. (1999). "Popular ensemble methods: An empirical study". *Journal of Artificial Intelligence Research*. **11**: 169–198. doi:10.1613/jair.614.
[5] https://en.wikipedia.org/wiki/Real_estate
[6] https://en.wikipedia.org/wiki/Real_estate_appraisal
[7] https://www.franchisehelp.com/industry-reports/real-estate-franchise-industry-report/
[8] Harrison, D. and Rubinfeld, D. L. (1978). “Hedonic Housing Prices and the Demand for Clean Air,” Journal of Environmental Economics and Management, 5, 81-102.
[9] Journal of Statistics Education Volume 19, Number 3 (2011), www.amstat.org/publications/jse/v19n3/decock.pdf
[10] https://www.kaggle.com
[11] https://www.kaggle.com/c/house-prices-advanced-regression-techniques
[12] https://www.kaggle.com/c/house-prices-advanced-regression-techniques/data
[13] 2.2.4. Correlation
[14] 1.2.4.
[15] 1.3. Evaluation Metrics

to the benchmark model[16]. Later on, feature engineering and grid search will be evaluated as a proposed improvement[17] to see if the RSMLE score can be increased using these techniques.

The tasks involved are the following:

1.2.1. EDA (Exploratory Data Analysis)[18]: to get a better understanding of the data.
1.2.2. Feature Engineering[19] [20]: to improve the performance of the estimators.
1.2.3. Feature Selection[21]: to select the most significant features.
1.2.4. The following regression techniques will be implemented: DecisionTreeRegressor[22], SVR[23][24], ElasticNet[25], Lasso[26], LassoLars[27], BayesianRidge[28], GradientBoostingRegressor[29], ExtraTreesRegressor[30], BaggingRegressor[31], AdaBoostRegressor[32] and XGBRegressor[33].

---

[16] 2.4. Benchmark Model
[17] 5.3. Improvement
[18] Tukey, John W. (1977). *Exploratory Data Analysis*. Pearson. ISBN 978-0201076165.
[19] "Discover Feature Engineering, How to Engineer Features and How to Get Good at It - Machine Learning Mastery". *Machine Learning Mastery*. Retrieved 2015-11-11.
[20] 2.1. Data Exploration
[21] Guyon, Isabelle; Elisseeff, André (2003). "An Introduction to Variable and Feature Selection". *JMLR*. **3**
[22] "sklearn.tree.DecisionTreeRegressor — scikit-learn 0.18.1 ...." http://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeRegressor.html. Accessed 18 Jan. 2017.
[23] "sklearn.svm.SVR — scikit-learn 0.18.1 documentation." http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVR.html. Accessed 18 Jan. 2017.
[24] "A tutorial on support vector regression - Alex Smola." https://alex.smola.org/papers/2004/SmoSch04.pdf. Accessed 18 Jan. 2017.
[25] "Elastic Net - Stanford University." http://web.stanford.edu/~hastie/TALKS/enet_talk.pdf. Accessed 18 Jan. 2017.
[26] "Regression shrinkage and selection via the lasso. - Department of ...." 15 Feb. 2003, http://statweb.stanford.edu/~tibs/lasso/lasso.pdf. Accessed 18 Jan. 2017.
[27] "linear_model.LassoLARS - Scikit-learn." http://scikit-learn.sourceforge.net/0.7/modules/generated/scikits.learn.linear_model.LassoLARS.html. Accessed 18 Jan. 2017.
[28] "Bayesian Ridge Regression — scikit-learn 0.18.1 documentation." http://scikit-learn.org/stable/auto_examples/linear_model/plot_bayesian_ridge.html. Accessed 18 Jan. 2017.
[29] "3.2.4.3.6. sklearn.ensemble.GradientBoostingRegressor — scikit-learn ...." http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingRegressor.html. Accessed 18 Jan. 2017.
[30] "3.2.4.3.4. sklearn.ensemble.ExtraTreesRegressor — scikit-learn 0.18 ...." http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.ExtraTreesRegressor.html. Accessed 18 Jan. 2017.
[31] "sklearn.ensemble.BaggingRegressor — scikit-learn 0.18.1 ...." http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.BaggingRegressor.html. Accessed 18 Jan. 2017.
[32] "sklearn.ensemble.AdaBoostRegressor — scikit-learn 0.18.1 ...." http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostRegressor.html. Accessed 18 Jan. 2017.
[33] "GitHub - dmlc/xgboost: Scalable, Portable and Distributed Gradient ...." https://github.com/dmlc/xgboost. Accessed 18 Jan. 2017.

1.2.5. Hyperparameter Optimization[34]: to tune those models through the grid search technique[35] [36].
1.2.6. Model selection and evaluation[37]: the results from these regressions will be analyzed and the best models will be chosen, according to their performances (RMSLE score[38]).
1.2.7. Ensemble averaging generation[39]: to obtain the mean of the individual models predictions.
1.2.8. Compare to benchmark: the RMSLE of the ensemble will be calculated and compared to the one of the benchmark model[40], to determine if the target was reached.

1.3. Evaluation Metrics

To quantify the model's performance R^2 and RMSLE will be used, MSE and RMSE are also defined here to provide the reader a better understanding of RMSLE. R^2 will provide a good measure of how well the regression approximates the real data points and RMSLE will provide an absolute measure of fit.

R^2 was selected as an evaluation metric because is a useful statistic in regression analysis, to see how good the model is at making predictions. Using this metric can be seen how much of the total variability of the predicted feature can be explain by the model. However, in this problem R^2 is not sufficient, because a measure of dispersion explaining how far these predictions are from the real values is also needed, this would be mse. RSMLE was selected by Kaggle to be used in this competition, what makes sense since the logarithm present in this formula means that errors in predicting expensive houses and cheap houses will affect the result equally.

- Coefficient of determination[41] [42]: $R^2 = \left(\frac{1}{n} \times \sum_{i=1}^{n} \frac{(x_i - x) - (y_i - y)}{(\sigma_x - \sigma_y)} \right)^2$

Where n is the number of observations used to fit the model, Σ is the summation symbol, x_i is the x value for observation i, x is the mean x value, y_i is the y value for observation i, y is the mean y value, σ_x is the standard deviation of x, and σ_y is the standard deviation of y.

The values for R^2 range from 0 to 1, which captures the percentage of squared correlation between the predicted and actual values of the target variable. A model with an R^2 of 0 is no better than a model that always predicts the mean of the target variable, whereas a model with an R^2 of 1 perfectly predicts the target variable. R^2 values indicates what percentage of the

---

[34] https://en.wikipedia.org/wiki/Hyperparameter_optimization
[35] http://scikit-learn.org/stable/modules/grid_search.html
[36] http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html
[37] https://sebastianraschka.com/blog/2016/model-evaluation-selection-part1.html
[38] 3. Evaluation Metrics
[39] Naftaly, U., N. Intrator, and D. Horn. "Optimal ensemble averaging of neural networks." Network: Computation in Neural Systems 8, no. 3 (1997): 283–296.
[40] 2.4. Benchmark Model
[41] http://stattrek.com/statistics/dictionary.aspx?definition=coefficient_of_determination
[42] http://scikit-learn.org/stable/modules/generated/sklearn.metrics.r2_score.html

target variable, using this model, can be explained by the features. A model can be given a negative R^2 as well, which indicates that the model is arbitrarily worse than one that always predicts the mean of the target variable.

- Mean Squared Error (MSE)[43] [44]: $MSE = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2$

Where $\hat{y}_i = predictions$ and $y_i = true\ values$.

MSE measures the average of the squares of the errors or deviations and is a measure of the quality of an estimator. It is always non-negative, and values closer to zero are better.

- Root Mean Square Deviation (RMSD) or Root Mean Square Error (RMSE)[45]: $RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2} = \sqrt{1 - R^2} \times \sigma_y$

RMSE is used as a measure of the differences between values (sample and population values) predicted by a model or an estimator and the values actually observed. The RMSE represents the sample standard deviation of the differences between predicted values and observed values. RMSE is a good measure of accuracy, but only to compare forecasting errors of different models for a particular variable.

Always between 0 and 1, since R is between -1 and 1. It tells us how much smaller the RSME will be than the SD.

For example, if all the points lie exactly on a line with positive slope, then R will be 1, and the RMSE will be 0.

- Root Mean Squared Logarithmic Error (RMSLE)[46]: $RMSLE = \varepsilon = \sqrt{\frac{1}{n} \times \sum_{i=1}^{n} \left(log\left(\hat{y}_i + 1\right) - log\left(y_i + 1\right)\right)^2}$

Where: n is the total number of observations in the (public/private) data set, $\hat{y}_i$ is the prediction, and y_i is the actual response for i and log(x) is the natural logarithm of x.

RMSLE penalizes an under-predicted estimate greater than an over-predicted estimate. In the case of the Ames dataset, logs means that errors in predicting expensive houses and cheap houses will affect the result equally. Values closer to zero are better.

2. Analysis

2.1. Data Exploration

The Ames Housing dataset[47] compiled by Dean De Cock for use in data science education, in particular the version available in Kaggle[48], will be used in this project. This dataset is divided into two parts: the train dataset has 1460 data points with 81 features each and the test dataset has 1459 data points with 80 features each. In the test dataset the feature 'SalePrice' have

[43] https://en.wikipedia.org/wiki/Mean_squared_error
[44] http://scikit-learn.org/stable/modules/generated/sklearn.metrics.mean_squared_error.html
[45] https://en.wikipedia.org/wiki/Root-mean-square_deviation
[46] https://www.kaggle.com/wiki/RootMeanSquaredLogarithmicError
[47] http://www.amstat.org/publications/jse/v19n3/decock.pdf
[48] https://www.kaggle.com/c/house-prices-advanced-regression-techniques/data

been removed, because this is the predicted feature. The dataset describes the sale of individual residential properties in Ames, Iowa from 2006 to 2010. There are a large number of explanatory features (23 nominal, 23 ordinal, 14 discrete, and 20 continuous) involved in assessing home values. In the following table are the 81 features, along with its data type and description[49], it can also be seen also that there are some features that have large number of missing values:

Feature	Type	Description	Null / NA Values Train Set	Null / NA Values Test Set
MSSubClass	Nominal	Identifies the type of dwelling involved in the sale	0	0
MSZoning	Nominal	Identifies the general zoning classification of the sale	0	4
LotFrontage	Continuous	Linear feet of street connected to property	259	227
LotArea	Continuous	Lot size in square feet	0	0
Street	Nominal	Type of road access to property	0	0
Alley	Nominal	Type of alley access to property	1369	1352
LotShape	Ordinal	General shape of property	0	0
LandContour	Nominal	Flatness of the property	0	0
Utilities	Ordinal	Type of utilities available	0	2
LotConfig	Nominal	Lot configuration	0	0
LandSlope	Ordinal	Slope of property	0	0
Neighborhood	Nominal	Physical locations within Ames city limits	0	0
Condition1	Nominal	Proximity to various conditions	0	0
Condition2	Nominal	Proximity to various conditions (if more than one is present)	0	0
BldgType	Nominal	Type of dwelling	0	0
HouseStyle	Nominal	Style of dwelling	0	0
OverallQual	Ordinal	Rates the overall material and finish of the house	0	0
OverallCond	Ordinal	Rates the overall condition of the house	0	0
YearBuilt	Discrete	Original construction date	0	0
YearRemodAdd	Discrete	Remodel date (same as construction date if no remodeling or additions)	0	0
RoofStyle	Nominal	Type of roof	0	0
RoofMatl	Nominal	Roof material	0	0
Exterior1st	Nominal	Exterior covering on house	0	1
Exterior2nd	Nominal	Exterior covering on house (if more than one material)	0	1

[49] More information on data/house_prices/data_description.txt

MasVnrType	Nominal	Masonry veneer type	8	16
MasVnrArea	Continuous	Masonry veneer area in square feet	8	15
ExterQual	Ordinal	Evaluates the quality of the material on the exterior	0	0
ExterCond	Ordinal	Evaluates the present condition of the material on the exterior	0	0
Foundation	Nominal	Type of foundation	0	0
BsmtQual	Ordinal	Evaluates the height of the basement	37	44
BsmtCond	Ordinal	Evaluates the general condition of the basement	37	45
BsmtExposure	Ordinal	Refers to walkout or garden level walls	38	44
BsmtFinType1	Ordinal	Rating of basement finished area	37	42
BsmtFinSF1	Continuous	Type 1 finished square feet	0	1
BsmtFinType2	Ordinal	Rating of basement finished area (if multiple types	0	42
BsmtFinSF2	Continuous	Type 2 finished square feet	0	1
BsmtUnfSF	Continuous	Unfinished square feet of basement area	1	1
TotalBsmtSF	Continuous	Total square feet of basement area	0	1
Heating	Nominal	Type of heating	0	0
HeatingQC	Ordinal	Heating quality and condition	0	0
CentralAir	Nominal	Central air conditioning	0	0
Electrical	Ordinal	Electrical system	0	0
1stFlrSF	Continuous	First Floor square feet	0	0
2ndFlrSF	Continuous	Second floor square feet	0	0
LowQualFinSF	Continuous	Low quality finished square feet (all floors)	0	0
GrLivArea	Continuous	Above grade (ground) living area square feet	0	0
BsmtFullBath	Discrete	Basement full bathrooms	0	2
BsmtHalfBath	Discrete	Basement half bathrooms	0	2
FullBath	Discrete	Full bathrooms above grade	0	0
HalfBath	Discrete	Half baths above grade	0	0
BedroomAbvGr	Discrete	Bedrooms above grade (does NOT include basement bedrooms)	0	0
KitchenAbvGr	Discrete	Kitchens above grade	0	0
KitchenQual	Ordinal	Kitchen quality	0	1
TotRmsAbvGrd	Discrete	Total rooms above grade (does not include bathrooms)	0	0
Functional	Ordinal	Home functionality (Assume typical unless deductions are warranted)	0	2
Fireplaces	Discrete	Number of fireplaces	0	0
FireplaceQu	Ordinal	Fireplace quality	690	730
GarageType	Nominal	Garage location	81	76

GarageYrBlt	Discrete	Year garage was built	81	78
GarageFinish	Ordinal	Interior finish of the garage	81	78
GarageCars	Discrete	Size of garage in car capacity	0	1
GarageArea	Continuous	Size of garage in square feet	0	1
GarageQual	Ordinal	Garage quality	81	78
GarageCond	Ordinal	Garage condition	81	78
PavedDrive	Ordinal	Paved driveway	0	0
WoodDeckSF	Continuous	Wood deck area in square feet	0	0
OpenPorchSF	Continuous	Open porch area in square feet	0	0
EnclosedPorch	Continuous	Enclosed porch area in square feet	0	0
3SsnPorch	Continuous	Three season porch area in square feet	0	0
ScreenPorch	Continuous	Screen porch area in square feet	0	0
PoolArea	Continuous	Pool area in square feet	0	0
PoolQC	Ordinal	Pool quality	1453	1456
Fence	Ordinal	Fence quality	1179	1169
MiscFeature	Nominal	Miscellaneous feature not covered in other categories	1406	1408
MiscVal	Continuous	Value of miscellaneous feature	0	0
MoSold	Discrete	Month Sold (MM)	0	0
YrSold	Discrete	Year Sold (YYYY)	0	0
SaleType	Nominal	Type of sale	0	1
SaleCondition	Nominal	Condition of sale	0	0
SalePrice	Continuous	Predicted feature (only present in the train set)	0	0

Table 1 - Features description

The 20 continuous variables relate to various area dimensions for each observation. The 14 discrete variables typically quantify the number of items occurring within the house. There are a large number of categorical variables (23 nominal, 23 ordinal). They range from 2 to 28 classes. The nominal variables identify various types of dwellings, garages, materials, and environmental conditions. The ordinal variables rate various items within the property.

Two example data points, randomly selected from both data sets, can be seen in Table 4 and 5 respectively.

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	...	MoSold	YrSold	SaleType	SaleCondition	SalePrice
1192	1193	50	RM	60	9600	...	7	2007	WD	Normal	125000
527	528	60	RL	67	14948	...	11	2008	New	Partial	446261

Table 3 - Sample data points from train set

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	...	MoSold	YrSold	SaleType	SaleCondition
1192	2653	20	RL	110	16163	...	7	2006	WD	Normal

527	1988	60	RL	80	9024	...	7	2008	WD	Normal

Table 4 - Sample data points from test set

Due to the large number of features, only some of them will be selected for further statistical analysis and for a first implementation of the regression algorithms.

	OverallQual	GrLivArea	GarageCars	GarageArea	TotalBsmtSF	1stFlrSF	FullBath	TotRmsAbvGrd	YearBuilt	YearRemodAdd	SalePrice
count	1460.0	1460.0	1460.0	1460.0	1460.0	1460.0	1460.0	1460.0	1460.0	1460.0	1460.0
mean	6.1	1515.5	1.8	473.0	1057.4	1162.6	1.6	6.5	1971.3	1984.9	180921.2
std	1.4	525.5	0.7	213.8	438.7	386.6	0.6	1.6	30.2	20.6	79442.5
min	1.0	334.0	0.0	0.0	0.0	334.0	0.0	2.0	1872.0	1950.0	34900.0
0.3	5.0	1129.5	1.0	334.5	795.8	882.0	1.0	5.0	1954.0	1967.0	129975.0
0.5	6.0	1464.0	2.0	480.0	991.5	1087.0	2.0	6.0	1973.0	1994.0	163000.0
0.8	7.0	1776.8	2.0	576.0	1298.3	1391.3	2.0	7.0	2000.0	2004.0	214000.0
max	10.0	5642.0	4.0	1418.0	6110.0	4692.0	3.0	14.0	2010.0	2010.0	755000.0

Table 5 - Statistical analysis of selected features of train set

	OverallQual	GrLivArea	GarageCars	GarageArea	TotalBsmtSF	1stFlrSF	FullBath	TotRmsAbvGrd	YearBuilt	YearRemodAdd
count	1459.0	1459.0	1458.0	1458.0	1458.0	1459.0	1459.0	1459.0	1459.0	1459.0
mean	6.1	1486.0	1.8	472.8	1046.1	1156.5	1.6	6.4	1971.4	1983.7
std	1.4	485.6	0.8	217.0	442.9	398.2	0.6	1.5	30.4	21.1
min	1.0	407.0	0.0	0.0	0.0	407.0	0.0	3.0	1879.0	1950.0
0.3	5.0	1117.5	1.0	318.0	784.0	873.5	1.0	5.0	1953.0	1963.0
0.5	6.0	1432.0	2.0	480.0	988.0	1079.0	2.0	6.0	1973.0	1992.0
0.8	7.0	1721.0	2.0	576.0	1307.0	1382.5	2.0	7.0	2001.0	2004.0
max	10.0	5095.0	5.0	1488.0	5095.0	5095.0	4.0	15.0	2010.0	2010.0

Table 6 - Statistical analysis of selected features of test set

2.2. Exploratory Visualization

2.2.1. Null values

Figures 1 and 2 shows, orderly, the numbers of null values. This will be useful for working with the data in order to make the predictions.

Figure 1: barplot - null values in train set

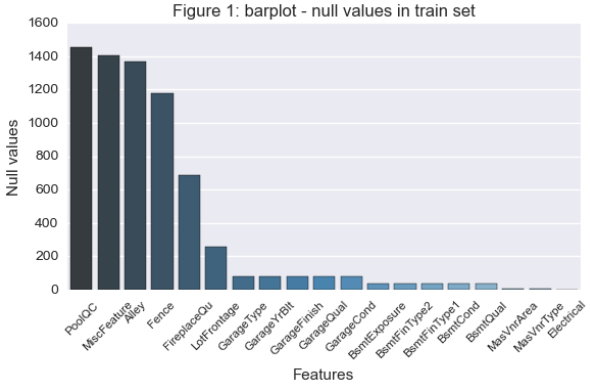
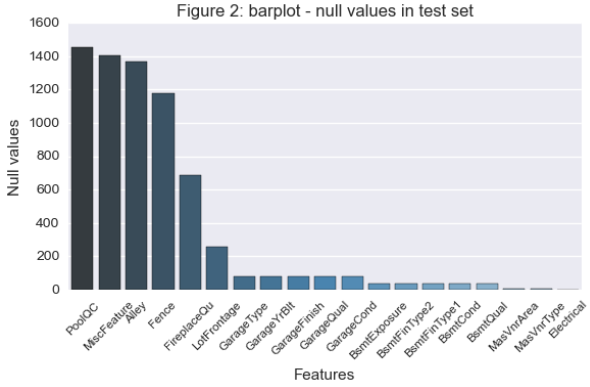


Figure 2: barplot - null values in test set



2.2.2. Barplots

Figures 3, 4 and 5 shows, grouped by similar mean values, an estimate of central tendency for the numeric variables with the height of each rectangle and provides some indication of the uncertainty around that estimate using error bars.

Figure 3: Barplot - mean and standard error

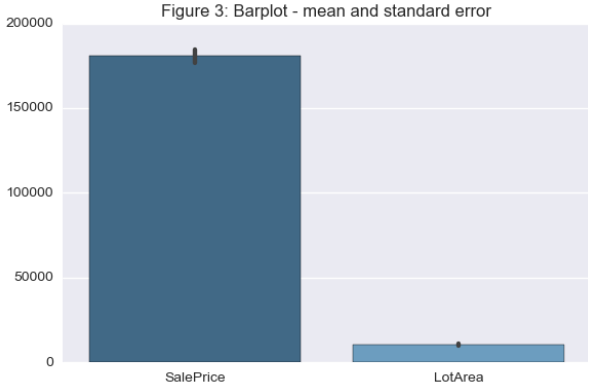


Figure 4: Barplot - mean and standard error

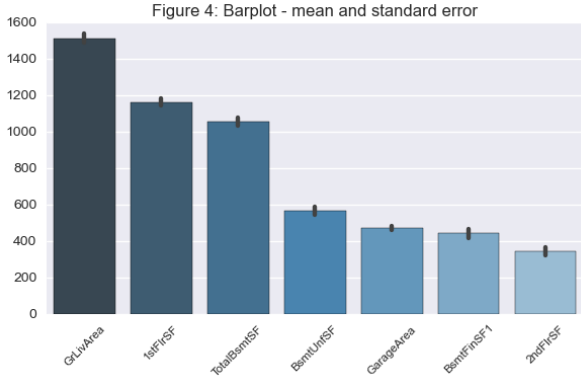
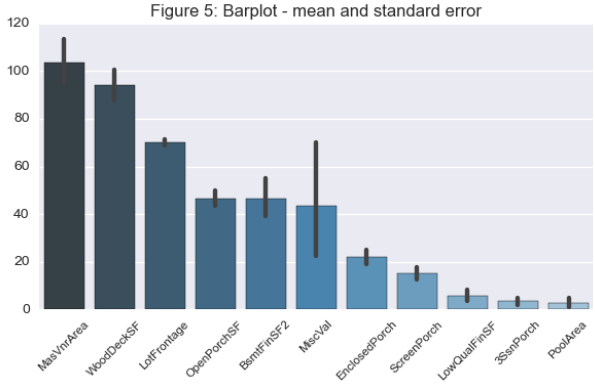


Figure 5: Barplot - mean and standard error



2.2.3. Horizontal Box Plots

Figures 6, 7 and 8 shows, grouped by similar mean values, horizontal box plot exhibiting the distribution of quantitative data in a way that facilitates comparisons between variables. The box shows the quartiles of the dataset while the whiskers extend to show the rest of the distribution, except for points that are determined to be “outliers” using a method that is a function of the inter-quartile range.

Figure 6: horizontal box plot

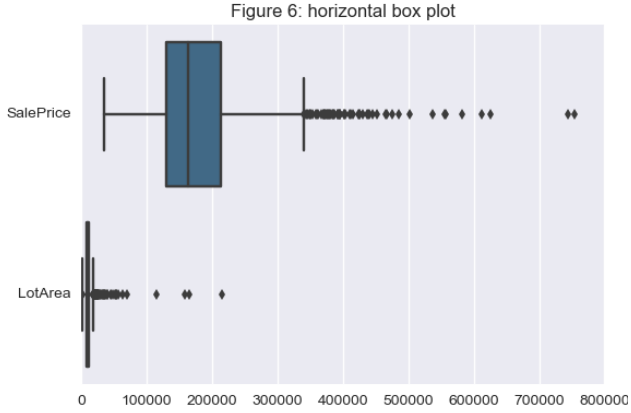


Figure 7: horizontal box plot

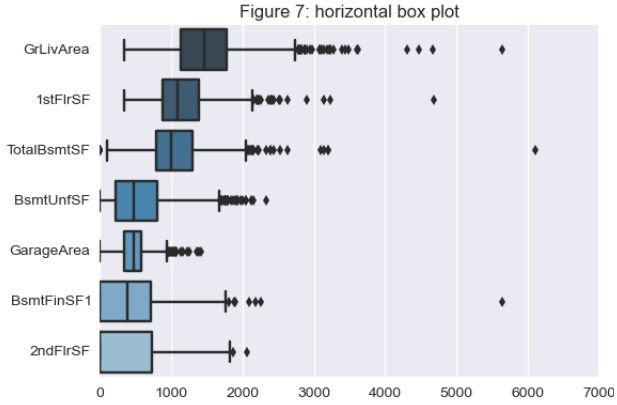
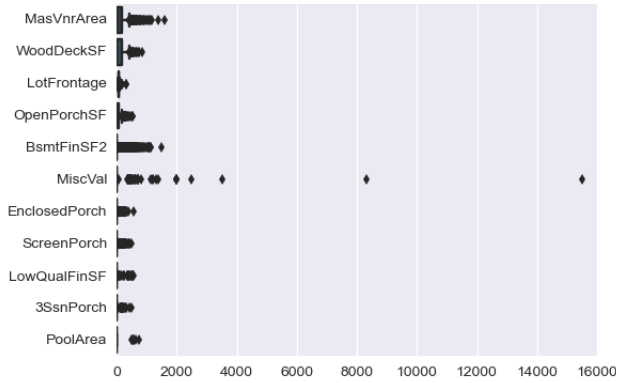


Figure 8: horizontal box plot

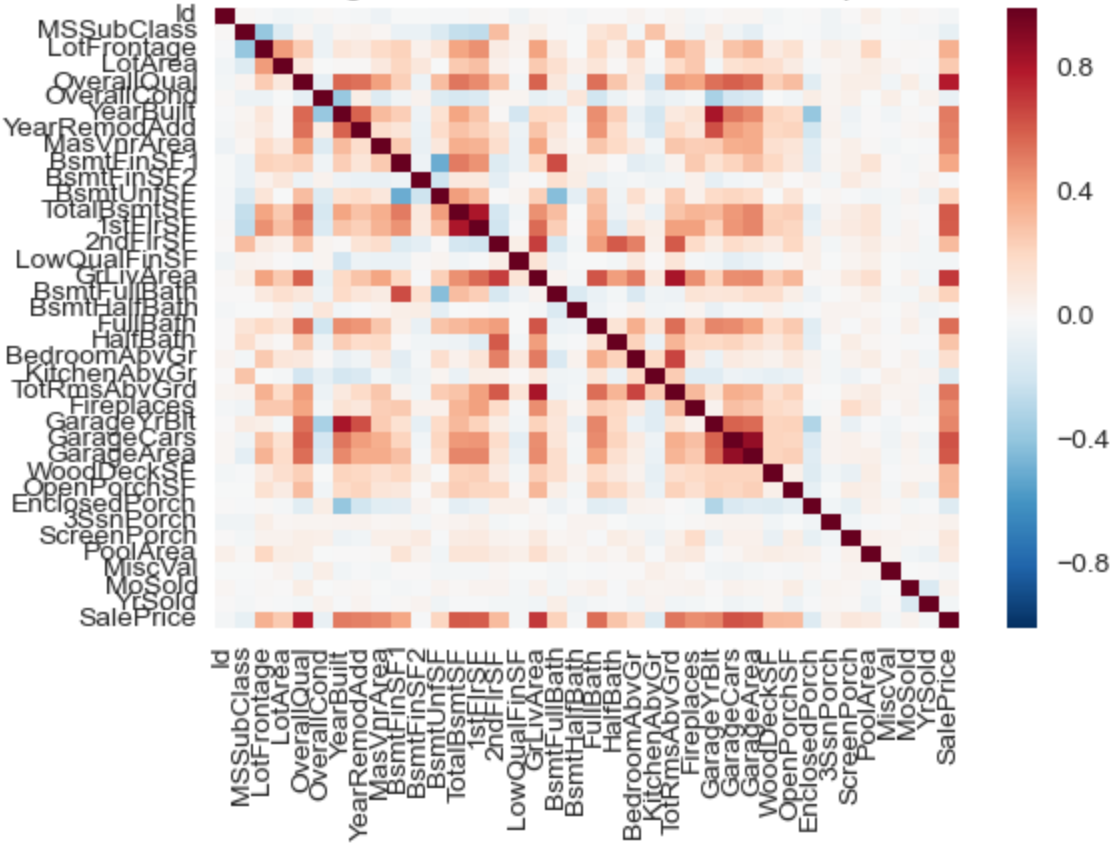


2.2.4. Correlation - Default Features

Figures 9, 10 and 11 shows a mutual relationship or connection between the quantitative features.

Table 4 shows the correlation between the quantitative features and 'SalePrice'. This information will serve to select the first two and the first ten most correlated features, to be used as a first approach to solve the problem before feature engineering is applied.

Figure 9: Correlation matrix heatmap



Feature	Correlation	Feature	Correlation	Feature	Correlation
OverallQual	0.790982	Fireplaces	0.466929	PoolArea	0.092404
GrLivArea	0.708624	BsmtFinSF1	0.38642	MoSold	0.046432
GarageCars	0.640409	LotFrontage	0.351799	3SsnPorch	0.044584
GarageArea	0.623431	WoodDeckSF	0.324413	BsmtFinSF2	-0.011378
TotalBsmtSF	0.613581	2ndFlrSF	0.319334	BsmtHalfBath	-0.016844
1stFlrSF	0.605852	OpenPorchSF	0.315856	MiscVal	-0.02119
FullBath	0.560664	HalfBath	0.284108	LowQualFinSF	-0.025606
TotRmsAbvGrd	0.533723	LotArea	0.263843	YrSold	-0.028923
YearBuilt	0.522897	BsmtFullBath	0.227122	OverallCond	-0.077856
YearRemodAdd	0.507101	BsmtUnfSF	0.214479	MSSubClass	-0.084284
GarageYrBlt	0.486362	BedroomAbvGr	0.168213	EnclosedPorch	-0.128578
MasVnrArea	0.477493	ScreenPorch	0.111447	KitchenAbvGr	-0.135907

Table 7: Correlation of default features

Figure 10: ten most corralated features in train set heatmap

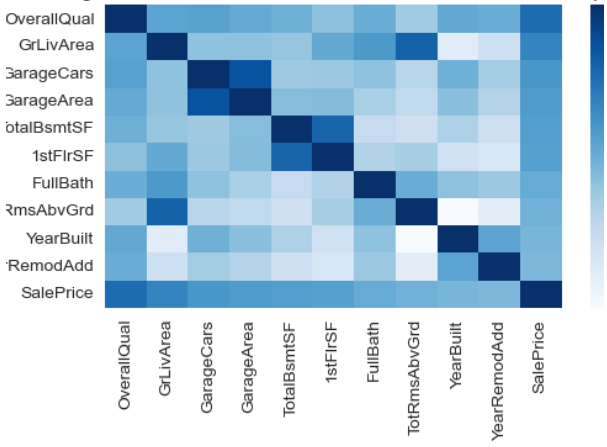
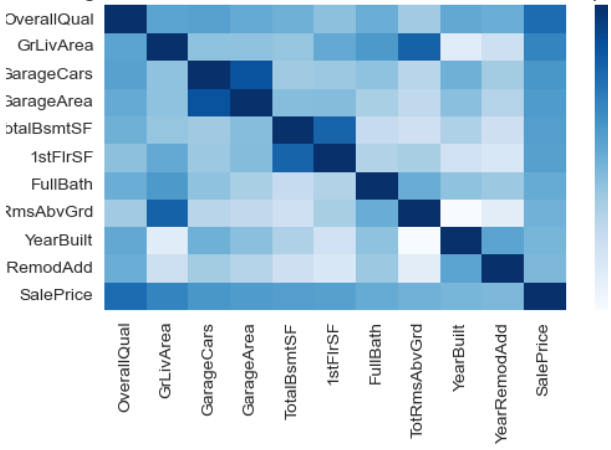


Figure 11: ten most corralated features in test set heatmap



2.2.5. Scatterplot Matrix

Figure 12 shows a scatter matrix plot that can be used to determine if there are any relationships between the various features in the data set. Also, it is possible to use the matrix to determine the density of each of these features. Due to the large number of features presented in the dataset, using the information in 2.2.4[50], the first ten most correlated features have been selected to be represented in this figure.

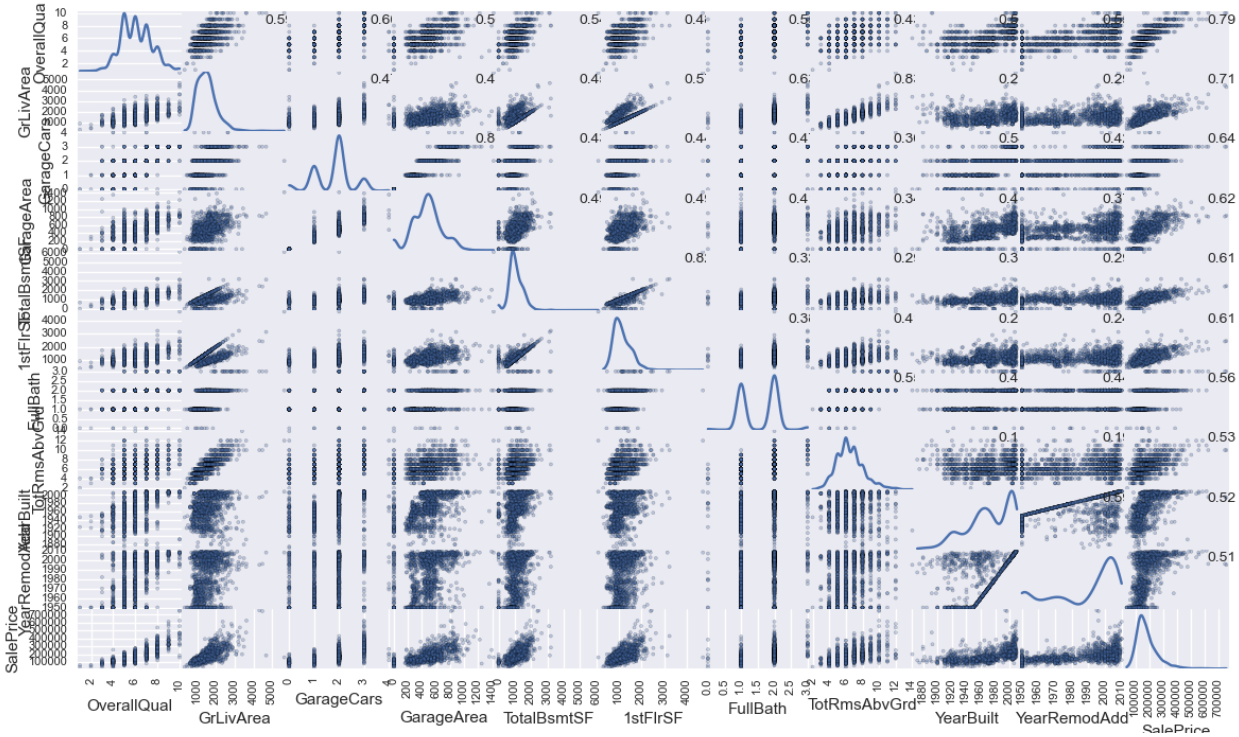


Figure 12: Scatterplot Matrix

[50] 2.2.4 Correlation - Default Features

2.2.6 Further Analysis of two continuous most correlated features

The first two continuous most correlated features according to 2.2.4.[51] are: 'GrLivArea' and 'GarageArea'.

Figure 13 and 14 shows the distribution plot, to get a sense for how the variables are distributed, and the a visualization of the linear relationship between 'GrLivArea' and ‘SalePrice’ Figure 15 and 16 shows the same information for features 'GarageArea' and ‘SalePrice’.

Figure 13: 'GrLivArea' Distribution plot

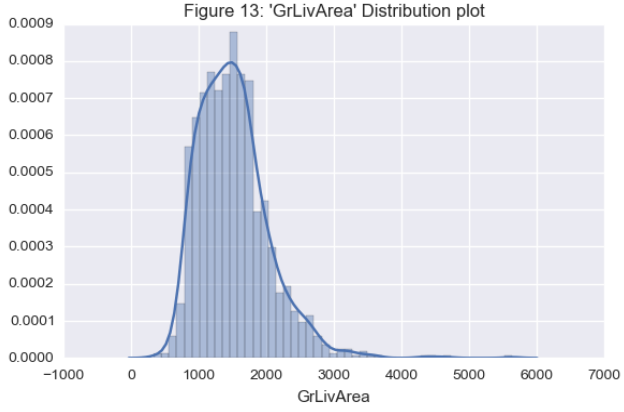


Figure 14: Linear Relationship 'GrLivArea' and 'SalePrice'

Figure 15: 'GarageArea' Distribution plot

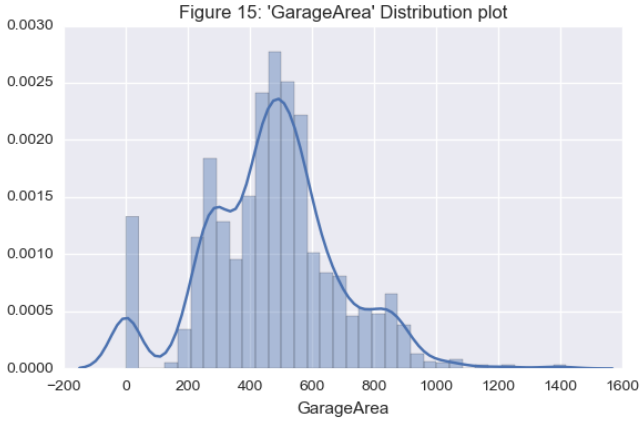


Figure 16: Linear Relationship 'GarageArea' and 'SalePrice'

[51] 2.2.4. Correlation - Default Features

2.3. Algorithms and Techniques

This is a supervised learning problem, and more specifically one of regression, there are many algorithms available for training a regressor and to make the corresponding predictions. As stated in 2.2.1[52] the train dataset has 1460 data points with 81 features each and the test dataset has 1459 data points with 80 features each. Taking this into account as well as the nature of the problem, and according to sklearn machine learning map[53] it is convenient to use SVR or some of the Ensemble Regressors such as: GradientBoostingRegressor, ExtraTreesRegressor, BaggingRegressor, AdaBoostRegressor and XGBRegressor. Some other regressors will also be tested: DecisionTreeRegressor, SVR, ElasticNet, Lasso, LassoLars and BayesianRidge. To tune these models the grid search technique[54] [55] [56] will be used. To perform this grid search, a number of parameters were selected for each algorithm.

2.3.1. DecisionTreeRegressor

Decision Trees (DTs) are a non-parametric supervised learning method used for classification and regression. Creates a model that predicts the value of a target variable by learning simple decision rules inferred from the data features.

2.3.1.1. Grid search parameters

1. splitter: The strategy used to choose the split at each node.
2. max_features: The number of features to consider when looking for the best split.
3. max_depth: The maximum depth of the tree.
4. min_samples_split: The minimum number of samples required to split an internal node.
5. min_samples_leaf: The minimum number of samples required to be at a leaf node.

2.3.2. SVR

Support vector machine (SVM) learns a hyperplane to classify data into 2 classes. The method of Support Vector Classification can be extended to solve regression problems. This method is called Support Vector Regression.

2.3.1.1. Grid search parameters

1. C: Penalty parameter C of the error term.
2. kernel: It specifies the epsilon-tube within which no penalty is associated in the training loss function with points predicted within a distance epsilon from the actual value.
3. max_iter: Hard limit on iterations within solver, or -1 for no limit.

---

[52] 2.1. Data Exploration
[53] http://scikit-learn.org/stable/tutorial/machine_learning_map/
[54] https://en.wikipedia.org/wiki/Hyperparameter_optimization
[55] http://scikit-learn.org/stable/modules/grid_search.html
[56] http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html

2.3.3. ElasticNet

"The elastic net is a regularized regression method that linearly combines the L1 and L2 penalties of the lasso and ridge methods."[57]

2.3.1.1. Grid search parameters

1. alpha: Constant that multiplies the penalty terms.
2. tol: The tolerance for the optimization.

2.3.4. Lasso

Is a regression analysis method that performs both variable selection and regularization in order to enhance the prediction accuracy and interpretability of the statistical model it produces.[58] Its trained with L1 prior as regularizer (aka the Lasso).

2.3.1.1. Grid search parameters

1. alpha: Constant that multiplies the L1 term.
2. tol: The tolerance for the optimization.
3. selection: If set to 'random', a random coefficient is updated every iteration rather than looping over features sequentially by default.

2.3.5. LassoLars[59]

Lasso model fit with Least Angle Regression a.k.a. Lars. It is a Linear Model trained with an L1 prior as regularizer.

2.3.1.1. Grid search parameters

1. alpha: Constant that multiplies the penalty term.
2. normalize: If True, the regressors X will be normalized before regression.

2.3.6. BayesianRidge

Bayesian regression techniques can be used to include regularization parameters in the estimation procedure: the regularization parameter is not set in a hard sense but tuned to the data at hand. Bayesian Ridge estimates a probabilistic model of the regression problem as described above. The prior for the parameter w is given by a spherical Gaussian. Optimize the regularization parameters lambda (precision of the weights) and alpha (precision of the noise).

---

[57] "Elastic net regularization - Wikipedia." https://en.wikipedia.org/wiki/Elastic_net_regularization. Accessed 18 Jan. 2017.
[58] Tibshirani, Robert. 1996. "Regression Shrinkage and Selection via the lasso". Journal of the Royal Statistical Society. Series B (methodological) 58 (1). Wiley: 267–88. http://www.jstor.org/stable/2346178
[59] "Least Angle Regression - Department of Statistics - Stanford University." http://statweb.stanford.edu/~imj/WEBLIST/2004/LarsAnnStat04.pdf. Accessed 18 Jan. 2017.

2.3.1.1. Grid search parameters

1. tol: Stop the algorithm if w has converged.
2. n_iter: Maximum number of iterations. Default is 300.
3. normalize: If True, the regressors X will be normalized before regression.

2.3.7. ExtraTreesRegressor

Implements a meta estimator that fits a number of randomized decision trees (a.k.a. extra-trees) on various sub-samples of the dataset and use averaging to improve the predictive accuracy and control over-fitting.

2.3.1.1. Grid search parameters

1. min_samples_split: The minimum number of samples required to split an internal node.
2. min_samples_leaf: The minimum number of samples required to be at a leaf node.

2.3.8. BaggingRegressor

Is an ensemble meta-estimator that fits base regressors each on random subsets of the original dataset and then aggregate their individual predictions (either by voting or by averaging) to form a final prediction.

2.3.1.1. Grid search parameters

1. n_estimators: The number of base estimators in the ensemble.

2.3.9. AdaBoostRegressor

“Boosting is an ensemble learning algorithm which takes multiple learning algorithms (e.g. decision trees) and combines them. The goal is to take an ensemble or group of weak learners[60] and combine them to create a single strong learner. AdaBoost takes a sample of the training dataset and tests to see how accurate each learner is. The end result is we find the best learner. Samples that are misclassified are given a heavier weight, so that they have a higher chance of being picked in the next round.”[61]

2.3.1.1. Grid search parameters

1. learning_rate: Learning rate shrinks the contribution of each regressor.
2. loss: The loss function to use when updating the weights after each boosting iteration.

---

[60] A weak learner classifies with accuracy barely above chance. A popular example of a weak learner is the decision stump which is a one-level decision tree.

[61] "Top 10 data mining algorithms in plain English | rayli.net." 2 May. 2015, http://rayli.net/blog/data/top-10-data-mining-algorithms-in-plain-english/. Accessed 18 Jan. 2017.

2.3.10. GradientBoostingRegressor

Builds an additive model in a forward stage-wise fashion; it allows for the optimization of arbitrary differentiable loss functions. In each stage a regression tree is fit on the negative gradient of the given loss function.

2.3.1.1. Grid search parameters

1. max_depth: The maximum depth of the tree.
2. criterion: The function to measure the quality of a split.
3. max_features: The number of features to consider when looking for the best split.
4. min_samples_split: The minimum number of samples required to split an internal node.
5. Min_samples_leaf: The minimum number of samples required to be at a leaf node.

2.3.11. XGBRegressor

XGBoost is short for “Extreme Gradient Boosting”[62]. XGBoost is a library designed and optimized for boosting trees algorithms. Through multi-threads and imposing regularization, XGBoost is able to utilize more computational power and get more accurate prediction.

2.3.1.1. Grid search parameters[63]

1. colsample_bytree: Subsample ratio of columns when constructing each tree.
2. gamma: Minimum loss reduction required to make a further partition on a leaf node of the tree.
3. learning_rate: Boosting learning rate.
4. max_depth: Maximum tree depth for base learners.
5. min_child_weight: Minimum sum of instance weight (hessian) needed in a child.
6. n_estimators: Number of boosted trees to fit.
7. reg_alpha: L1 regularization term on weights.
8. reg_lambda: L2 regularization term on weights
9. subsample: Subsample ratio of the training instance.

After selecting the best models, according to their performances, an ensemble generation[64] will be implemented. The performance metrics calculated and compared to the ones of the benchmark model[65], to determine if the target was reached.

---

[62] "Greedy Function Approximation: A Gradient Boosting Machine Author ...."
http://luthuli.cs.uiuc.edu/~daf/courses/optimization/papers/2699986.pdf. Accessed 18 Jan. 2017.
[63] "Python API Reference — xgboost 0.6 documentation."
http://xgboost.readthedocs.io/en/latest/python/python_api.html. Accessed 18 Jan. 2017.
[64] http://link.springer.com/chapter/10.1007%2F3-540-33019-4_19
[65] 2.4. Benchmark Model

2.4. Benchmark Model

The House Prices: Advanced Regression Techniques competition uses RMSLE[66] as evaluation metric. The public leaderboard for this competition presents the following stats for RMSLE values of the competing teams:

min	0.03839
25%	0.12401
50%	0.14071
75%	0.17778

Table 8: Public leaderboard statistics

Due to the limited resources of the computer in which this research is performed (Ubuntu 14.04.5 LTS[67] via crouton[68] on a Chromebook Acer C720-2802[69]), an RMSLE above the top 25% of the 3306 participating teams will be considered a success.
That would mean that the developed model should obtain, at least, an RMSLE value of approximately 0.12401.

3. Methodology

3.1. Data Preprocessing

As said before[70], there are numerous features, therefore an exhaustive feature engineering is required. A brief description of the process that took place in this section is detailed next.
As a first approach the first two and the first ten most correlated features have been selected for an initial implementation of the proposed method. In both of these new datasets all the missing values have been filled with zeros.
The definitely approach (working with all the relevant features) required some extensive work on the features. Outliers were detected in some features, for example 'GrLivArea' presents 4 outliers above 4000 square feets, these entries were removed. Entries with missing data have been replaced with some value: features of data type 'Object' with the most frequent value and features of other types with the mean. Some features have been entirely dropped since they would simply serve to complicate the results, for example 'SaleCondition' (the different types of sales), 'Street', 'Alley', etc. The feature 'SalePrice' have been logarithmically scaled and will be rescaled back in the predictions of the public test set before submitting to kaggle.

---

[66] 6. Evaluation Metrics
[67] http://releases.ubuntu.com/14.04/
[68] https://github.com/dnschneid/crouton
[69] http://laptops.specout.com/l/2780/C720-2802
[70] 2.1. Data Exploration

The ordinal features have been converted into labels with value between 0 and n_classes-1, through LabelEncoder[71].
To the nominal features an encoding have been applied (through One Hot Encoder[72] a group of n dummy features[73], Yes/No (1/0) features have been created).

3.2. Implementation

All the performed steps to carry this research are organized in the following five Ipython Notebooks[74]:

3.2.1. 01_EDA.ipynb

Contains all the tasks performed to get a better understanding of the data and to obtain the insights presented in sections 2.1. Data Exploration[75] and 2.2. Exploratory Visualization[76].

3.2.2. 02_features_engineering.ipynb

As stated in 3.1. Data Preprocessing[77] throughout this research we will be working on three sets of features:

- The first two most correlated features with all the missing values fill with zeros (**top_2**)
- The first ten most correlated features with all the missing values fill with zeros (**top_10**)
- All the engineered features (**all**)

In each of the three cases the dataset corresponding to: the **features (features)** of the train set, the logarithmically scaled **price (log_price)** of 'SalePrice' feature of the train set and the **public features (public_features)** used to make the predictions to be submitted to Kaggle are exported to a .pkl file using Pickle[78][79].

3.2.3. 03_default_regression_models.ipynb

The **features**.pkl, **log_price**.pkl and **public_features**.pkl files are loaded again using pickle. Then the train dataset is divided into a training and testing set, using 'train_test_split'[80] from

---

[71] "sklearn.preprocessing.LabelEncoder — scikit-learn 0.18.1 ...."
http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html. Accessed 19 Jan. 2017.
[72] http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.OneHotEncoder.html
[73] http://pandas.pydata.org/pandas-docs/version/0.18.1/generated/pandas.get_dummies.html
[74] "The Jupyter Notebook — IPython." https://ipython.org/notebook.html. Accessed 18 Jan. 2017.
[75] 2.1. Data Exploration
[76] 2.2. Exploratory Visualization
[77] 3.1. Data Preprocessing
[78] "Pickle - Wikipedia." https://en.wikipedia.org/wiki/Pickle. Accessed 18 Jan. 2017.
[79] "11.1. pickle — Python object serialization — Python 2.7.13 ...."
https://docs.python.org/2/library/pickle.html. Accessed 18 Jan. 2017.
[80] "sklearn.model_selection.train_test_split — scikit-learn 0.18.1 ...."
http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html. Accessed 18 Jan. 2017.

sklearn.cross_validation[81], to shuffle and split the features and gross data into the training and the testing sets.
Once the data set had been split up into the training and the testing sets, the regressors are trained. To accomplish this and to obtain the R^2 score and the RSMLE score of each of the regressors with their default parameters a for loop is implemented. This for loop uses three procedures: 'train_regressor', 'make_prediction' and 'train_predict'.
The trained regressors are then saved in a regs_dict.dict file using pickle.
The structure of this dictionary is as follows:

```
regs_dict = { 'top_2': {'untuned': {}, 'tuned': {} }, 'top_10': {'untuned': {}, 'tuned': {} },
'all': {'untuned': {}, 'tuned': {} } }
```

This dictionary is then loaded in the ensemble to make the ensemble predictions[82].

3.2.4. 04_grid_search.ipynb

The features.pkl, log_price.pkl and public_features.pkl files are loaded again using pickle.
Then the train dataset is divided into the training and the testing set, using 'train_test_split' from sklearn.cross_validation, to shuffle and split the features and gross data into the training and the testing sets.
Once the data set had been split up into the training and the testing sets. Several parameters for each of the regressors were fed to the Hyperparameter Optimization: to tune those models through the sklearn GridSearchCV. The grid search is performed using K-fold cross-validation (k = 4) on a regressor using each possible permutation of the input parameters. To do the grid search the procedure 'tune_parameters' is defined taking as input a tuple[83] of all regressors names and of all the parameters. In this procedure a for loop performs the grid search on each of the regressors and output the R^2 and the RMSLE of each of the tuned regressors, it also outputs a dictionary of these regressors ('tuned_regressors' = {}).
The tuned_regressors dictionary is then saved in the regs_dict.dict file using pickle.

3.2.5. 05_ensemble_generation.ipynb

The features.pkl, log_price.pkl and public_features.pkl files are loaded again using pickle.
Then the train dataset is divided into a the training and the testing sets. The dictionary of the models ('regs_dict.dict') is loaded using pickle. Afterward, the ensemble class is defined and the selected models from the dictionary loaded into the ensemble. Then the R^2 and the RMSLE of the ensemble calculated.
The final step consists in calculating the predictions on the public test set and export those results into a .csv file for the corresponding kaggle submission.

---

[81] "3.1. Cross-validation: evaluating estimator performance — scikit-learn ...." http://scikit-learn.org/stable/modules/cross_validation.html. Accessed 18 Jan. 2017.
[82] 3.2.5. 05_ensemble_generation.ipynb
[83] "Python Tuples - Tutorialspoint." https://www.tutorialspoint.com/python/python_tuples.htm. Accessed 18 Jan. 2017.

A first approach to this problem consisted in testing the regressors with their default parameters for both the first two and for the first ten most correlated features. Then the ensemble of these parameters and sets were calculated. Meaning that as a first attempt to make the predictions only 3.2.1. 01_EDA.ipynb, 3.2.3. 03_default_regression_models.ipynb and 3.2.5. 05_ensemble_generation.ipynb were implemented.

The primary complication that occurred during the implementation phase of this section was related to finding the best way to save the regressors and their respective parameters both untuned and tuned for further experimentation. The best solution found was to create the regressors dictionary using pickle.

3.3. Refinement

As a refinement, two things were implemented:

- Grid search
- Feature engineering

3.3.1. Grid search

In the case of the grid search, to perform the Hyperparameter Optimization and tune the models through the sklearn GridSearchCV the following parameters were used, as shown in Table 9:

Regressor	Parameter	Values
DecisionTreeRegressor	splitter	['best', 'random']
	max_features	['auto', 'sqrt', 'log2']
	max_depth_range	np.arange(1, 11)
	min_samples_split	np.arange(2, 9)
	min_samples_leaf	np.arange(1, 9)
SVR	C	[0.9, 1.0, 1.1]
	kernel	['linear', 'rbf']
	max_iter	[1500]
ElasticNet	alpha	[1.0, 1.1]
	tol	[0.00005, 0.0001, 0.00015, 0.0002]
Lasso	alpha	[0.01, 1, 10]
	tol	[0.00005, 0.0001, 0.00015, 0.0002]
	selection	['cyclic', 'random']
BayesianRidge	tol	[0.0001, 0.001, 0.01, 0.1, 1, 10, 50, 100, 500]
	n_iter	[1000]
	normalize	['True', 'False']
GradientBoostingRegressor	min_samples_split	np.arange(2, 6)
	min_samples_leaf	np.arange(1, 6)
BaggingRegressor	n_estimators	np.arange(50, 500, 50)

AdaBoostRegressor	learning_rate	[0.001, 0.01, 1.0]
	loss	['linear', 'square', 'exponential']
XGBRegressor	colsample_bytree	[0.9, 1.0]
	gamma	[0.0, 0.1]
	learning_rate	[0.1, 0.5]
	max_depth	np.arange(3, 5)
	min_child_weight	np.arange(1, 4)
	n_estimators	np.linspace(100, 10000, 100)
	reg_alpha	[0.0, 0.05, 0.1, 0.2, 0.3, 0.4]
	reg_lambda	[0.6, 0.7, 0.8, 0.9, 0.95, 1.0]
	subsample	[0.6, 0.7, 0.8, 0.9, 0.95, 1.0]

Table 9: Parameters values for GridSearchCV

The steps performed to run the grid search can be seen in 3.2.4. 04_grid_search.ipynb[84], once the tuned regressors are in the 'regs_dict.dict', the metrics of the ensemble of these tuned regressors can be calculated.[85]
The total time in performing the grid search can be seen in Figure 17, Figure 18 and Figure 19 in 4.1. Model Evaluation and Validation[86].

Due to the fact that GridSearcCV is, in some cases, extremely resourceful intensive and also due to the limited resources in the environment in which this research have been performed, some of the parameters used to feed the grid search had to be subsequently removed.

3.3.2. Feature engineering

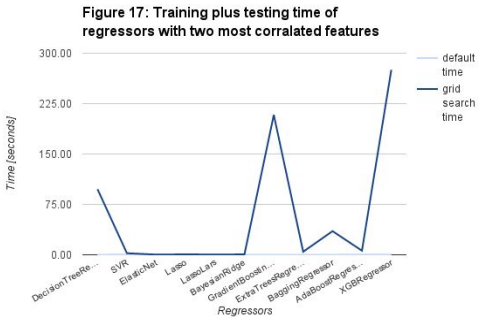
In order to perform the tasks described in 3.1. Data Preprocessing[87]. The following procedures were defined: "DataFrameImputer" to fill the missing values, 'label_encoding' to convert the ordinal features into labels with value between 0 and n_classes-1 and 'one_hot' to convert nominal variables into dummy variables. Once the dataset with all the engineered features is developed, the R^2 and the RMSLE of the default and tuned regressors were calculated, as well as these metrics for the ensemble. This was done following the steps discussed in 3.2. Implementation[88]. And the effect of the feature engineering in the RSMLE scores can be seen in 4.1. Model Evaluation and Validation[89].
The complication arisen here have to do with the vast number of variables that made this task extremely difficult.

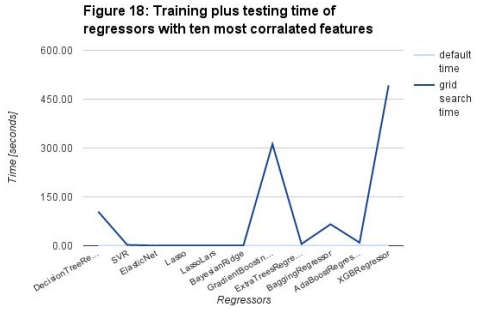
[84] 3.2.4. 04_grid_search.ipynb
[85] 3.2.5. 05_ensemble_generation.ipynb
[86] 4.1. Model Evaluation and Validation
[87] 3.1. Data Preprocessing
[88] 3.2. Implementation
[89] 4.1. Model Evaluation and Validation

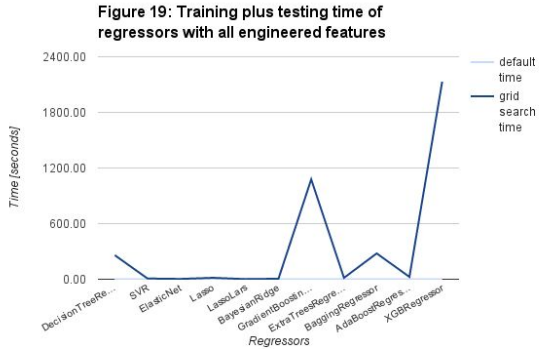
4. Results

4.1. Model Evaluation and Validation

In Figure 17, Figure 18 and Figure 19 the combined training and testing times, for each of the regressors in each of the selected features, can be seen. Clearly, performing grid search on the regressors is really slow compared to train the regressors with their default parameters. Also, it is important to notice that as the number of features increases the training and testing times also increases. Of all the regressors the three slowest in performing the grid search are: XGBRegressor, GradientBoostingRegressor and BaggingRegressor, this has to do with the fact these are the ones that have the large grid for parameters tuning.







In Figures 20, 21 and 22, the RSMLE score can be seen for each of the regressors. The better performing regressors are:

- Two most correlated features:

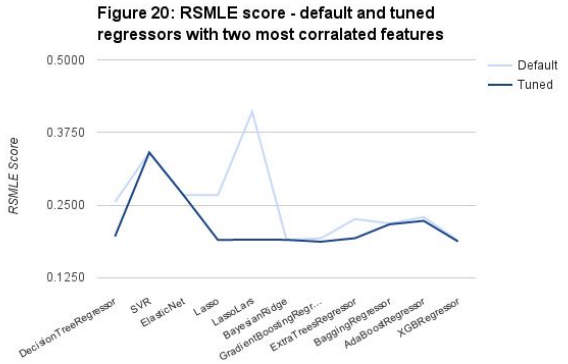
Lasso (tuned), LassoLars (tuned), BayesianRidge (tuned or untuned), GradienTreeBoosting (tuned) and XGBRegressor (tuned or untuned).

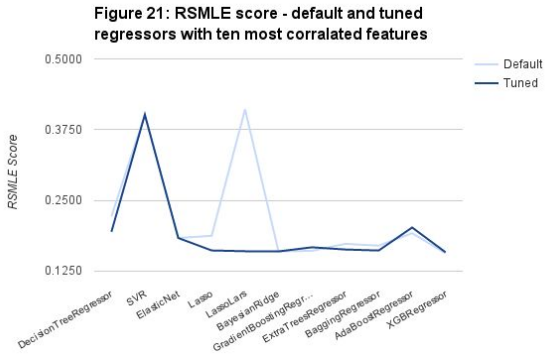
- Ten most correlated features:

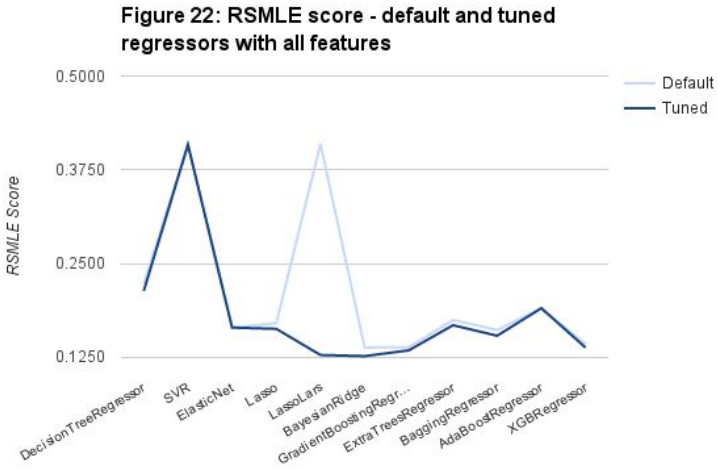
In the case of the ten most correlated features the better regressors are: Lasso (tuned), LassoLars (tuned), BayesianRidge (tuned or untuned), GradientBoostingRegressor (tuned), ExtraTreeRegressor (tuned), BaggingRegressor (tuned) and XGBRegressor (tuned or untuned).

- All engineered features:

ElasticNet (tuned or untuned), Lasso (tuned or untuned), LassoLars (tuned), BayesianRidge (tuned), GradientBoostingRegressor (tuned), ExtraTreeRegressor (tuned), BaggingRegressor (tuned) and XGBRegressor (tuned or untuned).







In Figures 23 and 24, the RSMLE score and the R^2 score of the ensemble, respectively, can be seen for each of the selected features. It is also shown here these metrics for the public test set submitted to kaggle. In Figure 23 can be seen that the ensemble performed better with all the engineered features and with the regressors tuned through the grid search technique. Also a selection of the best models was tested, these models are: BayesianRidge (tuned) and GradientBoostingRegressor (tuned). The ensemble with these selected models performed really well with an RSMLE of 0.12994 on the public test set.

In figure 24 can be seen how the ensemble is steadily improving in explaining the variance of the predicted variable.

Figure 23: RSMLE score of ensemble

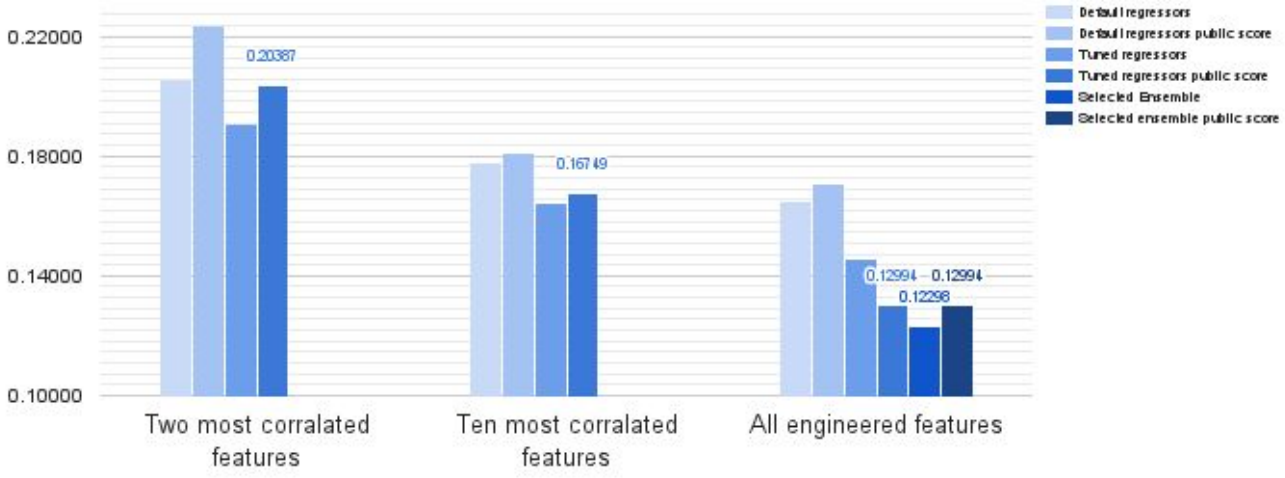
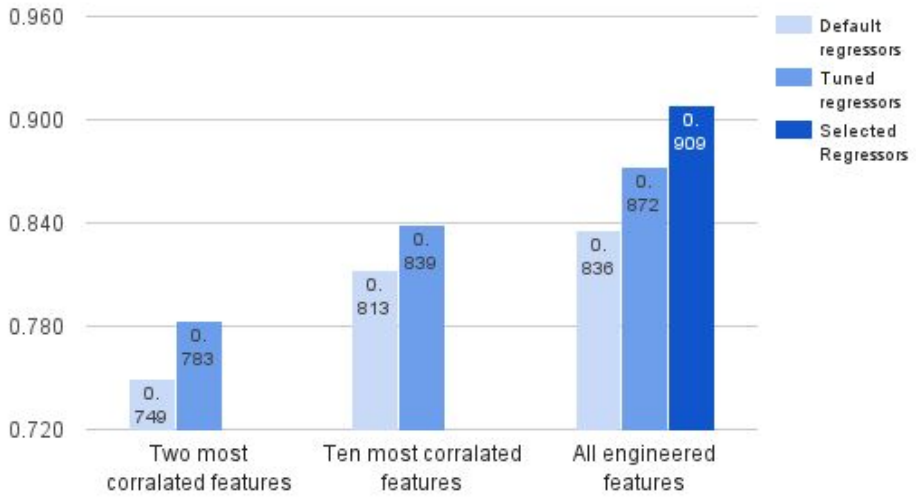


Figure 24: R2 score of ensemble



4.2. Justification

As stated in 2.4. Benchmark Model[90] the RMSLE score proposed to consider the ensemble a success was 0.12401. And as shown in Figure 23[91] the final RSMLE score obtained with the

[90] 2.4. Benchmark Model

[91] 4.1. Model Evaluation and Validation

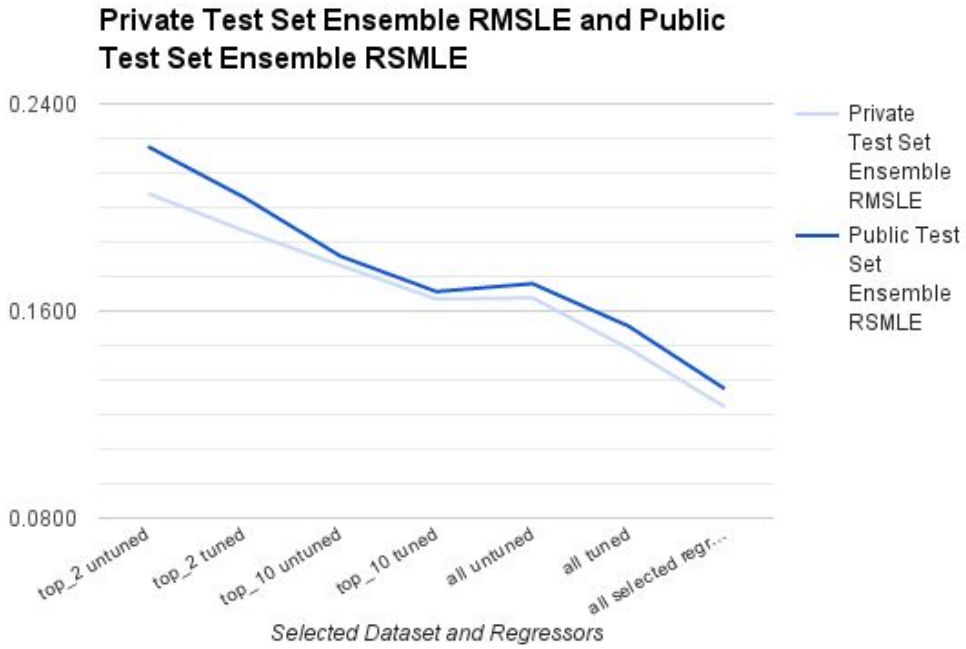
ensemble of BayesianRidge (tuned) and GradientBoostingRegressor (tuned) is 0.1299 which is really close proposed benchmark model. For that reasons the proposed solution for this problem can be considered successful. Besides the public leaderboard is calculated on approximately 50% of the test data. Therefore the final results will be based on the other 50%, so the final standings may be different and there is always the risk of overfitting on the mentioned first half of the public test set.
It is important to notice here however, that this final result was only obtain after performing feature engineering and after tuning the parameters, that is the ensemble on its own would not have resolved this problem.

5. Conclusion

5.1. Free-From Visualization

5.1.1. RSMLE score - private test set versus public test set

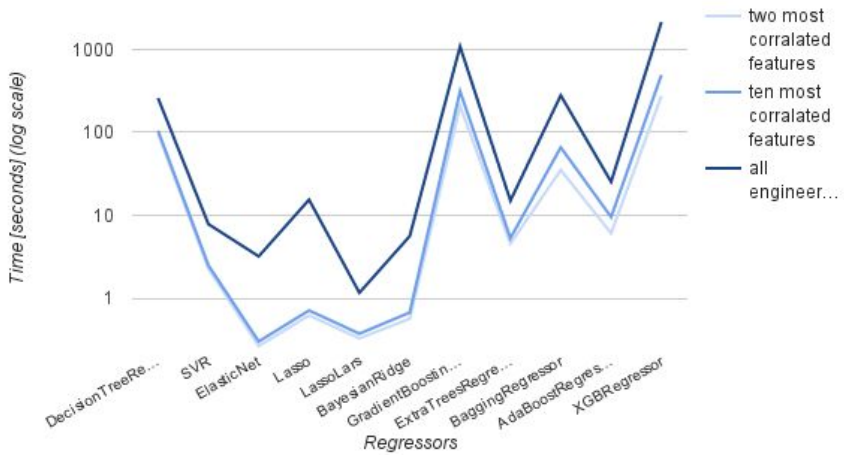
It can be observed in Figure 25 that the RMSLE obtained by the ensemble in the public test set is always higher than the one obtained in the private test set, meaning there is some overfitting.



5.1.2. Comparison Times

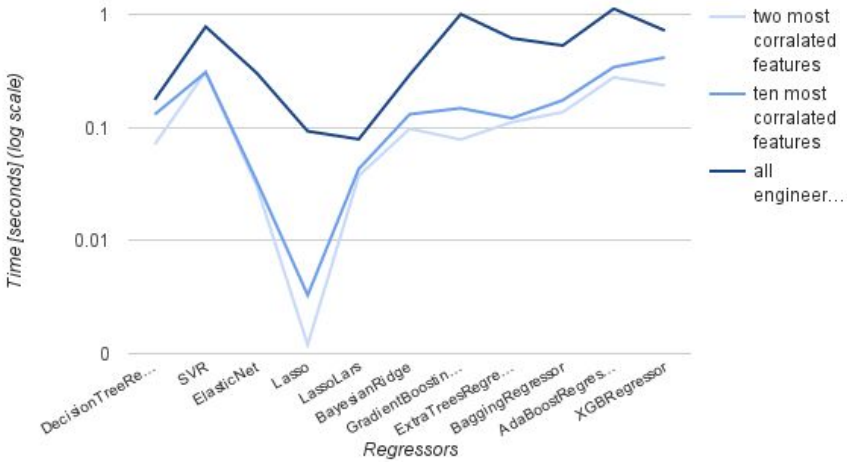
In Figure 26 can be observed that there are some regressors that are more sensitive to a grid search for hyper parameter optimization as the number of features increases. For example: Lasso increases its running time considerably when performing a grid search with all the engineered features compared to when it uses the other two sets of selected features.

Figure 26: Grid search time



In figure 27 the relative increase of training and testing time of the regressors can be observed as the numbers of features is increased. It is important to notice here that ElasticNet and GradientBoostingRegressor increases considerably their execution times as more features are added.

Figure 27: Training and testing time



5.2. Reflection

In this project we have used the following libraries NumPy[92], SciPy[93], matplotlib[94], Seaborn[95], pandas[96], sklearn[97], xgboost[98] and pickle[99].

[92] http://www.numpy.org/
[93] https://www.scipy.org/
[94] http://matplotlib.org/
[95] http://seaborn.pydata.org/
[96] http://pandas.pydata.org/
[97] http://scikit-learn.org/stable/

To predict the housing prices in the Kaggle version of the Ames dataset, first an Exploratory Data Analysis was performed to get a better understanding of the data. Then as a first approach the first two and the first ten most correlated features have been selected for an initial implementation of the proposed method. In both of these new datasets all the missing values have been filled with zeros. Then the steps described in 3.2. Implementation[100] and the RSMLE scores of the two mentioned sets were calculated for all the regressors with their default parameters, their tuned parameters and for the ensemble of those.

The definitely approach (working with all the relevant features) required some extensive work on the features, as it is detailed in 3.1. Data Preprocessing[101]. Outliers were detected in some features, for example 'GrLivArea' presents 4 outliers above 4000 square feets, these entries were removed. Entries with missing data have been replaced with some value: features of data type 'Object' with the most frequent value and features of other types with the mean. Some features have been entirely dropped since they would simply serve to complicate the results, for example 'SaleCondition' (the different types of sales), 'Street', 'Alley', etc. The feature 'SalePrice' have been logarithmically scaled and is then rescaled back in the predictions of the public test set before submitting to kaggle. The ordinal features have been converted into labels with value between 0 and n_classes-1, through LabelEncoder. To the nominal features an encoding have been applied (through One Hot Encoder a group of n dummy features, Yes/No (1/0) features have been created).

Before performing each of the training and prediction iterations the training dataset was divided into two parts, using train_test_split from sklearn.cross_validation, to shuffle and split the features and 'SalePrice' data into a training and a testing set.
In all the cases the feature 'SalePrice' have been logarithmically scaled and rescaled back in the predictions of the public test set before submitting to kaggle.

A dictionary with all the parameters in each of the six analysis cases[102] was created to deliver a more flexible way to test and try different combinations of regressors in the ensemble and to learn how the results were affected by the presence of this different regressors. Once the best models were selected, according to their performances, an ensemble averaging of those regressors was implemented. The performance metrics calculated and compared with the ones of the benchmark model, determining that the target was reached and the proposed approach to solve this problem was a success. The selected models were BayesianRidge (tuned) and GradientBoostingRegressor (tuned) and the final RSMLE score was 0.1299 just a little higher than the one of the benchmark model. Meaning that the proposed solution was a success.

---

[98] https://github.com/dmlc/xgboost
[99] https://docs.python.org/2/library/pickle.html
[100] 3.2. Implementation
[101] 3.1. Data Preprocessing
[102] 3.2.3. 03_default_regression_models.ipynb

As a final reflection, it is important to comment that working with algorithms or techniques resourceful intensive is extremely difficult in the current environment.

5.3. Improvement

One aspect of the proposed solution to this problem that can be definitely improved is the GridSearchCV performed to tune the parameters. Due to the limited resources of the computer in which this research was performed, a more exhaustive grid search was impossible to implement. The run times for GridSearchCV increase extremely quickly when adding just a few parameters to the search. It would be likely that the RSMLE score of some regressor that obtained really good RMSLE scores but did not improved much after the gridsearch could obtain much lower RSMLE scores if a more exhaustive grid search is performed. For example, in Figure 22[103] can be seen that except for LassoLars that made a considerable improvement after the gridsearch, the other regressors did not improved much. And ElasticNet, Lasso, BayesianRidge, GradientBoostingRegressor and XGBRegressor obtain really low RSMLE scores but except for GradientBoostingRegressor and XGBRegressor the grid search performed in the other mentioned regressors was not so exhaustive as it could be. Improving the RSMLE scores of the constituting regressor would mean an improve in the overall performance of the ensemble.

Another approach to this problem could be the implementation of a deep learning algorithm, like a Neural Network, unfortunately due to the limited resources available the implementation and testing of such solution will be certainly unfeasible.

[103] 4.1. Model Evaluation and Validation