Machine Learning Engineer Nanodegree

Capstone Project

Arbaaz Muslim

January 14, 2018

I. Definition

Project Overview

The problem this project addresses relates to the field of education. It involves taking student information and using it to determine the student's final math grade. Therefore, the project performs a regression task. The data I plan on using was featured in a report authored by Associate Professor Paulo Cortez at the University of Minho, who provided the same data for the UCI Machine Learning Repository (BRITO, A. ; TEIXEIRA, J., eds. lit. – "Proceedings of 5th Annual Future Business Technology Conference, Porto, 2008". [S.l. : EUROSIS, 2008]. ISBN 978-9077381-39-7. p. 5-12.). The student information provides input features and the student's final math grade is the target feature.

A model that accurately performs such a regression task can provide great insight to the factors that impact academic success. Once a relatively accurate model is trained, its feature weights can provide a look at the relative effects of different factors on academic success.

Problem Statement

In one sentence, the problem can be phrased as follows: given information about a student, how can one determine the student's final math grade? The general solution to this problem is a supervised learner. More specifically, finding a solution to this problem involves the following steps:

- Separating the features (student information) from the target (final math grades)
- Removing irrelevant features

- One-hot encoding categorical features, or translating non-numerical features into arrays of numbers
- Normalizing continuous features (i.e. applying mathematical functions to the features in order to distribute them in a way similar to a normal distribution)
- Scaling all non-categorical features (i.e. discrete and continuous features)
- Removing Tukey outliers from the continuous data features
- Shuffling and splitting the data into training and testing sets through an 80/20 split
- Further dividing the training set using k-fold cross-validation
- Training and evaluating many unoptimized regression models on the divided training data, starting with a benchmark linear regressor
- Taking the unoptimized model with the lowest error and tuning its hyperparameters via gridsearch

Metrics

The foremost evaluation metric that is most relevant to this problem is the mean squared error. This metric is commonly used for regression problems and provides a single number that illustrates the accuracy of predictions in comparison to actual values. The formula for mean squared error is $\frac{1}{n} \sum_{i=1}^{n} (\hat{X}_i - X_i)^2$, where n is the number of data points, X_i is an actual value, and X hat is the predicted value. For each input value, the formula involves finding the difference between the predicted and actual value and then squaring it to ensure that the result is positive. These error squares are then averaged. In the context of this project, this metric provides a number that illustrates how far predicted grades are from the actual grades of students.

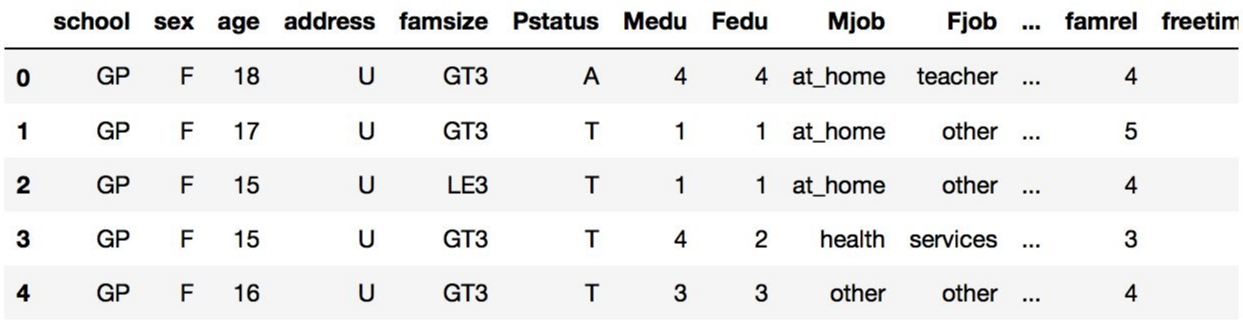
Another evaluation metric relevant to this problem is the R^2 value, or correlation coefficient. This value illustrates the proportion of variance in an output variable that can be explained by an input variable. The formula for the R^2 value is $R^2 \equiv 1 - \frac{SS_{\text{res}}}{SS_{\text{tot}}}.$, where $SS_{\text{res}} = \sum_i (y_i - f_i)^2$ and $SS_{\text{tot}} = \sum_i (y_i - \bar{y})^2,$. In these formulas, f_i is the same as y-hat (the prediction of the output value) and y-bar is the average of the actual output values. In the context of this project, this metric shows how much of the variance in the students' actual grades is explained by the chosen model. This metric will serve as a sanity check in this project. As error decreases, the correlation coefficient should increase, thereby confirming the quality of the model.

II. Analysis

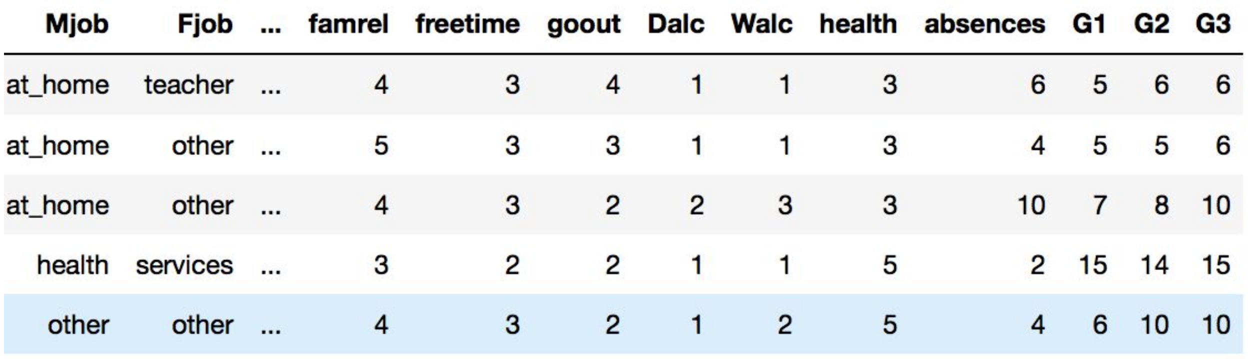
Data Exploration

The publicly available dataset used for this project is the Student Performance Data Set from the UCI Machine Learning Repository. The data comes from Associate Professor Paulo Cortez at the University of Minho. The data comes from students in secondary education in two Portuguese schools and was collected through surveys and questionnaires. There are 395 data points available. Since this dataset is very small, outlier removal should be minimal in order to prevent information loss. For this reason, the top five outliers from the continuous features are removed. The data is then split into a training test (312 points) and a testing set (78 points) after removing the top five outliers. The training set is then further divided into 10 folds in order to facilitate k-fold cross validation.

Each data point has 33 attributes. These attributes include both student data as well as student grades, making the data both relevant and useful for a solution to the problem. The attribute information can be found at https://archive.ics.uci.edu/ml/datasets/Student+Performance. Here is a data sample prior to any preprocessing:



	school	sex	age	address	famsize	Pstatus	Medu	Fedu	Mjob	Fjob	...	famrel	freetim
0	GP	F	18	U	GT3	A	4	4	at_home	teacher	...	4	
1	GP	F	17	U	GT3	T	1	1	at_home	other	...	5	
2	GP	F	15	U	LE3	T	1	1	at_home	other	...	4	
3	GP	F	15	U	GT3	T	4	2	health	services	...	3	
4	GP	F	16	U	GT3	T	3	3	other	other	...	4	

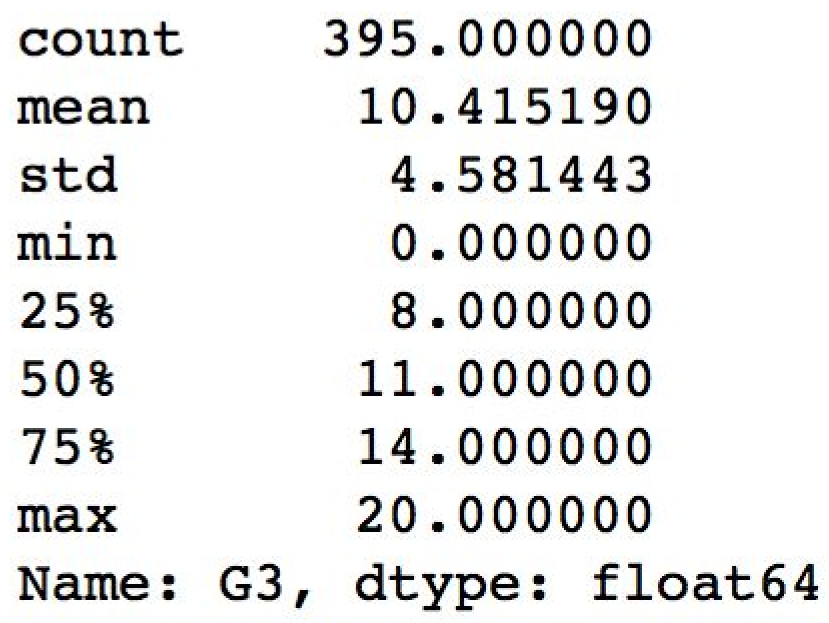


Mjob	Fjob	...	famrel	freetime	goout	Dalc	Walc	health	absences	G1	G2	G3
at_home	teacher	...	4	3	4	1	1	3	6	5	6	6
at_home	other	...	5	3	3	1	1	3	4	5	5	6
at_home	other	...	4	3	2	2	3	3	10	7	8	10
health	services	...	3	2	2	1	1	5	2	15	14	15
other	other	...	4	3	2	1	2	5	4	6	10	10

30 of the 33 features are used as input (excluding the three features discussed below).

Three of the attributes present are G1, G2, and G3. G1 refers to the grade for the first grading period, G2 refers to the grade for the second grading period, and so on. In this project, G3, the final grade, is the target variable. G1 and G2 will be disregarded since “G3 has a strong correlation with attributes G2 and G1” and “it is more difficult to predict G3 without G2 and G1, but such prediction is much more useful”, according to the site.

A description of G3 is provided below, followed by a summary of all features and their types.



```
count    395.000000
mean      10.415190
std        4.581443
min        0.000000
25%        8.000000
50%       11.000000
75%       14.000000
max       20.000000
Name: G3, dtype: float64
```

Variable	**Type**
School	Categorical
Sex	Categorical
Age	Continuous
Address	Categorical
Family Size	Categorical
Parent’s cohabitation status	Categorical
Mother’s education	Discrete

Father’s education	Discrete
Mother’s job	Categorical
Father’s job	Categorical
Reason to choose school	Categorical
Guardian	Categorical
Home to school travel time	Discrete
Weekly study time	Discrete
Class failures	Discrete
Educational support	Categorical
Family educational support	Categorical
Extracurricular activities	Categorical
Attended nursery school	Categorical
Wants to take higher education	Categorical
Internet access at home	Categorical
Family relationship qualities	Discrete
Free time	Discrete
Going out with friends frequency	Discrete
Workday alcohol consumption	Discrete
Weekend alcohol consumption	Discrete
Health status	Discrete
Number of absences	Continuous

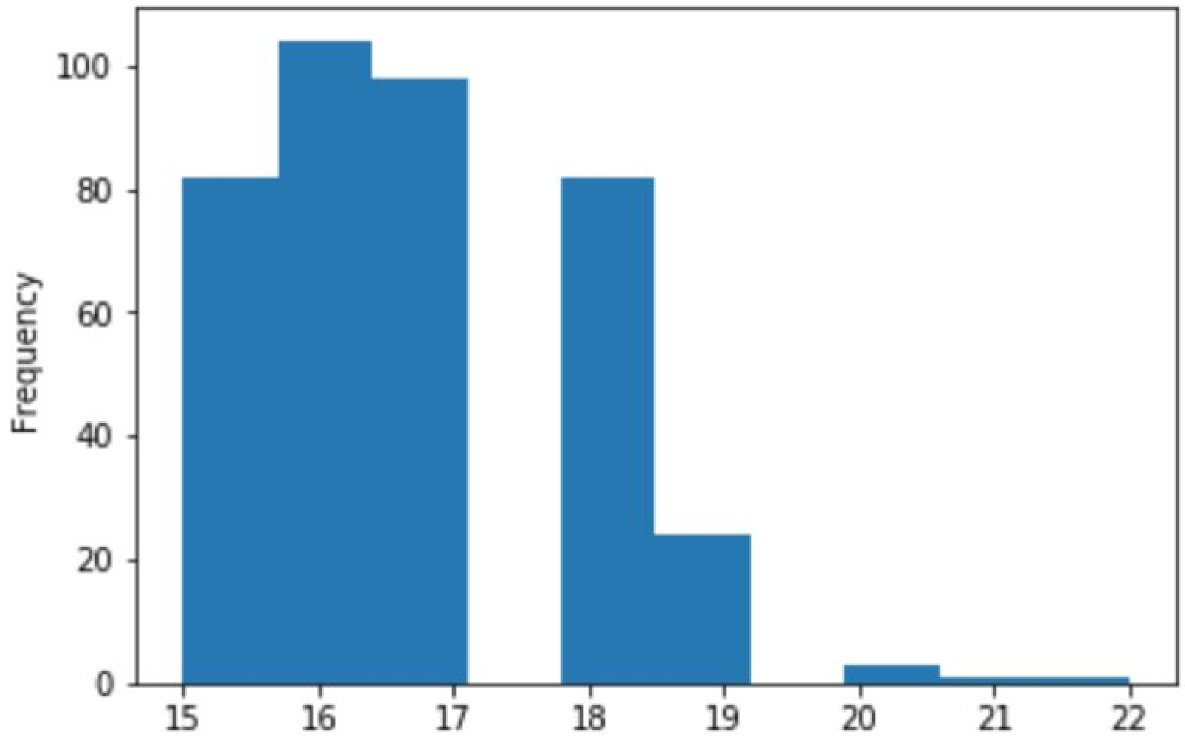
In the above table, categorical features need to be one-hot encoded. Discrete features are numerical but limited to a range, while continuous features are numerical and are not limited to a range. Continuous features, of which there are only two, need to be normalized and then scaled. Discrete features only need to be scaled.

Before After

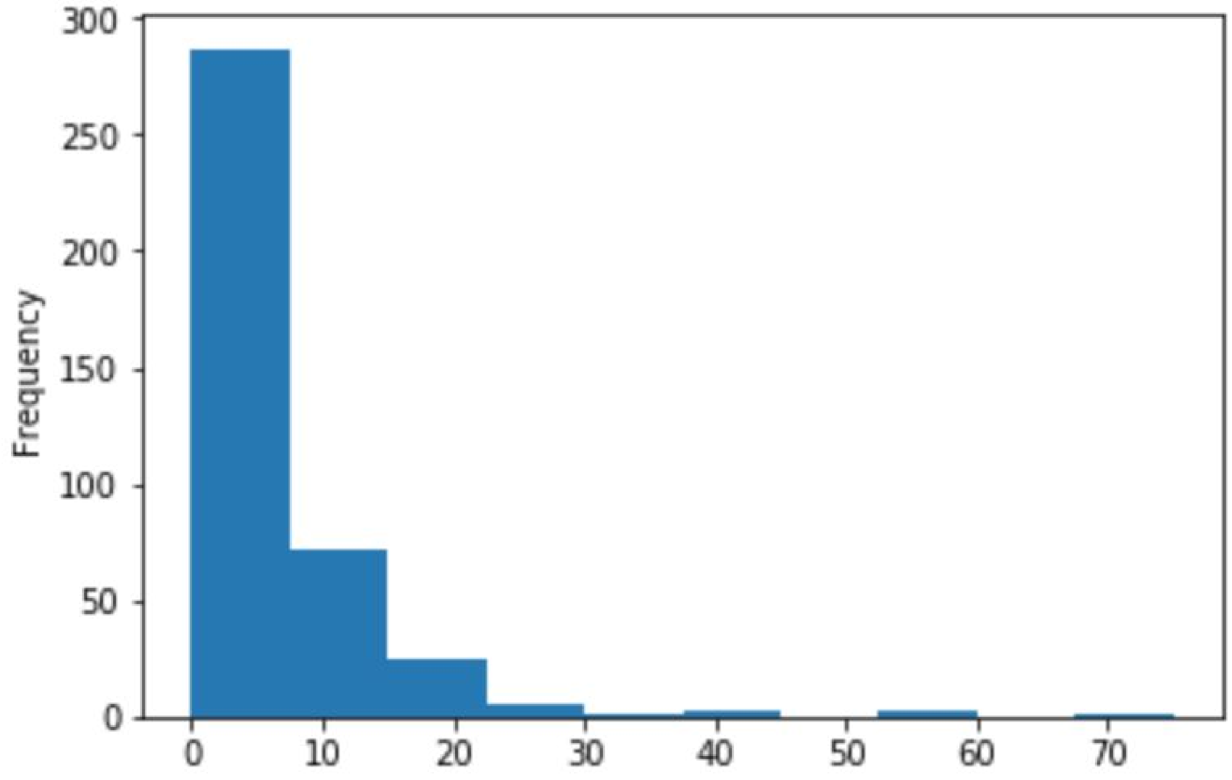
Exploratory Visualization

One of the tasks involved in this project was normalizing the continuous features. The following visuals illustrate the contrast before and after Box Cox and log normalization. The visuals begin with histograms of the two continuous features prior to normalization. These are followed by a histogram of the target feature. Finally, density plots are provided for the two continuous features before and after normalization.

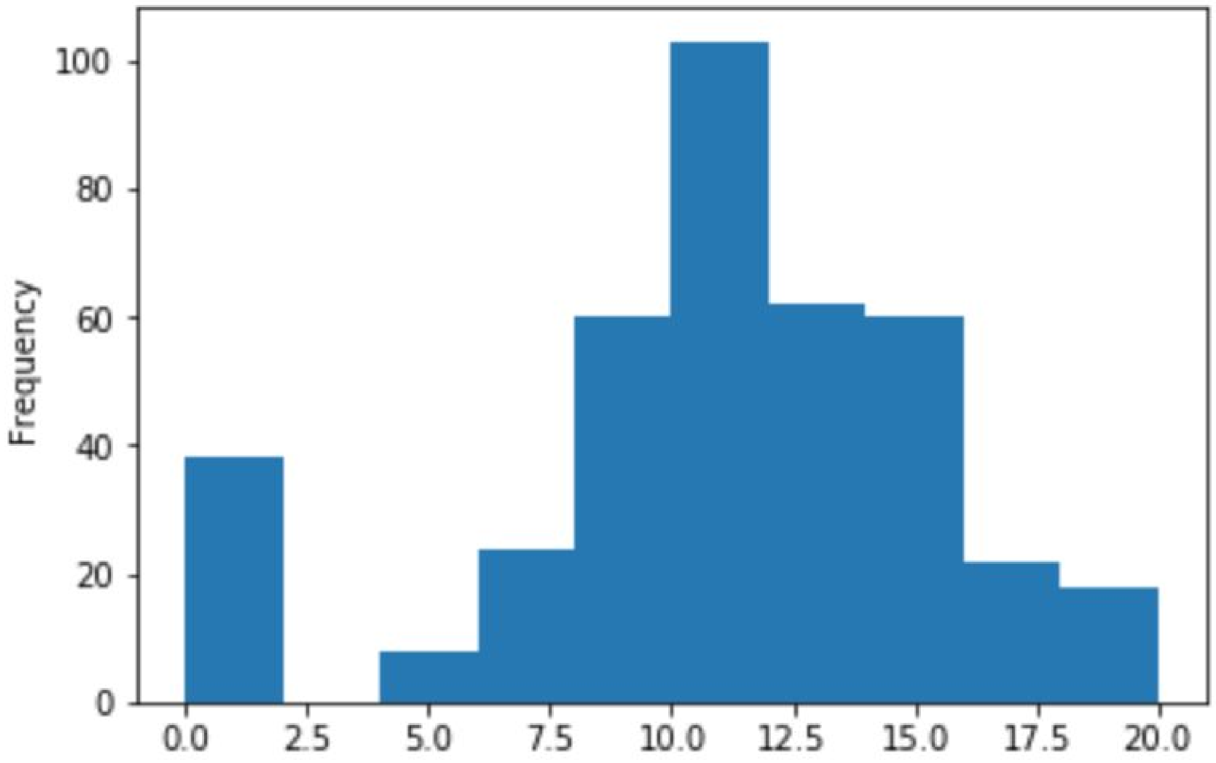
Age Histogram



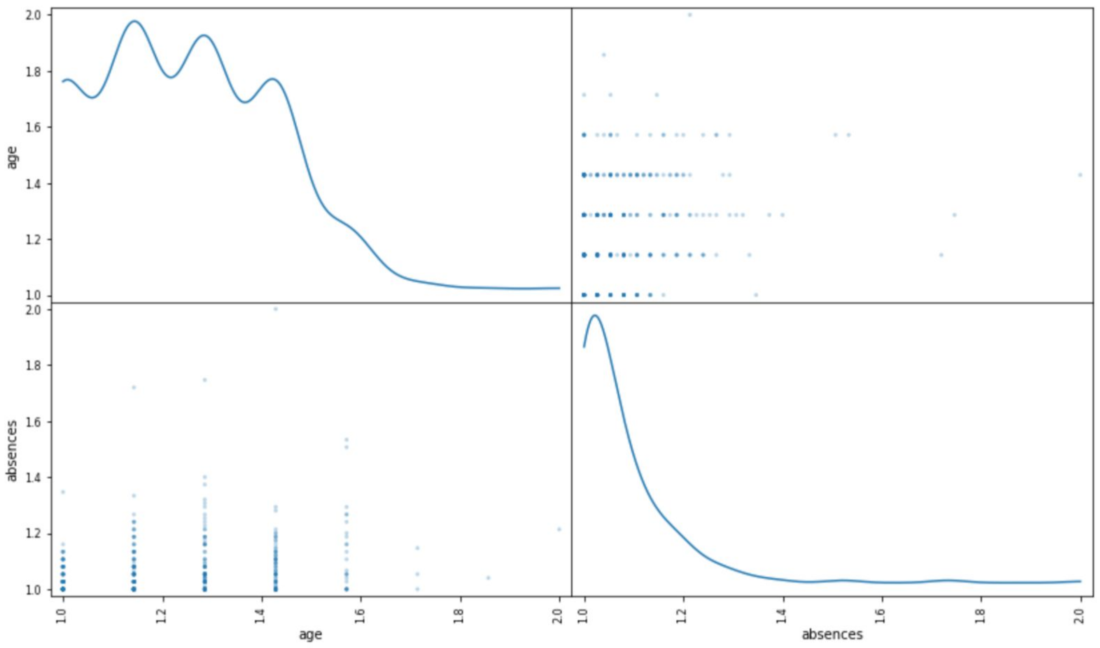
Absences Histogram



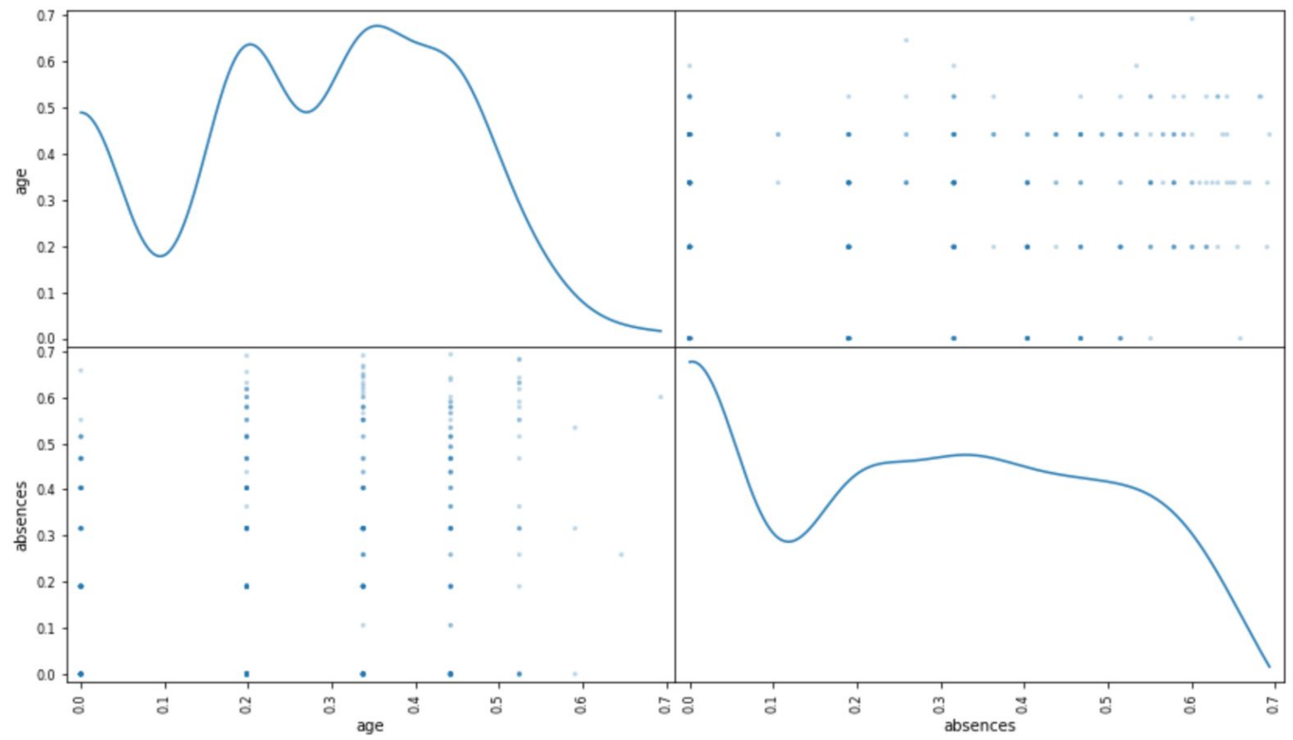
Final Math Grade Histogram



Continuous Features Pre-normalization

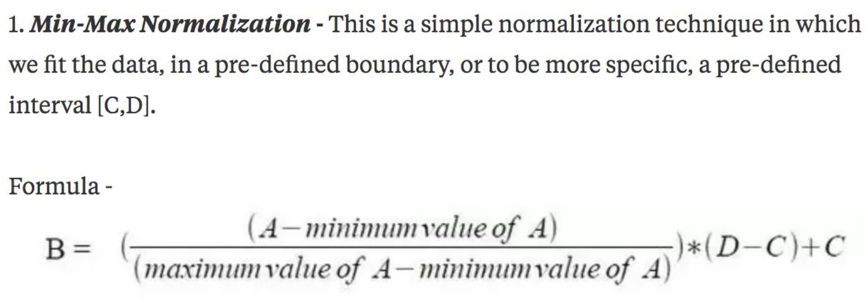


Continuous Features Post-normalization



Algorithms and Techniques

One of the first techniques used in this project was a custom defined function that applied the min-max scaling formula to continuous features.



1. ***Min-Max Normalization*** - This is a simple normalization technique in which we fit the data, in a pre-defined boundary, or to be more specific, a pre-defined interval [C,D].

Formula -

$$B = \left(\frac{(A - minimum\,value\,of\;A)}{(maximum\,value\,of\;A - minimum\,value\,of\;A)}\right) * (D - C) + C$$

The function fit the data to a range of [1,2]. This made it possible to apply a log transformation and the Box Cox function to the data in order to give it a more normal distribution.

A second technique used was the use of a log transformation in conjunction with the Box Cox function in order to fit the continuous features to a normal or Gaussian distribution. The log transformation returned the natural logarithm of all data points. The Box Cox transformation applied the formula y = (x**lmbda - 1) / lmbda to all data points. The value of lambda was found by the function and was chosen in a way that "maximize[d] the log-likelihood function", according to scipy documentation.

In order to determine the best way to fit the continuous features to a normal distribution, the normaltest function from scipy was used. Different transformations were applied to the data and the resulting p-values were examined. The p-values, which were returned by the normaltest function, showed the probability that data did not come from a normal distribution. Therefore, the objective was to minimize the p-value of inputted data. This objective was achieved when the data underwent a log transformation followed by a Box Cox transformation.

In order to further preprocess the data, outliers were also removed. Since the dataset was quite small to begin with (395 data points), a minimal number of outliers were removed in order to prevent major information loss. The method used to remove outliers was the Tukey method, which considers all points that are more than 1.5 interquartile ranges from the first or third quartile as outliers. This is known as the Tukey method.

Finally, a number of regressors were tested on the data in order to determine which regressor's performance merited further tuning. The models tested were: linear regressor, logistic regressor, stochastic gradient descent regressor, Gaussian process regressor, support vector regressor, gradient boosting regressor, decision tree regressor, multilayer perceptron, extra tree regressor, random forest regressor, extra trees regressor, bagging regressor, and Adaboost regressor. These algorithms were all imported from sklearn libraries.

A linear regressor works by determining a line that minimizes the squares of the distances between the line and the data points. The points on this line are then used to make predictions based off of given input values.

A random forest is a model that begins by taking random samples of the training data and then fitting a decision tree to each sample. The model is complete once this "forest" has been assembled. When a value is inputted, it is passed through each decision tree. The prediction from each individual tree is taken into account and an average of all the predictions is outputted by the entire model.

A bagging regressor works much in the same way as a random forest, but does not remain bound to a decision tree as its individual unit. Instead, a bagging regressor may use any kind of estimator, such as a linear or logistic estimator. However, a bagging regressor's default argument for the unit estimator is a decision tree.

Benchmark

The benchmark used in this project was a simple linear regressor. This benchmark was chosen because of its relative simplicity in comparison to the other methods tested. It produced a mean squared error of 31.1161358173.

A second benchmark used was from the previously referenced paper published by Paulo Cortez. This paper used root mean squared error as a metric. Its lowest root mean squared error was 3.90 (obtained with a random forest regressor) while its highest error was 4.59 (obtained with a naive predictor). These translate to mean squared errors of 15.21 and 21.07 respectively.

III. Methodology

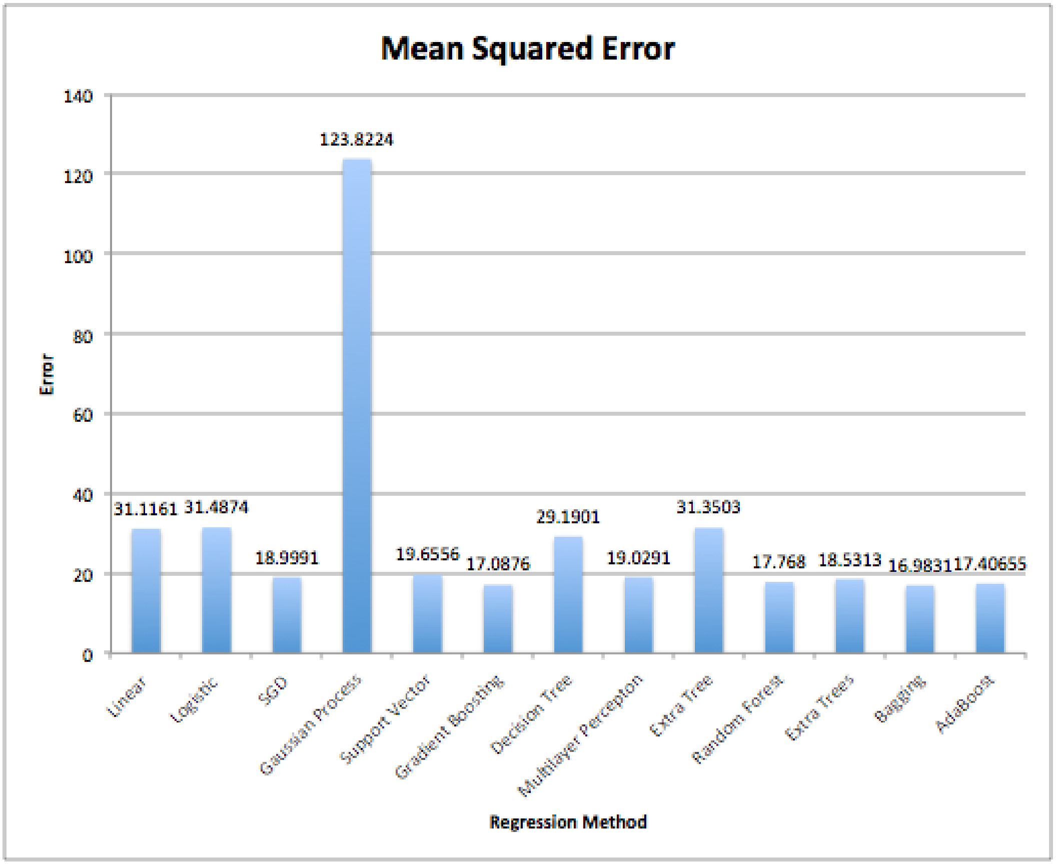
Data Preprocessing

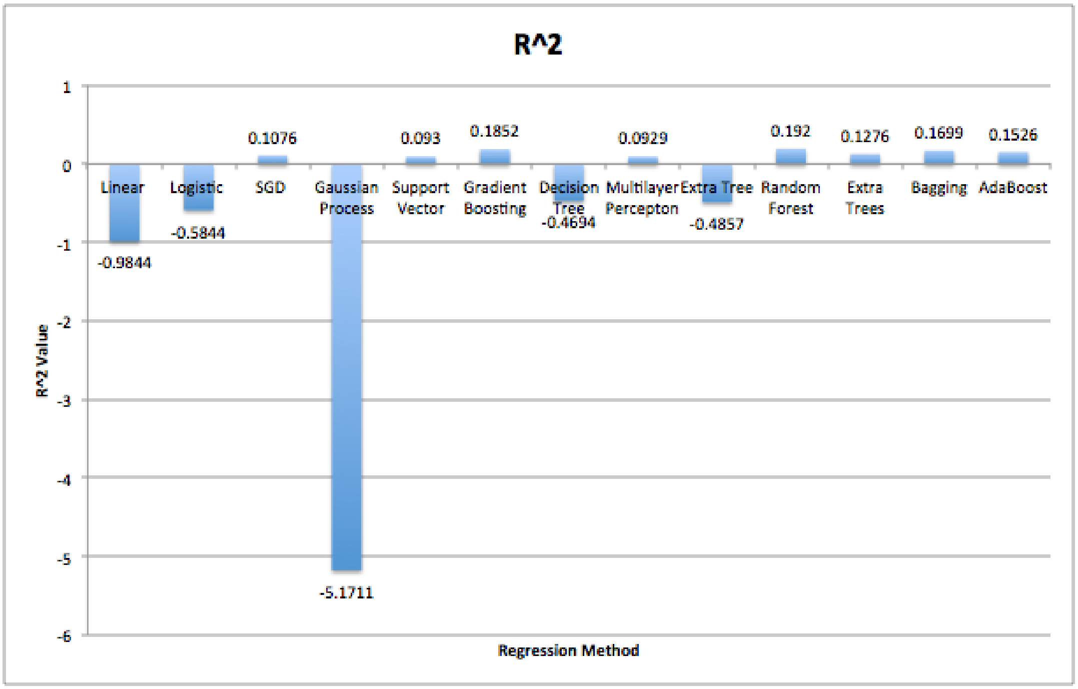
The first step of preprocessing the data was to separate the input features, the target features, and irrelevant features. Specifically, this meant deleting the G1 and G2 features and then separating the remaining data from the G3 feature. This was followed by one-hot encoding the categorical features. After this, discrete features were scaled to have a mean of 0 and a standard deviation of 1. This was accomplished by manually applying the z-score function to the data. The continuous features were then scaled between 1 and 2 through a custom defined scale feature which applied the Min-Max scaling formula as described previously. After this, the Box Cox function was applied to the data. The data was then log-transformed. These two steps served the purpose of normalizing the continuous data. The continuous data was then also scaled to have a mean of 0 and a standard deviation of 1. Finally, Tukey outliers (described previously) were removed.

Implementation

The models that were tested included linear regressor, logistic regressor, stochastic gradient descent regressor, Gaussian process regressor, support vector regressor, gradient boosting regressor, decision tree regressor, multilayer perceptron, extra tree regressor, random forest regressor, extra trees regressor, bagging regressor, and Adaboost regressor. All of these models were imported from scikit-learn. They were trained and evaluated on a training set that was divided into 10 folds. This meant that, by the end of this training and evaluation, there was a testing set that was still unknown to all of the models.

For each model that was tested, the mean squared error was used as a metric of evaluation. This metric was passed in as an argument to the cross_val_score function which was imported from scikit-learn. The same was carried out for the R^2 score. The model with the lowest mean squared error was then selected for hyperparameter optimization. Below are visualizations of the mean squared errors and R^2 scores for all unoptimized models that were tested.





Refinement

Based off of the comparison of unoptimized model, the bagging regressor performed the best, with a mean squared error of 16.983. Its R^2 score, 0.170, was also one of the highest. I then used gridsearch in order to find the combination of hyperparameters that would minimize the mean squared error. The hyperparameters tested were the number of estimators, the maximum number of features used, the maximum number of samples used, whether out of the bag samples were used, whether samples were drawn with replacement, and whether features were drawn with replacement. Gridsearch used 3 fold cross validation on the training set in order to determine the optimal hyperparameters, and then used that optimal model to make predictions from the testing set. The optimized model had a mean squared error of 13.6330 and an R^2 value of 0.1305 on the testing set. These values show that the optimized model outperformed not only the unoptimized model and the benchmark linear regressor, but also the results reported in the paper.

IV. Results

Model Evaluation and Validation

The final model was a bagging regressor that:

- Used 100 estimators
- Used 55 maximum features
- Used 160 maximum samples
- Did not use out of the bag samples
- Did not draw samples with replacement
- Did not draw features with replacement

This model was chosen based off of a comparison of a number of unoptimized models. It was then optimized using a gridsearch algorithm as described above. The model is robust and generalizes well. This is shown not only by its mean squared error and R^2 score, but also through the fact that these metrics were derived from a testing set that was not used until the absolute end.

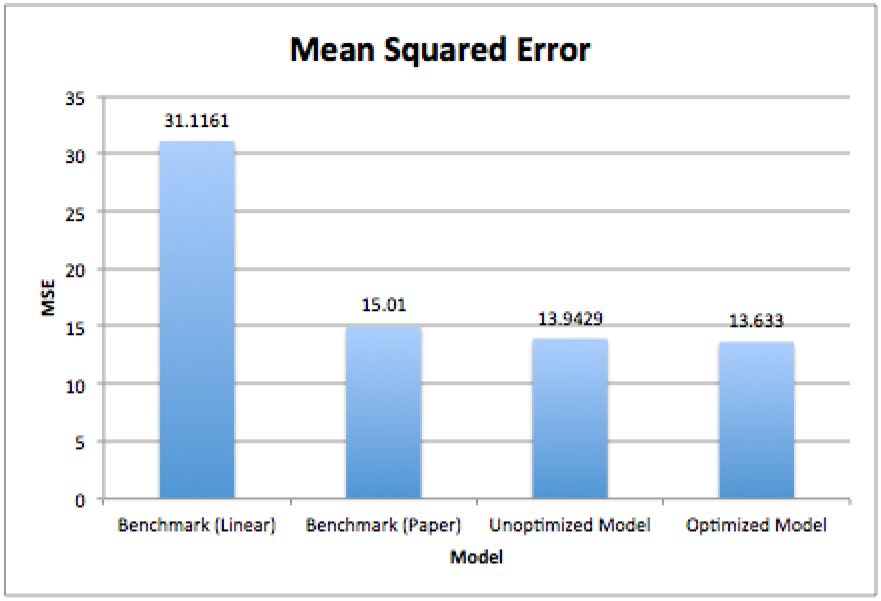
Justification

	Mean Squared Error	R^2
Benchmark (Linear)	31.1161	-0.9844
Benchmark (Paper)	15.01(lower)/21.07(higher)	N/A
Unoptimized Model (Testing)	13.9429	0.1108
Optimized Model (Training Cross-Validation)	16.9831	0.1699
Optimized Model (Testing)	13.6330	0.1305

Considering that the model has achieved a lower error than all benchmarks, the results and the solution are significant enough to have solved the problem.

V. Conclusion

Free-Form Visualization



The above graph illustrates the contrast between the benchmarks and final model(s) chosen. It is clear that the optimized model had the lowest mean squared error. Since this was the principal metric being used, it is safe to say that the problem has successfully been solved.

Reflection

In general, I found it surprising how much more time and effort it took to preprocess the data than it did to choose an optimal model. Using pandas greatly facilitated one-hot encoding categorical features as well as separating the inputs and target. I was also surprised by how I personally found it easier to translate scaling functions into code rather than using existing functions such as MinMaxScaler to perform the same task. One of the most interesting parts of the preprocessing step was discovering the existence of the normalscore function. It was fascinating to find a function that managed to return a numerical translation of a shape. This was also the first time I used a combination of functions to normalize data instead of just using a simple log transformation. It proved to be much more effective. In addition, the counter class made it easy to find outliers shared among multiple features.

Sklearn facilitated much of the process overall, but in particular, it made fitting and testing different models relatively painless as well as the gridsearch process. I was also surprised by the final result and how much better the optimal model did than the published result. I noticed that the target feature (the final math grade) was a single number between 0 and 20, making it too small to be a percentage . This is either a grading format unique to Portugal or a scaled version of the true grades. In either case, this model cannot be generalized to other schools due to this limitation.

Improvement

One improvement that could be made to this solution is during the optimization stage of the process. Not all arguments taken by the bagging regressor constructor were tested with gridsearch. In particular, the base_estimator and warm_start arguments were not part of the parameter dictionary. I think a better solution is possible, particularly through manipulation of the base_estimator argument. The default value of this argument is decision tree, but other regressors could be attempted.

References

- BRITO, A. ; TEIXEIRA, J., eds. lit. – "Proceedings of 5th Annual Future Business Technology Conference, Porto, 2008". [S.l. : EUROSIS, 2008]. ISBN 978-9077381-39-7. p. 5-12

- https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.boxcox.html
- http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.BaggingRegressor.html#sklearn.ensemble.BaggingRegressor
- http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html