Test Score Prediction - PISA 2009

Machine Learning Engineer Nanodegree
Capstone Project

Shinto Theruvil Manuel

December 11th, 2016

I. Definition

Project Overview

The **Programme for International Student Assessment (PISA)** is a worldwide study by the Organisation for Economic Co-operation and Development (OECD) of 15-year-old school pupils' scholastic performance on mathematics, science, and reading. It was first performed in 2000 and then repeated every three years. It is done with a view to improving education policies and outcomes. It measures problem solving and cognition in daily life[1].

PISA aims at testing literacy in three competence fields: reading, mathematics, science on a 1000-point scale.

Each student takes a two-hour handwritten test. Part of the test is multiple-choice and part involves fuller answers. There are six and a half hours of assessment material, but each student is not tested on all the parts. Following the cognitive test, participating students spend nearly one more hour answering a questionnaire on their background including learning habits, motivation, and family. School directors fill in a questionnaire describing school demographics, funding, etc.

In this project, we will predict the reading scores of students using the background information provided by student and the School. We use PISA 2009 dataset as reading literacy was the main domain assessed in 2009.

Problem Statement

The dataset contain information about the demographics and reading scores for American students taking the exam, derived from 2009 PISA Public-Use Data Files distributed by the United States National Center for Education Statistics(NCES). Each row in the dataset represents one student taking the exam. The variable, "**readingScore**" will be used as target and the other 23 variables for demographical information.

The goal is to train a model to predict the reading score of student using their demographical information. Since this is a problem of supervised learning of type regression. Several regression algorithms will be explored, as discussed in Phase II, Analysis.

Metrics

The prediction outcome (test score) is a numeric value between 1 and 1000. When the outcome is a number, the most common method for characterizing a model's predictive capabilities is to use the root mean squared error (**RMSE**). This metric is a function of the model residuals, which are the observed values minus the model predictions. The mean squared error (**MSE**) is calculated by squaring the residuals, summing them and dividing by the number of samples. The RMSE is then calculated by taking the square root of the MSE so that it is in the same units as the original data [2].

Compared to the mean squared error, RMSE amplifies and severely punishes large errors. The RMSE is calculated using below formula.

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

In Python using sklearn:

```
from sklearn.metrics import mean_squared_error
RMSE = mean_squared_error(y, y_pred)**0.5
```

Another common metric is the R^2. It provides an indication of goodness of fit of a set of predictions to the actual values. This measure is called coefficient of determination. In other words, it can be interpreted as the proportion of the information in the data that is explained by the model. This is a value between 0 and 1 for no-fit and the perfect fit respectively. Thus, an R^2 value of 0.75 implies that the model can explain three-quarters of the variation in the outcome. Mostly R^2 is a measure of correlation, not accuracy. It's the correlation coefficient between the observed and predicted values and squares it.

In Python using sklearn:

```
from sklearn.metrics import r2_score
R2 = r2_score(y, y_pred)
```

We use **RMSE** and R^2 for this problem.

II. Analysis

Data Exploration [exploratory_analysis.ipynb]

The dataset **pisa2009.csv** contain information about the demographics and schools for American students taking the exam, derived from 2009 PISA Public-Use Data Files distributed by the United States National Center for Education Statistics (NCES). While the datasets are not supposed to contain identifying information about students taking the test, by using the data you are bound by the NCES data use agreement, which prohibits any attempt to determine the identity of any student in the datasets.

Dataset Size

5233 rows, 24 fields

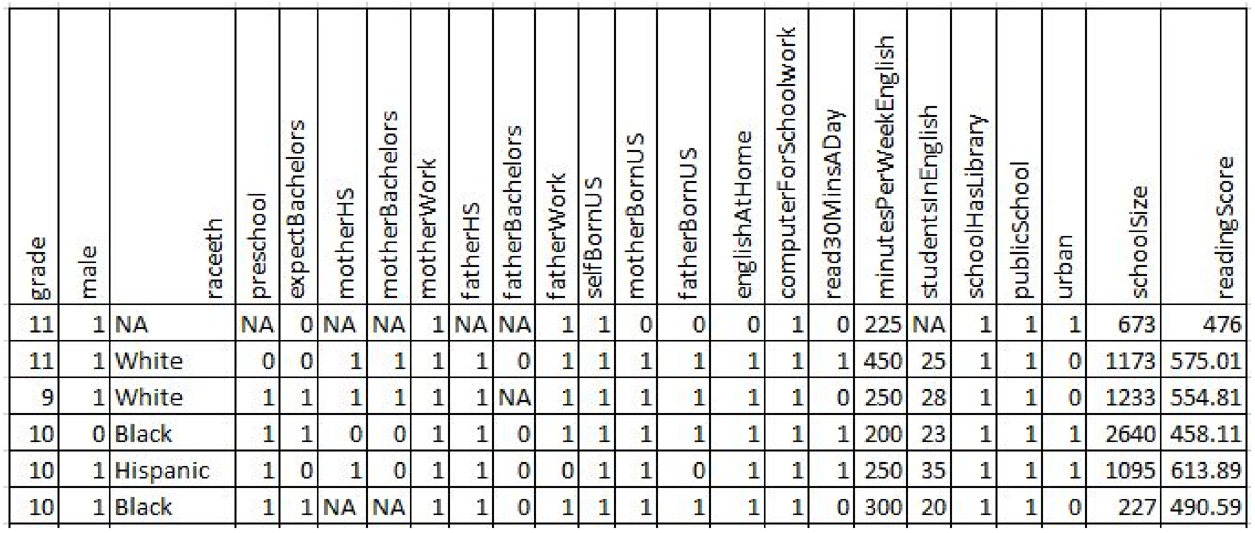
Fields

The datasets have the following variables:

- **grade**: The grade in school of the student (most 15-year-olds in America are in 10th grade)
- **male**: Whether the student is male (1/0)
- **raceeth**: The race/ethnicity composite of the student
- **preschool**: Whether the student attended preschool (1/0)
- **expectBachelors**: Whether the student expects to obtain a bachelor's degree (1/0)
- **motherHS**: Whether the student's mother completed high school (1/0)
- **motherBachelors**: Whether the student's mother obtained a bachelor's degree (1/0)
- **motherWork**: Whether the student's mother has part-time or full-time work (1/0)
- **fatherHS**: Whether the student's father completed high school (1/0)
- **fatherBachelors**: Whether the student's father obtained a bachelor's degree (1/0)
- **fatherWork**: Whether the student's father has part-time or full-time work (1/0)
- **selfBornUS**: Whether the student was born in the United States of America (1/0)
- **motherBornUS**: Whether the student's mother was born in the United States of America (1/0)
- **fatherBornUS**: Whether the student's father was born in the United States of America (1/0)
- **englishAtHome**: Whether the student speaks English at home (1/0)
- **computerForSchoolwork**: Whether the student has access to a computer for schoolwork (1/0)
- **read30MinsADay**: Whether the student reads for pleasure for 30 minutes/day (1/0)
- **minutesPerWeekEnglish**: The number of minutes per week the student spend in English class

- **studentsInEnglish**: The number of students in this student's English class at school
- **schoolHasLibrary**: Whether this student's school has a library (1/0)
- **publicSchool**: Whether this student attends a public school (1/0)
- **urban**: Whether this student's school is in an urban area (1/0)
- **schoolSize**: The number of students in this student's school
- **readingScore**: The student's reading score, on a 1000-point scale (This is the target variable)

Preview of dataset



grade	male	raceeth	preschool	expectBachelors	motherHS	motherBachelors	motherWork	fatherHS	fatherBachelors	fatherWork	selfBornUS	motherBornUS	fatherBornUS	englishAtHome	computerForSchoolwork	read30MinsADay	minutesPerWeekEnglish	studentsInEnglish	schoolHasLibrary	publicSchool	urban	schoolSize	readingScore
11	1	NA	NA	0	NA	NA	1	NA	NA	1	1	0	0	0	1	0	225	NA	1	1	1	673	476
11	1	White	0	0	1	1	1	1	0	1	1	1	1	1	1	1	450	25	1	1	0	1173	575.01
9	1	White	1	1	1	1	1	1	NA	1	1	1	1	1	1	0	250	28	1	1	0	1233	554.81
10	0	Black	1	1	0	0	1	1	0	1	1	1	1	1	1	1	200	23	1	1	1	2640	458.11
10	1	Hispanic	1	0	1	0	1	1	0	0	1	1	0	1	1	1	250	35	1	1	1	1095	613.89
10	1	Black	1	1	NA	NA	1	1	0	1	1	1	1	1	1	0	300	20	1	1	0	227	490.59

Only one variable, **'raceeth'** is text. We will process this a categorical type. Among other numerical variables, three are continuous type. They are 1) **studentsInEnglish**, 2) **minutesPerWeekEnglish** and 3) **schoolSize**. Our prediction target, **readingScore** is also continuous. All other numeric variables, except **grade** are of type binary having values either 0 or 1. The **grade** has four discrete values, 8, 9, 10, or 12.

Distribution of continuous variables

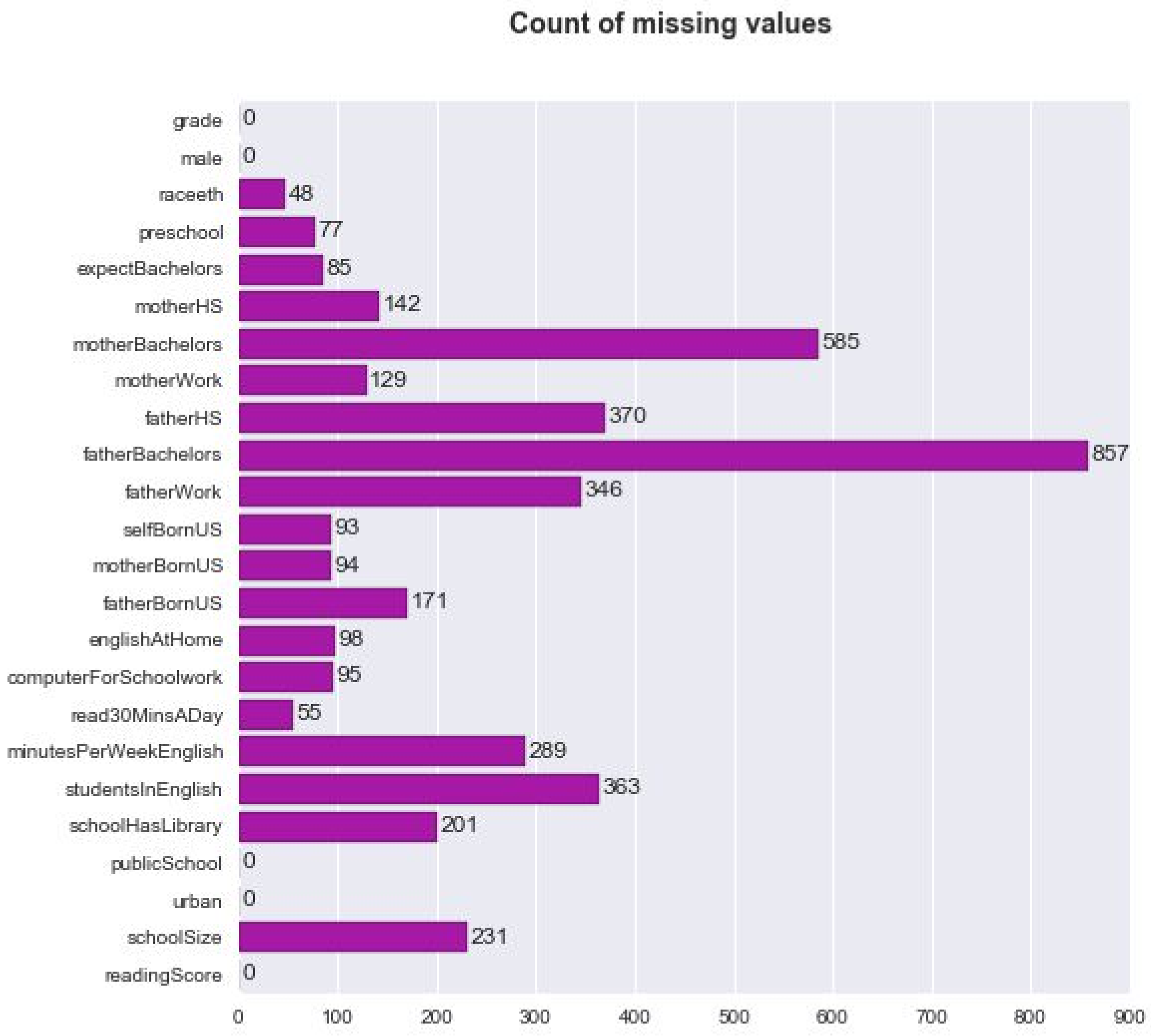
	studentsInEnglish	**minutesPerWeekEnglish**	**schoolSize**	**readingScore**
count	4870.000000	4944.000000	5002.000000	5233.000000
mean	24.559754	265.717840	1374.367653	497.591875
std	7.139661	149.591118	870.424790	95.598917
min	1.000000	0.000000	100.000000	156.380000
25%	20.000000	225.000000	712.000000	431.270000
50%	25.000000	250.000000	1233.000000	499.580000
75%	30.000000	300.000000	1900.000000	565.510000

max	90.000000	2400.000000	6694.000000	772.460000

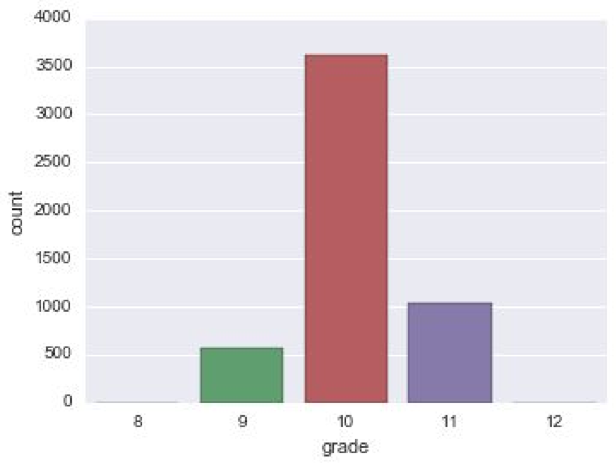
The min and max values as well are the means vary a lot for the continuous variables. We are likely going to get better results by rescaling the data in some way in the preprocessing stage.

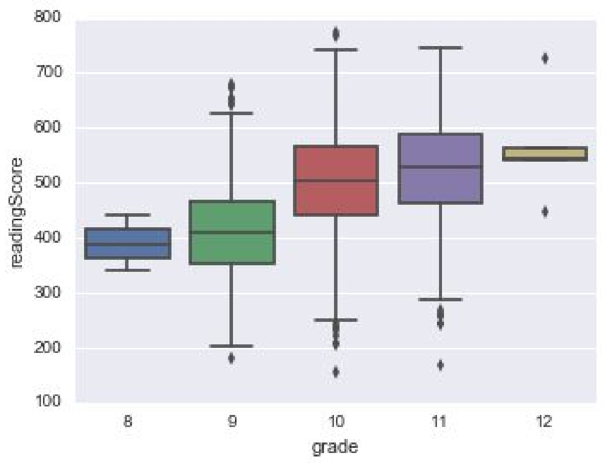
Exploratory Visualization

Missing values

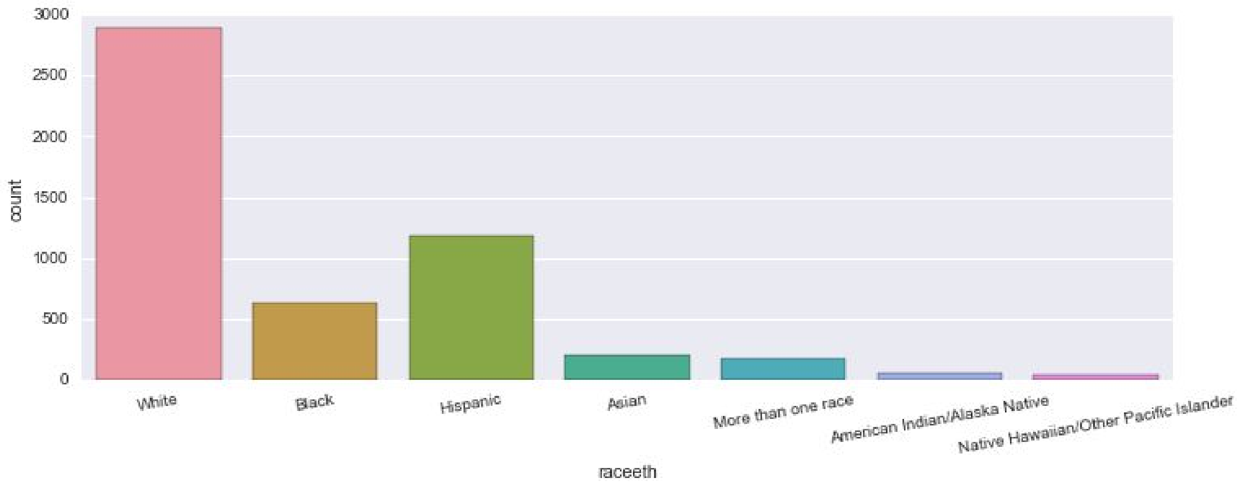


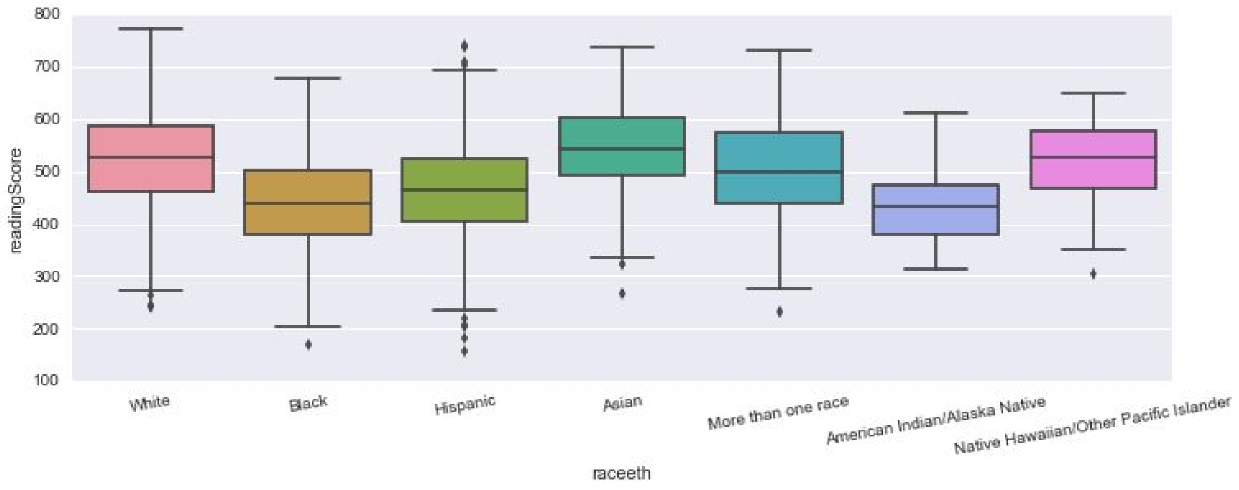
grade and readingScore



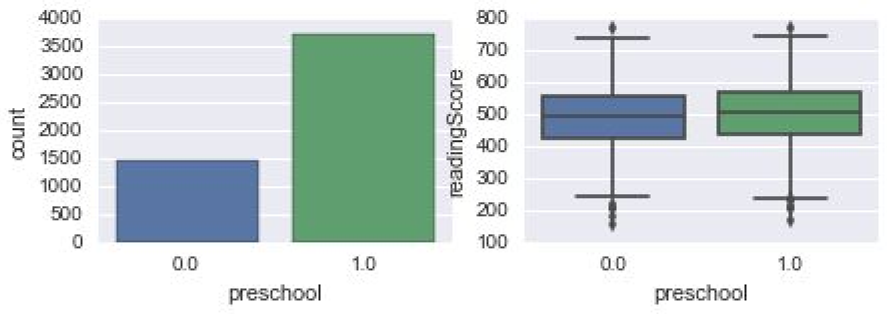


Raceeth and readingScore

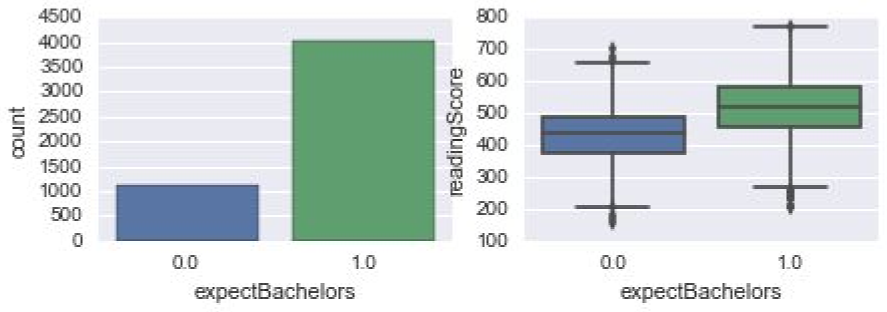




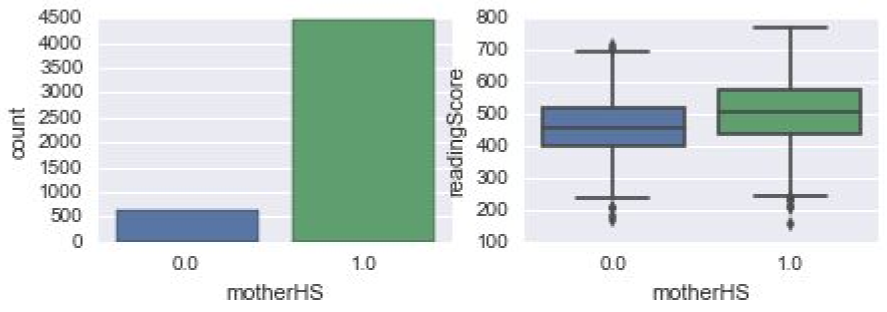
preSchool and readingScore



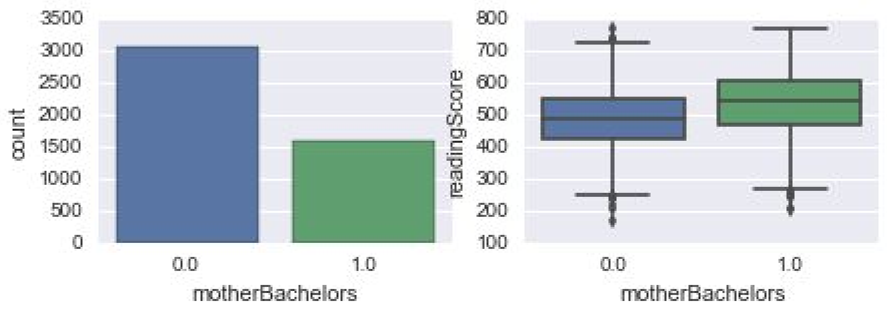
expectBachelors and readingScore



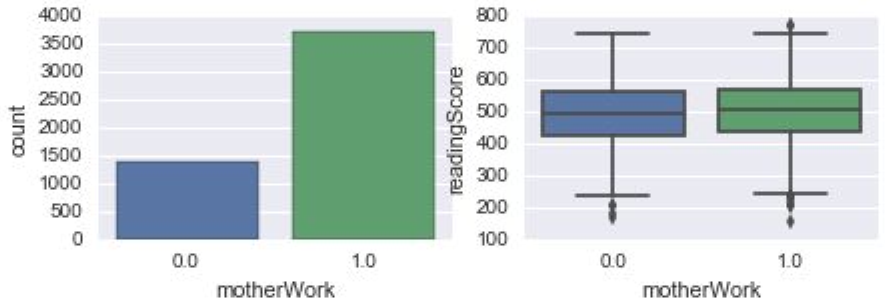
motherHS and readingScore



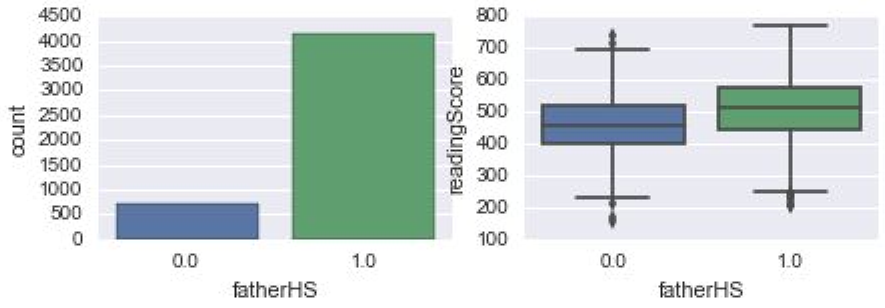
motherBachelors and readingScore



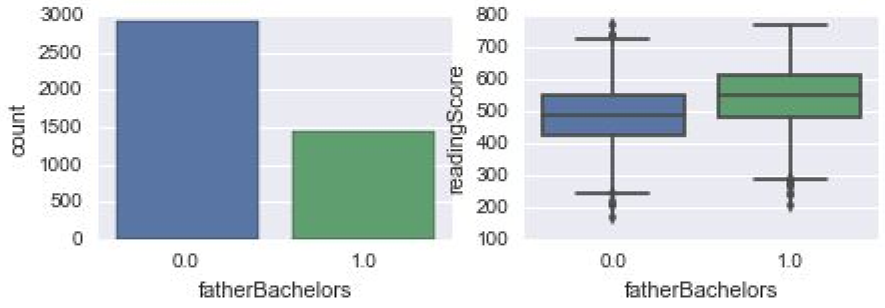
motherWork and readingScore



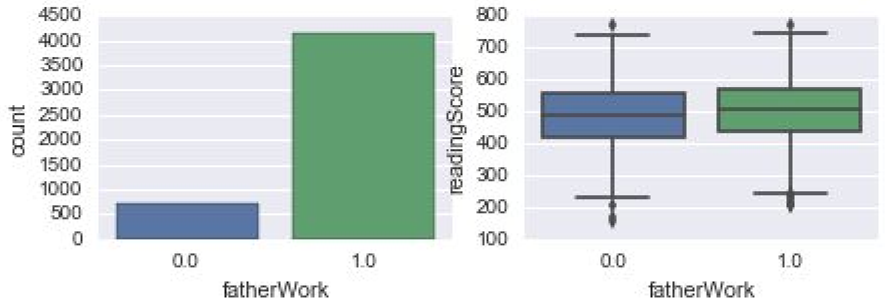
fatherHS and readingScore



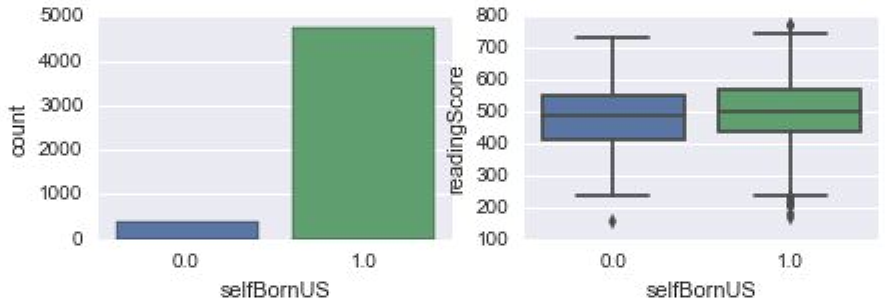
fatherBachelors and readingScore



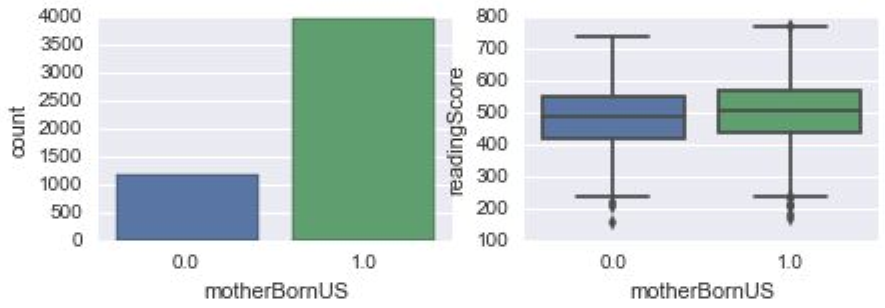
fatherWork and readingScore



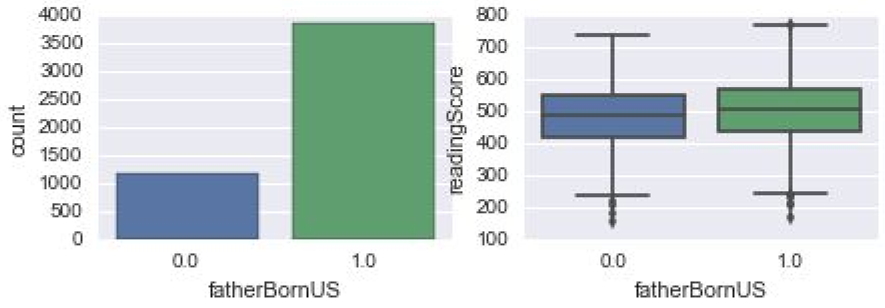
selfBornUS and readingScore



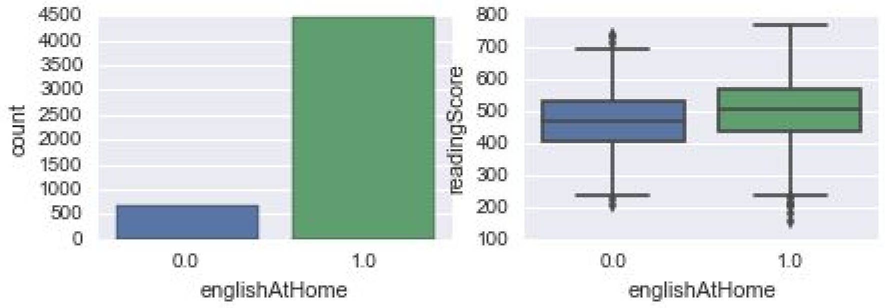
motherBornUS and readingScore



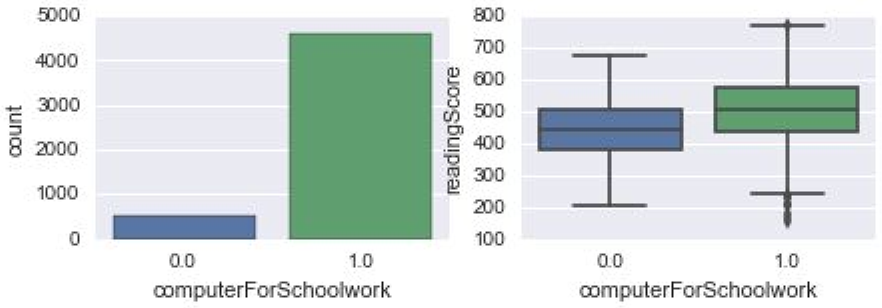
fatherBornUS and readingScore



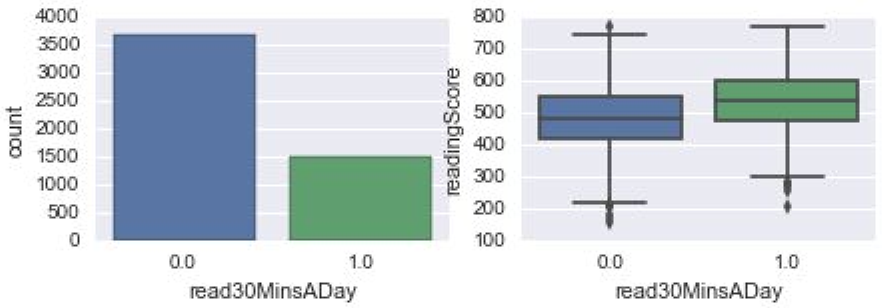
englishAtHome and readingScore



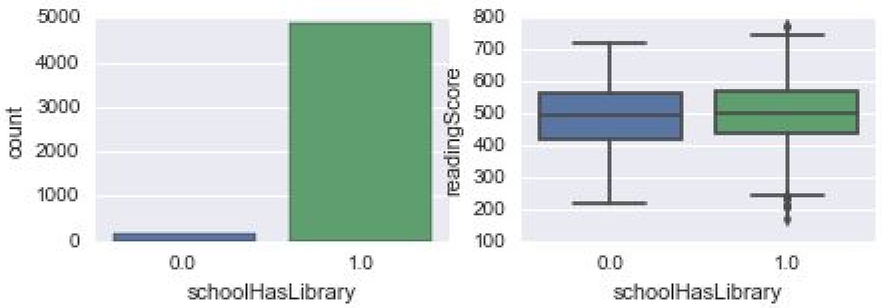
computerForSchoolwork and readingScore



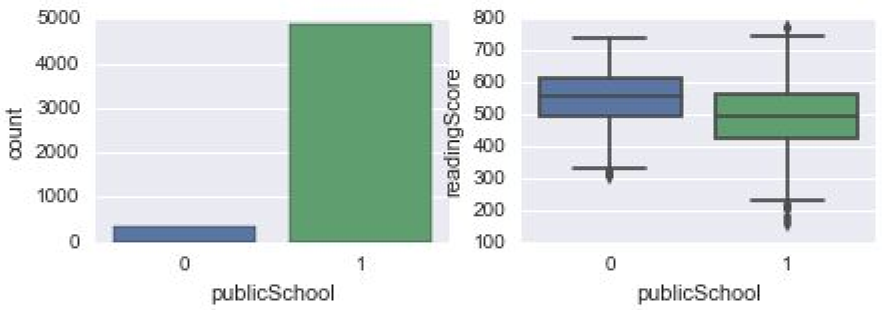
read30MinsADay and readingScore



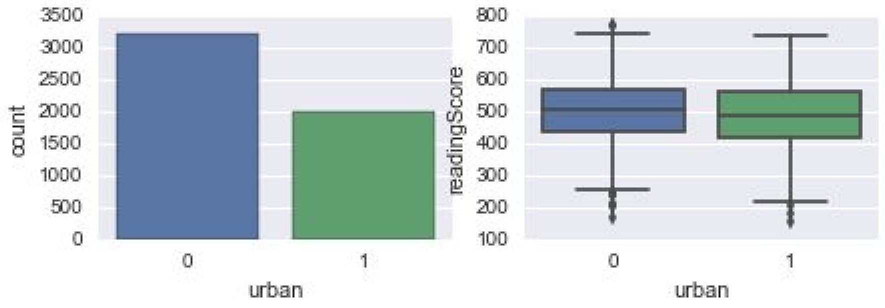
schoolHasLibrary and readingScore



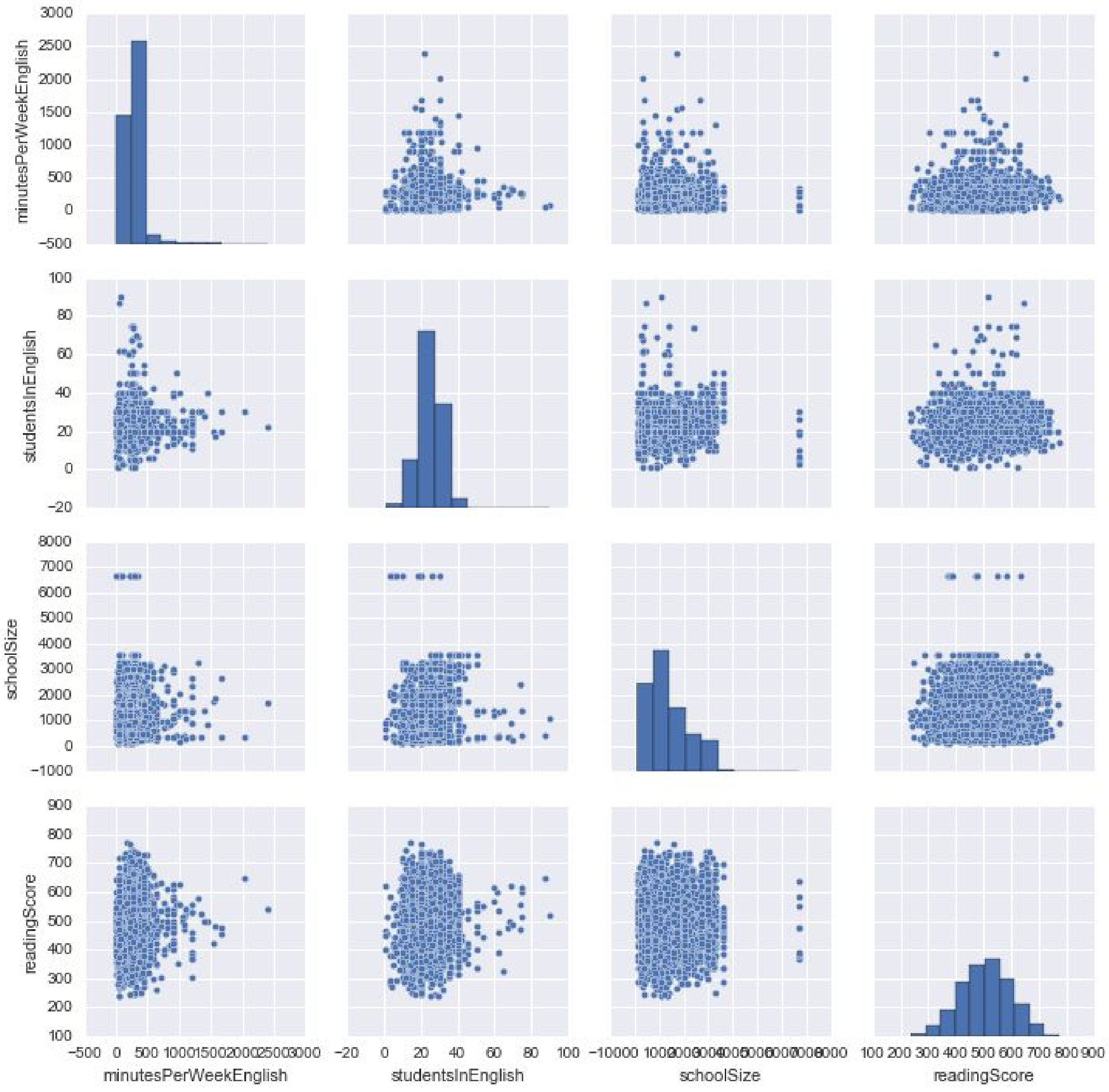
publicSchool and readingScore



urban and readingScore

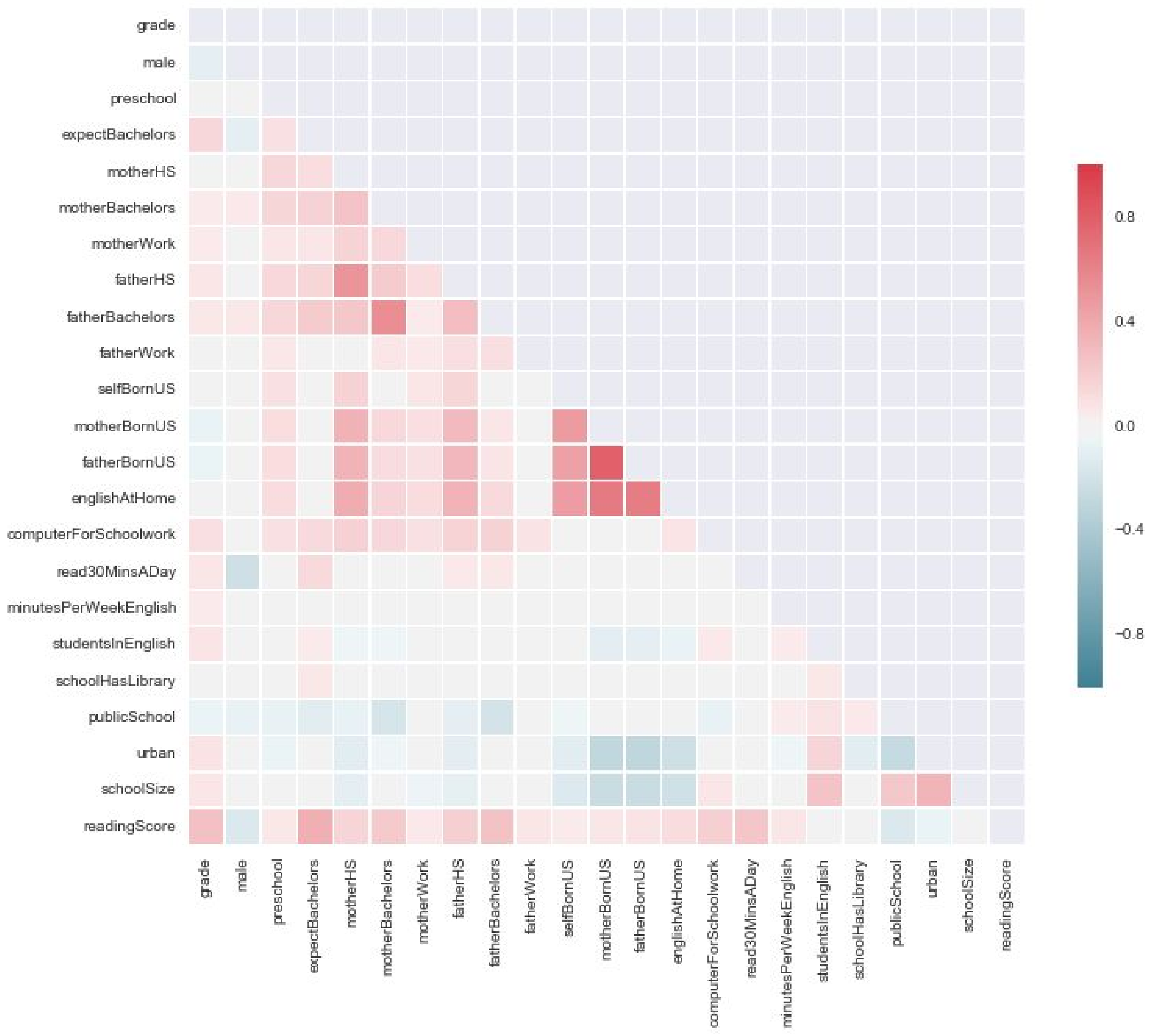


Continuous variables distribution and reading score



The target variable **readingScore** has a normal distribution. The variables **schoolSize** and **minutesPerWeekEnglish** shows some skewness.

Heatmap showing correlation among features



Algorithms and Techniques

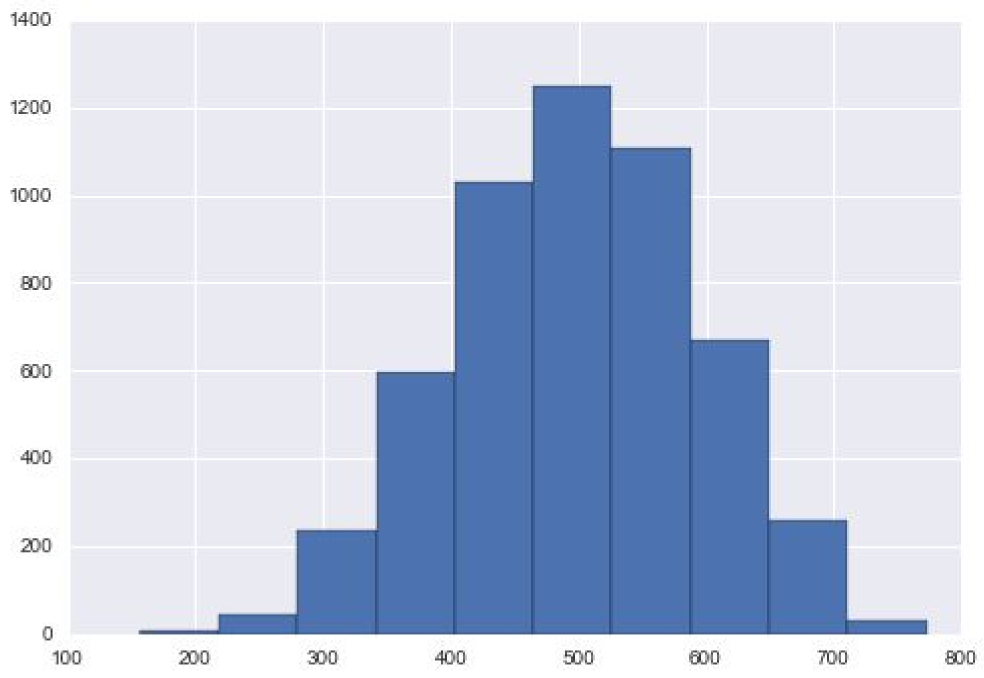
There are many good algorithms for regression problems and it is difficult to say which algorithm will do well on a problem in advance. We will try a number of different algorithms. The twelve algorithms considered are:

- **Linear Algorithms**: Linear Regression , Lasso Regression, Ridge, Bayesian Ridge and ElasticNet .
- **Nonlinear Algorithms**: Decision Tree Regressor, Support Vector Regression and k-Nearest Neighbors.
- **Ensemble Algorithms**: AdaBoost, Gradient Boosting, Random Forest Regressor, and ExtraTrees Regressor.

We will use a test harness with 10-fold cross validation. We will evaluate algorithms using the Root Mean Squared Error (RMSE) metric. RMSE will give a gross idea about the accuracy of predictions (0 is perfect).

Benchmark

The primary goal of a benchmark is to find out the best choice of algorithm. The reading scores has normal distribution. So we will use the mean score as the predicted score and calculate RMSE.



Mean reading score = 497.60
Benchmark RMSE = 95.59
Benchmark R^2 = 0.0

Our goal is to get a lower RMSE and higher R^2 value than our benchmark .

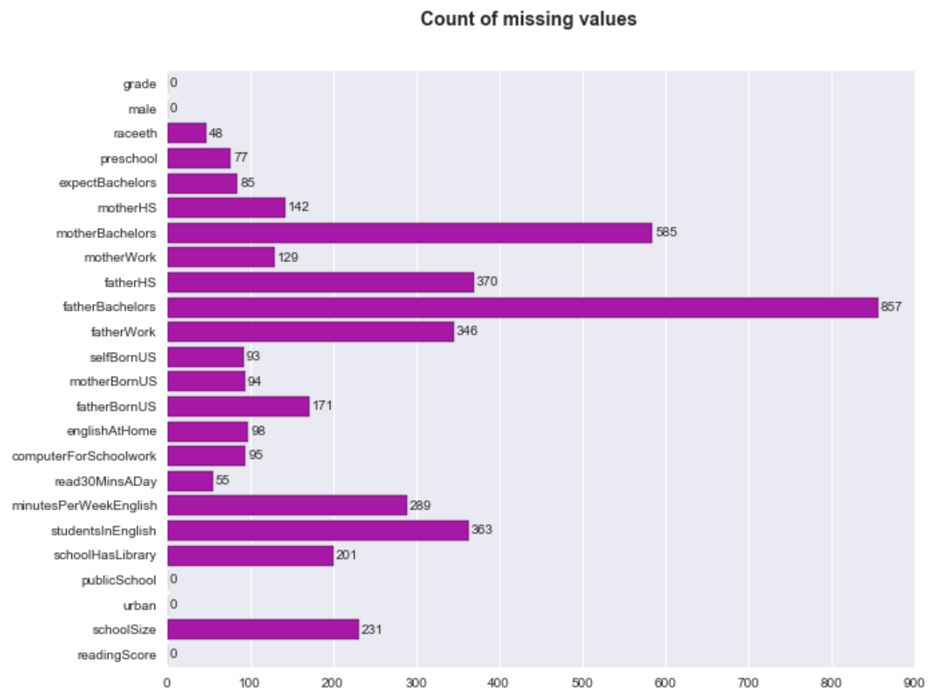
III. Methodology

Data Preprocessing [PreProcessing.ipynb]

Handling missing values

In the real world data is rarely clean and often can have corrupt or missing values. It is important to identify, and handle missing data when developing machine learning models.

There are 4329 missing data values in the dataset. Except four features (grade, male, public school, urban) all the remaining 19 features have some missing values. The highest number of missing values, 857 is seen in feature, **'fatherBachelors'** in the training set. The lowest count of missing values, 48 is seen in feature, **'raceeth'**.



Replacing the missing values with an average of the column (called **Mean Imputation**) is quick and simple but at the expense of underestimate the variance. A method called **KNN Imputation**, which iterate over all the data points and perform calculation of each missing value, with the assumption that missing value is correlated with with values from other features. Another method called **Regression Imputation** in which a regression model is used to predict the missing values based on other variables and that model is then used to impute values.

Following are approached used for this dataset:
For the continuous variables, the missing values were replaced by average of columns.
For the categorical variable, ‘raceeth’ the missing value is replaced by string ‘NoRace’.
For all binary variables, the missing values were replaced by value 0.5

Encoding Categorical values

Our data has only one categorical variable, “**raceeth**”. It contains a number of discrete categories, such as ‘White’, ‘Black’, ‘Asian’ etc.,. Most machine learning algorithms require all input in numeric form.
There are two common ways to convert discrete text labels to numeric form:

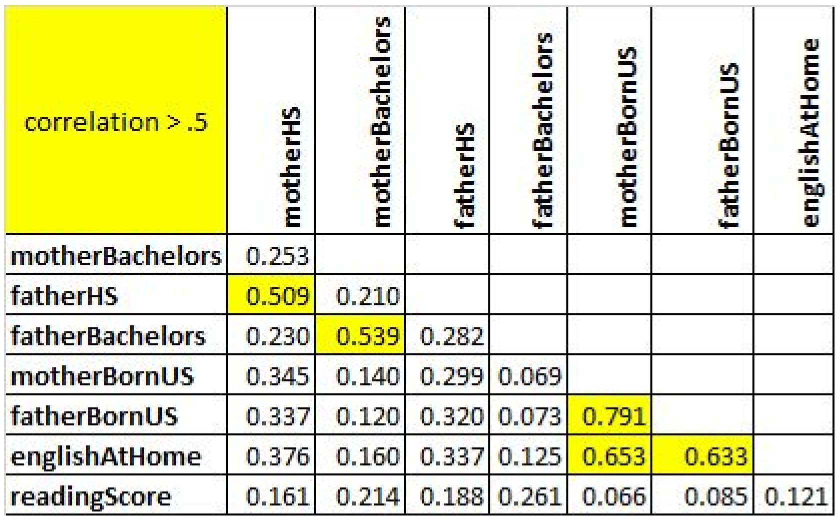
1. Represent each item by a number. Like ‘white’ is 1, ‘Asian’ is 2.
2. Binary encode each unique label as a new variable.

The first option is useful if there is an inherent order in the labels. In our case there no order among the values of **‘raceeth’**. So we will go with second option, create new variable for each unique label.

Between-Predictor Correlations (Collinearity)

Collinearity is the situation where a pair of predictor variables have substantial correlation with each other.

Below table highlights the collinearity among the features and their correlation to the target variable, readingScore.



correlation > .5	motherHS	motherBachelors	fatherHS	fatherBachelors	motherBornUS	fatherBornUS	englishAtHome
motherBachelors	0.253						
fatherHS	0.509	0.210					
fatherBachelors	0.230	0.539	0.282				
motherBornUS	0.345	0.140	0.299	0.069			
fatherBornUS	0.337	0.120	0.320	0.073	0.791		
englishAtHome	0.376	0.160	0.337	0.125	0.653	0.633	
readingScore	0.161	0.214	0.188	0.261	0.066	0.085	0.121

Highly correlated features in models like Linear Regression can result in degraded predictive performance. When two features have high correlation only one, which is more correlated to the target is retained.

So from the above table following four are retained:

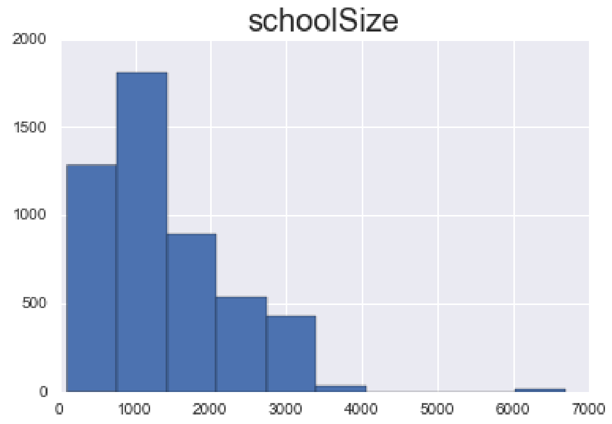
- **englishAtHome**
- **fatherHS**
- **motherHS**
- **fatherBachelors**

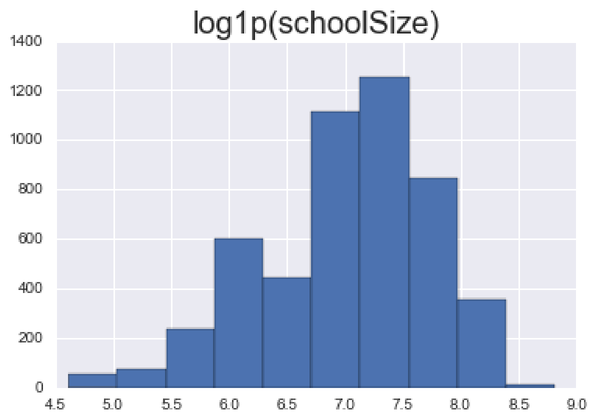
And the following three features are removed :

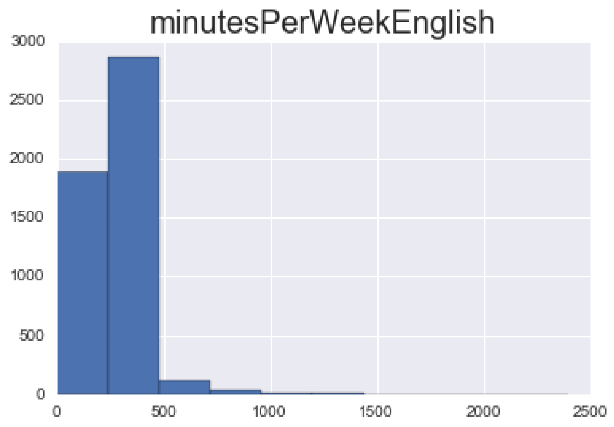
- **motherBachelors**
- **motherBornUS**
- **fatherBornUS**

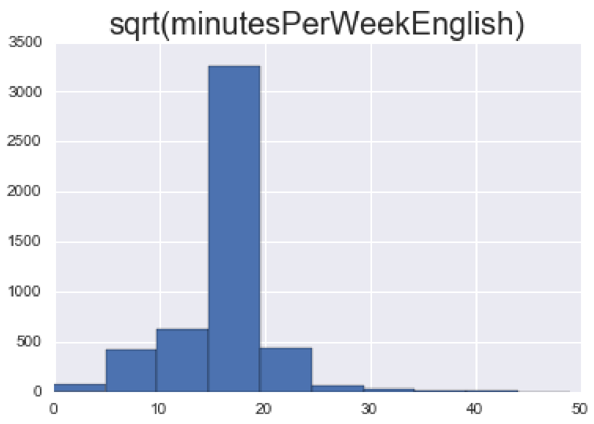
Skew correction

The two features, **‘schoolSize’** and **‘minutesPerWeek’** do not have the bell shaped normal distribution. They skewed and we can use transformations to reduce the skewness.









Implementation [Algorithms.ipynb]

A. Linear algorithms

Linear models describe a continuous response variable as a function of one or more predictor variables. Linear models are highly interpretable. They are appropriate for when the relationship between the predictors and response falls along a hyper plane. Following five linear models a tested for the pisa score prediction.

- **Linear Regression**: Most basic and commonly used linear model, assume that input variables have a Gaussian distribution. It is also assumed that input variables are relevant to the output variable and they are not highly correlated with each other (collinearity).

- **Lasso Regression:** The Least Absolute Shrinkage and Selection Operator(LASSO) is a modification of linear regression. Similar to Ridge regression in the sense that regression is penalized by some amount.

- **Ridge Regression**: Another modification of linear regression. Helps to overcome the linear regression's shortfalls by introducing a regularization parameter to shrink the coefficients. By imposing a penalty on the size of coefficients, the ridge regression minimizes residual sum of squares.

- **Bayesian Ridge Regression:** Similar to Ridge Regression. It can be used to include regularization parameter in the estimation procedure. The main advantage of this model is that it adapts to data at hand.

- **ElasticNet**: It is a form of regularization regression that combines the properties of both Ridge Regression and LASSO regression.

B. Non-linear algorithms

- **K-Nearest Neighbors**: KNN locates the K most similar instances in the training dataset for a new observation. From the K neighbors, a mean or median output variable is taken as the prediction.
- **Decision Tree Regressor**: Use the training data to select the best points to split the data in order to minimize a cost metric. The default metric is the mean squared error.
- **Support Vector Regresso**r: Initially developed for binary classification Support Vector Machines were extended for regression type problems. They are based on the concept of decisionplanes that decision boundaries

C. Ensemble algorithms

- **AdaBoost:** Short for adaptive boosting, use a set of weak learners and by learning the accuracy of each it will determine the weights of building a strong learner.
- **Gradient Boosting:** Identifies difficult observations by large residuals computed in the previous iterations. In other words, Gradient Boosting is a technique that learns from its mistakes.
- **Random Forests:** Ensembles of decision trees, makes predictions by combining decisions from a sequence of base models. The base models are constructed independently using a different subsample of the data.
- **Extra Trees:** Implements a meta estimator that fits a number of randomized decision trees on various subsamples of the dataset and use averaging to improve the predictive accuracy.

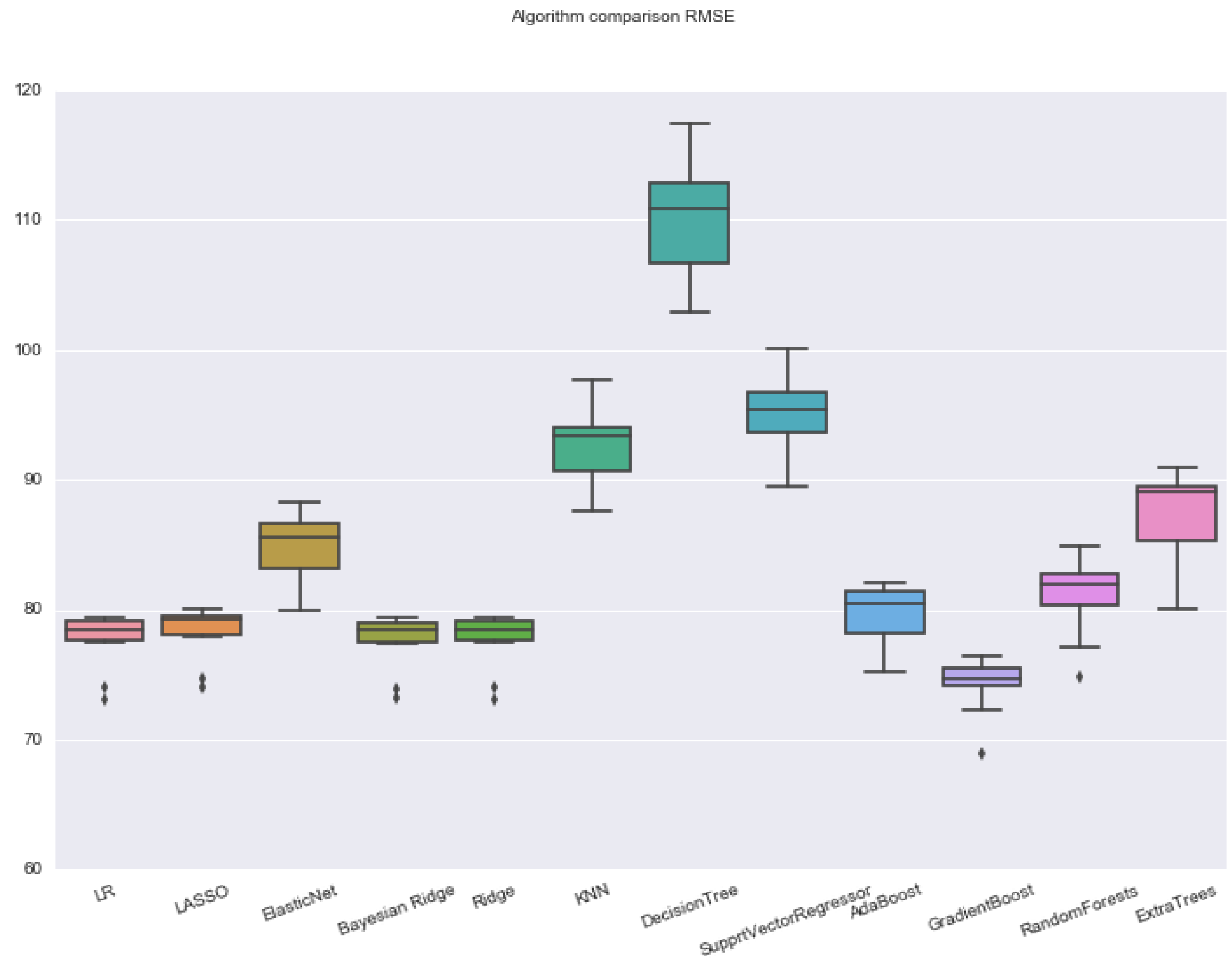
10 fold cross validation scores from the linear and non-linear models

Model	**Type**	**RMSE (std)**	R^2 **(std)**
Linear Regression	Linear	77.638 (2.122)	0.341 (0.035)
Lasso	Linear	78.297 (2.052)	0.330 (0.029)
ElasticNet	Linear	85.144 (2.580)	0.209 (0.012)
Bayesian Ridge	Linear	77.620 (2.108)	0.341 (0.034)
Ridge	Linear	77.635 (2.120)	0.341 (0.035)

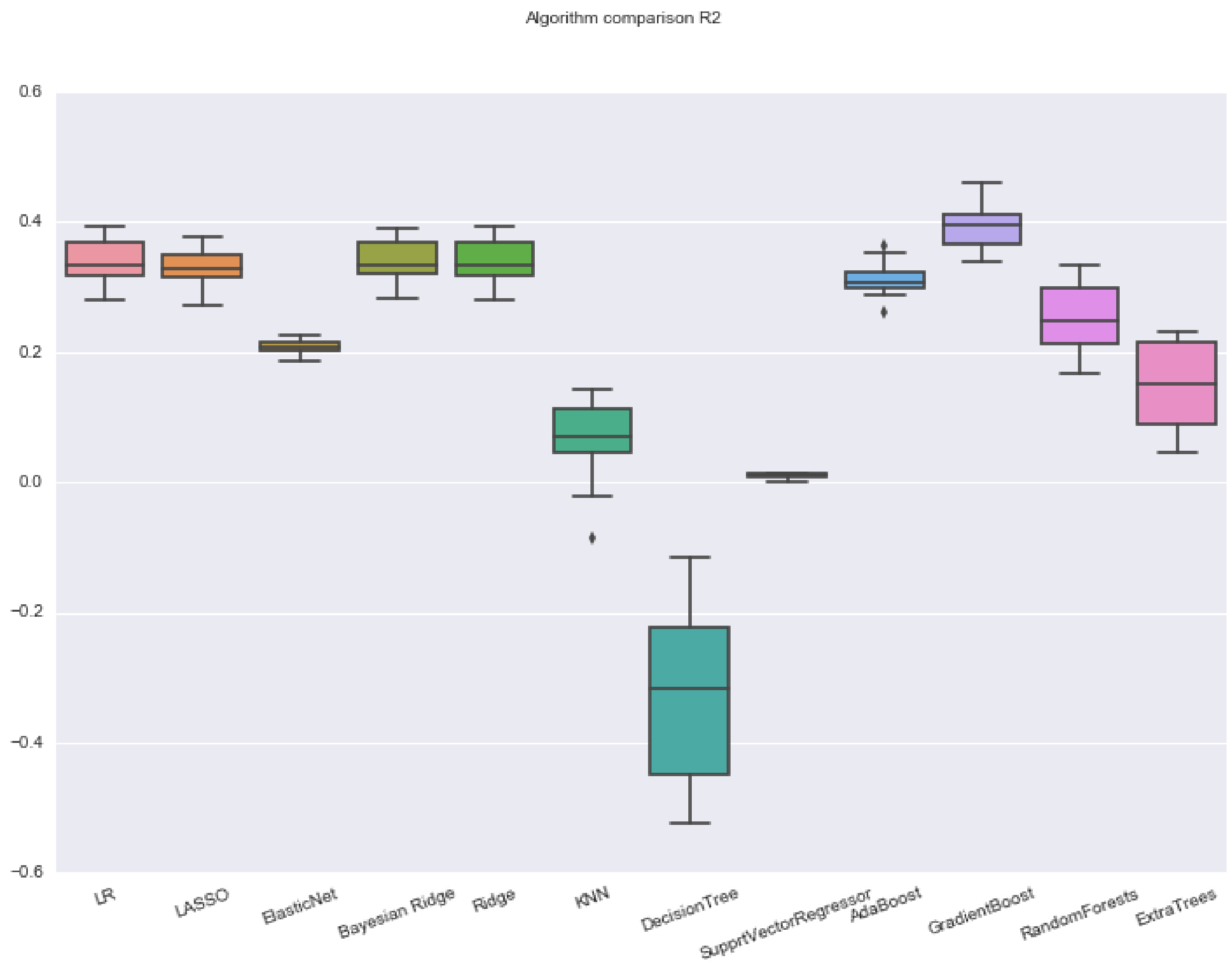
K-Nearest Neighbors	Non-linear	92.656 (3.191)	0.061 (0.066)
Decision Tree Regressor	Non-linear	109.029 (3.884)	-0.308 (0.127)
Support Vector Regressor	Non-linear	95.231 (3.198)	0.010 (0.005)
AdaBoost Regressor	Ensemble (boosting)	79.295 (2.448)	0.311 (0.028)
Gradient Boosting Regressor	Ensemble (boosting)	74.311 (2.140)	0.396 (0.038)
Random Forests Regressor	Ensemble (bagging)	81.568 (2.607)	0.271 (0.061)
Extra Trees Regressor	Ensemble (bagging)	87.502 (3.028)	0.166 (0.069)

Now we have 12 models and performance estimation (RMSE and R^2) for each. From comparing the models we can see that Gradient Boost Regressor has the highest R^2 (0.39) and lowest RMSE (74.3). Below plot compares the spread and the mean RMSE value of each model. There is a population accuracy measure for each model as it was evaluated 10-fold cross validation.

Algorithm comparison - RMSE



Algorithm comparison - R^2



Make Prediction with validation set

Gradient Boost Regressor was the most accurate model among the 12 models tested. On testing this model with our validation set.

Test set rmse: 72.16 is better compared to baseline 94.41
Test set r2: 0.42 is better than baseline 0.00

Refinement [ModelRefinement_GBRegressor.ipynb]

Our selected model, **Gradient Boosting Regressor**(GBR) is a technique that learns from its mistakes. It starts with a bunch of weak learners. Looking at individually, each learner has poor accuracy but together they can have very good accuracy.

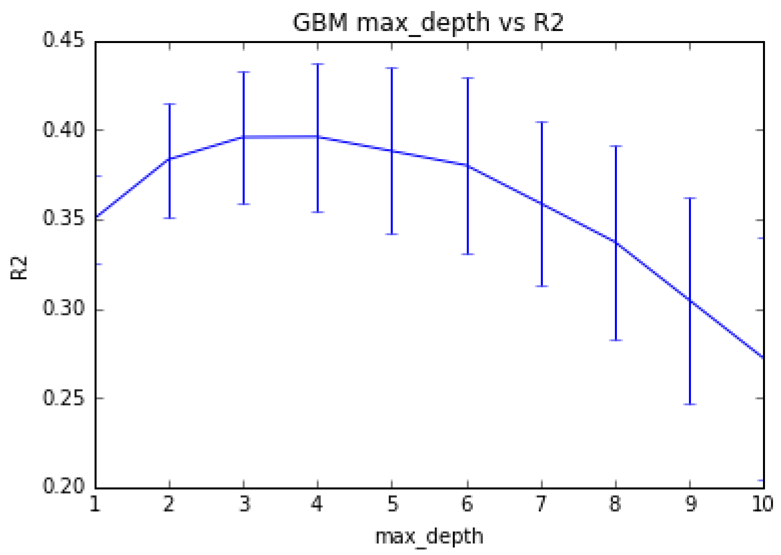
Currently the GBR has better accuracy among the 12 models tested. We can make several modifications to the GBR model which might improve performance.

Gradient Boosting Regressor - Parameters:

max_depth: Controls the number of nodes produced for individual trees. In GBR individual learners are trees.

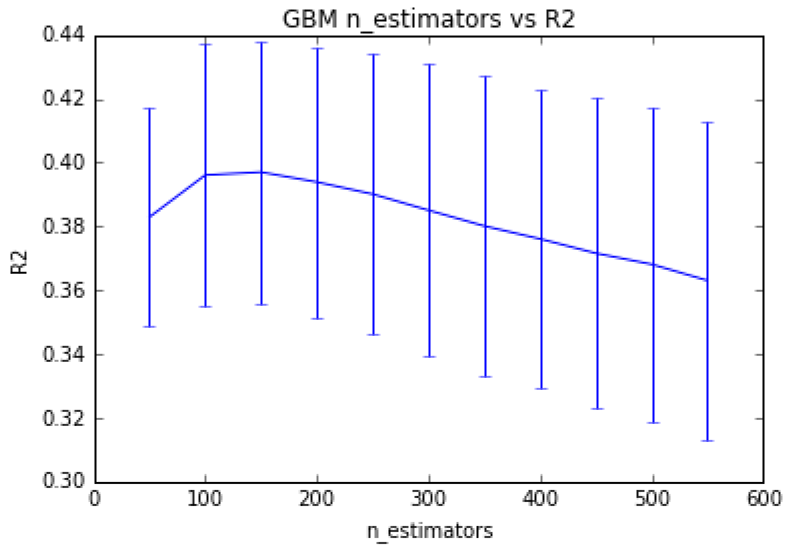
Grid search is performed with 10-fold cross validation to look for the optimal value of max_depth. Results are below:

```
Best: 0.396097 using {'max_depth': 4}
0.350008 (0.024771) with: {'max_depth': 1}
0.383342 (0.031714) with: {'max_depth': 2}
0.395863 (0.037489) with: {'max_depth': 3}
0.396097 (0.041184) with: {'max_depth': 4}
0.388151 (0.046541) with: {'max_depth': 5}
0.380201 (0.049167) with: {'max_depth': 6}
0.358921 (0.045949) with: {'max_depth': 7}
0.337311 (0.054641) with: {'max_depth': 8}
0.304730 (0.057597) with: {'max_depth': 9}
0.272152 (0.067554) with: {'max_depth': 10}
```



n_estimators: The number of weak learners that are used. Higher is better but at the cost of processing power.

```
Best: 0.397017 using {'n_estimators': 150}
0.382914 (0.034040) with: {'n_estimators': 50}
0.396097 (0.041184) with: {'n_estimators': 100}
0.397017 (0.041212) with: {'n_estimators': 150}
0.393913 (0.042431) with: {'n_estimators': 200}
0.390130 (0.044083) with: {'n_estimators': 250}
0.385075 (0.045792) with: {'n_estimators': 300}
0.380137 (0.046892) with: {'n_estimators': 350}
0.376094 (0.046991) with: {'n_estimators': 400}
0.371601 (0.048583) with: {'n_estimators': 450}
0.368139 (0.049196) with: {'n_estimators': 500}
0.363062 (0.049780) with: {'n_estimators': 550}
```



loss: Controls the ‘loss’ function, which determines the error. The ‘ls’ parameter is the default, which stands for least squares. Other options are Least absolute deviation, Huber loss, and qantiles.

```
Best: 0.396097 using {'loss': 'ls'}
0.396097 (0.041184) with: {'loss': 'ls'}
0.396052 (0.036434) with: {'loss': 'lad'}
0.395521 (0.037696) with: {'loss': 'huber'}
-0.591284 (0.114796) with: {'loss': 'quantile'}
```

IV. Results

Model Evaluation and Validation

Verifying with test data

The test data is kept away from the model from the beginning. The test data is underwent the same preprocessing and feature selection process. The trained model, Gradient Boosting Regressor is used to predict the reading scores from the test data. The test set predictions has an RMSE 74.49 and an R^2 score of 0.4.

Justification

Best result achieved so far is with an ensemble model, Gradient Boosting Regressor. Below table shows comparison with the benchmark scores. For the benchmark we used the average of reading scores.

Comparison with baseline performance

Metric	**Baseline**	**Gradient Boosting Regressor**
R^2	0.00	0.40
RMSE	95.79	74.49

The R^2 score is improved from 0 to 0.4. Though an R^2 of 0.4 is not a great score for a model, but in comparison with the benchmark our model shows much better predictability.

V. Conclusion

Reflection

In this project we have worked through regression predictive modeling machine learning problem from end-to-end using Python and scikit-learn.

Here is the summary:

The first step was problem definition. The problem defined is predicting students reading test scores from their demographical information. The two metrics decided to evaluate the algorithms Root Mean Squared Error (RMSE) and R^2. The average score is taken for calculating benchmark RMSE and R^2.

The second step is the exploratory analysis of the data. Different visualizations and statistical summarizations are used to uncover relevant characteristics about the data. We have found about missing values in data. Skewed distributions of some features were identified. The exploratory analysis helped to plan for preprocessing, the next step.

The third step is the preprocessing of data. This included imputing the missing values, encoding categorical features, fixing collinearity and correcting skewness through data transforms.

The fourth step is model evaluation and validation. After preprocessing the dataset is split into training and test sets. It is hard to know which algorithm is best suited for the machine learning problem beforehand. We used trial and error to discover a short-list of algorithms that do well on the problem. Twelve algorithms were tested on the training set using 10-fold cross validation. Gradient Boosting Regressor is selected for the best RMSE and R^2.

The fifth step is the model refinement. In this step, appropriate parameter values found through grid search. The fine tuned model is tested against the validation dataset.

Compared to the benchmark, the model showed significant improvement in RMSE and R^2. The R^2 value is 0.4 and I believe it is not enough to be used in a general setting to solve this type of problems.

Improvement

Feature Engineering is the first thing in mind when thought about improvement. Binning the predictors can be considered. This is done by taking a numeric predictor and pre-categorize or ‘bin’ it into two or more groups prior to data analysis.

Another improvement is to apply different feature selection methods such as SelectKBest, Recursive Feature Elimination (RFE), Principal Component Analysis (PCA) etc to select relevant features.

References

Gareth James, Daniela Witten, Trevor Hastie and Robert Tibshirani. An Introduction to Statistical Learning Ed. 2, Springer, 2015.

Max Kuhn and Kjell Johnson. Applied Predictive Modeling. Springer, 2016.

Stuart Russel and Peter Norvig. Artificial Intelligence A Modern Approach Ed. 2. Pearson, 2011.

Joel Grus, Data Science from Scratch. O’reilly, 2015

Jason Brownlee. Machine Learning Mastery with Python. 2016.

Rachel Schutt and Cathy O’Neil. Doing Data Science. O’Reilly, 2015.

John Paul Muller and Luca Massaron. Python for Data Science. Wiley, 2015.

Sklearn documentation. http://scikit-learn.org/stable/index.html