Test Score Prediction - PISA 2009

Machine Learning Engineer Nanodegree
Capstone Project

Shinto Theruvil Manuel

January 14th, 2016

1. Definition

1.1 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.

1.2 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.

1.3 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 RMSE can be calculated in Python using sklearn:

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

Another common metric is the R^2 (R-squared). It provides an indication of goodness of fit of a set of predictions to the actual values. By definition it is the percentage of the response variable variation that is explained by a linear model. 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. [1]

R-squared does not indicate whether a regression model is adequate. You can have a low R-squared value for a good model, or a high R-squared value for model does not fit the data[2].

R-squared can be calculated In Python using sklearn:

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

[1] https://en.wikipedia.org/wiki/Coefficient_of_determination

[2] http://blog.minitab.com/blog/adventures-in-statistics/regression-analysis-how-do-i-interpret-r-squared-and-assess-the-goodness-of-fit

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

2. Analysis

2.1 Data Exploration

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

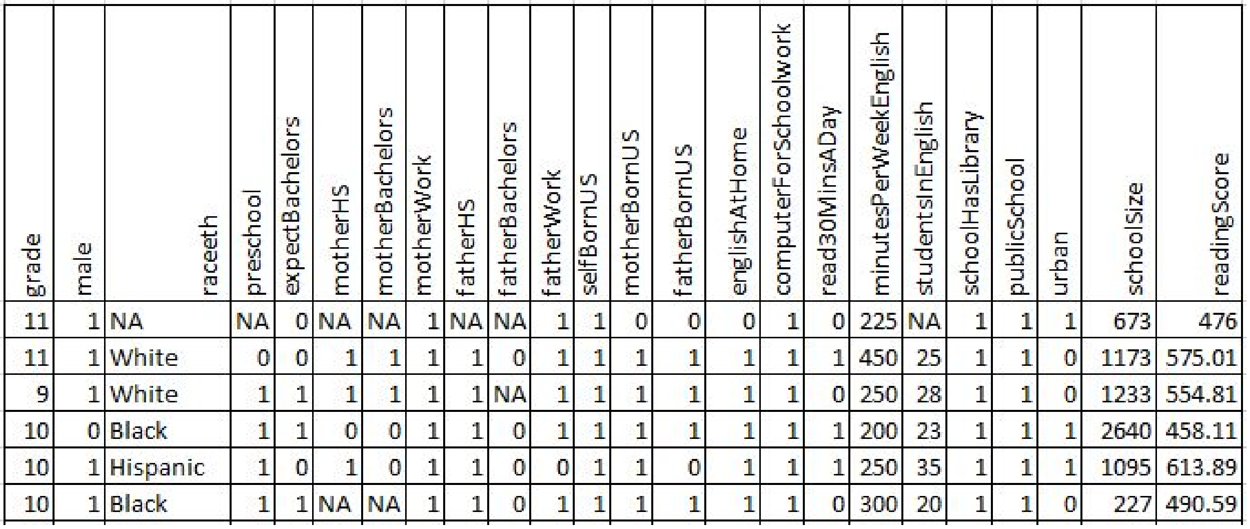
Fields

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

Categorical predictors:

Only one variable, **'raceeth'** is text. We will process this a categorical type.

Continuous predictors:

Among other numerical predictors, three are continuous type. They are:
1) **studentsInEnglish**
2) **minutesPerWeekEnglish**
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. The **grade** can be considered as an ordered categorical predictor.

Description 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. The predictor, ‘studentsInEnglish’ is in the range of 1-90, while another predictor, ‘schoolSize’ is in the range of 100-6694. We are likely going to get better results by rescaling the data in some way in the preprocessing stage. The max value of ‘minutesPerWeekEnglish’ is 2400(40 hours) signalling outliers. The outliers will be investigated further in exploratory visualization.

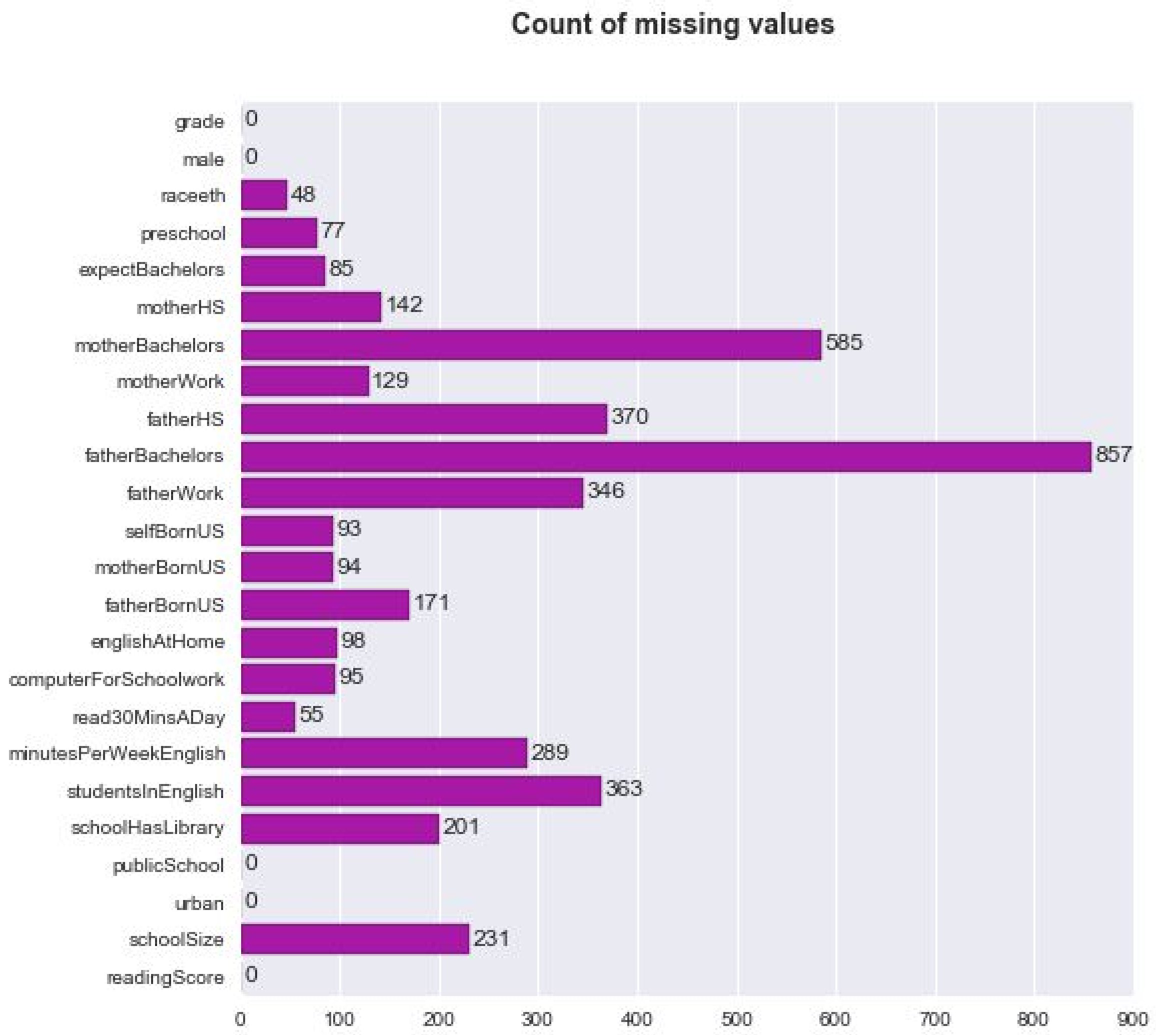
2.2 Exploratory Visualization

Exploratory Visualization is an open process where the user has no set goal and/or is looking for no particular outcome - their intent is to understand their data better and perhaps to satisfy their curiosity[3].

Missing values

In this data many features (predictors) have no values for given sample. It is important to understand the distribution of missing data as it will help to decide on the appropriate preprocessing techniques. For example, the percentage of missing data is substantial enough to remove that predictor from subsequent modeling activities.

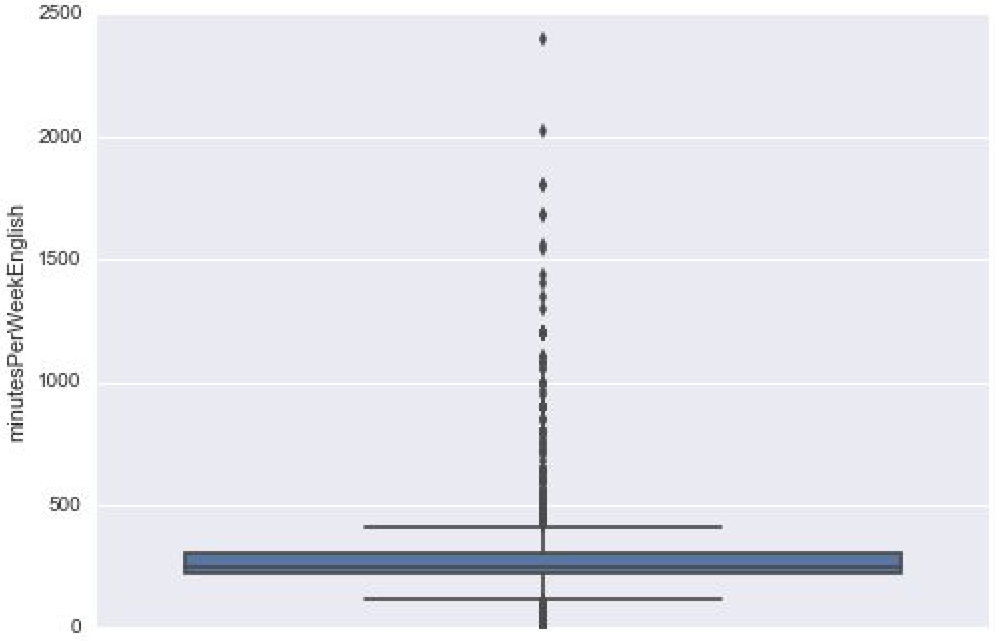
[3] http://www.igi-global.com/dictionary/exploratory-visualization/10619



In the above figure it shows that predictor, ‘fatherBachelors’ (857 samples out of 5233, that is 16%) is having the highest number of missing values, followed by ‘motherBachelors’ (585 samples out of 5233). Also note that four predictors (grade, male, publicSchool, urban) has no missing values.

Detecting Outliers

Descriptive statistics showed signs of outliers in ‘minutesPerEnglish’. An outlier is an observation that is numerically distant from the rest of the data. A boxplot below shows outliers, as data points located outside whiskers:



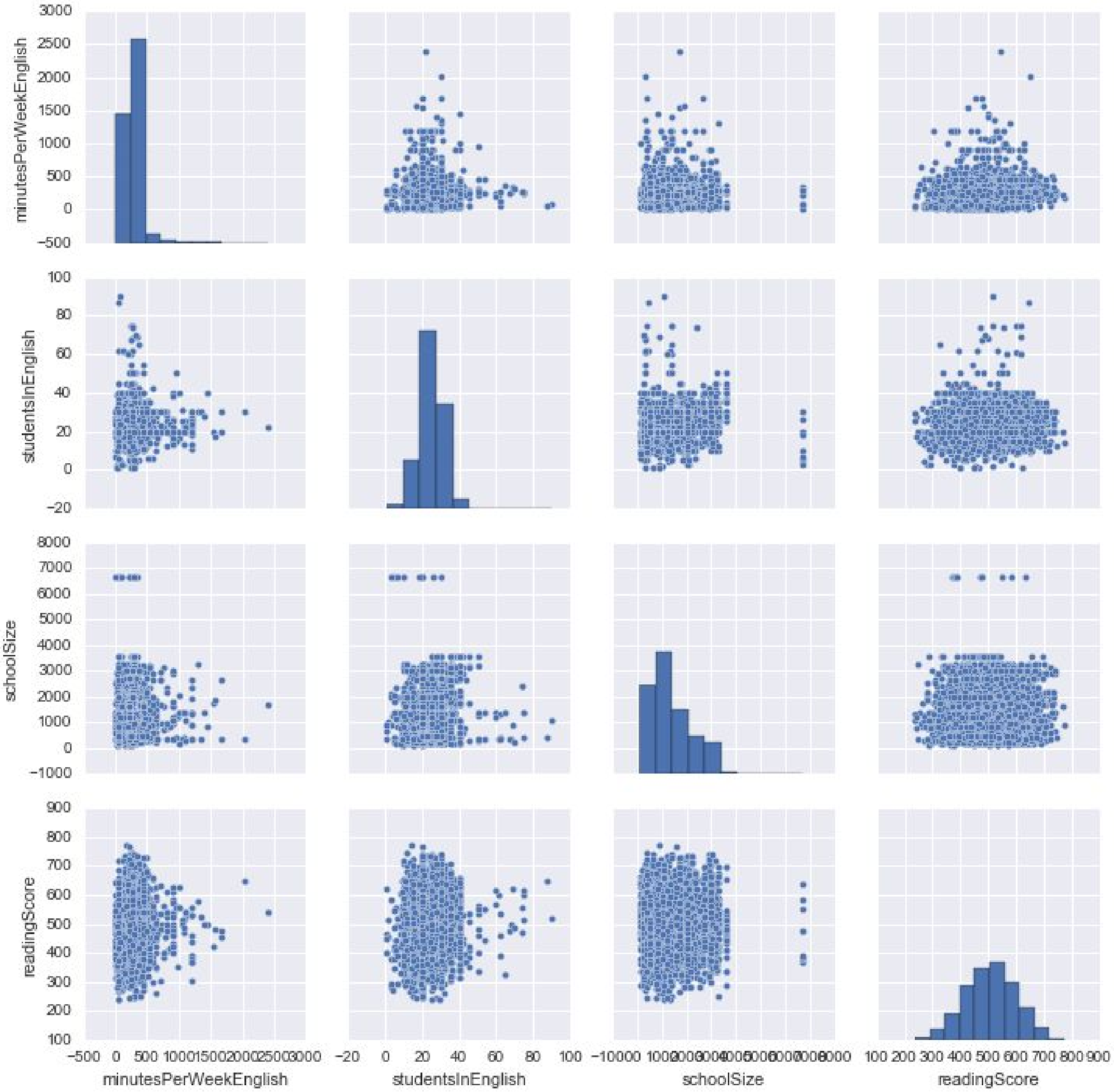
It’s unlikely that a student spend 30-40 hours per week in English classes. So these outliers are assumed to be errors or mistakes and need to be handled in preprocessing.

Visualizing pairwise relationships - continuous predictors

Below diagram shows pairwise bivariate distributions of continuous variables in our dataset. Univariate distribution of each variable is shown on diagonal axis.

A quick glance at univariate distribution at the diagonal axis shows that the target variable (readingScore) has normal distribution. Other two variables, minutesPerEnglish and schoolSize are right skewed. Data transformation at the preprocess stage would help change this to a normal distribution..

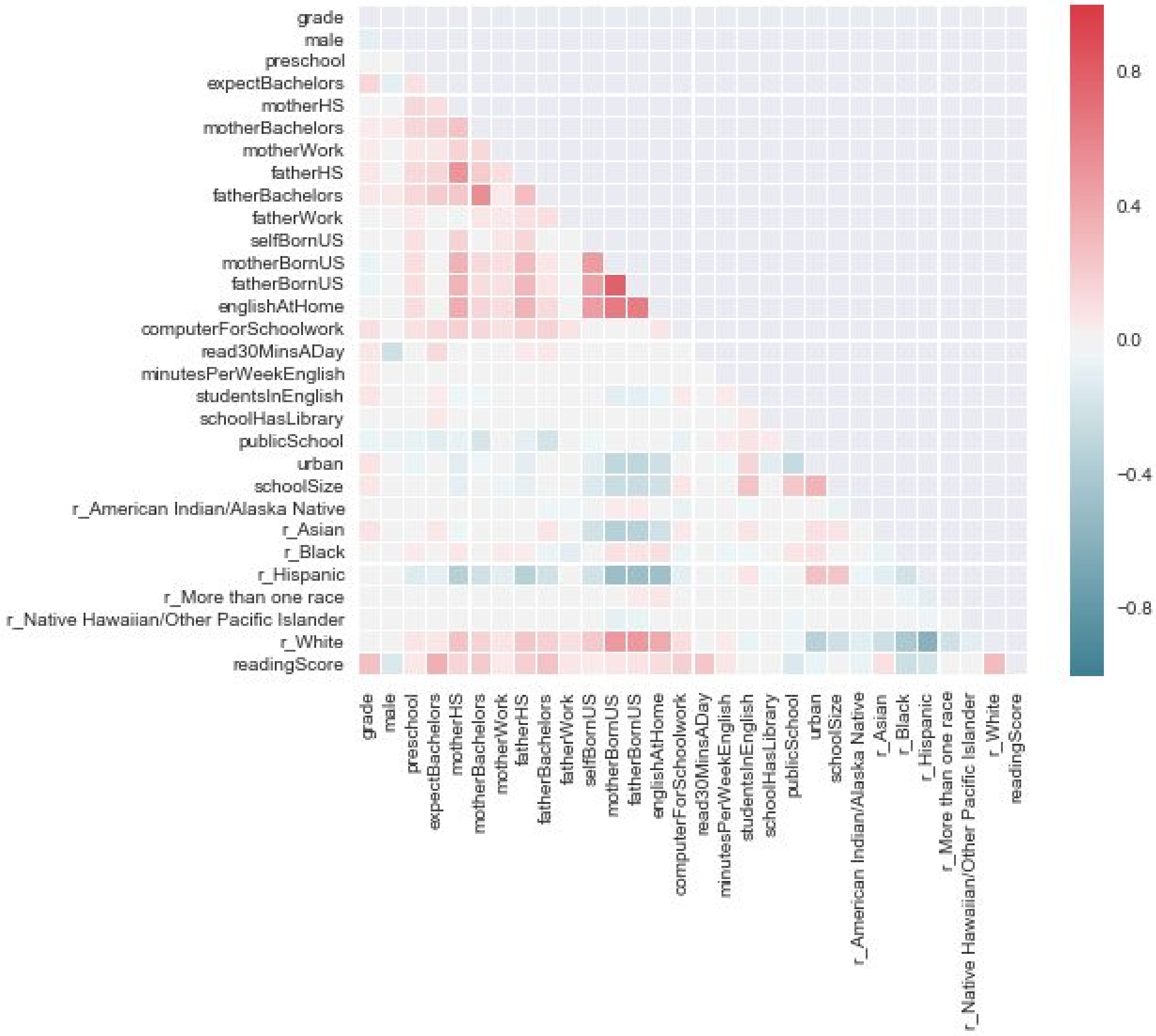
Importantly, the bivariate plots does not show a linear relationship. The large amounts of scatter indicates weak pairwise relationships among these four variables.



The relation between the target and the three predictors is almost zero (neither positive nor negative). Though it looks like these three predictors do not contribute to our prediction much, it says there is no clear linear relationship between our target and predictors.It is just an indication that linear models may not do well. Let us see correlations in the data next.

Correlations in data

The heatmap below shows each pairwise correlation computed and colored according to its magnitude. Please note that the category predictor, ‘raceeth’ is converted to dummy variables to include that as well. In the below heatmap dark red colors indicate strong positive correlations, dark blue is used for strong negative correlations, and white implies no empirical relationship between the predictors.



In the heatmap we can see that that dark red is more often between predictors. For example, there is a high correlation between fatherBornUS and motherBornUS. This situation is called collinearity, where a pair of predictor variables have substantial correlation with each other.

In other words, both variables provide the same information and using highly correlated predictors in techniques like linear regression can result in highly unstable models, numerical errors, and degraded predictive performance[4]. We will handle correlated predictors in preprocessing.

[4] Max Khun and Kjell Johnson, Applied Predictive Modeling, Springer 2016. pg.46

2.3 Algorithms and Techniques

Following are the some specific characteristics from the Analysis phase and require specialized handling:

1. Missing values
2. Outliers
3. Skewness
4. Collinearity
5. Categorical data

Handling Missing Values

Missing values are commonplace in the real world of data analysis. There are different ways to handle missing values.

a) **Delete samples with missing data**: With 1829 observations missing value, we will be losing 35% of data.

b) **Delete features with missing data**:There are only 4 features without any missing data and all other 19 features need to be discarded.

c) **Educated guess**: All the fields that missing value are from the survey questionnaire filled by students and school. Teenage student who has immigrant parents may choose not to answer the fields like ‘fatherBornUS’, ‘selfBornUS’. Likewise students who left out the field ‘fatherBachelors’ might have parents who are not having Bachelors. Considering this we can guess the value 0 for imputing fields such as fatherBornUS, motherBornUS, selfBornUS, fatherHS, motherHS, fatherWork, motherWork, englishAtHome, computerForSchoolWork, read30MinsADay, and schoolHasLibrary.

d) **Average imputation**: Use the average value of other students to fill in the missing value. We will use this approach for the continuous fields such as minutesPerEnglishWeek, studentsInEnglish, and schoolSize.

e) **Common point imputation**: It replaces missing values the most common value.

f) **Regression substitution**: Regression substitution predicts the missing value from the other values. We have 19 predictors with missing data and 19 different models. So for now we not using this method.

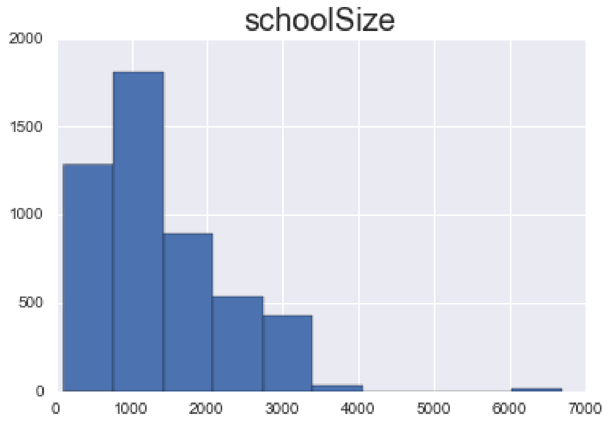
Handling Outliers

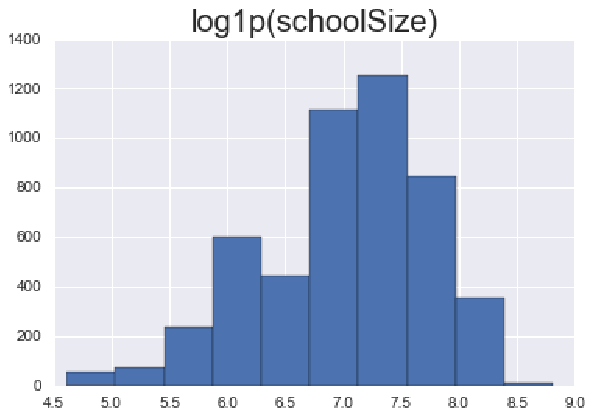
Outliers are seen in feature, ‘minutesPerWeekEnglish’. A value above 400 minutes of English per week is treated as outliers. It is not an invalid value but not normal. We use binning to convert this feature from continuous type to categorical type. All outliers (value above 400) will be put in a separate bin so the outliers will be capped to max value of 400 minutes.

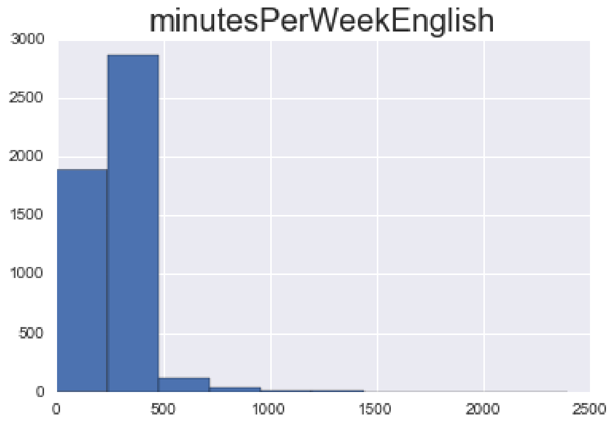
Handling Skewness

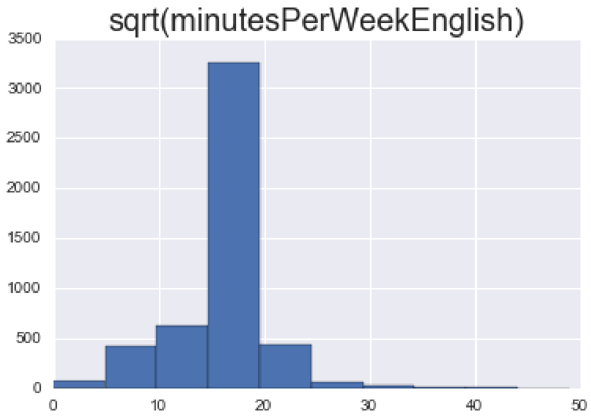
The two features, **‘schoolSize’** and **‘minutesPerWeek’** do not have the bell shaped normal distribution. They skewed and we can use transformations to reduce the skewness.

For schoolSize log1p transform is used and for minutesPerWeekEnglish, square root transform is used.







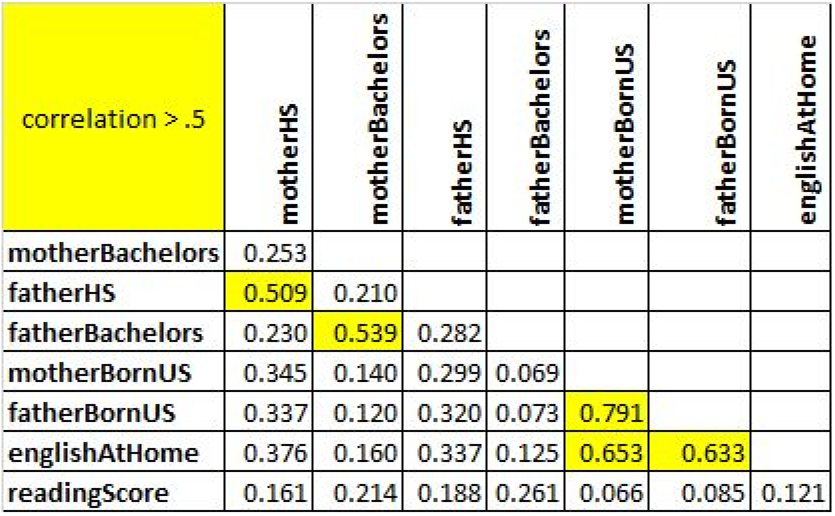


Handling Collinearity

. 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, and other is not used.

So from the above table following four are retained:

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

And the following three features are removed :

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

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.

Algorithms

The problem is to predict the target variable, ‘readingScore’ using a set of predictors. So it is a supervised learning problem. The target variable is a continuous value so we will be using regression algorithms.

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 algorithms considered are:

- Linear Regression.
- Ridge.
- Support Vector Regressor.
- Gradient Boosting Regressor.
- Random Forest Regressor.

1. **Linear Regression:** Most basic commonly used linear mode, 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(multi-collinearity). This algorithm is used as the benchmark.

2. **Ridge Regression:** Closely related to linear regression, Ridge Regression is technique for analyzing multiple regression data. It 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.

   Hyper parameters:

   i) alpha: Regularization strength.

3. **Support Vector Regressor**: Initially developed for binary classification Support Vector Machines were extended for regression type problems. They are based on the concept of decision planes that decision boundaries

   Hyper parameters:

   i) kernel: Kernel to be type used by the algorithm
   ii) C: Penalty parameter

4. **Gradient Boosting Regressor**: Identifies difficult observations by large residuals computed in the previous iterations. In other words, Gradient Boosting is a technique that learns from its mistakes.

Hyper parameters:

i) learning_rate: rate that shrinks the contribution of each tree.
ii) max_depth: maximum depth of individual estimators.
ii) n_estimators: The number of boosting stages to perform.

5. **Random Forest Regressor**: Ensemble of decision trees. A decision tree use the training data to select the best points to split the data in order to minimize a cost metric. Random Forests make predictions by combining decisions from a sequence of base models. These models are constructed independently using a different subsample of the data.

Hyper parameters:

i) max_depth: maximum depth of individual estimators.
ii) n_estimators: The number of boosting stages to perform.
iii) min_samples_split:
iv) min_samples_leaf:
v) max_features:

We will use a test harness with 10-fold cross validation. We will evaluate algorithms using the Root Mean Squared Error (RMSE) and R-squared metric.

2.4 Benchmark

Benchmark is a standard against which we compare results (RMSE and R^2 in our case), to get a feel if the solutions are better or worse. Usually the benchmark is based on some previous work. Since no previous works with this dataset are found, a Linear Regression model is built to be used as a benchmark. The dataset is divided into training and test sets. The results from the benchmark model on the test set is as below:

Benchmark RMSE = 78.36
Benchmark R^2 = 0.32

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

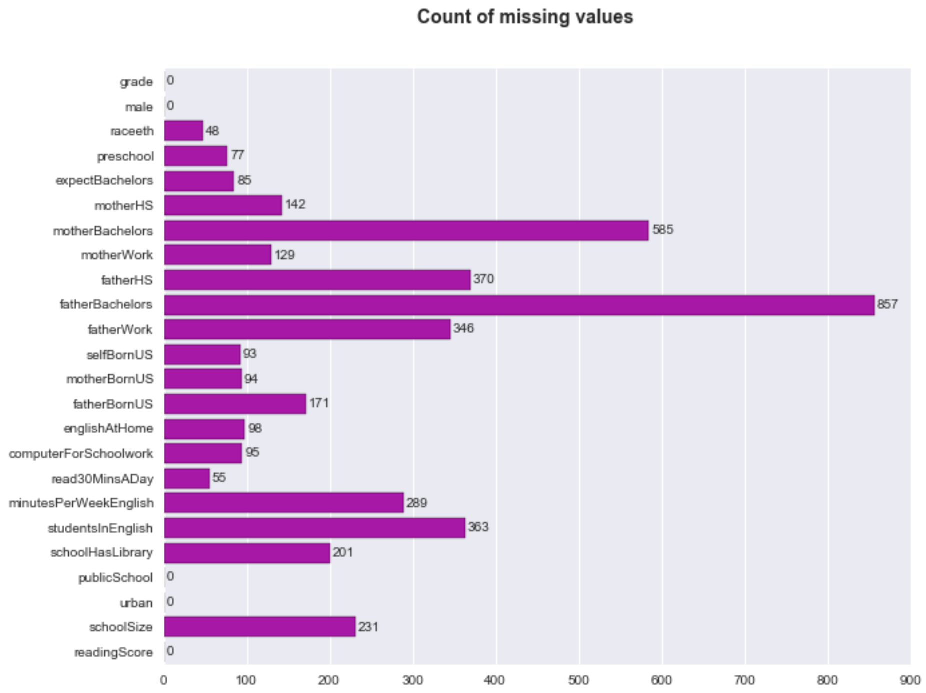
3. Methodology

3.1 Data Preprocessing

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

Binning Continuous Features and handling outliers

Binnig is the process of converting from numerical to categorical variables. This useful feature engineering technique can not only help to interpret and visualize the numerical variable, but also can add additional feature that could increase the performance of the predictive model by reducing noise or non-linearity.

1. **minutesPerWeekEnglish** binned into 6 categories: '<100', '100-200', '200-300', '300-400', '400-2500'.
2. **studentsInEnglish** binned into 6 categories: '<10', '10-20', '20-30', '30-40', '40-50', '50-100'.
3. **schoolSize** binned into 6 categories: '<100', '100-500', '500-1000', '1000-2000', '2000-3000', '3000-7000'.

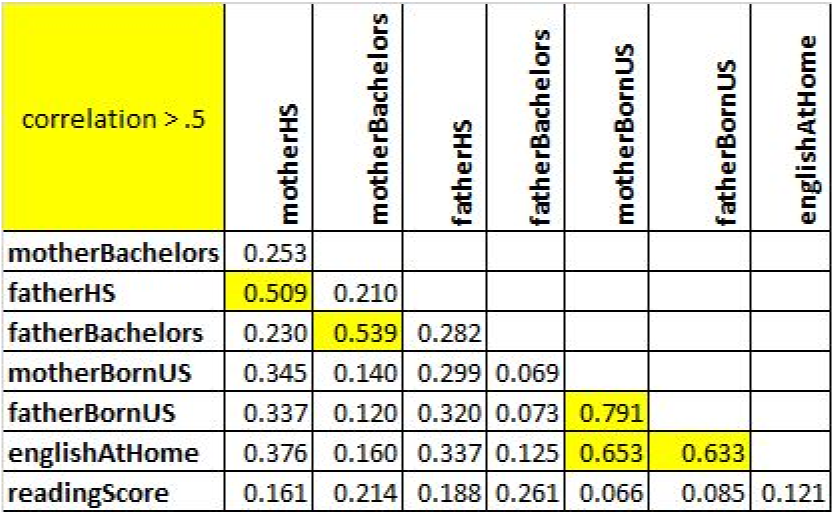
Handling Categorical values - dummy coding

Dummy coding the process of recoding the categorical variable into a number of separate, dichotomous variables. Our data has only one original categorical variable, "**raceeth**". After binning the continuous features in the previous step, we have three more categorical variables from **minutesPerWeekEnglish**, **studentsInEnglish** and **schoolSize**. All the four categorical features are dummy coded into several new dichotomous features.

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

Since all the continuous features are binned into categorical features, no skew correction is required.

Train / Test Split

The dataset is split into 80% training data and 20% test data so that we can test our model against unseen data.

3.2 Implementation

Finding the right algorithm is not an easy job. We will spot check different algorithms with 10-fold cross validation and observe the learning curves.

Following helper functions are used:

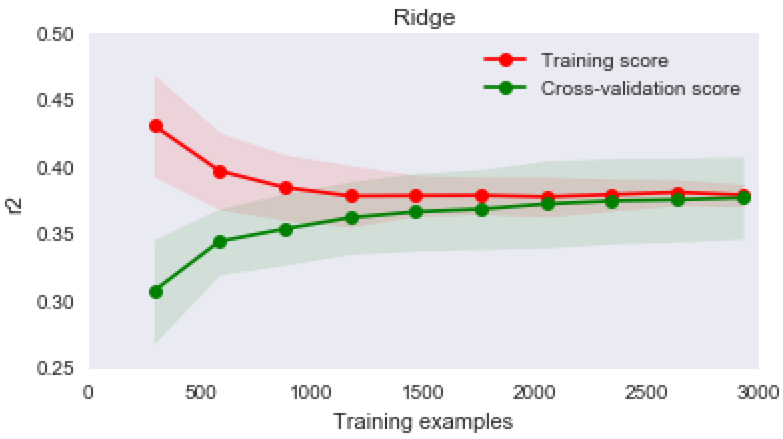
Function	**Description**
cross_validate_models	Perform 10-fold cross validation on multiple models. Prints the RMSE and R2 scores of each model for comparison.
test_set_validation	Train the models on training set and tests the trained models on validation data.

plot_learning_curve	Plot the cross validation and training scores against different number training data used.
plot_validation_curve	Plots training and test scores for varying parameter values.
grid_search_validation_curve	Perform grid search for a varying parameter and plots the validation curve.
select_hyperparams	Finds the best hyperparameters for the model using grid search with 5x2 cross validation.

Learning curves

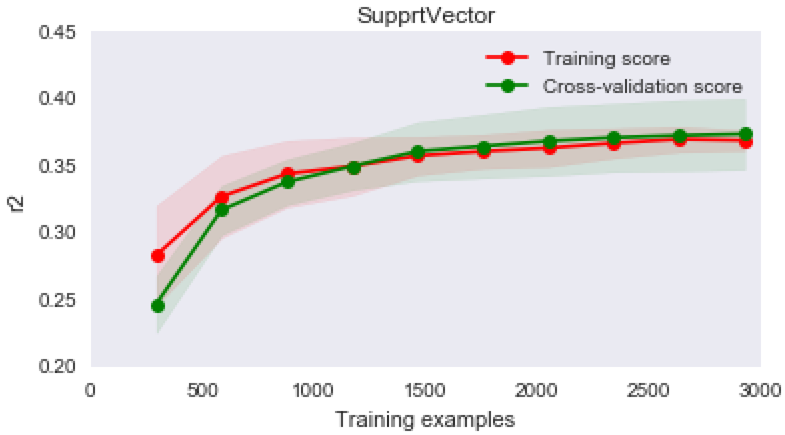
A learning curve shows the training scores and the test scores for varying number of training samples. It is a tool to find out how much we benefit from adding more training data and whether the estimator suffers more from a variance error or bias error[5].

Bias and variance are inherent properties of estimators and our goal is to select estimators and their hyperparameters so that both bias and variance are as low as possible.

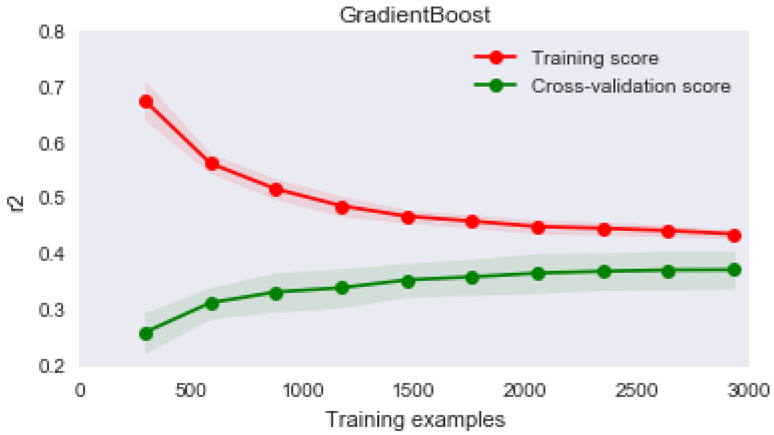


The Ridge model initially started with high variance and as more sample added the variance got reduced.

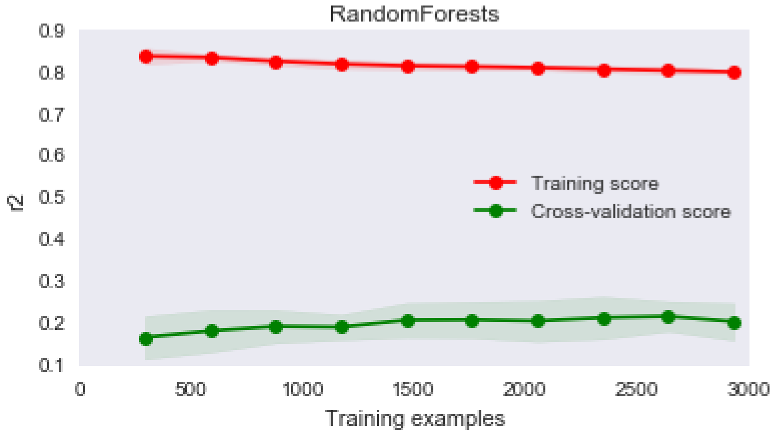
[5] http://scikit-learn.org/stable/modules/learning_curve.html



The SVM model started with high bias and low variance. The bias got improved as more samples added. The training and test curves got converged much early.



The GradientBoost started with high variance, showed underfitting. The variance got reduced as more samples added. The curves are still not met with 3000 samples. Still this is the model with best scores.



Clear overfitting is shown by RandomForest. It looks like that adding more training data is not going to help this model. The high bias will not be fixed by more training samples because the underlying model is not able to approximate the correct function.

Make Prediction with validation set

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

Test set rmse: 72.16 is better compared to baseline 78.36
Test set r2: 0.40 is better than baseline 0.32

3.3 Refinement

The refinement of models consists of two parts:

1) Feature Selection
2) Hyperparameter tuning

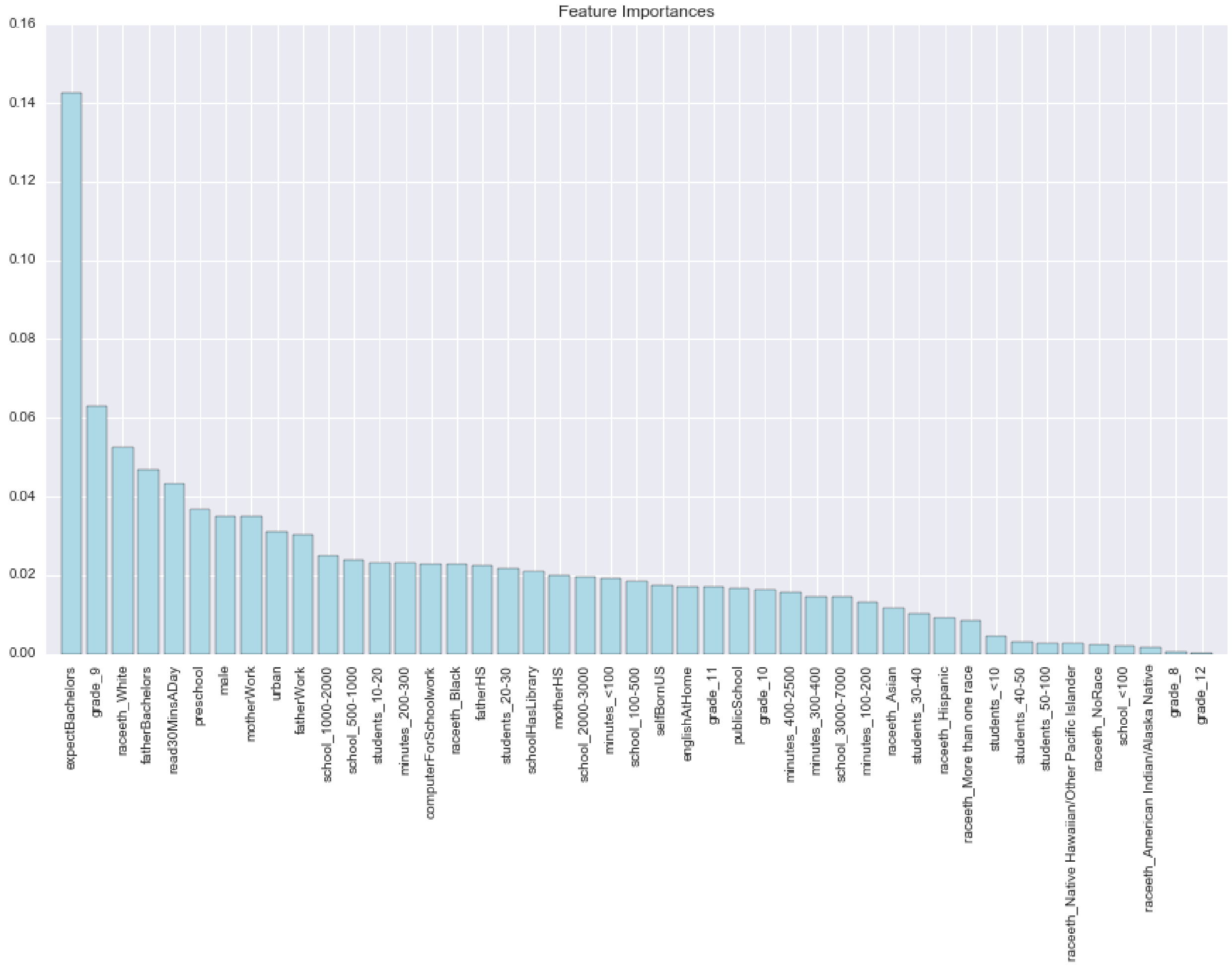
3.3.1 Feature selection

It is the process of selecting a subset of relevant features(variables, predictors) for use in model construction[6]. More often a small subset of features give better performance than whole set of features. Also irrelevant features can negatively impact model performance. A small set of features enable machine learning algorithm to train faster. Proper selection of features reduces the complexity of a model and makes it easier to interpret.

a) **Feature importance through RandomForests**

[6] https://en.wikipedia.org/wiki/Feature_selection

We have a total of 45 predictors. RandomForest is used to estimate importance of these predictors(features) and the importance is plotted below.



Top 25 features from RandomForest feature importance are selected and a 10-fold cross validation is performed on 4 models.

b) SelectKBest

SelectKBest is used to remove all but the 25 highest scoring features. A 10-fold cross validation is then performed to assess the performance on each of the four models.

c) Principal Component Analysis (PCA)

It is a statistical procedure to convert a set of observations of possibly correlated variables into a set of linearly uncorrelated variables called principal components. PCA is used to select 25 components and a 10-fold cross validation is then performed to assess the four models.

d) Recursive Feature Elimination(RFE)

RFE works by recursively removing features and building a model on those features that remain. It uses the model accuracy to identify which combination of features contribute the most. RFE is used to select the top 25 features then a 10-fold cross validation is performed to assess the models.

Among the above four methods of feature selection it is found that RFE provided the best scores of RMSE and R2. In a grid search manner RFE is tried with different number of components and it is found 27 features provided the best score of RMSE and R2.

The feature selection helped to reduce the number of features from 45 to 27 without compromising RMSE and R2 scores.

3.3.1 Hyperparameter tuning

Machine learning models are parameterized so that their behavior can be tuned for a given problem[7]. Models can have many parameters and finding the best combination of parameters can be treated as a search problem.

Separate grid searches is performed for narrow down the range of each hyperparameters. Once an optimal range of values for each parameter is found a combined grid search is performed to select the best combination of parameters. This process is repeated for each of the four models and the optimized parameters are as below:

Gradient Boosting Regressor - optimized parameters:

learning_rate=0.1
max_depth=2
max_features='sqrt'
min_samples_leaf=8,
min_samples_split=3
n_estimators=250
subsample=0.6

Support Vector Regressor(SVR)- optimized parameters:

C=3
epsilon=14

Ridge Regressor - optimized parameters:

alpha=15

Random Forest Regressor- optimized parameters:

max_depth=8
max_features='sqrt'
min_samples_leaf=1

[7] http://machinelearningmastery.com/how-to-tune-algorithm-parameters-with-scikit-learn/

min_samples_split=21
n_estimators=250

After hyperparameter tuning, the scores of all the four models came close. Gradient Boosting Regressor found to be having the best scores of RMSE and R2. Significant gain is observed in the performance of Random Forest algorithm.

4. Results

4.1 Model Evaluation and Validation

Validation with unseen 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. All four models, after preprocessing, refinement, tested with a hold out dataset, the test set. The performance is as in the below table:

Model	**RMSE**	**R2**	**Time taken**
GradientBoost	74.52	0.386	0 seconds
SVR	75.00	0.378	0 seconds
Ridge	75.01	0.378	0 seconds
RandomForest	75.29	0.373	1 second

4.2 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 simple Linear Regression.

Comparison with baseline performance

Metric	**Baseline**	**Gradient Boosting Regressor (Final Model)**
R^2	0.32	0.39
RMSE	78.36	74.52

The R^2 score is improved from 0.32 to 0.39. Though an R^2 of 0.39 is not a great score for a model, but in comparison with the benchmark our model shows much better predictability.

Cross validation RMSE changes during different stages

	Preprocessing (45 predictors)	Feature Selection (27 predictors)	Hyperparameter tuning (27 predictors)
Ridge	76.223 (3.725)	76.102 (3.509)	75.849 (3.622)
SupportVector	76.372 (3.559)	76.330 (3.412)	76.084 (3.453)
GradientBoost	76.118 (3.663)	75.833 (3.531)	75.849 (3.622)
RandomForests	85.190 (3.407)	85.500 (3.310)	76.984 (3.734

Cross validation R2 changes during different stages

	Preprocessing (45 predictors)	Feature Selection (27 predictors)	Hyperparameter tuning (27 predictors)
Ridge	0.363 (0.035)	0.365 (0.033)	0.365 (0.031)
SupportVector	0.360 (0.030)	0.361 (0.027)	0.365 (0.031)
GradientBoost	0.365 (0.030)	0.369 (0.032)	0.369 (0.032)
RandomForests	0.199 (0.057)	0.189 (0.049)	0.351 (0.023)

Test Set RMSE changes during different stages

	Preprocessing (45 predictors)	Feature Selection (27 predictors)	Hyperparameter tuning (27 predictors)
Ridge	75.09	75.09	75.01
SupportVector	75.12	75.20	75.00
GradientBoost	73.95	73.97	73.97
RandomForests	81.45	84.95	75.29

Test Set R2 changes during different stages

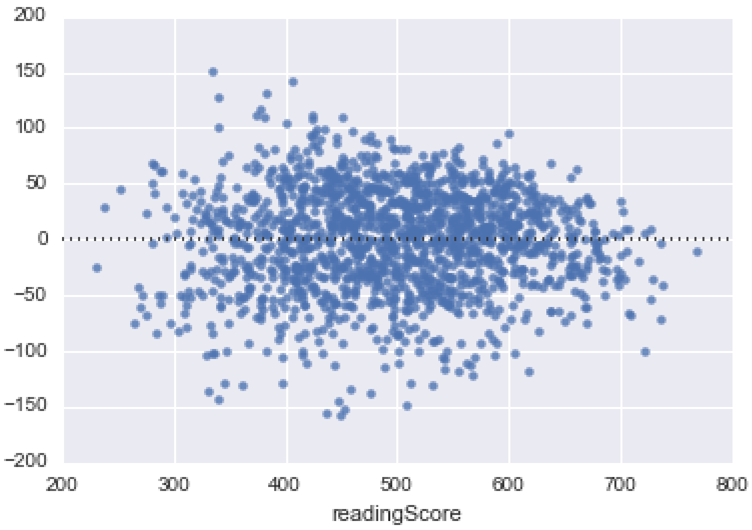
	Preprocessing (45 predictors)	Feature Selection (27 predictors)	Hyperparameter tuning (27 predictors)
Ridge	0.38	0.376	0.378
SupportVector	0.38	0.375	0.378
GradientBoost	0.40	0.395	0.386
RandomForests	0.27	0.202	0.373

5. Conclusion

5.1 Free-Form Visualization

Residuals plot

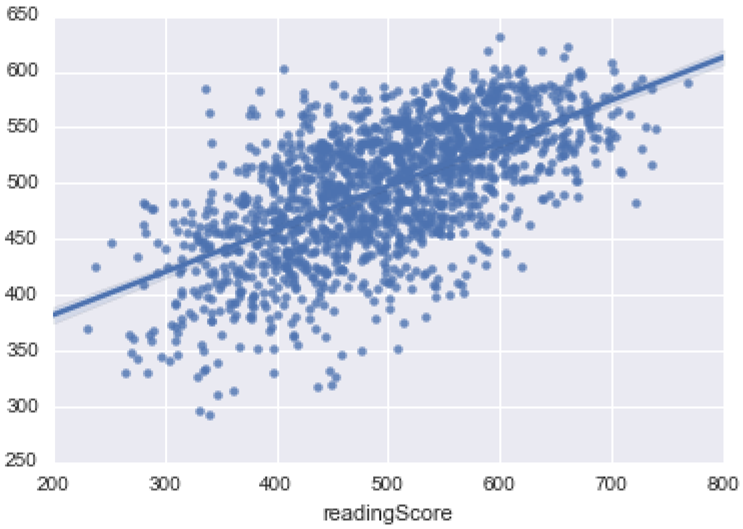
Residuals are a sum of deviations from the regression line. Because a linear regression is not always the best choice, residuals help to figure out if the regression model is a good fit for your data



The above residual plot shows residuals on the vertical axis and the target variable, readingScore on the horizontal axis. The above residual plot shows a random pattern. A line of best fit is a good approximation of the data.

Actual vs Predicted Plot

Scatter plots of Actual Vs Predicted are useful in analyzing the performance of the model. Ideally, all points should be close to a regressed diagonal line



5.2 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. A simple Linear Regression model is used as the benchmark..

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, handling outliers, binning continuous predictors.

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. Four 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, feature selection and tuning of hyper parameters were done.

The feature selection through Recursive Feature Elimination (RFE) helped to reduce the number of features from 45 to 27 without compromising the performance of the models.

All the four fine tuned models are tested against the validation dataset.

Compared to the benchmark, all the four models showed improvement in RMSE and R^2. Gradient Boosting Regressor is selected for the best RMSE and R^2. The R^2 value is 0.39 improved from the baseline value of 0.32. The RMSE is improved from 78.36 to 74.52. Though feature processing and parameter tuning made some improvements in the model performance I still believe it is not enough to be used in a general setting to solve this type of problems.

5.3 Improvement

Feature Engineering is the first thing in mind when thought about improvement. Creating some interaction features can be considered. This is done by taking a numeric predictor and multiplied by another predictor. Multiplication is just one type of interaction. Another example is adding a quadratic term.

Another attempt is to create an ensemble of different models, and take the average of their predictions..

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