Investigative Data Analysis and Predictive Modelling of GCSE Ethnicity Dataset

Fawwaz Chowdhury
Group size: 3

Student	Effort
Fawwaz Chowdhury	33%
Matthew Bieda	33%
Nabila Alli	33%

GCSE Data Exploration

Fawwaz, Matthew, Nabila

I. INTRODUCTION

The academic curriculum of schools in England makes it easier for students to narrow their future options at an early age. Students normally take their GCSE exams in secondary school usually occurring in year 10 and 11. The results are normally an indication to which sixth form a student can go to and can also influence a person's desired career choices. However, given the coronavirus pandemic the government were then forced to cancel any exams and replace them with assessments and algorithms given by teachers.

Even though this seemed like a suitable approach at first, the results could have had a bias effect on students from disadvantaged backgrounds as the situation shines light on long-term flaws in the assessment, exams and admission systems that systematically disadvantage pupils from certain groups. Mr Pran Patelr, who's an equity activist refers back to a 2009 study which found that Pakistani pupils were 62.9% more likely than white pupils to be predicted a lower score than they achieved in their Key Stage 2 English. But some other educators found this fact contradicting as a research done by Kaili Rimfeld at King's College London based on data with >10,000 pupils found that teacher assessments are generally good predictors of future exam performance.

It has also been known that due to fears over grade inflation caused by teachers assessing their own students, the marks are not being used in isolation. Because of the coronavirus situation, those potentially biased teacher assessments were modified, considering the school's historical performance and not the individual student's performance. This means that a pupil in a poorly performing school may have seen their grades downgraded because last year's pupils did not perform well in their exams. "Children from a certain background may find their assessment is downgraded," says Stephen Curran, a teacher and education expert. This is what happened in Scotland, where children from poorer backgrounds were twice as likely to have their results downgraded than those from richer areas [1].

This data analytic report aims to identify the ways in which a pupil's ethnicity/race, and socio-economic background will influence their achieved GCSE results. By thoroughly assessing and analysing our dataset, this would allow us to identify whether demographic does play a role in GCSE results.

OBJECTIVES

-Identify the extent to which ethnicity/race and socio-economic background influence a pupil's achieved GCSE results

-Conduct Exploratory Data Analysis to find how Ethnicity and Gender in the dataset relate to Attainment and Progress of pupils

-Use a minimum of two classification/prediction models to deliver an outcome.

II. LITERATURE REVIEW

The report "A compendium of evidence on ethnic minority resilience to the effects of deprivation on attainment" by *Stokes, Rolfe, Hudson- Sharp* and *Stevens* [2], discusses further on this topic. The report talks about how there are ways to raise ethnic minority attainment of disadvantaged pupils from all ethnic groups.[2] The report listed importance of high-quality school leadership, a school ethos that embraces diversity and has high expectations of all pupils, monitoring and tracking of pupils, a flexible and inclusive curriculum, and engagement with parents and the wider community as effective ways.[2] Another study by *Roberts and Bolton* [3], shows that across the black major ethnic group, 59% of pupils attained a standard pass in English and Maths GCSE in 2018/19. The report also concluded that white pupils make the least progress of any major ethnic group, at -0.11. Black pupils typically make more progress, at +0.13, but lag behind Chinese pupils (+0.86) and Asian pupils (+0.47). Another study which looked at gender disparity within GCSE grades is "Gender differences in GCSE" by *Bramley, Vidal Rodeiro and Vitello* [4]. The study found that females performed better than males when it came to English GCSE, by on average, half a grade. However, no gender differences were found in GCSE mathematics, although females performed better in mathematics and science in primary school. The study "The effects of social class and ethnicity on gender differences in GCSE attainment: a secondary analysis of the Youth Cohort Study of England and Wales 1997– 2001" partaken by *Conolly* who reported that "*while girls have been about one and a half times more likely to gain five or more GCSE grades A*-C than boys, those from the highest social class backgrounds have been between five and nine times more likely than those from the lowest social class backgrounds"* [5] is adherent to the latter study. The gender disparity may also be linked to other factors like curriculum choices said in "Explaining Sex Differences in Educational Choice: An empirical assessment of a rational choice model" by Jonsson [6], who argues that gender difference may persist because boys have an advantage in technical subjects e.g., engineering while girls perceive a relative advantage in humanities subjects e.g., languages, making them a more attractive choice of learning to each gender. Last point to make is the influence of school experiences to GCSE outcomes. “Influences on students’ GCSE attainment and progress at age 16” by *Sammons et al*., [7], speaks more on this topic by stating that students GCSE attainment and progress were increased if they attended a secondary school rated as having a more favorable overall school ‘behavior climate’ in KS3 which means a greater progression rate. The highlighted effects were particularly noticeable for Maths and English grades and the number of full GCSE entries. There needs to be a further exploration in this area of research as there seems to be several worthy literatures to back the possibility of various external influences in GCSE grading system.

III. DATA MANAGEMENT

Data Source and Description

The dataset used in this report was collected from London dataset. Data retrieval was based on GCSE results by Ethnicity. The dataset initially had 8 columns along with the target column. The column feature consists of: “Code”, “Area”, “Year”, “Sex”, “Ethnicity”, “Pupils”,” Attainment8” and “Progress”. The Code column describes the area code the pupils attend school at e.g., E09000002. The Area is the area/location of the pupil’s school e.g., Bromley. The Year describes the year the GCSE results were taken, this could vary from 2015/2016 to 2018/2019 which is when the data was collected. The Sex is the grouped into two main categories: Female and Male. The ethnicity collected in the data is based on these races: Asian, Black, Chinese, Mixed and White. Attainment8 is results from the measurement of a pupil’s average grade across 8 subjects. Pupil8 is a value-added measure, that measures how much value a school is adding compared to other schools.

The first 4 rows of the dataset are shown below in the table:

Data Cleaning

To be able to properly perform the analysis, various data cleaning techniques had to be carried out. This is a crucial point in analysis because cleaning datasets by removing duplicates or null values helps remove various errors which could later affect the exploration by producing inaccurate/ bias results and could also degrade the visual results.

The data cleaning methods adopted were:

- Identification of invalid values.
- Replacing any invalid value with “NaN” to transform data.
- Strip any whitespace between values.
- Drop unnecessary column that was not needed for exploratory “Code”, which helped modify the column data.
- Removing outliers with Inter-Quartile range filtering
- Replacing NaN values with imputed median values. (This can be justified in our data set because we have a small fraction of NaN values and our data follows a normal distribution.)
- Applying a log transformation in our data to make it more closely follow a normal distribution. This makes statistical analysis simpler and improves the accuracy of some machine learning algorithms.

Data Representation

Python was used as the main language for the data manipulation and analysis. Python is a high-level programming language which contains easily readable syntax to build effective solutions even in complex scenarios.

The following external libraries were used:

- *Pandas:* An open-source python package, one can manipulate and present data. Provides in-memory 2d table object called Dataframe.
- *NumPy:* Library that includes support for large multidimensional arrays and matrices, extensively used as it facilitates advanced mathematical and other types of operations on large numbers of data.
- *Scikit-learn:* A machine learning library in Python that contains a lot of efficient tools for machine learning and statistical modelling including classification, regression, clustering and dimensionality reduction.
- *Matplotlib:* A plotting library for visualisation of data used for 2D graphics, there are different types of graphs that can be developed using this library e.g., plots, charts, etc. It can be used in python scripts, shell, etc.
- *Seaborn:* Data visualization library based on matplotlib in Python. Provides a high-level interface for drawing attractive, sophisticated and informative statistical graphics.

- *Scipy:* A free and open-source library that provides advanced support for scientific computing and statistical analysis.
- *statsmodels.api:* Python module capable of approximations of statistical models, tests and data exploration.

IV. METHODOLOGY

The approach taken for the prediction of student's performance based on various factors analysed in the report uses two predictive models: Simple/Multiple Linear Regression and Random Forest Regression. Regression models were used because the dependent variable (Attainment) is a numeric quality, if it were Categorical we could explore some Classification algorithms such as SVM, KNN and Logistic Regression. These models have various advantages and disadvantages but they were chosen due to the suitability of our dataset, explanation and reasoning behind these models is explained as follows:

Simple/Multiple Linear Regression

Multiple linear regression is has similarity to simple linear regression except that it uses more than one independent variables / large number of predictors and a dependent / response variable by fitting a linear equation to the observed data.

For the simple/multiple linear regression mode has three different hypothesis tests for slopes that could conduct be.

They are as follows:

- Hypothesis test for testing that *one* slope parameter is 0
- Hypothesis test for testing that *all* of the slope parameters are 0
- Hypothesis test for testing that a *subset* — more than one, but not all — of the slope parameters are 0

Area	Year	Ethnicity	Attainment8	Progress8	log Attainment8	log Progress8
Barking and Dagenham	2018/19	White	41.7	-0.21	3.730501	NaN
Barking and Dagenham	2018/19	Mixed	44.7	0.05	3.799974	-2.995732
Barking and Dagenham	2018/19	Asian	53.3	0.60	3.975936	-0.510826
Barking and Dagenham	2018/19	Black	48.0	0.37	3.871201	-0.994252
Barking and Dagenham	2018/19	All Pupils	46.4	0.16	3.837299	-1.832581
...	...	...	...	...	...	...
Yorkshire and the Humber	2016/17	Chinese	60.8	1.16	4.107590	0.148420
Yorkshire and the Humber	2016/17	Mixed	43.8	-0.08	3.779634	NaN
Yorkshire and the Humber	2015/16	Black	46.7	0.28	3.843744	-1.272966
Yorkshire and the Humber	2015/16	Chinese	63.1	0.83	4.144721	-0.186330
Yorkshire and the Humber	2015/16	Mixed	48.1	-0.11	3.873282	NaN

798 rows × 6 columns

A population model for a multiple linear regression model is written as

$$\boxed{Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_i X_i}$$

Y : Dependent variable
β_0 : Intercept
β_i : Slope for X_i
X = Independent variable

Figure 1: Equation for Multiple Regression Model

The aim of this regression is to determine the values of the weights b_0, b_1, and b_2 such that it is as close as possible to the actual responses and yield the minimal sum of squared residuals (SSR) with the Ordinary Least Squares implementation.

Random Forest Algorithm

The random forest algorithm is a popularly known supervised learning algorithm. The "forest" it creates, is a combination of decision trees, trained with the bagging method. The bagging method uses the idea that a combination of learning models have a high chance to increase the overall result. The benefit of using a random forest is that it can be used for any type of classification and regression problems, random forest makes it very easy to measure the relative importance of each feature on the prediction.

In random forest, instead of searching for the most vital feature while splitting a node, it searches for the best feature among a random subset of features. This then results in a wide diversity that generally results in a better model. This means that in random forest, only a random subset of the features is taken into consideration by the algorithm for splitting a node.

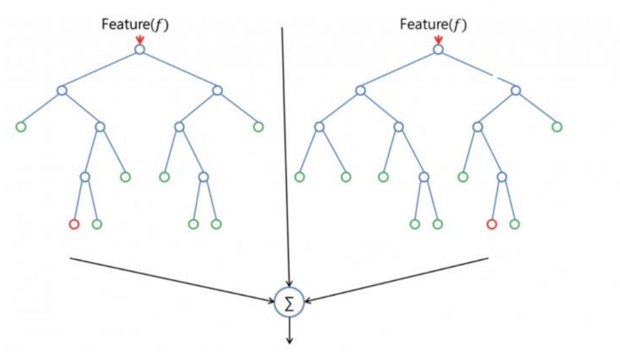
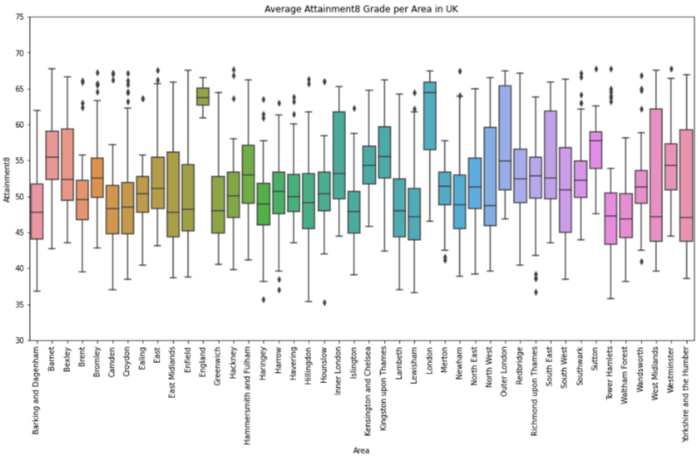


Figure 2: Random Forest Simplified

V. DATA EXPLORATION

Now that we have installed the appropriate analytical packages and decided on a methodology for the machine learning portion, we are ready to begin exploring the data using Python and Jupyter Notebook to see what insights we can gleam. Firstly, we will examine the data and determine how it must be cleaned.



50%	406.000000	50.500000	0.240000
75%	1463.000000	55.300000	0.530000
max	540006.000000	89.000000	2.820000

	Code	Area	Year	Sex	Ethnicity	Pupils	Attainment8	Progress8
0	E09000002	Barking and Dagenham	2018/19	All	White	903.0	41.7	-0.21
1	E09000002	Barking and Dagenham	2018/19	All	Mixed	227.0	44.7	0.05
2	E09000002	Barking and Dagenham	2018/19	All	Asian	530.0	53.3	0.60
3	E09000002	Barking and Dagenham	2018/19	All	Black	642.0	48.0	0.37
4	E09000002	Barking and Dagenham	2018/19	All	Chinese	3.0	69.0	0.93

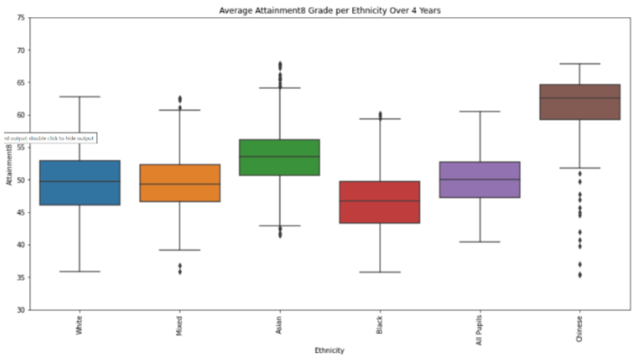
```
 1   Area          3240 non-null   object
 2   Year          3240 non-null   object
 3   Sex           3240 non-null   object
 4   Ethnicity     3240 non-null   object
 5   Pupils        3110 non-null   float64
 6   Attainment8   3106 non-null   float64
 7   Progress8     3048 non-null   float64
dtypes: float64(3), object(5)
memory usage: 202.6+ KB
```

By running some simple calculations, we found there are 4,863,974 GCSE students represented, 2,391,378 females and 2,472,596 males. Next, we checked for null values, finding a negligible amount.

```
Code           0
Area           0
Year           0
Sex            0
Ethnicity      0
Pupils       130
Attainment8  134
Progress8    192
dtype: int64
```

We decided to use IQR filtering to identify outliers. We also experimented with using z-scores but using IQR filtering provided a more normal distribution of the data. Now that we had removed outlying values, we had to deal with the present NaN values. Since these only comprised a few percent of our data they could be simply discarded. Alternatively, we could have used an imputation method such as simple mean/median imputation (both would be similar as we have a normal distribution. However, this would have introduced a spike at certain values in histogram plots. After the removal of outliers and NaN values we had 2423 rows remaining.

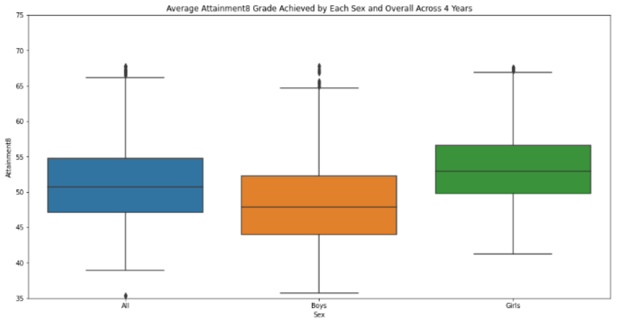
Now we proceeded to create some boxplots using Seaborn to graphically show insights into the data.



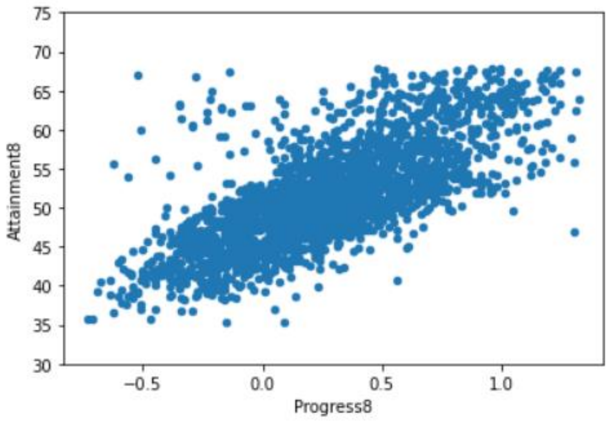
Here we can see that students of Chinese ethnicity significantly outperformed any other group and white students had the largest variation in achievement. Next, we plotted Attainment8 by each area.

We can see from this that despite also containing the area with the lowest median value (Tower Hamlets/Waltham Forest), London is the highest performing region on average. There is a clear correlation between socio-economic status and attainment in London, for example compare Barking and Dagenham versus Kensington and Chelsea.

Then finally we explored a boxplot of attainment versus sex of the student. As is historically known girls outperformed boys at GCSE level.



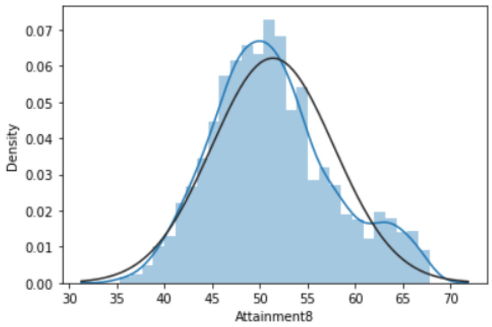
Subsequently we created a scatter plot to explore the relationship between progress and attainment.

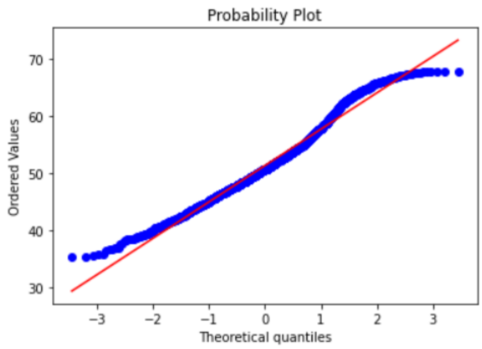


We can see that they are positively linearly correlated, meaning that children who performed well in primary school tend to continue this high performance in secondary education.

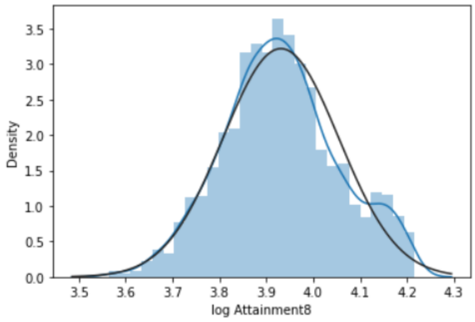
Following this we investigated the distribution of our data with a histogram and probability plot, as this has implications for statistical analysis and machine learning methods. We found our distribution to be essentially normal but imperfect, as can be seen on the probability plot. We then experimented with using a log transform to smooth the distribution, which worked rather successfully.

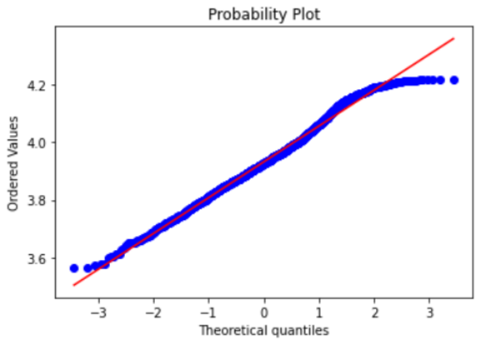
Before log transform:





After log transform:





Transforming Data

Identifying each ethnicity and their Attainment8 and Progress8 scores required data transformation. The initial step was to remove the columns: Code and Pupils as the former was unneeded and the latter is summarised by the Attainment8 and Progress8 columns. The dataset was then indexed by area and filtered for all genders results to be displayed and then the Sex column was also dropped. The dataset was then filtered for a specific ethnicity and their results.

```
In [22]: #Selecting 'White' Ethnicity, dropping 'Ethnicity' column and then Standardising
         #Then dropped 'Year' Column and used groupby for 'Area' column and then took the
         #Standardised values
         df_white_grades = df_all_sex[(df_all_sex['Ethnicity']=='White')]
         df_w_eth_dropped = df_white_grades.drop(columns={'Ethnicity'})
         dfw=df_w_eth_dropped.reset_index()
         dfw1 = dfw.sort_values(['Area', 'Year']).set_index(['Area'])
         dfw2 = dfw1.rename(columns={'Attainment8':'W Att8', 'Progress8':'W Prog8' })
         df_w_mean = dfw2.drop(columns={'Year'}).groupby('Area').agg({'W Att8':'mean', 'W

In [23]: #Selecting 'Black' Ethnicity, dropping 'Ethnicity' column and then Standardising
         #Then dropped 'Year' Column and used groupby for 'Area' column and then took the
         #Standardised values
         df_black_grades = df_all_sex[(df_all_sex['Ethnicity']=='Black')]
         df_b_eth_dropped = df_black_grades.drop(columns={'Ethnicity'})
         dfb=df_b_eth_dropped.reset_index()
         dfb1 = dfb.sort_values(['Area', 'Year']).set_index(['Area'])
         dfb2 = dfb1.rename(columns={'Attainment8':'B Att8', 'Progress8':'B Prog8' })
         df_b_mean = dfb2.drop(columns={'Year'}).groupby('Area').agg({'B Att8':'mean', 'B
```

The above shows how this was achieved for White and Black ethnicities. The same code was applied with respect to Asian, Mixed and Chinese.

The code involves an initial selection of a specific ethnicity, dropping ethnicity column, reset of index, sorting values by area and year, renaming columns, and then finally grouping by area and calculation of a mean for Attainment8 and Progress8. A mean/average was calculated as the dataset contained 4 academic years' worth of Attainment8 and Progress8 scores. Each of these averages was concatenated into a full new data frame.

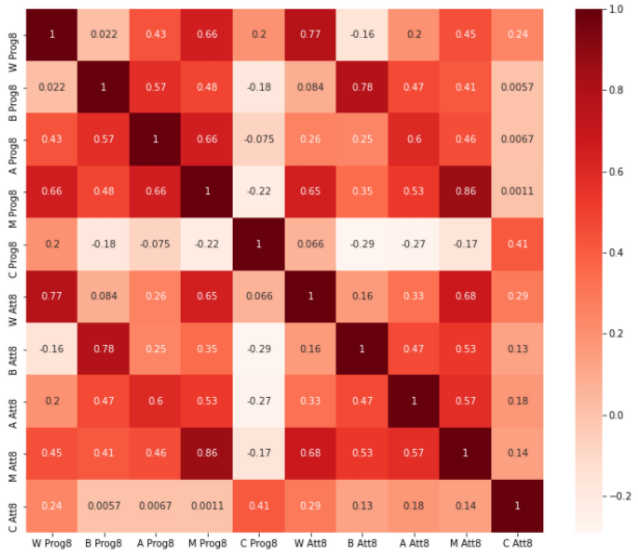
```
In [27]: #Concatenate each filtered datasets above into a new dataset containing Standard
         #Progress8 Scores
         full_df = pd.concat([df_w_mean, df_b_mean, df_a_mean, df_m_mean, df_c_mean], axi
         fdf = full_df.reset_index().set_index('index')
         fdf
```

Out[27]:

index	W Att8	W Prog8	B Att8	B Prog8	A Att8	A Prog8	M Att8	M Prog8
Barking and Dagenham	43.550	-0.122500	48.375000	0.390000	53.900000	0.597500	44.633333	-0.053333
Barnet	55.350	0.417500	48.775000	0.270000	65.033333	0.813333	55.775000	0.380000
Bexley	47.225	-0.277500	55.400000	0.382500	59.600000	0.532500	50.375000	-0.080000
Brent	47.875	0.342500	44.525000	0.045000	52.825000	0.820000	48.533333	0.086667
Bromley	50.800	0.042500	50.725000	0.302500	64.533333	0.703333	52.900000	0.202500
Camden	51.025	0.005000	45.450000	0.002500	47.775000	0.162500	48.500000	-0.097500
Croydon	48.000	0.023333	44.100000	-0.017500	52.200000	0.555000	47.950000	-0.020000
Ealing	51.575	0.345000	45.525000	0.220000	51.925000	0.742500	51.066667	0.166667
Enfield	46.225	0.015000	44.900000	0.077500	57.725000	0.445000	49.150000	0.012500
Greenwich	44.525	-0.320000	47.150000	0.097500	51.650000	0.362500	46.450000	-0.180000

Feature Selection

Missing values within a column were then imputed with median values of each column respectively. A correlation heatmap was used to identify features with a correlation higher than 0.5 for a given Attainment8 grade per ethnicity. As the output is the Attainment8 score the focus for feature selection was on Progress8 scores with a higher correlation value of 0.5. This would determine if a feature such as White Attainment8 score would be coupled with another ethnic group's Progress8 to be modelled in a specific Linear Regression.



This showed very few correlations of the Progress8 score between ethnicities for a specific Attainment8 score of an ethnicity. However, for two ethnic groups Attainment8 scores did yield correlations in Progress8 with another ethnic group. For Whites Attainment8 score showed a correlation in White Progress8 and Mixed Progress8 greater than 0.5. Likewise, for Asians Attainment8 score showed a correlation in Asian Progress8 and Mixed Progress8 scores greater than 0.5. Ethnic groups demonstrating Attainment8 scores that did not correlate with another group would undergo simple linear regression modelling.

Our Models:

Multiple Linear Regression

As the White Attainment8 score showed a correlation between White and Mixed Progress8 scores higher than 0.5 a multiple linear regression model was evaluated for prediction accuracy. Results below shows similarities between coefficient values from sklearn and statsmodel libraries. The prediction coefficient value was similar to the coefficient value found in the OLS Regression Results summary. The accuracy however was low with a R^2 score of 0.42. Therefore, multiple linear regression to predict future White Attainment8 score was unreliable.

```
RMSE:
 2.0185795612479325
R2:
 0.4169192809673864
Intercept:
 [48.28332931]
Coefficients:
 [[10.83101151  3.40029468]]
Predicted Attainment8 Score:
 [[48.46804701]
 [47.53933637]
 [52.41260756]
 [49.35730558]
 [49.4911179 ]
 [49.67813411]
 [48.00595563]
 [50.31066629]
 [49.9448767 ]]
                            OLS Regression Results
==============================================================================
Dep. Variable:                 W Att8   R-squared:                       0.628
Model:                            OLS   Adj. R-squared:                  0.610
Method:                 Least Squares   F-statistic:                     34.57
Date:                Tue, 06 Apr 2021   Prob (F-statistic):           1.60e-09
Time:                        23:19:18   Log-Likelihood:                -94.545
No. Observations:                  44   AIC:                             195.1
Df Residuals:                      41   BIC:                             200.4
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         48.4520      0.366    132.333      0.000      47.713      49.191
W Prog8       10.1872      2.127      4.790      0.000       5.892      14.482
M Prog8        4.7508      2.384      1.993      0.053      -0.063       9.565
==============================================================================
Omnibus:                        1.198   Durbin-Watson:                   1.976
Prob(Omnibus):                  0.549   Jarque-Bera (JB):                0.478
Skew:                           0.180   Prob(JB):                        0.787
Kurtosis:                       3.362   Cond. No.                         9.05
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctl
y specified.
```

The other test of multiple linear regression was for the Attainment8 score for the Asian ethnicity group. Their Attainmnet8 score correlated with the Progress8 score of the Mixed ethnic group. The R^2 Value was 0.53 and RMSE was 2.96. The coefficient predicted values differed to the values displayed on the summary. It can be said the model's future predictions of Attainment8 scores for the Asian ethnic group should be treated with scepticism.

```
RMSE:
 2.961706534090485
R2:
 0.5301911944540938
Intercept:
 [50.08134858]
Coefficients:
 [[6.73827809 5.48653072]]
Predicted Attainment8 Score:
 [[53.7113623 ]
 [53.31598653]
 [57.35929707]
 [52.7456306 ]
 [53.81436699]
 [53.8718636 ]
 [50.64138202]
 [53.94547424]
 [55.62755343]]
                            OLS Regression Results
==============================================================================
Dep. Variable:                 A Att8   R-squared:                       0.392
Model:                            OLS   Adj. R-squared:                  0.362
Method:                 Least Squares   F-statistic:                     13.21
Date:                Tue, 06 Apr 2021   Prob (F-statistic):           3.73e-05
Time:                        23:19:18   Log-Likelihood:                -107.74
No. Observations:                  44   AIC:                             221.5
Df Residuals:                      41   BIC:                             226.8
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         47.6572      2.162     22.039      0.000      43.290      52.024
A Prog8       10.8993      3.987      2.734      0.009       2.847      18.952
M Prog8        4.7905      3.223      1.486      0.145      -1.718      11.299
==============================================================================
Omnibus:                       13.503   Durbin-Watson:                   2.091
Prob(Omnibus):                  0.001   Jarque-Bera (JB):               14.427
Skew:                           1.157   Prob(JB):                     0.000737
Kurtosis:                       4.587   Cond. No.                         13.4
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctl
y specified.
```

Simple Linear Regression

The remaining ethnicity groups were evaluated with simple linear regression models.

```
RMSE:
 2.334113139048039
R2:
 0.6431113359472609
Intercept:
 [44.74606405]
Coefficients:
 [[12.77502373]]
Predicted Attainment8 Score:
 [[44.52250113]
 [46.02356642]
 [47.58850683]
 [44.44798016]
 [45.38481523]
 [46.94975564]
 [44.77800161]
 [43.50049923]
 [49.91994866]]
                            OLS Regression Results
==============================================================================
Dep. Variable:                 B Att8   R-squared:                       0.602
Model:                            OLS   Adj. R-squared:                  0.593
Method:                 Least Squares   F-statistic:                     63.61
Date:                Tue, 06 Apr 2021   Prob (F-statistic):           6.05e-10
Time:                        23:19:18   Log-Likelihood:                -88.954
No. Observations:                  44   AIC:                             181.9
Df Residuals:                      42   BIC:                             185.5
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         44.4208      0.394    112.672      0.000      43.625      45.216
B Prog8       15.0270      1.884      7.976      0.000      11.225      18.829
==============================================================================
Omnibus:                        9.058   Durbin-Watson:                   2.057
Prob(Omnibus):                  0.011   Jarque-Bera (JB):                8.581
Skew:                           0.820   Prob(JB):                       0.0137
Kurtosis:                       4.412   Cond. No.                         6.83
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctl
y specified.
```

Figure 1 Black Simple Linear Regression Results

The results above regard the black ethnicity group on its own. R^2 score was 0.64 showing a better than average line fit for predictions. RMSE score was 2.33. Coefficient predicted value was found to be similar in value to the OLS Regression Results summary.

The R^2 score was 0.61 and RMSE score was 2.68 for Mixed Ethnicity score. Coefficient prediction was close in value to OLS Regression Results coefficient value. Predicted Attainment8 scores can be remarked as realistic to previous test scores.

```
2.6835051758085506
0.6094576563641783
Intercept:
 [48.62543025]
Coefficients:
 [[13.24249494]]
Predicted Attainment8 Score:
 [[48.36058035]
 [47.73156184]
 [52.89613487]
 [48.06262421]
 [48.16194293]
 [48.89028015]
 [47.33428699]
 [49.98278598]
 [53.12787853]]
                            OLS Regression Results
==============================================================================
Dep. Variable:                 M Att8   R-squared:                       0.740
Model:                            OLS   Adj. R-squared:                  0.734
Method:                 Least Squares   F-statistic:                     119.7
Date:                Tue, 06 Apr 2021   Prob (F-statistic):           7.18e-14
Time:                        23:19:18   Log-Likelihood:                -84.301
No. Observations:                  44   AIC:                             172.6
Df Residuals:                      42   BIC:                             176.2
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         48.7919      0.260    187.401      0.000      48.266      49.317
M Prog8       15.4172      1.409     10.940      0.000      12.573      18.261
==============================================================================
Omnibus:                        8.297   Durbin-Watson:                   2.257
Prob(Omnibus):                  0.016   Jarque-Bera (JB):                9.022
Skew:                           0.627   Prob(JB):                       0.0110
Kurtosis:                       4.830   Cond. No.                         5.57
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctl
y specified.
```

The R^2 score was -11.47 and RMSE was 3.84 for the Chinese Attainment8 group. However, the predicted coefficient value was close in value to the OLS Regression Results coefficient. Predicted Attainment8 scores were found to be close in value to previous years scores. The predicted results should be treated with scepticism.

```
3.835507826800202
-11.472753738211756
Intercept:
 [56.34599476]
Coefficients:
 [[7.62976366]]
Predicted Attainment8 Score:
 [[61.07644823]
 [61.45793641]
 [62.22727091]
 [62.83129387]
 [62.62147537]
 [61.68682932]
 [62.27177787]
 [62.57696841]
 [62.22727091]]
                            OLS Regression Results
==============================================================================
Dep. Variable:                 C Att8   R-squared:                       0.171
Model:                            OLS   Adj. R-squared:                  0.151
Method:                 Least Squares   F-statistic:                     8.646
Date:                Wed, 07 Apr 2021   Prob (F-statistic):            0.00531
Time:                        00:33:04   Log-Likelihood:                -113.35
No. Observations:                  44   AIC:                             230.7
Df Residuals:                      42   BIC:                             234.3
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         56.6852      1.743     32.523      0.000      53.168      60.203
C Prog8        6.8355      2.325      2.940      0.005       2.144      11.527
==============================================================================
Omnibus:                        7.276   Durbin-Watson:                   1.319
Prob(Omnibus):                  0.026   Jarque-Bera (JB):                6.248
Skew:                          -0.726   Prob(JB):                       0.0440
Kurtosis:                       4.139   Cond. No.                         7.26
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctl
y specified.
```

Random Forest Regression: Data Transformation

Data retrieval and loading process was the same as done previously.

```
In [1]: # Pandas is used for data manipulation
        import pandas as pd
        # Read in data and display first 5 rows
        feature = pd.read_csv('Fixed_data.csv', converters={'Ethnicity': str.strip})
        feature.head(5)
```

Out[1]:

	Code	Area	Year	Sex	Ethnicity	Pupils	Attainment8	Progress8
0	E09000002	Barking and Dagenham	2018/19	All	White	903.0	41.7	-0.21
1	E09000002	Barking and Dagenham	2018/19	All	Mixed	227.0	44.7	0.05
2	E09000002	Barking and Dagenham	2018/19	All	Asian	530.0	53.3	0.60
3	E09000002	Barking and Dagenham	2018/19	All	Black	642.0	48.0	0.37
4	E09000002	Barking and Dagenham	2018/19	All	Chinese	3.0	69.0	0.93

Column named ‘Code’ was dropped and categorical features underwent one-hot encoding using pandas get_dummies.

	Area_Bexley	Area_Brent	Area_Bromley	Area_Camden	Area_City of London	Area_Croydon	Area_Ealing
0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0

5 rows × 56 columns

Attainment8 column was converted to a Numpy array and then split into test and train sets.

Baseline Prediction

```
# The baseline predictions are the historical averages
baseline_preds = test_features[:, feature_list.index('Progress8')]
# Baseline errors, and display average baseline error
baseline_errors = abs(baseline_preds - test_labels)
print('Average baseline error: ', round(np.mean(baseline_errors), 2))

Average baseline error:  51.58
```

A baseline was used to evaluate predictions from the Random Forest Regressor model. If the model betters the baseline value, then it will be evaluated to be a success. The above image shows Attainment baseline error value to be 51.58.

MODEL

The Random Forest Regressor was set to instantiate 1000 decision trees with 42 random states. It was trained and predictions were generated from the test_features set. The Mean Absolute Error value was 1.77.

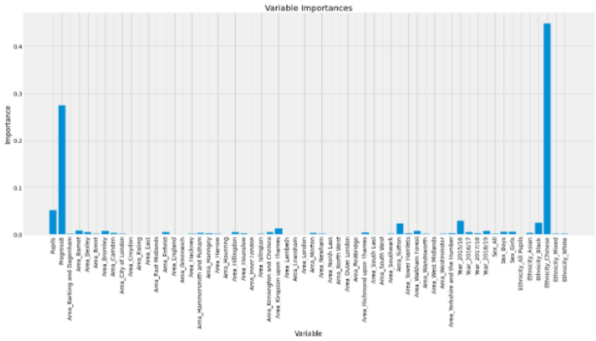
```
# Use the forest's predict method on the test data
predictions = rf.predict(test_features)
# Calculate the absolute errors
errors = abs(predictions - test_labels)
# Print out the mean absolute error (mae)
print('Mean Absolute Error:', round(np.mean(errors), 2))

Mean Absolute Error: 1.77
```

The mean absolute percentage error (MAPE) was 96.38%. This indicates a very high and positive model predictive accuracy.

```
# Calculate mean absolute percentage error (MAPE)
mape = 100 * (errors / test_labels)
# Calculate and display accuracy
accuracy = 100 - np.mean(mape)
print('Accuracy:', round(accuracy, 2), '%.')
```

```
Accuracy: 96.38 %.
```

A record of the decision tree was made for analysis and viewing.

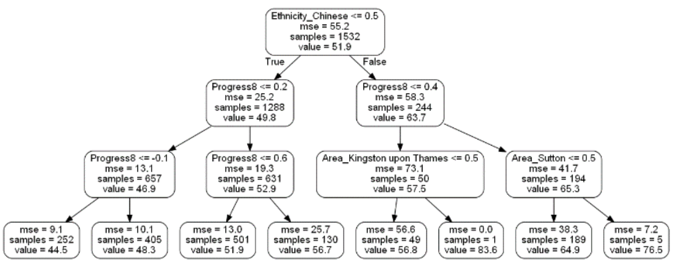


The graph above shows the importance of each variable clearly and hence, ethnicity Chinese and Progress8 was chosen as the best two variable inputs.

Feature Selection and Prediction Results

```
Variable: Ethnicity_Chinese    Importance: 0.45
Variable: Progress8            Importance: 0.27
Variable: Pupils               Importance: 0.05
Variable: Year_2015/16         Importance: 0.03
Variable: Ethnicity_Black      Importance: 0.03
Variable: Area_Sutton          Importance: 0.02
Variable: Area_Barnet          Importance: 0.01
Variable: Area_Bexley          Importance: 0.01
Variable: Area_Bromley         Importance: 0.01
Variable: Area_Kensington and Chelsea Importance: 0.01
Variable: Area_Kingston upon Thames Importance: 0.01
Variable: Area_Waltham Forest  Importance: 0.01
Variable: Year_2018/19         Importance: 0.01
Variable: Sex_Boys             Importance: 0.01
Variable: Sex_Girls            Importance: 0.01
Variable: Area_Barking and Dagenham Importance: 0.0
Variable: Area_Brent           Importance: 0.0
Variable: Area_Camden          Importance: 0.0
Variable: Area_City of London  Importance: 0.0
Variable: Area_Croydon         Importance: 0.0
Variable: Area_Ealing          Importance: 0.0
Variable: Area_East            Importance: 0.0
Variable: Area_East Midlands   Importance: 0.0
Variable: Area_Enfield         Importance: 0.0
Variable: Area_England         Importance: 0.0
Variable: Area_Greenwich       Importance: 0.0
Variable: Area_Hackney         Importance: 0.0
Variable: Area_Hammersmith and Fulham Importance: 0.0
Variable: Area_Haringey        Importance: 0.0
Variable: Area_Harrow          Importance: 0.0
Variable: Area_Havering        Importance: 0.0
Variable: Area_Hillingdon      Importance: 0.0
Variable: Area_Hounslow        Importance: 0.0
Variable: Area_Inner London    Importance: 0.0
Variable: Area_Islington       Importance: 0.0
Variable: Area_Lambeth         Importance: 0.0
Variable: Area_Lewisham        Importance: 0.0
Variable: Area_London          Importance: 0.0
Variable: Area_Merton          Importance: 0.0
```

Identifying the correct features is paramount to the success of a Random Forest Regression model. Displayed above is the ranking of importance of every feature that exists in the dataset. Ethnicity_Chinese is of the most importance with a value of 0.45 followed by Progress8 with 0.27. These two features will be the focal input for the Random Forest Regression model.



Once variables are inputted, the results show an accuracy in predictive ability of 93.59%. This shows the Random Forest Regression model is more apt at prediction of future Attainment8 grades for the Chinese ethnicity. Similarly, the other ethnic groups showed an accuracy in predictive percentage above 90% consistently.

```
Mean Absolute Error: 3.23
Accuracy: 93.59 %.
```

Figure 2 Accuracy for Chinese Ethnic Group

VI. Conclusions

The Random Forest Regression is proven to be not only highly accurate but a reliable model for prediction of future results. Compared to predictions from Linear Regression models which consisted of high errors and minimal predictive accuracy, it is advised to use Random Forest Regression as the main model. Our findings have shown that ethnicities in the UK vary greatly in results. It also shown that correlations between races are rare and have minimal effect on improving predictability of their future scores through some machine learning models. This may be due to the level of simplicity in some models such as Linear Regression compared to Random Forest Regression.

VII. Recommendations

From the findings of our report, we can be able to recommend different solutions that the government, teachers and parents can take to be able to combat those factors. One way is that teachers should commit to setting high- quality homework, consistent in -class assessment, after school extra-lesson tutoring for pupil's who need the extra support, and parents/guardians should commit to ensuring that the homework set is completed and completed in due time and should also have regular contact with the school to discuss pupil's progress. Another recommendation is that government should force all schools to publish data on training provision and turnover rates for teachers (especially in their entry level career stage) in different schools and across multi-academy trusts. This should be produced in a systematised form to promote comparability and shine a light on pupil's memory and retention plus development problems.

References

[1] Katwala A, "Results Day Is A Diversity Disaster. Here'S All The Proof You Need,". *WIRED UK*, 2021.

[2] ~ P. Patel, "2020 Exams Result and Bias.," *The Teacherist*.2020. URL:https://theteacherist.com/2020/03/29/teacher-bias/

[3] L. Stokes, H. Rolfe, N. Hudson- Sharp and S. Stevens, "A compendium of evidence on ethnic minority resilience to the effects of deprivation on attainment,". National Institute of Economic and Social Research, 2015.

[4] N. Roberts, and P. Bolton, "Educational outcomes of Black pupils and students,". House of Commons Library, 2020.

[5] T. Bramley, C. Vidal Rodeiro, and S. Vitello, "Gender differences in GCSE", Cambridge Assessment Research Report, 2015.

[6] P. Connolly,"The effects of social class and ethnicity on gender differences in GCSE attainment: a secondary analysis of the Youth Cohort Study of England and Wales 1997–2001", British educational research journal, 32(1), 3-21, 2006.

[7] J.O. Jonsson, "Explaining Sex Differences in Educational Choice: An empirical assessment of a rational choice model", European Sociological Review, 15(4): 391-404, 1999.

[8] P. Sammons, K. Sylva, E. Melhuish, I. Siraj, B. Taggart,, K. Toth, and R. Smees, "Influences on students' GCSE attainment and progress at age 16", Effective Pre-School, Primary & Secondary Education Project (EPPSE), 2014.

[9] D. Kurniawan, "Multiple Linear Regression for Manufacturing Analysis," *Medium*, 2020. URL: https://towardsdatascience.com/multiple-linear-regression-for-manufacturing-analysis-c057d4af718b.

[10] N. Donges, "A complete guide to the random forest algorithm," *Built In*. 2021. URL:https://builtin.com/data-science/random-forest-algorithm.

[11] Clegg, R., Allen, R., Freedman, S., Kinnock MP, S. and Fernandes MP, S., *Commission on Inequality in Education*. Smf.co.uk. 2021. URL:https://www.smf.co.uk/wp-content/uploads/2017/07/Education-Commission-final-web-report.pdf

[12]

[13] .

Code
https://github.com/inspectorsherlock/Data-Analysis.git

In [1]:
```
#Importing all necessary library for Data Analysis
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from scipy.stats import norm
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from scipy import stats
import warnings
warnings.filterwarnings('ignore')
%matplotlib inline
from pandas.plotting import scatter_matrix
from matplotlib import pyplot
```

In [2]:
```
#Loading csv dataset and using converters to remove leading & trailing whitespac
df = pd.read_csv('Fixed_data.csv', converters={'Ethnicity': str.strip})
```

In [3]:
```
#Code shows the first 10 datapoints in the dataset
df.head()
```

Out[3]:

	Code	Area	Year	Sex	Ethnicity	Pupils	Attainment8	Progress8
0	E09000002	Barking and Dagenham	2018/19	All	White	903.0	41.7	-0.21
1	E09000002	Barking and Dagenham	2018/19	All	Mixed	227.0	44.7	0.05
2	E09000002	Barking and Dagenham	2018/19	All	Asian	530.0	53.3	0.60
3	E09000002	Barking and Dagenham	2018/19	All	Black	642.0	48.0	0.37
4	E09000002	Barking and Dagenham	2018/19	All	Chinese	3.0	69.0	0.93

In [4]:
```
#Describes the dataset
df.describe()
```

Out[4]:

	Pupils	Attainment8	Progress8
count	3110.000000	3106.000000	3048.000000
mean	6146.238585	52.011719	0.285722
std	30810.214114	7.770027	0.430250
min	1.000000	15.000000	-0.860000
25%	124.250000	46.900000	-0.020000
50%	406.000000	50.500000	0.240000
75%	1463.000000	55.300000	0.530000
max	540006.000000	89.000000	2.820000

In [5]:
```
#Summary of dataset
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3240 entries, 0 to 3239
Data columns (total 8 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   Code         3240 non-null   object 
 1   Area         3240 non-null   object 
 2   Year         3240 non-null   object 
 3   Sex          3240 non-null   object 
 4   Ethnicity    3240 non-null   object 
 5   Pupils       3110 non-null   float64
 6   Attainment8  3106 non-null   float64
 7   Progress8    3048 non-null   float64
dtypes: float64(3), object(5)
memory usage: 202.6+ KB
```

In [6]:
```
#Calculation for total number of students in dataset between academic years 201
all_students_ethnicity_sex = df[(df['Sex'] == 'All') &
                                (df['Ethnicity'] == 'All Pupils')]
total_students = all_students_ethnicity_sex['Pupils'].sum()
print("There exists a total number of %d students in this GCSE dataset between a
```

```
There exists a total number of 4863974 students in this GCSE dataset between ac
ademic years 2015-2019
```

In [7]:
```
#Calculation for total number of female students in dataset between academic ye
girls_student_all_ethnics = df[((df['Sex'] == 'Girls') &
                                (df['Ethnicity'] == 'All Pupils'))]
total_girls = girls_student_all_ethnics['Pupils'].sum()
print("There exists a total number of %d of girls in the GCSE dataset between ac
```

```
There exists a total number of 2391378 of girls in the GCSE dataset between aca
demic years 2015-2019.
```

In [8]:
```
#Calculation for total number of male students in dataset between academic year
boys_student_all_ethnics = df[((df['Sex'] == 'Boys') &
                               (df['Ethnicity'] == 'All Pupils'))]
total_boys = boys_student_all_ethnics['Pupils'].sum()
print("There exists a total number of %d of boys in the GCSE dataset between aca
```

```
There exists a total number of 2472596 of boys in the GCSE dataset between acad
emic years 2015-2019.
```

In [9]:
```
#Total number of null/NaN values in the dataset
df.isnull().sum()
```

Out[9]:
```
Code            0
Area            0
Year            0
Sex             0
Ethnicity       0
Pupils        130
Attainment8   134
Progress8     192
dtype: int64
```

In [10]:
```
#Calculation of quantiles, IQR and removal of outliers
Q1 = df.quantile(0.25)
Q3 = df.quantile(0.75)
IQR = Q3 - Q1
(df < (Q1 - 1.5 * IQR)) |(df > (Q3 + 1.5 * IQR)).any(axis=1)
df_outlier = df[~((df < (Q1 - 1.5 * IQR)) |(df > (Q3 + 1.5 * IQR))).any(axis=1)]
df_outlier_removed = df_outlier.dropna()
df_cleaned = df_outlier_removed.set_index('Area').reset_index()
print(IQR)
df_cleaned
```

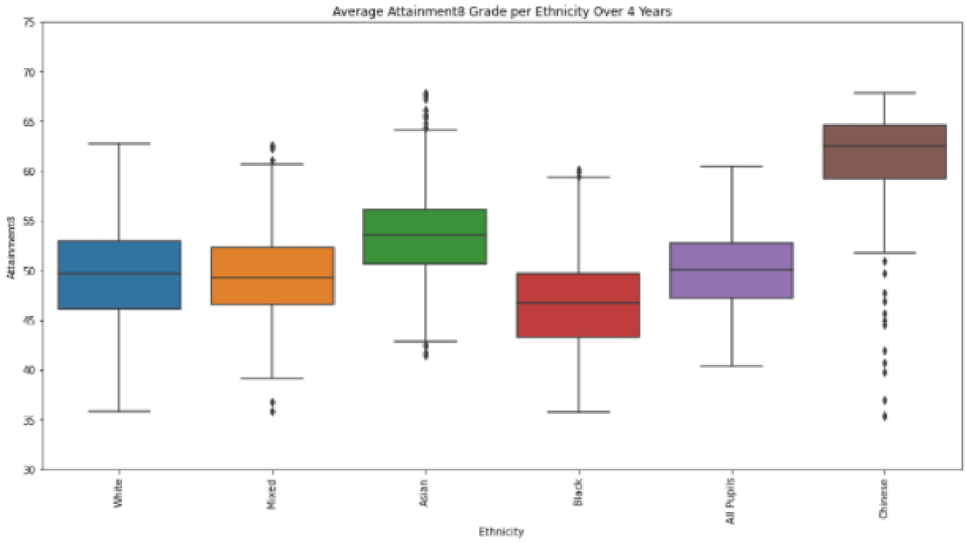
```
Pupils         1338.75
Attainment8       8.40
Progress8         0.55
dtype: float64
```

Out[10]:

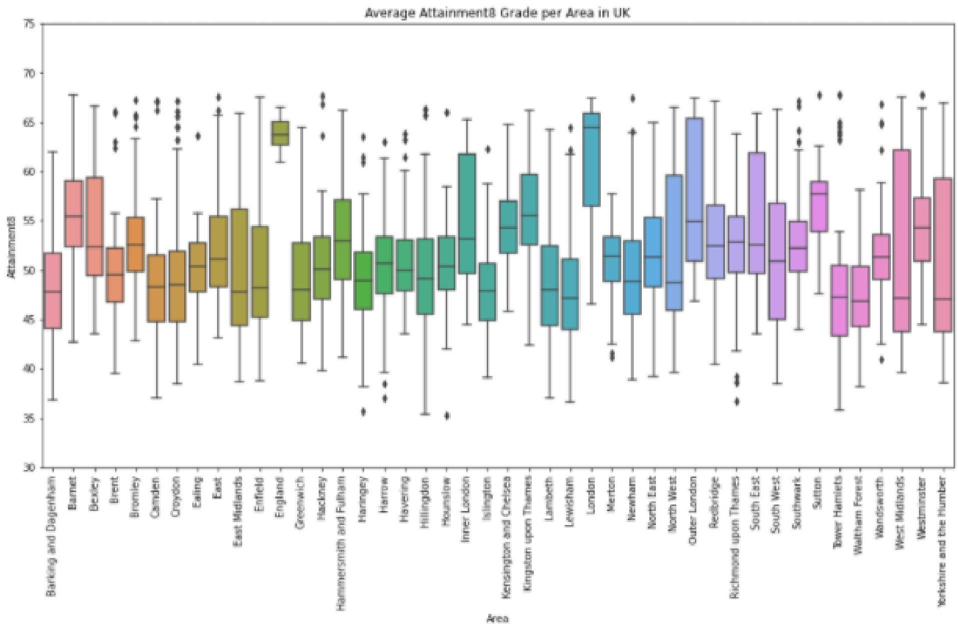
	Area	Code	Year	Sex	Ethnicity	Pupils	Attainment8	Progress8
0	Barking and Dagenham	E09000002	2018/19	All	White	903.0	41.7	-0.21
1	Barking and Dagenham	E09000002	2018/19	All	Mixed	227.0	44.7	0.05
2	Barking and Dagenham	E09000002	2018/19	All	Asian	530.0	53.3	0.60
3	Barking and Dagenham	E09000002	2018/19	All	Black	642.0	48.0	0.37
4	Barking and Dagenham	E09000002	2018/19	All	All Pupils	2353.0	46.4	0.16
...	...	...	...	...	...	...	...	...
2418	Yorkshire and the Humber	E12000003	2015/16	Boys	Mixed	849.0	45.7	-0.29
2419	Yorkshire and the Humber	E12000003	2015/16	Girls	Asian	2938.0	50.5	0.28
2420	Yorkshire and the Humber	E12000003	2015/16	Girls	Black	530.0	50.4	0.42
2421	Yorkshire and the Humber	E12000003	2015/16	Girls	Chinese	64.0	64.0	0.96
2422	Yorkshire and the Humber	E12000003	2015/16	Girls	Mixed	809.0	50.5	0.08

2423 rows × 8 columns

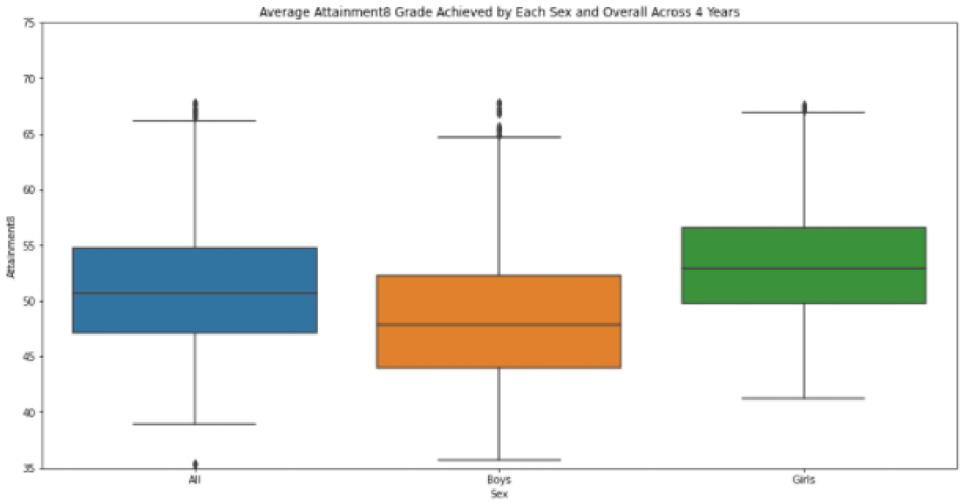
In [11]:
```
#Ethnicity with highest and lowest results across 4 years and
var = 'Ethnicity'
data = pd.concat([df_cleaned['Attainment8'], df_cleaned[var]], axis=1)
f, ax = plt.subplots(figsize=(16, 8))
fig = sns.boxplot(x=var, y="Attainment8", data=data)
fig.axis(ymin=30, ymax=75);
plt.title('Average Attainment8 Grade per Ethnicity Over 4 Years')
plt.xticks(rotation=90);
```



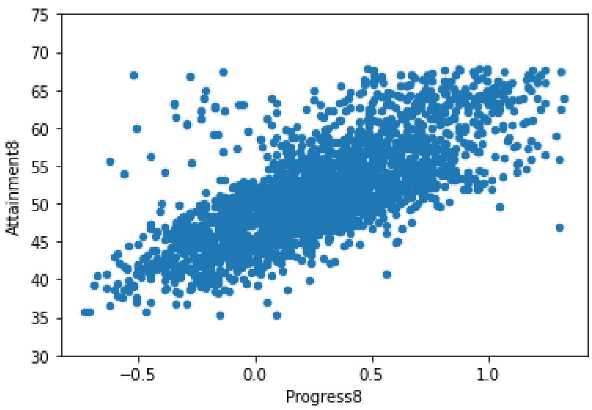
In [12]:
```
#Boxplot of Average Attainment8 Grade per Area in UK
var = 'Area'
data = pd.concat([df_cleaned['Attainment8'], df_cleaned[var]], axis=1)
f, ax = plt.subplots(figsize=(16, 8))
fig = sns.boxplot(x=var, y="Attainment8", data=data)
fig.axis(ymin=30, ymax=75);
plt.title('Average Attainment8 Grade per Area in UK')
plt.xticks(rotation=90);
```



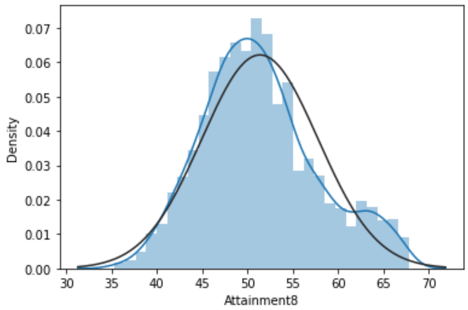
In [13]:
```
#Boxplot of Average Attainment8 Grade Achieved by Each Sex and Overall Across 4 /
var = 'Sex'
data = pd.concat([df_cleaned['Attainment8'], df_cleaned[var]], axis=1)
f, ax = plt.subplots(figsize=(16, 8))
fig = sns.boxplot(x=var, y="Attainment8", data=data)
fig.axis(ymin=35, ymax=75);
plt.title('Average Attainment8 Grade Achieved by Each Sex and Overall Across 4 Y
plt.xticks(rotation=0);
```

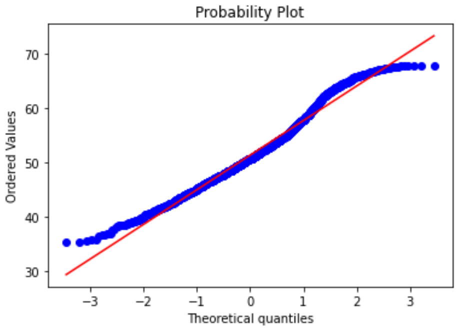


In [14]:
```
#Scatter Plot Graph of Progress8 Scores against Attainment8 Scores
var = 'Progress8'
data = pd.concat([df_cleaned['Attainment8'], df_cleaned[var]], axis=1)
data.plot.scatter(x=var, y='Attainment8', ylim=(30,75));
```



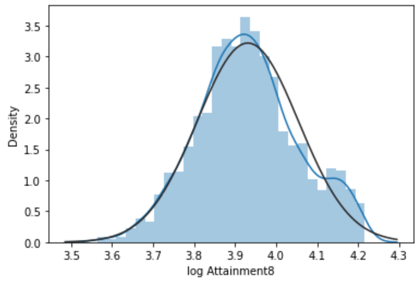
In [15]:
```
# Inital Histogram and normal probability plot after IQR filtering
sns.distplot(df_cleaned['Attainment8'], fit=norm);
fig = plt.figure()
res = stats.probplot(df_cleaned['Attainment8'], plot=plt)
```

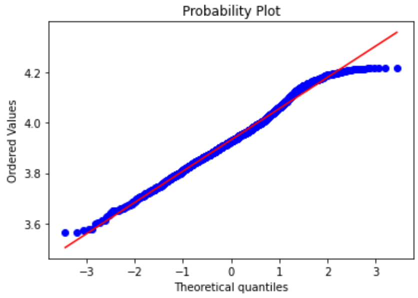




```
In [16]: # Applying log transformation to more perfectly match the Gaussian distribution
df_cleaned['log Attainment8'] = np.log(df_cleaned['Attainment8'])
df_cleaned['log Progress8'] = np.log(df_cleaned['Progress8'])
```

```
In [17]: # Histogram and normal probability plot after log transformation ready for ML an
sns.distplot(df_cleaned['log Attainment8'], fit=norm);
fig = plt.figure()
res = stats.probplot(df_cleaned['log Attainment8'], plot=plt)
```





In [18]:
```
#Skewness and Kurtosis Value of Attainment8
print("Skewness: %f" % df_cleaned['Attainment8'].skew())
print("Kurtosis: %f" % df_cleaned['Attainment8'].kurt())
```

```
Skewness: 0.422188
Kurtosis: -0.171318
```

In [19]:
```
#Dropping unneeded columns such as Code and Pupils from the dataset
df_nocode_pup = df_cleaned.drop(columns={'Code', 'Pupils'})
```

In [20]:
```
#Setting Area as the new Index for the Dataset
df_area_ix = df_nocode_pup.set_index('Area')
```

In [21]:
```
#Selecting 'ALL' Sexes from dataset into a variable and then dropping 'Sex' Colu
#into a new Variable
df_all_sexes = df_area_ix[(df_area_ix['Sex']=='All')]
df_all_sex = df_all_sexes.drop(columns={'Sex'})
df_all_sex
```

Out[21]:

	Year	Ethnicity	Attainment8	Progress8	log Attainment8	log Progress8
Area						
Barking and Dagenham	2018/19	White	41.7	-0.21	3.730501	NaN
Barking and Dagenham	2018/19	Mixed	44.7	0.05	3.799974	-2.995732
Barking and Dagenham	2018/19	Asian	53.3	0.60	3.975936	-0.510826
Barking and Dagenham	2018/19	Black	48.0	0.37	3.871201	-0.994252
Barking and Dagenham	2018/19	All Pupils	46.4	0.16	3.837299	-1.832581
...	...	...	...	...	...	...
Yorkshire and the Humber	2016/17	Chinese	60.8	1.16	4.107590	0.148420
Yorkshire and the Humber	2016/17	Mixed	43.8	-0.08	3.779634	NaN
Yorkshire and the Humber	2015/16	Black	46.7	0.28	3.843744	-1.272966
Yorkshire and the Humber	2015/16	Chinese	63.1	0.83	4.144721	-0.186330
Yorkshire and the Humber	2015/16	Mixed	48.1	-0.11	3.873282	NaN

798 rows × 6 columns

In [22]:
```
#Selecting 'White' Ethnicity, dropping 'Ethnicity' column and then Standardising
#Then dropped 'Year' Column and used groupby for 'Area' column and then took the
#Standardised values
df_white_grades = df_all_sex[(df_all_sex['Ethnicity']=='White')]
df_w_eth_dropped = df_white_grades.drop(columns={'Ethnicity'})
dfw=df_w_eth_dropped.reset_index()
dfw1 = dfw.sort_values(['Area', 'Year']).set_index(['Area'])
dfw2 = dfw1.rename(columns={'Attainment8':'W Att8', 'Progress8':'W Prog8' })
df_w_mean = dfw2.drop(columns={'Year'}).groupby('Area').agg({'W Att8':'mean', 'W
```

In [23]:
```
#Selecting 'Black' Ethnicity, dropping 'Ethnicity' column and then Standardising
#Then dropped 'Year' Column and used groupby for 'Area' column and then took the
#Standardised values
df_black_grades = df_all_sex[(df_all_sex['Ethnicity']=='Black')]
df_b_eth_dropped = df_black_grades.drop(columns={'Ethnicity'})
dfb=df_b_eth_dropped.reset_index()
dfb1 = dfb.sort_values(['Area', 'Year']).set_index(['Area'])
dfb2 = dfb1.rename(columns={'Attainment8':'B Att8', 'Progress8':'B Prog8' })
df_b_mean = dfb2.drop(columns={'Year'}).groupby('Area').agg({'B Att8':'mean', 'B
```

In [24]:
```
#Selecting 'Asian' Ethnicity, dropping 'Ethnicity' column and then Standardising
#Then dropped 'Year' Column and used groupby for 'Area' column and then took the
#Standardised values
df_asian_grades = df_all_sex[(df_all_sex['Ethnicity']=='Asian')]
df_a_eth_dropped = df_asian_grades.drop(columns={'Ethnicity'})
dfa=df_a_eth_dropped.reset_index()
dfa1 = dfa.sort_values(['Area', 'Year']).set_index(['Area'])
dfa2 = dfa1.rename(columns={'Attainment8':'A Att8', 'Progress8':'A Prog8' })
df_a_mean = dfa2.drop(columns={'Year'}).groupby('Area').agg({'A Att8':'mean', 'A
```

In [25]:
```
#Selecting 'Mixed' Ethnicity, dropping 'Ethnicity' column and then Standardising
#Then dropped 'Year' Column and used groupby for 'Area' column and then took the
#Standardised values
df_mixed_grades = df_all_sex[(df_all_sex['Ethnicity']=='Mixed')]
df_m_eth_dropped = df_mixed_grades.drop(columns={'Ethnicity'})
dfm=df_m_eth_dropped.reset_index()
dfm1 = dfm.sort_values(['Area', 'Year']).set_index(['Area'])
dfm2 = dfm1.rename(columns={'Attainment8':'M Att8', 'Progress8':'M Prog8' })
df_m_mean = dfm2.drop(columns={'Year'}).groupby('Area').agg({'M Att8':'mean', 'M
```

In [26]:
```python
#Selecting 'Chinese' Ethnicity, dropping 'Ethnicity' column and then Standardisi
#Then dropped 'Year' Column and used groupby for 'Area' column and then took the
#Standardised values
df_chinese_grades = df_all_sex[(df_all_sex['Ethnicity']=='Chinese')]
df_c_eth_dropped = df_chinese_grades.drop(columns={'Ethnicity'})
dfc = df_c_eth_dropped.reset_index()
dfc1 = dfc.sort_values(['Area', 'Year']).set_index(['Area'])
dfc2 = dfc1.rename(columns={'Attainment8':'C Att8', 'Progress8':'C Prog8' })
df_c_mean = dfc2.drop(columns={'Year'}).groupby('Area').agg({'C Att8':'mean', 'C
```

In [48]:
```python
#Concatenate each filtered datasets above into a new dataset containing Standard
#Progress8 Scores
full_df = pd.concat([df_w_mean, df_b_mean, df_a_mean, df_m_mean, df_c_mean], axi
fdf = full_df.reset_index().set_index('index')
fdf.head()
```

Out[48]:

index	W Att8	W Prog8	B Att8	B Prog8	A Att8	A Prog8	M Att8	M Prog8	C Att8
Barking and Dagenham	43.550	-0.1225	48.375	0.3900	53.900000	0.597500	44.633333	-0.053333	54.800000
Barnet	55.350	0.4175	48.775	0.2700	65.033333	0.813333	55.775000	0.380000	NaN
Bexley	47.225	-0.2775	55.400	0.3825	59.600000	0.532500	50.375000	-0.080000	63.333333
Brent	47.875	0.3425	44.525	0.0450	52.825000	0.820000	48.533333	0.086667	64.800000
Bromley	50.800	0.0425	50.725	0.3025	64.533333	0.703333	52.900000	0.202500	65.700000

In [28]:
```python
#Median scores were used as imputation for Null/NaN values
fdf['W Att8'] = fdf['W Att8'].fillna(fdf['W Att8'].median())
fdf['W Prog8'] = fdf['W Prog8'].fillna(fdf['W Prog8'].median())
fdf['B Att8'] = fdf['B Att8'].fillna(fdf['B Att8'].median())
fdf['B Prog8'] = fdf['B Prog8'].fillna(fdf['B Prog8'].median())
fdf['A Att8'] = fdf['A Att8'].fillna(fdf['A Att8'].median())
fdf['A Prog8'] = fdf['A Prog8'].fillna(fdf['A Prog8'].median())
fdf['M Att8'] = fdf['M Att8'].fillna(fdf['M Att8'].median())
fdf['M Prog8'] = fdf['M Prog8'].fillna(fdf['M Prog8'].median())
fdf['C Att8'] = fdf['C Att8'].fillna(fdf['C Att8'].median())
fdf['C Prog8'] = fdf['C Prog8'].fillna(fdf['C Prog8'].median())
```

In [49]:
```python
#Dataset renamed index to 'Area' and assigned to a new variable name: fdf_cleane
fdf_cleaned = fdf.reset_index().rename(columns={'index':'Area'})
fdf_cleaned.head()
```

Out[49]:

	Area	W Att8	W Prog8	B Att8	B Prog8	A Att8	A Prog8	M Att8	M Prog8	C Att8
0	Barking and Dagenham	43.550	-0.1225	48.375	0.3900	53.900000	0.597500	44.633333	-0.053333	54.80000(
1	Barnet	55.350	0.4175	48.775	0.2700	65.033333	0.813333	55.775000	0.380000	Nal
2	Bexley	47.225	-0.2775	55.400	0.3825	59.600000	0.532500	50.375000	-0.080000	63.33333:
3	Brent	47.875	0.3425	44.525	0.0450	52.825000	0.820000	48.533333	0.086667	64.80000(
4	Bromley	50.800	0.0425	50.725	0.3025	64.533333	0.703333	52.900000	0.202500	65.70000(

In [30]:
```
#filtered cleaned dataset for each ethnicity's Prog8 scores and assigned to new
newx = fdf_cleaned.filter(['W Prog8', 'B Prog8', 'A Prog8', 'M Prog8', 'C Prog8'
```

In [31]:
```
#filtered cleaned dataset for each ethnicity's Attainment8 scores and assigned t
newy = fdf_cleaned.filter(['W Att8', 'B Att8', 'A Att8', 'M Att8', 'C Att8'], ax
```

In [50]:
```
#Concatenated into a new dataframe and variable called new_full_df, dataframe sh
new_full_df = pd.concat([newx, newy], axis=1)
new_full_df.head()
```

Out[50]:

	W Prog8	B Prog8	A Prog8	M Prog8	C Prog8	W Att8	B Att8	A Att8	M Att8	C Att8
0	-0.1225	0.3900	0.597500	-0.053333	0.180000	43.550	48.375	53.900000	44.633333	54.800000
1	0.4175	0.2700	0.813333	0.380000	0.770833	55.350	48.775	65.033333	55.775000	62.375000
2	-0.2775	0.3825	0.532500	-0.080000	0.496667	47.225	55.400	59.600000	50.375000	63.333333
3	0.3425	0.0450	0.820000	0.086667	0.806667	47.875	44.525	52.825000	48.533333	64.800000
4	0.0425	0.3025	0.703333	0.202500	0.470000	50.800	50.725	64.533333	52.900000	65.700000

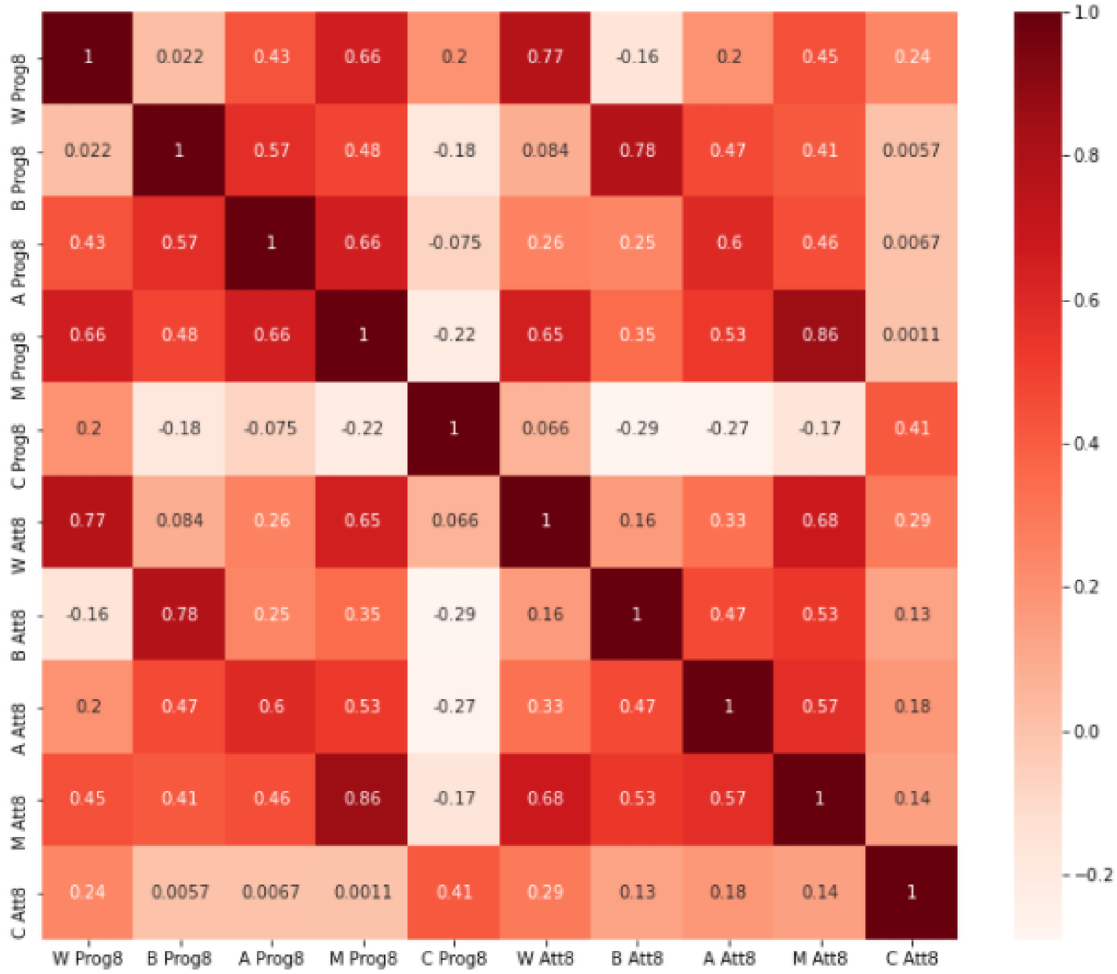
In [33]:
```
#Correlation values between features for feature selection purposes
new_full_df.corr()
```

Out[33]:

	W Prog8	B Prog8	A Prog8	M Prog8	C Prog8	W Att8	B Att8	A Att8	M
W Prog8	1.000000	0.022343	0.434880	0.655474	0.196081	0.769188	-0.159289	0.195305	0.446
B Prog8	0.022343	1.000000	0.572556	0.482270	-0.182704	0.084449	0.776091	0.469868	0.412
A Prog8	0.434880	0.572556	1.000000	0.656946	-0.075391	0.258392	0.247948	0.599283	0.455
M Prog8	0.655474	0.482270	0.656946	1.000000	-0.216725	0.647603	0.347975	0.530175	0.860
C Prog8	0.196081	-0.182704	-0.075391	-0.216725	1.000000	0.066193	-0.289936	-0.272727	-0.167
W Att8	0.769188	0.084449	0.258392	0.647603	0.066193	1.000000	0.158865	0.326250	0.679
B Att8	-0.159289	0.776091	0.247948	0.347975	-0.289936	0.158865	1.000000	0.467847	0.533
A Att8	0.195305	0.469868	0.599283	0.530175	-0.272727	0.326250	0.467847	1.000000	0.574
M Att8	0.446604	0.412752	0.455371	0.860364	-0.167650	0.679788	0.533864	0.574550	1.000
C Att8	0.242326	0.005666	0.006687	0.001138	0.413182	0.289560	0.133003	0.178188	0.142

In [34]:

```
#Using Pearson Correlation heatmap for correlation identification and feature se
plt.figure(figsize=(12,10))
cor = new_full_df.corr()
sns.heatmap(cor, annot=True, cmap=plt.cm.Reds)
plt.show()
```



	W Prog8	B Prog8	A Prog8	M Prog8	C Prog8	W Att8	B Att8	A Att8	M Att8	C Att8
W Prog8	1	0.022	0.43	0.66	0.2	0.77	-0.16	0.2	0.45	0.24
B Prog8	0.022	1	0.57	0.48	-0.18	0.084	0.78	0.47	0.41	0.0057
A Prog8	0.43	0.57	1	0.66	-0.075	0.26	0.25	0.6	0.46	0.0067
M Prog8	0.66	0.48	0.66	1	-0.22	0.65	0.35	0.53	0.86	0.0011
C Prog8	0.2	-0.18	-0.075	-0.22	1	0.066	-0.29	-0.27	-0.17	0.41
W Att8	0.77	0.084	0.26	0.65	0.066	1	0.16	0.33	0.68	0.29
B Att8	-0.16	0.78	0.25	0.35	-0.29	0.16	1	0.47	0.53	0.13
A Att8	0.2	0.47	0.6	0.53	-0.27	0.33	0.47	1	0.57	0.18
M Att8	0.45	0.41	0.46	0.86	-0.17	0.68	0.53	0.57	1	0.14
C Att8	0.24	0.0057	0.0067	0.0011	0.41	0.29	0.13	0.18	0.14	1

In [35]:
```
#Correlation with output variable
cor_target = abs(cor["W Att8"])
#Selecting highly correlated features
relevant_features = cor_target[cor_target>0.5]
relevant_features
```

Out[35]:
```
W Prog8    0.769188
M Prog8    0.647603
W Att8     1.000000
M Att8     0.679788
Name: W Att8, dtype: float64
```

In [36]:
```
#Correlation with output variable
cor_target = abs(cor["B Att8"])
#Selecting highly correlated features
relevant_features = cor_target[cor_target>0.5]
relevant_features
```

Out[36]:
```
B Prog8    0.776091
B Att8     1.000000
M Att8     0.533864
Name: B Att8, dtype: float64
```

In [37]:
```
#Correlation with output variable
cor_target = abs(cor["A Att8"])
#Selecting highly correlated features
relevant_features = cor_target[cor_target>0.5]
relevant_features
```

Out[37]:
```
A Prog8    0.599283
M Prog8    0.530175
A Att8     1.000000
M Att8     0.574550
Name: A Att8, dtype: float64
```

In [38]:
```
#Correlation with output variable
cor_target = abs(cor["M Att8"])
#Selecting highly correlated features
relevant_features = cor_target[cor_target>0.5]
relevant_features
```

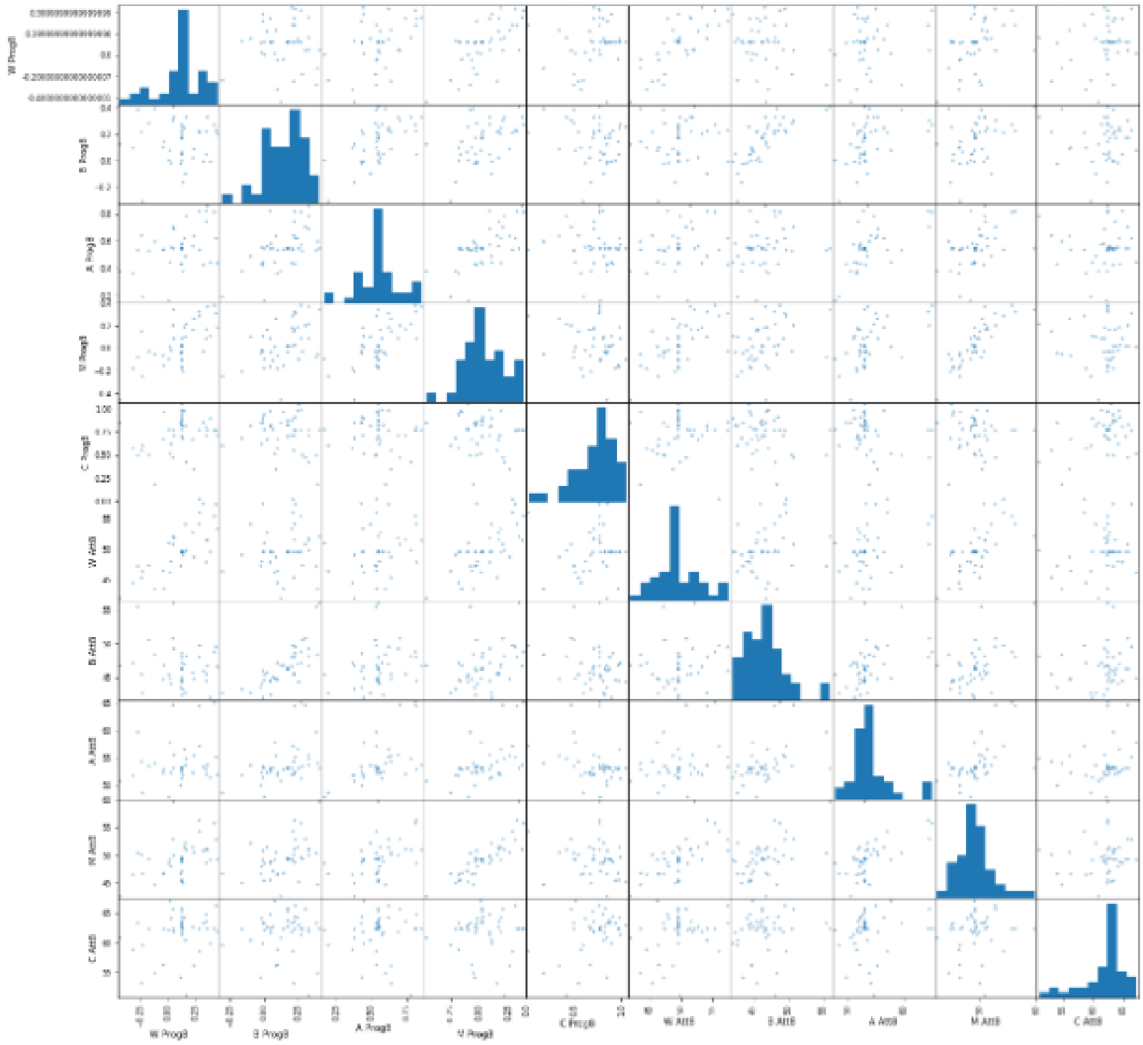
Out[38]:
```
M Prog8    0.860364
W Att8     0.679788
B Att8     0.533864
A Att8     0.574550
M Att8     1.000000
Name: M Att8, dtype: float64
```

In [39]:
```
#Correlation with output variable
cor_target = abs(cor["C Att8"])
#Selecting highly correlated features
relevant_features = cor_target[cor_target>0.5]
relevant_features
```

Out[39]:
```
C Att8    1.0
Name: C Att8, dtype: float64
```

In [40]:

```
#Scatter Matrix for Data Exploration Purposes
scatter_matrix(new_full_df, figsize=(20,20))
plt.xticks(rotation=90);
plt.yticks(rotation=0);
pyplot.show()
```



In [41]:

```
#White Multiple Linear Regression Model
import pandas as pd
from sklearn import linear_model
import statsmodels.api as sm
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score

fxw = new_full_df[['W Prog8', 'M Prog8']]
fyw = new_full_df[['W Att8']]

Xw = fxw
yw = fyw

X_train, X_test, y_train, y_test = train_test_split(Xw, yw, test_size = 0.2, ran

# with sklearn
regr = linear_model.LinearRegression()
regr.fit(X_train, y_train)

y_pred = regr.predict(X_test)

test_set_rmse = (np.sqrt(mean_squared_error(y_test, y_pred)))

test_set_r2 = r2_score(y_test, y_pred)

print('RMSE: \n', test_set_rmse)
print('R2: \n', test_set_r2)


print('Intercept: \n', regr.intercept_)
print('Coefficients: \n', regr.coef_)

# prediction with sklearn
print ('Predicted Attainment8 Score: \n', regr.predict(X_test))

# with statsmodels
Xw = sm.add_constant(Xw) # adding a constant

model = sm.OLS(yw, Xw).fit()
predictions = model.predict(Xw)

print_model = model.summary()
print(print_model)
```

```
RMSE:
 2.0185795612479325
R2:
 0.4169192809673864
Intercept:
 [48.28332931]
Coefficients:
 [[10.83101151  3.40029468]]
Predicted Attainment8 Score:
 [[48.46804701]
 [47.53933637]
 [52.41260756]
 [49.35730558]
 [49.4911179 ]
 [49.67813411]
 [48.00595563]
 [50.31066629]
 [49.9448767 ]]
                            OLS Regression Results
==============================================================================
Dep. Variable:                 W Att8   R-squared:                       0.628
Model:                            OLS   Adj. R-squared:                  0.610
Method:                 Least Squares   F-statistic:                     34.57
Date:                Tue, 06 Apr 2021   Prob (F-statistic):           1.60e-09
Time:                        23:19:18   Log-Likelihood:                -94.545
No. Observations:                  44   AIC:                             195.1
Df Residuals:                      41   BIC:                             200.4
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         48.4520      0.366    132.333      0.000      47.713      49.191
W Prog8       10.1872      2.127      4.790      0.000       5.892      14.482
M Prog8        4.7508      2.384      1.993      0.053      -0.063       9.565
==============================================================================
Omnibus:                        1.198   Durbin-Watson:                   1.976
Prob(Omnibus):                  0.549   Jarque-Bera (JB):                0.478
Skew:                           0.180   Prob(JB):                        0.787
Kurtosis:                       3.362   Cond. No.                         9.05
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctl
y specified.
```

]:
```
#Testing Black Linear Regression Model

import pandas as pd
from sklearn import linear_model
import statsmodels.api as sm
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score

fxb = new_full_df[['B Prog8']]
fyb = new_full_df[['B Att8']]

Xb = fxb
yb = fyb

X_train, X_test, y_train, y_test = train_test_split(Xb, yb, test_size = 0.2, ran

# with sklearn
regr = linear_model.LinearRegression()
regr.fit(X_train, y_train)

y_pred = regr.predict(X_test)

test_set_rmse = (np.sqrt(mean_squared_error(y_test, y_pred)))

test_set_r2 = r2_score(y_test, y_pred)

print('RMSE: \n', test_set_rmse)
print('R2: \n', test_set_r2)


print('Intercept: \n', regr.intercept_)
print('Coefficients: \n', regr.coef_)

# prediction with sklearn
print ('Predicted Attainment8 Score: \n', regr.predict(X_test))

# with statsmodels
Xb = sm.add_constant(Xb) # adding a constant

model = sm.OLS(yb, Xb).fit()
predictions = model.predict(Xb)

print_model = model.summary()
print(print_model)
```

```
RMSE:
 2.334113139048039
R2:
 0.6431113359472609
Intercept:
 [44.74606405]
Coefficients:
 [[12.77502373]]
Predicted Attainment8 Score:
 [[44.52250113]
 [46.02356642]
 [47.58850683]
 [44.44798016]
 [45.38481523]
 [46.94975564]
 [44.77800161]
 [43.50049923]
 [49.91994866]]
                            OLS Regression Results
==============================================================================
Dep. Variable:                 B Att8   R-squared:                       0.602
Model:                            OLS   Adj. R-squared:                  0.593
Method:                 Least Squares   F-statistic:                     63.61
Date:                Tue, 06 Apr 2021   Prob (F-statistic):           6.05e-10
Time:                        23:19:18   Log-Likelihood:                -88.954
No. Observations:                  44   AIC:                             181.9
Df Residuals:                      42   BIC:                             185.5
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         44.4208      0.394    112.672      0.000      43.625      45.216
B Prog8       15.0270      1.884      7.976      0.000      11.225      18.829
==============================================================================
Omnibus:                        9.058   Durbin-Watson:                   2.057
Prob(Omnibus):                  0.011   Jarque-Bera (JB):                8.581
Skew:                           0.820   Prob(JB):                       0.0137
Kurtosis:                       4.412   Cond. No.                         6.83
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctl
y specified.
```

In [43]:

```
#Asian Multiple Linear Regression Test
import pandas as pd
from sklearn import linear_model
import statsmodels.api as sm
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score

fxa = new_full_df[['A Prog8', 'M Prog8']]
fya = new_full_df[['A Att8']]

Xa = fxa
ya = fya

X_train, X_test, y_train, y_test = train_test_split(Xa, ya, test_size = 0.2, ran

# with sklearn
regr = linear_model.LinearRegression()
regr.fit(X_train, y_train)

y_pred = regr.predict(X_test)

test_set_rmse = (np.sqrt(mean_squared_error(y_test, y_pred)))

test_set_r2 = r2_score(y_test, y_pred)

print('RMSE: \n', test_set_rmse)
print('R2: \n', test_set_r2)

print('Intercept: \n', regr.intercept_)
print('Coefficients: \n', regr.coef_)

# prediction with sklearn
print ('Predicted Attainment8 Score: \n', regr.predict(X_test))

# with statsmodels
Xa = sm.add_constant(Xa) # adding a constant

model = sm.OLS(ya, Xa).fit()
predictions = model.predict(Xa)

print_model = model.summary()
print(print_model)
```

```
RMSE:
 2.961706534090485
R2:
 0.5301911944540938
Intercept:
 [50.08134858]
Coefficients:
 [[6.73827809 5.48653072]]
Predicted Attainment8 Score:
 [[53.7113623 ]
 [53.31598653]
 [57.35929707]
 [52.7456306 ]
 [53.81436699]
 [53.8718636 ]
 [50.64138202]
 [53.94547424]
 [55.62755343]]
                            OLS Regression Results
==============================================================================
Dep. Variable:                 A Att8   R-squared:                       0.392
Model:                            OLS   Adj. R-squared:                  0.362
Method:                 Least Squares   F-statistic:                     13.21
Date:                Tue, 06 Apr 2021   Prob (F-statistic):           3.73e-05
Time:                        23:19:18   Log-Likelihood:                -107.74
No. Observations:                  44   AIC:                             221.5
Df Residuals:                      41   BIC:                             226.8
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         47.6572      2.162     22.039      0.000      43.290      52.024
A Prog8       10.8993      3.987      2.734      0.009       2.847      18.952
M Prog8        4.7905      3.223      1.486      0.145      -1.718      11.299
==============================================================================
Omnibus:                       13.503   Durbin-Watson:                   2.091
Prob(Omnibus):                  0.001   Jarque-Bera (JB):               14.427
Skew:                           1.157   Prob(JB):                     0.000737
Kurtosis:                       4.587   Cond. No.                         13.4
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctl
y specified.
```

In [46]:

```
#Mixed Linear Regression Model Predictions
import pandas as pd
from sklearn import linear_model
import statsmodels.api as sm
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score

fxm = new_full_df[['M Prog8']]
fym = new_full_df[['M Att8']]

Xm = fxm
ym = fym

X_train, X_test, y_train, y_test = train_test_split(Xm, ym, test_size = 0.2, ran


# with sklearn
regr = linear_model.LinearRegression()
regr.fit(X_train, y_train)

y_pred = regr.predict(X_test)

test_set_rmse = (np.sqrt(mean_squared_error(y_test, y_pred)))

test_set_r2 = r2_score(y_test, pred)

print(test_set_rmse)
print(test_set_r2)

print('Intercept: \n', regr.intercept_)
print('Coefficients: \n', regr.coef_)

# prediction with sklearn
print ('Predicted Attainment8 Score: \n', regr.predict(X_test))

# with statsmodels
Xm = sm.add_constant(Xm) # adding a constant

model = sm.OLS(ym, Xm).fit()
predictions = model.predict(Xm)

print_model = model.summary()
print(print_model)
```

```
2.6835051758085506
0.6094576563641783
Intercept:
 [48.62543025]
Coefficients:
 [[13.24249494]]
Predicted Attainment8 Score:
 [[48.36058035]
 [47.73156184]
 [52.89613487]
 [48.06262421]
 [48.16194293]
 [48.89028015]
 [47.33428699]
 [49.98278598]
 [53.12787853]]
                            OLS Regression Results
==============================================================================
Dep. Variable:                 M Att8   R-squared:                       0.740
Model:                            OLS   Adj. R-squared:                  0.734
Method:                 Least Squares   F-statistic:                     119.7
Date:                Wed, 07 Apr 2021   Prob (F-statistic):           7.18e-14
Time:                        00:32:42   Log-Likelihood:                -84.301
No. Observations:                  44   AIC:                             172.6
Df Residuals:                      42   BIC:                             176.2
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         48.7919      0.260    187.401      0.000      48.266      49.317
M Prog8       15.4172      1.409     10.940      0.000      12.573      18.261
==============================================================================
Omnibus:                        8.297   Durbin-Watson:                   2.257
Prob(Omnibus):                  0.016   Jarque-Bera (JB):                9.022
Skew:                           0.627   Prob(JB):                       0.0110
Kurtosis:                       4.830   Cond. No.                         5.57
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctl
y specified.
```

In [47]:

```
#Chinese Linear Regression Model Predictions
import pandas as pd
from sklearn import linear_model
import statsmodels.api as sm
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score

fxc = new_full_df[['C Prog8']]
fyc = new_full_df[['C Att8']]

Xc = fxc
yc = fyc

X_train, X_test, y_train, y_test = train_test_split(Xc, yc, test_size = 0.2, ran


# with sklearn
regr = linear_model.LinearRegression()
regr.fit(X_train, y_train)

y_pred = regr.predict(X_test)

test_set_rmse = (np.sqrt(mean_squared_error(y_test, y_pred)))

test_set_r2 = r2_score(y_test, pred)

print(test_set_rmse)
print(test_set_r2)

print('Intercept: \n', regr.intercept_)
print('Coefficients: \n', regr.coef_)

# prediction with sklearn
print ('Predicted Attainment8 Score: \n', regr.predict(X_test))

# with statsmodels
Xc = sm.add_constant(Xc) # adding a constant

model = sm.OLS(yc, Xc).fit()
predictions = model.predict(Xc)

print_model = model.summary()
print(print_model)
```

```
3.835507826800202
-11.472753738211756
Intercept:
 [56.34599476]
Coefficients:
 [[7.62976366]]
Predicted Attainment8 Score:
 [[61.07644823]
 [61.45793641]
 [62.22727091]
 [62.83129387]
 [62.62147537]
 [61.68682932]
 [62.27177787]
 [62.57696841]
 [62.22727091]]
                            OLS Regression Results
==============================================================================
Dep. Variable:                 C Att8   R-squared:                       0.171
Model:                            OLS   Adj. R-squared:                  0.151
Method:                 Least Squares   F-statistic:                     8.646
Date:                Wed, 07 Apr 2021   Prob (F-statistic):            0.00531
Time:                        00:33:04   Log-Likelihood:                -113.35
No. Observations:                  44   AIC:                             230.7
Df Residuals:                      42   BIC:                             234.3
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         56.6852      1.743     32.523      0.000      53.168      60.203
C Prog8        6.8355      2.325      2.940      0.005       2.144      11.527
==============================================================================
Omnibus:                        7.276   Durbin-Watson:                   1.319
Prob(Omnibus):                  0.026   Jarque-Bera (JB):                6.248
Skew:                          -0.726   Prob(JB):                       0.0440
Kurtosis:                       4.139   Cond. No.                         7.26
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctl
y specified.
```

In [4]:
```python
# Pandas is used for data manipulation
import pandas as pd
# Read in data and display first 5 rows
feature = pd.read_csv('Fixed_data.csv', converters={'Ethnicity': str.strip})
feature.head(5)
```

Out[4]:

	Code	Area	Year	Sex	Ethnicity	Pupils	Attainment8	Progress8
0	E09000002	Barking and Dagenham	2018/19	All	White	903.0	41.7	-0.21
1	E09000002	Barking and Dagenham	2018/19	All	Mixed	227.0	44.7	0.05
2	E09000002	Barking and Dagenham	2018/19	All	Asian	530.0	53.3	0.60
3	E09000002	Barking and Dagenham	2018/19	All	Black	642.0	48.0	0.37
4	E09000002	Barking and Dagenham	2018/19	All	Chinese	3.0	69.0	0.93

In [5]:
```python
feature['Pupils'] = feature['Pupils'].fillna(feature['Pupils'].mean())
feature['Attainment8'] = feature['Attainment8'].fillna(feature['Attainment8'].me
feature['Progress8'] = feature['Progress8'].fillna(feature['Progress8'].mean())
```

In [6]:
```python
feature.isnull().sum()
```

Out[6]:
```
Code           0
Area           0
Year           0
Sex            0
Ethnicity      0
Pupils         0
Attainment8    0
Progress8      0
dtype: int64
```

In [7]:
```python
print('The shape of our features is:', feature.shape)
```

```
The shape of our features is: (3240, 8)
```

In [8]:
```
feature.describe()
```

Out[8]:

	Pupils	Attainment8	Progress8
count	3240.000000	3240.000000	3240.000000
mean	6146.238585	52.011719	0.285722
std	30185.585452	7.607603	0.417303
min	1.000000	15.000000	-0.860000
25%	131.750000	47.000000	0.010000
50%	450.000000	50.800000	0.270000
75%	1766.750000	54.900000	0.510000
max	540006.000000	89.000000	2.820000

In [9]:
```
features = feature.drop(columns='Code')
```

In [10]:
```
# One-hot encode the data using pandas get_dummies
features = pd.get_dummies(features)
# Display the first 5 rows of the last 12 columns
features.iloc[:,5:].head(5)
```

Out[10]:

	Area_Bexley	Area_Brent	Area_Bromley	Area_Camden	Area_City of London	Area_Croydon	Area_Ealing
0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0

5 rows × 56 columns

In [11]:

```
# Use numpy to convert to arrays
import numpy as np
# Labels are the values we want to predict
labels = np.array(features['Attainment8'])
# Remove the labels from the features
# axis 1 refers to the columns
features= features.drop('Attainment8', axis = 1)
# Saving feature names for later use
feature_list = list(features.columns)
# Convert to numpy array
features = np.array(features)
```

In [12]:

```
# Using Skicit-learn to split data into training and testing sets
from sklearn.model_selection import train_test_split
# Split the data into training and testing sets
train_features, test_features, train_labels, test_labels = train_test_split(feat
```

In [13]:

```
print('Training Features Shape:', train_features.shape)
print('Training Labels Shape:', train_labels.shape)
print('Testing Features Shape:', test_features.shape)
print('Testing Labels Shape:', test_labels.shape)
```

```
Training Features Shape: (2430, 60)
Training Labels Shape: (2430,)
Testing Features Shape: (810, 60)
Testing Labels Shape: (810,)
```

In [30]:

```
# The baseline predictions are the historical averages
baseline_preds = test_features[:, feature_list.index('Progress8')]
# Baseline errors, and display average baseline error
baseline_errors = abs(baseline_preds - test_labels)
print('Average baseline error: ', round(np.mean(baseline_errors), 2))
```

```
Average baseline error:  51.58
```

In [18]:

```
# Import the model we are using
from sklearn.ensemble import RandomForestRegressor
# Instantiate model with 1000 decision trees
rf = RandomForestRegressor(n_estimators = 1000, random_state = 42)
# Train the model on training data
rf.fit(train_features, train_labels);
```

In [19]:

```
# Use the forest's predict method on the test data
predictions = rf.predict(test_features)
# Calculate the absolute errors
errors = abs(predictions - test_labels)
# Print out the mean absolute error (mae)
print('Mean Absolute Error:', round(np.mean(errors), 2))
```

```
Mean Absolute Error: 1.77
```

In [20]:

```
# Calculate mean absolute percentage error (MAPE)
mape = 100 * (errors / test_labels)
# Calculate and display accuracy
accuracy = 100 - np.mean(mape)
print('Accuracy:', round(accuracy, 2), '%.')
```

```
Accuracy: 96.38 %.
```

In [21]:

```
# Import tools needed for visualization
from sklearn.tree import export_graphviz
import pydot
# Pull out one tree from the forest
tree = rf.estimators_[5]
# Import tools needed for visualization
from sklearn.tree import export_graphviz
import pydot
# Pull out one tree from the forest
tree = rf.estimators_[5]
# Export the image to a dot file
export_graphviz(tree, out_file = 'tree.dot', feature_names = feature_list, round
# Use dot file to create a graph
(graph, ) = pydot.graph_from_dot_file('tree.dot')
# Write graph to a png file
graph.write_png('tree.png')
```

In [22]:

```
# Limit depth of tree to 3 levels
rf_small = RandomForestRegressor(n_estimators=10, max_depth = 3)
rf_small.fit(train_features, train_labels)
# Extract the small tree
tree_small = rf_small.estimators_[5]
# Save the tree as a png image
export_graphviz(tree_small, out_file = 'small_tree.dot', feature_names = feature_
(graph, ) = pydot.graph_from_dot_file('small_tree.dot')
graph.write_png('small_tree.png');
```

In [23]:

```
# Get numerical feature importances
importances = list(rf.feature_importances_)
# List of tuples with variable and importance
feature_importances = [(feature, round(importance, 2)) for feature, importance i
# Sort the feature importances by most important first
feature_importances = sorted(feature_importances, key = lambda x: x[1], reverse 
# Print out the feature and importances
[print('Variable: {:20} Importance: {}'.format(*pair)) for pair in feature_impor
```

```
Variable: Ethnicity_Chinese    Importance: 0.45
Variable: Progress8            Importance: 0.27
Variable: Pupils               Importance: 0.05
Variable: Year_2015/16         Importance: 0.03
Variable: Ethnicity_Black      Importance: 0.03
Variable: Area_Sutton          Importance: 0.02
Variable: Area_Barnet          Importance: 0.01
Variable: Area_Bexley          Importance: 0.01
Variable: Area_Bromley         Importance: 0.01
Variable: Area_Kensington and Chelsea Importance: 0.01
Variable: Area_Kingston upon Thames Importance: 0.01
Variable: Area_Waltham Forest  Importance: 0.01
Variable: Year_2018/19         Importance: 0.01
Variable: Sex_Boys             Importance: 0.01
Variable: Sex_Girls            Importance: 0.01
Variable: Area_Barking and Dagenham Importance: 0.0
Variable: Area_Brent           Importance: 0.0
Variable: Area_Camden          Importance: 0.0
Variable: Area_City of London  Importance: 0.0
Variable: Area_Croydon         Importance: 0.0
Variable: Area_Ealing          Importance: 0.0
Variable: Area_East            Importance: 0.0
Variable: Area_East Midlands   Importance: 0.0
Variable: Area_Enfield         Importance: 0.0
Variable: Area_England         Importance: 0.0
Variable: Area_Greenwich       Importance: 0.0
Variable: Area_Hackney         Importance: 0.0
Variable: Area_Hammersmith and Fulham Importance: 0.0
Variable: Area_Haringey        Importance: 0.0
Variable: Area_Harrow          Importance: 0.0
Variable: Area_Havering        Importance: 0.0
Variable: Area_Hillingdon      Importance: 0.0
Variable: Area_Hounslow        Importance: 0.0
Variable: Area_Inner London    Importance: 0.0
Variable: Area_Islington       Importance: 0.0
```

```
Variable: Area_Lambeth         Importance: 0.0
Variable: Area_Lewisham        Importance: 0.0
Variable: Area_London          Importance: 0.0
Variable: Area_Merton          Importance: 0.0
Variable: Area_Newham          Importance: 0.0
Variable: Area_North East      Importance: 0.0
Variable: Area_North West      Importance: 0.0
Variable: Area_Outer London    Importance: 0.0
Variable: Area_Redbridge       Importance: 0.0
Variable: Area_Richmond upon Thames Importance: 0.0
Variable: Area_South East      Importance: 0.0
Variable: Area_South West      Importance: 0.0
Variable: Area_Southwark       Importance: 0.0
Variable: Area_Tower Hamlets   Importance: 0.0
Variable: Area_Wandsworth      Importance: 0.0
Variable: Area_West Midlands   Importance: 0.0
Variable: Area_Westminster     Importance: 0.0
Variable: Area_Yorkshire and the Humber Importance: 0.0
Variable: Year_2016/17         Importance: 0.0
Variable: Year_2017/18         Importance: 0.0
Variable: Sex_All              Importance: 0.0
Variable: Ethnicity_All Pupils Importance: 0.0
Variable: Ethnicity_Asian      Importance: 0.0
Variable: Ethnicity_Mixed      Importance: 0.0
Variable: Ethnicity_White      Importance: 0.0
```

In [24]:
```
# New random forest with only the two most important variables
rf_most_important = RandomForestRegressor(n_estimators= 1000, random_state=42)
# Extract the two most important features
important_indices = [feature_list.index('Ethnicity_Chinese'), feature_list.index
train_important = train_features[:, important_indices]
test_important = test_features[:, important_indices]
# Train the random forest
rf_most_important.fit(train_important, train_labels)
# Make predictions and determine the error
predictions = rf_most_important.predict(test_important)
errors = abs(predictions - test_labels)
# Display the performance metrics
print('Mean Absolute Error:', round(np.mean(errors), 2))
mape = np.mean(100 * (errors / test_labels))
accuracy = 100 - mape
print('Accuracy:', round(accuracy, 2), '%.')
```

```
Mean Absolute Error: 3.23
Accuracy: 93.59 %.
```

In [ ]:

In [25]:
```
# New random forest with only the two most important variables
rf_most_important = RandomForestRegressor(n_estimators= 1000, random_state=42)
# Extract the two most important features
important_indices = [feature_list.index('Ethnicity_Black'), feature_list.index('
train_important = train_features[:, important_indices]
test_important = test_features[:, important_indices]
# Train the random forest
rf_most_important.fit(train_important, train_labels)
# Make predictions and determine the error
predictions = rf_most_important.predict(test_important)
errors = abs(predictions - test_labels)
# Display the performance metrics
print('Mean Absolute Error:', round(np.mean(errors), 2))
mape = np.mean(100 * (errors / test_labels))
accuracy = 100 - mape
print('Accuracy:', round(accuracy, 2), '%.')
```

```
Mean Absolute Error: 3.49
Accuracy: 93.32 %.
```

In [26]:
```python
# New random forest with only the two most important variables
rf_most_important = RandomForestRegressor(n_estimators= 1000, random_state=42)
# Extract the two most important features
important_indices = [feature_list.index('Ethnicity_Mixed'), feature_list.index('
train_important = train_features[:, important_indices]
test_important = test_features[:, important_indices]
# Train the random forest
rf_most_important.fit(train_important, train_labels)
# Make predictions and determine the error
predictions = rf_most_important.predict(test_important)
errors = abs(predictions - test_labels)
# Display the performance metrics
print('Mean Absolute Error:', round(np.mean(errors), 2))
mape = np.mean(100 * (errors / test_labels))
accuracy = 100 - mape
print('Accuracy:', round(accuracy, 2), '%.')
```

```
Mean Absolute Error: 3.66
Accuracy: 92.95 %.
```

In [ ]:

In [27]:
```python
# New random forest with only the two most important variables
rf_most_important = RandomForestRegressor(n_estimators= 1000, random_state=42)
# Extract the two most important features
important_indices = [feature_list.index('Ethnicity_White'), feature_list.index('
train_important = train_features[:, important_indices]
test_important = test_features[:, important_indices]
# Train the random forest
rf_most_important.fit(train_important, train_labels)
# Make predictions and determine the error
predictions = rf_most_important.predict(test_important)
errors = abs(predictions - test_labels)
# Display the performance metrics
print('Mean Absolute Error:', round(np.mean(errors), 2))
mape = np.mean(100 * (errors / test_labels))
accuracy = 100 - mape
print('Accuracy:', round(accuracy, 2), '%.')
```

```
Mean Absolute Error: 3.74
Accuracy: 92.78 %.
```

In [28]:
```
# New random forest with only the two most important variables
rf_most_important = RandomForestRegressor(n_estimators= 1000, random_state=42)
# Extract the two most important features
important_indices = [feature_list.index('Ethnicity_Asian'), feature_list.index('I
train_important = train_features[:, important_indices]
test_important = test_features[:, important_indices]
# Train the random forest
rf_most_important.fit(train_important, train_labels)
# Make predictions and determine the error
predictions = rf_most_important.predict(test_important)
errors = abs(predictions - test_labels)
# Display the performance metrics
print('Mean Absolute Error:', round(np.mean(errors), 2))
mape = np.mean(100 * (errors / test_labels))
accuracy = 100 - mape
print('Accuracy:', round(accuracy, 2), '%.')
```

```
Mean Absolute Error: 3.69
Accuracy: 92.86 %.
```

In [29]:

```
# Import matplotlib for plotting and use magic command for Jupyter Notebooks
import matplotlib.pyplot as plt
%matplotlib inline

plt.figure(figsize=(25 ,10))
# Set the style
plt.style.use('fivethirtyeight')
# list of x locations for plotting
x_values = list(range(len(importances)))
# Make a bar chart
plt.bar(x_values, importances, orientation = 'vertical')
# Tick labels for x axis
plt.xticks(x_values, feature_list, rotation='vertical')
# Axis labels and title
plt.ylabel('Importance'); plt.xlabel('Variable'); plt.title('Variable Importance
```