2020

Statistical data analysis using Bayesian linear regression

COS7005

ISMAIL BHAMJEE

Word count 3200 excluding tables and appendix

Contents

Introduction

This report will explain the development and implementation of the Bayesian statistics in the form of a linear regression. We will also go over same pre-processing methods of data and visualisations to aid in the preparation. Next, we will construct the network and compare it of a regular OLS linear regression. Lastly, we will try and predict new values which we will construct to see how it fairs.

Part 1

In this first part we will examine the chosen data set with visual aid. This is to allow us to get an understand of what the data represents and not just a cluster of numbers.

The data set is from UCL machine learning org and is a collection of data regarding a student, from academic prior data such as previous results and failures, personal circumstances such as parents work ethic and situation and personal aspirations such as wanting to pursue higher education.

The database consists of 650 records which span 34 attributes. They are three main type of records which are nominal which contain an array of specific values such as male job and female job.

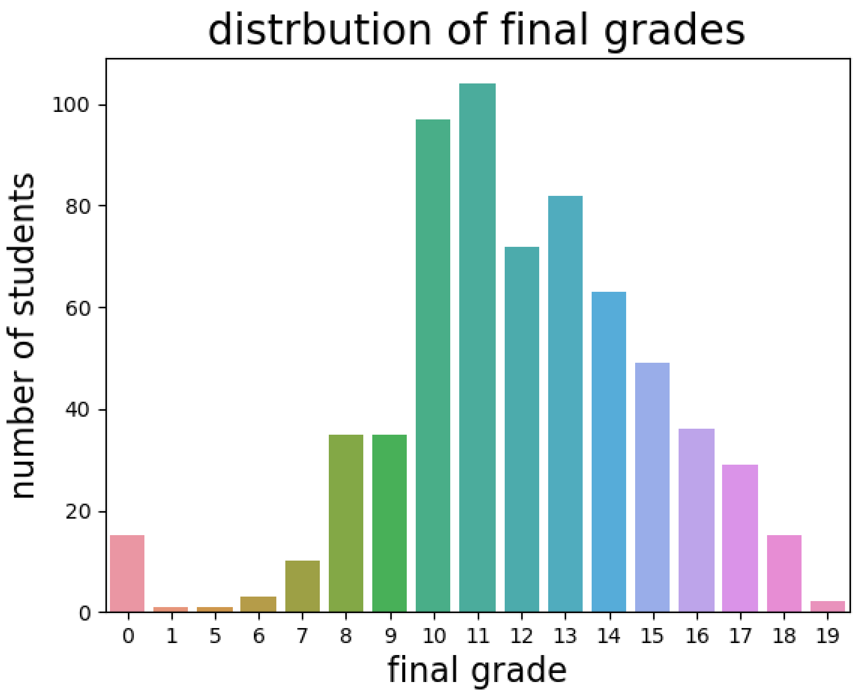
Next, we have numerical data such as the previous grades achieved by the student. Lastly, we have binary data which consists of two outcomes such as wanting to go on to higher education or romantic interests during the year.

A. Data reading and examining

For this task the datafile was given in a csv text file. So, during data analysis it was converted into a excel csv file and the quotation marks had to be removed as well as allocating the right value to the right column. An excel file was chosen as it allowed manual analysis of the data better than the text file.

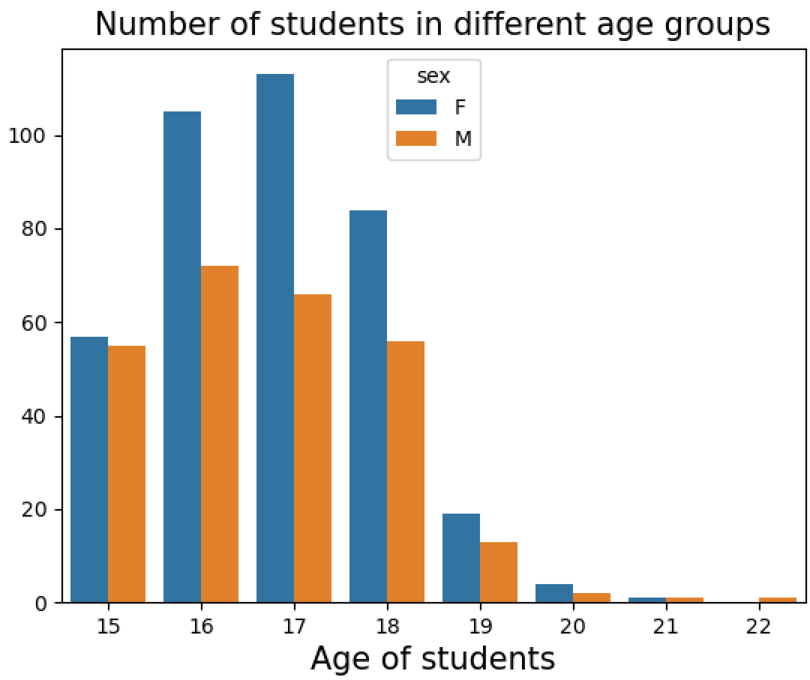
This was done using by using Pandas data frame which allowed the data frame to save to the excel file.

Distribution of grades



This figure shows us how many students got what grade and it allows us to see the normal distribution of the grades. furthermore, it shows us that most students received a final grade between 10 and 15. Next it shows that a small population of students failed.

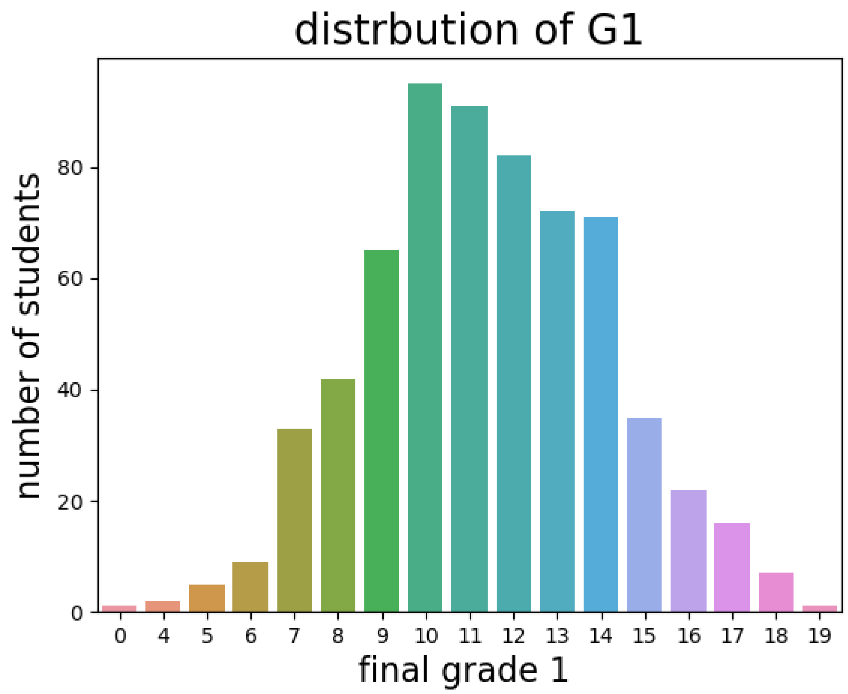
Numbers of students in age groups



This graph shows the age groups of the students and their gender. As we can see they are more female students than male students. The majority of them between 15 and 19.

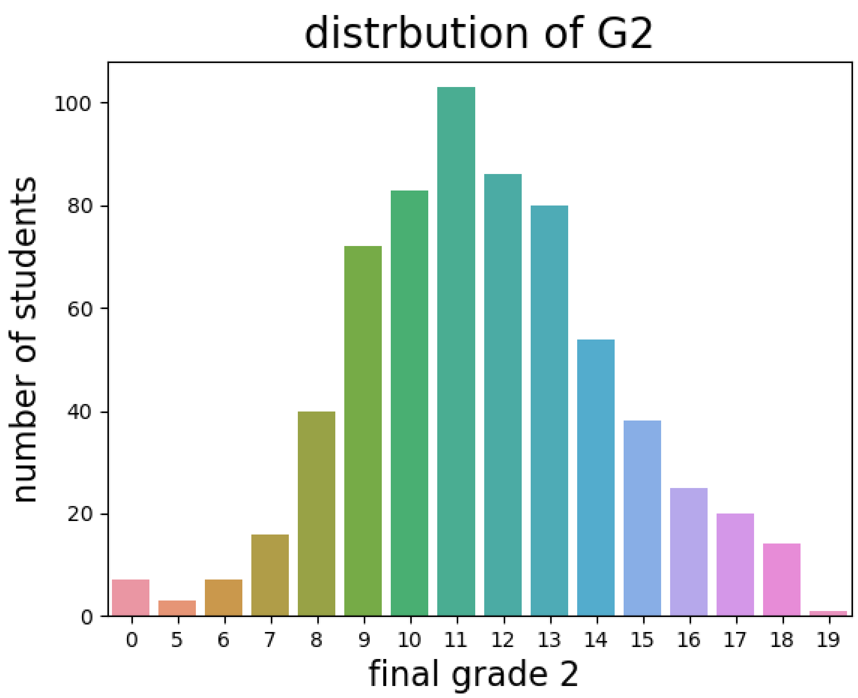
.

Distribution of G1



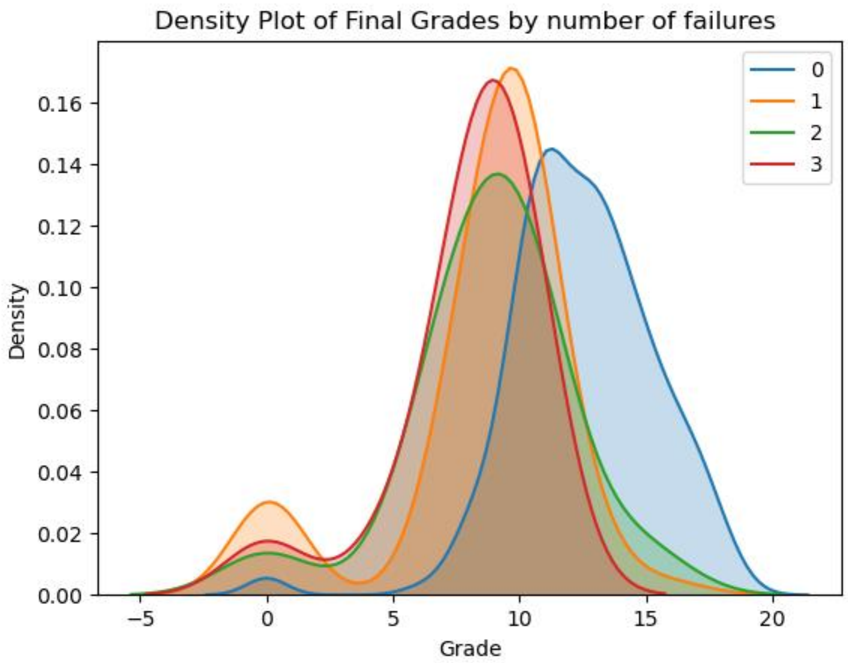
The above plot shows the distribution of the first grade and what the density of student was according to the score. We can see that the majority of students were between 9 and 14. Also there went as much failures as grade 3.

Distribution of G2



Grade 2 shows a slightly different story than the grade 1 as more student achieved higher grades and more students failed. But the overall distribution remains the same with middle scores being the most achieved.

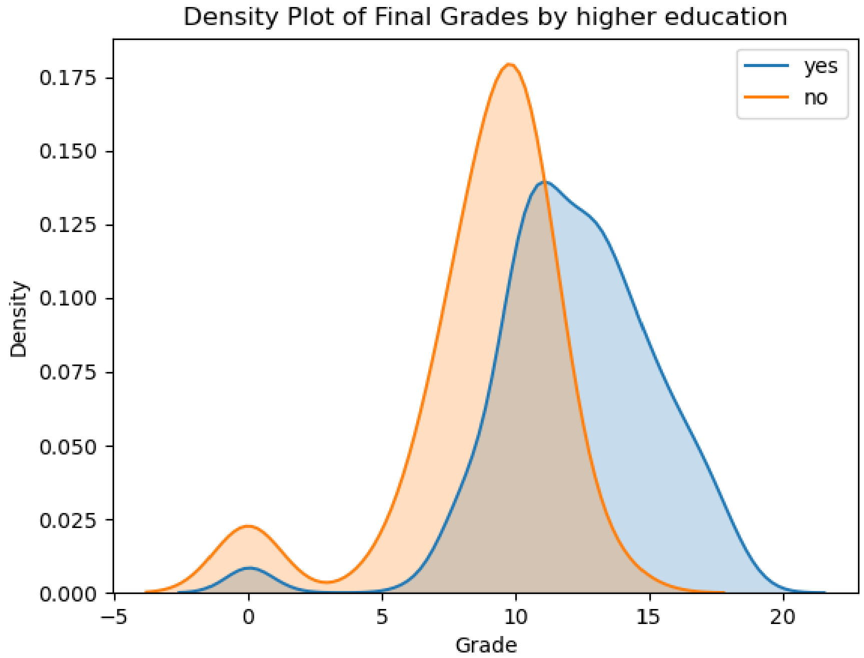
Grades by failures



The following is a density plot and it shows final grades with accordance with number of failures. Straight away you can see that students with 0 failures are prone to higher grades with a positive shift in the latter range furthermore the graph shows the density of students who got 0 marks to be lower than the rest.

On the contrary students with 3 failures tented to have more failures and overall lower marks, this is also shown in the peak as it is before the rest with a sharp decline not exceeding 15

Higher education vs non higher education aspirations



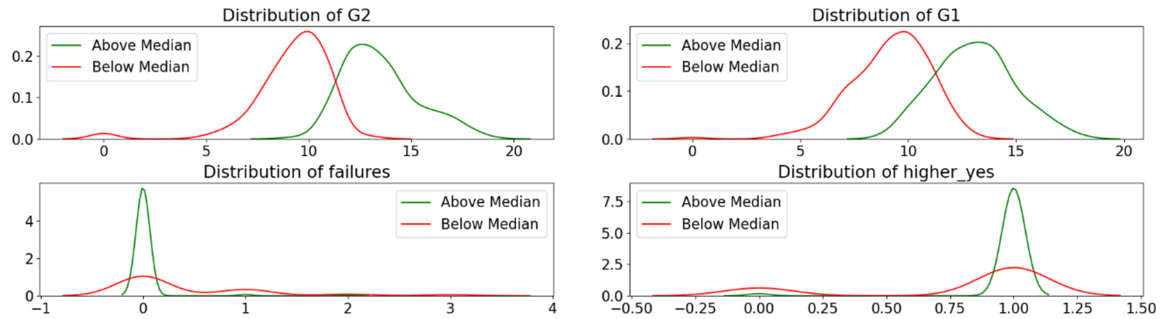
The above plot shows the density of students who want to go into higher education and those who do not. Straight away we can see the ones who want to go into higher have a broader span of latter grades which shows more achieved them. Next there the rise starts at a later score unlike the ones do not wish to go.

B. View statistical details like percentile, mean, std etc. of a data frame

	G3	G2	G1	Higher_yes	Failures
count	649	649	649	649	649
mean	11.906009	11.570108	11.399076	0.893683	0.221880
std	3.230656	2.913639	2.745265	0.308481	0.593235
min	0.000000	0.000000	0.000000	0	0
25%	10.000000	10.000000	10.000000	1.0	0
50%	12.000000	11.000000	11.000000	1.0	0
75%	14.000000	13.000000	13.000000	1.0	0
max	19.000000	19.000000	19.000000	10	3.0

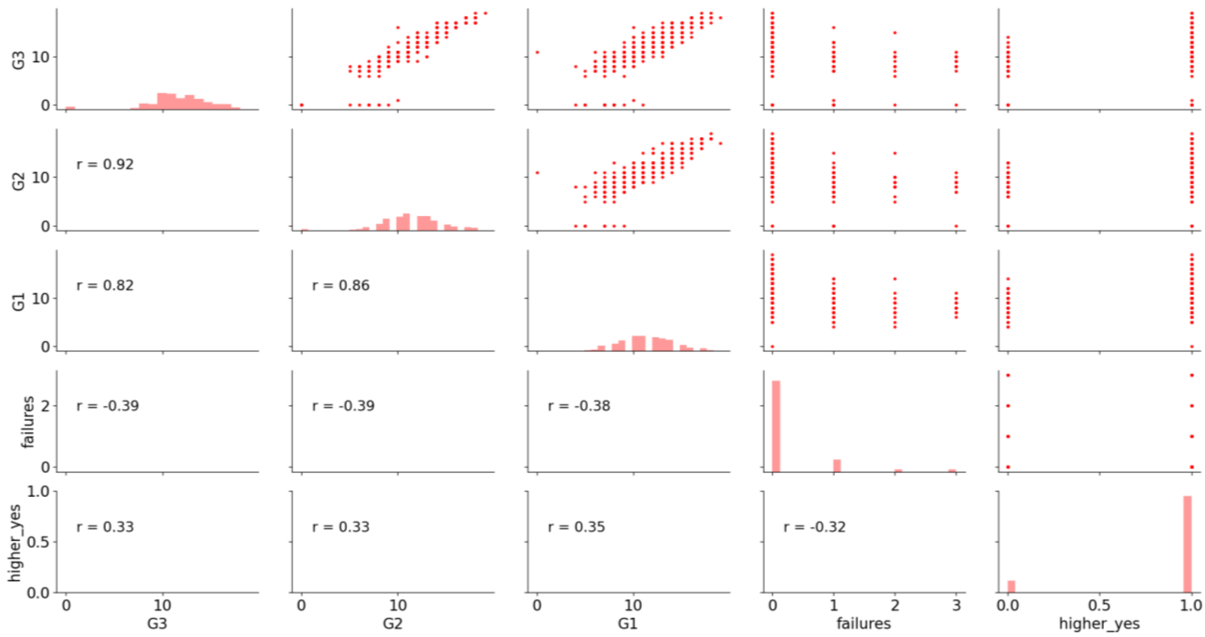
The above table shows the statistical data derived from the chosen attributes. With the 3 grade columns the numbers are fairly similar

C. Plot the distribution of various features.



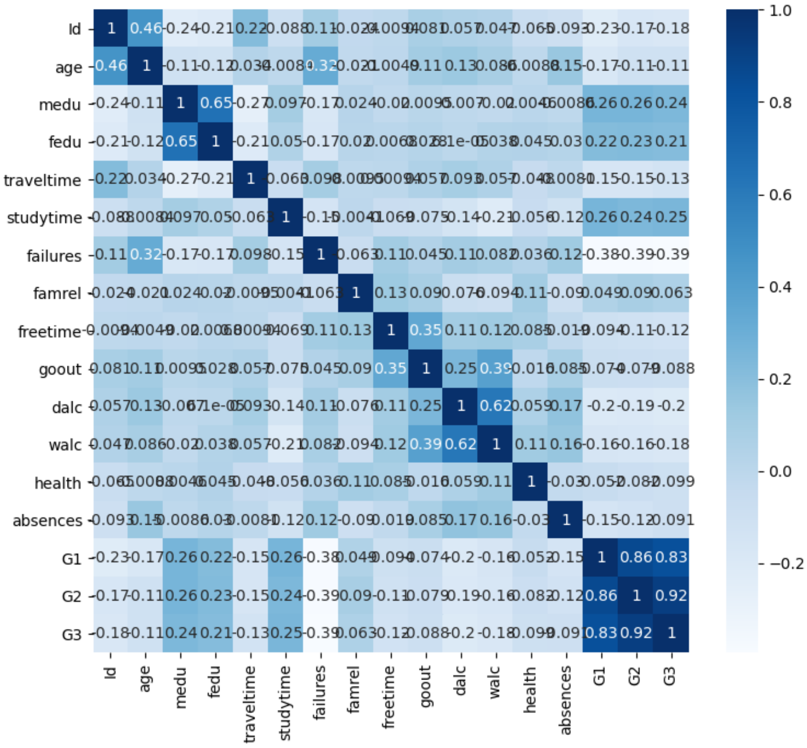
The green line in the above graphs shows students with a higher median and the red shows students which are below the median.

In the Distribution of failures plot it shows that students which 0 failures were well above the median grade 3. this is again seen the higher_yes. Students who wish to peruse a higher education are well above the median.



D.Calculated the features correlations with the final grade. This for both categorical and numerical values.

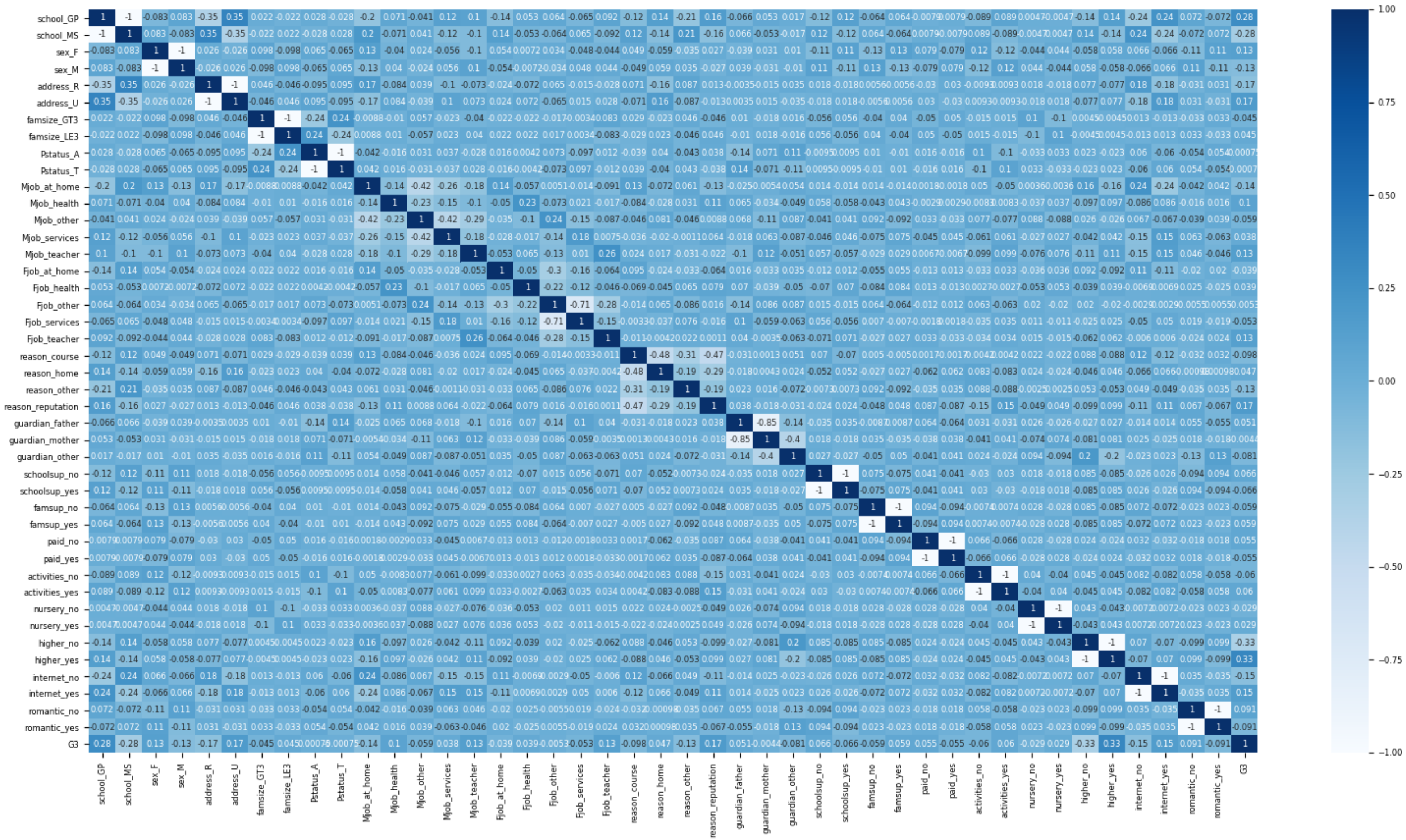
Correlation coefficient of numerical values



The correlation coefficient of nominal values using Pearson correlation coefficient. Fist they had to be converted into numerical using one hot encoding

Correlation coefficient of categorical and non-numerical values

The table bellow shows all coeffcites of the catergorical data using one hot encoding this is when the values are transformed into numerical values for example Higher_yes has two out puts a “yes” and a “no”, this will then be converted into a 1 or a 0, if theyre more than 2 outputs for an attribute then columns will be made for each output and a “1” will be in place of a true value.


school_GP
school_MS
sex_F
sex_M
address_R
address_U
famsize_GT3
famsize_LE3
Pstatus_A
Pstatus_T
Mjob_at_home
Mjob_health
Mjob_other
Mjob_services
Mjob_teacher
Fjob_at_home
Fjob_health
Fjob_other
Fjob_services
Fjob_teacher
reason_course
reason_home
reason_other
reason_reputation
guardian_father
guardian_mother
guardian_other
schoolsup_no
schoolsup_yes
famsup_no
famsup_yes
paid_no
paid_yes
activities_no
activities_yes
nursery_no
nursery_yes
higher_no
higher_yes
internet_no
internet_yes
romantic_no
romantic_yes
G3
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00

Top 4 most correlated attributes

G3	1.000000
G2	0.918548
G1	0.826387
failures	0.393316
higher_yes	0.332172

E. Implement a function that can select the "n" most correlated variables with the final score.

```
df = pd.read_csv("stats.csv")
# creates the lable by using the df column 'G3'
labels = df['G3']

# One-Hot Encoding of Categorical Variables
df_dummy = pd.get_dummies(df)

# Find correlations with the G3 both types.
most_correlated = df_dummy.corr().abs()['G3'].sort_values(ascending=False)

# Maintain the top x most correlation features with Grade
## can be used to reduce dimentionality
most_correlated2 = most_correlated[:6]

df3 = df_dummy.loc[:, most_correlated2.index]
#df = df.drop(columns='higher_no')

#Split into training and testing sets with with a desiried split
X_train, X_test, y_train, y_test = train_test_split(df3, labels,test_size=0.10,random_state=31)
```

This function allows the user to select the number of attributes which are corelated with the final grade" G3", this works for both nominal and numerical data, this is important when choosing the dimensionality of the model, as too many attributes can lead to slow processing times and over fitting.

Part two

For this part we will be selecting a performance measure. common performance measures for regression problems are MAE and RMSE. They give a figure on how much error the system makes in its prediction. The rmse uses the Euclidian distance between the sum of squares and the MAE uses the Manhattan distance.

The RMSE is more sensitive to outliers than the MAE. # add more

Rmse

Root mean square error this tells us the average of the squared differences between prediction and actual observation. This also means that larger errors have a higher weight. This method uses the Euclidian distance between instances (A straight path).

$$\mathrm{RMSE}(\mathbf{X}, h) = \sqrt{\frac{1}{m} \sum_{i=1}^{m} \left(h\left(\mathbf{x}^{(i)}\right) - y^{(i)}\right)^2}$$

Mae

This is the average over the given test sample of the absolute differences between actual observation and the prediction. Given all individual differences have equal weight. This method uses the Manhattan distance between instances (given path along the designated routes). With both measures the lower the better.

$$\mathrm{MAE}(\mathbf{X}, h) = \frac{1}{m} \sum_{i=1}^{m} \left| h\left(\mathbf{x}^{(i)}\right) - y^{(i)} \right|$$

The rsme and the mae changes as the dimensionality of data frame changes also it changes on the percentage of the test and train data frames

A.Implement a function to evaluate several statistical models by training on

the training set and testing on the test set.

In this section we will use alternative techniques to see where they lie in comparison to the baseline.

Here we have used 4 different models to compare with each of the two metrics

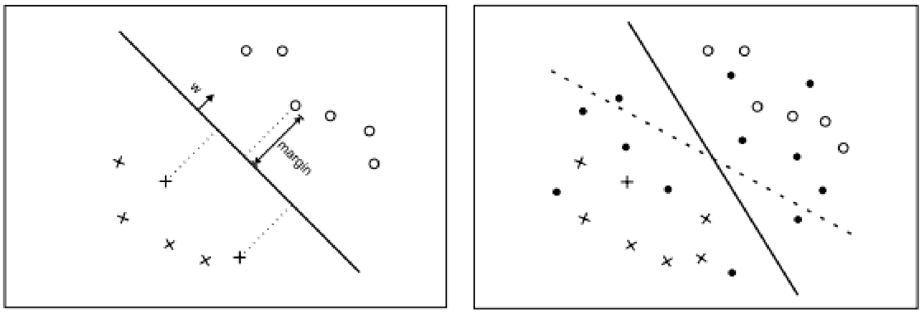
Linear regression

Linear regression is when there is a relationship between the independent variable and dependant variable given the function. This is often denoted at a straight line between the X and Y. so, as the independent variable changes the depend will changed according to the values. The regression line is based upon the least square's method. This is to minimise the difference between the estimate and the actual value known as error

B0 is the intercept t +b1 which is the slope and x being the value of independent variable

SVM

A support vector machine fines a Hyperplane between the data points. This a separation between them with the largest possible margin to provide a higher confidence when classifying a point.



(a) (b)

http://www.jmlr.org/papers/volume2/tong01a/tong01a.pdf

Gradient boost

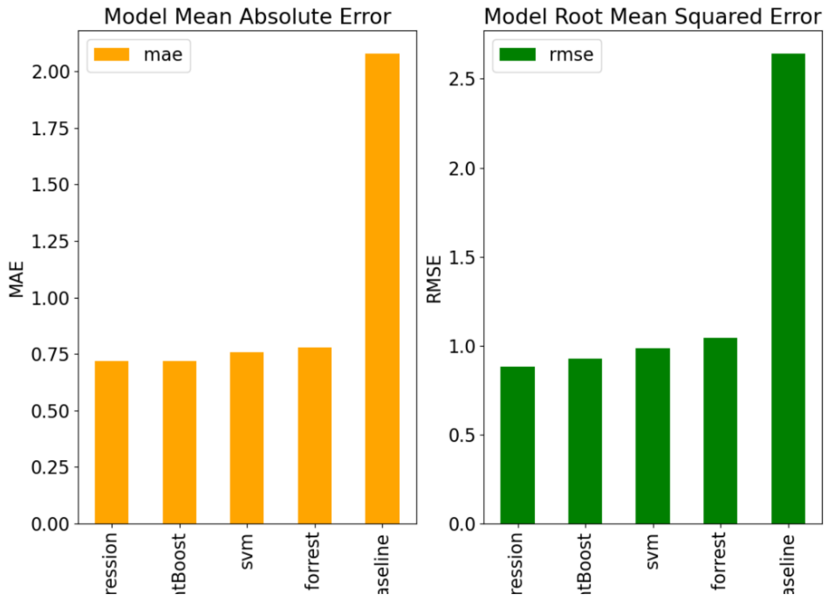
Gradient boosting is when weights are assigned in proportion of the strength of an outcome. It does this by assigning a loss function to the decision. The loss function is a measure showing how good the model's coefficient are at fitting the given data.in this regression task the loss function would be the error between the predicted outcome of grade and the actual outcome of grade.

Random Forest

‘This decision tree algorithm is based on the section of a random sample from the data set, when the data splits it adds another node to the tree and so on and so forth generating a ‘

In this given function the parameters taken in are the percentage split of the X and Y data. From there the models are then declared using python libraries and methods. Next each of the models will be given the X and Y train data to predict the X test.

Two figures are then derived to visualise where each of the models compares with each other and the baseline.



Results of them

model	mae	rmse
linear regression	0.718334	0.880918
random forrest	0.777933	1.04493
svm	0.757032	0.983363
Gradient Boost	0.71915	0.927297
Baseline	2.07692	2.63993

As we can see from all of the models the errors in both cases are less than the baseline error.

B. Implementing Bayesian Linear Regression

Model.

Linear regression

Linear regression allows to find a relationship between the independent variable (x) and the dependant variable (y) using a straight line through the points. So as the independent changes the dependant changes accordingly. This change can be either positive or negative depending on the independent.

To conduct regression, we take observations and try and find a line of best fit through all of the points. This is based on the least square's method, this is to minimise the distance between estimated value (given by the regression line) and the actual points.

This is done by using the formula

$$Y_i = \beta_0 + \beta_1 X_i + \varepsilon_i$$

Y_i is the dependant variable.

B_0 is the y intercept.

B_1 is the slope of the line.

X_i is the independent variable.

E_i is the random error.

Least squares method.

Given a set of points along an X and Y axis the regression line has to go through both means of X and Y. this is done by taking the distance between all observations and the given axis mean.

Bayesian linear regression is when a regression line is derived from using the probability distribution rather than pervious points estimate. Y is not a single value but rather a probability from the distribution.

The model for the Y prediction is derived from the normal distribution from the following formula.

$$y \sim N(X\beta, \sigma^2 \mathrm{I})$$

Y is given by the gaussian matrix.

- N is the normal distribution
- $\beta = (\beta 1....., \beta k)$ is the linear regression
- X is (N x K) matrix with each row being (Xi1....,Xik) of the independent variable
- σ^2 is the variance

The model parameter is also generated from a probability distribution. Hence the posterior probability is conditional to the training inputs and outputs. The is is given by the following formula as a version of Bayesian probability.

$$P(\beta|y,X) = \frac{P(y|\beta,X) * P(\beta|X)}{P(y|X)}$$

- The postior probability is **β** given **y** and **X**
- Likelihood is **y** given **β** and **X**
- Prior is **β** given **X**
- Predictor prior probability is **y** given **X**

Advantages of Bayesian linear regression.

Benefits of using Bayesian linear regression.

A OLS parameters are given by the data, however if we have overall knowledge of the process then we can guess the parameters.

The point of performing a Bayesian line regression is to gain a distribution of model parameters and this allows us to gain an idea of what the uncertainty of the model is in a numerical form.

Data points

The quantity of data affects the posterior distribution. The lower the count the distribution will be further spread out; where as if the count is higher the likelihood replaces the prior and with a potential infinite data stream they would be the same results as a linear regression.

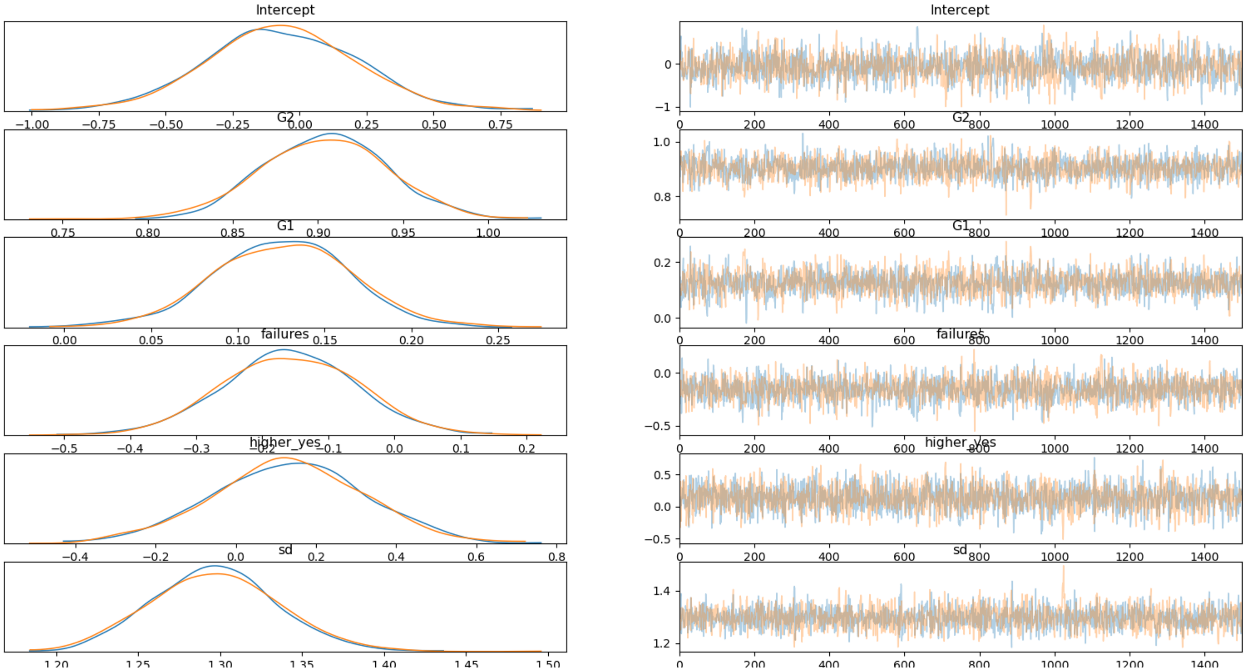
Sampling using the monte Carlo markov chain.

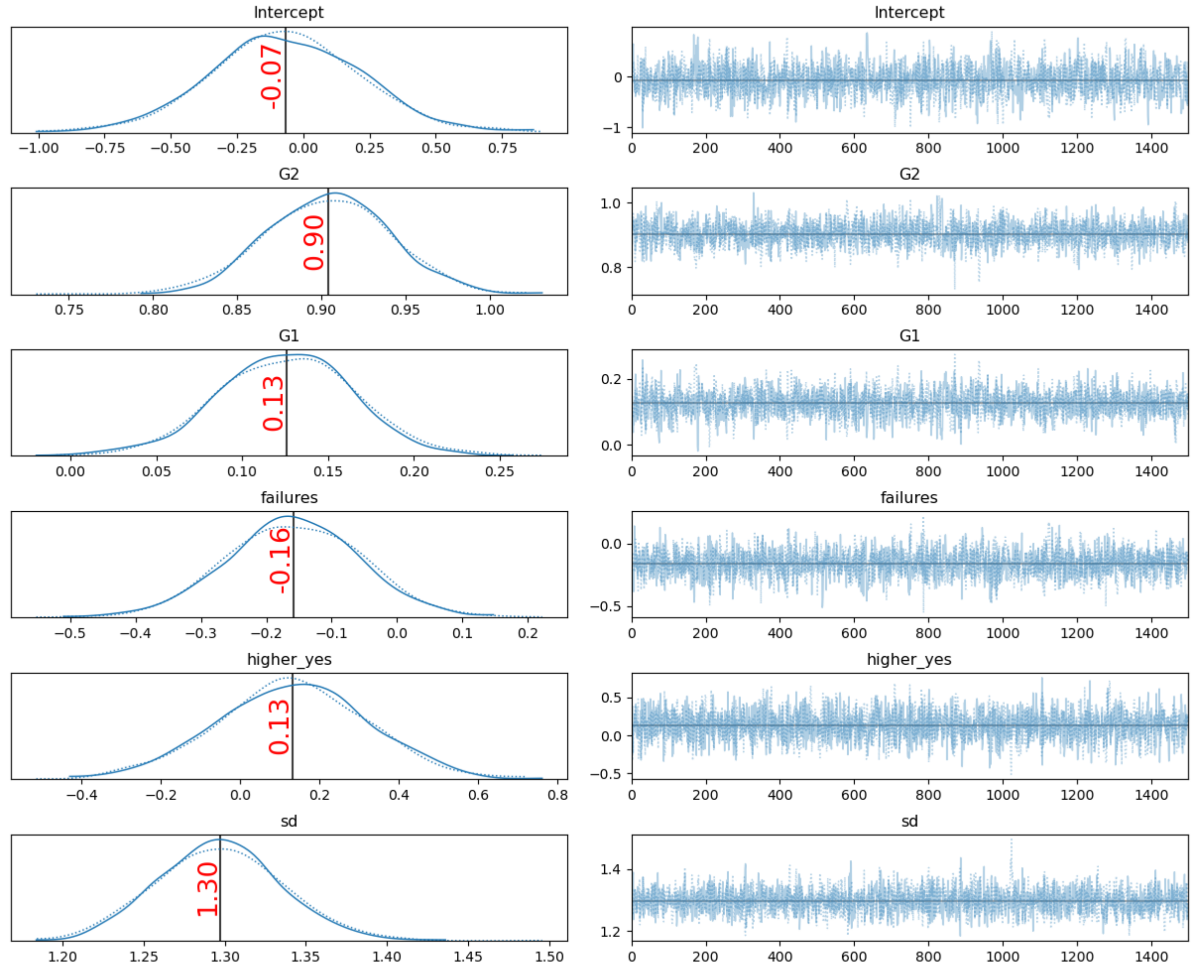
This method allows us to draw samples from the posterior in order to gauge the posterior.

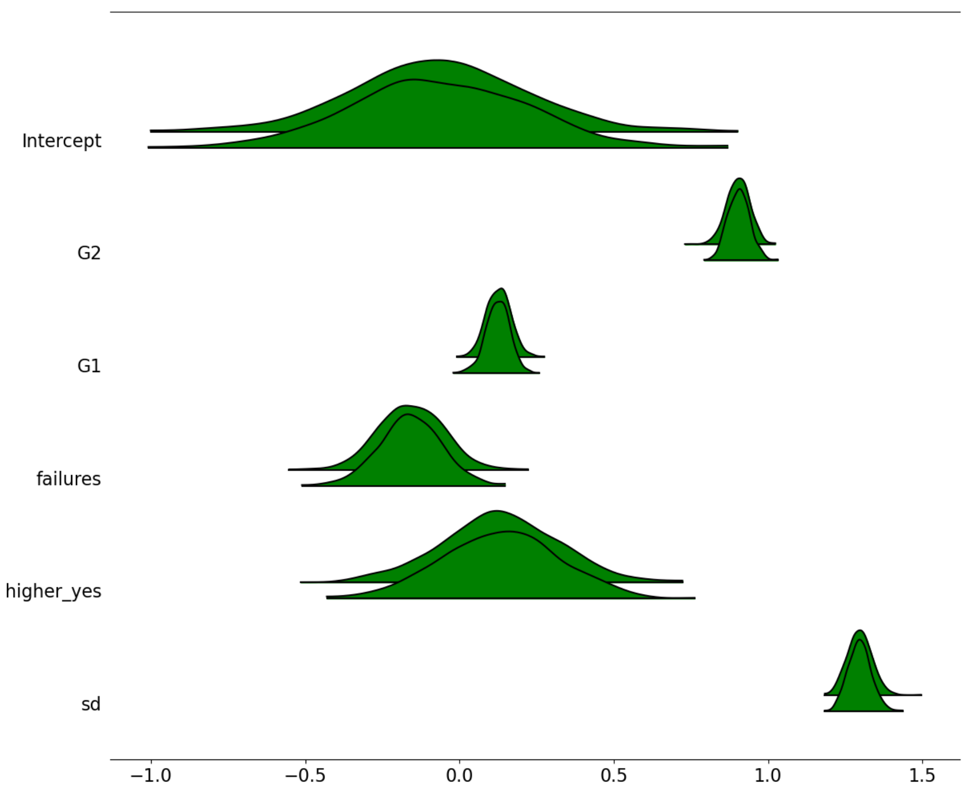
This method allows us to sample for a most likely distribution of the posterior, as we cannot directly calculate the distribution, we have use draws to see how close we are to the actual distribution. Usually the more draws the better accuracy which leads to a true distribution. The markov chain allows us to proceed with the next draw in accordance with the first. In the MCMC chain we used 1500 draws and 300 tunes with a chain length of 2. This was due to hardware restrictions.

Given at 1500 draws and 300 tures

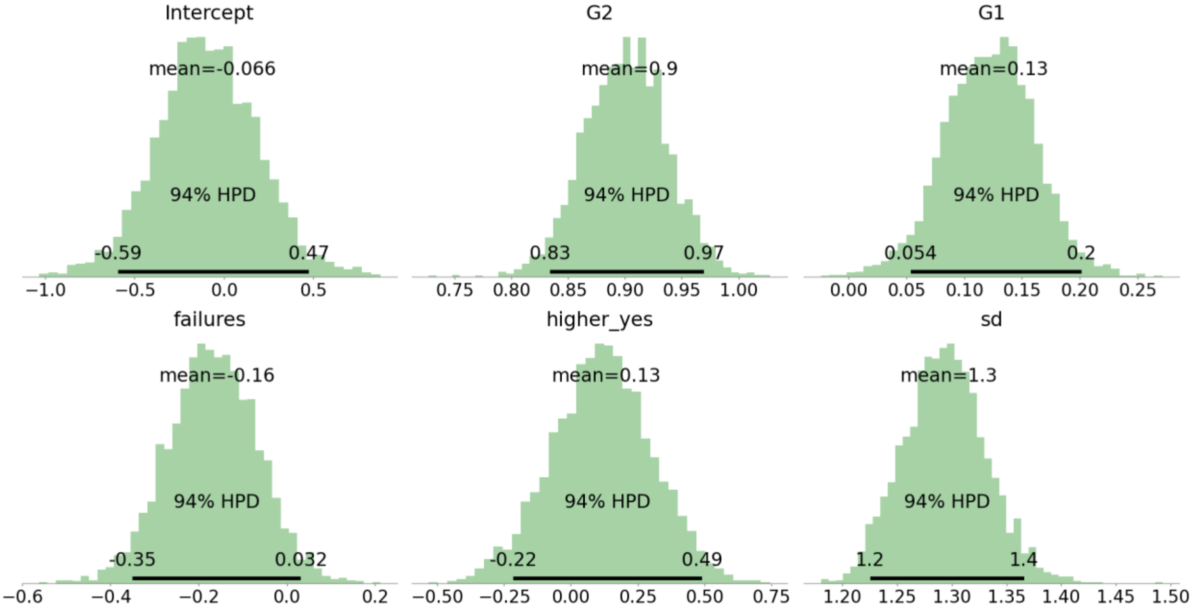
The given plots show the posterior distribution for the given model parameters and the samples drawn by both chains on the right. As we can see there in no single value but a distribution of the model parameters and the mean can be taken as the most likely.




Intercept
-0.07
G2
0.90
G1
0.13
failures
-0.16
higher_yes
0.13
sd
1.30


Intercept
G2
G1
failures
higher_yes
sd

The given histograms show what is the most likely estimate given for each parameter is the mean. The Highest posterior density is also given as well as a credible interval for the chosen parameters. This is a confidence level that the data point will be in this region.



Results for the histograms	
variable	Mean weight in the model
Intercept	-0.0659
G2	0.9041
G1	0.1257
failures	-0.1582
higher_yes	0.1325
sd_log__	0.2598
sd	1.2972

Here we can see the summary of the results.

G2 has a positive weight of 0.904 which means that this attribute highly affects the final grade. Its HPD is between 0.834 and 0.970 which means at the lowest HPD it is still positively weights and affects the final grade positively. next the standard deviation is low at 0.036 which shows us that a minute sample deviate from the given mean value of the group.

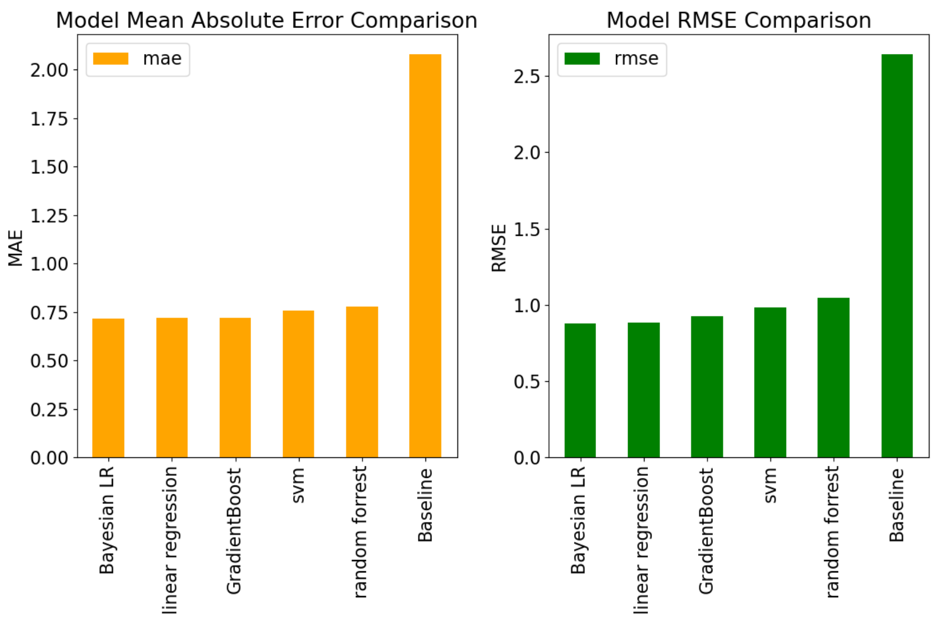
On the contrary the failures column has a low negative weight given by the mean of –0.158. Its HPD is between 0.032 and 0.032 which shows that effect on the model is unclear. In addition, the standard deviation is 0.102 is high, which shows that over 10% of the data sample deviate from the mean value of the group. This is also seen in the “higher_yes” column. However its HPD range shows that its more likely to have a passive effect on the final grade.

	mean	sd	hpd_3%	hpd_97%	...	ess_sd	ess_bulk	ess_tail	r_hat
Intercept	-0.066	0.285	-0.593	0.474	...	1515.0	1590.0	1746.0	1.0
G2	0.904	0.036	0.834	0.970	...	2026.0	2048.0	1700.0	1.0
G1	0.126	0.039	0.054	0.201	...	1833.0	1886.0	1597.0	1.0
failures	-0.158	0.102	-0.350	0.032	...	1863.0	2121.0	1994.0	1.0
higher_yes	0.132	0.187	-0.215	0.491	...	1630.0	2010.0	2004.0	1.0
sd	1.297	0.038	1.225	1.366	...	2380.0	2433.0	1824.0	1.0

Model MAE: 0.7174

Model RMSE: 0.8794

Here we can see how the Bayesian linear regression did in comparison to the other methods, including the ols linear regression



Results of the bar table with bayserian lr

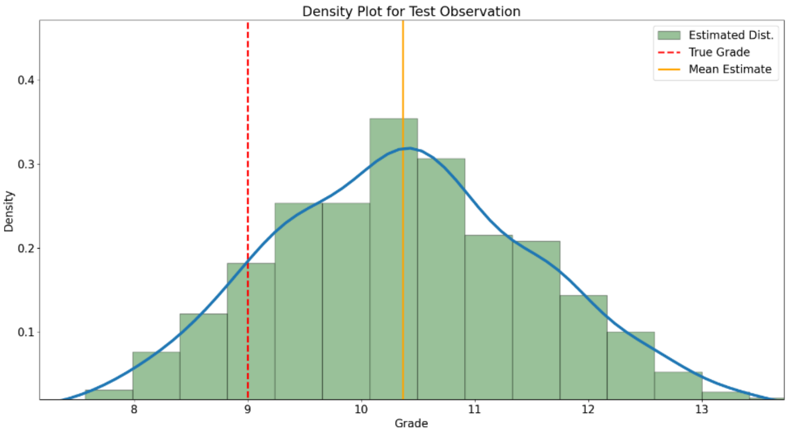
Model name	MAE	RMSE
linear regression	0.718334	0.880918
random forrest	0.777933	1.04493
svm	0.757032	0.983363
Gradient Boost	0.71915	0.927297
Baseline	2.07692	2.63993
Bayesian LR	0.71741	0.87937

Predictions from the model

In this part we will test the model on the existing data.

The first test was done on the 64th item in the test array, this was to see how far off the prediction was in terms of the actual result.

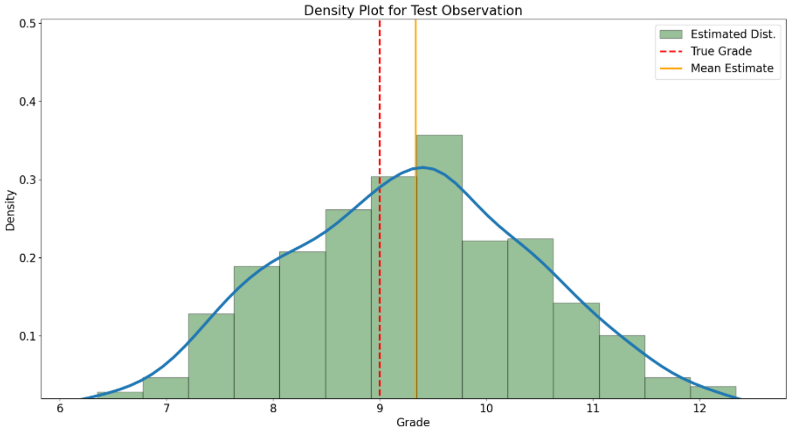
The graph bellow shows the likely distribution of the final grade with the true grade, as we can see the estimated grade shows the prediction for G3 to be 10.36, however the true results was 9. this in a linear perspective seems to be an outlier as the selected student has consistently gotten higher than 9 in the past two tests. As well as a solidified interest in higher education. In addition, the student has not had any failures. However, it's still in the range of 5% to 95%



G3		9	
G2		10	
G1		10	
failures		0	
Higher_yes		1	
Intercept		1	
True grade		9	
Average estimate		10.36	
5% estimate	8.28	95% estimate	12.58

In this test we chose the 48th record in the test array. To see how the prediction faired against the actual result.

The graph bellow shows the Distibution of the probability of the final grade. Its mean in the most likely prediction. In the test we can see that the prediction mean is equivalent to the actual results. The Distibution shows that the average grade is 9.33 with the true grade being 9. this is well in the 5% and 95% percent range.



G3		9	
G2		9	
G1		9	
failures		0	
Higher_yes		1	
Intercept		1	
True grade		9	
Average estimate		9.33	
5% estimate	7.26	95% estimate	11.38

C. Examine Bayesian Linear Regression Results

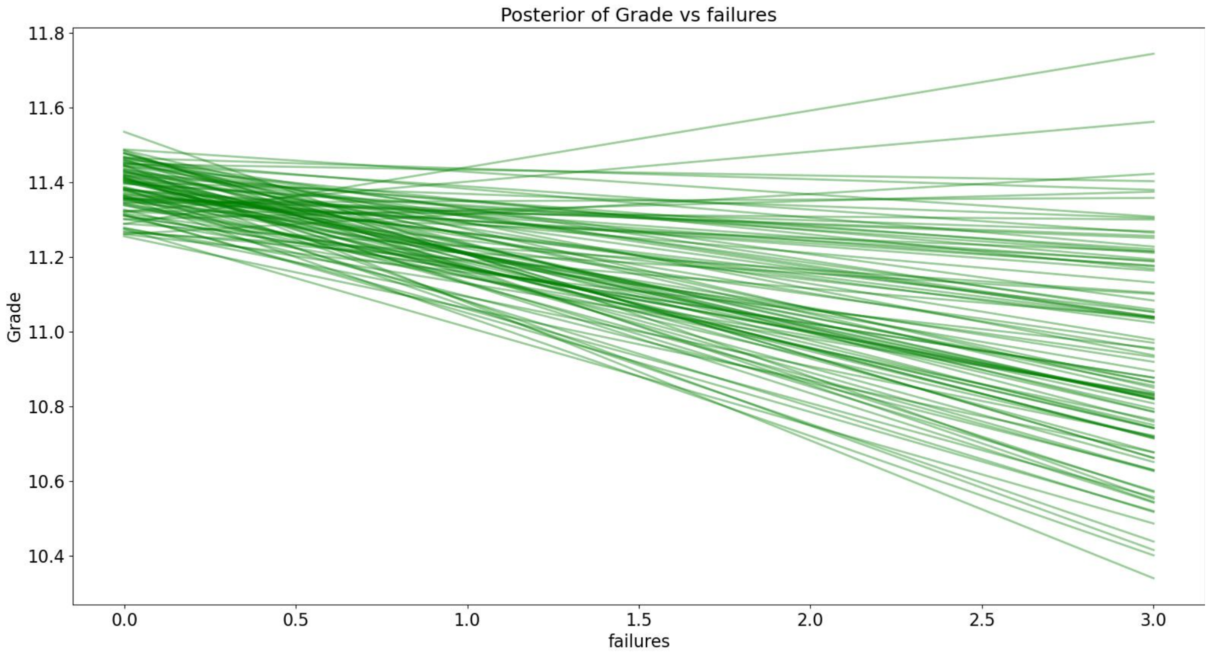
Post plot1

To see what effect a single variable has on the grade we can use a posterior predictive plot. This will show us the distribution of lines in the model

Each line is drawn by choosing a set of model parameters from the posterior and evaluating the predicted grade. The distribution of the lines shows us uncertainty in the model parameters, meaning the more spread out it is the less sure the model is about the effect of the variable.

Plot1 G3 against failures

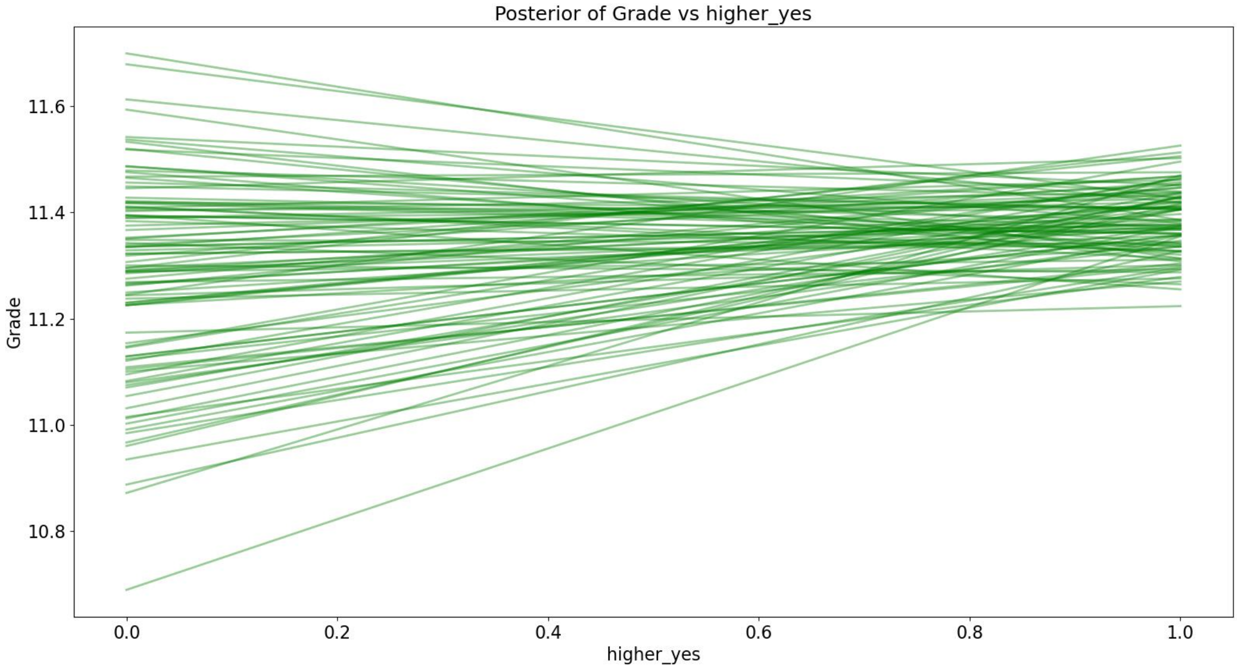
In this attribute we can see that the overall situation is that if the failures increase then the grades will decrease, only in a few cases the grades increase. However, as failures reach their peak the spread is not tight meaning that its not an accurate measure.



Plot 2 G3 against higher_yes.

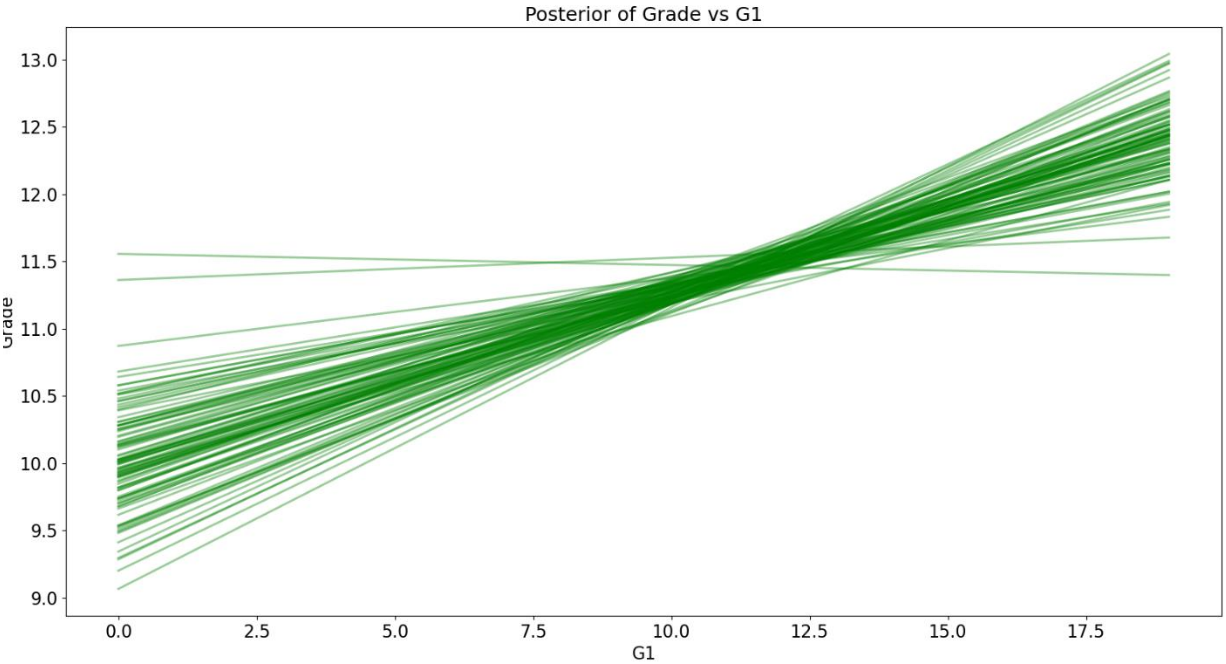
In this plot we can see that if the student does not want to go into higher education, they are not necessarily susceptible to a lower grade as the spread is more or less evenly distributed across the highest and lowest grade.

But on the other side we can clearly see that the spread grows tights and towards the higher grades as the students wants to go into higher education.



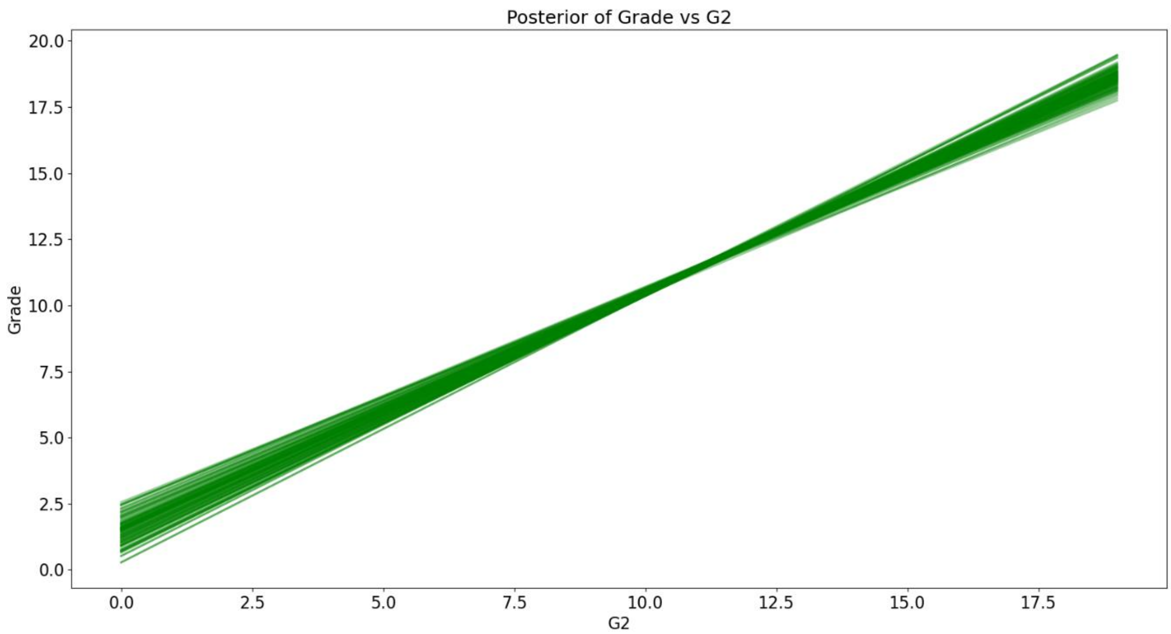
Plot 3 G3 against G1.

In this plot we can see that the Distibution is positive linear with the exception of a few cases. The students will generally have a higher outcome on grade 3 if their grade 1 results are high and vice versa if they are low.



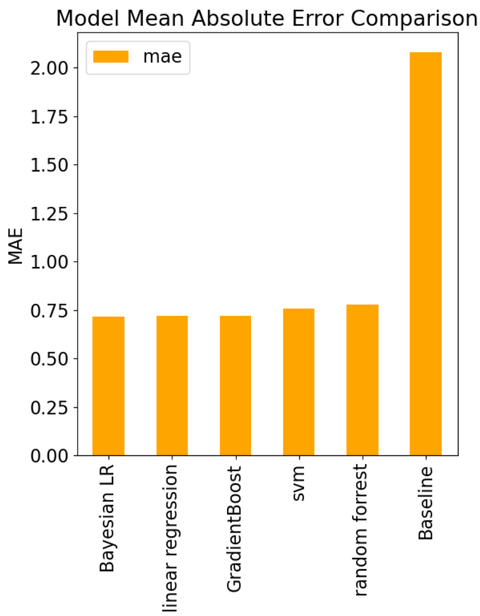
Plot 4 G3 against G2.

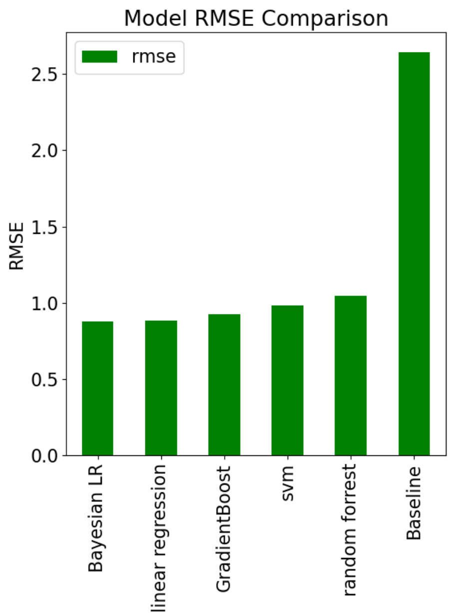
Lastly, we can see strong positive correlation between the results of the second grade and the result of the third. We can see a small spread at each end on the plot as this maybe to external factors such as the student managing to gain a few marks in the case of the bottom left.



D. Evaluate Bayesian Model Using Mean of Model Parameters

In this part we will compare the Bayesian approach to the others including the linear regression. As we can see the Bayesian linear regression is extreme similar to the linear regression. This may be due to a number of factors such as limited data set and dimensionality of the model.





Results of the bar graph with Bayesian linear regression.

Model name	MAE	RMSE
linear regression	0.718334	0.880918
random forest	0.777933	1.04493
SVM	0.757032	0.983363
Gradient Boost	0.71915	0.927297
Baseline	2.07692	2.63993
Bayesian LR	0.71741	0.87937

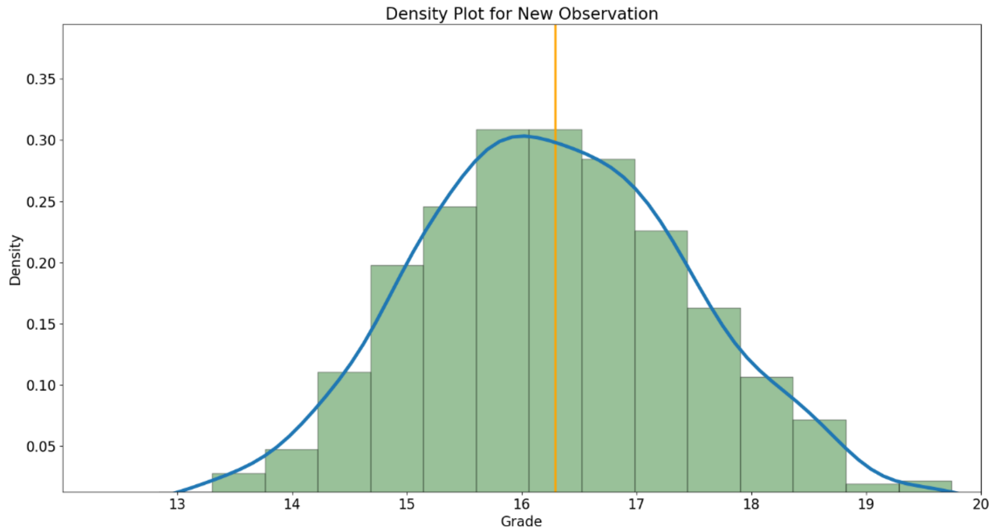
E. Make Predictions from Model on unseen (test) dataset.

New observation

Data 1

This plot shows the Prediction of a new point which was created. In this we have chosen values in both G1 & G2 to be greater than the last which would result in a positive linear prediction. Furthermore, we have caused to the student to peruse higher education and have no failures.

In this the average estimate for the new student is 16.29 but the range that the student can possibly get it is between 14.30 and 18.44.



Attribute	value
Intercept	1
failures	0
higher_yes	1
G2	16
G1	14
Average Estimate =	16.2922
5% Estimate =	14.2951
95% Estimate =	18.4436

Conclusion

To conclude we found out that the Bayesian linear regression did not make much difference to the prediction of grades compared to normal OLS linear regression. However, it gave a distribution of what the probability was and this gave us a better understanding of what is more likely of a grade than a concrete number.

problems to this project could have been the size of the data set at it was very small in comparison to others. Next a different type of leaning algorithms could be used such as a multi- layered perceptron to act as a direct competitor to the Bayesian theory with the introduction of forward and backward passes to ajust the weights for training and “learning”

Appendix 1

Imports and libraries

```

import pandas as pd
import numpy as np
import arviz as az
np.random.seed(42)
# Matplotlib and seaborn for plotting
import matplotlib.pyplot as plt
import matplotlib
matplotlib.rcParams['font.size'] = 16
matplotlib.rcParams['figure.figsize'] = (9, 9)
import seaborn as sns
from IPython.core.pylabtools import figsize
# Scipy helper functions
from scipy.stats import percentileofscore
from scipy import stats
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import ElasticNet
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import ExtraTreesRegressor
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.svm import SVR
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error, mean_absolute_error, median_absolute_error
import scipy
import pymc3 as pm
from sklearn.model_selection import train_test_split
```

Part1

```python
#distrbution of grades
#change G1 to appopriate column
ax = sns.countplot(data['G3'])
ax.axes.set_title('distrbution of G3', fontsize = 20 )
ax.set_xlabel('final grade 3 ', fontsize= 16)
ax.set_ylabel('number of students', fontsize = 16)
plt.show()

#bar graph of grades and count
b = sns.countplot('age',hue='sex', data= data)
b.axes.set_title('Number of students in different age groups',fontsize=15)
b.set_xlabel("Age of students",fontsize=15)
b.set_ylabel("grades of students",fontsize=7)
plt.show()

#this shows student age groups
sns.kdeplot(data.loc[data['address'] == 'U', 'G3'], label='Urban', shade = True)
sns.kdeplot(data.loc[data['address'] == 'R', 'G3'], label='Rural', shade = True)
plt.title('Do urban students score higher than rural students?', fontsize = 20)
plt.xlabel(' final grade ', fontsize = 20);
plt.ylabel('Density', fontsize = 20)
plt.show()

# this shows grades in retrospect of asparation of wanting to proceed into higer
sns.kdeplot(data.loc[data['higher'] == 'yes', 'G3'], label = 'yes', shade = True)
sns.kdeplot(data.loc[data['higher'] == 'no', 'G3'], label = 'no', shade = True)
plt.xlabel('Grade'); plt.ylabel('Density'); plt.title('Density Plot of Final Grades by higher education');

#this shows failures
sns.kdeplot(data.loc[data['failures'] == 0, 'G3'], label='0', shade = True)
sns.kdeplot(data.loc[data['failures'] == 1, 'G3'], label='1', shade = True)
sns.kdeplot(data.loc[data['failures'] == 2, 'G3'], label='2', shade = True)
sns.kdeplot(data.loc[data['failures'] == 3, 'G3'], label='3', shade = True)
plt.xlabel('Grade'); plt.ylabel('Density'); plt.title('Density Plot of Final Grades by number of failures ');
```

Part2

```python
# guardian of the student
sns.kdeplot(data.loc[data['guardian'] == 'father', 'G3'], label = 'Father', shade = True)
sns.kdeplot(data.loc[data['guardian'] == 'mother', 'G3'], label = 'Mother', shade = True)
sns.kdeplot(data.loc[data['guardian'] == 'other', 'G3'], label = 'Other', shade = True)
plt.xlabel('Grade'); plt.ylabel('Density'); plt.title('Density Plot of Final Grades by Guardian');

#escription of grades
print(data['G3'].describe())
print(data['G2'].describe())
print(data['G1'].describe())
#data['G3'].describe()

plt.subplots(figsize=(8,12))
grade_counts = data['G3'].value_counts().sort_values().plot.barh(width=.9)
grade_counts.axes.set_title('Number of students who scored a particular grade',fontsize=30)
grade_counts.set_xlabel('Number of students', fontsize=30)
grade_counts.set_ylabel('Final Grade', fontsize=30)
plt.show()

# corrolation coefficent
#works as a sheet
plt.figure(figsize=(12,10))
cor = data.corr()
sns.heatmap(cor, annot= True, cmap=plt.cm.Blues)
plt.show()
```

Part3

Split the data

```
df = pd.read_csv("stats.csv")
# creates the lable by using the df column 'G3'
labels = df['G3']
# One-Hot Encoding of Categorical Variables
df_dummy = pd.get_dummies(df)
# Find correlations with the G3 both types.
most_correlated = df_dummy.corr().abs()['G3'].sort_values(ascending=False)
# Maintain the top x most correlation features with Grade
## can be used to reduce dimentionality
most_correlated2 = most_correlated[:5]
df3 = df_dummy.loc[:, most_correlated2.index]

#Split into training and testing sets with with a desired split
X_train, X_test, y_train, y_test = train_test_split(df3, labels,test_size=0.10,random_state=31)
```

Part 4

Correlation coefficient and pairs plot on the data

```
def corrfunc(x, y):
  r, _ = stats.pearsonr(x, y)
  ax = plt.gca()
  ax.annotate("r = {:.2f}".format(r),xy=(.1, .6), xycoords=ax.transAxes,size=15)
  return x,y
cmap = sns.cubehelix_palette(light=1, dark=0.1,hue=0.5, as_cmap=True)



sns.set_context(font_scale=5)
#pairs plot
# Pair grid set up
g = sns.PairGrid(X_train)
# Scatter plot on the upper triangle
g.map_upper(plt.scatter, s=5, color='red')
# Distribution on the diagonal
g.map_diag(sns.distplot, kde=False, color='red')
# Density Plot and Correlation coefficients on the lower triangle
plt.tight_layout()
plt.show(g.map_lower(corrfunc))

#fuction and plot
```

Part 5

Mean compaarson

```python
#mean comarason between the atrributies
#Selected Variables Distribution by Relation to Median
#deoes that

X_plot = X_train.copy()
X_plot['relation_median'] = (X_plot['G3'] >= 12)
X_plot['relation_median'] = X_plot['relation_median'].replace({True: 'above', False: 'below'})
X_plot = X_plot.drop(columns='G3')

plt.figure(figsize=(12, 12))
# Plot the distribution of each variable colored
# by the relation to the median grade
for i, col in enumerate(X_plot.columns[:-1]):
    plt.subplot(4, 2, i + 1)

    subset_above = X_plot[X_plot['relation_median'] == 'above']
    subset_below = X_plot[X_plot['relation_median'] == 'below']
    sns.kdeplot(subset_above[col], label='Above Median', color='green')
    sns.kdeplot(subset_below[col], label='Below Median', color='red')
    plt.legend();
    plt.title('Distribution of %s' % col)

plt.tight_layout()
plt.show()
```

Part 6

metrics

```python
# #metric fuction

def eval (pred, true):
    mae = np.mean(abs(pred - true))
    rmse = np.sqrt(np.mean((pred-true) ** 2))
    return mae, rmse
# baseline is  the median
med_pred = X_train['G3'].median()
med_preds = [med_pred for _ in range(len(X_test))]
true = X_test['G3']
#display metrics
mb_mae, mb_rmse = eval(med_preds, true)
print('mae is : {:.3f}'.format(mb_mae))
print('rmse is : {:.3f}'.format(mb_rmse))
```

Part 7

model comparison

```
def eval2(X_train, X_test, y_train, y_test):
    model_name = ['linear regression', 'random forrest','svm','GradientBoost']
    X_train = X_train.drop(columns = 'G3')
    X_test = X_test.drop(columns='G3')

    model1 = LinearRegression()
    model2 = RandomForestRegressor(n_estimators=50)
    model3 = SVR(kernel='rbf', degree=3, C=1.0, gamma='auto')
    model4 = GradientBoostingRegressor(n_estimators=20)

    results = pd.DataFrame(columns=['mae','rmse'], index= model_name)

    for i,model in enumerate([model1,model2,model3, model4]):
        model.fit(X_train,y_train)
        predictions = model.predict(X_test)

        # Metrics
        mae = np.mean(abs(predictions - y_test))
        rmse = np.sqrt(np.mean((predictions - y_test) ** 2))

        modelname = model_name[i]
        results.loc[modelname, :]= [mae, rmse]

    baseline = np.median(y_train)
    baseline_mae = np.mean(abs(baseline - y_test))
    baseline_rmse = np.sqrt(np.mean((baseline - y_test) ** 2))

    results.loc['Baseline', :] = [baseline_mae, baseline_rmse]
    return results

results = eval2(X_train, X_test, y_train, y_test)
```

Plots

```
# works produces two distict bar graphs
figsize(12, 8)
matplotlib.rcParams['font.size'] = 16
# Root mean squared error
ax =  plt.subplot(1, 2, 1)
results.sort_values('mae', ascending = True).plot.bar(y = 'mae', color = 'orange', ax = ax)
plt.title('Model Mean Absolute Error'); plt.ylabel('MAE');

# Median absolute error
ax = plt.subplot(1, 2, 2)
results.sort_values('rmse', ascending = True).plot.bar(y = 'rmse', color = 'green', ax = ax)
plt.title('Model Root Mean Squared Error'); plt.ylabel('RMSE');
```

Ols linear regression

```python
ols_fomula = 'G3 = %0.2f +' % lr.intercept_
for i, col in enumerate(X_train.columns[1:]):
    ols_fomula += '%0.2f * %s +' % (lr.coef_[i], col)

' '.join(ols_fomula.split(' ')[:-1])
print(ols_fomula)
```

Part 8

Bayesian linear regression formula and MCMC

```python
# bayesian linar regression formula
blr = 'G3 ~ ' + ' + '.join(['%s' % var for var in X_train.columns[1:]])
print(blr)

# markov chain monte carlo
with pm.Model() as normal_model:
    family = pm.glm.families.Normal()
    #takes in blr formula
    pm.GLM.from_formula(blr, data = X_train,family = family)
    # how man draws
    normal_trace = pm.sample(draws=1500, chains= 2, tune= 300, cores=-1)
```

Part 9

Trace and plot of the posterior

```python
# Traceplot of All Samples
def plot_trace(trace):
    ax2 = pm.traceplot(trace, figsize(14, len(trace.varnames)*1.8),
        lines=[(k, {}, [v['mean']]) for k, v in pm.summary(trace).iterrows()])

    matplotlib.rcParams['font.size'] = 16
    for i, mn in enumerate (pm.summary(trace)['mean']):
        ax2[i, 0].annotate('{:0.2f}'.format(mn), xy=(mn, 0), xycoords='data', size=8,xytext=(-18, 18),
         textcoords='offset points', rotation=90,va='bottom', fontsize='large', color='red')
```

Plot outputs.

```python
plot_trace(normal_trace)
#pm.traceplot(normal_trace)
for variable in normal_trace.varnames:
    print('Variable: {:15} Mean weight in model: {:.4f}'.format(variable, np.mean(normal_trace[variable])))

print(pm.summary(normal_trace))

az.plot_trace(normal_trace)
az.plot_forest(normal_trace,  kind='ridgeplot',colors='green' )
pm.plot_posterior(normal_trace, figsize(14,14), textsize = 20, kind='hist', color = 'green')

plt.show()
```

Part 10

```python
model_formula = 'G3 ='
# Formula from Bayesian Inference
for variable in normal_trace.varnames:
    model_formula += ' %0.2f * %s +' % (np.mean(normal_trace[variable]), variable)
' '.join(model_formula.split(' ')[:-1])
```

```python
def eval_trace(trace, X_train, X_test, y_train, y_test, model_results):

    # dict for samples values
    var_dict = {}
    for variable in trace.varnames:
        var_dict[variable] = trace[variable]

    #results in data frame
    var_weights = pd.DataFrame(var_dict)
    var_means = var_weights.mean(axis = 0)
    # intercept coloumn
     # this gives error

    X_test.loc['intercept'] = 1

   # X_test.__setitem__('intercept'), 1
    names = X_test.columns[1:]
    X_test = X_test.loc[:, names]
    var_means = var_means[names]

    # estemate each test observation
    results = pd.DataFrame(index=X_test.index, columns=['est'])

    for row in X_test.iterrows():
        results.loc[row[0], 'est'] = np.dot(np.array(var_means), np.array(row[1]))
```

```python

    for row in X_test.iterrows():
        results.loc[row[0], 'est'] = np.dot(np.array(var_means), np.array(row[1]))

     # metrics
    actual = np.array(y_test)
    errors = results.loc['est'] - actual
    mae = np.mean(abs(errors))
    rmse = np.sqrt(np.mean(errors ** 2))

    print('Model  MAE: {:.4f}\nModel RMSE: {:.3f}'.format(mae, rmse))
    # Adds the results to the dataframe
    model_results.loc['Bayesian LR', :] = [mae, rmse]

    plt.figure(figsize=(12, 8))
    ax = plt.subplot(1, 2, 1)
    model_results.sort_values('mae', ascending=True).plot.bar(y='mae', color='r', ax=ax)
    plt.title('Model Mean Absolute Error Comparison');
    plt.ylabel('MAE');

    ax = plt.subplot(1, 2, 2)
    model_results.sort_values('rmse', ascending=True).plot.bar(y='rmse', color='b', ax=ax)
    plt.title('Model RMSE Comparison');
    plt.ylabel('RMSE')

    plt.tight_layout()
    plt.show()
    return model_results
```

Part 11

```python

def test(trace, test_observation):
    var_dict = {}
    for variable in trace.varnames:
        var_dict[variable] = trace[variable]

    weights = pd.DataFrame(var_dict)

    sd = weights['sd'].mean()

    actual = test_observation['G3']

    # intercept term
    test_observation['intercept'] = 1
    test_observation = test_observation.drop('G3')
    weights = weights[test_observation.index]
    # means fro the weights
    means = weights.mean(axis=0)
    mean_loc = np.dot(means, test_observation)
    estimate = np.random.normal(loc = mean_loc, scale=sd, size=1000)

    plt.figure(figsize(8, 8))
    sns.distplot(estimate, hist=True, kde=True, bins=19,hist_kws={'edgecolor': 'k', 'color': 'darkgreen'},
                 kde_kws={'linewidth': 4},
                 label='Estimated Dist.');
```

```python
    plt.vlines(x=actual, ymin=0, ymax=5,linestyles='--', colors='red',label='True Grade',linewidth=2.5);
     # mean esimate
    plt.vlines(x=mean_loc, ymin=0, ymax=5,linestyles='-', colors='orange',label='Mean Estimate',linewidth=2.5);

    plt.legend(loc=1)
    plt.title('Density Plot for Test Observation');
    plt.xlabel('G1');
    plt.ylabel('Density');


    print('True Grade = %d' % actual);
    print('Average Estimate = %0.4f' % mean_loc);
    print('5%% Estimate = %0.4f    95%% Estimate = %0.4f' % (np.percentile(estimate, 5),np.percentile(estimate, 95)));
    plt.show()


test(normal_trace, X_test.iloc[30])
```

Part 12

New observation

```python
# Make predictions for a new data point from the model trace
def query(trace, new_obs):
    print(new_obs)
    var_dict = {}
    for variable in trace.varnames:
        var_dict[variable]= trace[variable]

    sdval = var_dict['sd'].mean()
    varweights = pd.DataFrame(var_dict)
    varweights= varweights[new_obs.index]
    var_means = varweights.mean(axis =0)
    mean_loc = np.dot(var_means, new_obs)

    estim = np.random.normal(loc = mean_loc, scale= sdval, size= 1000)
    plt.figure(figsize(6,6))
    sns.distplot(estim,hist=True, kde=True, bins=19,hist_kws={'edge colour' : 'k', 'colour': 'darkblue'},
                 kde_kws={'linewidth' : 4}, label = 'estimate distance')

    plt.vlines(x=mean_loc, ymin= 0, ymax= 5, linestyles= '-', colors='red', linewidth = 3)
    plt.title('Density Plot for New Observation')
    plt.xlabel('G3')
    plt.ylabel('Density')
    print('Average Estimate = %0.4f' % mean_loc)
    print('5%% Estimate = %0.4f    95%% Estimate = %0.4f' % (np.percentile(estim, 5), np.percentile(estim, 95)))
# end of fuc
```

Input for the observation

```python
observation3 = pd.Series({'Intercept': 1,  'failures': 0,  'higher_yes': 1, 'G2': 16,'G1': 14})
```

Part13

```
def effect(query, trace, X):
    # Variables that do not change
    perm_vars = list(X.columns)
    perm_vars.remove(query)

    #Linear Model

    def lm(value, sample):

        prediction = sample['Intercept'] + sample[query] * value
        for var in perm_vars:

            prediction += sample[var] * X[var].median()
        return prediction

    figsize(8, 8)
    #Find the minimum and maximum values for the range of the query
    var_min = X[query].min()
    var_max = X[query].max()
   #plot
    pm.plot_posterior_predictive_glm(trace, eval=np.linspace(var_min, var_max, 100), lm=lm, samples=100, color='green',alpha=0.4, lw=2)
    plt.xlabel('%s' % query, size=16)
    plt.ylabel('Grade', size=16)
    plt.title("Posterior of Grade vs %s" % query, size=18)
    plt.show()


effect('higher_yes', normal_trace, X_train.drop(columns = 'G3'))
effect('failures', normal_trace, X_train.drop(columns = 'G3'))
effect('G1', normal_trace, X_train.drop(columns = 'G3'))
effect('G2', normal_trace, X_train.drop(columns = 'G3'))
```

Appendix 2

I think I've completed most if not all of the given tasks to the best of my ability with the help of resources given and would consider this coursework to be 70%. Naturally some understanding may be missing.