DTSC-670 Foundations of Machine Learning

Final Exam

Name: Calvin Gray

Table of Contents

Problem Framing & Big Picture

Opening overview

The initial thought is to create a ML model that would accurately depict a student's G3 grade with or without the use of the G1 and G2 grades.

In order to acheieve this goal I will choose a regression model, and have it to be supervised. The learning will be offline.

The Following various models will be accessed, Linear Regression, Logistic Regresson, Ridge Regression, and Lasso.

The reason why we are choosing a regression instead of a classification, is that the classification task will label the result esentially as pass/fail. I would like to know if there are borderline predictions that are made.

Get the Data

Import the data

In [1]:

```
# standard imports
import pandas as pd
import numpy as np

#more imports
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import make_pipeline
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import OneHotEncoder

#readcsv
student = pd.read_csv('student-mat.csv')
students = student

#data check
student.head()


```

Out[1]:

	school	sex	age	address	famsize	Pstatus	Medu	Fedu	Mjob	Fjob	...	goout	Dalc
0	GP	F	18.0	U	GT3	A	4	4	at_home	teacher	...	4	1
1	GP	F	17.0	U	GT3	T	1	1	at_home	other	...	3	1
2	GP	F	15.0	U	LE3	T	1	1	at_home	other	...	2	2
3	GP	F	15.0	U	GT3	T	4	2	health	services	...	2	1
4	GP	F	NaN	U	GT3	T	3	3	other	other	...	2	1

5 rows × 35 columns

Explore the data

Data List/Available Features

Study Attributes and Features

Attributes

1. school - student's school (binary: "GP" - Gabriel Pereira or "MS" - Mousinho da Silveira)
2. sex - student's sex (binary: "F" - female or "M" - male)
3. age - student's age (numeric: from 15 to 22)
4. address - student's home address type (binary: "U" - urban or "R" - rural)

5. famsize - family size (binary: "LE3" - less or equal to 3 or "GT3" - greater than 3)
6. Pstatus - parent's cohabitation status (binary: "T" - living together or "A" - apart)
7. Medu - mother's education (numeric: 0 - none, 1 - primary education (4th grade), 2 – 5th to 9th grade, 3 – secondary education or 4 – higher education)
8. Fedu - father's education (numeric: 0 - none, 1 - primary education (4th grade), 2 – 5th to 9th grade, 3 – secondary education or 4 – higher education)
9. Mjob - mother's job (nominal: "teacher", "health" care related, civil "services" (e.g. administrative or police), "at_home" or "other")
10. Fjob - father's job (nominal: "teacher", "health" care related, civil "services" (e.g. administrative or police), "at_home" or "other")
11. reason - reason to choose this school (nominal: close to "home", school "reputation", "course" preference or "other")
12. guardian - student's guardian (nominal: "mother", "father" or "other")
13. traveltime - home to school travel time (numeric: 1 - <15 min., 2 - 15 to 30 min., 3 - 30 min. to 1 hour, or 4 - >1 hour)
14. studytime - weekly study time (numeric: 1 - <2 hours, 2 - 2 to 5 hours, 3 - 5 to 10 hours, or 4 - >10 hours)
15. failures - number of past class failures (numeric: n if 1<=n<3, else 4)
16. schoolsup - extra educational support (binary: yes or no)
17. famsup - family educational support (binary: yes or no)
18. paid - extra paid classes within the course subject (Math or Portuguese) (binary: yes or no)
19. activities - extra-curricular activities (binary: yes or no)
20. nursery - attended nursery school (binary: yes or no)
21. higher - wants to take higher education (binary: yes or no)
22. internet - Internet access at home (binary: yes or no)
23. romantic - with a romantic relationship (binary: yes or no)
24. famrel - quality of family relationships (numeric: from 1 - very bad to 5 - excellent)
25. freetime - free time after school (numeric: from 1 - very low to 5 - very high)
26. goout - going out with friends (numeric: from 1 - very low to 5 - very high)
27. Dalc - workday alcohol consumption (numeric: from 1 - very low to 5 - very high)
28. Walc - weekend alcohol consumption (numeric: from 1 - very low to 5 - very high)
29. health - current health status (numeric: from 1 - very bad to 5 - very good)
30. absences_G1 - number of school absences for G1 term (numeric)
31. absences_G2 - number of school absences for G2 term (numeric)
32. absences_G3 - number of school absences for G3 term (numeric)

these grades are related with the course math subject

33. G1 - first term grade (numeric: from 0 to 20)
34. G2 - second term grade (numeric: from 0 to 20)
35. G3 - final grade (numeric: from 0 to 20)

Prepare the data

drop NA rows

In [2]:
```
#drop any rows with NA values
student.dropna(axis=0,inplace=True)
```

Get the data

Check the size and data types

In [3]:
```
#DF check
student.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 369 entries, 0 to 394
Data columns (total 35 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   school       369 non-null    object 
 1   sex          369 non-null    object 
 2   age          369 non-null    float64
 3   address      369 non-null    object 
 4   famsize      369 non-null    object 
 5   Pstatus      369 non-null    object 
 6   Medu         369 non-null    int64  
 7   Fedu         369 non-null    int64  
 8   Mjob         369 non-null    object 
 9   Fjob         369 non-null    object 
 10  reason       369 non-null    object 
 11  guardian     369 non-null    object 
 12  traveltime   369 non-null    int64  
 13  studytime    369 non-null    int64  
 14  failures     369 non-null    int64  
 15  schoolsup    369 non-null    object 
 16  famsup       369 non-null    object 
 17  paid         369 non-null    object 
 18  activities   369 non-null    object 
 19  nursery      369 non-null    object 
 20  higher       369 non-null    object 
 21  internet     369 non-null    object 
 22  romantic     369 non-null    object 
 23  famrel       369 non-null    int64  
 24  freetime     369 non-null    int64  
 25  goout        369 non-null    int64  
 26  Dalc         369 non-null    int64  
 27  Walc         369 non-null    int64  
 28  health       369 non-null    int64  
 29  absences_G1  369 non-null    float64
 30  absences_G2  369 non-null    float64
 31  absences_G3  369 non-null    float64
 32  G1           369 non-null    int64  
 33  G2           369 non-null    int64  
 34  G3           369 non-null    int64  
dtypes: float64(4), int64(14), object(17)
memory usage: 103.8+ KB
```

Get The data

Target Attribute

Target or predicted Attribute is G3 Since this is the target we will remove it from the dataset and call it G3_label

In [4]:
```
#Cutting the Target and removing from dataset
G3_label = student['G3']

students = student


#G3_label
student = student.drop('G3',axis=1)
```

Get the data

Split data training and testing

In [5]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(
    student, G3_label, test_size=0.2, random_state=42)
```

In [6]:
```
X_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 295 entries, 367 to 113
Data columns (total 34 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   school       295 non-null    object 
 1   sex          295 non-null    object 
 2   age          295 non-null    float64
 3   address      295 non-null    object 
 4   famsize      295 non-null    object 
 5   Pstatus      295 non-null    object 
 6   Medu         295 non-null    int64  
 7   Fedu         295 non-null    int64  
 8   Mjob         295 non-null    object 
 9   Fjob         295 non-null    object 
 10  reason       295 non-null    object 
 11  guardian     295 non-null    object 
 12  traveltime   295 non-null    int64  
 13  studytime    295 non-null    int64  
 14  failures     295 non-null    int64  
 15  schoolsup    295 non-null    object 
 16  famsup       295 non-null    object 
 17  paid         295 non-null    object 
 18  activities   295 non-null    object 
 19  nursery      295 non-null    object 
 20  higher       295 non-null    object 
 21  internet     295 non-null    object 
 22  romantic     295 non-null    object 
 23  famrel       295 non-null    int64  
 24  freetime     295 non-null    int64  
 25  goout        295 non-null    int64  
 26  Dalc         295 non-null    int64  
 27  Walc         295 non-null    int64  
 28  health       295 non-null    int64  
 29  absences_G1  295 non-null    float64
 30  absences_G2  295 non-null    float64
 31  absences_G3  295 non-null    float64
 32  G1           295 non-null    int64  
 33  G2           295 non-null    int64  
dtypes: float64(4), int64(13), object(17)
memory usage: 80.7+ KB
```

In [7]:
```
X_test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 74 entries, 349 to 313
Data columns (total 34 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   school       74 non-null     object 
 1   sex          74 non-null     object 
 2   age          74 non-null     float64
 3   address      74 non-null     object 
 4   famsize      74 non-null     object 
 5   Pstatus      74 non-null     object 
 6   Medu         74 non-null     int64  
 7   Fedu         74 non-null     int64  
 8   Mjob         74 non-null     object 
 9   Fjob         74 non-null     object 
 10  reason       74 non-null     object 
 11  guardian     74 non-null     object 
 12  traveltime   74 non-null     int64  
 13  studytime    74 non-null     int64  
 14  failures     74 non-null     int64  
 15  schoolsup    74 non-null     object 
 16  famsup       74 non-null     object 
 17  paid         74 non-null     object 
 18  activities   74 non-null     object 
 19  nursery      74 non-null     object 
 20  higher       74 non-null     object 
 21  internet     74 non-null     object 
 22  romantic     74 non-null     object 
 23  famrel       74 non-null     int64  
 24  freetime     74 non-null     int64  
 25  goout        74 non-null     int64  
 26  Dalc         74 non-null     int64  
 27  Walc         74 non-null     int64  
 28  health       74 non-null     int64  
 29  absences_G1  74 non-null     float64
 30  absences_G2  74 non-null     float64
 31  absences_G3  74 non-null     float64
 32  G1           74 non-null     int64  
 33  G2           74 non-null     int64  
dtypes: float64(4), int64(13), object(17)
memory usage: 20.2+ KB
```

In [8]:
```
y_train.info()
```

```
<class 'pandas.core.series.Series'>
Index: 295 entries, 367 to 113
Series name: G3
Non-Null Count  Dtype
--------------  -----
295 non-null    int64
dtypes: int64(1)
memory usage: 4.6 KB
```

In [9]:
```
y_test.info()
```

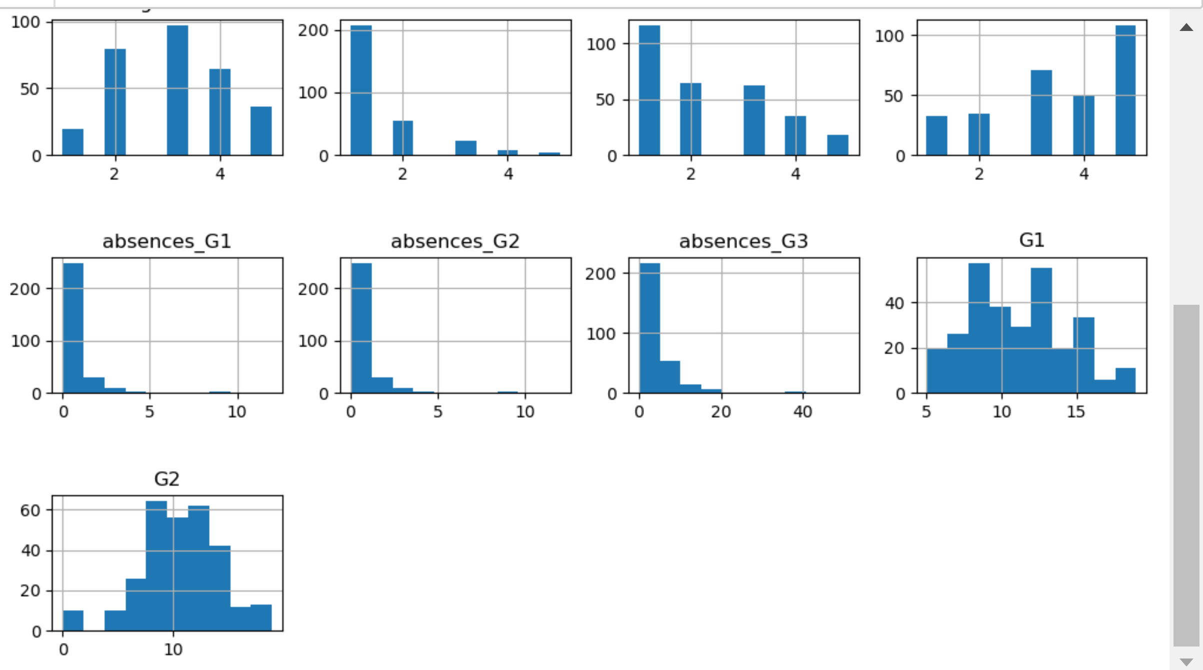
```
<class 'pandas.core.series.Series'>
Index: 74 entries, 349 to 313
Series name: G3
Non-Null Count  Dtype
--------------  -----
74 non-null     int64
dtypes: int64(1)
memory usage: 1.2 KB
```

Explore the Data

Attibutes and characteristics

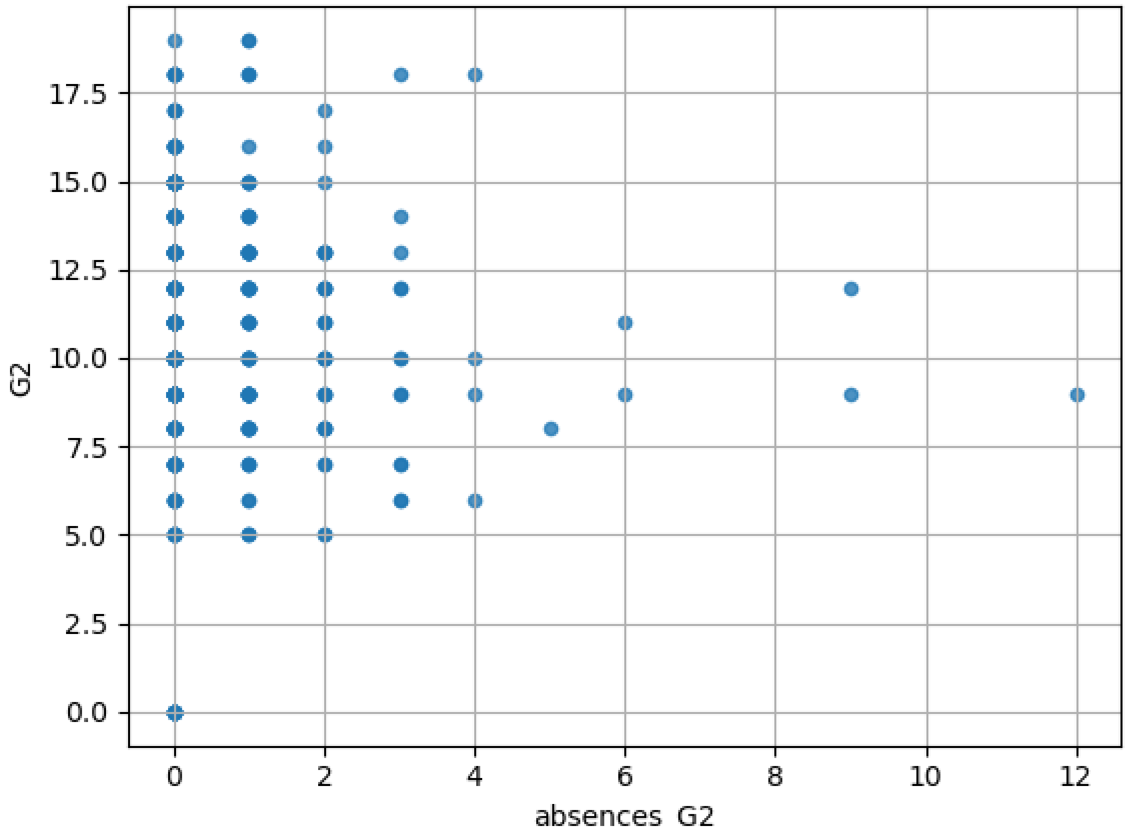
Data Visualizations

In [10]:
```
import matplotlib.pyplot as plt



# plots histograms of numerical data
X_train.hist(figsize=(12, 12))
plt.subplots_adjust(hspace=.75, wspace=.25)

plt.show()
```



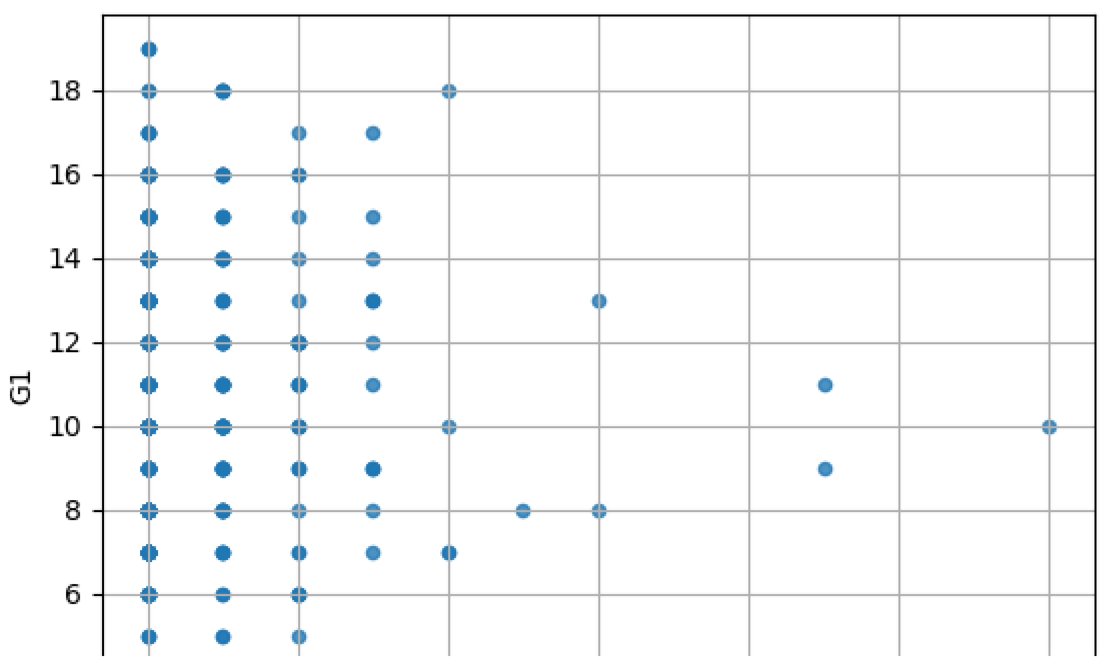
In [11]:

```
students.plot(kind="scatter", x="absences_G2", y="G2",
              alpha=.8, grid=True)
plt.show()
```

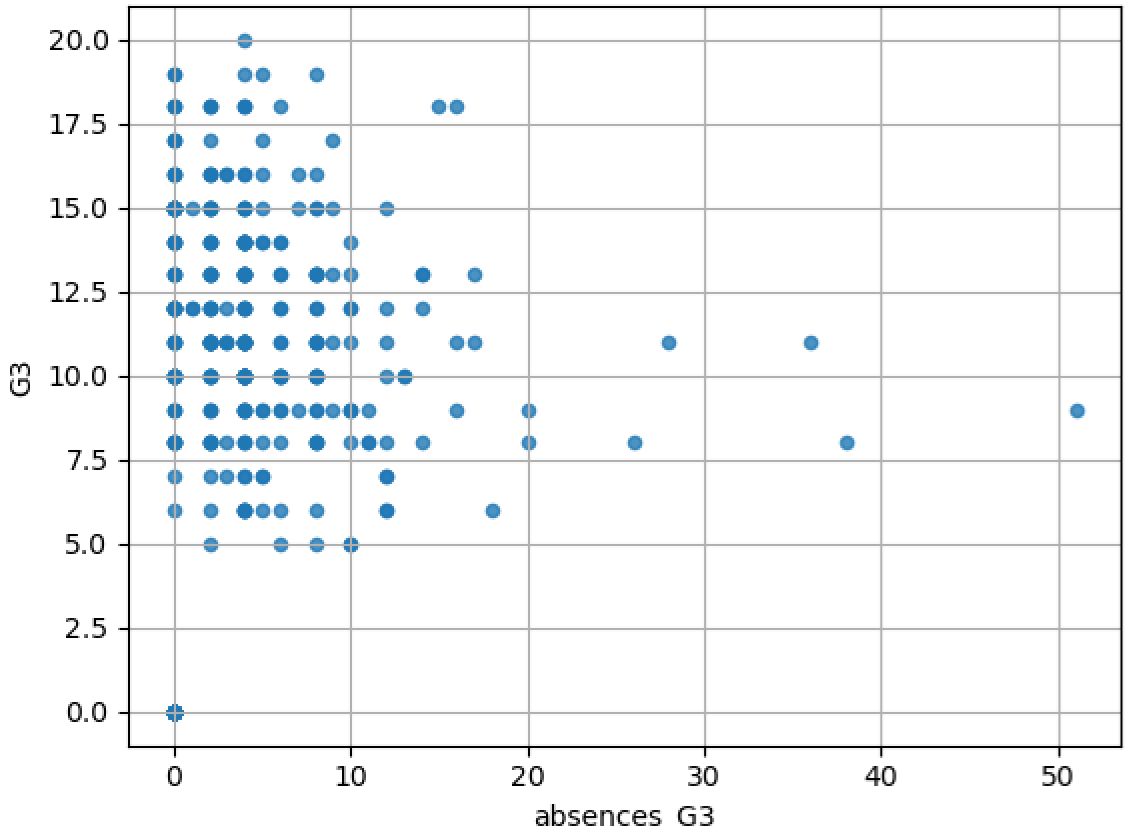


In [12]:

```
students.plot(kind="scatter", x="absences_G1", y="G1",
              alpha=.8, grid=True)
plt.show()
```



In [13]:
```
students.plot(kind="scatter", x="absences_G3", y="G3",
              alpha=.8, grid=True)
plt.show()
```



Explore the Data

Study the Correlations

In [14]:
```
#correlation Matrix to determine the correlations on the numeric columnns

nums_cols =  students._get_numeric_data()

#this can be done simply by calling the 'student' df with .corr()

correlation_matrix = nums_cols.corr()
correlation_matrix

correlation_matrix['G3'].sort_values(ascending=False)
```

Out[14]:
```
G3              1.000000
G2              0.902604
G1              0.802530
Medu            0.219222
Fedu            0.144649
studytime       0.087619
absences_G3     0.059360
famrel          0.057301
freetime        0.020067
absences_G1     0.005600
absences_G2     0.005600
Walc           -0.045631
Dalc           -0.049142
health         -0.074775
traveltime     -0.121912
goout          -0.122035
age            -0.152843
failures       -0.347706
Name: G3, dtype: float64
```

In [15]:

```
# Reindexing the columns so that the column transformer will drop the corr
X_Train = X_train.iloc[:,[29,30,31,32,33,0,1,2,3,4,5,6,7,8,9,10,11,14,15,1
X_Test = X_test.iloc[:,[29,30,31,32,33,0,1,2,3,4,5,6,7,8,9,10,11,14,15,16,


#column position check
X_Train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 295 entries, 367 to 113
Data columns (total 32 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   absences_G1  295 non-null    float64
 1   absences_G2  295 non-null    float64
 2   absences_G3  295 non-null    float64
 3   G1           295 non-null    int64
 4   G2           295 non-null    int64
 5   school       295 non-null    object
 6   sex          295 non-null    object
 7   age          295 non-null    float64
 8   address      295 non-null    object
 9   famsize      295 non-null    object
 10  Pstatus      295 non-null    object
 11  Medu         295 non-null    int64
 12  Fedu         295 non-null    int64
 13  Mjob         295 non-null    object
```

Prepare the Data

Custom Transfomer

In [16]:

```
from sklearn.base import BaseEstimator, TransformerMixin
 #['age', 'failures', 'absences_G1', 'absences_G2', 'absences_G3', 'G1', '
# column index
absences_G1_x, absences_G2_x, absences_G3_x = 0, 1, 2
G1_x, G2_x = 3,4

class final_project_transformer(BaseEstimator, TransformerMixin):
    def __init__(self, drop_columns = True): # no *args or **kargs
        self.drop_columns = drop_columns

    def fit(self, X, y=None): # pipelines require the fit() method to have
        return self  # nothing else to do

    def transform(self, X):
        total_absences = X[:, absences_G1_x] + X[:, absences_G2_x] + X[:,
        X = np.delete(X, 2, axis=1)
        X = np.delete(X, 1, axis=1)
        X = np.delete(X, 0, axis=1)

        if self.drop_columns:
            #X = np.delete(X, 33, axis=1)
            #X = np.delete(X, 32, axis=1)
            X = np.delete(X, 0, axis=1)
            X = np.delete(X, 0, axis=1)



            return np.c_[X, total_absences]
        else:
            return np.c_[X, total_absences]

```

In [17]:
```
#code check - must check to see if column transformer is working as intend
X_Trains = X_Train._get_numeric_data()
X_Trains.head()
print(X_Trains.head(2))
addme = final_project_transformer(drop_columns=True)
transform = addme.transform(X_Trains.values)
print(transform[:2])

#X_Trained = X_Trains.drop(columns = ['absences_G1','absences_G2','absence
X_Trained = X_Trains.drop(columns = ['absences_G1','absences_G2','absences

custom_df = pd.DataFrame(
    transform,
    columns=list(X_Trained.columns)+ ['total_absences'],
    index=X_Trained.index)

custom_df.head()

```

```
     absences_G1  absences_G2  absences_G3  G1  G2   age  Medu  Fedu  \
367          0.0          0.0          0.0   7   6  17.0     1     1
207          1.0          1.0          8.0  11  12  16.0     4     3

     failures  famrel  freetime  goout  Dalc  Walc  health
367         1       5         2      1     1     2       1
207         0       1         3      2     1     1       1
[[17.  1.  1.  1.  5.  2.  1.  1.  2.  1.  0.]
 [16.  4.  3.  0.  1.  3.  2.  1.  1.  1. 10.]]
```

Out[17]:

	age	Medu	Fedu	failures	famrel	freetime	goout	Dalc	Walc	health	total_absences
367	17.0	1.0	1.0	1.0	5.0	2.0	1.0	1.0	2.0	1.0	0.0
207	16.0	4.0	3.0	0.0	1.0	3.0	2.0	1.0	1.0	1.0	10.0
84	15.0	1.0	1.0	0.0	4.0	3.0	2.0	2.0	3.0	4.0	2.0
93	16.0	4.0	2.0	0.0	5.0	3.0	3.0	1.0	1.0	1.0	0.0
380	18.0	4.0	4.0	0.0	3.0	2.0	4.0	1.0	4.0	2.0	4.0

Prepare the data

perform feature selection to get the most applicable (numeric) attributes

feature selection is inside the numeric portion of the column transformer

Prepare the data

Pipeline for numeric data

In [18]:

```
#Start the numerical pipeline with the necessary transformations

from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.feature_selection import VarianceThreshold

numeric_pipeline = make_pipeline(
    SimpleImputer(missing_values=np.nan,strategy='most_frequent'),
    final_project_transformer(),
    StandardScaler(),
    VarianceThreshold(threshold=(.8 * (1 - .8))))


```

Prepare the data

Pipeline for categorical data

In [19]:

```
from sklearn.preprocessing import OneHotEncoder

categorical_pipeline = make_pipeline(
    SimpleImputer(missing_values=np.nan,strategy="most_frequent"),
    OneHotEncoder(drop=None, sparse_output=False))


```

Prepare the data

Pipeline for the ordinal data

In [20]:

```
from sklearn.preprocessing import OrdinalEncoder

school=['GP', 'MS']
sex=['F', 'M']
famsize=['GT3', 'LE3']
address=['R', 'U']
Pstatus=['A', 'T']
schoolsup=['no', 'yes']
famsup=['no', 'yes']
paid=['no', 'yes']
activities=['no', 'yes']
nursery=['no', 'yes']
higher=['no', 'yes']
internet=['no', 'yes']
romantic=['no', 'yes']

tryme = X_train[['school','sex','famsize','address','Pstatus','schoolsup',
                 'famsup','paid','activities','nursery','higher','internet

o_encode = OrdinalEncoder(categories=[school,sex,famsize,address,Pstatus,s
                                      famsup,paid,activities,nursery,highe

X_TRAINED = o_encode.fit_transform(X_train[['school','sex','famsize','addr
                 'famsup','paid','activities','nursery','higher','internet
#print(X_TRAINED)

#check the Data Frame
#this is ensures that the Ordinal Encoder is properly working
#before it is used in the pipeline

to_dfts = pd.DataFrame(X_TRAINED,
                       columns=o_encode.get_feature_names_out(),
                       index=X_train.index)
to_dfts

```

Out[20]:

	school	sex	famsize	address	Pstatus	schoolsup	famsup	paid	activities	nursery	hi
367	1.0	0.0	0.0	0.0	1.0	0.0	1.0	1.0	0.0	1.0	
207	0.0	0.0	0.0	1.0	1.0	0.0	0.0	1.0	1.0	1.0	
84	0.0	0.0	0.0	1.0	1.0	0.0	1.0	0.0	1.0	0.0	
93	0.0	0.0	0.0	1.0	1.0	0.0	1.0	1.0	1.0	1.0	
380	1.0	1.0	0.0	1.0	1.0	0.0	0.0	1.0	1.0	0.0	
...	...	...	...	...	...	...	...	...	...	...	
80	0.0	1.0	0.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	
117	0.0	1.0	0.0	1.0	1.0	0.0	0.0	0.0	1.0	1.0	
290	0.0	1.0	0.0	1.0	1.0	0.0	1.0	1.0	1.0	1.0	
371	1.0	1.0	1.0	0.0	1.0	0.0	1.0	1.0	1.0	1.0	
113	0.0	1.0	1.0	1.0	1.0	0.0	0.0	0.0	0.0	1.0	

In [21]:

```
from sklearn.preprocessing import OrdinalEncoder

ordinal_pipeline = make_pipeline(
    SimpleImputer(missing_values=np.nan,strategy="most_frequent"),
    OrdinalEncoder(categories=[school,sex,famsize,address,Pstatus,schoolsu
                                          famsup,paid,activities,nursery,highe

```

Column Transformer

Transform the data

Normal state is WITHOUT the G1/G2 Columns

In [22]:
```
#Passing the data through all three pipelines.
#Default state for final_project_transformer is "drop_columns = True"

from sklearn.compose import ColumnTransformer
from sklearn import set_config


nums_attribs = ['absences_G1',  'absences_G2',  'absences_G3',  'G1',  'G2
                'failures',  'famrel' , 'freetime',  'goout' , 'Dalc' , 'W

cats_attribs = ['Mjob','Fjob','reason','reason']

ords_attribs = ['school','sex','famsize','address','Pstatus','schoolsup',
                 'famsup','paid','activities','nursery','higher','internet



preprocessing = ColumnTransformer([

        ("cat", categorical_pipeline, cats_attribs),
        ("ord", ordinal_pipeline, ords_attribs),
        ("num", numeric_pipeline, nums_attribs)])

X_Train_prepared = preprocessing.fit_transform(X_Train)

```

In [23]:
```
#Output Check
X_Train_prepared.shape

```

Out[23]: (295, 42)

Column Transformer

Transform the data

With the G1/G2 Columns

In [24]:
```
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.feature_selection import VarianceThreshold

numeric_pipeline_drops = make_pipeline(
    SimpleImputer(missing_values=np.nan,strategy='most_frequent'),
    final_project_transformer(drop_columns = False),
    StandardScaler(),
    VarianceThreshold(threshold=(.8 * (1 - .8))))
```

In [25]:
```
#Rerunning final_project_transformer with drop_columns=False

nums_attribs = ['absences_G1',  'absences_G2',  'absences_G3',  'G1',  'G2
                'failures',  'famrel' , 'freetime',  'goout' , 'Dalc' , 'W

cats_attribs = ['Mjob','Fjob','reason','reason']

ords_attribs = ['school','sex','famsize','address','Pstatus','schoolsup',
                 'famsup','paid','activities','nursery','higher','internet



preprocessings = ColumnTransformer([

        ("cat", categorical_pipeline, cats_attribs),
        ("ord", ordinal_pipeline, ords_attribs),
        ("num", numeric_pipeline_drops, nums_attribs)])

X_Train_prepared_no_drops = preprocessings.fit_transform(X_Train)
```

In [26]:
```
#Output Check
X_Train_prepared_no_drops.shape
```

Out[26]: (295, 44)

Shortlist Promising Models

Fit 3 or more Promising models to the data

Compare Models with/without G1&G2

In [27]:
```
from sklearn.linear_model import LogisticRegression

# instantiate a Logistic Regression Class
# increasing the maximum number of iterations taken for the solvers to con
log_clf = LogisticRegression(random_state=42, max_iter=1500)


```

In [28]:

```
from sklearn.model_selection import cross_val_predict

# Log Reg Preds without G1/G2
log_reg_y_train_preds_a  = cross_val_predict(log_clf, X_Train_prepared, y_
# Log Reg Preds with G1/G2
log_reg_y_train_preds_b  = cross_val_predict(log_clf, X_Train_prepared_no_

print(log_reg_y_train_preds_a[:5])
print(y_test.iloc[:5].values)
print('\n')
print(log_reg_y_train_preds_b[:5])
print(y_test.iloc[:5].values)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\model_selection\_split.py:
725: UserWarning: The least populated class in y has only 1 members, which is
less than n_splits=5.
  warnings.warn(
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\model_selection\_split.py:
725: UserWarning: The least populated class in y has only 1 members, which is
less than n_splits=5.
  warnings.warn(

[10 11 11 10 15]
[13 13 10 10 12]


[10 11 11 10 15]
[13 13 10 10 12]
```

In [29]:

```
from sklearn.linear_model import LinearRegression

lin_reg = LinearRegression()

```

In [30]:

```
# Lin Reg Preds without G1/G2
ols_y_train_preds_a  = cross_val_predict(lin_reg, X_Train_prepared, y_trai
# Lin Reg Preds with G1/G2
ols_y_train_preds_b  = cross_val_predict(lin_reg, X_Train_prepared_no_drop

print(ols_y_train_preds_a[:5])
print(y_test.iloc[:5].values)
print('\n')
print(ols_y_train_preds_b[:5])
print(y_test.iloc[:5].values)
```

```
[ 9.50390625  9.578125    9.03320312  9.83203125 11.62890625]
[13 13 10 10 12]


[ 5.55273438  8.89111328  9.58398438  8.72949219 13.9375    ]
[13 13 10 10 12]
```

In [31]:
```
from sklearn.linear_model import Ridge

ridge_reg = Ridge(alpha=0.1, solver="cholesky")

```

In [32]:
```
#Ridge Reg Preds without G1/G2
ridge_y_train_preds_a  = cross_val_predict(ridge_reg, X_Train_prepared, y_
# Ridge Reg Preds with G1/G2
ridge_y_train_preds_b  = cross_val_predict(ridge_reg, X_Train_prepared_no_

print(ridge_y_train_preds_a[:5])
print(y_test.iloc[:5].values)
print('\n')
print(ridge_y_train_preds_b[:5])
print(y_test.iloc[:5].values)
```

```
[ 9.47906312  9.56465559  9.0493415   9.8390047  11.59443469]
[13 13 10 10 12]


[ 5.55035657  8.89277734  9.58552635  8.73224881 13.92651431]
[13 13 10 10 12]
```

In [33]:
```
from sklearn.linear_model import Lasso

lasso_reg = Lasso(alpha=0.1)

```

In [34]:
```
#lasso Reg Preds without G1/G2
lasso_y_train_preds_a  = cross_val_predict(lasso_reg, X_Train_prepared, y_
#lasso Reg Preds with G1/G2
lasso_y_train_preds_b  = cross_val_predict(lasso_reg, X_Train_prepared_no_

print(lasso_y_train_preds_a[:5])
print(y_test.iloc[:5].values)
print('\n')
print(lasso_y_train_preds_b[:5])
print(y_test.iloc[:5].values)
```

```
[ 9.66791437 10.74944002 10.00331228 11.21810024 11.1145995 ]
[13 13 10 10 12]


[ 5.01198318 10.6335893   9.56606828  9.56773496 13.69584772]
[13 13 10 10 12]
```

Fine-Tune the System

Pick one model and fine tune with GridSearchCV

In [35]:
```
#I will be choosing the Lasso Linear Model for the GridSearch
```

In [36]:
```
from sklearn.model_selection import GridSearchCV

# setup the hyperparameter values to search - we'll learn more about these
lasso_params = {'alpha':[.001,.01,.025,.05,.1,.25,.3,.35,.4,.45,.5,1]}



# instantiate grid search
grid_search = GridSearchCV(lasso_reg, lasso_params, cv=5,
scoring='neg_mean_squared_error',
return_train_score=True)

# run grid search
grid_search.fit(X_Train_prepared, y_train)
#grid_search.fit(X_Train_prepared_no_drops, y_train)
```

Out[36]:
```
▸ GridSearchCV
▸ estimator: Lasso
   ▸ Lasso
```

In [37]:
```
#get the best alpha
grid_search.best_params_
```

Out[37]: {'alpha': 0.3}

Fine-Tune the System

Correctly transform the testing data

In [38]:
```
# Transform Test Data, Do not Fit_Transform
X_Test_prepared = preprocessing.transform(X_Test)
#Output Check
X_Test_prepared.shape
```

Out[38]: (74, 42)

In [39]:

```
# Transform Test Data, Do not Fit_Transform
X_Test_prepared_no_drops = preprocessings.transform(X_test)
#Output Check
X_Test_prepared_no_drops.shape
```

Out[39]: (74, 44)

Fine Tune the System

Select final model and measure its performance on the test set

In [40]:

```


lasso_tuned_a = Lasso(alpha=0.3)
lasso_tuned_b =Lasso(alpha=0.3)

final_lasso_a = lasso_tuned_a.fit(X_Train_prepared, y_train)
final_lasso_b = lasso_tuned_b.fit(X_Train_prepared_no_drops, y_train)

final_predictions_a = final_lasso_a.predict(X_Test_prepared)
final_predictions_b = final_lasso_b.predict(X_Test_prepared_no_drops)

print(final_predictions_a[:5].round(2))
print(y_test.iloc[:5].values)
print('\n')
print(final_predictions_b[:5].round(2))
print(y_test.iloc[:5].values)

```

```
[ 9.5  10.62 11.37 10.59 11.25]
[13 13 10 10 12]


[12.49 12.92  9.24  9.65 12.74]
[13 13 10 10 12]
```

In [41]:

```
lasso_score_a = lasso_tuned_a.score(X_Test_prepared[:74], y_test.iloc[:74]
print(lasso_score_a.round(2))

lasso_score_b = lasso_tuned_b.score(X_Test_prepared_no_drops[:74], y_test.
print(lasso_score_b.round(2))
```

```
0.17
0.95
```

Present Your Solution See below

Conclusions

There is some difficulty in running the getting the prediction close without the G1/G2 data with the Linear Models. The linear, Logistic and Ridge models did not seem to accurately depict the y_test data. After a brief deliberation, I have then chose to move foward with the the Lasso Model. The thought that the lasso has produced a closeness to predict the y_train data. It does not seem to be overfitted either. There is some variance between the predicted values and the y_train values.

I did run a lasso score on the with and without G1/G1 Test sets. It shows that the "B" set which has high correlation values with G1 & G2 preformed exceptionally well(.95 of 1.0) but the lasso model without the highly correlated G1 & G2 values ("A" model) performed well under the "B" model with a .17 score. A score this close to 0 (with 0 being a deregard for input values), cannot be relied upon to make concrete predictions. The model must be futher improved, at the risk of overfitting the training data.

Increasing the value of threshold value in the VarianceThreshold Feature Selector May be a way to improve results by further reducing low variance data in the model. This would have to be tested in a few different iterations to improve the "A" scores.