01

February 4, 2021

1 Linear Regression

[1]:
```
import pandas as pd
import numpy as np
import sklearn
from sklearn import linear_model
from sklearn.utils import shuffle
import os
```

[2]:
```
# Prepare directories and paths
dir_here = os.path.abspath("")
dir_base = os.path.dirname(dir_here)
dir_data = os.path.join(dir_base, "data")
path_data_csv = os.path.join(dir_data, "student-mat.csv")
```

[3]:
```
# Read data
data = pd.read_csv(path_data_csv, sep=";")
print(data.head())
```

```
  school sex  age address famsize Pstatus  Medu  Fedu     Mjob      Fjob  … \
0     GP   F   18       U     GT3       A     4     4  at_home   teacher  …
1     GP   F   17       U     GT3       T     1     1  at_home     other  …
2     GP   F   15       U     LE3       T     1     1  at_home     other  …
3     GP   F   15       U     GT3       T     4     2   health  services  …
4     GP   F   16       U     GT3       T     3     3    other     other  …

  famrel freetime  goout  Dalc  Walc health absences  G1  G2  G3
0      4        3      4     1     1      3        6   5   6   6
1      5        3      3     1     1      3        4   5   5   6
2      4        3      2     2     3      3       10   7   8  10
3      3        2      2     1     1      5        2  15  14  15
4      4        3      2     1     2      5        4   6  10  10

[5 rows x 33 columns]
```

[4]:
```
# Filter the data that interests me
data = data[['G1', 'G2', 'G3', 'studytime', 'failures', 'absences']]
```

```
print(data.head())
```

```
   G1  G2  G3  studytime  failures  absences
0   5   6   6          2         0         6
1   5   5   6          2         0         4
2   7   8  10          2         3        10
3  15  14  15          3         0         2
4   6  10  10          2         0         4
```

1.1 Some Nomenclature

Atributes are the different variables, i.e. the fields or names of the columns in the dataframe.

Strictly speaking the variable to predict is not an attribute, and will have to be dropped from the dataframe.

We are going to predict the variable 'G3'.

1.2 Data preparation

1.2.1 Select the variable to predict

[5]:
```
predict = "G3"
x = np.array(data.drop([predict], 1))
y = np.array(data[predict])
print("x:")
print(x)
print("---")
print("y:")
print(y)
```

```
x:
[[ 5  6  2  0  6]
 [ 5  5  2  0  4]
 [ 7  8  2  3 10]
 …
 [10  8  1  3  3]
 [11 12  1  0  0]
 [ 8  9  1  0  5]]
---
y:
[ 6  6 10 15 10 15 11  6 19 15  9 12 14 11 16 14 14 10  5 10 15 15 16 12
  8  8 11 15 11 11 12 17 16 12 15  6 18 15 11 13 11 12 18 11  9  6 11 20
 14  7 13 13 10 11 13 10 15 15  9 16 11 11  9  9 10 15 12  6  8 16 15 10
  5 14 11 10 10 11 10  5 12 11  6 15 10  8  6 14 10  7  8 18  6 10 14 10
 15 10 14  8  5 17 14  6 18 11  8 18 13 16 19 10 13 19  9 16 14 13  8 13
 15 15 13 13  8 12 11  9  0 18  0  0 12 11  0  0  0  0 12 15  0  9 11 13
  0 11  0 11  0 10  0 14 10  0 12  8 13 10 15 12  0  7  0 10  7 12 10 16
  0 14  0 16 10  0  9  9 11  6  9 11  8 12 17  8 12 11 11 15  9 10 13  9
```

```
 8 10 14 15 16 10 18 10 16 10 10  6 11  9  7 13 10  7  8 13 14  8 10 15
 4  8  8 10  6  0 17 13 14  7 15 12  9 12 14 11  9 13  6 10 13 12 11  0
12 12  0 12  0 18 13  8  5 15  8 10  8  8 12  8 13 11 14  0 18  8 12  9
 0 17 10 11 10  0  9 14 11 14 10 12  9  9  8 10  8 10 12 10 11 11 19 12
14 15 11 15 13 18 14 11  0  8 14 16 11 10 14 18 13 12 18  8 12 10  0 13
11 11 13 11  0  9 10 11 13  9 11 15 15 11 16 10  9 14  8 14  0  0  0 15
13  0 17 10 11  0 15  0 10 14 16  9 15 13  8 13  8  8 11  9 13 11 10 16
13 12 10 15 12 10 13  0 10 11  9 12 11  5 19 10 15 10 15 10 14  7 10  0
 5 10  6  0  8  0  9 16  7 10  9]
```

[6]:
```
print(x.shape)
print("---")
print(y.shape)
```

```
(395, 5)
---
(395,)
```

1.2.2 Split the data between train and select batches

[7]:
```
x_train, x_test, y_train, y_test = sklearn.model_selection.train_test_split(x,␣
↪y, test_size=0.1)
```

[8]:
```
print(x_train)
print("---")
print(y_train)
```

```
[[12 13  2  0  2]
 [ 5  6  2  0  6]
 [10 10  1  0  2]
 …
 [ 9  9  2  0  2]
 [18 16  3  0  0]
 [11 10  1  0  0]]
---
[13  6 10  9  0  8  8 18 18 13  6 12 10 11  4 10  6  0 10 10  8  5 13 10
 11  7 14  0  8 11 11  6 10  7  0  8  7 15 10 11 14 10 15  9  5 15  0 11
  9 10  8 13  0  0 14  0  9  9  8 10  0  9  8 15 15  8  8  0 11 11  0 10
  5 15 12 12 12 13 10 12  9  8 14 12 15 10 12  5 17 14  0 11 14 14  6 10
 10 15 11  9 19  0 14  8  9 13 11 11 10  0 15  0  0 10  8 10  0 15 17 16
  0 16 11 12 13 12 14 10  9 10 11 11  0 10  6 14  9 13 14 15  7  8  9 10
 16 14  6 13  9 15 10 13 12 11 12 13 10 16 10 12 16  8 12  0  8 16  0  8
 18 14  0 10  6 11  8  7 14 15 10 15 13  8 14 11 10  9 10  9  9 14  0 11
 15  8  6 14 10 12 15 10 10  9 16 13 11 12 11 10 13  8 19 15 11 17 13  7
 13  0 10 12  8 11 15 10 10 15 17 18  9 12  0 13 11 12 17 16  0 13 11 19
 14 12 13 15  9 10 11 12  7  9  8  9 10  9 13 12 13  9 12 10  0  0 14 11
  6  8 10 11 15 11 18 10 11 12  6 13 18 14 10 11 15 11  6  0 12 14 13  0
 20 16 15  8  9 13 14 16 14 12  5 18  0 13 15  7  6 12  8  6 11  6 14 10
```

```
 11 15  8  0  8 11  0 13 10  0 11 10 19 14 19  8  9 11 15 16 16 10 11 14
  0 13 11 16 11  5 11  8 10  9  7 15  9 11 15 10 10 16 10]
```

[9]:
```
print(x_train.shape)
print("---")
print(y_train.shape)
```

```
(355, 5)
---
(355,)
```

1.2.3 Create a linear regression model

[10]:
```
# Create linear regression object
linear = sklearn.linear_model.LinearRegression()
```

[11]:
```
# Fit linear regression to the train data
linear.fit(x_train, y_train)
# The fit is stored in the linear object already created
```

[11]:
```
LinearRegression()
```

[12]:
```
# Check accuracy with the test data
acc = linear.score(x_test, y_test)
print(acc)
```

```
0.8187133465709397
```

We get a 79% accuracy.

For the case that's decent enough.

1.2.4 Check the linear regression

[13]:
```
# Check the coefficients
print(f"Coefficients: {linear.coef_}")
print(f"Intercept: {linear.intercept_}")
```

```
Coefficients: [ 0.14796322  0.98384801 -0.1994292  -0.32181276  0.03758345]
Intercept: -1.422785306660396
```

[14]:
```
# Check some predictions made by the model
predictions = linear.predict(x_test)
for k in range(len(predictions)):
    print(f"{y_test[k]:02d} ; {predictions[k]:06.3f} ; {x_test[k]}")
```

```
11 ; 07.954 ; [10  8  1  0  6]
12 ; 11.734 ; [14 12  2  1  0]
00 ; -1.848 ; [5 0 1 3 0]
```

```
18 ; 18.976 ; [18 18  1  0  6]
15 ; 13.479 ; [15 14  3  2  4]
11 ; 10.208 ; [ 9 11  3  0  2]
15 ; 15.717 ; [16 15  2  0 11]
15 ; 15.008 ; [14 15  2  0  0]
10 ; 09.647 ; [10 10  2  0  4]
09 ; 09.424 ; [ 9 10  2  0  2]
13 ; 12.967 ; [13 13  2  0  2]
11 ; 11.906 ; [13 11  3  1 40]
14 ; 14.024 ; [14 14  2  0  0]
11 ; 09.868 ; [12 10  2  0  2]
08 ; 07.752 ; [11  8  2  0  2]
10 ; 09.774 ; [11 10  2  1 12]
10 ; 10.147 ; [12 10  2  1 18]
18 ; 18.255 ; [16 18  2  0  0]
13 ; 12.275 ; [10 13  4  0  6]
17 ; 16.363 ; [16 16  2  0  2]
16 ; 16.363 ; [16 16  2  0  2]
00 ; 05.317 ; [7 6 1 0 0]
12 ; 13.270 ; [12 13  2  0 14]
12 ; 11.293 ; [10 12  2  1  4]
11 ; 10.184 ; [ 8 11  2  0  0]
18 ; 18.774 ; [19 18  2  0  2]
12 ; 13.091 ; [13 13  1  0  0]
15 ; 15.008 ; [14 15  2  0  0]
15 ; 15.156 ; [15 15  2  0  0]
08 ; 07.057 ; [9 8 4 0 2]
18 ; 19.192 ; [17 18  2  0 21]
05 ; 04.389 ; [ 6  5  1  1 14]
18 ; 19.331 ; [18 18  1  1 24]
00 ; 09.175 ; [10 10  2  1  0]
12 ; 12.892 ; [13 13  2  0  0]
10 ; 10.530 ; [11 11  4  0  8]
13 ; 11.948 ; [14 11  1  0 18]
10 ; 07.451 ; [8 9 1 3 0]
13 ; 12.871 ; [15 13  3  2 14]
16 ; 15.231 ; [15 15  2  0  2]
```

2 Saving Models and Plotting Data

2.1 Save the model with Pickle

[15]:
```
import pickle # standard Python module to save objects
```

[16]:
```
with open("studentmodel.pickle", "wb") as fid:
    pickle.dump(linear, fid)
```

[17]:
```
# Now we can retrieve the model
pickle_in = open("studentmodel.pickle", "rb")
linear_2 = pickle.load(pickle_in)
# Check the coefficients
print(f"Coefficients: {linear.coef_}")
print(f"Intercept: {linear.intercept_}")
```

```
Coefficients: [ 0.14796322  0.98384801 -0.1994292  -0.32181276  0.03758345]
Intercept: -1.422785306660396
```

It's the same model as before

2.2 Train several models and keep the best

[18]:
```
def train_linear_model(x,y):
    x_train, x_test, y_train, y_test = sklearn.model_selection.
 ↪train_test_split(x, y, test_size=0.1)
    linear = sklearn.linear_model.LinearRegression()
    linear.fit(x_train, y_train)
    acc = linear.score(x_test, y_test)
    return linear, acc
```

[19]:
```
# Train N models and keep the best
N = 1000
max_acc = 0
for _ in range(N):
    _linear, _acc = train_linear_model(x,y)
    if _acc > max_acc:
        linear_3 = _linear
        acc_3 = _acc
print("Best model accuracy: ", acc_3)
print("... saving model to file 'studentmodel.pickle'")
with open("studentmodel.pickle", "wb") as fid:
    pickle.dump(linear_3, fid)
```

```
Best model accuracy:  0.8220903411761525
… saving model to file 'studentmodel.pickle'
```

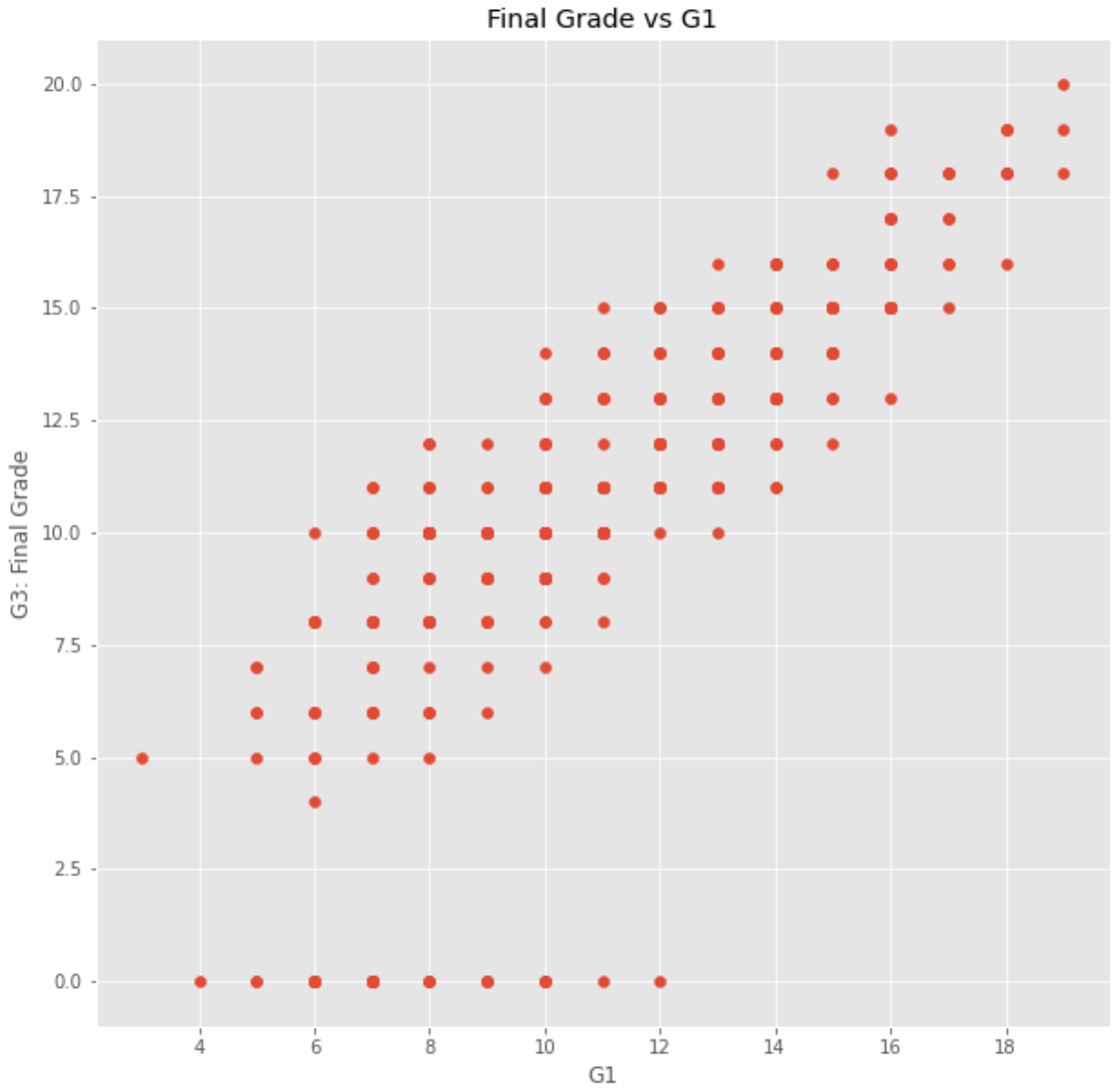
2.3 Plotting

[20]:
```
from matplotlib import pyplot as plt
from matplotlib import style
%matplotlib inline
```

[21]:
```
style.use("ggplot")
```

[22]:
```
fig, ax = plt.subplots(figsize=(10,10))
p = 'G1'
ax.scatter(data[p], data['G3'])
ax.set_xlabel(p)
ax.set_ylabel('G3: Final Grade')
ax.set_title('Final Grade vs G1')
```

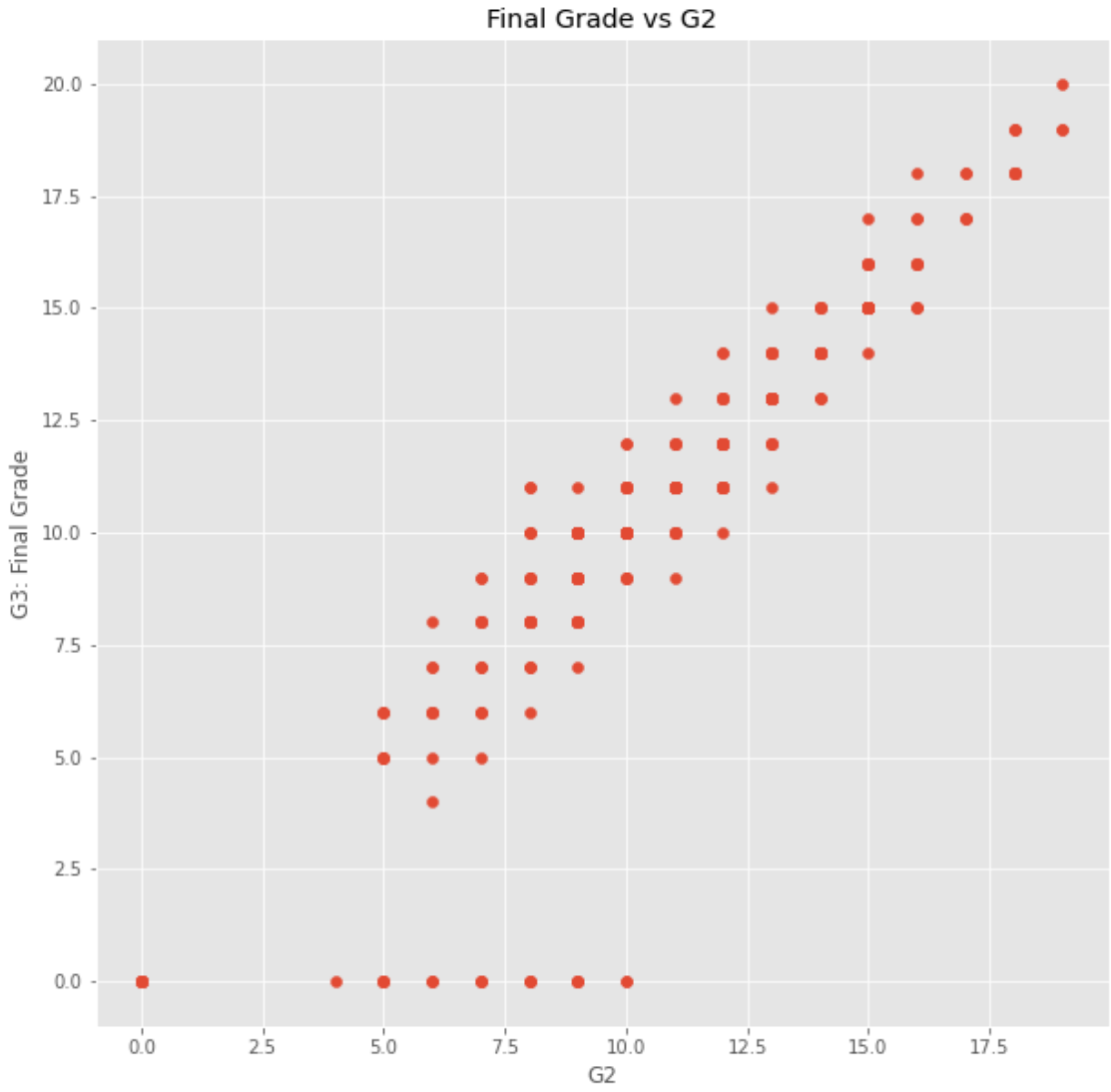
[22]:
```
Text(0.5, 1.0, 'Final Grade vs G1')
```



[23]:
```
fig, ax = plt.subplots(figsize=(10,10))
p = 'G2'
ax.scatter(data[p], data['G3'])
ax.set_xlabel(p)
```

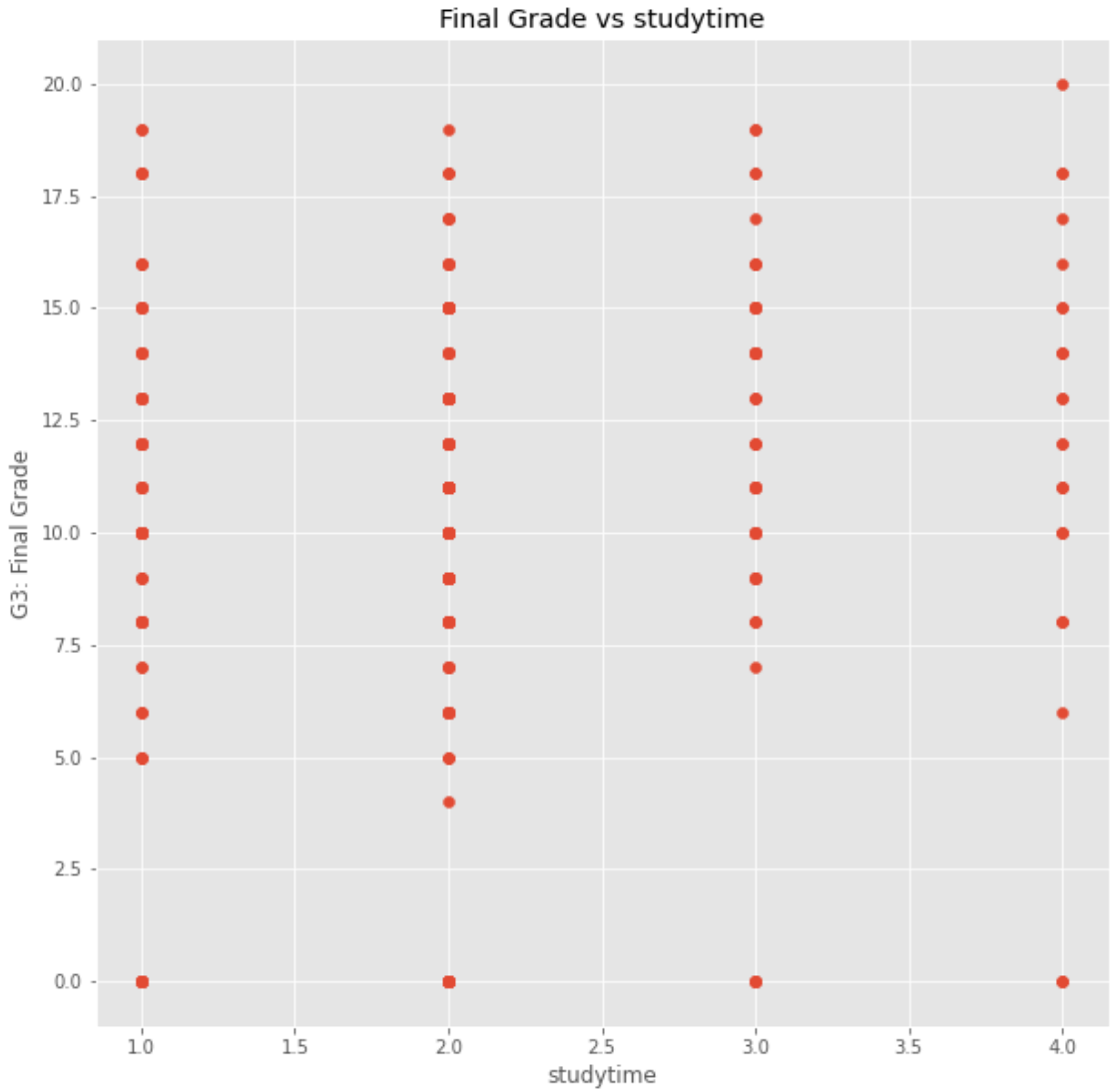
```
ax.set_ylabel('G3: Final Grade')
ax.set_title(f'Final Grade vs {p}')
```

[23]: Text(0.5, 1.0, 'Final Grade vs G2')



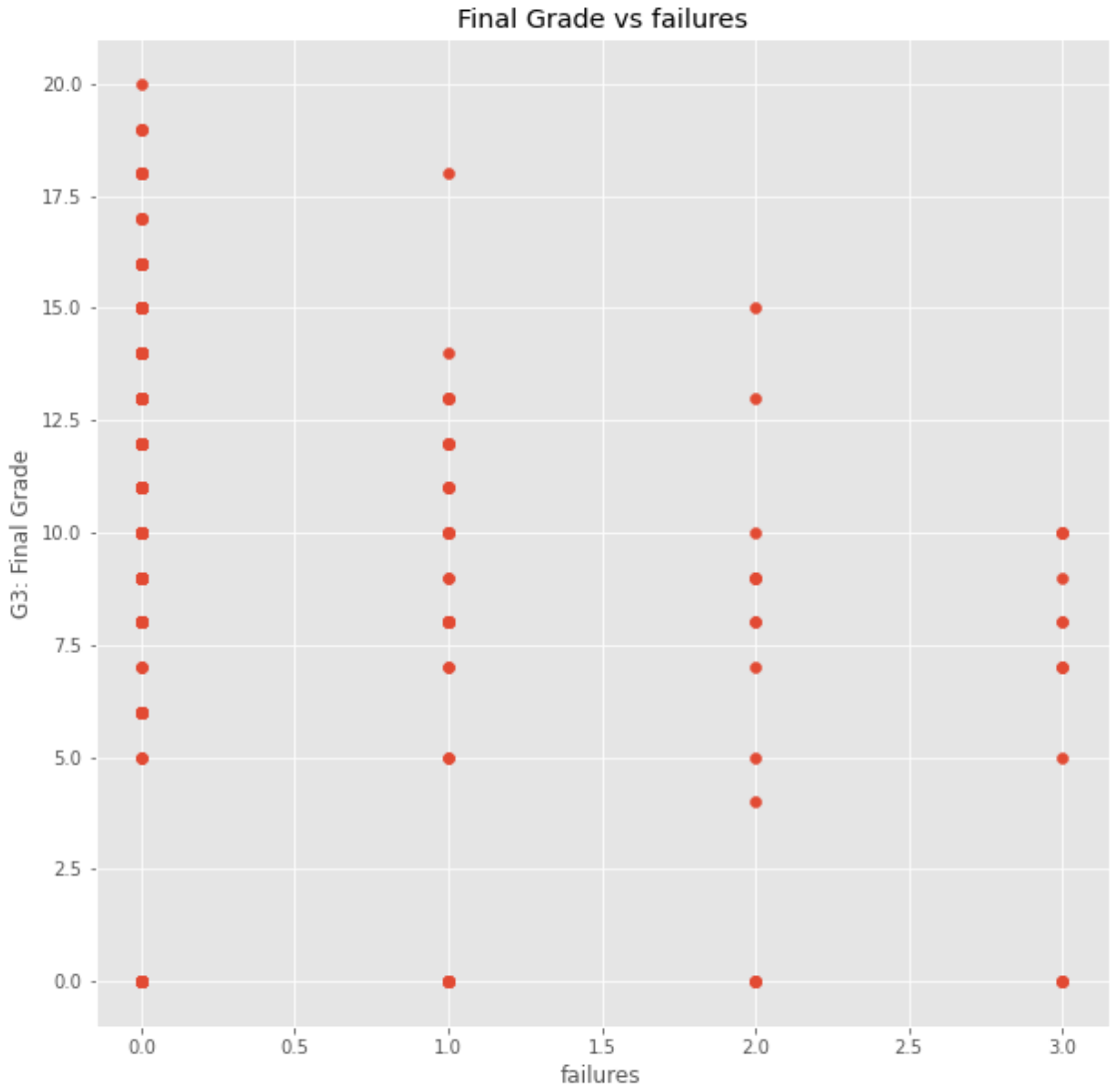
[24]:
```
fig, ax = plt.subplots(figsize=(10,10))
p = 'studytime'
ax.scatter(data[p], data['G3'])
ax.set_xlabel(p)
ax.set_ylabel('G3: Final Grade')
ax.set_title(f'Final Grade vs {p}')
```

[24]: Text(0.5, 1.0, 'Final Grade vs studytime')



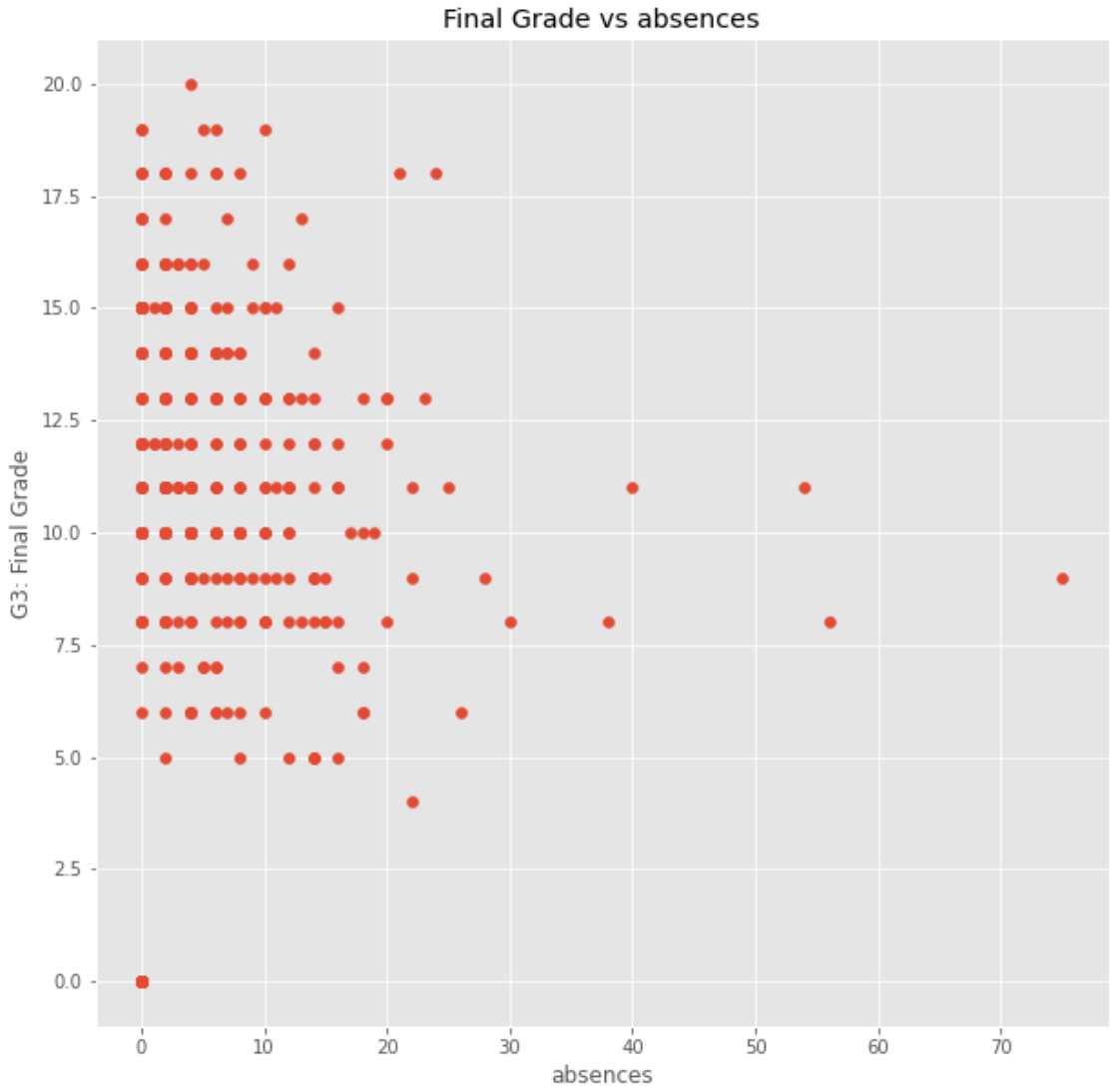
[25]:
```
fig, ax = plt.subplots(figsize=(10,10))
p = 'failures'
ax.scatter(data[p], data['G3'])
ax.set_xlabel(p)
ax.set_ylabel('G3: Final Grade')
ax.set_title(f'Final Grade vs {p}')
```

[25]: Text(0.5, 1.0, 'Final Grade vs failures')



[26]:
```
fig, ax = plt.subplots(figsize=(10,10))
p = 'absences'
ax.scatter(data[p], data['G3'])
ax.set_xlabel(p)
ax.set_ylabel('G3: Final Grade')
ax.set_title(f'Final Grade vs {p}')
```

[26]: Text(0.5, 1.0, 'Final Grade vs absences')

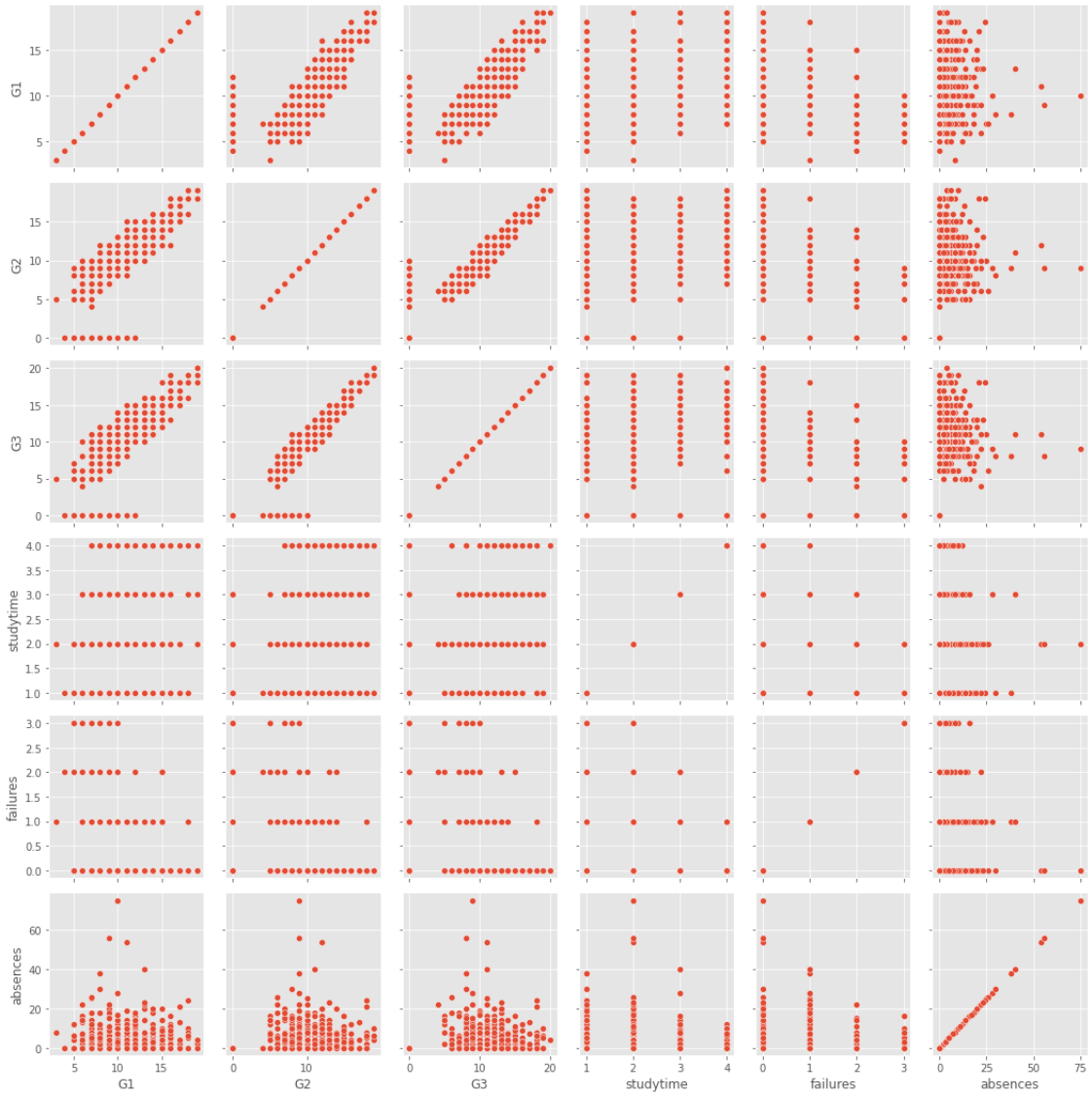


2.4 Try with seaborn

[27]:
```
import seaborn as sns
```

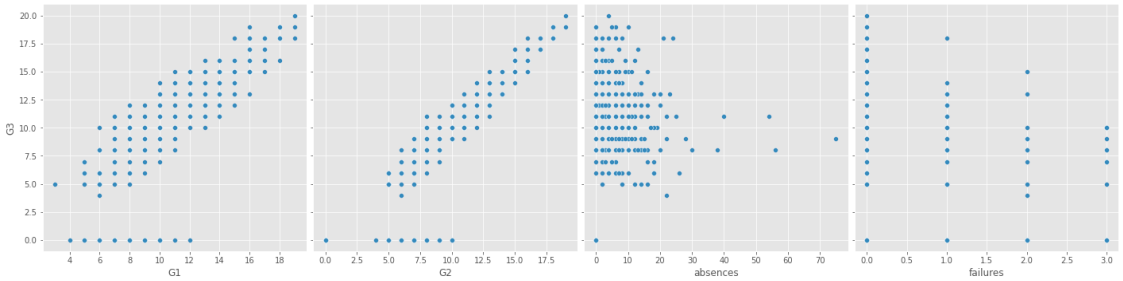
[28]:
```
g = sns.PairGrid(data)
g.map(sns.scatterplot)
```

[28]:
```
<seaborn.axisgrid.PairGrid at 0x7f4e500b6a90>
```



[29]:
```
# Again
g = sns.pairplot(data=data, x_vars=["G1", "G2", "absences", "failures"],␣
 ↪y_vars=["G3"], height=5)
g.map(sns.scatterplot)
```

[29]: <seaborn.axisgrid.PairGrid at 0x7f4e4a1e73d0>


G3
G1
G2
absences
failures

[ ]: