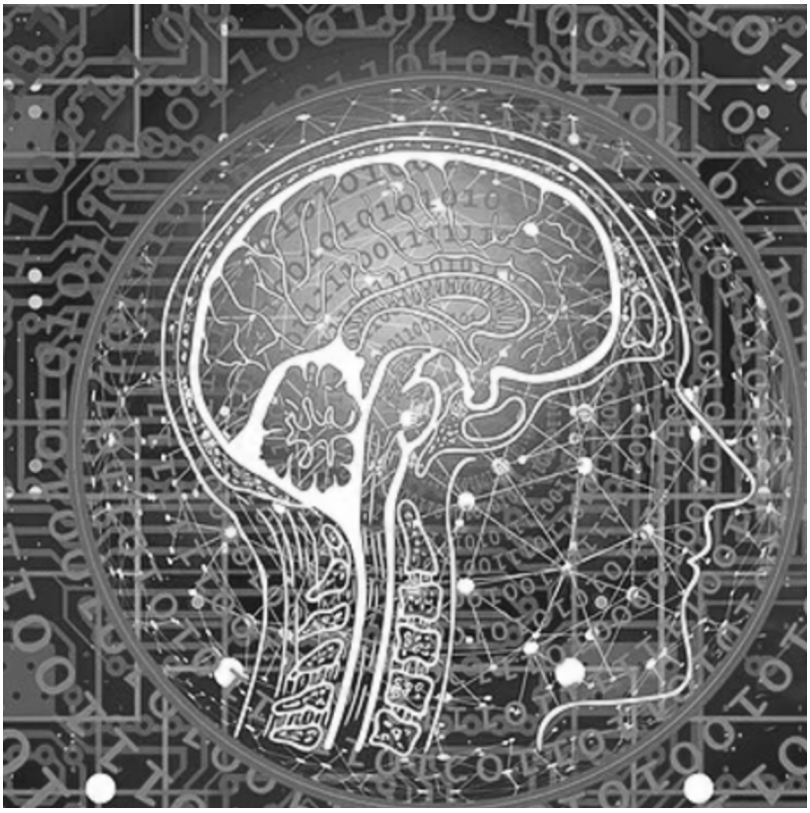


Coursera IBM-- Supervised Machine Learning: Classification Final Project

By: Liam Webster

6/27/2022

Objective:

The main objective of this analysis was to engineer an interpretable model that successfully predicts a students academic performance. This analysis will largely focus on uncovering interpretable model features that heavily influence a student's academic performance. For students this model can help them predict their academic performance in reflection of their individual current features, although this should be taken lightly. Largely this model could be used to help clarify important behaviors and habits in student success.

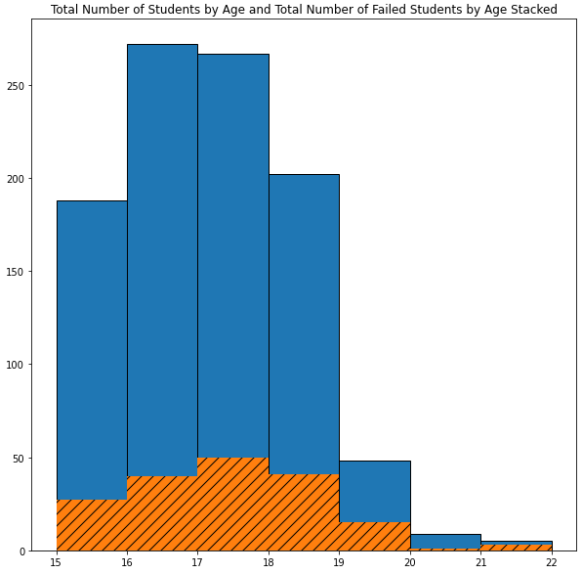
Dataset:

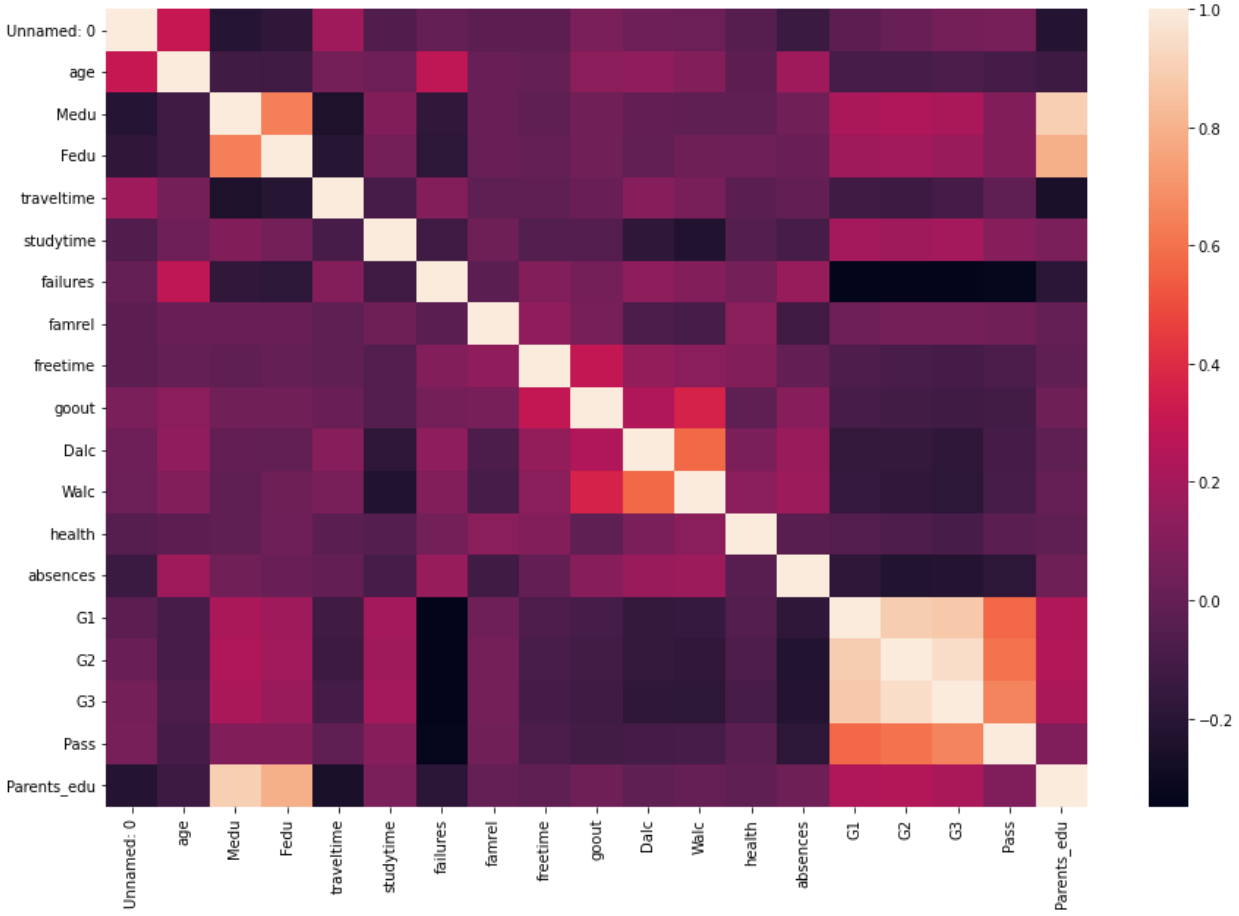
The data set I have chosen contains information about students'--aged 15-22–academic performance. The dataset has 19 columns– four label columns and 15 feature columns. The label columns consist of three columns containing test scores and a column containing whether the student passed or failed the course. A couple of the feature columns include: sex, age, family size, weekend alchol consumption, and parental education history. With this analysis I aim to uncover some of the most important features in regards to academic performance.

Exploratory Data Analysis:

An initial manual inspection of the dataset in VS Code's Excel viewer was done. Then some initial plotting was done. Which included a breakdown of the Pass to Fail ratio of the dataset. Which was about 80% 'Pass' observations and 20% 'Fail' observations. I then began to make inferences about data trends and feature importance. I plotted a histogram illustrating the trend between the number of previously failed classes versus the number of 'Fail' observations. It was found that as the number of previously failed classes increased the likelihood of failing increased significantly. A full correlation heat map was created. It was observed from the heatmap that "Pass" is positively correlated with each of the class test scores("G1","G2","G3"); this makes intuitive sense. It was also observed that "Pass" is negatively correlated with "absences", "Walc(weekend alcohol consumption", "Dalc(weekday alcohol consumption", "freetime", "failures", and "age". All of these correlations make intuitive sense except the "age" correlation, which was investigated further. By observation of plotting the total number of students by age and total number of failed students by age stacked it was clear that due to the low number of older student samples an unbalanced class was present. This lead to a synthetic negative correlation between age and student pass rate thus the 'age' feature column was not be used in this analysis. Concluding the exploratory data analysis it was determined to only use 'absences', 'Walc', 'Dalc', 'goout', 'freetime', and 'failures' as the feature columns and 'Pass' as the label column.


Pass Fail Distribution of Dataset
Passed
Failure


Total Number of Students by Age and Total Number of Failed Students by Age Stacked
250
200
150
100
50
0
15
16
17
18
19
20
21
22


Unnamed: 0
age
Medu
Fedu
traveltime
studytime
failures
famrel
freetime
goout
Dalc
Walc
health
absences
G1
G2
G3
Pass
Parents_edu
1.0
0.8
0.6
0.4
0.2
0.0
−0.2

Models:

To preface the same stratified data split was used across all models. The first model used was a simple logistic classifier, using the ‘liblinear’ solver. This model produced the following results:

	Train	Test
Accuracy	0.831	0.844
Precision	0.841	0.851
Recall	0.980	0.982
F1	0.905	0.912

This model was quite accurate and provided good interpretability. The coefficients with the largest magnitude were ‘failures’, ‘goout’, and ‘absences’. All were negative coefficients.
The second model used was a decision tree classifier. This model produced the following results:

	Train	Test
Accuracy	0.965	0.704
Precision	0.997	0.842
Recall	0.960	0.785
F1	0.978	0.813

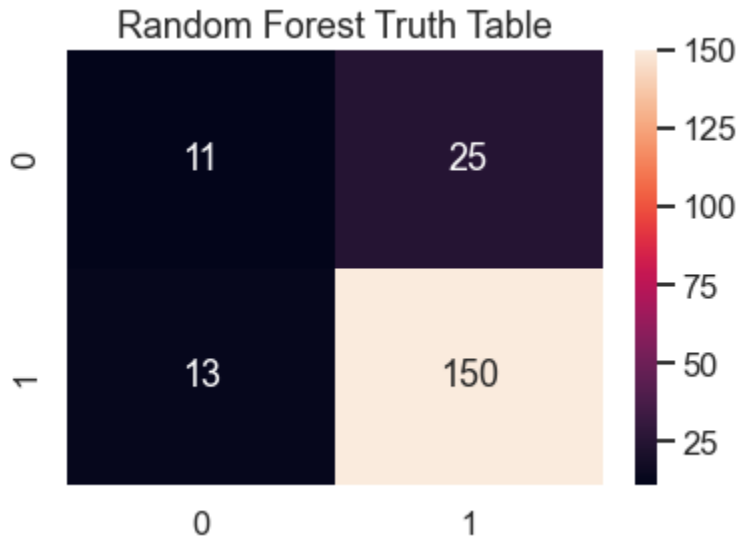
This model produced very accurate training results but did not generalize well and produced subpar testing results. This decision tree model was not interpretable with it being composed of 477 nodes and having a depth of 14. This overfitting trend was not surprising and the next steps were to produce a decision tree with better generalization thus Grid Search was used to find optimal parameters. This model produced the following results:

	Train	Test
Accuracy	0.837	0.839
Precision	0.843	0.847
Recall	0.985	0.982
F1	0.909	0.909

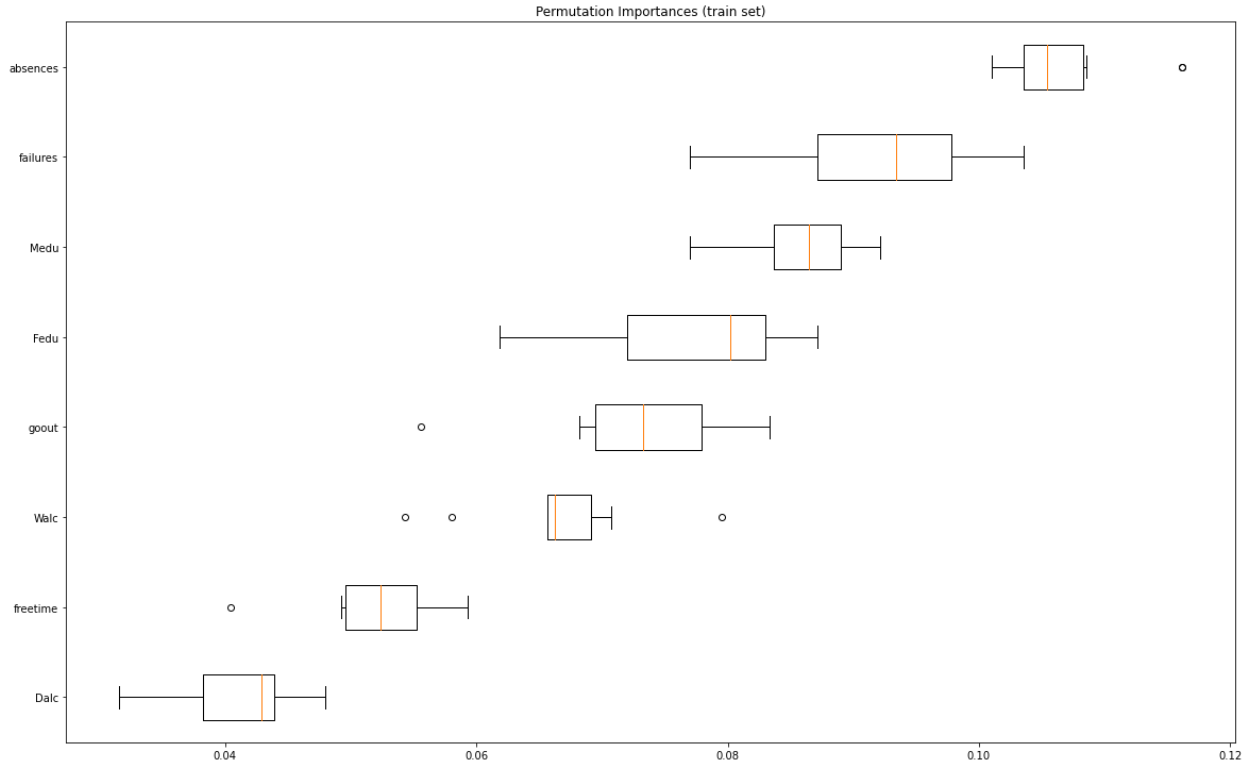
This decision tree produced much better generalized results. The decision tree was composed of 15 nodes and a depth of 3, thus a much smaller and quicker classifying model.

The next model used was an ensemble model composed of a random forest and a logistic regression as a surrogate model. The random forest model produced the following results:

	Train	Test
Accuracy	0.837	0.839
Precision	0.843	0.847
Recall	0.985	0.982
F1	0.909	0.909



This random forest produced similar results as the decision tree classifier. A permutation importance algorithm was run on the random forest to determine the most important features. The following plot was produced:



Next a logistic regression was fit to our black box random forest. It correctly predicted our blackbox random forest model 90% of the time. With the logistic regression fit it was easy to observe feature importance, with ‘failures’, ‘absences’, and ‘freetime’ having the largest magnitude.

Model Summary:

The best performing model was our black box random forest. It produced the best general results. The vanilla logistic classification was a close second, producing almost replicable results. The logistic regression model was more interpretable and it was discovered that ‘failures’, ‘goout’, and ‘absences’ were the most important features for the logistic classification model. Using a surrogate logistic classification model for the random forest it was observed that ‘failures’, ‘absences’, and ‘freetime’ were the most important features. With this in mind it is recommended to use the vanilla logistic regression model. It produced near replicable results as the random forest model, while keeping complexity to a minimum, and interpretability to a maximum.

Key Findings and Insights:

From a culmination of all the models it can be reasoned that the number of previously failed classes, number of absences, amount of nights going out, and amount of freetime were the biggest factors in determining a students success. Thus to increase the likelihood of student success, behaviors that influence or produce these features should be evaluated and disciplined.

Further Research:

To further this research the next steps to be taken could be to collect more data and continue to fit and test our logistic model. Specifically the logistic classifier model correctly predicted passing students extremely well, but did worse at predicting failing students. This is likely due to the data set being a skewed distribution. There were many more passing observations than failing observations. Thus more failing observations could help improve the model. New features could be introduced into the model to improve accuracy. Possibly previous features could be combined to simplify the model and create more robust individual features.

project2

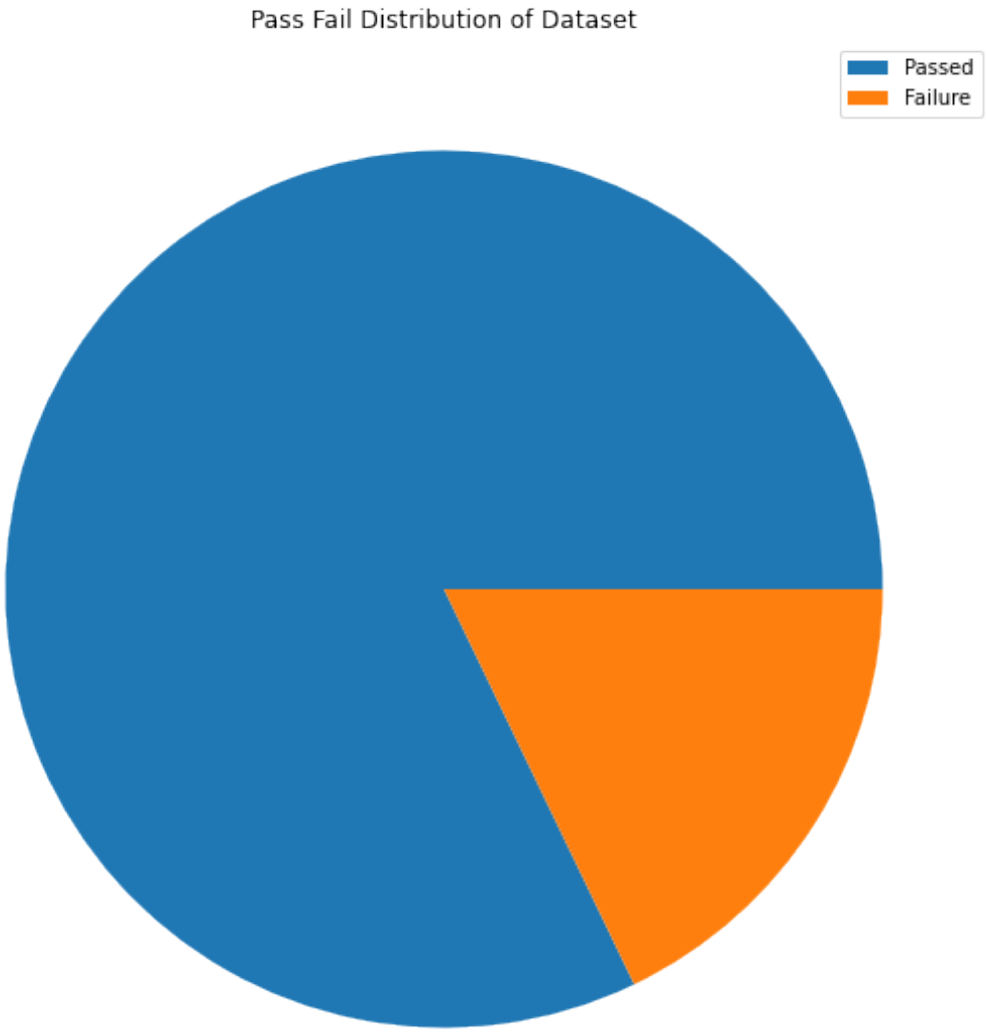
June 27, 2022

[227]:
```
# Imports
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

import warnings
warnings.filterwarnings("ignore", category=UserWarning)
warnings.filterwarnings("ignore", category=RuntimeWarning)
```

[228]:
```
# Reading Data
filepath = 'data\original_data.csv'

og_data = pd.read_csv(filepath)
#og_data.head()
```

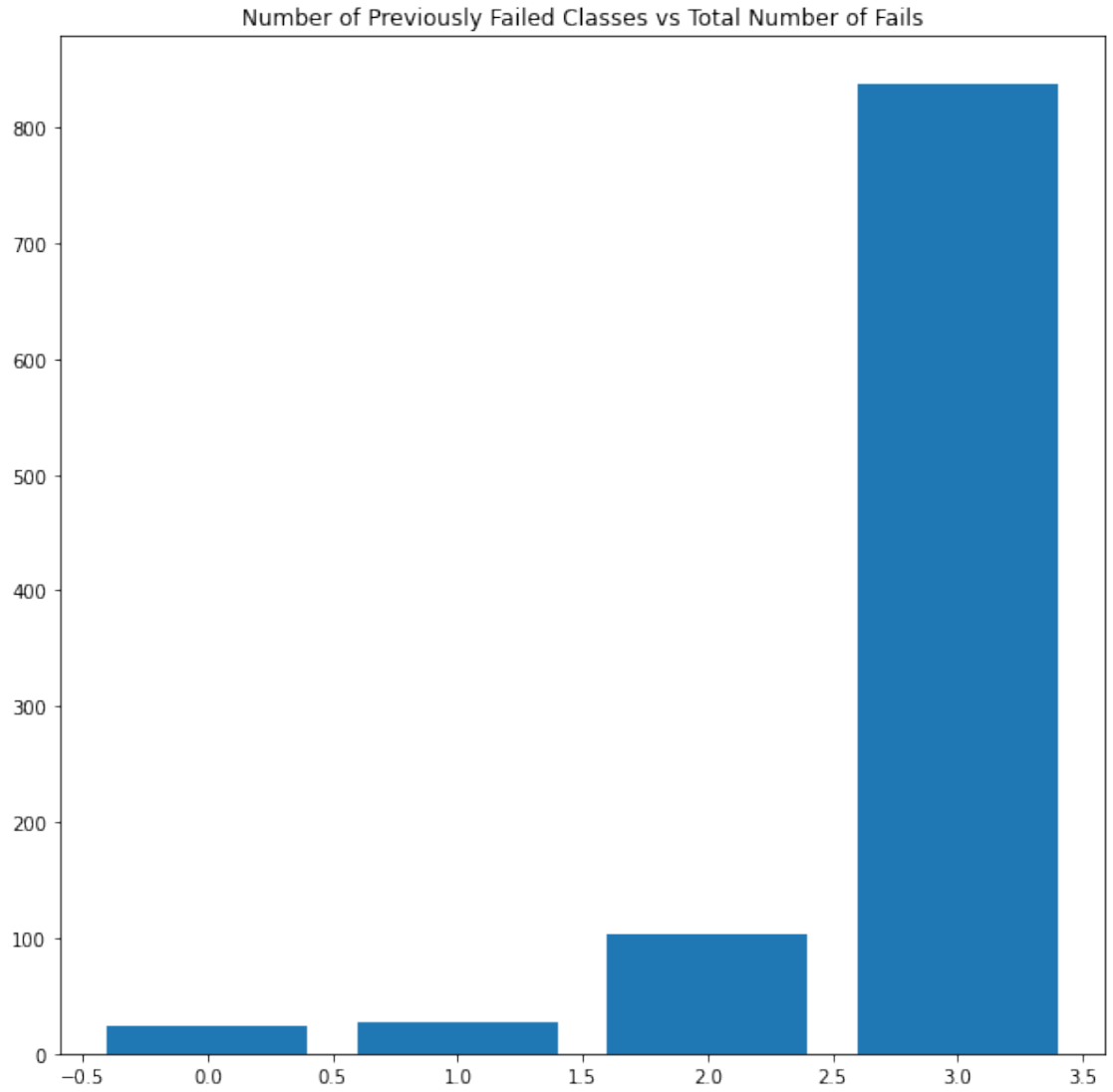
[229]:
```
# Performing Initial EDA
og_data.dtypes.value_counts()
og_data.dtypes

fig = plt.figure(figsize=(10,10))
ax = plt.axes()

ax.pie(x=og_data['Pass'].value_counts())
labels = ['Passed','Failure']
ax.legend(labels)
ax.set_title('Pass Fail Distribution of Dataset')
```

[229]:
```
Text(0.5, 1.0, 'Pass Fail Distribution of Dataset')
```



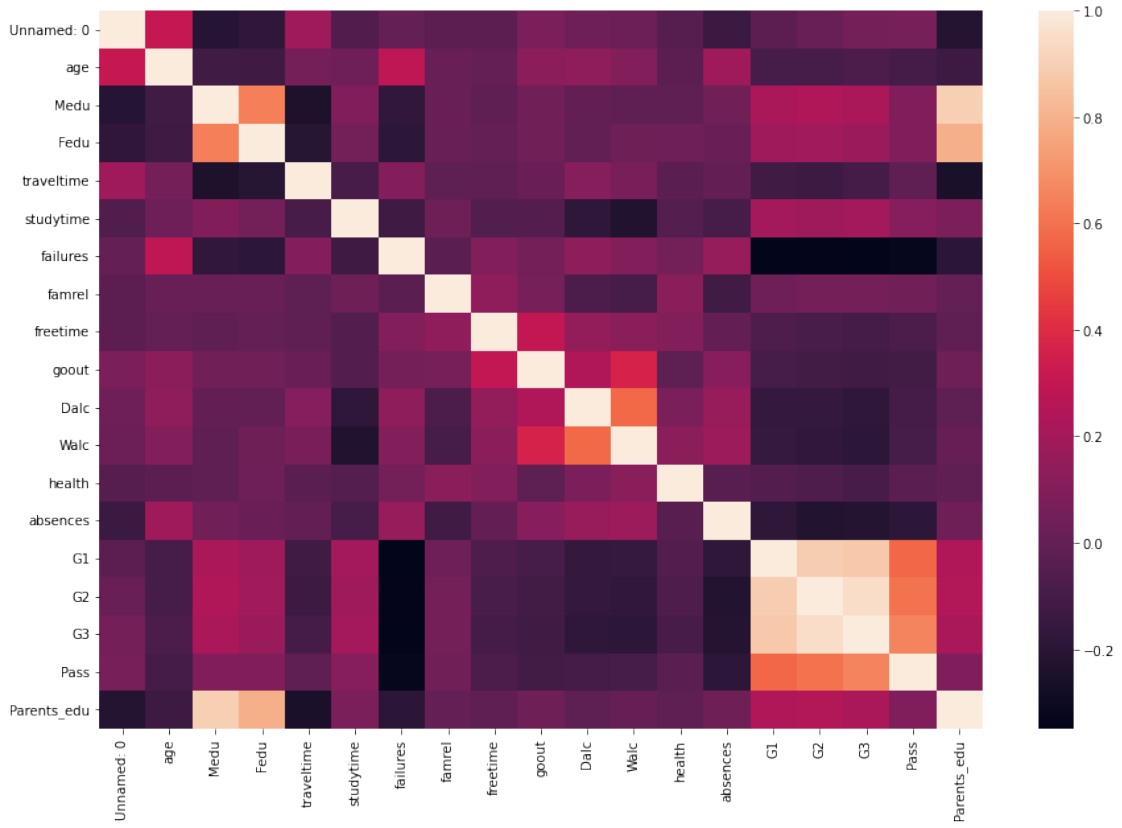
[230]:
```
three_fails = pd.DataFrame(og_data[og_data['failures'] == 3]['Pass']).shape[0]
two_fails = pd.DataFrame(og_data[og_data['failures'] == 2]['Pass']).shape[0]
one_fails = pd.DataFrame(og_data[og_data['failures'] == 1]['Pass']).shape[0]
zero_fails = pd.DataFrame(og_data[og_data['failures'] == 0]['Pass']).shape[0]
y_axis = [three_fails, two_fails, one_fails, zero_fails]
x_axis = [0,1,2,3]

fig = plt.figure(figsize=(10,10))
ax = plt.axes()
ax.bar(x=x_axis,height=y_axis)
ax.set_title('Number of Previously Failed Classes vs Total Number of Fails')
```

[230]: Text(0.5, 1.0, 'Number of Previously Failed Classes vs Total Number of Fails')



[231]:
```
fig, ax = plt.subplots(figsize=(15,10))
sns.heatmap(og_data.corr())
```

[231]: <AxesSubplot:>

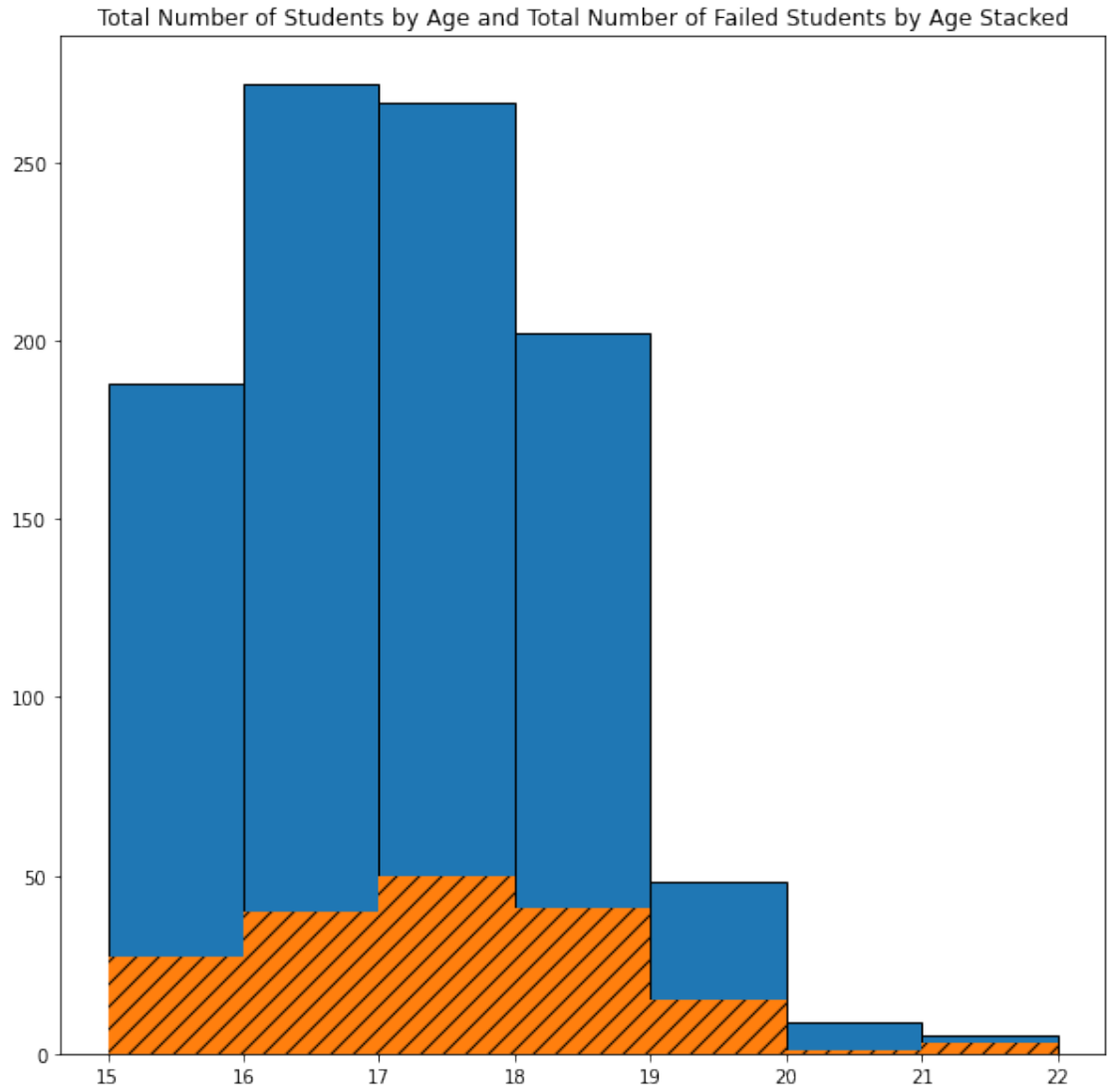


[232]:
```
# It can be observed from the heatmap that "Pass" is positively correlated with␣
↪each of the test scores("G1","G2","G3"); this makes intuitive sense. Its␣
↪also observed that "Pass" is negatively correlated with "absences",␣
↪"Walc(weekend alcohol consumption", "Dalc(weekday alcohol consumption",␣
↪"freetime", "failures", and "age". All of these correlations make intuitive␣
↪sense except the "age" correlation thus we will investigate that correlation␣
↪further.
```

[233]:
```
# Investigation into the correlation between age and pass
mybins = [15,16,17,18,19,20,21,22]
list = og_data['age']
list2 = og_data[og_data['Pass'] == False]['age']
fig = plt.figure(figsize=(10,10))
ax = plt.axes()
_, bins, _ = ax.hist(list, bins=mybins, edgecolor='black', align='mid')
ax.hist(list2, bins=bins, hatch='//')
ax.set_title('Total Number of Students by Age and Total Number of Failed␣
↪Students by Age Stacked')
print('Total Number of Students Age 22: ', og_data[(og_data['age'] ==␣
↪22)]['age'].shape[0])
```

```
print('Total Number of Students Age 22 and Failed: ', og_data[(og_data['age']␣
 ↪== 22) & (og_data['Pass'] == False)].shape[0])
```

```
Total Number of Students Age 22:  2
Total Number of Students Age 22 and Failed:  2
```



[234]:
```
# By observation of the plot it can be seen that due to the low number of older␣
 ↪student samples we have an unbalanced class. This is leading to a synthetic␣
 ↪negative correlation between age and student pass rate thus the 'age'␣
 ↪feature column will not be used in this analysis.
# To begin with I will use a simple model using only 'Pass', 'absences',␣
 ↪'Walc', 'Dalc', 'goout', 'freetime', 'failures', 'G1', 'G2', and 'G3'.
```

[235]:
```
# Feature selection and cleaning
featureCols1 = ['absences', 'Fedu', 'Medu', 'Walc', 'Dalc', 'goout',␣
↪'freetime', 'failures', 'G1', 'G2', 'G3']
labelCol1 = ['Pass']
x_data1 = og_data[featureCols1]
y_data1 = og_data[labelCol1]

y_data1 = y_data1.replace(to_replace=[True, False], value=[1,0])
```

[252]:
```
from sklearn.model_selection import StratifiedShuffleSplit

strat_shuf_split = StratifiedShuffleSplit(n_splits=1,
                                          test_size=0.3,
                                          random_state=42)

train_idx, test_idx = next(strat_shuf_split.split(x_data1, y_data1))

X_train = x_data1.loc[train_idx,:]
y_train = y_data1.loc[train_idx,:]

X_test  = x_data1.loc[test_idx,:]
y_test  = y_data1.loc[test_idx,:]
```

[268]:
```
from sklearn.metrics import accuracy_score, precision_score, recall_score,␣
↪f1_score

def measure_error(y_true, y_pred, label):
    return pd.Series({'accuracy':accuracy_score(y_true, y_pred),
                      'precision': precision_score(y_true, y_pred),
                      'recall': recall_score(y_true, y_pred),
                      'f1': f1_score(y_true, y_pred)},
                      name=label)
```

[253]:
```
# Standard logistic regression
from sklearn.linear_model import LogisticRegression

lr = LogisticRegression(solver='liblinear').fit(X_train, y_train)
```

[269]:
```
# The error on the training and test data sets
y_train_pred = lr.predict(X_train)
y_test_pred = lr.predict(X_test)

train_test_full_error = pd.concat([measure_error(y_train, y_train_pred,␣
↪'train'),
                                   measure_error(y_test, y_test_pred, 'test')],
                                   axis=1)
```

```
train_test_full_error
```

[269]:
```
              train      test
accuracy   0.945166  0.959732
precision  0.941764  0.953307
recall     0.994728  1.000000
f1         0.967521  0.976096
```

[270]:
```
# Decision Tree Model
### BEGIN SOLUTION
from sklearn.tree import DecisionTreeClassifier

dt = DecisionTreeClassifier(random_state=42)
dt = dt.fit(X_train, y_train)


### BEGIN SOLUTION
from sklearn.model_selection import GridSearchCV

param_grid = {'max_depth':range(1, dt.tree_.max_depth+1, 2),
              'max_features': range(1, len(dt.feature_importances_)+1)}

GR = GridSearchCV(DecisionTreeClassifier(random_state=42),
                  param_grid=param_grid,
                  scoring='accuracy',
                  n_jobs=-1)

GR = GR.fit(X_train, y_train)
```

[272]:
```
GR.best_estimator_.tree_.node_count, GR.best_estimator_.tree_.max_depth
```

[272]:
```
(3, 1)
```

[271]:
```
y_train_pred_gr = GR.predict(X_train)
y_test_pred_gr = GR.predict(X_test)

train_test_gr_error = pd.concat([measure_error(y_train, y_train_pred_gr,␣
↪'train'),
                                 measure_error(y_test, y_test_pred_gr, 'test')],
                                axis=1)
train_test_gr_error
```

[271]:
```
           train  test
accuracy     1.0   1.0
precision    1.0   1.0
recall       1.0   1.0
f1           1.0   1.0
```

[275]:
```
dt.tree_.node_count, dt.tree_.max_depth
```

[275]:
```
(3, 1)
```

[280]:
```
list = og_data.dtypes == int
```

[ ]:
```

```