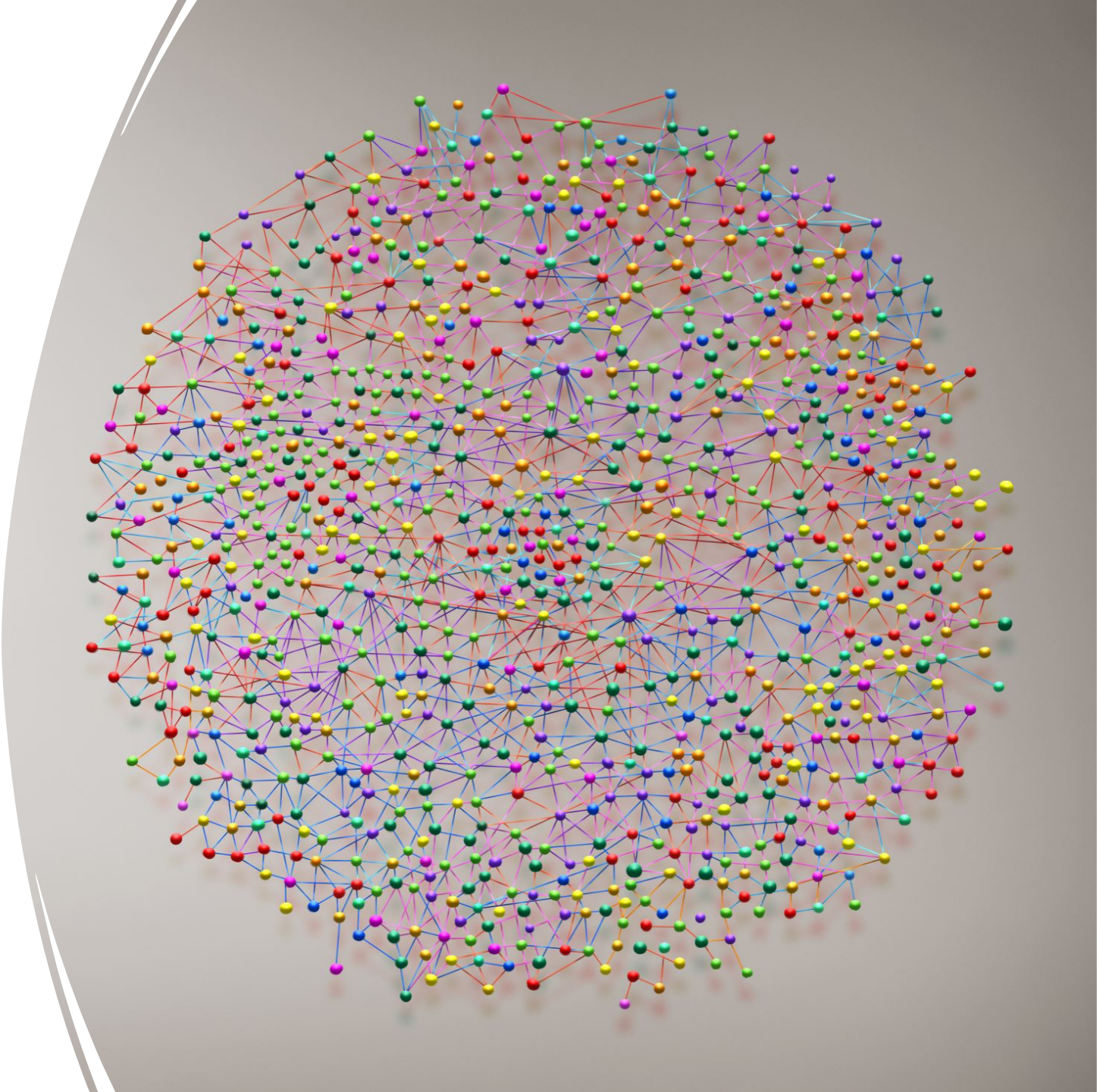
PREDICTING PORTUGESE SECONDARY SCHOOL STUDENT PERFORMANCE

https://github.com/annieptba/data1030_project_-portugese-secondary-student-performance.git

ANNIE PHAN (Banner ID: B01309278)



INTRODUCTION

Problem:

Regression: predict students' final scores (continuous, integer values from 0 to 20) using first and second period grades and other social, economic, and education factors

Classification: classify students' final performance (categorical values 'good', 'fair', 'poor') using first and second period grades and other social, economic, and education factors

Data overview and source

https://archive.ics.uci.edu/ml/datasets/student+performance

1,044 instances (students) including 395 Mathematics class students and 649 Portuguese language class students

33 features

Implications:

What type of courses can be offered to attract more students? Is it possible to predict student performance?

What are the factors that affect student achievement?

My model explorse these similar questions but look further into demographic factors such as family support, romantic relationships, alcohol consumption, and internet access.

Original research:

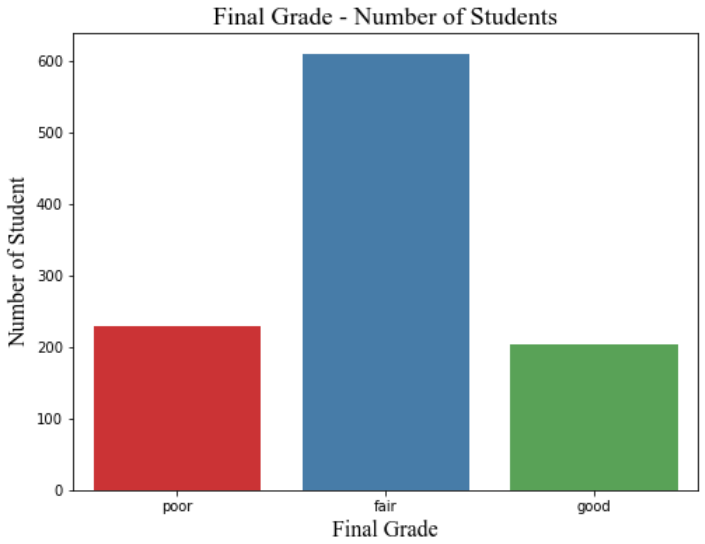
Project: "Using Data Mining To Predict Secondary School Student Performance" by Cortez and Silva in 2008

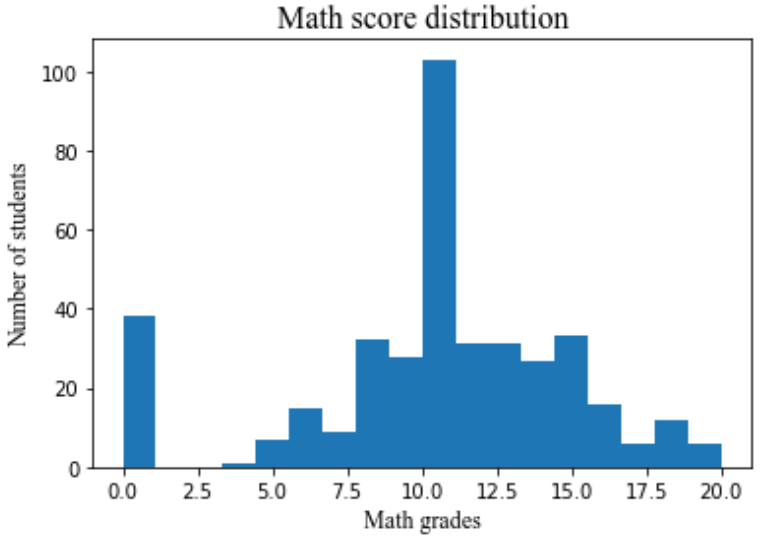
Method: binary classification (pass/fail), classification with 5 levels (I very good → V insufficient), regression, with a numeric output that ranges between (0%) and (100%)

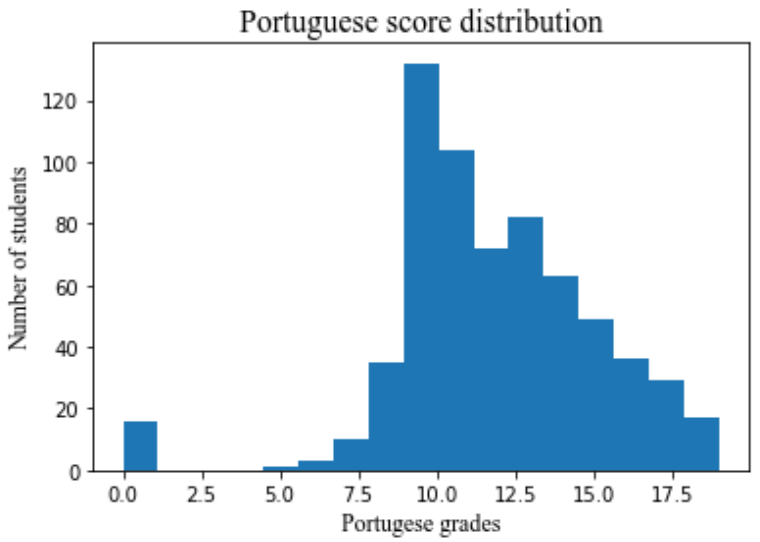
Result: students' final grades can be predicted by the first and/or second school period grades and also other relevant features

EDA - HISTOGRAMS AND BAR CHARTS

- Show us what you think was the most interesting, unexpected, or important insight that you gained during EDA. Present at least three different informative visualization types.
- Around 60% of the students have a fair performance, but 20% of them have a poor and 20% have good performance. This is a very high % of students have a poor performance and fit with the idea that many Portuguese students were underperforming.
- It seems like more students have a higher Portuguese score than Math scores. This poses questions about Math vs Portuguese teaching level of Portuguese schools.

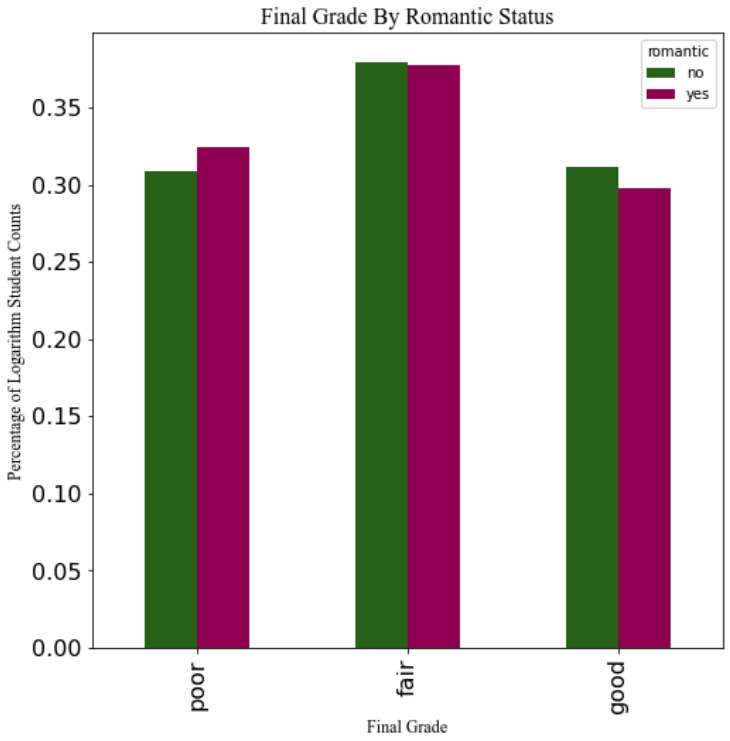




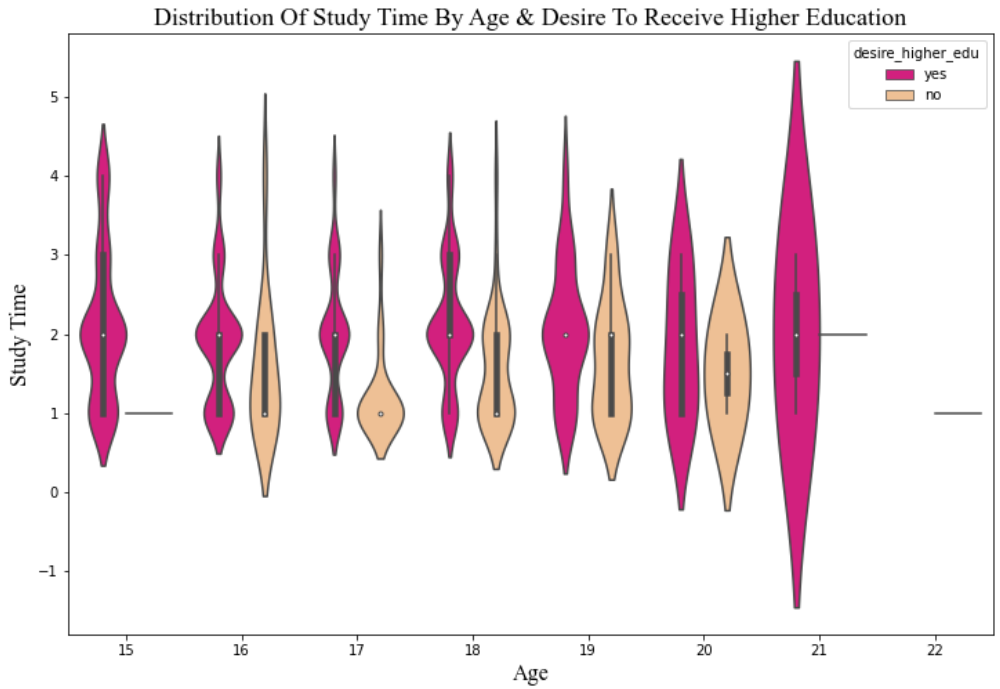


EDA - BAR & VIOLIN PLOTS

- more students who don't have a relationship have better final grade performance and vice versa, though the disparity isn't significant.
- signals that students should not be in a relationship

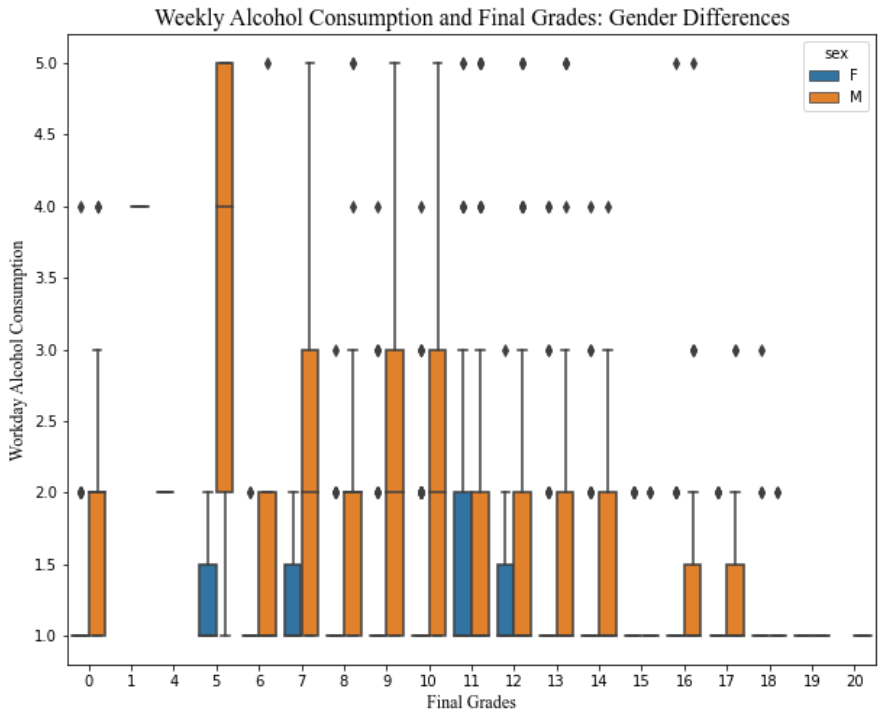


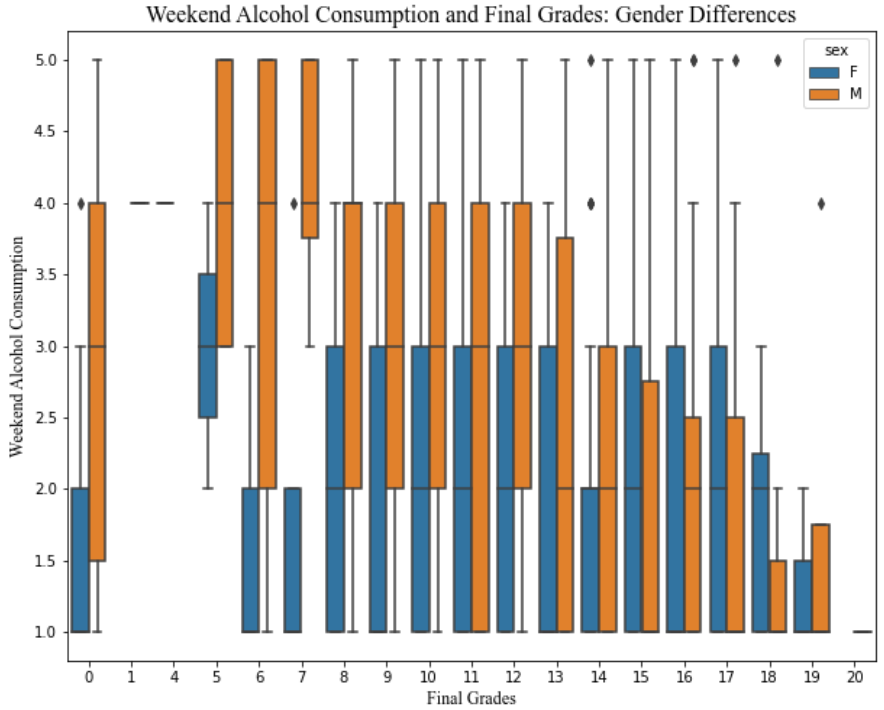
- positive relationship between study time and desire to achieve higher education.
- the older the students get, the more they want to achieve higher education
- students who don't want to achieve higher education quite close to those who want to



EDA - BOXPLOTS

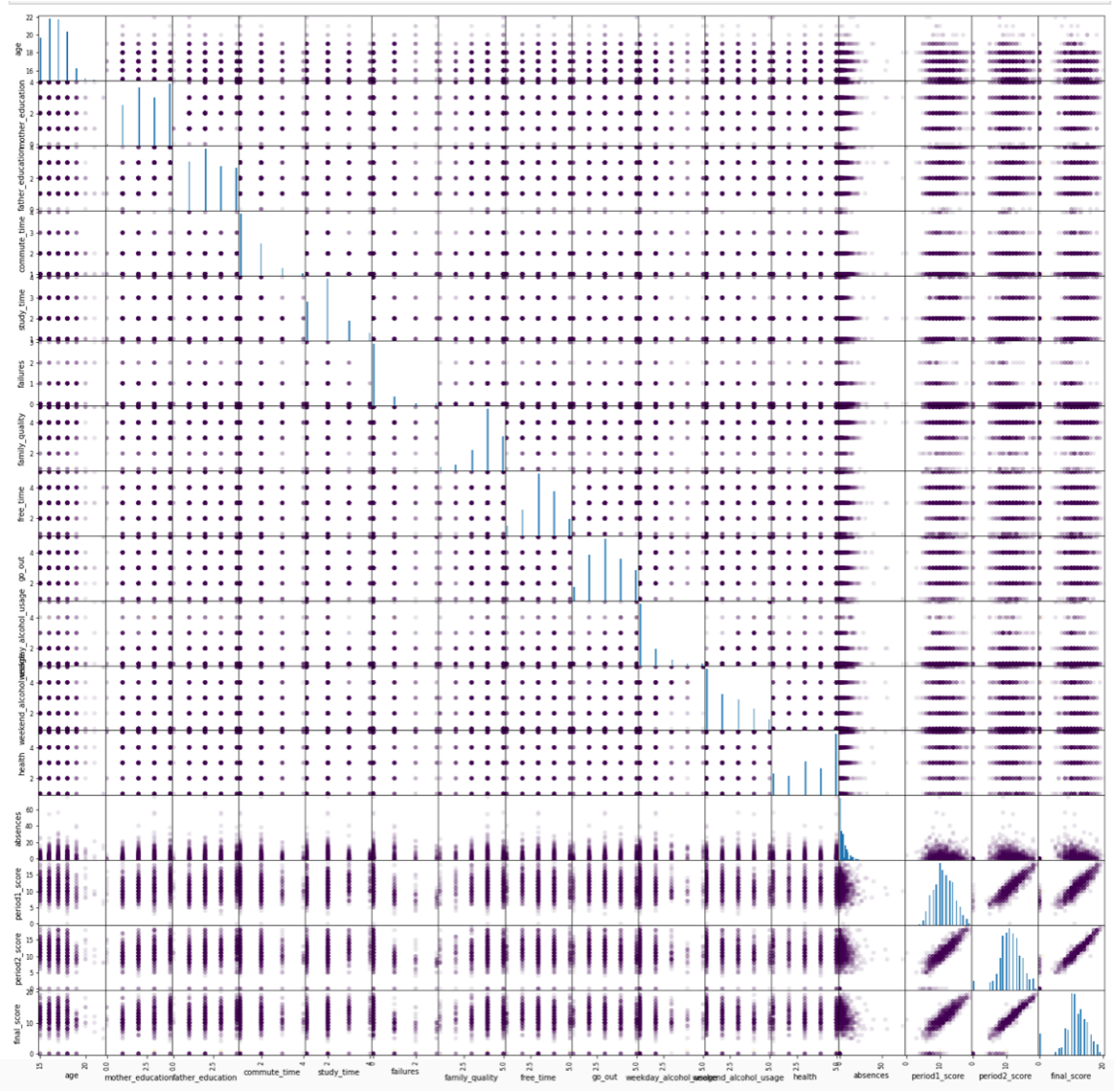
- demonstrate final grades across measures of distributions of students' weekend and weekday alcohol consumption, classified by gender.
- students who have less alcohol consumption tend to get higher grades and vice versa
- more male students consume alcohol than female students
- drink significantly more on the weekend than weekday
- some intervention and educational efforts should be executed to teach students about appropriate drink habits





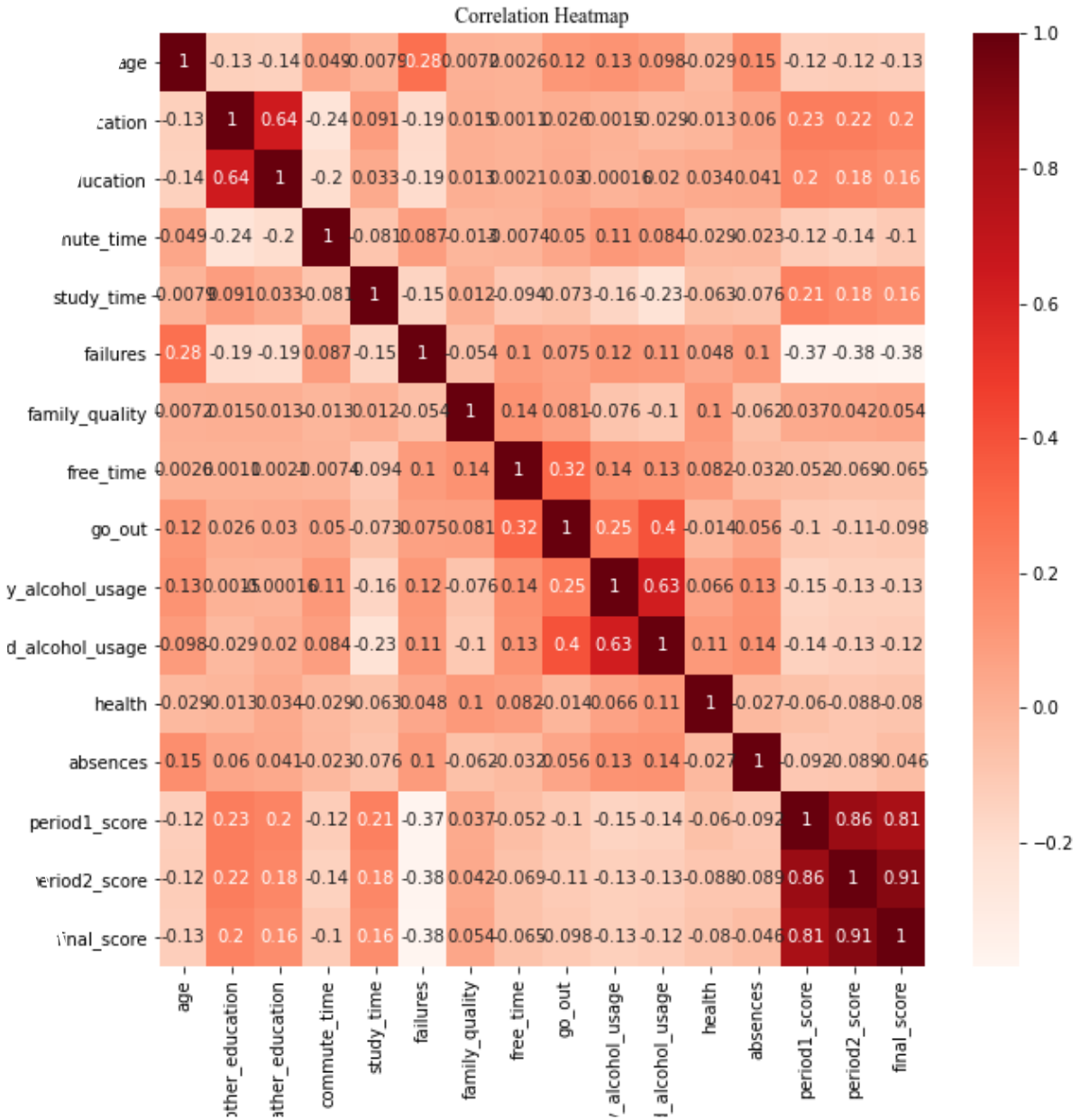
EDA - SCATTER MATRIX

- a strong positive relationship between the 1st and 2nd period scores and final scores, signaling that score prediction can be quite accurate.
- relationships between final scores and other variables aren't s obvious.



EDA - CORRELATION HEATMAP

- a strong positive relationship between the 1st and 2nd period scores and final scores, signaling that score prediction can be quite accurate
- final scores also seem to have quite strong positive relationships with the mother and father's education and study time
- other positive but weaker relationships between finals cores and other variables include family quality and free time
- a negative relationship between final age and scores, which can be because the materials get harder
- a weak negative relationship between health and final scores, raising concerns about mental health.



```
# merge datasets
df = pd.concat([mat,por])
```

```
# rename column labels
df.columns = ['school','sex','age','address','family_size','parents_status','mother_education','father_education',
              'mother_job','father_job','reason','guardian','commute_time','study_time','failures','school_support',
             'family_support','paid_classes','activities','nursery','desire_higher_edu','internet','romantic','family_quality',
             'free_time','go_out','weekday_alcohol_usage','weekend_alcohol_usage','health','absences','period1_score','period2_score','final_score
print(df.columns)
```

```
# convert final_score to categorical variable # Good:15~20 Fair:10~14 Poor:0~9
df['final_grade'] = 'na'
df.loc[(df.final_score >= 15) & (df.final_score <= 20), 'final_grade'] = 'good'
df.loc[(df.final_score >= 10) & (df.final_score <= 14), 'final_grade'] = 'fair'
df.loc[(df.final_score >= 0) & (df.final_score <= 9), 'final_grade'] = 'poor'
df.head(5)
```

PREPROCESSING - MERGE AND CREATE NEW VARIABLE

- Merge two datasets
- Create 'final_grade" column
  - good' performance if score from 15 to 20
  - 'fair' performance if score from 10 to 14
  - 'poor' performance if score below 9

PREPROCESSING - REGRESSION

- Target variable: final score
- Basic split: 60% of the data in train, 40% validation, and 40% set because the dataset is IID.
- 626 points in the train set, 209 in the validation and 209 in the test set

```
# collect all the encoders of the feature matrix for regression problems
preprocessor = ColumnTransformer(
    transformers=[
        ('onehot', OneHotEncoder(sparse=False,handle_unknown='ignore'), onehot_ftrs),
         ('minmax', MinMaxScaler(), minmax_ftrs)],
            remainder='passthrough')

clf = Pipeline(steps=[('preprocessor', preprocessor)]) # for now we only preprocess
                                                       # later on we will add other steps here
X_reg_train_prep = clf.fit_transform(X_reg_train)
X_reg_val_prep = clf.transform(X_reg_val)
X_reg_test_prep = clf.transform(X_reg_test)

print("Feature variable for regression training set", X_reg_train_prep.shape)
print("Feature variable for regression validation set",X_reg_val_prep.shape)
print("Feature variable for regression test set", X_reg_test_prep.shape)
```

```
Feature variable for regression training set (626, 67)
Feature variable for regression validation set (209, 67)
Feature variable for regression test set (209, 67)
```

PREPROCESSING - CLASSIFICATION

- Target variable: final_grade (, drop the "final_score" column
- K-fold split to shuffle the data , ensure enough randomness and reduce errors because the dataset is small. 20% of the data in test and split the other set into 5.
- 668 points in the train set, 167 in validation and 209 points in the test set.
- Apply a Label Encoder to the target variable 'final_grade'

```
# collect all the encoders for classification problem
#Feature Matrix
preprocessor = ColumnTransformer(
    transformers=[
        ('onehot', OneHotEncoder(sparse=False,handle_unknown='ignore'), onehot_ftrs),
         ('minmax', MinMaxScaler(), minmax_ftrs)],
            remainder='passthrough')

clf = Pipeline(steps=[('preprocessor', preprocessor)]) # for now we only preprocess
                                                       # later on we will add other steps here

X_class_train_prep = clf.fit_transform(X_class_train)
X_class_val_prep = clf.transform(X_class_val)
X_class_test_prep = clf.transform(X_class_test)

print("Feature variable for classifcation training set", X_class_train_prep.shape)
print("Feature variable for classifcation validation set",X_class_val_prep.shape)
print("Feature variable for classifcation test set", X_class_test_prep.shape)

#apply label encoder to target variable
le = LabelEncoder()
y_class_train_prep = le.fit_transform(y_class_train)
print('Target variable for classifcation training prep set:', y_class_train.shape, y_class_train.head(3))
y_class_val_prep = le.transform(y_class_val)
print('Target variable for classifcation validation prep set:', y_class_val.shape, y_class_val.head(3))
y_class_test_prep = le.transform(y_class_test)
print('Target variable for classifcation validation prep set:', y_class_test.shape, y_class_test.head(3))
```

```
Feature variable for classifcation training set (668, 66)
Feature variable for classifcation validation set (167, 66)
Feature variable for classifcation test set (209, 66)
Target variable for classifcation training prep set: (668,) 136    fair
350    poor
60     fair
Name: final_grade, dtype: object
Target variable for classifcation validation prep set: (167,) 256    fair
535    fair
397    fair
Name: final_grade, dtype: object
Target variable for classifcation validation prep set: (209,) 576    fair
280    poor
141    fair
Name: final_grade, dtype: object
```

PREPROCESSING - ENCODERS FOR FEATURE MATRIX

- only apply encoders to the columns that haven't been processed yet
- apply the MixMaxEncoder to the 'age' and 'absences' columns because they both have bounded features ('age' ranges from 15 to 22 and 'absences' ranges from 0 to 93)
- apply the One Hot encoders to the categorical columns that have 'yes'-'no' features because they haven't been encoded yet (the 'school', 'sex', 'age'', address', 'family_size'', parents_status', 'mother_job', 'father_job'', reason',guardian', 'school_support', 'family_support', 'paid_classes'', activities', 'nursery', 'desire_higher_edu', 'internet', 'romantic' columns).

REGRESSION

```
## Apply MixMax Scaler to Continous Columns with Bounds - 'Age' and 'Absences' column
#collect all continous bounded features that have not been processed
minmax_ftrs = ['age','absences']
#initialize the encoder
scaler = MinMaxScaler()
#fit X training
scaler.fit(X_reg_train[minmax_ftrs])
#transform X train
scaler_reg_train = scaler.transform(X_reg_train[minmax_ftrs])
print("Train set", scaler_reg_train.shape)
#transform X val
scaler_reg_val = scaler.transform(X_reg_val[minmax_ftrs])
print("Validation set",scaler_reg_val.shape)
#transform X set
scaler_reg_test = scaler.transform(X_reg_test[minmax_ftrs])
print("Test set",scaler_reg_test.shape)
```

```
## Apply One Hot Enconder to categorical columns
#collect all categorical features that have not been processed
onehot_ftrs = ['school','sex','age','address','family_size','parents_status',
               'mother_job','father_job','reason','guardian','school_support',
               'family_support','paid_classes','activities','nursery','desire_higher_edu','internet','romanti
# initialize the encoder
enc = OneHotEncoder(sparse=False,handle_unknown='ignore') # by default, OneHotEncoder returns a sparse matrix
# fit the training data
enc.fit(X_class_train[onehot_ftrs])
print('Feature names:',enc.get_feature_names(onehot_ftrs))
# transform X_train
onehot_class_train = enc.transform(X_class_train[onehot_ftrs])
print('Transformed train features:', onehot_class_train)
# transform X_val
onehot_class_val = enc.transform(X_class_val[onehot_ftrs])
print('Transformed val features:', onehot_class_val)
# transform X_test
onehot_class_test = enc.transform(X_class_test[onehot_ftrs])
print('Transformed test features:', onehot_class_test)
```

CLASSIFICATION

```
## Apply MixMax Scaler to Continous Columns with Bounds - 'Age' and 'Absences' column
#collect all continous bounded features that have not been processed
minmax_ftrs = ['age','absences']
#initialize the encoder
scaler = MinMaxScaler()
# fit the training data
scaler.fit(X_class_train[minmax_ftrs])
#transform X train
scaler_class_train = scaler.transform(X_class_train[minmax_ftrs])
print("Train set", scaler_class_train.shape)
#transform X val
scaler_class_val = scaler.transform(X_class_val[minmax_ftrs])
print("Validation set",scaler_class_val.shape)
#transform X set
scaler_class_test = scaler.transform(X_class_test[minmax_ftrs])
print("Test set",scaler_class_test.shape)
```

```
## Apply One Hot Enconder to categorical columns
#collect all categorical features that have not been processed
onehot_ftrs = ['school','sex','age','address','family_size','parents_status',
               'mother_job','father_job','reason','guardian','school_support',
               'family_support','paid_classes','activities','nursery','desire_higher_edu','internet','romanti
# initialize the encoder
enc = OneHotEncoder(sparse=False,handle_unknown='ignore') # by default, OneHotEncoder returns a sparse matrix
# fit the training data
enc.fit(X_reg_train[onehot_ftrs])
print('Feature names:',enc.get_feature_names(onehot_ftrs))
# transform X_train
onehot_reg_train = enc.transform(X_reg_train[onehot_ftrs])
print('Transformed train features:', onehot_reg_train)
# transform X_val
onehot_reg_val = enc.transform(X_reg_val[onehot_ftrs])
print('Transformed val features:', onehot_reg_val)
# transform X_test
onehot_reg_test = enc.transform(X_reg_test[onehot_ftrs])
print('Transformed test features:', onehot_reg_test)
```