CSS 429 Data Analysis Systems

Students Alcohol Consumption Data Analysis

Team 15

Team members:

Bekmaganbetov Zhanbolat

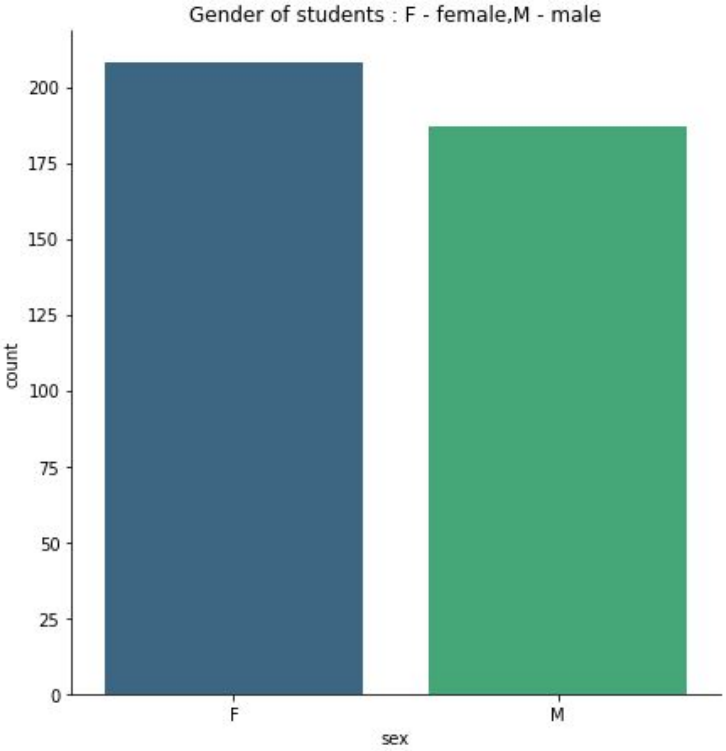
Kuatkyzy Gulnaz

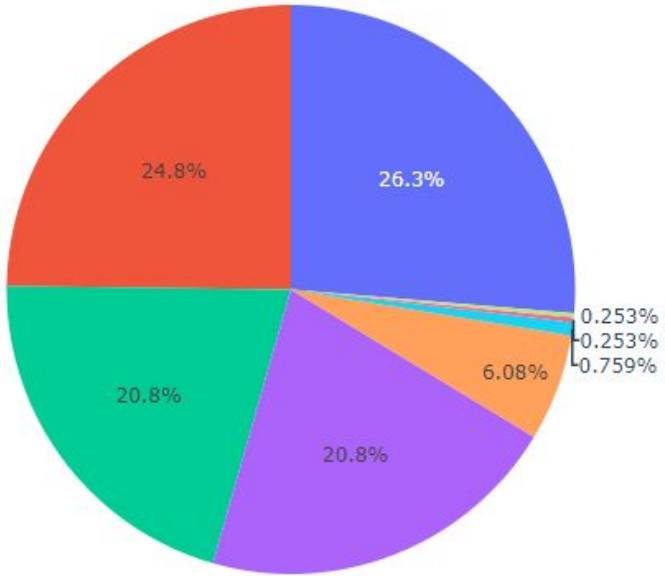
Phazyl Kaisar

Sissimbayev Dilshat

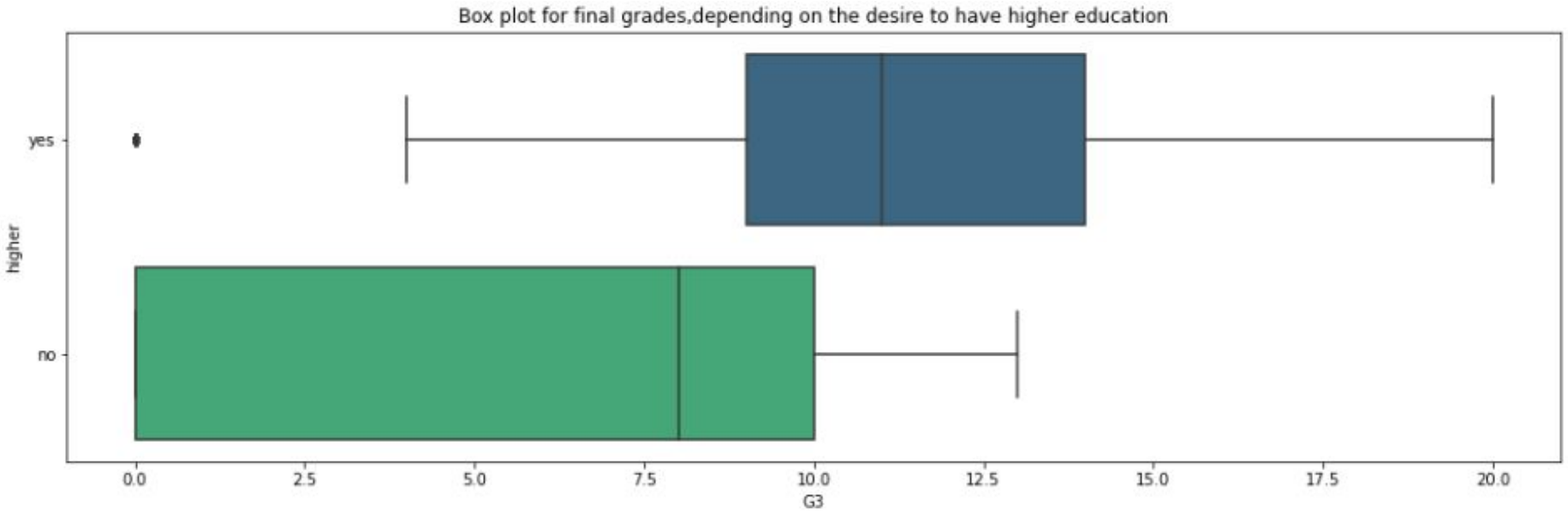
Tolegen Assel

Data understanding

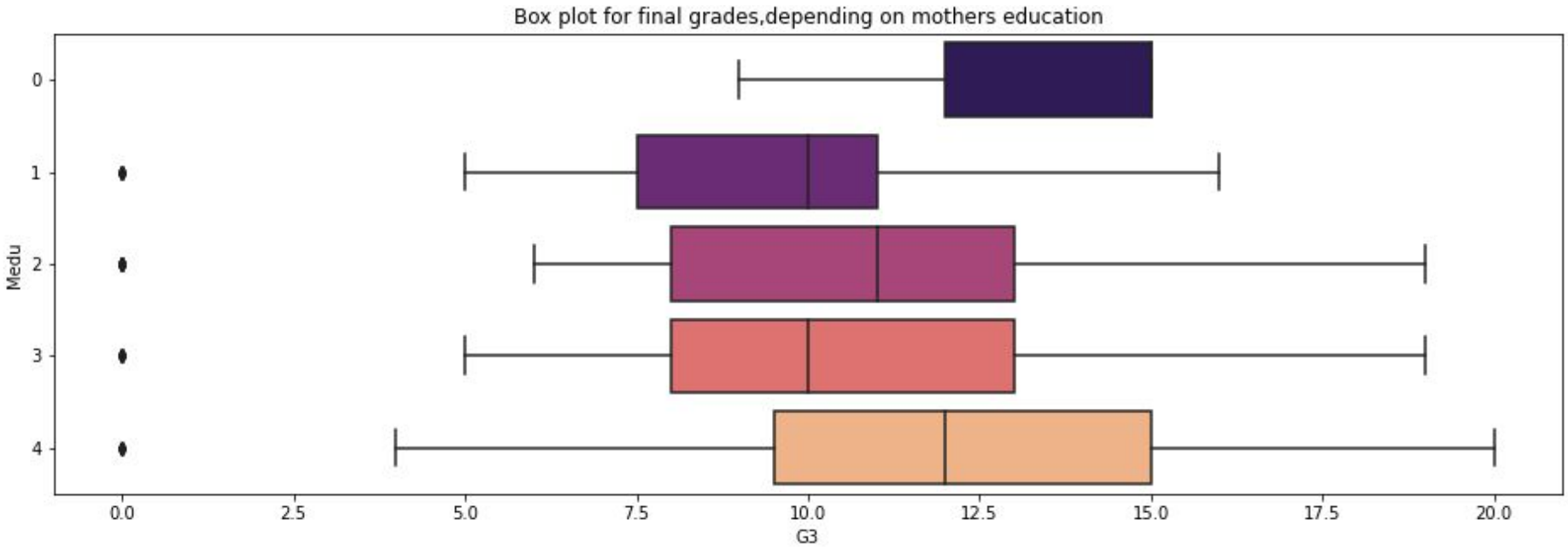




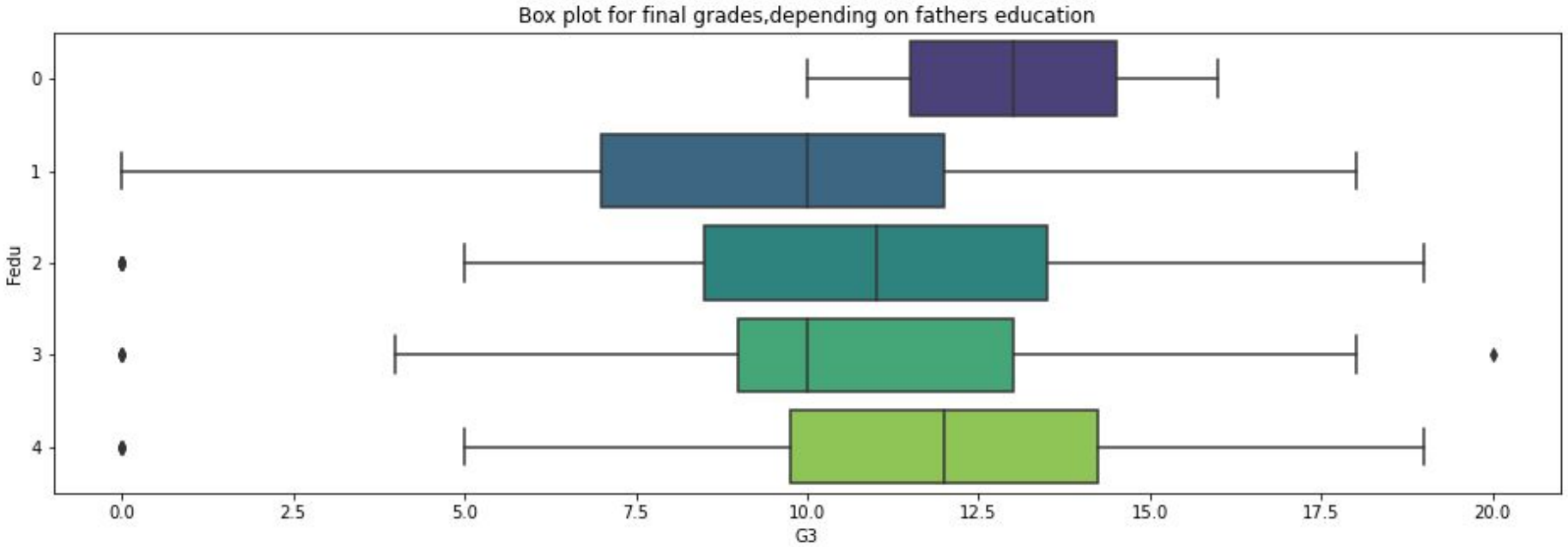
Does the desire to get higher education influence the final grades?



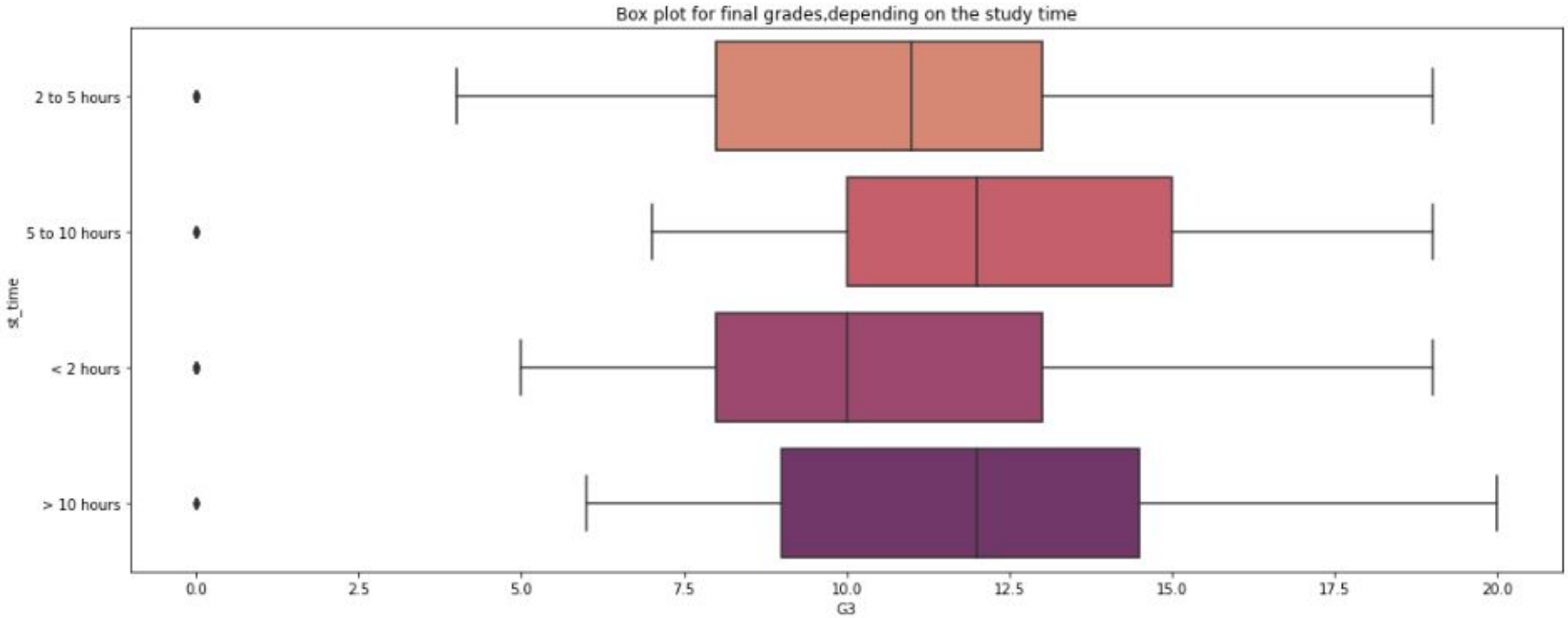
Final grades depending on mother's education



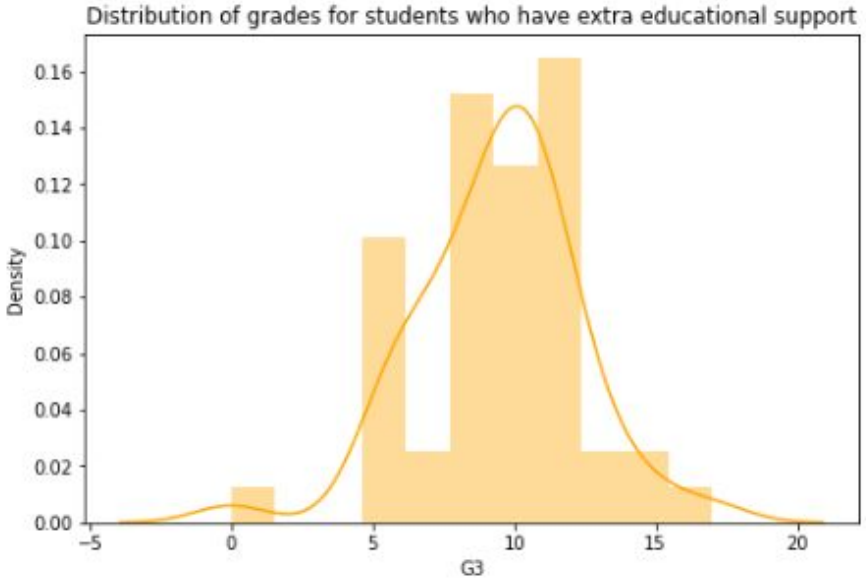
Final grades depending on father's education

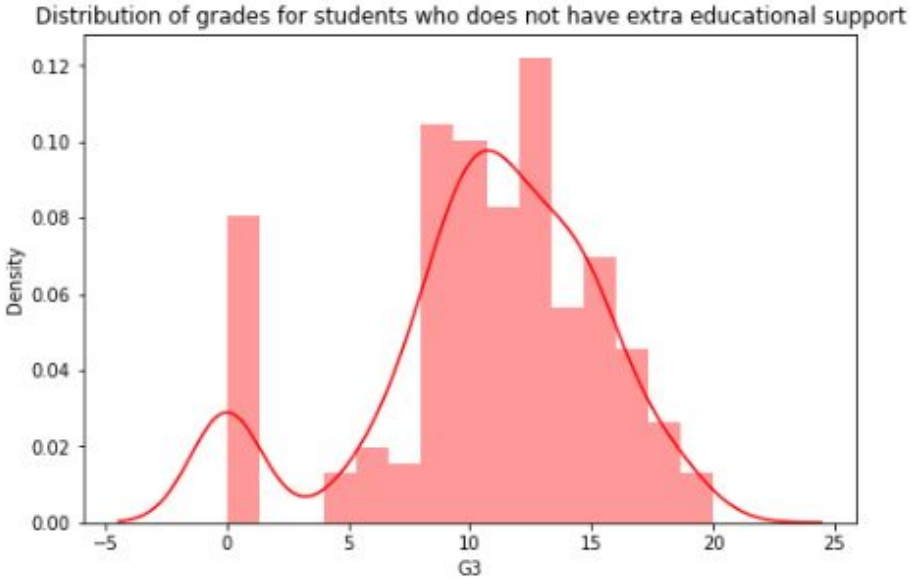


Final grades, depending on study time

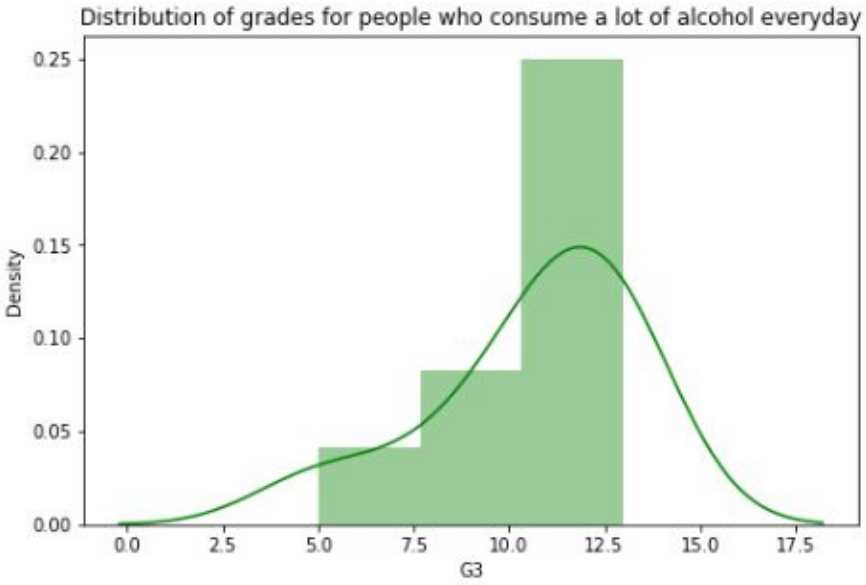


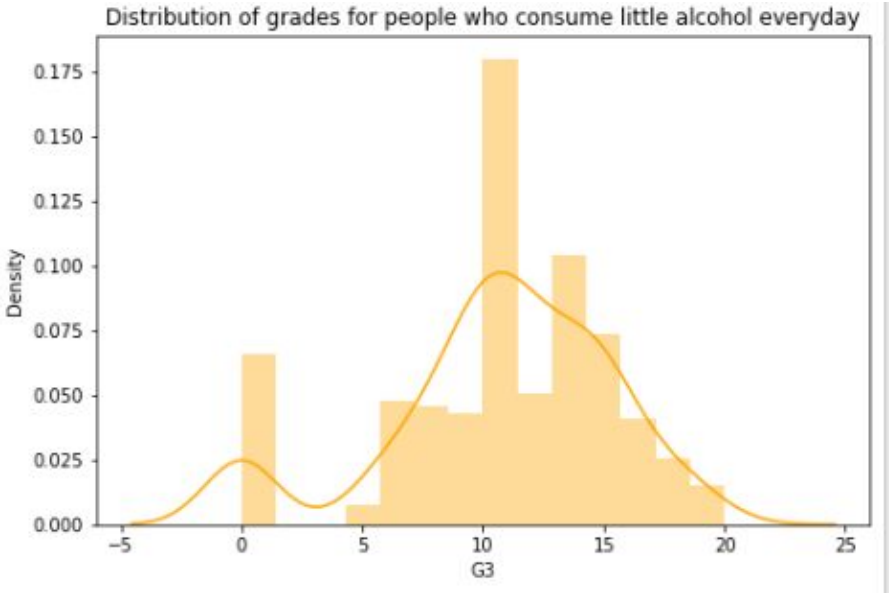
Final grades depending on extra educational support



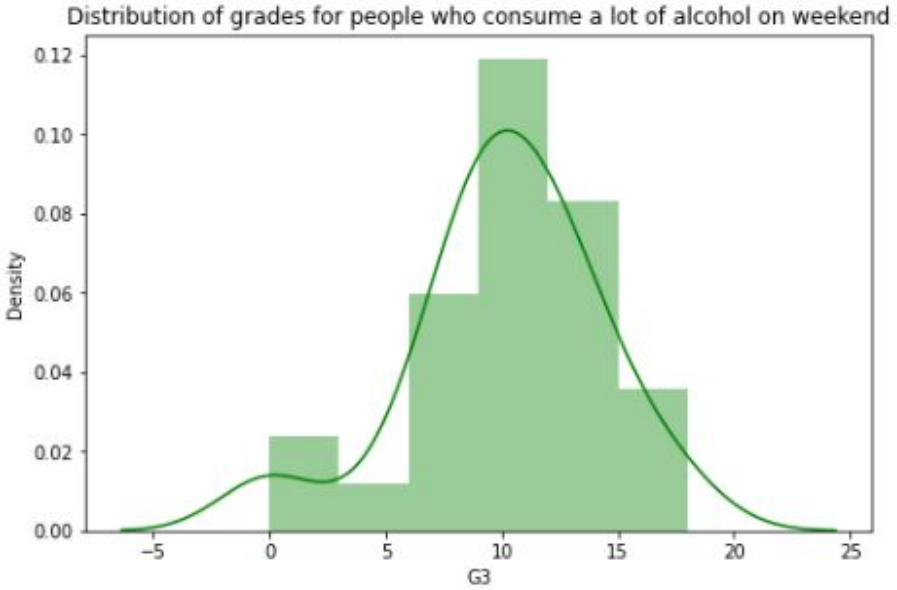


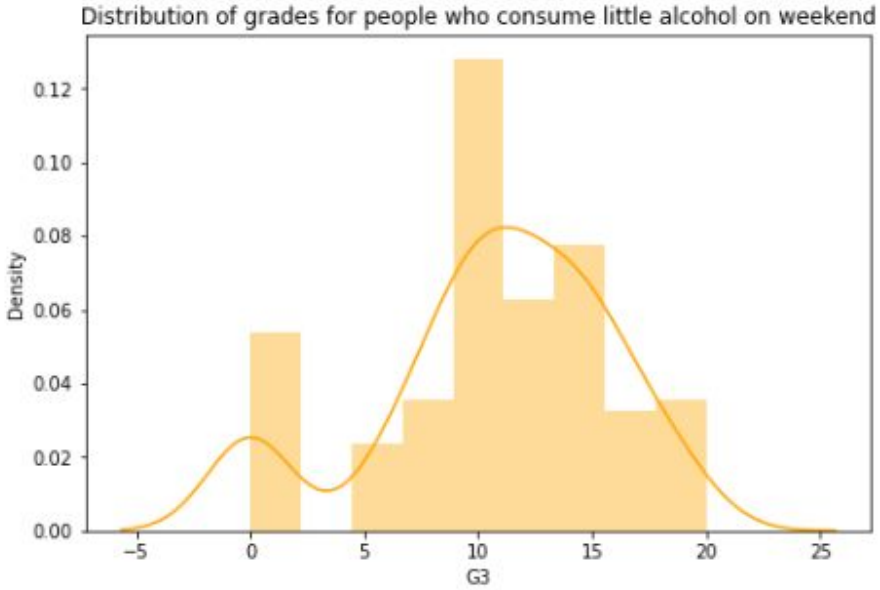
Daily alcohol consumption



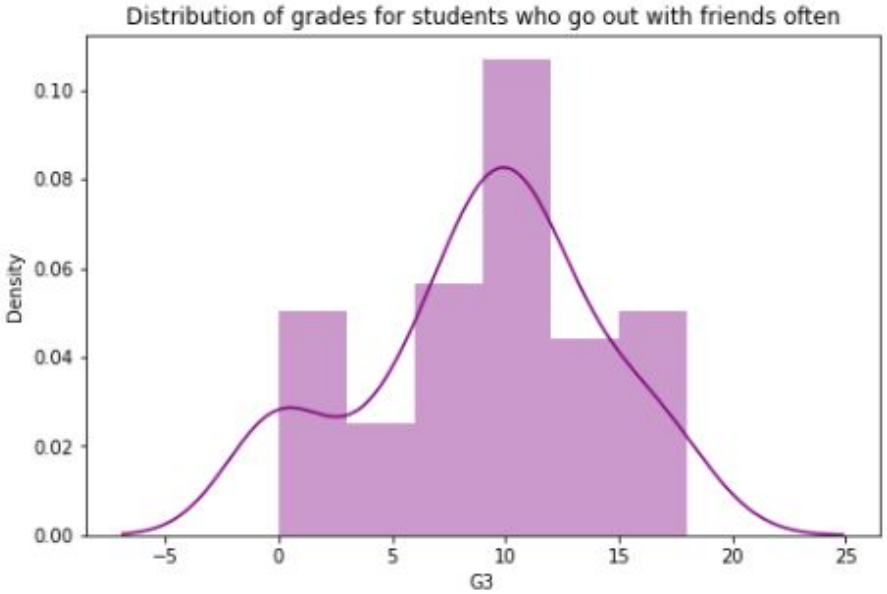


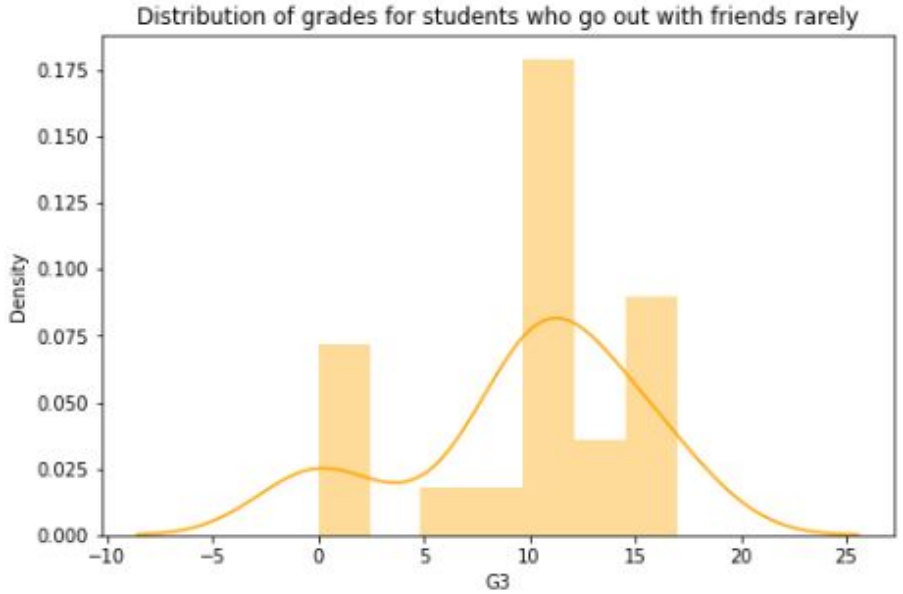
Weekend alcohol consumption



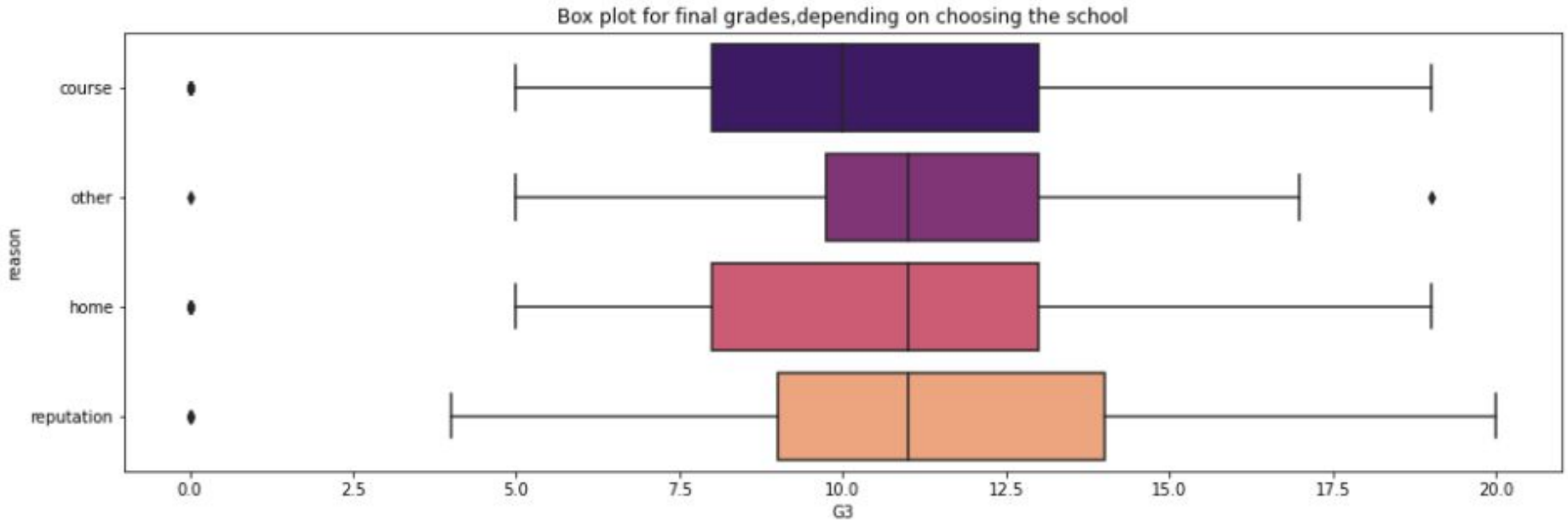


Going out with friends or staying at home?

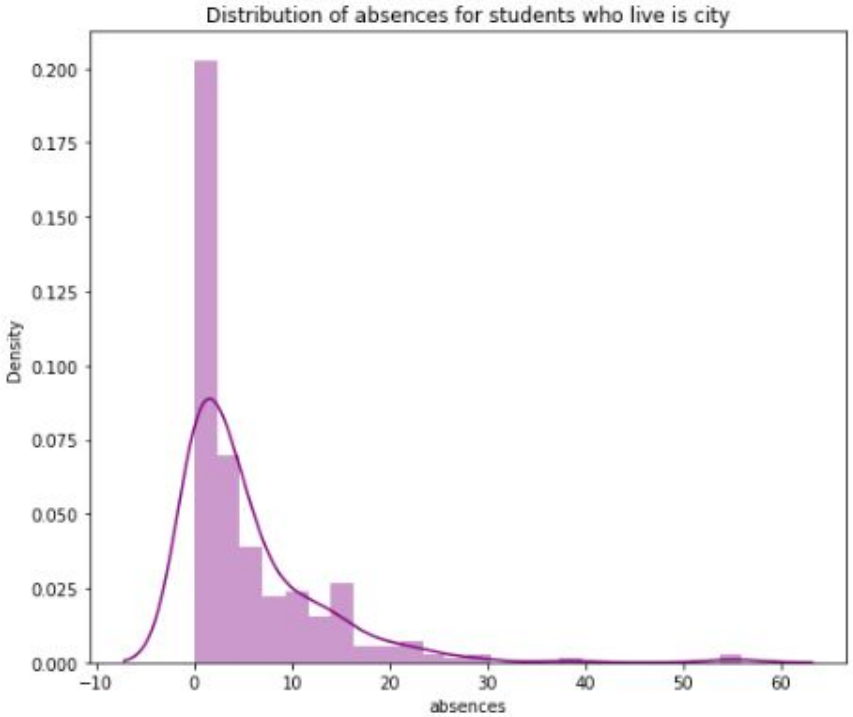


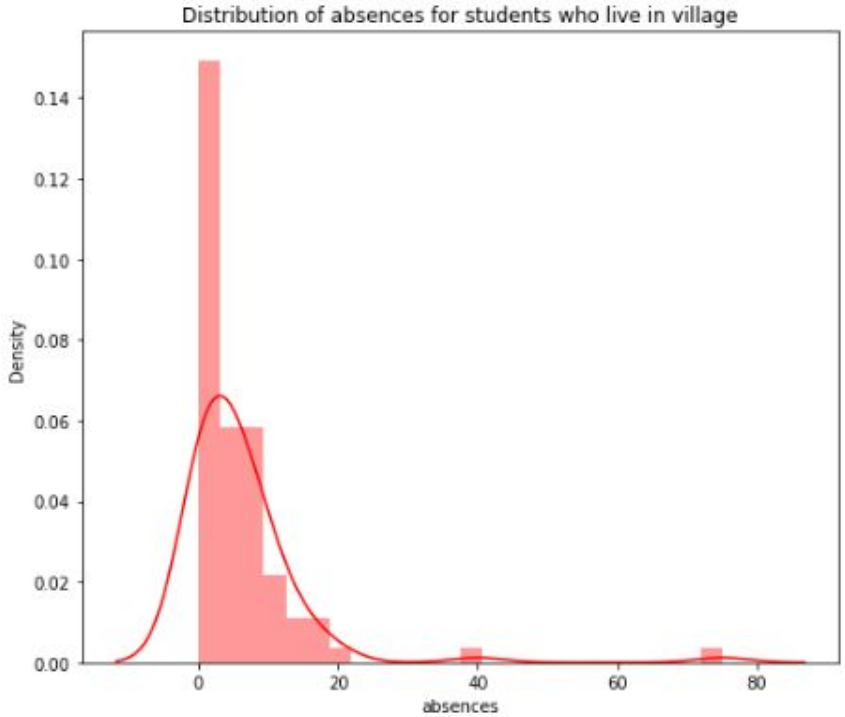


Reason to choose this school

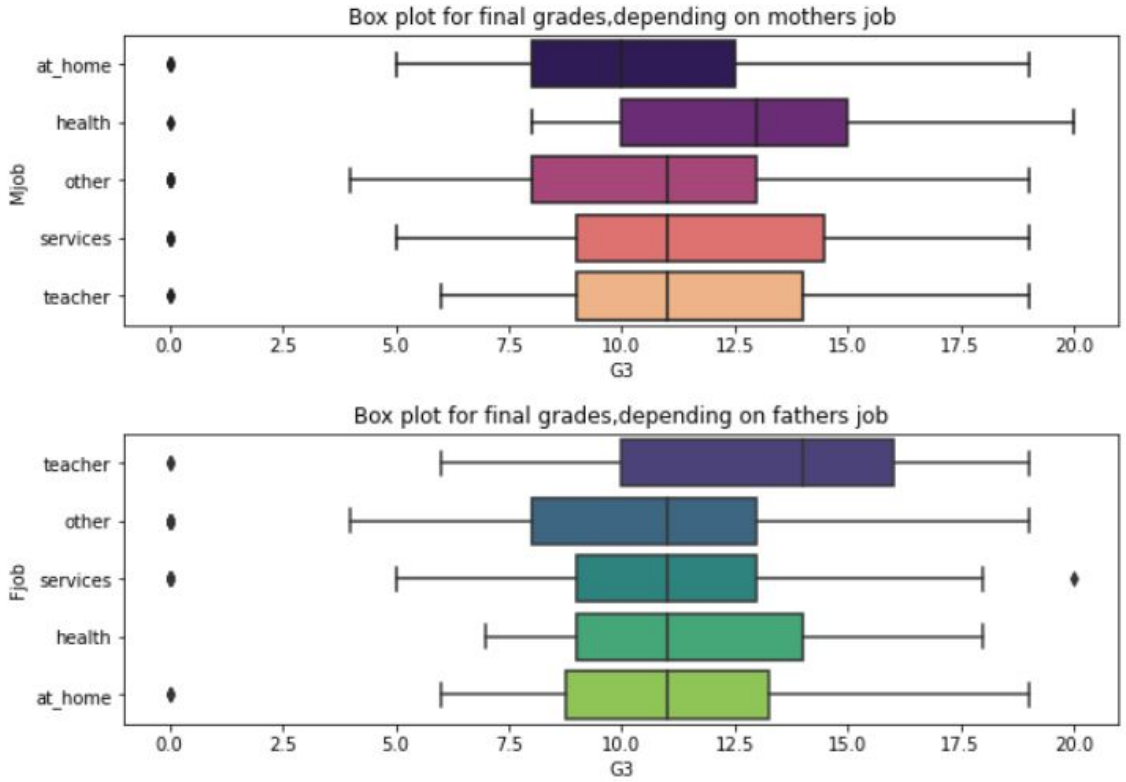


Distribution of absences

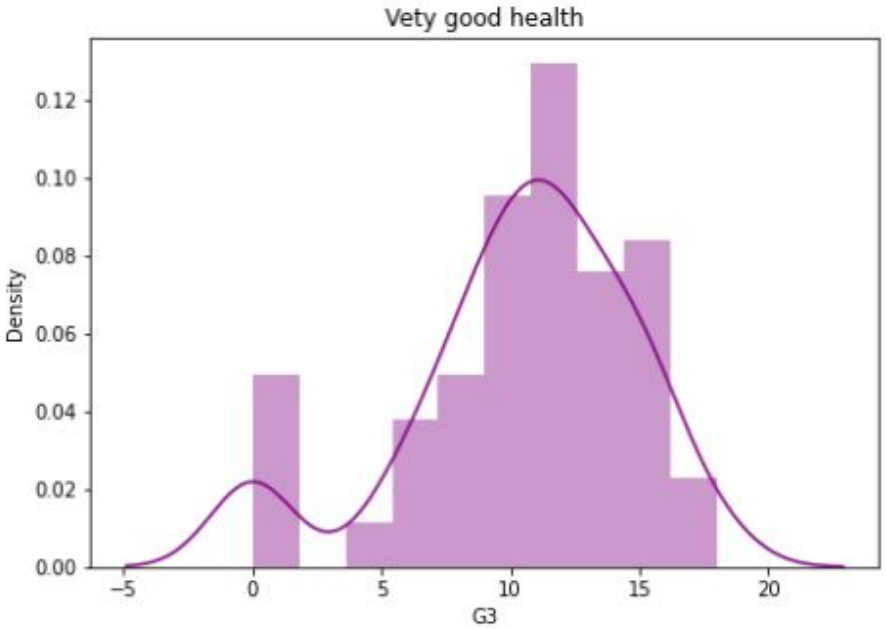


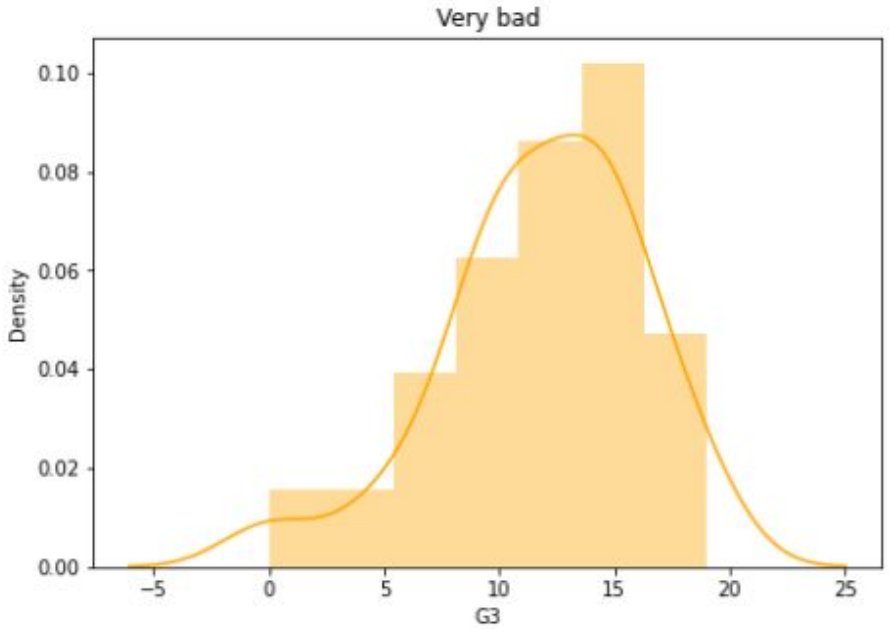


Final grades, depending on parents profession

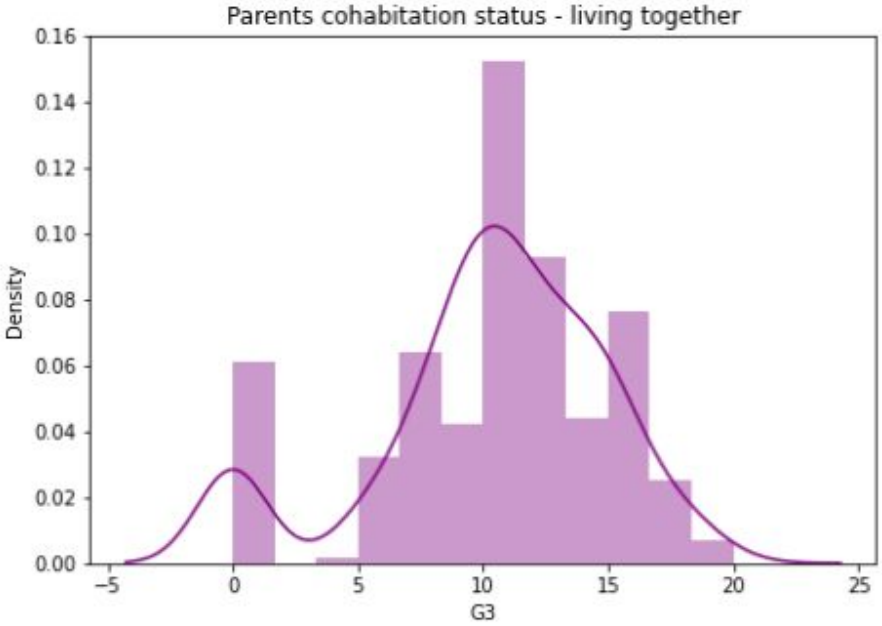


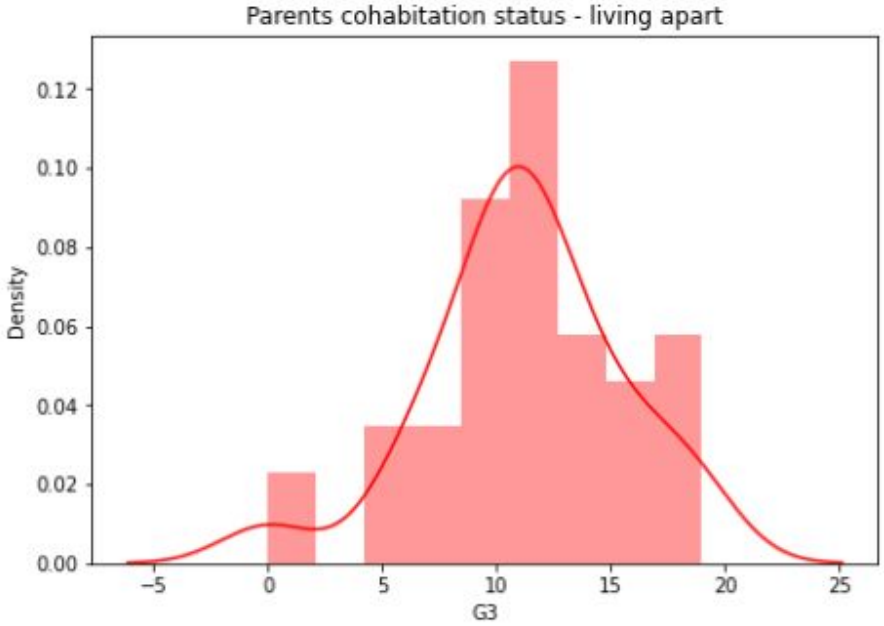
Final grades depending on health condition



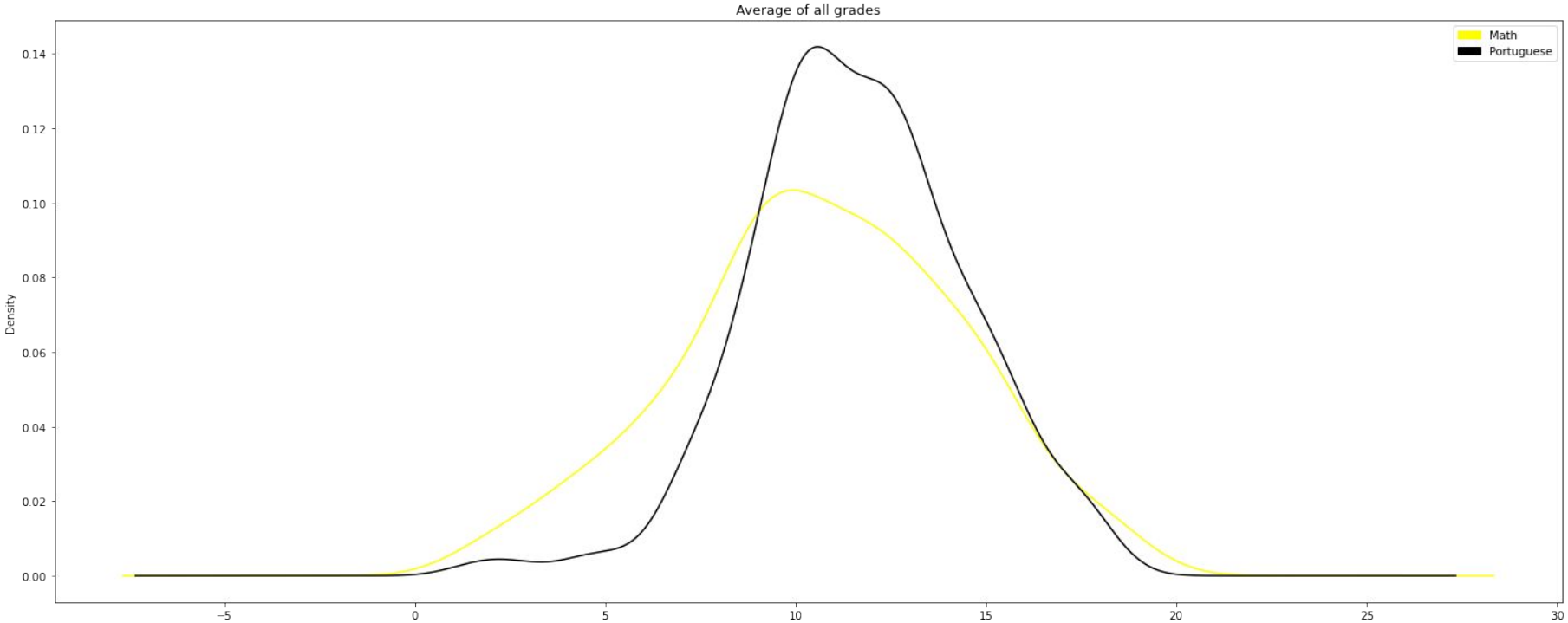


Final grades, depending on parents cohabitation

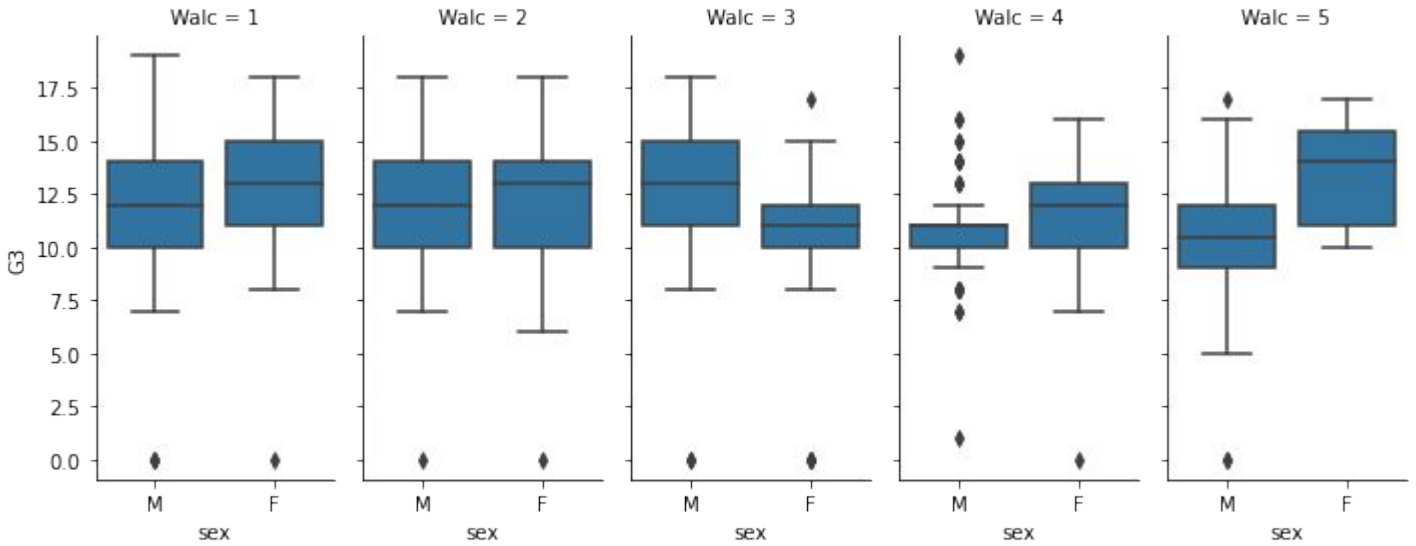




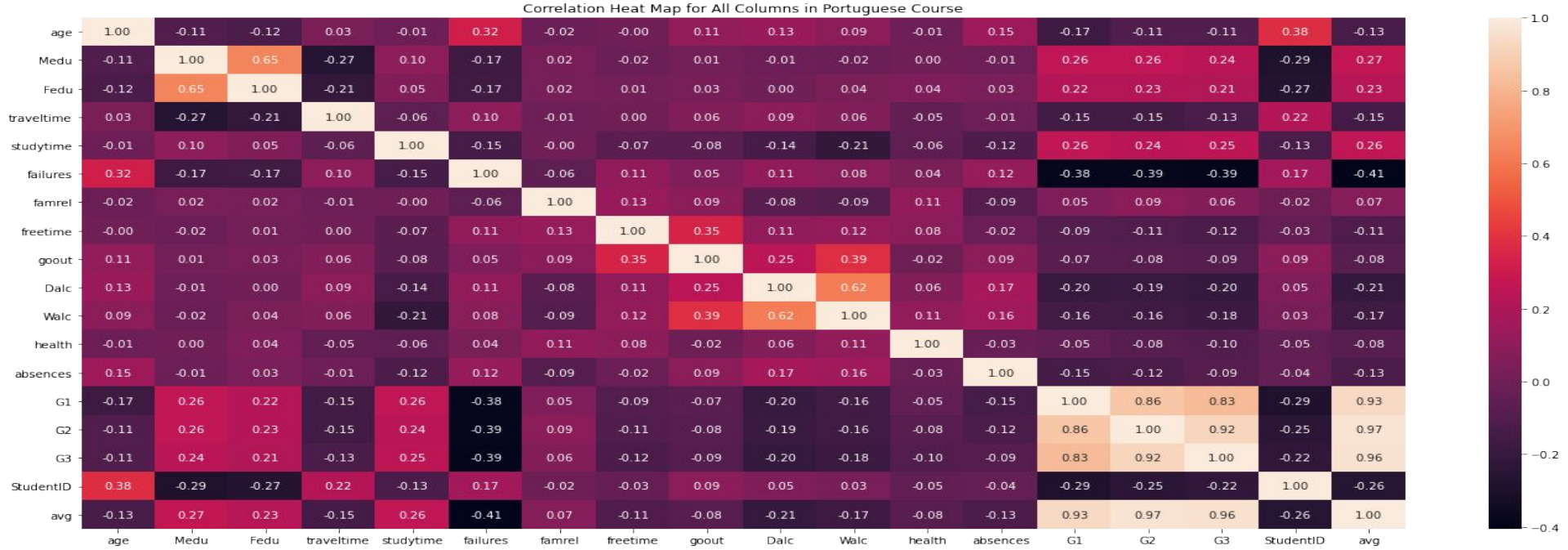
Average of all grades



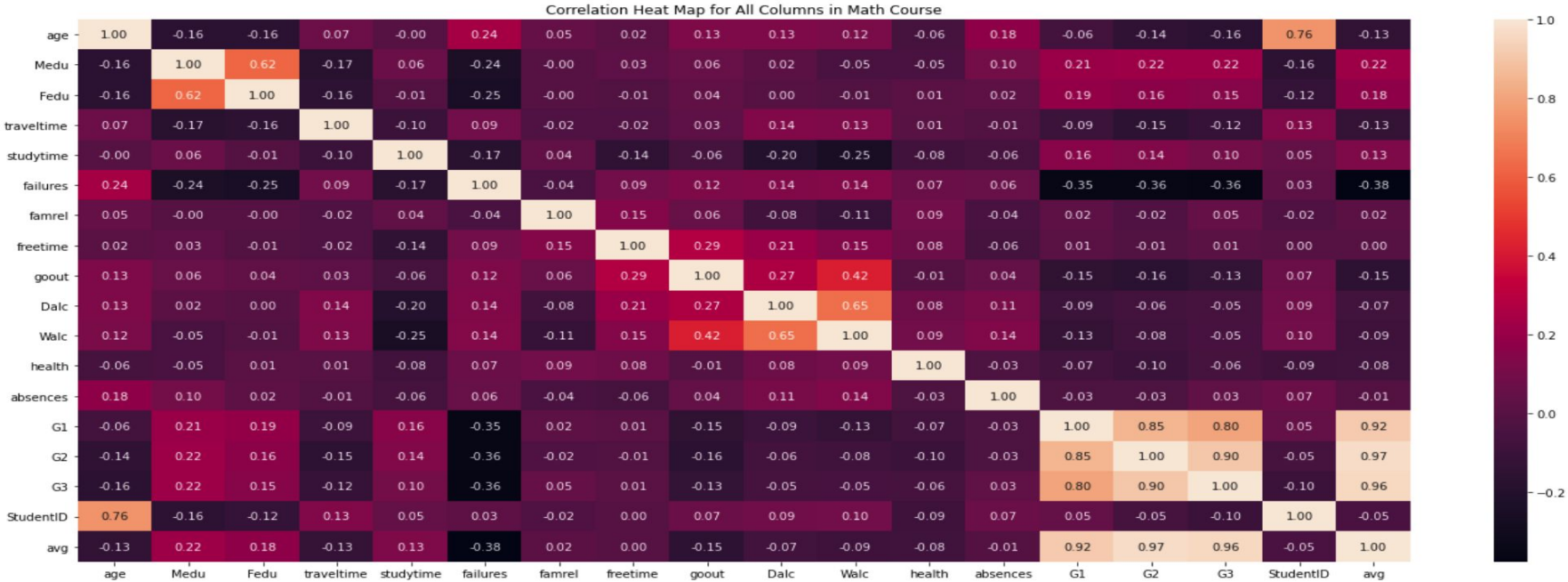
Box whisker for sex vs walc and grades



Heat Map for All Columns in Portuguese Course



Heat Map for All Columns in Mathematical Course



Modeling

We were discussing and choosed 4 data mining algorithms and built prediction models for our purposes.

Modeling

In [65]:
```
# Getting dummies

#Data split
#Lasso Regression
lX = data_two.drop(['G3'],axis=1)
lX_train, lX_test, ly_train, ly_test = train_test_split(pd.get_dummies(lX), data_two.G3, test_size=0.3)
#Linear Regression
lnX_train, lnX_test, lny_train, lny_test = train_test_split(pd.get_dummies(lX), data_two.G3, test_size=0.3)
#Random Forest
rfX_train, rfX_test, rfy_train, rfy_test = train_test_split(pd.get_dummies(lX), data_two.G3, test_size=0.3)
#DecisionTree
dX = data_two.drop(['G3'],axis=1)
dXtrain, dX_test, dy_train, dy_test = train_test_split(pd.get_dummies(dX), data_two.G3, test_size=0.3)
```

Train and fitting

In [66]:
```
#G1 - first period grade
#G2 - second period grade
#G3 - third period grade

#KFolds
cv = RepeatedKFold(n_splits=10, n_repeats=3, random_state=1)
#Linear regression
linearRegression = LinearRegression()
linearRegression.fit(rfX_train,rfy_train)

#RandomForestClassifier
randomForest = RandomForestClassifier(criterion='gini', n_estimators=200,max_depth=3)
randomForest.fit(rfX_train, rfy_train)

#Lasso Regression
lassoRegression = Lasso(alpha=1.0,max_iter=3000,random_state=30)
lassoRegression.fit(lX_train,ly_train)

#DecisionRegressor
decisionTree = DecisionTreeRegressor()
decisionTree.fit(dXtrain,dy_train)
```

Out[66]:
```
DecisionTreeRegressor()
```

Modeling

Score

In [68]:
```
# Mean scrore and Standart deviation
scores = cross_val_score(randomForest, rfX_test, rfy_test, cv=10)
print("Accuracy RandForest: %0.2f (+/- %0.2f)" % (scores.mean(), scores.std() * 2))
scores = cross_val_score(linearRegression, lnX_test, lny_test, cv=10)
print("Accuracy LinearReg: %0.2f (+/- %0.2f)" % (scores.mean(), scores.std() * 2))
scores = cross_val_score(lassoRegression, lX_test, ly_test, cv=10)
print("Accuracy Lasso: %0.2f (+/- %0.2f)" % (scores.mean(), scores.std() * 2))
scores = cross_val_score(decisionTree, dX_test, dy_test, cv=10)
print("Accuracy DecisionTree: %0.2f (+/- %0.2f)" % (scores.mean(), scores.std() * 2))

#print("MEAN AVERAGE ERROR: ", mean_absolute_error(y_test, y_pred))
```

```
/opt/anaconda3/lib/python3.8/site-packages/sklearn/model_selection/_split.py:670: UserWarning: The least populated
class in y has only 1 members, which is less than n_splits=10.
  warnings.warn(("The least populated class in y has only %d"
```

```
Accuracy RandForest: 0.35 (+/- 0.12)
Accuracy LinearReg: 0.81 (+/- 0.20)
Accuracy Lasso: 0.87 (+/- 0.19)
Accuracy DecisionTree: 0.52 (+/- 0.46)
```

In [69]:
```
randomForest.feature_importances_
```

Out[69]:
```
array([0.0105912 , 0.00118818, 0.01042924, 0.00447445, 0.00258399,
       0.00090509, 0.0127442 , 0.01914447, 0.01578392, 0.00821754,
       0.01335288, 0.00732683, 0.05847934, 0.00703071, 0.00695194,
       0.01883604, 0.00168761, 0.00362937, 0.00643978, 0.00870414,
       0.00950597, 0.01094184, 0.01203665, 0.01166532, 0.05311826,
       0.27623579, 0.37089449, 0.03710075])
```

In [70]:
```
#defining parameters for decision tree regressor
params= {
    'n_estimators':[1000],
    'max_depth':[20],
        }

reg_tree = RandomForestClassifier()
gs = GridSearchCV(estimator=reg_tree, param_grid=params, cv=5, n_jobs=-1) #validate model with his parameters
gs.fit(rfX_train, rfy_train) #fitting training set

reg_tree = gs.best_estimator_
print(reg_tree) #printing best estimator values

pred_tree = reg_tree.predict(rfX_test)
```

Modeling

In [69]:
```
randomForest.feature_importances_
```

Out[69]:
```
array([0.0105912 , 0.00118818, 0.01042924, 0.00447445, 0.00258399,
       0.00090509, 0.0127442 , 0.01914447, 0.01578392, 0.00821754,
       0.01335288, 0.00732683, 0.05847934, 0.00703071, 0.00695194,
       0.01883604, 0.00168761, 0.00362937, 0.00643978, 0.00870414,
       0.00950597, 0.01094184, 0.01203665, 0.01166532, 0.05311826,
       0.27623579, 0.37089449, 0.03710075])
```

In [70]:
```
#defining parameters for decision tree regressor
params= {
    'n_estimators':[1000],
    'max_depth':[20],
        }

reg_tree = RandomForestClassifier()
gs = GridSearchCV(estimator=reg_tree, param_grid=params, cv=5, n_jobs=-1) #validate model with his parameters
gs.fit(rfX_train, rfy_train) #fitting training set

reg_tree = gs.best_estimator_
print(reg_tree) #printing best estimator values

pred_tree = reg_tree.predict(rfX_test)

#printing scores
dt_score = r2_score(rfy_test, pred_tree)
dt_mae = mean_absolute_error(rfy_test, pred_tree)

#Printing Mean Absolute Error and Score
print('MAE: %.2f' %dt_mae)
print('Score: %.2f' %dt_score)

importances = reg_tree.feature_importances_
indices= np.argsort(importances)[::-1]
# summarize feature importance
for i,v in enumerate(importances):
    print("Id - %d. future_name %s(%.3f)" % (i + 1, rfX_test.columns.values[indices[i]], importances[indices[i]]))
```

```
/opt/anaconda3/lib/python3.8/site-packages/sklearn/model_selection/_split.py:670: UserWarning: The least populated
class in y has only 1 members, which is less than n_splits=5.
  warnings.warn(("The least populated class in y has only %d"
```

```
RandomForestClassifier(max_depth=20, n_estimators=1000)
MAE: 1.01
Score: 0.77
```