MACHINE LEARNING 2CS501

TERM ASSIGNMENT

ROLL NO. : 19BCE080,19BCE086

STUDENT PERFORMANCE ANALYSIS DATASET

Dataset available at: [http://archive.ics.uci.edu/ml/datasets/Student+Performance#](http://archive.ics.uci.edu/ml/datasets/Student+Performance)

**Abstract:**

Predict student performance in senior secondary education or high school.

**Dataset Information:**

Data Set Characteristics: Multivariate Associated Tasks: Classification, regression

Attribute Characteristics: Integers Number of instances, attributes : 649 , 33

In this dataset we are provided with student achievement in Portuguese schools. The data attributes include student grades, demographic, social and school related features and it was collected by using school reports and questionnaires.

We have been provided with two datasets which are mathematics and Portuguese respectively.

We have classified these marks of students into three different categories:

1)Good 2) Average 3) Poor

Based on these students’ performance in their final exam score we have analyzed and compared the relationship their marks share with various different features which have been provided in the dataset some of these are Romantic Status, Alcohol Consumption, Parents Education Level, Frequency of Going Out, Desire of Higher Education and Living Area.

Thus, we have done hypothesis testing, tuning of the parameter and implemented and chosen different machine learning algorithms and compared all of them to select the best one of these algorithms used.

**DESCRIBING THE CODE FOR STUDENT PERFORMANCE DATASET:**

#### Table of Contents

1. Import Packages
2. Load Dataset
3. Data Preparation
4. EDA
   * 4.1 Final Grade Distribution
   * 4.2 Correlation Heatmap
   * 4.3 Romantic Status
   * 4.4 Alcohol Consumption
   * 4.5 Parents Education Level
   * 4.6 Frequency of Going Out
   * 4.7 Desire of Higher Education
   * 4.8 Urban Vs. Rural Students
5. Classification
   * 5.1 Prepare Dataset for Modelling
   * 5.2 Decision Tree Classifier
   * 5.3 Random Forest Classifier
   * 5.4 Support Vector Classifier
   * 5.5 Logistic Regression Classifier
   * 5.6 Model Selection
6. Conclusion

1) First we have imported all necessary packages required in the dataset.

2) Then we have loaded both the datasets which are math and por respectively.

3) Data preparation.

We merge both datasets using the “concat” function and rename column labels for better understanding of the features.

4**)**Categorizing students based on their performance in exams:

Students are categorized based on their marks achieved into three categories:

1)Good: 15-20marks

2)Average: 10-14 marks

3) Poor: 0-9 marks

5) After categorizing the students, we check the information related to dataset such as size, length, info etc. and check If there are any missing variables or not.

6) Now we perform exploratory data analysis

EDA will help us reveal beyond the formal modeling or hypothesis testing task and provides a better understanding of data set variables and the relationships between them.

7) Now we use final grade distribution and plot final grade count plot this will show that majority of the students fall under the average category.

8) Plot a correlation heatmap which is graphical representation of correlation matrix representing correlation between different variables,

9) Applying hypothesis testing for different parameters and factors and checking its relationship with student’s grades.

It is found that Romantic Status, Alcohol Consumption, Parents Education Level, Frequency of Going Out, Desire of Higher Education and Living Area all these attributes have an impact on the student’s performance.

Romantic Status, Alcohol Consumption, Frequency of going out affect their performance in a negative way.

Whereas parents’ education level, desire of higher education and living area have a positive impact in their performance.

**SUGGESTIONS FOR STUDENTS TO IMPROVE THEIR PERFORMANCE BASED ON THE ANALYSIS:**

1) **No high school romance** as it is found that students who are not having a romantic status are more likely to score better marks.

2) **No alcohol consumption:** It is found that students who consume alcohol more often perform poor.

3) **Parents education plays a significant role:** Especially a mother’s education

4) **Limit the frequency of going out with friends:** It was found that students who go out twice a week score better marks then students who go out 4-5 times a week.

5) **Interest for college or higher education:** It was also found that students who wanted to study further for college or university performed significantly better than others so students should be encouraged for higher education.

6**) Better living area**: It was found that students who stay in urban places perform better than students who stay in rural areas.

CLASSIFICATION AND COMPARING RESULTS FOR DIFFERENT ALGORITHMS:

1. DECISION TREE CLASSIFIER
2. RANDOM FOREST CLASSIFIER
3. SUPPORT VECTOR CLASSIFIER
4. LOGISITIC REGRESSION CLASSIFIER

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Decision Tree Classifier | Random Forest Classifier | Support Vector Classifier | Logistic Regression Classifier |
| Model Score | 0.8904109589041096 | 0.9726027397260274 | 0.8849315068493151 | 0.8972602739726028 |
| Cross- Validation Score | 0.8821656050955414 | 0.8630573248407644 | 0.8694267515923567 | 0.2229299363057325 |

As you can see Logistic Regression is the Best Model

CODE:

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

%matplotlib inline

import seaborn as sns

sns.set\_style('whitegrid')

import statsmodels.api as sm

# load datasets for two subjects, Math and Portuguese

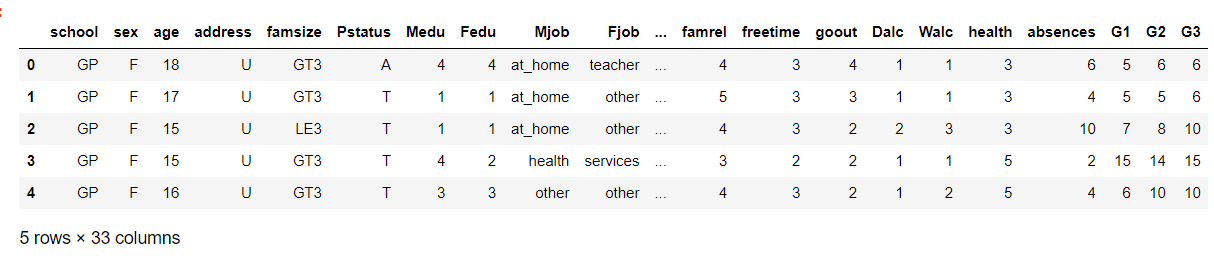
mat = pd.read\_csv("student-mat.csv", sep=';')

por = pd.read\_csv("student-por.csv", sep=';')

#merging the datasets

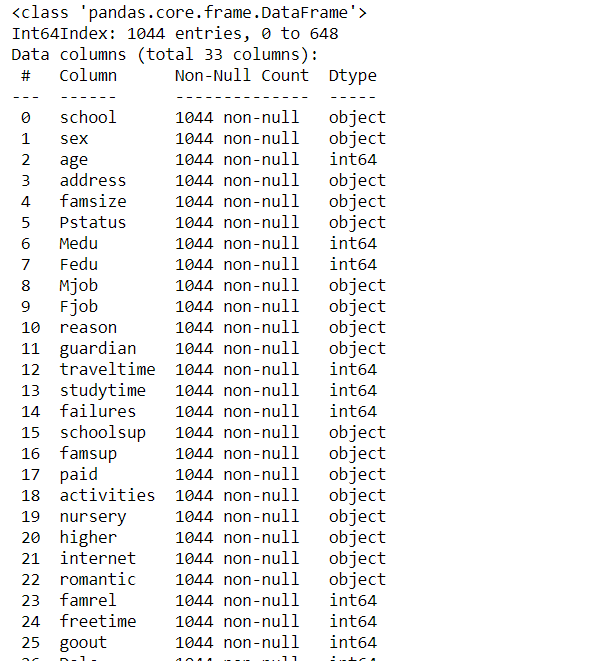
df = pd.concat([mat,por])

df.head(5)



#information about the dataset

df.info()



#shape of the dataset rows,columns

print(df.shape)

print(df.size)



print(len(mat))

print(len(por))

print(len(df))



# 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']

# convert final\_score to categorical variable # Good:15~20 Fair:10~14 Poor:0~9

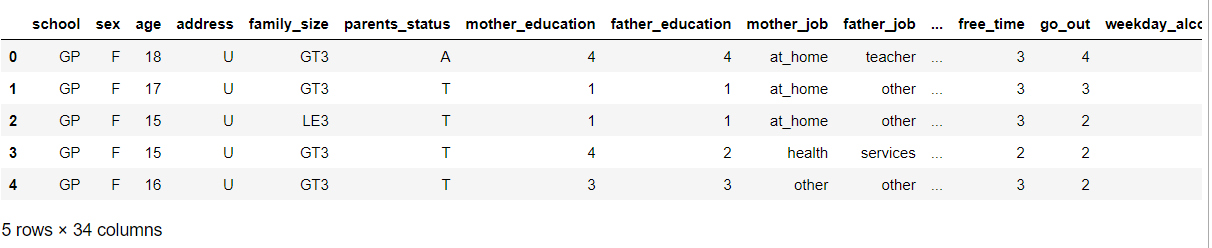
df['final\_grade'] = 'na'

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'] = 'average'

df.loc[(df.final\_score >= 0) & (df.final\_score <= 9), 'final\_grade'] = 'poor'

df.head(5)

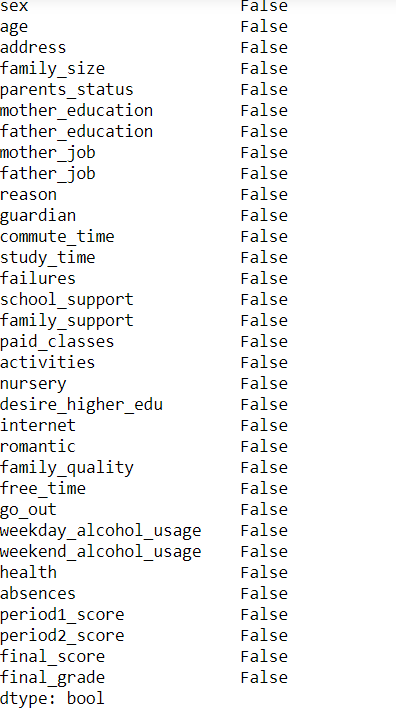


#Analysing the variables

#checking for missing variables

df.isnull().any()

#It is found that there are no missing variables in the dataset



#Exploratory data analysis on the student dataset

#EDA will help us reveal beyond the formal modeling or hypothesis testing task and provides a better understanding of data set

#variables and the relationships between them.

#Final Grade Distribution

# Final Grade Countplot

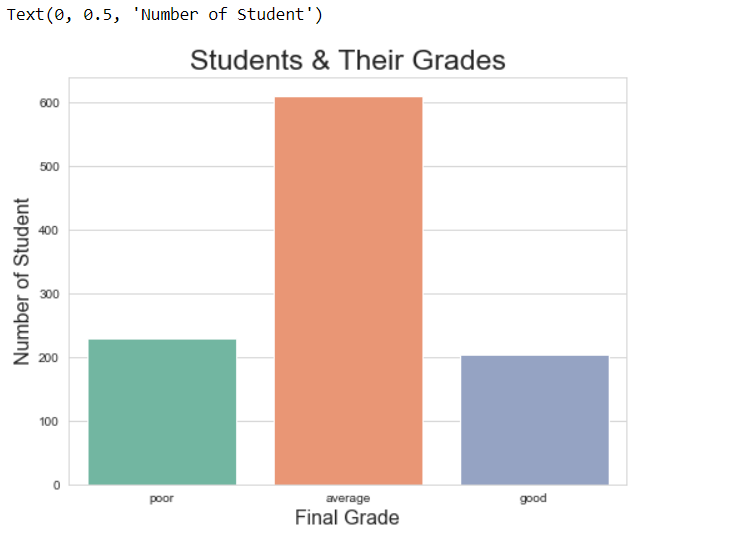
plt.figure(figsize=(8,6))

sns.countplot(df.final\_grade, order=["poor","average","good"], palette='Set2')

plt.title('Students & Their Grades',fontsize=23)

plt.xlabel('Final Grade', fontsize=16)

plt.ylabel('Number of Student', fontsize=16)



#Corelation heatmap

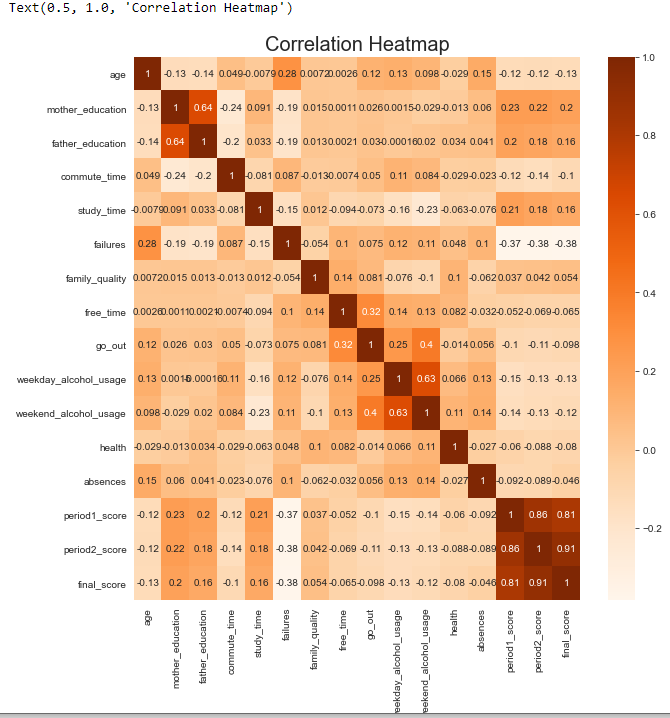
#we can use a correlation heatmap to see correlation between variables

corr = df.corr()

plt.figure(figsize=(10,10))

sns.heatmap(corr, annot=True, cmap="Oranges")

plt.title('Correlation Heatmap', fontsize=20)



#Applying hypothesis testing for different parameters and factors

#checking its relationship with students grades

# romantic status

perc = (lambda col: col/col.sum())

index = ['poor','average','good']

romance\_tab1 = pd.crosstab(index=df.final\_grade, columns=df.romantic)

romance\_tab = np.log(romance\_tab1)

romance\_perc = romance\_tab.apply(perc).reindex(index)

plt.figure()

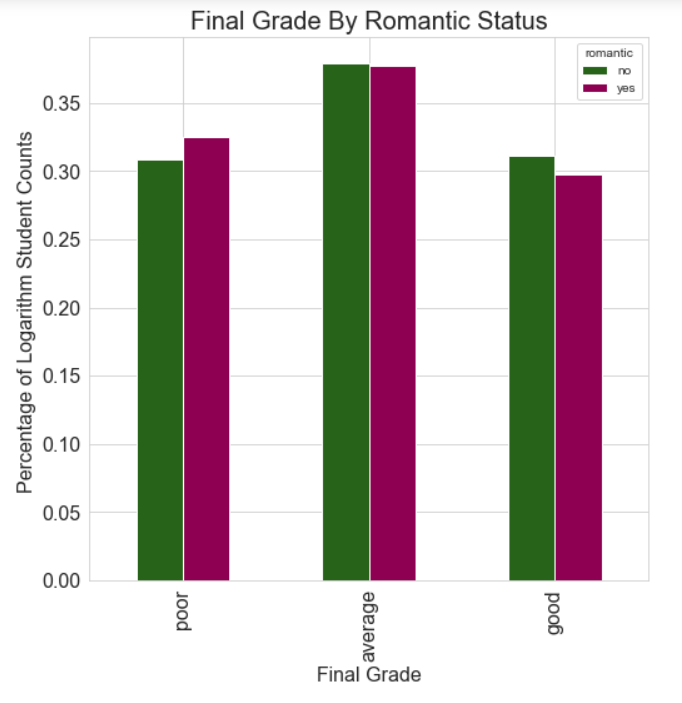
romance\_perc.plot.bar(colormap="PiYG\_r", fontsize=16, figsize=(8,8))

plt.title('Final Grade By Romantic Status', fontsize=20)

plt.ylabel('Percentage of Logarithm Student Counts ', fontsize=16)

plt.xlabel('Final Grade', fontsize=16)

plt.show()



# chi-square test result -- significant!

import statsmodels.api as sm

romance\_table = sm.stats.Table(romance\_tab1)

romance\_rslt = romance\_table.test\_nominal\_association()

romance\_rslt.pvalue



# weekend alcohol consumption

alc\_tab1 = pd.crosstab(index=df.final\_grade, columns=df.weekend\_alcohol\_usage)

alc\_tab = np.log(alc\_tab1)

alc\_perc = alc\_tab.apply(perc).reindex(index)

# create good student dataframe

good = df.loc[df.final\_grade == 'good']

good['good\_alcohol\_usage']=good.weekend\_alcohol\_usage

# create poor student dataframe

poor = df.loc[df.final\_grade == 'poor']

poor['poor\_alcohol\_usage']=poor.weekend\_alcohol\_usage

plt.figure(figsize=(10,6))

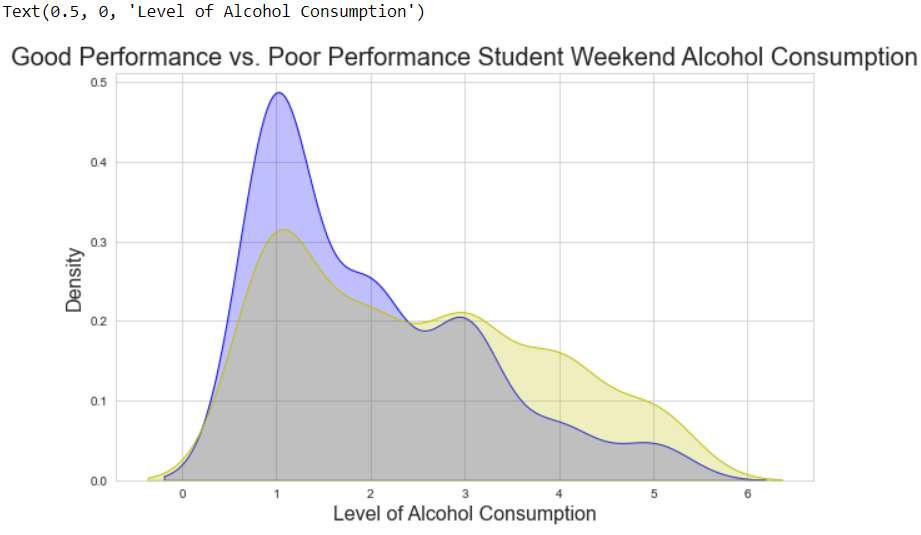
p1=sns.kdeplot(good['good\_alcohol\_usage'], shade=True, color="b")

p1=sns.kdeplot(poor['poor\_alcohol\_usage'], shade=True, color="y")

plt.title('Good Performance vs. Poor Performance Student Weekend Alcohol Consumption', fontsize=20)

plt.ylabel('Density', fontsize=16)

plt.xlabel('Level of Alcohol Consumption', fontsize=16)

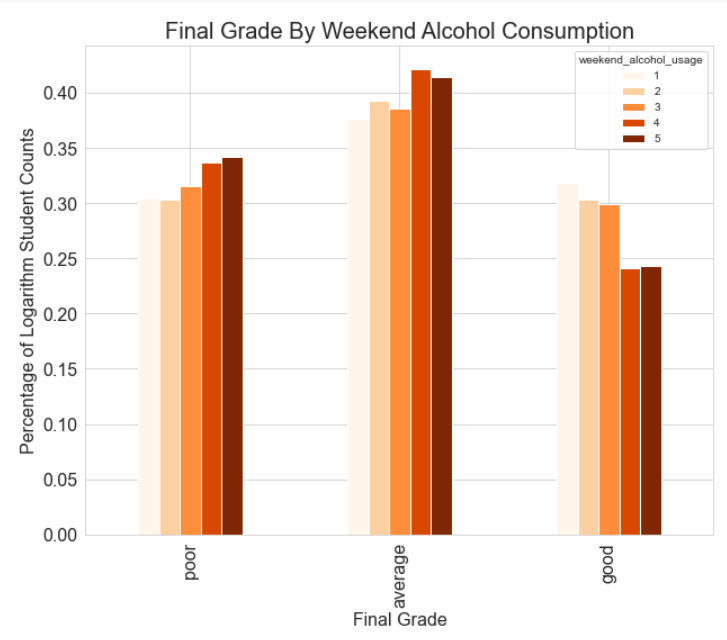


alc\_perc.plot.bar(colormap="Oranges", figsize=(10,8), fontsize=16)

plt.title('Final Grade By Weekend Alcohol Consumption', fontsize=20)

plt.ylabel('Percentage of Logarithm Student Counts', fontsize=16)

plt.xlabel('Final Grade', fontsize=16)



# chi-square test result -- significant!

import statsmodels.api as sm

alc\_table = sm.stats.Table(alc\_tab1)

alc\_rslt = alc\_table.test\_nominal\_association()

alc\_rslt.pvalue



#parents education

good['good\_student\_father\_education'] = good.father\_education

poor['poor\_student\_father\_education'] = poor.father\_education

good['good\_student\_mother\_education'] = good.mother\_education

poor['poor\_student\_mother\_education'] = poor.mother\_education

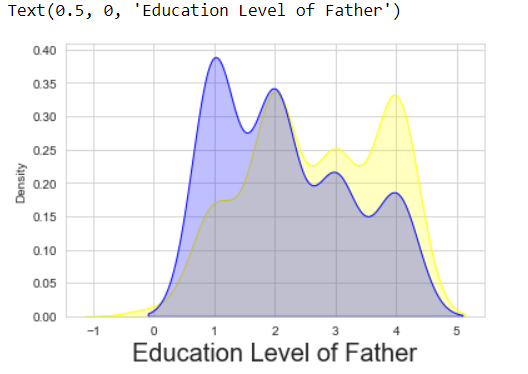
# see the difference between good and poor performers' father education level(numeric: from 1 - very low to 5 - very high)

plt.figure(figsize=(6,4))

p2=sns.kdeplot(good['good\_student\_father\_education'], shade=True, color="yellow")

p2=sns.kdeplot(poor['poor\_student\_father\_education'], shade=True, color="b")

plt.xlabel('Education Level of Father', fontsize=20)



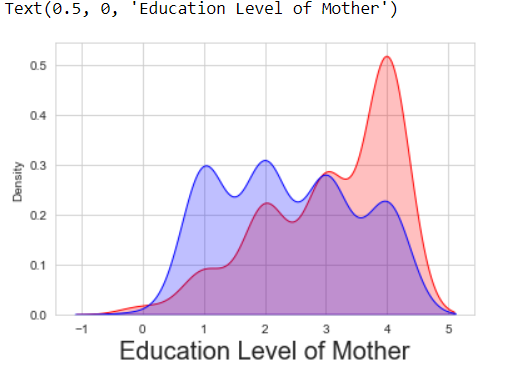
# see the difference between good and poor performers' mother education level(numeric: from 1 - very low to 5 - very high)

plt.figure(figsize=(6,4))

p3=sns.kdeplot(good['good\_student\_mother\_education'], shade=True, color="r")

p3=sns.kdeplot(poor['poor\_student\_mother\_education'], shade=True, color="b")

plt.xlabel('Education Level of Mother', fontsize=20)



# OLS tells that parents' education level has a positive correlation with students' final score.

#Comparatively, mother's education level has bigger influence than father's education level!

# use OLS to see coefficients

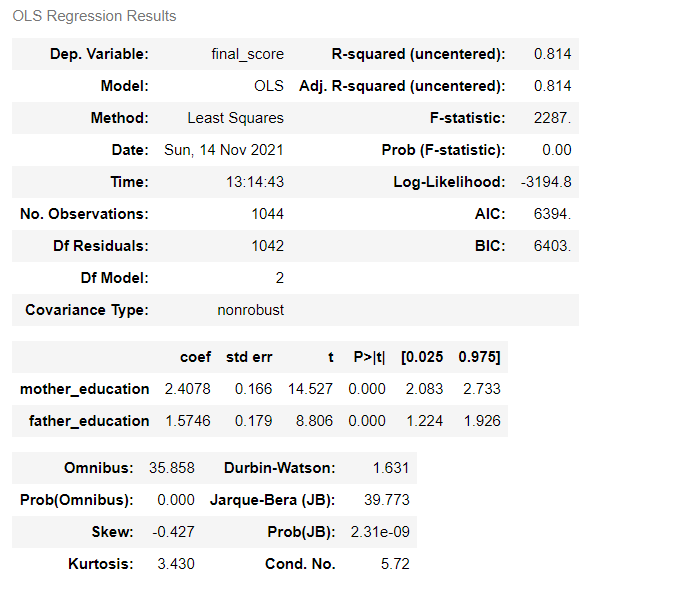
X\_edu = df[['mother\_education','father\_education']]

y\_edu = df.final\_score

edu = sm.OLS(y\_edu, X\_edu)

results\_edu = edu.fit()

results\_edu.summary()



#Final Grade By Frequency Of Going Out

# going out with friends (numeric: from 1 - very low to 5 - very high)

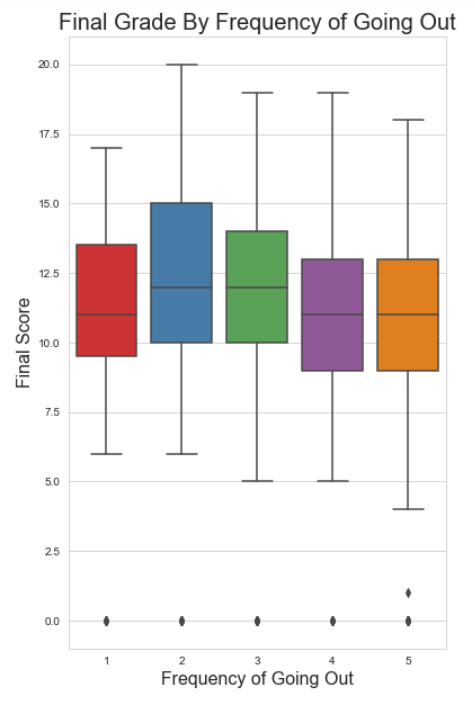
plt.figure(figsize=(6,10))

sns.boxplot(x='go\_out', y='final\_score', data=df, palette='Set1')

plt.title('Final Grade By Frequency of Going Out', fontsize=20)

plt.ylabel('Final Score', fontsize=16)

plt.xlabel('Frequency of Going Out', fontsize=16)



out\_tab = pd.crosstab(index=df.final\_grade, columns=df.go\_out)

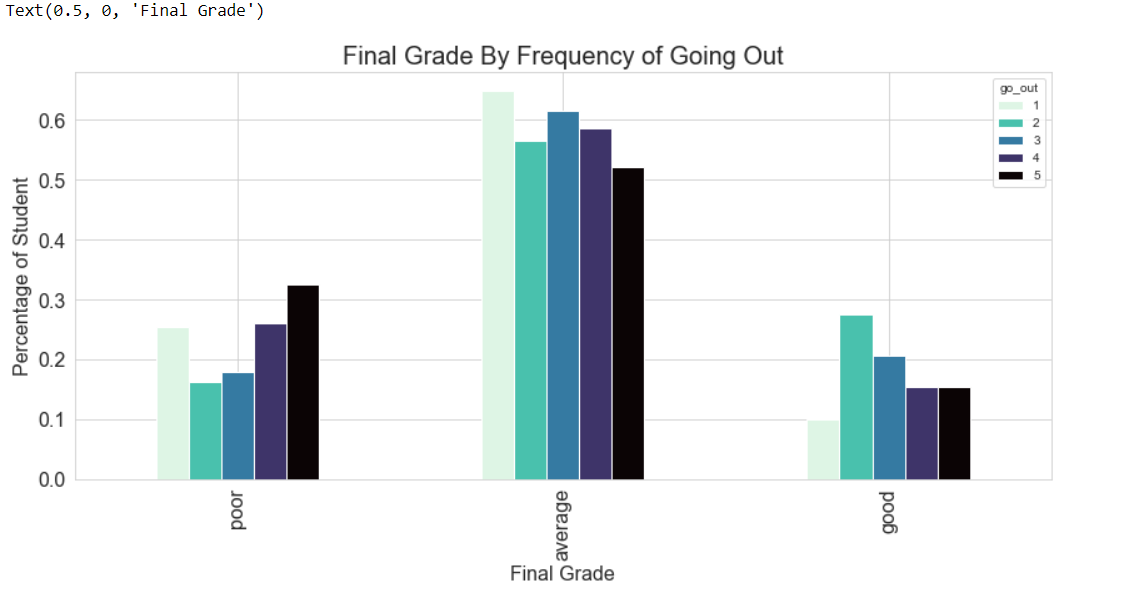
out\_perc = out\_tab.apply(perc).reindex(index)

out\_perc.plot.bar(colormap="mako\_r", fontsize=16, figsize=(14,6))

plt.title('Final Grade By Frequency of Going Out', fontsize=20)

plt.ylabel('Percentage of Student', fontsize=16)

plt.xlabel('Final Grade', fontsize=16)



# chi-square test result -- significant!

out\_table = sm.stats.Table(out\_tab)

out\_rslt = out\_table.test\_nominal\_association()

out\_rslt.pvalue



#Final Grade By Desire To Go To College

# Desire for higher education and study time by age

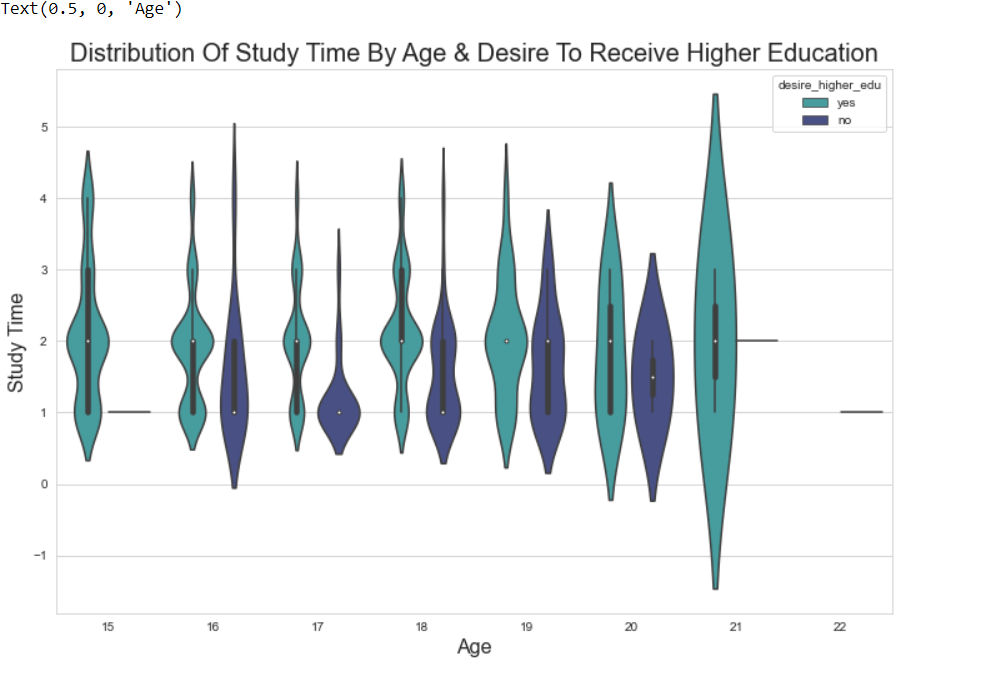
plt.figure(figsize=(12,8))

sns.violinplot(x='age', y='study\_time', hue='desire\_higher\_edu', data=df, palette="mako\_r", ylim=(1,6))

plt.title('Distribution Of Study Time By Age & Desire To Receive Higher Education', fontsize=20)

plt.ylabel('Study Time', fontsize=16)

plt.xlabel('Age', fontsize=16)



higher\_tab = pd.crosstab(index=df.final\_grade, columns=df.desire\_higher\_edu)

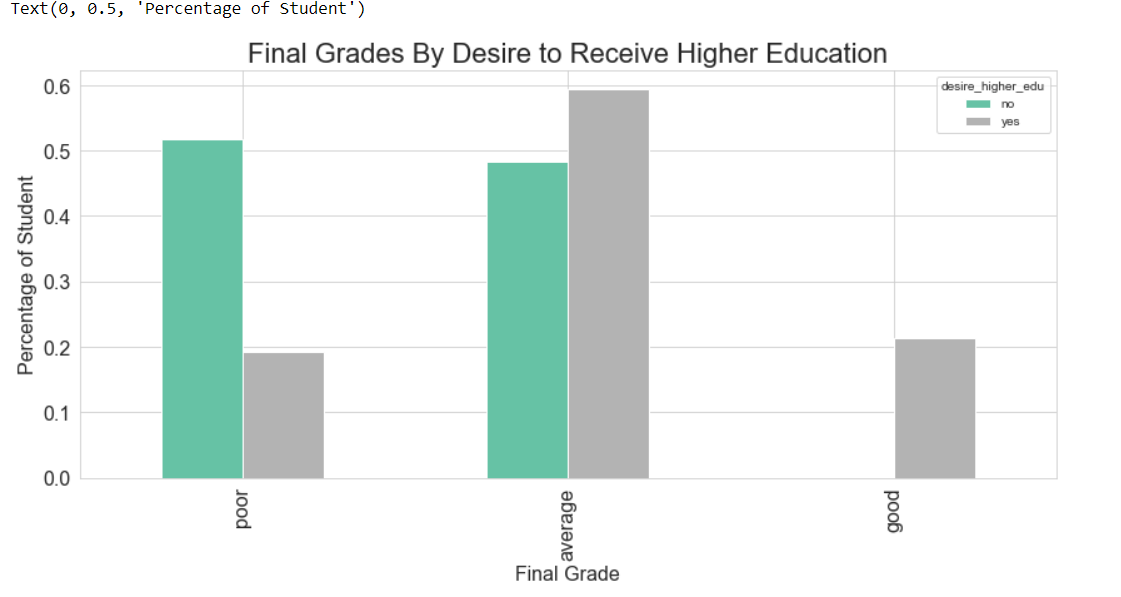
higher\_perc = higher\_tab.apply(perc).reindex(index)

higher\_perc.plot.bar(colormap="Set2", figsize=(14,6), fontsize=16)

plt.title('Final Grades By Desire to Receive Higher Education', fontsize=22)

plt.xlabel('Final Grade', fontsize=16)

plt.ylabel('Percentage of Student', fontsize=16)



# chi-square test result -- significant!

import statsmodels.api as sm

higher\_table = sm.stats.Table(higher\_tab)

higher\_rslt = higher\_table.test\_nominal\_association()

higher\_rslt.pvalue



#Final Grade By Living Area

# living area: urban vs. rural

df.address = df.address.map({'U':'Urban', 'R':'Rural'})

plt.figure(figsize=(6,6))

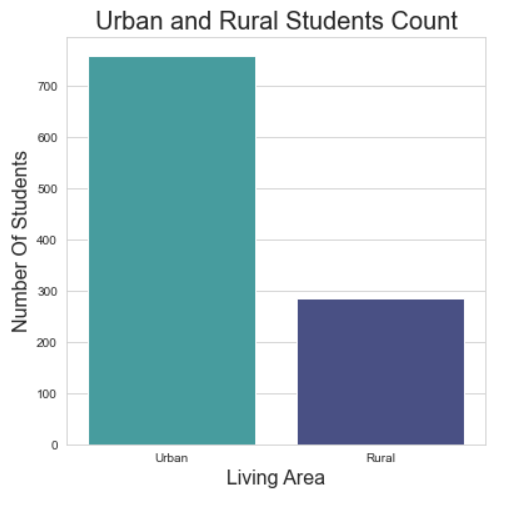
sns.countplot(df.address, palette='mako\_r')

plt.title('Urban and Rural Students Count', fontsize=20)

plt.xlabel('Living Area', fontsize=16)

plt.ylabel('Number Of Students', fontsize=16)

plt.show()



ad\_tab1 = pd.crosstab(index=df.final\_grade, columns=df.address)

ad\_tab = np.log(ad\_tab1)

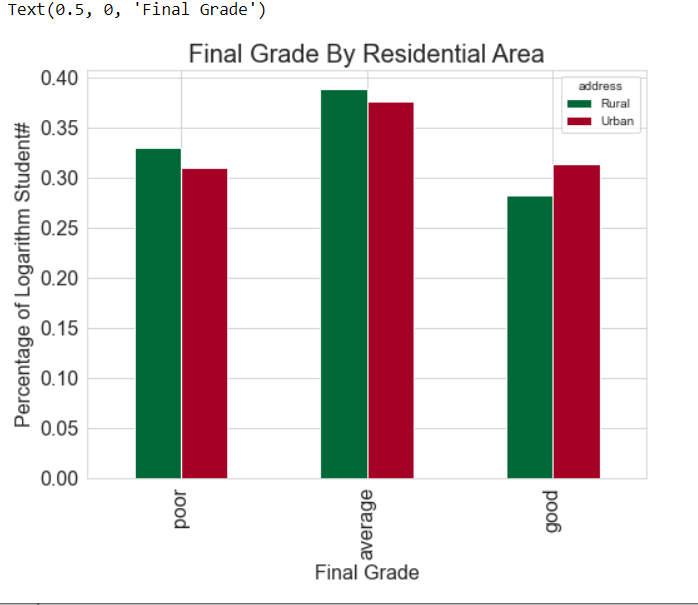
ad\_perc = ad\_tab.apply(perc).reindex(index)

ad\_perc.plot.bar(colormap="RdYlGn\_r", fontsize=16, figsize=(8,6))

plt.title('Final Grade By Residential Area', fontsize=20)

plt.ylabel('Percentage of Logarithm Student#', fontsize=16)

plt.xlabel('Final Grade', fontsize=16)



# chi-square test result -- significant!

ad\_table = sm.stats.Table(ad\_tab1)

ad\_rslt = ad\_table.test\_nominal\_association()

ad\_rslt.pvalue



# explore other variables via OLS

dfl = df.copy()

X\_ols = dfl.drop(['period1\_score', 'period2\_score', 'final\_score','final\_grade', 'failures','study\_time','absences'], axis=1)

X\_ols = pd.get\_dummies(X\_ols)

mod = sm.OLS(df.final\_score, X\_ols)

mod = mod.fit()

# Data Model Preparation

# create dataframe dfd for classification

dfd = df.copy()

dfd = dfd.drop([ 'final\_score'], axis=1)

# label encode final\_grade

from sklearn import preprocessing

le = preprocessing.LabelEncoder()

dfd.final\_grade = le.fit\_transform(dfd.final\_grade)

# dataset train\_test\_split

from sklearn.model\_selection import train\_test\_split

X = dfd.drop('final\_grade',axis=1)

y = dfd.final\_grade

X\_train,X\_test,y\_train,y\_test=train\_test\_split(X,y,test\_size=0.3)

# get dummy varibles

X\_train = pd.get\_dummies(X\_train)

X\_test = pd.get\_dummies(X\_test)

# see total number of features

len(list(X\_train))



# Decision Tree Classifier

# find the optimal number of minimum samples leaf

from sklearn.tree import DecisionTreeClassifier

import matplotlib.pyplot as plt

msl=[]

for i in range(1,58):

tree = DecisionTreeClassifier(min\_samples\_leaf=i)

t= tree.fit(X\_train, y\_train)

ts=t.score(X\_test, y\_test)

msl.append(ts)

msl = pd.Series(msl)

msl.where(msl==msl.max()).dropna()

# final model

tree = DecisionTreeClassifier(min\_samples\_leaf=17)

t= tree.fit(X\_train, y\_train)

print("Decision Tree Model Score" , ":" , t.score(X\_train, y\_train) , "," ,

"Cross Validation Score" ,":" , t.score(X\_test, y\_test))



# Random Forest Classifier

# find a good # of estimators

from sklearn.ensemble import RandomForestClassifier

ne=[]

for i in range(1,58):

forest = RandomForestClassifier()

f = forest.fit(X\_train, y\_train)

fs = f.score(X\_test, y\_test)

ne.append(fs)

ne = pd.Series(ne)

ne.where(ne==ne.max()).dropna()

# find a good # of min\_samples\_leaf

from sklearn.ensemble import RandomForestClassifier

ne=[]

for i in range(1,58):

forest = RandomForestClassifier(n\_estimators=36, min\_samples\_leaf=i)

f = forest.fit(X\_train, y\_train)

fs = f.score(X\_test, y\_test)

ne.append(fs)

ne = pd.Series(ne)

ne.where(ne==ne.max()).dropna()

# final model

forest = RandomForestClassifier(n\_estimators=36, min\_samples\_leaf=2)

f = forest.fit(X\_train, y\_train)

print("Random Forest Model Score" , ":" , f.score(X\_train, y\_train) , "," ,

"Cross Validation Score" ,":" , f.score(X\_test, y\_test))



# Support Vector Classification

from sklearn.svm import SVC

svc = SVC()

s= svc.fit(X\_train, y\_train)

print("SVC Model Score" , ":" , s.score(X\_train, y\_train) , "," ,

"Cross Validation Score" ,":" , s.score(X\_test, y\_test))



# Logistic Regression

from sklearn.linear\_model import LogisticRegression

lr = LogisticRegression(multi\_class='multinomial', solver='newton-cg',fit\_intercept=True)

# find optimal # of features to use in the model

from sklearn.feature\_selection import SelectKBest, chi2

ks=[]

for i in range(1,58):

sk = SelectKBest(chi2, k=i)

x\_new = sk.fit\_transform(X\_train,y\_train)

x\_new\_test=sk.fit\_transform(X\_test,y\_test)

l = lr.fit(x\_new, y\_train)

ll = l.score(x\_new\_test, y\_test)

ks.append(ll)

ks = pd.Series(ks)

ks = ks.reindex(list(range(1,58)))

ks

plt.figure(figsize=(10,5))

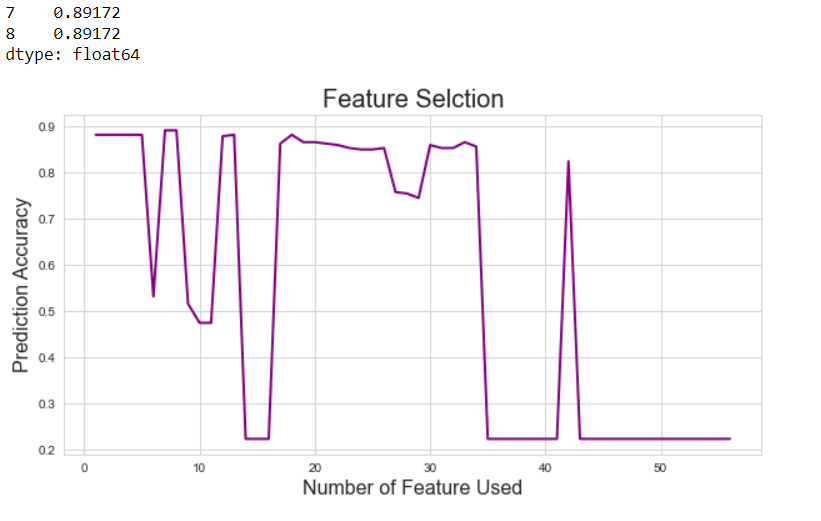
ks.plot.line(linewidth=2, color="purple")

plt.title('Feature Selction', fontsize=20)

plt.xlabel('Number of Feature Used', fontsize=16)

plt.ylabel('Prediction Accuracy', fontsize=16)

ks.where(ks==ks.max()).dropna()



# final model

sk = SelectKBest(chi2, k=8)

x\_new = sk.fit\_transform(X\_train,y\_train)

x\_new\_test=sk.fit\_transform(X\_test,y\_test)

lr = lr.fit(x\_new, y\_train)

print("Logistic Regression Model Score" , ":" , lr.score(x\_new, y\_train) , "," ,

"Cross Validation Score" ,":" , lr.score(x\_new\_test, y\_test))



Conclusion:

Logistic Regression helps us to group a range of values into two or more specific categories. It also helps in converting non numeric data into numeric data that can be plotted onto a graph. Thus judging from the output, we can conclude that for the dataset we have been presented with the logistic regression model is the most accurate and precise model. It gives us the highest score. Logistic Regression helps us to group a range of values into two or more specific categories.