Wilhelm von Arndt

Pascal Wallisch

Introduction to Data Science (DS-UA-112)

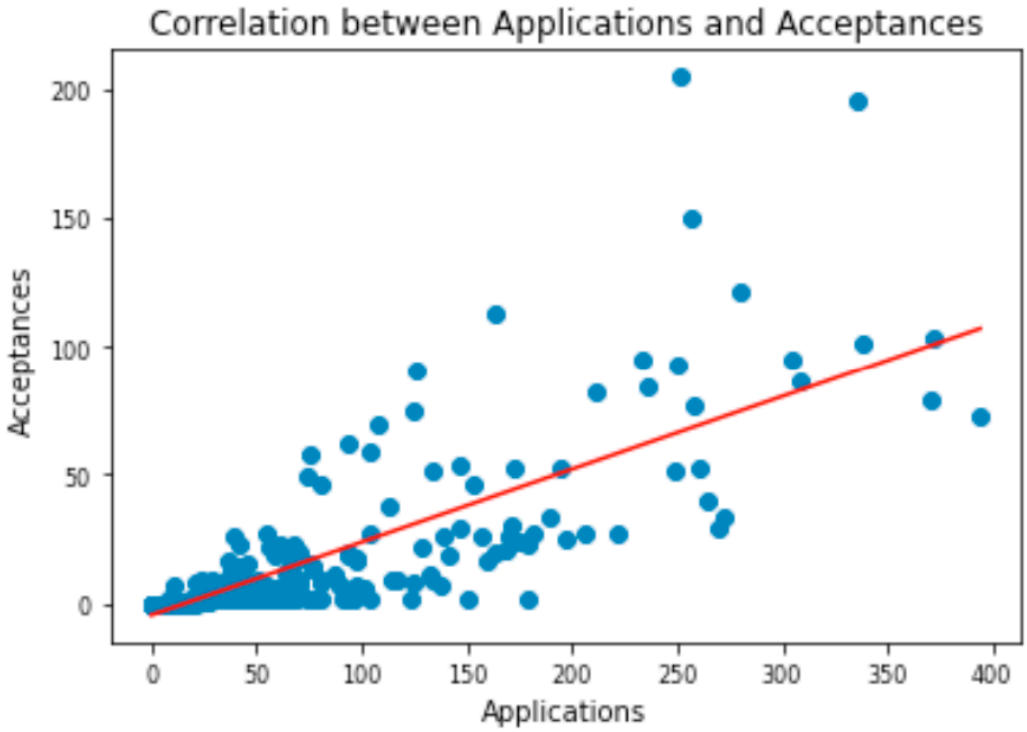
20 May 2021

Analysis of elementary school data of NYC

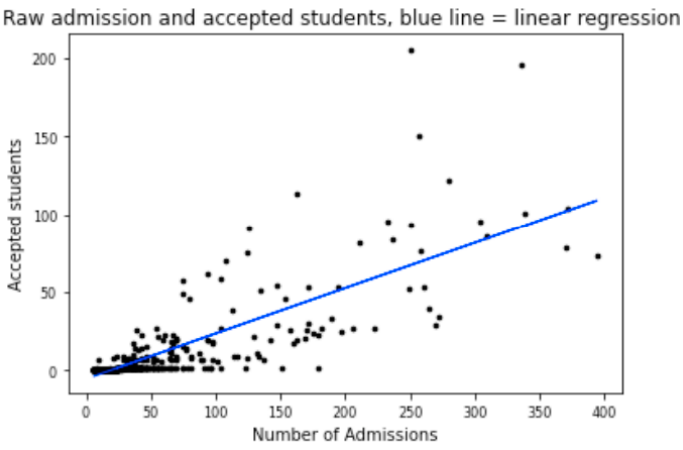
Managing missing data: As the data is not perfect, I will be using methods to correct the missing data throughout this project. This will be done by NaN removal of necessary rows (schools). With this being said, I will not remove **all** rows with >=1 nan as we'd like to keep as much data as possible. What you will see throughout the project is concatenating smaller data-frames from our original data and implementing the nan-removal method on this smaller set. For example, if we are working on a dataset with three columns (which we know contains NaNs): A, B, and C. I want to correlate column A with B, instead of removing all rows that contain one or more nan (regardless of its placement), we will group A and B before performing this action (on A and B **only**). This will keep the potential rows of data in which the C column contained a NaN, instead of sorting out perfect data. Beyond this technique I use the function nanRemover for Numpy array, which is defined between rows 20-37.

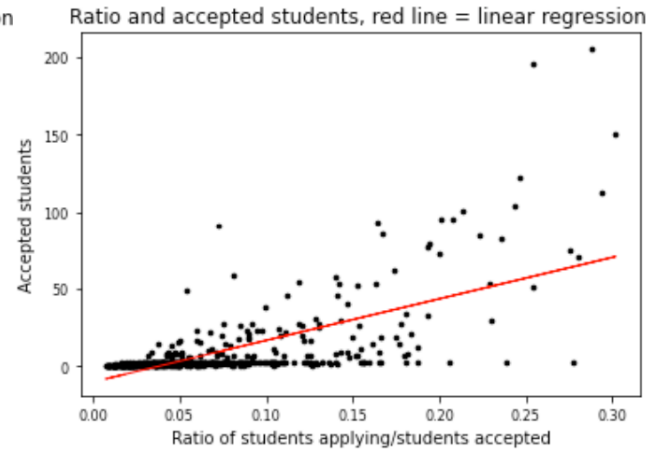
The plotting will not be commented as it's fairly straight forward and can easily be understood by the plots itself, all my plotting in this analysis uses MatPlotLib.

1. The correlation between the number of applications and admissions to HSPHS is **0.8017**.

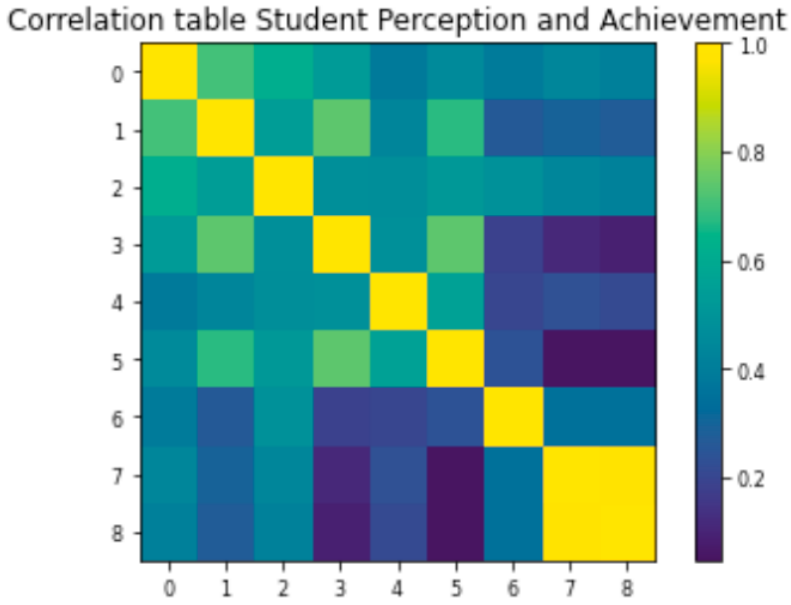


2. Removal of schools with 0 applicants were removed from this calculation as their 0/0 application/admissions rate as this would appear the same as a school with 0 acceptances but with >0 applications. Then two linear regressions were performed in which the admissions (Y) were predicted by the number of raw applications (x) and the application ratio (x) respectively. The findings were that **raw applications** are a **better predictor** as it's coefficient of determination (R^2) was **0.655** (blue line), vs. a 0.440 coefficient for the application-rate model (red line).

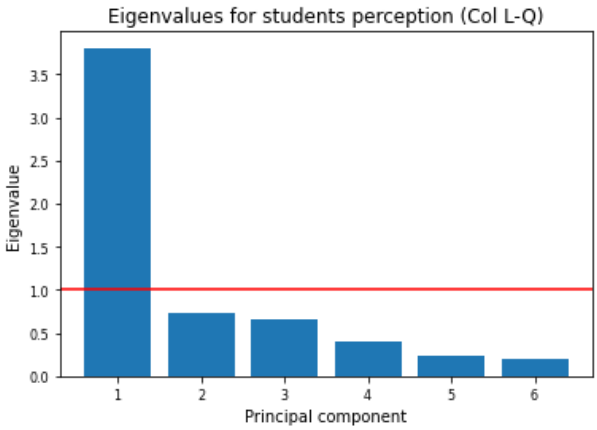


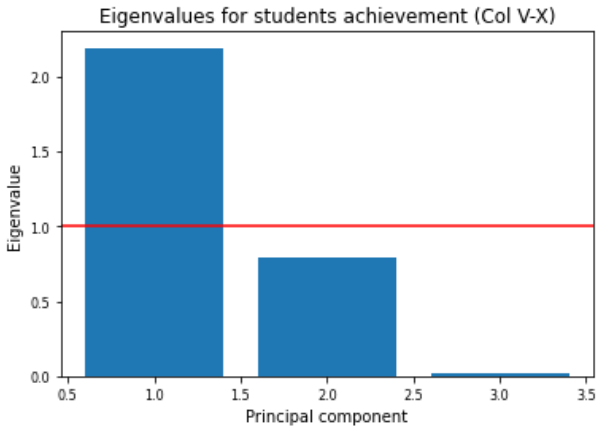


3. The ratio was calculated by examining the acceptances/school size ratio. The school with the highest per student odds HSPHS was Christa McAuliffe School with odds of **3.26 to 1** for sending someone to HSPHS. Here is a plot of the correlation:
4. The first step to determining if there are any relationship in the form of correlation:



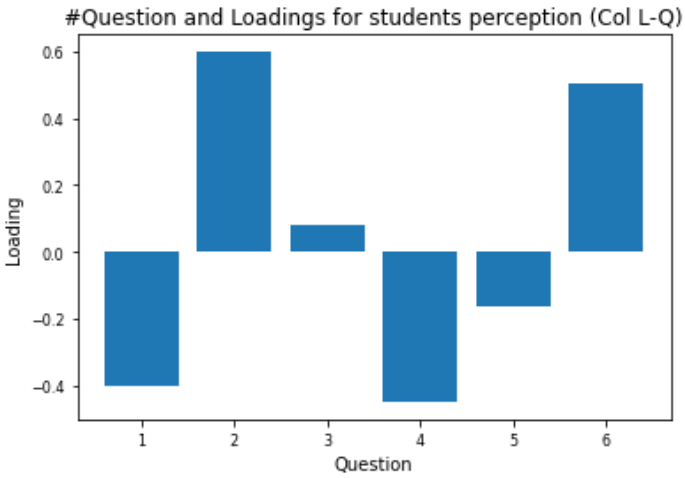
From here we see that there are multiple correlations within, which creates incentive to perform a dimension reduction in the form of a two PCA's, before this we normalize the data.

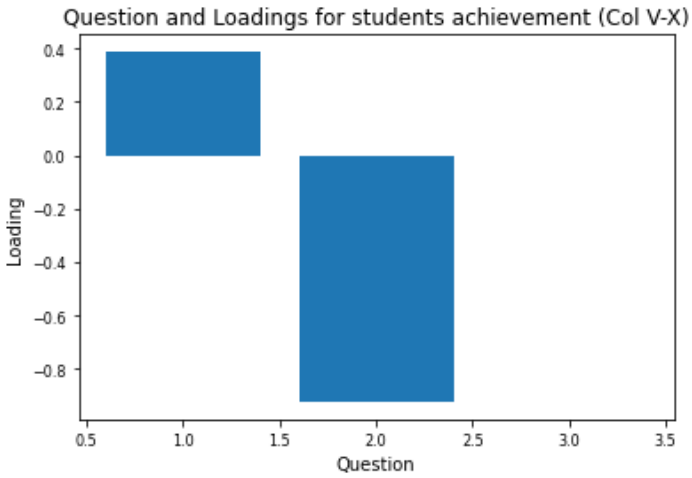




I will qualify the Eigenvalues from the Kaiser-criteron which is marked in red. For both PCA's we get one important important factor to reatin.

We then look at the scores of the two factor loadings:





For the loadings of Perception (left chart), we find four different loadings that are intersting to us and that we'd like to analyze (two stronger ones and two weaker ones), where they represent Rigorous Instructions (1) , Collaborative Teachers (2), Effective School Leadership (4), and Trust(6). For the loadings of student achievement (right) we find one important loadings which is Reading Scores Exceeded (2). We then conduct an ANOVA to analyze the variance among these, getting the following table:

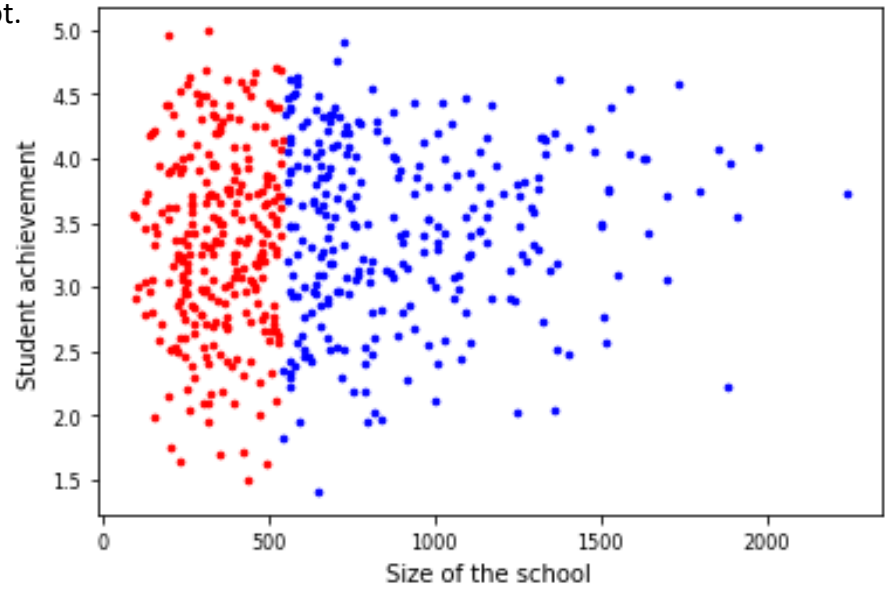
Index	sum_sq	df	F	PR(>F)
collaborative_teachers	0.641695	1	1.67985	0.195528
trust	6.02203	1	15.7646	8.19752e−05
reading_scores_exceed	10.9013	1	28.5377	1.38386e−07
effective_school_leadership	0.625124	1	1.63647	0.201389
rigorous_instruction	9.276	1	24.283	1.12422e−06
collaborative_teachers:trust	0.410197	1	1.07382	0.300571
collaborative_teachers:reading_scores_exceed	0.247054	1	0.646746	0.421651
collaborative_teachers:effective_school_leadership	0.108141	1	0.283094	0.594911
collaborative_teachers:rigorous_instruction	0.66768	1	1.74787	0.186734
trust:reading_scores_exceed	0.183792	1	0.481135	0.488223
trust:effective_school_leadership	0.167732	1	0.439095	0.507857
trust:rigorous_instruction	0.0686083	1	0.179605	0.671891
reading_scores_exceed:effective_school_leadership	0.0926928	1	0.242654	0.622507
reading_scores_exceed:rigorous_instruction	0.307625	1	0.80531	0.369932
effective_school_leadership:rigorous_instruction	0.313521	1	0.820744	0.365388
Residual	195.964	513	nan	nan

Our model didn't give us too much information on the interaction effects, both in regards of significance but also in F-values and the sums of squares. The main effects are more interesting, we see that trust, reading scores exceed, and rigorous instruction all gives us significant p-values with good F-values and sums of squares (Keep in mind that the loading for rigorous instruction was relatively low). As this model didn't give us a single interaction effect, I decided to narrow it down and instead on interpreting the loadings with an absolute value of ~0.4 (=> loading another ANOVA without 'rigorous instruction' and 'effective school leadership'). That ANOVA table looks like following:

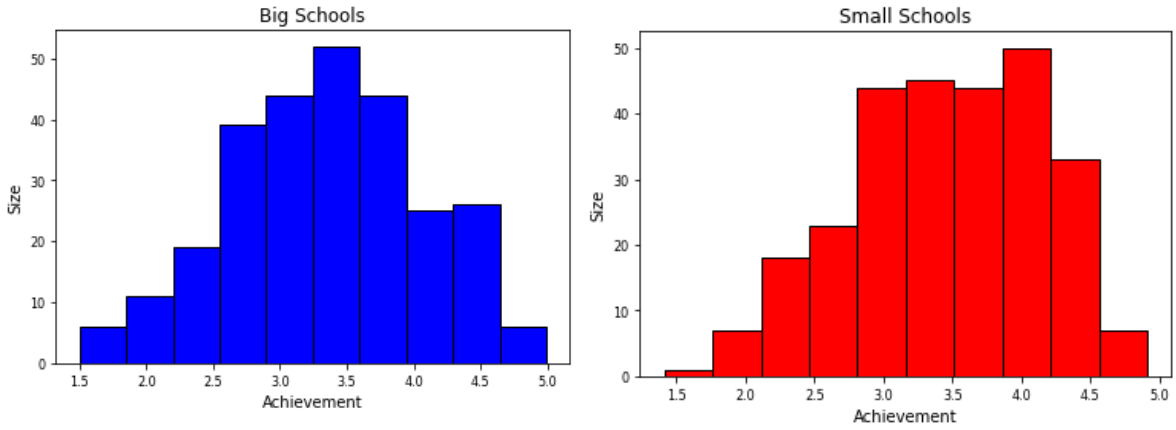
Index	sum_sq	df	F	PR(>F)
collaborative_teachers	0.0624379	1	0.156928	0.692163
trust	5.71095	1	14.3536	0.000169134
reading_scores_exceed	23.1489	1	58.1813	1.14031e-13
collaborative_teachers:trust	2.84764	1	7.15711	0.00770114
collaborative_teachers:reading_scores_exceed	0.072305	1	0.181727	0.670069
trust:reading_scores_exceed	0.492487	1	1.23779	0.26641
Residual	207.691	522	nan	nan

From here we get that the 'reading scores exceed' matters more than we initially thought, it is also confirmed the importance of 'trust'. We also find an interesting interaction effect between 'collaborative teachers' and 'trust'. From these findings I conclude that there is in fact a relationship between student perception and the measurement, it might not be as strong as the importance of reading scores, but the perception matters.

5. Here the school size is dived into large and small, this separation is done by their median. We then plot a scatter plot to see what we can derive intuitively from the plot.



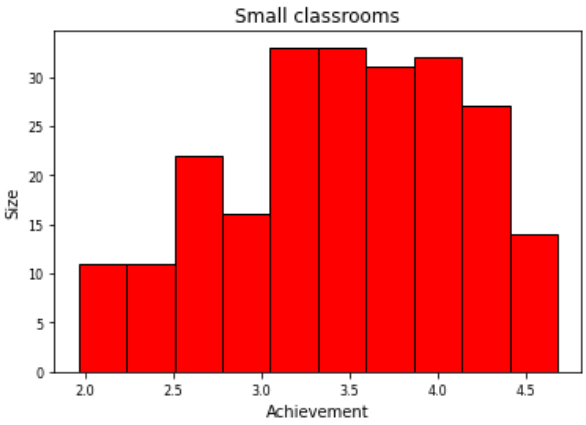
Intuitively there is no difference in the achievement depending on small schools (red), and big schools (blue). Let's investigate this further. My null hypothesis is that school size is not going to change the student achievement. To select the right test to test my null hypothesis I first check if the achievement is normally distributed:

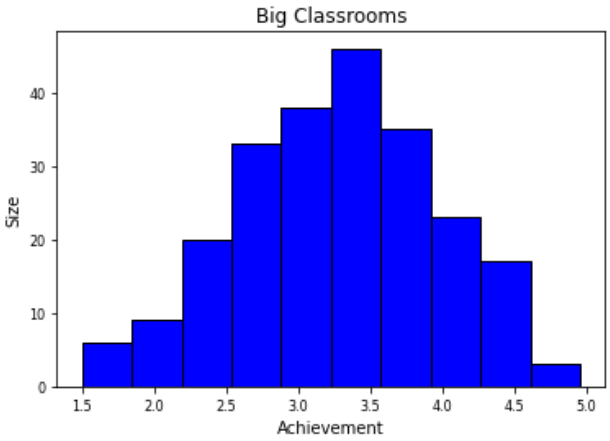


I would say they are normally distributed, hence we do an independent t-test between the achievement of the small schools and achievement of big schools. The

p-value of this test is ≈ **0.06517**, which means that we're unable to reject the null hypothesis – the size of the school does not impact the student achievement.

6. We perform a similar test but now instead of testing the school size we test the average classroom size as the dependent variable, with the null hypothesis that the classroom size does not affect student achievement.

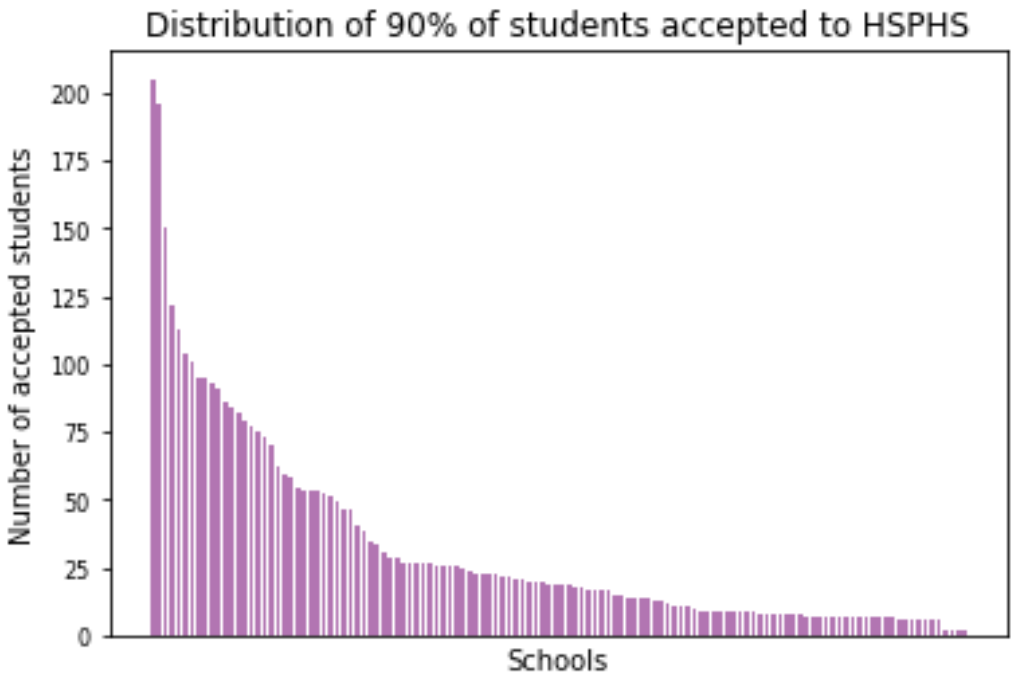




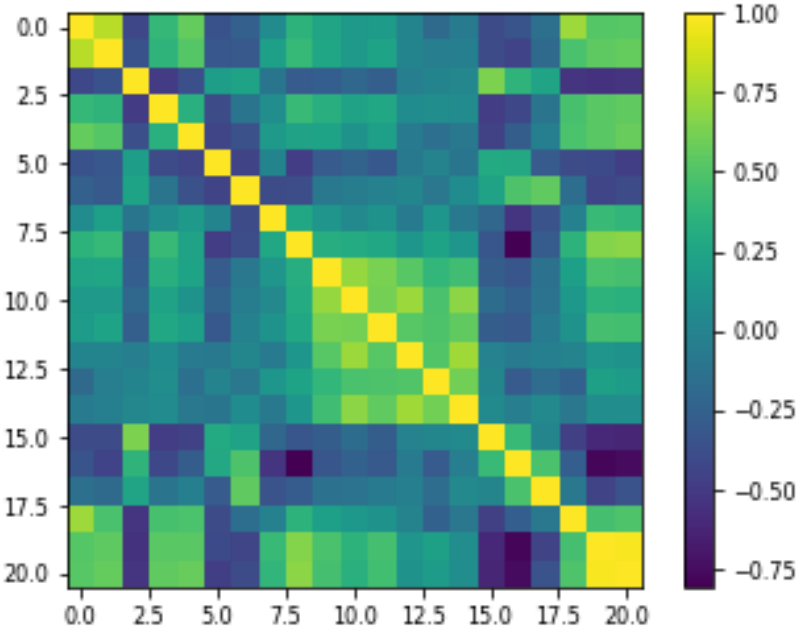
Here it is harder to tell wheteher the data is normally distributed for the 'Small Classrooms' plot, therefore we'll use a Mann Whitney U test just to be on the sure side. The test gives us a **p-value** of **0.001556** meaning that we reject our null hypothesis that calssroom size doesn't affect student achievement.

Doing some further calculations and running a linear regression between student achievement and the average classroom size we get a **r^2 coefficient** of **0.933**, which is huge and really confirms how important this is.

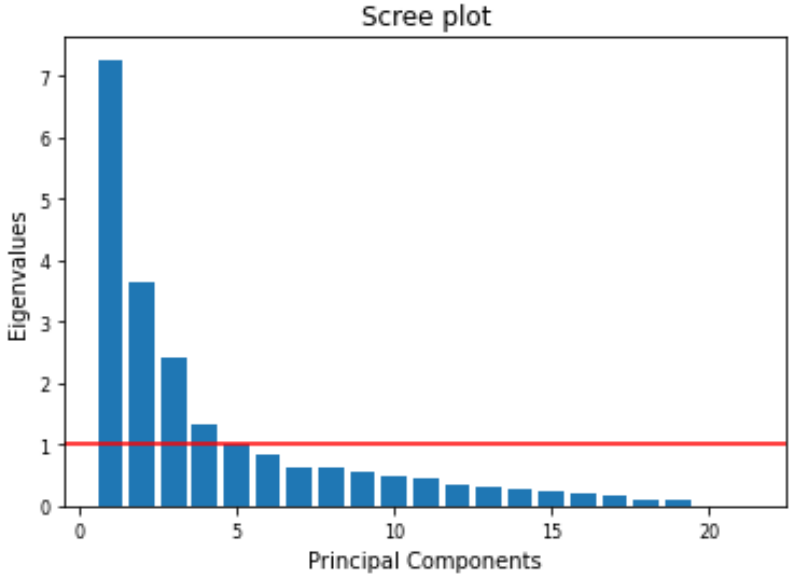
7. We sort the number of accepances from most per school to least, we then find out how many students = 90% of all accepted students. After that we iterate over the sorted list and add to total students until that equals or is larger than the absolute value of students that account for the 90%. We find that 20.7% of all the schools account for 90% of all the acceptances to HSPHS.



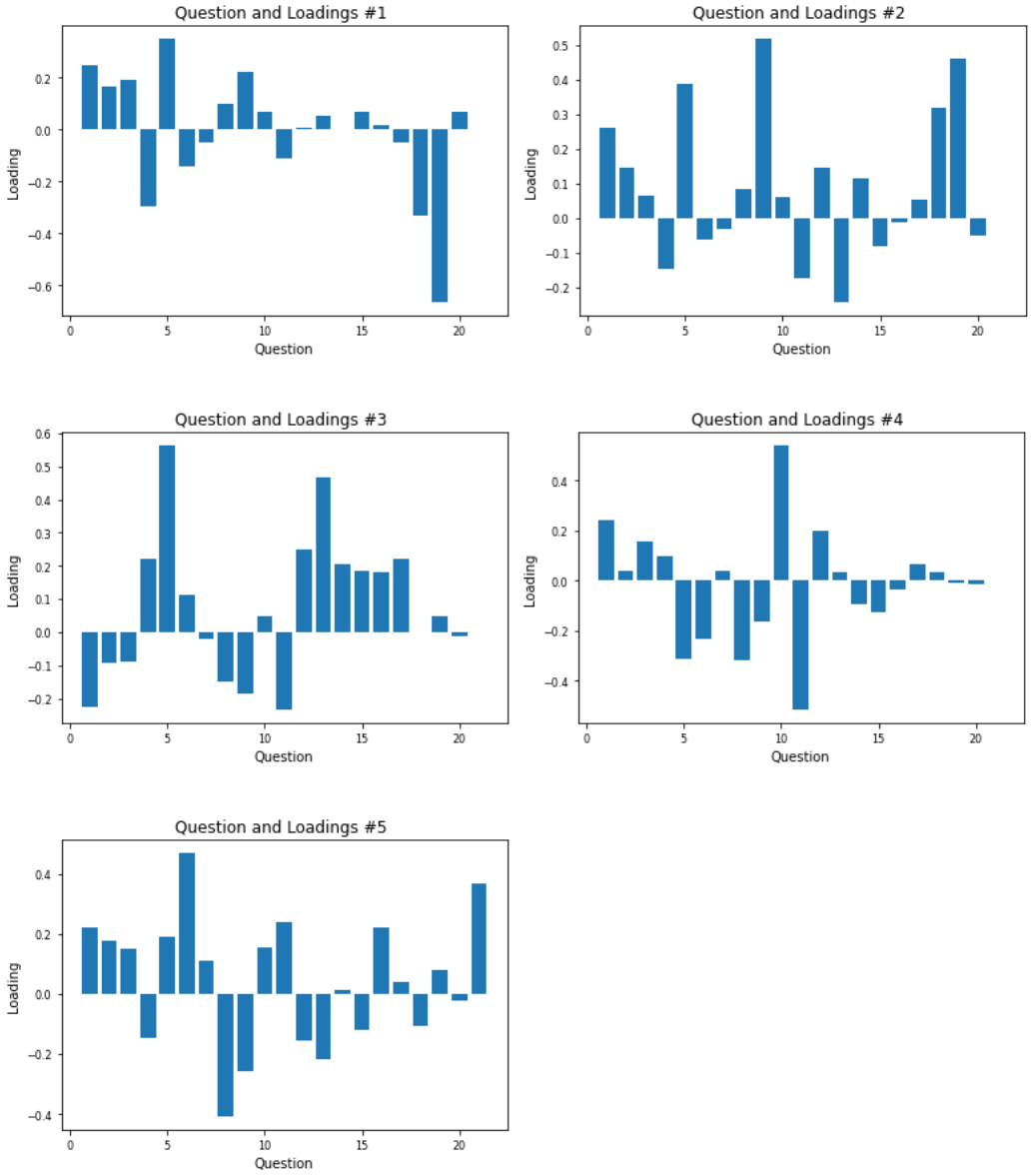
8. Before I could start building a model I contucted a dimension reduction. The method is exactly the same as Q4. Since we're looking at the bigger picture of the data we've been analyzing earlier we know that if correlation exists within the data but just for clarity we'll run a correlation matrix of the whole set (after removing dbn and name columns):



We then calculate our eigenvalues for our 21x21 matrix and find 5 different loadings we want to examine if we can interpret:

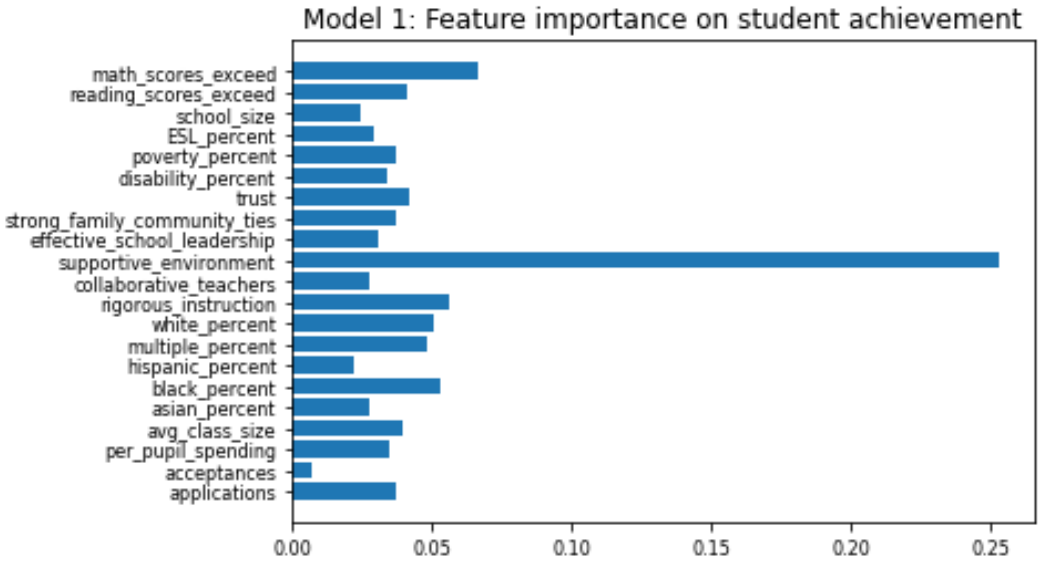


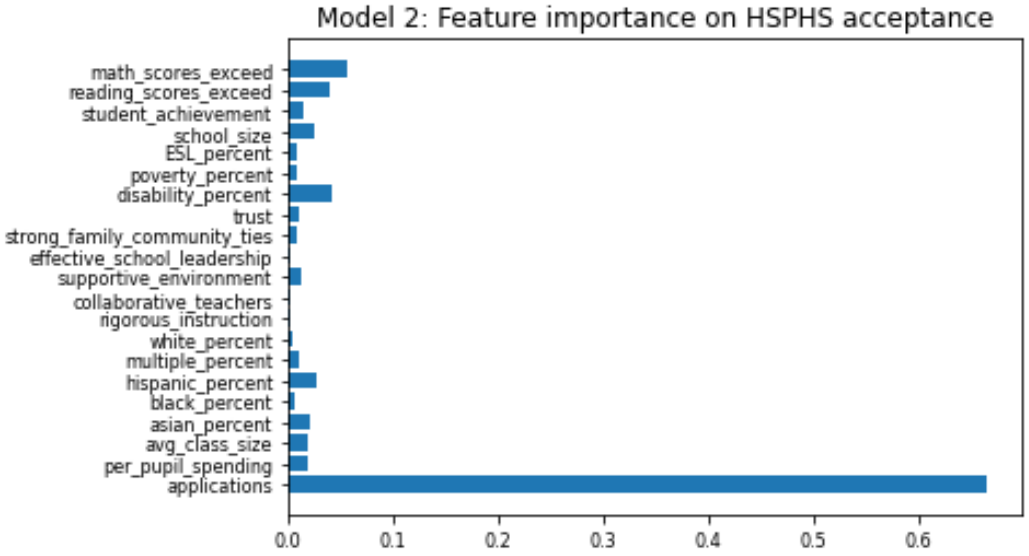
Next step (just like Q4) is to examine the loadings:



Because of the big dimensionality it is impossible for me to comprehend when each loading represent. This is why I decided to build two separate unsupervised models by using two random forrests, one for predicting sending students to HSPHS and one for achievming high scores on objective measures of achievement.

Both models were trained on different training/test sets of data, but with the same proportions (80% training, 20% test). This is done so that we can ultimately explain the feauture importance of the two dependent variables (in the two different models).





When building the models, the n_estimator (number of trees we start with) did not impact the result when ranging from 100-1000 which lead me to run it with 100 just because of improved speed. This is because the reasult are so obvious we will still interpret the model the same regardless of our n_estimator. What our first model show is obvious, that a supportive environment is key for student achievement in

our model. The second model shows us how raw applications is of the highest significance when trying to predict the number of accepted students to HSPHS.

9. There are a lot to be said, one thing to remember is that this is complex data on the real world – if there was an easy fix, this wouldn't be an issue we'd need to resolve. With that being said, it has been showed in the various analyses that the reading scores exceed are key as it is an indicator of student success. The social environment of the students are found the very important, a supportive environment is key for success it seems. It's shown to be more about how the students are feeling rather than pumping the students full of material resources. Altough the resources are important, they're not key – one of the most important findings being a small classroom size, something that intuitevly might mean more 1:1 time with the teacher for the student, which then could improve the the supportive enironment as a whole.

10.

a) From the linear regression and from our model, it is clear that the number of applications to these schools are very important. On paper it is obvious as you can't get in if you don't apply. But the prestigious nature of the HSPHS schools might disencourage students to even apply in the first place. We know that the supportive environment of the student is important, this might mean a higher encouragement from the teachers and surroundings to even considering wanting to go to a HSPHS school. This would lead to wanting to go and eventuelly applying which in turn increases your chances for acceptance from 0% to 0%<.

b) **Hire passionate teachers**. Budgeting towards renovating schools and suppling the teachers/students with the material they need is of course very important. But a student without a **teacher** can't learn, that's really the **bottom line.** I think you need to go one step further and investigate how you might increase the **incentive to become a teacher**, once that's identified you resolve it, whether that means hiring the wages or looking at the core of the schooling system. Hire **passionate teachers** and give them and their students oppurtunity for 1:1 interaction to build important relationships. Lastly, focus on teaching reading to kids at an early age – in this way they might develop a genuine intrest for reading and in worst case they just become good readers (which will set them up well for success further down the line).

Code

```
#Final project Data Science 2021
#Wilhelm von Arndt
#Fall 2021, Intro to Data Science Pascal Wallisch

import os
import numpy as np
from scipy import stats
import pandas as pd
from sklearn import linear_model
import matplotlib.pyplot as plt
import statsmodels.api as sm
from sklearn.decomposition import PCA
from statsmodels.formula.api import ols


os.chdir("/Users/admin/Desktop/Data Science/FinalProject")

data = pd.read_csv("middleSchoolData.csv")

#%%
#FUNCTIONS (written at earlier stage of class)

def nanRemover (index1, index2, dataset):
    counter=0
    outputData =[]
    for ii in dataset[index1]:
        if np.isnan(ii) == True:
            counter+=1
            pass
        elif np.isnan(dataset[index2][counter]) == True:
            counter+=1
            pass
        else:

outputData.append([dataset[index1][counter],dataset[index2][counter]])
            counter+=1
    outputData = np.array(outputData)
    return outputData
#%%
#Question 1
#What is the correlation between the number of applications and
admissions to HSPHS?
x1 = data['applications']
y1 = data['acceptances']
Q1_correlation = data['applications'].corr(data['acceptances'])

plt.scatter(x1, y1)
plt.title('Correlation between Applications and Acceptances')
plt.xlabel('Applications')
plt.ylabel('Acceptances')
plt.plot(np.unique(x1), np.poly1d(np.polyfit(x1, y1,
1))(np.unique(x1)), color='red')
plt.show()

#Correlation of 0.8017, apply more, get more people accepted relation
#%%
#Question 2
```

```
#we want to remove schools with 0 applicants as they would appear as a
0% acceptence school
#replace all 0s in applications with nans and clear rows with nans

apps2 = data['applications'].replace(0,np.nan)
size2 = data['school_size']
accept2 = data['acceptances']
names2 = data['school_name']

df_q2_comb = pd.concat([names2,apps2,size2,accept2],axis=1)
df_q2_comb = df_q2_comb.dropna()

appRate = df_q2_comb['applications']/df_q2_comb['school_size'] #how
many students at a particular school applies.
appRate = appRate.to_frame() #predictor #1

y_admissions = df_q2_comb['acceptances'] #this will be our y a.k.a our
outcome (what we're predicting)
y_admissions = y_admissions.to_frame() #Converting from series to
dataframe for skl

x_rawApp = df_q2_comb['applications']
x_rawApp = x_rawApp.to_frame() #Dataframe with our raw applications,
predictor #2

regr_appRaw = linear_model.LinearRegression()
regr_appRaw.fit(x_rawApp,y_admissions)

appRaw_r_sq = regr_appRaw.score(x_rawApp,y_admissions) #getting the r^2
value of raw applications
#plotting Raw applications and # of admissions

plt.scatter(x_rawApp, y_admissions,  color='black',s=7)
plt.plot(x_rawApp, regr_appRaw.predict(x_rawApp), color='blue',
linewidth=1)
plt.title('Raw admission and accepted students, blue line = linear
regression ')
plt.ylabel('Accepted students')
plt.xlabel('Number of Admissions')
plt.show()
#%%
#Question 2 cont.
#since we have nulls here I'll combine cols,remove nuls, separate them
tempComb = pd.concat([y_admissions,appRate],axis=1)
tempComb = tempComb.to_numpy()
tempComb = tempComb.transpose()
tempComb = nanRemover(0,1,tempComb) #removes nan and outputs a 2d array
that will be usable for our linear regression
tempComb = tempComb.transpose() #transpose for easier indexing, [0]=y;
[1]=x

appRate_nonan = tempComb[1].reshape(-1,1) #reshaping for fitting the
linear regression model
y_admissions_nonan = tempComb[0].reshape(-1,1)

regr_appRatio = linear_model.LinearRegression()
regr_appRatio.fit(appRate,y_admissions)

plt.scatter(appRate_nonan,y_admissions_nonan,color='black',s=7)
```

```
plt.plot(appRate_nonan, regr_appRatio.predict(appRate_nonan),
color='red', linewidth=1)
plt.ylabel('Accepted students')
plt.xlabel('Ratio of students applying/students accepted')
plt.title('Ratio and accepted students, red line = linear regression ')
plt.show()

modelRatio = sm.OLS(y_admissions,appRate).fit()
predictionsRatio = modelRatio.predict(appRate)

print(modelRatio.summary())

modelRaw = sm.OLS(y_admissions,x_rawApp).fit()
predictionsRaw = modelRaw.predict(x_rawApp)

print(modelRaw.summary())

#For application rate
#R-squared (uncentered):                   0.440

#For raw applications
#R-squared (uncentered):                   0.655

#%%
#Question 3
#Best per student odds of sending someone to school
df_admission_rate =
(data['acceptances']/data['school_size']).to_frame() #acceptances over
school size gives us ratio
df_admission_rate.columns = ['admission_rate']
df_school_names = (data['school_name']).to_frame()
df_admissionrate_schools =
(pd.concat([df_school_names,df_admission_rate],
join='outer',axis=1)).dropna()

x=1/0.235
y=x-1
print("Odds are", round(y,2),"to 1")


# Highest admissions rate per students is     school_name THE CHRISTA
MCAULIFFE SCHOOL with the rate of ≈ 23.5%
# 0.235 => 3.26 to 1
#%%
#Question 4

df_perception = data[data.columns[11:17]] #columns L-Q
df_achievement = data[data.columns[21:24]] #columns V-Z

df_contemp = (pd.concat([df_perception,df_achievement],
join='outer',axis=1)).dropna() #removes nan necessary

df_perception_nonan = df_contemp[df_contemp.columns[:6]] #slices
df_achievement_nonan = df_contemp[df_contemp.columns[6:]]

df_perception_performance = df_contemp.copy()

#Creating a correlation matrix

r4 = np.corrcoef(df_perception_performance,rowvar=False)
```

```
plt.rc('xtick', labelsize=8)
plt.rc('ytick', labelsize=8)
plt.title('Correlation table Student Perception and Achievement')
plt.imshow(r4)
plt.colorbar()


#From here we can see a clear correlation aka, we should examine it

#%%
#Question 4 cont...

#DOING PCA for Students perception of the school

zDataPerc = stats.zscore(df_perception_nonan) #normalizing data
pca = PCA()
pca.fit(zDataPerc)

eigValsPerc = pca.explained_variance_

loadingsPerc = pca.components_

rotatedData = pca.fit_transform(zDataPerc)

covarExplained = eigValsPerc/sum(eigValsPerc)*100
numVariables = 6

#Plotting the Screeplot and showing the Kaiser criterion (y=1)
plt.xlabel('Principal component')
plt.ylabel('Eigenvalue')
plt.axhline(y = 1, color = 'r')
plt.bar(np.linspace(1,numVariables,numVariables),eigValsPerc)
plt.title("Eigenvalues for students perception (Col L-Q)")

#%%
#Question 4 cont...
#Plotting the barchart with eigenvalues for student percention

plt.bar(np.linspace(1,6,6),loadingsPerc[:,0]) #0 for 1 with eigenvalue
>1
plt.xlabel('Question')
plt.ylabel('Loading')
plt.title("#Question and Loadings for students perception (Col L-Q)")

#Findings
#postive factors are collaborative teachers and trust
#negative factors are rigorous instructions and effective school
leadership
#%%
#Question 4 cont...
#DOING PCA for Students achievement

zDataPerf = stats.zscore(df_achievement_nonan)
pca = PCA()
pca.fit(zDataPerf)

eigValsPerf = pca.explained_variance_

loadingsPerf = pca.components_
```

```
rotatedData = pca.fit_transform(zDataPerf)

covarExplained = eigValsPerf/sum(eigValsPerc)*100
numVariables = 3

#Plotting the Screeplot and showing the Kaiser criterion (y=1)
plt.xlabel('Principal component')
plt.ylabel('Eigenvalue')
plt.axhline(y = 1, color = 'r')
plt.bar(np.linspace(1,numVariables,numVariables),eigValsPerf)
plt.title("Eigenvalues for students achievement (Col V-X)")
#%%

#Question 4 cont...
#Doing PCA for Student achievement

plt.bar(np.linspace(1,3,3),loadingsPerf[:,0])
plt.xlabel('Question')
plt.ylabel('Loading')
plt.title("Question and Loadings for students achievement (Col V-X)")

#Findings
# reading scores are very important with a loading of -0.8
#%%
#Actually performing the ANOVA
#look at the correlation between student achievement and collaborative
teachers

corr_achievement_collab =
df_perception_nonan['collaborative_teachers'].corr(df_achievement['stud
ent_achievement'])
#For the Anova:
achievement = data['student_achievement']
teachers = data['collaborative_teachers']
trust = data['trust']
reading = data['reading_scores_exceed']
leadership = data['effective_school_leadership']
instruction = data['rigorous_instruction']

postPCAData =
pd.concat([achievement,teachers,trust,reading,leadership,instruction],a
xis=1).dropna()
# postPCAData.columns=newNames

r = np.corrcoef(postPCAData,rowvar=False)
plt.imshow(r)
plt.colorbar()

corrMatrix = postPCAData.corr()
#Here we look all the above variables as main effects, before then
combining them all into one big ANOVA in an attempt to
#grasp the interaction effects between them.
model1 = ols('student_achievement ~ (collaborative_teachers) + (trust)
+ (reading_scores_exceed) + (effective_school_leadership)\
            + (rigorous_instruction) +
(collaborative_teachers):(trust) +
(collaborative_teachers):(reading_scores_exceed) +
(collaborative_teachers):(effective_school_leadership)\
                     + (collaborative_teachers):(rigorous_instruction)
+ (trust):(reading_scores_exceed)\
```

```
                          + (trust):(effective_school_leadership) +
(trust):(collaborative_teachers)\
                              + (trust):(rigorous_instruction) +
(reading_scores_exceed):(effective_school_leadership)\
                                   +
(reading_scores_exceed):(collaborative_teachers) +
(reading_scores_exceed):(rigorous_instruction)\
                                        +
(effective_school_leadership):(collaborative_teachers) +
(effective_school_leadership):(rigorous_instruction) + \

(collaborative_teachers):(rigorous_instruction)',data=postPCAData).fit(
)
anova_table1 = sm.stats.anova_lm(model1, typ=2)
print(anova_table1)

#only one significant P-value and a high residual sum_sq so we'll try
with only the largest aboslut loading values
#%%

smallPostPCAData = postPCAData.copy()
smallPostPCAData =
smallPostPCAData.drop(['rigorous_instruction','effective_school_leaders
hip'],axis=1)

model1 = ols('student_achievement ~ (collaborative_teachers) + (trust)
+ (reading_scores_exceed) + (collaborative_teachers):(trust) +
(collaborative_teachers):(reading_scores_exceed) +\
                       (trust):(reading_scores_exceed) +
(trust):(collaborative_teachers) + \

(reading_scores_exceed):(collaborative_teachers)',data=smallPostPCAData
).fit()
anova_table2 = sm.stats.anova_lm(model1, typ=2)
print(anova_table2)

#We find something more intersting here, teachers combined with trust
aswell as trust reading scores exceed is
#important on student achievement


#%%

#Question 5
#null hypothesis: There is no different in student performance wether
the in a smaller or bigger SCHOOL


sizeTotal = data['school_size']
achievementTotal = data['student_achievement']

sizeAchievementSS =
(pd.concat([sizeTotal,achievementTotal],axis=1)).dropna() #drops na's
no charters

medianSSize = sizeAchievementSS['school_size'].median() #gives us the
media which will be used to sort big/small class
#545.0
```

```
sizeAchievementSize = (sizeAchievementSS.sort_values(by =
'school_size') ).reset_index(drop=True)

achievementBSize = sizeAchievementSize.iloc[:272,:] #slicing
achievementSSize = sizeAchievementSize.iloc[273:,:]

u2,p2 =
stats.mannwhitneyu(achievementSSize['student_achievement'],achievementB
Size['student_achievement'])
#Significant => Difference between performance depending on the school
size


plt.scatter(achievementSSize['school_size'],achievementSSize['student_a
chievement'],color='red',s=7)
plt.scatter(achievementBSize['school_size'],achievementBSize['student_a
chievement'],color='blue',s=7)


plt.ylabel('Student achievement')
plt.xlabel('Size of the school')
plt.show()


plt.hist(achievementBSize['student_achievement'], color = 'blue',
edgecolor = 'black')
plt.title('Big Schools')
plt.ylabel('Size')
plt.xlabel('Achievement')
plt.show()

plt.hist(achievementSSize['student_achievement'], color = 'red',
edgecolor = 'black')
plt.title('Small Schools')
plt.ylabel('Size')
plt.xlabel('Achievement')
plt.show()

sizeschool_achievement_corr =
data['school_size'].corr(data['student_achievement'])
#low correlation
stats.ttest_rel(achievementBSize['student_achievement'],achievementSSiz
e['student_achievement'])
#Shows that there's no significant difference between the both

#%%
#Question 6
#null hypothesis: There is no different in student performance wether
they're in a smaller
#classroom or a bigger


sizeTotal = data['avg_class_size']
achievementTotal = data['student_achievement']

sizeAchievement =
(pd.concat([sizeTotal,achievementTotal],axis=1)).dropna() #drops na's
no charters
```

```
splitValueMed = sizeAchievement['avg_class_size'].median() #gives us
the media which will be used to sort big/small class
#22.2
#We have three classes with avg size of 22.2 so we will exclude these
three of the t-test.

sizeAchievementS = sizeAchievement.sort_values(by = 'avg_class_size')
sizeAchievementS = sizeAchievementS.reset_index(drop=True)

achievementSmall = sizeAchievementS.iloc[230:,:]
achievementBig = sizeAchievementS.iloc[:230,:]

u1,p1 =
stats.mannwhitneyu(achievementSmall['student_achievement'],achievementB
ig['student_achievement'])

#significant difference in achievement between student from small/big
classrooms.
#plot the

plt.hist(achievementBig['student_achievement'], color = 'blue',
edgecolor = 'black')
plt.title('Big Classrooms')
plt.ylabel('Size')
plt.xlabel('Achievement')
plt.show()

plt.hist(achievementSmall['student_achievement'], color = 'red',
edgecolor = 'black',)
plt.title('Small classrooms')
plt.ylabel('Size')
plt.xlabel('Achievement')
plt.show()

sizeAchieveRegr =
(data[['student_achievement','avg_class_size']]).dropna()

modelRaw =
sm.OLS(sizeAchieveRegr['student_achievement'],sizeAchieveRegr['avg_clas
s_size']).fit()
predictionsRaw = modelRaw.predict(sizeAchieveRegr['avg_class_size'])
#there's a difference, R^2 of 0.933
print(modelRaw.summary())
#%%
#Check if there's a correlation between school size and the student
achievement


regr_achievement_class = linear_model.LinearRegression()
regr_achievement_class.fit(x_rawApp,y_admissions)

appRaw_r_sq = regr_appRaw.score(x_rawApp,y_admissions) #getting the r^2
value of raw applications
#plotting Raw applications and # of admissions

plt.scatter(x_rawApp, y_admissions,  color='black',s=7)
plt.plot(x_rawApp, regr_appRaw.predict(x_rawApp), color='blue',
linewidth=1)
plt.xticks(())
plt.yticks(())
```

```
plt.ylabel('# of accepted students')
plt.xlabel('# of raw admissions')
plt.show()


#%%

#What proportion of schools accounts for 90% of accepted students to
HSPHS?

raw_accept = data['acceptances']
raw_accept = raw_accept.to_numpy() #convert to np
raw_accept = sorted(raw_accept,reverse=True) #Sort in decending order

total_accept = sum(raw_accept)
accept_90 = 0.9*total_accept #check how many student account for 90%

for i in raw_accept: #Just checking if we have nans, we do not.
    if np.isnan(i) == True:
        print("NAN")

tempCounter = 0
loopCounter = 0
for i in raw_accept: #simple for loop counting and checking if the
total students are equivalent of the # of "90%"
    if tempCounter>=accept_90:
        break
    else:
        tempCounter+=i
        loopCounter+=1

schools_number = len(raw_accept)

proportion = loopCounter/schools_number

# Roughly 20% of schools make up for 90% of acceptances to HSPHS
# 124 schools up until index 123

accept = data['acceptances']

acceptNames = pd.concat([names2,accept],axis=1)
acceptNames = (acceptNames.sort_values(by =
'acceptances',ascending=False).reset_index(drop=True))
acceptNames = acceptNames.iloc[:124,:]

x_pos = np.arange(len(acceptNames))

plt.bar(x_pos,acceptNames['acceptances'],color=(0.5,0.1,0.5,0.6),width=
0.7)
plt.xticks(x_pos,acceptNames['school_name'])
plt.xticks(())
plt.title('Distribution of 90% of students accepted to HSPHS')
plt.xlabel('Schools')
plt.ylabel('Number of accepted students')
plt.show()

#%%
#Building our model
#importing libraries needed for unsupervised models.
from sklearn.decomposition import PCA
```

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestRegressor
from sklearn import metrics

nonanData = data.dropna()


yOutcomes = (nonanData['student_achievement']).to_numpy()
predictors = nonanData
predictors = (predictors.drop(['dbn','school_name'],axis=1)) #drop
student_achievement later! 'applications','acceptances'

predictors, testSet = train_test_split(predictors, test_size=0.2)
#split between training and test set

yOutcomes = (predictors['student_achievement']).to_numpy()
yOutcomesTraining = (testSet['student_achievement']).to_numpy()

predictors = (predictors.drop(['student_achievement'],axis=1))
testSet = (testSet.drop(['student_achievement'],axis=1)).to_numpy()
predictorsNames = list(predictors.columns.values)
predictors = predictors.to_numpy()

#amnipulating the testSet

pcaTest = PCA()
pcaTest.fit(testSet)

eigValues1 = pcaTest.explained_variance_
loadings1 = pcaTest.components_ # sorted by explained_variance_
coordinatesTestData = pcaTest.fit_transform(testSet)

XTest =
np.transpose(np.array([coordinatesTestData[:,0],coordinatesTestData[:,1
],coordinatesTestData[:,2],coordinatesTestData[:,3]]))

r = np.corrcoef(predictors,rowvar=False)
plt.imshow(r)
plt.colorbar()

#Now we run a PCA

pca1 = PCA()
pca1.fit(predictors)

eigValues = pca1.explained_variance_
loadings = pca1.components_ # sorted by explained_variance_
origDataNewCoordinates = pca1.fit_transform(predictors)

#%%
#Creating and plotting Screeplot after normalizing our data

numPredictors = 21 #19
zscoredData = stats.zscore(predictors)
pca1.fit(zscoredData)
eigValues = pca1.explained_variance_
loadings = pca1.components_
origDataNewCoordinates = pca1.fit_transform(zscoredData)
```

```
plt.bar(np.linspace(1,numPredictors,numPredictors),eigValues)
plt.title('Scree plot')
plt.xlabel('Principal Components')
plt.ylabel('Eigenvalues')
plt.axhline(y = 1, color = 'r')

# Four meaningful predictors for student achievement

#%%
#Loadings for column 1
plt.bar(np.linspace(1,21,21),loadings[:,0])
plt.xlabel('Question')
plt.ylabel('Loading')
plt.title("Question and Loadings #1 ")

#Black percentage, white percentage, strong family community ties.
Black and white kids from influential families..?

#%%

plt.bar(np.linspace(1,21,21),loadings[:,1])
plt.xlabel('Question')
plt.ylabel('Loading')
plt.title("Question and Loadings #2 ")

# Collaborative teachers HUGE - this is good teaching, we have the next
value as rigorous instruction (~-0.9)

#%%

plt.bar(np.linspace(1,21,21),loadings[:,2])
plt.xlabel('Question')
plt.ylabel('Loading')
plt.title("Question and Loadings #3 ")

# Hispanic is huge(~0.8), reading comes in second (~0.4 )

#%%

plt.bar(np.linspace(1,21,21),loadings[:,3])
plt.xlabel('Question')
plt.ylabel('Loading')
plt.title("Question and Loadings #4 ")

#asian and black both have ~ 0.5
#%%
plt.bar(np.linspace(1,21,21),loadings[:,4])
plt.xlabel('Question')
plt.ylabel('Loading')
plt.title("Question and Loadings #5 ")


#%%
#model for predicting student achievement

y = (nonanData['student_achievement'])
X = nonanData
X = (X.drop(['dbn','school_name','student_achievement'],axis=1)) #drop
student_achievement later!
```

```
xTrain, xTest, yTrain, yTest = train_test_split(X,y, test_size=0.2)
#split between training and test set

scaler = StandardScaler() #Scaling the different columns as the
measured units differ from col to col.
xTrain = scaler.fit_transform(xTrain)
xTest = scaler.fit_transform(xTest)

#Training the model

regrM = RandomForestRegressor(n_estimators=100,random_state = 200)
regrM.fit(xTrain, yTrain)
print(regrM.feature_importances_)
plt.barh(X.columns, regrM.feature_importances_)
plt.title('Model 1: Feature importance on student achievement')

yPred = regrM.predict(xTest)

print('Mean Absolute Error:', metrics.mean_absolute_error(yTest,
yPred))
print('Mean Squared Error:', metrics.mean_squared_error(yTest, yPred))
print('Root Mean Squared Error:',
np.sqrt(metrics.mean_squared_error(yTest, yPred)))


#%%
#Model predicting sending students to HSPHS


y = (nonanData['acceptances']) #using this as this was proven to be a
better predictor earlier in OLS
X = nonanData
X = (X.drop(['dbn','school_name','acceptances'],axis=1)) #drop
student_achievement later! 'applications'

xTrain, xTest, yTrain, yTest = train_test_split(X,y, test_size=0.2)
#split between training and test set

scaler = StandardScaler() #Scaling the different columns as the
measured units differ from col to col.
xTrain = scaler.fit_transform(xTrain)
xTest = scaler.fit_transform(xTest)

#Training the model

regrM = RandomForestRegressor(n_estimators=40,random_state = 20)
regrM.fit(xTrain, yTrain)
print(regrM.feature_importances_)
plt.barh(X.columns, regrM.feature_importances_)
plt.title('Model 2: Feature importance on HSPHS acceptance')


#Acceptances play a huge roll, it skews our model
```