Fail, Pass, Succeed: Bob Edition

STAT 195: Python for Data

Tuan Dang and Emeka Nkuku

November 20th 2021

1 Overview:

We have a dataset of past students who took STAT 2103 (Introductory Statistics). We will use this dataset to build a model and make predictions of the performance of future students. We were able to build an SVM model of best accuracy at 73%.

Our goal is to predict the performance of Bob in this course. The features of the dataset as well as information about Bob will be discussed in the following sections.

2 The Data:

2.1 Dataset features:

The dataset contains records of 643 students, where the columns are their academic information. There are 9 features in total from the original dataset, one of which being our dependent variable:

Numeric variables:
gpa: the student's GPA
reading: SAT Reading Score
math: SAT Math Score
composite: SAT Score in total
ACTcomposite: ACT Score in total

Categorical variables:
gender: Female and Male
encoded_gender: 0 - Female, 1 - Male

level: the student's class year in college
encoded_level: 0 - Freshman, 1 - Junior, 2 - Senior, 3 - Sophomore

highschool: the student's high school background
encoded_highschool: 0 - Home School, 1 - Private High School, 2 - Public High School, 3 - Transfer

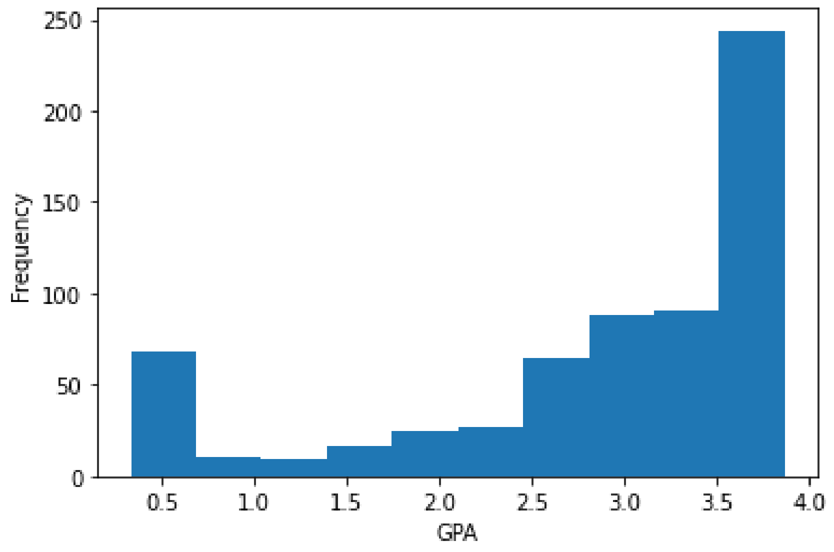
Dependent variable:
STAT2103: the student's overall performance in the class - 'Fail', 'Pass', 'Succeed'

Bob's data:

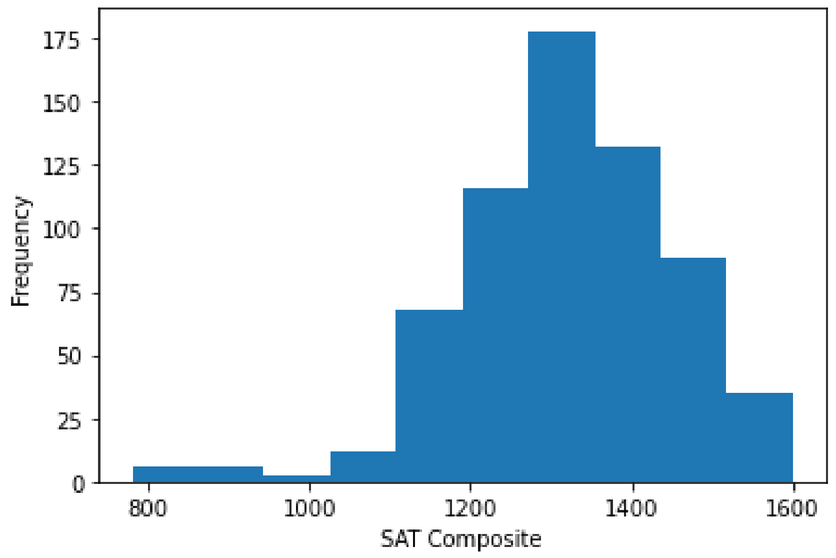
GPA: 2.33	Gender: Male
SAT Reading: 750	Class Year: Senior
SAT Math: 720	High School: Private
SAT Composite: 1470	ACT Composite: 25

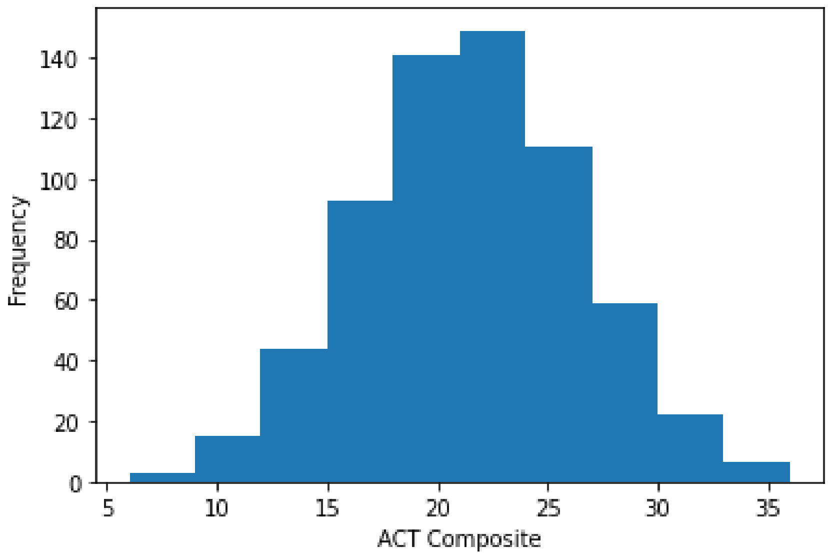
2.2 Descriptive statistics:

We will take a closer look at our data. This will give us some clues as to how we should build our model as well as some assumptions to look out for.

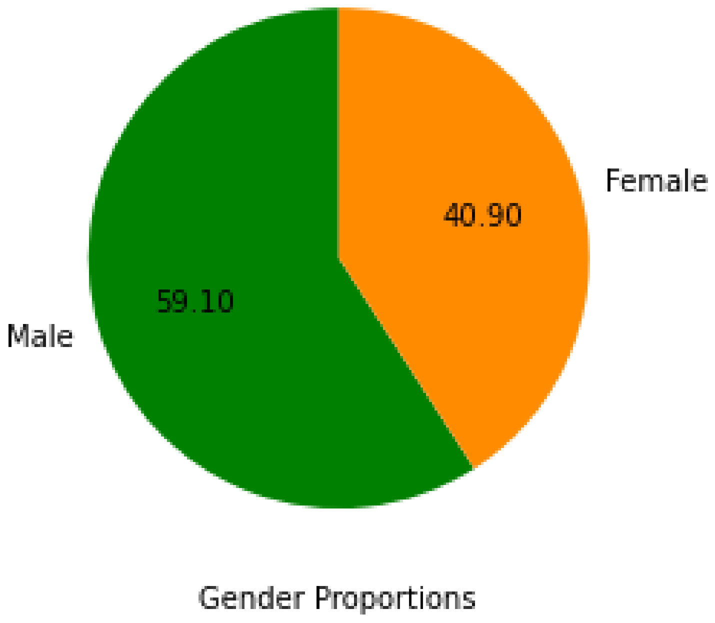


As we can see from this histogram of the distribution of GPA, the data set is definitely skewed with this variable. Most students seem to be on the higher end in GPA, which is definitely worth noting since this may affect the predictive power of this variable when we use it to train our model.

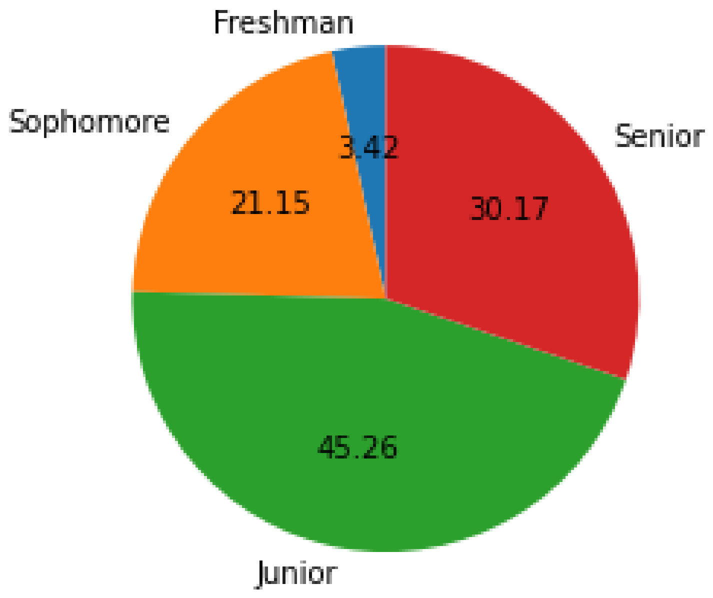




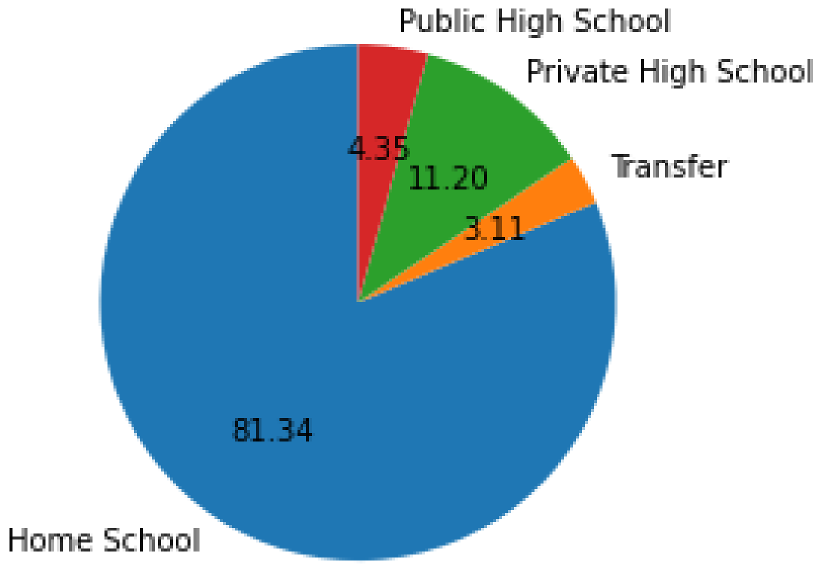
SAT and ACT scores seem to follow a somewhat Normal distribution, where SAT is more skewed than ACT.



Almost 60% of the subjects in the dataset is male, which is important to note since this may affect the model and cause gender to be significant later on. This is similar to Class Year and High School Background. Almost half of the students are of Junior level, and 75% are upperclassmen, which could mean this course is a higher level course and requires some prerequisites before enrollment. Surprisingly, up to 81% of the students in the data set were homeschooled in highschool, which might not be very good for the variable.

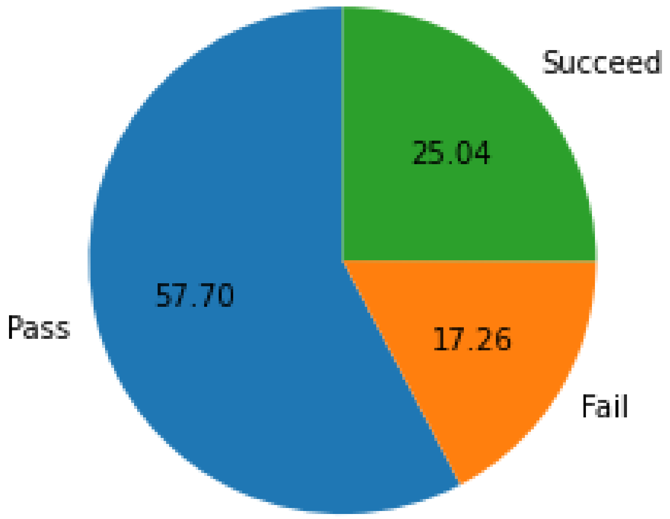


Proportion of Class Year



Proportion of Highschool Background

Last but not least, our dependent variable, performance in STAT 2103, seems normal. We should expect most students to pass the class, while some succeed and fail on two ends of the spectrum.



Proportion of Student Performance in STAT 2103

3 Part 1: Base Model

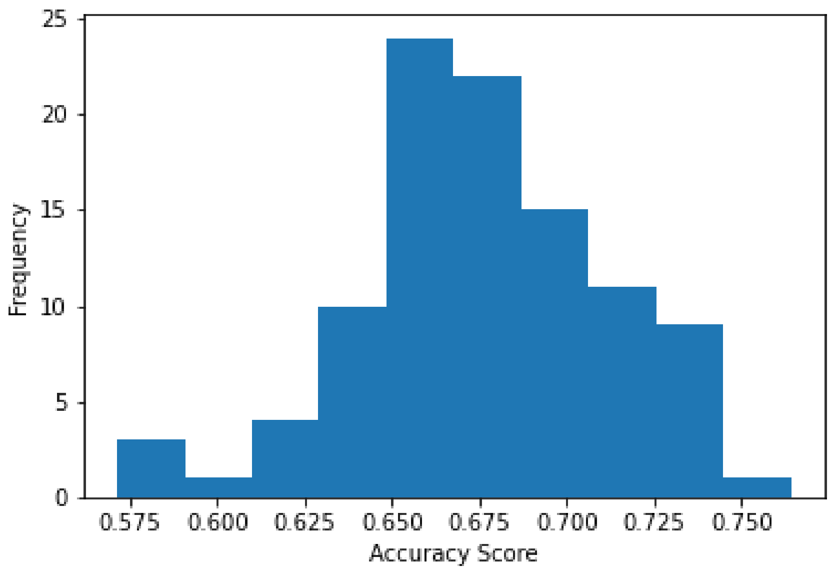
We decided to stick to the SVM model for the scope of this course, although there is plenty of room to experiment with other types of models as well, which might produce better accuracy than SVM.

Apparently, the SVM model only takes in numeric values, so before we can build the model, we will need to encode the categorical variables into numeric values.

In our first model, we will include all the independent variables to get a baseline. Later on, for our second model, we will figure out which variables will have the most significance to the model, and also which variables are insignificant enough to be discarded.

We will split the dataset into two sets for training and testing. The testing dataset will consist of 25% of the data. This will allow us to use the training set to build the model, and the testing set to test its accuracy.

We can get the accuracy score of a model using the score() function from the scikit-learn library, but this is not enough to get a valid interpretation of the true accuracy since there is a randomness component to it. We decided to address this by repeatedly calculating the accuracy score for 100 times using a for loop and calculating the mean of that instead.



Histogram of the base model's accuracy score

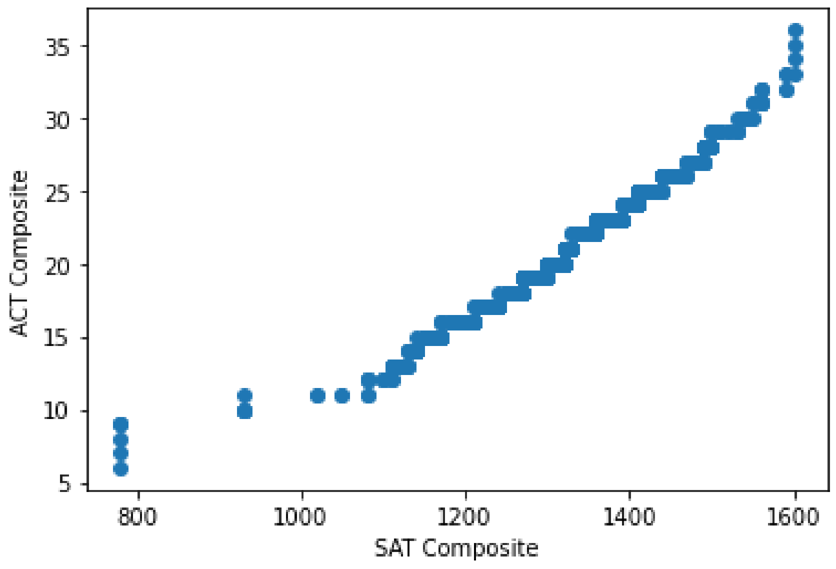
The accuracy of this base model comes out to be around 67%. This model predicts that Bob will succeed in this Statistics course.

4 Part 2: Improved Model

Now, we will try to take a deeper look at the variables and their relationship with one another to figure out what variables we should keep in the model. This can be done using the Wald test, but we couldn't find a good way to do this in Python. As an alternative, we will use some graphs to show relationships of variables that are correlated, and also a process of trial-and-error to test which combinations of variables produce the best model.

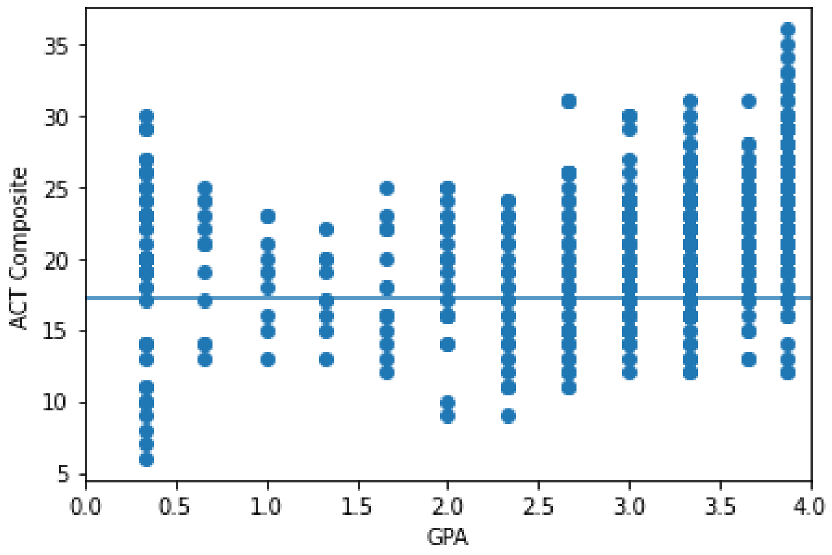
A key that we want to focus on is the relationship between 'math', 'reading', and 'composite.' Since 'composite' is just the sum of 'math' and 'reading,' we probably only need one of the two in the model since they give us the same information.

Next, we also compared 'composite' (or 'math' and 'reading') to 'ACTcomposite,' since we should expect them to be highly correlated. We wanted to see between the SAT and the ACT which one would be better for the model (or perhaps both of them can be included). As it turns out, the SAT is actually not a very good predictor of success in this course, while the ACT proves to be very significant. This is why we decided to remove the SAT completely from the model while keeping the ACT.



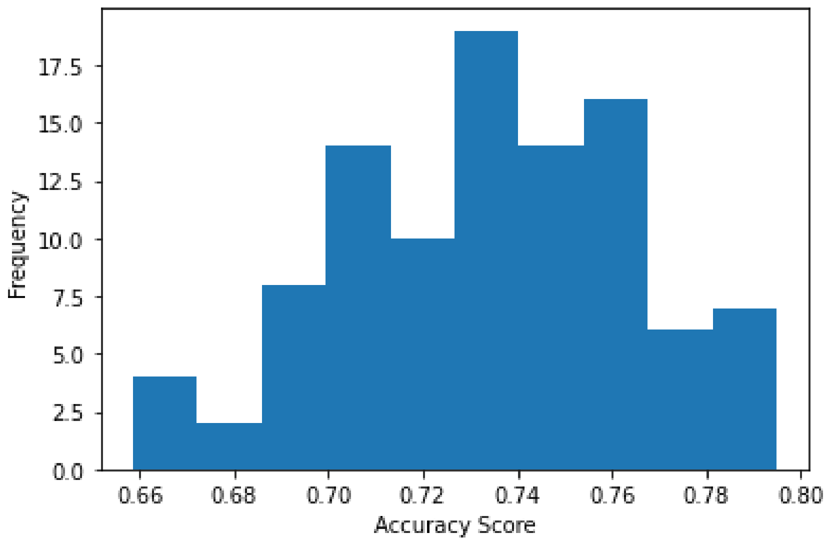
Correlation between SAT composite and ACT composite

We will also compare 'gpa' and 'ACTcomposite,' since they both fall on the same line of measuring a student's academic performance. It doesn't seem like they are correlated at all based on the scatter plot below, so we will include both of them in the model.



Correlation between GPA and ACT composite

Similarly, through more trial-and-error, we found the SVM model with best accuracy of 73% which only uses three independent variables: GPA, gender, and ACT score. This model predicts that Bob will pass this course, but not succeed in it like what we found with the base model.



Histogram of the improved model's accuracy score

5 Accuracy, Uncertainty, and Future Improvements:

As mentioned previously, the analysis for these models would be better if we could use the Wald test. A closer look into the data that we have available might also be useful. For example, 60% of the subjects from the dataset were Male, which could play a part in the significance of gender on the model. Regardless, we managed to do a decent job and found a model with moderately good accuracy.

I believe the model can be even better if we have a more expansive dataset. For example, if this Statistics course has other courses as its prerequisites, then it could be helpful to measure them as well.

Lastly, the SVM model was used for this project, but as stated earlier, we can continue to improve the results by experimenting with other classification models.

6 Data Source:

$\langle https://courses.kvasaheim.com/stat195/project/forsbergCollege.csv\rangle$

7 Python Code Appendix:

Part 1: Base Model

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

from scipy.cluster.vq import kmeans, vq, whiten
from sklearn.model_selection import train_test_split
import sklearn as skl
from sklearn import svm
from numpy import arange

import scipy.stats as ss




### Load data
url = "https://courses.kvasaheim.com/stat195/project/forsbergCollege.
    csv"
dt = pd.read_csv(url)
#print(dt.head)     ## inspect the data

n = dt.shape[0]     ## number of rows / sample size
#print(n)


### Encode categorical data into numerical values
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()
gender_encoded = le.fit_transform(dt['gender'])
dt['encoded_gender'] = gender_encoded
#print(gender_encoded)

level_encoded = le.fit_transform(dt['level'])
dt['encoded_level'] = level_encoded
#print(level_encoded)

highschool_encoded = le.fit_transform(dt['highschool'])
dt['encoded_highschool'] = highschool_encoded
#print(highschool_encoded)

#encoded_gender: 0 - Female, 1 - Male
#encoded_level: 0 - Freshman, 1 - Junior, 2 - Senior, 3 - Sophomore
#encoded_highschool: 0 - Home School, 1 - Private High School, 2 -
    Public High School, 3 - Transfer



### Sample Statistics
#print(dt.loc[:, ['gpa','reading','math','composite','ACTcomposite']].
    describe())
```

```
#print(dt.loc[:, ['encoded_gender','encoded_level','encoded_highschool
    ']].describe())
#print(dt.loc[:, ['gender','level','highschool','STAT2103']].describe()
    )
#print(dt['gender'].unique())
#print(dt['level'].unique())
#print(dt['highschool'].unique())
#print(dt['STAT2103'].unique())



### Base Model - includes all independent variables
dx = dt.loc[:, ['gpa','reading','math','composite','encoded_gender','
    encoded_level','encoded_highschool','ACTcomposite']]
#print(dx.head)         # checks the first few values of dx
tt = dt['STAT2103']     # target / dependent variable



### Do SVM
# split the dataset into train and test sets
X_train, X_test, y_train, y_test = train_test_split(dx, tt, test_size
     = 0.25)
svmModel = svm.SVC()
svmModel.fit(X_train, y_train)
#print(svmModel.score(X_test, y_test))      # accuracy score



### Prediction
print(svmModel.predict([[2.33,750,720,1470,1,2,1,25]]))



### Create histogram of accuracy scores
preds = []

for i in range(100):
    X_train, X_test, y_train, y_test = train_test_split(dx, tt, test_
    size = 0.25)
    svmModel = svm.SVC()
    svmModel.fit(X_train, y_train)
    preds.append(svmModel.score(X_test, y_test))

plt.hist(preds)
plt.xlabel('Accuracy Score')
plt.ylabel('Frequency')
plt.show()

print(np.mean(preds))       # take average of the accuracy scores
```

Part 2: Improved Model

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

from scipy.cluster.vq import kmeans, vq, whiten
from sklearn.model_selection import train_test_split
import sklearn as skl
from sklearn import svm
from numpy import arange

import scipy.stats as ss




### Load data
url = "https://courses.kvasaheim.com/stat195/project/forsbergCollege.
    csv"
dt = pd.read_csv(url)
#print(dt.head)         ## inspect the data

n = dt.shape[0]         ## number of rows / sample size
#print(n)



### Encode categorical data into numerical values
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()
gender_encoded = le.fit_transform(dt['gender'])
dt['encoded_gender'] = gender_encoded
#print(gender_encoded)

level_encoded = le.fit_transform(dt['level'])
dt['encoded_level'] = level_encoded
#print(level_encoded)

highschool_encoded = le.fit_transform(dt['highschool'])
dt['encoded_highschool'] = highschool_encoded
#print(highschool_encoded)


### Sample Statistics
#print(dt.loc[:, ['gpa','reading','math','composite','ACTcomposite']].
    describe())
#print(dt.loc[:, ['encoded_gender','encoded_level','encoded_highschool
    ']].describe())


### Graphs
#plt.scatter(dt['composite'], dt['ACTcomposite'])
#plt.xlabel('SAT Composite')
#plt.ylabel('ACT Composite')
```

```
#plt.xlim(0,4)

#plt.scatter(dt['gpa'], dt['ACTcomposite'])
#m, b = np.polyfit(dt['gpa'], dt['ACTcomposite'], 1)
#print(m)
#plt.plot(m*dt['gpa']+b)
#plt.xlabel('GPA')
#plt.ylabel('ACT Composite')
#plt.show()

#ols = ss.linregress(dt['gpa'], dt['ACTcomposite'])
#print(ols[0])



### Model with best accuracy
dx = dt.loc[:, ['gpa','encoded_gender','ACTcomposite']]

#print(dx.head)
tt = dt['STAT2103']



### Do SVM
# split the dataset into train and test sets
X_train, X_test, y_train, y_test = train_test_split(dx, tt, test_size
     = 0.25)
svmModel = svm.SVC()
svmModel.fit(X_train, y_train)
#print(svmModel.score(X_test, y_test))      # accuracy score



### Prediction
print(svmModel.predict([[2.33,1,25]]))



### Create histogram of accuracy scores
preds = []

for i in range(100):
    X_train, X_test, y_train, y_test = train_test_split(dx, tt, test_
    size = 0.25)
    svmModel = svm.SVC()
    svmModel.fit(X_train, y_train)
    preds.append(svmModel.score(X_test, y_test))

plt.hist(preds)
plt.xlabel('Accuracy Score')
plt.ylabel('Frequency')
plt.show()

print(np.mean(preds))       # take average of the accuracy scores
```