Data Mining Project Report

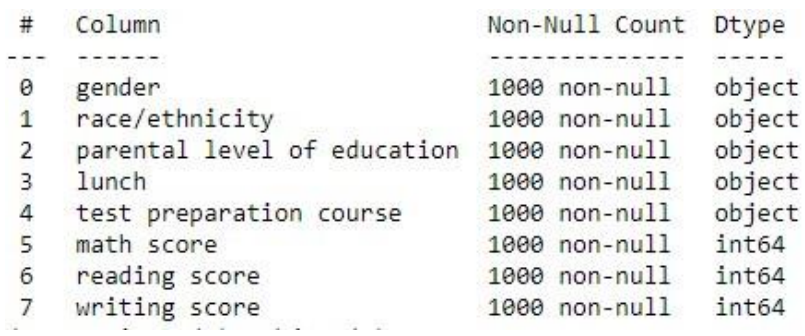
Topic: Student performance evaluation

Section: A

Submitted to: Dr.Waseem Shehzad

The project was about Student performance evaluation. We started the project by finding a suitable dataset which not only had numerical data (subject score), but also other type of string data (parental level of education, lunch types, etc.). We chose this data because now we not only had number but also some external forces which could help us build a suitable model.

The columns in our dataset.

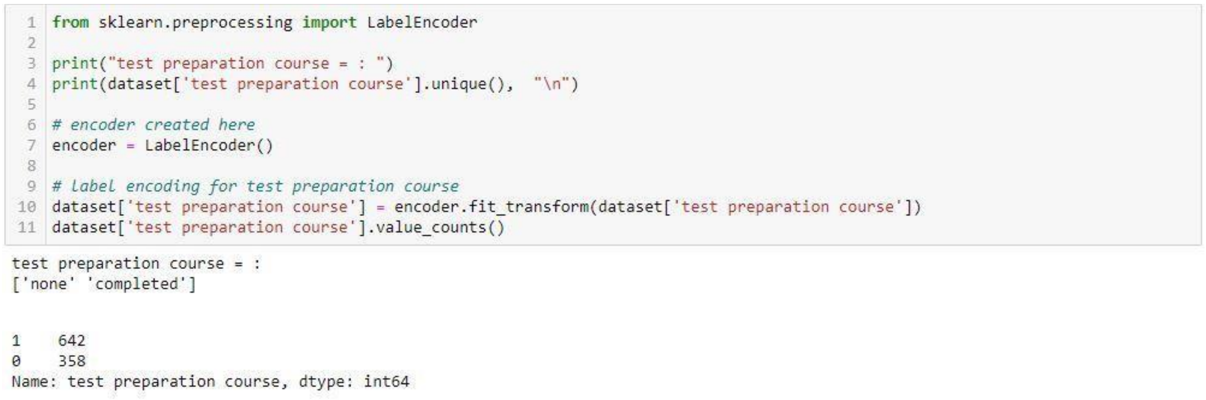


```
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0   gender                       1000 non-null   object
 1   race/ethnicity               1000 non-null   object
 2   parental level of education  1000 non-null   object
 3   lunch                        1000 non-null   object
 4   test preparation course      1000 non-null   object
 5   math score                   1000 non-null   int64
 6   reading score                1000 non-null   int64
 7   writing score                1000 non-null   int64
```

We started by importing our data into a dataframe called ‘dataset’. We checked the shape of the data, followed by the information of the ‘dataset’ which tells us the columns data types. We then checked if our columns had null values or not, which we did not have. Because of this we were not able to perform data imputation.

We then added a new column called ‘class’. This column basically told us whether the student had passed or failed to a given criteria. The criteria which we had chosen was that the student had to have greater than 50 score in all 3 subjects to have passed.

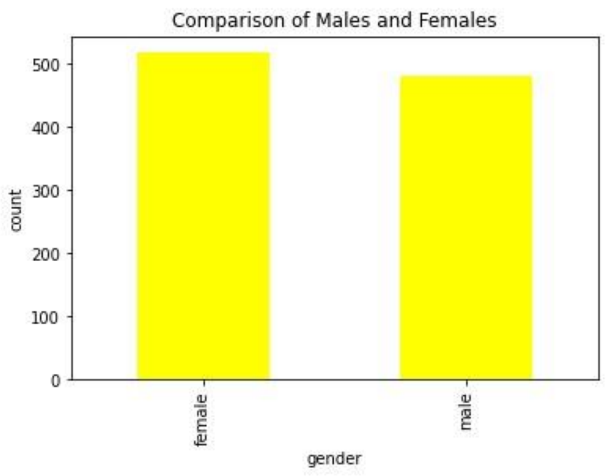
We found that there were a few columns which had string data, so we left that as it is till the end where we used ‘Label encoder from SKLearn’ and gave labels to our data columns so that they were now also in numerical format. The following is the encoder code for the columns test preparation course.



```python
from sklearn.preprocessing import LabelEncoder

print("test preparation course = : ")
print(dataset['test preparation course'].unique(),  "\n")

# encoder created here
encoder = LabelEncoder()

# label encoding for test preparation course
dataset['test preparation course'] = encoder.fit_transform(dataset['test preparation course'])
dataset['test preparation course'].value_counts()
```

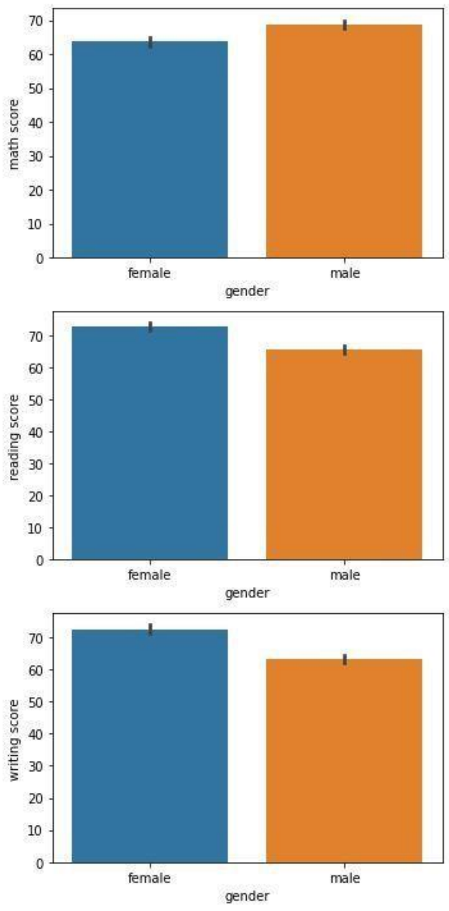
```
test preparation course = :
['none' 'completed']


1    642
0    358
Name: test preparation course, dtype: int64
```

Before this we displayed a few graphs to help us check that which column are to be dropped and which are to be taken. To check the number of male and female students, so that we knew they were almost equal and the data was not imbalanced.

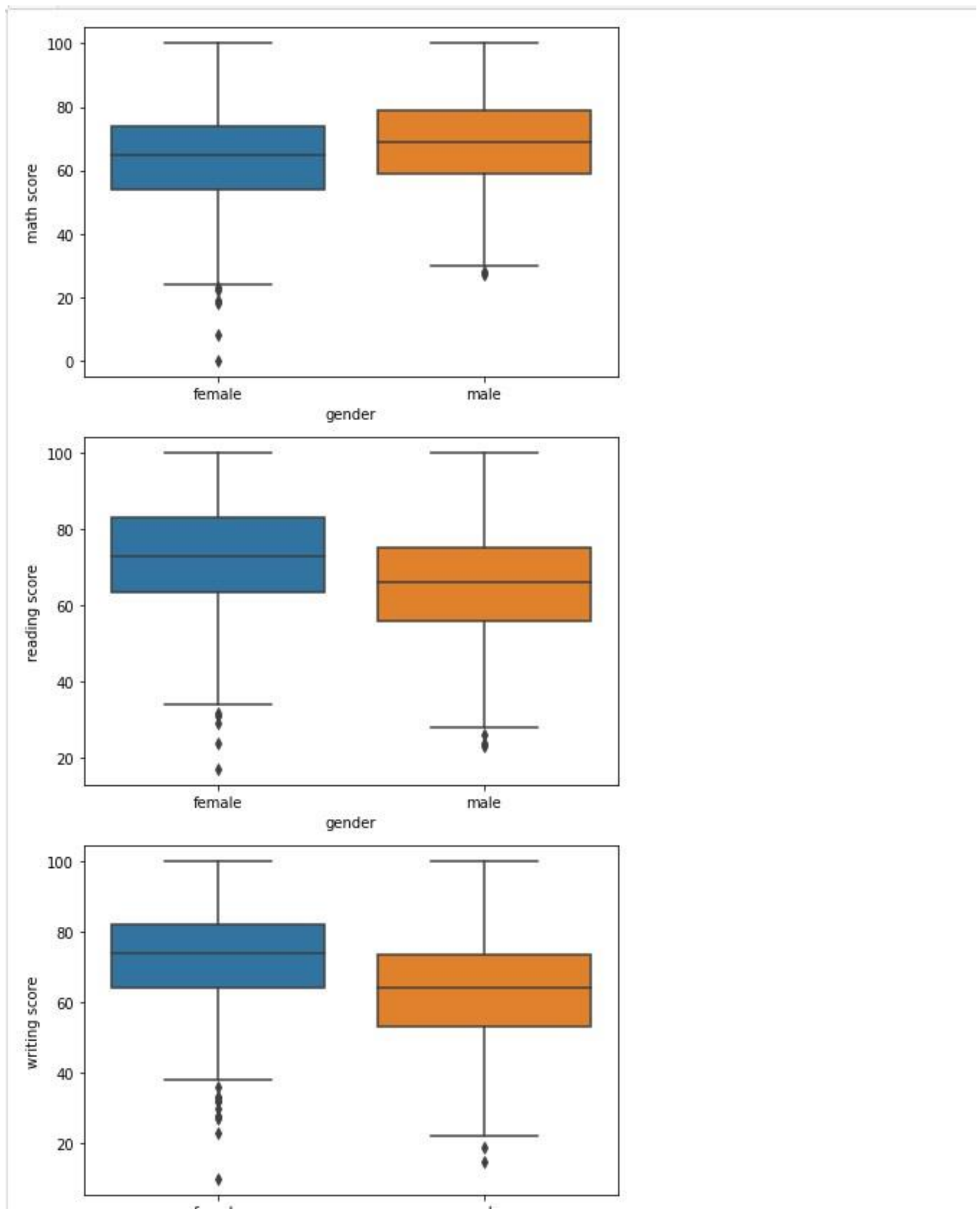


Scores of math’s, writing and reading were cross-referenced with male and female students.

```
# In the following plot we see that the male students have performed well in the math, and females have performed
# well in reading and writing
```

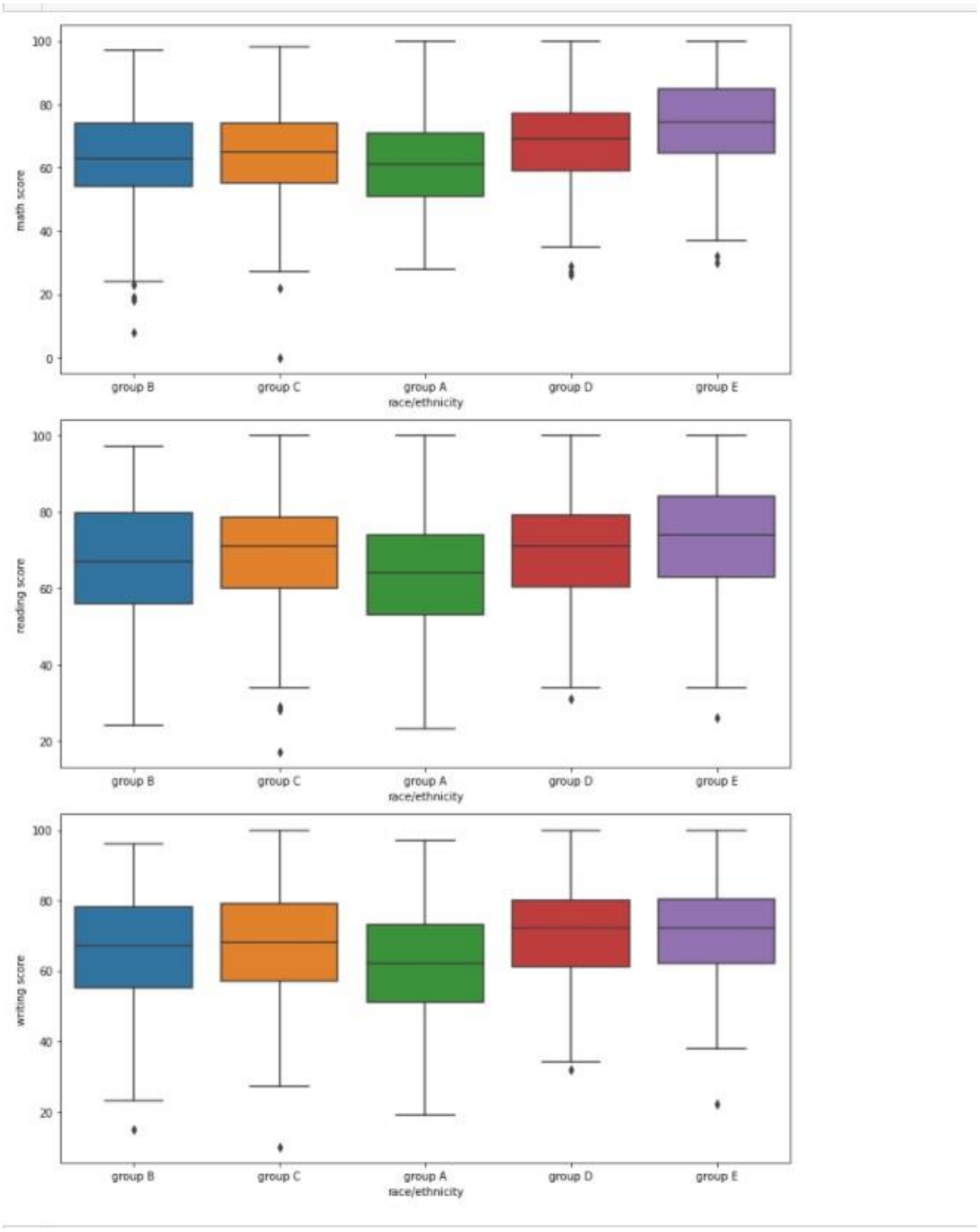


Box Pot of scores of math’s, writing and reading with male and female students was plotted to check who had done well.

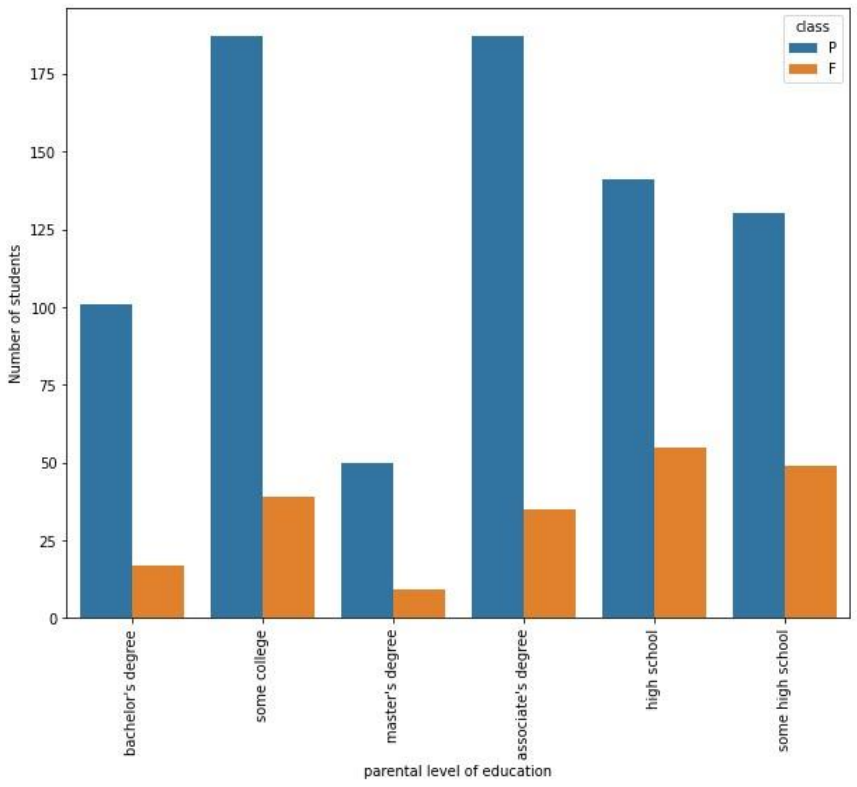


Here above we can see the outlier with the median in all 3 subjects. This can be compared to above plot, which shows that males have performed will in maths and females have performed well in reading and writing, with less outliers.

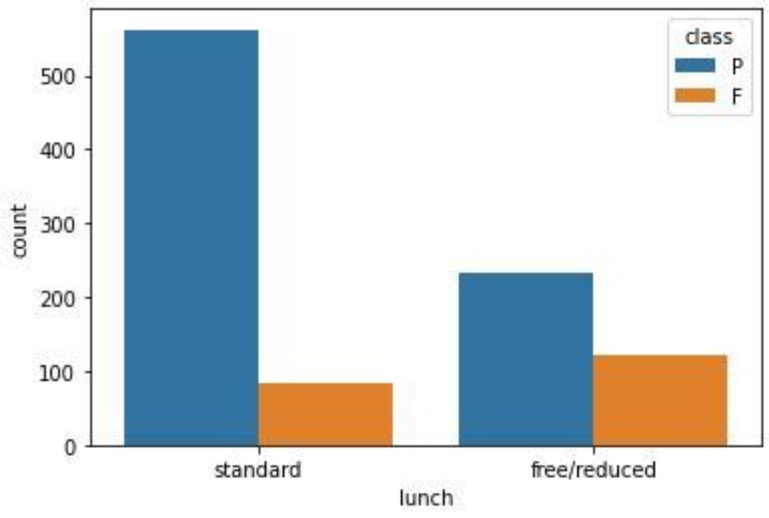
Then, boxplot of ethnicity with scores was performed. This showed that ethnicity also was a factor for different scores of students in each subject.



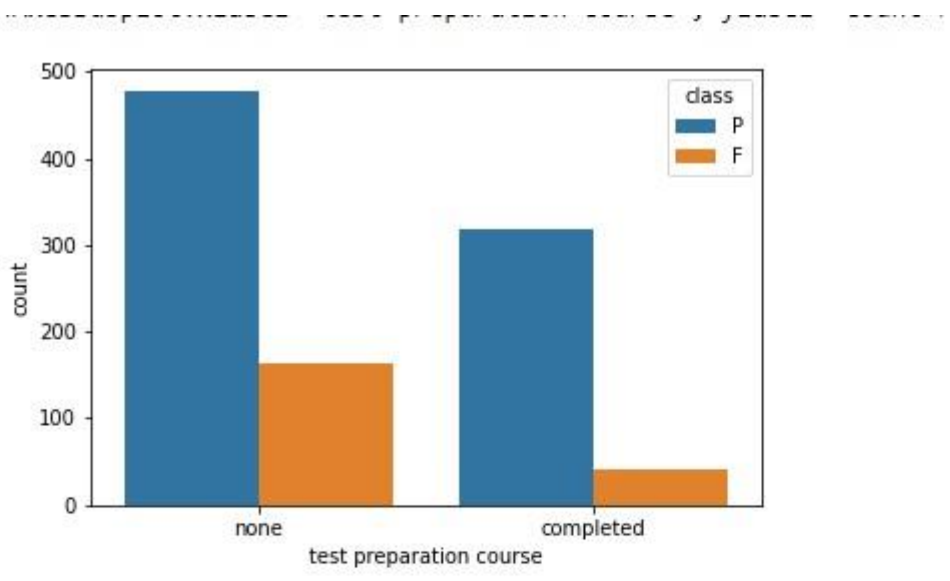
Then we plotted parental level of education against 'class' column to check whether the level of education of parents impacted the number of students who had passed or failed. This showed us that there was a big difference so this could also be a possible factor of student performance evaluation.



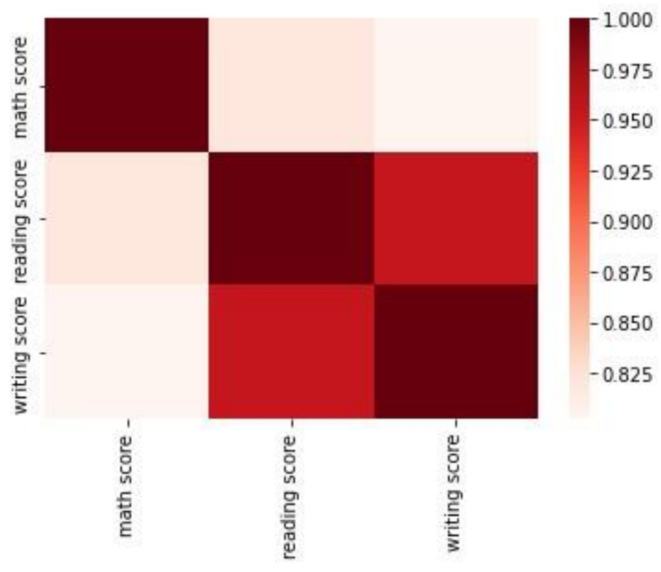
Then, Lunch was plotted against ‘class’ column. This tells that lunch also affected student grades.



Then, test preparation course was plotted against ‘class’ column.



After examining the above graphs, we thought of our current numerical data, which are the 3 columns (math score, writing score, reading score) were correlated or not, for that we used correlation heat map.

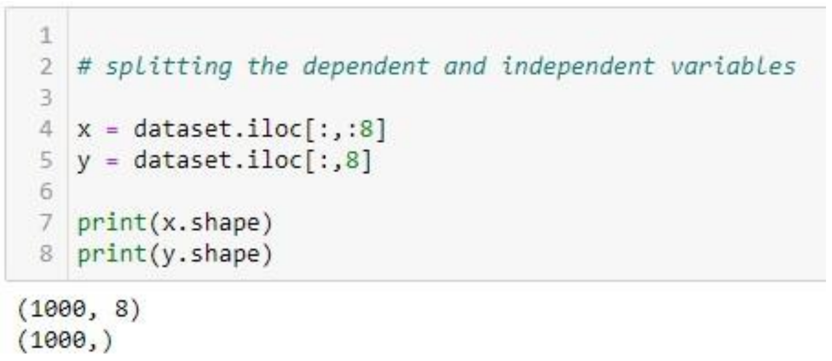


The heat map showed us that indeed our 3 columns were correlated and now we moved onto our model.

Here, we first encoded our model, as told earlier in the report. We then split our dataset to

1. x values (all the columns which we need to predict)
2. y values(the column which we have to predict) For our model which would be created later.

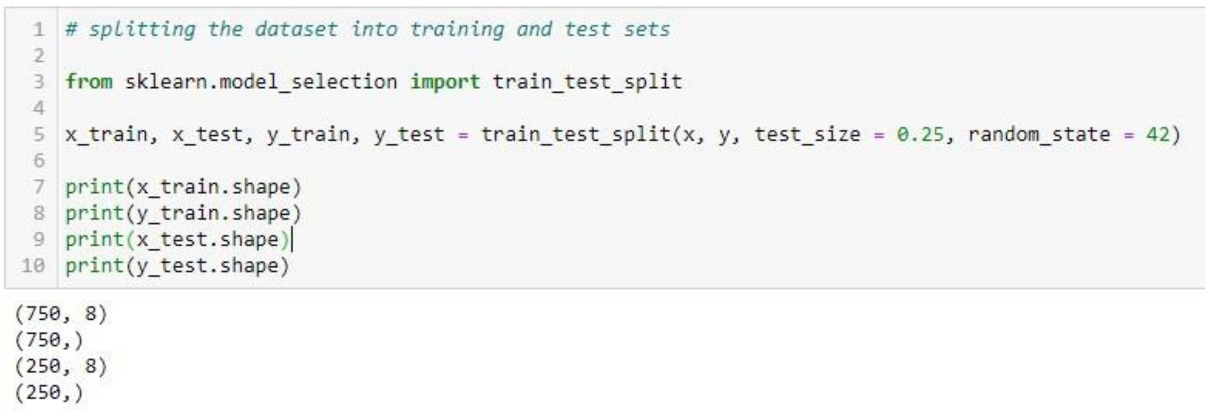
The following is the split code.



```
# splitting the dependent and independent variables

x = dataset.iloc[:,:8]
y = dataset.iloc[:,8]

print(x.shape)
print(y.shape)
```

```
(1000, 8)
(1000,)
```

Now we use 'train_test_split from sklearn' to split our data into training data and testing data. Training data would be used to train our model, and testing data would be used to test our model.



```
# splitting the dataset into training and test sets

from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.25, random_state = 42)

print(x_train.shape)
print(y_train.shape)
print(x_test.shape)
print(y_test.shape)
```

```
(750, 8)
(750,)
(250, 8)
(250,)
```

After splitting our data we used min-max scaler to convert our data to a single scale, because our encoder converted data into 0 and 1 as our data had only 2 strings, then if we see our 3 initial numerical columns, we see that they are out of 100, so we used min-max scaler to convert our data into a same scale.

After this, we use 3 model to train and predict data, we analyze them and check which model is better suited.

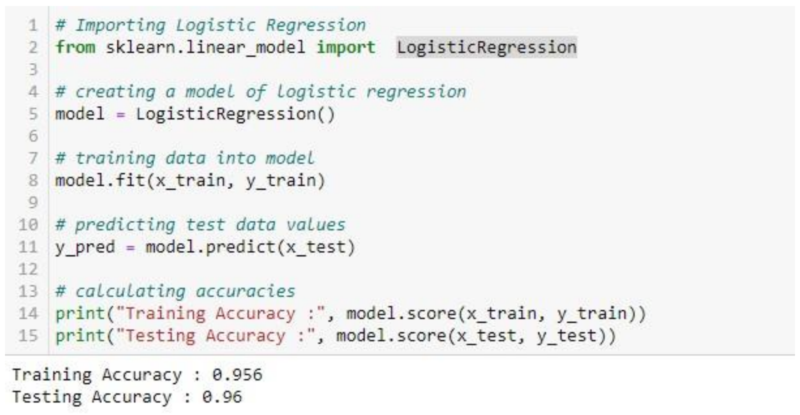
The 3 models were:

1. Logistic Regression
2. Random Forest Classifier
3. Decision Tree Classifier

From these 3 classifier Decision Tree Classifier gave the most correct result followed by Random Forest Classifier, and then Logistic Regression.

We not only used the above models, we displayed the confusion matrix and used k fold cross validation to avoid overfitting of our model.

We imported the each classifier from sklearn, used .fit method to train our model and .predict to predict class of the given input. The following is the code for logistic regression which we used.



```python
# Importing Logistic Regression
from sklearn.linear_model import  LogisticRegression

# creating a model of logistic regression
model = LogisticRegression()

# training data into model
model.fit(x_train, y_train)

# predicting test data values
y_pred = model.predict(x_test)

# calculating accuracies
print("Training Accuracy :", model.score(x_train, y_train))
print("Testing Accuracy :", model.score(x_test, y_test))
```

```
Training Accuracy : 0.956
Testing Accuracy : 0.96
```

If we compare our model results to the previous made models, we get to see that our overall accuracy of our model is greater than the existing models. We have used different criteria for pass/fail of students and we are not using the total marks to select the pass/fail students because what if a student had performed really well in math’s and not good in reading and writing. According to the total marks the students would have passed. E.g. A student has 100 in maths, and 40 in reading and writing, his total will be 180. This is greater than 150 so the student has passed. But if we see the student marks in each subject, we see that the student has failed in reading and writing. This gives our model an upper edge.