Part 1[¶](file:///C:\Users\Soukhna\AppData\Local\Packages\Microsoft.MicrosoftEdge_8wekyb3d8bbwe\TempState\Downloads\11.2%20Exercise%20Original%20Case%20Study%20Part%203%20(3).html#Part-1)

6.3 Exercise: Original Analysis Case Study Part 1[¶](file:///C:\Users\Soukhna\AppData\Local\Packages\Microsoft.MicrosoftEdge_8wekyb3d8bbwe\TempState\Downloads\11.2%20Exercise%20Original%20Case%20Study%20Part%203%20(3).html#6.3-Exercise:-Original-Analysis-Case-Study-Part-1)

As you go through this tutorial, using everything you’ve learned in the first half of this class, it is time to start developing your own original Analysis Case Study.

Provide a short narrative describing an original idea for an analysis problem. Find or create appropriate data that can be analyzed.

Write the step-by-step instructions for completing the Graph Analysis part of your case study.

You will post Part 1 of your Case Study (including the data) to the Week 7 Discussion Forum. This is due by the end of Week 6. This is a cumulative project so it is imperative that you do not fall behind.

This data set consists of the marks secured by the students in various subjects.

To understand the influence of various factors like economic, personal and social on the students performance Inferences would be : How to imporve the students performance in each test ? What are the major factors influencing the test scores ? Effectiveness of test preparation course?

load libraries[¶](file:///C:\Users\Soukhna\AppData\Local\Packages\Microsoft.MicrosoftEdge_8wekyb3d8bbwe\TempState\Downloads\11.2%20Exercise%20Original%20Case%20Study%20Part%203%20(3).html#load-libraries)

In [615]:

#Import the required libraries

import pandas as pd # This is used to create data frames and also read and write files

import numpy as np # importing the numpy library for scientific calculation

import matplotlib.pyplot as plt # This is also used for plotting

import seaborn as sns

%matplotlib inline

Step 1: Load data into a dataframe[¶](file:///C:\\Users\\Soukhna\\AppData\\Local\\Packages\\Microsoft.MicrosoftEdge_8wekyb3d8bbwe\\TempState\\Downloads\\11.2%20Exercise%20Original%20Case%20Study%20Part%203%20(3).html" \l "Step-1:--Load-data-into-a-dataframe)

In [617]:

url = "https://www.kaggle.com/spscientist/students-performance-in-exams.html"

In [618]:

#Step 1: Load data into a dataframe

# Read in the StudentsPerformance data set (given as a .csv file) from the local directory

Students\_Performance = pd.read\_csv("StudentsPerformance.csv")

df = Students\_Performance

# showing the dataset includes variables(1000) and observations(8)

df.head()

Out[618]:

|  | **gender** | **race/ethnicity** | **parental level of education** | **lunch** | **test preparation course** | **math score** | **reading score** | **writing score** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | female | group B | bachelor's degree | standard | none | 72 | 72 | 74 |
| 1 | female | group C | some college | standard | completed | 69 | 90 | 88 |
| 2 | female | group B | master's degree | standard | none | 90 | 95 | 93 |
| 3 | male | group A | associate's degree | free/reduced | none | 47 | 57 | 44 |
| 4 | male | group C | some college | standard | none | 76 | 78 | 75 |

In [619]:

# show the last five records(rows)

Students\_Performance.tail()

Out[619]:

|  | **gender** | **race/ethnicity** | **parental level of education** | **lunch** | **test preparation course** | **math score** | **reading score** | **writing score** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 995 | female | group E | master's degree | standard | completed | 88 | 99 | 95 |
| 996 | male | group C | high school | free/reduced | none | 62 | 55 | 55 |
| 997 | female | group C | high school | free/reduced | completed | 59 | 71 | 65 |
| 998 | female | group D | some college | standard | completed | 68 | 78 | 77 |
| 999 | female | group D | some college | free/reduced | none | 77 | 86 | 86 |

In [620]:

#df.info()

In [621]:

df.isnull().head(6)

Out[621]:

|  | **gender** | **race/ethnicity** | **parental level of education** | **lunch** | **test preparation course** | **math score** | **reading score** | **writing score** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | False | False | False | False | False | False | False | False |
| 1 | False | False | False | False | False | False | False | False |
| 2 | False | False | False | False | False | False | False | False |
| 3 | False | False | False | False | False | False | False | False |
| 4 | False | False | False | False | False | False | False | False |
| 5 | False | False | False | False | False | False | False | False |

In [622]:

# see the list of the columns

df.columns

Out[622]:

Index(['gender', 'race/ethnicity', 'parental level of education', 'lunch',

'test preparation course', 'math score', 'reading score',

'writing score'],

dtype='object')

In [625]:

# print the first five record of df (articles1)

df.head()

Out[625]:

|  | **gender** | **race/ethnicity** | **parental level of education** | **lunch** | **test preparation course** | **math score** | **reading score** | **writing score** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | female | group B | bachelor's degree | standard | none | 72 | 72 | 74 |
| 1 | female | group C | some college | standard | completed | 69 | 90 | 88 |
| 2 | female | group B | master's degree | standard | none | 90 | 95 | 93 |
| 3 | male | group A | associate's degree | free/reduced | none | 47 | 57 | 44 |
| 4 | male | group C | some college | standard | none | 76 | 78 | 75 |

Step 2 : Shape of the dataset(size of dataframe) / check the dimension of the table[¶](file:///C:\Users\Soukhna\AppData\Local\Packages\Microsoft.MicrosoftEdge_8wekyb3d8bbwe\TempState\Downloads\11.2%20Exercise%20Original%20Case%20Study%20Part%203%20(3).html#Step-2-:-Shape-of-the-dataset(size-of--dataframe)-/-check-the-dimension-of-the-table)

In [626]:

# Step 2: check the dimension of the table or the size of dataframe

print("The dimension of the table is: ", df.shape)

#Looking at the current shape of the dataset under consideration

#df.shape

The dimension of the table is: (1000, 8)

The below table gives statistical description of each attribute.[¶](file:///C:\Users\Soukhna\AppData\Local\Packages\Microsoft.MicrosoftEdge_8wekyb3d8bbwe\TempState\Downloads\11.2%20Exercise%20Original%20Case%20Study%20Part%203%20(3).html#The-below--table-gives-statistical-description-of-each-attribute.)

In [627]:

# display some statistical summaries of the numerical columns data.

df.describe().head()

Out[627]:

|  | **math score** | **reading score** | **writing score** |
| --- | --- | --- | --- |
| count | 1000.00000 | 1000.000000 | 1000.000000 |
| mean | 66.08900 | 69.169000 | 68.054000 |
| std | 15.16308 | 14.600192 | 15.195657 |
| min | 0.00000 | 17.000000 | 10.000000 |
| 25% | 57.00000 | 59.000000 | 57.750000 |

In [660]:

#Lets add columns of total score and average

df["Total\_Score"] = df["math score"] + df["reading score"] + df["writing score"]

df["Percentage"] = (df["Total\_Score"]/300)\*100

Boxplot-vertical[¶](file:///C:\Users\Soukhna\AppData\Local\Packages\Microsoft.MicrosoftEdge_8wekyb3d8bbwe\TempState\Downloads\11.2%20Exercise%20Original%20Case%20Study%20Part%203%20(3).html#Boxplot-vertical)

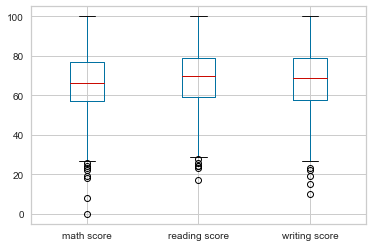
In [629]:

# showing plotbox

df[['math score','reading score','writing score']].plot.box()

Out[629]:

<matplotlib.axes.\_subplots.AxesSubplot at 0x1c7f23b45c8>



Boxplot- horizontal shows additional plot grid to help to see more precisely the essential parts of the plots, such as the median, outlier boundaries, etc.. By setting the parameter vert = False and grid = True.[¶](file:///C:\Users\Soukhna\AppData\Local\Packages\Microsoft.MicrosoftEdge_8wekyb3d8bbwe\TempState\Downloads\11.2%20Exercise%20Original%20Case%20Study%20Part%203%20(3).html#Boxplot--horizontal-shows-additional-plot-grid-to-help-to-see-more-precisely-the-essential-parts-of-the-plots,-such-as-the-median,-outlier-boundaries,-etc..-By-setting-the-parameter-vert-=-False-and-grid-=-True.)

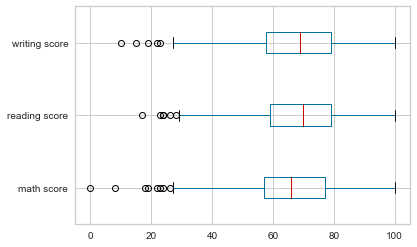
In [630]:

#shows horizontal plotbox with setting the parametter (vert = False and grid = True)

df[['math score','reading score','writing score']].plot.box(vert = False, grid = True)

Out[630]:

<matplotlib.axes.\_subplots.AxesSubplot at 0x1c7f2604188>



In [631]:

# lets check the percentage of missing data in each columns present in the data

no\_of\_columns = df.shape[0]

percentage\_of\_missing\_df = df.isnull().sum()/no\_of\_columns

percentage\_of\_missing\_df

Out[631]:

gender 0.0

race/ethnicity 0.0

parental level of education 0.0

lunch 0.0

test preparation course 0.0

math score 0.0

reading score 0.0

writing score 0.0

Total\_Score 0.0

Percentage 0.0

dtype: float64

The above result in this dataframe showing that there is no null values or missing values. Note that,in reality we need to work on dataset with a lot of missing values

Part 2[¶](file:///C:\Users\Soukhna\AppData\Local\Packages\Microsoft.MicrosoftEdge_8wekyb3d8bbwe\TempState\Downloads\11.2%20Exercise%20Original%20Case%20Study%20Part%203%20(3).html#Part-2)

7.3 Exercise: Original Case Study Part 2[¶](file:///C:\Users\Soukhna\AppData\Local\Packages\Microsoft.MicrosoftEdge_8wekyb3d8bbwe\TempState\Downloads\11.2%20Exercise%20Original%20Case%20Study%20Part%203%20(3).html#7.3-Exercise:-Original-Case-Study-Part-2)

Review and provide feedback to one of your classmate’s Case Study Part 1 assignments which are posted to the Week 7 Discussion Forum. Also review the feedback you receive on your Part 1 assignment. Make any changes you think necessary.

Create Part 2 of your Analysis Case Study project. Part 2 should consist of Dimensionality and Feature Reduction. You can use any methods/tools you think are most appropriate.

Write the step-by-step instructions for .

You will post Part 2 of your Case Study (including the data) to the Week 8 Discussion Forum. This is due by the end of Week 7. This is a cumulative project so it is imperative that you do not fall behind.

Submit the Sample Code for Part 1&2 to the Submission Link. Do not post the Sample Code to the Week 8 Discussion Forum with your project!

Step instructions for completing the Dimensionality and Feature Reduction[¶](file:///C:\Users\Soukhna\AppData\Local\Packages\Microsoft.MicrosoftEdge_8wekyb3d8bbwe\TempState\Downloads\11.2%20Exercise%20Original%20Case%20Study%20Part%203%20(3).html#Step-instructions-for-completing-the-Dimensionality-and-Feature-Reduction)

In [632]:

#Step 2: check the dimension of the table

print("The dimension of the table is: ", df.shape)

The dimension of the table is: (1000, 10)

In [633]:

df.columns

Out[633]:

Index(['gender', 'race/ethnicity', 'parental level of education', 'lunch',

'test preparation course', 'math score', 'reading score',

'writing score', 'Total\_Score', 'Percentage'],

dtype='object')

Statistical description of each attribute[¶](file:///C:\Users\Soukhna\AppData\Local\Packages\Microsoft.MicrosoftEdge_8wekyb3d8bbwe\TempState\Downloads\11.2%20Exercise%20Original%20Case%20Study%20Part%203%20(3).html#Statistical-description-of-each-attribute)

In [634]:

#understand about the basic information of the data, like min, max, mean and standard deviation

df.describe()

Out[634]:

|  | **math score** | **reading score** | **writing score** | **Total\_Score** | **Percentage** |
| --- | --- | --- | --- | --- | --- |
| count | 1000.00000 | 1000.000000 | 1000.000000 | 1000.000000 | 1000.000000 |
| mean | 66.08900 | 69.169000 | 68.054000 | 203.312000 | 67.770667 |
| std | 15.16308 | 14.600192 | 15.195657 | 42.771978 | 14.257326 |
| min | 0.00000 | 17.000000 | 10.000000 | 27.000000 | 9.000000 |
| 25% | 57.00000 | 59.000000 | 57.750000 | 175.000000 | 58.333333 |
| 50% | 66.00000 | 70.000000 | 69.000000 | 205.000000 | 68.333333 |
| 75% | 77.00000 | 79.000000 | 79.000000 | 233.000000 | 77.666667 |
| max | 100.00000 | 100.000000 | 100.000000 | 300.000000 | 100.000000 |

Bar Charts of some categorical variables[¶](file:///C:\Users\Soukhna\AppData\Local\Packages\Microsoft.MicrosoftEdge_8wekyb3d8bbwe\TempState\Downloads\11.2%20Exercise%20Original%20Case%20Study%20Part%203%20(3).html#Bar-Charts-of-some-categorical-variables)

Case Study: Testing Hypothesis: To understand the influence of the parents background, test preparation etc on students performance And more likely to be published by "Students Performance in Exams" sources

In [635]:

#Some EDA

# Let's talk about gender ,Race,Parental education and lunch

gender=Students\_Performance["gender"]

#Race

race=Students\_Performance["race/ethnicity"]

#Some Parental Education

PE=Students\_Performance["parental level of education"]

#Some Lunch Data

lunch=Students\_Performance["lunch"]

Bar Charts of some categorical variables[¶](file:///C:\Users\Soukhna\AppData\Local\Packages\Microsoft.MicrosoftEdge_8wekyb3d8bbwe\TempState\Downloads\11.2%20Exercise%20Original%20Case%20Study%20Part%203%20(3).html#Bar-Charts-of-some-categorical-variables)

In [636]:

#Bar Charts of some categorical variables

#Import the required libraries

import matplotlib.pyplot as plt

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

ax1 = plt.subplot2grid((4,3), (0,0), rowspan=1, colspan=1)

ax2 = plt.subplot2grid((4,3), (0,1), rowspan=1, colspan=1)

ax3 = plt.subplot2grid((4,3), (0,2), rowspan=1, colspan=1)

ax4 = plt.subplot2grid((4,3), (1,0), rowspan=1, colspan=2)

ax5 = plt.subplot2grid((4,3), (1,2), rowspan=1, colspan=1)

#sns.barplot(x=gender.unique(), y=gender.value\_counts(), palette="rocket", ax=axs[0,0])

ax1.bar(gender.unique(),gender.value\_counts())

ax1.set\_xlabel('gender Bar Chart')

ax2.bar(race.unique(),race.value\_counts())

ax2.set\_xlabel('race Bar Chart')

ax3.bar(lunch.unique(),lunch.value\_counts())

ax3.set\_xlabel('lunch Bar Chart')

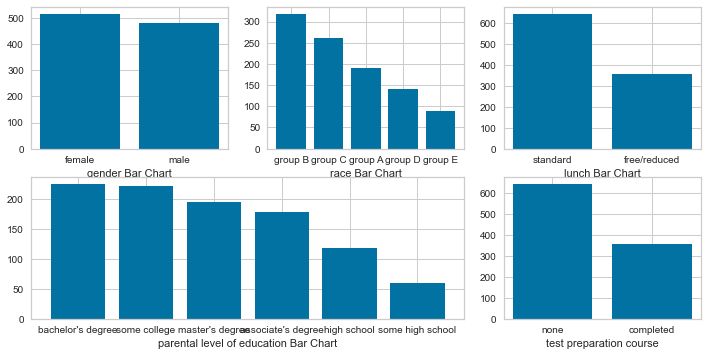
ax4.bar(PE.unique(),PE.value\_counts())

ax4.set\_xlabel('parental level of education Bar Chart')

ax5.bar(Students\_Performance["test preparation course"].unique(),lunch.value\_counts())

ax5.set\_xlabel('test preparation course')

plt.show()



In [650]:

# lets check the no. of unique items present in the categorical column

df.select\_dtypes('object').nunique()

Out[650]:

gender 2

race/ethnicity 5

parental level of education 6

lunch 2

test preparation course 2

dtype: int64

Add columns of total score and average[¶](file:///C:\Users\Soukhna\AppData\Local\Packages\Microsoft.MicrosoftEdge_8wekyb3d8bbwe\TempState\Downloads\11.2%20Exercise%20Original%20Case%20Study%20Part%203%20(3).html#Add-columns-of-total-score-and-average)

In [642]:

#Lets add columns of total score and average

df["Total\_Score"]=df["math score"]+df["reading score"]+df["writing score"]

df["Percentage"]=(df["Total\_Score"]/300)\*100

In [643]:

# print some result

print(df[df["Percentage"]==max(df["Percentage"])])

#So 2 Female out of 3 got 100% marks.

gender race/ethnicity parental level of education lunch \

458 female group E bachelor's degree standard

916 male group E bachelor's degree standard

962 female group E associate's degree standard

test preparation course math score reading score writing score \

458 none 100 100 100

916 completed 100 100 100

962 none 100 100 100

Total\_Score Percentage

458 300 100.0

916 300 100.0

962 300 100.0

In [644]:

# print some result

print(df[df["Percentage"]==min(df["Percentage"])])

#One Female got lowest percentage

gender race/ethnicity parental level of education lunch \

59 female group C some high school free/reduced

test preparation course math score reading score writing score \

59 none 0 17 10

Total\_Score Percentage

59 27 9.0

Plotting all the Distrbutions of all marks[¶](file:///C:\Users\Soukhna\AppData\Local\Packages\Microsoft.MicrosoftEdge_8wekyb3d8bbwe\TempState\Downloads\11.2%20Exercise%20Original%20Case%20Study%20Part%203%20(3).html#Plotting-all-the-Distrbutions-of-all-marks)

In [645]:

#Plotting all the Distrbutions of all marks

fig, axs = plt.subplots(2, 2, figsize=(15, 15))

axs[0,0].hist(df["math score"],edgecolor="black")

axs[0,0].set\_xlabel('Math Score Distribution')

axs[0,1].hist(df["reading score"],edgecolor="black")

axs[0,1].set\_xlabel('Reading Score Distribution')

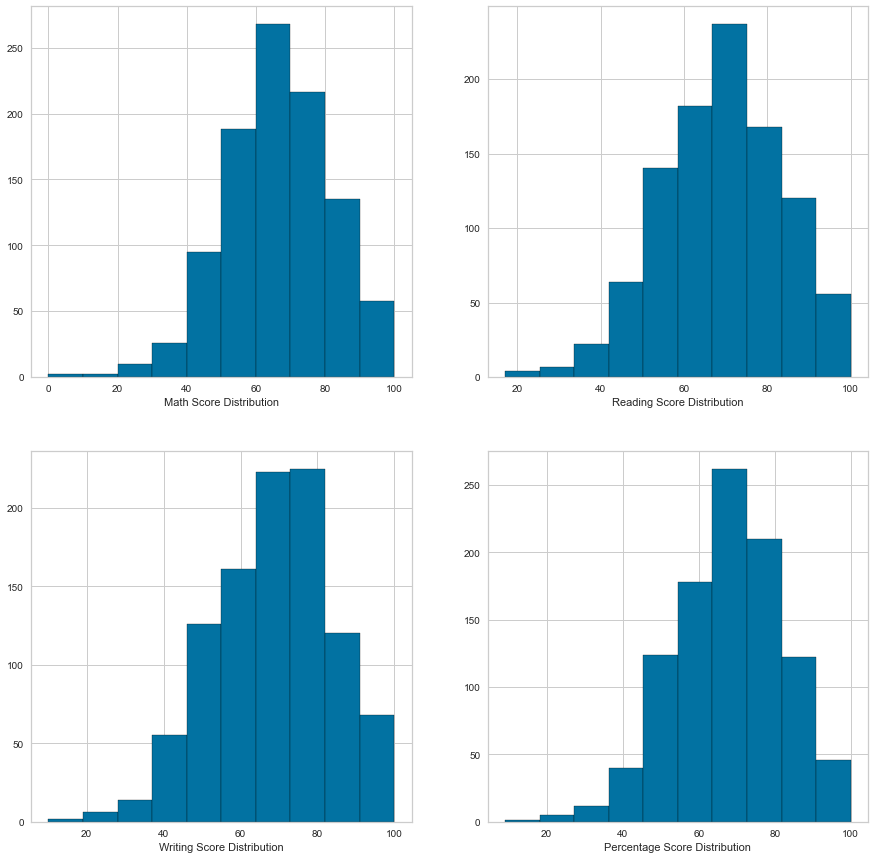
axs[1,0].hist(df["writing score"],edgecolor="black")

axs[1,0].set\_xlabel('Writing Score Distribution')

axs[1,1].hist(df["Percentage"],edgecolor="black")

axs[1,1].set\_xlabel('Percentage Score Distribution')

plt.show()



Distributions of score based on different categories[¶](file:///C:\Users\Soukhna\AppData\Local\Packages\Microsoft.MicrosoftEdge_8wekyb3d8bbwe\TempState\Downloads\11.2%20Exercise%20Original%20Case%20Study%20Part%203%20(3).html#Distributions-of-score-based-on-different-categories)

In [646]:

import seaborn as sns

#Distributions of score based on different categories

fig = plt.figure(figsize=(15,15))

ax1 = plt.subplot2grid((4,3), (0,0), rowspan=1, colspan=1)

ax2 = plt.subplot2grid((4,3), (0,1), rowspan=1, colspan=1)

ax3 = plt.subplot2grid((4,3), (0,2), rowspan=1, colspan=1)

ax4 = plt.subplot2grid((4,3), (1,0), rowspan=1, colspan=2)

ax5 = plt.subplot2grid((4,3), (1,2), rowspan=1, colspan=1)

sns.boxplot(x=df["gender"], y=df["Total\_Score"], palette=["m", "g"],ax=ax1)

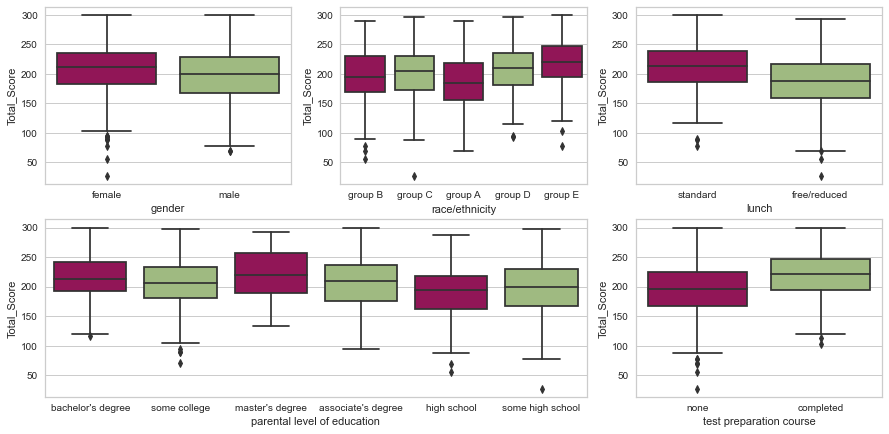
sns.boxplot(x=df["race/ethnicity"], y=df["Total\_Score"], palette=["m", "g"],ax=ax2)

sns.boxplot(x=df["lunch"], y=df["Total\_Score"], palette=["m", "g"],ax=ax3)

sns.boxplot(x=df["parental level of education"], y=df["Total\_Score"], palette=["m", "g"],ax=ax4)

sns.boxplot(x=df["test preparation course"], y=df["Total\_Score"], palette=["m", "g"],ax=ax5)

plt.show()



Data Exploration[¶](file:///C:\Users\Soukhna\AppData\Local\Packages\Microsoft.MicrosoftEdge_8wekyb3d8bbwe\TempState\Downloads\11.2%20Exercise%20Original%20Case%20Study%20Part%203%20(3).html#Data-Exploration)

In [647]:

# Data Exploration

df\_correlation = df.corr()

df\_correlation

Out[647]:

|  | **math score** | **reading score** | **writing score** | **Total\_Score** | **Percentage** |
| --- | --- | --- | --- | --- | --- |
| math score | 1.000000 | 0.817580 | 0.802642 | 0.918746 | 0.918746 |
| reading score | 0.817580 | 1.000000 | 0.954598 | 0.970331 | 0.970331 |
| writing score | 0.802642 | 0.954598 | 1.000000 | 0.965667 | 0.965667 |
| Total\_Score | 0.918746 | 0.970331 | 0.965667 | 1.000000 | 1.000000 |
| Percentage | 0.918746 | 0.970331 | 0.965667 | 1.000000 | 1.000000 |

Correlation[¶](file:///C:\Users\Soukhna\AppData\Local\Packages\Microsoft.MicrosoftEdge_8wekyb3d8bbwe\TempState\Downloads\11.2%20Exercise%20Original%20Case%20Study%20Part%203%20(3).html#Correlation)

In [648]:

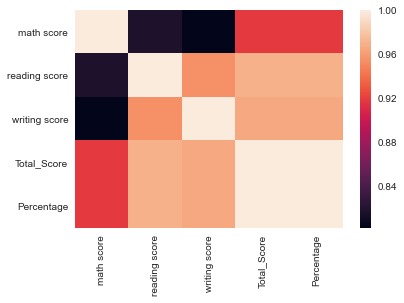
#COORELATION

Mdf = df

Corr=Mdf.corr()

sns.heatmap(Corr)

plt.show()



In [657]:

# explore the Reading score

sns.countplot(x="reading score", data = df, palette="muted")

plt.show()



In [664]:

#Initialize the required values - set the minimum marks to 40 to pass in a exam

passmark=50

How many studends passed in reading ?[¶](file:///C:\Users\Soukhna\AppData\Local\Packages\Microsoft.MicrosoftEdge_8wekyb3d8bbwe\TempState\Downloads\11.2%20Exercise%20Original%20Case%20Study%20Part%203%20(3).html#How-many-studends-passed-in-reading-?)

In [663]:

#To find out the number of studends who passed in reading

df['Reading\_PassStatus'] = np.where(df['reading score']<passmark, 'F', 'P')

df.Reading\_PassStatus.value\_counts()

Out[663]:

P 974

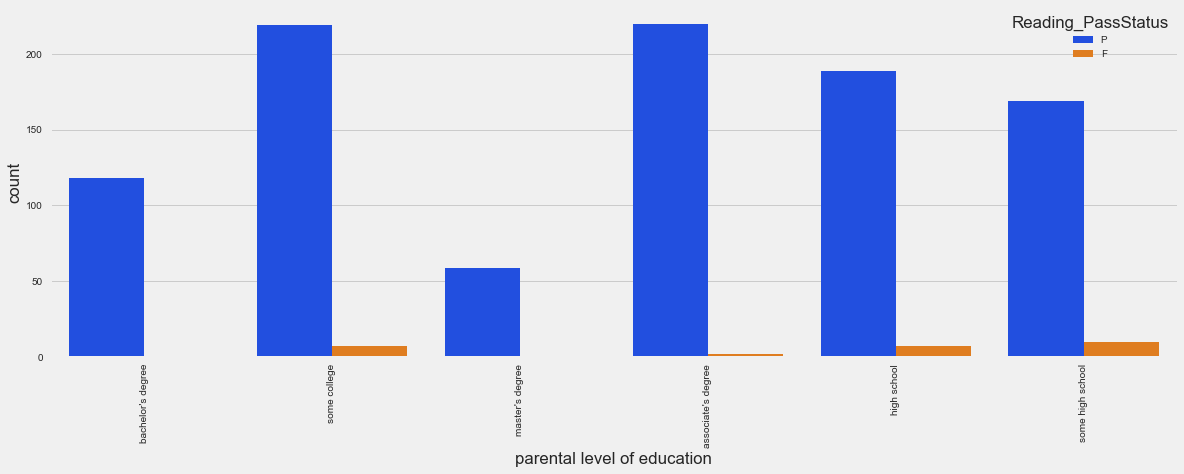
F 26

Name: Reading\_PassStatus, dtype: int64

In [665]:

p = sns.countplot(x='parental level of education', data = df, hue='Reading\_PassStatus', palette='bright')

\_ = plt.setp(p.get\_xticklabels(), rotation=90)

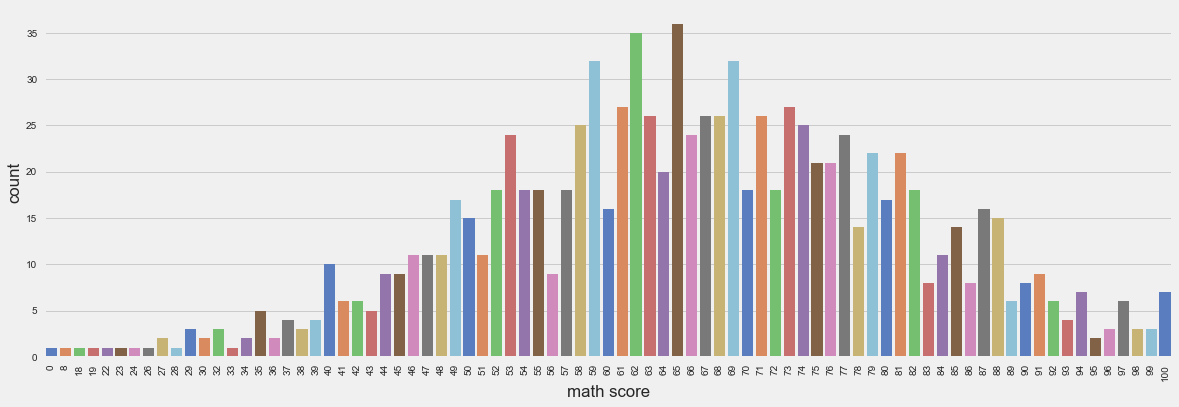


In [666]:

#To explore the Math Score first

p = sns.countplot(x="math score", data = df, palette="muted")

\_ = plt.setp(p.get\_xticklabels(), rotation=90)



In [667]:

#how many students passed in Math exam ?

df['Math\_PassStatus'] = np.where(df['math score']<passmark, 'F', 'P')

df.Math\_PassStatus.value\_counts()

Out[667]:

P 865

F 135

Name: Math\_PassStatus, dtype: int64

Standardize the feature matrix[¶](file:///C:\Users\Soukhna\AppData\Local\Packages\Microsoft.MicrosoftEdge_8wekyb3d8bbwe\TempState\Downloads\11.2%20Exercise%20Original%20Case%20Study%20Part%203%20(3).html#Standardize-the-feature-matrix)

In [559]:

from sklearn.preprocessing import StandardScaler

from sklearn.decomposition import PCA

from sklearn import datasets

#load the data

#datasets = StudentsPerformance

digits = datasets.load\_digits()

# standardize the feature matrix

features = StandardScaler().fit\_transform(digits.data)

#create a PCA that will retain 99% of variance

pca = PCA(n\_components=0.99,whiten=True)

#conduct PCA

features\_pca =pca.fit\_transform(features)

features\_pca

#show results

print("Original number of features: ,features.shape[1]")

print("Reduced number of features: ,features\_pca.shape[1]")

Original number of features: ,features.shape[1]

Reduced number of features: ,features\_pca.shape[1]

In [560]:

features.shape

Out[560]:

(1797, 64)

In [561]:

features\_pca.shape

Out[561]:

(1797, 54)

Bar Plots[¶](file:///C:\Users\Soukhna\AppData\Local\Packages\Microsoft.MicrosoftEdge_8wekyb3d8bbwe\TempState\Downloads\11.2%20Exercise%20Original%20Case%20Study%20Part%203%20(3).html#Bar-Plots)

In [562]:

fig, axes = plt.subplots(2, 1)

df = pd.Series(np.random.rand(16), index=list('abcdefghijklmnop'))

df.plot.bar(ax=axes[0], color='k', alpha=0.7)

df.plot.barh(ax=axes[1], color='k', alpha=0.7)

C:\Users\Soukhna\Anaconda3\lib\site-packages\pandas\plotting\\_matplotlib\tools.py:307: MatplotlibDeprecationWarning:

The rowNum attribute was deprecated in Matplotlib 3.2 and will be removed two minor releases later. Use ax.get\_subplotspec().rowspan.start instead.

layout[ax.rowNum, ax.colNum] = ax.get\_visible()

C:\Users\Soukhna\Anaconda3\lib\site-packages\pandas\plotting\\_matplotlib\tools.py:307: MatplotlibDeprecationWarning:

The colNum attribute was deprecated in Matplotlib 3.2 and will be removed two minor releases later. Use ax.get\_subplotspec().colspan.start instead.

layout[ax.rowNum, ax.colNum] = ax.get\_visible()

C:\Users\Soukhna\Anaconda3\lib\site-packages\pandas\plotting\\_matplotlib\tools.py:313: MatplotlibDeprecationWarning:

The rowNum attribute was deprecated in Matplotlib 3.2 and will be removed two minor releases later. Use ax.get\_subplotspec().rowspan.start instead.

if not layout[ax.rowNum + 1, ax.colNum]:

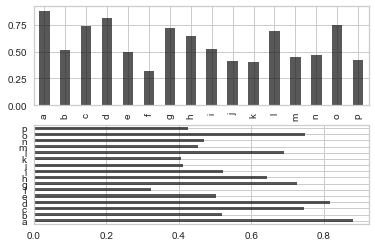
C:\Users\Soukhna\Anaconda3\lib\site-packages\pandas\plotting\\_matplotlib\tools.py:313: MatplotlibDeprecationWarning:

The colNum attribute was deprecated in Matplotlib 3.2 and will be removed two minor releases later. Use ax.get\_subplotspec().colspan.start instead.

if not layout[ax.rowNum + 1, ax.colNum]:

Out[562]:

<matplotlib.axes.\_subplots.AxesSubplot at 0x1c7f1391348>



Convert categorical data to numbers[¶](file:///C:\Users\Soukhna\AppData\Local\Packages\Microsoft.MicrosoftEdge_8wekyb3d8bbwe\TempState\Downloads\11.2%20Exercise%20Original%20Case%20Study%20Part%203%20(3).html#Convert-categorical-data-to-numbers)

In [563]:

Students\_Performance.dtypes

Out[563]:

gender object

race/ethnicity object

parental level of education object

lunch object

test preparation course object

math score int64

reading score int64

writing score int64

Total\_Score int64

Percentage float64

dtype: object

In [564]:

#get the categorical data

cat\_features = ['gender', 'race/ethnicity', 'parental level of education', 'lunch', 'test preparation course']

Students\_Performance\_cat = Students\_Performance[cat\_features]

Students\_Performance\_cat = Students\_Performance\_cat.replace({'Pclass': {1: '1st', 2: '2nd', 3: '3rd'}})

In [565]:

# One Hot Encoding

Students\_Performance\_cat\_dummies = pd.get\_dummies(Students\_Performance\_cat)

# check the dataset

print(Students\_Performance\_cat\_dummies.head(6))

gender\_female gender\_male race/ethnicity\_group A race/ethnicity\_group B \

0 1 0 0 1

1 1 0 0 0

2 1 0 0 1

3 0 1 1 0

4 0 1 0 0

5 1 0 0 1

race/ethnicity\_group C race/ethnicity\_group D race/ethnicity\_group E \

0 0 0 0

1 1 0 0

2 0 0 0

3 0 0 0

4 1 0 0

5 0 0 0

parental level of education\_associate's degree \

0 0

1 0

2 0

3 1

4 0

5 1

parental level of education\_bachelor's degree \

0 1

1 0

2 0

3 0

4 0

5 0

parental level of education\_high school \

0 0

1 0

2 0

3 0

4 0

5 0

parental level of education\_master's degree \

0 0

1 0

2 1

3 0

4 0

5 0

parental level of education\_some college \

0 0

1 1

2 0

3 0

4 1

5 0

parental level of education\_some high school lunch\_free/reduced \

0 0 0

1 0 0

2 0 0

3 0 1

4 0 0

5 0 0

lunch\_standard test preparation course\_completed \

0 1 0

1 1 1

2 1 0

3 0 0

4 1 0

5 1 0

test preparation course\_none

0 1

1 0

2 1

3 1

4 1

5 1

In [566]:

Students\_Performance.keys()

#df.keys()

Out[566]:

Index(['gender', 'race/ethnicity', 'parental level of education', 'lunch',

'test preparation course', 'math score', 'reading score',

'writing score', 'Total\_Score', 'Percentage'],

dtype='object')

PCA visualization[¶](file:///C:\Users\Soukhna\AppData\Local\Packages\Microsoft.MicrosoftEdge_8wekyb3d8bbwe\TempState\Downloads\11.2%20Exercise%20Original%20Case%20Study%20Part%203%20(3).html#PCA-visualization)

"As we know it is difficult to visualize the data with so many features i.e high dimensional data so we can use PCA to find the two principal components hence visualize the data in two-dimensional space with a single scatter plot. But, before that, we need to pre-process the data i.e we need to scale the data such that each feature has unit variance and has not a greater impact than the other one." <https://datascienceplus.com/principal-component-analysis-pca-with-python/>

In [567]:

import pandas as pd

import numpy as np

from sklearn.decomposition import PCA

df = pd.DataFrame(data=np.random.normal(0, 1, (20, 10)))

pca = PCA(n\_components=5)

pca.fit(df)

Out[567]:

PCA(copy=True, iterated\_power='auto', n\_components=5, random\_state=None,

svd\_solver='auto', tol=0.0, whiten=False)

In [568]:

# access the components themselves

pca.components\_

Out[568]:

array([[ 0.19720902, 0.03440352, -0.03080231, 0.24625556, 0.23891697,

0.22261508, 0.50601464, 0.05234779, 0.43670248, -0.58497475],

[ 0.092611 , -0.17011321, 0.54381219, -0.00409001, 0.45413374,

0.59678741, -0.10018496, -0.10413138, -0.26827891, 0.10718982],

[ 0.43923183, 0.02220277, -0.36236382, -0.0936854 , -0.35283989,

0.33627715, 0.35896758, -0.35969646, -0.40315833, 0.09024259],

[ 0.05600072, 0.23221105, 0.14008904, 0.42945167, -0.36843843,

0.2803349 , -0.31176092, -0.35655582, 0.49668964, 0.23135734],

[ 0.31150065, -0.16798696, 0.062979 , 0.48517809, -0.0871496 ,

-0.04596102, 0.21696158, 0.61657451, -0.05939713, 0.441488 ]])

In [569]:

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

scaler.fit(df)

Out[569]:

StandardScaler(copy=True, with\_mean=True, with\_std=True)

In [570]:

scaled\_data = scaler.transform(df)

In [571]:

#specify number of components as 2

from sklearn.decomposition import PCA

pca = PCA(n\_components=2)

pca.fit(scaled\_data)

Out[571]:

PCA(copy=True, iterated\_power='auto', n\_components=2, random\_state=None,

svd\_solver='auto', tol=0.0, whiten=False)

In [572]:

#Now we can transform this data to its first 2 principal components.

x\_pca = pca.transform(scaled\_data)

In [573]:

#Now let us check the shape of data before and after PCA

scaled\_data.shape

Out[573]:

(20, 10)

In [574]:

x\_pca.shape

Out[574]:

(20, 2)

In the above lines, we have reduced 10 dimensions to just 2.

In [575]:

#important python libraries required for this algorithm

import matplotlib.pyplot as plt

import pandas as pd

import numpy as np

import seaborn as sns

%matplotlib inline

In [576]:

#Import the dataset from the python library sci-kit-learn.

from sklearn.datasets import load\_breast\_cancer

cancer = load\_breast\_cancer()

In [577]:

#Since the datset is in a form of a dictionary, So we will check what all key values are there in dataset.

cancer.keys()

Out[577]:

dict\_keys(['data', 'target', 'target\_names', 'DESCR', 'feature\_names', 'filename'])

In [578]:

#Now lets make the Dataframe for the given data and check its head value.

df = pd.DataFrame(cancer['data'],columns=cancer['feature\_names'])

df.head()

Out[578]:

|  | **mean radius** | **mean texture** | **mean perimeter** | **mean area** | **mean smoothness** | **mean compactness** | **mean concavity** | **mean concave points** | **mean symmetry** | **mean fractal dimension** | **...** | **worst radius** | **worst texture** | **worst perimeter** | **worst area** | **worst smoothness** | **worst compactness** | **worst concavity** | **worst concave points** | **worst symmetry** | **worst fractal dimension** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 17.99 | 10.38 | 122.80 | 1001.0 | 0.11840 | 0.27760 | 0.3001 | 0.14710 | 0.2419 | 0.07871 | ... | 25.38 | 17.33 | 184.60 | 2019.0 | 0.1622 | 0.6656 | 0.7119 | 0.2654 | 0.4601 | 0.11890 |
| 1 | 20.57 | 17.77 | 132.90 | 1326.0 | 0.08474 | 0.07864 | 0.0869 | 0.07017 | 0.1812 | 0.05667 | ... | 24.99 | 23.41 | 158.80 | 1956.0 | 0.1238 | 0.1866 | 0.2416 | 0.1860 | 0.2750 | 0.08902 |
| 2 | 19.69 | 21.25 | 130.00 | 1203.0 | 0.10960 | 0.15990 | 0.1974 | 0.12790 | 0.2069 | 0.05999 | ... | 23.57 | 25.53 | 152.50 | 1709.0 | 0.1444 | 0.4245 | 0.4504 | 0.2430 | 0.3613 | 0.08758 |
| 3 | 11.42 | 20.38 | 77.58 | 386.1 | 0.14250 | 0.28390 | 0.2414 | 0.10520 | 0.2597 | 0.09744 | ... | 14.91 | 26.50 | 98.87 | 567.7 | 0.2098 | 0.8663 | 0.6869 | 0.2575 | 0.6638 | 0.17300 |
| 4 | 20.29 | 14.34 | 135.10 | 1297.0 | 0.10030 | 0.13280 | 0.1980 | 0.10430 | 0.1809 | 0.05883 | ... | 22.54 | 16.67 | 152.20 | 1575.0 | 0.1374 | 0.2050 | 0.4000 | 0.1625 | 0.2364 | 0.07678 |

5 rows × 30 columns

In [579]:

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

scaler.fit(df)

Out[579]:

StandardScaler(copy=True, with\_mean=True, with\_std=True)

In [580]:

scaled\_data = scaler.transform(df)

In [581]:

#specify number of components as 2

from sklearn.decomposition import PCA

pca = PCA(n\_components=2)

pca.fit(scaled\_data)

Out[581]:

PCA(copy=True, iterated\_power='auto', n\_components=2, random\_state=None,

svd\_solver='auto', tol=0.0, whiten=False)

In [582]:

#Now we can transform this data to its first 2 principal components.

x\_pca = pca.transform(scaled\_data)

In [583]:

#Now let us check the shape of data before and after PCA

scaled\_data.shape

Out[583]:

(569, 30)

In [584]:

x\_pca.shape

Out[584]:

(569, 2)

In the above lines, we have reduced 30 dimensions to just 2.

Plot these two dimensions out![¶](file:///C:\Users\Soukhna\AppData\Local\Packages\Microsoft.MicrosoftEdge_8wekyb3d8bbwe\TempState\Downloads\11.2%20Exercise%20Original%20Case%20Study%20Part%203%20(3).html#Plot-these-two-dimensions-out!)

In [585]:

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

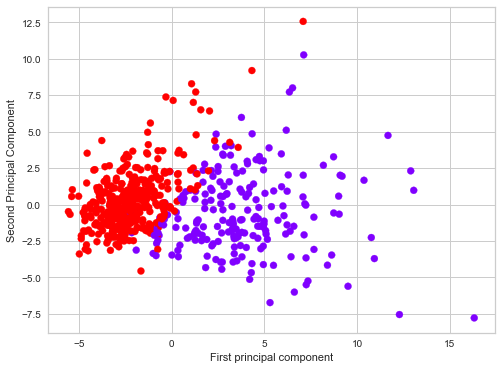
plt.scatter(x\_pca[:,0],x\_pca[:,1],c=cancer['target'],cmap='rainbow')

plt.xlabel('First principal component')

plt.ylabel('Second Principal Component')

Out[585]:

Text(0, 0.5, 'Second Principal Component')



The above lines showing clearly two components that can be use to easily separate these two classes.[¶](file:///C:\Users\Soukhna\AppData\Local\Packages\Microsoft.MicrosoftEdge_8wekyb3d8bbwe\TempState\Downloads\11.2%20Exercise%20Original%20Case%20Study%20Part%203%20(3).html#The-above-lines-showing-clearly-two-components-that-can--be-use-to-easily-separate-these-two-classes.)

Interpreting the components[¶](file:///C:\Users\Soukhna\AppData\Local\Packages\Microsoft.MicrosoftEdge_8wekyb3d8bbwe\TempState\Downloads\11.2%20Exercise%20Original%20Case%20Study%20Part%203%20(3).html#Interpreting-the-components)

"Its not easy to understand these component reduction.The components correspond to combinations of the original features, the components themselves are stored as an attribute of the fitted PCA object:

In [586]:

pca.components\_

Out[586]:

array([[ 0.21890244, 0.10372458, 0.22753729, 0.22099499, 0.14258969,

0.23928535, 0.25840048, 0.26085376, 0.13816696, 0.06436335,

0.20597878, 0.01742803, 0.21132592, 0.20286964, 0.01453145,

0.17039345, 0.15358979, 0.1834174 , 0.04249842, 0.10256832,

0.22799663, 0.10446933, 0.23663968, 0.22487053, 0.12795256,

0.21009588, 0.22876753, 0.25088597, 0.12290456, 0.13178394],

[-0.23385713, -0.05970609, -0.21518136, -0.23107671, 0.18611302,

0.15189161, 0.06016536, -0.0347675 , 0.19034877, 0.36657547,

-0.10555215, 0.08997968, -0.08945723, -0.15229263, 0.20443045,

0.2327159 , 0.19720728, 0.13032156, 0.183848 , 0.28009203,

-0.21986638, -0.0454673 , -0.19987843, -0.21935186, 0.17230435,

0.14359317, 0.09796411, -0.00825724, 0.14188335, 0.27533947]])

In [587]:

#visualize the above using heatmap

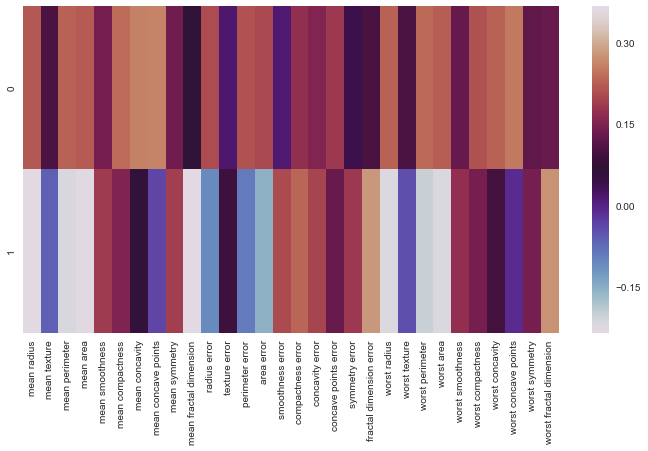
map= pd.DataFrame(pca.components\_,columns=cancer['feature\_names'])

plt.figure(figsize=(12,6))

sns.heatmap(map,cmap='twilight')

Out[587]:

<matplotlib.axes.\_subplots.AxesSubplot at 0x1c7f07bb0c8>



The above heatmap and the color bar basically represent the correlation between the various feature and the principal component itself.[¶](file:///C:\Users\Soukhna\AppData\Local\Packages\Microsoft.MicrosoftEdge_8wekyb3d8bbwe\TempState\Downloads\11.2%20Exercise%20Original%20Case%20Study%20Part%203%20(3).html#The-above-heatmap-and-the-color-bar-basically-represent-the-correlation-between-the-various-feature-and-the-principal-component-itself.)

"This is useful when you are dealing with the high dimensional dataset. Stay tuned for more fun!"

<https://datascienceplus.com/principal-component-analysis-pca-with-python/>

<https://towardsdatascience.com/intro-to-feature-selection-methods-for-data-science-4cae2178a00a>

<https://github.com/WillKoehrsen/feature-selector/blob/master/Feature%20Selector%20Usage.ipynb>

8.3 Exercise Original Case Study Part 3 - Create Part 3 should consist of Model Evaluation and Selection.[¶](file:///C:\Users\Soukhna\AppData\Local\Packages\Microsoft.MicrosoftEdge_8wekyb3d8bbwe\TempState\Downloads\11.2%20Exercise%20Original%20Case%20Study%20Part%203%20(3).html#8.3-Exercise-Original-Case-Study-Part-3---Create-Part-3-should-consist-of-Model-Evaluation-and-Selection.)

Part 3[¶](file:///C:\Users\Soukhna\AppData\Local\Packages\Microsoft.MicrosoftEdge_8wekyb3d8bbwe\TempState\Downloads\11.2%20Exercise%20Original%20Case%20Study%20Part%203%20(3).html#Part-3)

1. Training - Split your data into two sets: Training and Testing.
2. Evaluation – Remember, we are trying to predict if a passenger has survived or not so this is a classification problem. There are many algorithms that could be used but we’re going to use logistic regression.  
   a. Metrics for the evaluation:  
   i. Confusion Matrix (you should get 84% - pretty good) ii. Precision, Recall & F1 score (all 3 were very good) iii. ROC curve (the dotted line is the randomly guessed so anything above that is good metric)

14. Training - Split data into two sets: Training and Testing[¶](file:///C:\Users\Soukhna\AppData\Local\Packages\Microsoft.MicrosoftEdge_8wekyb3d8bbwe\TempState\Downloads\11.2%20Exercise%20Original%20Case%20Study%20Part%203%20(3).html#14.-Training---Split-data-into-two-sets:--Training-and-Testing)

Create a whole features dataset that can be used for train and validation data splitting[¶](file:///C:\Users\Soukhna\AppData\Local\Packages\Microsoft.MicrosoftEdge_8wekyb3d8bbwe\TempState\Downloads\11.2%20Exercise%20Original%20Case%20Study%20Part%203%20(3).html#Create-a-whole-features-dataset-that-can-be-used-for-train-and-validation-data-splitting)

In [588]:

#Step 14 - create a whole features dataset that can be used for train and validation data splitting

# here we will combine the numerical features and the dummie features together

features\_model = ['math score', 'reading score','writing score']

Students\_Performance\_model\_X = pd.concat([Students\_Performance[features\_model], Students\_Performance\_cat\_dummies], axis=1)

In [589]:

#display the fisrt rows

Students\_Performance\_model\_X.head()

Out[589]:

|  | **math score** | **reading score** | **writing score** | **gender\_female** | **gender\_male** | **race/ethnicity\_group A** | **race/ethnicity\_group B** | **race/ethnicity\_group C** | **race/ethnicity\_group D** | **race/ethnicity\_group E** | **parental level of education\_associate's degree** | **parental level of education\_bachelor's degree** | **parental level of education\_high school** | **parental level of education\_master's degree** | **parental level of education\_some college** | **parental level of education\_some high school** | **lunch\_free/reduced** | **lunch\_standard** | **test preparation course\_completed** | **test preparation course\_none** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 72 | 72 | 74 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| 1 | 69 | 90 | 88 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 |
| 2 | 90 | 95 | 93 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 |
| 3 | 47 | 57 | 44 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| 4 | 76 | 78 | 75 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 |

Create a whole target dataset that can be used for train and validation data splitting[¶](file:///C:\Users\Soukhna\AppData\Local\Packages\Microsoft.MicrosoftEdge_8wekyb3d8bbwe\TempState\Downloads\11.2%20Exercise%20Original%20Case%20Study%20Part%203%20(3).html#Create-a-whole-target-dataset-that-can-be-used-for-train-and-validation-data-splitting)

In [590]:

# create a whole target dataset that can be used for train and validation data splitting

Students\_Performance\_model\_y = Students\_Performance.replace({'test preparation course':

{'completed':'test preparation course', 'none':

'Not\_test preparation course'}})['test preparation course']

#Students\_Performance\_model\_y

In [591]:

# create a whole target dataset that can be used for train and validation data splitting

#Students\_Performance\_model\_y = Students\_Performance.replace({'gender':{'male':'gender', 'female':'Not\_gender'}})['gender']

#Students\_Performance\_model\_y

Separate data into training and validation and check the details of the datasets[¶](file:///C:\Users\Soukhna\AppData\Local\Packages\Microsoft.MicrosoftEdge_8wekyb3d8bbwe\TempState\Downloads\11.2%20Exercise%20Original%20Case%20Study%20Part%203%20(3).html#Separate-data-into-training-and-validation-and-check-the-details-of-the-datasets)

In [592]:

# separate data into training and validation and check the details of the datasets

# import packages

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

# split the data

X\_train, X\_val, y\_train, y\_val = train\_test\_split(Students\_Performance\_model\_X, Students\_Performance\_model\_y, test\_size =0.3, random\_state=11)

Number of samples in each set[¶](file:///C:\Users\Soukhna\AppData\Local\Packages\Microsoft.MicrosoftEdge_8wekyb3d8bbwe\TempState\Downloads\11.2%20Exercise%20Original%20Case%20Study%20Part%203%20(3).html#Number-of-samples-in-each-set)

In [593]:

# number of samples in each set

print("No. of samples in training set: ", X\_train.shape[0])

print("No. of samples in validation set:", X\_val.shape[0])

No. of samples in training set: 700

No. of samples in validation set: 300

In [594]:

# test preparation course and not-test preparation course

print('\n')

print('No. of test preparation course and not-test preparation course in the training set:')

print(y\_train.value\_counts())

print('\n')

print('No. of test preparation course and not-test preparation course in the validation set:')

print(y\_val.value\_counts())

No. of test preparation course and not-test preparation course in the training set:

Not\_test preparation course 445

test preparation course 255

Name: test preparation course, dtype: int64

No. of test preparation course and not-test preparation course in the validation set:

Not\_test preparation course 197

test preparation course 103

Name: test preparation course, dtype: int64

15. Evaluation - There are many algorithms that could be used but we’re going to use logistic regression.[¶](file:///C:\Users\Soukhna\AppData\Local\Packages\Microsoft.MicrosoftEdge_8wekyb3d8bbwe\TempState\Downloads\11.2%20Exercise%20Original%20Case%20Study%20Part%203%20(3).html#15.-Evaluation---There-are-many-algorithms-that-could-be-used-but-we’re-going-to-use-logistic-regression.)

a. Metrics for the evaluation: i. Confusion Matrix (you should get 84% - pretty good) ii. Precision, Recall & F1 score (all 3 were very good) iii. ROC curve (the dotted line is the randomly guessed so anything above that is good metric)

In [595]:

# Step 15 - Eval Metrics

from sklearn.linear\_model import LogisticRegression

from yellowbrick.classifier import ConfusionMatrix

from yellowbrick.classifier import ClassificationReport

from yellowbrick.classifier import ROCAUC

# Instantiate the classification model

model = LogisticRegression()

#The ConfusionMatrix visualizer taxes a model

classes = ['Not\_test preparation course','test preparation course']

cm = ConfusionMatrix(model, classes=classes, percent=False)

#Fit fits the passed model. This is unnecessary if you pass the visualizer a pre-fitted model

cm.fit(X\_train, y\_train)

#To create the ConfusionMatrix, we need some test data. Score runs predict() on the data

#and then creates the confusion\_matrix from scikit learn.

cm.score(X\_val, y\_val)

C:\Users\Soukhna\Anaconda3\lib\site-packages\sklearn\base.py:197: FutureWarning: From version 0.24, get\_params will raise an AttributeError if a parameter cannot be retrieved as an instance attribute. Previously it would return None.

FutureWarning)

C:\Users\Soukhna\Anaconda3\lib\site-packages\sklearn\linear\_model\\_logistic.py:940: ConvergenceWarning: lbfgs failed to converge (status=1):

STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max\_iter) or scale the data as shown in:

https://scikit-learn.org/stable/modules/preprocessing.html

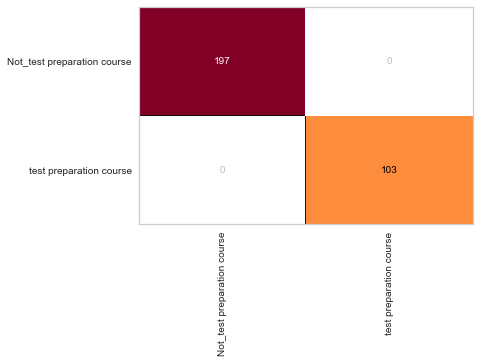
Please also refer to the documentation for alternative solver options:

https://scikit-learn.org/stable/modules/linear\_model.html#logistic-regression

extra\_warning\_msg=\_LOGISTIC\_SOLVER\_CONVERGENCE\_MSG)

Out[595]:

1.0



In [596]:

# change fontsize of the labels in the figure

for label in cm.ax.texts:

label.set\_size(20)

#How did we do?

cm.poof()

# Precision, Recall, and F1 Score

# set the size of the figure and the font size

#%matplotlib inline

plt.rcParams['figure.figsize'] = (15, 7)

plt.rcParams['font.size'] = 20

# Instantiate the visualizer

visualizer = ClassificationReport(model, classes=classes)

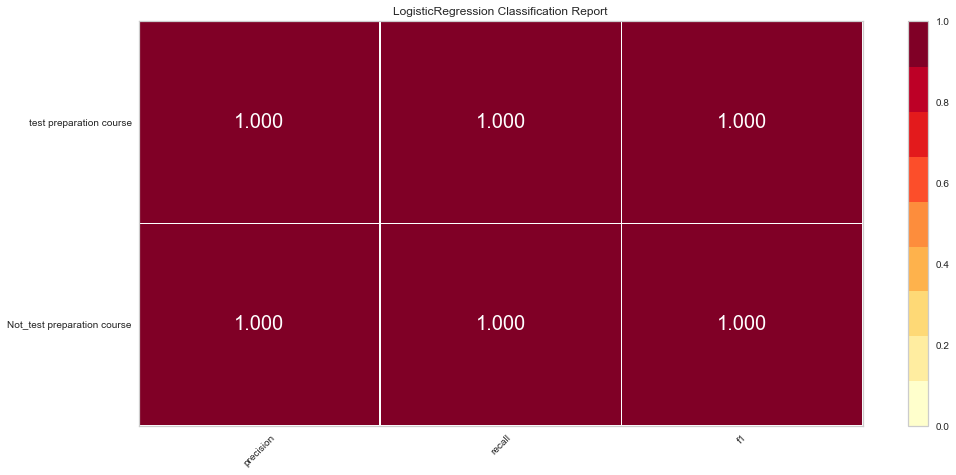
visualizer.fit(X\_train, y\_train) # Fit the training data to the visualizer

visualizer.score(X\_val, y\_val) # Evaluate the model on the test data

g = visualizer.poof()

C:\Users\Soukhna\Anaconda3\lib\site-packages\sklearn\base.py:197: FutureWarning: From version 0.24, get\_params will raise an AttributeError if a parameter cannot be retrieved as an instance attribute. Previously it would return None.

FutureWarning)



In [597]:

# ROC and AUC

#Instantiate the visualizer

visualizer = ROCAUC(model)

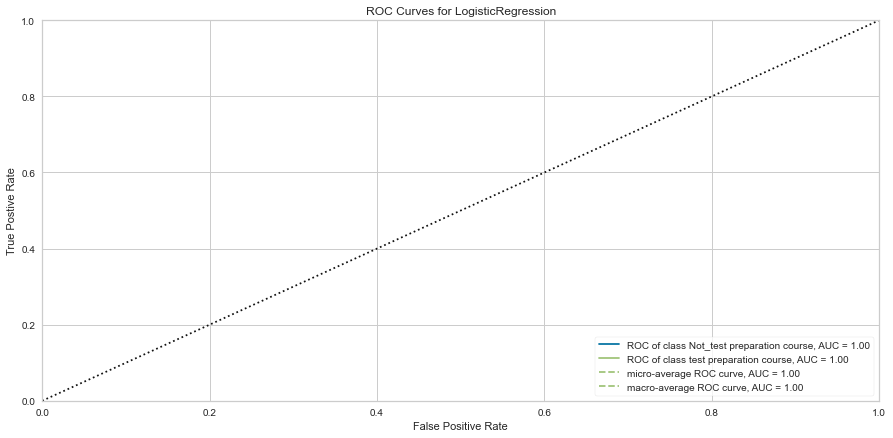
visualizer.fit(X\_train, y\_train) # Fit the training data to the visualizer

visualizer.score(X\_val, y\_val) # Evaluate the model on the test data

g = visualizer.poof()

C:\Users\Soukhna\Anaconda3\lib\site-packages\sklearn\base.py:197: FutureWarning: From version 0.24, get\_params will raise an AttributeError if a parameter cannot be retrieved as an instance attribute. Previously it would return None.

FutureWarning)



THE END