Importing data

In [121…
```
import pandas as pd
import numpy as np
```

In [122…
```
nilai = pd.read_csv(r"D:\SEMESTER 4\UTS\datmodelling\onsite-DataModeling_nilaiSiswa_UTS.csv", delimiter=',')
nilai.head(5)
```

Out[122]:

	gender	race	parental level of education	lunch	test preparation course	math score	reading score	writing score	total_score	result
0	female	group B	bachelor's degree	standard	none	72	72	74	72.67	PASS
1	female	group C	some college	standard	completed	69	90	88	82.33	PASS
2	female	group B	master's degree	standard	none	90	95	93	92.67	PASS
3	male	group A	associate's degree	free/reduced	none	47	57	44	49.33	FAIL
4	male	group C	some college	standard	none	76	78	75	76.33	PASS

In [123…
```
nilai.tail(2)
```

Out[123]:

	gender	race	parental level of education	lunch	test preparation course	math score	reading score	writing score	total_score	result
998	female	group D	some college	standard	completed	68	78	77	74.33	NaN
999	female	group D	some college	free/reduced	none	77	86	86	83.00	PASS

In [124…
```
nilai.info()
nilai.shape
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 10 columns):
 #   Column                       Non-Null Count  Dtype  
---  ------                       --------------  -----  
 0   gender                       1000 non-null   object 
 1   race                         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  
 8   total_score                  1000 non-null   float64
 9   result                       939 non-null    object 
dtypes: float64(1), int64(3), object(6)
memory usage: 78.2+ KB
```

Out[124]:
```
(1000, 10)
```

In [125…
```
nilai.describe()
```

Out[125]:

	math score	reading score	writing score	total_score
count	1000.00000	1000.000000	1000.000000	1000.000000
mean	66.08900	69.169000	68.054000	67.770580
std	15.16308	14.600192	15.195657	14.257311
min	0.00000	17.000000	10.000000	9.000000
25%	57.00000	59.000000	57.750000	58.330000
50%	66.00000	70.000000	69.000000	68.330000
75%	77.00000	79.000000	79.000000	77.670000
max	100.00000	100.000000	100.000000	100.000000

Question 1(fill the missing values)

In [126…
```
#menampilkan informasi untuk mengetahui kolum/atribut data yang memiliki data null (empty) dalam barisnya
print(nilai.isnull().sum())
```

```
gender                         0
race                           0
parental level of education    0
lunch                          0
test preparation course        0
math score                     0
reading score                  0
writing score                  0
total_score                    0
result                        61
dtype: int64
```

In [127…
```
#menggunakan condition dan apply untuk mengisi nilai PASS dan FAil yang masih NaN

nilai['result'] = nilai.apply(lambda row: 'PASS' if row['total_score'] > 60 else 'FAIL' if row['total_score'] i
nilai.head(20)
```

Out[127]:

	gender	race	parental level of education	lunch	test preparation course	math score	reading score	writing score	total_score	result
0	female	group B	bachelor's degree	standard	none	72	72	74	72.67	PASS
1	female	group C	some college	standard	completed	69	90	88	82.33	PASS
2	female	group B	master's degree	standard	none	90	95	93	92.67	PASS
3	male	group A	associate's degree	free/reduced	none	47	57	44	49.33	FAIL
4	male	group C	some college	standard	none	76	78	75	76.33	PASS
5	female	group B	associate's degree	standard	none	71	83	78	77.33	PASS
6	female	group B	some college	standard	completed	88	95	92	91.67	PASS
7	male	group B	some college	free/reduced	none	40	43	39	40.67	FAIL
8	male	group D	high school	free/reduced	completed	64	64	67	65.00	PASS
9	female	group B	high school	free/reduced	none	38	60	50	49.33	FAIL
10	male	group C	associate's degree	standard	none	58	54	52	54.67	FAIL
11	male	group D	associate's degree	standard	none	40	52	43	45.00	FAIL
12	female	group B	high school	standard	none	65	81	73	73.00	PASS
13	male	group A	some college	standard	completed	78	72	70	73.33	PASS
14	female	group A	master's degree	standard	none	50	53	58	53.67	FAIL
15	female	group C	some high school	standard	none	69	75	78	74.00	PASS
16	male	group C	high school	standard	none	88	89	86	87.67	PASS
17	female	group B	some high school	free/reduced	none	18	32	28	26.00	FAIL
18	male	group C	master's degree	free/reduced	completed	46	42	46	44.67	FAIL
19	female	group C	associate's degree	free/reduced	none	54	58	61	57.67	FAIL

In [128…
```
#checking whether is it already fill or not
nilai.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 10 columns):
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0   gender                       1000 non-null   object
 1   race                         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
 8   total_score                  1000 non-null   float64
 9   result                       1000 non-null   object
dtypes: float64(1), int64(3), object(6)
memory usage: 78.2+ KB
```

Question 2

find the average of each score of each course, then display the highest and lowest mean of total_score(using groupby)

In [129…
```
# melihat average score dari setiap mata pelajaran

averagescorecourse = nilai.groupby('test preparation course')['total_score'].mean()
print(averagescorecourse)
```

```
test preparation course
completed    72.669469
none         65.038801
Name: total_score, dtype: float64
```

bisa dilihat bahwa total_score untuk completed dan none 72.6694 sekian dan none 65.038801

In [130…
```
# tampilkan nilai paling tinggi dan paling rendah mean dari total_score!!!

tertinggi = averagescorecourse.max()
terendah = averagescorecourse.min()

print(f'nilai terendah untuk averagenya adalah:  {terendah:2f}')
print(f'nilai tertinggi untuk averagenya adalah:  {tertinggi:2f}')
```

```
nilai terendah untuk averagenya adalah:  65.038801
nilai tertinggi untuk averagenya adalah:  72.669469
```

Question 3

a.use the same dataset that has no missing values

In [131…
```
samedata = nilai.copy()
samedata.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 10 columns):
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0   gender                       1000 non-null   object
 1   race                         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
 8   total_score                  1000 non-null   float64
 9   result                       1000 non-null   object
dtypes: float64(1), int64(3), object(6)
memory usage: 78.2+ KB
```

In [132…
```
#encoding

def encode_data(feature_name):
    mapping_dict = {}

    unique_values = list(samedata[feature_name].unique())


    for idx in range(len(unique_values)):

        mapping_dict[unique_values[idx]] = idx

    return mapping_dict
```

```
samedata['gender'].replace({'male':0,'female': 1}, inplace = True)

samedata['race'].replace({'group A':0,'group B': 1, 'group C':2, 'group D':3, 'group E':4}, inplace = True)

samedata['parental level of education'].replace(encode_data('parental level of education'), inplace = True)

samedata['lunch'].replace(encode_data('lunch'), inplace = True)

samedata['test preparation course'].replace(encode_data('test preparation course'), inplace = True)

samedata['result'].replace({'FAIL':0,'PASS': 1}, inplace = True)
```

In [133…
```
#checking the data again is it already encoded or not?
samedata.info()
```

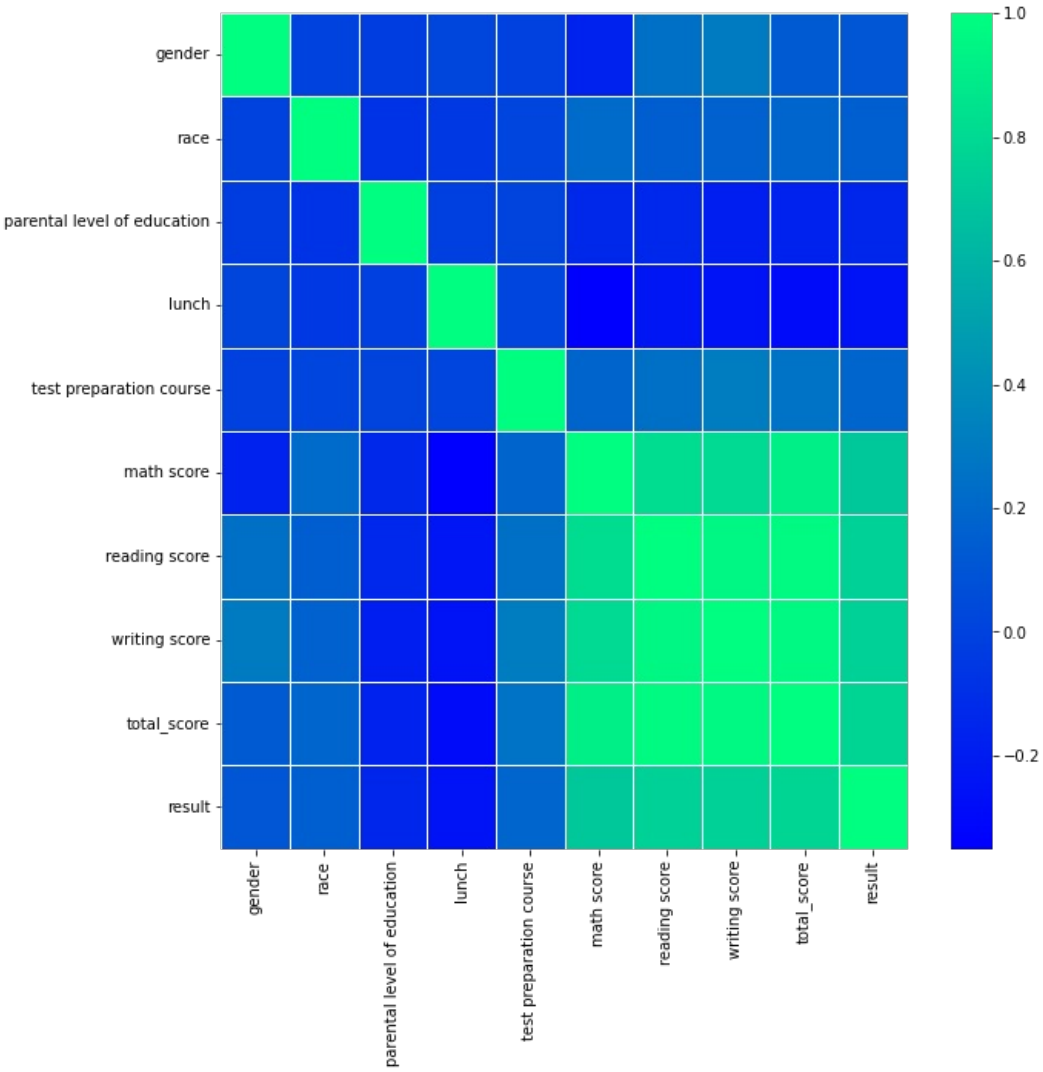
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 10 columns):
 #   Column                       Non-Null Count  Dtype  
---  ------                       --------------  -----  
 0   gender                       1000 non-null   int64  
 1   race                         1000 non-null   int64  
 2   parental level of education  1000 non-null   int64  
 3   lunch                        1000 non-null   int64  
 4   test preparation course      1000 non-null   int64  
 5   math score                   1000 non-null   int64  
 6   reading score                1000 non-null   int64  
 7   writing score                1000 non-null   int64  
 8   total_score                  1000 non-null   float64
 9   result                       1000 non-null   int64  
dtypes: float64(1), int64(9)
memory usage: 78.2 KB
```

In [134…
```
#import more libraries
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
```

visualisasi biar keren

In [135…
```
plt.figure(figsize=(10,10))
sns.heatmap(samedata.corr(), linewidth=0.5, annot=False, fmt=".2f", cmap = 'winter')
```

Out[135]: <AxesSubplot:>



b.divide training and test data 80:20

In [142…
```
#menentukan nilai x dan y untuk prediksi apakah lulus(PASS) atau tidak(FAIL)

X = samedata.drop(['result'], axis= 1)
y = samedata['result']

X_train, X_test, y_train, y_test = train_test_split(X,y, test_size = 0.2)

print("Panjang dari Training Data: {}".format(len(X_train)))
print("Panjang dari Testing Data: {}".format(len(X_test)))
```

```
Panjang dari Training Data: 800
Panjang dari Testing Data: 200
```

c.use classification algorithm

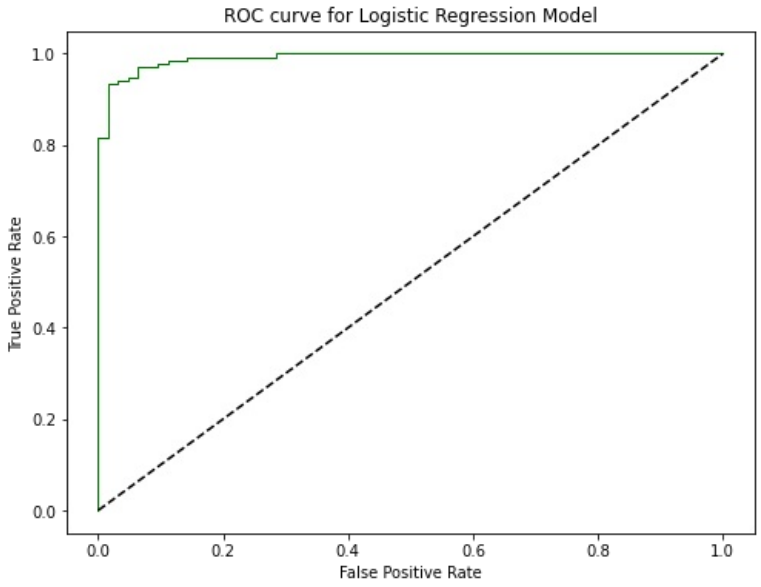
In [143…
```
#memasukkan data ke dalam model Logistic Regression
logregression = LogisticRegression(solver='liblinear')
logregression.fit(X_train, y_train)
```

Out[143]:
```
LogisticRegression(solver='liblinear')
```

d. model evalution on logistic regression

In [144…
```
#evaluating data using ROC Curve
#hasil yang kemungkinan terjadi pada model evaluation ini adalah

#True Positive: prediksi lulus, dan beneran PASS
#True Negative: prediksi tidak lulus, dan beneran FAIL

print("Accuracy : ", logregression.score(X_test, y_test))

from sklearn. metrics import roc_curve

y_pred_logreg_proba = logregression.predict_proba(X_test)[:,1]
fpr, tpr, thresholds = roc_curve(y_test, y_pred_logreg_proba)
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, '-g', linewidth=1)
plt.plot([0,1],[0,1], 'k--')
plt.title('ROC curve for Logistic Regression Model')
plt.xlabel("False Positive Rate")
plt.ylabel('True Positive Rate')
plt.show()
```

```
Accuracy :  0.95
```



In [145…
```
!jupyter nbconvert --to html "./Christopher Darren_00000054804_IS411_CLHY_UTS" --output-dir="./"
```

```
This application is used to convert notebook files (*.ipynb)
        to various other formats.

        WARNING: THE COMMANDLINE INTERFACE MAY CHANGE IN FUTURE RELEASES.

Options
=======
The options below are convenience aliases to configurable class-options,
as listed in the "Equivalent to" description-line of the aliases.
To see all configurable class-options for some <cmd>, use:
    <cmd> --help-all

--debug
    set log level to logging.DEBUG (maximize logging output)
    Equivalent to: [--Application.log_level=10]
--show-config
```

```
    Show the application's configuration (human-readable format)
    Equivalent to: [--Application.show_config=True]
--show-config-json
    Show the application's configuration (json format)
    Equivalent to: [--Application.show_config_json=True]
--generate-config
    generate default config file
    Equivalent to: [--JupyterApp.generate_config=True]
-y
    Answer yes to any questions instead of prompting.
    Equivalent to: [--JupyterApp.answer_yes=True]
--execute
    Execute the notebook prior to export.
    Equivalent to: [--ExecutePreprocessor.enabled=True]
--allow-errors
    Continue notebook execution even if one of the cells throws an error and include the error message in the c
ell output (the default behaviour is to abort conversion). This flag is only relevant if '--execute' was specif
ied, too.
    Equivalent to: [--ExecutePreprocessor.allow_errors=True]
--stdin
    read a single notebook file from stdin. Write the resulting notebook with default basename 'notebook.*'
    Equivalent to: [--NbConvertApp.from_stdin=True]
--stdout
    Write notebook output to stdout instead of files.
    Equivalent to: [--NbConvertApp.writer_class=StdoutWriter]
--inplace
    Run nbconvert in place, overwriting the existing notebook (only
            relevant when converting to notebook format)
    Equivalent to: [--NbConvertApp.use_output_suffix=False --NbConvertApp.export_format=notebook --FilesWriter.
build_directory=]
--clear-output
    Clear output of current file and save in place,
            overwriting the existing notebook.
    Equivalent to: [--NbConvertApp.use_output_suffix=False --NbConvertApp.export_format=notebook --FilesWriter.
build_directory= --ClearOutputPreprocessor.enabled=True]
--no-prompt
    Exclude input and output prompts from converted document.
    Equivalent to: [--TemplateExporter.exclude_input_prompt=True --TemplateExporter.exclude_output_prompt=True]
--no-input
    Exclude input cells and output prompts from converted document.
            This mode is ideal for generating code-free reports.
    Equivalent to: [--TemplateExporter.exclude_output_prompt=True --TemplateExporter.exclude_input=True --Templ
ateExporter.exclude_input_prompt=True]
--allow-chromium-download
    Whether to allow downloading chromium if no suitable version is found on the system.
    Equivalent to: [--WebPDFExporter.allow_chromium_download=True]
--disable-chromium-sandbox
    Disable chromium security sandbox when converting to PDF..
    Equivalent to: [--WebPDFExporter.disable_sandbox=True]
--show-input
    Shows code input. This flag is only useful for dejavu users.
    Equivalent to: [--TemplateExporter.exclude_input=False]
--embed-images
    Embed the images as base64 dataurls in the output. This flag is only useful for the HTML/WebPDF/Slides expo
rts.
    Equivalent to: [--HTMLExporter.embed_images=True]
--log-level=<Enum>
    Set the log level by value or name.
    Choices: any of [0, 10, 20, 30, 40, 50, 'DEBUG', 'INFO', 'WARN', 'ERROR', 'CRITICAL']
    Default: 30
    Equivalent to: [--Application.log_level]
--config=<Unicode>
    Full path of a config file.
    Default: ''
    Equivalent to: [--JupyterApp.config_file]
--to=<Unicode>
    The export format to be used, either one of the built-in formats
            ['asciidoc', 'custom', 'html', 'latex', 'markdown', 'notebook', 'pdf', 'python', 'rst', 'script', '
slides', 'webpdf']
            or a dotted object name that represents the import path for an
            ``Exporter`` class
    Default: ''
    Equivalent to: [--NbConvertApp.export_format]
--template=<Unicode>
    Name of the template to use
    Default: ''
    Equivalent to: [--TemplateExporter.template_name]
--template-file=<Unicode>
    Name of the template file to use
    Default: None
    Equivalent to: [--TemplateExporter.template_file]
--theme=<Unicode>
    Template specific theme(e.g. the name of a JupyterLab CSS theme distributed
    as prebuilt extension for the lab template)
    Default: 'light'
    Equivalent to: [--HTMLExporter.theme]
--writer=<DottedObjectName>
    Writer class used to write the
                                        results of the conversion
```

```
    Default: 'FilesWriter'
    Equivalent to: [--NbConvertApp.writer_class]
--post=<DottedOrNone>
    PostProcessor class used to write the
                                        results of the conversion
    Default: ''
    Equivalent to: [--NbConvertApp.postprocessor_class]
--output=<Unicode>
    overwrite base name use for output files.
                can only be used when converting one notebook at a time.
    Default: ''
    Equivalent to: [--NbConvertApp.output_base]
--output-dir=<Unicode>
    Directory to write output(s) to. Defaults
                                    to output to the directory of each notebook. To recover
                                    previous default behaviour (outputting to the current
                                    working directory) use . as the flag value.
    Default: ''
    Equivalent to: [--FilesWriter.build_directory]
--reveal-prefix=<Unicode>
    The URL prefix for reveal.js (version 3.x).
            This defaults to the reveal CDN, but can be any url pointing to a copy
            of reveal.js.
            For speaker notes to work, this must be a relative path to a local
            copy of reveal.js: e.g., "reveal.js".
            If a relative path is given, it must be a subdirectory of the
            current directory (from which the server is run).
            See the usage documentation
            (https://nbconvert.readthedocs.io/en/latest/usage.html#reveal-js-html-slideshow)
            for more details.
    Default: ''
    Equivalent to: [--SlidesExporter.reveal_url_prefix]
--nbformat=<Enum>
    The nbformat version to write.
            Use this to downgrade notebooks.
    Choices: any of [1, 2, 3, 4]
    Default: 4
    Equivalent to: [--NotebookExporter.nbformat_version]

Examples
--------

    The simplest way to use nbconvert is

            > jupyter nbconvert mynotebook.ipynb --to html

            Options include ['asciidoc', 'custom', 'html', 'latex', 'markdown', 'notebook', 'pdf', 'python', 'r
st', 'script', 'slides', 'webpdf'].

            > jupyter nbconvert --to latex mynotebook.ipynb

            Both HTML and LaTeX support multiple output templates. LaTeX includes
            'base', 'article' and 'report'.  HTML includes 'basic', 'lab' and
            'classic'. You can specify the flavor of the format used.

            > jupyter nbconvert --to html --template lab mynotebook.ipynb

            You can also pipe the output to stdout, rather than a file

            > jupyter nbconvert mynotebook.ipynb --stdout

            PDF is generated via latex

            > jupyter nbconvert mynotebook.ipynb --to pdf

            You can get (and serve) a Reveal.js-powered slideshow

            > jupyter nbconvert myslides.ipynb --to slides --post serve

            Multiple notebooks can be given at the command line in a couple of
            different ways:

            > jupyter nbconvert notebook*.ipynb
            > jupyter nbconvert notebook1.ipynb notebook2.ipynb

            or you can specify the notebooks list in a config file, containing::

                c.NbConvertApp.notebooks = ["my_notebook.ipynb"]

            > jupyter nbconvert --config mycfg.py

To see all available configurables, use `--help-all`.
```

```
[NbConvertApp] WARNING | pattern './Template Laporan Tugas Mingguan Lab IF540 Genap2223.ipynb' matched no files
```

In [ ]: