In [105…

```
# Badr TADJER | Naoufal ARRADI | Leo TRAN
# M1-LS1-APP
```

Assignment 2

In this Assignment, you will explore the **FIFA 19** dataset, which contains detailed attributes for every player registered in the latest edition of FIFA 19 database.
It was scrapped from this website, and you can find the source code here.

Your goal is to find out how the `Overall` score by player is calculated

There are 2 main parts:

- Data Import & Cleaning, the output of this part is provided in the csv file `Assignment_2_data_cleaned`, **so you do not need to have everything right in this part to do the second one**
- Modeling with the cleaned data

In the notebook, there will be cells in the form `assert condition` like the next one. They are used to check if an answer is correct. Execute the next one and you will get no errors

In [106…

```
assert 3 < 5
```

Execute the next one and you will get an error

In [107…

```
assert 3 > 5
```

```
---------------------------------------------------------------------------
AssertionError                            Traceback (most recent call last)
Input In [107], in <module>
----> 1 assert 3 > 5

AssertionError:
```

Data Import

In [108…

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

1.1. Load the csv file `data.csv` into a dataframe called `df` and print its shape. (Set the right parameters when reading the csv file)

In [109…

```
df = pd.read_csv('Assignment_2_data.csv', index_col = 0)
```

In [110…

```
# check if your answer is correct
assert df.shape == (18207, 88)
```

1.2. print the head of `df`

In [111…

```
df.head()
```

Out[111]:

	ID	Name	Age	Photo	Nationality	Flag	Overall	Potential	Cl
0	158023	L. Messi	31	https://cdn.sofifa.org/players/4/19/158023.png	Argentina	https://cdn.sofifa.org/flags/52.png	94	94	Barcelo
1	20801	Cristiano Ronaldo	33	https://cdn.sofifa.org/players/4/19/20801.png	Portugal	https://cdn.sofifa.org/flags/38.png	94	94	Juvent
2	190871	Neymar Jr	26	https://cdn.sofifa.org/players/4/19/190871.png	Brazil	https://cdn.sofifa.org/flags/54.png	92	93	Paris Sai Germ
3	193080	De Gea	27	https://cdn.sofifa.org/players/4/19/193080.png	Spain	https://cdn.sofifa.org/flags/45.png	91	93	Manches Unit
4	192985	K. De Bruyne	27	https://cdn.sofifa.org/players/4/19/192985.png	Belgium	https://cdn.sofifa.org/flags/7.png	91	92	Manches C

5 rows × 88 columns

1.3. Print how many columns that are in df columns types

In [112…

```
df.dtypes.value_counts()
```

Out[112]:

```
object     45
float64    38
int64       5
dtype: int64
```

1.4. `to_drop` is a list containing columns that are not useful for modeling, remove them and print the new shape of `df`

In [113…

```
to_drop =['ID', 'Name', 'Photo','Nationality', 'Flag','Club','Club Logo', 'Real Face', 'Joined', 'Loaned From'
# your code here
df = df.drop(columns = to_drop)
```

In [114…

```
# check if your answer is correct
assert df.shape == (18207, 77)
```

Data Cleaning

Handling missing values

2.1. Build a dataframe called `missing` which has the following format:

- `pct` is the percentage of missing values, **takes values between `0` and `100`**
- the index is the column names

	pct
......	.....
Strength	0.263635
.....	.....

In [115…

```
missing = pd.DataFrame((df.isnull().sum() / len(df)) * 100, columns={"pct"})
missing
```

Out[115]:

	pct
Age	0.000000
Overall	0.000000
Potential	0.000000
Value	0.000000
Wage	0.000000
...	...
GKHandling	0.263635
GKKicking	0.263635
GKPositioning	0.263635
GKReflexes	0.263635
Release Clause	8.590103

77 rows × 1 columns

2.2. Remove from `missing`, rows with `pct` = 0
sort `missing` in ascending order of `pct` and print its head

In [116…

```
missing = missing[missing.pct != 0]
missing = missing.sort_values(by = ['pct'])
missing
```

Out[116]:

	pct
Preferred Foot	0.263635
Strength	0.263635
Stamina	0.263635
Jumping	0.263635
ShotPower	0.263635
...	...
RWB	11.451639
LB	11.451639
LCB	11.451639
RCB	11.451639
RB	11.451639

71 rows × 1 columns

2.3. Now, let's fill missing values where the % of missing is lower than 1 (1%).
First identify these columns in a list named `cols_to_fill`

In [117…

```
# your code here
cols_to_fill = missing.index[missing['pct'] < 1].tolist()
print(len(cols_to_fill), type(cols_to_fill))
```

```
44 <class 'list'>
```

In [118…

```
# check if your answer is correct
assert len(cols_to_fill) == 44; assert isinstance(cols_to_fill, list)
```

2.4. define a function (`fill_nas_by_type`) to fill null values by column type:

- if a column type is `Object`, fill it with the **most frequent value**
- otherwise, fill it with the **median value**

In [119…

```
def fill_nas_by_type(df, col_name):
    """Fill null values in df according to col_name type

    Parameters
    ----------
    df : dataframe, (default=None)
        input dataframe
    col_name : str, (default=None)
        column with null values to fill

    Returns
    -------
    df with filled values in col_name
    """
    if df[col_name].dtypes == 'object':
        df.loc[df[col_name].isnull(), col_name] = df[col_name].mode
    else:
        df.loc[df[col_name].isnull(), col_name] = df[col_name].median()

    return df
```

Loop through `cols_to_fill` and apply the defined function `fill_nas_by_type` to fill null values

In [120…

```
for col_name in cols_to_fill:
  fill_nas_by_type(df, col_name)
```

In [121…

```
# check if your answer is correct
assert df[cols_to_fill].isnull().sum().sum() == 0
```

For the remaining missing values, let's just remove them.
Print the shape of `df` before and after removing any rows with missing observations

In [122…

```
df.shape
df = df.dropna()
df.shape
```

Out[122]:

```
(14743, 77)
```

In [123…

```
# check if your answer is correct
assert df.shape == (14743, 77); assert df.isnull().sum().sum() == 0
```

Correct some columns format

Monetary columns

In [124…

```
money_cols = ['Value','Wage', 'Release Clause']
df[money_cols].head()
```

Out[124]:

	Value	Wage	Release Clause
0	€110.5M	€565K	€226.5M
1	€77M	€405K	€127.1M
2	€118.5M	€290K	€228.1M
4	€102M	€355K	€196.4M
5	€93M	€340K	€172.1M

3.1. Build a function which extracts the monetary value from a string. It should return a number with no decimals.
Your function should pass the three tests in the cell after

In [125…

```
def get_value(value_text):
    """Extract the monetary value from a string

    Parameters
    ----------
    value_text: str, (default=None)
        a string containing a number ending with M, K or nothing

    Returns
    -------
    a float with no decimals

    Examples
    --------
    >>> get_value('€7.1K')
    7100.0
    """
    multiplier = value_text[-1]
    if multiplier == 'M':
        number = float(value_text[1:-1])
        return number * 1000000
    elif multiplier == 'K':
        number = float(value_text[1:-1])
        return number * 1000
    else:
        return float(value_text[1:])
```

In [126…

```
# check if your answer is correct
assert get_value('€110.5M') == 110500000; assert get_value('€7.1K') == 7100; assert get_value('€200') == 200
```

3.2. Loop through `money_cols` and apply the defined function `get_value` to convert them to numeric

In [127…

```
for f in money_cols:
    df[f] =df[f].apply(get_value)
    print(f, df[f].dtype, df[f].isnull().sum())
```

```
Value float64 0
Wage float64 0
Release Clause float64 0
```

In [128…

```
# check if your answer is correct
assert df[money_cols].isnull().sum().sum() == 0
```

Height and Weight columns

4.1. Start by printing the unique values for `Height`

In [129…

```
# print unique values for Height
df['Height'].unique()
```

Out[129]:

```
array(["5'7", "6'2", "5'9", "5'11", "5'8", "6'0", "5'6", "5'10", "6'1",
       "5'4", "6'3", "6'4", "5'5", "6'6", "6'5", "5'3", "5'2", "6'7",
       "5'1", "6'8"], dtype=object)
```

4.2. Write a function (`get_height`) which converts the Height from a string in feet to a number in `cm` with no decimals.
1 feet = 30.48 cm. For example `get_height("5'10")` = `155`

In [130…

```
def get_height(x):
    return  round(float(x.replace("'", ".")) * 30.48, 0)
```

In [131…

```
# check if your answer is correct
assert get_height("5'10") == 155; assert get_height("6'8") == 207
```

Apply the previous defined function on `Height`

In [132…

```
df['Height'] = df['Height'].apply(get_height)
```

In [133…

```
# check if your answer is correct
assert df['Height'].dtype == 'float64'; assert df['Height'].isnull().sum() == 0
```

4.3. The same thing with `Weight`, print the unique values

In [134…

```
# print unique values for Weight
df['Weight'].unique()
```

Out[134]:

```
array(['159lbs', '183lbs', '150lbs', '154lbs', '163lbs', '146lbs',
       '190lbs', '181lbs', '176lbs', '168lbs', '172lbs', '148lbs',
       '165lbs', '196lbs', '161lbs', '170lbs', '187lbs', '157lbs',
       '185lbs', '130lbs', '174lbs', '203lbs', '207lbs', '134lbs',
       '141lbs', '152lbs', '179lbs', '132lbs', '198lbs', '201lbs',
       '209lbs', '214lbs', '143lbs', '192lbs', '137lbs', '194lbs',
       '139lbs', '220lbs', '205lbs', '216lbs', '126lbs', '123lbs',
       '128lbs', '223lbs', '212lbs', '121lbs', '115lbs', '218lbs',
       '117lbs', '243lbs', '110lbs', '119lbs', '234lbs'], dtype=object)
```

4.4. Write a function (`get_weight`) which converts the **Weight** from a string in `lbs` to a number in `kg` with no decimals.
1 lbs = 0.453592 kg. For example `get_weight("115lbs")` = `52`

In [135…

```
def get_weight(x):
    return  round(float(x.split('lbs')[0]) * 0.453592, 0)
```

In [136…

```
# check if your answer is correct
assert get_weight("115lbs") == 52; assert get_weight("234lbs") == 106
```

Apply the previous defined function on `Weight`

In [137…

```
df['Weight'] = df['Weight'].apply(get_weight)
```

In [138…

```
# check if your answer is correct
assert df['Weight'].dtype == 'float64'; assert df['Weight'].isnull().sum() == 0
```

Convert text columns to numeric

5.1. Identify non-numeric text columns in a list called `text_cols`

In [139…

```
# your code here
text_cols = text_cols = [item for item in df if df.dtypes[item] == object]
print(len(text_cols))
```

```
30
```

5.2. Build a list named `cols_to_remove` containing columns from `text_cols`, if a column has a number of unique values greater than **10** (`> 10`)

In [140…

```
# your code here
cols_to_remove = [item for item in text_cols if len(df[item].unique()) > 10]
print(len(cols_to_remove))
```

```
27
```

remove `cols_to_remove` columns from `df` and print its shape

In [141…

```
# your code here
df = df.drop(columns = cols_to_remove)
```

In [142…

```
# check if your answer is correct
assert df.shape == (14743, 50)
```

5.3. Identify the remaining text columns in `text_cols` as `remaining_text_cols`, make sur it passes the test after

In [143…

```
# your code here
remaining_text_cols = [item for item in text_cols if item not in cols_to_remove]
print(len(remaining_text_cols))
```

```
3
```

In [144…

```
# check if your answer is correct
assert remaining_text_cols == ['Preferred Foot', 'Work Rate', 'Body Type']
```

5.4. Loop through `remaining_text_cols` and convert them to numerical values

In [145…

```
# your code here
for column in remaining_text_cols:
    values_list = df[column].unique()
    df[column] = df[column].map({value: key for key, value in enumerate(values_list)})
```

In [146…

```
df.shape
```

Out[146]:

```
(14743, 50)
```

Model building

As stated before, you can do this part without completing the previous one

6.1. Load the cleaned dataset `Assignment_2_data_cleaned.csv` into `df_clean` and print its shape.

In [147…

```
# your code here
df_clean = pd.read_csv("Assignment_2_data_cleaned.csv")
df_clean.shape
```

Out[147]:

```
(14743, 50)
```

6.2. Load the target variable `Overall` into a dataframe and name it `y`. Then, load the features into a second dataframe and name it `X`. Plot a histogram of `y`, choose the number of bins as 100.

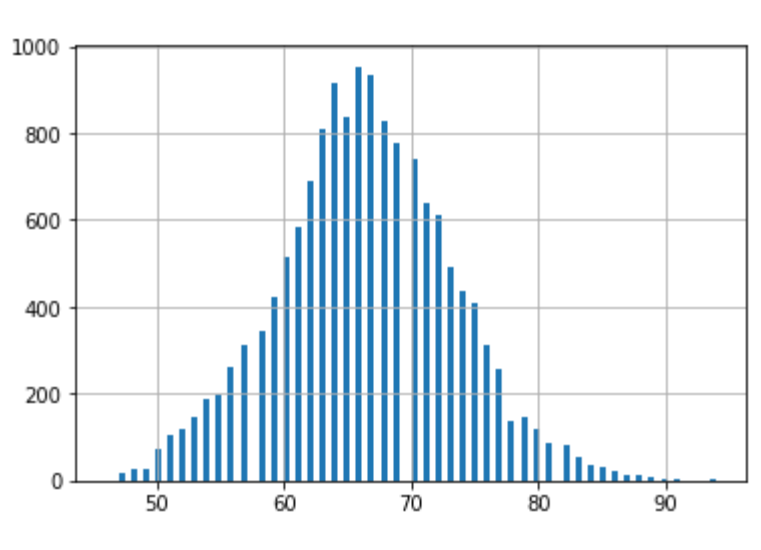
In [148…

```
# your code here
y = df_clean['Overall']
X = df_clean.drop(columns = 'Overall')

y.hist(bins = 100)
```

Out[148]:

```
<AxesSubplot:>
```



1. Split the data set into a training set and a test set. Choose `test_size` = 0.3 and `random_state` = 123
   Print train and test size
   Attention: You are asked to use `sklearn.model_selection`

In [149…

```
# your code here
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state = 123)

print(X_train.shape)
print(X_test.shape)
```

```
(10320, 49)
(4423, 49)
```

1. Fit a linear model to the training set, and then report the training and testing errors obtained (the R2 statistic).
   Calculate and print the following metrics: mse, rmse, mae for the test_set

In [150…

```
# your code here
from sklearn.linear_model import LinearRegression

model = LinearRegression()
model.fit(X_train, y_train)

y_prediction = model.predict(X_test)
```

In [151…

```
# your code here
import math
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import r2_score

r2 = r2_score(y_test, y_prediction)
mse = mean_squared_error(y_test, y_prediction)
rmse = mean_squared_error(y_test, y_prediction, squared = False)
mea = mean_absolute_error(y_test, y_prediction)

print("r2 score : ", r2)
print("mse score : ", mse)
print("rmse score : ", rmse)
print("mea score : ", mea)

print('r2 = {}, mse = {} \rmse = {} mea = {}'.format(r2, mse, rmse, mea))
```

```
r2 score :  0.9357188804619977
mse score :  3.0074136038322967
rmse score :  1.734189610115427
mea score :  1.3561926131562998
mse = 1.734189610115427 mea = 1.35619261315629987
```

Check residuals

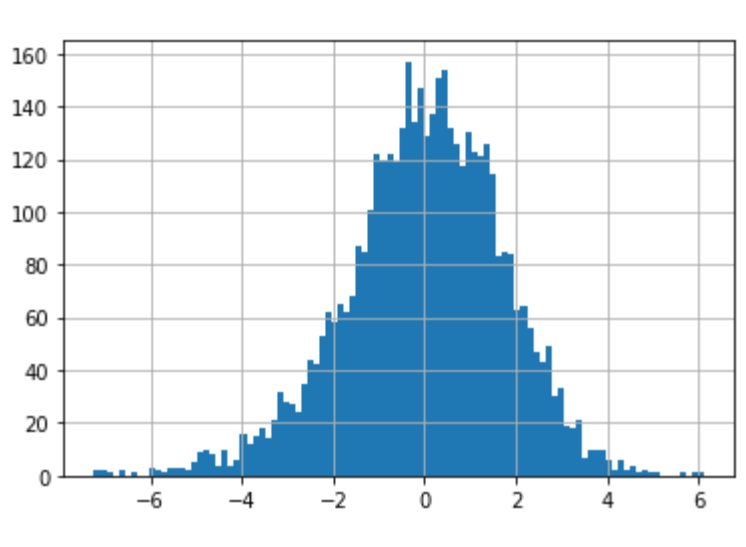
9.1. Plot a histogram of the residuals (difference between `y_test` and `y_pred`

In [152…

```
# your code here
residuals = y_test - y_prediction
residuals.hist(bins = 100)
```

Out[152]:

```
<AxesSubplot:>
```



9.2. Plot a scatter plot where `y_test` is in the **x** axis and `y_pred` is in the **y** axis

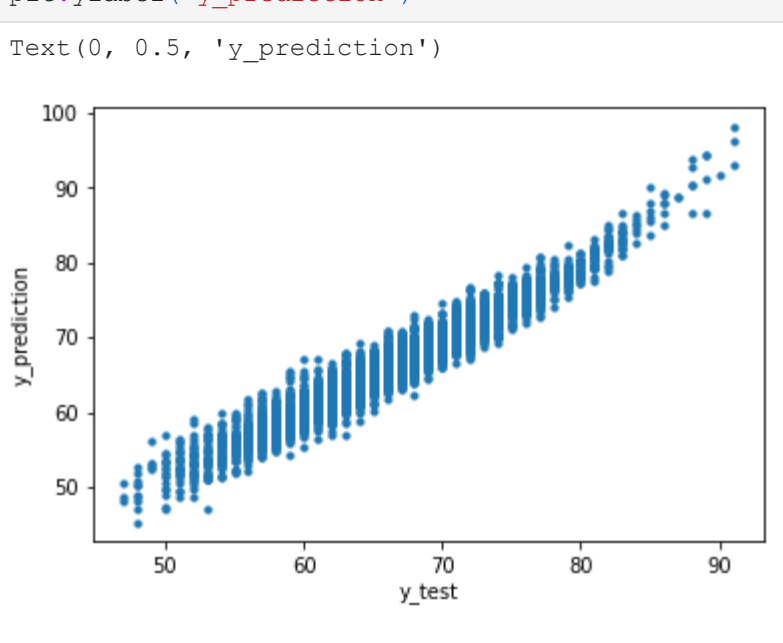
In [153…

```
# your code here
import matplotlib.pyplot as plt

plt.scatter(x = y_test, y = y_prediction, s = 10)
plt.xlabel('y_test')
plt.ylabel('y_prediction')
```

Out[153]:

```
Text(0, 0.5, 'y_prediction')
```



1. Try to improve the performance of your model, by adding new features

In [154…

```
# your code here
```