Tao Li

Student number: 220106979

I declare that I have personally prepared this assignment. The work is my own, carried out personally by me unless otherwise stated and has not been generated using Artificial Intelligence tools unless specified as a clearly stated approved component of the assessment brief. All sources of information, including quotations, are acknowledged by means of the appropriate citations and references. I declare that this work has not gained credit previously for another module at this or another University. I understand that plagiarism, collusion, copying another student and commissioning (which for the avoidance of doubt includes the use of essay mills and other paid for assessment writing services, as well as unattributed use of work generated by Artificial Intelligence tools) are regarded as offences against the University's Assessment Regulations and may result in formal disciplinary proceedings. I understand that by submitting this assessment, I declare myself fit to be able to undertake the assessment and accept the outcome of the assessment as valid.

Part A: Recommender system

Question 1) Import the data "train.txt" and identify the total number Users and the total number of Products.

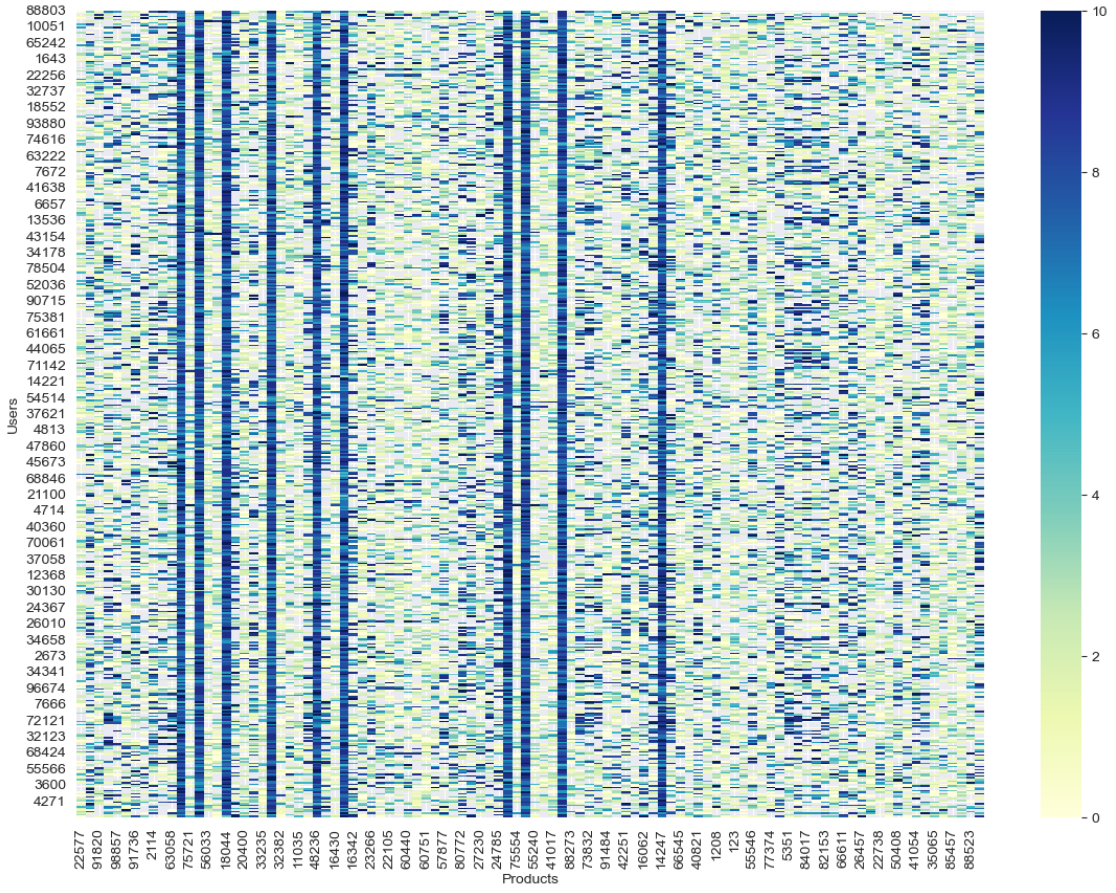
>> number of users is 500, number of Products 100

Question 2) Create a data frame (or matrix, or array) Y which consists of the ratings, such that y_{nd} represents the rating given by User n to Product d.

1. What are the dimensions of Y?

    > The dimensions of Y is 500 * 100

2. Plot the matrix Y



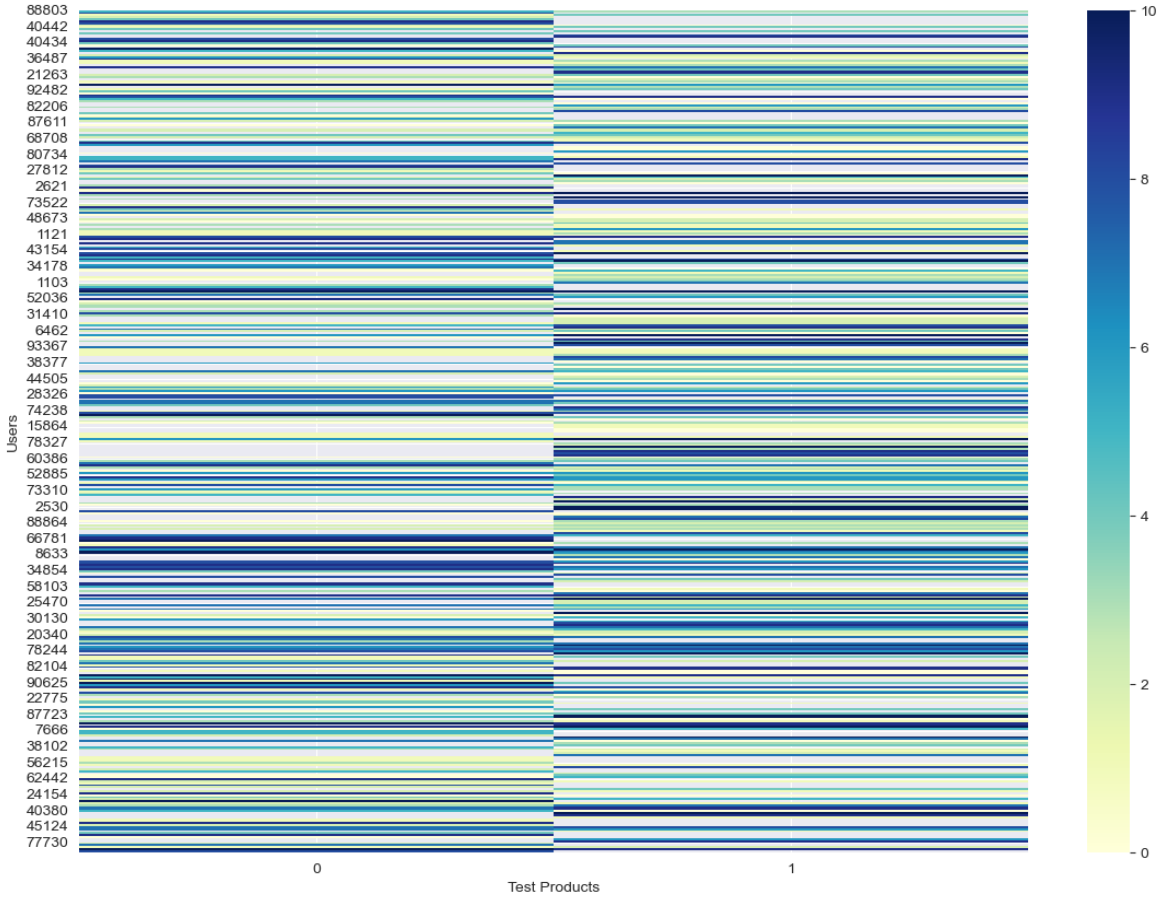
Question 3) Find the top 5 Products. Explain your method and results.

> Top 5 products is 18044,14247,71562,48236,7343.My method is to group the data by product and calculate the mean of the rating. Then sort the products by their average rating and print the top 5 products.

Question 4) Import the data “test.txt” and identify the total number Users and the total number of Products.

> number of users is 421, number of Products is 2

Question 5) Create a data frame (or matrix, or array) X which consists of the ratings, such that X_{nd} represents the rating given by User n to Product d. What are the dimensions of x?



> Dimensions of X: (421, 2)

Question 6) For each Product in "test.txt", find the most similar product in "train.txt". To compute the distance between Product n (from $\mathbf{X}$) and Product m (from $\mathbf{Y}$) use the following
$For i = 1, \ldots, N_{user}$

If x_{in} and y_{im} exist $d_{nm} = d_{nm} + (x_{in} - y_{im})^2$

Where N_{user} is the total number of users, and d_{nm} represents the distance between Product n (from $\mathbf{X}$) and Product m (from $\mathbf{Y}$).\

> I got an matix shape as (2*100), calculate the distance between each product in test.txt and train.txt.

Question 7) Using the method from Question 6) Find the top 5 similar product in “train.txt”, for each Product in “test.txt”.

> test product 0 similar to top 5 train product is [50408,58577,26457,60734,38851] , test product 1 similar to top 5 train product is [24785,26457,40821,50408,41232]

Question 8) Explain the main limitation of the method in Question 6)

> The main limitation of the method in Question 6) is that it only considers the rating of the users who rated both the products. It does not consider the rating of the users who rated only one of the products. This can be improved by considering the rating of the users who rated only one of the products by using the average rating of the product as the rating of the user for the product.

Question 9) Propose an alternative to the distance method described in Question 6) by completing:
$For i = 1, \ldots, N_{\max}$
$If \# Answer here$
$d_{nm} = d_{nm} + (x_{in} - y_{im})^2$
$If \# Answer here$
$d_{nm} = d_{nm} + \#Answer$ here

> my asnswer is If x_{in} and y_{im} exist $d_{nm} = d_{nm} + (x_{in} - y_{im})^2$ else if x_{in} exist $d_{nm} = d_{nm} + (x_{in} - \bar{y}_m)^2$ elif If y_{im} exist $d_{nm} = d_{nm} + (\bar{x}_n - y_{im})^2$

Question 10) Using your solution to Question 10), find the top 5 similar products in "train.txt", for each Product in "test.txt". Marking scheme

> test product 0 similar to top 5 train product is [91736,80772,42251,16062,41232], test product 1 similar to top 5 train product is [66611,26457,16062,42251,41232]

Part B: University Module Data

Title: In-depth Analysis and Projection of Student Performance in the AM41AB and AM41MN Modules over a Five-Year Period

Synopsis: This report focuses on analysing student performance in two distinct university modules: AM41AB and AM41MN, over a span of five years (2015-2019). The objective is not only to understand the past performance trends but also to project the average marks for each module for the subsequent year, 2020. The analysis considers several influencing factors including, but not limited to, the distribution of marks, teaching consistency, and variations in marking criteria.

Data Overview:

The data set comprises 12 files, detailing student performance for each module from 2015 to 2019. Among these, the 'Information.txt' file was found to be superfluous, while the 'moduleInformation.xlsx' file included valuable information about the course instructors. The remaining files comprehensively provide the students' marks for the respective module and year.

Data Preparation:

Prior to analysis, data pre-processing was carried out which involved converting score strings into integers using regular expressions. Unknown or missing values were replaced with the mean scores of the respective classes, enabling a more accurate analysis.

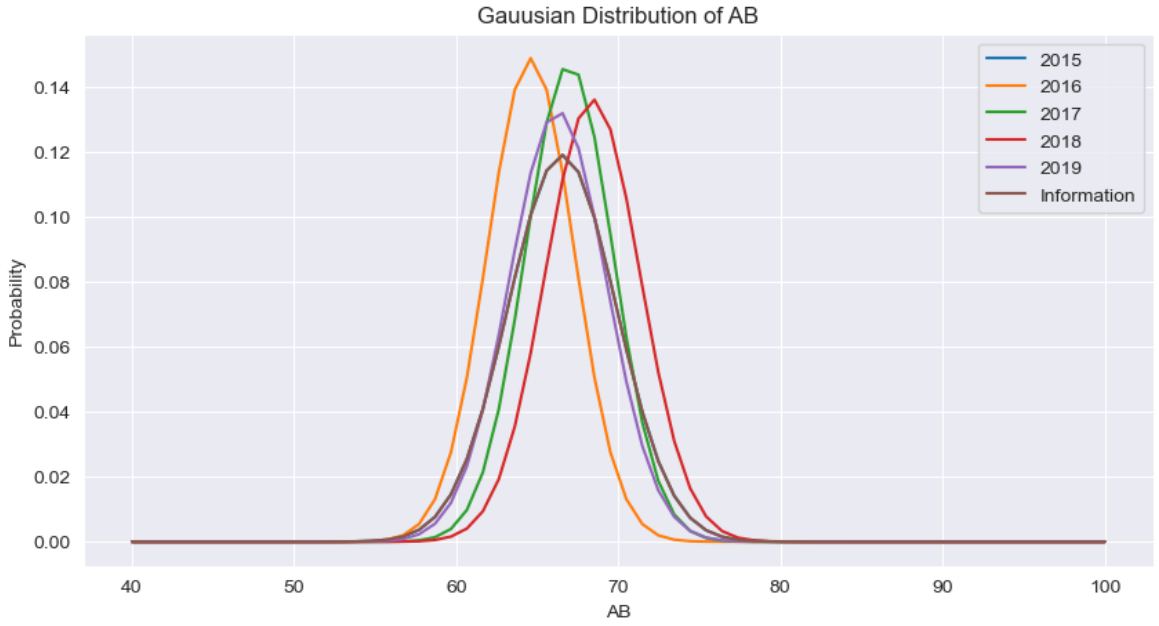
Data Analysis:

The data was analysed using several methods:

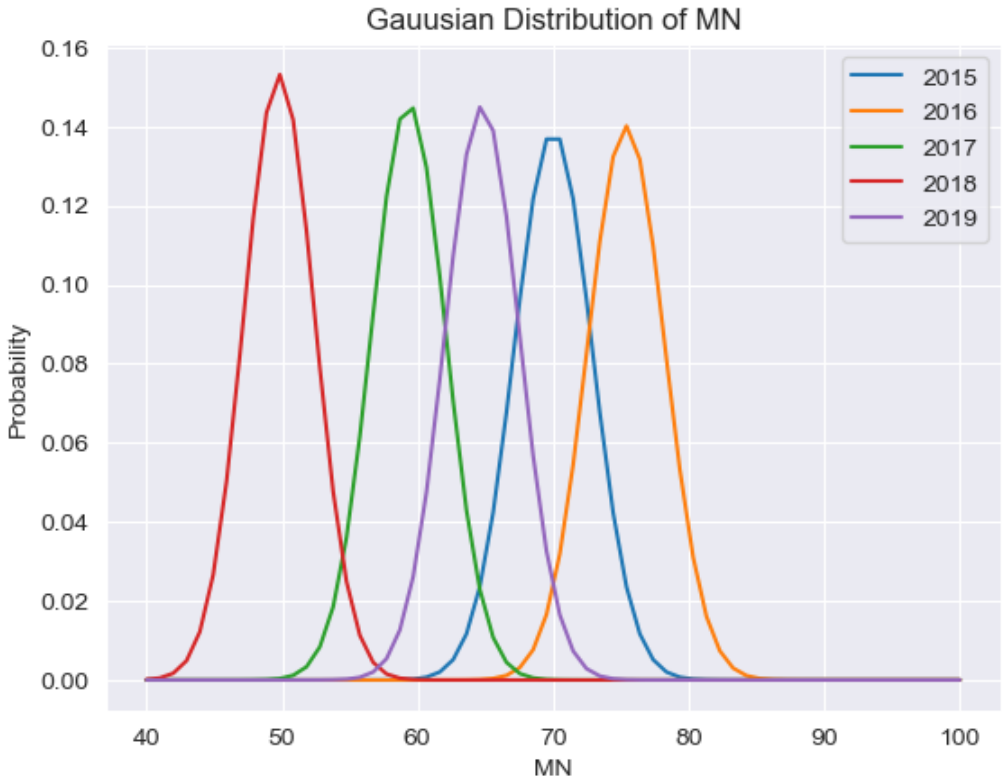
a) Aggregate Analysis: Given that AM41AB and AM41MN had an identical number of students each year, an aggregate analysis of the scores was conducted as a preliminary step. I didn't split the data into a training set and a test set because the data wasn't big enough. If I split it up, the impact would be huge.

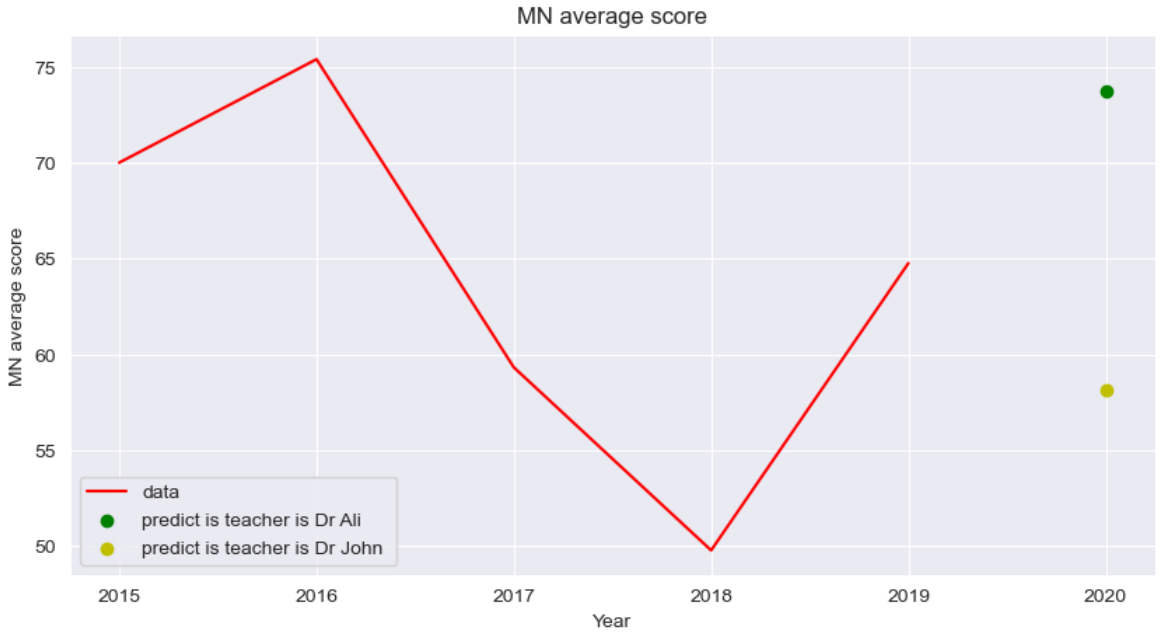
b) Gaussian Distribution: Based on the assumption that students' scores follow a Gaussian distribution, p-tests were conducted, confirming this hypothesis with p-values exceeding 0.05 for both modules, indicating a statistically significant normal distribution of scores.

c) AM41AB Module: Dr. Sam, who taught the AM41AB module consistently over four years, ensured a level of consistency in teaching methodology and marking criteria. The score distribution exhibited a similar pattern year after year. Thus, using these historical trends, the mean score for 2020 was projected by averaging the means of the Gaussian distributions from the previous years. The projected mean score for 2020 is 66.55.



d) AM41MN Module: Unlike the AM41AB module, the AM41MN module demonstrated a more independent scoring pattern, suggesting that it may not have a direct correlation with the former module. Another point of note is the change in instructors: Dr. John taught the course in 2015 and 2016, while Dr. Ali took over from 2017 to 2019. Thus, the impact of the instructor on the scores was considered, and a linear regression model was used to project the scores for 2020, factoring in the potential instructor. If Dr. Ali continues to teach in 2020, the predicted mean score is 58.18. Conversely, if Dr. John were to teach, the projected mean score is 73.70.





Conclusion:

The analysis reveals that student performance in both modules aligns with a Gaussian distribution. The projected mean score for the AM41AB module in 2020 is 66.55. For the AM41MN module, the predicted mean score for 2020 is contingent upon the instructor: 58.18 if taught by Dr. Ali and 73.70 if Dr. John assumes the role.

Question 1) Import the data "train.txt" and identify the total number Users and the total number of Products.

In [1]:
```
import pandas as pd
import numpy as np

def parse_data(file):
    data = pd.read_csv(file, sep=",", header=None, names=["user", "product", "rating"])
    data["user"] = data["user"].str.extract(r"(\d+)",expand=False).to_numpy().astype(int)
```

```
    data["product"] = data["product"].str.extract(r"(\d+)",expand=False).to_numpy().astype(int
    data["rating"] = data["rating"].str.extract(r"([0-9.]+)",expand=False).to_numpy().astype(f
    data.columns = ["user", "product", "rating"]
    return data


data_pd = parse_data("./Part A data/train.txt")
total_users = data_pd["user"].unique().shape[0]
total_product = data_pd["product"].unique().shape[0]
print(data_pd.shape)
print("number Users ", total_users)
print("number of Products ",total_product)
```

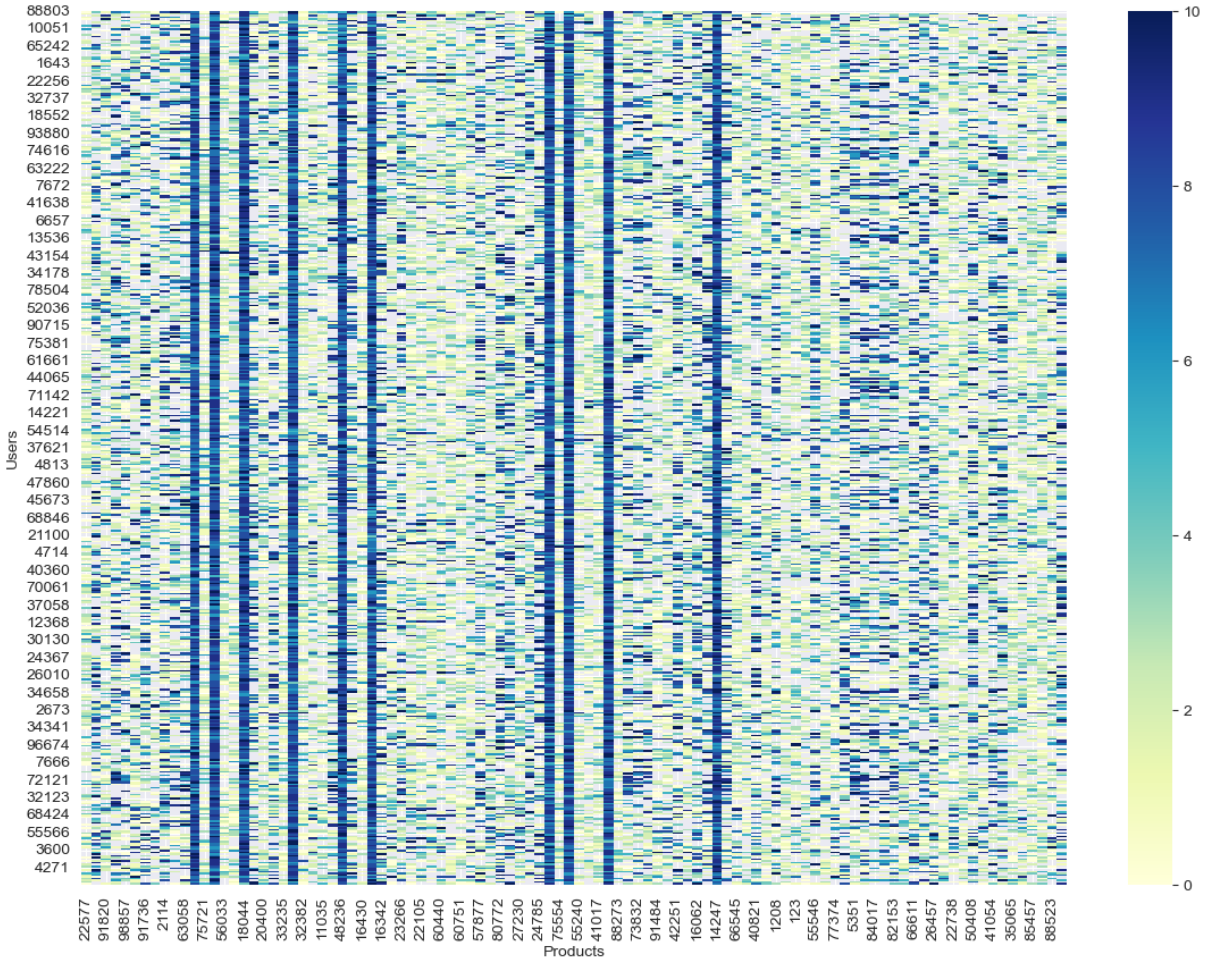
```
(31763, 3)
number Users  500
number of Products  100
```

Question 2) Create a data frame (or matrix, or array) Y which consists of the ratings, such that y_{nd} represents the rating given by User n to Product d.

1. What are the dimensions of Y?
2. Plot the matrix Y

In [33]:
```
import matplotlib.pyplot as plt
import seaborn as sns

Y = np.full((total_users, total_product),np.nan)
Y = pd.DataFrame(Y, columns=data_pd["product"].unique(), index=data_pd["user"].unique())
for index, row in data_pd.iterrows():
    Y.loc[row["user"], row["product"]] = row["rating"]

f, ax = plt.subplots(figsize = (14, 10))
sns.heatmap(Y, cmap="YlGnBu",ax=ax)
ax.set_xlabel("Products")
ax.set_ylabel("Users")
print("Dimensions of Y:", Y.shape)
```

```
Dimensions of Y: (500, 100)
```



Question 3) Find the top 5 Products. Explain your method and results.

> My method is to group the data by product and calculate the mean of the rating. Then sort the products by their average rating and print the top 5 products.

In [34]:
```
product_ratings = data_pd.groupby("product")["rating"].mean()

# Sort the Products by their average rating
top_products = product_ratings.sort_values(ascending=False).head(5)

# Print the top 5 Products
print("Top 5 Products:")
print(top_products)
```

```
Top 5 Products:
product
18044    8.090
14247    8.058
71562    8.036
48236    8.018
7343     7.996
Name: rating, dtype: float64
```

Question 4) Import the data "test.txt" and identify the total number Users and the total number of Products.

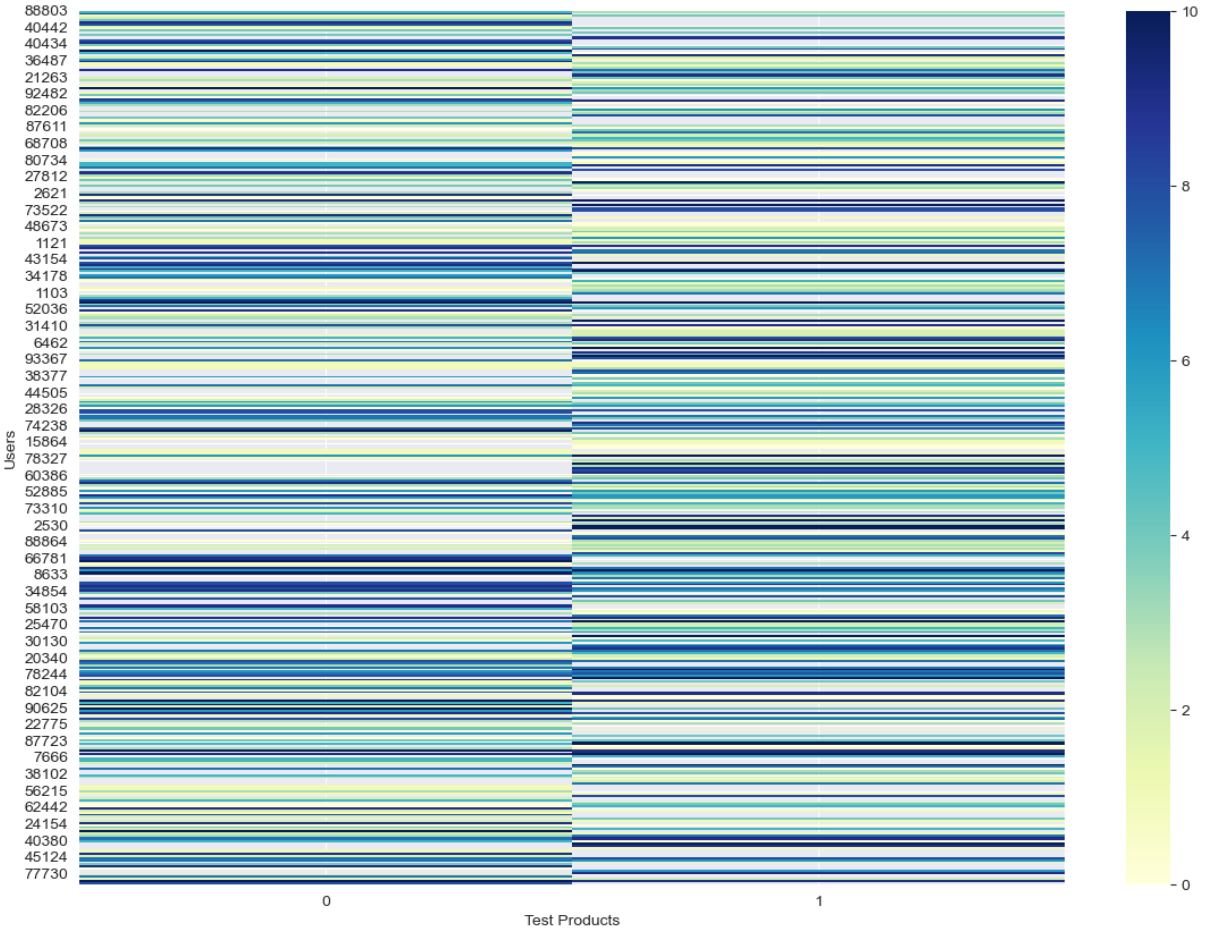
In [35]:
```
test_data_pd = parse_data("./Part A data/test.txt")
total_users_test = test_data_pd["user"].unique().shape[0]
total_product_test = test_data_pd["product"].unique().shape[0]
print(test_data_pd.shape)
print("number Users ", total_users_test)
print("number of Products ",total_product_test)
```

```
(585, 3)
number Users  421
number of Products  2
```

Question 5) Create a data frame (or matrix, or array) X which consists of the ratings, such that X_{nd} represents the rating given by User n to Product d. What are the dimensions of x?

In [36]:
```
X = np.full((total_users_test, total_product_test),np.nan)
X = pd.DataFrame(X, columns=test_data_pd["product"].unique(), index=test_data_pd["user"].unique
for index, row in test_data_pd.iterrows():
    X.loc[row["user"], row["product"]] = row["rating"]

f, ax = plt.subplots(figsize = (14, 10))
sns.heatmap(X, cmap="YlGnBu",ax=ax)
ax.set_xlabel("Test Products")
ax.set_ylabel("Users")
print("Dimensions of X:", X.shape)
```

```
Dimensions of X: (421, 2)
```



Question 6) For each Product in "test.txt", find the most similar product in "train.txt". To compute the distance between Product n (from $\mathbf{X}$) and Product m (from $\mathbf{Y}$) use the following
$For i = 1, \ldots, N_{user}$

If x_{in} and y_{im} exist $d_{nm} = d_{nm} + (x_{in} - y_{im})^2$

Where N_{user} is the total number of users, and d_{nm} represents the distance between Product n (from $\mathbf{X}$) and Product m (from $\mathbf{Y}$).

In [37]:
```
import math
def distance_matrix(X, Y):
    distance_matrix = np.full((X.shape[1], Y.shape[1]),np.nan)
    distance_pd = pd.DataFrame(distance_matrix, columns=Y.columns, index=X.columns)
    # print(distance_pd)
    for x_userid, x_row in X.iterrows():
        for x_product_id,x_rating in x_row.items():
```

```
            if not math.isnan(x_rating):
                for y_product_id,y_rating in Y.loc[x_userid].items():
                    if not math.isnan(y_rating):
                        #print(x_userid,x_product_id,x_rating,y_product_id,y_rating)
                        if np.isnan(distance_pd.loc[x_product_id, y_product_id]):
                            distance_pd.loc[x_product_id, y_product_id] = 0
                        distance_pd.loc[x_product_id, y_product_id] += (x_rating - y_rating) **

    return distance_pd

distance_pd = distance_matrix(X, Y)
print(distance_pd.shape)
# print(distance_pd)
```

```
(2, 100)
```

Question 7) Using the method from Question 6) Find the top 5 similar product in “train.txt”, for each Product in “test.txt”.

In [38]:
```
for x_product_id,x_row in distance_pd.iterrows():
    top_products = x_row.sort_values(ascending=True).head(5)
    print("test product ",x_product_id,"  similar to top 5 train product is ",top_products.inde
```

```
test product  0   similar to top 5 train product is  [50408 58577 26457 60734 38851]
test product  1   similar to top 5 train product is  [24785 26457 40821 50408 41232]
```

Question 8) Explain the main limitation of the method in Question 6)

> The main limitation of the method in Question 6) is that it only considers the rating of the users who rated both the products. It does not consider the rating of the users who rated only one of the products. This can be improved by considering the rating of the users who rated only one of the products by using the average rating of the product as the rating of the user for the product.

Question 9) Propose an alternative to the distance method described in Question 6) by completing: $ For i=1, \ldots, N_{\max } \ If #Answer here \

d{*n m*}=*d*{n m}+\left(x{*i n*}-*y*{i m}\right)^{2}\

If #Answer here \

d{*n m*}=*d*{n m}+# A n s w e r \text { here } \

$

> my asnswer is If x_{in} and y_{im} exist $d_{nm} = d_{nm} + (x_{in} - y_{im})^2$ else if x_{in} exist $d_{nm} = d_{nm} + (x_{in} - \bar{y}_m)^2$ elif If y_{im} exist $d_{nm} = d_{nm} + (\bar{x}_n - y_{im})^2$

In [39]:
```
def distance_matrix_avg(X, Y):
    distance_matrix = np.full((X.shape[1], Y.shape[1]),np.nan)
    distance_pd = pd.DataFrame(distance_matrix, columns=Y.columns, index=X.columns)
    # print(distance_pd)
    for x_userid, x_row in X.iterrows():
        for x_product_id,x_rating in x_row.items():
            for y_product_id,y_rating in Y.loc[x_userid].items():
                #print(x_userid,x_product_id,x_rating,y_product_id,y_rating)
                if np.isnan(distance_pd.loc[x_product_id, y_product_id]):
                    distance_pd.loc[x_product_id, y_product_id] = 0
                if not math.isnan(y_rating)  and not math.isnan(x_rating):
                    distance_pd.loc[x_product_id, y_product_id] += (x_rating - y_rating) ** 2
                elif  math.isnan(y_rating) and not math.isnan(x_rating) :
```

```
                    #print("x_rating",x_rating,"y_rating",Y[y_product_id].dropna().mean())
                    distance_pd.loc[x_product_id, y_product_id] += (x_rating - Y[y_product_id].
                elif not math.isnan(y_rating)  and math.isnan(x_rating) :
                    # print("x_rating",x_rating,"y_rating",Y[y_product_id].dropna().mean(),y_rat
                    distance_pd.loc[x_product_id, y_product_id] += (X[x_product_id].dropna().me
    return distance_pd

distance_avg_pd = distance_matrix_avg(X, Y)
print(distance_pd.shape)
# print(distance_pd)
```

```
(2, 100)
```

Question 10) Using your solution to Question 10), find the top 5 similar products in "train.txt", for each Product in "test.txt". Marking scheme

In [47]:
```
for x_product_id,x_row in distance_avg_pd.iterrows():
    top_products = x_row.sort_values(ascending=True).head(5)
    print("test product ",x_product_id,"  similar to top 5 train product is ",top_products.index
    # print(top_products)
```

```
test product  0   similar to top 5 train product is  [91736 80772 42251 16062 41232]
test product  1   similar to top 5 train product is  [66611 26457 16062 42251 41232]
```

Part B: University Module Data

You've been tasked to analyse the marks of students for two modules over a five-year period. The data is stored in the following "Part B data". You are instructed to create a short 3-page report which gives an overview of the performance of students over the two modules and predicts the average mark of each module for 2020 (assuming COVID did not take place). 你的任务是分析五年内两个模块的学生的分数。这些数据存储在以下 "B部分数据 "中。你被要求创建一份3页的简短报告，概述学生在这两个模块中的表现，并预测2020年每个模块的平均分数（假设没有发生COVID）。

In [41]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
import os
warnings.filterwarnings('ignore')
import re
```

In [42]:
```
df_ab = None
df_mn = None
current_df = None
df_infor = None
all_csv_list = os.listdir("./Part B data")
print(all_csv_list)

def parse_data(pd_input,file_name):
    # print(file_name)
    try:
        year = file_name.replace(".txt","").split("_")[1]
    except:
        year = "Information"
    res_data = []
    for index, row in pd_input.iterrows():
        if "UNKNOWN" in row.values[0]:
            res_data.append(np.nan)
        else:
            res_data.append(float(re.findall(r"([0-9.]+)",row.values[0])[0]))
    pd_out = pd.DataFrame(res_data,columns=[year])
    pd_out = pd_out.fillna(pd_out.mean())
    return pd_out


for count, single_csv in enumerate(all_csv_list):
```

```
    if "AM41" in single_csv:
        current_df = pd.read_csv("./Part B data/%s"% single_csv,encoding='utf-8',sep="\t")
        res = parse_data(current_df,single_csv)
        if "AB" in single_csv:
            if df_ab is None:
                df_ab = res
            else:
                df_ab = pd.concat([df_ab,res],axis=1)
        elif "MN" in single_csv:
            if df_mn is None:
                df_mn = res
            else:
                df_mn = pd.concat([df_mn,res],axis=1)
    elif "Information.txt" in single_csv:
        current_df = pd.read_csv("./Part B data/%s"% single_csv,encoding='utf-8',sep="\t")
        res = parse_data(current_df,single_csv)
        df_infor = res
```

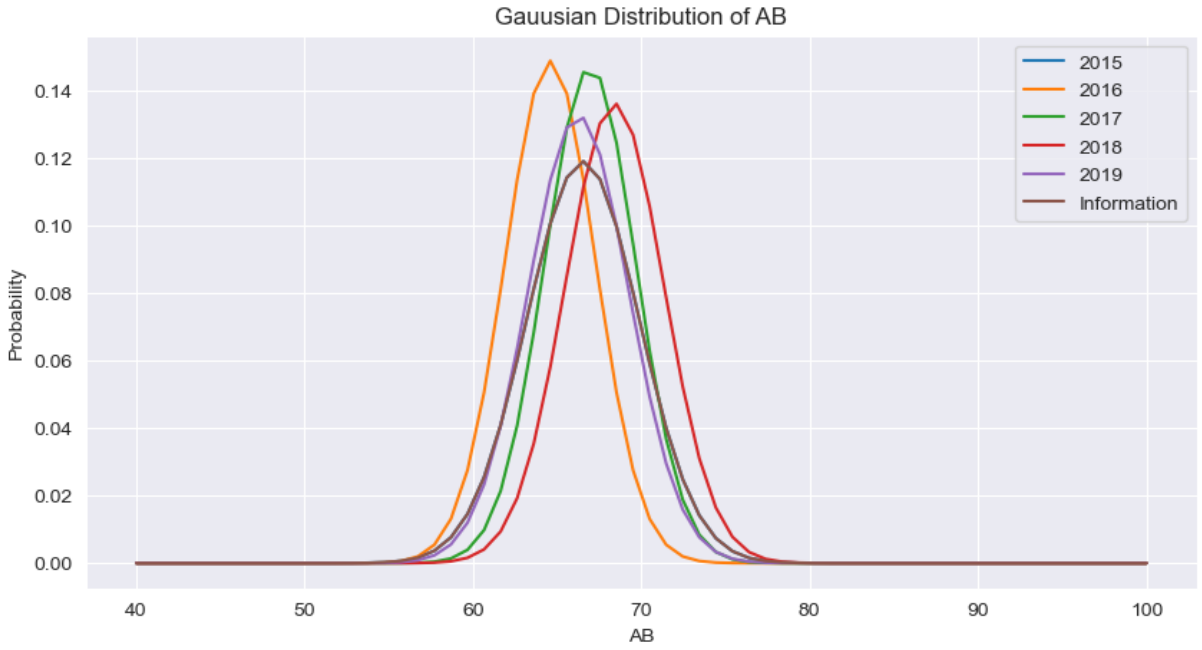
```
['AM41AB_2015.txt', 'AM41AB_2016.txt', 'AM41AB_2017.txt', 'AM41AB_2018.txt', 'AM41AB_2019.txt',
'AM41MN_2015.txt', 'AM41MN_2016.txt', 'AM41MN_2017.txt', 'AM41MN_2018.txt', 'AM41MN_2019.txt',
'Information.txt', 'ModuleInformation.xlsx']
```

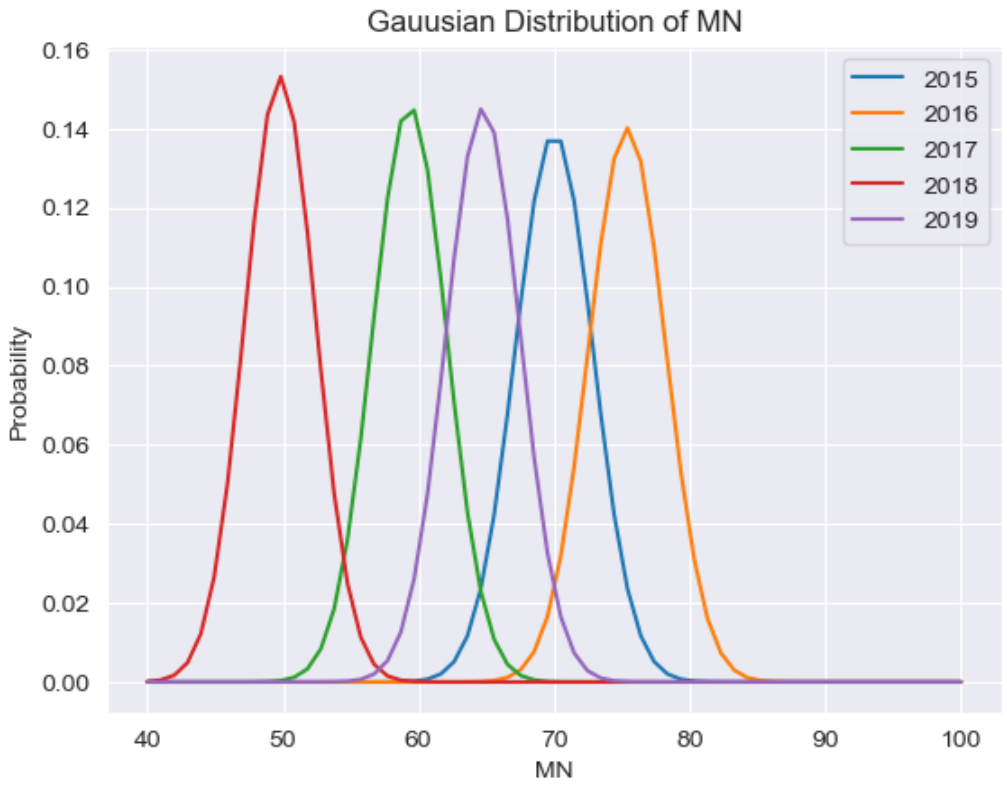
In [43]:

```
from scipy.stats import norm
import numpy as np
# function of maximum likelihood
from scipy import stats as st

#get_r0 = lambda x: np.sum(x, axis=0) / len(x)
# get_sigma = lambda x, mu: (x - mu).T.dot(x - mu) / (len(x) )
data_ab_set_list = []
data_mn_set_list = []
data_ab_r0_list = []
data_mn_r0_list = []
data_ab_sigma_list = []
data_mn_sigma_list = []
years = ["2015","2016","2017","2018","2019"]
x = np.linspace(40, 100, 62)
for y in years:
    r0, sigma = norm.fit(df_ab[y].dropna())

    res = st.shapiro(df_ab[y].dropna())
    print("AB",y,"r0",r0,"sigma",sigma,res)
    data_ab_r0_list.append(r0)
    data_ab_sigma_list.append(sigma)
    data_ab_set_list.append(st.norm.pdf(x, r0, np.sqrt(sigma)))
    r0, sigma = norm.fit(df_mn[y].dropna())
    res = st.shapiro(df_ab[y].dropna())
    print("MN",y,"r0",r0,"sigma",sigma,res)
    data_mn_r0_list.append(r0)
    data_mn_sigma_list.append(sigma)
    data_mn_set_list.append(st.norm.pdf(x, r0, np.sqrt(sigma)))

g_data_pd = pd.DataFrame({"ab_r0":data_ab_r0_list,"ab_sigma":data_ab_sigma_list,"mn_r0":data_mn
r0, sigma =norm.fit(df_infor["Information"].dropna())
print("Information ","r0",r0,"sigma",sigma)
data_infor_norm = st.norm.pdf(x, r0, np.sqrt(sigma))

plt.figure(figsize=(10,5))
for n,i in enumerate(data_ab_set_list):
    plt.plot(x, i,label=years[n])
plt.plot(x,data_infor_norm,label="Information")
plt.xlabel("AB")
plt.ylabel("Probability")
plt.title("Gauusian Distribution of AB")
plt.legend()
plt.show()


for n,i in enumerate(data_mn_set_list):
    plt.plot(x, i,label=years[n])
```

```
plt.xlabel("MN")
plt.ylabel("Probability")
plt.title("Gauusian Distribution of MN")
plt.legend()
plt.show()
```

```
AB 2015 r0 66.53333333333333 sigma 11.2055750535482 ShapiroResult(statistic=0.9432141184806824, pvalue=0.3010416626930237)
MN 2015 r0 70.0 sigma 8.252591299579468 ShapiroResult(statistic=0.9432141184806824, pvalue=0.3010416626930237)
AB 2016 r0 64.58974358974359 sigma 7.169119220397328 ShapiroResult(statistic=0.9336419105529785, pvalue=0.014008289203047752)
MN 2016 r0 75.3888888888889 sigma 8.096794477327482 ShapiroResult(statistic=0.9336419105529785, pvalue=0.014008289203047752)
AB 2017 r0 66.9622641509434 sigma 7.341689322173039 ShapiroResult(statistic=0.9716718792915344, pvalue=0.16854214668273926)
MN 2017 r0 59.327586206896555 sigma 7.481973814780764 ShapiroResult(statistic=0.9716718792915344, pvalue=0.16854214668273926)
AB 2018 r0 68.40983606557377 sigma 8.564099889753493 ShapiroResult(statistic=0.9806907773017883, pvalue=0.33503642678260803)
MN 2018 r0 49.791044776119406 sigma 6.779334024635367 ShapiroResult(statistic=0.9806907773017883, pvalue=0.33503642678260803)
AB 2019 r0 66.26388888888889 sigma 9.041561059291084 ShapiroResult(statistic=0.9846985340118408, pvalue=0.45719072222709656)
MN 2019 r0 64.7605633802817 sigma 7.54563580610076 ShapiroResult(statistic=0.9846985340118408, pvalue=0.45719072222709656)
Information  r0 66.53333333333333 sigma 11.2055750535482
```





From the grpah and value we can see that Information is the same as 2019,It maybe is a mistake. And consider the teacher's informathion: Year Module Tutor 2015 AM41AB Dr Sam 2016 AM41AB Dr Sam 2017 AM41AB Dr Sam 2018 AM41AB Dr Sam 2015 AM41MN Dr Ali 2016 AM41MN Dr Ali 2017 AM41MN Dr John 2018 AM41MN Dr John 2019 AM41MN Dr John

Because marks is an independent data, I do the K-S test for the data. Whether the data is Gaussian distributed or not. and result shows that all the P value is bigger than 0.05, so we can assume the data is Gaussian distributed. AM41AB have the same tutor in 2015-2018, and the distribution are very similar. Because of lacking of 2019's teachers data. I have to assume 2020 and 2019 don't change teacher.So that we can predict the 2020 data using average.

In [44]:
```
ab_2020 = g_data_pd["ab_r0"].mean()
print("AB 2020",ab_2020)
```

```
AB 2020 66.55181320569659
```

and AM41MN have different tutor in 2015-2018, and the distribution are very different. So that I want to use two demotion Gaussian distribution to fit the data by using teacher and year.

In [45]:
```
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LinearRegression

regressor = LinearRegression()
# merge the data to one dataframe,and add the teacher information,use linear regression to predi
data_mn = pd.concat([df_mn["2015"].dropna(),df_mn["2016"].dropna(),df_mn["2017"].dropna(),df_mn
data_ab = pd.concat([df_ab["2015"].dropna(),df_ab["2016"].dropna(),df_ab["2017"].dropna(),df_ab
years_list = [[2015]*len(df_mn["2015"].dropna())+[2016]*len(df_mn["2016"].dropna())+[2017]*len
years_list = np.array(years_list).reshape(-1)
data_mn = pd.DataFrame({"value_mn":data_mn,"value_ab":data_ab,"year":years_list})

# add teacher information
data_mn["teacher"] = ["Dr Ali"]*len(df_mn["2015"].dropna())+["Dr Ali"]*len(df_mn["2016"].dropna

data_mn['teacher'] = data_mn['teacher'].replace({'Dr Ali': 1, 'Dr John': 2})

liner = LinearRegression()
model = liner.fit(data_mn[["teacher","value_ab"]],data_mn["value_mn"])
```

```
missing=model.predict([[1,66.55],[2,66.55]])
print("MN 2020",missing)
```

```
MN 2020 [73.69668908 58.17568285]
```

In [46]:
```
plt.figure(figsize=(10,5))
plt.plot([2015,2016,2017,2018,2019],data_mn_r0_list,label="data",c="r")
plt.scatter([2020],missing[0],label="predict is teacher is Dr Ali ",c="g")
plt.scatter([2020],missing[1],label="predict is teacher is Dr John ",c="y")
plt.xlabel("Year")
plt.ylabel("MN average score")
plt.title("MN average score")
plt.legend()
plt.show()
```