**ABSTRACTION**

There is 2 decisions in this thesis such as assign the right task to the right person and suggesting the right task to the new employee. Before working, attributes have been specified and written to Excel table. It’s discussed with domain expert and non domain expert people. Tasks have been added into the file as sheets. Per sheets represent a task for employees. Earlier, they always have been working at the same task. Thus, in order to find a right task to employees, they will be worked at each new task. Under favour of this result, this program will suggest correct task to new employees as for that their attributes. Decision tree regressor and naive bayes algorithms have been used as suggestion algorithm to find correct suggestion. It doesn’t show us optimal, however it shows which group of people do better on attribute basis. Hence, we decide based on presend datas to assign correct task to new employees.

Keywords: K-means, Python, Task Assignment, Naive Bayes, Regression

1. **INTRODUCTION**

Nowadays, while most of companies is deciding to employ new joiner, they go around the different criterias. Ages, experiences, travel restriction are exampled for these. Of course, after recruitment, employer need to give the right decision to assign task. Because although employer have senior and successfull worker, they have difficulty in adaption in company. I’ll emphasize that recruitment is not just about success and experience.

BÖLÜM BÖLÜM NERDE NE YAPTIĞIM ANLATILACAK. YAPILDI, EDİLDİ ŞEKLİNDE ANLATTIM DAHA ÇOK.

1. **ATTRIBUTES of WORKERS**

Attributes are specified in order to show their performance in tasks. Attributes are age, experience, genre, marriage, childbearing, graduation, former job quantity, seniority, title, estimated time of arrival such as independent. Actually, I wanted to show whether these attributes affect their success or not. Each workers will work in each tasks. We’ll calculate their success according to make fixed bug numbers. They will have success point in every task. Hence, information of success will be compared according to each attributes. For instance, older men is more successfull than older women.

1. **BUG TRACKER SYSTEM**

Software group designs and implements all requirements of system. R&D test group’s task is to find bugs of the system. Their bugs are resolved by software group and software is released again. Until R&D group don’t see, this loop goes on. Then the software that there is no issue delivers quality department. Quality assurance tests the same parts. When they see any issues, R&D software group evaluate those bugs in their side. If it’s major or can be reproducable, it will be resolved by software group. After test is finished, software group releases new version. R&D test group re-test again, then quality assurance group re-test again. I’ll approach R&D software group’s success.

1. **SUCCESS CRITERIA**

Workers will collect their points entering bugs, task complete speed and quality assurance response. If their bugs are made fixed by sw group, it will affect positively. Bug priority is depand on bug’s criteria. Showstopper bugs’s coefficient is 20, high bugs’s is 10, medium one is 5 and low one is 1. Else if these bugs are set to re-test in Jira system, it will affect evenly, negatively. Workers’s score will be decrease with the same coefficient. Complete speed will play a role in task due date inversely proportional. Let’s say that due date is 4 hours. Complete speed is (¼)\*20 . 20 is a number which I specify to show score as integer number.

Quality assurance department’s task is to catch escaper issues of R&D test group. If tester can’t catch bugs with their test suites, it affects their score negatively. If those bugs have high priority, it gives more negative point. The same priority rule progresses during this process.

1. **SUGGESTION MECHANISM**

Scores that have been collected is classified as successfully and failed. Each test is represented as sheets in different file. For once, independent variable is our decision. If testers succeed in their task, the related file will be updated as “yes”. In order to be successfull, they should get min 50 points in each task. Starting from this, the system will recommend us with different ways to assign new employee.

I used different regression and classification algorithms to find a good recommendation. Decision tree regressor, simple vector regressor and naive bayes algorithms have been used. After user enters inputs in the program, the application will say that whether new employee can do this task or not. I will work through in the following.

In these algorithms, the same independent variables is used with different file. However, some string variables are defined as string. For example, sex variable can be male or female. It doesn’t represent integer in excel file. Thus, it converted to integer in the background. We had already calculated their scores in before. So, it was updated as yes or no in dependent variable. If the score is greater than 50, our dependent variable that is accept will be “yes”.

1. **NAIVE BAYES ALGORITHM**
2. **Manual Calculation:**

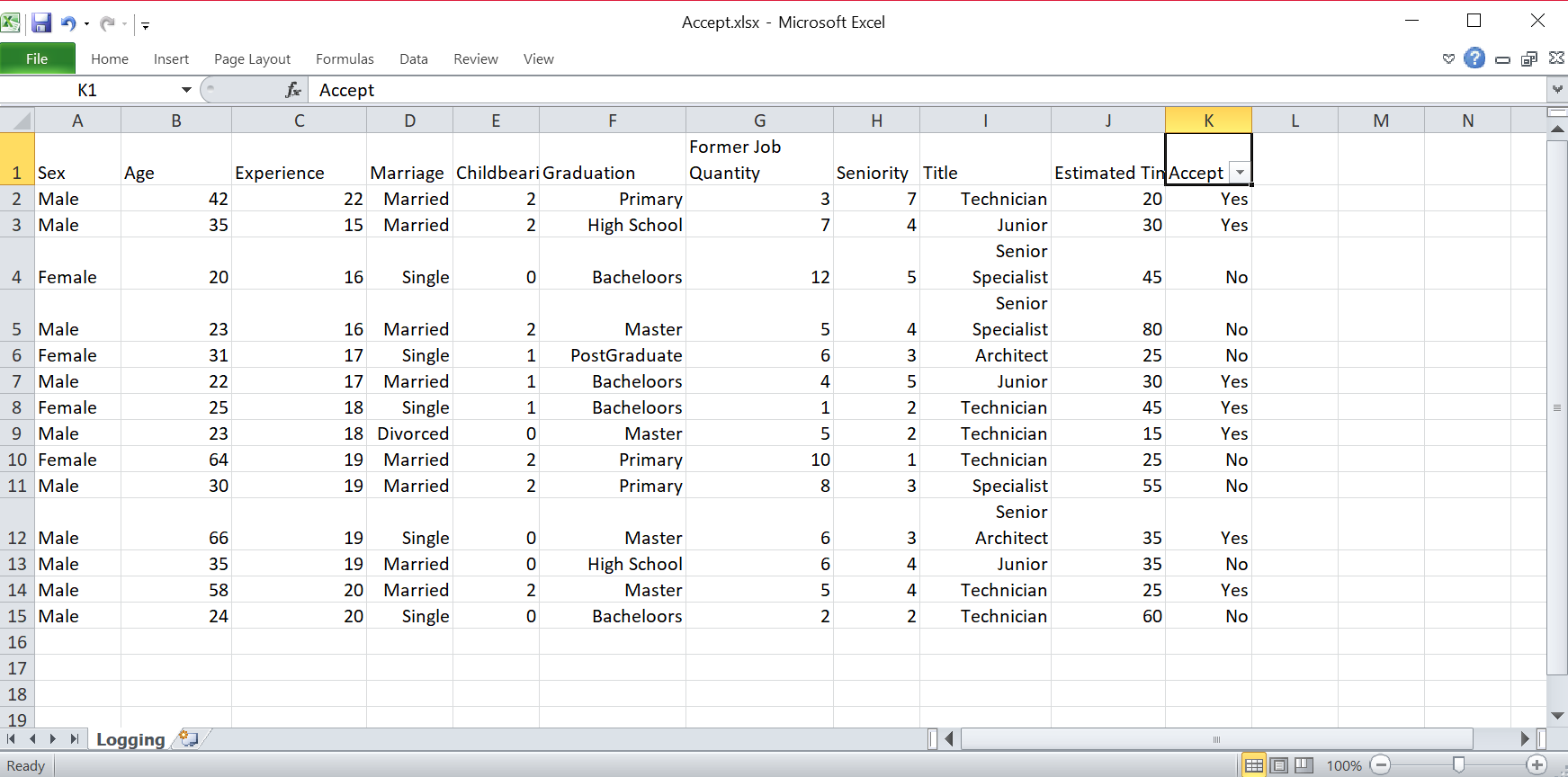
Suppose that our features is in the below.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Sex | Age | Experience | Maritial Status | Childbearing | Graduation | Former Jobs | Seniority | Title | Estimated Time of Arrival |
| Male | 20 | 17 | Single | 2 | High School | 1 | 2 | Technician | 45 |

We need to find probabilities in case of “Yes” and “No”. According to Naive Bayes Algorithm, the probability is in the following for each dependent result such as yes and no.

For “yes” dependent value result = P(Yes) \* P(M|Yes) \* P(20|Yes) \* P(17|Yes) \* P(“Single”|Yes) \* P(2|Yes) \* P(“High School”|Yes) \* P(1|Yes) \* P(2|Yes) \* P(“Technician”|Yes) \* P(45|Yes)

For “no” dependent value result = P(No) \* P(M|No) \* P(20|No) \* P(17|No) \* P(“Single”|No) \* P(2|No) \* P(“High School”|No) \* P(1|No) \* P(2|No) \* P(“Technician”|No) \* P(45|No)



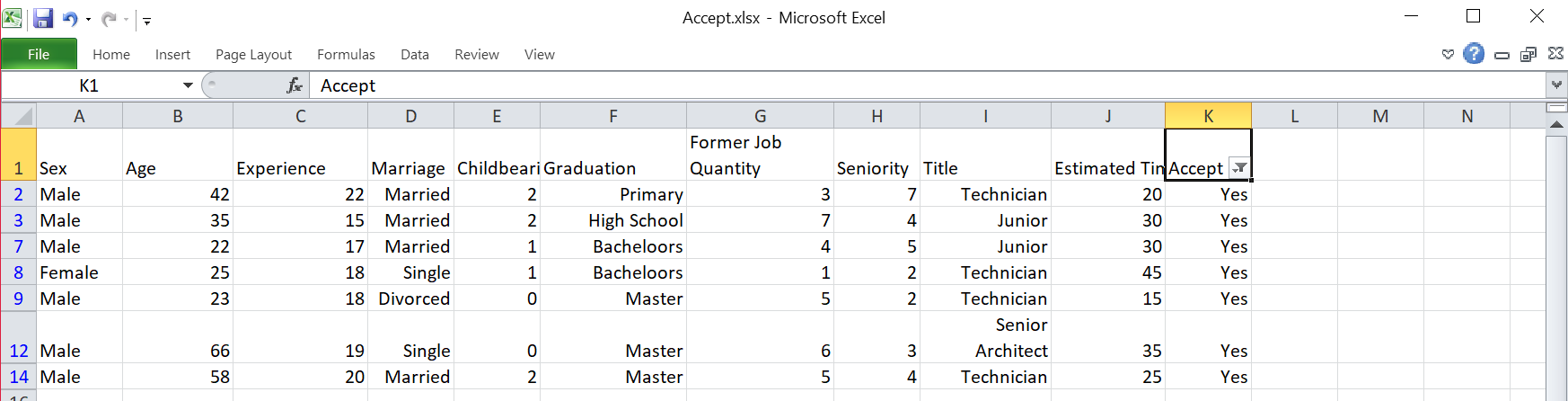
For “Yes” selection:

The probability of “yes” for all testers is 7/14 = ½ 🡪 P(Yes)

We will calculate probabilities for each independent result in case of “yes”. It will be used in the following formula:

P(Male|Yes) = [ (Number of “Male” for all “Yes” ) + 1 ] / [ Number of all values for “Yes” + Number of all different values for “Yes” ]

We filtered our file for “Yes” selection.



Number of “Male” for all “Yes” is 6.

Number of all values for “Yes” is 7.

Number of all different values for “Yes” is 2.

P(Male|Yes) = (6+1) / (7+2) = 7/9

Other probabilies are found in the following:

P(20|Yes) = (0+1) / (7+7) = 1/14, P(17|Yes) = (1+1) / (7+6) = 2/13,

P(“Single”|Yes) = (2+1) / (7+3) = 3/10, P(2|Yes) = (3+1) / (7+3) = 4/10,

P(“High School”|Yes) = (1+1) / (7+4) = 2/11, P(1|Yes) = (1+1) / (7+6) = 2/13,

P(2|Yes) = (2+1) / (7+5) = 3/12, P(“Technician”|Yes) = (4+1) / (7+3) = 5/10,

P(45|Yes) = (1+1) / (7+6) = 2/13

For “yes” dependent value result = (7/14)\*(7/9)\*(1/14)\*(2/13)\*(3/10)\*(4/10)\*(2/11)\*(2/13)\*(3/12)\*(5/10)\*(2/13) = 1/362505

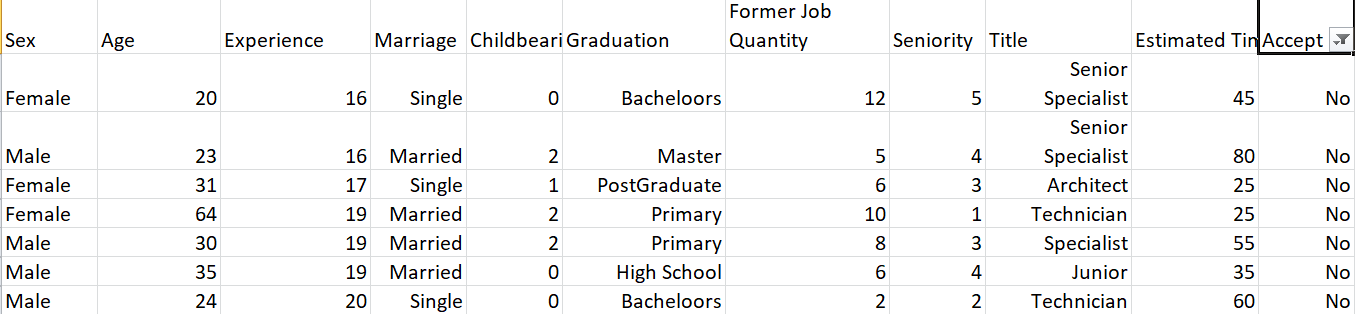
For “No” selection:

The probability of “no” for all testers is 7/14 = ½ 🡪 P(Yes)

We will calculate probabilities for each independent result in case of “no”. It will be used in the following formula:

P(Male|No) = [ (Number of “Male” for all “No” ) + 1 ] / [ Number of all values for “No” + Number of all different values for “No” ]

Now we filtered for “No” selection.



Number of “Male” for all “No” is 4.

Number of all values for “No” is 7.

Number of all different values for “No” is 2.

P(Male|No) = (4+1) / (7+2) = 5/9

Other probabilies are found in the following:

P(20|No) = (1+1) / (7+7) = 2/14, P(17|No) = (1+1) / (7+4) = 2/11,

P(“Single”|No) = (3+1) / (7+2) = 4/9, P(2|No) = (3+1) / (7+3) = 4/10,

P(“High School”|No) = (1+1) / (7+5) = 2/12, P(1|No) = (0+1) / (7+6) = 1/13,

P(2|No) = (1+1) / (7+5) = 2/12, P(“Technician”|No) = (2+1) / (7+5) = 3/12,

P(45|No) = (1+1) / (7+6) = 2/13

For “no” dependent value result = (7/14)\*(5/9)\*(2/14)\*(2/11)\*(4/9)\*(4/10)\*(2/12)\*(1/13)\*(2/12)\*(3/12)\*(2/13) = 1/1054053

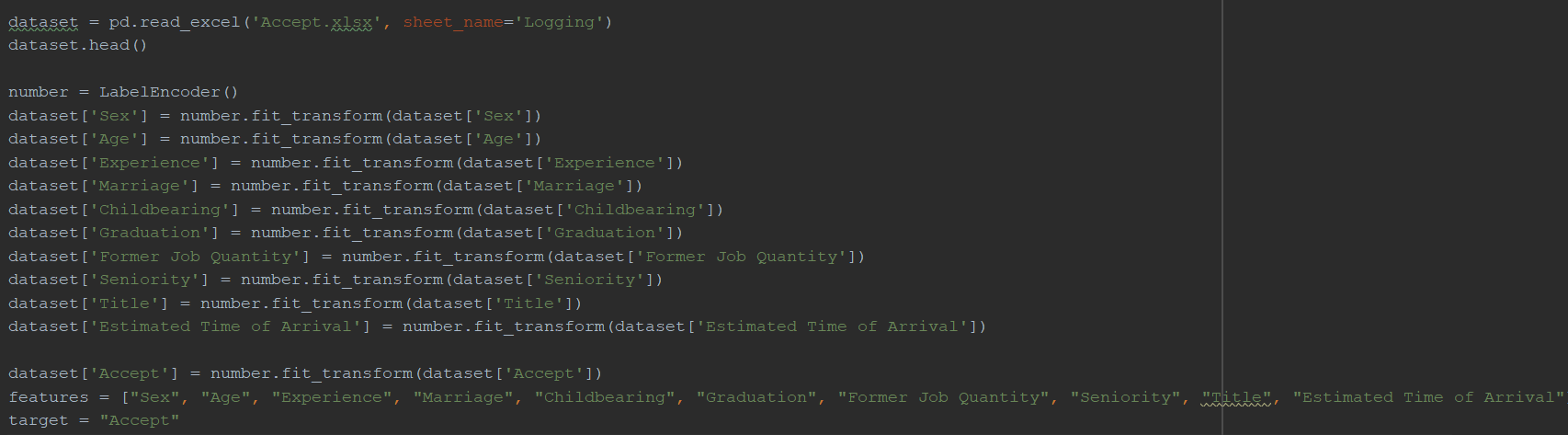
When we compare “yes” and “no” results, we see the probability of “yes” is greater than “no”.

1/362505(yes) < 1/1054053(no)

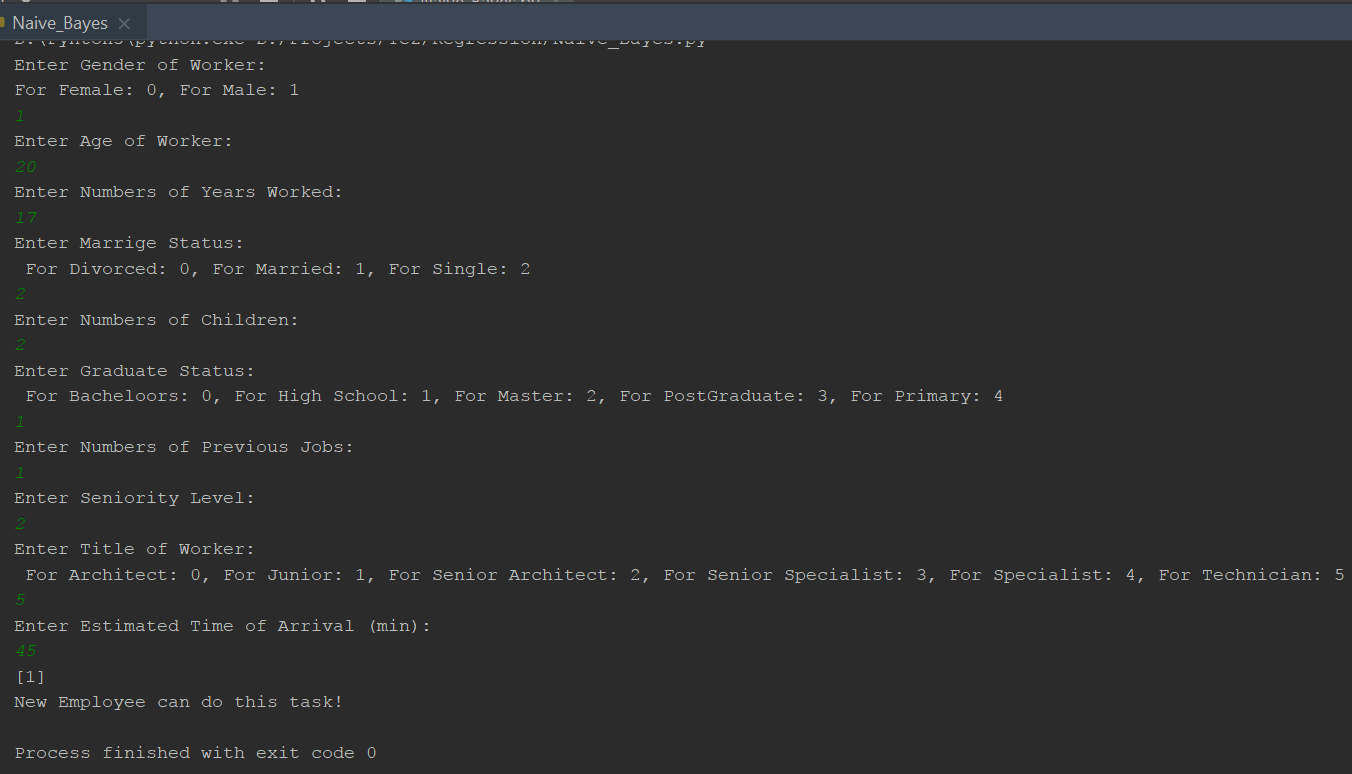
The result is “yes” !

1. **Automatic Calculation:**

We have a data set that are integer values and string values. In order to calculate in programming level, we need to convert string values to integer values. Hence, we used LabelEncoder library in Python language. It arranges in an alphabetical order. For instance, when we check graduation feature in our file, we have 6 different string values. When we use LabelEncoder library, it assigns first element that is “Architect” to 0 and the last element that is “Technician” to 5. Thus, we don’t need to use string values during calculation. Secondly, we separated dependent and independent values in our algorithm. Independent value is defined as target value.



To find recommendation with Naive Bayes, we splitted datas as test and train. Test size is specified as 0.33. Other size is for train data. An instance is created in Naive Bayes class. Dependent variables have been trained to make decision. In the next step, inputs are filled according to new employee’s features.



After this calculation, the program says either new employee can do this task or not according to “yes” and “no” result.

1. **Features of Algorithm:**

* It’s stable.
* The more we have datas the more results are correct.
* It assumes that features follow a normal distribution in classification.
* It can be given good result for text analytics problem.

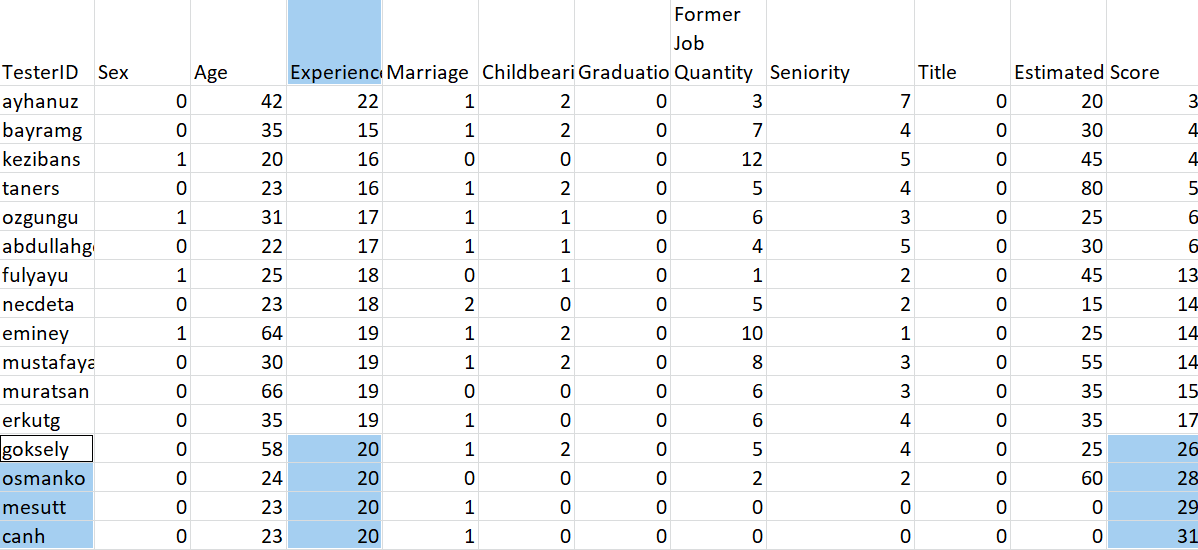
1. **DECISION TREE REGRESSION**

Decision tree regression algorithm returns a result according to arithmetic mean of the related feature. If the value of feature is not in this range,

* If the value of feature that we enter in system is close to greater, it returns us the arithmetic mean of score of greater ones.
* If the value of feature that we enter in system is close to less, it returns us the arithmetic mean of score of less ones.
* If the value of feature that we enter in system is in the middle, it returns us the arithmetic mean of score of less ones.

1. **Manual Calculation**
2. **Calculation with One Feature**

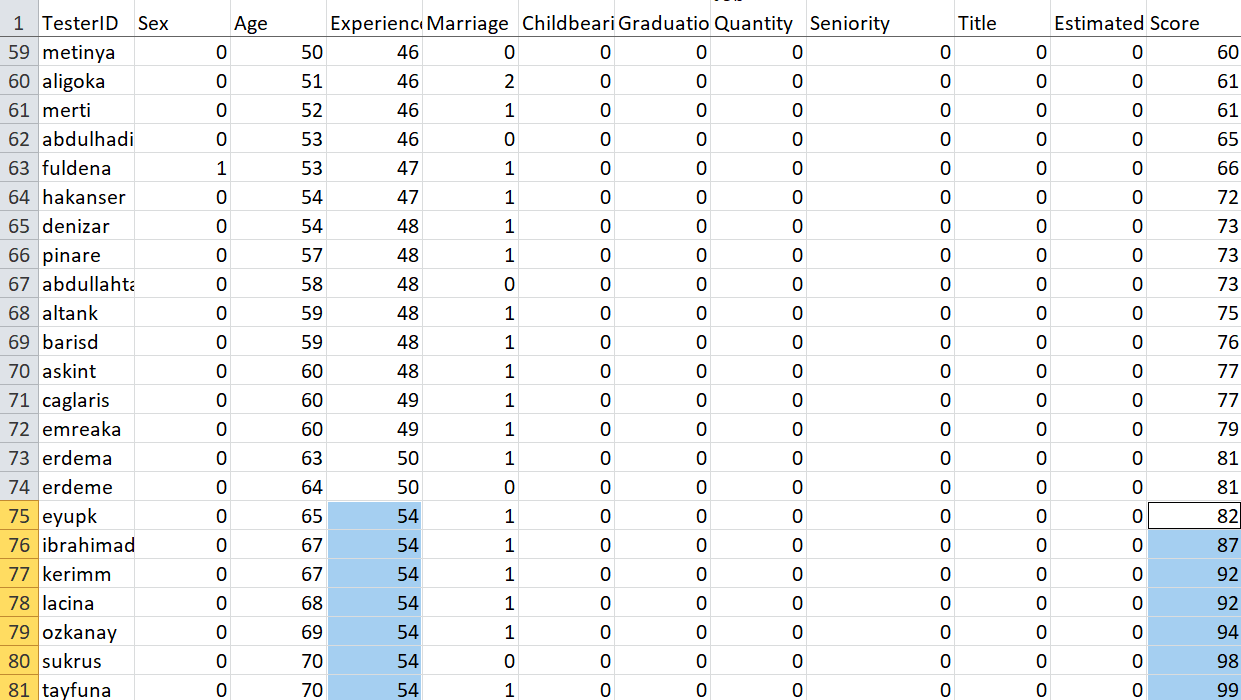
We will use Attributes.xlsx file for one feature.



When we enter 20 to system for experience feature, decision tree regressor algorithm returns the arithmetic mean of score of these people.

Result is (26+28+29+31) / 4 = 28.5.

When we enter 53 to system for experience feature, decision tree regressor algorithm can’t find 53 in the related feature. Thus, it returns the arithmetic mean of score of closer people.



The result is (82+87+92+92+94+98+99) / 7 = 92.

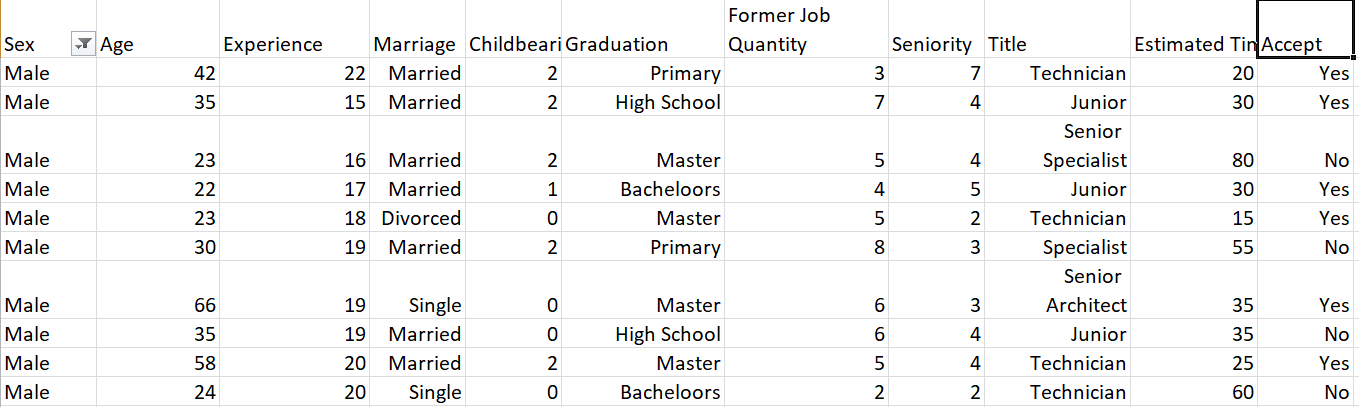
Let’s suppose our experience value is 52. It has the same distance to 50 and 54. Decision tree regressor algorithm returns the mean of less values of experience feature. So, (81+81) / 2 = 81.

1. **Calculation with More Feature**

We will use Accept.xlsx for more feature.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Sex | Age | Experience | Maritial Status | Childbearing | Graduation | Former Jobs | Seniority | Title | Estimated Time of Arrival |
| Male | 35 | 20 | Divorced | 1 | PostGraduate | 7 | 1 | Junior | 15 |

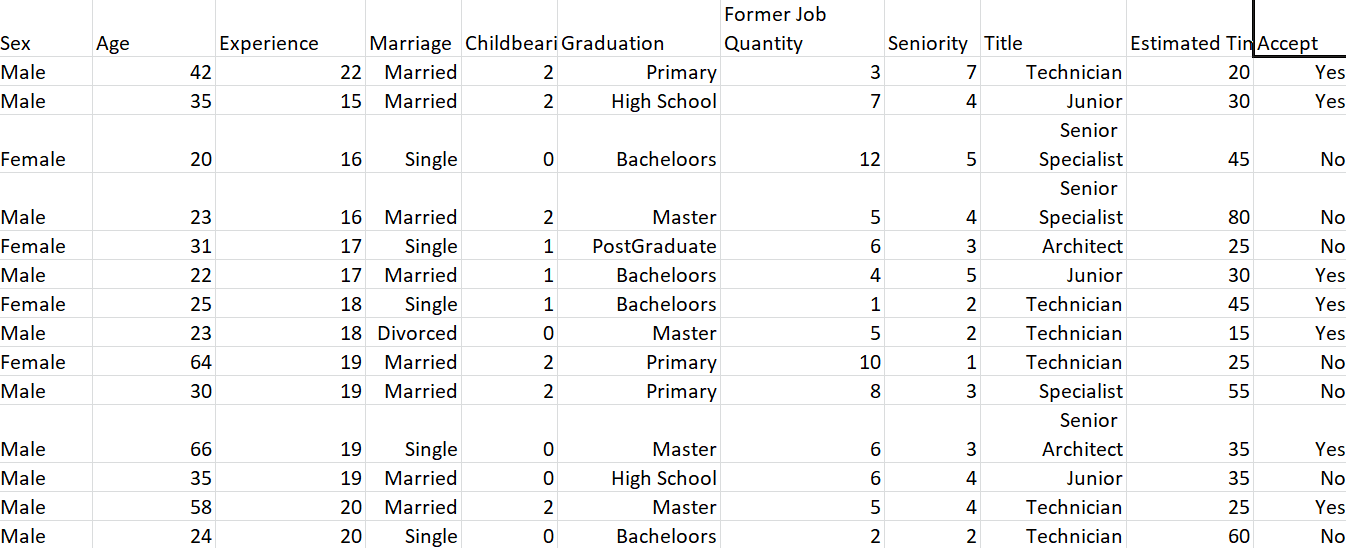
For male selection, sex feature is filtered as only “male”. Results are calculated according to dependent values.



It’s already assigned as 1 and 0 with LabelEncoder method. 1 is used for “Yes” and 0 is used for “No”.

1+1+0+1+1+0+1+0+1+0 = 6 . According to DecisionTreeRegressor rules, arithmetic mean will be calculated. 6/10 = 0.6 , so we can say “Yes”. Because 0.6 is closer to 1 than 0.

Another values are calculated with the same method.



35 has either “no” and “yes” result. Mean is (1+0) / 2 = 0.5. 20 has also either “yes” and “no”. We need to calculate the arithmetic mean. 🡪 (1+0) / 2 = 0.5 . In case of “Divorced”, we can say “Yes” clearly. For 1 child, we need to calculate the arithmetic mean. 🡪 (0+1+1) / 3 = 2/3 , we can say “yes”. There is only one “PostGradeuate” value, so we can find “yes”. For 7 in former job quantity feature, we have one result. It’s “yes”. Seniority feature has the same as former job. There is only one result that is “no”. In case of “Junior” selection, we need to calculate arithmetic mean. 🡪 (1+1+0) / 3 = 2/3. Finally, when we enter 15 for estimated time, we find “yes”. So, number of “yes” is greater than number of “no”. Our result is “yes”.

When we calculate the arithmetic mean of all result, we will find whether new employee can do this task or not.

Male 🡪 0.6; Age 🡪 0.5; Experience 🡪 0.5; Maritial Status 🡪 1;

Numbers of Children 🡪 0.66; Graduation 🡪 0; Numbers of Previous Jobs 🡪 1;

Seniority Level 🡪 0; Title of Worker 🡪 0.66 ; Estimated Time of Arrival 🡪 1

Sum = 0.6+0.5+0.5+1+0.66+0+1+0+0.66+1 = 5.92

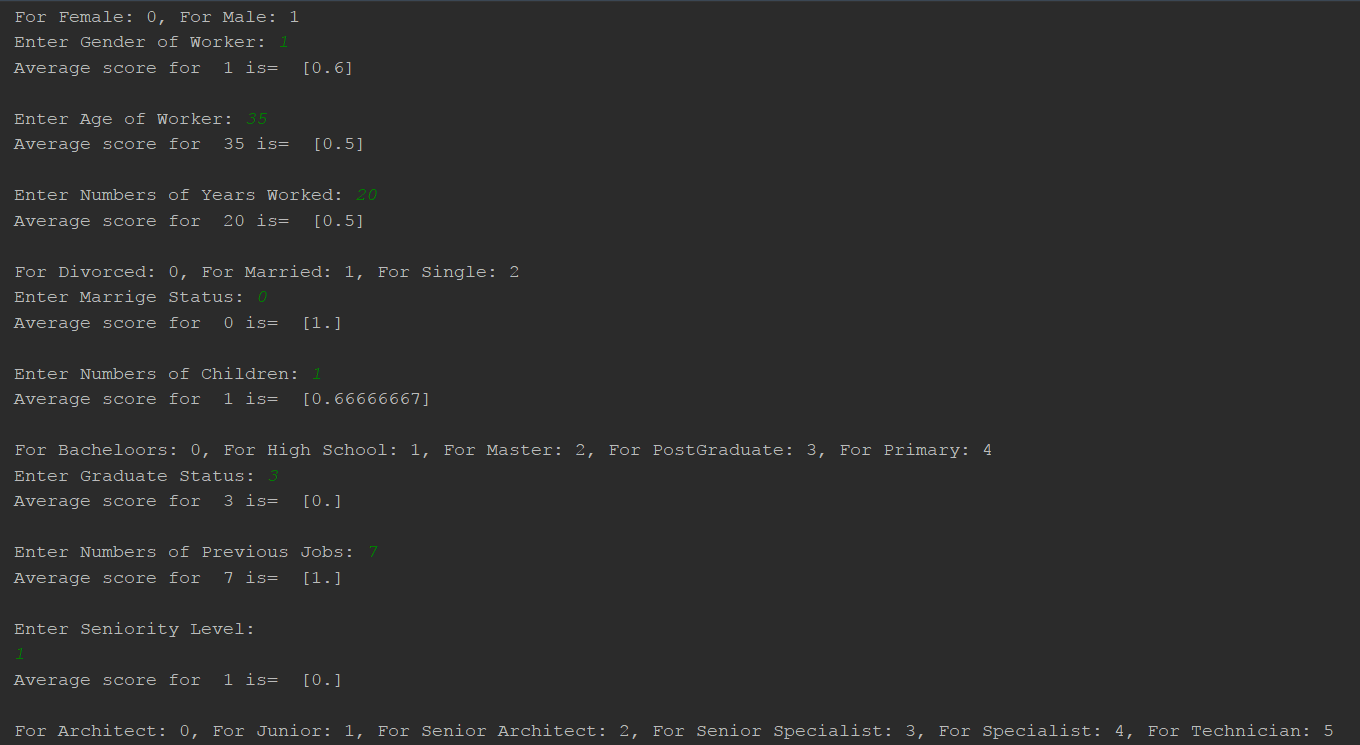
Arithmetic mean = 0.592

New employee can do this task because 0.592 is greater than 0.5 !

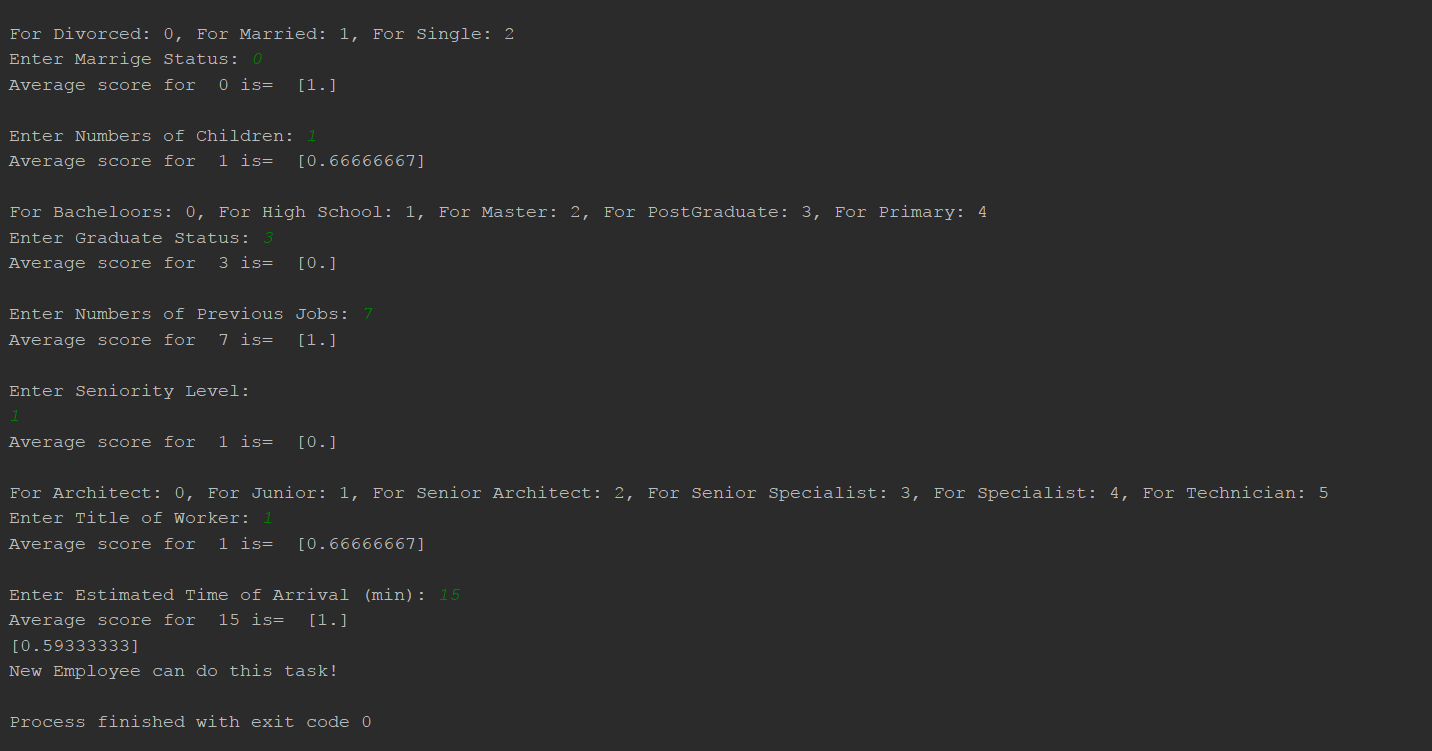
1. **Automatic Calculation**

LabelEncoder method is also used in decision tree regression algorithm for string values. In order to find correct prediction, we enter inputs in the system. Datas were fetched from the .xlsx file. An instance was created in DecisionTreeRegressor class. Datas that we entered into system were trained and got results per feature. The result is between 0 and 1. If the result is greater than 0.5, employee can do this task according to this feature. When we made calculate the arithmetic mean of these results, we got prediction per task.

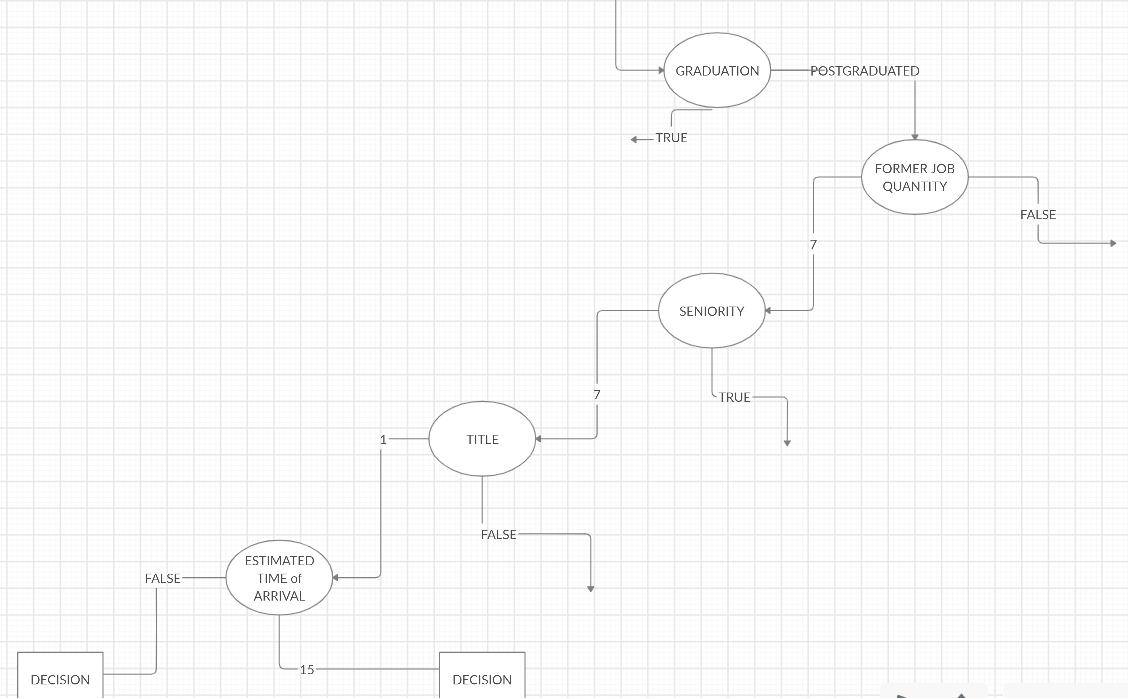
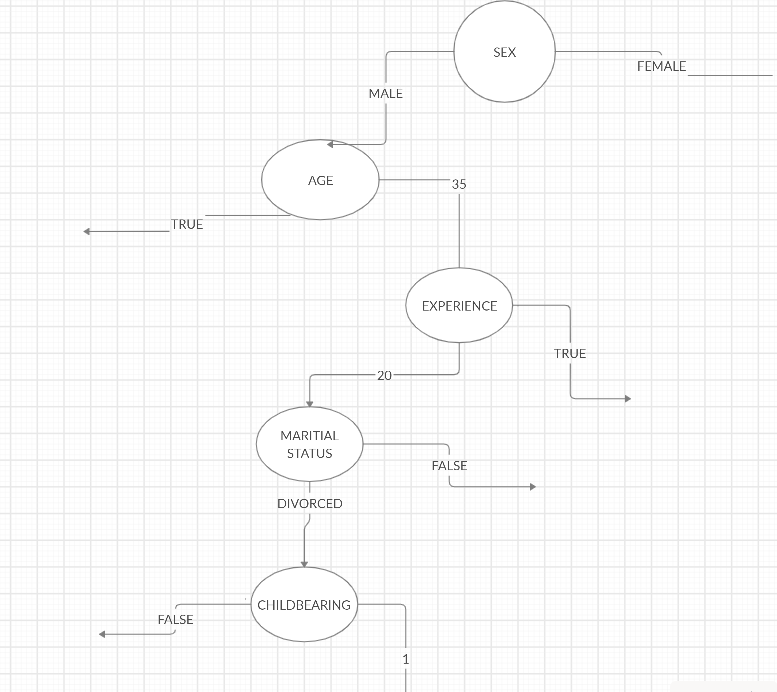
Datas that we entered into system in the below.



Decision is printed on the screen.



1. **Decision Tree Regressor Data Structure**



1. SIMPLE VECTOR REGRESSION

It separetes the datas as successfull and failed. The working principle of this algoritm is that maximizes between the distances of two classes. It is a method that used to normalize the range of properties of arguments or data.

With this data set and the polynomial model we will establish, it will be possible to predict whether a tester can do this task according to the problem level we will try to solve. Thus, we can conclude that our target variable y is “Accept” and the remaining properties are independent variables.

The SVR scikit-learn library is defined as a class of the Support Vector Machine module. To train our model, we create an object named regressor. Then we give our object x, y as parameters to the fit () method of this object. Thus, we have installed our machine. In the next step, the arguments entered from the keyboard were estimated.

1. **Calculation with One Feature**

We will use Attributes.xlsx file for one feature. In this part, we will evaluate whether datas need to scale or not one by one.

Why do we need to scale datas?

* + It is used to use data under the same metric. Think of this process as roughly turning both of them into meters before processing the kilometer and millimeter. Data mining algorithms can not separate them.
  + It associates and normalizes similar datas.

WILL BE FILLED AFTER WE DECIDE WHICH DATAS NEED TO SCALING