3/23/2017

Chinmay Ajnadkar

Sri Abbirami Gnanasekaran

Urvi Bahety

**CloudFactory**

Team 2

Table of Contents

[ Objective 2](#_Toc480406845)

[ Data 2](#_Toc480406846)

[ Predictors variables 2](#_Toc480406847)

[ From the log file 2](#_Toc480406848)

[ From demographic data 4](#_Toc480406849)

[ Some graphs 7](#_Toc480406850)

[ Correlation matrix between 11 Variables and ‘error’ 9](#_Toc480406851)

[ Response Variable 9](#_Toc480406852)

[ Random Forest Algorithm 10](#_Toc480406853)

[ Confusion matrix 10](#_Toc480406854)

[ ROC Curve 10](#_Toc480406855)

[ Cap Curve 11](#_Toc480406856)

[ Real-time-error-prediction 11](#_Toc480406857)

[ Cost Function Analysis 12](#_Toc480406858)

# **Objective**

To develop a model to predict accuracy on a current task, for which the input is provided.

# **Data**

Data consists of 5000 rows with each row as a unique task, 176 variables (features) and 1 Y-variable. Aim is to predict the value of Y variable

# **Predictors variables**

## **From the log file**

Variables: keytype\_changed, mouseclick\_button, mousemove\_distance, keytype\_changed\_time, mouseclick\_button\_time, mousemove\_distance\_time, work\_duration, speed, 169 variables with keytype changed

1) keytype\_changed – Average number of times keytype is changed based on total number of keyspressed

keytype\_changed = Number of times keytype is changed from previous row to next row  
 Number of Rows in the task

2) mouseclick\_button - Average number of times mouse is clicked

Mouseclick\_button = Number of times mouse button is clicked in a particular task . Number of rows in the task

3) mousemove\_distance - Average number of pixel mouse pointer is moved

mousemove\_distance = Total number of pixel mouse pointer is moved in a task

Number of rows in the task

4) keytype\_changed – Average number of times keytype is changed on task duration

keytype\_changed = Number of times keytype is changed from previous row to next row  
 task duration

5) mouseclick\_button - Average number of times mouse is clicked in task duration

Mouseclick\_button = Number of times mouse button is clicked in a particular task . task duration

6) mousemove\_distance - Average number of pixel mouse pointer is moved in task duration

mousemove\_distance = Total number of pixel mouse pointer is moved in a task

task duration

7) work\_duration – Duration of the task

8) speed – number of entries per unit time

speed = Total number of rows (entries) in the task  
 task duration

9) 169 predictor variables were calculated using 13\*13 transition matrix

All the Nan (not a number) values are replaced by 0, as when the mouse is used, there is no value recorded for the keytype pressed. After replacing Nan with 0, we have 13 keytypes. (0 to 12)

In a particular task transition matrix, Mi,j denotes the number of times keytype is changed from i to j.

After calculating the transition matrix, every element is divided by total number of rows in the task so that matrix is generalized irrespective of the length of task.

The method is demonstrated below.

For example, for task id = 57a280bc296d6d75f3d34350, the actual data table is given below

|  |  |
| --- | --- |
| **task\_id** | **keypress\_keytypeint** |
| 57a280bc296d6d75f3d34350 | 0 |
| 57a280bc296d6d75f3d34350 | 1 |
| 57a280bc296d6d75f3d34350 | 2 |
| 57a280bc296d6d75f3d34350 | 3 |
| 57a280bc296d6d75f3d34350 | 4 |
| 57a280bc296d6d75f3d34350 | 0 |
| 57a280bc296d6d75f3d34350 | 5 |
| 57a280bc296d6d75f3d34350 | 0 |
| 57a280bc296d6d75f3d34350 | 0 |
| 57a280bc296d6d75f3d34350 | 1 |
| 57a280bc296d6d75f3d34350 | 1 |
| 57a280bc296d6d75f3d34350 | 2 |
| 57a280bc296d6d75f3d34350 | 0 |
| 57a280bc296d6d75f3d34350 | 3 |
| 57a280bc296d6d75f3d34350 | 3 |
| 57a280bc296d6d75f3d34350 | 3 |
| 57a280bc296d6d75f3d34350 | 4 |
| 57a280bc296d6d75f3d34350 | 5 |
| 57a280bc296d6d75f3d34350 | 6 |
| 57a280bc296d6d75f3d34350 | 4 |
| 57a280bc296d6d75f3d34350 | 8 |

For this task the transition matrix (not normalized) looks like this

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **F**  **R**  **O**  **M** | **TO** | | | | | | | | | | | | | |
|  | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** |
| **0** | 1 | 2 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **1** | 0 | 1 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **2** | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **3** | 0 | 0 | 0 | 2 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **4** | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| **5** | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| **6** | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **7** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **8** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **9** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **10** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **11** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **12** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

As there are 21 rows in the task, to normalize this matrix, we divided each element of this matrix by 21.

Resulting matrix looks like this,

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** |
| **0** | 0.047 | 0.095 | 0 | 0.0476 | 0 | 0.0476 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **1** | 0 | 0.047 | 0.0952 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **2** | 0.047 | 0 | 0 | 0.0476 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **3** | 0 | 0 | 0 | 0.0952 | 0.0952 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **4** | 0.047 | 0 | 0 | 0 | 0 | 0.0476 | 0 | 0 | 0.0476 | 0 | 0 | 0 | 0 |
| **5** | 0.047 | 0 | 0 | 0 | 0 | 0 | 0.0476 | 0 | 0 | 0 | 0 | 0 | 0 |
| **6** | 0 | 0 | 0 | 0 | 0.0476 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **7** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **8** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **9** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **10** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **11** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **12** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

We reshaped this matrix as a row vector and used each element as a X variable.

## **From demographic data**

Variables: Experience, gender, education

From the demographic data, we derived the above variables.

1) Experience - Experience is calculated from starting date and the date on which data was given to us. The experience is calculated in years.

2) Gender – Dummy variables are created from Demographic data as following.

Male as 1

Female as 0

3) Education – There are 11 unique values of education in the given data. According to level of education, they are converted into 4 values (0,1,2,3), with 3 being the highest level of education and 0 being the lowest level of education. For those employees, if the education values are

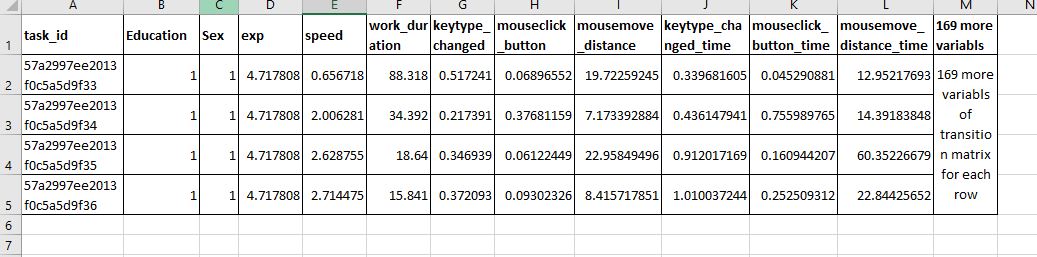
not present, 1 is assigned. The 11 education values and their respective dummy variables are as follows

|  |  |
| --- | --- |
| **Education as mentioned in data** | **Assigned values/ dummy variable** |
| Higher Secondary or equivalent level | 1 |
| Bachelor\'s or equivalent level | 2 |
| Master\'s or equivalent level | 3 |
| Secondary or equivalent level | 0 |
| Master\'s or equivalent level", "Bachelor\'s or equivalent level", "Higher Secondary or equivalent level | 3 |
| Higher Secondary or equivalent level", "Bachelor\'s or equivalent level", "Master\'s or equivalent level | 3 |
| Bachelor\'s or equivalent level", "Higher Secondary or equivalent level", "Secondary or equivalent level | 2 |
| Bachelor\'s or equivalent level", "Master\'s or equivalent level | 2 |
| Higher Secondary or equivalent level", "Bachelor\'s or equivalent level | 2 |
| Higher Secondary or equivalent level", "Secondary or equivalent level | 2 |
| Education level not available | 1 |

From this data, we formed a matrix with worked id as index and ‘Education’, ‘Experience’ and ‘Gender’ as columns. The example worker information matrix is given below

|  | **worker\_id** | **Education** | **Sex** | **exp** |
| --- | --- | --- | --- | --- |
| **0** | 501767d1cb022453170004dc | 1 | 1 | 4.717808 |
| **1** | 503b89aacb02245d320001a4 | 1 | 1 | 4.643836 |
| **2** | 50531712cb022432f3000f27 | 1 | 1 | 4.594521 |
| **3** | 5059b9b4cb02246c1200017d | 2 | 1 | 4.580822 |
| **4** | 50b59592ed964941970005b6 | 1 | 1 | 4.389041 |
| **5** | 50b6078eed9649419f000758 | 2 | 0 | 4.389041 |
| **6** | 50c08ac0c32206041f000109 | 3 | 1 | 4.367123 |
| **7** | 50c95201ed9649711300004f | 1 | 1 | 4.347945 |

We merged this matrix with the matrix created earlier to get entire X variable matrix. Each row of this matrix represents the one task.



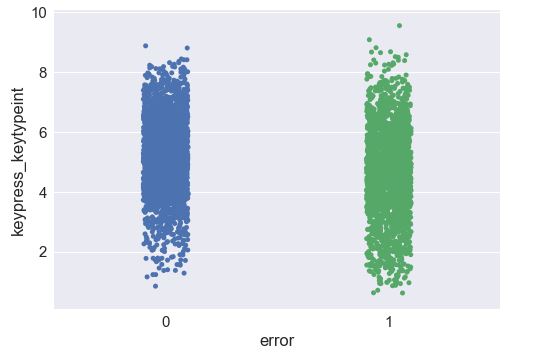
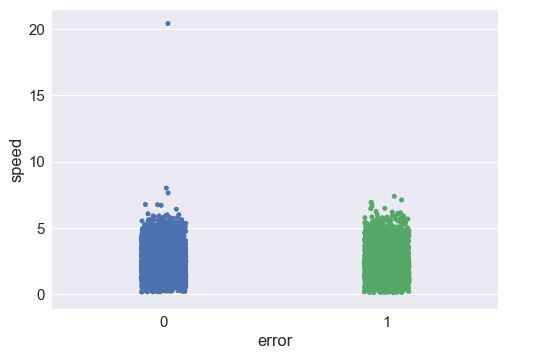
This table is attached as .CSV file as

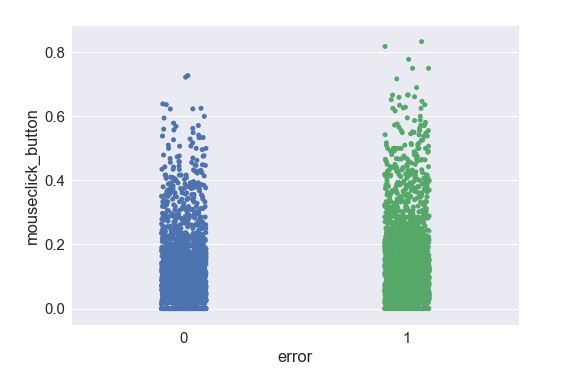
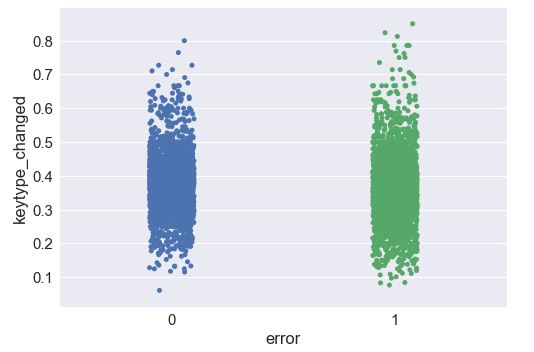
Let us call this table ‘master\_matrix’ for the convenience.

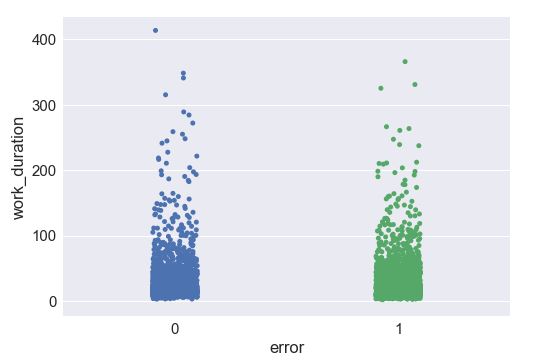
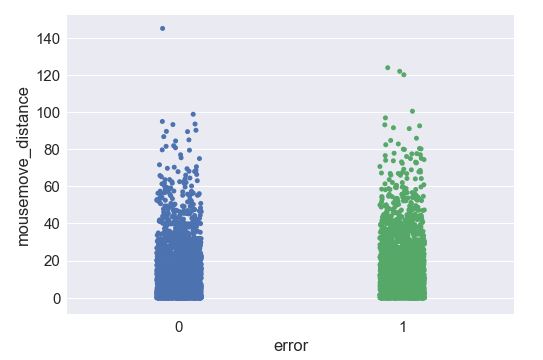
## **Some graphs**

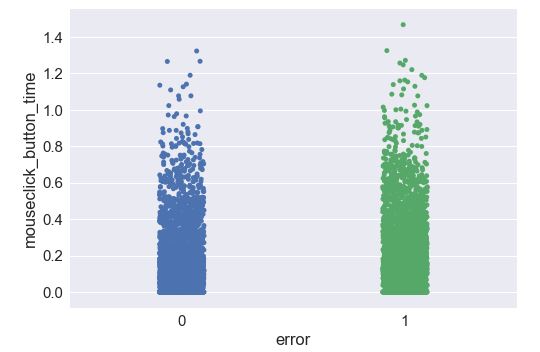
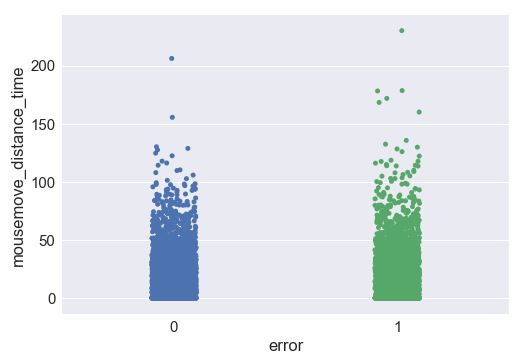
When we plot all the derived variables against the ‘error’, we get following plots.

As can be seen from the plot, there is no specific pattern between variables and ‘error’.

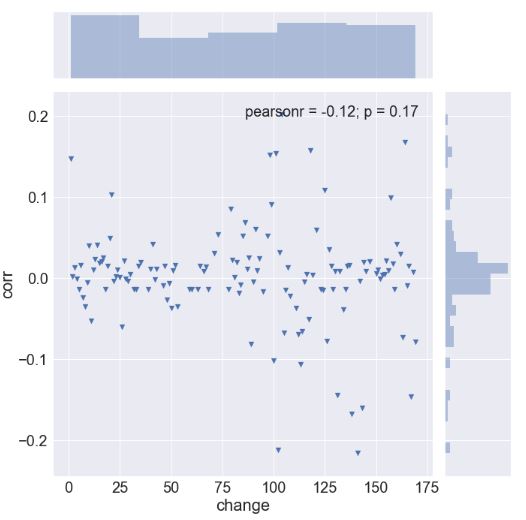
****

****

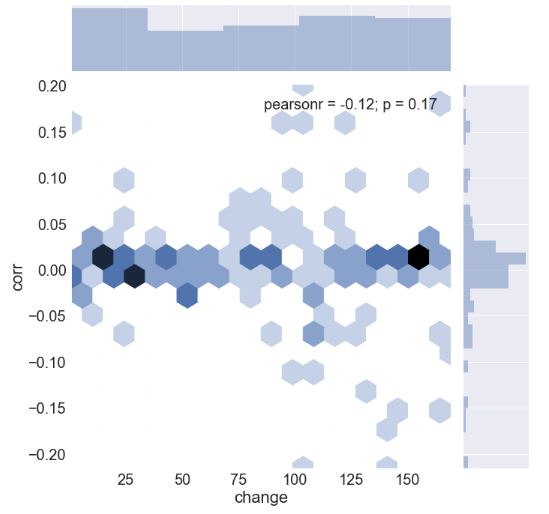
****

****

We also plotted the graph of 169 (13\*13) vs ‘error’ transition matrix. Most of the individual variables have correlation between -0.15 to 0.15, which is not significant. Although when we used those variables in algorithm, results improved significantly.

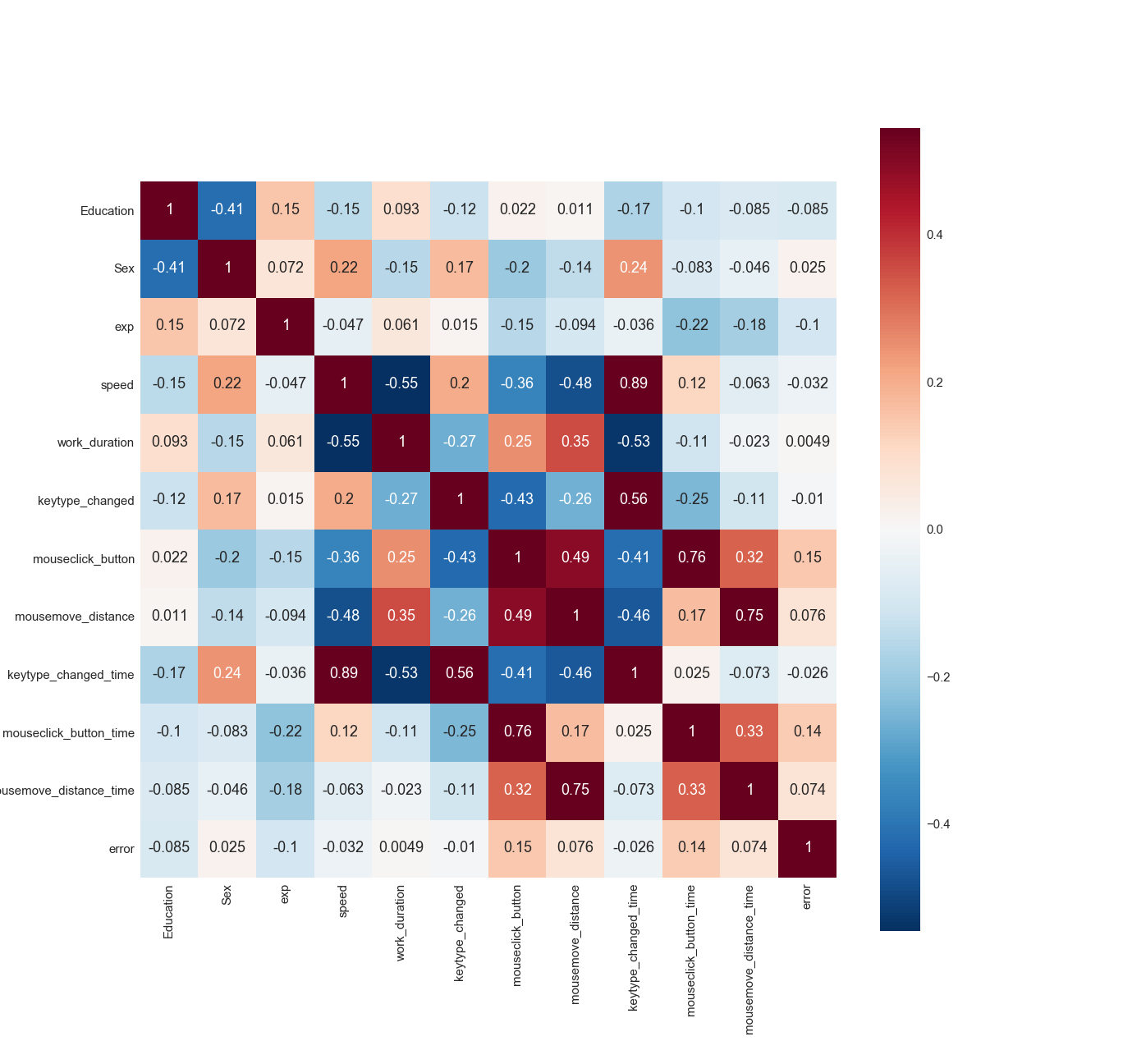


The figure on the show individual correlation of each variable with error.



The figure on the right shows the density graph of the correlations.

## **Correlation matrix between 11 Variables and ‘error’**

****

* **Response Variable** **-** Error

As only those tasks are values which has accuracy 100%, we created another variable ‘error’ with binary values 0 or 1.

1 means the task has errors and 0 means it has no errors and the accuracy is 100%

Using these X and Y variables, we ran several algorithms on the data. We used 50% data (2500 tasks) as training set and 50% data (2500 tasks) as test set. Results are as follows.

# **Random Forest Algorithm**

## **Confusion matrix**

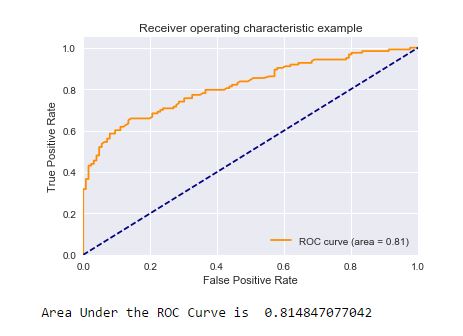
|  |  |  |  |
| --- | --- | --- | --- |
|  | **Prediction** | | |
| **ACTUAL** |  | **1** | **0** |
| **1** | 748 | 473 |
| **0** | 208 | 1055 |

Precision = 0.79

Accuracy = 0.61

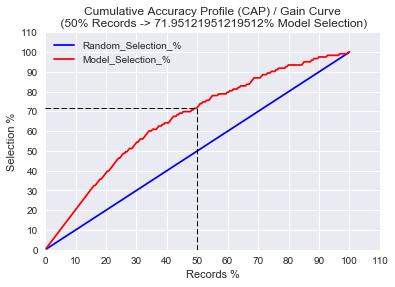
Percentage information gain = 15.99%

## **ROC Curve**



Area under the ROC curve is 0.81.

## **Cap Curve**



If we increase the training set and decrease the test set, then we see improved results

We also tried all the other classification algorithms. All the other algorithms are giving us the area under the ROC curve between 0.62 to 0.72.

# **Real-time-error-prediction**

The same algorithm can also be used for real-time error prediction.

When an employee starts working on a start, every time the log file is updated with a new row, algorithms repeats the procedure mentioned above to calculate ‘master\_matrix’ for that task. The previously trained random forest classifier can be used to predict the probability of error of the task at each step.

Although, we can set the classification boundary accordingly. We think the boundary should be 0.65. i.e. whenever the probability of error is more than 0.65 algorithm will flash a message of error. As 95% of the time, if the probability of error is more than 0.65, actual task have errors, we selected 0.65 as the classification boundary.

# **Cost Function Analysis**

We developed a model to predict error in real-time and using the 75-25 train-test split, we got the following results for error detection at 25%, 50%, 75% and 100% completion of task.

For our analysis, as our goal is to minimize the number of errors made, hence we have taken error as 1 and error free as 0

1. Confusion Matrix at 25% task completion:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Predicted | | |
| Actual |  | 1 | 0 |
| 1 | 544 | 48 |
| 0 | 553 | 97 |

1. Confusion Matrix at 50% task completion:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Predicted | | |
| Actual |  | 1 | 0 |
| 1 | 485 | 107 |
| 0 | 400 | 250 |

1. Confusion Matrix at 75% task completion:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Predicted | | |
| Actual |  | 1 | 0 |
| 1 | 389 | 203 |
| 0 | 173 | 477 |

1. Confusion Matrix at 100% task completion:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Predicted | | |
| Actual |  | 1 | 0 |
| 1 | 395 | 197 |
| 0 | 125 | 525 |

**Assumption 1 :** To minimize the loss function, we have made one reasonable assumption that once the worker is asked to redo the task in real-time, he will become very cautious and will not make any typing error, because for the data given to us, the workers have not been interrupted in between and it will be reasonable to assume that the 4 confusion matrices are independent.

**Assumption 2 :** Another assumption that we have made is that the worker gets paid according to the number of tasks he has completed and not the amount of time spent on it.

**Assumption 3:** If the worker has done one task in certain time (say, y), he will take 1.5 of that time (1.5 y) to redo that task.

Let X be the amount the worker gets paid for completing 1 task.

Let Y be the amount of time the worker takes to complete the task initially

**Case 1:** Interrupted when 25% of the task was completed

From the Confusion matrix 1:

Out of 1242 tasks:

The algorithm predicts that 544+533 =1077 tasks might have an error.

And 48+97 =145 tasks are error free. These error free tasks might have some false negatives and to catch this, because a FN can be very detrimental to business, we will triple key tasks and compare manually all tasks which algorithm deems error free.

As per our assumption 1, all tasks which were once asked to be redone will be error free in the second go, our loss function equation in terms of $ value would be:

(1077\*1.25\*X) + (145\*3\*X) = 1781.25\*X -------(1)

And our cost function in terms of time:

(1077\*(1.5+0.25) \*Y) + (145\*3\*Y) = 2319.75\*Y -------(2)

Cost for triple keying all tasks:

1242\*3\*X = 3726\*X -------(3)

1242\*3\*Y = 3726\*Y -------(4)

Adding equation (1) & (2) and (3) & (4)

1781.25\*X + 2319.75\*Y -------(5)

3726\*(X+Y) -------(6)

**Case 2:** Interrupted when 50% of the task was completed

Similarly, from Confusion matrix 2:

Out of 1242 tasks:

The algorithm predicts that 485+400 =885 tasks might have an error.

And 107+250 =357 tasks are error free. Our loss function equation in terms of $ value:

(885\*1.5\*X) + (357\*3\*X) = 2398.5\*X -------(7)

Loss function in terms of time:

(885\*2\*Y) + (357\*3\*Y) = 2841\*Y -------(8)

Adding equation (7) & (8)

2398.5\*X + 2841\*Y -------(9)

Cost for triple keying all tasks: 1242\*3\*X +1242\*Y = 3726\*(X+Y)

**Case 3:** Interrupted when 75% of the task was completed

Similarly, from Confusion matrix 3:

Out of 1242 tasks:

The algorithm predicts that 389+173 =562 tasks might have an error.

And 203+477 =680 tasks are error free. Our loss function equation in terms of $ value:

(562\*1.75\*X) + (680\*3\*X) = 3023.5\*X -------(10)

Loss function in terms of time:

(562\*2.25\*Y) + (680\*3\*X) =3304.5\*X -------(11)

Adding equation (10) & (11)

3023.5\*X + 3304.5\*Y -------(12)

Cost for triple keying all tasks: 1242\*3\*X +1242\*Y = 3726\*(X+Y)

Hence, we can see that using the real-time error prediction algorithm will decrease costs the trade-off here would be that interrupting the worker and asking him to redo the task will increase the time taken to complete the task and will also decrease his typing speed for that task as he will be cautious of not making any error.