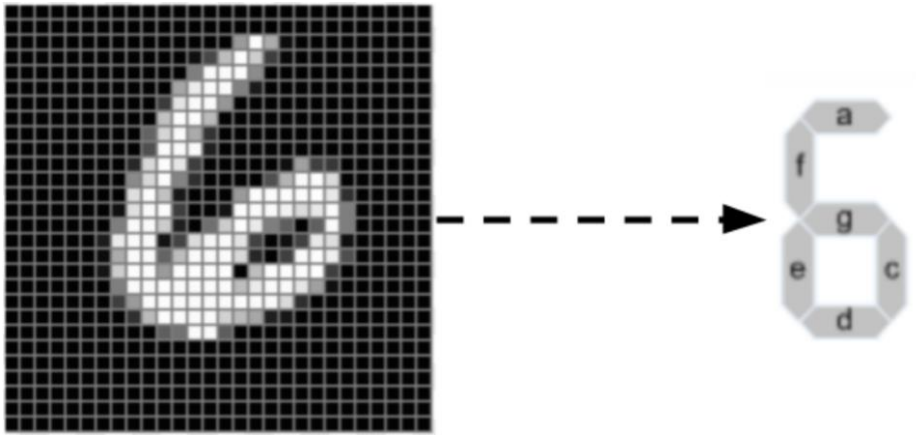
The topic of this project is to implement Gradient Boosting algorithm to transform a 28 x 28 grayscale image of a hand-written digit into its seven segment display representation.



Use mnist dataset to build predict models and classify each image according to the activation patterns of a seven segments display. Use accuracy score of each model by to measure the reliability of models. The dataset belongs to 70000 instances of 10 handwriting number pictures, each of which is identified with a numeric label (0 through 9). 60000 are training set and 10000 are testing set.

Decimal Digit	Individual Segments Illuminated						
	a	b	c	d	e	f	g
0	×	×	×	×	×	×	
1		×	×				
2	×	×		×	×		×
3	×	×	×	×			×
4		×	×			×	×
5	×		×	×		×	×
6	×		×	×	×	×	×
7	×	×	×				
8	×	×	×	×	×	×	×
9	×	×	×			×	×

To classify the original labels (0-9)into new labels(a-g), we use integer encoding to transfer 10 numbers into 1-7, which stands for model a-g. 0 is always involved as a 'False' signal to represent unpair or missing prediction. This report is mainly focus on the process of the results in steps and final conclusions, along with the reasons behind some important commands, i.e. why the reported numbers behave the way they do and some code/formula that displayed Gradient Boosting algorithm.

Step

1. Import data and do the cleaning
2. Transfer 10 different digits into 7 labels with integer encoding
3. Set training X, training Y, testing X and testing Y
4. Fit the model and make predict based on Gradient Boosting with dataset prepared in step 3
5. Build confusion matrix and accuracy score look up model accuracy roughly
6. After 7 model has been built, build confusion matrix of test_y and predicted_y around test dateset, combined with regression accuracy scores to illustrate model reliability

Work & Findings

```
from google.colab import files
import pandas as pd
import numpy as np
import seaborn as sb
from sklearn.ensemble import GradientBoostingClassifier as GBC
from sklearn.metrics import confusion_matrix as CM
import warnings
warnings.filterwarnings('ignore')
```

```
train = pd.read_csv('mnist_train.csv',header = None)
test = pd.read_csv('mnist_test.csv',header = None)
```

```
# Rename the first column as 'initial' to stands for original labels
train.rename(columns={0:'initial'}, inplace=True)
test.rename(columns={0:'initial'}, inplace=True)
print(train)
print(test)
```

The first column in each dataset stands for labels that listed in the introduction part. The integers in 1 to 785 columns stands for pixels in each instance by row..

Here is the following dataset that will be applied into the training algorithm. Only 10 columns are displayed to give a look.

Train##

```
       initial  1  2  3  4  5  6  7  ...  777  778  779  780  781  782  783  784
0            5  0  0  0  0  0  0  0  ...    0    0    0    0    0    0    0    0
1            0  0  0  0  0  0  0  0  ...    0    0    0    0    0    0    0    0
2            4  0  0  0  0  0  0  0  ...    0    0    0    0    0    0    0    0
3            1  0  0  0  0  0  0  0  ...    0    0    0    0    0    0    0    0
4            9  0  0  0  0  0  0  0  ...    0    0    0    0    0    0    0    0
...        ... .. .. .. .. .. .. .. ..  ...  ...  ...  ...  ...  ...  ...  ...
59995        8  0  0  0  0  0  0  0  ...    0    0    0    0    0    0    0    0
59996        3  0  0  0  0  0  0  0  ...    0    0    0    0    0    0    0    0
59997        5  0  0  0  0  0  0  0  ...    0    0    0    0    0    0    0    0
59998        6  0  0  0  0  0  0  0  ...    0    0    0    0    0    0    0    0
59999        8  0  0  0  0  0  0  0  ...    0    0    0    0    0    0    0    0

[60000 rows x 785 columns]
```

##Test##

```
      initial  1  2  3  4  5  6  7  ...  777  778  779  780  781  782  783  784
0           7  0  0  0  0  0  0  0  ...    0    0    0    0    0    0    0    0
1           2  0  0  0  0  0  0  0  ...    0    0    0    0    0    0    0    0
2           1  0  0  0  0  0  0  0  ...    0    0    0    0    0    0    0    0
3           0  0  0  0  0  0  0  0  ...    0    0    0    0    0    0    0    0
4           4  0  0  0  0  0  0  0  ...    0    0    0    0    0    0    0    0
...       ... .. .. .. .. .. .. .. ..  ...  ...  ...  ...  ...  ...  ...  ...
9995        2  0  0  0  0  0  0  0  ...    0    0    0    0    0    0    0    0
9996        3  0  0  0  0  0  0  0  ...    0    0    0    0    0    0    0    0
9997        4  0  0  0  0  0  0  0  ...    0    0    0    0    0    0    0    0
9998        5  0  0  0  0  0  0  0  ...    0    0    0    0    0    0    0    0
9999        6  0  0  0  0  0  0  0  ...    0    0    0    0    0    0    0    0

[10000 rows x 785 columns]
```

```
######### Model a ########
### train_a
train_a = train
train_a['a_relabel'] = 1
train_a.loc[train_a['initial']==1,'a_relabel']=0
train_a.loc[train_a['initial']==4,'a_relabel']=0 # mark 'relabel' as 0 when initial = 1 and 4
print(train_a)
# print(train_a['relabel'])
### test_a
test_a = test
test_a['a_relabel'] = 1
test_a.loc[test_a['initial']==1,'a_relabel']=0
test_a.loc[test_a['initial']==4,'a_relabel']=0 # mark 'relabel' as 0 when initial = 1 and 4
print(test_a)
```

Integer encoding is where the original label in categorical variable is removed and new variable of integers is added for each category in the variable. In this example where train_zero and test_zero has been made, train and test sets are managed respectively. First, set an all-zero column as 'relabel', when the initial label is equal to the model's name (zero at here), 'relabel' will change 0 into 1 for model a, 2 for model b, 3 for model c, 4 for model e, 5 for model f etc. In this way, we can not only redo the encoding but also classify the labels when all pixel data are in the same matrix.

```
######### relabel 1 ########
### train_one
train_one = train
train_one['relabel'] = 0
train_one.loc[train_one['initial']==1,'relabel']=1 # replace label 1 as 1, label m as 0
print(train_one)
### test_one
test_one = test
test_one['relabel'] = 0
test_one.loc[test_one['initial']==1,'relabel']=1 # set label 1 as 1, label m as 0
print(test_one)
```

```
       initial  1  2  3  4  5  6  ...  779  780  781  782  783  784  a_relabel
0            5  0  0  0  0  0  0  ...    0    0    0    0    0    0          1
1            0  0  0  0  0  0  0  ...    0    0    0    0    0    0          1
2            4  0  0  0  0  0  0  ...    0    0    0    0    0    0          0
3            1  0  0  0  0  0  0  ...    0    0    0    0    0    0          0
4            9  0  0  0  0  0  0  ...    0    0    0    0    0    0          1
...        ... .. .. .. .. .. ..  ...  ...  ...  ...  ...  ...  ...        ...
59995        8  0  0  0  0  0  0  ...    0    0    0    0    0    0          1
59996        3  0  0  0  0  0  0  ...    0    0    0    0    0    0          1
59997        5  0  0  0  0  0  0  ...    0    0    0    0    0    0          1
59998        6  0  0  0  0  0  0  ...    0    0    0    0    0    0          1
59999        8  0  0  0  0  0  0  ...    0    0    0    0    0    0          1

[60000 rows x 786 columns]
      initial  1  2  3  4  5  6  ...  779  780  781  782  783  784  a_relabel
0           7  0  0  0  0  0  0  ...    0    0    0    0    0    0          1
1           2  0  0  0  0  0  0  ...    0    0    0    0    0    0          1
2           1  0  0  0  0  0  0  ...    0    0    0    0    0    0          0
3           0  0  0  0  0  0  0  ...    0    0    0    0    0    0          1
4           4  0  0  0  0  0  0  ...    0    0    0    0    0    0          0
...       ... .. .. .. .. .. ..  ...  ...  ...  ...  ...  ...  ...        ...
9995        2  0  0  0  0  0  0  ...    0    0    0    0    0    0          1
9996        3  0  0  0  0  0  0  ...    0    0    0    0    0    0          1
9997        4  0  0  0  0  0  0  ...    0    0    0    0    0    0          0
9998        5  0  0  0  0  0  0  ...    0    0    0    0    0    0          1
9999        6  0  0  0  0  0  0  ...    0    0    0    0    0    0          1

[10000 rows x 786 columns]
```

```
### train x
train_a_x = train_a.iloc[:,1:785]
### train y
train_a_y = train_a['a_relabel']
### test_x
test_a_x = test_a.iloc[:,1:785]
### test y
test_a_y = test_a['a_relabel']
```

```
clf_a = GBC(random_state=0).fit(train_a_x, train_a_y)
pred_a = clf_a.predict(test_a_x)
print(clf_a.score(test_a_x, test_a_y))
```

```
0.9787
```

In order to show the accuracy of the regression model, it needs to be examined using different metrics and presentations. The first is the accuracy score and confusion matrix of each single model, which presents in a two-dimensional matrix the number of errors between the test set y-values obtained by model prediction and the true y-values of the test set under different labels. In this confusion matrix, we shared 7 models, each with labels of 0 and 1, so the output of the confusion matrix is a 2*2 matrix.

```
print(CM(pred_a, test_a_y))
```

```
[[1964   60]
 [ 153 7823]]
```

```
#########  relabel  2  ########
###  train_two
train_two  =  train
train_two['relabel']  =  0
train_two.loc[train_two['initial']==2,'relabel']=1  #  replace  label  2  as  1,  label  m  as  0
print(train_two)
###  test_two
test_two  =  test
test_two['relabel']  =  0
test_two.loc[test_two['initial']==2,'relabel']=1  #  set  label  2  as  1,  label  m  as  0
print(test_two)
```

```
       initial  1  2  3  4  5  6  ...  779  780  781  782  783  784  b_relabel
0            5  0  0  0  0  0  0  ...    0    0    0    0    0    0          0
1            0  0  0  0  0  0  0  ...    0    0    0    0    0    0          2
2            4  0  0  0  0  0  0  ...    0    0    0    0    0    0          2
3            1  0  0  0  0  0  0  ...    0    0    0    0    0    0          2
4            9  0  0  0  0  0  0  ...    0    0    0    0    0    0          2
...        ... .. .. .. .. .. .. ..  ...  ...  ...  ...  ...  ...  ...        ...
59995        8  0  0  0  0  0  0  ...    0    0    0    0    0    0          2
59996        3  0  0  0  0  0  0  ...    0    0    0    0    0    0          2
59997        5  0  0  0  0  0  0  ...    0    0    0    0    0    0          0
59998        6  0  0  0  0  0  0  ...    0    0    0    0    0    0          0
59999        8  0  0  0  0  0  0  ...    0    0    0    0    0    0          2

[60000 rows x 786 columns]
      initial  1  2  3  4  5  6  ...  779  780  781  782  783  784  b_relabel
0           7  0  0  0  0  0  0  ...    0    0    0    0    0    0          2
1           2  0  0  0  0  0  0  ...    0    0    0    0    0    0          2
2           1  0  0  0  0  0  0  ...    0    0    0    0    0    0          2
3           0  0  0  0  0  0  0  ...    0    0    0    0    0    0          2
4           4  0  0  0  0  0  0  ...    0    0    0    0    0    0          2
...       ... .. .. .. .. .. .. ..  ...  ...  ...  ...  ...  ...  ...        ...
9995        2  0  0  0  0  0  0  ...    0    0    0    0    0    0          2
9996        3  0  0  0  0  0  0  ...    0    0    0    0    0    0          2
9997        4  0  0  0  0  0  0  ...    0    0    0    0    0    0          2
9998        5  0  0  0  0  0  0  ...    0    0    0    0    0    0          0
9999        6  0  0  0  0  0  0  ...    0    0    0    0    0    0          0

[10000 rows x 786 columns]
```

```
train_b_x  =  train_b.iloc[:,1:785]
train_b_y  =  train_b['b_relabel']
test_b_x  =  test_b.iloc[:,1:785]
test_b_y  =  test_b['b_relabel']
```

```
clf_b  =  GBC(random_state=0).fit(train_b_x,  train_b_y)
pred_b  =  clf_b.predict(test_b_x)
print(clf_b.score(test_b_x,  test_b_y))
```

```
0.9716
```

```
print(CM(pred_b,test_b_y))
```

```
[[1631   65]
 [ 219 8085]]
```

```
#########  Model  c  ########
###  train_c
train_c  =  train
train_c['c_relabel']  =  3
train_c.loc[train_c['initial']==2,'c_relabel']=0  #  mark  'relabel'  as  0  when  initial  =  2
print(train_c)
###  test_c
test_c  =  test
test_c['c_relabel']  =  3
test_c.loc[test_c['initial']==2,'c_relabel']=0  #  mark  'relabel'  as  0  when  initial  =  2
print(test_c)
```

```
       initial  1  2  3  4  5  ...  782  783  784  a_relabel  b_relabel  c_relabel
0            5  0  0  0  0  0  ...    0    0    0          1          0          3
1            0  0  0  0  0  0  ...    0    0    0          1          2          3
2            4  0  0  0  0  0  ...    0    0    0          0          2          3
3            1  0  0  0  0  0  ...    0    0    0          0          2          3
4            9  0  0  0  0  0  ...    0    0    0          1          2          3
...        ... .. .. .. .. .. ..  ...  ...  ...  ...        ...        ...        ...
59995        8  0  0  0  0  0  ...    0    0    0          1          2          3
59996        3  0  0  0  0  0  ...    0    0    0          1          2          3
59997        5  0  0  0  0  0  ...    0    0    0          1          0          3
59998        6  0  0  0  0  0  ...    0    0    0          1          0          3
59999        8  0  0  0  0  0  ...    0    0    0          1          2          3

[60000 rows x 788 columns]
      initial  1  2  3  4  5  ...  782  783  784  a_relabel  b_relabel  c_relabel
0           7  0  0  0  0  0  ...    0    0    0          1          2          3
1           2  0  0  0  0  0  ...    0    0    0          1          2          0
2           1  0  0  0  0  0  ...    0    0    0          0          2          3
3           0  0  0  0  0  0  ...    0    0    0          1          2          3
4           4  0  0  0  0  0  ...    0    0    0          0          2          3
...       ... .. .. .. .. .. ..  ...  ...  ...  ...        ...        ...        ...
9995        2  0  0  0  0  0  ...    0    0    0          1          2          0
9996        3  0  0  0  0  0  ...    0    0    0          1          2          3
9997        4  0  0  0  0  0  ...    0    0    0          0          2          3
9998        5  0  0  0  0  0  ...    0    0    0          1          0          3
9999        6  0  0  0  0  0  ...    0    0    0          1          0          3

[10000 rows x 788 columns]
```

```
train_c_x  =  train_c.iloc[:,1:785]
train_c_y  =  train_c['c_relabel']
test_c_x  =  test_c.iloc[:,1:785]
test_c_y  =  test_c['c_relabel']
```

```
clf_c  =  GBC(random_state=0).fit(train_c_x,  train_c_y)
pred_c  =  clf_c.predict(test_c_x)
print(clf_c.score(test_c_x,  test_c_y))
```

```
0.9851
```

```
print(CM(pred_c,test_c_y))
```

```
[[ 904   21]
 [ 128 8947]]
```

```
######### Model d ########
### train_d
train_d = train
train_d['d_relabel'] = 4
train_d.loc[train_d['initial']==1,'d_relabel']=0
train_d.loc[train_d['initial']==4,'d_relabel']=0 # mark 'relabel' as 0 when initial = 1,4,7,9
train_d.loc[train_d['initial']==7,'d_relabel']=0
train_d.loc[train_d['initial']==9,'d_relabel']=0
print(train_d)
### test_d
test_d = test
test_d['d_relabel'] = 4
test_d.loc[test_d['initial']==1,'d_relabel']=0
test_d.loc[test_d['initial']==4,'d_relabel']=0
test_d.loc[test_d['initial']==7,'d_relabel']=0
test_d.loc[test_d['initial']==9,'d_relabel']=0 # mark 'relabel' as 0 when initial = 1,4,7,9
print(test_d)
```

```
       initial  1  2  3  4  ...  784  a_relabel  b_relabel  c_relabel  d_relabel
0            5  0  0  0  0  ...    0          1          0          3          4
1            0  0  0  0  0  ...    0          1          2          3          4
2            4  0  0  0  0  ...    0          0          2          3          0
3            1  0  0  0  0  ...    0          0          2          3          0
4            9  0  0  0  0  ...    0          1          2          3          0
...        ... .. .. .. ..  ...  ...        ...        ...        ...        ...
59995        8  0  0  0  0  ...    0          1          2          3          4
59996        3  0  0  0  0  ...    0          1          2          3          4
59997        5  0  0  0  0  ...    0          1          0          3          4
59998        6  0  0  0  0  ...    0          1          0          3          4
59999        8  0  0  0  0  ...    0          1          2          3          4

[60000 rows x 789 columns]
      initial  1  2  3  4  ...  784  a_relabel  b_relabel  c_relabel  d_relabel
0           7  0  0  0  0  ...    0          1          2          3          0
1           2  0  0  0  0  ...    0          1          2          0          4
2           1  0  0  0  0  ...    0          0          2          3          0
3           0  0  0  0  0  ...    0          1          2          3          4
4           4  0  0  0  0  ...    0          0          2          3          0
...       ... .. .. .. ..  ...  ...        ...        ...        ...        ...
9995        2  0  0  0  0  ...    0          1          2          0          4
9996        3  0  0  0  0  ...    0          1          2          3          4
9997        4  0  0  0  0  ...    0          0          2          3          0
9998        5  0  0  0  0  ...    0          1          0          3          4
9999        6  0  0  0  0  ...    0          1          0          3          4

[10000 rows x 789 columns]
```

```
train_d_x  =  train_d.iloc[:,1:785]
train_d_y  =  train_d['d_relabel']
test_d_x  =  test_d.iloc[:,1:785]
test_d_y  =  test_d['d_relabel']
```

```
clf_d  =  GBC(random_state=0).fit(train_d_x,  train_d_y)
pred_d  =  clf_d.predict(test_d_x)
print(clf_d.score(test_d_x,  test_d_y))
```

0.9723

```
print(CM(pred_d,  test_d_y))
```

```
[[4008  131]
 [ 146 5715]]
```

```
#########  Model  e  ########
###  train_e
train_e  =  train
train_e['e_relabel']  =  0
train_e.loc[train_e['initial']==0,'e_relabel']=5
train_e.loc[train_e['initial']==2,'e_relabel']=5  #  mark  'relabel'  as  5  when  initial  =  1,4,7,9
train_e.loc[train_e['initial']==6,'e_relabel']=5
train_e.loc[train_e['initial']==8,'e_relabel']=5
print(train_e)
###  test_e
test_e  =  test
test_e['e_relabel']  =  0
test_e.loc[test_e['initial']==0,'e_relabel']=5
test_e.loc[test_e['initial']==2,'e_relabel']=5
test_e.loc[test_e['initial']==6,'e_relabel']=5
test_e.loc[test_e['initial']==8,'e_relabel']=5  #  mark  'relabel'  as  5  when  initial  =  1,4,7,9
print(test_e)
```

```
       initial  1  2  3  ...  b_relabel  c_relabel  d_relabel  e_relabel
0            5  0  0  0  ...         0          3          4          0
1            0  0  0  0  ...         2          3          4          5
2            4  0  0  0  ...         2          3          0          0
3            1  0  0  0  ...         2          3          0          0
4            9  0  0  0  ...         2          3          0          0
...        ... .. .. ..  ...       ...        ...        ...        ...
59995        8  0  0  0  ...         2          3          4          5
59996        3  0  0  0  ...         2          3          4          0
59997        5  0  0  0  ...         0          3          4          0
59998        6  0  0  0  ...         0          3          4          5
59999        8  0  0  0  ...         2          3          4          5

[60000 rows x 790 columns]
      initial  1  2  3  ...  b_relabel  c_relabel  d_relabel  e_relabel
0           7  0  0  0  ...         2          3          0          0
1           2  0  0  0  ...         2          0          4          5
2           1  0  0  0  ...         2          3          0          0
3           0  0  0  0  ...         2          3          4          5
4           4  0  0  0  ...         2          3          0          0
...       ... .. .. ..  ...       ...        ...        ...        ...
9995        2  0  0  0  ...         2          0          4          5
9996        3  0  0  0  ...         2          3          4          0
9997        4  0  0  0  ...         2          3          0          0
9998        5  0  0  0  ...         0          3          4          0
9999        6  0  0  0  ...         0          3          4          5

[10000 rows x 790 columns]
```

```
train_e_x  =  train_e.iloc[:,1:785]
train_e_y  =  train_e['e_relabel']
test_e_x  =  test_e.iloc[:,1:785]
test_e_y  =  test_e['e_relabel']
```

```
clf_e  =  GBC(random_state=0).fit(train_e_x,  train_e_y)
pred_e  =  clf_e.predict(test_e_x)
print(clf_e.score(test_e_x,  test_e_y))
```

```
0.9574
```

```
print(CM(pred_e,  test_e_y))
```

```
[[5845  215]
 [ 211 3729]]
```

```
######### Model f ########
### train_f
train_f = train
train_f['f_relabel'] = 6
train_f.loc[train_f['initial']==1,'f_relabel']=0
train_f.loc[train_f['initial']==2,'f_relabel']=0 # mark 'relabel' as 0 when initial = 1,2,3,7
train_f.loc[train_f['initial']==3,'f_relabel']=0
train_f.loc[train_f['initial']==7,'f_relabel']=0
print(train_f)
### test_f
test_f = test
test_f['f_relabel'] = 6
test_f.loc[test_f['initial']==1,'f_relabel']=0
test_f.loc[test_f['initial']==2,'f_relabel']=0
test_f.loc[test_f['initial']==3,'f_relabel']=0
test_f.loc[test_f['initial']==7,'f_relabel']=0 # mark 'relabel' as 0 when initial = 1,2,3,7
print(test_f)
```

```
       initial  1  2  3  ...  c_relabel  d_relabel  e_relabel  f_relabel
0            5  0  0  0  ...          3          4          0          6
1            0  0  0  0  ...          3          4          5          6
2            4  0  0  0  ...          3          0          0          6
3            1  0  0  0  ...          3          0          0          0
4            9  0  0  0  ...          3          0          0          6
...        ... .. .. ..  ...        ...        ...        ...        ...
59995        8  0  0  0  ...          3          4          5          6
59996        3  0  0  0  ...          3          4          0          0
59997        5  0  0  0  ...          3          4          0          6
59998        6  0  0  0  ...          3          4          5          6
59999        8  0  0  0  ...          3          4          5          6

[60000 rows x 791 columns]
      initial  1  2  3  ...  c_relabel  d_relabel  e_relabel  f_relabel
0           7  0  0  0  ...          3          0          0          0
1           2  0  0  0  ...          0          4          5          0
2           1  0  0  0  ...          3          0          0          0
3           0  0  0  0  ...          3          4          5          6
4           4  0  0  0  ...          3          0          0          6
...       ... .. .. ..  ...        ...        ...        ...        ...
9995        2  0  0  0  ...          0          4          5          0
9996        3  0  0  0  ...          3          4          0          0
9997        4  0  0  0  ...          3          0          0          6
9998        5  0  0  0  ...          3          4          0          6
9999        6  0  0  0  ...          3          4          5          6

[10000 rows x 791 columns]
```

```
train_f_x = train_f.iloc[:,1:785]
train_f_y = train_f['f_relabel']
test_f_x = test_f.iloc[:,1:785]
test_f_y = test_f['f_relabel']
```

```
clf_f = GBC(random_state=0).fit(train_f_x, train_f_y)
pred_f = clf_f.predict(test_f_x)
print(clf_f.score(test_f_x, test_f_y))
```

```
0.9617
```

```
print(CM(pred_f, test_f_y))
```

```
[[3978  156]
 [ 227 5639]]
```

```
######### Model g ########
### train_g
train_g = train
train_g['g_relabel'] = 7
train_g.loc[train_g['initial']==0,'g_relabel']=0
train_g.loc[train_g['initial']==1,'g_relabel']=0 # mark 'relabel' as 0 when initial = 0,1,7
train_g.loc[train_g['initial']==7,'g_relabel']=0
print(train_g)
### test_g
test_g = test
test_g['g_relabel'] = 7
test_g.loc[test_g['initial']==0,'g_relabel']=0
test_g.loc[test_g['initial']==1,'g_relabel']=0 # mark 'relabel' as 0 when initial = 0,1,7
test_g.loc[test_g['initial']==7,'g_relabel']=0
print(test_g)
```

```
       initial  1  2  3  ...  d_relabel  e_relabel  f_relabel  g_relabel
0            5  0  0  0  ...          4          0          6          7
1            0  0  0  0  ...          4          5          6          0
2            4  0  0  0  ...          0          0          6          7
3            1  0  0  0  ...          0          0          0          0
4            9  0  0  0  ...          0          0          6          7
...        ... .. .. ..  ...        ...        ...        ...        ...
59995        8  0  0  0  ...          4          5          6          7
59996        3  0  0  0  ...          4          0          0          7
59997        5  0  0  0  ...          4          0          6          7
59998        6  0  0  0  ...          4          5          6          7
59999        8  0  0  0  ...          4          5          6          7

[60000 rows x 792 columns]
      initial  1  2  3  ...  d_relabel  e_relabel  f_relabel  g_relabel
0           7  0  0  0  ...          0          0          0          0
1           2  0  0  0  ...          4          5          0          7
2           1  0  0  0  ...          0          0          0          0
3           0  0  0  0  ...          4          5          6          0
4           4  0  0  0  ...          0          0          6          7
...       ... .. .. ..  ...        ...        ...        ...        ...
9995        2  0  0  0  ...          4          5          0          7
9996        3  0  0  0  ...          4          0          0          7
9997        4  0  0  0  ...          0          0          6          7
9998        5  0  0  0  ...          4          0          6          7
9999        6  0  0  0  ...          4          5          6          7

[10000 rows x 792 columns]
```

```
train_g_x = train_g.iloc[:,1:785]
train_g_y = train_g['g_relabel']
test_g_x = test_g.iloc[:,1:785]
test_g_y = test_g['g_relabel']
```

```
clf_g = GBC(random_state=0).fit(train_g_x, train_g_y)
pred_g = clf_g.predict(test_g_x)
print(clf_g.score(test_g_x, test_g_y))
```

```
0.9663
```

```
print(CM(pred_g, test_g_y))
```

```
[[2924  118]
 [ 219 6739]]
```

```
test_relabel = pd.concat(
        [test_g['a_relabel'], test_g['b_relabel'], test_g['c_relabel'], test_g['d_relabel'],
          test_g['e_relabel'], test_g['f_relabel'], test_g['g_relabel']], axis = 0
         )
pred_a = pd.Series(pred_a, dtype = 'uint8')
pred_b = pd.Series(pred_b, dtype = 'uint8')
pred_c = pd.Series(pred_c, dtype = 'uint8')
pred_d = pd.Series(pred_d, dtype = 'uint8')
pred_e = pd.Series(pred_e, dtype = 'uint8')
pred_f = pd.Series(pred_f, dtype = 'uint8')
pred_g = pd.Series(pred_g, dtype = 'uint8')
test_pred = pd.concat([pred_a, pred_b, pred_c, pred_d, pred_e, pred_f, pred_g], axis = 0)
test_relabel
test_pred
```

```
print(CM(test_pred, test_relabel, labels = [0, 1, 2, 3, 4, 5, 6, 7]))
```

```
[[21254    60    65    21   131   215   156   118]
 [  153  7823     0     0     0     0     0     0]
 [  219     0  8085     0     0     0     0     0]
 [  128     0     0  8947     0     0     0     0]
 [  146     0     0     0  5715     0     0     0]
 [  211     0     0     0     0  3729     0     0]
 [  227     0     0     0     0     0  5639     0]
 [  219     0     0     0     0     0     0  6739]]
```

```
from sklearn.metrics import accuracy_score
print(accuracy_score(test_pred, test_relabel))
```

```
0.9704428571428572
```

Accuracy score of 7 models are listed below.

Model	a	b	c	d	e	f	g	Test_pred
Accuracy Score	0.9787	0.9716	0.9851	0.9723	0.9574	0.9617	0.9633	0.9704

The predicted results of the test dataset for all seven models, and the true results of the dataset were all fitted above 0.95. This indicates a good fit. The best performing model in this case is model c with a score of 0.9851. The worst performing model is model e with a score of 0.9574.

Finally, we integrate all a-g models and list the confusion matrix of the test set. This is also reflected in the individual confusion matrices for each model and the confusion matrix for the test set as a whole. The diagonal values are the number of data correctly predicted for each model. Model c has the highest number of diagonals, model e has the lowest number of diagonals.

The accuracy score of the confusion matrix is 0.9704, which is close to the simple average accuracy score of model a-g. This indicates that the seven models are well distributed across the test set and all are well predicted.

References

Brownlee J. (2020, June 12). Ordinal and One-Hot Encodings for Categorical Data. Retrieved from, https://machinelearningmastery.com/one-hot-encoding-for-categorical-data/
N.N. (n.d.) w3schools. https://www.w3schools.com/python/python_arrays.asp