COL865: Special Topics in Computer Application

rajat singh - 2020CSZ8507

October 2022

1 Dataset 1: BitcoinHeist Ransomware Dataset

1.1 Dataset Analysis

Shape of the dataset: (2916697, 10) where we have 2916697 rows and 10 columns in the dataset.
Total classes: 29
Number of samples per class:

```
{'princetonCerber': 9223,
 'princetonLocky': 6625,
 'montrealCryptoLocker': 9315,
 'montrealCryptXXX': 2419,
 'paduaCryptoWall': 12390,
 'montrealWannaCry': 28,
 'montrealDMALockerv3': 354,
 'montrealCryptoTorLocker2015': 55,
 'montrealSamSam': 62,
 'montrealFlyper': 9,
 'montrealNoobCrypt': 483,
 'montrealDMALocker': 251,
 'montrealGlobe': 32,
 'montrealEDA2': 6,
 'paduaKeRanger': 10,
 'montrealVenusLocker': 7,
 'montrealXTPLocker': 8,
 'paduaJigsaw': 2,
 'montrealGlobev3': 34,
 'montrealJigSaw': 4,
 'montrealXLockerv5.0': 7,
 'montrealXLocker': 1,
 'montrealRazy': 13,
 'montrealCryptConsole': 7,
 'montrealGlobeImposter': 55,
 'montrealSam': 1,
 'montrealComradeCircle': 1,
 'montrealAPT': 11,
 'white': 2875284}
```

Figure 1: Number of samples per class

Unique values in each column:

```
Unique values in column 1 : 2631095
Unique values in column 2 : 8
Unique values in column 3 : 365
Unique values in column 4 : 73
Unique values in column 5 : 785669
Unique values in column 6 : 11572
Unique values in column 7 : 10168
Unique values in column 8 : 814
Unique values in column 9 : 1866365
Unique values in column 10 : 29
```

Figure 2: Unique values in each column

1.2 Models

1. Random Forests

Since column1 contain all the values as the unique values thus it will not help in learning anything. Thus I dropped the column. Now column 10 contain large number of unique values as compared to other columns.

CASE 1: I split the dataset into test and train set by randomly selecting the samples from the dataset. But the issue was that most of the time i was getting only one class samples in the training dataset, which make the prediction of one class only. Figure 3

Accuracy: 0.9857973538678854

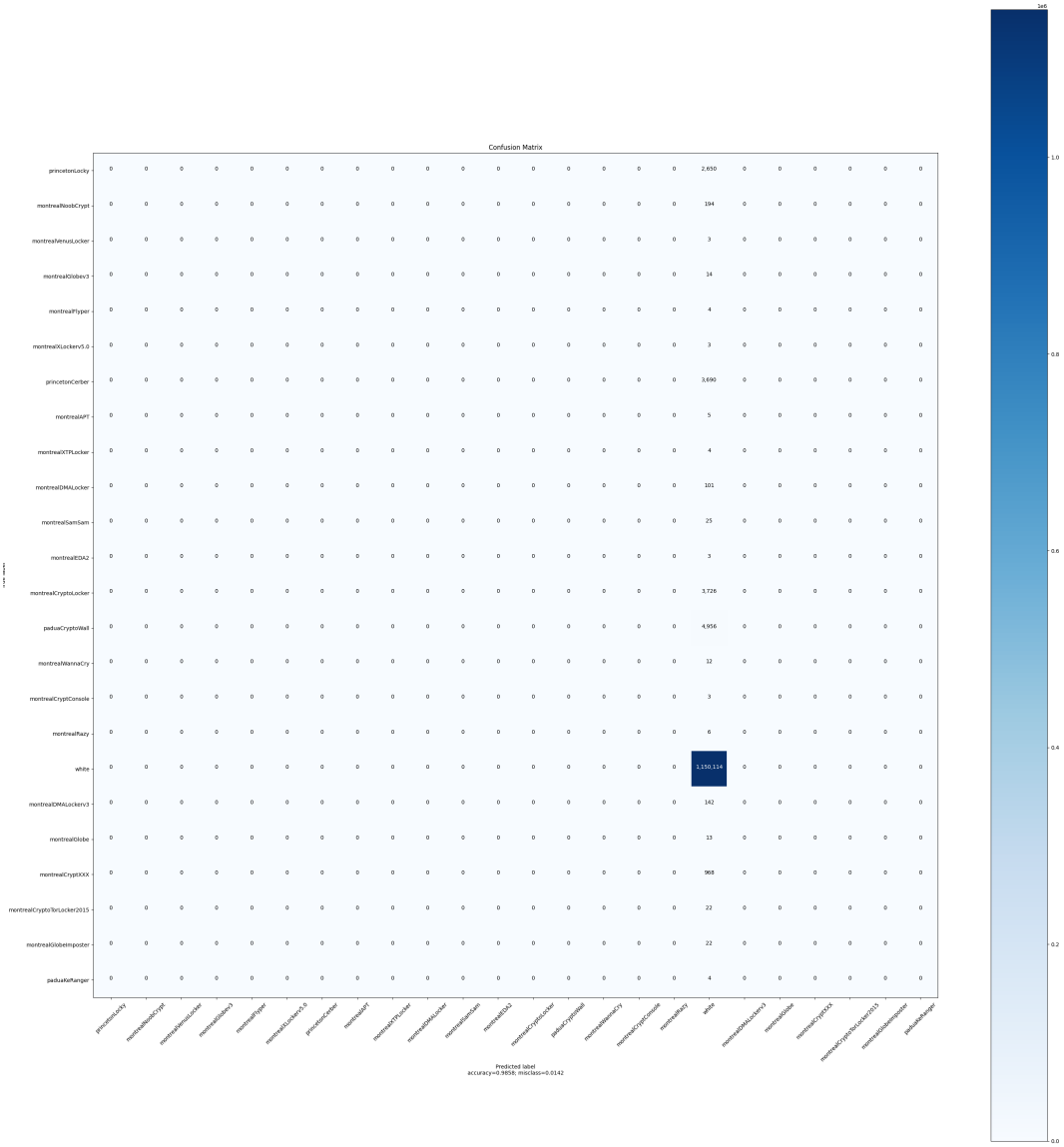

Figure 3: Results - Random Forests case1

CASE 2: From each class I sampled X% of samples as the train set and (100-X)% as the training set. It does not improve the accuracy. I was still getting the same result and the only prediction I was getting was "white". Figure 4

Accuracy: 0.9857973538678854

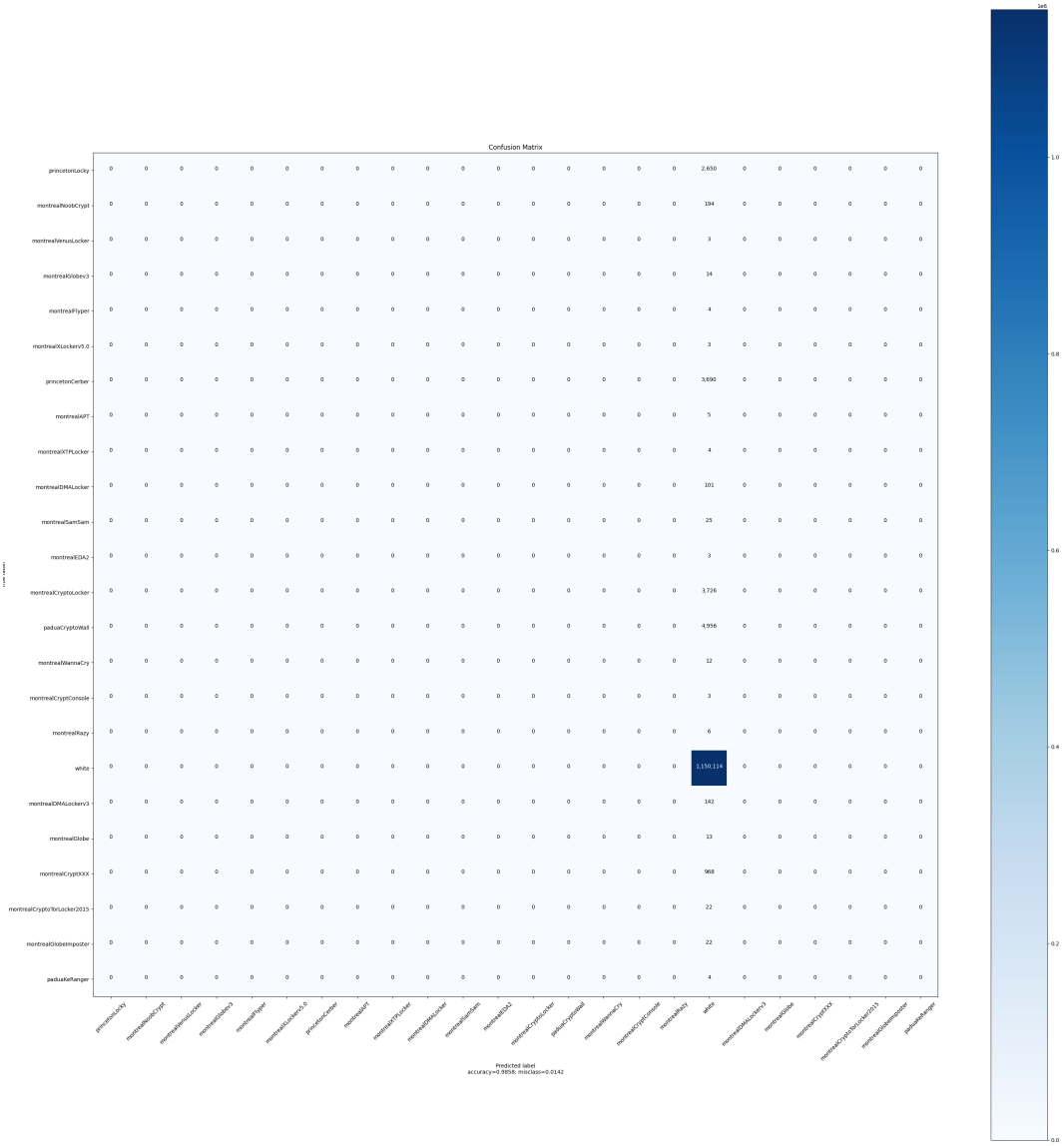

Figure 4: Results - Random Forests case2

CASE 3: From each class I sampled X% of samples as the train set and (100-X)% as the training set. And assigned weight to each class (1 - (number of samples in the class/total number of samples)). In this case I was getting better accuracy and I started getting other classes as the classification. Figure 5

Accuracy: 0.0.9880524632205464

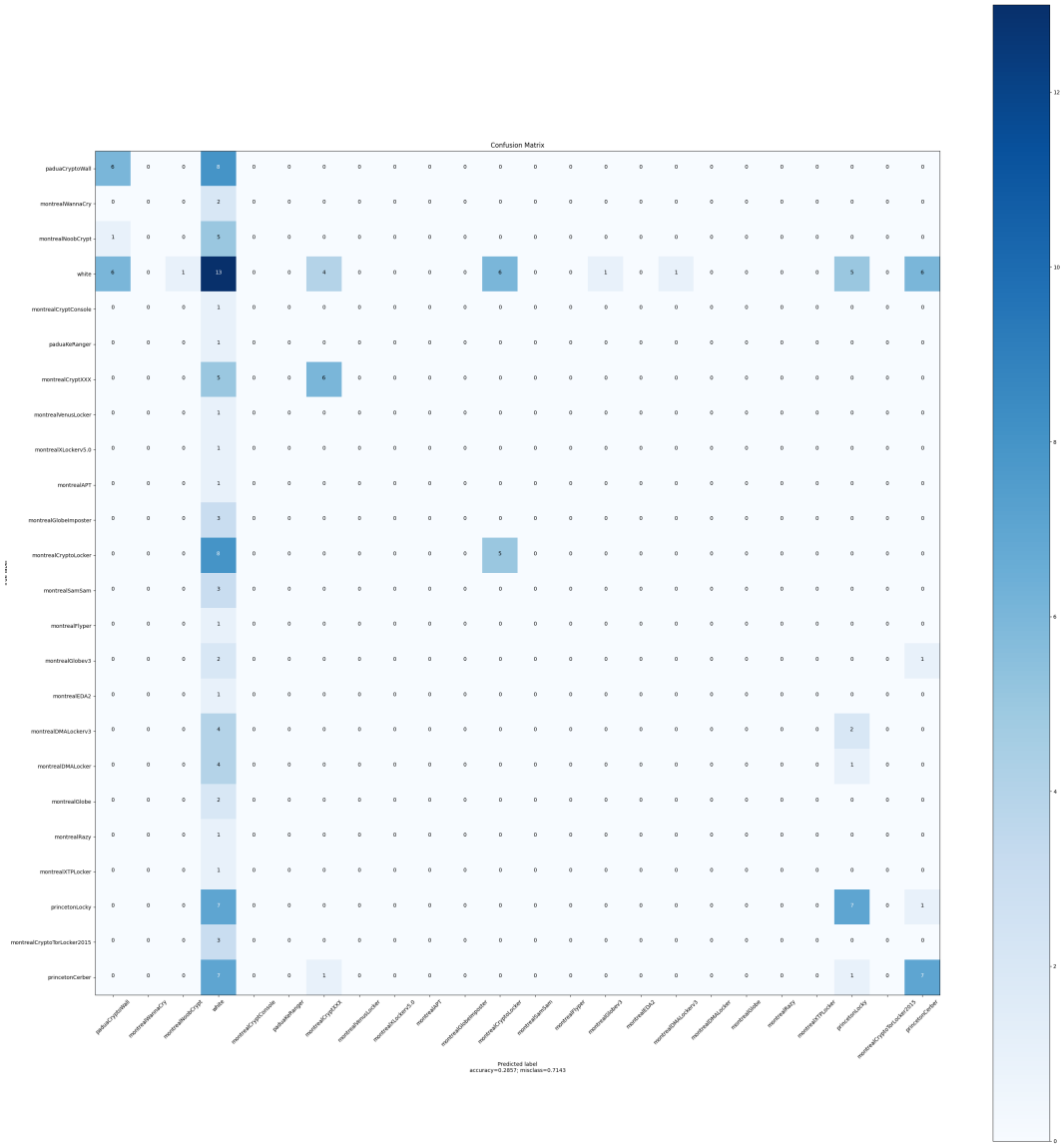

Figure 5: Results - Random Forests case3

2. Oblique decision tree For oblique decision tree we have only regression so I changed all the columns into floating values (already numerical), and then train a regression task. I round-of the prediction values accordingly to the nearest values.

Accuracy: 0.8235464870436952

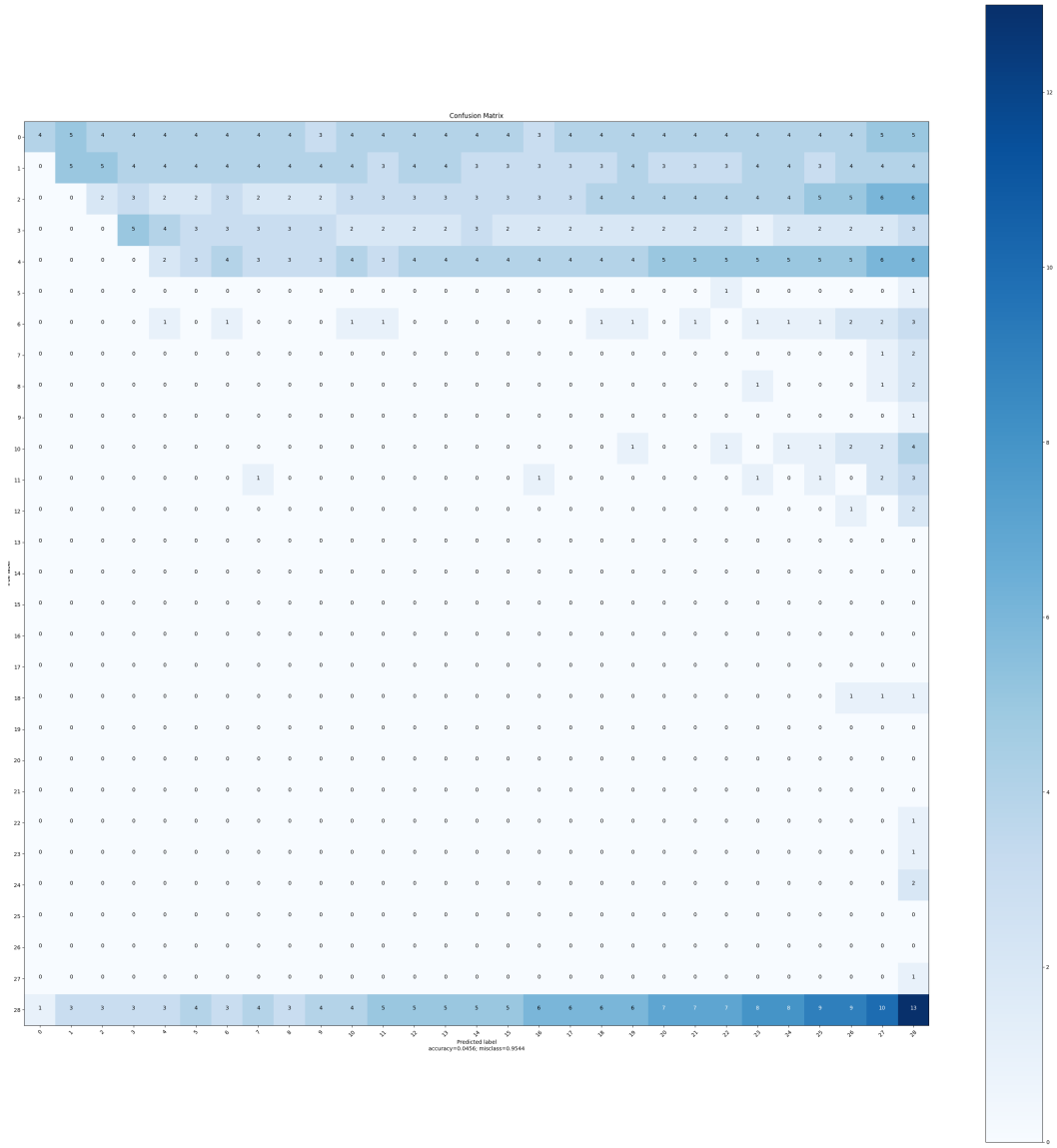

Figure 6: Results - Oblique decision tree

3. TabNet Here first I converted the final prediction column into numbers. Conberted all other columns into "float32". And then did the classification.

Accuracy: 0.9857973538678854

```
cpu
Device used : cpu
epoch 0  | loss: 0.11717 | val_0_accuracy: 0.9858  |  0:01:22s
epoch 1  | loss: 0.07086 | val_0_accuracy: 0.9858  |  0:02:34s
epoch 2  | loss: 0.06917 | val_0_accuracy: 0.98579 |  0:03:56s
epoch 3  | loss: 0.0681  | val_0_accuracy: 0.9858  |  0:05:17s
epoch 4  | loss: 0.0675  | val_0_accuracy: 0.9858  |  0:06:39s
Stop training because you reached max_epochs = 5 with best_epoch = 0 and best_val_0_accuracy = 0.9858
Best weights from best epoch are automatically used!
```

Figure 7: Per epoch loss and accuracy

```
train_X
```

```
array([[2.0160000e+03, 8.5000000e+01, 4.0000000e+00, ..., 0.0000000e+00,
        2.0000000e+00, 1.2500000e+08],
       [2.0160000e+03, 1.4800000e+02, 5.8000000e+01, ..., 0.0000000e+00,
        2.0000000e+00, 1.2500000e+08],
       [2.0160000e+03, 2.8200000e+02, 1.4400000e+02, ..., 0.0000000e+00,
        2.0000000e+00, 1.0000000e+08],
       ...,
       [2.0120000e+03, 2.2200000e+02, 1.4400000e+02, ..., 1.4070000e+03,
        2.0000000e+00, 1.0014240e+08],
       [2.0140000e+03, 1.1600000e+02, 4.0000000e+00, ..., 0.0000000e+00,
        2.0000000e+00, 5.0000000e+07],
       [2.0180000e+03, 9.2000000e+01, 1.4400000e+02, ..., 3.9340000e+03,
        2.0000000e+00, 2.5829937e+09]], dtype=float32)
```

Figure 8: trainX

```
train_y
```

```
array([ 0,  0,  0, ..., 28, 28, 28])
```

Figure 9: trainy

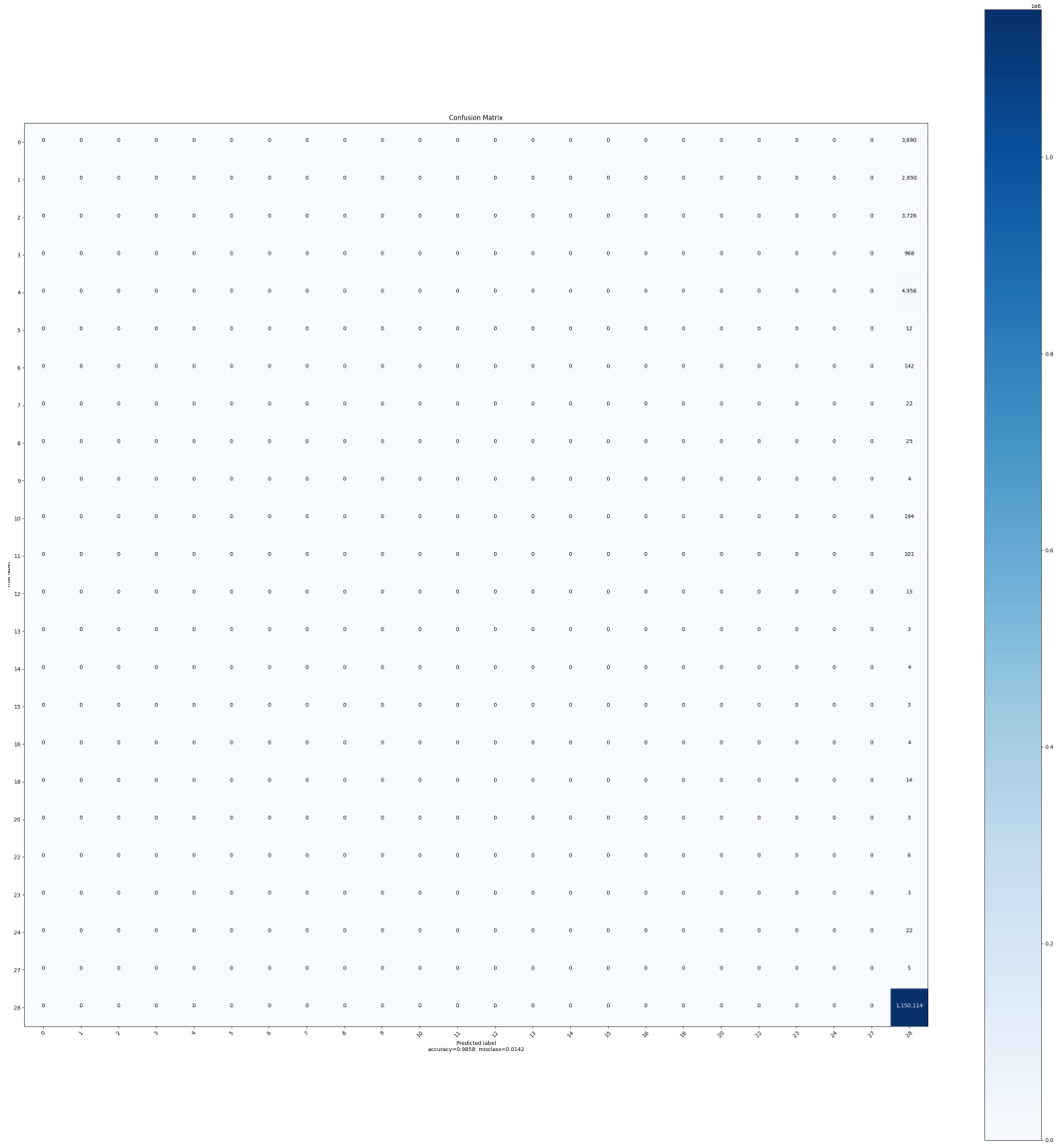

Figure 10: Result - TabNet

2 Dataset 2: Traffic Violations

2.1 Dataset Analysis

Shape of the dataset: (1578154, 43) where we have 1578154 rows and 43 columns in the dataset.
Total classes: 4
Number of missing values and unique values in each column: Figure 11 12

```
Number of missing values in column 1 : 0 , Unique Values : 895102
Number of missing values in column 2 : 0 , Unique Values : 2810
Number of missing values in column 3 : 0 , Unique Values : 1440
Number of missing values in column 4 : 0 , Unique Values : 1
Number of missing values in column 5 : 0 , Unique Values : 9
Number of missing values in column 6 : 9 , Unique Values : 14268
Number of missing values in column 7 : 2 , Unique Values : 214969
Number of missing values in column 8 : 0 , Unique Values : 302145
Number of missing values in column 9 : 0 , Unique Values : 343293
Number of missing values in column 10 : 0 , Unique Values : 2
Number of missing values in column 11 : 0 , Unique Values : 2
Number of missing values in column 12 : 0 , Unique Values : 2
Number of missing values in column 13 : 0 , Unique Values : 2
Number of missing values in column 14 : 0 , Unique Values : 2
Number of missing values in column 15 : 0 , Unique Values : 2
Number of missing values in column 16 : 0 , Unique Values : 2
Number of missing values in column 17 : 0 , Unique Values : 2
Number of missing values in column 18 : 0 , Unique Values : 2
Number of missing values in column 19 : 0 , Unique Values : 2
Number of missing values in column 20 : 600828 , Unique Values : 3
Number of missing values in column 21 : 1509055 , Unique Values : 6
Number of missing values in column 22 : 619994 , Unique Values : 6
Number of missing values in column 23 : 1509055 , Unique Values : 10
Number of missing values in column 24 : 600998 , Unique Values : 752
Number of missing values in column 25 : 1509063 , Unique Values : 5
Number of missing values in column 26 : 1531303 , Unique Values : 10
Number of missing values in column 27 : 59 , Unique Values : 71
Number of missing values in column 28 : 0 , Unique Values : 33
Number of missing values in column 29 : 9647 , Unique Values : 355
Number of missing values in column 30 : 9762 , Unique Values : 3985
```

Figure 11: Dataset2 - missing values and unique values

Since we have huge amount of unique values in this dataset and the values itself is not numeric, we can't create one hot encoding of these values.

Fist: I dropped the columns which have large ammount of missing values as they are of not any useful as seen in the figure, for example column 21, 23 etc. I dropped the column 4 as it contain only one value. Second: I assigned unique id to each variable in the dataset.

```
Number of missing values in column 30 : 9762 , Unique Values : 3985
Number of missing values in column 31 : 9909 , Unique Values : 19463
Number of missing values in column 32 : 18657 , Unique Values : 27
Number of missing values in column 33 : 0 , Unique Values : 4
Number of missing values in column 34 : 0 , Unique Values : 1086
Number of missing values in column 35 : 76873 , Unique Values : 5
Number of missing values in column 36 : 0 , Unique Values : 2
Number of missing values in column 37 : 0 , Unique Values : 6
Number of missing values in column 38 : 0 , Unique Values : 3
Number of missing values in column 39 : 387 , Unique Values : 8088
Number of missing values in column 40 : 11 , Unique Values : 69
Number of missing values in column 41 : 929 , Unique Values : 71
Number of missing values in column 42 : 0 , Unique Values : 19
Number of missing values in column 43 : 0 , Unique Values : 774176
```

Figure 12: Dataset2 - missing values and unique values

2.2 Models

1. Random Forests

We have done same as in the previous dataset (dataset1). We got the accuracy as shown below:
Accuracy: 0.9477586996228197

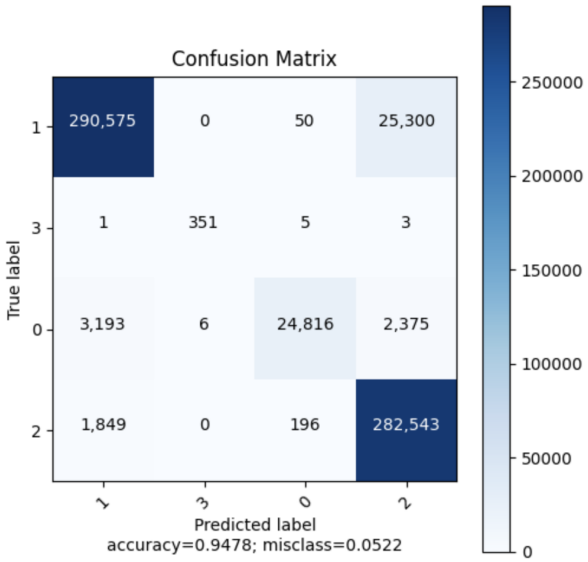

Figure 13: Dataset2 - Random forests

2. Oblique decision tree Oblique decision tree used here is same as previous implementation for dataset1.

Accuracy: 0.8079485095752483

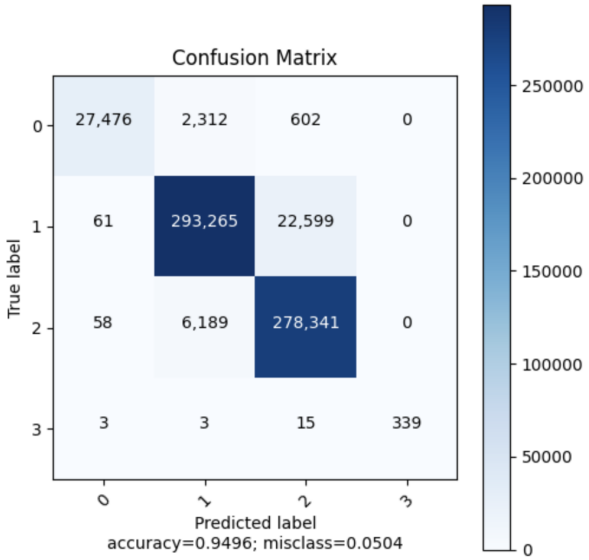

Figure 14: Dataset2 - Oblique decision tree

3. TabNet tabnet used here is same as previous implementation for dataset1.
Accuracy: 0.9405065717458492

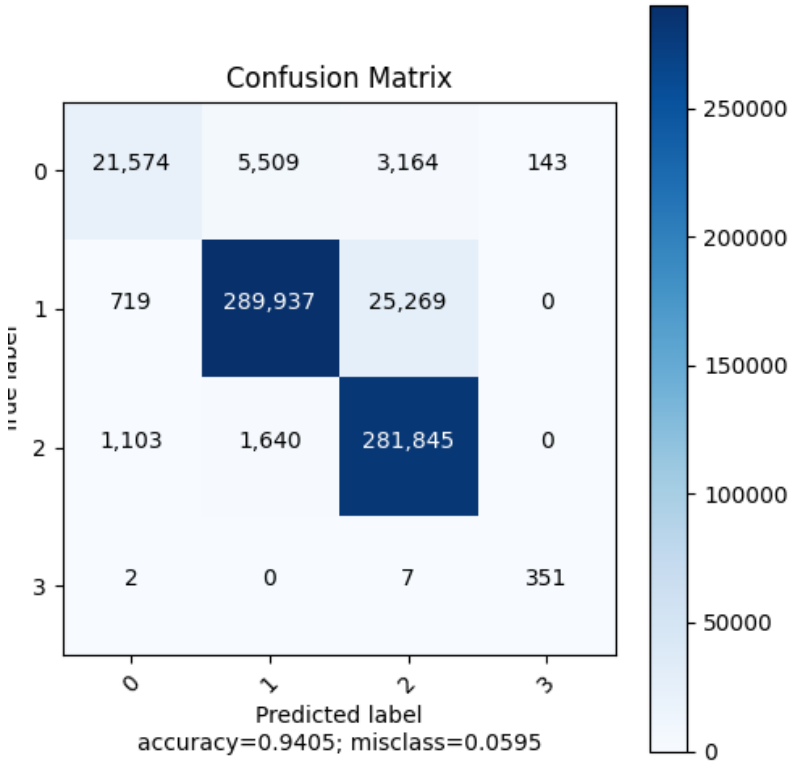

Figure 15: Dataset2 - Tabnet

3 Dataset 3: German Credit data

3.1 Dataset Analysis

Dataset3 is whole numerical (binary classification). The only thing is that "It is worse to class a customer as good when they are bad, than it is to class a customer as bad when they are good."

To avoid this in random forests we can increase the weightage of certain class (that is more important).

```
: dataset

: array([[ 1.,  6.,  4., ...,  0.,  1.,  1.],
        [ 2., 48.,  2., ...,  0.,  1.,  2.],
        [ 4., 12.,  4., ...,  1.,  0.,  1.],
        ...,
        [ 4., 12.,  2., ...,  0.,  1.,  1.],
        [ 1., 45.,  2., ...,  0.,  1.,  2.],
        [ 2., 45.,  4., ...,  0.,  1.,  1.]])
```

Figure 16: Dataset3

3.2 Models

1. Random Forests

No data cleaning is done for this dataset. Except I change the weightage of the classes yes/no.

Accuracy: 0.655

Here I have given class weigh as classweight=1:0.2, 2:0.8

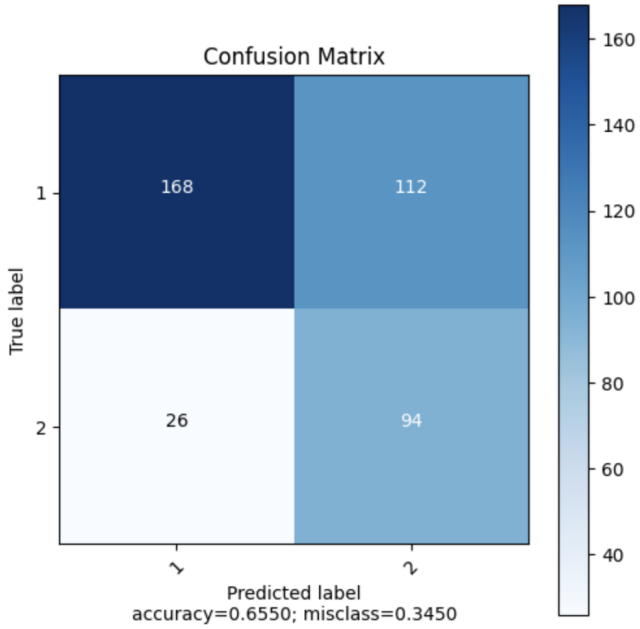

Figure 17: Dataset3 - Random Forest

2. **Oblique decision tree** In this i was not able to set the weight of the classes.
   Accuracy: 0.725

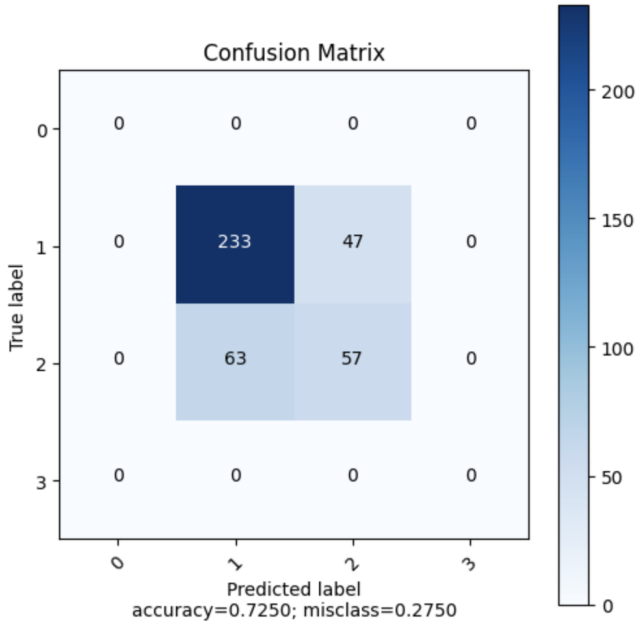

Figure 18: Dataset3 - Oblique

3. TabNet Accuracy: 0.68

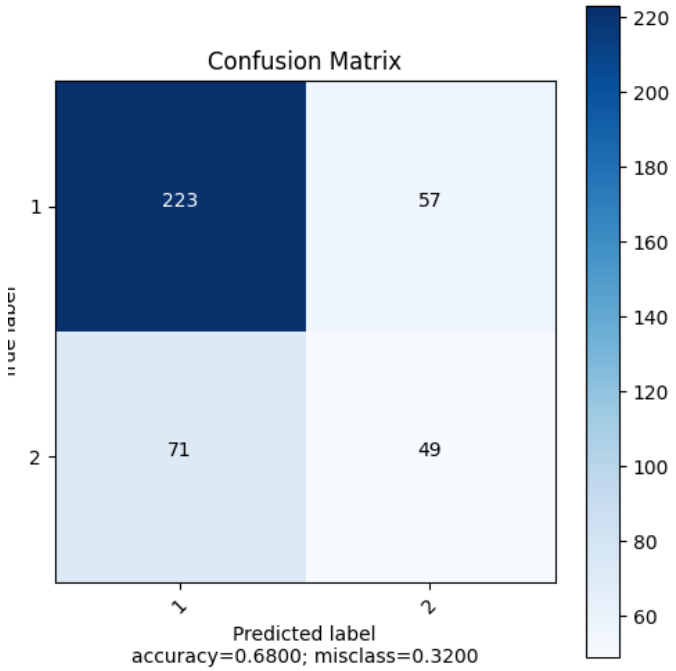

Figure 19: Dataset3 - Tabnet