Credit Card Fraud Detection Using Daimensions

In this notebook, we will be using a dataset from Worldline and the Machine Learning Group (http://mlg.ulb.ac.be) of ULB (Université Libre de Bruxelles). This dataset has 30 attribute columns to describe a credit card transaction and one target column to determine if it is a fraudulant transaction. The dataset can be found on Kaggle: https://www.kaggle.com/mlg-ulb/creditcardfraud

Below is a sample of the data. All of the features that start with "V" are the result of a PCA transformation on the sensitive data relevant to the transaction. We are trying to predict the "Class" column, and it has the labels "1" for fraudulent transactions and "0" for regular ones. Also, the dataset is highly unbalanced, with only 0.17% of the transactions being fraudulent.

In [6]:

```
! head creditcard.csv
# file needs to be unzipped
```

```
"Time","V1","V2","V3","V4","V5","V6","V7","V8","V9","V10","V11","V12","V13","V14","V15","
V16","V17","V18","V19","V20","V21","V22","V23","V24","V25","V26","V27","V28","Amount","Cl
ass"
0,-1.3598071336738,-0.0727811733098497,2.53634673796914,1.37815522427443,-0.3383207699425
18,0.462387777762292,0.239598554061257,0.0986979012610507,0.363786969611213,0.09079417197
89316,-0.551599533260813,-0.617800855762348,-0.991389847235408,-0.311169353699879,1.46817
697209427,-0.470400525259478,0.207971241929242,0.0257905801985591,0.403992960255733,0.251
412098239705,-0.018306777944153,0.277837575558899,-0.110473910188767,0.0669280749146731,0
.128539358273528,-0.189114843888824,0.133558376740387,-0.0210530534538215,149.62,"0"
0,1.19185711131486,0.26615071205963,0.16648011335321,0.448154078460911,0.060017649282243
,-0.0823608088155687,-0.0788029833323113,0.0851016549148104,-0.255425128109186,-0.1669744
14004614,1.61272666105479,1.06523531137287,0.48909501589608,-0.143772296441519,0.63555809
3258208,0.463917041022171,-0.114804663102346,-0.183361270123994,-0.145783041325259,-0.069
0831352230203,-0.225775248033138,-0.638671952771851,0.101288021253234,-0.339846475529127,
0.167170404418143,0.125894532368176,-0.00898309914322813,0.0147241691924927,2.69,"0"
1,-1.35835406159823,-1.34016307473609,1.77320934263119,0.379779593034328,-0.503198133318
193,1.80049938079263,0.791460956450422,0.247675786588991,-1.51465432260583,0.207642865216
696,0.624501459424895,0.066083685268831,0.717292731410831,-0.165945922763554,2.34586494901
581,-2.89008319444231,1.10996937869599,-0.121359313195888,-2.26185709530414,0.52497972522
4404,0.247998153469754,0.771679401917229,0.909412262347719,-0.689280956490685,-0.32764183
3735251,-0.139096571514147,-0.0553527940384261,-0.0597518405929204,378.66,"0"
1,-0.966271711572087,-0.185226008082898,1.79299333957872,-0.863291275036453,-0.0103088796
030823,1.24720316752486,0.23760893977178,0.377435874652262,-1.38702406270197,-0.054951922
4713749,-0.226487263835401,0.178228225877303,0.507756869957169,-0.28792374549456,-0.63141
8117709045,-1.0596472454325,-0.684092786345479,1.96577500349538,-1.2326219700892,-0.20803
7781160366,-0.108300452035545,0.00527359678253453,-0.190320518742841,-1.17557533186321,0.
647376034602038,-0.221928844458407,0.0627228487293033,0.0614576285006353,123.5,"0"
2,-1.15823309349523,0.877736754848451,1.548717846511,0.403033933955121,-0.407193377311653
,0.0959214624684256,0.592940745385545,-0.270532677192282,0.817739308235294,0.753074431976
354,-0.822842877946363,0.53819555014995,1.3458515932154,-1.11966983471731,0.1751211300089
94,-0.451449182813529,-0.237033239362776,-0.0381947870352842,0.803486924960175,0.40854236
0392758,-0.00943069713232919,0.79827849458971,-0.137458079619063,0.141266983824769,-0.206
009587619756,0.502292224181569,0.219422229513348,0.215153147499206,69.99,"0"
2,-0.425965884412454,0.960523044882985,1.14110934232219,-0.168252079760302,0.420986880772
19,-0.0297275516639742,0.476200948720027,0.260314333074874,-0.56867137571251,-0.371407196
834471,1.34126198001957,0.359893837038039,-0.358090652573631,-0.137133700217612,0.5176168
06555742,0.401725895589603,-0.0581328233640131,0.068653149442543 2,-0.0331937877876282,0.0
849676720682049,-0.208253514656728,-0.559824796253248,-0.0263976679795373,-0.371426583174
346,-0.232793816737034,0.105914779097957,0.253844224739337,0.0810802569229443,3.67,"0"
4,1.22965763450793,0.141003507049326,0.0453707735899449,1.20261273673594,0.19188098859764
5,0.272708122899098,-0.00515900288250983,0.0812129398830894,0.464959994783886,-0.09925432
11289237,-1.41690724314928,-0.153825826253651,-0.75106271556262,0.16737196252175,0.050143
5942254188,-0.443586797916727,0.00282051247234708,-0.61198733994012,-0.0455750446637976,-
0.21963255278686,-0.167716265815783,-0.270709726172363,-0.154103786809305,-0.780055415004
671,0.75013693580659,-0.257236845917139,0.0345074297438413,0.00516776890624916,4.99,"0"
7,-0.644269442348146,1.41796354547385,1.0743803763556,-0.492199018495015,0.94893409476415
7,0.428118462833089,1.12063135838353,-3.80786423873589,0.615374730667027,1.24937617815176
,-0.619467796121913,0.291474353088705,1.75796421396042,-1.32386521970526,0.68613250439438
```

```
3,-0.0761269994382006,-1.2221273453247,-0.358221569869078,0.324504731321494,-0.1567418524
88285,1.94346533978412,-1.01545470979971,0.057503529867291,-0.649709005559993,-0.41526656
6234811,-0.0516342969262494,-1.20692108094258,-1.08533918832377,40.8,"0"
7,-0.89428608220282,0.286157196276544,-0.113192212729871,-0.271526130088604,2.66959865959
86,3.72181806112751,0.370145127676916,0.851084443200905,-0.392047586798604,-0.41043043284
8439,-0.705116586646536,-0.110452261733098,-0.286253632470583,0.0743553603016731,-0.32878
3050303565,-0.210077268148783,-0.499767968800267,0.118764861004217,0.57032816746536,0.052
7356691149697,-0.0734251001059225,-0.268091632235551,-0.204232669947878,1.0115918018785,0
.373204680146282,-0.384157307702294,0.0117473564581996,0.14240432992147,93.2,"0"
```

For this dataset, our objective is to understand which attributes are most important, and then be able to build a model that detects credit card fraud. Daimension's has an option to enable attribute ranking, which is extremely helpful in finding the features that are most correlated with the target class.

1. Get Measurements

Before we build the predictor for the dataset, it would be wise to measure it. This allows us to find the most optimal model, without even having to build one. For more information about how to use Daimensions and why we want to measure our data beforehand, check out the Titanic notebook.

In [7]:

```
! btc creditcard.csv -measureonly
```

```
WARNING: Could not detect a GPU. Neural Network generation will be slow.

Brainome Daimensions(tm) 0.99 Copyright (c) 2019 - 2021 by Brainome, Inc. All Rights Rese
rved.
Licensed to:                 Alexander Makhratchev  (Evaluation)
Expiration Date:             2021-04-30   56 days left
Number of Threads:           1
Maximum File Size:           30 GB
Maximum Instances:           unlimited
Maximum Attributes:          unlimited
Maximum Classes:             unlimited
Connected to:                daimensions.brainome.ai  (local execution)



Command:
    btc creditcard.csv -measureonly

Start Time:                 03/05/2021, 02:28


Data:
    Input:                        creditcard.csv
    Target Column:                Class
    Number of instances:          284807
    Number of attributes:         30
    Number of classes:            2
    Class Balance:                0: 99.83%, 1: 0.17%

Learnability:
    Best guess accuracy:           99.83%
    Data Sufficiency:             Maybe enough data to generalize. [yellow]

Capacity Progression:             at [ 5%, 10%, 20%, 40%, 80%, 100% ]
    Optimal Machine Learner:            7,    8,    9,    9,   10,   10


Estimated Memory Equivalent Capacity for...
    Decision Tree:                    938 parameters
    Neural Networks:                    1 parameters
    Random Forest:                     65 parameters

Risk that model needs to overfit for 100% accuracy using...
    Decision Tree:                 95.49%
    Neural Networks:                9.09%
    Random Forest:                27.20%

Expected Generalization using...
    Decision Tree:                  5.57 bits/bit
    Neural Network:             142182.00 bits/bit
    Random Forest:               4381.65 bits/bit


Recommendations:

    Warning: Data has high information density. Expect varying results and increase --eff
ort.



Time to Build Estimates:
    Decision Tree:             less than a minute
    Neural Network:              13 minutes
```

2. Neural Network with -O

From the daimensions measurements, we can see that the best model for this dataset would be a neural network. It has the highest generalization and lowest memory equivalent capacity. However, the neural network has a much higher risk for overfit. Because the dataset is so unbalanced, we will be using the -O command line option in order optimize the true positive rate (TPR). After the -O, we specify the label to focus on, and in our case it is the fradulent charges "1".

In [9]:

```
! btc creditcard.csv -f NN -O 1 --yes
```

```
WARNING: Could not detect a GPU. Neural Network generation will be slow.

Brainome Daimensions(tm) 0.99 Copyright (c) 2019 - 2021 by Brainome, Inc. All Rights Rese
rved.
Licensed to:              Alexander Makhratchev  (Evaluation)
Expiration Date:          2021-04-30   56 days left
Number of Threads:        1
Maximum File Size:        30 GB
Maximum Instances:        unlimited
Maximum Attributes:       unlimited
Maximum Classes:          unlimited
Connected to:             daimensions.brainome.ai  (local execution)



Command:
    btc creditcard.csv -f NN -O 1 --yes

Start Time:                 03/05/2021, 03:37


Data:
    Input:                      creditcard.csv
    Target Column:              Class
    Number of instances:        284807
    Number of attributes:       30
    Number of classes:          2
    Class Balance:              0: 99.83%, 1: 0.17%

Learnability:
    Best guess accuracy:          99.83%
    Data Sufficiency:            Maybe enough data to generalize. [yellow]

Capacity Progression:            at [ 5%, 10%, 20%, 40%, 80%, 100% ]
    Optimal Machine Learner:           7,    8,    9,    9,   10,   10
```

```
Estimated Memory Equivalent Capacity for...
    Decision Tree:                   938 parameters
    Neural Networks:                   1 parameters
    Random Forest:                    65 parameters

Risk that model needs to overfit for 100% accuracy using...
    Decision Tree:                95.49%
    Neural Networks:               9.09%
    Random Forest:                27.20%

Expected Generalization using...
    Decision Tree:                  5.57 bits/bit
    Neural Network:             142182.00 bits/bit
    Random Forest:               4381.65 bits/bit


Recommendations:

    Warning: Data has high information density. Expect varying results and increase --eff
ort.
    Note: Machine learner type NN given by user.


Time to Build Estimates:

    Neural Network:              18 minutes


System Meter:                          a.py
    Classifier Type:                   Neural Network
    System Type:                       Binary classifier
    Training/Validation Split:           50% : 50%
    Accuracy:
        Best-guess accuracy:               99.82%
        Training accuracy:                  1.44% (2064/142403 correct)
        Validation Accuracy:                0.80% (1152/142404 correct)
        Overall Model Accuracy:             1.12% (3216/284807 correct)
        Improvement over best guess:      -98.70% of possible   0.18%

    Model Capacity (MEC):                       1 bit
    Generalization Ratio:                   37.46 bits/bit
    Model Efficiency:                      -98.69 /parameter

    Training Confusion Matrix (count):
                   0 |    1821  140339
                   1 |       0     243

    Validation Confusion Matrix (count):
                   0 |     903  141252
                   1 |       0     249

    Full Confusion Matrix (count):
                   0 |    2724  281591
                   1 |       0     492

    Accuracy by Class:
               class |      TP      FP      TN      FN     TPR      TNR      PPV      NP
V       F1       TS
                   0 |    2724  281591    492        0  100.00%  100.00%    0.96%  100.0
0%    1.90%    0.96%
                   1 |     492       0    2724  281591    0.17%    0.96%  100.00%    0.9
6%    0.35%    0.17%

End Time:
Runtime Duration:
```

The neural network had a very poor overall accuracy on the validation set. However, the true positive rate is 100%, signifying that every transaction that was fraudulent was identified.

Now we will re-run the previous command, but this time we will add the -e command in order to increase the training effort of the model.

In [4]:

```
! btc creditcard.csv -f NN -O 1 --yes -e 5
```

```
WARNING: Could not detect a GPU. Neural Network generation will be slow.

Brainome Table Compiler 0.99
Copyright (c) 2019-2021 Brainome, Inc. All Rights Reserved.
Licensed to:                 Alexander Makhratchev  (Evaluation)
Expiration Date:             2021-04-30   55 days left
Maximum File Size:           30 GB
Maximum Instances:           unlimited
Maximum Attributes:          unlimited
Maximum Classes:             unlimited
Connected to:                daimensions.brainome.ai  (local execution)

Command:
    btc creditcard.csv -f NN -O 1 --yes -e 5

Start Time:                 03/06/2021, 00:49 UTC

Data:
    Input:                      creditcard.csv
    Target Column:              Class
    Number of instances:     284807
    Number of attributes:        30
    Number of classes:            2
    Class Balance:                0: 99.83%, 1: 0.17%

Learnability:
    Best guess accuracy:          99.83%
    Data Sufficiency:             Maybe enough data to generalize. [yellow]

Capacity Progression:             at [ 5%, 10%, 20%, 40%, 80%, 100% ]
    Ideal Machine Learner:              7,    8,    9,    9,   10,   10

Estimated Memory Equivalent Capacity for...
    Decision Tree:               938 parameters
    Neural Networks:               1 parameters
    Random Forest:                65 parameters

Risk that model needs to overfit for 100% accuracy using...
    Decision Tree:                95.49%
    Neural Networks:               9.09%
    Random Forest:                27.20%

Expected Generalization using...
    Decision Tree:                 5.57 bits/bit
    Neural Network:           142182.00 bits/bit
    Random Forest:              4381.65 bits/bit


Recommendations:
    Warning: Data has high information density. Expect varying results and increase --eff
ort.
    Note: Machine learner type NN given by user.

Time to Build Estimates:
    Neural Network:                  13 minutes


System Meter:                     a.py
    Classifier Type:              Neural Network
    System Type:                  Binary classifier
    Training/Validation Split:  50% : 50%
    Accuracy:
      Best-guess accuracy:        99.82%
      Training accuracy:           1.44% (2064/142403 correct)
      Validation Accuracy:         0.80% (1152/142404 correct)
      Overall Model Accuracy:      1.12% (3216/284807 correct)

    Architecture Capacity (MEC):   1    bits
    Generalization Ratio:         37.46 bits/bit
    Architecture Efficiency:            /parameter

    Training Confusion Matrix (count):
                   0 |    1821  140339
                   1 |       0     243

    Validation Confusion Matrix (count):
                   0 |     903  141252
                   1 |       0     249

    Full Confusion Matrix (count):
                   0 |    2724  281591
                   1 |       0     492

    Accuracy by Class:
              class |      TP      FP       TN       FN      TPR       TNR       PPV       NP
V       F1       TS
                  0 |    2724  281591      492        0  100.00%   100.00%     0.96%   100.0
0%    1.90%    0.96%
                  1 |     492       0     2724   281591     0.17%     0.96%   100.00%     0.9
6%    0.35%    0.17%


End Time:           03/06/2021, 01:24 UTC
Runtime Duration:   34m 29s
```

3. Decision Tree with -O

We can also try to a decision tree for the dataset by simply replacing the NN command with DT.

In [10]:

```
! btc creditcard.csv -rank -f DT -O 1 --yes
```

```
WARNING: Could not detect a GPU. Neural Network generation will be slow.

Brainome Daimensions(tm) 0.99 Copyright (c) 2019 - 2021 by Brainome, Inc. All Rights Rese
rved.
Licensed to:                 Alexander Makhratchev  (Evaluation)
Expiration Date:             2021-04-30   56 days left
Number of Threads:           1
Maximum File Size:           30 GB
Maximum Instances:           unlimited
Maximum Attributes:          unlimited
Maximum Classes:             unlimited
Connected to:                daimensions.brainome.ai  (local execution)



Command:
    btc creditcard.csv -rank -f DT -O 1 --yes

Start Time:                    03/05/2021, 04:37

Attribute Ranking:
    Important columns:           V17, V14, V10, V9, V25
    Overfit risk:                    0.0%
    Ignoring columns:            Time, V1, V2, V3, V4, V5, V6, V7, V8, V11, V12, V13, V15,
V16, V18, V19, V20, V21, V22, V23, V24, V26, V27, V28, Amount



Data:
    Input:                       creditcard.csv
    Target Column:               Class
    Number of instances:         284807
    Number of attributes:       5
    Number of classes:          2
    Class Balance:              0: 99.83%, 1: 0.17%

Learnability:
    Best guess accuracy:          99.83%
    Data Sufficiency:            Not enough data to generalize. [red]

Capacity Progression:            at [ 5%, 10%, 20%, 40%, 80%, 100% ]
    Optimal Machine Learner:           5,   6,   7,   8,   8,   9


Estimated Memory Equivalent Capacity for...
    Decision Tree:                   289 parameters
    Neural Networks:                   1 parameters
    Random Forest:                    63 parameters

Risk that model needs to overfit for 100% accuracy using...
    Decision Tree:                29.42%
    Neural Networks:               9.08%
    Random Forest:                24.61%

Expected Generalization using...
    Decision Tree:                 18.08 bits/bit
    Neural Network:             142315.00 bits/bit
    Random Forest:               4520.75 bits/bit


Recommendations:

    Warning: Data has high information density. Expect varying results and increase --eff
ort.
    Note: Machine learner type DT given by user.


Time to Build Estimates:
    Decision Tree:             less than a minute


System Meter:                          a.py
    Classifier Type:                   Decision Tree
    System Type:                       Binary classifier
    Training/Validation Split:           50% : 50%
    Accuracy:
        Best-guess accuracy:               99.82%
        Training accuracy:                100.00% (142403/142403 correct)
        Validation Accuracy:               99.90% (142264/142404 correct)
        Overall Model Accuracy:            99.95% (284667/284807 correct)
        Improvement over best guess:        0.13% of possible   0.18%

    Model Capacity (MEC):                     149 bits
    Generalization Ratio:                   17.35 bits/bit
    Model Efficiency:                        0.00 /parameter
    Generalization Index:                    0.01
    Percent of Data Memorized:            17897.46%

    Training Confusion Matrix (count):
                   0 |  142160       0
                   1 |       0     243

    Validation Confusion Matrix (count):
                   0 |  142074      81
                   1 |      59     190

    Full Confusion Matrix (count):
                   0 |  284234      81
                   1 |      59     433

    Accuracy by Class:
               class |       TP      FP      TN      FN    TPR      TNR       PPV      NP
V       F1       TS
                   0 |  284234      81     433      59   99.98%   88.01%  99.97%     88.0
```

```
1%   99.98%   99.95%
                  1 |      433       59  284234       81    84.24%    99.97%    88.01%    99.9
7%   86.08%   75.57%

End Time:
Runtime Duration:
```

The decion tree was able to predict most of the fraudelent charges with 99.98% accuracy. The use of attribute ranking significantly reduces the noise in a dataset and improves accuracy.

4. Neural Netork with -balance

Now we will try the -balance command which optimizes the true positive rate for each class, instead of a specific one.

In [11]:

```
! btc creditcard.csv -f NN -balance --yes
```

```
WARNING: Could not detect a GPU. Neural Network generation will be slow.

Brainome Daimensions(tm) 0.99 Copyright (c) 2019 - 2021 by Brainome, Inc. All Rights Rese
rved.
Licensed to:                 Alexander Makhratchev  (Evaluation)
Expiration Date:             2021-04-30   56 days left
Number of Threads:           1
Maximum File Size:           30 GB
Maximum Instances:           unlimited
Maximum Attributes:          unlimited
Maximum Classes:             unlimited
Connected to:                daimensions.brainome.ai  (local execution)



Command:
    btc creditcard.csv -f NN -balance --yes

Start Time:                 03/05/2021, 04:57


Data:
    Input:                      creditcard.csv
    Target Column:              Class
    Number of instances:        284807
    Number of attributes:       30
    Number of classes:          2
    Class Balance:              0: 99.83%, 1: 0.17%

Learnability:
    Best guess accuracy:          99.83%
    Data Sufficiency:           Maybe enough data to generalize. [yellow]

Capacity Progression:           at [ 5%, 10%, 20%, 40%, 80%, 100% ]
    Optimal Machine Learner:           7,   8,   9,   9,  10,  10


Estimated Memory Equivalent Capacity for...
    Decision Tree:                        938 parameters
    Neural Networks:                        1 parameters
    Random Forest:                         65 parameters

Risk that model needs to overfit for 100% accuracy using...
    Decision Tree:                95.49%
    Neural Networks:               9.09%
    Random Forest:                27.20%

Expected Generalization using...
```

```
    Decision Tree:                    5.57 bits/bit
    Neural Network:               142182.00 bits/bit
    Random Forest:                  4381.65 bits/bit


Recommendations:

    Warning: Data has high information density. Expect varying results and increase --eff
ort.
    Note: Machine learner type NN given by user.


Time to Build Estimates:

    Neural Network:               1:30:50


System Meter:                           a.py
    Classifier Type:                    Neural Network
    System Type:                        Binary classifier
    Training/Validation Split:            50% : 50%
    Accuracy:
        Best-guess accuracy:                99.82%
        Training accuracy:                  94.57% (134683/142403 correct)
        Validation Accuracy:                96.38% (137259/142404 correct)
        Overall Model Accuracy:             95.48% (271942/284807 correct)
        Improvement over best guess:        -4.34% of possible   0.18%

    Model Capacity (MEC):                    30389 bits
    Generalization Ratio:                     0.08 bits/bit
    Model Efficiency:                         0.00 /parameter
    Generalization Index:                     0.02
    Percent of Data Memorized:             6614.64%

    Training Confusion Matrix (count):
                   0 |  134556    7604
                   1 |     116     127

    Validation Confusion Matrix (count):
                   0 |  137182    4973
                   1 |     172      77

    Full Confusion Matrix (count):
                   0 |  271738   12577
                   1 |     288     204

    Accuracy by Class:
               class |      TP       FP       TN       FN      TPR       TNR       PPV       NP
V       F1        TS
                   0 |  271738   12577      204      288   99.89%    41.46%    95.58%    41.4
6%   97.69%   95.48%
                   1 |     204      288   271738    12577    1.60%    95.58%    41.46%    95.5
8%    3.07%    1.56%

End Time:
Runtime Duration:
```

Unfortunately, our model performs slightly worse than best guess on the dataset, but the true positive rate is 99.89%.

Now we will re run the following command, but will use the -e command to increase the amount of effort in training the model.

In [5]:

```
! btc creditcard.csv -f NN -balance --yes -e 5
```

```
WARNING: Could not detect a GPU. Neural Network generation will be slow.
```

```
Brainome Table Compiler 0.99
Copyright (c) 2019-2021 Brainome, Inc. All Rights Reserved.
Licensed to:                 Alexander Makhratchev  (Evaluation)
Expiration Date:             2021-04-30   55 days left
Maximum File Size:           30 GB
Maximum Instances:           unlimited
Maximum Attributes:          unlimited
Maximum Classes:             unlimited
Connected to:                daimensions.brainome.ai  (local execution)

Command:
    btc creditcard.csv -f NN -balance --yes -e 5

Start Time:                 03/06/2021, 01:24 UTC

Data:
    Input:                      creditcard.csv
    Target Column:              Class
    Number of instances:     284807
    Number of attributes:        30
    Number of classes:            2
    Class Balance:                0: 99.83%, 1: 0.17%

Learnability:
    Best guess accuracy:          99.83%
    Data Sufficiency:             Maybe enough data to generalize. [yellow]

Capacity Progression:             at [ 5%, 10%, 20%, 40%, 80%, 100% ]
    Ideal Machine Learner:              7,   8,   9,   9,  10,  10

Estimated Memory Equivalent Capacity for...
    Decision Tree:               938 parameters
    Neural Networks:               1 parameters
    Random Forest:                65 parameters

Risk that model needs to overfit for 100% accuracy using...
    Decision Tree:                95.49%
    Neural Networks:               9.09%
    Random Forest:                27.20%

Expected Generalization using...
    Decision Tree:                 5.57 bits/bit
    Neural Network:           142182.00 bits/bit
    Random Forest:              4381.65 bits/bit


Recommendations:
    Warning: Data has high information density. Expect varying results and increase --eff
ort.
    Note: Machine learner type NN given by user.

Time to Build Estimates:
    Neural Network:                 50 minutes


System Meter:                     a.py
    Classifier Type:              Neural Network
    System Type:                  Binary classifier
    Training/Validation Split:  50% : 50%
    Accuracy:
      Best-guess accuracy:        99.82%
      Training accuracy:           0.17% (250/142403 correct)
      Validation Accuracy:         0.16% (242/142404 correct)
      Overall Model Accuracy:      0.17% (492/284807 correct)

    Architecture Capacity (MEC):   1    bits
    Generalization Ratio:          4.65 bits/bit
    Architecture Efficiency:            /parameter
    Generalization Index:          0.87
    Percent of Data Memorized:   114.37%

    Training Confusion Matrix (count):
```

```
                   0 |       0  142153
                   1 |       0     250

    Validation Confusion Matrix (count):
                   0 |       0  142162
                   1 |       0     242

    Full Confusion Matrix (count):
                   0 |       0  284315
                   1 |       0     492

    Accuracy by Class:
               class |      TP       FP       TN       FN      TPR      TNR      PPV      NP
V       F1       TS
                   0 |       0  284315      492        0     nan%  100.00%    0.00%  100.0
0%    0.00%    0.00%
                   1 |     492        0        0   284315    0.17%    0.00%  100.00%    0.0
0%    0.34%    0.17%


End Time:           03/06/2021, 20:07 UTC
Runtime Duration:   18h 43m 37s
```

From the results, it looks like our model did not perform well. The validation accuracy was very low, because the model simply guessed all of the charges are fraudulent.

5. Random Forest

In the newest version of the Brainome Table Compiler, the random forest model is included. We can run it on the dataset and increase the effort level to improve the accuracy.

In [13]:

```
! btc creditcard.csv -f RF --yes -e 5
```

```
WARNING: Could not detect a GPU. Neural Network generation will be slow.

Brainome Table Compiler 0.99
Copyright (c) 2019-2021 Brainome, Inc. All Rights Reserved.
Licensed to:                 Alexander Makhratchev  (Evaluation)
Expiration Date:             2021-04-30   53 days left
Maximum File Size:           30 GB
Maximum Instances:           unlimited
Maximum Attributes:          unlimited
Maximum Classes:             unlimited
Connected to:                daimensions.brainome.ai  (local execution)

Command:
    btc creditcard.csv -f RF --yes -e 5

Start Time:                 03/08/2021, 21:38 UTC

Data:
    Input:                      creditcard.csv
    Target Column:              Class
    Number of instances:     284807
    Number of attributes:        30
    Number of classes:            2
    Class Balance:                0: 99.83%, 1: 0.17%

Learnability:
    Best guess accuracy:          99.83%
    Data Sufficiency:             Maybe enough data to generalize. [yellow]

Capacity Progression:             at [ 5%, 10%, 20%, 40%, 80%, 100% ]
    Ideal Machine Learner:              7,   8,   9,   9,  10,   10

Estimated Memory Equivalent Capacity for...
    Decision Tree:               938 parameters
    Neural Networks:               1 parameters
    Random Forest:              65 parameters

Risk that model needs to overfit for 100% accuracy using...
    Decision Tree:              95.49%
    Neural Networks:             9.09%
    Random Forest:              27.20%

Expected Generalization using...
    Decision Tree:               5.57 bits/bit
    Neural Network:         142182.00 bits/bit
    Random Forest:            4381.65 bits/bit


Recommendations:
    Warning: Data has high information density. Expect varying results and increase --eff
ort.
    Note: Machine learner type RF given by user.


System Meter:                       a.py
    Classifier Type:                Random Forest
    System Type:                    Binary classifier
    Training/Validation Split:  50% : 50%
    Accuracy:
      Best-guess accuracy:          99.82%
      Training accuracy:          100.00% (142403/142403 correct)
      Validation Accuracy:          99.95% (142342/142404 correct)
      Overall Model Accuracy:       99.97% (284745/284807 correct)

    Architecture Capacity (MEC):   8    bits
    Generalization Ratio:         323.08 bits/bit
    Architecture Efficiency:             /parameter
    Generalization Index:          60.75
    Percent of Data Memorized:      1.65%

    Training Confusion Matrix (count):
                   0 |  142160       0
                   1 |       0     243

    Validation Confusion Matrix (count):
                   0 |  142142      13
                   1 |      49     200

    Full Confusion Matrix (count):
                   0 |  284302      13
                   1 |      49     443

    Accuracy by Class:
                 class |      TP       FP       TN       FN      TPR       TNR       PPV       NP
V        F1        TS
                   0 |  284302      13      443      49   99.98%    90.04%  100.00%    90.0
4%   99.99%   99.98%
                   1 |     443      49   284302      13   97.15%   100.00%    90.04%   100.0
0%   93.46%   87.72%


End Time:            03/08/2021, 22:17 UTC
Runtime Duration:    39m 11s
```

The Random Forest model did better than best guess on the validation data. Additionally, the True Positive Rate is almost near 100%, which signifies that a majority of the fraudulent transactions were detected.

6. Random Forest with -O and -rank

We can run the same command as we did above, but now we will utilize the -O command in order to optimize the True Positive Rate.

In [11]:

```
!btc creditcard.csv -f RF --yes -e 5 -O 1 -rank
```

```
WARNING: Could not detect a GPU. Neural Network generation will be slow.

Brainome Table Compiler 0.99
Copyright (c) 2019-2021 Brainome, Inc. All Rights Reserved.
Licensed to:                 Alexander Makhratchev  (Evaluation)
Expiration Date:             2021-04-30   55 days left
Maximum File Size:           30 GB
Maximum Instances:           unlimited
Maximum Attributes:          unlimited
Maximum Classes:             unlimited
Connected to:                daimensions.brainome.ai  (local execution)

Command:
    btc creditcard.csv -f RF --yes -e 5 -O 1 -rank

Start Time:                 03/06/2021, 21:20 UTC


Attribute Ranking:
    Important columns:          V17, V14, V10, V9, V25
    Overfit risk:                  0.0%
    Ignoring columns:           Time, V1, V2, V3, V4, V5, V6, V7, V8, V11, V12, V13, V15,
V16, V18, V19, V20, V21, V22, V23, V24, V26, V27, V28, Amount

Data:
    Input:                      creditcard.csv
    Target Column:              Class
    Number of instances:     284807
    Number of attributes:         5
    Number of classes:            2
    Class Balance:                0: 99.83%, 1: 0.17%

Learnability:
    Best guess accuracy:          99.83%
    Data Sufficiency:             Not enough data to generalize. [red]

Capacity Progression:             at [ 5%, 10%, 20%, 40%, 80%, 100% ]
    Ideal Machine Learner:              5,    6,    7,    8,    8,    9

Estimated Memory Equivalent Capacity for...
    Decision Tree:               289 parameters
    Neural Networks:               1 parameters
    Random Forest:                63 parameters

Risk that model needs to overfit for 100% accuracy using...
    Decision Tree:                29.42%
    Neural Networks:               9.08%
    Random Forest:                24.61%

Expected Generalization using...
    Decision Tree:                18.08 bits/bit
    Neural Network:            142315.00 bits/bit
    Random Forest:              4520.75 bits/bit


Recommendations:
    Warning: Data has high information density. Expect varying results and increase --eff
ort.
    Note: Machine learner type RF given by user.


System Meter:                     a.py
    Classifier Type:              Random Forest
    System Type:                  Binary classifier
    Training/Validation Split:  50% : 50%
    Accuracy:
      Best-guess accuracy:        99.82%
      Training accuracy:         100.00% (142403/142403 correct)
      Validation Accuracy:        99.94% (142331/142404 correct)
      Overall Model Accuracy:     99.97% (284734/284807 correct)
    Architecture Capacity (MEC):   8    bits
    Generalization Ratio:        323.08 bits/bit
    Architecture Efficiency:            /parameter

    Training Confusion Matrix (count):
                   0 |  142160       0
                   1 |       0     243

    Validation Confusion Matrix (count):
                   0 |  142136      19
                   1 |      54     195

    Full Confusion Matrix (count):
                   0 |  284296      19
                   1 |      54     438

    Accuracy by Class:
               class |      TP       FP       TN       FN      TPR       TNR       PPV       NP
V        F1        TS
                   0 |  284296      19      438       54   99.98%   89.02%   99.99%   89.0
2%   99.99%   99.97%
                   1 |     438      54  284296       19   95.84%   99.99%   89.02%   99.9
9%   92.31%   85.71%


End Time:           03/07/2021, 03:46 UTC
Runtime Duration:   6h 26m 2s
```

The validation score is higher than best guess, and 99.98% of fraudulent transactions were identified. However, only 89.02% of the regular transactions were identified.