USPS Data Prediction Using Daimensions

This dataset is from OpenML who describes the data as, "Normalized handwritten digits, automatically scanned from envelopes by the U.S. Postal Service."

0. Setup

We'll get the csv from the OpenML link and use a pandas dataframe to split it into training and validation data in csv's.

In [2]:

```
# using pandas to get csv as a dataframe and see how it looks
import pandas as pd
from sklearn.model_selection import train_test_split

dataset_url = 'https://www.openml.org/data/get_csv/19329737/usps.csv'
data = pd.read_csv(dataset_url)
data.describe()
```

Out[2]:

	int0	double1	double2	double3	double4	double5	double6	double7	double8
count	9298.000000	9298.000000	9298.000000	9298.000000	9298.000000	9298.000000	9298.000000	9298.000000	9298.000000
mean	4.892020	-0.991800	-0.972226	-0.930421	-0.852805	-0.733673	-0.578239	-0.391187	-0.228260
std	3.001086	0.050814	0.118296	0.195285	0.284053	0.372653	0.435317	0.452878	0.454537
min	1.000000	-1.000000	-1.000000	-1.000000	-1.000000	-1.000000	-1.000000	-1.000000	-1.000000
25%	2.000000	-1.000000	-1.000000	-1.000000	-0.999914	-0.996085	-0.963110	-0.787003	-0.620084
50%	5.000000	-1.000000	-0.999992	-0.999608	-0.991661	-0.932991	-0.747495	-0.447743	-0.138583
75%	7.000000	-0.999969	-0.998444	-0.979572	-0.861493	-0.589829	-0.260331	0.000547	0.143727
max	10.000000	0.000308	0.332928	0.479436	0.523534	0.527370	0.531509	0.531319	0.531368

8 rows × 257 columns

In [3]:

```
# split data into training and testing csv's, y is for the target column (int0)
y = data.int0
X = data.drop('int0', axis=1)
X_train, X_test, y_train, y_test = train_test_split(X, y,test_size=0.2)
pd.concat([X_train, y_train], axis=1).to_csv('usps_train.csv',index=False)
pd.concat([X_test, y_test], axis=1).to_csv('usps_valid.csv',index=False)
```

1. Get Measurements

We always want to measure our data before building our predictor in order to ensure we are building the right model. For more information about how to use Daimensions and why we want to measure our data beforehand, check out the Titanic notebook. Don't forget to use -target int0 because the target column is not on the very right for this dataset.

In [1]:

```
! btc -measureonly usps_train.csv -target int0
```

```
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 -measureonly usps_train.csv -target int0

Start Time:                 03/05/2021, 18:30


Data:
    Input:                      usps_train.csv
    Target Column:              int0
    Number of instances:        7438
    Number of attributes:       256
    Number of classes:          10
    Class Balance:              0: 7.56%, 1: 8.86%, 2: 16.79%, 3: 13.70%, 4: 9.75%, 5: 8.
67%, 6: 9.38%, 7: 9.08%, 8: 8.55%, 9: 7.66%

Learnability:
    Best guess accuracy:          16.79%
    Data Sufficiency:            Maybe enough data to generalize. [yellow]

Capacity Progression:            at [ 5%, 10%, 20%, 40%, 80%, 100% ]
    Optimal Machine Learner:          10,  11,  12,  12,  13,  13


Estimated Memory Equivalent Capacity for...
    Decision Tree:                  5973 parameters
    Neural Networks:                 241 parameters
    Random Forest:                   132 parameters

Risk that model needs to overfit for 100% accuracy using...
    Decision Tree:                90.00%
    Neural Networks:               9.89%
    Random Forest:                 4.11%

Expected Generalization using...
    Decision Tree:                  4.07 bits/bit
    Neural Network:                15.42 bits/bit
    Random Forest:                 56.35 bits/bit


Recommendations:




Time to Build Estimates:
    Decision Tree:             a few seconds
    Neural Network:              40 minutes

Messages:
Warning: Remapped class labels to be contiguous. Use -cm if DET/ROC-based accuracy measur
ements are wrong.

[+] Building 0.0s (0/1)
[+] Building 0.1s (2/2)
 => [internal] load build definition from btc-dockerfile.12650             0.0s
 => => transferring dockerfile: 239B                                       0.0s
 => [internal] load .dockerignore                                          0.0s
 => => transferring context: 2B                                            0.0s
```

```
[+] Building 0.2s (5/7)
 => [internal] load build definition from btc-dockerfile.12650              0.0s
 => => transferring dockerfile: 239B                                       0.0s
 => [internal] load .dockerignore                                          0.0s
 => => transferring context: 2B                                            0.0s
 => [internal] load metadata for docker.io/brainome/btc_local_cpu:alpha    0.0s
 => [internal] load build context                                          0.0s
 => => transferring context: 1.37kB                                        0.0s
 => [1/3] FROM docker.io/brainome/btc_local_cpu:alpha                      0.0s
[+] Building 0.3s (8/8) FINISHED
 => [internal] load build definition from btc-dockerfile.12650              0.0s
 => => transferring dockerfile: 239B                                       0.0s
 => [internal] load .dockerignore                                          0.0s
 => => transferring context: 2B                                            0.0s
 => [internal] load metadata for docker.io/brainome/btc_local_cpu:alpha    0.0s
 => [internal] load build context                                          0.0s
 => => transferring context: 1.37kB                                        0.0s
 => [1/3] FROM docker.io/brainome/btc_local_cpu:alpha                      0.0s
 => CACHED [2/3] RUN adduser --disabled-password --gecos '' --uid 501 --g  0.0s
 => CACHED [3/3] COPY --chown=501:20 .daimensions.key /btc-alex            0.0s
 => exporting to image                                                     0.1s
 => => exporting layers                                                    0.0s
 => => writing image sha256:43dd56f18fcffcdd02c290bfcce87c7a6cd0b9ccf4db3  0.0s
 => => naming to docker.io/library/btc-alex:latest                         0.0s
Docker image btc-alex:latest updated successfully.
```

2. Build the Predictor

Based on our measurements, Daimensions recommends we use a neural network (higher expected generalization) and more effort for this dataset. Don't forget to use -target because the target column isn't on the very right.

In [8]:

```
! btc -f NN usps_train.csv -o usps_predict.py -target int0 -e 5 --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 -f NN usps_train.csv -o usps_predict.py -target int0 -e 5 --yes

Start Time:                 03/05/2021, 20:09


Data:
    Input:                      usps_train.csv
    Target Column:              int0
    Number of instances:        7438
    Number of attributes:       256
    Number of classes:          10
    Class Balance:              0: 7.56%, 1: 8.86%, 2: 16.79%, 3: 13.70%, 4: 9.75%, 5: 8.
67%, 6: 9.38%, 7: 9.08%, 8: 8.55%, 9: 7.66%

Learnability:
    Best guess accuracy:          16.79%
    Data Sufficiency:            Maybe enough data to generalize. [yellow]
```

```
Capacity Progression:           at [ 5%, 10%, 20%, 40%, 80%, 100% ]
    Optimal Machine Learner:          10,  11,  12,  12,  13,  13


Estimated Memory Equivalent Capacity for...
    Decision Tree:                 5973 parameters
    Neural Networks:                241 parameters
    Random Forest:                  132 parameters

Risk that model needs to overfit for 100% accuracy using...
    Decision Tree:                90.00%
    Neural Networks:               9.89%
    Random Forest:                 4.11%

Expected Generalization using...
    Decision Tree:                 4.07 bits/bit
    Neural Network:               15.42 bits/bit
    Random Forest:                56.35 bits/bit


Recommendations:


    Note: Machine learner type NN given by user.


Time to Build Estimates:

    Neural Network:             42 minutes



Messages:
Warning: Remapped class labels to be contiguous. Use -cm if DET/ROC-based accuracy measur
ements are wrong.
Error Error: Predictor building failed. Output:
```

3. Validate the Model

Now we can validate our model on our test data, a separate set of data that wasn't used for training.

In [4]:

```
! python3 usps_predict.py -validate usps_valid.csv
```

```
Classifier Type:                    Neural Network
System Type:                        10-way classifier
Best-guess accuracy:                16.34%
Model accuracy:                     93.49% (1739/1860 correct)
Improvement over best guess:        77.15% (of possible 83.66%)
Model capacity (MEC):               574 bits
Generalization ratio:               9.89 bits/bit
Model efficiency:                   0.13%/parameter
Confusion Matrix:
 [6.88% 0.22% 0.05% 0.00% 0.11% 0.05% 0.32% 0.05% 0.05% 0.11%]
 [0.05% 8.39% 0.00% 0.00% 0.05% 0.00% 0.00% 0.00% 0.05% 0.32%]
 [0.05% 0.05% 15.81% 0.00% 0.22% 0.00% 0.00% 0.05% 0.05% 0.11%]
 [0.05% 0.05% 0.00% 13.17% 0.05% 0.05% 0.00% 0.05% 0.00% 0.00%]
 [0.22% 0.05% 0.16% 0.00% 10.05% 0.00% 0.27% 0.05% 0.00% 0.16%]
 [0.22% 0.00% 0.00% 0.00% 0.11% 8.60% 0.27% 0.00% 0.22% 0.05%]
 [0.00% 0.00% 0.16% 0.00% 0.00% 0.27% 7.74% 0.00% 0.00% 0.11%]
 [0.05% 0.00% 0.05% 0.00% 0.11% 0.00% 0.16% 8.17% 0.00% 0.00%]
 [0.00% 0.11% 0.11% 0.00% 0.05% 0.11% 0.05% 0.00% 7.96% 0.00%]
 [0.11% 0.38% 0.22% 0.05% 0.00% 0.11% 0.11% 0.11% 0.05% 6.72%]
```

Hooray! We have validated the accuracy of our model and found that it has a 92.9% accuracy for the test data. We can also see the confusion matrix, which tells us the percentage of data points from each class (columns) that were predicted to be in a certain class (rows). The diagonals are correctly predicted data points.