H2O_Neural_Networks

January 22, 2019

```
In [1]: # imports
        import h2o
        import numpy as np
        import pandas as pd
        from h2o.estimators.deeplearning import H2ODeepLearningEstimator
        from h2o.grid.grid_search import H2OGridSearch
```

```
In [2]: # display matplotlib graphics in notebook
        %matplotlib inline
```

```
In [3]: # start and connect to h2o server
        h2o.init()
```

```
Checking whether there is an H2O instance running at http://localhost:54321... not found.
Attempting to start a local H2O server...
  Java Version: openjdk version "1.8.0_121"; OpenJDK Runtime Environment (Zulu 8.20.0.5-macosx)
  Starting server from /Users/bear/anaconda/lib/python3.6/site-packages/h2o/backend/bin/h2o.jar
  Ice root: /var/folders/lh/42j8mfjx069d1bkc2wlf2pw40000gn/T/tmpw_a3y2kf
  JVM stdout: /var/folders/lh/42j8mfjx069d1bkc2wlf2pw40000gn/T/tmpw_a3y2kf/h2o_bear_started_from
  JVM stderr: /var/folders/lh/42j8mfjx069d1bkc2wlf2pw40000gn/T/tmpw_a3y2kf/h2o_bear_started_from
  Server is running at http://127.0.0.1:54321
Connecting to H2O server at http://127.0.0.1:54321... successful.
```

```
--------------------------  ----------------------------------------
H2O cluster uptime:         01 secs
H2O cluster timezone:       America/New_York
H2O data parsing timezone:  UTC
H2O cluster version:        3.22.1.1
H2O cluster version age:    25 days
H2O cluster name:           H2O_from_python_bear_d9y5oa
H2O cluster total nodes:    1
H2O cluster free memory:    3.556 Gb
H2O cluster total cores:    8
H2O cluster allowed cores:  8
H2O cluster status:         accepting new members, healthy
H2O connection url:         http://127.0.0.1:54321
H2O connection proxy:
```

```
H2O internal security:      False
H2O API Extensions:         XGBoost, Algos, AutoML, Core V3, Core V4
Python version:             3.6.5 final
--------------------------  ----------------------------------------
```

In [4]:
```
# load clean data
path = 'data/loan.csv'
```

In [5]:
```
# define input variable measurement levels
# strings automatically parsed as enums (nominal)
# numbers automatically parsed as numeric
col_types = {'bad_loan': 'enum'}
```

In [6]:
```
frame = h2o.import_file(path=path, col_types=col_types) # import from url
```

```
Parse progress: || 100%
```

In [7]:
```
frame.describe() # summarize data
```

```
Rows:163987
Cols:15
```

In [8]:
```
# split into 40% training, 30% validation, and 30% test
train, valid, test = frame.split_frame([0.4, 0.3])
```

In [9]:
```
# assign target and inputs
y = 'bad_loan'
X = [name for name in frame.columns if name != y]
print(y)
print(X)
```

```
bad_loan
['loan_amnt', 'term', 'int_rate', 'emp_length', 'home_ownership', 'annual_inc', 'purpose', 'addr
```

In [10]:
```
# determine column types
reals, enums = [], []
for key, val in frame.types.items():
    if key in X:
        if val == 'enum':
            enums.append(key)
        else:
            reals.append(key)

print(enums)
print(reals)
```

```
['term', 'home_ownership', 'purpose', 'addr_state', 'verification_status']
['loan_amnt', 'int_rate', 'emp_length', 'annual_inc', 'dti', 'delinq_2yrs', 'revol_util', 'total
```

```
In [11]: # impute missing values
         _ = frame[reals].impute(method='mean')
```

```
In [12]: # set target to factor - for binary classification
         # just to be safe ...
         train[y] = train[y].asfactor()
         valid[y] = valid[y].asfactor()
         test[y] = test[y].asfactor()
```

```
In [13]: # neural network

         # initialize nn model
         nn_model = H2ODeepLearningEstimator(
             epochs=50,                        # read over the data 50 times, but in mini-batches
             hidden=[100],                     # 100 hidden units in 1 hidden layer
             input_dropout_ratio=0.2,          # randomly drop 20% of inputs for each iteration, hel
             hidden_dropout_ratios=[0.05], # randomly set 5% of hidden weights to 0 each iterati
             activation='TanhWithDropout', # bounded activation function that allows for dropout
             l1=0.001,                         # L1 penalty can help generalization
             l2=0.01,                          # L2 penalty can increase stability in presence of hi
             adaptive_rate=True,               # adjust magnitude of weight updates automatically (+
             stopping_rounds=5,                # stop after validation error does not decrease for 5
             score_each_iteration=True,        # score validation error on every iteration
             model_id='nn_model')              # for easy lookup in flow

         # train nn model
         nn_model.train(
             x=X,
             y=y,
             training_frame=train,
             validation_frame=valid)

         # print model information
         nn_model

         # view detailed results at http://localhost:54321/flow/index.html
```

```
deeplearning Model Build progress: || 100%
Model Details
=============
H2ODeepLearningEstimator :  Deep Learning
Model Key:  nn_model


ModelMetricsBinomial: deeplearning
```

** Reported on train data. **

MSE: 0.14012752126328837
RMSE: 0.3743361073464439
LogLoss: 0.4443499533129185
Mean Per-Class Error: 0.35657985163704065
AUC: 0.6915638683615544
pr_auc: 0.3288191580005072
Gini: 0.38312773672310874
Confusion Matrix (Act/Pred) for max f1 @ threshold = 0.19205222874853198:

	0	1	Error	Rate
0	5439	2757	0.3364	(2757.0/8196.0)
1	704	1158	0.3781	(704.0/1862.0)
Total	6143	3915	0.3441	(3461.0/10058.0)

Maximum Metrics: Maximum metrics at their respective thresholds

metric	threshold	value	idx
max f1	0.192052	0.4009	241
max f2	0.102222	0.558789	340
max f0point5	0.276488	0.364826	166
max accuracy	0.560144	0.815271	16
max precision	0.61433	0.636364	4
max recall	0.0491335	1	397
max specificity	0.640472	0.999878	0
max absolute_mcc	0.241647	0.231823	194
max min_per_class_accuracy	0.185285	0.638561	248
max mean_per_class_accuracy	0.176703	0.64342	257

Gains/Lift Table: Avg response rate: 18.51 %, avg score: 19.14 %

	group	cumulative_data_fraction	lower_threshold	lift	cumulative_lift	respons
	1	0.0100418	0.528289	2.56715	2.56715	0.47524
	2	0.0200835	0.491005	2.40671	2.48693	0.44554
	3	0.0300259	0.466029	2.16069	2.3789	0.4
	4	0.0400676	0.444594	2.29974	2.35906	0.42574
	5	0.0500099	0.426483	2.53881	2.3948	0.47
	6	0.10002	0.355973	2.06189	2.22834	0.38171

7	0.15003	0.309743	1.57863	2.01177	0.29224
8	0.20004	0.274472	1.71824	1.93839	0.31809
9	0.30006	0.222695	1.29942	1.7254	0.24055
10	0.39998	0.188336	1.12334	1.575	0.20796
11	0.5	0.159245	0.945032	1.44898	0.17495
12	0.60002	0.136281	0.805425	1.3417	0.14910
13	0.69994	0.11686	0.585858	1.2338	0.10845
14	0.79996	0.0984589	0.617493	1.15674	0.11431
15	0.89998	0.0794413	0.467147	1.08011	0.08648
16	1	0.0412045	0.279214	1	0.05168

ModelMetricsBinomial: deeplearning
** Reported on validation data. **

MSE: 0.13897766905487463
RMSE: 0.37279708831330033
LogLoss: 0.4413991844691089
Mean Per-Class Error: 0.35983832063361787
AUC: 0.6906684424910284
pr_auc: 0.32383478570708496
Gini: 0.38133688498205687
Confusion Matrix (Act/Pred) for max f1 @ threshold = 0.1974062202858475:

	0	1	Error	Rate
0	27617	12638	0.3139	(12638.0/40255.0)
1	3672	5313	0.4087	(3672.0/8985.0)
Total	31289	17951	0.3312	(16310.0/49240.0)

Maximum Metrics: Maximum metrics at their respective thresholds

metric	threshold	value	idx
max f1	0.197406	0.394491	235
max f2	0.113771	0.55799	327
max f0point5	0.264697	0.352595	179
max accuracy	0.534647	0.817953	26
max precision	0.657312	1	0
max recall	0.0420548	1	399
max specificity	0.657312	1	0
max absolute_mcc	0.206718	0.222814	226
max min_per_class_accuracy	0.182287	0.638604	250

max mean_per_class_accuracy 0.186885   0.640162 245

Gains/Lift Table: Avg response rate: 18.25 %, avg score: 18.94 %

	group	cumulative_data_fraction	lower_threshold	lift	cumulative_lift	respons
	1	0.0100122	0.527923	2.82349	2.82349	0.51521
	2	0.0200041	0.492837	2.30571	2.56487	0.42073
	3	0.0300162	0.468294	2.31215	2.48057	0.42190
	4	0.0400081	0.444579	2.2166	2.41465	0.40447
	5	0.05	0.425445	2.11635	2.35504	0.38617
	6	0.1	0.353973	1.97663	2.16583	0.36068
	7	0.15	0.305637	1.66055	1.9974	0.30300
	8	0.2	0.270679	1.56483	1.88926	0.28554
	9	0.3	0.22087	1.37563	1.71805	0.25101
	10	0.4	0.18541	1.133	1.57179	0.20674
	11	0.5	0.157594	0.915971	1.44062	0.16714
	12	0.6	0.134695	0.830273	1.3389	0.15150
	13	0.7	0.114595	0.706733	1.24859	0.12896
	14	0.8	0.0964226	0.545353	1.16068	0.09951
	15	0.9	0.0786919	0.445186	1.08118	0.08123
	16	1	0.0412108	0.269338	1	0.04914

Scoring History:

	timestamp	duration	training_speed	epochs	iterations	samples	traini
	2019-01-22 22:06:37	0.000 sec		0	0	0	nan
	2019-01-22 22:06:40	3.425 sec	61287 obs/sec	1.52077	1	99898	0.3763
	2019-01-22 22:06:42	5.076 sec	67805 obs/sec	3.0409	2	199754	0.3757
	2019-01-22 22:06:43	6.674 sec	70761 obs/sec	4.56525	3	299887	0.3758
	2019-01-22 22:06:46	10.050 sec	57386 obs/sec	6.08385	4	399642	0.3750
	2019-01-22 22:06:48	11.742 sec	60042 obs/sec	7.60751	5	499730	0.3749
	2019-01-22 22:06:50	13.418 sec	61783 obs/sec	9.12995	6	599737	0.3793
	2019-01-22 22:06:52	15.069 sec	63289 obs/sec	10.6502	7	699598	0.3802
	2019-01-22 22:06:53	16.646 sec	64747 obs/sec	12.1739	8	799692	0.3765
	2019-01-22 22:06:55	18.213 sec	65944 obs/sec	13.6961	9	899682	0.3743
	2019-01-22 22:06:57	20.124 sec	65649 obs/sec	15.2189	10	999716	0.3773
	2019-01-22 22:06:57	20.420 sec	65615 obs/sec	15.2189	10	999716	0.3743

Variable Importances:

```
variable                         relative_importance    scaled_importance    percentage
-------------------------------  ---------------------  -------------------  --------------------
purpose.small_business           1.0                    1.0                  0.06090408556279186
addr_state.CO                    0.5995688438415527     0.5995688438415527   0.03651619216611012
annual_inc                       0.5280225872993469     0.5280225872993469   0.03215873283596617
int_rate                         0.5251196622848511     0.5251196622848511   0.03198193284250094
addr_state.WV                    0.45759496092796326    0.45759496092796326  0.02786940265345907
---                              ---                    ---                  ---
addr_state.missing(NA)           0.0                    0.0                  0.0
purpose.missing(NA)              0.0                    0.0                  0.0
home_ownership.missing(NA)       0.0                    0.0                  0.0
term.missing(NA)                 0.0                    0.0                  0.0
verification_status.missing(NA)  0.0                    0.0                  0.0
```

```
See the whole table with table.as_data_frame()
```

Out[13]:

In [14]:
```python
# measure nn AUC
print(nn_model.auc(train=True))
print(nn_model.auc(valid=True))
print(nn_model.model_performance(test_data=test).auc())
```

```
0.6915638683615544
0.6906684424910284
0.6956753398991676
```

In [15]:
```python
# NN with random hyperparameter search
# train many different NN models with random hyperparameters
# and select best model based on validation error

# define random grid search parameters
hyper_parameters = {'hidden':[[170, 320], [80, 190], [320, 160, 80], [100], [50, 50, 50
                    'l1':[s/1e4 for s in range(0, 1000, 100)],
                    'l2':[s/1e5 for s in range(0, 1000, 100)],
                    'input_dropout_ratio':[s/1e2 for s in range(0, 20, 2)]}

# define search strategy
search_criteria = {'strategy':'RandomDiscrete',
                   'max_models':20,
                   'max_runtime_secs':600}

# initialize grid search
gsearch = H2OGridSearch(H2ODeepLearningEstimator,
                        hyper_params=hyper_parameters,
```

```
                               search_criteria=search_criteria)

        # execute training w/ grid search
        gsearch.train(x=X,
                      y=y,
                      training_frame=train,
                      validation_frame=valid)

        # view detailed results at http://ip:port/flow/index.html
```

```
deeplearning Grid Build progress: || 100%
```

```
In [16]: # show grid search results
         gsearch.show()

         # select best model
         nn_model2 = gsearch.get_grid()[0]

         # print model information
         nn_model2
```

```
              hidden input_dropout_ratio    l1     l2  \
0              [100]                 0.0   0.0  0.003
1              [100]                0.16   0.0  0.006
2         [170, 320]                 0.0   0.0  0.009
3         [170, 320]                0.06  0.07  0.006
4          [80, 190]                0.12  0.05  0.002
5   [50, 50, 50, 50]                0.18  0.08  0.009
6         [170, 320]                 0.0  0.01  0.009
7              [100]                 0.1  0.04  0.008
8   [50, 50, 50, 50]                0.12  0.03  0.008
9          [80, 190]                 0.0  0.02  0.006
10    [320, 160, 80]                0.18  0.08    0.0
11  [50, 50, 50, 50]                0.04  0.07    0.0
12    [320, 160, 80]                0.02  0.07  0.004
13    [320, 160, 80]                0.08  0.02  0.003
14        [170, 320]                0.16  0.04  0.009
15             [100]                0.06  0.06  0.007
16         [80, 190]                0.16  0.01  0.006
17        [170, 320]                 0.0  0.02  0.001
18    [320, 160, 80]                0.18  0.02  0.002
19             [100]                0.02  0.05  0.001

                                             model_ids  \
0   Grid_DeepLearning_py_7_sid_b0d1_model_python_1548212791865_24_model_4
1   Grid_DeepLearning_py_7_sid_b0d1_model_python_1548212791865_24_mode...
2   Grid_DeepLearning_py_7_sid_b0d1_model_python_1548212791865_24_model_5
```

```
3   Grid_DeepLearning_py_7_sid_b0d1_model_python_1548212791865_24_mode...
4   Grid_DeepLearning_py_7_sid_b0d1_model_python_1548212791865_24_mode...
5   Grid_DeepLearning_py_7_sid_b0d1_model_python_1548212791865_24_mode...
6   Grid_DeepLearning_py_7_sid_b0d1_model_python_1548212791865_24_model_9
7   Grid_DeepLearning_py_7_sid_b0d1_model_python_1548212791865_24_mode...
8   Grid_DeepLearning_py_7_sid_b0d1_model_python_1548212791865_24_model_2
9   Grid_DeepLearning_py_7_sid_b0d1_model_python_1548212791865_24_mode...
10  Grid_DeepLearning_py_7_sid_b0d1_model_python_1548212791865_24_model_7
11  Grid_DeepLearning_py_7_sid_b0d1_model_python_1548212791865_24_model_8
12  Grid_DeepLearning_py_7_sid_b0d1_model_python_1548212791865_24_model_1
13  Grid_DeepLearning_py_7_sid_b0d1_model_python_1548212791865_24_mode...
14  Grid_DeepLearning_py_7_sid_b0d1_model_python_1548212791865_24_mode...
15  Grid_DeepLearning_py_7_sid_b0d1_model_python_1548212791865_24_mode...
16  Grid_DeepLearning_py_7_sid_b0d1_model_python_1548212791865_24_model_6
17  Grid_DeepLearning_py_7_sid_b0d1_model_python_1548212791865_24_model_3
18  Grid_DeepLearning_py_7_sid_b0d1_model_python_1548212791865_24_mode...
19  Grid_DeepLearning_py_7_sid_b0d1_model_python_1548212791865_24_mode...

                logloss
0   0.44092351643259303
1    0.4443169689842515
2   0.44777615659277836
3    0.4751245521436998
4   0.47526618794661657
5    0.4754214359385578
6   0.47548375735728876
7    0.4755765022643118
8    0.4756250068385829
9    0.4756706372390496
10   0.4761257167663083
11   0.4761717953604289
12   0.4762053305368573
13   0.4763941041967503
14   0.4764412458354611
15   0.4812486711359482
16  0.48376347302381506
17  0.48376541523613664
18  0.48427833738692344
19   0.4863340143327703
Model Details
=============
H2ODeepLearningEstimator :  Deep Learning
Model Key:  Grid_DeepLearning_py_7_sid_b0d1_model_python_1548212791865_24_model_4


ModelMetricsBinomial: deeplearning
** Reported on train data. **
```

MSE: 0.1373519914200528
RMSE: 0.3706102958905119
LogLoss: 0.4349174231468685
Mean Per-Class Error: 0.3410720687074309
AUC: 0.7145815998888247
pr_auc: 0.3537231416142696
Gini: 0.4291631997776495
Confusion Matrix (Act/Pred) for max f1 @ threshold = 0.19309119092123808:

	0	1	Error	Rate
0	5653	2578	0.3132	(2578.0/8231.0)
1	704	1159	0.3779	(704.0/1863.0)
Total	6357	3737	0.3251	(3282.0/10094.0)

Maximum Metrics: Maximum metrics at their respective thresholds

metric	threshold	value	idx
max f1	0.193091	0.413929	232
max f2	0.0949289	0.575918	318
max f0point5	0.29572	0.383836	158
max accuracy	0.557329	0.817218	29
max precision	0.720939	1	0
max recall	0.00795781	1	397
max specificity	0.720939	1	0
max absolute_mcc	0.23079	0.250845	202
max min_per_class_accuracy	0.17891	0.653991	243
max mean_per_class_accuracy	0.144566	0.658928	272

Gains/Lift Table: Avg response rate: 18.46 %, avg score: 17.98 %

	group	cumulative_data_fraction	lower_threshold	lift	cumulative_lift	respons
	1	0.0100059	0.560417	3.05776	3.05776	0.56435
	2	0.0200119	0.523473	2.73589	2.89683	0.50495
	3	0.0300178	0.488575	2.41402	2.73589	0.44554
	4	0.0400238	0.464198	1.87757	2.52131	0.34653
	5	0.0500297	0.444922	2.25309	2.46767	0.41584
	6	0.100059	0.375695	1.97414	2.2209	0.36435
	7	0.14999	0.325201	1.93505	2.12574	0.35714
	8	0.20002	0.28431	1.78101	2.03952	0.32871

9	0.29998	0.225927	1.29413	1.79114	0.23885
10	0.40004	0.17985	1.12654	1.62491	0.20792
11	0.5	0.14624	1.07396	1.51476	0.19821
12	0.59996	0.115711	0.751774	1.38764	0.13875
13	0.70002	0.0901271	0.622282	1.27824	0.11485
14	0.79998	0.0649302	0.488653	1.17958	0.09018
15	0.899941	0.0413871	0.359778	1.08852	0.06640
16	1	0.000106829	0.203851	1	0.03762

ModelMetricsBinomial: deeplearning
** Reported on validation data. **

MSE: 0.13879047534556194
RMSE: 0.3725459372286348
LogLoss: 0.44092351643259303
Mean Per-Class Error: 0.35729763796421077
AUC: 0.6967998458906275
pr_auc: 0.33044848884930395
Gini: 0.393599691781255
Confusion Matrix (Act/Pred) for max f1 @ threshold = 0.1811819503331:

	0	1	Error	Rate
0	26221	14034	0.3486	(14034.0/40255.0)
1	3289	5696	0.3661	(3289.0/8985.0)
Total	29510	19730	0.3518	(17323.0/49240.0)

Maximum Metrics: Maximum metrics at their respective thresholds

metric	threshold	value	idx
max f1	0.181182	0.396726	244
max f2	0.0848574	0.56023	327
max f0point5	0.314375	0.364242	153
max accuracy	0.632333	0.817973	21
max precision	0.866518	1	0
max recall	0.00422494	1	398
max specificity	0.866518	1	0
max absolute_mcc	0.226436	0.231077	210
max min_per_class_accuracy	0.177682	0.641635	247
max mean_per_class_accuracy	0.179213	0.642702	246

Gains/Lift Table: Avg response rate: 18.25 %, avg score: 18.04 %

	group	cumulative_data_fraction	lower_threshold	lift	cumulative_lift	respons
	1	0.0100122	0.572463	2.74568	2.74568	0.50101
	2	0.0200041	0.527374	2.52849	2.63719	0.46138
	3	0.0300162	0.497283	2.37885	2.55102	0.43407
	4	0.0400081	0.473143	2.18319	2.45916	0.39837
	5	0.05	0.452193	2.16091	2.39955	0.39430
	6	0.1	0.376891	1.98553	2.19254	0.36230
	7	0.15	0.323595	1.84085	2.07531	0.33590
	8	0.2	0.285333	1.50473	1.93267	0.27457
	9	0.3	0.22468	1.36895	1.74476	0.24979
	10	0.4	0.180849	1.09516	1.58236	0.19983
	11	0.5	0.145926	0.934891	1.45287	0.17059
	12	0.6	0.11661	0.838063	1.3504	0.15292
	13	0.7	0.0902488	0.666667	1.25272	0.12164
	14	0.8	0.0645946	0.583194	1.16903	0.10641
	15	0.9	0.0405852	0.420701	1.08588	0.07676
	16	1	8.19897e-07	0.227045	1	0.04142

Scoring History:

	timestamp	duration	training_speed	epochs	iterations	samples	traini
	2019-01-22 22:07:46	0.000 sec		0	0	0	nan
	2019-01-22 22:07:47	48.439 sec	72987 obs/sec	1	1	65689	0.3805
	2019-01-22 22:07:51	52.769 sec	131430 obs/sec	10	10	656890	0.3706

Variable Importances:

variable	relative_importance	scaled_importance	percentage
addr_state.MO	1.0	1.0	0.01519030535232567
purpose.small_business	0.9904335141181946	0.9904335141181946	0.01504498751063234
addr_state.TN	0.9665326476097107	0.9665326476097107	0.01468192605018329
purpose.renewable_energy	0.9472734928131104	0.9472734928131104	0.01438937360799523
addr_state.DC	0.9444687366485596	0.9444687366485596	0.01434676850541688
---	---	---	---
addr_state.missing(NA)	0.0	0.0	0.0
purpose.missing(NA)	0.0	0.0	0.0
home_ownership.missing(NA)	0.0	0.0	0.0

```
term.missing(NA)                0.0                     0.0                   0.0
verification_status.missing(NA) 0.0                     0.0                   0.0
```

```
See the whole table with table.as_data_frame()
```

Out[16]:

```
In [17]: # measure nn AUC
         print(nn_model2.auc(train=True))
         print(nn_model2.auc(valid=True))
         print(nn_model2.model_performance(test_data=test).auc())
```

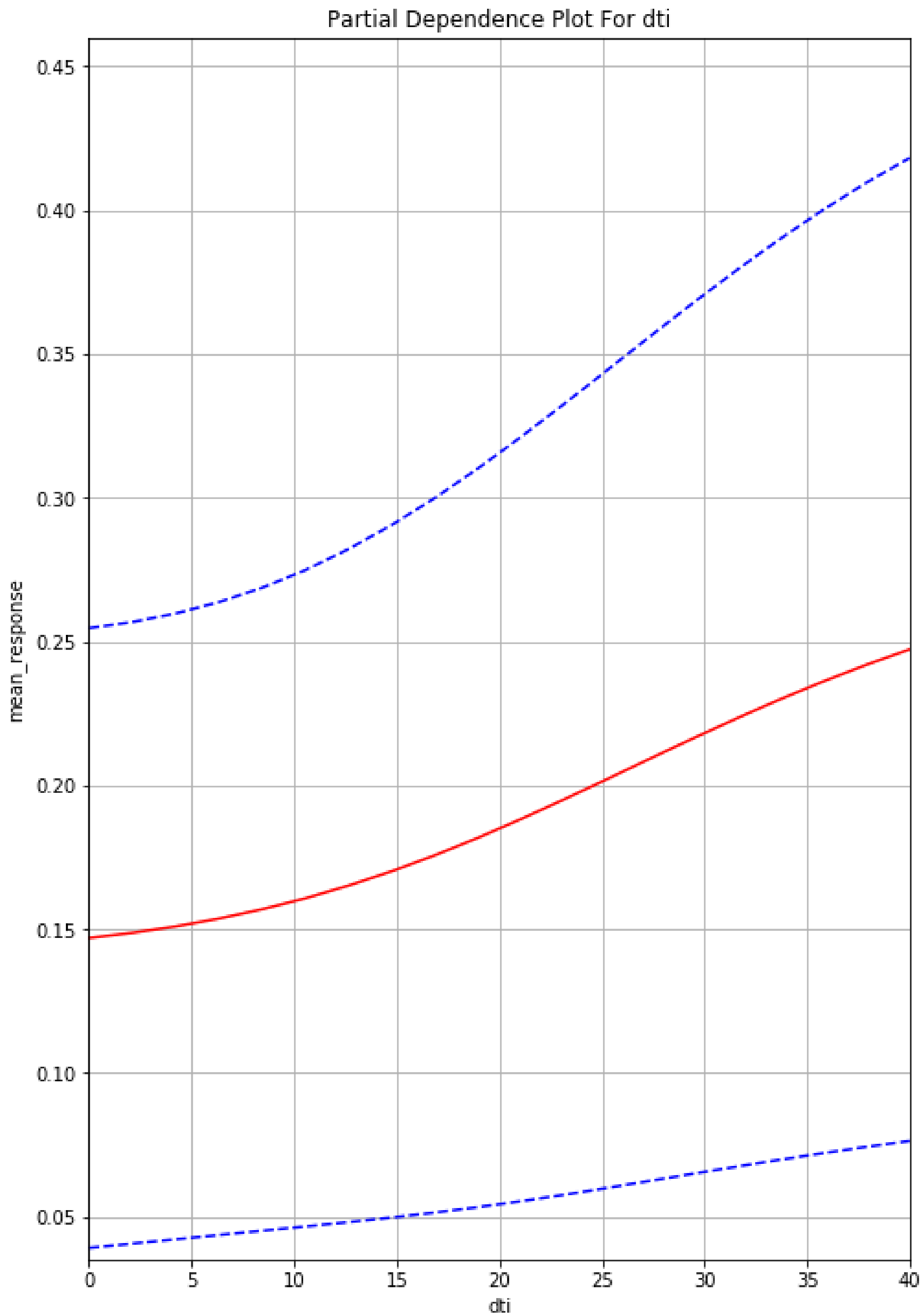
```
0.7145815998888247
0.6967998458906275
0.7041325607237255
```

0.1 Partial dependence plots

The partial dependence plot (short PDP or PD plot) shows the marginal effect one or two features have on the predicted outcome of a machine learning model. A partial dependence plot can show whether the relationship between the target and a feature is linear, monotonous or more complex. https://christophm.github.io/interpretable-ml-book/pdp.html

```
In [18]: # partial dependence plots are a powerful machine learning interpretation tool
         # to calculate partial dependence across the domain a variable
         # hold column of interest at constant value
         # find the mean prediction of the model with this column constant
         # repeat for multiple values of the variable of interest
         # h2o has a built-in function for partial dependence as well
         par_dep_dti1 = nn_model2.partial_plot(data=train, cols=['dti'], server=True, plot=True)
```

```
PartialDependencePlot progress: || 100%
```



Partial Dependence Plot For dti
0.45
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
mean_response
0
5
10
15
20
25
30
35
40
dti

```
In [19]: # shutdown h2o
         # be careful ... this can erase your work
         h2o.cluster().shutdown()
```

```
H2O session _sid_b0d1 closed.
```