H2O_automl_model

November 20, 2018

1 AutoML: Automatic Machine Learning

AutoML: Automatic Machine Learning

H2O's AutoML is used for automating the machine learning workflow, which includes automatic training and tuning of many models within a user-specified time-limit. Stacked Ensembles will be automatically trained on collections of individual models to produce highly predictive ensemble models which, in most cases, will be the top performing models in the AutoML Leaderboard.

1.1 Tutorials

- Intro to AutoML + Hands-on Lab - Erin LeDell, Machine Learning Scientist... https://youtu.be/42Oo8TOl85I
- Scalable Automatic Machine Learning in H2O https://youtu.be/j6rqrEYQNdo
- Scalable Automatic Machine Learning in H2O https://youtu.be/j6rqrEYQNdo

1.2 Installing H2O and h2o python

See http://docs.h2o.ai/h2o/latest-stable/h2o-docs/downloading.html

Click the Download H2O button on the http://h2o-release.s3.amazonaws.com/h2o/latest_stable.html page. This downloads a zip file that contains everything you need to get started.

```
cd ~/Downloads
unzip h2o-3.20.0.1.zip
cd h2o-3.20.0.1
java -jar h2o.jar
```

Point your browser to http://localhost:54321.

Install in Python

Install dependencies (prepending with sudo if needed):

```
pip install requests
pip install tabulate
pip install scikit-learn
pip install colorama
pip install future
```

Remove any existing H2O module for Python.

```
pip uninstall h2o
```

Use pip to install this version of the H2O Python module.

```
pip install -f http://h2o-release.s3.amazonaws.com/h2o/latest_stable_Py.html h2o
```

Note: When installing H2O from pip in OS X El Capitan, users must include the --user flag. For example:

```
pip install -f http://h2o-release.s3.amazonaws.com/h2o/latest_stable_Py.html h2o --user
```

Initialize H2O in Python and run a demo to see H2O at work.

```
python
import h2o
h2o.init()
h2o.demo("glm")
```

1.3 Saving data

H2O model file file will be saved in one of two formats.

There are two ways to save the leader model -- binary format and MOJO format. If you're taking your leader model to production, then we'd suggest the MOJO format since it's optimized for production use.

See http://docs.h2o.ai/h2o/latest-stable/h2o-docs/save-and-load-model.html

```
# save the model
model_path = h2o.save_model(model=model, path="/tmp/mymodel", force=True)

# or

h2o.save_model(aml.leader, path = "./models")

# or

aml.leader.download_mojo(path = "./models")

# load the model
saved_model = h2o.load_model(model_path)
```

Saving data from runs

Stats about the models can be saved as text or csv or put directly in a database.

Much of the data is gathered by converting H2O objects to pandas data frame. So anything that a pandas data frame can be saved as is supported. https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.html

```
data_pd = object.as_data_frame(use_pandas=True)
```

Otherwise data is returned as python dictionaries or lists.

```
[('addr_state', 258199.28125, 1.0, 0.19965953057652525), ('int_rate', 203347.0625, 0.78755859240
```

1.4 Starting H2O server

```
In [65]: # import h2o package and specific estimator
         import h2o
         from h2o.automl import H2OAutoML
```

h2o.init with python seems very sensitive the the H2O version. If the H2O cluster version is 3.20.0.1 and the python h2o library is 3.19.0 it will fail so we set strict_version_check=False

If the H2O cluster isn't found h2o.init will start one.

Note that the current script starts each H2O instance on a different port. It's not clear why but should we do this we should choose from only the higher ports.

A port number is a 16-bit unsigned integer, thus ranging from 0 to 65535. There is no reason to choose a port less than 10000.

```
In [66]: h2o.init(strict_version_check=False) # start h2o
```

```
Checking whether there is an H2O instance running at http://localhost:54321. connected.
```

```
--------------------------  ----------------------------------------
H2O cluster uptime:         10 mins 06 secs
H2O cluster timezone:       America/New_York
H2O data parsing timezone:  UTC
H2O cluster version:        3.20.0.1
H2O cluster version age:    5 months and 13 days !!!
H2O cluster name:           H2O_from_python_bear_07058y
H2O cluster total nodes:    1
H2O cluster free memory:    2.129 Gb
H2O cluster total cores:    8
H2O cluster allowed cores:  8
H2O cluster status:         locked, healthy
H2O connection url:         http://localhost:54321
H2O connection proxy:
H2O internal security:      False
H2O API Extensions:         XGBoost, Algos, AutoML, Core V3, Core V4
Python version:             3.6.5 final
--------------------------  ----------------------------------------
```

1.5 h2o.automl Parameters

http://docs.h2o.ai/h2o/latest-stable/h2o-docs/automl.html

NB: Eventually one wants to expose all the parameters to the expert user.

Required Data Parameters

y: This argument is the name (or index) of the response column.

training_frame: Specifies the training set.

The user gives the name of the depenent variable and training file name.

Required Stopping Parameters

One of the following stopping strategies (time or number-of-model based) must be specified. When both options are set, then the AutoML run will stop as soon as it hits one of either of these limits.

max_runtime_secs: This argument controls how long the AutoML run will execute for. This defaults to 3600 seconds (1 hour).

max_models: Specify the maximum number of models to build in an AutoML run, excluding the Stacked Ensemble models. Defaults to NULL/None.

1.5.1 Optional Parameters

Optional Data Parameters

x: A list/vector of predictor column names or indexes. This argument only needs to be specified if the user wants to exclude columns from the set of predictors. If all columns (other than the response) should be used in prediction, then this does not need to be set.

validation_frame: This argument is used to specify the validation frame used for early stopping of individual models and early stopping of the grid searches (unless max_models or max_runtime_secs overrides metric-based early stopping).

leaderboard_frame: This argument allows the user to specify a particular data frame use to score & rank models on the leaderboard. This frame will not be used for anything besides leaderboard scoring. If a leaderboard frame is not specified by the user, then the leaderboard will use cross-validation metrics instead (or if cross-validation is turned off by setting nfolds = 0, then a leaderboard frame will be generated automatically from the validation frame (if provided) or the training frame).

fold_column: Specifies a column with cross-validation fold index assignment per observation. This is used to override the default, randomized, 5-fold cross-validation scheme for individual models in the AutoML run.

weights_column: Specifies a column with observation weights. Giving some observation a weight of zero is equivalent to excluding it from the dataset; giving an observation a relative weight of 2 is equivalent to repeating that row twice. Negative weights are not allowed.

ignored_columns: (Optional, Python only) Specify the column or columns (as a list/vector) to be excluded from the model. This is the converse of the x argument.

Optional Miscellaneous Parameters

nfolds: Number of folds for k-fold cross-validation of the models in the AutoML run. Defaults to 5. Use 0 to disable cross-validation; this will also disable Stacked Ensembles (thus decreasing the overall best model performance).

balance_classes: Specify whether to oversample the minority classes to balance the class distribution. This option is not enabled by default and can increase the data frame size. This option is only applicable for classification. Majority classes can be undersampled to satisfy the max_after_balance_size parameter.

class_sampling_factors: Specify the per-class (in lexicographical order) over/under-sampling ratios. By default, these ratios are automatically computed during training to obtain the class balance.

max_after_balance_size: Specify the maximum relative size of the training data after balancing class counts (balance_classes must be enabled). Defaults to 5.0. (The value can be less than 1.0).

stopping_metric: Specifies the metric to use for early stopping of the grid searches and individual models. Defaults to "AUTO". The available options are:

AUTO: This defaults to logloss for classification, deviance for regression deviance (mean residual deviance) logloss MSE RMSE MAE RMSLE AUC lift_top_group misclassification

mean_per_class_error

stopping_tolerance: This option specifies the relative tolerance for the metric-based stopping criterion to stop a grid search and the training of individual models within the AutoML run. This value defaults to 0.001 if the dataset is at least 1 million rows; otherwise it defaults to a bigger value determined by the size of the dataset and the non-NA-rate. In that case, the value is computed as 1/sqrt(nrows * non-NA-rate).

stopping_rounds: This argument is used to stop model training when the stopping metric (e.g. AUC) doesn't improve for this specified number of training rounds, based on a simple moving average. In the context of AutoML, this controls early stopping both within the random grid searches as well as the individual models. Defaults to 3 and must be an non-negative integer. To disable early stopping altogether, set this to 0.

sort_metric: Specifies the metric used to sort the Leaderboard by at the end of an AutoML run. Available options include:

AUTO: This defaults to AUC for binary classification, mean_per_class_error for multinomial classification, and deviance for regression. deviance (mean residual deviance) logloss MSE RMSE MAE RMSLE AUC mean_per_class_error

seed: Integer. Set a seed for reproducibility. AutoML can only guarantee reproducibility if max_models is used because max_runtime_secs is resource limited, meaning that if the available compute resources are not the same between runs, AutoML may be able to train more models on one run vs another. Defaults to NULL/None.

project_name: Character string to identify an AutoML project. Defaults to NULL/None, which means a project name will be auto-generated based on the training frame ID. More models can be trained and added to an existing AutoML project by specifying the same project name in muliple calls to the AutoML function (as long as the same training frame is used in subsequent runs).

exclude_algos: List/vector of character strings naming the algorithms to skip during the model-building phase. An example use is exclude_algos = ["GLM", "DeepLearning", "DRF"] in Python or exclude_algos = c("GLM", "DeepLearning", "DRF") in R. Defaults to None/NULL, which means that all appropriate H2O algorithms will be used, if the search stopping criteria allow. The algorithm names are:

GLM DeepLearning GBM DRF (This includes both the Random Forest and Extremely Randomized Trees (XRT) models. Refer to the Extremely Randomized Trees section in the DRF chapter and the histogram_type parameter description for more information.) StackedEnsemble keep_cross_validation_predictions: Specify whether to keep the predictions of the cross-validation predictions. If set to FALSE, then running the same AutoML object for repeated runs will cause an exception because CV predictions are are required to build additional Stacked Ensemble models in AutoML. This option defaults to TRUE.

keep_cross_validation_models: Specify whether to keep the cross-validated models. Deleting cross-validation models will save memory in the H2O cluster. This option defaults to TRUE.

```
In [67]: # Assume the following are passed by the user from the web interface

         '''
         Need a user id and project id?

         '''
         target='bad_loan'
         data_file='loan.csv'
         run_time=333
```

```
run_id='SOME_ID_20180617_221529' # Just some arbitrary ID
server_path='/Users/bear/Documents/INFO_7390/H2O'
classification=True
scale=False
max_models=None
balance_y=False # balance_classes=balance_y
balance_threshold=0.2
project ="automl_test"  # project_name = project
```

In [68]:
```
# Use local data file or download from some type of bucket
import os

data_path=os.path.join(server_path,data_file)
data_path
```

Out[68]: '/Users/bear/Documents/INFO_7390/H2O/loan.csv'

In [69]:
```
# Use local data file or download from some type of bucket
if not os.path.isfile(data_path):
  data_path = 'https://raw.githubusercontent.com/h2oai/app-consumer-loan/master/data/lo

# Load data into H2O
df = h2o.import_file(data_path)
```

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

In [70]:
```
df.describe()
```

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

In [71]:
```
# assign target and inputs for logistic regression
y = target
X = [name for name in df.columns if name != y]
print(y)
print(X)
```

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

In [72]:
```
# determine column types
ints, reals, enums = [], [], []
for key, val in df.types.items():
    if key in X:
```

```
                if val == 'enum':
                    enums.append(key)
                elif val == 'int':
                    ints.append(key)
                else:
                    reals.append(key)

        print(ints)
        print(enums)
        print(reals)
```

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

```
In [73]: # impute missing values
         _ = df[reals].impute(method='mean')
         _ = df[ints].impute(method='median')

         if scale:
             df[reals] = df[reals].scale()
             df[ints] = df[ints].scale()
```

```
In [74]: # set target to factor for classification by default or if user specifies classificatio
         if classification:
             df[y] = df[y].asfactor()
```

```
In [75]: df[y].levels()
```

```
Out[75]: [['0', '1']]
```

1.5.2 balance_classes check

If one class in two class classification is less than 20% of the total then one should set balance_classes=True

That is,

balance_classes=balance_y

```
In [76]: if classification:
             class_percentage = y_balance=df[y].mean()[0]/(df[y].max()-df[y].min())
             if class_percentage < balance_threshold:
                 balance_y=True
```

```
In [77]: print(run_time)
         type(run_time)
```

```
333
```

```
Out[77]: int
```

1.6 Cross-validate rather than take a test training split

Cross-validation rather than taking a test training split reduces the variance of the estimates of goodness of fit statistics. In rare cases one should take a test training split but this should be left to the expert users.

This also means the pro user can just upload the data and not worry about taking a test training split.

We can pass the original, full dataset, `df` (without passing a `leaderboard_frame`). This is a more efficient use of our data since we can use 100% of the data for training, rather than 80% or so. This time our leaderboard will use cross-validated metrics. It also gives better estimates of goodness of fit statistics.

Note: Using an explicit `leaderboard_frame` for scoring may be useful in some cases, which is why the option is available.

But it's not preferable in most cases. Leave it as an expert option.

```
In [78]: # automl
         # runs for run_time seconds then builds a stacked ensemble
         aml = H2OAutoML(max_runtime_secs=run_time,project_name = project,balance_classes=balanc
         aml.train(x=X,
                   y=y,
                   training_frame=df)
```

```
AutoML progress: || 100%
```

1.7 Leaderboard

Next, we will view the AutoML Leaderboard. Since we did not specify a `leaderboard_frame` in the `H2OAutoML.train()` method for scoring and ranking the models, the AutoML leaderboard uses cross-validation metrics to rank the models.

A default performance metric for each machine learning task (binary classification, multiclass classification, regression) is specified internally and the leaderboard will be sorted by that metric. In the case of binary classification, the default ranking metric is Area Under the ROC Curve (AUC). In the future, the user will be able to specify any of the H2O metrics so that different metrics can be used to generate rankings on the leaderboard.

The leader model is stored at `aml.leader` and the leaderboard is stored at `aml.leaderboard`.

```
In [79]: # view leaderboard
         lb = aml.leaderboard
         lb
```

```
Out[79]:
```

Now we will view a snapshot of the top models. Here we should see the two Stacked Ensembles at or near the top of the leaderboard. Stacked Ensembles can almost always outperform a single model.

```
In [80]: aml.leader
```

```
Model Details
=============
H2OStackedEnsembleEstimator :  Stacked Ensemble
Model Key:  StackedEnsemble_AllModels_0_AutoML_20181120_154205
No model summary for this model


ModelMetricsBinomialGLM: stackedensemble
** Reported on train data. **

MSE: 0.12685990330985406
RMSE: 0.3561739789904002
LogLoss: 0.40703958453227496
Null degrees of freedom: 130954
Residual degrees of freedom: 130943
Null deviance: 124374.13927893189
Residual deviance: 106607.73758484815
AIC: 106631.73758484815
AUC: 0.7734040182726795
Gini: 0.5468080365453589
Confusion Matrix (Act/Pred) for max f1 @ threshold = 0.21708051286464972:


       0      1      Error    Rate
-----  -----  -----  -------  ------------------
0      83212  23869  0.2229   (23869.0/107081.0)
1      9376   14498  0.3927   (9376.0/23874.0)
Total  92588  38367  0.2539   (33245.0/130955.0)


Maximum Metrics: Maximum metrics at their respective thresholds



metric                        threshold    value     idx
----------------------------  -----------  --------  -----
max f1                        0.217081     0.465867  238
max f2                        0.141019     0.606743  309
max f0point5                  0.333843     0.456869  155
max accuracy                  0.439085     0.82901   98
max precision                 0.800677     1         0
max recall                    0.0676438    1         394
max specificity               0.800677     1         0
max absolute_mcc              0.233584     0.3269    224
max min_per_class_accuracy    0.185566     0.699396  265
max mean_per_class_accuracy   0.176134     0.700621  274


Gains/Lift Table: Avg response rate: 18.23 %
```

	group	cumulative_data_fraction	lower_threshold	lift	cumulative_lift	respons
	1	0.0100034	0.58099	4.22909	4.22909	0.77099
	2	0.0200069	0.535031	3.48795	3.85852	0.63587
	3	0.0300027	0.500774	3.19729	3.63822	0.58288
	4	0.0400061	0.471843	2.94361	3.46454	0.53664
	5	0.0500019	0.448387	2.87044	3.34577	0.5233
	6	0.100004	0.362304	2.44441	2.89509	0.44563
	7	0.150006	0.30793	1.98786	2.59268	0.36240
	8	0.2	0.268446	1.71168	2.37246	0.31205
	9	0.300004	0.213201	1.41027	2.05172	0.25710
	10	0.4	0.176563	1.12177	1.81924	0.20450
	11	0.500004	0.150314	0.874558	1.6303	0.15943
	12	0.6	0.130034	0.676913	1.47141	0.12340
	13	0.699996	0.112956	0.495956	1.33206	0.09041
	14	0.8	0.0978849	0.392043	1.21455	0.07147
	15	0.899996	0.0824531	0.219494	1.104	0.04001
	16	1	0.0597755	0.064084	1	0.01168

ModelMetricsBinomialGLM: stackedensemble
** Reported on validation data. **

MSE: 0.13813682407268132
RMSE: 0.37166762580655494
LogLoss: 0.4379927029517987
Null degrees of freedom: 33031
Residual degrees of freedom: 33020
Null deviance: 31732.354674481932
Residual deviance: 28935.54992780763
AIC: 28959.54992780763
AUC: 0.7122628231156057
Gini: 0.4245256462312115
Confusion Matrix (Act/Pred) for max f1 @ threshold = 0.18933454821029921:

	0	1	Error	Rate
0	18652	8238	0.3064	(8238.0/26890.0)
1	2386	3756	0.3885	(2386.0/6142.0)
Total	21038	11994	0.3216	(10624.0/33032.0)

Maximum Metrics: Maximum metrics at their respective thresholds

metric	threshold	value	idx
max f1	0.189335	0.414204	256
max f2	0.119741	0.573853	330
max f0point5	0.312996	0.381781	160
max accuracy	0.601024	0.816057	29
max precision	0.766658	1	0
max recall	0.064008	1	398
max specificity	0.766658	1	0
max absolute_mcc	0.209272	0.247569	238
max min_per_class_accuracy	0.176285	0.652556	268
max mean_per_class_accuracy	0.157733	0.654789	286

Gains/Lift Table: Avg response rate: 18.59 %

	group	cumulative_data_fraction	lower_threshold	lift	cumulative_lift	respons
	1	0.0100206	0.578369	3.13584	3.13584	0.58308
	2	0.0200109	0.529317	2.57495	2.85582	0.47878
	3	0.0300012	0.495518	2.41198	2.70802	0.44848
	4	0.0400218	0.468825	2.48593	2.65241	0.46223
	5	0.0500121	0.44556	2.21641	2.56532	0.41212
	6	0.100024	0.361287	2.06072	2.31302	0.38317
	7	0.150006	0.306878	1.78508	2.13711	0.33192
	8	0.200018	0.267574	1.57565	1.99673	0.29297
	9	0.300012	0.213245	1.37586	1.78979	0.25582
	10	0.400006	0.176494	1.12999	1.62485	0.21011
	11	0.5	0.149726	0.996478	1.49919	0.18528
	12	0.599994	0.129639	0.758756	1.37579	0.14108
	13	0.699988	0.113067	0.628498	1.26904	0.11686
	14	0.799982	0.0982229	0.525919	1.17615	0.09778
	15	0.899976	0.0826149	0.398917	1.08979	0.07417
	16	1	0.0605773	0.192073	1	0.03571

ModelMetricsBinomialGLM: stackedensemble
** Reported on cross-validation data. **

MSE: 0.1369765435568584
RMSE: 0.3701034227845757
LogLoss: 0.4355309009590542

Null degrees of freedom: 130954
Residual degrees of freedom: 130943
Null deviance: 124375.46514588114
Residual deviance: 114069.89827018589
AIC: 114093.89827018589
AUC: 0.7071203432205223
Gini: 0.4142406864410446
Confusion Matrix (Act/Pred) for max f1 @ threshold = 0.17843972441369682:

	0	1	Error	Rate
0	73561	33520	0.313	(33520.0/107081.0)
1	9215	14659	0.386	(9215.0/23874.0)
Total	82776	48179	0.3263	(42735.0/130955.0)

Maximum Metrics: Maximum metrics at their respective thresholds

metric	threshold	value	idx
max f1	0.17844	0.406895	261
max f2	0.113675	0.566567	334
max f0point5	0.287994	0.370805	176
max accuracy	0.566359	0.818457	36
max precision	0.785924	1	0
max recall	0.062087	1	399
max specificity	0.785924	1	0
max absolute_mcc	0.181598	0.241225	258
max min_per_class_accuracy	0.167173	0.649882	272
max mean_per_class_accuracy	0.161654	0.651283	278

Gains/Lift Table: Avg response rate: 18.23 %

	group	cumulative_data_fraction	lower_threshold	lift	cumulative_lift	respons
	1	0.0100034	0.560981	2.93943	2.93943	0.53587
	2	0.0200069	0.513676	2.60445	2.77194	0.47480
	3	0.0300027	0.480217	2.43463	2.65956	0.44385
	4	0.0400061	0.454197	2.29878	2.56935	0.41908
	5	0.0500019	0.430441	2.3592	2.52734	0.43009
	6	0.100004	0.348992	2.06744	2.29739	0.37690
	7	0.150006	0.295739	1.75833	2.1177	0.32055
	8	0.2	0.25691	1.58266	1.98396	0.28852

9	0.300004	0.203217	1.36796	1.77862	0.24938
10	0.4	0.167994	1.12847	1.61609	0.20572
11	0.500004	0.142859	0.930265	1.47892	0.16959
12	0.6	0.123886	0.814306	1.36816	0.14845
13	0.699996	0.108111	0.658901	1.26684	0.12012
14	0.8	0.0941428	0.518536	1.1733	0.09453
15	0.899996	0.0801775	0.394587	1.08677	0.07193
16	1	0.0597666	0.219058	1	0.03993

Out[80]:

```
In [81]: aml.leader.algo
```

Out[81]: 'stackedensemble'

```
In [82]: dir(aml.leader)
```

```
Out[82]: ['F0point5',
          'F1',
          'F2',
          '__class__',
          '__delattr__',
          '__dict__',
          '__dir__',
          '__doc__',
          '__eq__',
          '__format__',
          '__ge__',
          '__getattr__',
          '__getattribute__',
          '__gt__',
          '__hash__',
          '__init__',
          '__init_subclass__',
          '__le__',
          '__lt__',
          '__module__',
          '__ne__',
          '__new__',
          '__reduce__',
          '__reduce_ex__',
          '__repr__',
          '__setattr__',
          '__sizeof__',
          '__str__',
```

```
'__subclasshook__',
'__weakref__',
'_bc',
'_bcin',
'_check_targets',
'_compute_algo',
'_estimator_type',
'_future',
'_get_metrics',
'_have_mojo',
'_have_pojo',
'_id',
'_is_xvalidated',
'_job',
'_keyify_if_h2oframe',
'_metrics_class',
'_model_json',
'_parms',
'_plot',
'_requires_training_frame',
'_resolve_model',
'_verify_training_frame_params',
'_xval_keys',
'accuracy',
'actual_params',
'aic',
'algo',
'auc',
'base_models',
'biases',
'catoffsets',
'coef',
'coef_norm',
'confusion_matrix',
'cross_validation_fold_assignment',
'cross_validation_holdout_predictions',
'cross_validation_metrics_summary',
'cross_validation_models',
'cross_validation_predictions',
'deepfeatures',
'default_params',
'download_mojo',
'download_pojo',
'error',
'fallout',
'find_idx_by_threshold',
'find_threshold_by_max_metric',
'fit',
```

```
'fnr',
'fpr',
'full_parameters',
'gains_lift',
'get_params',
'get_xval_models',
'gini',
'have_mojo',
'have_pojo',
'is_cross_validated',
'join',
'keep_levelone_frame',
'levelone_frame_id',
'logloss',
'mae',
'max_per_class_error',
'mcc',
'mean_per_class_error',
'mean_residual_deviance',
'metalearner',
'metalearner_algorithm',
'metalearner_fold_assignment',
'metalearner_fold_column',
'metalearner_nfolds',
'metalearner_params',
'metric',
'missrate',
'mixin',
'model_id',
'model_performance',
'mse',
'normmul',
'normsub',
'null_degrees_of_freedom',
'null_deviance',
'params',
'parms',
'partial_plot',
'plot',
'pprint_coef',
'precision',
'predict',
'predict_leaf_node_assignment',
'r2',
'recall',
'residual_degrees_of_freedom',
'residual_deviance',
'respmul',
```

```
 'response_column',
 'respsub',
 'rmse',
 'rmsle',
 'roc',
 'rotation',
 'save_model_details',
 'save_mojo',
 'score_history',
 'scoring_history',
 'seed',
 'sensitivity',
 'set_params',
 'show',
 'specificity',
 'start',
 'std_coef_plot',
 'summary',
 'tnr',
 'tpr',
 'train',
 'training_frame',
 'type',
 'validation_frame',
 'varimp',
 'varimp_plot',
 'weights',
 'xval_keys',
 'xvals']
```

1.8 Ensemble Exploration

To understand how the ensemble works, let's take a peek inside the Stacked Ensemble "All Models" model. The "All Models" ensemble is an ensemble of all of the individual models in the AutoML run. This is often the top performing model on the leaderboard.

```
In [83]: aml_leaderboard_df=aml.leaderboard.as_data_frame()
         aml_leaderboard_df
```

Out[83]:

	model_id	auc	logloss
0	StackedEnsemble_AllModels_0_AutoML_20181120_15...	0.707120	0.435531
1	StackedEnsemble_AllModels_0_AutoML_20181120_15...	0.706222	0.435883
2	StackedEnsemble_BestOfFamily_0_AutoML_20181120...	0.705212	0.436220
3	GBM_grid_0_AutoML_20181120_154205_model_0	0.703839	0.435055
4	GBM_grid_0_AutoML_20181120_153506_model_0	0.703216	0.435227
5	GBM_grid_0_AutoML_20181120_153506_model_1	0.702313	0.435862
6	GBM_grid_0_AutoML_20181120_154205_model_1	0.702177	0.435943
7	GBM_grid_0_AutoML_20181120_153506_model_2	0.700280	0.437029

```
8           GBM_grid_0_AutoML_20181120_154205_model_2  0.699971  0.437370
9               DeepLearning_0_AutoML_20181120_154205  0.699148  0.439850
10          GLM_grid_0_AutoML_20181120_154205_model_0  0.697503  0.438455
11          GLM_grid_0_AutoML_20181120_153506_model_0  0.697503  0.438455
12              DeepLearning_0_AutoML_20181120_153506  0.696297  0.439618
13          GBM_grid_0_AutoML_20181120_154205_model_4  0.694225  0.443191
14          GBM_grid_0_AutoML_20181120_153506_model_4  0.694205  0.442884
15          GBM_grid_0_AutoML_20181120_153506_model_3  0.690842  0.443542
16          GBM_grid_0_AutoML_20181120_154205_model_3  0.689891  0.443962
17                      XRT_0_AutoML_20181120_154205  0.688110  0.471744
18                      XRT_0_AutoML_20181120_153506  0.687225  0.472021
19                      DRF_0_AutoML_20181120_154205  0.684609  0.479048
20                      DRF_0_AutoML_20181120_153506  0.684478  0.479350
21                      DRF_0_AutoML_20181120_153207  0.683488  0.479453

    mean_per_class_error      rmse       mse
0               0.349509  0.370103  0.136977
1               0.351685  0.370262  0.137094
2               0.350829  0.370407  0.137202
3               0.353507  0.370225  0.137067
4               0.352786  0.370273  0.137102
5               0.353887  0.370509  0.137277
6               0.354178  0.370554  0.137310
7               0.354784  0.370872  0.137546
8               0.354552  0.371075  0.137696
9               0.355085  0.372547  0.138791
10              0.354668  0.371441  0.137968
11              0.354668  0.371441  0.137968
12              0.357903  0.371964  0.138357
13              0.361761  0.372926  0.139073
14              0.358243  0.372862  0.139026
15              0.362873  0.373139  0.139233
16              0.363898  0.373311  0.139361
17              0.364619  0.382286  0.146143
18              0.365550  0.382442  0.146262
19              0.366846  0.383622  0.147166
20              0.367447  0.383614  0.147160
21              0.368185  0.383608  0.147155
```

1.9 Getting models

Individul models can ne found through a search of the leader board or directly by the name.

```
In [84]: # Get model ids for all models in the AutoML Leaderboard
         model_ids = list(aml.leaderboard['model_id'].as_data_frame().iloc[:,0])
         # Get the "All Models" Stacked Ensemble model
         se = h2o.get_model([mid for mid in model_ids if "StackedEnsemble_AllModels" in mid][0])
         # Get the Stacked Ensemble metalearner model
```

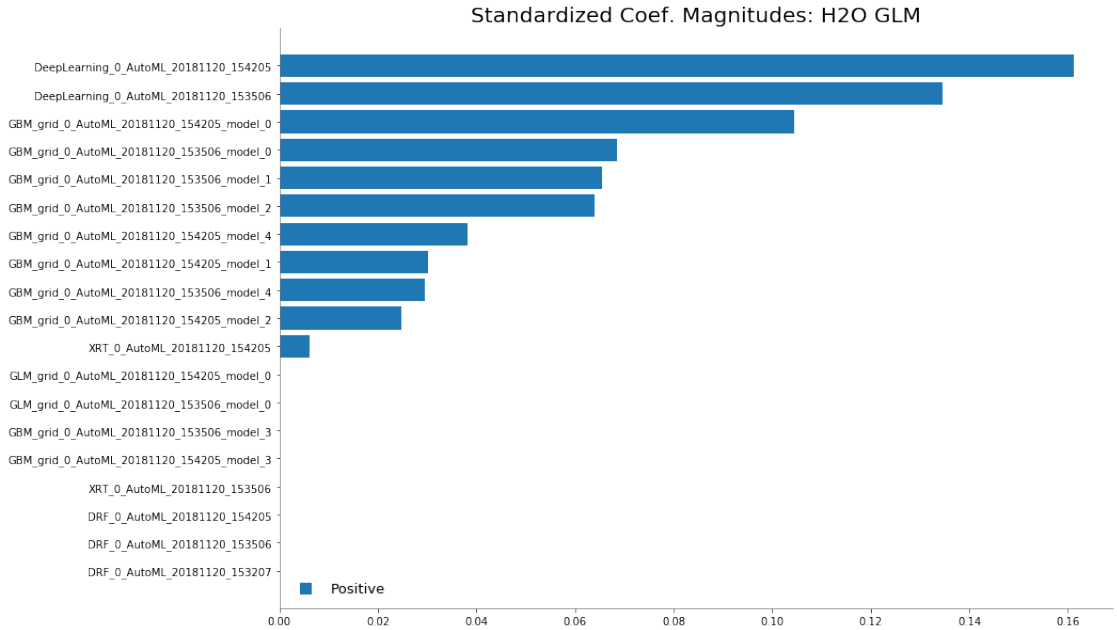
```
        metalearner = h2o.get_model(aml.leader.metalearner()['name'])
```

```
In [85]: metalearner.coef_norm()
```

```
Out[85]: {'DRF_0_AutoML_20181120_153207': 0.0,
          'DRF_0_AutoML_20181120_153506': 0.0,
          'DRF_0_AutoML_20181120_154205': 0.0,
          'DeepLearning_0_AutoML_20181120_153506': 0.1344838464922838,
          'DeepLearning_0_AutoML_20181120_154205': 0.16114849497175607,
          'GBM_grid_0_AutoML_20181120_153506_model_0': 0.06857343556357838,
          'GBM_grid_0_AutoML_20181120_153506_model_1': 0.06556630998879215,
          'GBM_grid_0_AutoML_20181120_153506_model_2': 0.06399896524231677,
          'GBM_grid_0_AutoML_20181120_153506_model_3': 0.0,
          'GBM_grid_0_AutoML_20181120_153506_model_4': 0.029475852258499607,
          'GBM_grid_0_AutoML_20181120_154205_model_0': 0.10436142902876248,
          'GBM_grid_0_AutoML_20181120_154205_model_1': 0.030089245243887962,
          'GBM_grid_0_AutoML_20181120_154205_model_2': 0.024832084306272362,
          'GBM_grid_0_AutoML_20181120_154205_model_3': 0.0,
          'GBM_grid_0_AutoML_20181120_154205_model_4': 0.03820562737398706,
          'GLM_grid_0_AutoML_20181120_153506_model_0': 0.0,
          'GLM_grid_0_AutoML_20181120_154205_model_0': 0.0,
          'Intercept': -1.648080783111384,
          'XRT_0_AutoML_20181120_153506': 0.0,
          'XRT_0_AutoML_20181120_154205': 0.006042541276377857}
```

```
In [86]: %matplotlib inline
         metalearner.std_coef_plot()
```

```
/Users/bear/anaconda/lib/python3.6/site-packages/matplotlib/cbook/deprecation.py:107: Matplotlib
  warnings.warn(message, mplDeprecation, stacklevel=1)
```



Getting a model directly by name

```
In [87]: aml_leaderboard_df.head()
```

Out[87]:

	model_id	auc	logloss	mean_per_class_error	rmse	mse
0	StackedEnsemble_AllModels_0_AutoML_20181120_15...	0.707120	0.435531	0.349509	0.370103	0.136977
1	StackedEnsemble_AllModels_0_AutoML_20181120_15...	0.706222	0.435883	0.351685	0.370262	0.137094
2	StackedEnsemble_BestOfFamily_0_AutoML_20181120...	0.705212	0.436220	0.350829	0.370407	0.137202
3	GBM_grid_0_AutoML_20181120_154205_model_0	0.703839	0.435055	0.353507	0.370225	0.137067
4	GBM_grid_0_AutoML_20181120_153506_model_0	0.703216	0.435227	0.352786	0.370273	0.137102

```
In [88]: m_id=''
         for model in aml_leaderboard_df['model_id']:
             if 'StackedEnsemble' not in model:
               print (model)
               if m_id=='':
                     m_id=model
         print ("model_id ", m_id)
```

```
GBM_grid_0_AutoML_20181120_154205_model_0
GBM_grid_0_AutoML_20181120_153506_model_0
GBM_grid_0_AutoML_20181120_153506_model_1
GBM_grid_0_AutoML_20181120_154205_model_1
GBM_grid_0_AutoML_20181120_153506_model_2
GBM_grid_0_AutoML_20181120_154205_model_2
DeepLearning_0_AutoML_20181120_154205
GLM_grid_0_AutoML_20181120_154205_model_0
GLM_grid_0_AutoML_20181120_153506_model_0
DeepLearning_0_AutoML_20181120_153506
GBM_grid_0_AutoML_20181120_154205_model_4
GBM_grid_0_AutoML_20181120_153506_model_4
GBM_grid_0_AutoML_20181120_153506_model_3
GBM_grid_0_AutoML_20181120_154205_model_3
XRT_0_AutoML_20181120_154205
XRT_0_AutoML_20181120_153506
DRF_0_AutoML_20181120_154205
DRF_0_AutoML_20181120_153506
DRF_0_AutoML_20181120_153207
model_id  GBM_grid_0_AutoML_20181120_154205_model_0
```

```
In [89]: non_stacked= h2o.get_model(m_id)
         print (non_stacked.algo)
```

```
gbm
```

```
In [90]: dir(non_stacked)
```

```
Out[90]: ['F0point5',
          'F1',
          'F2',
          '__class__',
          '__delattr__',
          '__dict__',
          '__dir__',
          '__doc__',
          '__eq__',
          '__format__',
          '__ge__',
          '__getattr__',
          '__getattribute__',
          '__gt__',
          '__hash__',
          '__init__',
          '__init_subclass__',
          '__le__',
          '__lt__',
          '__module__',
          '__ne__',
          '__new__',
          '__reduce__',
          '__reduce_ex__',
          '__repr__',
          '__setattr__',
          '__sizeof__',
          '__str__',
          '__subclasshook__',
          '__weakref__',
          '_bc',
          '_bcin',
          '_check_targets',
          '_compute_algo',
          '_estimator_type',
          '_future',
          '_get_metrics',
          '_have_mojo',
          '_have_pojo',
          '_id',
```

```
'_is_xvalidated',
'_job',
'_keyify_if_h2oframe',
'_metrics_class',
'_model_json',
'_parms',
'_plot',
'_requires_training_frame',
'_resolve_model',
'_verify_training_frame_params',
'_xval_keys',
'accuracy',
'actual_params',
'aic',
'algo',
'auc',
'balance_classes',
'biases',
'build_tree_one_node',
'calibrate_model',
'calibration_frame',
'categorical_encoding',
'catoffsets',
'checkpoint',
'class_sampling_factors',
'coef',
'coef_norm',
'col_sample_rate',
'col_sample_rate_change_per_level',
'col_sample_rate_per_tree',
'confusion_matrix',
'cross_validation_fold_assignment',
'cross_validation_holdout_predictions',
'cross_validation_metrics_summary',
'cross_validation_models',
'cross_validation_predictions',
'custom_metric_func',
'deepfeatures',
'default_params',
'distribution',
'download_mojo',
'download_pojo',
'error',
'fallout',
'find_idx_by_threshold',
'find_threshold_by_max_metric',
'fit',
'fnr',
```

```
'fold_assignment',
'fold_column',
'fpr',
'full_parameters',
'gains_lift',
'get_params',
'get_xval_models',
'gini',
'have_mojo',
'have_pojo',
'histogram_type',
'huber_alpha',
'ignore_const_cols',
'ignored_columns',
'is_cross_validated',
'join',
'keep_cross_validation_fold_assignment',
'keep_cross_validation_predictions',
'learn_rate',
'learn_rate_annealing',
'levelone_frame_id',
'logloss',
'mae',
'max_abs_leafnode_pred',
'max_after_balance_size',
'max_confusion_matrix_size',
'max_depth',
'max_hit_ratio_k',
'max_per_class_error',
'max_runtime_secs',
'mcc',
'mean_per_class_error',
'mean_residual_deviance',
'metalearner',
'metric',
'min_rows',
'min_split_improvement',
'missrate',
'mixin',
'model_id',
'model_performance',
'mse',
'nbins',
'nbins_cats',
'nbins_top_level',
'nfolds',
'normmul',
'normsub',
```

```
'ntrees',
'null_degrees_of_freedom',
'null_deviance',
'offset_column',
'params',
'parms',
'partial_plot',
'plot',
'pprint_coef',
'precision',
'pred_noise_bandwidth',
'predict',
'predict_leaf_node_assignment',
'quantile_alpha',
'r2',
'r2_stopping',
'recall',
'residual_degrees_of_freedom',
'residual_deviance',
'respmul',
'response_column',
'respsub',
'rmse',
'rmsle',
'roc',
'rotation',
'sample_rate',
'sample_rate_per_class',
'save_model_details',
'save_mojo',
'score_each_iteration',
'score_history',
'score_tree_interval',
'scoring_history',
'seed',
'sensitivity',
'set_params',
'show',
'specificity',
'start',
'std_coef_plot',
'stopping_metric',
'stopping_rounds',
'stopping_tolerance',
'summary',
'tnr',
'tpr',
'train',
```

```
 'training_frame',
 'tweedie_power',
 'type',
 'validation_frame',
 'varimp',
 'varimp_plot',
 'weights',
 'weights_column',
 'xval_keys',
 'xvals']
```

Note that since this is a pandas dataframe the data can be saved.

The type of exploration depends on the learner. If the learner isn't an ensemble then ensemble exploration doesn't make sense.

Examine the variable importance of the metalearner (combiner) algorithm in the ensemble. This shows us how much each base learner is contributing to the ensemble. The AutoML Stacked Ensembles use the default metalearner algorithm (GLM with non-negative weights), so the variable importance of the metalearner is actually the standardized coefficient magnitudes of the GLM.

1.10 Save Leader Model

There are two ways to save the leader model -- binary format and MOJO format. If you're taking your leader model to production, then we'd suggest the MOJO format since it's optimized for production use.

```
In [91]: h2o.save_model(aml.leader, path = "./models")
```

```
Out[91]: '/Users/bear/Documents/INFO_7390/H2O/models/StackedEnsemble_AllModels_0_AutoML_20181120
```

```
In [92]: aml.leader.download_mojo(path = "./models")
```

```
Out[92]: '/Users/bear/Documents/INFO_7390/H2O/models/StackedEnsemble_AllModels_0_AutoML_20181120
```

1.11 Making predictions

If one wants predictions the user will do this on new data.

Here we are taking 10% of original file just to show the syntax

```
In [93]: # split into training and test for showing how to predict
         train, test = df.split_frame([0.8])
```

1.12 Predict Using Leader Model

If you need to generate predictions on a test set, you can make predictions on the "H2OAutoML" object directly, or on the leader model object.

```
In [94]: pred = aml.predict(test)
         pred.head()
```

```
stackedensemble prediction progress: || 100%
```

Out[94]:

1.13 model_performance()

The standard `model_performance()` method can be applied to the AutoML leader model and a test set to generate an H2O model performance object.

In [95]:
```
perf = aml.leader.model_performance(test)
perf
```

```
ModelMetricsBinomialGLM: stackedensemble
** Reported on test data. **

MSE: 0.13055715415749164
RMSE: 0.36132693527813786
LogLoss: 0.41655140391705325
Null degrees of freedom: 32682
Residual degrees of freedom: 32671
Null deviance: 31504.82333185961
Residual deviance: 27228.299068442102
AIC: 27252.299068442102
AUC: 0.764167884838059
Gini: 0.528335769676118
Confusion Matrix (Act/Pred) for max f1 @ threshold = 0.21295302564978025:

       0      1     Error    Rate
-----  -----  ----  -------  ----------------
0      20323  6247  0.2351   (6247.0/26570.0)
1      2383   3730  0.3898   (2383.0/6113.0)
Total  22706  9977  0.2641   (8630.0/32683.0)

Maximum Metrics: Maximum metrics at their respective thresholds

metric                       threshold    value     idx
---------------------------  -----------  --------  -----
max f1                       0.212953     0.463642  234
max f2                       0.140722     0.60664   306
max f0point5                 0.321952     0.453994  156
max accuracy                 0.426017     0.823517  99
max precision                0.672695     0.888889  9
max recall                   0.067973     1         393
```

```
max specificity              0.755967     0.999962  0
max absolute_mcc             0.229425     0.318063  221
max min_per_class_accuracy   0.185164     0.690823  260
max mean_per_class_accuracy  0.180249     0.693508  265
```

```
Gains/Lift Table: Avg response rate: 18.70 %
```

	group	cumulative_data_fraction	lower_threshold	lift	cumulative_lift	respon
	1	0.0100052	0.579259	3.92402	3.92402	0.7339
	2	0.0200104	0.533039	3.36812	3.64607	0.6299
	3	0.0300156	0.499334	3.07381	3.45532	0.5749
	4	0.0400208	0.472152	2.69776	3.26593	0.5045
	5	0.050026	0.448412	2.94301	3.20134	0.5504
	6	0.100021	0.361291	2.38203	2.79181	0.4455
	7	0.150017	0.307647	1.9992	2.52766	0.3739
	8	0.200012	0.268194	1.71454	2.32441	0.3206
	9	0.300003	0.214777	1.38733	2.01208	0.2594
	10	0.399994	0.17798	1.0994	1.78393	0.2056
	11	0.500015	0.150762	0.888081	1.60473	0.1661
	12	0.600006	0.130132	0.685487	1.45154	0.1282
	13	0.699997	0.11267	0.513707	1.31757	0.0960
	14	0.799988	0.0979212	0.44663	1.20871	0.0835
	15	0.899979	0.0825174	0.248673	1.10205	0.0465
	16	1	0.0605814	0.0817754	1	0.0152

Out[95]:

```
In [96]: dir(perf)
```

```
Out[96]: ['F0point5',
          'F1',
          'F2',
          '__class__',
          '__delattr__',
          '__dict__',
          '__dir__',
          '__doc__',
          '__eq__',
          '__format__',
          '__ge__',
          '__getattr__',
```

```
'__getattribute__',
'__getitem__',
'__gt__',
'__hash__',
'__init__',
'__init_subclass__',
'__le__',
'__lt__',
'__module__',
'__ne__',
'__new__',
'__reduce__',
'__reduce_ex__',
'__repr__',
'__setattr__',
'__sizeof__',
'__str__',
'__subclasshook__',
'__weakref__',
'_algo',
'_bc',
'_bcin',
'_has',
'_metric_json',
'_on_train',
'_on_valid',
'_on_xval',
'accuracy',
'aic',
'auc',
'confusion_matrix',
'custom_metric_name',
'custom_metric_value',
'error',
'fallout',
'find_idx_by_threshold',
'find_threshold_by_max_metric',
'fnr',
'fpr',
'fprs',
'gains_lift',
'gini',
'logloss',
'mae',
'make',
'max_per_class_error',
'mcc',
'mean_per_class_error',
```

```
          'mean_residual_deviance',
          'metric',
          'missrate',
          'mse',
          'nobs',
          'null_degrees_of_freedom',
          'null_deviance',
          'plot',
          'precision',
          'r2',
          'recall',
          'residual_degrees_of_freedom',
          'residual_deviance',
          'rmse',
          'rmsle',
          'sensitivity',
          'show',
          'specificity',
          'tnr',
          'tpr',
          'tprs']
```

```
In [97]: d=perf.confusion_matrix()
         d
```

```
Confusion Matrix (Act/Pred) for max f1 @ threshold = 0.21295302564978025:


       0      1     Error    Rate
-----  -----  ----  -------  ----------------
0      20323  6247  0.2351   (6247.0/26570.0)
1      2383   3730  0.3898   (2383.0/6113.0)
Total  22706  9977  0.2641   (8630.0/32683.0)
```

```
Out[97]:
```

```
In [98]: dir(perf)
```

```
Out[98]: ['F0point5',
          'F1',
          'F2',
          '__class__',
          '__delattr__',
          '__dict__',
          '__dir__',
          '__doc__',
          '__eq__',
          '__format__',
```

```
'__ge__',
'__getattr__',
'__getattribute__',
'__getitem__',
'__gt__',
'__hash__',
'__init__',
'__init_subclass__',
'__le__',
'__lt__',
'__module__',
'__ne__',
'__new__',
'__reduce__',
'__reduce_ex__',
'__repr__',
'__setattr__',
'__sizeof__',
'__str__',
'__subclasshook__',
'__weakref__',
'_algo',
'_bc',
'_bcin',
'_has',
'_metric_json',
'_on_train',
'_on_valid',
'_on_xval',
'accuracy',
'aic',
'auc',
'confusion_matrix',
'custom_metric_name',
'custom_metric_value',
'error',
'fallout',
'find_idx_by_threshold',
'find_threshold_by_max_metric',
'fnr',
'fpr',
'fprs',
'gains_lift',
'gini',
'logloss',
'mae',
'make',
'max_per_class_error',
```

```
          'mcc',
          'mean_per_class_error',
          'mean_residual_deviance',
          'metric',
          'missrate',
          'mse',
          'nobs',
          'null_degrees_of_freedom',
          'null_deviance',
          'plot',
          'precision',
          'r2',
          'recall',
          'residual_degrees_of_freedom',
          'residual_deviance',
          'rmse',
          'rmsle',
          'sensitivity',
          'show',
          'specificity',
          'tnr',
          'tpr',
          'tprs']
```

In R we get plots like:

```
#compute performance
```

perf <- h2o.performance(automl_leader,conv_data.hex) h2o.confusionMatrix(perf) h2o.accuracy(perf) h2o.tpr(perf)

```
In [99]: aml.leader.algo

Out[99]: 'stackedensemble'

In [100]: dir(aml.leader)

Out[100]: ['F0point5',
           'F1',
           'F2',
           '__class__',
           '__delattr__',
           '__dict__',
           '__dir__',
           '__doc__',
           '__eq__',
           '__format__',
           '__ge__',
           '__getattr__',
```

```
'__getattribute__',
'__gt__',
'__hash__',
'__init__',
'__init_subclass__',
'__le__',
'__lt__',
'__module__',
'__ne__',
'__new__',
'__reduce__',
'__reduce_ex__',
'__repr__',
'__setattr__',
'__sizeof__',
'__str__',
'__subclasshook__',
'__weakref__',
'_bc',
'_bcin',
'_check_targets',
'_compute_algo',
'_estimator_type',
'_future',
'_get_metrics',
'_have_mojo',
'_have_pojo',
'_id',
'_is_xvalidated',
'_job',
'_keyify_if_h2oframe',
'_metrics_class',
'_model_json',
'_parms',
'_plot',
'_requires_training_frame',
'_resolve_model',
'_verify_training_frame_params',
'_xval_keys',
'accuracy',
'actual_params',
'aic',
'algo',
'auc',
'base_models',
'biases',
'catoffsets',
'coef',
```

```
'coef_norm',
'confusion_matrix',
'cross_validation_fold_assignment',
'cross_validation_holdout_predictions',
'cross_validation_metrics_summary',
'cross_validation_models',
'cross_validation_predictions',
'deepfeatures',
'default_params',
'download_mojo',
'download_pojo',
'error',
'fallout',
'find_idx_by_threshold',
'find_threshold_by_max_metric',
'fit',
'fnr',
'fpr',
'full_parameters',
'gains_lift',
'get_params',
'get_xval_models',
'gini',
'have_mojo',
'have_pojo',
'is_cross_validated',
'join',
'keep_levelone_frame',
'levelone_frame_id',
'logloss',
'mae',
'max_per_class_error',
'mcc',
'mean_per_class_error',
'mean_residual_deviance',
'metalearner',
'metalearner_algorithm',
'metalearner_fold_assignment',
'metalearner_fold_column',
'metalearner_nfolds',
'metalearner_params',
'metric',
'missrate',
'mixin',
'model_id',
'model_performance',
'mse',
'normmul',
```

```
 'normsub',
 'null_degrees_of_freedom',
 'null_deviance',
 'params',
 'parms',
 'partial_plot',
 'plot',
 'pprint_coef',
 'precision',
 'predict',
 'predict_leaf_node_assignment',
 'r2',
 'recall',
 'residual_degrees_of_freedom',
 'residual_deviance',
 'respmul',
 'response_column',
 'respsub',
 'rmse',
 'rmsle',
 'roc',
 'rotation',
 'save_model_details',
 'save_mojo',
 'score_history',
 'scoring_history',
 'seed',
 'sensitivity',
 'set_params',
 'show',
 'specificity',
 'start',
 'std_coef_plot',
 'summary',
 'tnr',
 'tpr',
 'train',
 'training_frame',
 'type',
 'validation_frame',
 'varimp',
 'varimp_plot',
 'weights',
 'xval_keys',
 'xvals']
```

In [101]:
```
aml.leader.model_performance(test).auc()
```

Out[101]:
```
0.764167884838059
```

```
In [102]: best_perf = aml.leader.model_performance()
          best_perf
```

```
ModelMetricsBinomialGLM: stackedensemble
** Reported on train data. **

MSE: 0.12685990330985406
RMSE: 0.3561739789904002
LogLoss: 0.40703958453227496
Null degrees of freedom: 130954
Residual degrees of freedom: 130943
Null deviance: 124374.13927893189
Residual deviance: 106607.73758484815
AIC: 106631.73758484815
AUC: 0.7734040182726795
Gini: 0.5468080365453589
Confusion Matrix (Act/Pred) for max f1 @ threshold = 0.21708051286464972:
```

	0	1	Error	Rate
0	83212	23869	0.2229	(23869.0/107081.0)
1	9376	14498	0.3927	(9376.0/23874.0)
Total	92588	38367	0.2539	(33245.0/130955.0)

```
Maximum Metrics: Maximum metrics at their respective thresholds
```

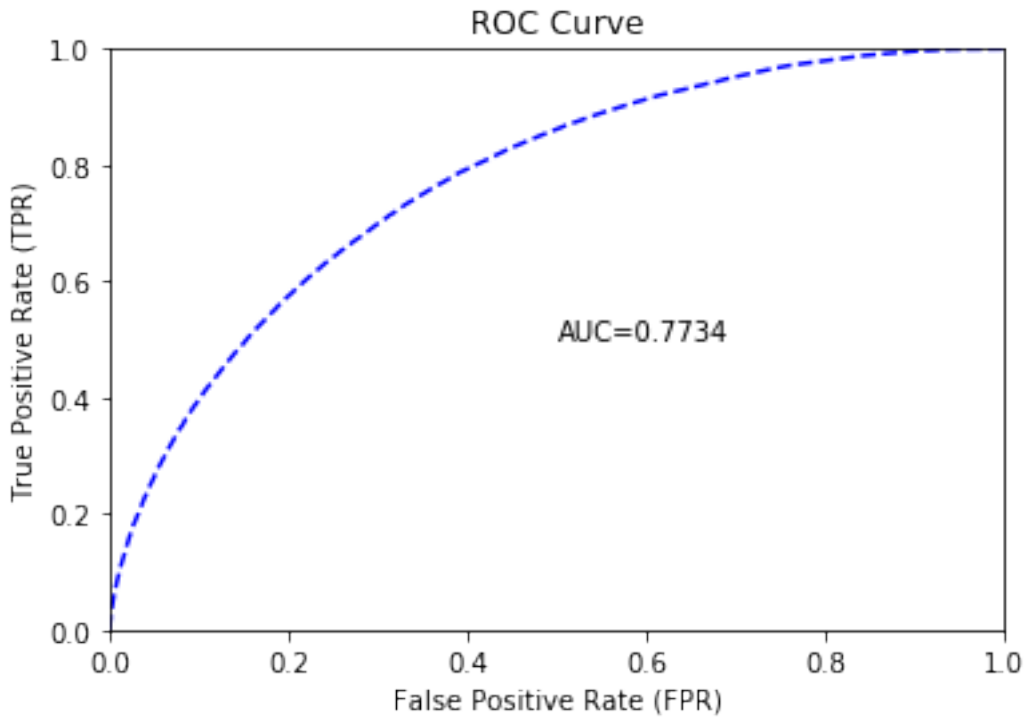
metric	threshold	value	idx
max f1	0.217081	0.465867	238
max f2	0.141019	0.606743	309
max f0point5	0.333843	0.456869	155
max accuracy	0.439085	0.82901	98
max precision	0.800677	1	0
max recall	0.0676438	1	394
max specificity	0.800677	1	0
max absolute_mcc	0.233584	0.3269	224
max min_per_class_accuracy	0.185566	0.699396	265
max mean_per_class_accuracy	0.176134	0.700621	274

```
Gains/Lift Table: Avg response rate: 18.23 %
```

	group	cumulative_data_fraction	lower_threshold	lift	cumulative_lift	respons
	1	0.0100034	0.58099	4.22909	4.22909	0.77099
	2	0.0200069	0.535031	3.48795	3.85852	0.63587
	3	0.0300027	0.500774	3.19729	3.63822	0.58288
	4	0.0400061	0.471843	2.94361	3.46454	0.53664
	5	0.0500019	0.448387	2.87044	3.34577	0.5233
	6	0.100004	0.362304	2.44441	2.89509	0.44563
	7	0.150006	0.30793	1.98786	2.59268	0.36240
	8	0.2	0.268446	1.71168	2.37246	0.31205
	9	0.300004	0.213201	1.41027	2.05172	0.25710
	10	0.4	0.176563	1.12177	1.81924	0.20450
	11	0.500004	0.150314	0.874558	1.6303	0.15943
	12	0.6	0.130034	0.676913	1.47141	0.12340
	13	0.699996	0.112956	0.495956	1.33206	0.09041
	14	0.8	0.0978849	0.392043	1.21455	0.07147
	15	0.899996	0.0824531	0.219494	1.104	0.04001
	16	1	0.0597755	0.064084	1	0.01168

Out[102]:

In [103]: best_perf.plot()



```
In [104]: aml.leader.confusion_matrix()

Confusion Matrix (Act/Pred) for max f1 @ threshold = 0.21708051286464972:


       0      1      Error    Rate
-----  -----  -----  -------  ------------------
0      83212  23869  0.2229   (23869.0/107081.0)
1      9376   14498  0.3927   (9376.0/23874.0)
Total  92588  38367  0.2539   (33245.0/130955.0)


Out[104]:

In [105]: roc=aml.leader.roc()
          roc

Out[105]: ([0.0,
            0.0,
            0.0,
            0.0,
            0.0,
            9.33872489050345e-06,
            9.33872489050345e-06,
            2.8016174671510353e-05,
            4.669362445251725e-05,
            5.6032349343020706e-05,
            9.33872489050345e-05,
            0.00011206469868604141,
            0.0001307421484670483,
            0.0001680970480290621 2,
            0.00023346812226258625,
            0.0002988391964961104,
            0.0003922264454011449,
            0.00044825879474416565,
            0.0005229685938681932,
            0.000579000943211214,
            0.0006256945676637312,
            0.0007097430916782623,
            0.0008031303405832968,
            0.0008871788645978278,
            0.0009805661135028623,
            0.0011019695370794072,
            0.0012046955108749452,
            0.0012794053099989728,
            0.001344776384232497,
```

```
0.0014475023580280348,
0.0015782445064950832,
0.001680970480290621,
0.0018584062532101867,
0.0020358420261297524,
0.002166584174596806,
0.0022879875981733455,
0.0023720361221878767,
0.0025307944453264353,
0.002689552768464994,
0.00285764981649406,
0.003063101764085132,
0.0032405375370046976,
0.003558054183281815,
0.00379152230554401,
0.00400631297802598,
0.004258458550069574,
0.004547959021675181,
0.00489349184262381,
0.005117621239995891,
0.005453815336054015,
0.00579934815700264,
0.006051493729046236,
0.006397026549994864,
0.006686527021600471,
0.00704139856743960,
0.007424286287950243,
0.007536350986636285,
0.00790989998225642,
0.008330142602329078,
0.00870369159794921,
0.009123934218021872,
0.00955351556298503,
0.009917725833714664,
0.010263258654663293,
0.01072085617429796,
0.01085159832276501,
0.01120646986860414,
0.011626712488676796,
0.012168358532325996,
0.012401826654588582,
0.01287810162400426,
0.0132609893445149,
0.013625199615244535,
0.013802635388164,
0.014456346130499342,
0.01477386277677646,
0.015212782846630121,
```

```
0.01572641271560781,
0.01628673620903802,
0.016529543056191108,
0.017127221449183327,
0.017603496418599006,
0.018145142462248205,
0.018313239510277267,
0.01889224045348848,
0.019387192872685165,
0.019816774217648322,
0.020349081536407018,
0.020769324156479675,
0.021254937850785853,
0.02178724516954455,
0.022496988261222813,
0.023225408802682083,
0.02389779699479833,
0.024243329815746957,
0.024803653309177165,
0.025326621903045358,
0.025933639020928084,
0.02640991399034376,
0.027082302182460008,
0.02782006144880978,
0.028408401116911498,
0.029276902531728318,
0.030033339247859097,
0.03077109851420887,
0.031518196505449146,
0.032358681745594454,
0.033068424837272715,
0.03366610323026494,
0.0343758463219432,
0.03509492813851197,
0.035711283981285195,
0.035982107003109796,
0.0368039147934741,
0.037494980435371354,
0.03806464265369206,
0.03876504702047982,
0.039577516085953626,
0.040455356225660946,
0.041286502740915755,
0.04230442375398063,
0.04335969966660752,
0.043929361884928235,
0.04503133142200764,
0.04589983283682446,
```

0.046693624452517256,
0.04754344841755307,
0.048402611107479385,
0.04940185467076325,
0.050298372260251585,
0.05110150260083488,
0.05202603636499473,
0.05288519905492104,
0.05340816764878924,
0.05437939503740159,
0.05537863860068546,
0.056424575788421846,
0.057218367404114644,
0.05832033694119405,
0.05908611238221533,
0.06037485641710481,
0.061803681325351835,
0.06276556998907369,
0.06405431402396317,
0.06506289631213755,
0.06606213987542141,
0.06743493243432541,
0.06785517505439807,
0.06905053184038251,
0.07018985627702394,
0.07082488956957816,
0.07188016548220506,
0.073047506093518,
0.07407476583147338,
0.07521409026811479,
0.07639076960431823,
0.07763282001475519,
0.07832388565665244,
0.07960329096665142,
0.08105079332467945,
0.08187260111504376,
0.08297457065212316,
0.08421662106256012,
0.08515049355161046,
0.0866166733594195,
0.08827896638992912,
0.08962374277416162,
0.09083777700992707,
0.09220123084394057,
0.09357402340284457,
0.09505888066043462,
0.09646902811890065,
0.0979912402760527,

0.09945742008386176,
0.10049401854670763,
0.1017547464069256,
0.10317423259028212,
0.1044723153500621,
0.10593849515787114,
0.10753541711414723,
0.10908564544597081,
0.11065455122757539,
0.11293320010085822,
0.1147542514545064,
0.11632315723611099,
0.1179667728168396,
0.11959171094778719,
0.12121664907873479,
0.12291629700880642,
0.12363537882537519,
0.12525097823143228,
0.12702533596062793,
0.12906117798675767,
0.13066743866792427,
0.1323857640477769,
0.13414144432719158,
0.13600918930529227,
0.13778354703448792,
0.13955790476368357,
0.14108011692083564,
0.14295720062382683,
0.14510510734864263,
0.14703822340097683,
0.1491207590515591,
0.15137139175017042,
0.15327649162783313,
0.15515357533082433,
0.15689991688534846,
0.15868361333943462,
0.16039259999439676,
0.16310083021264277,
0.16551022123439266,
0.16764878923431795,
0.1699647930071628,
0.17145898898964335,
0.17334541141752505,
0.1753812534436548,
0.17756651506803262,
0.17949029239547631,
0.18143274717270105,
0.18338454067481627,

0.18549509250007004,
0.18736283747817073,
0.18926793735583344,
0.1916212960282403,
0.19307813711115884,
0.19529141491020816,
0.1975420476088195,
0.1997646641327593,
0.20199661938158964,
0.20434997805399652,
0.2059469000102726,
0.2083749684818035,
0.21084973057778691,
0.21316573435063177,
0.2157152062457392,
0.21818062961683213,
0.22035655251631941,
0.22290602441142687,
0.22610920704886955,
0.2292376798871882,
0.23176847433251463,
0.23457009179966568,
0.2373903867165977,
0.2403227463322158,
0.2430216378255713,
0.24567383569447426,
0.24723340275118835,
0.2500256814934489,
0.25297671855884796,
0.255740981126437,
0.258430533894902,
0.2614095871349726,
0.2642112046021236,
0.2670221607941652,
0.26999187530934526,
0.2719997011608035,
0.2749974318506551,
0.2784060664356889,
0.28237502451415286,
0.28546614245290947,
0.2884918893174326,
0.2914242489330507,
0.2946647864700554,
0.29756912991100193,
0.30060421550041555,
0.30380739813785823,
0.30697322587573894,
0.3108954903297504,

0.3138838822947115,
0.3171804521810592,
0.3204770220674069,
0.3237922694035356,
0.3275744529841895,
0.3293021170889327,
0.3325800095254994,
0.3360353377349857,
0.3403498286343983,
0.34413201221505213,
0.3475780016996479,
0.3511267171580392,
0.35575872470372893,
0.35985842493065995,
0.3632577207908032,
0.3668064362491945,
0.37149447614422726,
0.3762105322139315,
0.3801608128426145,
0.38395233514815885,
0.3887431010169871,
0.3926280105714366,
0.3966903558988056,
0.40118228257113775,
0.407289808649527,
0.41139884760134854,
0.4155359027278415,
0.419476844631634,
0.42529487023841767,
0.4292544895919911,
0.43314873787133107,
0.4371457121244665,
0.44103062167891594,
0.44697938943416665,
0.45131255778336027,
0.45561770995788237,
0.4595493131367843,
0.464031901084226,
0.4684211017827626,
0.47288501228042323,
0.47761040707501795,
0.48214902737180265,
0.488555392646688,
0.4941212726814281,
0.49882799002624184,
0.5046646930828065,
0.5091192648555766,
0.5149839840868128,

0.5197373950560791,
0.5243507251519878,
0.529048103771911,
0.5334466431953381,
0.5404973804876683,
0.5451200493084675,
0.5500602347755438,
0.5548790168190435,
0.5595577179891857,
0.5667485361548734,
0.5715206245739207,
0.5767129556130406,
0.5829325463901159,
0.5892735405907678,
0.5944565329049971,
0.5996955575685696,
0.6049439209570325,
0.6111821891838888,
0.6159916325024981,
0.6218096581092818,
0.6269926504235112,
0.6324371270346747,
0.6392263800300707,
0.6472483447110131,
0.6564469887281591,
0.6622743530598332,
0.6674013130247196,
0.6740224689720865,
0.6818296429805475,
0.6880585724825132,
0.6919248045871816,
0.6972105228752066,
0.7026176445868081,
0.7106676254424221,
0.7176716691102997,
0.7231161457214632,
0.7317077726207264,
0.7390853652842241,
0.7450808266639273,
0.7504132385764047,
0.7560911833098308,
0.7630298559034749,
0.7687358168115725,
0.7760573771257272,
0.782267629177912,
0.7881603645838198,
0.7940157450901654,
0.7994228668017669,

```
 0.8062401359718344,
 0.8134402928624126,
 0.8217424192900701,
 0.8287838178575098,
 0.8343123429926877,
 0.8395513676562603,
 0.8448370859442852,
 0.8530925187474903,
 0.8600498687909153,
 0.8668671379609828,
 0.8699769333495204,
 0.8768035412444785,
 0.882229340405861,
 0.8880100111130826,
 0.8963214762656307,
 0.9017939690514657,
 0.9071077035141621,
 0.9127669707978072,
 0.9204807575573631,
 0.9262987831641468,
 0.9325930837403461,
 0.937897479478152,
 0.9431551815915055,
 0.9474696724909181,
 0.9537639730671175,
 0.9591617560538284,
 0.967557269730391,
 0.9740476835292909,
 0.9785863038260756,
 0.9817894864635183,
 0.9877382542187689,
 0.9927718269347503,
 0.9959002997730689,
 0.9989634015371541,
 1.0],
[8.377314233056883e-05,
 0.00016754628466113765,
 0.0004607522828181285,
 0.0007539582809751194,
 0.001130937421462679,
 0.0014660299907849543,
 0.0019267822736030828,
 0.002345647985255927,
 0.0029320599815699086,
 0.003769791404875597,
 0.004691295970511854,
 0.005361481109156405,
 0.006408645388288514,
```

0.006995057384602496,
0.007832788807908185,
0.008670520231213872,
0.01026220993549468,
0.011435033928122644,
0.012105219066767195,
0.01357124905755215,
0.0155818044734858,
0.016838401608444332,
0.017885565887576442,
0.018765183882047416,
0.0201474407305018,
0.020943285582642204,
0.022241769288766023,
0.02366591270838569,
0.024419870989360812,
0.026137220407137472,
0.027770796682583563,
0.029613805813856076,
0.031121722375806318,
0.03292284493591355,
0.034346988355533216,
0.03543603920583061,
0.03690206919661557,
0.03866130518555751,
0.0408394068861523,
0.042808075730920665,
0.04532127000083773,
0.047289938845606096,
0.04946804054620089,
0.051604255675630394,
0.05336349166457234,
0.05545782022283656,
0.05771969506576192,
0.06027477590684427,
0.06249476417860434,
0.0647566390215297,
0.06794001843009131,
0.07016000670185138,
0.07288263382759487,
0.07443243696071039,
0.07610789980732177,
0.07803468208092486,
0.07908184636005697,
0.08159504062997402,
0.08406634832872581,
0.08657954259864288,
0.08988858172070034,

0.09185725056546871,
0.0940353522660635,
0.0969674122476334,
0.09943871994638519,
0.10094663650833542,
0.1034179442070872,
0.10563793247884729,
0.1080254670352685,
0.1093658373125576,
0.11238167043645807,
0.11443411242355701,
0.11669598726648236,
0.1179525844014409,
0.12071709809834967,
0.12201558180447349,
0.12486386864371282,
0.1273351763424646,
0.12972271089888582,
0.13148194688782777,
0.133576275446092,
0.1365083354276619,
0.13881209684175252,
0.14015246711904164,
0.14287509424478512,
0.14538828851470217,
0.1485716679232638,
0.15112674876434615,
0.15351428332076736,
0.15657200301583313,
0.1590014241434196,
0.16205914383848538,
0.16482365753539416,
0.16746251151880706,
0.16934740722124486,
0.17144173577950908,
0.17449945547457485,
0.17755717516964062,
0.18053112172237581,
0.18249979056714416,
0.18501298483706125,
0.1867303342548379,
0.1905001256597135,
0.1931389796431264,
0.19590349334003518,
0.1987936667504398,
0.20051101616821648,
0.20356873586328222,
0.2058306107062076,

0.2087207841166122,
0.21161095752701684,
0.2141241517969339,
0.2154226355030577,
0.21848035519812348,
0.2216637346066851,
0.22426070201893272,
0.22681578286001508,
0.22983161598391555,
0.23226103711150206,
0.23494177766608026,
0.23858590935746,
0.24164362905252576,
0.24323531875680657,
0.247130769875178,
0.2504398089972355,
0.2536650749769624,
0.25601072296221833,
0.25860769037446596,
0.26262880120633325,
0.26539331490324203,
0.26773896288849797,
0.27062913629890256,
0.2742732679902823,
0.27599061740805897,
0.2787132445338025,
0.2822317165116863,
0.2849962302085951,
0.2877188573343386,
0.29060903074474326,
0.2926614727318422,
0.2963893775655525,
0.29999162268576696,
0.3032587752366591,
0.3065678143587166,
0.3092066683421295,
0.31247382089302167,
0.316494931724889,
0.31796096171567395,
0.32059981569908685,
0.3239507413923096,
0.3251235653849376,
0.32868392393398677,
0.33216050934070535,
0.33513445589344054,
0.33831783530200216,
0.34137555499706795,
0.3440981821228114,

0.3455223255424311,
0.34891513780681915,
0.3519309709307196,
0.35339700092150456,
0.3570411326128843,
0.3603082851637765,
0.36344977800117284,
0.36633995141157744,
0.37006785624528776,
0.3734606685096758,
0.37647650163357627,
0.3793247884728156,
0.3830526933065259,
0.3859009801457653,
0.3884141744156823,
0.3915137806819134,
0.39457150037697913,
0.3970009215045656,
0.40014241434196196,
0.40345145346401945,
0.40655105973025046,
0.4101533048504649,
0.41333668425902653,
0.4165619502387535,
0.41982910278964564,
0.4239758733350088,
0.4277875513110497,
0.4305939515791237,
0.43352601156069365,
0.43708637010974283,
0.4403116360894697,
0.4433274692133702,
0.4451285917734774,
0.4483538577532043,
0.4515372371617659,
0.4551813688531457,
0.45849040797520313,
0.4621764262377482,
0.4653179190751445,
0.46896205076652425,
0.47273184217139985,
0.4758314484376309,
0.47834464270754795,
0.48165368182960544,
0.4850046075228282,
0.48781100779090225,
0.4919158917651001,
0.49660718773561197,

```
0.5,
0.5036022451202145,
0.5062829856747927,
0.5097595710815113,
0.512817290776577,
0.5172991538912625,
0.5211108318673033,
0.5241685515623691,
0.5276032503979224,
0.529907011812013,
0.5330485046494094,
0.5363994303426322,
0.5394990366088632,
0.5428080757309207,
0.5459914551394823,
0.5491329479768786,
0.5520231213872833,
0.5555834799363324,
0.5590181787718858,
0.5626623104632654,
0.5645472061657033,
0.567604925860769,
0.5713747172656446,
0.5746837563877021,
0.577322610371115,
0.5805478763508419,
0.5828516377649325,
0.5866633157409734,
0.5903912205746837,
0.5935745999832454,
0.5970511853899639,
0.6006115439390132,
0.6037949233475748,
0.6072715087542934,
0.6114182792996565,
0.6151461841333669,
0.6184971098265896,
0.6222250146602999,
0.6263298986344977,
0.6299740303258775,
0.6334925023037614,
0.6377649325626205,
0.6393985088380665,
0.6417860433944877,
0.6446343302337271,
0.6486554410655944,
0.6520482533299824,
0.6548127670268912,
```

0.6582474658624445,
0.6618078244114937,
0.6645723381084024,
0.667588171232303,
0.6707296640696992,
0.674373795761079,
0.6785205663064422,
0.6817877188573344,
0.6856412834045406,
0.6889922090977633,
0.6928038870738041,
0.696615565049845,
0.6997989444584066,
0.7028147775823071,
0.7062075898466952,
0.7108569992460417,
0.7145011309374215,
0.7170980983496691,
0.7205327971852225,
0.7239256094496105,
0.7282399262796347,
0.7305436876937254,
0.7336432939599564,
0.7363659210856999,
0.7409734439138812,
0.7440730501801123,
0.7481779341543101,
0.7512356538493759,
0.7556337438217308,
0.759654854653598,
0.762880120633325,
0.7656027477590684,
0.7690374465946218,
0.7733936499958114,
0.777330987685348,
0.7801373879534221,
0.7847449107816035,
0.7881377230459915,
0.7906928038870739,
0.794253162436123,
0.7990701181201307,
0.8015833123900478,
0.8044315992292871,
0.8074474323531876,
0.8113847700427244,
0.8147775823071124,
0.8177934154310128,
0.8205998156990869,

0.8244114936751278,
0.8280556253665075,
0.8309039122057469,
0.8336265393314903,
0.8360559604590768,
0.8392812264388038,
0.8417525341375555,
0.8448940269749519,
0.8482868392393399,
0.8508419200804223,
0.8546535980564631,
0.8582139566055123,
0.8609365837312558,
0.8644131691379744,
0.8674708888330401,
0.8704867219569407,
0.8732512356538493,
0.87555499706794,
0.8784451704783447,
0.88070704532127,
0.8846024964396415,
0.887031917567228,
0.8895869984083103,
0.8918069866800704,
0.8940269749518305,
0.8978805394990366,
0.8999329814861355,
0.9026137220407138,
0.905587668593449,
0.9085197285750188,
0.9111166959872665,
0.9133785708301918,
0.9155147859596213,
0.9182374130853649,
0.9203736282147943,
0.9225936164865544,
0.9243947390466617,
0.9264052944625953,
0.9289184887325124,
0.9317248890005864,
0.9355365669766273,
0.9374214626790651,
0.9396414509508252,
0.9419033257937506,
0.9445421797771635,
0.9474742397587333,
0.9486470637513613,
0.9507413923096255,

```
0.9526262880120633,
0.9556421211359638,
0.9577783362653933,
0.9601239842506493,
0.9626371785205663,
0.9648990533634917,
0.9664488564966072,
0.9683337521990449,
0.970092988187987,
0.9719778838904247,
0.9729412750272263,
0.9745329647315071,
0.9757476752953004,
0.9769204992879283,
0.978051436709391,
0.97939180698668,
0.9804389712658121,
0.9819887743989277,
0.9837480103878696,
0.9854653598056463,
0.986596297227109,
0.987643461506241,
0.988523079500712,
0.9895702437798441,
0.9903660886319846,
0.9916226857669431,
0.992041551478596,
0.9931724889000586,
0.9937589008963726,
0.9946385188908436,
0.9953505906006535,
0.9959788891681327,
0.9964396414509509,
0.9970260534472648,
0.9976962385859094,
0.9980732177263969,
0.9985339700092151,
0.998827176007372,
0.9991622685766943,
0.9992041551478597,
0.9994135880036861,
0.9996649074306777,
0.9997905671441736,
0.9998743402865041,
0.9999581134288347,
1.0,
1.0,
1.0,
```

```
            1.0,
            1.0,
            1.0])
```

```
In [106]: aml.leader.tnr
```

```
Model Details
=============
H2OStackedEnsembleEstimator :  Stacked Ensemble
Model Key:  StackedEnsemble_AllModels_0_AutoML_20181120_154205
No model summary for this model


ModelMetricsBinomialGLM: stackedensemble
** Reported on train data. **

MSE: 0.12685990330985406
RMSE: 0.3561739789904002
LogLoss: 0.40703958453227496
Null degrees of freedom: 130954
Residual degrees of freedom: 130943
Null deviance: 124374.13927893189
Residual deviance: 106607.73758484815
AIC: 106631.73758484815
AUC: 0.7734040182726795
Gini: 0.5468080365453589
Confusion Matrix (Act/Pred) for max f1 @ threshold = 0.21708051286464972:


       0      1      Error    Rate
-----  -----  -----  -------  ------------------
0      83212  23869  0.2229   (23869.0/107081.0)
1      9376   14498  0.3927   (9376.0/23874.0)
Total  92588  38367  0.2539   (33245.0/130955.0)


Maximum Metrics: Maximum metrics at their respective thresholds



metric                       threshold    value     idx
---------------------------  -----------  --------  -----
max f1                       0.217081     0.465867  238
max f2                       0.141019     0.606743  309
max f0point5                 0.333843     0.456869  155
max accuracy                 0.439085     0.82901   98
max precision                0.800677     1         0
max recall                   0.0676438    1         394
max specificity              0.800677     1         0
```

max absolute_mcc	0.233584	0.3269	224
max min_per_class_accuracy	0.185566	0.699396	265
max mean_per_class_accuracy	0.176134	0.700621	274

Gains/Lift Table: Avg response rate: 18.23 %

	group	cumulative_data_fraction	lower_threshold	lift	cumulative_lift	respons
	1	0.0100034	0.58099	4.22909	4.22909	0.77099
	2	0.0200069	0.535031	3.48795	3.85852	0.63587
	3	0.0300027	0.500774	3.19729	3.63822	0.58288
	4	0.0400061	0.471843	2.94361	3.46454	0.53664
	5	0.0500019	0.448387	2.87044	3.34577	0.5233
	6	0.100004	0.362304	2.44441	2.89509	0.44563
	7	0.150006	0.30793	1.98786	2.59268	0.36240
	8	0.2	0.268446	1.71168	2.37246	0.31205
	9	0.300004	0.213201	1.41027	2.05172	0.25710
	10	0.4	0.176563	1.12177	1.81924	0.20450
	11	0.500004	0.150314	0.874558	1.6303	0.15943
	12	0.6	0.130034	0.676913	1.47141	0.12340
	13	0.699996	0.112956	0.495956	1.33206	0.09041
	14	0.8	0.0978849	0.392043	1.21455	0.07147
	15	0.899996	0.0824531	0.219494	1.104	0.04001
	16	1	0.0597755	0.064084	1	0.01168

ModelMetricsBinomialGLM: stackedensemble
** Reported on validation data. **

MSE: 0.13813682407268132
RMSE: 0.37166762580655494
LogLoss: 0.4379927029517987
Null degrees of freedom: 33031
Residual degrees of freedom: 33020
Null deviance: 31732.354674481932
Residual deviance: 28935.54992780763
AIC: 28959.54992780763
AUC: 0.7122628231156057
Gini: 0.4245256462312115
Confusion Matrix (Act/Pred) for max f1 @ threshold = 0.18933454821029921:

	0	1	Error	Rate

0	18652	8238	0.3064	(8238.0/26890.0)
1	2386	3756	0.3885	(2386.0/6142.0)
Total	21038	11994	0.3216	(10624.0/33032.0)

Maximum Metrics: Maximum metrics at their respective thresholds

metric	threshold	value	idx
max f1	0.189335	0.414204	256
max f2	0.119741	0.573853	330
max f0point5	0.312996	0.381781	160
max accuracy	0.601024	0.816057	29
max precision	0.766658	1	0
max recall	0.064008	1	398
max specificity	0.766658	1	0
max absolute_mcc	0.209272	0.247569	238
max min_per_class_accuracy	0.176285	0.652556	268
max mean_per_class_accuracy	0.157733	0.654789	286

Gains/Lift Table: Avg response rate: 18.59 %

	group	cumulative_data_fraction	lower_threshold	lift	cumulative_lift	respons
	1	0.0100206	0.578369	3.13584	3.13584	0.58308
	2	0.0200109	0.529317	2.57495	2.85582	0.47878
	3	0.0300012	0.495518	2.41198	2.70802	0.44848
	4	0.0400218	0.468825	2.48593	2.65241	0.46223
	5	0.0500121	0.44556	2.21641	2.56532	0.41212
	6	0.100024	0.361287	2.06072	2.31302	0.38317
	7	0.150006	0.306878	1.78508	2.13711	0.33192
	8	0.200018	0.267574	1.57565	1.99673	0.29297
	9	0.300012	0.213245	1.37586	1.78979	0.25582
	10	0.400006	0.176494	1.12999	1.62485	0.21011
	11	0.5	0.149726	0.996478	1.49919	0.18528
	12	0.599994	0.129639	0.758756	1.37579	0.14108
	13	0.699988	0.113067	0.628498	1.26904	0.11686
	14	0.799982	0.0982229	0.525919	1.17615	0.09778
	15	0.899976	0.0826149	0.398917	1.08979	0.07417
	16	1	0.0605773	0.192073	1	0.03571

ModelMetricsBinomialGLM: stackedensemble
** Reported on cross-validation data. **

MSE: 0.1369765435568584
RMSE: 0.3701034227845757
LogLoss: 0.4355309009590542
Null degrees of freedom: 130954
Residual degrees of freedom: 130943
Null deviance: 124375.46514588114
Residual deviance: 114069.89827018589
AIC: 114093.89827018589
AUC: 0.7071203432205223
Gini: 0.4142406864410446
Confusion Matrix (Act/Pred) for max f1 @ threshold = 0.17843972441369682:

	0	1	Error	Rate
0	73561	33520	0.313	(33520.0/107081.0)
1	9215	14659	0.386	(9215.0/23874.0)
Total	82776	48179	0.3263	(42735.0/130955.0)

Maximum Metrics: Maximum metrics at their respective thresholds

metric	threshold	value	idx
max f1	0.17844	0.406895	261
max f2	0.113675	0.566567	334
max f0point5	0.287994	0.370805	176
max accuracy	0.566359	0.818457	36
max precision	0.785924	1	0
max recall	0.062087	1	399
max specificity	0.785924	1	0
max absolute_mcc	0.181598	0.241225	258
max min_per_class_accuracy	0.167173	0.649882	272
max mean_per_class_accuracy	0.161654	0.651283	278

Gains/Lift Table: Avg response rate: 18.23 %

	group	cumulative_data_fraction	lower_threshold	lift	cumulative_lift	respons
	1	0.0100034	0.560981	2.93943	2.93943	0.53587
	2	0.0200069	0.513676	2.60445	2.77194	0.47480

3	0.0300027	0.480217	2.43463	2.65956	0.44385
4	0.0400061	0.454197	2.29878	2.56935	0.41908
5	0.0500019	0.430441	2.3592	2.52734	0.43009
6	0.100004	0.348992	2.06744	2.29739	0.37690
7	0.150006	0.295739	1.75833	2.1177	0.32055
8	0.2	0.25691	1.58266	1.98396	0.28852
9	0.300004	0.203217	1.36796	1.77862	0.24938
10	0.4	0.167994	1.12847	1.61609	0.20572
11	0.500004	0.142859	0.930265	1.47892	0.16959
12	0.6	0.123886	0.814306	1.36816	0.14845
13	0.699996	0.108111	0.658901	1.26684	0.12012
14	0.8	0.0941428	0.518536	1.1733	0.09453
15	0.899996	0.0801775	0.394587	1.08677	0.07193
16	1	0.0597666	0.219058	1	0.03993

Out[106]: <bound method H2OBinomialModel.tnr of >

In [107]: aml.leader.tpr

```
Model Details
=============
H2OStackedEnsembleEstimator :  Stacked Ensemble
Model Key:  StackedEnsemble_AllModels_0_AutoML_20181120_154205
No model summary for this model


ModelMetricsBinomialGLM: stackedensemble
** Reported on train data. **

MSE: 0.12685990330985406
RMSE: 0.3561739789904002
LogLoss: 0.40703958453227496
Null degrees of freedom: 130954
Residual degrees of freedom: 130943
Null deviance: 124374.13927893189
Residual deviance: 106607.73758484815
AIC: 106631.73758484815
AUC: 0.7734040182726795
Gini: 0.5468080365453589
Confusion Matrix (Act/Pred) for max f1 @ threshold = 0.21708051286464972:


       0      1      Error    Rate
-----  -----  -----  -------  ------------------
0      83212  23869  0.2229   (23869.0/107081.0)
```

```
1      9376   14498  0.3927   (9376.0/23874.0)
Total  92588  38367  0.2539   (33245.0/130955.0)
```

Maximum Metrics: Maximum metrics at their respective thresholds

metric	threshold	value	idx
max f1	0.217081	0.465867	238
max f2	0.141019	0.606743	309
max f0point5	0.333843	0.456869	155
max accuracy	0.439085	0.82901	98
max precision	0.800677	1	0
max recall	0.0676438	1	394
max specificity	0.800677	1	0
max absolute_mcc	0.233584	0.3269	224
max min_per_class_accuracy	0.185566	0.699396	265
max mean_per_class_accuracy	0.176134	0.700621	274

Gains/Lift Table: Avg response rate: 18.23 %

	group	cumulative_data_fraction	lower_threshold	lift	cumulative_lift	respons
	1	0.0100034	0.58099	4.22909	4.22909	0.77099
	2	0.0200069	0.535031	3.48795	3.85852	0.63587
	3	0.0300027	0.500774	3.19729	3.63822	0.58288
	4	0.0400061	0.471843	2.94361	3.46454	0.53664
	5	0.0500019	0.448387	2.87044	3.34577	0.5233
	6	0.100004	0.362304	2.44441	2.89509	0.44563
	7	0.150006	0.30793	1.98786	2.59268	0.36240
	8	0.2	0.268446	1.71168	2.37246	0.31205
	9	0.300004	0.213201	1.41027	2.05172	0.25710
	10	0.4	0.176563	1.12177	1.81924	0.20450
	11	0.500004	0.150314	0.874558	1.6303	0.15943
	12	0.6	0.130034	0.676913	1.47141	0.12340
	13	0.699996	0.112956	0.495956	1.33206	0.09041
	14	0.8	0.0978849	0.392043	1.21455	0.07147
	15	0.899996	0.0824531	0.219494	1.104	0.04001
	16	1	0.0597755	0.064084	1	0.01168

ModelMetricsBinomialGLM: stackedensemble

** Reported on validation data. **

MSE: 0.13813682407268132
RMSE: 0.37166762580655494
LogLoss: 0.4379927029517987
Null degrees of freedom: 33031
Residual degrees of freedom: 33020
Null deviance: 31732.354674481932
Residual deviance: 28935.54992780763
AIC: 28959.54992780763
AUC: 0.7122628231156057
Gini: 0.4245256462312115
Confusion Matrix (Act/Pred) for max f1 @ threshold = 0.18933454821029921:

	0	1	Error	Rate
0	18652	8238	0.3064	(8238.0/26890.0)
1	2386	3756	0.3885	(2386.0/6142.0)
Total	21038	11994	0.3216	(10624.0/33032.0)

Maximum Metrics: Maximum metrics at their respective thresholds

metric	threshold	value	idx
max f1	0.189335	0.414204	256
max f2	0.119741	0.573853	330
max f0point5	0.312996	0.381781	160
max accuracy	0.601024	0.816057	29
max precision	0.766658	1	0
max recall	0.064008	1	398
max specificity	0.766658	1	0
max absolute_mcc	0.209272	0.247569	238
max min_per_class_accuracy	0.176285	0.652556	268
max mean_per_class_accuracy	0.157733	0.654789	286

Gains/Lift Table: Avg response rate: 18.59 %

	group	cumulative_data_fraction	lower_threshold	lift	cumulative_lift	respons
	1	0.0100206	0.578369	3.13584	3.13584	0.58308
	2	0.0200109	0.529317	2.57495	2.85582	0.47878
	3	0.0300012	0.495518	2.41198	2.70802	0.44848

4	0.0400218	0.468825	2.48593	2.65241	0.46223
5	0.0500121	0.44556	2.21641	2.56532	0.41212
6	0.100024	0.361287	2.06072	2.31302	0.38317
7	0.150006	0.306878	1.78508	2.13711	0.33192
8	0.200018	0.267574	1.57565	1.99673	0.29297
9	0.300012	0.213245	1.37586	1.78979	0.25582
10	0.400006	0.176494	1.12999	1.62485	0.21011
11	0.5	0.149726	0.996478	1.49919	0.18528
12	0.599994	0.129639	0.758756	1.37579	0.14108
13	0.699988	0.113067	0.628498	1.26904	0.11686
14	0.799982	0.0982229	0.525919	1.17615	0.09778
15	0.899976	0.0826149	0.398917	1.08979	0.07417
16	1	0.0605773	0.192073	1	0.03571

ModelMetricsBinomialGLM: stackedensemble
** Reported on cross-validation data. **

MSE: 0.1369765435568584
RMSE: 0.3701034227845757
LogLoss: 0.4355309009590542
Null degrees of freedom: 130954
Residual degrees of freedom: 130943
Null deviance: 124375.46514588114
Residual deviance: 114069.89827018589
AIC: 114093.89827018589
AUC: 0.7071203432205223
Gini: 0.4142406864410446
Confusion Matrix (Act/Pred) for max f1 @ threshold = 0.17843972441369682:

	0	1	Error	Rate
0	73561	33520	0.313	(33520.0/107081.0)
1	9215	14659	0.386	(9215.0/23874.0)
Total	82776	48179	0.3263	(42735.0/130955.0)

Maximum Metrics: Maximum metrics at their respective thresholds

metric	threshold	value	idx
max f1	0.17844	0.406895	261
max f2	0.113675	0.566567	334
max f0point5	0.287994	0.370805	176

```
max accuracy                  0.566359     0.818457  36
max precision                 0.785924     1         0
max recall                    0.062087     1         399
max specificity               0.785924     1         0
max absolute_mcc              0.181598     0.241225  258
max min_per_class_accuracy    0.167173     0.649882  272
max mean_per_class_accuracy   0.161654     0.651283  278


Gains/Lift Table: Avg response rate: 18.23 %
```

	group	cumulative_data_fraction	lower_threshold	lift	cumulative_lift	respons
	1	0.0100034	0.560981	2.93943	2.93943	0.53587
	2	0.0200069	0.513676	2.60445	2.77194	0.47480
	3	0.0300027	0.480217	2.43463	2.65956	0.44385
	4	0.0400061	0.454197	2.29878	2.56935	0.41908
	5	0.0500019	0.430441	2.3592	2.52734	0.43009
	6	0.100004	0.348992	2.06744	2.29739	0.37690
	7	0.150006	0.295739	1.75833	2.1177	0.32055
	8	0.2	0.25691	1.58266	1.98396	0.28852
	9	0.300004	0.203217	1.36796	1.77862	0.24938
	10	0.4	0.167994	1.12847	1.61609	0.20572
	11	0.500004	0.142859	0.930265	1.47892	0.16959
	12	0.6	0.123886	0.814306	1.36816	0.14845
	13	0.699996	0.108111	0.658901	1.26684	0.12012
	14	0.8	0.0941428	0.518536	1.1733	0.09453
	15	0.899996	0.0801775	0.394587	1.08677	0.07193
	16	1	0.0597666	0.219058	1	0.03993

```
Out[107]: <bound method H2OBinomialModel.tpr of >

In [108]: aml.leader.weights

Model Details
=============
H2OStackedEnsembleEstimator :  Stacked Ensemble
Model Key:  StackedEnsemble_AllModels_0_AutoML_20181120_154205
No model summary for this model


ModelMetricsBinomialGLM: stackedensemble
** Reported on train data. **
```

```
MSE: 0.12685990330985406
RMSE: 0.3561739789904002
LogLoss: 0.40703958453227496
Null degrees of freedom: 130954
Residual degrees of freedom: 130943
Null deviance: 124374.13927893189
Residual deviance: 106607.73758484815
AIC: 106631.73758484815
AUC: 0.7734040182726795
Gini: 0.5468080365453589
Confusion Matrix (Act/Pred) for max f1 @ threshold = 0.21708051286464972:
```

	0	1	Error	Rate
0	83212	23869	0.2229	(23869.0/107081.0)
1	9376	14498	0.3927	(9376.0/23874.0)
Total	92588	38367	0.2539	(33245.0/130955.0)

Maximum Metrics: Maximum metrics at their respective thresholds

metric	threshold	value	idx
max f1	0.217081	0.465867	238
max f2	0.141019	0.606743	309
max f0point5	0.333843	0.456869	155
max accuracy	0.439085	0.82901	98
max precision	0.800677	1	0
max recall	0.0676438	1	394
max specificity	0.800677	1	0
max absolute_mcc	0.233584	0.3269	224
max min_per_class_accuracy	0.185566	0.699396	265
max mean_per_class_accuracy	0.176134	0.700621	274

Gains/Lift Table: Avg response rate: 18.23 %

	group	cumulative_data_fraction	lower_threshold	lift	cumulative_lift	respons
	1	0.0100034	0.58099	4.22909	4.22909	0.77099
	2	0.0200069	0.535031	3.48795	3.85852	0.63587
	3	0.0300027	0.500774	3.19729	3.63822	0.58288
	4	0.0400061	0.471843	2.94361	3.46454	0.53664

5	0.0500019	0.448387	2.87044	3.34577	0.5233
6	0.100004	0.362304	2.44441	2.89509	0.44563
7	0.150006	0.30793	1.98786	2.59268	0.36240
8	0.2	0.268446	1.71168	2.37246	0.31205
9	0.300004	0.213201	1.41027	2.05172	0.25710
10	0.4	0.176563	1.12177	1.81924	0.20450
11	0.500004	0.150314	0.874558	1.6303	0.15943
12	0.6	0.130034	0.676913	1.47141	0.12340
13	0.699996	0.112956	0.495956	1.33206	0.09041
14	0.8	0.0978849	0.392043	1.21455	0.07147
15	0.899996	0.0824531	0.219494	1.104	0.04001
16	1	0.0597755	0.064084	1	0.01168

ModelMetricsBinomialGLM: stackedensemble
** Reported on validation data. **

MSE: 0.13813682407268132
RMSE: 0.37166762580655494
LogLoss: 0.4379927029517987
Null degrees of freedom: 33031
Residual degrees of freedom: 33020
Null deviance: 31732.354674481932
Residual deviance: 28935.54992780763
AIC: 28959.54992780763
AUC: 0.7122628231156057
Gini: 0.4245256462312115
Confusion Matrix (Act/Pred) for max f1 @ threshold = 0.18933454821029921:

	0	1	Error	Rate
0	18652	8238	0.3064	(8238.0/26890.0)
1	2386	3756	0.3885	(2386.0/6142.0)
Total	21038	11994	0.3216	(10624.0/33032.0)

Maximum Metrics: Maximum metrics at their respective thresholds

metric	threshold	value	idx
max f1	0.189335	0.414204	256
max f2	0.119741	0.573853	330
max f0point5	0.312996	0.381781	160
max accuracy	0.601024	0.816057	29

max precision	0.766658	1	0
max recall	0.064008	1	398
max specificity	0.766658	1	0
max absolute_mcc	0.209272	0.247569	238
max min_per_class_accuracy	0.176285	0.652556	268
max mean_per_class_accuracy	0.157733	0.654789	286

Gains/Lift Table: Avg response rate: 18.59 %

	group	cumulative_data_fraction	lower_threshold	lift	cumulative_lift	respons
	1	0.0100206	0.578369	3.13584	3.13584	0.58308
	2	0.0200109	0.529317	2.57495	2.85582	0.47878
	3	0.0300012	0.495518	2.41198	2.70802	0.44848
	4	0.0400218	0.468825	2.48593	2.65241	0.46223
	5	0.0500121	0.44556	2.21641	2.56532	0.41212
	6	0.100024	0.361287	2.06072	2.31302	0.38317
	7	0.150006	0.306878	1.78508	2.13711	0.33192
	8	0.200018	0.267574	1.57565	1.99673	0.29297
	9	0.300012	0.213245	1.37586	1.78979	0.25582
	10	0.400006	0.176494	1.12999	1.62485	0.21011
	11	0.5	0.149726	0.996478	1.49919	0.18528
	12	0.599994	0.129639	0.758756	1.37579	0.14108
	13	0.699988	0.113067	0.628498	1.26904	0.11686
	14	0.799982	0.0982229	0.525919	1.17615	0.09778
	15	0.899976	0.0826149	0.398917	1.08979	0.07417
	16	1	0.0605773	0.192073	1	0.03571

ModelMetricsBinomialGLM: stackedensemble
** Reported on cross-validation data. **

MSE: 0.1369765435568584
RMSE: 0.3701034227845757
LogLoss: 0.4355309009590542
Null degrees of freedom: 130954
Residual degrees of freedom: 130943
Null deviance: 124375.46514588114
Residual deviance: 114069.89827018589
AIC: 114093.89827018589
AUC: 0.7071203432205223
Gini: 0.4142406864410446
Confusion Matrix (Act/Pred) for max f1 @ threshold = 0.17843972441369682:

	0	1	Error	Rate
0	73561	33520	0.313	(33520.0/107081.0)
1	9215	14659	0.386	(9215.0/23874.0)
Total	82776	48179	0.3263	(42735.0/130955.0)

Maximum Metrics: Maximum metrics at their respective thresholds

metric	threshold	value	idx
max f1	0.17844	0.406895	261
max f2	0.113675	0.566567	334
max f0point5	0.287994	0.370805	176
max accuracy	0.566359	0.818457	36
max precision	0.785924	1	0
max recall	0.062087	1	399
max specificity	0.785924	1	0
max absolute_mcc	0.181598	0.241225	258
max min_per_class_accuracy	0.167173	0.649882	272
max mean_per_class_accuracy	0.161654	0.651283	278

Gains/Lift Table: Avg response rate: 18.23 %

	group	cumulative_data_fraction	lower_threshold	lift	cumulative_lift	respons
	1	0.0100034	0.560981	2.93943	2.93943	0.53587
	2	0.0200069	0.513676	2.60445	2.77194	0.47480
	3	0.0300027	0.480217	2.43463	2.65956	0.44385
	4	0.0400061	0.454197	2.29878	2.56935	0.41908
	5	0.0500019	0.430441	2.3592	2.52734	0.43009
	6	0.100004	0.348992	2.06744	2.29739	0.37690
	7	0.150006	0.295739	1.75833	2.1177	0.32055
	8	0.2	0.25691	1.58266	1.98396	0.28852
	9	0.300004	0.203217	1.36796	1.77862	0.24938
	10	0.4	0.167994	1.12847	1.61609	0.20572
	11	0.500004	0.142859	0.930265	1.47892	0.16959
	12	0.6	0.123886	0.814306	1.36816	0.14845
	13	0.699996	0.108111	0.658901	1.26684	0.12012
	14	0.8	0.0941428	0.518536	1.1733	0.09453
	15	0.899996	0.0801775	0.394587	1.08677	0.07193
	16	1	0.0597666	0.219058	1	0.03993

```
Out[108]: <bound method ModelBase.weights of >
```

1.13.1 Test Data Sets for Binary Classifier

Some Kaggle Binary classification competitions The idea here is to get a range of datasets to test our H2O binary classification models as well as to understand which approaches work best for binary classification. The hope is to get a single model or set of models that perform well in these competitions as well as logic and tests to dynamically choose the best models and their parameters.

Santander Customer Satisfaction

Facebook Recruiting IV: Human or Robot?

DonorsChoose.org Application Screening Predict whether teachers' project proposals are accepted

Statoil/C-CORE Iceberg Classifier Challenge Ship or iceberg, can you decide from space?

WSDM - KKBox's Churn Prediction Challenge Can you predict when subscribers will churn?

Porto Seguro's Safe Driver Prediction Predict if a driver will file an insurance claim next year.

Porto Seguro's Safe Driver Prediction Predict if a driver will file an insurance claim next year.

Data Science Bowl 2017 Can you improve lung cancer detection?

Random Acts of Pizza Predicting altruism through free pizza

Last update: October 3, 2018