

```
In [1]: import numpy as np
        from scipy import sparse

        import h2o
        from h2o.automl import H2OAutoML
        h2o.init(nthreads = 4,max_mem_size = "12G")
```

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 "11.0.6" 2020-01-14; OpenJDK Runtime Environment (build 11.0.6+10-post-Ubuntu-1ubuntu118.04.1); OpenJDK 64-Bit Server VM (build 11.0.6+10-post-Ubuntu-1ubuntu118.04.1, mixed mode, sharing)

Starting server from /home/alex/miniconda3/envs/h2o/lib/python3.7/site-packages/h2o/backend/bin/h2o.jar

Ice root: /tmp/tmpobw4gjqq

JVM stdout: /tmp/tmpobw4gjqq/h2o\_alex\_started\_from\_python.out

JVM stderr: /tmp/tmpobw4gjqq/h2o\_alex\_started\_from\_python.err

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:	04 secs
H2O cluster timezone:	America/Los_Angeles
H2O data parsing timezone:	UTC
H2O cluster version:	3.28.0.3
H2O cluster version age:	26 days
H2O cluster name:	H2O_from_python_alex_rfazra
H2O cluster total nodes:	1
H2O cluster free memory:	12 Gb
H2O cluster total cores:	6
H2O cluster allowed cores:	4
H2O cluster status:	accepting new members, healthy
H2O connection url:	http://127.0.0.1:54321
H2O connection proxy:	{'http': None, 'https': None}
H2O internal security:	False
H2O API Extensions:	Amazon S3, XGBoost, Algos, AutoML, Core V3, TargetEncoder, Core V4
Python version:	3.7.6 final

```
In [2]: %%time
# lemmatized keywords, sparse, y included, latest text preprocessing with a total of ~13K features, tfidf

X_train = sparse.load_npz("../processed_data/full_lemma_keyword_pipeline_train_sparse.npz")
X_test = sparse.load_npz("../processed_data/full_lemma_keyword_pipeline_test_sparse.npz")

X_train = h2o.H2OFrame(X_train)
X_test = h2o.H2OFrame(X_test)

y = X_train.columns[-1]
x = X_train.columns
x.remove(y)

# Ensure binary target is a factor
X_train[y] = X_train[y].asfactor()

print(X_train.shape)
print(X_test.shape)

aml = H2OAutoML(max_models = 10,
                seed = 1,
                verbosity='debug',
                )
aml.train(x = x, y = y, training_frame = X_train)

# 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(se.metalearner()['name'])

metalearner.coef_norm()
```

```
Parse progress: |████████████████████████████████████████████████████████████████████████████████| 10
0%
Parse progress: |████████████████████████████████████████████████████████████████████████████████| 10
0%
(7613, 13583)
(3263, 13582)
AutoML progress: |
08:28:26.119: Project: AutoML_20200303_82826117
08:28:26.125: Setting stopping tolerance adaptively based on the training frame: 0.011460988720255175
08:28:26.125: Build control seed: 1
08:28:26.137: training frame: Frame key: automl_training_py_3_sid_b026 cols: 13583 rows: 7613 chunks: 1 size: 2026562 checksum: -4424098674846662324
08:28:26.137: validation frame: NULL
08:28:26.137: leaderboard frame: NULL
08:28:26.137: blending frame: NULL
08:28:26.137: response column: C13583
08:28:26.137: fold column: null
08:28:26.137: weights column: null
08:28:26.145: Loading execution steps: [{XGBoost : defaults}, {GLM : defaults}, {DRF : [def_1]}, {GBM : defaults}, {DeepLearning : defaults}, {DRF : [XRT]}, {XGBoost : grids}, {GBM : grids}, {DeepLearning : grids}, {StackedEnsemble : defaults}]
08:28:26.149: AutoML job created: 2020.03.03 08:28:26.117
08:28:26.150: AutoML build started: 2020.03.03 08:28:26.150
08:28:26.154: AutoML: starting XGBoost_1_AutoML_20200303_082826 model training

08:28:26.637: XGBoost_1_AutoML_20200303_082826 [XGBoost def_1] started

████████████████████████████████████████████████████████████████████████████████
08:53:34.774: XGBoost_1_AutoML_20200303_082826 [XGBoost def_1] complete
08:53:34.825: New leader: XGBoost_1_AutoML_20200303_082826, auc: 0.8155178719061653
08:53:34.826: AutoML: starting XGBoost_2_AutoML_20200303_082826 model training
08:53:34.855: XGBoost_2_AutoML_20200303_082826 [XGBoost def_2] started

████████████████████████████████████████████████████████████████████████████████
09:04:56.919: XGBoost_2_AutoML_20200303_082826 [XGBoost def_2] complete
09:04:56.923: AutoML: starting XGBoost_3_AutoML_20200303_082826 model training
09:04:56.957: XGBoost_3_AutoML_20200303_082826 [XGBoost def_3] started

████████████████████████████████████████████████████████████████████████████████
09:37:46.135: XGBoost_3_AutoML_20200303_082826 [XGBoost def_3] complete
09:37:46.141: New leader: XGBoost_3_AutoML_20200303_082826, auc: 0.8226190306873026
09:37:46.142: AutoML: starting GLM_1_AutoML_20200303_082826 model training
09:37:46.156: GLM_1_AutoML_20200303_082826 [GLM def_1] started

09:38:19.162: GLM_1_AutoML_20200303_082826 [GLM def_1] complete
09:38:19.168: New leader: GLM_1_AutoML_20200303_082826, auc: 0.8364690204286769
09:38:19.171: AutoML: starting DRF_1_AutoML_20200303_082826 model training
09:38:19.224: DRF_1_AutoML_20200303_082826 [DRF def_1] started

09:41:36.241: DRF_1_AutoML_20200303_082826 [DRF def_1] complete
09:41:36.250: AutoML: starting GBM_1_AutoML_20200303_082826 model training
09:41:36.283: GBM_1_AutoML_20200303_082826 [GBM def_1] started

09:55:29.564: GBM_1_AutoML_20200303_082826 [GBM def_1] complete
09:55:29.572: AutoML: starting GBM_2_AutoML_20200303_082826 model training
```

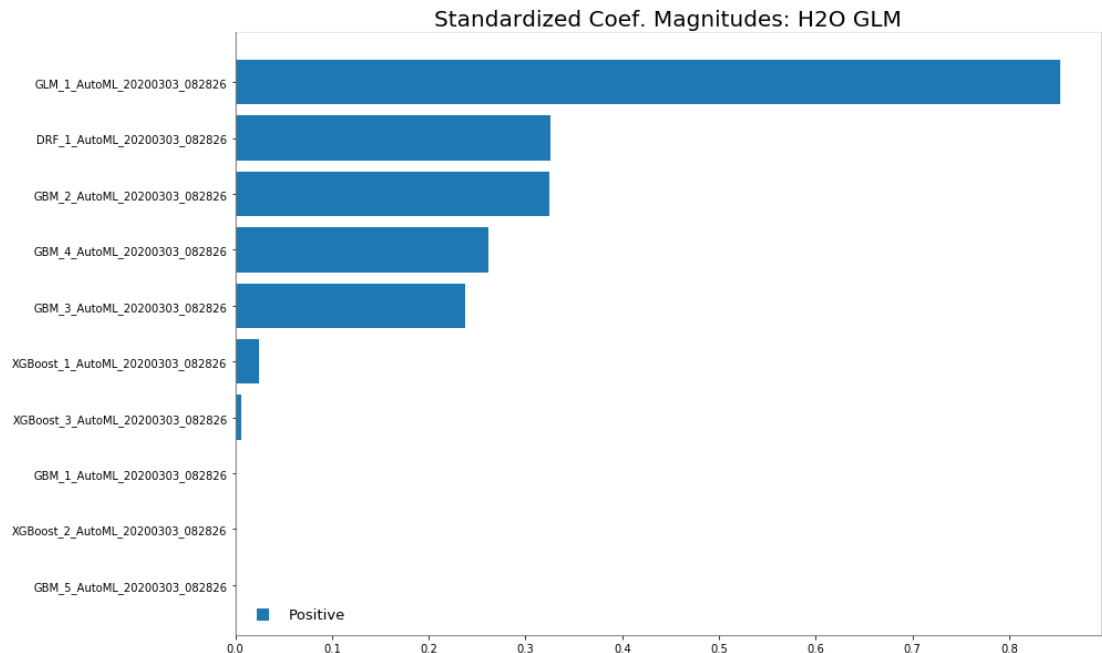
```
Out[2]: {'Intercept': -0.2376143735595635,
'GBM_4_AutoML_20200303_082826': 0.2616483799081072,
'GBM_3_AutoML_20200303_082826': 0.2379515650760144,
'GBM_2_AutoML_20200303_082826': 0.3248733556600385,
'GLM_1_AutoML_20200303_082826': 0.8522008500355571,
'GBM_1_AutoML_20200303_082826': 0.0,
'XGBoost_3_AutoML_20200303_082826': 0.005647509326949814,
'DRF_1_AutoML_20200303_082826': 0.325544544509857,
'XGBoost_1_AutoML_20200303_082826': 0.02457550648857595,
'XGBoost_2_AutoML_20200303_082826': 0.0,
'GBM_5_AutoML_20200303_082826': 0.0}
```

```
In [3]: # 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(se.metalearner()['name'])

print(metalearner.coef_norm())

%matplotlib inline
metalearner.std_coef_plot()
```

```
{'Intercept': -0.2376143735595635, 'GBM_4_AutoML_20200303_082826': 0.2616483799
081072, 'GBM_3_AutoML_20200303_082826': 0.2379515650760144, 'GBM_2_AutoML_20200
303_082826': 0.3248733556600385, 'GLM_1_AutoML_20200303_082826': 0.852200850035
5571, 'GBM_1_AutoML_20200303_082826': 0.0, 'XGBoost_3_AutoML_20200303_082826':
0.005647509326949814, 'DRF_1_AutoML_20200303_082826': 0.325544544509857, 'XGBoo
st_1_AutoML_20200303_082826': 0.02457550648857595, 'XGBoost_2_AutoML_20200303_0
82826': 0.0, 'GBM_5_AutoML_20200303_082826': 0.0}
```



```
In [4]: lb = aml.leaderboard
lb.head()
```

	model_id	auc	logloss	aucpr	mean_per_class_error	
	StackedEnsemble_AllModels_AutoML_20200303_082826	0.859209	0.44469	0.851111	0.216098	0
	StackedEnsemble_BestOfFamily_AutoML_20200303_082826	0.85898	0.445105	0.851193	0.213623	0
	GBM_4_AutoML_20200303_082826	0.840937	0.481935	0.829821	0.238195	0
	GBM_3_AutoML_20200303_082826	0.839382	0.487468	0.829392	0.239423	0
	GBM_2_AutoML_20200303_082826	0.838685	0.48796	0.82923	0.23973	0
	GLM_1_AutoML_20200303_082826	0.836469	0.488063	0.828124	0.229677	0
	GBM_1_AutoML_20200303_082826	0.833188	0.497765	0.820909	0.237652	0
	XGBoost_3_AutoML_20200303_082826	0.822619	0.512913	0.810639	0.257604	0
	DRF_1_AutoML_20200303_082826	0.818065	0.597941	0.814039	0.258379	0
	XGBoost_1_AutoML_20200303_082826	0.815518	0.513327	0.798369	0.265636	0

Out[4]:

```
In [5]: %%time
# Experiment 6 predictions
predictions6 = aml.predict(X_test)
predictions6.shape
```

stackedensemble prediction progress: |██| 10  
0%  
CPU times: user 92.2 ms, sys: 389 µs, total: 92.6 ms  
Wall time: 10.3 s

Out[5]: (3263, 3)

```
In [6]: import pandas as pd
test_df = pd.read_csv("../data/test.csv")
test_id = test_df['id']
test_predictions_df = pd.DataFrame([test_id, pd.Series(predictions6.as_data_frame().iloc[:, 0])]).T
test_predictions_df.columns = ['id', 'target']
test_predictions_df['id'] = test_predictions_df['id'].astype('int32')
test_predictions_df['target'] = test_predictions_df['target'].astype('int32')
```

```
In [7]: test_predictions_df['target'].sum()
```

Out[7]: 1405

```
In [8]: test_predictions_df.to_csv('test_preds_automl_006.csv', index=False)
```

```
In [10]: h2o.save_model(aml.leader, path = "../saved_models/automl_006_bin")
```

Out[10]: '/home/alex/Documents/mlbase/disaster\_tweet\_kaggle/h2o/saved\_models/automl\_006\_bin/StackedEnsemble\_AllModels\_AutoML\_20200303\_082826'

```
In [ ]:
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

```
In [2]: %%time
# Experiment 1
X_train = np.load("../processed_data/full_lemma_keyword_pca_50_pipeline_tf_train_
n_ndarray.npy")
X_test = np.load("../processed_data/full_lemma_keyword_pca_50_pipeline_tf_test_
ndarray.npy")

X_train = h2o.H2OFrame(X_train)
X_test = h2o.H2OFrame(X_test)

y = X_train.columns[-1]
x = X_train.columns
x.remove(y)

aml = H2OAutoML(max_models = 10,
                seed = 1,
                verbosity='debug',
                )
aml.train(x = x, y = y, training_frame = X_train)

# 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(se.metalearner()['name'])

metalearner.coef_norm()
```

```

Parse progress: | 10
0%
Parse progress: | 10
0%
AutoML progress: |
23:27:51.862: Project: AutoML_20200221_232751860
23:27:51.862: Setting stopping tolerance adaptively based on the training frame: 0.011460988720255175
23:27:51.862: Build control seed: 1
23:27:51.863: training frame: Frame key: automl_training_Key_Frame__upload_a64c4a5544f95b80afd19d09cb64459b.hex cols: 242 rows: 7613 chunks: 4 size: 3150109 checksum: -1202237540327242178
23:27:51.863: validation frame: NULL
23:27:51.863: leaderboard frame: NULL
23:27:51.863: blending frame: NULL
23:27:51.863: response column: C242
23:27:51.863: fold column: null
23:27:51.863: weights column: null
23:27:51.869: Loading execution steps: [{XGBoost : defaults}, {GLM : defaults}, {DRF : [def_1]}, {GBM : defaults}, {DeepLearning : defaults}, {DRF : [XRT]}, {XGBoost : grids}, {GBM : grids}, {DeepLearning : grids}, {StackedEnsemble : defaults}]
23:27:51.874: AutoML job created: 2020.02.21 23:27:51.860
23:27:51.875: AutoML build started: 2020.02.21 23:27:51.875
23:27:51.881: AutoML: starting XGBoost_1_AutoML_20200221_232751 model training
23:27:52.18: XGBoost_1_AutoML_20200221_232751 [XGBoost def_1] started

23:28:08.19: XGBoost_1_AutoML_20200221_232751 [XGBoost def_1] complete
23:28:08.27: New leader: XGBoost_1_AutoML_20200221_232751, mean_residual_deviance: 0.44106507474520734
23:28:08.28: AutoML: starting XGBoost_2_AutoML_20200221_232751 model training
23:28:08.28: XGBoost_2_AutoML_20200221_232751 [XGBoost def_2] started

23:28:21.32: XGBoost_2_AutoML_20200221_232751 [XGBoost def_2] complete
23:28:21.36: AutoML: starting XGBoost_3_AutoML_20200221_232751 model training
23:28:21.37: XGBoost_3_AutoML_20200221_232751 [XGBoost def_3] started

23:28:29.38: XGBoost_3_AutoML_20200221_232751 [XGBoost def_3] complete
23:28:29.44: AutoML: starting GLM_1_AutoML_20200221_232751 model training
23:28:29.44: GLM_1_AutoML_20200221_232751 [GLM def_1] started

23:28:30.44: GLM_1_AutoML_20200221_232751 [GLM def_1] complete
23:28:30.46: AutoML: starting DRF_1_AutoML_20200221_232751 model training
23:28:30.46: DRF_1_AutoML_20200221_232751 [DRF def_1] started

23:28:43.48: DRF_1_AutoML_20200221_232751 [DRF def_1] complete
23:28:43.52: AutoML: starting GBM_1_AutoML_20200221_232751 model training
23:28:43.53: GBM_1_AutoML_20200221_232751 [GBM def_1] started

23:28:48.54: GBM_1_AutoML_20200221_232751 [GBM def_1] complete
23:28:48.58: AutoML: starting GBM_2_AutoML_20200221_232751 model training
23:28:48.59: GBM_2_AutoML_20200221_232751 [GBM def_2] started

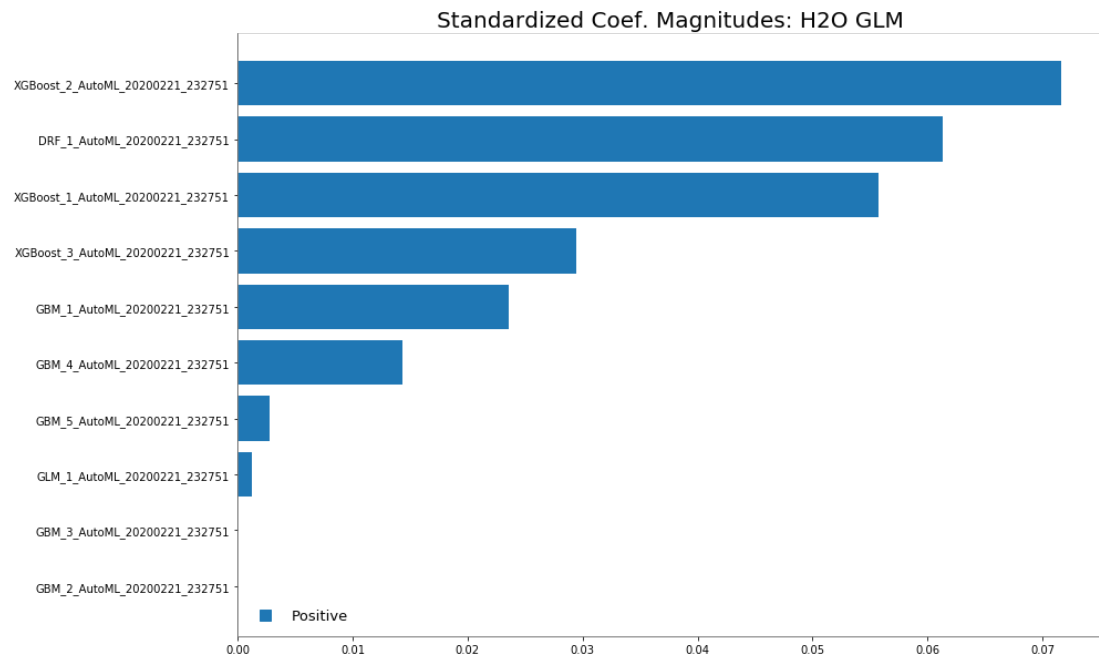
23:28:53.60: GBM_2_AutoML_20200221_232751 [GBM def_2] complete
23:28:53.64: AutoML: starting GBM_3_AutoML_20200221_232751 model training
23:28:53.65: GBM_3_AutoML_20200221_232751 [GBM def_3] started

```



```
Out[2]: {'Intercept': 0.34257191645868984,
'XGBoost_1_AutoML_20200221_232751': 0.05576652257573207,
'DRF_1_AutoML_20200221_232751': 0.061332121090884886,
'XGBoost_3_AutoML_20200221_232751': 0.02941413881503072,
'GBM_4_AutoML_20200221_232751': 0.01428765587910585,
'XGBoost_2_AutoML_20200221_232751': 0.07165427133901207,
'GBM_3_AutoML_20200221_232751': 0.0,
'GBM_1_AutoML_20200221_232751': 0.023564014208765,
'GBM_2_AutoML_20200221_232751': 0.0,
'GBM_5_AutoML_20200221_232751': 0.00280615106051427,
'GLM_1_AutoML_20200221_232751': 0.0012646711768132368}
```

```
In [3]: %matplotlib inline
metalearner.std_coef_plot()
```



```
In [5]: # Experiment 1 Prediction Test
predictions1 = aml.predict(X_test)
predictions1.shape
```

stackedensemble prediction progress: |██| 10  
0%

```
Out[5]: (3263, 1)
```

```
In [7]: %%time
# Experiment 2
X_train = sparse.load_npz("../processed_data/full_lemma_keyword_pca_50_pipeline_
_tt_train_sparse.npz")
X_test = sparse.load_npz("../processed_data/full_lemma_keyword_pca_50_pipeline_
tt_test_sparse.npz")

X_train = h2o.H2OFrame(X_train)
X_test = h2o.H2OFrame(X_test)

y = X_train.columns[-1]
x = X_train.columns
x.remove(y)

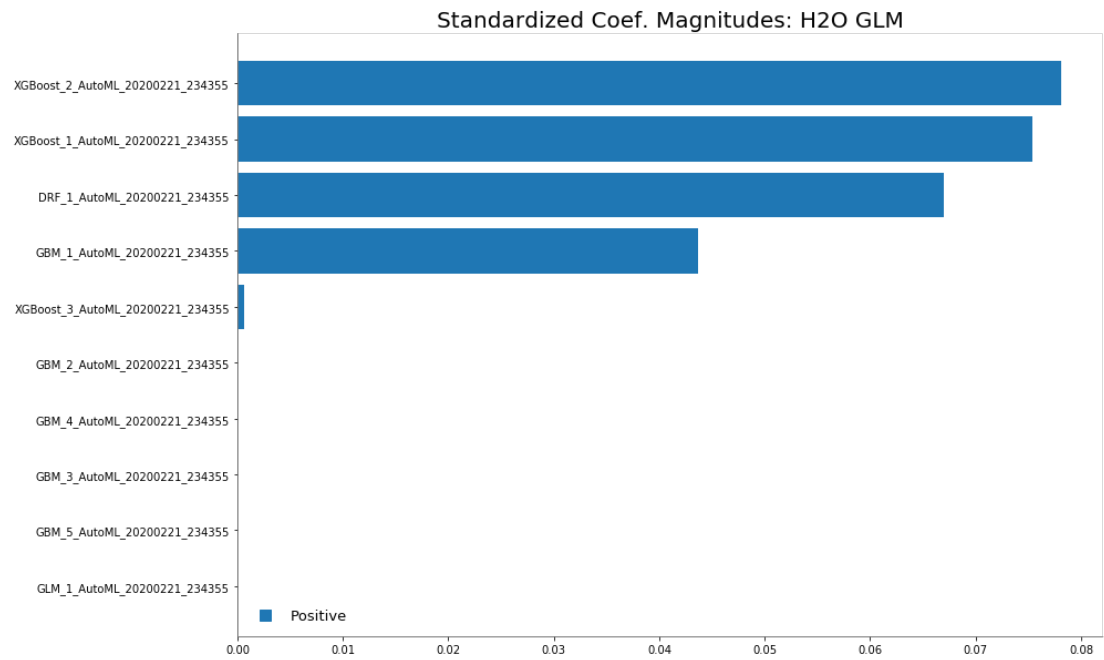
aml = H2OAutoML(max_models = 10,
                seed = 1,
                verbosity='debug',
                )
aml.train(x = x, y = y, training_frame = X_train)

# 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(se.metalearner()['name'])

print(metalearner.coef_norm())

%matplotlib inline
metalearner.std_coef_plot()
```

3/3/20, 11:27 AM



CPU times: user 6.95 s, sys: 156 ms, total: 7.11 s  
Wall time: 1min 31s

```
In [9]: # Experiment 2 Prediction Test
predictions2 = aml.predict(X_test)
predictions2.shape
```

stackedensemble prediction progress: |██| 10  
0%

```
Out[9]: (3263, 1)
```

```
In [2]: %%time
# Experiment 3 - bow pca 50
X_train = sparse.load_npz("../processed_data/full_raw_keyword_bow_pca_50_pipeline_tt_train_sparse.npz")
X_test = sparse.load_npz("../processed_data/full_raw_keyword_bow_pca_50_pipeline_tt_test_sparse.npz")

X_train = h2o.H2OFrame(X_train)
X_test = h2o.H2OFrame(X_test)

y = X_train.columns[-1]
x = X_train.columns
x.remove(y)

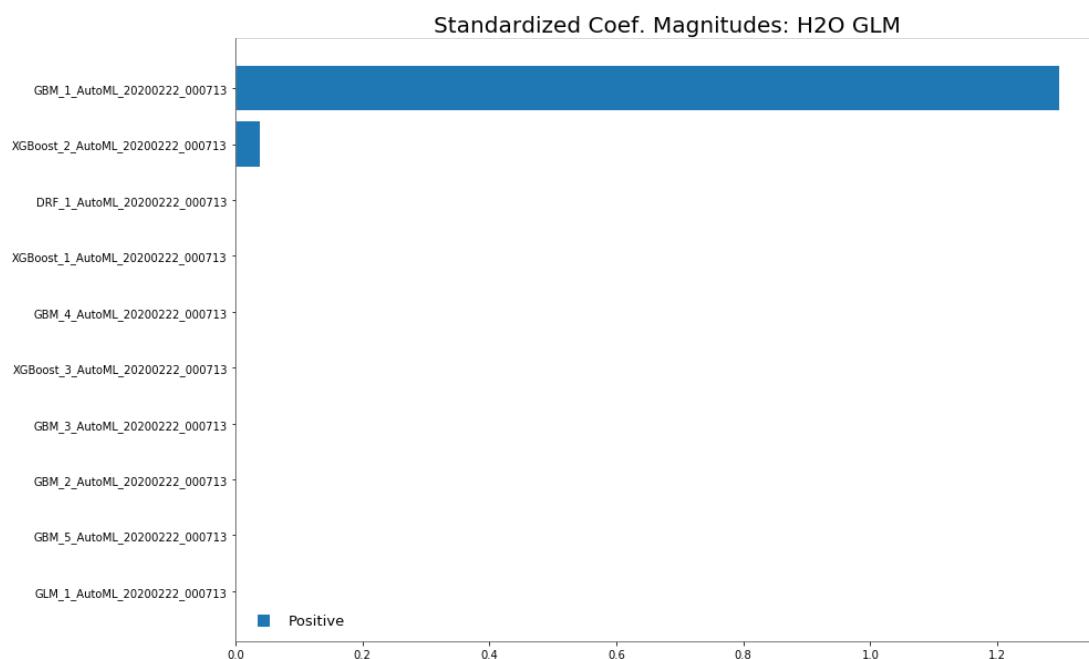
aml = H2OAutoML(max_models = 10,
                seed = 1,
                verbosity='info',
                )
aml.train(x = x, y = y, training_frame = X_train)

# 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(se.metalearner()['name'])

print(metalearner.coef_norm())

%matplotlib inline
metalearner.std_coef_plot()
```

```
Parse progress: | ██████████ | 10  
0%  
Parse progress: | ██████████ | 10  
0%  
AutoML progress: |  
00:07:13.501: Project: AutoML_20200222_00713499  
00:07:13.502: Setting stopping tolerance adaptively based on the training frame:  
e: 0.011460988720255175  
00:07:13.502: Build control seed: 1  
00:07:13.502: training frame: Frame key: automl_training_py_1_sid_a749 cols:  
275 rows: 7613 chunks: 3 size: 3139444 checksum: 5360252508036772992  
00:07:13.502: validation frame: NULL  
00:07:13.502: leaderboard frame: NULL  
00:07:13.502: blending frame: NULL  
00:07:13.502: response column: C275  
00:07:13.502: fold column: null  
00:07:13.502: weights column: null  
00:07:13.510: Loading execution steps: [{XGBoost : defaults}, {GLM : defaults},  
{DRF : [def_1]}, {GBM : defaults}, {DeepLearning : defaults}, {DRF : [XRT]}, {X  
GBoost : grids}, {GBM : grids}, {DeepLearning : grids}, {StackedEnsemble : defa  
ults}]  
00:07:13.515: AutoML job created: 2020.02.22 00:07:13.499  
00:07:13.515: AutoML build started: 2020.02.22 00:07:13.515  
00:07:13.519: AutoML: starting XGBoost_1_AutoML_20200222_000713 model training  
  
00:07:25.673: New leader: XGBoost_1_AutoML_20200222_000713, mean_residual_devia  
nce: 7.412241391589661  
00:07:25.673: AutoML: starting XGBoost_2_AutoML_20200222_000713 model training  
  
00:07:37.681: New leader: XGBoost_2_AutoML_20200222_000713, mean_residual_devia  
nce: 7.29133356164068  
00:07:37.681: AutoML: starting XGBoost_3_AutoML_20200222_000713 model training  
  
00:07:50.690: AutoML: starting GLM_1_AutoML_20200222_000713 model training  
  
00:07:51.692: AutoML: starting DRF_1_AutoML_20200222_000713 model training  
  
00:08:04.697: AutoML: starting GBM_1_AutoML_20200222_000713 model training  
  
00:08:14.702: New leader: GBM_1_AutoML_20200222_000713, mean_residual_deviance:  
6.571425276123931  
00:08:14.703: AutoML: starting GBM_2_AutoML_20200222_000713 model training  
  
00:08:19.706: AutoML: starting GBM_3_AutoML_20200222_000713 model training  
  
00:08:25.708: AutoML: starting GBM_4_AutoML_20200222_000713 model training  
  
00:08:31.714: AutoML: starting GBM_5_AutoML_20200222_000713 model training  
  
00:08:40.722: AutoML: starting StackedEnsemble_BestOfFamily_AutoML_20200222_000  
713 model training  
00:08:41.729: AutoML: starting StackedEnsemble_AllModels_AutoML_20200222_000713  
model training
```



CPU times: user 5.94 s, sys: 166 ms, total: 6.1 s  
Wall time: 1min 33s

```
In [3]: # Experiment 3 Prediction Test
predictions3 = aml.predict(X_test)
predictions3.shape
```

gbm prediction progress: |██| 10  
0%

Out[3]: (3263, 1)

```
In [ ]:
```

```
In [37]: # X_train = sparse.load_npz("../processed_data/full_raw_keyword_bow_pipeline_tt_train_sparse.npz")
X_train = np.load("../full_raw_keyword_bow_pipeline_tt_train_ndarray.npy")
# y_train = np.load("processed_data/raw_keyword_categorical_y_train.npy")
# y_train = h2o.H2OFrame(y_train)
# X_test = sparse.load_npz("../processed_data/full_raw_keyword_bow_pipeline_tt_test_sparse.npz")
X_test = np.load("../full_raw_keyword_bow_pipeline_tt_test_ndarray.npy")

X_train = h2o.H2OFrame(X_train)
X_test = h2o.H2OFrame(X_test)

# combined_train = X_train.cbind(y_train)
```

Parse progress: |██| 10  
0%  
Parse progress: |██| 10  
0%

```
In [41]: X_train.shape
```

Out[41]: (7613, 20338)

```
In [42]: X_test.shape
```

```
Out[42]: (3263, 20337)
```

```
In [43]: y = X_train.columns[-1]
x = X_train.columns
x.remove(y)

# Ensure binary target is a factor
X_train[y] = X_train[y].asfactor()
```

```
In [29]: # y = "C110000"
# x = combined_train.columns
# x.remove(y)
```

## Run AutoML



```
In [45]: aml = H2OAutoML(max_models = 20,
                        seed = 1,
                        verbosity='info',
                        # max_runtime_secs = 30, # For debugging
                        )
aml.train(x = x, y = y, training_frame = X_train)
```

```
AutoML progress: |
13:21:16.253: Project: AutoML_20200222_132116250
13:21:16.265: Setting stopping tolerance adaptively based on the training frame: 0.011460988720255175
13:21:16.265: Build control seed: 1
13:21:16.289: training frame: Frame key: automl_training_py_851_sid_a6a5 cols: 20338 rows: 7613 chunks: 148 size: 240807846 checksum: -8586601818324432240
13:21:16.289: validation frame: NULL
13:21:16.289: leaderboard frame: NULL
13:21:16.289: blending frame: NULL
13:21:16.289: response column: C20338
13:21:16.289: fold column: null
13:21:16.289: weights column: null
13:21:16.398: Loading execution steps: [{XGBoost : defaults}, {GLM : defaults}, {DRF : [def_1]}, {GBM : defaults}, {DeepLearning : defaults}, {DRF : [XRT]}, {XGBoost : grids}, {GBM : grids}, {DeepLearning : grids}, {StackedEnsemble : defaults}]
13:21:16.403: AutoML job created: 2020.02.22 13:21:16.250
13:21:16.404: AutoML build started: 2020.02.22 13:21:16.403
13:21:16.493: AutoML: starting XGBoost_1_AutoML_20200222_132116 model training

13:45:06.455: New leader: XGBoost_1_AutoML_20200222_132116, auc: 0.7996706889586066
13:45:06.455: AutoML: starting XGBoost_2_AutoML_20200222_132116 model training

13:56:47.651: AutoML: starting XGBoost_3_AutoML_20200222_132116 model training

14:31:02.519: New leader: XGBoost_3_AutoML_20200222_132116, auc: 0.8186019372960684
14:31:02.638: AutoML: starting GLM_1_AutoML_20200222_132116 model training

14:38:59.49: New leader: GLM_1_AutoML_20200222_132116, auc: 0.8350472115055452
14:38:59.86: AutoML: starting DRF_1_AutoML_20200222_132116 model training

15:02:37.189: AutoML: starting GBM_1_AutoML_20200222_132116 model training

17:58:57.743: New leader: GBM_1_AutoML_20200222_132116, auc: 0.836954210479401
17:58:57.743: AutoML: starting GBM_2_AutoML_20200222_132116 model training

21:34:55.124: New leader: GBM_2_AutoML_20200222_132116, auc: 0.8401902542069167
21:34:55.124: AutoML: starting GBM_3_AutoML_20200222_132116 model training

01:18:37.896: New leader: GBM_3_AutoML_20200222_132116, auc: 0.8408943113702047
01:18:37.896: AutoML: starting GBM_4_AutoML_20200222_132116 model training

05:50:18.751: New leader: GBM_4_AutoML_20200222_132116, auc: 0.845331677495842
05:50:18.751: AutoML: starting GBM_5_AutoML_20200222_132116 model training

06:43:32.76: AutoML: starting DeepLearning_1_AutoML_20200222_132116 model training
```

```
In [46]: aml.modeling_steps
```

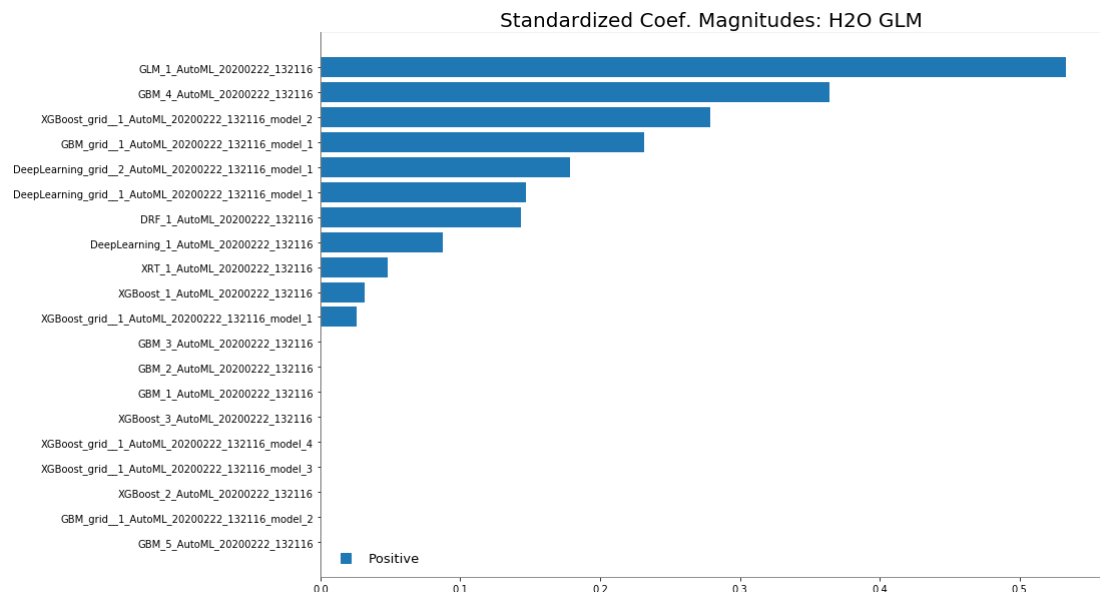
```
Out[46]: [{'name': 'XGBoost',  
  'steps': [{'id': 'def_1', 'weight': 10},  
    {'id': 'def_2', 'weight': 10},  
    {'id': 'def_3', 'weight': 10}]},  
{'name': 'GLM', 'steps': [{'id': 'def_1', 'weight': 10}]},  
{'name': 'DRF', 'steps': [{'id': 'def_1', 'weight': 10}]},  
{'name': 'GBM',  
  'steps': [{'id': 'def_1', 'weight': 10},  
    {'id': 'def_2', 'weight': 10},  
    {'id': 'def_3', 'weight': 10},  
    {'id': 'def_4', 'weight': 10},  
    {'id': 'def_5', 'weight': 10}]},  
{'name': 'DeepLearning', 'steps': [{'id': 'def_1', 'weight': 10}]},  
{'name': 'DRF', 'steps': [{'id': 'XRT', 'weight': 10}]},  
{'name': 'XGBoost', 'steps': [{'id': 'grid_1', 'weight': 100}]},  
{'name': 'GBM', 'steps': [{'id': 'grid_1', 'weight': 60}]},  
{'name': 'DeepLearning',  
  'steps': [{'id': 'grid_1', 'weight': 20}, {'id': 'grid_2', 'weight': 20}]},  
{'name': 'StackedEnsemble',  
  'steps': [{'id': 'best', 'weight': 10}, {'id': 'all', 'weight': 10}]}]
```

```
In [47]: # 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(se.metalearner()['name'])

print(metalearner.coef_norm())

%matplotlib inline
metalearner.std_coef_plot()
```

```
{'Intercept': -0.22084968506644476, 'XGBoost_grid__1_AutoML_20200222_132116_model_2': 0.27850699616898056, 'GBM_4_AutoML_20200222_132116': 0.36384241842698295, 'GBM_grid__1_AutoML_20200222_132116_model_1': 0.23165007332467588, 'GBM_3_AutoML_20200222_132116': 0.0, 'GBM_2_AutoML_20200222_132116': 0.0, 'GBM_1_AutoML_20200222_132116': 0.0, 'GLM_1_AutoML_20200222_132116': 0.5329696274679702, 'DeepLearning_grid__2_AutoML_20200222_132116_model_1': 0.1785443247147275, 'XGBoost_grid__1_AutoML_20200222_132116_model_1': 0.026158606025897122, 'XGBoost_3_AutoML_20200222_132116': 0.0, 'XGBoost_grid__1_AutoML_20200222_132116_model_4': 0.0, 'DRF_1_AutoML_20200222_132116': 0.14370500884265608, 'XGBoost_1_AutoML_20200222_132116': 0.031497896763378226, 'DeepLearning_1_AutoML_20200222_132116': 0.0878257223706418, 'DeepLearning_grid__1_AutoML_20200222_132116_model_1': 0.14719232235066243, 'XGBoost_grid__1_AutoML_20200222_132116_model_3': 0.0, 'XGBoost_2_AutoML_20200222_132116': 0.0, 'XRT_1_AutoML_20200222_132116': 0.04791984693762383, 'GBM_grid__1_AutoML_20200222_132116_model_2': 0.0, 'GBM_5_AutoML_20200222_132116': 0.0}
```



```
In [48]: # Save the model
h2o.save_model(aml.leader, path = "./saved_models/automl_005_bin")
```

```
Out[48]: '/home/alex/Documents/mlbase/disaster_tweet_kaggle/h2o/saved_models/automl_005_bin/StackedEnsemble_BestOfFamily_AutoML_20200222_132116'
```

```
In [49]: %%time
# Experiment 5 Prediction Test
predictions5 = aml.predict(X_test)
predictions5.shape
```

stackedensemble prediction progress: |██| 10  
0%  
CPU times: user 183 ms, sys: 31.1 ms, total: 214 ms  
Wall time: 14.4 s

Out[49]: (3263, 3)

```
In [53]: predictions5
```

predict	p0	p1
1	0.299266	0.700734
1	0.130853	0.869147
1	0.0740614	0.925939
1	0.403016	0.596984
1	0.048906	0.951094
1	0.229236	0.770764
0	0.742732	0.257268
0	0.806888	0.193112
0	0.819966	0.180034
0	0.797184	0.202816

Out[53]:

```
In [50]: import pandas as pd
test_df = pd.read_csv("../data/test.csv")
test_id = test_df['id']
test_predictions_df = pd.DataFrame([test_id, pd.Series(predictions5.as_data_frame().iloc[:, 0])]).T
test_predictions_df.columns = ['id', 'target']
```

```
In [51]: test_predictions_df.to_csv('test_preds_automl_005.csv', index=False)
```

```
In [52]: lb = aml.leaderboard
         lb.head()
```

	model_id	auc	logloss	aucpr	mean_per_class_error	
StackedEnsemble_BestOfFamily_AutoML_20200222_132116	0.858575	0.446024	0.852109	0.212056	0	
StackedEnsemble_AllModels_AutoML_20200222_132116	0.858222	0.446052	0.851982	0.214646	0	
XGBoost_grid__1_AutoML_20200222_132116_model_2	0.847469	0.470319	0.835908	0.221965	0	
GBM_4_AutoML_20200222_132116	0.845332	0.483274	0.835073	0.226435	0	
GBM_grid__1_AutoML_20200222_132116_model_1	0.841468	0.500601	0.832605	0.228817	0	
GBM_3_AutoML_20200222_132116	0.840894	0.494642	0.830746	0.23427	0	
GBM_2_AutoML_20200222_132116	0.84019	0.496928	0.830837	0.2335		
GBM_1_AutoML_20200222_132116	0.836954	0.503705	0.826798	0.229047	0	
GLM_1_AutoML_20200222_132116	0.835047	0.491352	0.826733	0.232259	0	
DeepLearning_grid__2_AutoML_20200222_132116_model_1	0.829	0.652747	0.587207	0.236082	0	

Out[52]:

```
In [21]: h2o.save_model(aml.leader, path = "./saved_models/automl_004_bin")
```

```
Out[21]: '/home/alex/Documents/mlbase/disaster_tweet_kaggle/h2o/saved_models/automl_004_bin/DeepLearning_grid__1_AutoML_20200222_032428_model_1'
```

```
In [23]: aml2 = h2o.load_model("./saved_models/automl_004_bin/DeepLearning_grid__1_AutoML_20200222_032428_model_1")
```

In [24]: `dir(aml2)`

```

Out[24]: ['F0point5',
          'F1',
          'F2',
          '_ModelBase__generate_partial_plots',
          '_ModelBase__generate_user_splits',
          '_ModelBase__grabValues',
          '_ModelBase__plot_1dpdp',
          '_ModelBase__plot_2dpdp',
          '_ModelBase__predFor3D',
          '_ModelBase__setAxs1D',
          '__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__',
          '_additional_used_columns',
          '_bc',
          '_bci',
          '_check_and_save_parm',
          '_check_targets',
          '_compute_algo',
          '_delegate_to_metrics',
          '_end_time',
          '_estimator_type',
          '_fillMultinomialDict',
          '_future',
          '_get_metrics',
          '_have_mojito',
          '_have_pojo',
          '_id',
          '_is_xvalidated',
          '_job',
          '_keyify_if_h2oframe',
          '_metrics_class',
          '_model_json',
          '_parms',
          '_plot',
          '_print_model_scoring_history',
          '_requires_training_frame',
          '_resolve_model',
          '_run_time',

```



In [25]: aml2.F1

Model Details  
=====

H2ODeepLearningEstimator : Deep Learning  
Model Key: DeepLearning\_grid\_\_1\_AutoML\_20200222\_032428\_model\_1

Status of Neuron Layers: predicting C20337, 2-class classification, bernoulli distribution, CrossEntropy loss, 4,067,802 weights/biases, 48.8 MB, 27,144 training samples, mini-batch size 1

	layer	units	type	dropout	I1	I2	mean_rate	rate_rms	momentum	mean_weight	weight_l
0	1	20336	Input	20							
1	2	200	RectifierDropout	40	0	0	0.977042	0.00153036	0	0.00727941	0.0481
2	3	2	Softmax		0	0	0.902415	0.244836	0	0.00986327	0.38

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

MSE: 1.7266850582090668e-09  
RMSE: 4.155340008000629e-05  
LogLoss: 9.267748319421941e-07  
Mean Per-Class Error: 0.0  
AUC: 1.0  
AUCPR: 0.0  
Gini: 1.0

Confusion Matrix (Act/Pred) for max f1 @ threshold = 0.9996075471647122:

		0	1	Error	Rate
0	0	7612.0	0.0	0.0	(0.0/7612.0)
1	1	0.0	1.0	0.0	(0.0/1.0)
2	Total	7612.0	1.0	0.0	(0.0/7613.0)

Maximum Metrics: Maximum metrics at their respective thresholds

		metric	threshold	value	idx
0		max f1	9.996075e-01	1.0	0.0
1		max f2	9.996075e-01	1.0	0.0
2		max f0point5	9.996075e-01	1.0	0.0
3		max accuracy	9.996075e-01	1.0	0.0
4		max precision	9.996075e-01	1.0	0.0
5		max recall	9.996075e-01	1.0	0.0
6		max specificity	9.996075e-01	1.0	0.0
7		max absolute_mcc	9.996075e-01	1.0	0.0
8	max min_per_class_accuracy		9.996075e-01	1.0	0.0
9	max mean_per_class_accuracy		9.996075e-01	1.0	0.0
10		max tns	9.996075e-01	7612.0	0.0
11		max fns	9.996075e-01	0.0	0.0
12		max fps	6.193928e-18	7612.0	399.0
13		max tps	9.996075e-01	1.0	0.0
14		max tnr	9.996075e-01	1.0	0.0
15		max fnr	9.996075e-01	0.0	0.0
16		max fpr	6.193928e-18	1.0	399.0
17		max tpr	9.996075e-01	1.0	0.0

Gains/Lift Table: Avg response rate: 0.01 %, avg score: 0.01 %

	group	cumulative_data_fraction	lower_threshold	lift	cumulative_lift	response_rate	score
0	1	0.010114	1.802922e-10	98.87013	98.870130	0.012987	1.306836e-02
1	2	0.020229	2.293305e-11	0.00000	49.435065	0.000000	6.132870e-11
2	3	0.030080	5.540042e-12	0.00000	33.244541	0.000000	1.280717e-11
3	4	0.040063	1.642750e-12	0.00000	24.960656	0.000000	2.900338e-12
4	5	0.050046	5.903790e-13	0.00000	19.981627	0.000000	1.013965e-12
5	6	0.100092	1.023800e-14	0.00000	9.990814	0.000000	1.296480e-13
6	7	0.150007	3.534926e-16	0.00000	6.666375	0.000000	2.963425e-15
7	8	0.200053	1.129864e-17	0.00000	4.998687	0.000000	1.001845e-16
8	9	0.300013	3.732061e-21	0.00000	3.333187	0.000000	1.588717e-18
9	10	0.399974	2.624096e-25	0.00000	2.500164	0.000000	4.595708e-22
10	11	0.500066	2.920532e-31	0.00000	1.999737	0.000000	1.992354e-26
11	12	0.600026	2.611918e-39	0.00000	1.666594	0.000000	1.884698e-32
12	13	0.699987	1.446788e-50	0.00000	1.428598	0.000000	1.350577e-40
13	14	0.799947	2.332991e-65	0.00000	1.250082	0.000000	4.724662e-52
14	15	0.899908	2.487951e-91	0.00000	1.111225	0.000000	3.352021e-67
15	16	1.000000	9.211654e-140	0.00000	1.000000	0.000000	3.588582e-93

ModelMetricsBinomial: deeplearning  
 \*\* Reported on cross-validation data. \*\*

MSE: 0.00013135426277789726  
 RMSE: 0.011460988734742621  
 LogLoss: 0.0030228245701911715  
 Mean Per-Class Error: 0.008867577509195979  
 AUC: 0.9822648449816079  
 AUCPR: 0.003676470588235294  
 Gini: 0.9645296899632159

Confusion Matrix (Act/Pred) for max f1 @ threshold = 1.0132437368732008e-10:

	0	1	Error	Rate
0	0	7477.0	135.0	0.0177 (135.0/7612.0)
1	1	0.0	1.0	0.0 (0.0/1.0)
2	Total	7477.0	136.0	0.0177 (135.0/7613.0)

Maximum Metrics: Maximum metrics at their respective thresholds

		<b>metric</b>	<b>threshold</b>	<b>value</b>	<b>idx</b>
<b>0</b>		max f1	1.013244e-10	0.014599	135.0
<b>1</b>		max f2	1.013244e-10	0.035714	135.0
<b>2</b>		max f0point5	1.013244e-10	0.009174	135.0
<b>3</b>		max accuracy	5.100234e-05	0.999737	0.0
<b>4</b>		max precision	1.013244e-10	0.007353	135.0
<b>5</b>		max recall	1.013244e-10	1.000000	135.0
<b>6</b>		max specificity	5.100234e-05	0.999869	0.0
<b>7</b>		max absolute_mcc	1.013244e-10	0.084986	135.0
<b>8</b>	max min_per_class_accuracy		1.013244e-10	0.982265	135.0
<b>9</b>	max mean_per_class_accuracy		1.013244e-10	0.991132	135.0
<b>10</b>		max tns	5.100234e-05	7611.000000	0.0
<b>11</b>		max fns	5.100234e-05	1.000000	0.0
<b>12</b>		max fps	1.421735e-18	7612.000000	399.0
<b>13</b>		max tps	1.013244e-10	1.000000	135.0
<b>14</b>		max tnr	5.100234e-05	0.999869	0.0
<b>15</b>		max fnr	5.100234e-05	1.000000	0.0
<b>16</b>		max fpr	1.421735e-18	1.000000	399.0
<b>17</b>		max tpr	1.013244e-10	1.000000	135.0

Gains/Lift Table: Avg response rate: 0.01 %, avg score: 0.00 %

	group	cumulative_data_fraction	lower_threshold	lift	cumulative_lift	response_rate	score
0	1	0.010114	8.512282e-10	0.000000	0.000000	0.000000	9.025751e
1	2	0.020097	6.870312e-11	100.171053	49.758170	0.013158	2.661291e
2	3	0.030080	6.435641e-12	0.000000	33.244541	0.000000	2.513683e
3	4	0.040063	1.272123e-12	0.000000	24.960656	0.000000	2.979778e
4	5	0.050046	2.141560e-13	0.000000	19.981627	0.000000	6.290080e
5	6	0.100092	1.556618e-16	0.000000	9.990814	0.000000	3.269805e
6	7	0.150007	1.083049e-20	0.000000	6.666375	0.000000	2.217211e
7	8	0.200053	4.798200e-26	0.000000	4.998687	0.000000	9.142823e
8	9	0.300013	3.443183e-39	0.000000	3.333187	0.000000	1.627817e
9	10	0.399974	4.852426e-52	0.000000	2.500164	0.000000	1.091184e
10	11	0.500066	3.679154e-64	0.000000	1.999737	0.000000	2.044856e
11	12	0.600026	1.424191e-76	0.000000	1.666594	0.000000	1.359427e
12	13	0.699987	1.016643e-87	0.000000	1.428598	0.000000	5.826658e
13	14	0.799947	4.678720e-101	0.000000	1.250082	0.000000	3.315469e
14	15	0.899908	1.590261e-117	0.000000	1.111225	0.000000	1.859891e-1
15	16	1.000000	8.635758e-164	0.000000	1.000000	0.000000	3.277727e-1

Cross-Validation Metrics Summary:

		mean	sd	cv_1_valid	cv_2_valid	cv_3_valid	cv_4_val
0	auc	0.19027595	0.42546996	0.0	0.9513798	0.0	(
1	aucpr	0.006666667	0.0	NaN	0.006666667	NaN	Ni
2	lift_top_group	0.0	0.0	NaN	0.0	NaN	Ni
3	logloss	0.0030220307	0.006757462	2.727688E-10	0.015110146	1.723711E-11	2.6087467E-
4	mean_per_class_error	0.024310118	0.0	NaN	0.024310118	NaN	Ni
5	mse	1.3131976E-4	2.9363993E-4	1.0240409E-16	6.5659883E-4	5.977388E-20	3.4193868E-
6	pr_auc	0.006666667	0.0	NaN	0.006666667	NaN	Ni
7	r2	-Infinity	NaN	-Infinity	-6.5703265E-4	-Infinity	-Infir
8	rmse	0.0051248944	0.011459451	1.0119491E-8	0.025624184	2.4448696E-10	5.8475524E-

Scoring History:

	timestamp	duration	training_speed	epochs	iterations	samples	training_rmse	training_logloss	traini
0	2020-02-22 11:17:38	0.000 sec	None	0.000000	0	0.0	NaN	NaN	
1	2020-02-22 11:17:44	39 min 9.779 sec	223 obs/sec	0.170235	1	1296.0	0.011461	4.536821e-03	-0.0
2	2020-02-22 11:19:54	41 min 19.682 sec	220 obs/sec	3.565480	21	27144.0	0.000042	9.267748e-07	0.9

Variable Importances:

	variable	relative_importance	scaled_importance	percentage
0	C2744	1.000000	1.000000	0.000114
1	C19770	0.987238	0.987238	0.000112
2	C13074	0.965126	0.965126	0.000110
3	C3981	0.961258	0.961258	0.000109
4	C19799	0.955598	0.955598	0.000109
5	C18052	0.947276	0.947276	0.000108
6	C13202	0.934454	0.934454	0.000106
7	C12733	0.914398	0.914398	0.000104
8	C164	0.911760	0.911760	0.000104
9	C17455	0.909137	0.909137	0.000103
10	C16127	0.907963	0.907963	0.000103
11	C18531	0.903513	0.903513	0.000103
12	C2610	0.899375	0.899375	0.000102
13	C18590	0.898254	0.898254	0.000102
14	C12670	0.896280	0.896280	0.000102
15	C5900	0.895458	0.895458	0.000102
16	C16629	0.891352	0.891352	0.000101
17	C2562	0.886023	0.886023	0.000101
18	C4525	0.883681	0.883681	0.000101
19	C2594	0.882354	0.882354	0.000100

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

Out[25]: <bound method H20BinomialModel.F1 of >

In [29]: di

```

-----
AttributeError                                Traceback (most recent call last)
<ipython-input-29-b3b7bb88a570> in <module>
----> 1 aml2.training_info

~/miniconda3/envs/h2o/lib/python3.7/site-packages/h2o/utils/metaclass.py in __getattribute__(self, name)
    191         if name in self._bci:
    192             return self._bci[name]
--> 193         return getattr(new_clz, name)
    194
    195         new_clz = extend_and_replace(clz, __init__=__init__, __getattribute__=__getattribute__)

~/miniconda3/envs/h2o/lib/python3.7/site-packages/h2o/utils/metaclass.py in __getattribute__(cls, name)
    233         if attr is not MetaFeature.NOT_FOUND:
    234             return attr
--> 235         return type.__getattribute__(cls, name)
    236
    237     def __setattr__(cls, name, value):

AttributeError: type object 'ModelBase' has no attribute 'training_info'

```

In [ ]:

In [6]: lb = aml.leaderboard

In [6]: lb.head()

	model_id	mean_residual_deviance	rmse	mse	mae
StackedEnsemble_BestOfFamily_AutoML_20200221_092127		0.148802	0.385749	0.148802	0.310458
StackedEnsemble_AllModels_AutoML_20200221_092127		0.148818	0.38577	0.148818	0.310552
XGBoost_2_AutoML_20200221_092127		0.157919	0.39739	0.157919	0.337343
GBM_4_AutoML_20200221_092127		0.16077	0.400961	0.16077	0.348566
GBM_3_AutoML_20200221_092127		0.163293	0.404095	0.163293	0.35532
XGBoost_1_AutoML_20200221_092127		0.164017	0.40499	0.164017	0.359857
GLM_1_AutoML_20200221_092127		0.164835	0.405998	0.164835	0.352993
GBM_2_AutoML_20200221_092127		0.165197	0.406444	0.165197	0.361149
GBM_1_AutoML_20200221_092127		0.16637	0.407885	0.16637	0.36481
XGBoost_3_AutoML_20200221_092127		0.171875	0.414578	0.171875	0.379498

Out[6]:



```
In [7]: lb.head()
```

	model_id	mean_residual_deviance	rmse	mse	
	StackedEnsemble_AllModels_AutoML_20200221_201104	0.000131376	0.0114619	0.000131376	0.000
	StackedEnsemble_BestOfFamily_AutoML_20200221_201104	0.000131376	0.0114619	0.000131376	0.000
	GBM_5_AutoML_20200221_201104	0.000132225	0.0114989	0.000132225	0.000
	GBM_4_AutoML_20200221_201104	0.00013967	0.0118182	0.00013967	0.000
	XGBoost_2_AutoML_20200221_201104	0.000140173	0.0118395	0.000140173	0.000
	GBM_3_AutoML_20200221_201104	0.000140341	0.0118465	0.000140341	0.000
	GBM_2_AutoML_20200221_201104	0.000141645	0.0119015	0.000141645	0.000
	XGBoost_1_AutoML_20200221_201104	0.000148843	0.0122001	0.000148843	0.000
	GLM_1_AutoML_20200221_201104	0.00016624	0.0128934	0.00016624	0.000
	XGBoost_3_AutoML_20200221_201104	0.000174279	0.0132015	0.000174279	0.000

Out[7]:

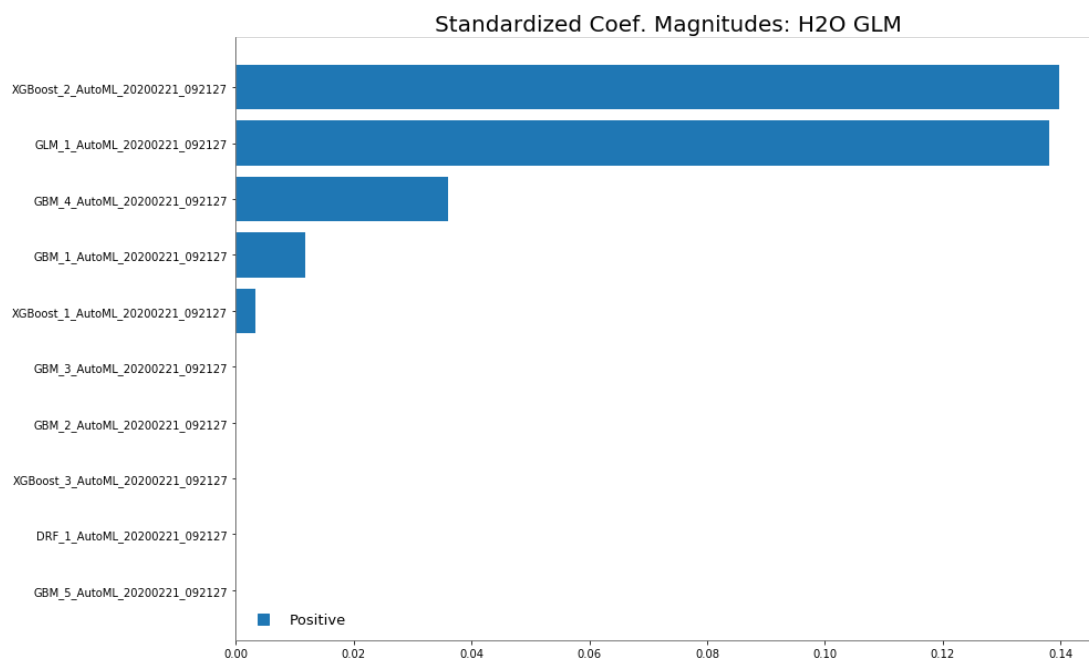
```
In [ ]:
```

```
In [7]: # 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(se.metalearner()['name'])
```

```
In [8]: metalearner.coef_norm()
```

```
Out[8]: {'Intercept': 0.42965979246026575,
'XGBoost_2_AutoML_20200221_092127': 0.13980509143405623,
'GBM_4_AutoML_20200221_092127': 0.036119599854358475,
'GBM_3_AutoML_20200221_092127': 0.0,
'XGBoost_1_AutoML_20200221_092127': 0.0034155997464314753,
'GLM_1_AutoML_20200221_092127': 0.1381498848377734,
'GBM_2_AutoML_20200221_092127': 0.0,
'GBM_1_AutoML_20200221_092127': 0.011880218602186476,
'XGBoost_3_AutoML_20200221_092127': 0.0,
'DRF_1_AutoML_20200221_092127': 0.0,
'GBM_5_AutoML_20200221_092127': 0.0}
```

```
In [9]: %matplotlib inline
        metalearner.std_coef_plot()
```

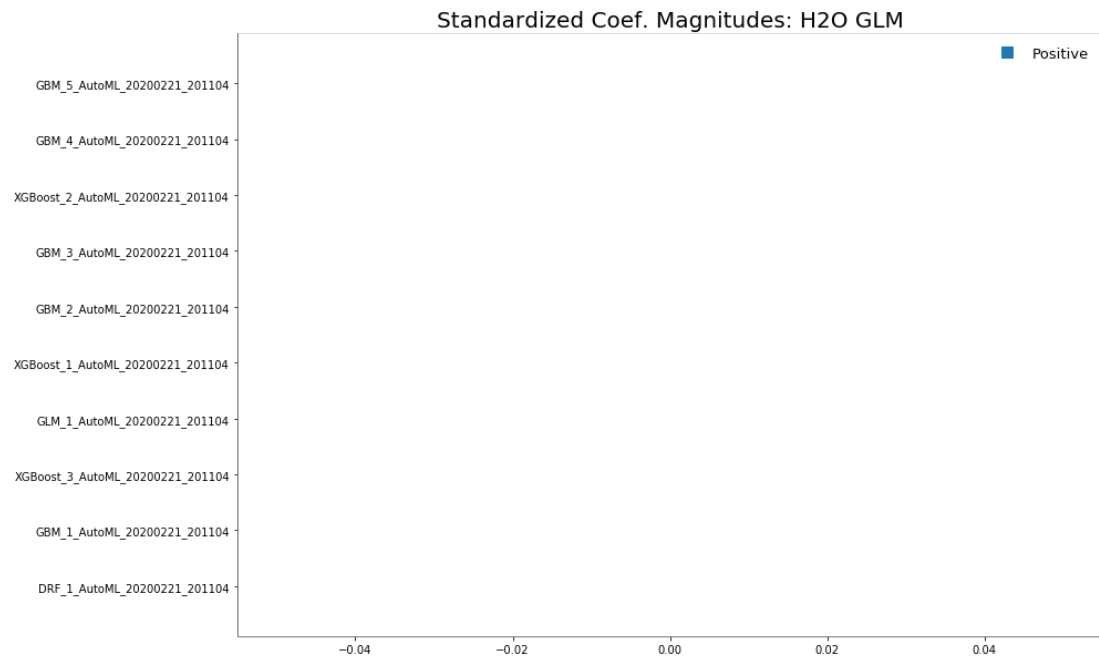


```
In [12]: # Get model ids for all models in the AutoML Leaderboard
        model_ids = list(aml.leaderboard['model_id'].as_data_frame().iloc[:,0])
        # Get the "All Models" Stacked Ensemble model
        se = h2o.get_model([mid for mid in model_ids if "StackedEnsemble_AllModels" in
        mid][0])
        # Get the Stacked Ensemble metalearner model
        metalearner = h2o.get_model(se.metalearner()['name'])
```

```
In [13]: metalearner.coef_norm()
```

```
Out[13]: {'Intercept': 0.00013135426244581638,
          'GBM_5_AutoML_20200221_201104': 0.0,
          'GBM_4_AutoML_20200221_201104': 0.0,
          'XGBoost_2_AutoML_20200221_201104': 0.0,
          'GBM_3_AutoML_20200221_201104': 0.0,
          'GBM_2_AutoML_20200221_201104': 0.0,
          'XGBoost_1_AutoML_20200221_201104': 0.0,
          'GLM_1_AutoML_20200221_201104': 0.0,
          'XGBoost_3_AutoML_20200221_201104': 0.0,
          'GBM_1_AutoML_20200221_201104': 0.0,
          'DRF_1_AutoML_20200221_201104': 0.0}
```

```
In [14]: %matplotlib inline
         metalearner.std_coef_plot()
```



```
In [8]: h2o.save_model(aml.leader, path = "./saved_models/automl_002_bin")
```

```
Out[8]: '/home/alex/Documents/mlbase/disaster_tweet_kaggle/h2o/saved_models/automl_002_bin/StackedEnsemble_AllModels_AutoML_20200221_201104'
```

```
In [9]: ?h2o.save_model
```

**Signature:** h2o.save\_model(model, path='', force=False)

**Docstring:**

Save an H2O Model object to disk. (Note that ensemble binary models can now be saved using this method.)

:param model: The model object to save.

:param path: a path to save the model at (hdfs, s3, local)

:param force: if True overwrite destination directory in case it exists, or throw exception if set to False.

:returns: the path of the saved model

:examples:

```
>>> from h2o.estimators.glm import H2OGeneralizedLinearEstimator
>>> h2o_df = h2o.import_file("http://s3.amazonaws.com/h2o-public-test-data/smalldata/prostate/prostate.csv.zip")
>>> my_model = H2OGeneralizedLinearEstimator(family = "binomial")
>>> my_model.train(y = "CAPSULE",
...               x = ["AGE", "RACE", "PSA", "GLEASON"],
...               training_frame = h2o_df)
>>> h2o.save_model(my_model, path='', force=True)
File:      ~/miniconda3/envs/h2o/lib/python3.7/site-packages/h2o/h2o.py
Type:      function
```

```
In [ ]:
```

```
X_test = sparse.load_npz("../processed_data/full_raw_keyword_bow_test_sparse.npz")
X_test = h2o.H2OFrame(X_test)
```

```
Parse progress: |██████████████████████████████████████| 10  
0%
```

```

In [19]: predictions = aml.predict(X_test)

stackedensemble prediction progress: |███████| (failed)

-----
OSError                                Traceback (most recent call last)
<ipython-input-19-abdfdb4e54e0> in <module>
----> 1 predictions = aml.predict(X_test)

~/miniconda3/envs/h20/lib/python3.7/site-packages/h2o/automl/autoh2o.py in predict
ict(self, test_data)
    516         leader = self.leader
    517         if leader is not None:
--> 518             return leader.predict(test_data)
    519         print("No model built yet...")
    520

~/miniconda3/envs/h20/lib/python3.7/site-packages/h2o/model/model_base.py in pr
edict(self, test_data, custom_metric, custom_metric_func)
    232         j = H2OJob(h2o.api("POST /4/Predictions/models/%s/frames/%s" %
(self.model_id, test_data.frame_id), data = {'custom_metric_func': custom_metri
c_func}),
    233                     self._model_json["algo"] + " prediction")
--> 234         j.poll()
    235         return h2o.get_frame(j.dest_key)
    236

~/miniconda3/envs/h20/lib/python3.7/site-packages/h2o/job.py in poll(self, poll
_updates)
    76         if (isinstance(self.job, dict)) and ("stacktrace" in list(s
elf.job)):
    77             raise EnvironmentError("Job with key {} failed with an
exception: {}\\nstacktrace: "
--> 78                                     "\\n{}".format(self.job_key, sel
f.exception, self.job["stacktrace"]))
    79         else:
    80             raise EnvironmentError("Job with key %s failed with an
exception: %s" % (self.job_key, self.exception))

OSError: Job with key $03017f00000132d4ffffffff$8196bb3b970f73b7e68fec28eb254b
20 failed with an exception: java.lang.NullPointerException
stacktrace:
java.lang.NullPointerException
  at water.MRTask.dfork(MRTask.java:453)
  at water.MRTask.doAll(MRTask.java:390)
  at water.MRTask.doAll(MRTask.java:397)
  at hex.glm.GLMMModel.predictScoreImpl(GLMMModel.java:1734)
  at hex.Model.score(Model.java:1470)
  at hex.ensemble.StackedEnsembleModel.predictScoreImpl(StackedEnsembleMo
del.java:146)
  at hex.Model.score(Model.java:1470)
  at water.api.ModelMetricsHandler$1.compute2(ModelMetricsHandler.java:38
1)
  at water.H2O$H2OCountedCompleter.compute(H2O.java:1468)
  at jsr166y.CountedCompleter.exec(CountedCompleter.java:468)
  at jsr166y.ForkJoinTask.doExec(ForkJoinTask.java:263)
  at jsr166y.ForkJoinPool$WorkQueue.runTask(ForkJoinPool.java:974)
  at jsr166y.ForkJoinPool.runWorker(ForkJoinPool.java:1477)
  at jsr166y.ForkJoinWorkerThread.run(ForkJoinWorkerThread.java:104)

```

In [ ]:

In [ ]:

In [15]:

```
# Shut down and release RAM
h2o.cluster().shutdown()

H2O session _sid_8a00 closed.
```

In [ ]: