AutoML - AutoGluon

In January 2020, Amazon introduced AutoGluon, an open-source library that utilizes Automatic Machine Learning (AutoML) by deploying ML into ML itself.

AutoML is considered to be more accurate, time-saving, and easy-to-build than traditional ML methods

https://autogluon.mxnet.io/index.html#

In [14]:
```
!python3 -m venv jatin_automl
!source jatin_automl/bin/activate
```

In [15]:
```
!pip install mxnet autogluon
```

```
Requirement already satisfied: mxnet in /Users/jatinmalhotra/anaconda3/li
b/python3.7/site-packages (1.5.1.post0)
Requirement already satisfied: autogluon in /Users/jatinmalhotra/anaconda
3/lib/python3.7/site-packages (0.0.5)
Requirement already satisfied: graphviz<0.9.0,>=0.8.1 in /Users/jatinmalh
otra/anaconda3/lib/python3.7/site-packages (from mxnet) (0.8.4)
Requirement already satisfied: numpy<2.0.0,>1.16.0 in /Users/jatinmalhotr
a/anaconda3/lib/python3.7/site-packages (from mxnet) (1.18.1)
Requirement already satisfied: requests<3,>=2.20.0 in /Users/jatinmalhotr
a/anaconda3/lib/python3.7/site-packages (from mxnet) (2.21.0)
Requirement already satisfied: tqdm>=4.38.0 in /Users/jatinmalhotra/anaco
nda3/lib/python3.7/site-packages (from autogluon) (4.42.0)
Requirement already satisfied: gluonnlp==0.8.1 in /Users/jatinmalhotra/an
aconda3/lib/python3.7/site-packages (from autogluon) (0.8.1)
Requirement already satisfied: boto3==1.9.187 in /Users/jatinmalhotra/ana
conda3/lib/python3.7/site-packages (from autogluon) (1.9.187)
Requirement already satisfied: scikit-optimize in /Users/jatinmalhotra/an
aconda3/lib/python3.7/site-packages (from autogluon) (0.7.1)
Requirement already satisfied: psutil>=5.0.0 in /Users/jatinmalhotra/anac
```

In [16]:
```
import autogluon as ag
from autogluon import TabularPrediction as task
```

In [17]:
```
import pandas as pd
```

In [18]:
```
# Titanic Dataset
df=pd.read_csv("train.csv",sep=",")
```

In [19]:
```
df.head()
```

Out[19]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN

In [20]:
```
df.Survived.value_counts()
```

Out[20]:
```
0    549
1    342
Name: Survived, dtype: int64
```

In [21]:
```
train_data = task.Dataset(df)
#train_data = train_data.head(10000) # subsample 500 data points for faster
train_data.head()
```

Out[21]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN

AutoGluon Hyper-Parameters Explanation

label - Name of the column that contains the target variable to predict.

output_directory - Path to directory where models and intermediate outputs should be saved.

problem_type - Type of prediction problem, i.e. is this a binary/multiclass classification or regression problem (options: 'binary', 'multiclass', 'regression'). If problem_type = None, the prediction problem type is inferred based on the label-values in provided dataset.

eval_metric - Metric by which predictions will be ultimately evaluated on test data. AutoGluon tunes factors such as hyperparameters, early-stopping, ensemble-weights, etc. in order to improve this metric on validation data.

stopping_metric - Metric which models use to early stop to avoid overfitting. stopping_metric is not used by weighted ensembles, instead weighted ensembles maximize eval_metric. Defaults to eval_metric value except when eval_metric='roc_auc', where it defaults to log_loss. Options are identical to options for eval_metric.

hyperparameter_tune - Whether to tune hyperparameters or just use fixed hyperparameter values for each model. Setting as True will increase fit() runtimes.

feature_prune - Whether or not to perform feature selection.

auto_stack - Whether to have AutoGluon automatically attempt to select optimal num_bagging_folds and stack_ensemble_levels based on data properties. Note: Overrides num_bagging_folds and stack_ensemble_levels values. Note: This can increase training time by up to 20x, but can produce much better results. Note: This can increase inference time by up to 20x.

num_bagging_folds - Number of folds used for bagging of models. When num_bagging_folds = k, training time is roughly increased by a factor of k (set = 0 to disable bagging). Disabled by default, but we recommend values between 5-10 to maximize predictive performance. Increasing num_bagging_folds will result in models with lower bias but that are more prone to overfitting. Values > 10 may produce diminishing returns, and can even harm overall results due to overfitting. To further improve predictions, avoid increasing num_bagging_folds much beyond 10 and instead increase num_bagging_sets.

stack_ensemble_levels - Number of stacking levels to use in stack ensemble. Roughly increases model training time by factor of stack_ensemble_levels+1 (set = 0 to disable stack ensembling). Disabled by default, but we recommend values between 1-3 to maximize predictive performance. To prevent overfitting, this argument is ignored unless you have also set num_bagging_folds >= 2.

enable_fit_continuation - Whether the predictor returned by this fit() call should be able to be further trained via another future fit() call. When enabled, the training and validation data are saved to disk for future reuse.

time_limits - Approximately how long fit() should run for (wallclock time in seconds). If not specified, fit() will run until all models have completed training, but will not repeatedly bag models unless num_bagging_sets is specified.

visualizer - How to visualize the neural network training progress during fit(). Options: [‘mxboard’, ‘tensorboard’, ‘none’].

verbosity - Verbosity levels range from 0 to 4 and control how much information is printed during fit(). Higher levels correspond to more detailed print statements (you can set verbosity = 0 to suppress warnings). If using logging, you can alternatively control amount of information printed via logger.setLevel(L), where L ranges from 0 to 50 (Note: higher values of L correspond to fewer print statements, opposite of verbosity levels)

search_strategy - Which hyperparameter search algorithm to use. Options include: ‘random’ (random search), ‘skopt’ (SKopt Bayesian optimization), ‘grid’ (grid search), ‘hyperband’ (Hyperband), ‘rl’ (reinforcement learner)

Autogluon Uses Python open source called Dask for Parallel Computing.

In [22]:

```
predictor = task.fit(train_data=train_data,
                     label="Survived",
                     output_directory="AutogluonModels/",
                     problem_type ="binary",
                     eval_metric="accuracy",
                     stopping_metric= None,
                     hyperparameter_tune=False,
                     feature_prune=False,
                     auto_stack = False,
                     num_bagging_folds=0,
                     stack_ensemble_levels=0,
                     enable_fit_continuation=False,
                     time_limits=300,
                     verbosity=2,
                     search_strategy="grid"

                    )
```

```
Beginning AutoGluon training ... Time limit = 300s
AutoGluon will save models to AutogluonModels/
Train Data Rows:    891
Train Data Columns: 12
Preprocessing data ...
Selected class <--> label mapping:  class 1 = 1, class 0 = 0
Feature Generator processed 891 data points with 33 features
Original Features:
        int features: 4
        object features: 5
        float features: 2
Generated Features:
        int features: 22
All Features:
        int features: 26
        object features: 5
        float features: 2
        Data preprocessing and feature engineering runtime = 0.28s ...
AutoGluon will gauge predictive performance using evaluation metric: accu
racy
To change this, specify the eval_metric argument of fit()
AutoGluon will early stop models using evaluation metric: accuracy
Fitting model: RandomForestClassifierGini ... Training model for up to 29
9.72s of the 299.72s of remaining time.
        0.8212   = Validation accuracy score
        0.39s    = Training runtime
        0.13s    = Validation runtime
Fitting model: RandomForestClassifierEntr ... Training model for up to 29
9.16s of the 299.16s of remaining time.
        0.8268   = Validation accuracy score
        0.46s    = Training runtime
        0.12s    = Validation runtime
Fitting model: ExtraTreesClassifierGini ... Training model for up to 298.
56s of the 298.56s of remaining time.
        0.8045   = Validation accuracy score
        0.4s     = Training runtime
        0.12s    = Validation runtime
Fitting model: ExtraTreesClassifierEntr ... Training model for up to 298.
```

```
02s of the 298.02s of remaining time.
        0.8101   = Validation accuracy score
        0.4s     = Training runtime
        0.12s    = Validation runtime
Fitting model: KNeighborsClassifierUnif ... Training model for up to 297.
48s of the 297.48s of remaining time.
        0.6089   = Validation accuracy score
        0.01s    = Training runtime
        0.11s    = Validation runtime
Fitting model: KNeighborsClassifierDist ... Training model for up to 297.
36s of the 297.36s of remaining time.
        0.6145   = Validation accuracy score
        0.01s    = Training runtime
        0.11s    = Validation runtime
Fitting model: LightGBMClassifier ... Training model for up to 297.23s of
the 297.23s of remaining time.
        0.8268   = Validation accuracy score
        0.29s    = Training runtime
        0.01s    = Validation runtime
Fitting model: CatboostClassifier ... Training model for up to 296.93s of
the 296.93s of remaining time.
        0.8156   = Validation accuracy score
        0.99s    = Training runtime
        0.01s    = Validation runtime
Fitting model: NeuralNetClassifier ... Training model for up to 295.92s o
f the 295.92s of remaining time.
        0.8212   = Validation accuracy score
        3.96s    = Training runtime
        0.21s    = Validation runtime
Fitting model: LightGBMClassifierCustom ... Training model for up to 291.
73s of the 291.73s of remaining time.
        0.8268   = Validation accuracy score
        0.52s    = Training runtime
        0.01s    = Validation runtime
Fitting model: weighted_ensemble_k0_l1 ... Training model for up to 299.7
2s of the 290.14s of remaining time.
        0.8436   = Validation accuracy score
        0.4s     = Training runtime
        0.0s     = Validation runtime
AutoGluon training complete, total runtime = 10.27s ...
```

Evaluation Summary of Multiple Model

In [23]:
```
print(predictor.fit_summary())
```

```
*** Summary of fit() ***
Number of models trained: 11
Types of models trained:
{'TabularNeuralNetModel', 'CatboostModel', 'KNNModel', 'LGBModel', 'RFMod
el', 'WeightedEnsembleModel'}
Validation performance of individual models: {'RandomForestClassifierGin
i': 0.8212290502793296, 'RandomForestClassifierEntr': 0.8268156424581006,
'ExtraTreesClassifierGini': 0.8044692737430168, 'ExtraTreesClassifierEnt
r': 0.8100558659217877, 'KNeighborsClassifierUnif': 0.6089385474860335,
'KNeighborsClassifierDist': 0.6145251396648045, 'LightGBMClassifier': 0.8
268156424581006, 'CatboostClassifier': 0.8156424581005587, 'NeuralNetClas
sifier': 0.8212290502793296, 'LightGBMClassifierCustom': 0.82681564245810
06, 'weighted_ensemble_k0_l1': 0.8435754189944135}
Best model (based on validation performance): weighted_ensemble_k0_l1
Hyperparameter-tuning used: False
Bagging used: False
Stack-ensembling used: False
User-specified hyperparameters:
{'NN': {'num_epochs': 500}, 'GBM': {'num_boost_round': 10000}, 'CAT': {'i
terations': 10000}, 'RF': {'n_estimators': 300}, 'XT': {'n_estimators': 3
00}, 'KNN': {}, 'custom': ['GBM']}
Plot summary of models saved to file: SummaryOfModels.html
*** End of fit() summary ***
{'model_types': {'RandomForestClassifierGini': 'RFModel', 'RandomForestCl
assifierEntr': 'RFModel', 'ExtraTreesClassifierGini': 'RFModel', 'ExtraTr
eesClassifierEntr': 'RFModel', 'KNeighborsClassifierUnif': 'KNNModel', 'K
NeighborsClassifierDist': 'KNNModel', 'LightGBMClassifier': 'LGBModel',
'CatboostClassifier': 'CatboostModel', 'NeuralNetClassifier': 'TabularNeu
ralNetModel', 'LightGBMClassifierCustom': 'LGBModel', 'weighted_ensemble_
k0_l1': 'WeightedEnsembleModel'}, 'model_performance': {'RandomForestClas
sifierGini': 0.8212290502793296, 'RandomForestClassifierEntr': 0.82681564
24581006, 'ExtraTreesClassifierGini': 0.8044692737430168, 'ExtraTreesClas
sifierEntr': 0.8100558659217877, 'KNeighborsClassifierUnif': 0.6089385474
860335, 'KNeighborsClassifierDist': 0.6145251396648045, 'LightGBMClassifi
er': 0.8268156424581006, 'CatboostClassifier': 0.8156424581005587, 'Neura
lNetClassifier': 0.8212290502793296, 'LightGBMClassifierCustom': 0.826815
6424581006, 'weighted_ensemble_k0_l1': 0.8435754189944135}, 'model_best':
'weighted_ensemble_k0_l1', 'model_paths': {'RandomForestClassifierGini':
'AutogluonModels/models/RandomForestClassifierGini/', 'RandomForestClassi
fierEntr': 'AutogluonModels/models/RandomForestClassifierEntr/', 'ExtraTr
eesClassifierGini': 'AutogluonModels/models/ExtraTreesClassifierGini/',
'ExtraTreesClassifierEntr': 'AutogluonModels/models/ExtraTreesClassifierE
ntr/', 'KNeighborsClassifierUnif': 'AutogluonModels/models/KNeighborsClas
sifierUnif/', 'KNeighborsClassifierDist': 'AutogluonModels/models/KNeighb
orsClassifierDist/', 'LightGBMClassifier': 'AutogluonModels/models/LightG
BMClassifier/', 'CatboostClassifier': 'AutogluonModels/models/CatboostCla
ssifier/', 'NeuralNetClassifier': 'AutogluonModels/models/NeuralNetClassi
fier/', 'LightGBMClassifierCustom': 'AutogluonModels/models/LightGBMClass
ifierCustom/', 'weighted_ensemble_k0_l1': 'AutogluonModels/models/weighte
d_ensemble_k0_l1/'}, 'model_fit_times': {'RandomForestClassifierGini': 0.
39109110832214355, 'RandomForestClassifierEntr': 0.4632601737976074, 'Ext
raTreesClassifierGini': 0.39534997940063477, 'ExtraTreesClassifierEntr':
0.39725279808044434, 'KNeighborsClassifierUnif': 0.008833885192871094, 'K
NeighborsClassifierDist': 0.009203910827636719, 'LightGBMClassifier': 0.2
862510681152344, 'CatboostClassifier': 0.9940118789672852, 'NeuralNetClas
```

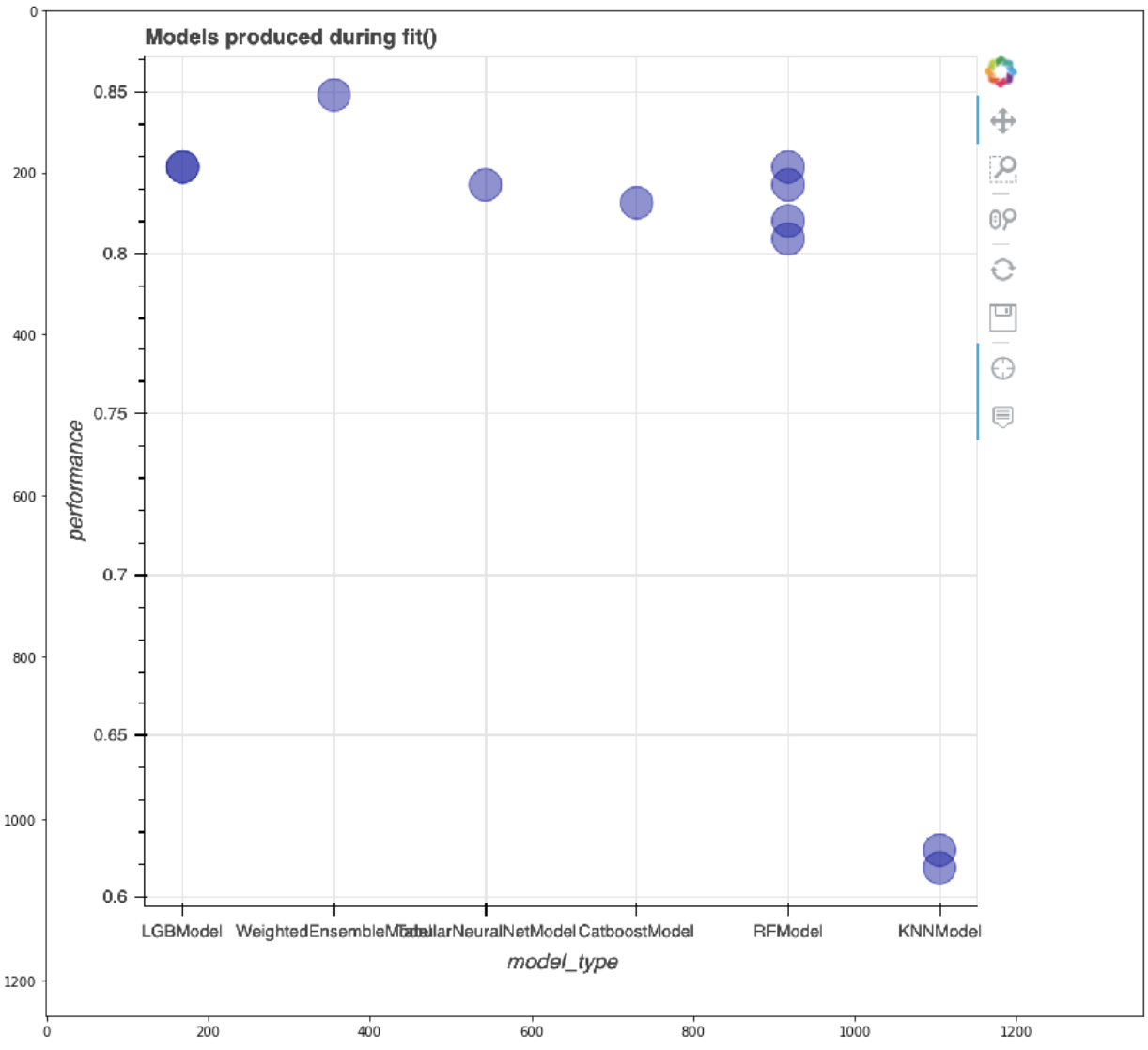
sifier': 3.9601919651031494, 'LightGBMClassifierCustom': 0.52193737030029
3, 'weighted_ensemble_k0_l1': 0.4029660224914551}, 'model_pred_times':
{'RandomForestClassifierGini': 0.13118886947631836, 'RandomForestClassifi
erEntr': 0.11670088768005371, 'ExtraTreesClassifierGini': 0.1210770606994
6289, 'ExtraTreesClassifierEntr': 0.11937832832336426, 'KNeighborsClassif
ierUnif': 0.11108112335205078, 'KNeighborsClassifierDist': 0.111308097839
35547, 'LightGBMClassifier': 0.012463092803955078, 'CatboostClassifier':
0.011341094970703125, 'NeuralNetClassifier': 0.2113039493560791, 'LightGB
MClassifierCustom': 0.014373779296875, 'weighted_ensemble_k0_l1': 0.00077
53372192382812}, 'num_bagging_folds': 0, 'stack_ensemble_levels': 0, 'fea
ture_prune': False, 'hyperparameter_tune': False, 'hyperparameters_usersp
ecified': {'NN': {'num_epochs': 500}, 'GBM': {'num_boost_round': 10000},
'CAT': {'iterations': 10000}, 'RF': {'n_estimators': 300}, 'XT': {'n_esti
mators': 300}, 'KNN': {}, 'custom': ['GBM']}, 'num_classes': 2, 'model_hy
perparams': {'RandomForestClassifierGini': {'model_type': 'rf', 'n_estima
tors': 300, 'n_jobs': -1, 'criterion': 'gini'}, 'RandomForestClassifierEn
tr': {'model_type': 'rf', 'n_estimators': 300, 'n_jobs': -1, 'criterion':
'entropy'}, 'ExtraTreesClassifierGini': {'model_type': 'xt', 'n_estimator
s': 300, 'n_jobs': -1, 'criterion': 'gini'}, 'ExtraTreesClassifierEntr':
{'model_type': 'xt', 'n_estimators': 300, 'n_jobs': -1, 'criterion': 'ent
ropy'}, 'KNeighborsClassifierUnif': {'weights': 'uniform', 'n_jobs': -1},
'KNeighborsClassifierDist': {'weights': 'distance', 'n_jobs': -1}, 'Light
GBMClassifier': {'num_boost_round': 10000, 'num_threads': -1, 'objectiv
e': 'binary', 'metric': 'binary_logloss,binary_error', 'verbose': -1, 'bo
osting_type': 'gbdt', 'two_round': True}, 'CatboostClassifier': {'iterati
ons': 10000, 'learning_rate': 0.1, 'random_seed': 0, 'eval_metric': 'Accu
racy'}, 'NeuralNetClassifier': {'num_epochs': 500, 'seed_value': None, 'p
roc.embed_min_categories': 4, 'proc.impute_strategy': 'median', 'proc.max
_category_levels': 100, 'proc.skew_threshold': 0.99, 'network_type': 'wid
edeep', 'layers': [256, 128], 'numeric_embed_dim': 329, 'activation': 're
lu', 'max_layer_width': 2056, 'embedding_size_factor': 1.0, 'embed_expone
nt': 0.56, 'max_embedding_dim': 100, 'y_range': None, 'y_range_extend':
0.05, 'use_batchnorm': True, 'dropout_prob': 0.1, 'batch_size': 512, 'los
s_function': SoftmaxCrossEntropyLoss(batch_axis=0, w=None), 'optimizer':
'adam', 'learning_rate': 0.0003, 'weight_decay': 1e-06, 'clip_gradient':
100.0, 'momentum': 0.9, 'epochs_wo_improve': 20, 'num_dataloading_worker
s': 6, 'ctx': cpu(0)}, 'LightGBMClassifierCustom': {'num_boost_round': 10
000, 'num_threads': -1, 'objective': 'binary', 'metric': 'binary_logloss,
binary_error', 'verbose': -1, 'boosting_type': 'gbdt', 'two_round': True,
'learning_rate': 0.03, 'num_leaves': 128, 'feature_fraction': 0.9, 'min_d
ata_in_leaf': 5, 'seed_value': 0}, 'weighted_ensemble_k0_l1': {'max_model
s': 25, 'max_models_per_type': 5}}}

Evaluation summary as graph

In [25]:
```
%pylab inline
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

plt.figure(figsize = (15,15))
img=mpimg.imread('models_evaluation_summary.png')
imgplot = plt.imshow(img)
plt.show()
```

```
Populating the interactive namespace from numpy and matplotlib
```



AutoML Leadership Board

In [26]:
```
predictor.leaderboard()
```

```
                          model  score_val  fit_time  pred_time_val  stack
_level
10     weighted_ensemble_k0_l1   0.843575  0.402966       0.000775
1
1   RandomForestClassifierEntr   0.826816  0.463260       0.116701
0
9     LightGBMClassifierCustom   0.826816  0.521937       0.014374
0
6           LightGBMClassifier   0.826816  0.286251       0.012463
0
0   RandomForestClassifierGini   0.821229  0.391091       0.131189
0
8         NeuralNetClassifier   0.821229  3.960192       0.211304
0
7          CatboostClassifier   0.815642  0.994012       0.011341
0
3     ExtraTreesClassifierEntr   0.810056  0.397253       0.119378
0
2     ExtraTreesClassifierGini   0.804469  0.395350       0.121077
0
5     KNeighborsClassifierDist   0.614525  0.009204       0.111308
0
4     KNeighborsClassifierUnif   0.608939  0.008834       0.111081
0
```

Out[26]:

	model	score_val	fit_time	pred_time_val	stack_level
10	weighted_ensemble_k0_l1	0.843575	0.402966	0.000775	1
1	RandomForestClassifierEntr	0.826816	0.463260	0.116701	0
9	LightGBMClassifierCustom	0.826816	0.521937	0.014374	0
6	LightGBMClassifier	0.826816	0.286251	0.012463	0
0	RandomForestClassifierGini	0.821229	0.391091	0.131189	0
8	NeuralNetClassifier	0.821229	3.960192	0.211304	0
7	CatboostClassifier	0.815642	0.994012	0.011341	0
3	ExtraTreesClassifierEntr	0.810056	0.397253	0.119378	0
2	ExtraTreesClassifierGini	0.804469	0.395350	0.121077	0
5	KNeighborsClassifierDist	0.614525	0.009204	0.111308	0
4	KNeighborsClassifierUnif	0.608939	0.008834	0.111081	0

AutoML Accuracy

In [27]:
```
predictor.model_performance
```

Out[27]:
```
{'RandomForestClassifierGini': 0.8212290502793296,
 'RandomForestClassifierEntr': 0.8268156424581006,
 'ExtraTreesClassifierGini': 0.8044692737430168,
 'ExtraTreesClassifierEntr': 0.8100558659217877,
 'KNeighborsClassifierUnif': 0.6089385474860335,
 'KNeighborsClassifierDist': 0.6145251396648045,
 'LightGBMClassifier': 0.8268156424581006,
 'CatboostClassifier': 0.8156424581005587,
 'NeuralNetClassifier': 0.8212290502793296,
 'LightGBMClassifierCustom': 0.8268156424581006,
 'weighted_ensemble_k0_l1': 0.8435754189944135}
```

THANK YOU

In [ ]:
```

```