In [ ]:
```
from IPython.display import display, HTML

display(HTML(data="""
<style>
    div#notebook-container    { width: 95%; }
    div#menubar-container     { width: 65%; }
    div#maintoolbar-container { width: 99%; }
</style>
"""))
```

Comparison of AutoML Frameworks

Introduction

Why AutoML?

https://www.infoworld.com/article/3430788/automated-machine-learning-or-automl-explained.html (https://www.infoworld.com/article/3430788/automated-machine-learning-or-automl-explained.html)

It aims to reduce or eliminate the need for skilled data scientists to build machine learning and deep learning models. Instead, an AutoML system allows you to provide the labeled training data as input and receive an optimized model as output.

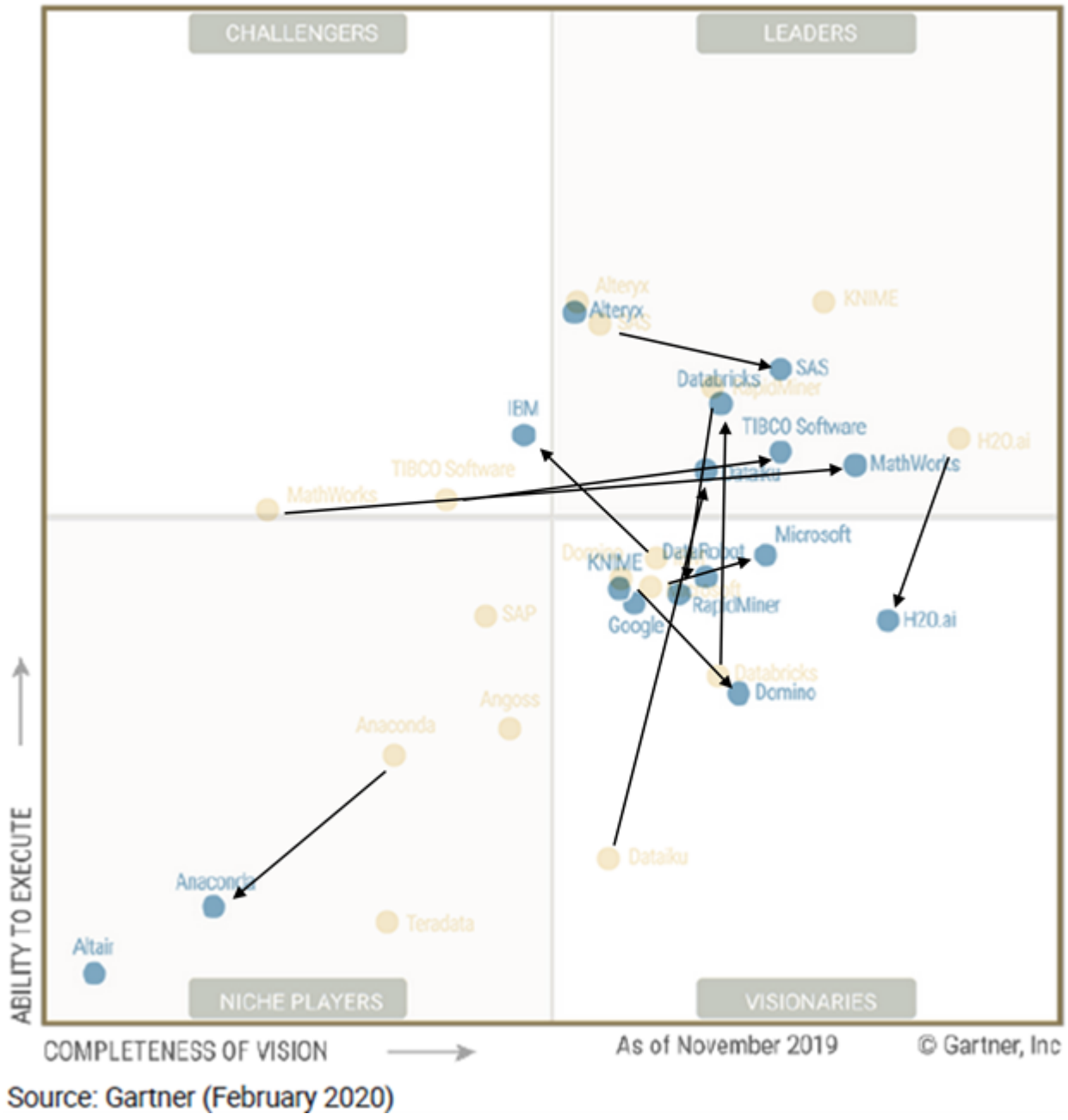
There are several ways of going about this.

1. Simply train every kind of model on the data and pick the one that works best.
2. Optimize the hyperparameters of the best model or models to train an even better model.

Gartner Data Science and Machine Learning Platforms (2018, 2019)

https://b2bsalescafe.files.wordpress.com/2019/09/gartner-magic-quadrant-for-data-science-and-machine-learning-platforms-january-2019.pdf (https://b2bsalescafe.files.wordpress.com/2019/09/gartner-magic-quadrant-for-data-science-and-machine-learning-platforms-january-2019.pdf)
https://b2bsalescafe.files.wordpress.com/2020/04/gartner-magic-quadrant-for-data-science-and-machine-learning-platforms-feb-2020.pdf (https://b2bsalescafe.files.wordpress.com/2020/04/gartner-magic-quadrant-for-data-science-and-machine-learning-platforms-feb-2020.pdf)

1. It assesses the availability of some pre-packaged content but does not assess service providers that can help jump-start or extend DSML projects throughout an organization and does not assess specialized vendors of industry-, or function-specific solutions.
2. There are 16 vendors of DSML, all are measured based on specific weighting criteria before pointed into the magic quadrant.
3. The magic quadrant consist of four (4) quadrants; Challengers, Leaders, Niche Players and Visionaries. These quadrants have their own specific criteria and characteristics.
4. Comparison are done between Gartner Magic Quadrant 2019 and 2018 to see the movement of DSML platforms across quadrant. This aims is to see the performance of vendors over different assessments in the market.
5. Here are platforms discarded from the magic quadrant due to unable to reached specific standards set by Gartner. However, there are also new plaforms introduced in Gartner Magic Quadrant 2019.



Source: Gartner (February 2020)

AutoML Table of Comparison

https://arxiv.org/pdf/1908.05557.pdf (https://arxiv.org/pdf/1908.05557.pdf)
https://arxiv.org/abs/1808.06492 (https://arxiv.org/abs/1808.06492)

		Open Source Version									
Tool		H20 AutoML	TPOT	Auto-Keras	TransmogrifAI	Auto-sklearn	Auto-ml	Ludwig	Auto-Weka	**Autogluon	**Microsoft NNI
Platform		AWS, GCP, Azure			Apache Spark					AWS	Azure
Input Data Sources	**Support Apache Spark	Y	N (*dask*)	N	Y	N (*sk-dist*)	N	N	N	N	N
	Spreadsheet Datasets	Y	Y	Y	Y	Y	Y	Y	Y	Y	
	Image, Text	N	N	Y	N	N	N	Y	N	Y	
Data pre-processing		Y	N	N	Y(*)	N	N	Y(*)	N	Y	
Data Type Detected	Numerical	Y	N	N	Y	N	N	Y	Y	Y	
	Categorical	Y	N	N	Y	N	N	Y	Y	Y	
	Datetime	Y	N	N	Y	N	N	N	N	Y	
	Time-series	Y	N	N	Y	N	N	Y	N		
	Others (Hierarchical types)	N	N	N	Y	N	N	Y	N		
Feature Engineering	Datetime, Categorical Processing	Y	N	N	Y	Y (2*)	Y	N	N	Y	
	Imbalance, Missing Values	Y	Y	Y	Y	Y	Y	Y	Y	Y	
	Feture Selection, Reduction	Y	N	Y	Y	Y	Y	Y	Y	Y	
	Advanced Feature Extraction	N	Y	N	Y	Y	Y	Y	N		
ML Task	Supervised Learning	Y	Y	Y	Y	Y	Y	Y	Y	Y	
	Unsupervised Learning	N	N	N	N	N	N	N	N	N	
Model Selection and Hyperparameter Optimization	Ensembled	Y	Y	N	Y	Y	Y	Y	Y	Y	
	Genetic Algorithm	N	Y	N	N	N	N	N	N		
	Random Search	Y	N	Y	Y	Y	Y	Y	Y		
	Bayesian Search	N	N	Y	Y	Y	Y	Y	Y		
	Neural Architecture Search	N	N	Y	N	N	N	Y	N		
Quick Start/Early Stop	Quick Finding of Starting Model	N	N	Y	N	Y	N	Y	N		
	Allow Maximum Limit of Search Time	Y	Y	Y		N	Y	N	Y		
	Restrict Time Consuming Combination of Components	Y	N	N		Y	N	N	Y		
Model Evaluation/ Results Analysis/ Visualization (11*)	Model Dashboard	Y	Y	Y	Y	Y	Y	Y	Y	Y	
	Feature Importance	Y	Y	N	Y	Y	Y	Y	N	Y	
	Model Explainability and Interpretation, and reason code	Y	N	Y		Y	Y	N	N	Y	
**Github	Star	5.1k	7.6k	7.6k	2k	5k		7.2k	266	2.7k	8.3k
	Watch										
	Fork										
**Count		17	9	12	17	14	13	18	12		

		Paid Version					
Tool		**Darwin (+)**	**DataRobot (+)**	**Google AutoML (+)**	**Mljar(+)**	**Azure ML (+)**	**H2O-Driverless Ai (+)**
Platform		**GCP**	**AWS, GCP, Azure**	**Google Cloud**	**MLJAR Cloud**	**Azure**	**AWS, GCP, Azure**
Input Data Sources	**Support Apache Spark	N	Y	Y	N	Y	Y
	Spreadsheet Datasets	Y	Y	N	Y (3*)	Y	Y (3*)
	Image, Text	N	Y	Y	N	Y	Y
Data pre-processing		Y	Y	Y	Y	Y(6*)	Y
Data Type Detected	Numerical	Y	Y		Y	Y	Y
	Categorical	Y	Y		Y	Y	Y
	Datetime	Y	Y		N	Y	Y
	Time-series	Y	Y		N	Y	Y
	Others (Hierarchical types)	N	N		N	N	Y
Feature Engineering	Datetime, Categorical Processing	Y	Y	N	Y	Y	Y
	Imbalance, Missing Values	Y	Y	Y	Y (4*)	Y	Y
	Feture Selection, Reduction	Y	Y	Y	N	Y	Y
	Advanced Feature Extraction	Y	Y	Y	N	Y	Y
ML Task	Supervised Learning	Y	Y	Y	Y (5*)	Y	Y
	Unsupervised Learning	Y	Y	Y	N	N	Y
Model Selection and Hyperparameter Optimization	Ensembled	Y	Y		Y	Y	Y
	Genetic Algorithm	Y	Y	Y	N	N	Y
	Random Search	N	Y	Y	Y	Y	Y
	Bayesian Search	N	Y	Y	N	Y	Y
	Neural Architecture Search	Y	N	Y	N	N	N
Quick Start/Early Stop	Quick Finding of Starting Model	Y	Y (12*)	Y	N		N
	Allow Maximum Limit of Search Time	Y	Y	Y	N	Y	N
	Restrict Time Consuming Combination of Components	N		Y	N	Y	Y
Model Evaluation/ Results Analysis/ Visualization (11*)	Model Dashboard	Y	Y	Y	Y	Y	Y
	Feature Importance	Y	Y	Y	Y	Y	Y
	Model Explainability and Interpretation, and reason code	Y	Y	Y	N		Y
**Github	Star				567		
	Watch						
	Fork						
**Count		20	22	18	11	19	22

Thus, ranking is done from best to worst based on the comparison above, considering priority given to support Spark, most features available and AutoMLs version.

1. H2O-Driverless AI (support Spark, 22, paid)
2. DataRobot (support Spark, 22, paid)
3. Azure ML (support Spark, 19, paid)
4. H2O AutoML, TransmogrifAI (support Spark, 17, open source)
5. Darwin (20, paid)
6. Ludwig (18, open source)
7. Auto-sklearn (14, open source)

There are some of the criteria considered as more important than the rest.

1. Input Data Source (Spreadsheet dataset)
2. Data type detected (Numerical, Categorical)
3. Feature Engineering (Datetime, categorical processing, Imbalance, missing value)
4. ML task (Supervised learning)
5. Model selection and hyperparameter optimization (Ensembled, Neural architecture search)
6. Quick start/early stop (Allow maximum limit of search time, Restrict time consuming combination of components)
7. Model evaluation/result analysis/visualization (Feature importance, model explain ability and interpretation, and reason code)
8. Github (Star)

Thus, ranking is done from best to worst based on the comparison above, considering priority given to support Spark, most features available and AutoMLs version.

1. H2O AutoML (support Spark, 14, open source)
2. DataRobot, H20-Driverless AI, Azure ML (support Spark, 12, paid)
3. TransmogrifAI (support Spark, 10, open source)
4. Google AutoML (support Spark, 9, paid)
5. Darwin (12, paid)
6. Autogluon (10, open source)
7. Ludwig (9, open source)
8. MLJar (9, paid)

H2O AutoML

https://docs.h2o.ai/h2o/latest-stable/h2o-docs/automl.html (https://docs.h2o.ai/h2o/latest-stable/h2o-docs/automl.html)

In [ ]:
```
import h2o
from h2o.automl import H2OAutoML

h2o.init()

# Import a sample binary outcome train/test set into H2O
train = h2o.import_file("https://s3.amazonaws.com/erin-data/higgs/higgs_train_
10k.csv")
test = h2o.import_file("https://s3.amazonaws.com/erin-data/higgs/higgs_test_5
k.csv")

# Identify predictors and response
x = train.columns
y = "response"
x.remove(y)

# For binary classification, response should be a factor
train[y] = train[y].asfactor()
test[y] = test[y].asfactor()

# Run AutoML for 20 base models (limited to 1 hour max runtime by default)
aml = H2OAutoML(max_models=20, seed=1)
aml.train(x=x, y=y, training_frame=train)

# View the AutoML Leaderboard
lb = aml.leaderboard
lb.head(rows=lb.nrows)  # Print all rows instead of default (10 rows)
```

AutoKeras

https://autokeras.com/tutorial/structured_data_classification/
(https://autokeras.com/tutorial/structured_data_classification/)

In [ ]:
```
import tensorflow as tf
import autokeras as ak

TRAIN_DATA_URL = "https://storage.googleapis.com/tf-datasets/titanic/train.csv"
TEST_DATA_URL = "https://storage.googleapis.com/tf-datasets/titanic/eval.csv"

train_file_path = tf.keras.utils.get_file("train.csv", TRAIN_DATA_URL)
test_file_path = tf.keras.utils.get_file("eval.csv", TEST_DATA_URL)

# Initialize the structured data classifier.
clf = ak.StructuredDataClassifier(
    overwrite=True,
    max_trials=3) # It tries 3 different models.
# Feed the structured data classifier with training data.
clf.fit(
    # The path to the train.csv file.
    train_file_path,
    # The name of the label column.
    'survived',
    epochs=10)
# Predict with the best model.
predicted_y = clf.predict(test_file_path)
# Evaluate the best model with testing data.
print(clf.evaluate(test_file_path, 'survived'))
```

TPOT

http://epistasislab.github.io/tpot/ (http://epistasislab.github.io/tpot/)

In [ ]:
```
from tpot import TPOTClassifier
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
import numpy as np

iris = load_iris()
X_train, X_test, y_train, y_test = train_test_split(iris.data.astype(np.float64),
    iris.target.astype(np.float64), train_size=0.75, test_size=0.25, random_state=42)

tpot = TPOTClassifier(generations=5, population_size=50, verbosity=2, random_state=42)
tpot.fit(X_train, y_train)
print(tpot.score(X_test, y_test))
#tpot.export('tpot_iris_pipeline.py')
```

Application

1. Data Analysis

	age	job	marital	education	default	housing	loan	contact	day	month	duration	campaign	pdays	previous	poutcome	deposit (target)
Numerical (int)	Y										Y	Y	Y	Y		
Categorical (string)		Y	Y	Y	Y	Y	Y	Y	Y	Y					Y	Y

1. Portuguese Bank Dataset https://archive.ics.uci.edu/ml/datasets/bank+marketing# (https://archive.ics.uci.edu/ml/datasets/bank+marketing#)
2. The classification goal is to predict if the client will subscribe (yes/no) a term deposit.
3. The dataset is retrieved from Kaggle repositories. It consists of multipleclass attributes with 41188 observations.
4. The dataset is split with the ratio of best practice, 70:30. However, the split ratio can be manipulated according to own preference if the end result is not satisfied.

Data Preprocessing in Apache Spark

		Numerical (int)	Categorical (string)	StringIndexer	OneHotEncoder	VectorAssembler	final form
1	age	Y				age	age
2	job		Y	indexedJob	jobVec	jobVec	jobVec
3	marital		Y	indexedMarital	maritalVec	maritalVec	maritalVec
4	education		Y	indexedEducation	educationVec	educationVec	educationVec
5	default		Y	indexedDefault	defaultVec	defaultVec	defaultVec
6	housing		Y	indexedHousing	housingVec	housingVec	housingVec
7	loan		Y	indexedLoan	loanVec	loanVec	loanVec
8	contact		Y	indexedContact	contactVec	contactVec	contactVec
9	day		Y	indexedDay	dayVec	dayVec	dayVec
10	month		Y	indexedMonth	monthVec	monthVec	monthVec
11	duration	Y				duration	duration
12	campaign		Y	indexedCampaign	campaignVec	campaignVec	campaignVec
13	pdays	Y				pdays	pdays
14	previous	Y				previous	previous
15	poutcome		Y	indexedPoutcome	poutcomeVec	poutcomeVec	poutcomeVec
16	deposit (target)		Y	indexedDeposit		indexedDeposit	indexedDeposit

1. pipeline_prepV6_DECISION_TREE.ipynb
2. pipeline_prepV6_GRADIENT_BOOSTED.ipynb
3. pipeline_prepV6_RANDOM_FOREST.ipynb
4. duplicate_LGB.ipynb

Data Preprocessing in Scikit Learn

		Numerical	Categorical	LabelEncoder	OneHotEncoder	final form
1	age	Y				age
2	job		Y	job	job (LabelEncoder)	job (LabelEncoder, OneHotEncoder)
3	marital		Y	marital	marital (LabelEncoder)	marital (LabelEncoder, OneHotEncoder)
4	education		Y	education	education (LabelEncoder)	education (LabelEncoder, OneHotEncoder)
5	default		Y	default	default (LabelEncoder)	default (LabelEncoder, OneHotEncoder)
6	housing		Y	housing	housing (LabelEncoder)	housing (LabelEncoder, OneHotEncoder)
7	loan		Y	loan	loan (LabelEncoder)	loan (LabelEncoder, OneHotEncoder)
8	contact		Y	contact	contact (LabelEncoder)	contact (LabelEncoder, OneHotEncoder)
9	day		Y	day	day (LabelEncoder)	day (LabelEncoder, OneHotEncoder)
10	month		Y	month	month (LabelEncoder)	month (LabelEncoder, OneHotEncoder)
11	duration	Y				duration
12	campaign		Y	campaign	campaign (LabelEncoder)	campaign (LabelEncoder, OneHotEncoder)
13	pdays	Y				pdays
14	previous	Y				previous
15	poutcome		Y	poutcome	poutcome (LabelEncoder)	poutcome (LabelEncoder, OneHotEncoder)
16	deposit (target)		Y	deposit		deposit (LabelEncoder)

1. duplicate_LGB_v2.ipynb
2. duplicate_TPOT.ipynb
3. duplicate_autoKeras.ipynb
4. duplicate_autoKeras_v2.ipynb

Data Preprocessing in H2O-ai

		Numerical	Categorical	asfactor	predictors	response_col	final form
1	age	Y		age			
2	job		Y		job		job (predictors)
3	marital		Y		marital		marital (predictors)
4	education		Y		education		education (predictors)
5	default		Y		default		default (predictors)
6	housing		Y		housing		housing (predictors)
7	loan		Y		loan		loan (predictors)
8	contact		Y		contact		contact (predictors)
9	day		Y		day		day (predictors)
10	month		Y		month		month (predictors)
11	duration	Y		duration			
12	campaign		Y		campaign		campaign (predictors)
13	pdays	Y		pdays			
14	previous	Y		previous			
15	poutcome		Y		poutcome		
16	deposit (target)		Y			deposit	deposit (response_col)

1. duplicate_H2O_v2_H2OGeneralizedLinearEstimator.ipynb

2. Benchmark Testing

note: some of the missing values of error metrics are due to the unavailability provided by the AutoML framework respectively.

	ML Scripts	Accuracy	F1	Precision	Recall	Specificity	MSE	RMSE	LogLoss	AUC	AUCPR	AUCROC
1	duplicate_autoKeras	0.4935										
2	duplicate_autoKeras_v2	0.79										
3	duplicate_H2O_v2_H2OGeneralizedLinearEstimator	0.892487	0.456814	1	1	1	0.086060319	0.293360392	0.301032429	0.760344048	0.391259682	
4	duplicate_H2O_AutoML											
5	duplicate_LGB			0.421176471				0.301634151				0.797927104
6	duplicate_LGB_v2			0.93220339				0.228975683				0.947866759
7	duplicate_TPOT	0.953702	0.953921	0.961846	0.946126		0.0462977		1.59908	0.953802		
8	pipeline_prepV6_DECISION_TREE	0.903546	0.891051	0.888666	0.903546							
9	pipeline_prepV6_GRADIENT_BOOSTED	0.909861	0.901462	0.899103	0.909861							
10	pipeline_prepV6_RANDOM_FOREST	0.898235	0.866466	0.887995	0.898235							

Model Explainability

1. LIME (Local Interpretable Model-Agnostic Explanations)

https://arxiv.org/abs/1602.04938 (https://arxiv.org/abs/1602.04938)

Lime can be used to get more insights into model prediction like explaining why models take a particular decision for an individual observation. It can also be quite useful while selecting between different models. The central idea behind Lime is that it explains locally in the vicinity of the instance being explained by perturbating the different features rather than producing explanations at the entire model level.

It does so by fitting a sparse model on the locally dispersed, noise-induced dataset. This helps convert a non-linear problem into a linear one. The indicator variables with the largest coefficients in the model are then returned as the drivers of the score.

2. SHAP (Shapley Additive Explanations)

It tells on how it got the score for an instance in an additive manner. SHAP has not only a generic explainer that works for any model but also a TreeExplainer for tree-based models. It theoretically guarantees consistency and is slower than Lime.

Additionally, the computational requirements of exploring all possible feature combinations grow exponentially in SHAP.

Conclusion

Based on the result table, Auto-Keras perform the worst compared to H2OGeneralizedLinearEstimator, LGB, LGB_v2, TPOT, and MMLSPark (Decision Tree, Gradient Boosted and Random Forest) with accuracy of 0.49. However, modified automl, Auto-Keras_v2 shows a better performance with accuracy of 0.79.

H2OGeneralizedLinearEstimator yields accuracy of 0.89, outperforms Auto-Keras and Auto-Keras_v2. It also provides other error metrics values as well such as F1, Precision, Recall, Specificity, MSE, RMSE, Logloss, AUC and AUCPR. The automl configuration is very simple. It also provides model explainability for further model diagnose. H2O automl experiment cannot be done due to the limited resources. Both H2O-ai and H2O automl provide leaderboard of models run during execution.

Both LGB and LGB_v2 does not provides accuracy values. But, according to their precision values, LGB_v2 performs better compared to LGB with value of 0.93. This value, however, cannot be used directly to compare with other models or automls.

TPOT yields the best performance among other automl with accuracy of 0.95 without any modification. TPOT provides option to export best-configured pipeline to (.py) format, similar to Auto-Keras. However, TPOT is limited in features availability and error metrics, thus it is not the best option to be implemented in the future.

MMLSPark (Decision Tree, Gradient Boosted and Random Forest) are used as the benchmark models for this experiment. Decision Tree, Gradient Boosted and Random Forest yield accuracy of 0.90, 0.91, 0.89 respectively. These accuracy values, however, are tuned several times (it costs time) to get the desired, on-par with automls default configuration.

In [ ]: