INDUSTRIAL TRAINING REPORT

At

Business Unit: Group Information & Ops Dev

(Group Data Science & Analytics)

AMBANK (M) BERHAD

MUHAMMAD AFFIQ BIN MOHD AZRIN

2017807094

UNIVERSITI TEKNOLOGI MARA

INDUSTRIAL TRAINING REPORT

At

Ambank (M) Berhad,
Wisma AmFirst, Ground Floor,
Jalan Stadium SS7/15,
SS7, 47301 Petaling Jaya

28th September 2020– 31st December 2020

MUHAMMAD AFFIQ BIN MOHD AZRIN
2017807094

Report submitted in partial fulfillment of the requirement for the degree of Bachelor of Information Technology (Hons.) Intelligent Systems Engineering

Faculty of Computer and Mathematical Sciences

December 2020

Approved by:

Name	:	TAN JUN SHENG
Department	:	Business Unit: Group Information & Ops Dev (Group Data Science & Analytics)
Signature	:	
Date	:	

ACKNOWLEDGEMENTS

The internship opportunity I had with AmBank (M) Berhad was a great chance for learning and professional development. Therefore, I consider myself as a very lucky individual as I was provided with an opportunity to be a part of it. I am also grateful for having a chance to meet so many wonderful people and professionals who led me though this internship period.

Bearing in mind previous I am using this opportunity to express my deepest gratitude and special thanks to my supervisor in charge, Mr. Tan Jun Sheng, Head of Data Science & Machine Learning who in spite of being extraordinarily busy with his duties, took time out to hear, guide and keep me on the correct path and allowing me to carry out my internship at their esteemed organization and extending during the training.

I perceive as this opportunity as a big milestone in my career development. I will strive to use gained skills and knowledge in the best possible way, and I will continue to work on their improvement, in order to attain desired career objectives. Hope to continue cooperation with all of you in the future. Thank you.

Table of Contents

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 Background of Industrial Training

The score for the report will carry 60% of your final grade. The report will be evaluated by two (2) examiners; your supervisor and a second examiner. You may discuss with your supervisor on who should be your second examiner. He/She can be anybody from the Faculty or any member of other faculties at UiTM. To make sure that you are committed to the write up of the report, your supervisor holds 10% of the final grade from your commitment. The evaluations from the company contribute 30% of the total marks. The number of pages of the report should not exceed 20 pages.

1.2 Background of Organization

AmBank Group is a leading financial services group with over 40 years of expertise in supporting the economic development of Malaysia. It has over three million customers and employ over 9,000 people.

The Group was listed on the Main Market of Bursa Malaysia in 1988. It is the sixth-largest banking group by assets in Malaysia, with a market capitalization of around RM9 billion and assets of RM169.2 billion as of 31 March 2020.

AmBank Group serves over three million individual and corporate customers. It provides services in wholesale banking, business banking, retail banking, investment banking and related financial services which include Islamic banking, general insurance, life insurance, family takaful, stock and share broking, futures broking, investment advisory and management services in assets, real estate investment trust and unit trusts.

1.3 Background of Department Attached

Business Unit: Group Information & Ops Dev (Group Data Science & Analytics) is responsible to conduct analytics and to provide reliable insights based on given data for upper management to make business decision.

It comprises teams of data governance, business intelligence and data science. Data governance team is responsible to manage, protect, and ensure the integrity and usefulness of university data. They identify the sensitivity and criticality of the data, ensure that appropriate business processes are in place to keep the data secure, maximize data accuracy, and ensure that responsible staff are trained to maintain data quality.

Business intelligence team is responsible to setting business requirements for BI tools, translating business requirements into technical ones, leading BI software development, deployment, and maintenance, report curation and data modeling, participation in data warehouse design, documenting contents in a data warehouse and meta-data storage, and creating technical documentation for BI tools.

Data science team is responsible for delivering complex projects where system analysis, software engineering, data engineering, and data science is used to deliver the final solution

Together these teams work to steer the direction of Ambank in the future.

CHAPTER 2

ASSIGNMENTS

2.1 Implementation of AutoML Framework and MLOps

2.1.1 Introduction

Automated machine learning, also referred to as automated ML or AutoML, is the process of automating the time consuming, iterative tasks of machine learning model development. It allows data scientists, analysts, and developers to build ML models with high scale, efficiency, and productivity all while sustaining model quality.

2.1.2 Objectives

1. Proposed autoML framework must be able to automate machine learning process from data pre-processing to execution to model validation.
2. Proposed autoML framework must be running in Python language, and works in notebook environment (Jupyter Notebook, Azure Databricks).
3. Proposed autoML framework must be able to accept Apache Spark module as a part of the automation process.
4. Proposed autoML framework must be able to produce higher accuracy or using other valid error metrics to measure performance compared to the given values in current running ML pipeline.
5. Proposed autoML framework must be able to execute similar tasks using gradient boosting algorithms such as LightGBM, XGBoost and Catboost or others related.
6. Proposed autoML framework must be able to run offline, within the local network.

2.1.2 Extended Discussion

Gartner Magic Quadrant of Data Science and Machine Learning (DSML) Platform (2019, 2018)

These magic quadrants discuss about vendors of DSML. They assess the availability of some pre-packaged content but do 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.

There are 16 vendors of DSML, all are measured based on specific weighting criteria before pointed into the magic quadrant. The magic quadrant consists of four (4) quadrants, Challengers, Leaders, Niche Players and Visionaries. These quadrants have their own specific criteria and characteristics.

Comparison are done between Gartner Magic Quadrant 2019 and 2018 to see the movement of DSML platforms across quadrant. This aim is to see the performance of vendors over different assessments in the market. There are platforms discarded from the magic quadrant due to unable to reached specific standards set by Gartner. However, there are also new platforms introduced in Gartner Magic Quadrant 2019.

2.1.3 General Comparison based on Objectives Accomplishment

Literature review shows that Gartner Magic Quadrant 2019 and 2018 do provide a big picture on how the DSML market moves over time. New features are introduced, licenses are constantly revised, and/or engaging community is growing fast.

As according to the objective (i), some of the DSML platforms do provide automatic machine learning feature but some do not.

As according to the objective (ii), some of the DSML platforms do provide ease of use of Python scripting as part of the services although user interface (UI) is getting common among domain users, but some do not.

As according to the objective (iii), some of the DSML platforms do provide integration of their services with Apache Spark module but some do not. This integration allows user to loosely configure ML models at ease without limited to their services which is sometimes not compatible with the current setup**.**

As according to the objectives (iv), (v), Gartner Magic Quadrant (2019, 2018) do not provide any assessment on platform models' performance. However, article (2)(b), does provide a general overview on performance which includes data pre-processing, feature engineering and prediction result analysis.

2.1.4 Summary

In summary, all Gartner's DSML platforms provides machine learning capabilities in term of different approaches. However, objective (i) stated the need of automatic machine learning. Thus, few platforms need to be filtered out and discarded from the watchlist, leaving out only those which fulfill the objective.

Objective (ii) stated that, platforms must comply with Apache Spark module. This module allows data parallelism method, aims to improve performance of the machine by cutting down the compute time. This enables machine learning frameworks to process Big Data at ease. Only platforms that achieve all the objectives are finalized and listed for further evaluation and prepared for benchmark testing**.**

2.1.4 Definitions in Table of Functionality for AutoML Framework.

1. (+): commercialized tools
2. (): the function is not very stable, it fails for some datasets
3. (2): categorical input must be converted into integers
4. (3): datasets have to include headers
5. (4): missing values must be represented as NA
6. (5): multiclass classification not provided
7. (6): need some users' input for dataset description such as column types
8. (7): ability to detect primitive data types and rich data types such as: text (id, url, phone), numerical (integer, real)
9. (8): advanced feature processing: bucketing of values, removing features with zero variance or features with drift over time
10. (9): supervised learning includes binary classification, multiclass classification, regression
11. (10): unsupervised learning includes clustering and anomaly detection
12. (11): model interpretation and explainability refers to techniques such as LIME, Shapley, Decision Tree Surrogate, Partial Dependence, Individual Conditional Expectation, Lift chart, feature fit, prediction distribution plot, accuracy over time, hot spot and reason codes
13. (12): confirmed by a company spokesperson, we could not find public documentation at the time of publication

In a few empty cells, it is not clear that the functionality is provided from documentations of the tools, to the best of our knowledge.

https://arxiv.org/pdf/1908.05557.pdf

Table 2.1: Table of Comparison for AutoML Framework (Open Source)

		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 Evauation/ Results Analysis/ Visualization (11*)	Model Dashboard	Y	Y	Y	Y	Y	Y	Y	Y	Y	
	Feature Importance and	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		

Table 2.2: Table of Comparison for AutoML Framework (Paid Version)

		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 Evauation/ 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

2.1.5 Results and Discussion

A total of 16 AutoML frameworks have been evaluated based of different criteria. There are 10 open source and 6 paid versions of AutoMLs repectively.

Open Source AutoML Frameworks

For open source AutoML framework, most features available are Ludwig (18), H2O AutoML (17), TransmogrifAI (17) and Auto-sklearn (14). However, between these three, only H2O AutoML (17) and TransmogrifAI (17) support Spark module. TransmogrifAI might not preferred because it runs in Scala, while H2O AutoML runs in both Python and Scala.

Paid Version AutoML Frameworks

For paid version AutoMLs, most features available are H2O-Driverless AI (22), DataRobot (22), Darwin (20) and Azure ML (19). However, between these four, all AutoMls support Spark module except Darwin. All paid version AutoMLs provide UI as well as traditional code editing style.

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.

a. H2O-Driverless AI (support Spark, 22, paid)
b. DataRobot (support Spark, 22, paid)
c. Azure ML (support Spark, 19, paid)
d. H2O AutoML, TransmogrifAI (support Spark, 17, open source)
e. Darwin (20, paid)
f. Ludwig (18, open source)
g. Auto-sklearn (14, open source)

Evaluation continues with simple classification experiments to measure AutoMLs performances versus MMLSpark models.

a. MMLSpark Random Forest
b. MMLSpark DecisionTree
c. Microsoft LightGBM
d. XGBoost
e. H2O AutoML (support Spark, 17, open source)
f. Auto-Keras (12, open source)
g. TPOT (9, open source)

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

a. Input Data Source (Spreadsheet dataset)
b. Data type detected (Numerical, Categorical)
c. Feature Engineering (Datetime, categorical processing, Imbalance, missing value)
d. ML task (Supervised learning)
e. Model selection and hyperparameter optimization (Ensembled, Neural architecture search)
f. Quick start/early stop (Allow maximum limit of search time, Restrict time consuming combination of components)
g. Model evaluation/result analysis/visualization (Feature importance, model explain ability and interpretation, and reason code)
h. Github (Star)

Reviewed Open Source AutoML Frameworks

Considering only filtered criteria as the new criteria, top three open source AutoMLs with the most features available are H2O AutoML (14), TransmogrifAI (10), Autogluon (10) and Ludwig (9). However, between these four, only H2O AutoML (14) and TransmogrifAI (10) support Spark module. TransmogrifAI might not preferred because it runs in Scala, while H2O AutoML runs in both Python and Scala.

Reviewed Paid Version AutoML Frameworks

While top three paid AutoMLs with the most features available are all listed; Darwin (12), DataRobot (12), Azure ML (12), H2O-Driverless AI (12), Google AutoML (9) and MLJar (9). However, between these six, only DataRobot (12), Azure ML (12), H2O-Driverless AI (12) and Google AutoML (9) support Spark. All paid version AutoMLs provide UI as well as traditional code editing style.

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.

a. H2O AutoML (support Spark, 14, open source)
b. DataRobot, H20-Driverless AI, Azure ML (support Spark, 12, paid)
c. TransmogrifAI (support Spark, 10, open source)
d. Google AutoML (support Spark, 9, paid)
e. Darwin (12, paid)
f. Autogluon (10, open source)
g. Ludwig (9, open source)
h. MLJar (9, paid)

Evaluation continues with simple classification experiments to measure AutoMLs performances versus MMLSpark models.

a. MMLSpark Random Forest
b. MMLSpark DecisionTree
c. Microsoft LightGBM
d. XGBoost
e. H2O AutoML (support Spark, 14, open source)
f. Auto-Keras (6, open source)
g. TPOT (6, open source)

CHAPTER 3

CONCLUSION AND RECOMMENDATIONS

3.1 Benefits of the Assignments

1. Throughout the 14 weeks of internship, the intern is given the opportunity to work with the team and adapt with the structure of the team.
2. The intern is exposed with a good data science working environment.
3. The intern is given the opportunity to apply data science knowledge to solve real industrial problem in the related domain.
4. The intern is given the opportunity to explore features of AutoML framework and suitability of it to be implemented into the ML pipeline without losing resources through the automation of machine learning tasks like pipeline creation and hyperparameter tuning, AutoML is enabling data science team to improve their productivity.
5. The intern is given the opportunity to learn new things such as Apache Spark, Microsoft Azure Databricks and other related tools, their configurations, functionality, and purposes and how they are used in the organization as main tools to conduct analytical tasks.

3.2 Summary of Industrial Training

Throughout 14 weeks of trainings undergoes by the intern at the organization, many things the intern had learnt and gain from the industry. Training in organization make the intern had fully utilized every skills and knowledge regarding data science. All the trainings given were mostly to test out and polish the skills and understandings possessed by the intern. Intern also being able to make relations between theoretical knowledge gained at the university and applied it into the tasks given. Every new knowledge and skills obtained from inside of the organization are going to profit the intern to communicate with employees regardless their position benefits the intern to build up connections with them for future reference. Along the training’s periods, there were criticism and advice given by the supervisors towards the intern.

To improve and boost the abilities of the intern, all the words are sincerely accepted for self-reflection.

The industrial training program offered by Ambank really benefits the intern. However, there is still got a room of improvement to further enhance the quality of the trainings towards the intern. As a recommendation toward the organization, intern would suggest the organization to allow the intern to involve with every production line in the organization. This will provide the intern with varieties of Information Technology skills and knowledge, not focused only on data science.

BIBLIOGRAPHY

Balaji, A., & Allen, A. (2018). Benchmarking Automatic Machine Learning Frameworks. arXiv:1808.06492. Retrieved from https://ui.adsabs.harvard.edu/abs/2018arXiv180806492B

Peter Krensky, P. d. H., Erick Brethenoux, Jim Hare, Carlie, & Idoine, A. L., Svetlana Sicular, Farhan Choudhary. (2020). Gartner Magic Quadrant for Data Science and Machine Learning Platforms 2020.

Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). *"Why Should I Trust You?": Explaining the Predictions of Any Classifier*. Paper presented at the Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, California, USA. https://doi.org/10.1145/2939672.2939778

Truong, A., Walters, A., Goodsitt, J., Hines, K., Bruss, C., & Farivar, R. (2019). *Towards Automated Machine Learning: Evaluation and Comparison of AutoML Approaches and Tools*.

APPENDICES

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.

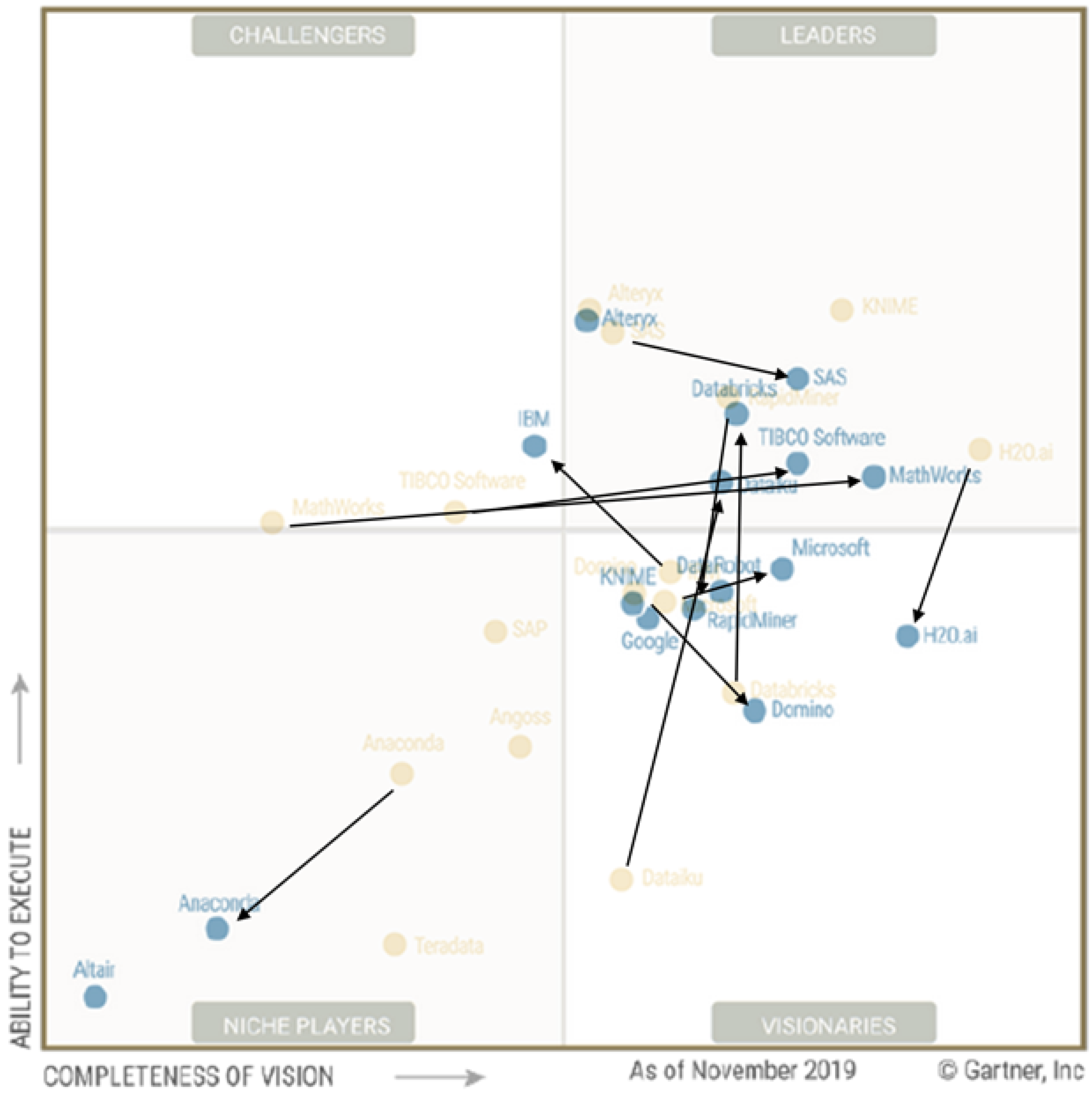
There are several ways of going about this.

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/pdf/1908.05557.pdf)
[https://arxiv.org/abs/1808.06492 (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 Evauation/ 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 Evauation/ 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 [ ]:
```python
import tensorflow as tf
import autokeras as ak

TRAIN_DATA_URL = "https://storage.googleapis.com/tf-datasets/titanic/train.cs
v"
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/)

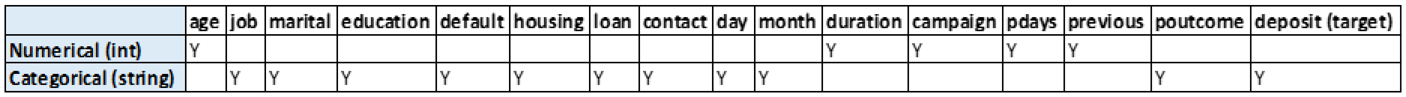
In [ ]:
```python
from tpot import TPOTClassifier
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
import numpy as np

iris = load_iris()
X_train, X_test, y_train, y_test = train_test_split(iris.data.astype(np.float6
4),
    iris.target.astype(np.float64), train_size=0.75, test_size=0.25, random_st
ate=42)

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

Application

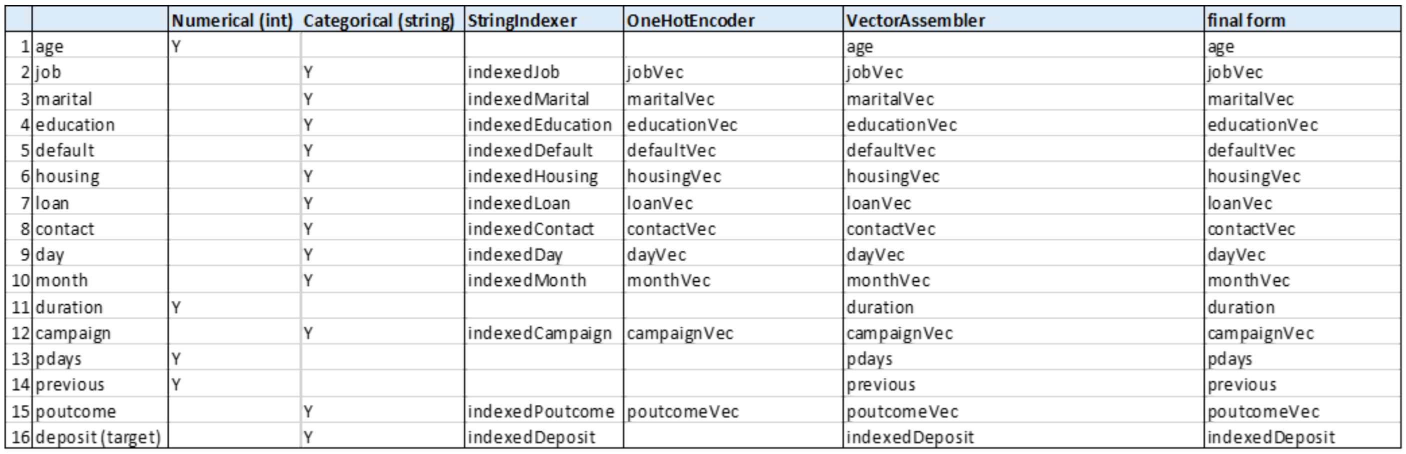
1. Data Analysis



	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.

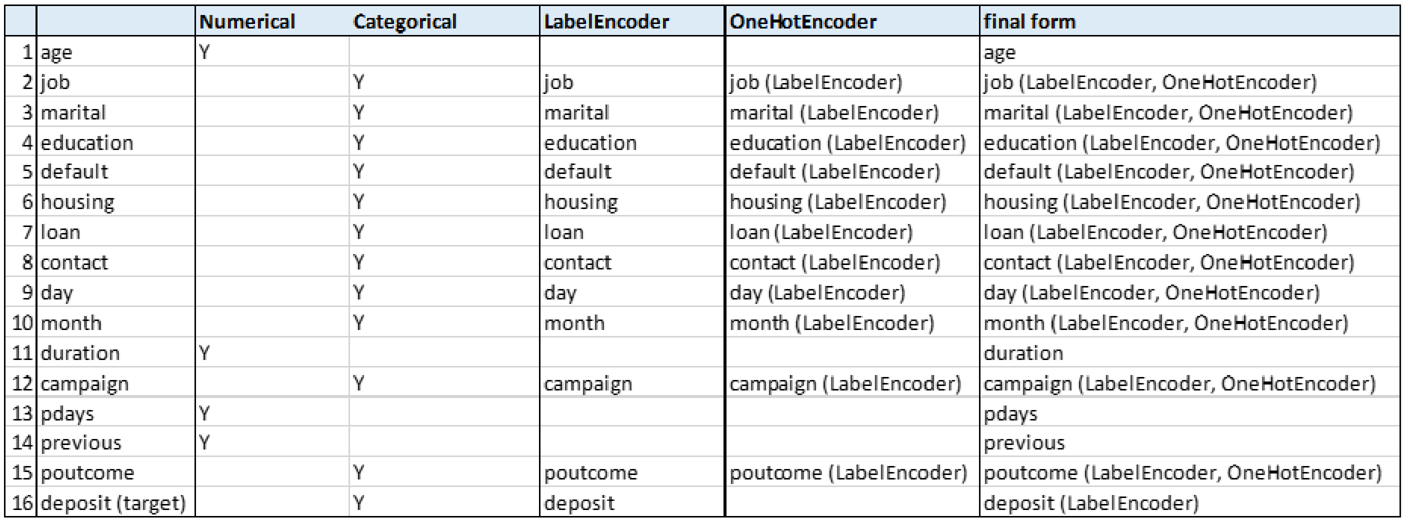
Data Preprocessing in Apache Spark



		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

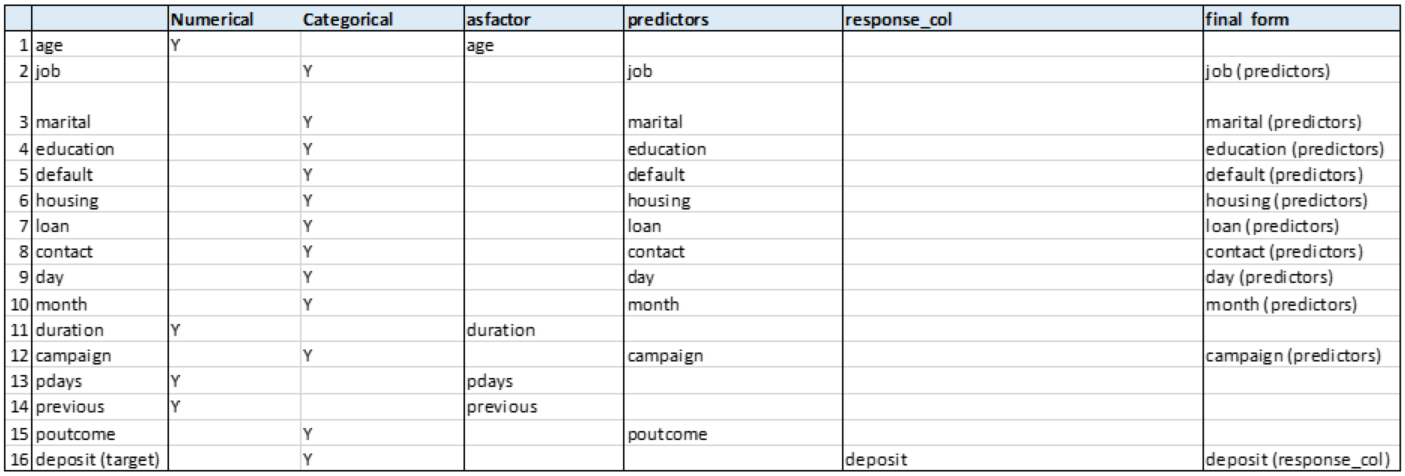
Data Preprocessing in Scikit Learn



		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

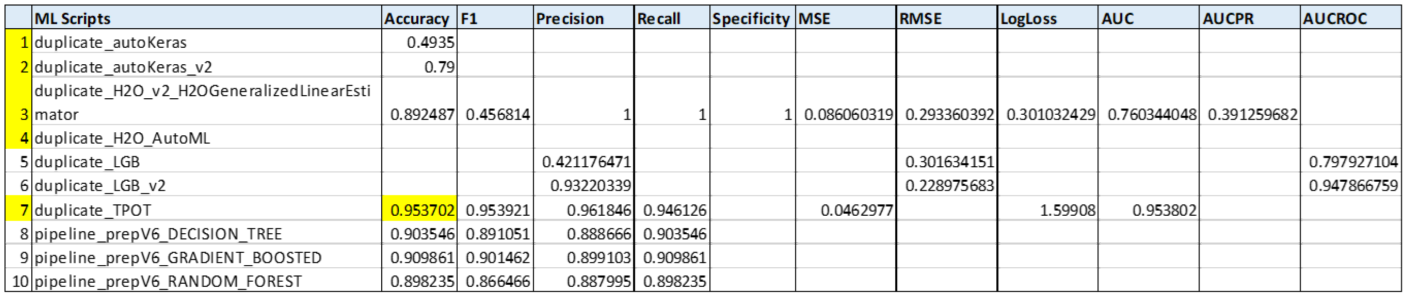


		Numerical	Categorical	asfactor	predictors	response_col	final form
1	age	Y		age			
2	job		Y		job		job (predictors)
3	marital		Y		marital		marital (predictors)
4	education		Y		education		education (predictors)
5	default		Y		default		default (predictors)
6	housing		Y		housing		housing (predictors)
7	loan		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 [ ]:

```

```