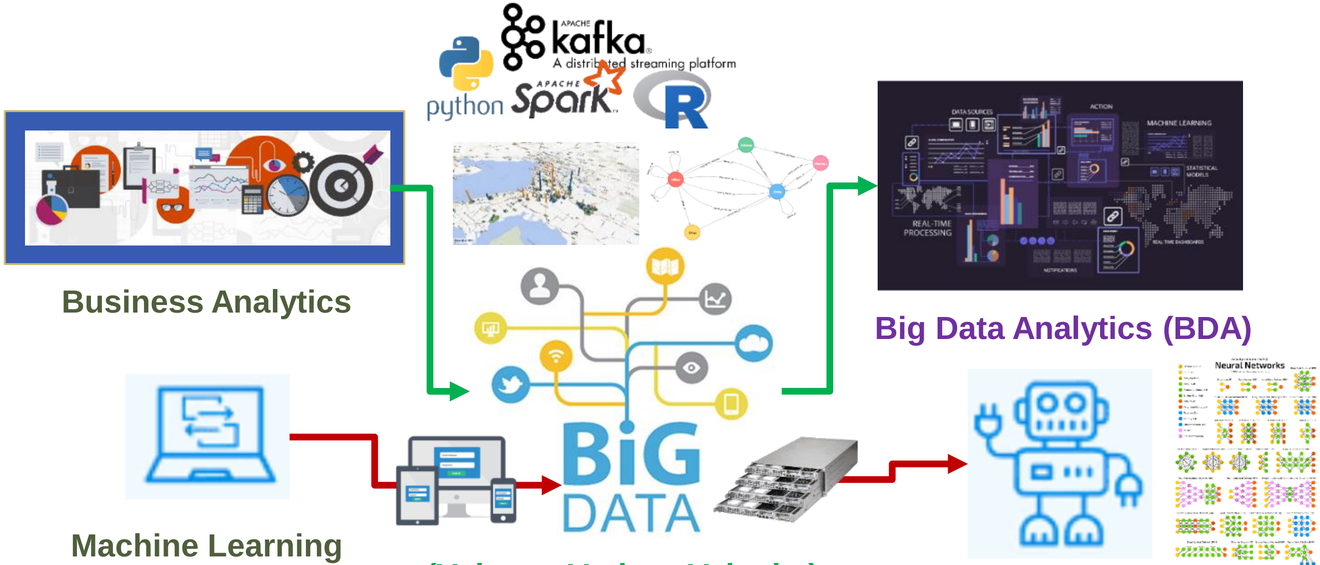
Data Science & Analytics

Accelerates Data Science with Automated Machine Learning

YC Chew, PhD
Principal Data Scientist, RHB

We Are Closer



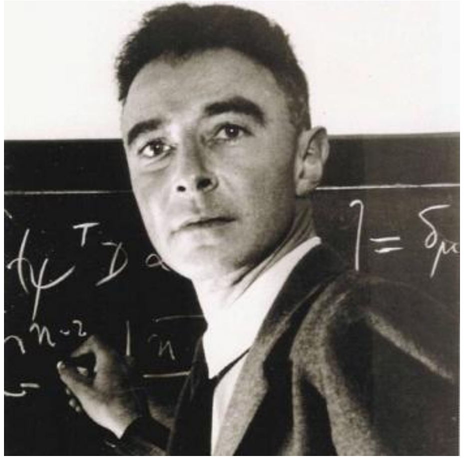
(Volume, Variety, Velocity)

Artificial Intelligence (AI)

More data are being generated, stored and processed (Big Data)

Powerful hardware & open source tools

Breakthrough in Machine Learning algorithms



J. Robert Oppenheimer

Theoretical physicist

father of the **atomic** bomb

"The optimist thinks this is the best of all possible worlds. The pessimist fears it is true."

Beginning of Artificial Intelligence (AI)

A Proposal for the Dartmouth Summer Research Project on Artificial Intelligence

August 31, 1955

John McCarthy, Marvin L. Minsky, Nathaniel Rochester, and Claude E. Shannon

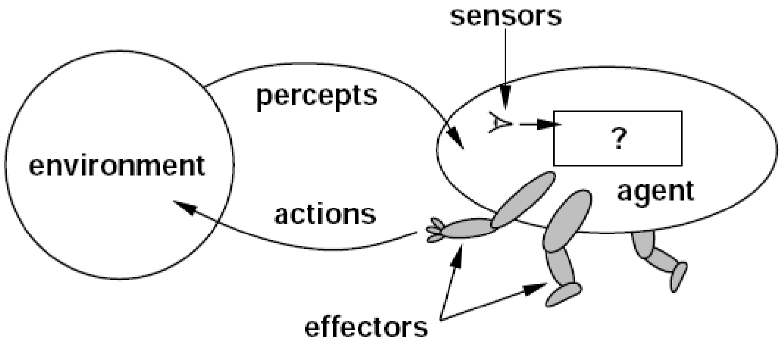
"We propose that a two-month, ten man study of artificial intelligence carried out during the summer of 1956…"

What is AI

- In computer science, **Artificial intelligence** (AI, also machine intelligence) research is defined as the study of "**intelligent agents**“.
- An intelligent agent perceives its environment via **sensors** and acts rationally upon that environment with its **effectors**.

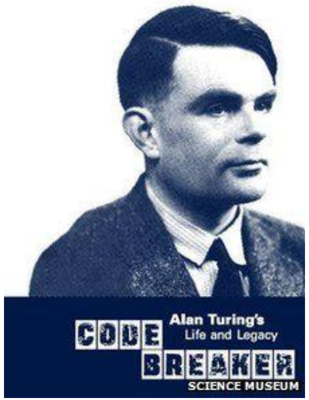
Properties:

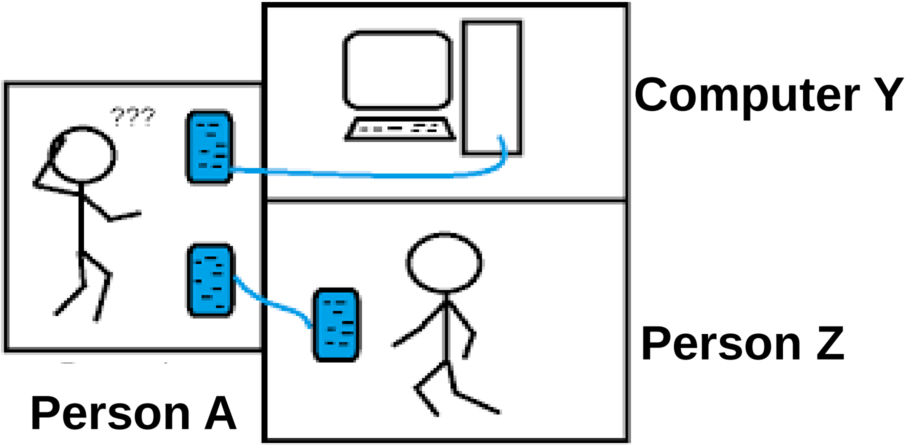
- Autonomous
- Pro-active
- Reactive to the environment
- Interacts with other agents



Is AI Exists

- Alan Turing proposed a method then be known as the **Turing Test** - if the human being conducting the test is unable to consistently determine whether an answer has been given by a computer or by another human being, then the computer is considered to have intelligence.





First Wave

- The **first wave** of AI came in the late 1950s. **It is basically a search program that based on predefined rules**.
- In order for the AI to be successful, human had to precisely defined every possibility. As machine can generally process faster than human, it doesn't bother the size of pattern to search for, as well as the rules that defined how to sort and search.
- The classic example of this type of AI is how machine plays chess. For every move, a machine simply calculates massive amounts of known patterns and evaluates the best board position.
- Unfortunately, even the DeepBlue still doesn't understand concepts of the game and just rely on brute force approaches to play.
- The first winter of AI – the **Frame Problem**.

Deep Blue vs. Kasparov

Deep Blue
IBM chess computer

Garry Kasparov
World Chess Champion

First match

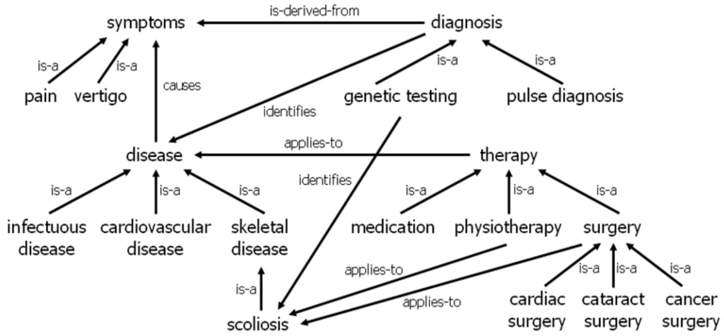
- February 10, 1996: takes place in Philadelphia, Pennsylvania
- Result: **Kasparov**–Deep Blue (4–2)
- Record set: First computer program to defeat a world champion in a *classical game* under tournament regulations

Second match (rematch)

- May 11, 1997: held in New York City, New York
- Result: **Deep Blue**–Kasparov (3½–2½)
- Record set: First computer program to defeat a world champion in a *match* under tournament regulations

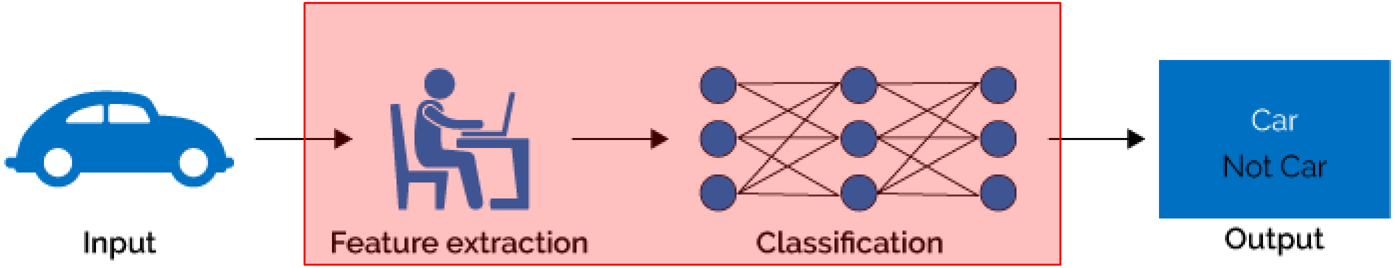
Second Wave

- The **second wave** of AI came in the 1980s. This time, its creators proposed **a systematic way to representing information about the world in a form that a computer can utilize**, it is called the Knowledge Representation and Reasoning.
- Effective KR requires a combination of common **vocabulary** and **automated reasoning engine**. The vocabulary, or **ontology**, provides a set of concepts and relationships between them. Unfortunately, it would requires tremendous effort to map reality into computerized vocabulary. Moreover, machines in KR are actually recognize the vocabulary and connections, not the concepts of the real world. The second winter of AI – the **Symbol Grounding Problem**.



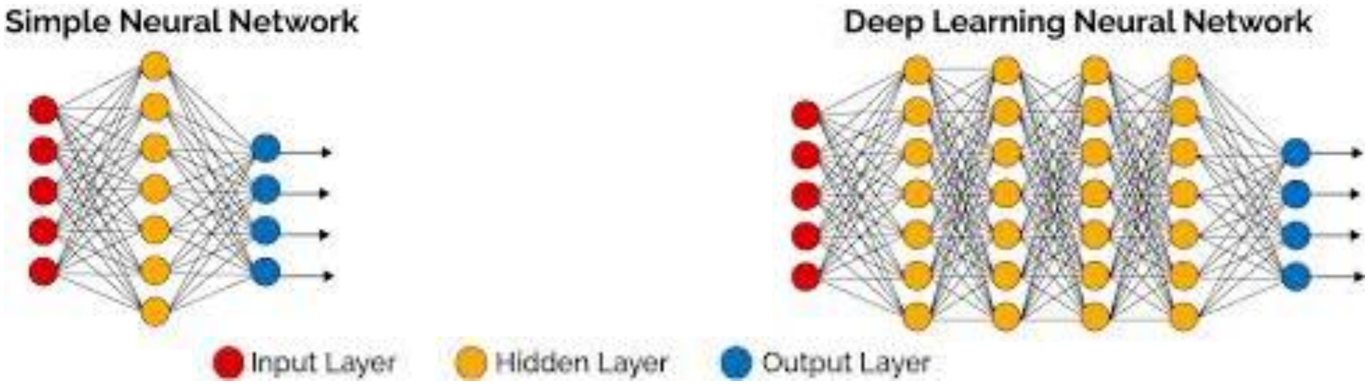
Third Wave

- The **third wave** of AI comes with Machine Learning (ML). Arthur Samuel in 1959 defined ML as **'giving computers the ability to learn without explicitly programmed'**. ML generally constructs a model that generated from one or more algorithms. Then, the model can be used to make predictions on new data.
- In simplified terms, every complex question can be replaced with a binary question. For example, the complex question 'Which author do you like?', can be replaced with binary questions like 'Do you like Haruki Murakami?', 'Do you like Milan Kundera?', 'Do you like Italo Calvino?'... Once in binary form, ML will suggest the best outcome from its evaluation.
- Unfortunately, a machine must at first learn correctly, before it can do prediction correctly. In all the ML before next wave, a machine by itself is not able to decide what is appropriate data and algorithm(s). The third winter of AI – the **Feature Engineering and Algorithm Selection Problem**.



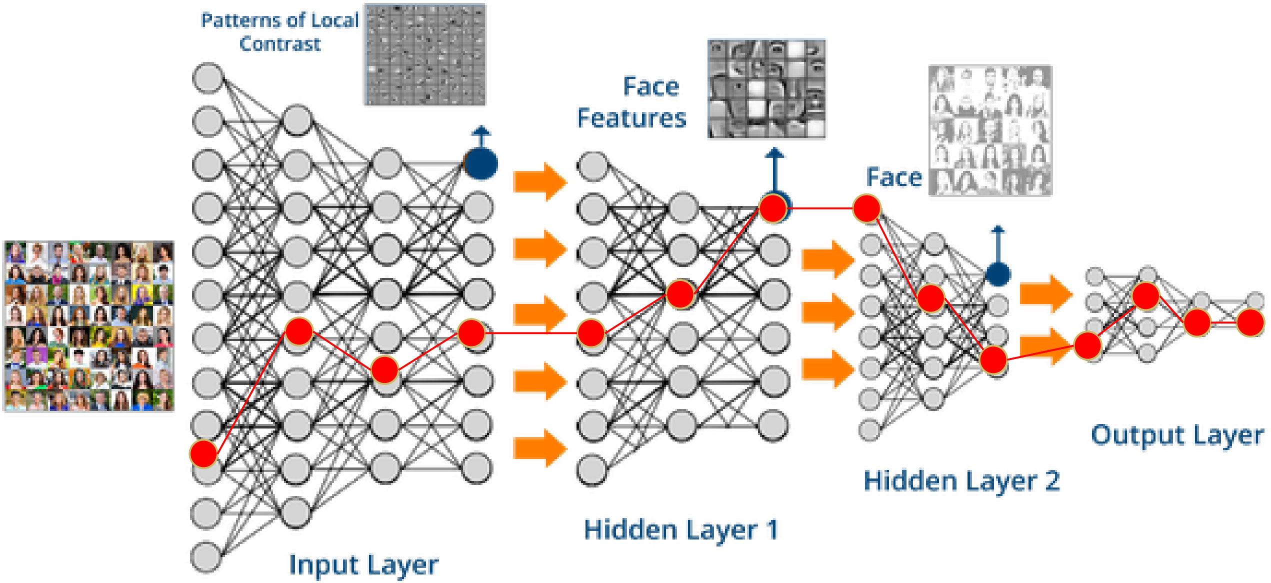
Fourth Wave

- **Deep Learning** (DL) is a new area of Machine Learning (ML) research, which has been introduced with the objective of moving ML closer to one of its original goals - Artificial Intelligence (AI).
- DL means using a neural network with **multiple layers of nodes** between input and output.
- The series of layers between input & output do feature identification and processing in a series of stages, just as our brains seem to.



Deep Neural Network (DNN)

- A deep neural network (**DNN**) is an artificial neural network (**ANN**) with multiple hidden layers between the input and output layers. DNNs are typically **feedforward** networks in which data flows from the input layer to the output layer without looping back.

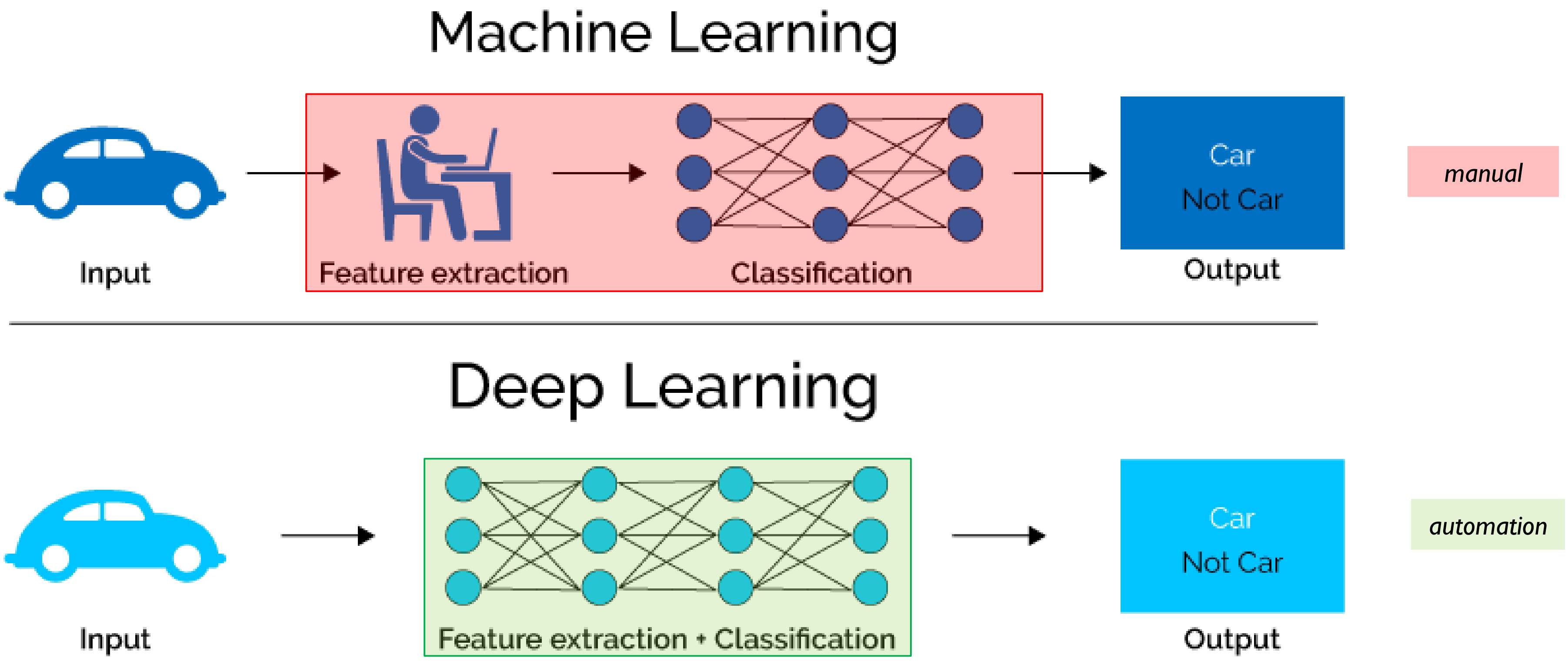


Deep Learning in ILSVRC 2012

- The **current hype** on AI is an extension of the ML, **Geoffrey Hinton** and his students managed to push neural network to next level, the **deep neural network**. In 2012, Geoffrey's **SuperVision** team stormed the world in ImageNet Large Scale Visual Recognition Challenge 2012 **(ILSVRC2012).**

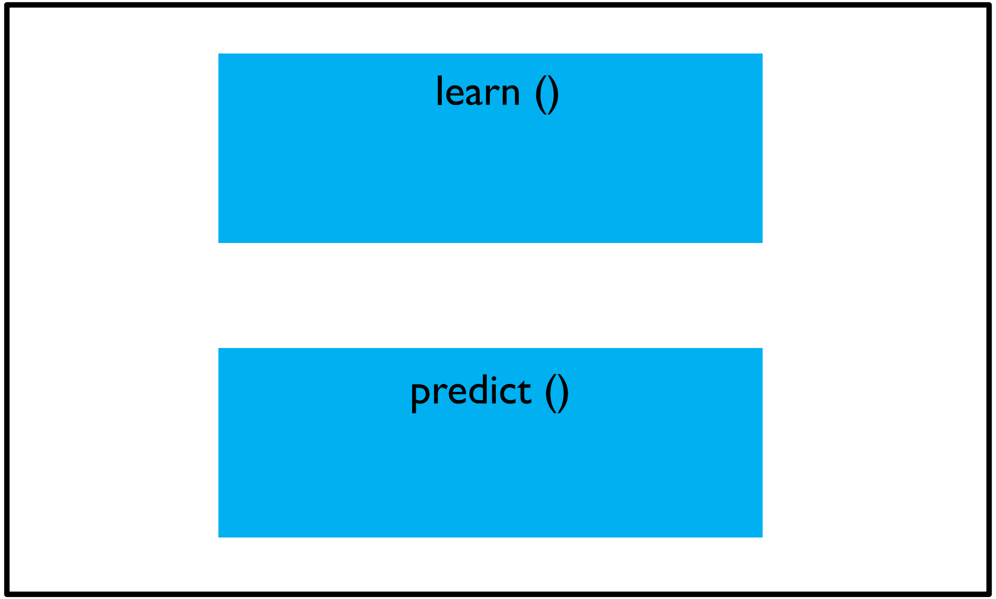
		Task		
	Team	Classification	Localization	Average
(Univ. of Toronto)	SuperVision	85%	66%	76%
(Univ. of Tokyo)	ISI	74%	46%	60%
(Univ. of Oxford)	OXFORD_VGG	73%	50%	61%
(Xerox Europe)	XRCE/INRIA	73%	N/A	N/A
	University of Amsterdam	70%	N/A	N/A

Deep Learning vs Machine Learning

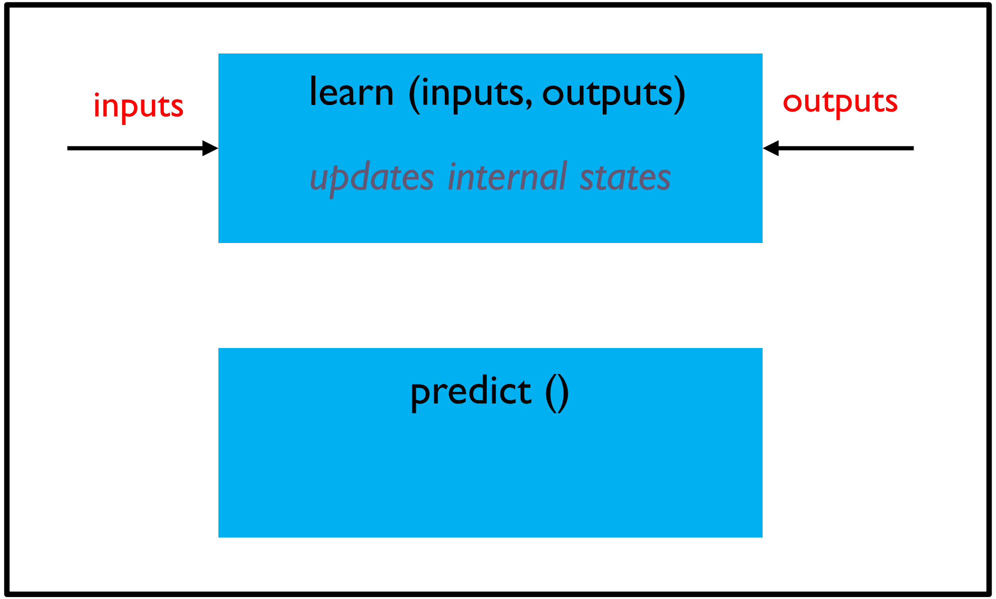


Automated ML: Backpropagation

- A supervised neural network, at the abstract representation, can be presented as a black box with 2 methods **learn()** and **predict()**

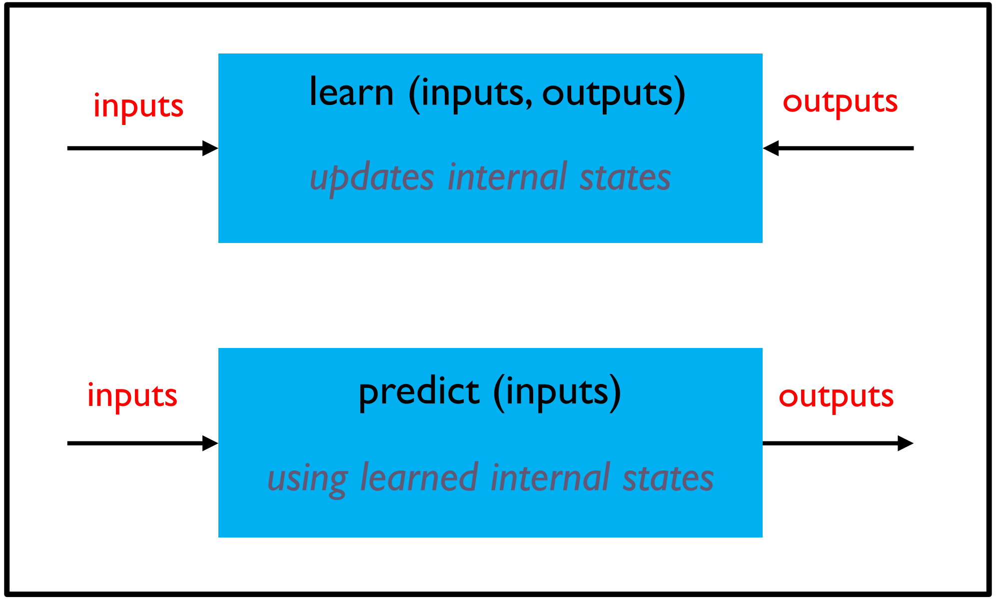


Automated ML: Backpropagation



the **learning** process takes the inputs and the desired outputs and updates its internal state accordingly, so the calculated output get as close as possible to the desired output

Automated ML: Backpropagation



the **predict** process takes input and using the learned internal state, to generate the most likely output according to its past "*learning experience*"

A Simple Example

- The example start with one input and one output and a linear relation between them.
- The goal of the supervised neural network is to try to search over all the possible linear functions which one best fits the data.
- For example, **y = *W*x**

(output = *weight* * input)

Input	Output
0	0
1	2
2	4
3	6
4	8
5	10

Automated ML: Backpropagation

Learning process **Step 1 - Model initialization**:

- A **random initialization** of the model is a common practice. The rational behind is that from wherever we start, if we are perseverant enough and through an iterative learning process, we can reach the pseudo-ideal model.
- Let's consider the following random initializations:
  - **(Model 1): y=3x**. The number ***3*** is generated at random.
  - **(Model 2): y=5x.** The number ***5*** is generated at random.
- We will explore later, how, through the learning process, all of these models can converge to the ideal solution **(y=2x)** (which we are trying to find).

Automated ML: Backpropagation

Learning process **Step 2 - Forward propagate**:

- The next step is to check performance of random initialized model.
- Start from the input, pass them through the network layer and calculate the actual output of the model.
- This step is called **forward-propagation**, because the calculation flow is going in the natural forward direction from the input -> through the supervised neural network -> to the output.

Input	Actual output of model 1 (y= 3x)
0	0
1	3
2	6
3	9
4	12
5	15

Automated ML: Backpropagation

Learning process **Step 3 - Loss function**:

- Defined a **loss function**, which is a performance metric on how well the algorithm manages to reach its goal of generating outputs as close as possible to the desired outputs.
- The most intuitive loss function is simply *loss = (desired output-actual output)*. This loss function returns positive values when the network **undershoot (prediction < desired output),** and negative values when the network **overshoot (prediction > desired output)**.
- If we want the loss function to reflect an **absolute error**, regardless if it's overshooting or undershooting we can define it as: *loss = absolute value of (desired—actual).*

Input	Actual output of model 1 (y= 3x)	Desired output	Absolute error
0	0	0	0
1	3	2	1
2	6	4	2
3	9	6	3
4	12	8	4
5	15	10	5

Automated ML: Backpropagation

Learning process **Step 3 - Loss function**:

- Several situations can lead to the same total sum of errors. For instance, lot of small errors or few big errors.
- We would like the prediction to work under **any** situation, it is more preferable to have a distribution of lot of small errors, rather than a few big ones.
- Hence, we define the loss function to be the **sum of squares** of the absolute errors.

Input	Actual output of model 1 (y= 3x)	Desired output	Absolute error	Square of absolute error
0	0	0	0	0
1	3	2	1	1
2	6	4	2	4
3	9	6	3	9
4	12	8	4	16
5	15	10	5	25
				Total 55

In short, the machine learning goal becomes to **minimize the loss function** (to reach as close as possible to 0).

Automated ML: Backpropagation

Learning process **Step 4 - Differentiation**:

- Goal is to **minimize the loss function**.
- **Differentiation** in mathematics can guide us how to optimize the weights.
- Imagine, how much the total error will change if the internal weight changed!
- Let's consider $\Delta W=0.0001$. (In reality it should be much smaller!)
- Let's recalculate the **sum of squares** of the absolute errors.

Input	Actual output of model 1 (y= 3.0001x)	Desired output	Square of absolute error
0	0	0	0
1	3.0001	2	1.0002
2	6.0002	4	4.0008
3	9.0003	6	9.0018
4	12.0004	8	16.0032
5	15.0005	10	25.0050
		Total	55.0110

Automated ML: Backpropagation

Learning process **Step 4 - Differentiation**:

- If we increase **W** from 3 to 3.0001, the sum of squares of error will increase from 55 to 55.0110.
- What is the rate of which the error changes **relatively** to the changes on the weight?
- The rate is **increase** by 0.0110 in the total error for each 0.0001 increase in weight.

Input	Actual output of model 1 (y= 3.0001x)	Desired output	Square error
0	0	0	0
1	3.0001	2	1.0002
2	6.0002	4	4.0008
3	9.0003	6	9.0018
4	12.0004	8	16.0032
5	15.0005	10	25.0050
		Total	55.0110

Automated ML: Backpropagation

Learning process **Step 4 - Differentiation**:

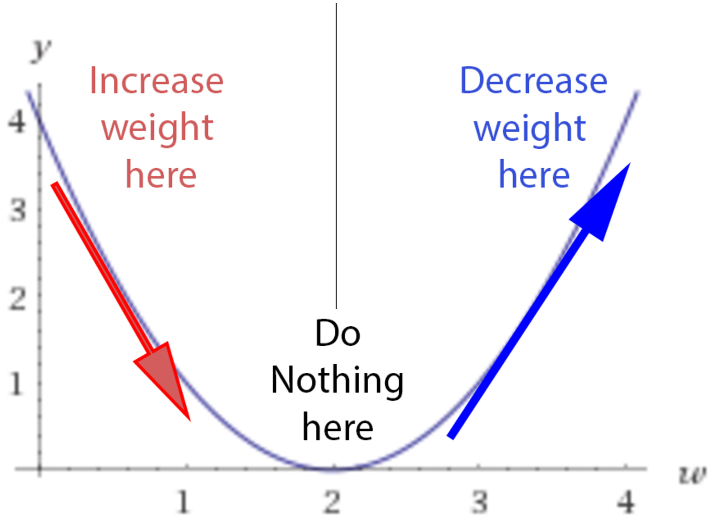
- If we decrease the weights by 0.0001, the sum of squares of error will decrease from 55 to 54.98900055.
- The rate is **decrease** by 0.0110 in the total error for each 0.0001 decrease in weight.

Input	Actual output of model 1 (y= 2.9999x)	Desired output	Square error
0	0	0	0
1	2.9999	2	0.99980001
2	5.9998	4	3.99920004
3	8.9997	6	8.99820009
4	11.9996	8	15.99680016
5	14.9995	10	24.99500025
		Total	54.98900055

Automated ML: Backpropagation

Learning process **Step 4 - Differentiation**:

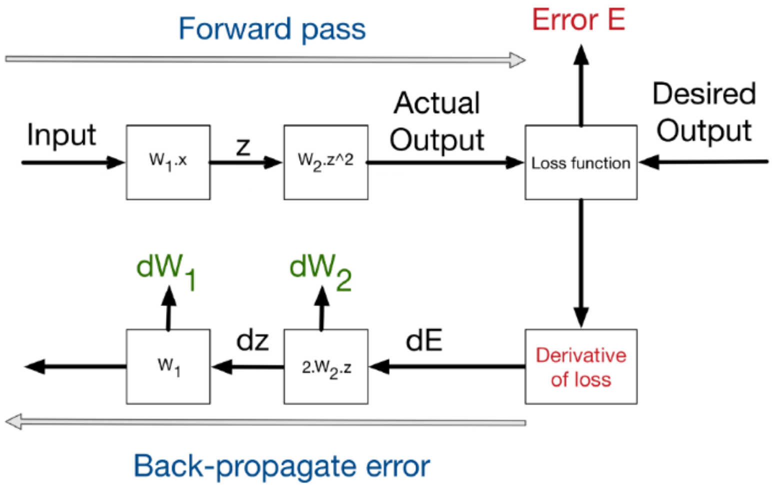
- Let's check the derivative, the rate at which error changes relatively to the changes on the weight.
- If *W* is < *desired W*, we have a positive loss function, but the derivative is negative, meaning that an increase of weight will decrease the loss function.
- If *W* is > *desired W*, we have a positive loss function, but the derivative is as well positive, meaning that any more increase in the weight, will increase the losses even more!
- If *W* is equal to *desired W*, we do nothing, we reach our stable point.



Automated ML: Backpropagation

Learning process **Step 5- Back-propagation**:

- A more complex problem – say **layer 1** is doing *3x* to generate a hidden output *z*, **layer 2** is doing z^2 to generate the final output.
- In order to solve the problem, luckily for us, derivative is **decomposable**, thus can be back-propagated.
- The process of back-propagating errors: Input -> Forward calls -> Loss function -> derivative -> back-propagation of errors.



Automated ML: Backpropagation

Learning process **Step 6- Weight Update**:

- A general rule of weight updates is the **delta rule.**
  new weight = old weight—Derivative Rate * learning rate
- The **learning rate** is introduced as a constant (usually very small), in order to force the weight to get updated very smoothly and slowly (to avoid big steps and chaotic behavior).
- If the derivative rate is positive, it means that an increase in weight will increase the error, thus the new weight should be smaller.
- If the derivative rate is negative, it means that an increase in weight will decrease the error, thus we need to increase the weights.
- If the derivative is 0, it means that we are in a stable minimum. Thus, no update on the weights is needed -> we reached a stable state.

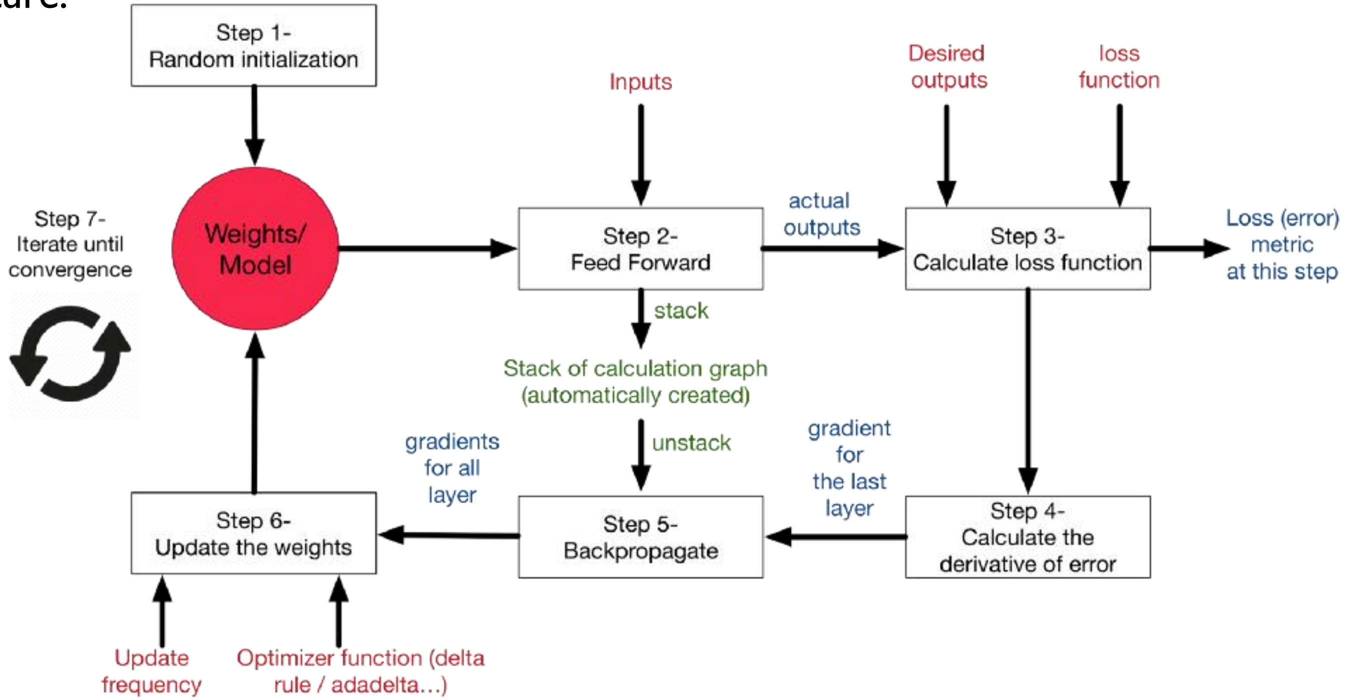
Automated ML: Backpropagation

Learning process **Step 7- Iterate Until Convergence**:

- A general rule of weight updates is the **delta rule.**
  new weight = old weight—Derivative Rate * learning rate
- The **learning rate** is introduced as a constant (usually very small), in order to force the weight to get updated very smoothly and slowly (to avoid big steps and chaotic behavior).
- If the derivative rate is positive, it means that an increase in weight will increase the error, thus the new weight should be smaller.
- If the derivative rate is negative, it means that an increase in weight will decrease the error, thus we need to increase the weights.
- If the derivative is 0, it means that we are in a stable minimum. Thus, no update on the weights is needed -> we reached a stable state.

Automated ML: Backpropagation

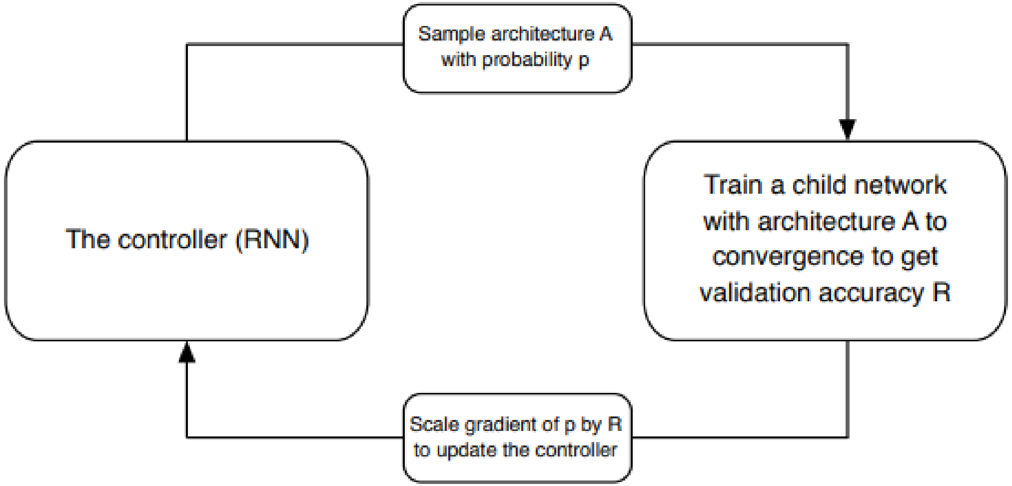
Overall picture:



Automated ML: Neural Architecture Search (NAS)

- Developing neural network models requires specialized skills and is challenging in general.
- NAS is an algorithm that *searches* for the **best *neural network architecture***:
  - Start off by defining a set of "building blocks" that can possibly be used for our network.
  - A controller Recurrent Neural Network (RNN) samples these building blocks, putting them together to create some kind of end-to-end architecture.
  - This new network architecture is then trained to convergence to obtain some accuracy on a held-out validation set.
  - The resulting accuracies are used to update the controller so that the controller will generate better architectures over time, perhaps by selecting better blocks or making better connections.
  - The controller weights are updated with policy gradient.

Automated ML: Neural Architecture Search (NAS)



Overview of Neural Architecture Search [71]. A controller RNN predicts architecture A from a search space with probability p. A child network with architecture A is trained to convergence achieving accuracy R. Scale the gradients of p by R to update the RNN controller.

Automated ML: Advances in Architecture Search

- NAS was quite inefficient and inaccessible to the common user. Using *450 GPUs* it took *3–4 days* of training to find that great architecture!
- Much of the latest research in NAS has thus focused on make this process more efficient.
- Progressive Neural Architecture Search (PNAS) proposes to use what is called a sequential model-based optimization (SMBO) strategy, rather than the reinforcement learning used in NASNet. This method of PNAS is 5–8 times more efficient (and thus much less expensive) than the original NAS.
- Efficient Neural Architecture Search (ENAS) is another shot at trying to make the general architecture search more efficient. The ENAS algorithm forces all models to share weights instead of training from scratch to convergence. Thus, we're essentially doing a transfer learning each time we train a new model, converging much faster! ENAS completed half a day of training with a single 1080Ti GPU!

Automated Machine Learning

- Google recently offering Cloud AutoML. Just upload your data and Google's NAS algorithm will find you an architecture, quick and easy!
- AutoKeras is an open source software library for automated machine learning.
- DataRobot's AutoML able to automate many of the tasks needed to develop artificial intelligence (AI) and machine learning applications.
- H2O's AutoML can be used for automating the machine learning workflow, which includes automatic training and tuning of many models within a user-specified time-limit.
- R's AutoML package fits from simple regression to highly customizable deep neural networks either with gradient descent or metaheuristic, using automatic hyper parameters tuning and custom cost function.
- Python's AutoML library automated machine learning for production and analytics.
- Auto-sklearn is an automated machine learning toolkit and a drop-in replacement for a scikit-learn estimator.

Thank You

Setup Parse

PARSE CONFIGURATION

Sources X_train.csv

ID Key_Frame__X_train1.hex

Parser CSV

Separator ,: '044'

Column Headers
- Auto
- First row contains column names
- First row contains data

Options
- Enable single quotes as a field quotation character
- Delete on done

EDIT COLUMN NAMES AND TYPES

Search by column name...

1	atm_type	Enum	0.0	0.0	0.0	0.0
2	branch_name	Enum	127.0	127.0	127.0	127.0
3	is_weekend	Enum	0.0	0.0	0.0	1.0
4	last_reloac	Numeric	0.920579633271809	0.920579633271809	0.920579633271809	0.920579633271809
5	last_reloac	Numeric	-1.1891177907825474	-1.0697977496481363	-1.0	-0.9504777085137253

Run AutoML

Project Name: atm-pred

Training Frame: Key_Frame__X_train.hex

Response Column: out_of_cash

Fold Column: (Select)

Weights Column: (Select)

Ignored Columns: Search...

Showing page 1 of 1.

☐ lat	REAL
☐ lon	REAL
☐ p2d_total_error_txn	REAL
☐ p2d_total_wd_amt	REAL
☐ p2d_total_wd_amt_since_last_reload	REAL
☐ p7d_total_wd_amt_since_last_reload	REAL
☐ term_id	ENUM(22880)
☐ txn_dow	ENUM(48976)
☐ txn_wom	ENUM(38387)

☑ All ☐ None

Only show columns with more than 0 % missing values.

Validation Frame: Key_Frame__X_val.hex

Leaderboard Frame: (Select)

Balance classes: ☐

Job

Run Time	02:00:03.90
Remaining Time	00:00:00.0
Type	Auto Model
Key	atm-pred
Description	AutoML build
Status	DONE
Progress	100%
	Done.
Actions	View

Leaderboard

Monitor Live

MODELS

models sorted in order of AUC, best first

	model_id	auc	logloss	mean_per_class_error	rmse	mse
0	GBM_grid_0_AutoML_20181113_120203_model_203	0.5073806953900392	10.923929023622673	0.43484550436215463	0.7248732213641473	0.525441187050836 1
1	DeepLearning_0_AutoML_20181113_120203	0.5093358411996883	4.200541759876456	0.49060827468162016	0.9719648644559118	0.9447156977367991
2	GBM_grid_0_AutoML_20181113_120203_model_22	0.655638867244881	3.117759938124701	0.3308946568395775	0.6236795709017424	0.3889762071601815
3	GBM_grid_0_AutoML_20181113_120203_model_112	0.7509077177564603	0.6923613534615316	0.42931217826714524	0.4952430246110214	0.245265653425872 7
4	GBM_grid_0_AutoML_20181113_120203_model_152	0.7661281566543615	0.5875090967587011	0.39587910508769014	0.37753723130834616	0.14253436102397166
5	GBM_grid_0_AutoML_20181113_120203_model_9	0.7668690203993	0.7058830392916294	0.30092560067444146	0.5063186183162819	0.2563585432537087
6	GBM_grid_0_AutoML_20181113_120203_model_25	0.76881857779708	0.3154900866750758	0.3965121923180094	0.310038317557286	0.09612375835375 25
7	GLM_grid_0_AutoML_20181113_120203_model_0	0.7725412584464854	0.277407664796 1704	0.40100926718356816	0.263483103685267 1	0.06942334592762123
8	GBM_grid_0_AutoML_20181113_120203_model_154	0.7777895755361682	0.523824542623807 8	0.41311616232580123	0.38614071520705096	0.14910465194061281
9	GBM_grid_0_AutoML_20181113_120203_model_34	0.7802396949685133	0.684164250558903 7	0.39151056369512177	0.4561002108731322	0.20802740235851566
10	GBM_grid_0_AutoML_20181113_120203_model_103	0.7806436573121974	0.755575279116592	0.3829778571154854	0.42974948528472356	0.18468462010248482
11	GBM_grid_0_AutoML_20181113_120203_model_82	0.7810476196558815	1.053313725620628 4	0.38483480654514796	0.5684054601211439	0.323084767095529 3
12	GBM_grid_0_AutoML_20181113_120203_model_39	0.7812208604238252	0.5278357305219928	0.39049107770127867	0.4166878846553893	0.17362879321858302
13	GBM_grid_0_AutoML_20181113_120203_model_67	0.784371926373472	1.2699168097620763	0.4362458007076526	0.594085803801969	0.352937942279031 6
14	GBM_grid_0_AutoML_20181113_120203_model_220	0.7844294071812689	0.6286376740625584	0.4169801499610408	0.46553860864136315	0.2167261961357363
15	GBM_grid_0_AutoML_20181113_120203_model_98	0.7865458185905706	0.6897316831381888	0.393056158749217 6	0.4637718043169175	0.2150842864793692

Model

Model ID: GBM_grid_0_AutoML_20181113_120203_model_324

Algorithm: Gradient Boosting Machine

Actions: Refresh | Predict... | Download POJO | Download Model Deployment Package (MOJO) | Export | Inspect | Delete | Download Gen Model

- MODEL PARAMETERS
- SCORING HISTORY - LOGLOSS
- ROC CURVE - TRAINING METRICS , AUC = 0.997559
- ROC CURVE - VALIDATION METRICS , AUC = 0.804742
- VARIABLE IMPORTANCES
- TRAINING METRICS - CONFUSION MATRIX ROW LABELS: ACTUAL CLASS; COLUMN LABELS: PREDICTED CLASS
- VALIDATION METRICS - CONFUSION MATRIX ROW LABELS: ACTUAL CLASS; COLUMN LABELS: PREDICTED CLASS
- TRAINING METRICS - GAINS/LIFT TABLE
- VALIDATION METRICS - GAINS/LIFT TABLE
- OUTPUT
- OUTPUT - MODEL SUMMARY
- OUTPUT - SCORING HISTORY
- OUTPUT - TRAINING_METRICS
- DOMAIN
- OUTPUT - TRAINING_METRICS - METRICS FOR THRESHOLDS (BINOMIAL METRICS AS A FUNCTION OF CLASSIFICATION THRESHOLDS)
- OUTPUT - TRAINING_METRICS - MAXIMUM METRICS (MAXIMUM METRICS AT THEIR RESPECTIVE THRESHOLDS)
- OUTPUT - TRAINING_METRICS - GAINS/LIFT TABLE (AVG RESPONSE RATE: 52.87 %, AVG SCORE: 52.86 %)

Model

Model ID: GBM_grid_0_AutoML_20181113_120203_model_324

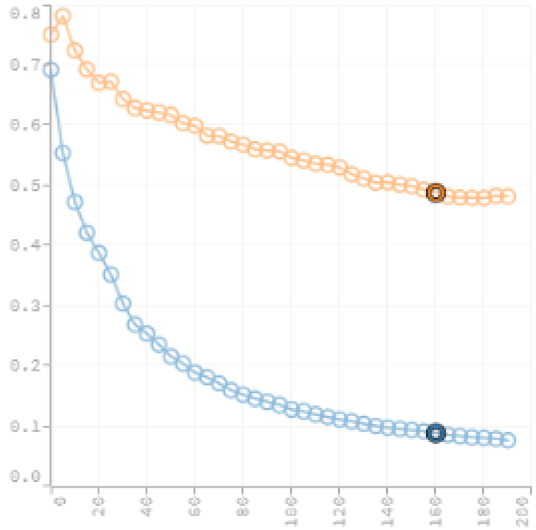
Algorithm: Gradient Boosting Machine

Actions: Refresh | Predict... | Download POJO | Download Model Deployment Package (MOJO) | Export | Inspect | Delete | Download Gen Model

MODEL PARAMETERS

Parameter	*Value*	*Description*
model_id	GBM_grid_0_AutoML_20181113_120203_model_324	Destination id for this model; auto-generated if not specified.
training_frame	_928dfd275f0bc66bf9d253e54f2a4f65	Id of the training data frame.
validation_frame	automl_validation_Key_Frame__X_train.hex	Id of the validation data frame.
keep_cross_validation_predictions	true	Whether to keep the predictions of the cross-validation models.
score_tree_interval	5	Score the model after every so many trees. Disabled if set to 0.
response_column	out_of_cash	Response variable column.
ignored_columns		Names of columns to ignore for training.
ntrees	10000	Number of trees.
min_rows	1	Fewest allowed (weighted) observations in a leaf.
stopping_rounds	3	Early stopping based on convergence of stopping_metric. Stop if simple moving average ol events (0 to disable)
stopping_tolerance	0.002851410881423444	Relative tolerance for metric-based stopping criterion (stop if relative improvement is not
max_runtime_secs	430	Maximum allowed runtime in seconds for model training. Use 0 to disable.
seed	-5783679485009037000	Seed for pseudo random number generator (if applicable)
distribution	bernoulli	Distribution function
sample_rate	0.8	Row sample rate per tree (from 0.0 to 1.0)
col_sample_rate	0.7	Column sample rate (from 0.0 to 1.0)
col_sample_rate_per_tree	0.4	Column sample rate per tree (from 0.0 to 1.0)

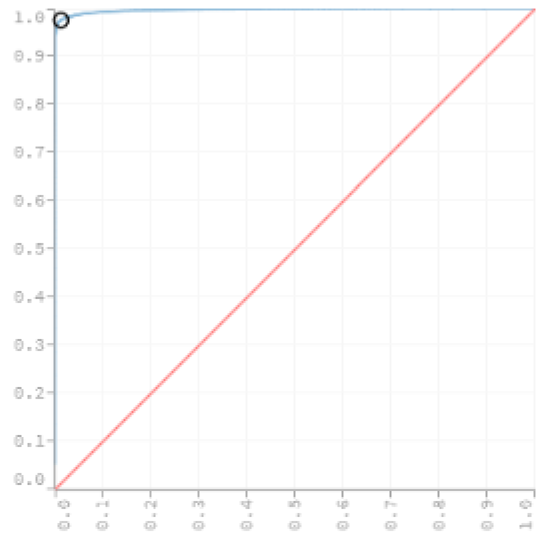
SCORING HISTORY - LOGLOSS



Selected mark(s):

timestamp	2018-11-13 13:55:00
duration	1:48:02.809
number_of_trees	160
training_rmse	0.1447
training_logloss	0.0896
training_auc	0.9969
training_lift	1.8914
training_classification_error	0.0207
validation_rmse	0.4138
validation_logloss	0.4899
validation_auc	0.8073
validation_lift	6.2549
validation_classification_error	0.0564

▾ ROC CURVE - TRAINING METRICS , AUC = 0.997559



Threshold: 0.46100644498097649

Criterion: max f1

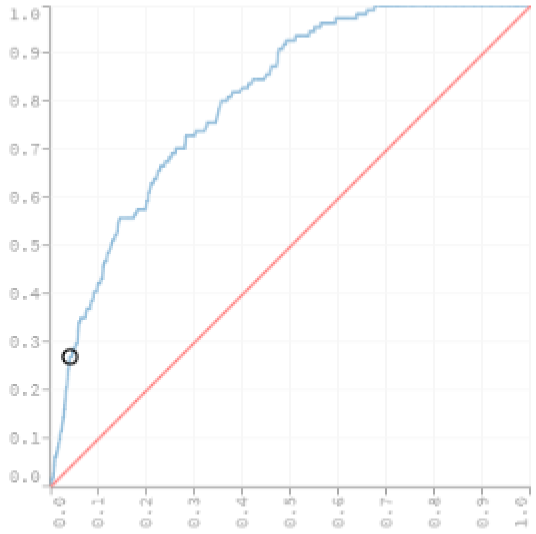
Selected mark(s):

threshold	0.4610
f1	0.9820
f2	0.9783
f0point5	0.9858
accuracy	0.9811
precision	0.9884
recall	0.9758
specificity	0.9871
absolute_mcc	0.9623
in_per_class_accuracy	0.9758
an_per_class_accuracy	0.9815
tns	57219
fns	1573
fps	747
tps	63454
tnr	0.9871
fnr	0.0242
fpr	0.0129
tpr	0.9758
idx	195

CM

Actual/Predicted	0.0	1.0	Error	Rate
0.0	57219	747	0.0129	747 / 57966
1.0	1573	63454	0.0242	1573 / 65027

ROC CURVE - VALIDATION METRICS , AUC = 0.804742



Threshold: 0.7981566463094314

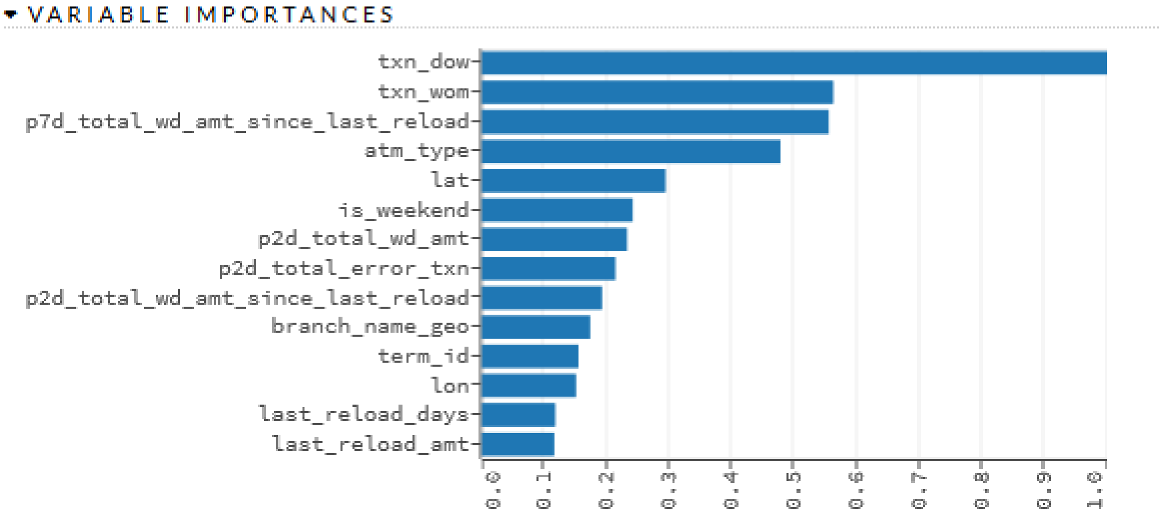
Criterion: max f1

Selected mark(s):

threshold	0.7982
f1	0.1685
f2	0.2177
f0point5	0.1375
accuracy	0.9447
precision	0.1224
recall	0.2703
specificity	0.9590
absolute_mcc	0.1563
min_per_class_accuracy	0.2703
mean_per_class_accuracy	0.6146
tns	5030
fns	81
fps	215
tps	30
tnr	0.9590
fnr	0.7297
fpr	0.0410
tpr	0.2703
idx	50

CM

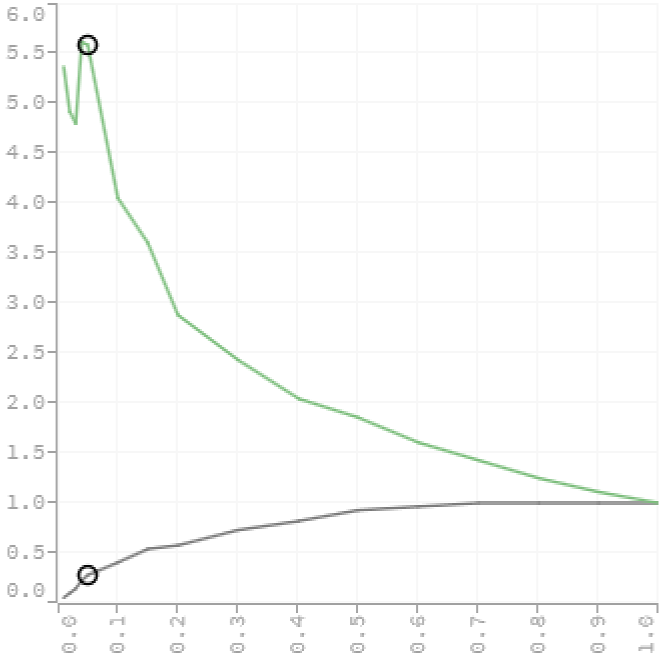
Actual/Predicted	0.0	1.0	Error	Rate
0.0	5030	215	0.0410	215 / 52
1.0	81	30	0.7297	81 / 111


VARIABLE IMPORTANCES
txn_dow
txn_wom
p7d_total_wd_amt_since_last_reload
atm_type
lat
is_weekend
p2d_total_wd_amt
p2d_total_error_txn
p2d_total_wd_amt_since_last_reload
branch_name_geo
term_id
lon
last_reload_days
last_reload_amt
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0

▾ TRAINING METRICS - CONFUSION MATRIX ROW LABELS: ACTUAL CLASS; COLUMN LABELS: PREDICTED CLASS

	0.0	1.0	Error	Rate	Recall
0.0	57219	747	0.0129	747 / 57,966	0.97
1.0	1573	63454	0.0242	1,573 / 65,027	0.99
Total	58792	64201	0.0189	2,320 / 122,993	
Precision	0.99	0.98			

VALIDATION METRICS - GAINS/LIFT TABLE



Selected mark(s):

group	5
cumulative_data_fraction	0.0500
lower_threshold	0.7885
lift	5.4625
cumulative_lift	5.5814
response_rate	0.1132
score	0.7972
cumulative_response_rate	0.1157
cumulative_score	0.8574
capture_rate	0.0541
cumulative_capture_rate	0.2793
gain	446.2519
cumulative_gain	458.1417

OUTPUT - SCORING HISTORY

timestamp	duration	number_of_trees	training_rmse	training_logloss	training_auc	training_lift
2018-11-13 13:54:54	1:47:57.111	0	0.4992	0.6915	0.5000	1.0
2018-11-13 13:54:55	1:47:57.253	5	0.4271	0.5541	0.9072	1.8914
2018-11-13 13:54:55	1:47:57.399	10	0.3837	0.4725	0.9217	1.8914
2018-11-13 13:54:55	1:47:57.548	15	0.3575	0.4214	0.9292	1.8914
2018-11-13 13:54:55	1:47:57.704	20	0.3411	0.3878	0.9369	1.8914
2018-11-13 13:54:55	1:47:57.862	25	0.3217	0.3519	0.9505	1.8914
2018-11-13 13:54:55	1:47:58.008	30	0.2923	0.3037	0.9700	1.8914
2018-11-13 13:54:55	1:47:58.160	35	0.2703	0.2688	0.9796	1.8914
2018-11-13 13:54:56	1:47:58.309	40	0.2617	0.2536	0.9816	1.8914
2018-11-13 13:54:56	1:47:58.461	45	0.2506	0.2355	0.9844	1.8914
2018-11-13 13:54:56	1:47:58.612	50	0.2379	0.2157	0.9871	1.8914
2018-11-13 13:54:56	1:47:58.795	55	0.2301	0.2040	0.9885	1.8914

References

1. https://ai100.stanford.edu/2016-report/appendix-i-short-history-ai
2. http://www.csis.pace.edu/~ctappert/dps/pdf/ai-chess-deep.pdf
3. https://skymind.ai/wiki/neural-network
4. https://medium.com/datathings/neural-networks-and-backpropagation-explained-in-a-simple-way-f540a3611f5e
5. http://ruder.io/optimizing-gradient-descent/
6. https://arxiv.org/pdf/1707.07012.pdf