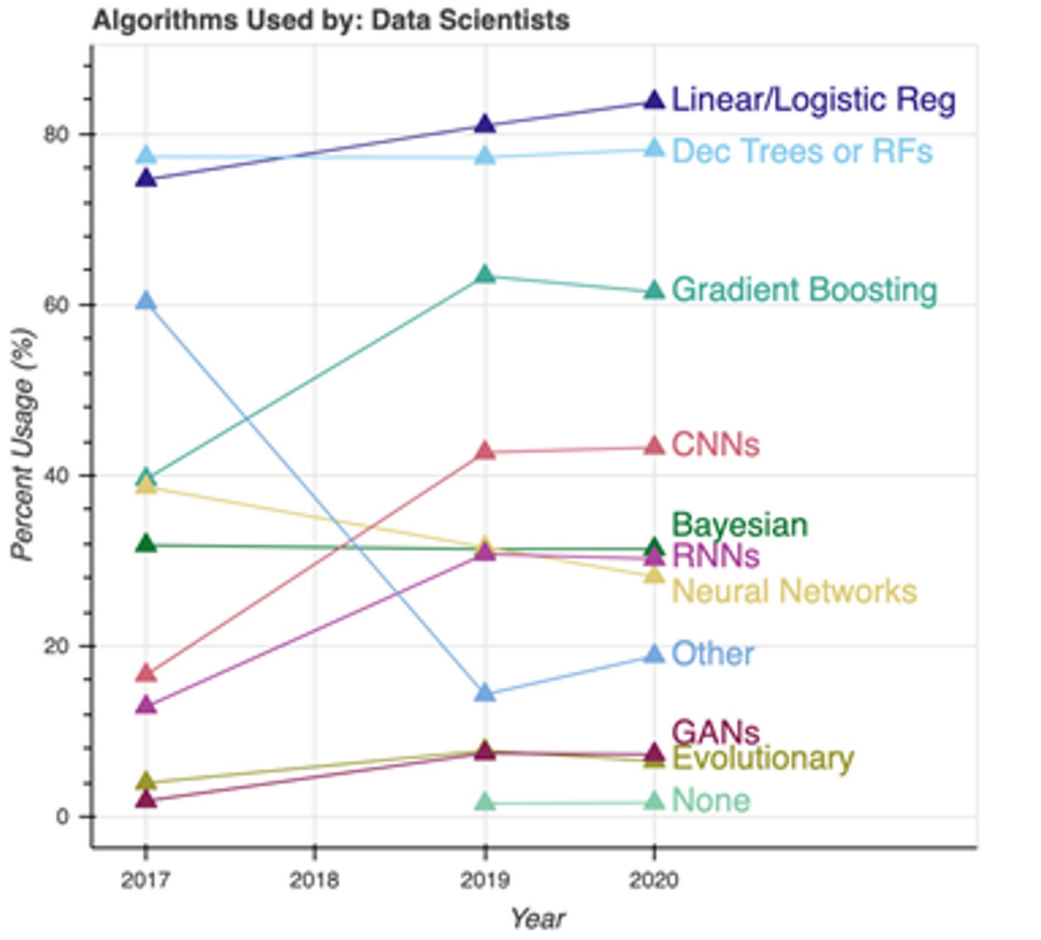
TRICKS OF THE TRADE:

Machine learning in the real world

Burning questions

- Which model to use?
- How should I improve my models?
  - Diagnosis and error analysis

What to use



https://medium.com/analytics-vidhya/ongoing-kaggle-survey-picks-the-topmost-data-science-trends-7c19ec7606a1

What's the best?

We evaluate **179 classifiers** arising from **17 families** (discriminant analysis, Bayesian, neural networks, support vector machines, decision trees, rule-based classifiers, boosting, bagging, stacking, random forests and other ensembles, generalized linear models, nearest-neighbors, partial least squares and principal component regression, logistic and multinomial regression, multiple adaptive regression splines and other methods), implemented in Weka, R (with and without the caret package), C and Matlab, including all the relevant classifiers available today. We use **121 data sets**, which represent **the whole UCI** data base (excluding the large-scale problems) and other own real problems, in order to achieve significant conclusions about the classifier behavior, not dependent on the data set collection. **The classifiers most likely to be the bests are the random forest** (RF) versions, the best of which (implemented in R and accessed via caret) achieves 94.1% of the maximum accuracy overcoming 90% in the 84.3% of the data sets. However, the difference is not statistically significant with the second best, the SVM with Gaussian kernel implemented in C using LibSVM, which achieves 92.3% of the maximum accuracy. A few models are clearly better than the remaining ones: random forest, SVM with Gaussian and polynomial kernels, extreme learning machine with Gaussian kernel, C5.0 and avNNet (a committee of multi-layer perceptrons implemented in R with the caret package). The random forest is clearly the best family of classifiers (3 out of 5 bests classifiers are RF), followed by SVM (4 classifiers in the top-10), neural networks and boosting ensembles (5 and 3 members in the top-20, respectively).

Most data sets are small (<1000) in this paper

http://jmlr.org/papers/volume15/delgado14a/delgado14a.pdf

Revisiting Deep Learning Models for Tabular Data

Yury Gorishniy[*†‡] **Ivan Rubachev**[†♣] **Valentin Khrulkov**[†] **Artem Babenko**[†♣]

† Yandex, Russia
‡ Moscow Institute of Physics and Technology, Russia
♣ National Research University Higher School of Economics, Russia

Abstract

The existing literature on deep learning for tabular data proposes a wide range of novel architectures and reports competitive results on various datasets. However, the proposed models are usually not properly compared to each other and existing works often use different benchmarks and experiment protocols. As a result, it is unclear for both researchers and practitioners what models perform best. Additionally, the field still lacks effective baselines, that is, the easy-to-use models that provide competitive performance across different problems.

In this work, we perform an overview of the main families of DL architectures for tabular data and raise the bar of baselines in tabular DL by identifying two simple and powerful deep architectures. The first one is a ResNet-like architecture which turns out to be a strong baseline that is often missing in prior works. The second model is our simple adaptation of the Transformer architecture for tabular data, which outperforms other solutions on most tasks. Both models are compared to many existing architectures on a diverse set of tasks under the same training and tuning protocols. We also compare the best DL models with Gradient Boosted Decision Trees and conclude that there is still no universally superior solution. The source code is available at https://github.com/yandex-research/rtdl.

https://arxiv.org/abs/2106.11959 2021

Table 1: Dataset properties. Notation: "RMSE" ~ root-mean-square error, "Acc." ~ accuracy.

	CA	AD	HE	JA	HI	AL	EP	YE	CO	YA	MI
#objects	20640	48842	65196	83733	98050	108000	500000	515345	581012	709877	1200192
#num. features	8	6	27	54	28	128	2000	90	54	699	136
#cat. features	0	8	0	0	0	0	0	0	0	0	0
metric	RMSE	Acc.	Acc.	Acc.	Acc.	Acc.	Acc.	RMSE	Acc.	RMSE	RMSE
#classes	–	2	100	4	2	1000	2	–	7	–	–

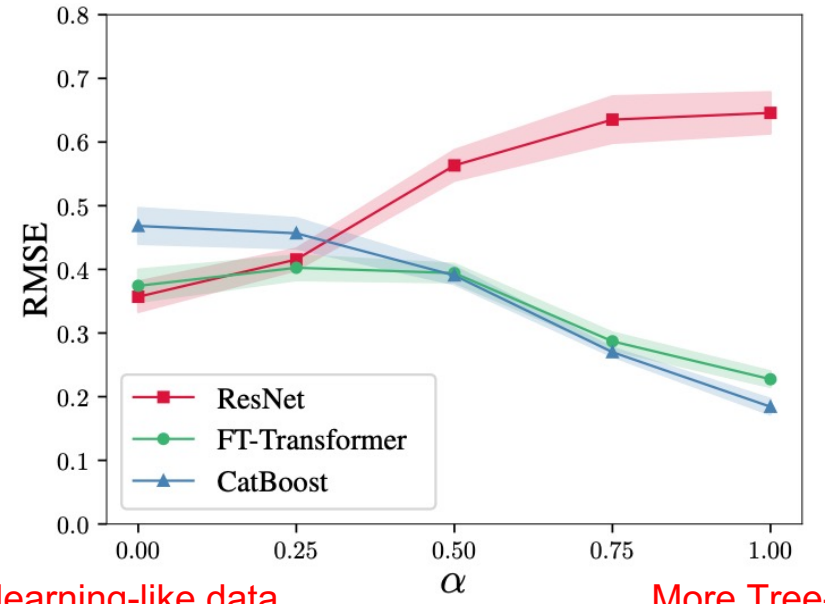
Table 4: Results for ensembles of GBDT and the main DL models. For each model-dataset pair, the metric value averaged over three ensembles is reported. See supplementary for standard deviations. Notation follows Table 3.

	CA ↓	AD ↑	HE ↑	JA ↑	HI ↑	AL ↑	EP ↑	YE ↓	CO ↑	YA ↓	MI ↓
Default hyperparameters											
XGBoost	0.462	**0.874**	0.348	0.711	0.717	0.924	0.8799	9.192	0.964	0.761	0.751
CatBoost	**0.428**	0.873	0.386	0.724	0.728	0.948	0.8893	8.885	0.910	0.749	0.744
FT-Transformer	0.454	0.860	**0.395**	**0.734**	**0.731**	**0.966**	**0.8969**	**8.727**	**0.973**	**0.747**	**0.742**
Tuned hyperparameters											
XGBoost	0.431	0.872	0.377	0.724	0.728	–	0.8861	8.819	0.969	**0.732**	0.742
CatBoost	**0.423**	**0.874**	0.388	0.727	0.729	–	0.8898	8.837	0.968	0.740	**0.741**
ResNet	0.478	0.857	0.398	0.734	0.731	0.966	0.8976	8.770	0.967	0.751	0.745
FT-Transformer	0.448	0.860	**0.398**	**0.739**	**0.731**	**0.967**	**0.8984**	**8.751**	**0.973**	0.747	0.743

Why is deep learning bad in tabular?

- Random dataset, generate from random tree/deep learning model.
- Gradient Boosting handles the mismatch better.

$$x \sim \mathcal{N}(0, I_k), \qquad y = \alpha \cdot f_{GBDT}(x) + (1-\alpha) \cdot f_{DL}(x).$$



More deep-learning-like data

More Tree-like data

NN are biased towards smooth functions (low frequency)

- While tree-based do not exhibits this bias.
  - Irregular functions hurt neural network performance
  - Smaller datasets are not as “smooth”

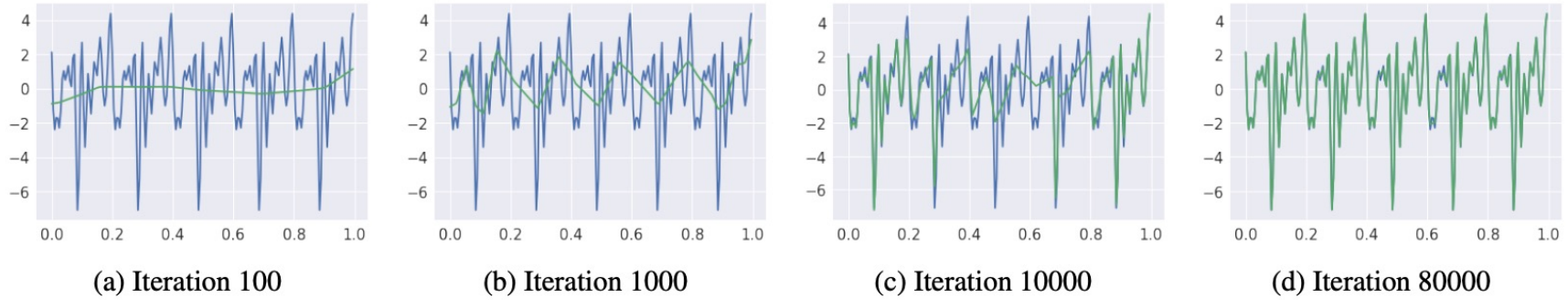


Figure 2. The learnt function (green) overlayed on the target function (blue) as the training progresses. The target function is a superposition of sinusoids of frequencies $\kappa = (5, 10, ..., 45, 50)$, equal amplitudes and randomly sampled phases.

On the Spectral Bias of Neural Networks (2018) https://arxiv.org/abs/1806.08734
Why do tree-based models still outperform deep learning on tabular data? (2022) https://arxiv.org/abs/2207.08815

NN are biased towards smooth functions (low frequency)

- While tree-based do not exhibits this bias.
  - Irregular functions hurt neural network performance
  - Smaller datasets are not as “smooth”

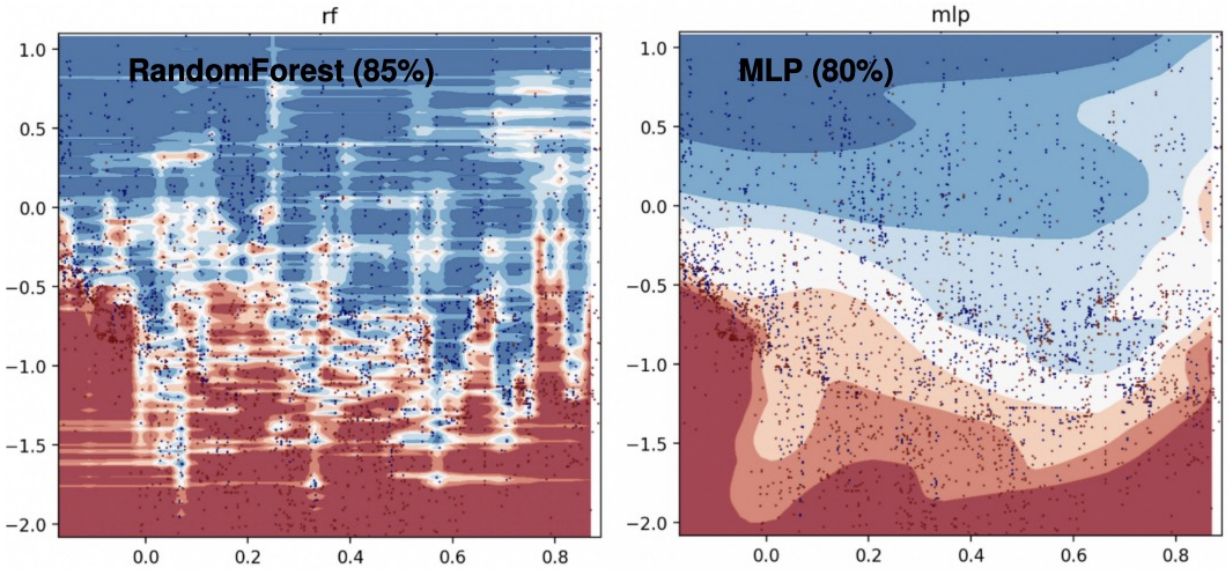


Figure 20: Decision boundaries of a default MLP and RandomForest for the 2 most important features of the *electricity* dataset

On the Spectral Bias of Neural Networks (2018) https://arxiv.org/abs/1806.08734
Why do tree-based models still outperform deep learning on tabular data? (2022) https://arxiv.org/abs/2207.08815

Neural network overfits easily

- Tabular datasets often contains uninformative or misleading features
- Deep learning gets hurt by this more than random forests

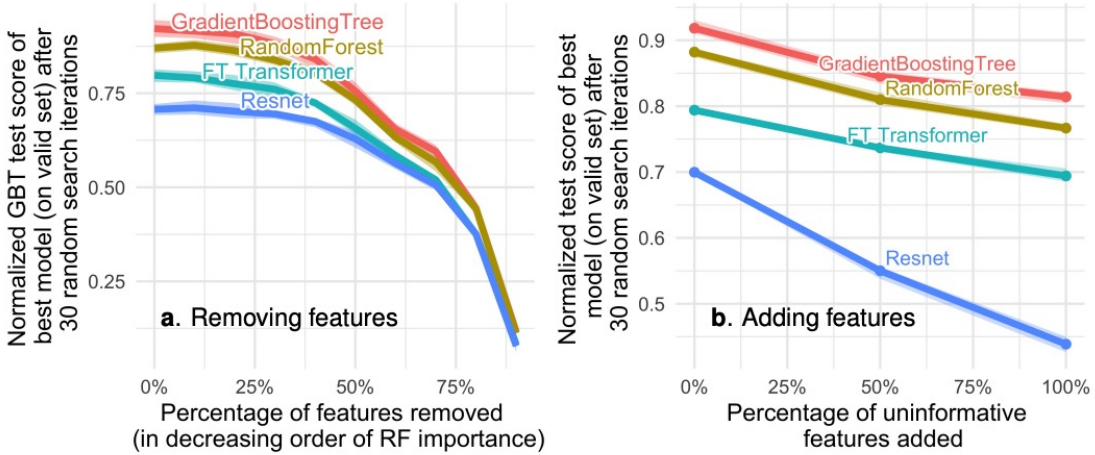


Figure 5: **Test accuracy changes when removing (a) or adding (b) uninformative features**. Features are removed in increasing order of feature importance (computed with a Random Forest). Added features are sampled from standard Gaussians uncorrelated with the target and with other features. Scores are averaged across datasets, and the ribbons correspond to the minimum and maximum score among the 30 different random search reorders (starting with the default models).

On the Spectral Bias of Neural Networks (2018) https://arxiv.org/abs/1806.08734
Why do tree-based models still outperform deep learning on tabular data? (2022) https://arxiv.org/abs/2207.08815

Neural networks are somewhat rotational invariant

- This helps in some task but hurts in tasks that do not require rotational invariance
  - Rotational invariant models suffer more from irrelavent features
    - See “Feature selection, L1 vs. L2 regularization, and rotational invariance” 2004 https://dl.acm.org/doi/10.1145/1015330.1015435

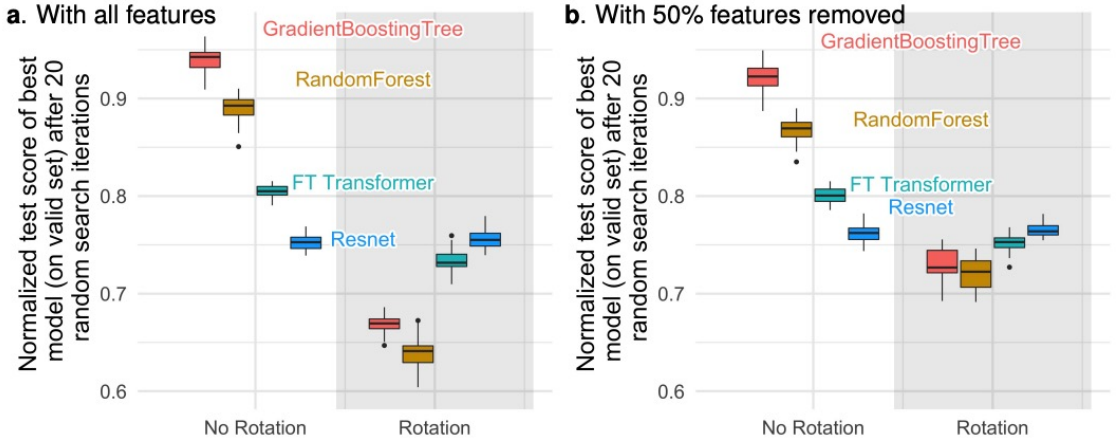


Figure 6: **Normalized test accuracy of different models when randomly rotating our datasets.** Here, the classification benchmark on numerical features was used. All features are Gaussianized before the random rotations. The scores are averaged across datasets, and the boxes depict the distribution across random search shuffles. Right: the features are removed before data rotation.

On the Spectral Bias of Neural Networks (2018) https://arxiv.org/abs/1806.08734

Why do tree-based models still outperform deep learning on tabular data? (2022) https://arxiv.org/abs/2207.08815

What can be done to improve deep learning in Tabular data?

FT-transformer

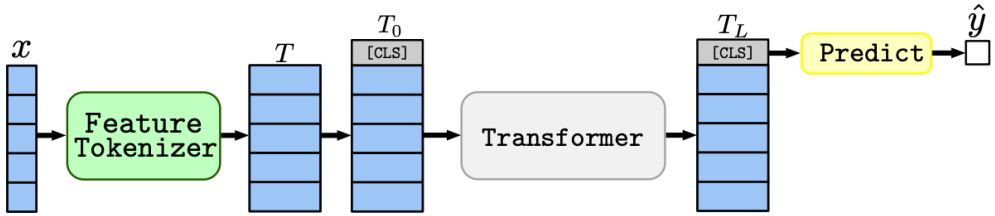


Figure 1: The FT-Transformer architecture. Firstly, Feature Tokenizer transforms features to embeddings. The embeddings are then processed by the Transformer module and the final representation of the [CLS] token is used for prediction.

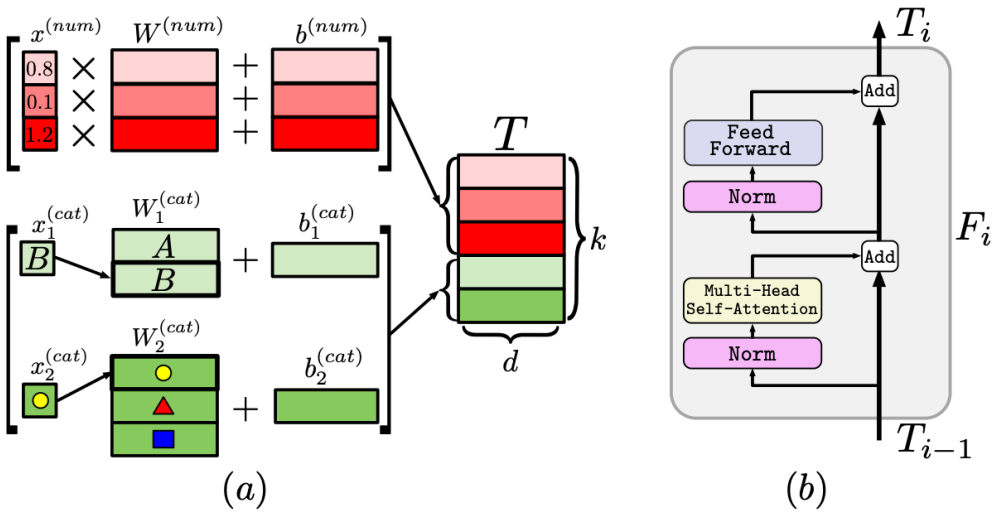


Figure 2: (a) Feature Tokenizer; in the example, there are three numerical and two categorical features; (b) One Transformer layer.

https://arxiv.org/abs/2106.11959

FT-transformer

Table 1: Dataset properties. Notation: "RMSE" ~ root-mean-square error, "Acc." ~ accuracy.

	CA	AD	HE	JA	HI	AL	EP	YE	CO	YA	MI
#objects	20640	48842	65196	83733	98050	108000	500000	515345	581012	709877	1200192
#num. features	8	6	27	54	28	128	2000	90	54	699	136
#cat. features	0	8	0	0	0	0	0	0	0	0	0
metric	RMSE	Acc.	Acc.	Acc.	Acc.	Acc.	Acc.	RMSE	Acc.	RMSE	RMSE
#classes	–	2	100	4	2	1000	2	–	7	–	–

Table 4: Results for ensembles of GBDT and the main DL models. For each model-dataset pair, the metric value averaged over three ensembles is reported. See supplementary for standard deviations. Notation follows Table 3.

	CA ↓	AD ↑	HE ↑	JA ↑	HI ↑	AL ↑	EP ↑	YE ↓	CO ↑	YA ↓	MI ↓
Default hyperparameters											
XGBoost	0.462	**0.874**	0.348	0.711	0.717	0.924	0.8799	9.192	0.964	0.761	0.751
CatBoost	**0.428**	0.873	0.386	0.724	0.728	0.948	0.8893	8.885	0.910	0.749	0.744
FT-Transformer	0.454	0.860	**0.395**	**0.734**	**0.731**	**0.966**	**0.8969**	**8.727**	**0.973**	**0.747**	**0.742**
Tuned hyperparameters											
XGBoost	0.431	0.872	0.377	0.724	0.728	–	0.8861	8.819	0.969	**0.732**	0.742
CatBoost	**0.423**	**0.874**	0.388	0.727	0.729	–	0.8898	8.837	0.968	0.740	**0.741**
ResNet	0.478	0.857	0.398	0.734	0.731	0.966	0.8976	8.770	0.967	0.751	0.745
FT-Transformer	0.448	0.860	**0.398**	**0.739**	**0.731**	**0.967**	**0.8984**	**8.751**	**0.973**	0.747	0.743

https://arxiv.org/abs/2106.11959

On the tuning

- Tree-based are much faster to tune with similar time budget.
- Regular MLP is a reasonable deep learning baseline for tabular data

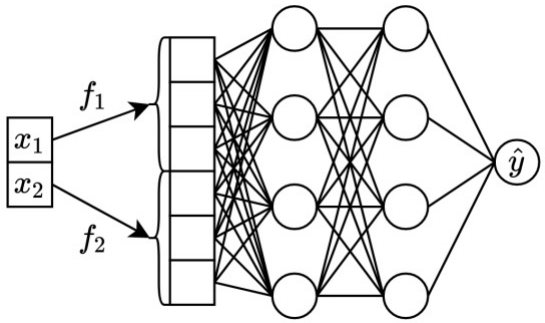
Table 11: Performance of tuned models with different tuning time budgets. Tuned model performance and the number of Optuna iterations (in parentheses) are reported (both metrics are averaged over five random seeds). Best results among DL models are in bold, overall best results are in bold red.

	0.25h	0.5h	1h	2h	3h	4h	5h	6h
				California Housing				
XGBoost	**0.437 (31)**	**0.436 (56)**	**0.434 (120)**	**0.433 (252)**	**0.433 (410)**	**0.432 (557)**	**0.433 (719)**	**0.432 (867)**
MLP	0.503(16)	0.496(42)	0.493(103)	0.488(230)	0.489(349)	0.489(466)	0.488(596)	0.488(724)
ResNet	0.488(7)	0.487(15)	0.483(30)	0.481(64)	0.482(101)	0.482(131)	0.482(164)	0.484(197)
FT-Transformer	**0.466 (4)**	**0.464 (9)**	**0.465 (20)**	**0.460 (47)**	**0.458 (74)**	**0.458 (99)**	**0.457 (124)**	**0.459 (153)**
				Adult				
XGBoost	**0.871 (165)**	**0.873 (311)**	**0.872 (638)**	**0.872 (1296)**	**0.872 (1927)**	**0.872 (2478)**	**0.872 (2999)**	**0.872 (3500)**
MLP	0.856(20)	0.857(37)	0.858(71)	0.857(130)	0.856(190)	0.856(247)	0.856(310)	0.856(375)
ResNet	0.856(8)	0.854(16)	0.854(32)	0.856(69)	0.855(105)	0.855(140)	0.856(174)	0.855(208)
FT-Transformer	**0.861 (6)**	**0.860 (12)**	**0.859 (27)**	**0.859 (52)**	**0.860 (78)**	**0.860 (99)**	**0.860 (125)**	**0.860 (148)**
				Higgs Small				
XGBoost	0.725(88)	0.725(153)	0.724(291)	0.725(573)	0.725(823)	0.726(1069)	0.725(1318)	0.725(1559)
MLP	0.721(16)	0.720(29)	0.723(62)	0.722(137)	0.724(220)	0.723(300)	0.724(375)	0.724(447)
ResNet	0.724(8)	0.727(14)	0.727(32)	0.728(61)	0.728(84)	0.728(107)	0.728(132)	0.728(154)
FT-Transformer	**0.727 (2)**	**0.729 (5)**	**0.728 (12)**	**0.728 (23)**	**0.729 (34)**	**0.729 (44)**	**0.730 (56)**	**0.729 (66)**

https://arxiv.org/abs/2106.11959

On embedding features for deep learning

- 1) Always use embedding features for categorical features
- 2) Bin numerical features -> project with embedding
- 3) Periodic features can help NN learn higher frequency functions



On Embeddings for Numerical Features in Tabular Deep Learning 2022 https://arxiv.org/abs/2203.05556

Binning numerical features

- Unsupervised by quatile
- Supervised by target label (imaging training a tree model)

$$\texttt{PLE}(x) = [e_1, \ldots, e_T] \in \mathbb{R}^T$$

$$e_t = \begin{cases} 0, & x < b_{t-1} \texttt{ AND } t > 1 \\ 1, & x \geq b_t \texttt{ AND } t < T \\ \frac{x - b_{t-1}}{b_t - b_{t-1}}, & \text{otherwise} \end{cases} \quad (1)$$

where PLE stands for "**p**eicewise **l**inear **e**ncoding". We provide the visualization in Figure 1.

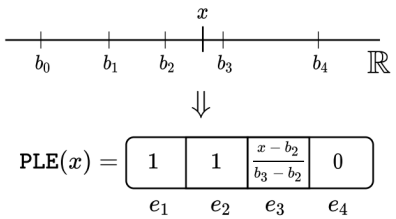


Figure 1: The piecewise linear encoding (PLE) in action for $T = 4$ (see Equation 1).

unsupervised

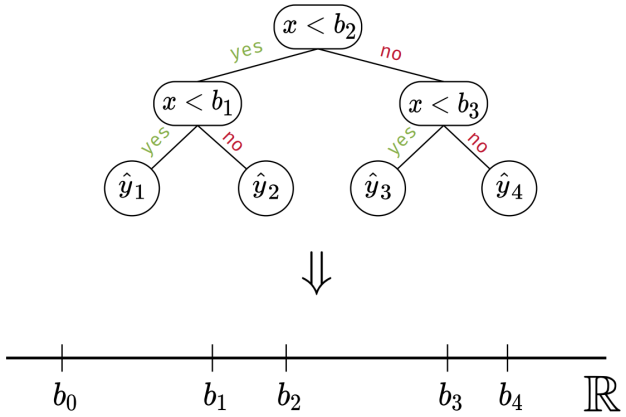


Figure 4: Obtaining bins for PLE from decision trees.

supervised

Periodic (fourier) features

It's been shown that using fourier features can help neural network learn high frequency functions better.

Figure 1: Fourier features improve the results of coordinate-based MLPs for a variety of high-frequency low-dimensional regression tasks, both with direct (b, c) and indirect (d, e) supervision. We visualize an example MLP (a) for an image regression task (b), where the input to the network is a pixel coordinate and the output is that pixel's color. Passing coordinates directly into the network (top) produces blurry images, whereas preprocessing the input with a Fourier feature mapping (bottom) enables the MLP to represent higher frequency details.

Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains 2020 https://arxiv.org/pdf/2006.10739.pdf

Learnable periodic features

- Can map input into (learnable) frequencies.

$$f_i(x) = \texttt{Periodic}(x) = \texttt{concat}[\sin(v),\ \cos(v)], \qquad v = [2\pi c_1 x,\ \ldots,\ 2\pi c_k x] \tag{2}$$

where c_i are trainable parameters initialized from $\mathcal{N}(0, \sigma)$. We observe that σ is an important hyperparameter. Both σ and k are tuned using validation sets.

On Embeddings for Numerical Features in Tabular Deep Learning 2022 https://arxiv.org/abs/2203.05556

On features

	CA ↓	HO ↓	HI ↑
XGBoost	0.436	3.160	0.724
XGBoost with `Periodic`	0.441	3.184	0.724
MLP	0.495	3.204	0.720
MLP-PL	0.467	3.113	0.727

Table 2: Embedding names. See subsection 4.3

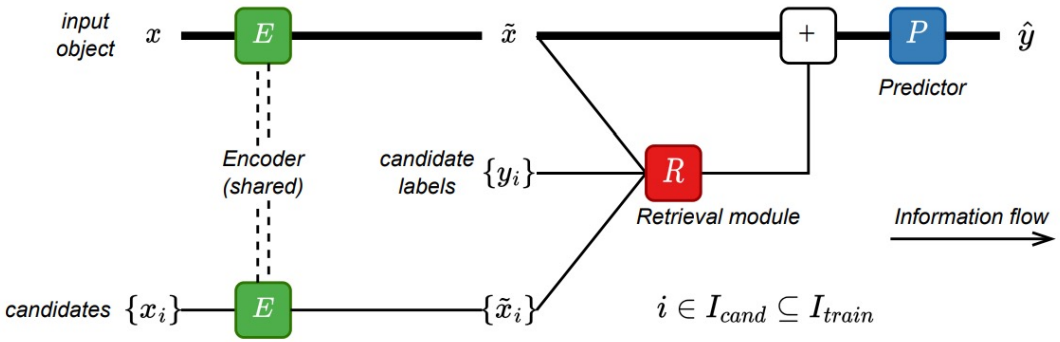
Name	Embedding function (f_i)
`L`	`Linear`
`LR`	`ReLU` ∘ `Linear`
`LRLR`	`ReLU` ∘ `Linear` ∘ `ReLU` ∘ `Linear`
`Q`	$\texttt{PLE}_q$
`Q-L`	`Linear` ∘ $\texttt{PLE}_q$
`Q-LR`	`ReLU` ∘ `Linear` ∘ $\texttt{PLE}_q$
`Q-LRLR`	`ReLU` ∘ `Linear` ∘ `ReLU` ∘ `Linear` ∘ $\texttt{PLE}_q$
`T`	$\texttt{PLE}_t$
`T-L`	`Linear` ∘ $\texttt{PLE}_t$
`T-LR`	`ReLU` ∘ `Linear` ∘ $\texttt{PLE}_t$
`T-LRLR`	`ReLU` ∘ `Linear` ∘ `ReLU` ∘ `Linear` ∘ $\texttt{PLE}_t$
`P`	`Periodic`
`PL`	`Linear` ∘ `Periodic`
`PLR`	`ReLU` ∘ `Linear` ∘ `Periodic`
`PLRLR`	`ReLU` ∘ `Linear` ∘ `ReLU` ∘ `Linear` ∘ `Periodic`

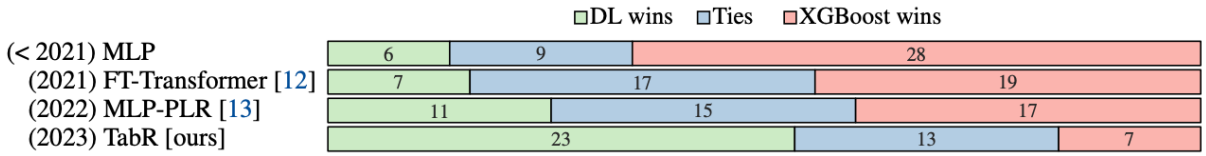
	GE ↑	CH ↑	CA ↓	HO ↓	AD ↑	OT ↑	HI ↑	FB ↓	SA ↑	CO ↑	MI ↓	Avg. Rank
CatBoost	0.692	0.861	0.430	3.093	0.873	0.825	0.727	5.226	0.924	0.967	**0.741**	3.6 ± 2.9
XGBoost	0.683	0.859	0.434	3.152	**0.875**	0.827	0.726	5.338	0.919	0.969	0.742	4.6 ± 2.7
MLP	0.665	0.856	0.486	3.109	0.856	0.822	0.727	5.616	0.913	0.968	0.746	8.5 ± 2.6
MLP-LR	0.679	0.861	0.463	3.012	0.859	0.826	0.731	5.477	0.924	0.972	0.744	5.5 ± 2.7
MLP-Q-LR	0.682	0.859	0.433	3.080	0.867	0.818	0.724	**5.144**	0.924	0.974	0.745	5.1 ± 1.9
MLP-T-LR	0.673	0.861	0.435	3.099	0.870	0.821	0.727	5.409	0.924	0.973	0.746	5.1 ± 1.7
MLP-PLR	**0.700**	0.858	0.453	**2.975**	0.874	**0.830**	**0.734**	5.388	**0.924**	0.975	0.743	3.0 ± 2.4
ResNet	0.690	0.861	0.483	3.081	0.856	0.821	0.734	5.482	0.918	0.968	0.745	6.7 ± 3.3
ResNet-LR	0.672	0.862	0.450	2.992	0.859	0.822	0.733	5.415	0.923	0.971	0.743	5.6 ± 2.7
ResNet-Q-LR	0.674	0.859	0.427	3.066	0.868	0.815	0.729	5.309	0.923	0.976	0.746	4.7 ± 2.0
ResNet-T-LR	0.683	0.862	**0.425**	3.030	0.872	0.822	0.731	5.471	0.923	0.975	0.744	4.1 ± 1.9
ResNet-PLR	0.691	0.861	0.443	3.040	**0.874**	0.825	0.734	5.400	0.924	0.975	0.743	3.2 ± 1.3
Transformer-L	0.668	0.861	0.455	3.188	0.860	0.824	0.727	5.434	0.924	0.973	0.743	5.9 ± 2.2
Transformer-LR	0.666	0.861	0.446	3.193	0.861	0.824	0.733	5.430	0.924	0.973	0.743	5.2 ± 2.2
Transformer-Q-LR	0.690	0.857	**0.425**	3.143	0.868	0.818	0.726	5.471	**0.924**	0.975	0.744	4.4 ± 2.2
Transformer-T-LR	0.686	0.862	**0.423**	3.149	0.871	0.823	0.733	5.515	0.924	**0.976**	0.744	3.7 ± 2.2
Transformer-PLR	0.686	**0.864**	0.449	3.091	0.873	0.823	0.734	5.581	**0.924**	0.975	0.743	3.9 ± 2.5

On Embeddings for Numerical Features in Tabular Deep Learning 2022 https://arxiv.org/abs/2203.05556

Retreival augment

- Can have a model retrieve relevant training data to augment the input to improve the performance further





TabR: Tabular Deep Learning Meets Nearest Neighbors in 2023 https://arxiv.org/abs/2307.14338

Retreival augment

Table 4: Comparing ensembles of TabR with ensembles of GBDT models. See subsection D.8 to learn how the "default" TabR-S was obtained. The notation follows Table 3.

	CH ↑	CA ↓	HO ↓	AD ↑	DI ↓	OT ↑	HI ↑	BL ↓	WE ↓	CO ↑	MI ↓	Avg. Rank
Tuned hyperparameters												
XGBoost	0.861	0.432	3.164	**0.872**	0.136	**0.832**	0.726	0.680	1.769	0.971	0.741	2.5 ± 0.9
CatBoost	0.859	0.426	3.106	**0.872**	0.133	0.827	0.727	0.681	1.773	0.969	**0.741**	2.5 ± 1.1
LightGBM	0.860	0.434	3.167	**0.872**	0.136	**0.832**	0.726	0.679	1.761	0.971	0.741	2.4 ± 0.9
TabR	**0.865**	**0.391**	**3.025**	**0.872**	**0.131**	**0.831**	**0.733**	**0.674**	**1.661**	**0.977**	0.748	1.3 ± 0.9
Default hyperparameters												
XGBoost	0.856	0.471	3.368	0.871	0.143	0.817	0.716	0.683	1.920	0.966	0.750	3.4 ± 0.9
CatBoost	0.861	0.432	3.108	**0.874**	0.132	0.822	**0.726**	0.684	1.886	0.924	**0.744**	2.1 ± 0.8
LightGBM	0.856	0.449	3.222	0.869	0.137	**0.826**	0.720	0.681	1.817	0.899	0.744	2.5 ± 0.9
TabR-S	**0.864**	**0.398**	**2.971**	0.859	**0.131**	0.824	0.724	0.688	**1.721**	**0.974**	0.752	2.0 ± 1.3

Table 14: Training times of the tuned configurations of several models. "MLP-EMB" is an MLP with the same embeddings for numerical features as the corresponding TabR on a given dataset. The format is `hh:mm:ss`.

	CH	CA	HO	AD	DI	OT	HI	BL	WE	CO	MI
XGBoost (GPU)	0:00:01	0:00:20	0:00:05	0:00:05	0:00:02	0:00:35	0:00:15	0:00:08	0:02:02	0:01:55	0:03:43
LightGBM (CPU)	0:00:01	0:00:04	0:00:01	0:00:01	0:00:03	0:00:34	0:00:10	0:00:07	0:06:40	0:06:22	0:06:45
MLP-EMB (GPU)	0:00:02	0:00:18	0:00:09	0:00:17	0:00:15	0:00:31	0:00:24	0:01:38	0:00:29	0:04:01	0:02:09
TabR (GPU)	0:00:16	0:00:40	0:00:55	0:01:30	0:01:24	0:01:47	0:06:22	0:04:14	1:03:18	0:37:03	1:46:07

Why retreival?

- Can do on-the-fly updates without full retraining by augmenting the retrieval pool with new data
- But suffers from slower training and slower inference
- Higher memory requirement

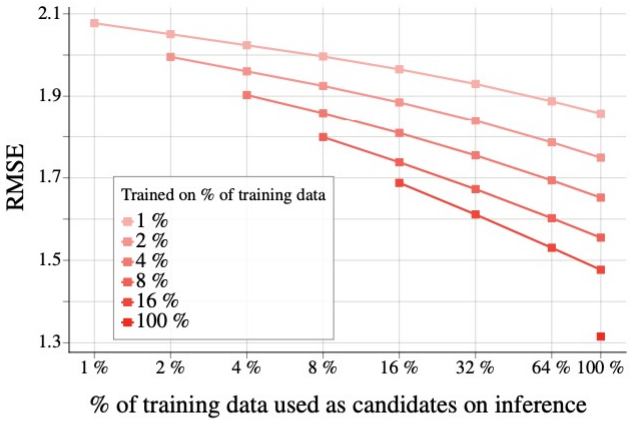
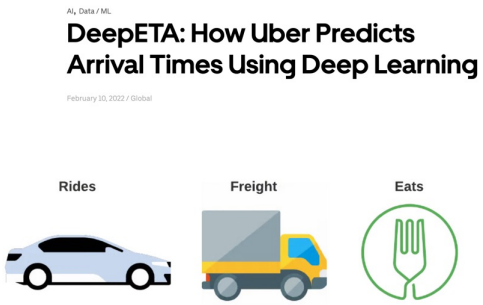


Figure 6: Training TabR-S on various portions of the training data of the full "Weather prediction" dataset and gradually adding the remaining unseen training data to the set of candidates *without retraining* as described in subsection 5.2. For each curve, the leftmost point corresponds to not adding any new data to the set of candidates after the training, and the rightmost point corresponds to adding all unseen training data to the set of candidates.

Is this really worth it?

- Maybe



For several years, Uber used gradient-boosted decision tree ensembles to refine ETA predictions. The ETA model and its training dataset grew steadily larger with each release. To keep pace with this growth, Uber's Apache Spark™ team contributed upstream improvements [1, 2] to XGBoost to allow the model to grow ever deeper, making it one of the largest and deepest XGBoost ensembles in the world at that time. Eventually, we reached a point where increasing the dataset and model size using XGBoost became untenable. To continue scaling the model and improving accuracy, we decided to explore deep learning because of the relative ease of scaling to large datasets using data-parallel SGD [3]. To justify switching to deep learning we needed to overcome three main challenges:

- **Latency:** The model must return an ETA within a few milliseconds at most.
- **Accuracy:** The mean absolute error (MAE) must improve significantly over the incumbent XGBoost model.
- **Generality:** The model must provide ETA predictions globally across all of Uber's lines of business such as mobility and delivery.

https://www.uber.com/en-ID/blog/deepeta-how-uber-predicts-arrival-times/

Cautionary notes

- “There is no free lunch.”
- The **“No Free Lunch”** theorem states that there is no one model that works best for every problem.
- Depends on
  - Nature of the task
  - Nature of the data
  - Amount of data
- Which model is the best?
  - Try it on your problem.

“Deep learning is not magical.”

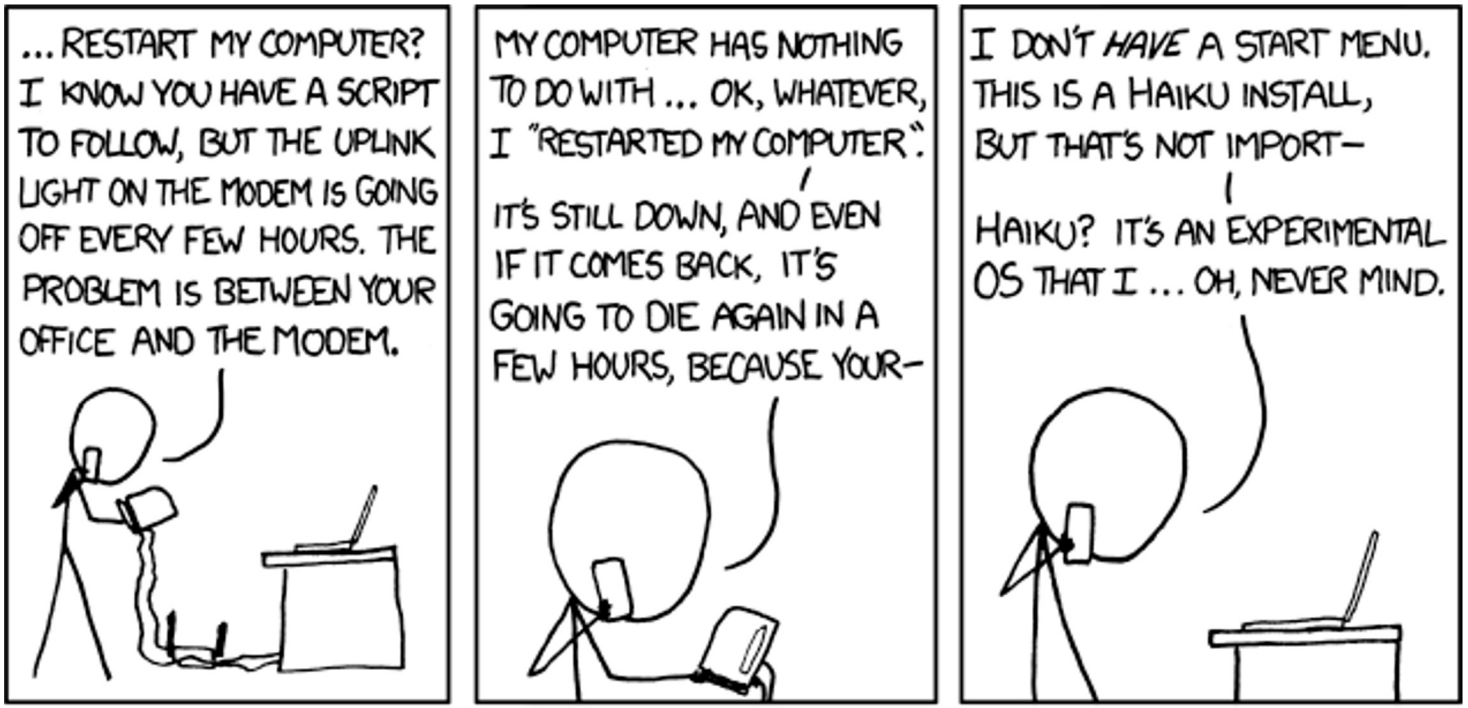
The Lack of A Priori Distinctions Between Learning Algorithms 1996
https://www.researchgate.net/publication/2755783_The_Lack_of_A_Priori_Distinctions_Between_Learning_Algorithms

What can deep learning do well?

- Unstructured data
  - Image, text (certain tasks), sensor signals, satellite image
- Generative tasks is pretty much only doable via deep learning

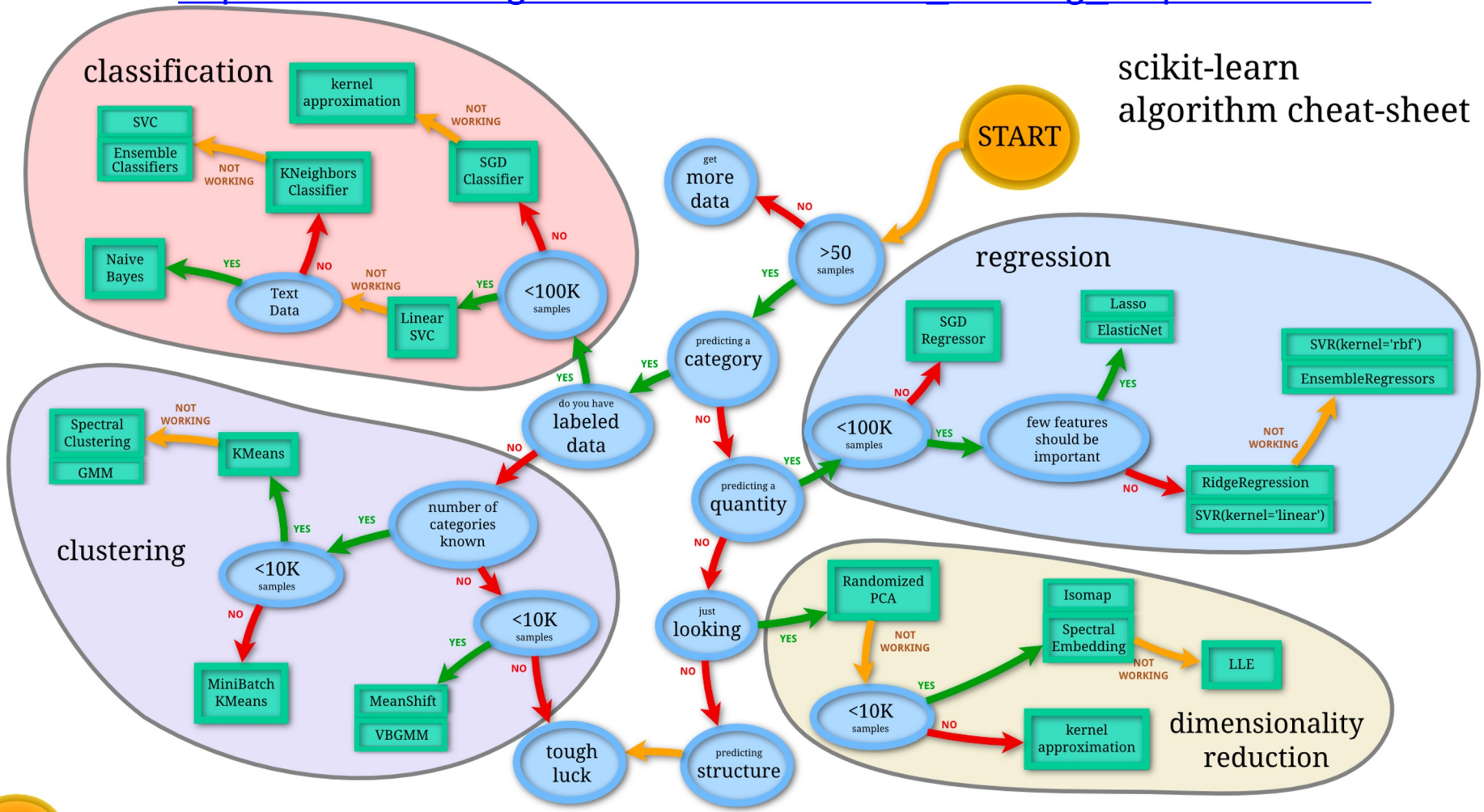
That's not so helpful…

Tech Support



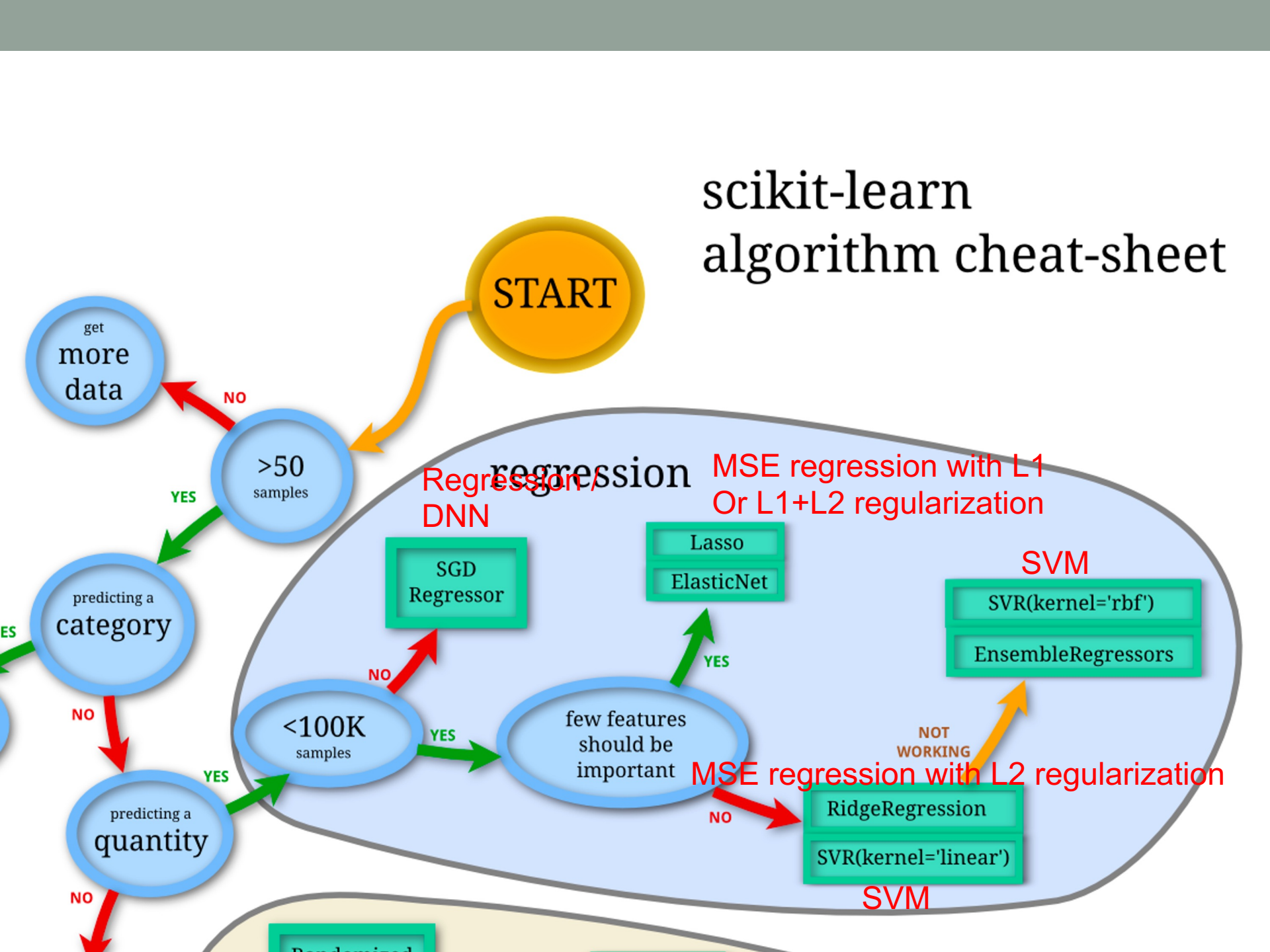
https://xkcd.com/806/

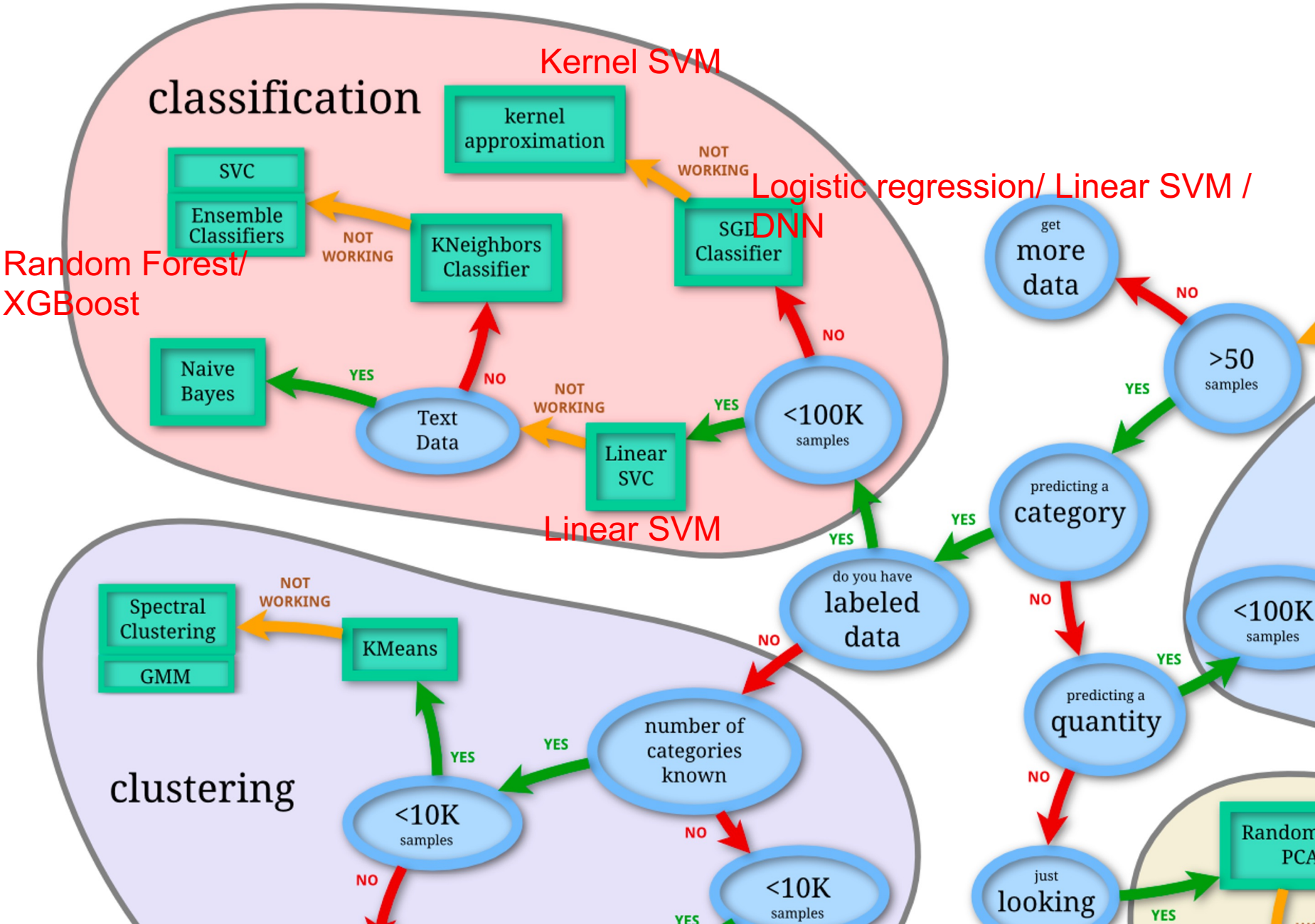
http://scikit-learn.org/stable/tutorial/machine_learning_map/index.html

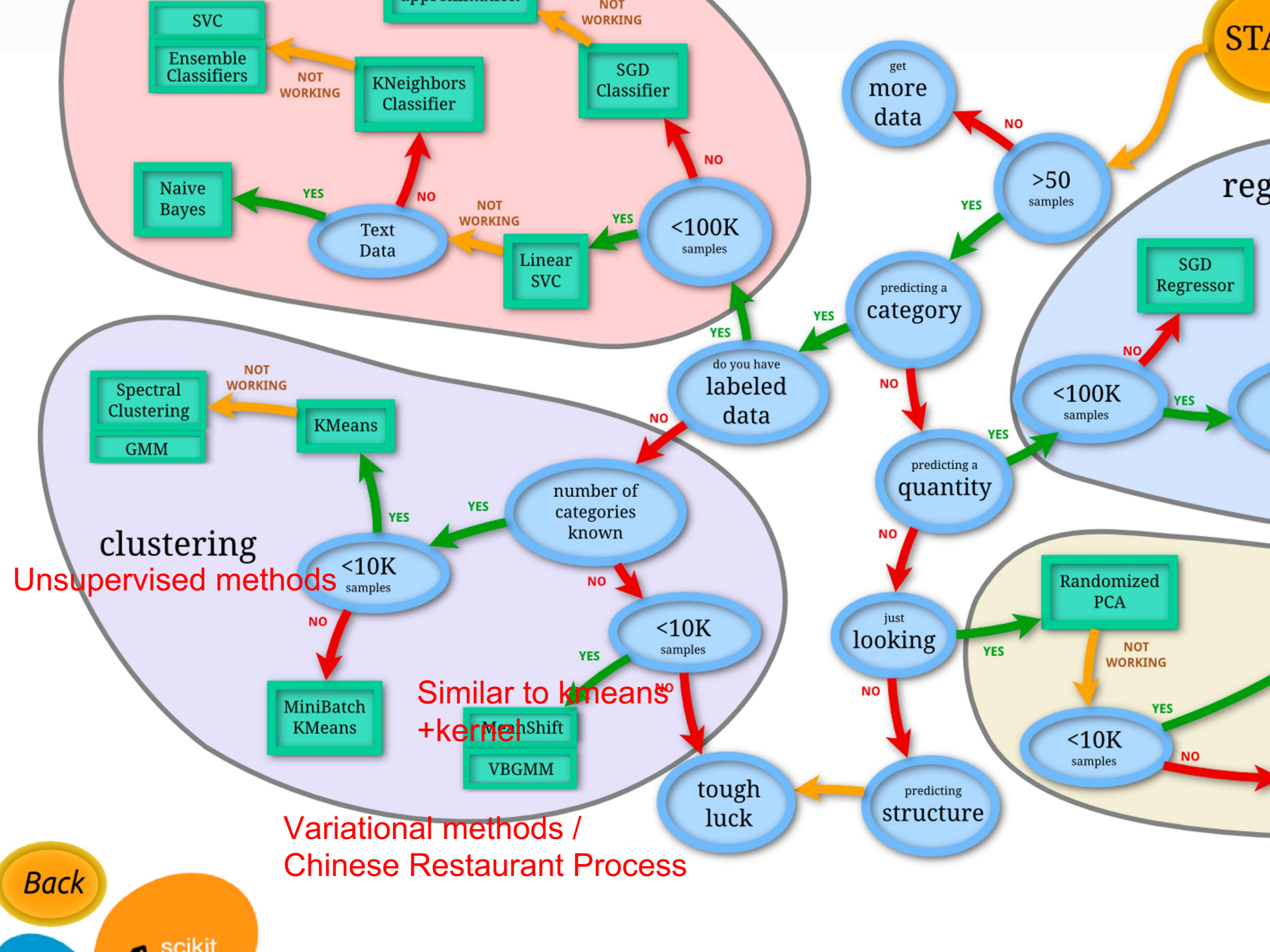


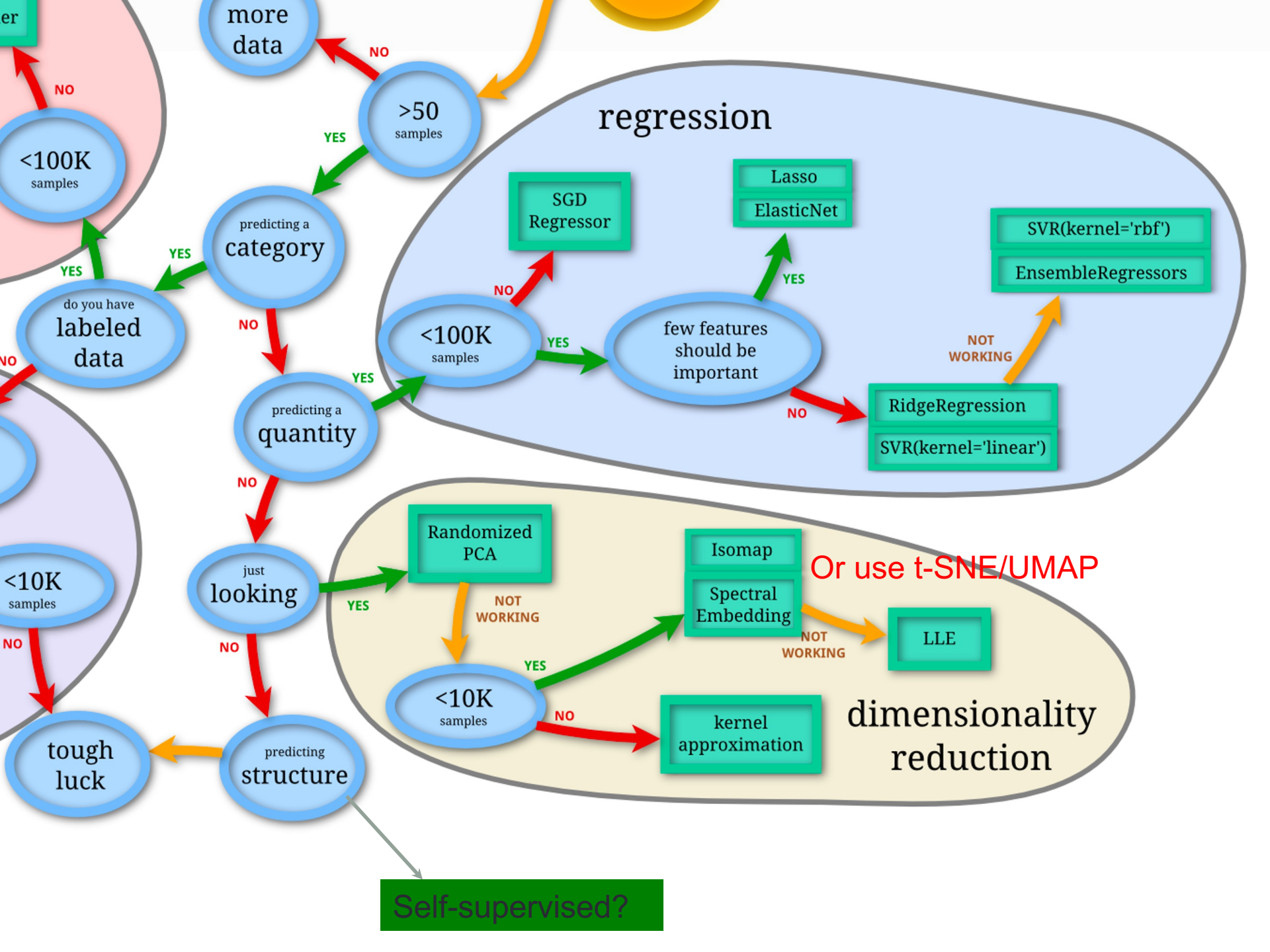
Back

Note: treat 100k, 10k samples as a guideline.
These numbers can go bigger or smaller depending on feature dimension and number of classes.


scikit-learn
algorithm cheat-sheet
START
get more data
NO
>50 samples
YES
predicting a category
NO
predicting a quantity
YES
NO
<100K samples
YES
NO
SGD Regressor
few features should be important
YES
NO
Lasso
ElasticNet
SVM
SVR(kernel='rbf')
EnsembleRegressors
NOT WORKING
RidgeRegression
SVR(kernel='linear')
SVM
regression
Regression /
DNN
MSE regression with L1
Or L1+L2 regularization
MSE regression with L2 regularization


classification
Kernel SVM
kernel approximation
NOT WORKING
SVC
Ensemble Classifiers
NOT WORKING
KNeighbors Classifier
SGD Classifier
Logistic regression/ Linear SVM / DNN
Random Forest/ XGBoost
NO
Naive Bayes
YES
Text Data
NO
NOT WORKING
Linear SVC
YES
<100K samples
NO
Linear SVM
get more data
NO
>50 samples
YES
predicting a category
YES
YES
do you have labeled data
NO
<100K samples
NOT WORKING
Spectral Clustering
GMM
KMeans
NO
YES
number of categories known
YES
predicting a quantity
YES
clustering
<10K samples
NO
NO
<10K samples
NO
just looking
YES
Random
PCA
YES


SVC
Ensemble Classifiers
NOT WORKING
KNeighbors Classifier
NOT WORKING
SGD Classifier
NO
Naive Bayes
YES
Text Data
NO
NOT WORKING
Linear SVC
YES
<100K samples
NO
get more data
NO
>50 samples
YES
predicting a category
YES
do you have labeled data
YES
NO
Spectral Clustering
GMM
NOT WORKING
KMeans
YES
number of categories known
YES
NO
<10K samples
NO
MiniBatch KMeans
<10K samples
YES
NO
MeanShift
VBGMM
Similar to kmeans +kernel
clustering
Unsupervised methods
tough luck
predicting structure
NO
just looking
NO
predicting a quantity
YES
NO
<100K samples
YES
NO
SGD Regressor
Randomized PCA
YES
NOT WORKING
<10K samples
YES
NO
Variational methods / Chinese Restaurant Process
Back

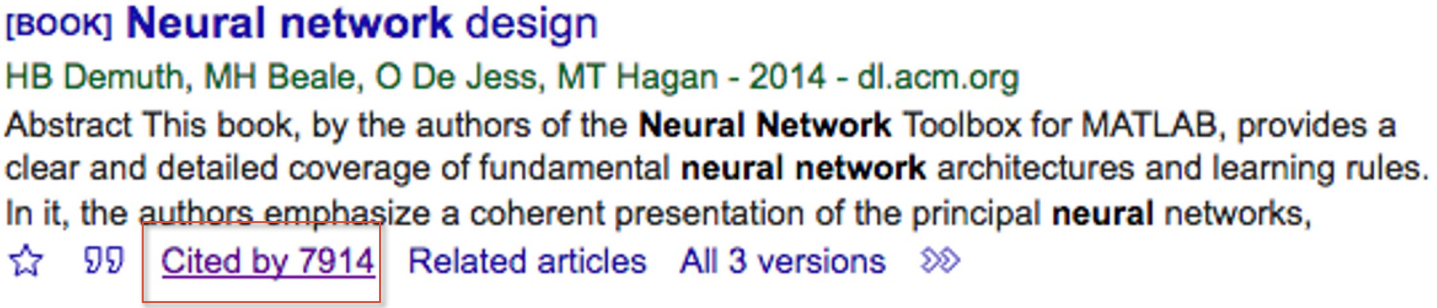

more data
NO
>50 samples
YES
<100K samples
NO
predicting a category
YES
do you have labeled data
YES
NO
predicting a quantity
YES
NO
just looking
YES
NO
<10K samples
NO
tough luck
predicting structure
regression
SGD Regressor
Lasso
ElasticNet
SVR(kernel='rbf')
EnsembleRegressors
NO
<100K samples
YES
few features should be important
YES
NOT WORKING
NO
RidgeRegression
SVR(kernel='linear')
Randomized PCA
Isomap
Spectral Embedding
Or use t-SNE/UMAP
YES
NOT WORKING
NOT WORKING
LLE
YES
<10K samples
NO
kernel approximation
dimensionality reduction
Self-supervised?

Do your literature review

- Check for the closest task
  - Does not need to be in the same domain
    - Ex: Strings of DNA -> NLP!
- Reading checklist
  - Amount of data
    - Number of classes
    - Data/classes
  - Features
  - Models
- Look at multiple papers

Literature review tricks

- Search forward and backward in https://scholar.google.com/
  - Citations
  - Cited by



Reading between the lines

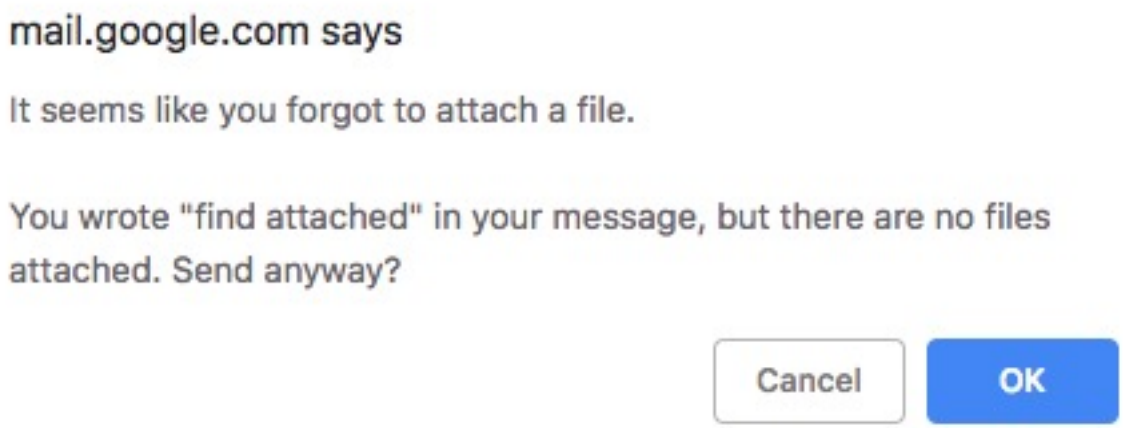
- If you don't understand something, re-read
  - Average of 5+ times to understand a paper completely
- Print it out, keep a pen/pencil at hand, and break down equations
- Try to explain it with what you already know

Elastic net Loss function $\min_w \frac{1}{2n_{samples}}||Xw - y||_2^2 + \alpha\rho||w||_1 + \frac{\alpha(1-\rho)}{2}||w||_2^2$

MSE Loss L1 L2

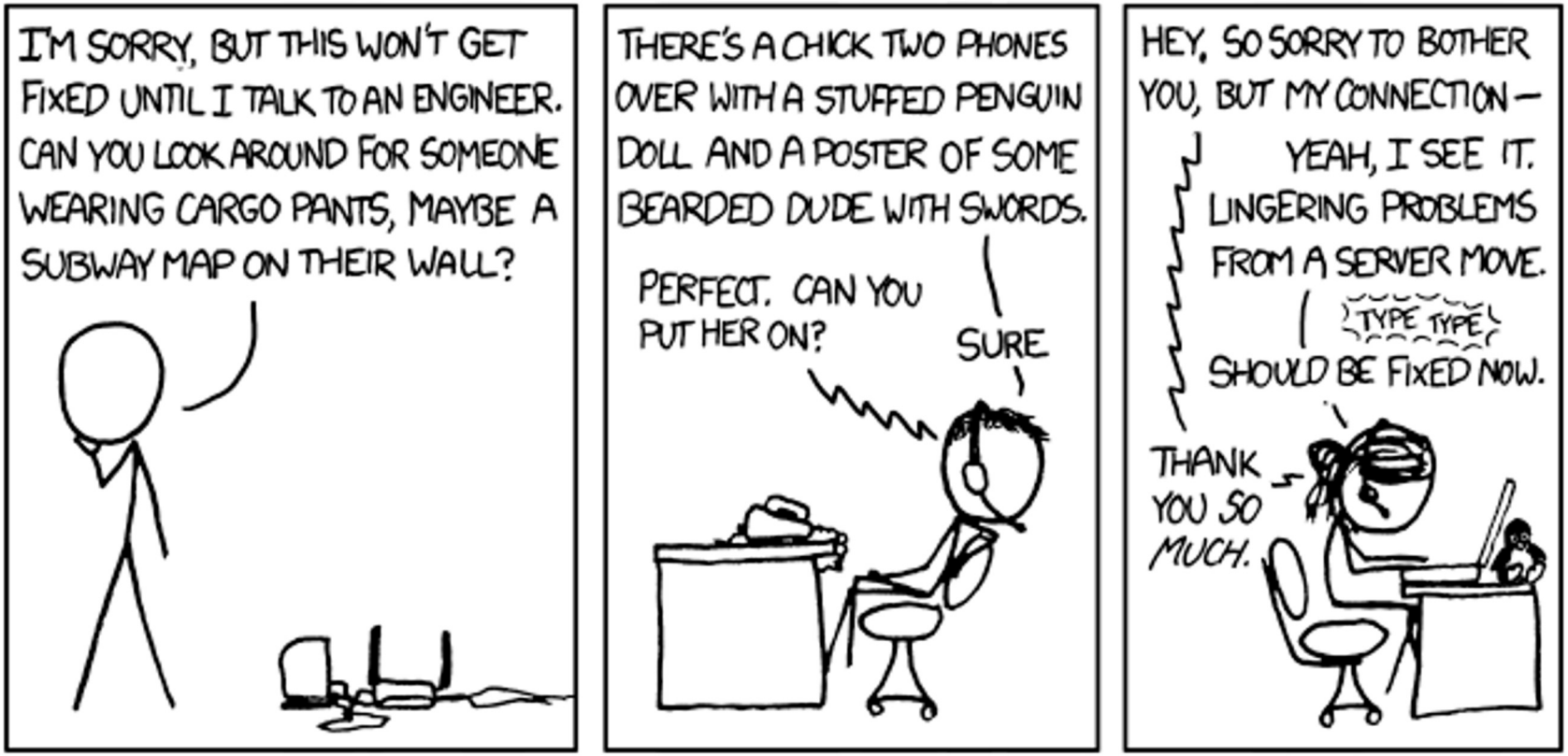
Rule #0: maybe no ML?

- Sometimes you don’t need a machine learning model
  - Launch a simpler system to collect data
    - If possible, have the user label your data.

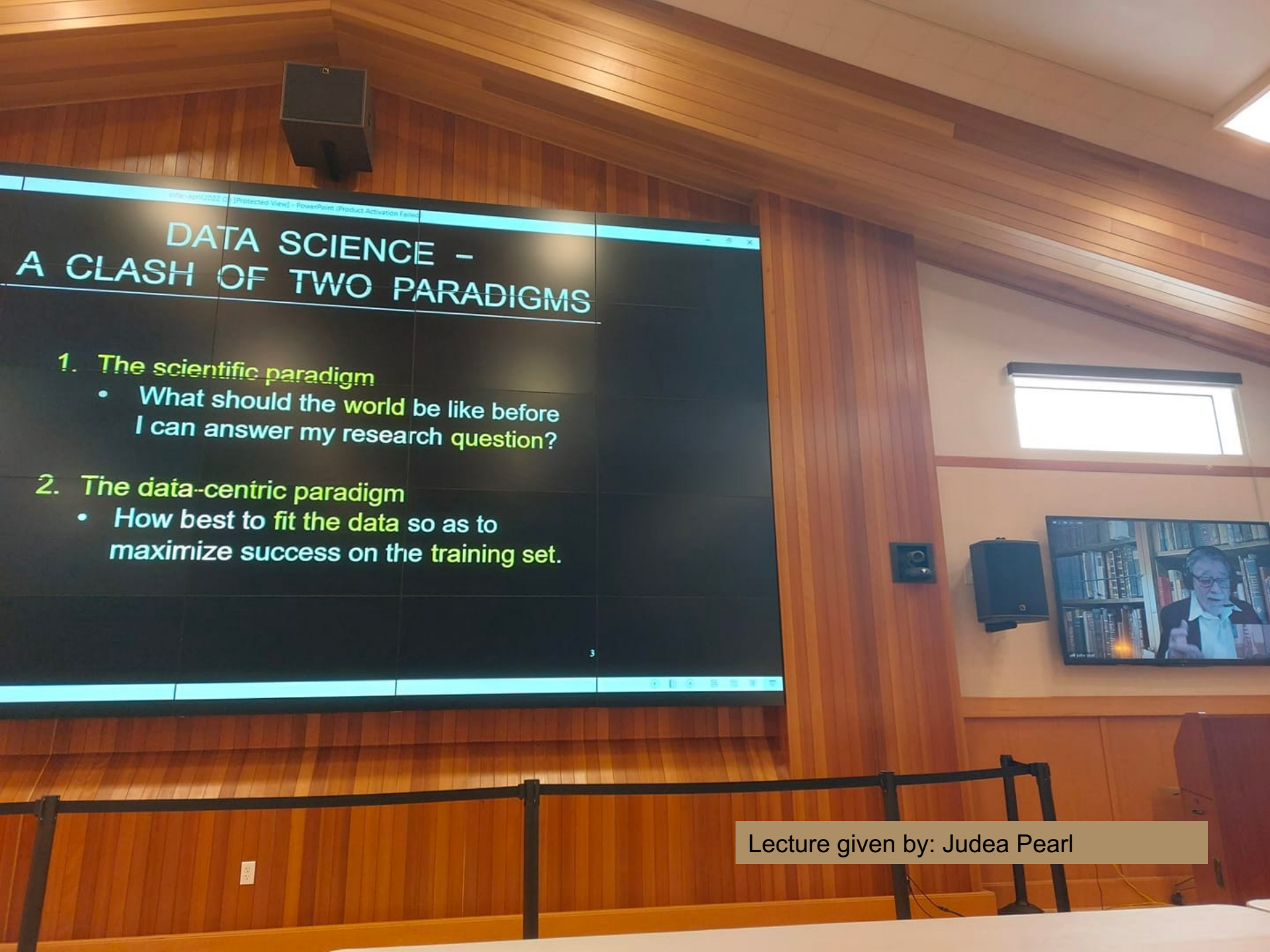


How to improve my results?

- More tech support



https://xkcd.com/806/



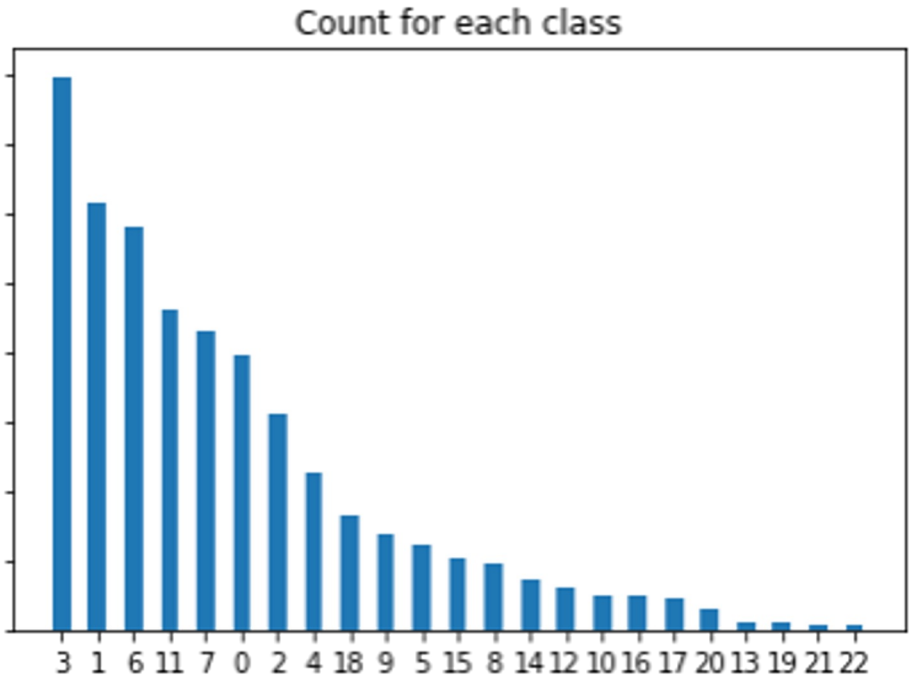
Lecture given by: Judea Pearl

Diagnosis and prescription

- Is my task proper?
- Should I add/remove feature XXX?
- Is my loss function appropriate?
- Is my model overfitting/underfitting?
- Should I try XXX?

Is my task proper? Do you suffer from class imbalance?

- Throwing away
- Refactoring
  - Split
  - Merge
- Data augmentation
- Biasing
  - Weighting the loss function
  - Bias in mini-batch sampling



Diagnosis

- Is my task proper?
- Should I add/remove feature XXX?
- Is my loss function appropriate?
- Is my model overfitting/underfitting?
- Should I try XXX?

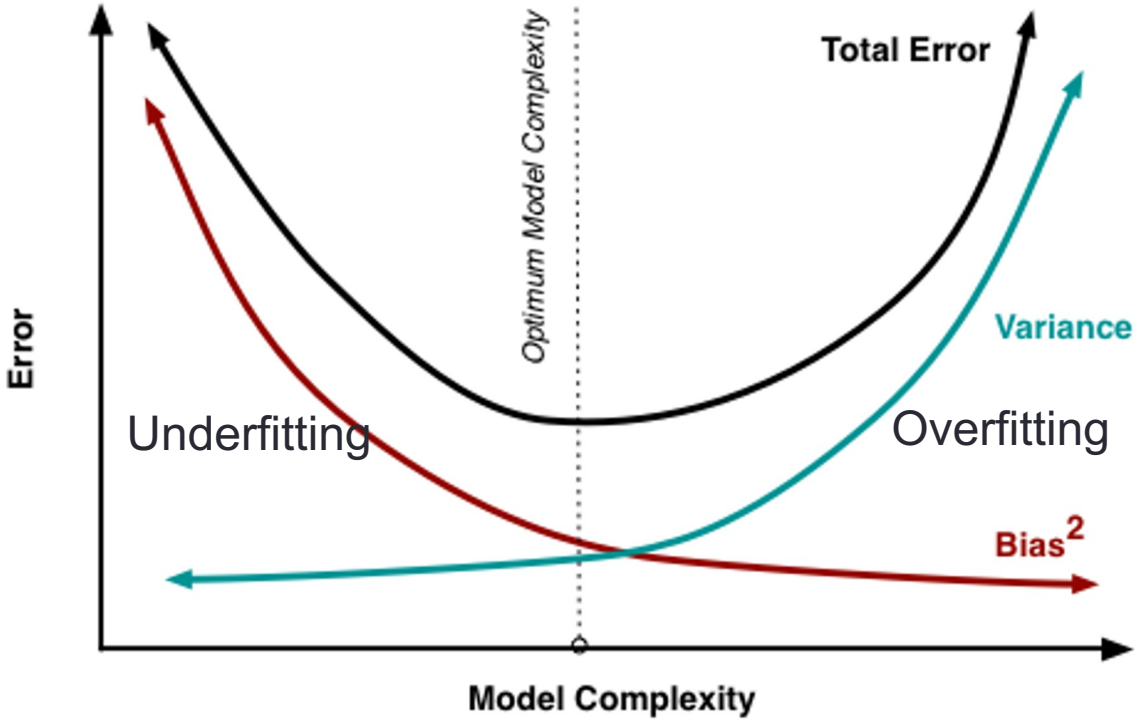
Short answer: Cross Validation

Can we do better?
Find the problem and fix it.

Understanding the bias variance trade-off

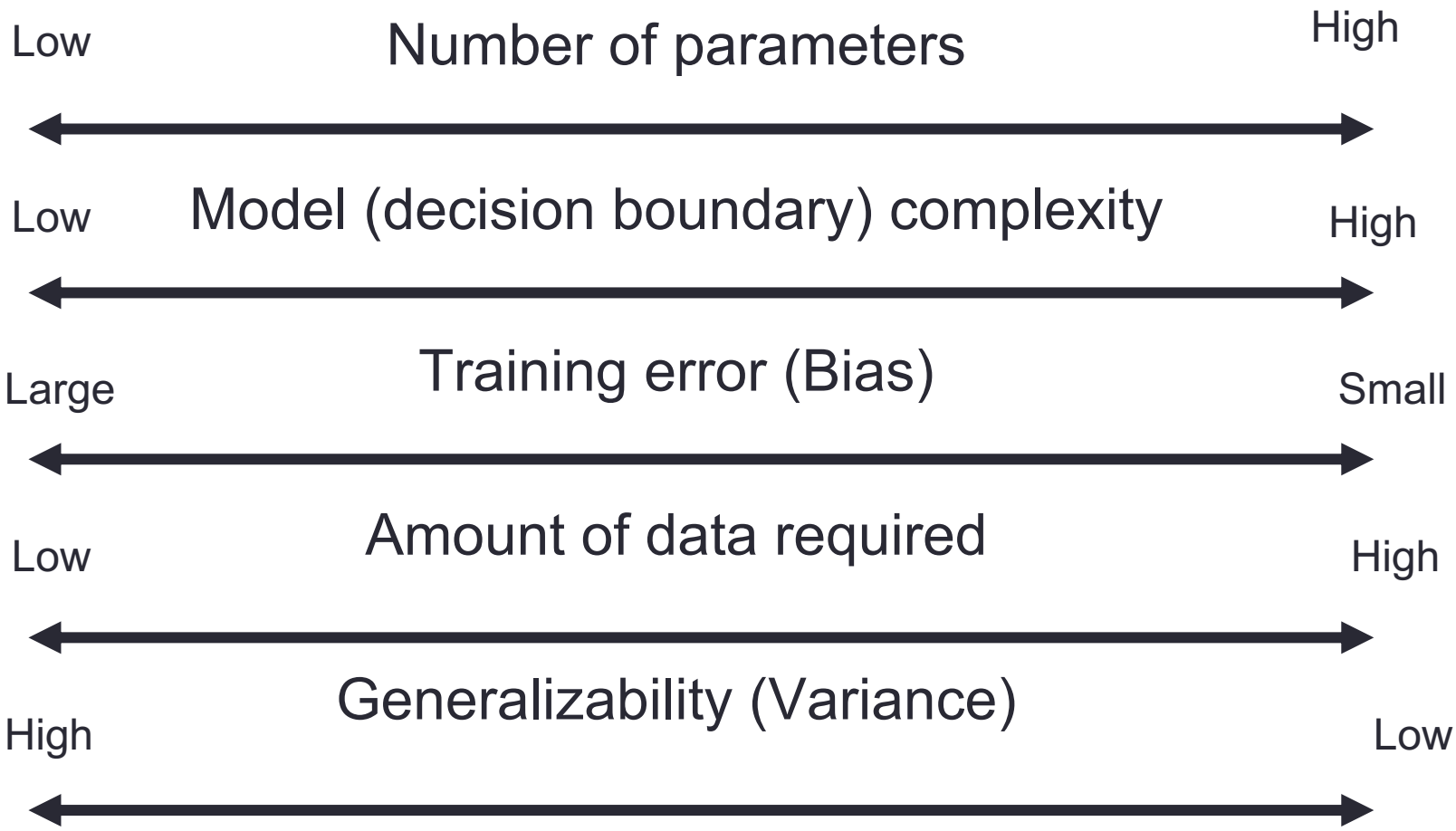
- Bias variance analysis can be helpful for diagnosis

$$\underbrace{E_{\mathbf{x},y,D}\left[\left(h_D(\mathbf{x})-y\right)^2\right]}_{\text{Expected Test Error}} = \underbrace{E_{\mathbf{x},D}\left[\left(h_D(\mathbf{x})-\bar{h}(\mathbf{x})\right)^2\right]}_{\text{Variance}} + \underbrace{E_{\mathbf{x},y}\left[\left(\bar{y}(\mathbf{x})-y\right)^2\right]}_{\text{Noise}} + \underbrace{E_{\mathbf{x}}\left[\left(\bar{h}(\mathbf{x})-\bar{y}(\mathbf{x})\right)^2\right]}_{\text{Bias}^2}$$



http://www.cs.cornell.edu/courses/cs4780/2017sp/lectures/lecturenote11.html

Bias-Variance overview



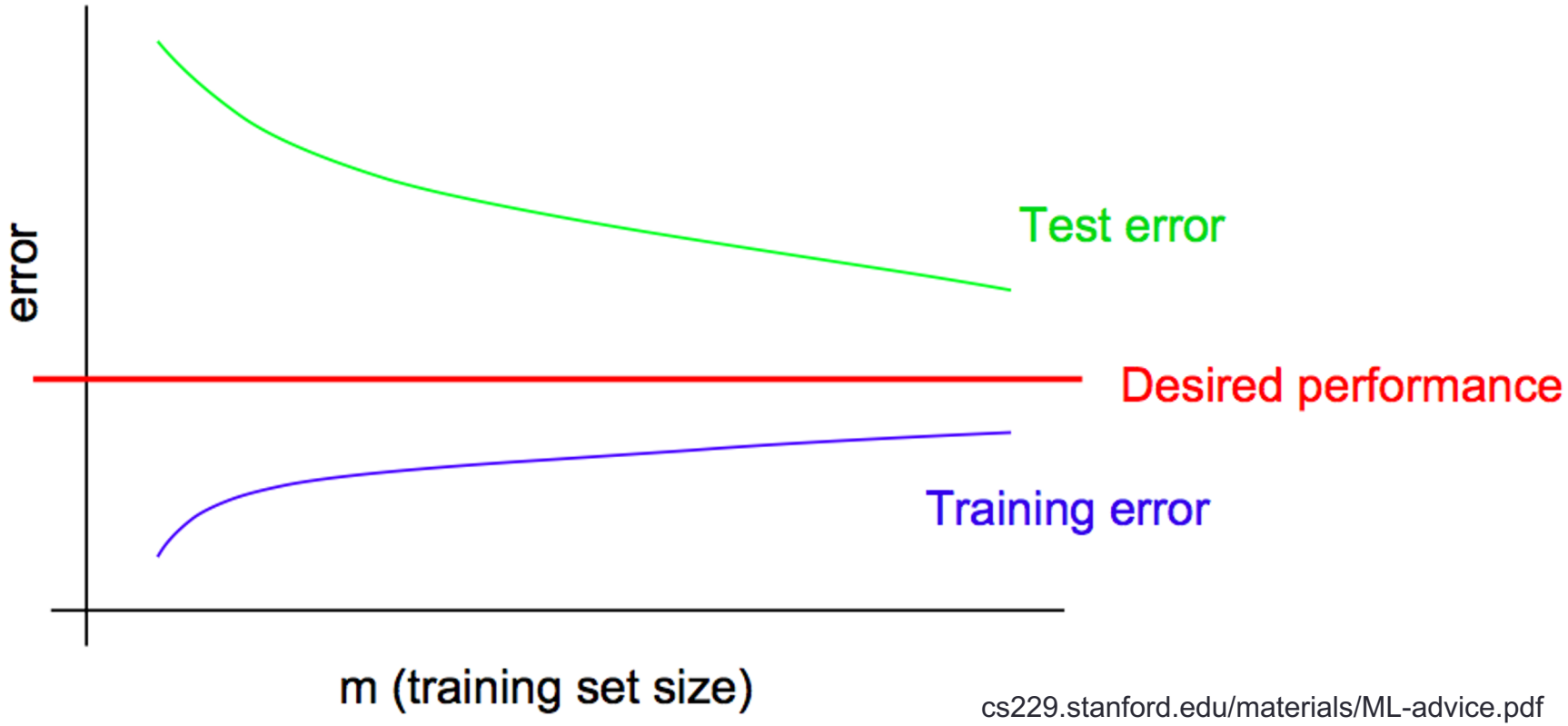
Bias-variance diagnosis

- Suppose the problem is either
  - Overfitting (high variance)
  - Too few features to classify properly (high bias)

- Symptoms
  - Variance: Training error is much lower than test/dev error.
  - Bias: Training error is also high

High variance case

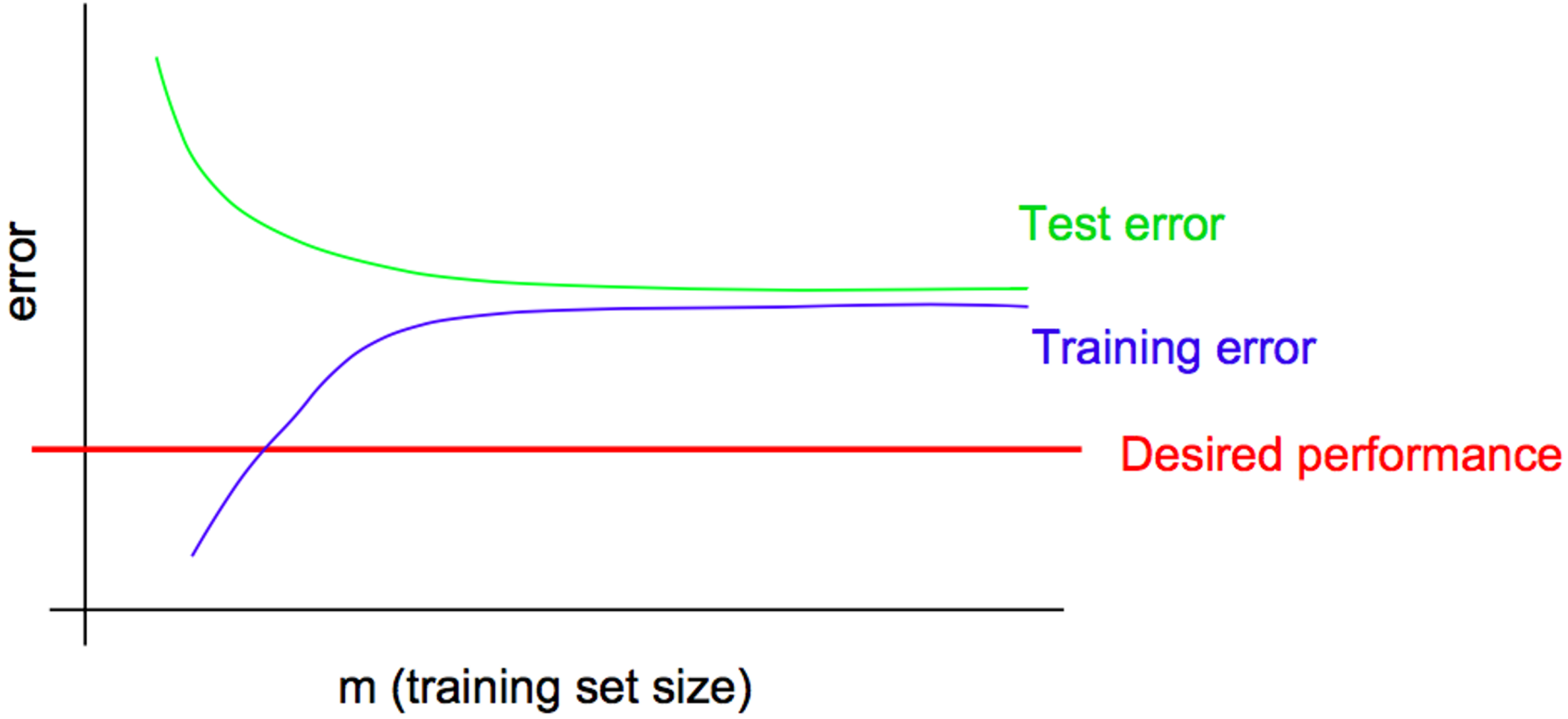
- Solution:
  - Reduce overfitting: regularization, reduce features, etc
  - Get more training set



cs229.stanford.edu/materials/ML-advice.pdf

High bias case

- Solution: more features, more layers, bigger models



cs229.stanford.edu/materials/ML-advice.pdf

Desired performance?

- The goal of the application
  - Possibilities
    - Target performance to launch the product
    - Human performance on the task
      - Have A and B do the same task, measure the difference.
- Knowing human performance gives several advantage
  - Knows when to stop
    - Beating human is sometimes a goal, but
    - Some errors might be labeling errors or judgment calls

Error diagnosis

- You're making a cat classifier. (cat/not cat)
- It sucks.
- You heard of this super new hype algorithm (for example: VIT with LoRA). Should you spend months to try it out?

This is not a cat

https://www.vat19.com/item/not-a-cat-cat-the-cat-that-isnt

Looking at the errors

- Spend an hour or two looking at your errors. Identify why. Keep a table.

	Blurred	Weird angle	Notes
Pic1	x		Stuffed toy
Pic2	x		
Pic3	x		
Pic4		x	Top view
…	…	…	…
	68%	2%	

Solution: Use a method to sharpen the image. Train on blurry images.

The table categories can expand as you look through more pictures and see frequently occurring error cases. So keep notes.

Diagnosis summary

- Simple analysis of the data can help you notice underlying problems
- Bias-variance diagnosis is a general method that can be applied to most tasks
- Other diagnosis depends on the application and need some understanding of the algorithms
- Error analysis can help guide your model improvements

Taking the time to do diagnosis can help you save months of trying random things.

Debugging neural networks

- Neural networks will give you something reasonable even with a buggy code.
  - Identifying them can be quite hard
- Some tips by Andrej Karpathy
  https://karpathy.github.io/2019/04/25/recipe/

Questions beyond classification/regression

Can I trust my model?

Why does it give this answer?

ML products

Customer facing

Recommendation systems, maps (traffic estimate), speech2text

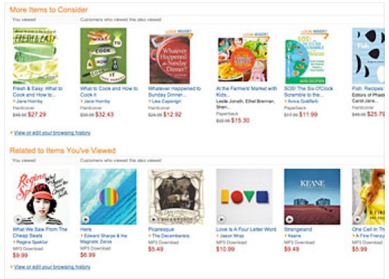
Best guess by the model
Mostly automatic (check deposit by app)

Internal facing

Loan applications, demand forecasting

Can say “I’m not sure”
Human in the loop. Machine-assisted

Requires confidence level

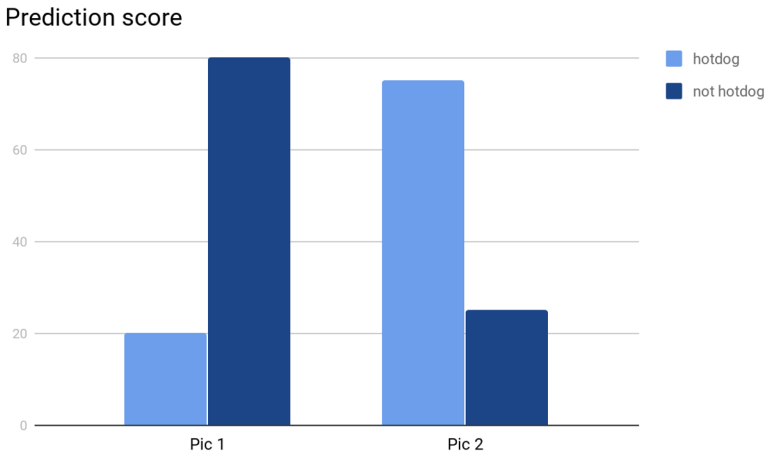


Slides courtesy of Phiradet Bangcharoensap

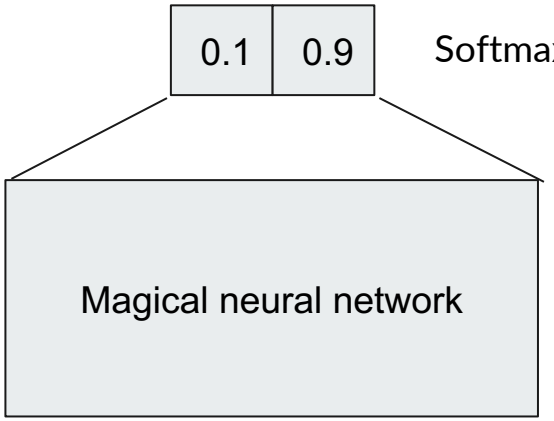
Confidence score

Practical models are not only accurate, but need to be able to state its confidence

Confidence = probability of being correct

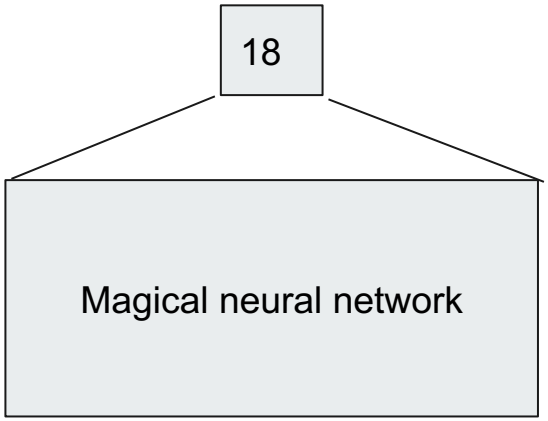


Naive way for confidence



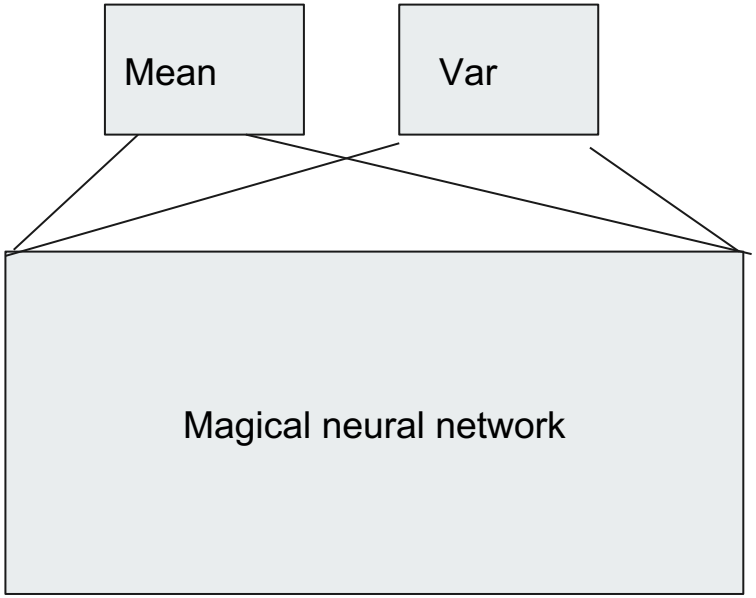
Confidence level = 0.9?

What about regression?



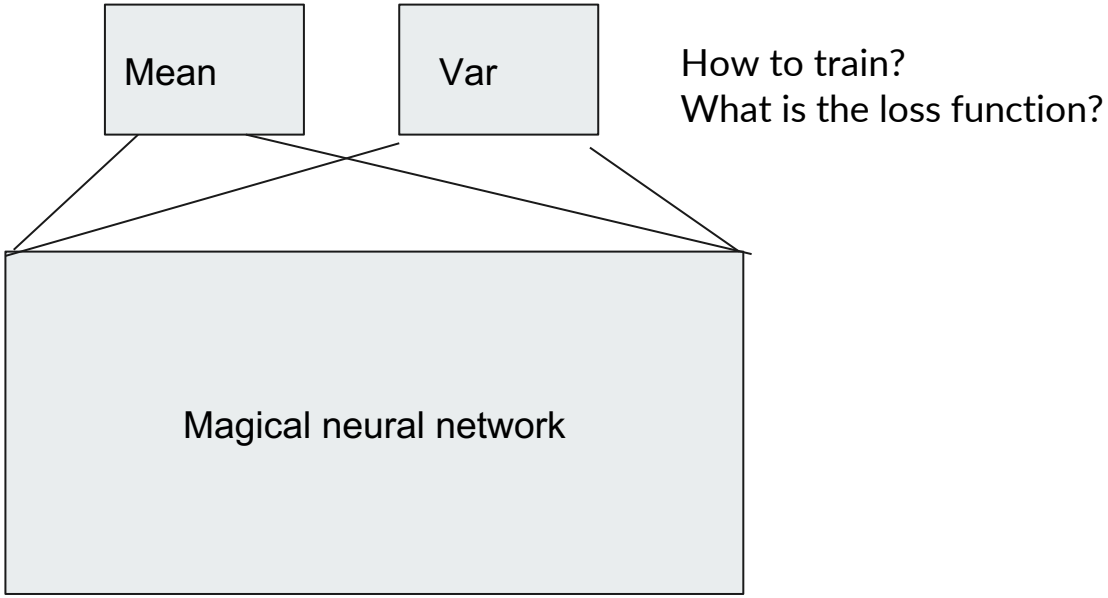
Confidence level = ???

A Naive way for regression (1994!)



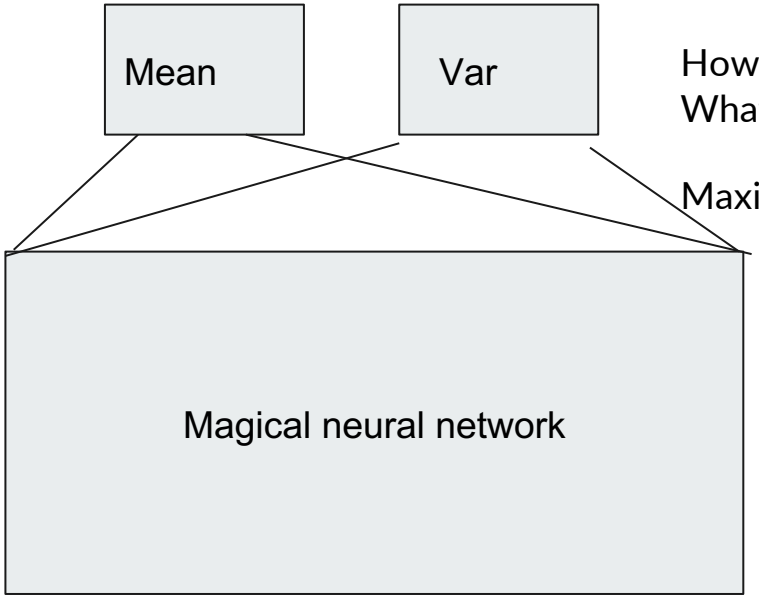
Estimating the mean and variance of the target probability distribution. IEEE 1994

A Naive way for regression (1994!)



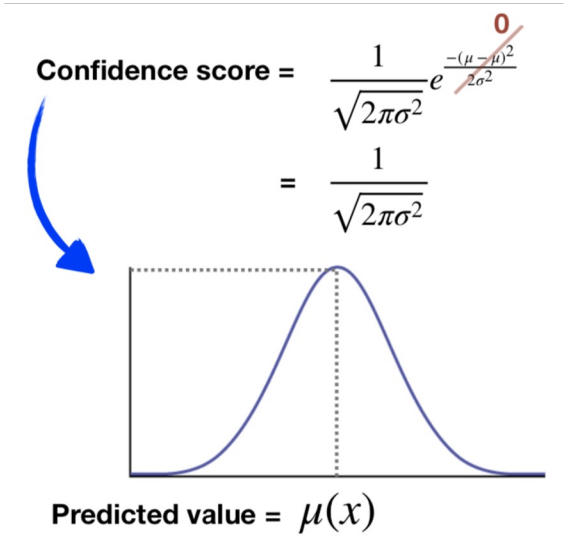
Estimating the mean and variance of the target probability distribution. IEEE 1994

A Naive way for regression (1994!)



How to train?
What is the loss function?

Maximum likelihood of the training data

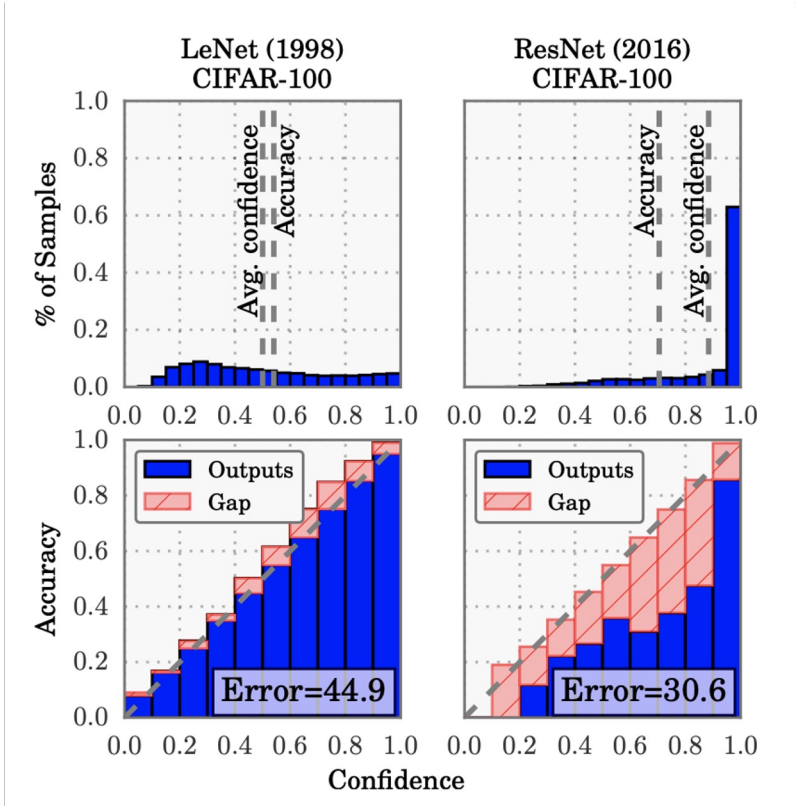


Estimating the mean and variance of the target probability distribution. IEEE 1994

Poorly calibrated confidence

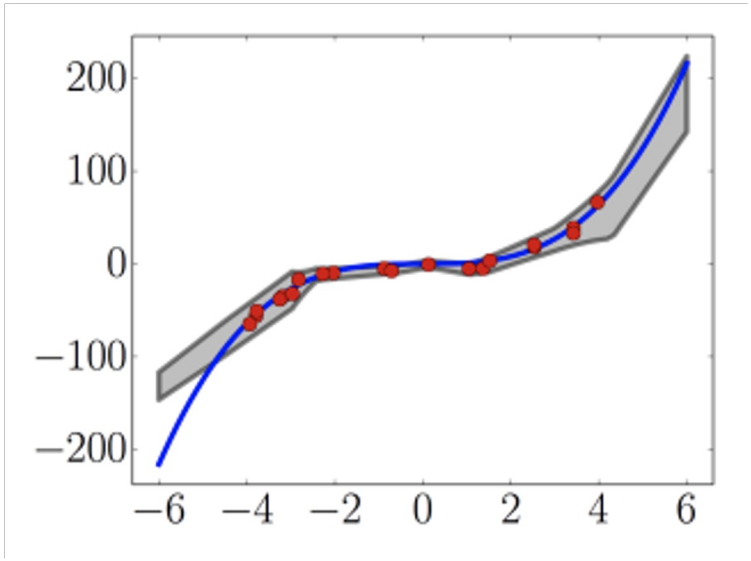
Deep Neural Networks are always overconfident!

Confidence = Probability of being correct

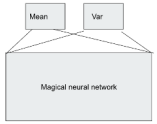


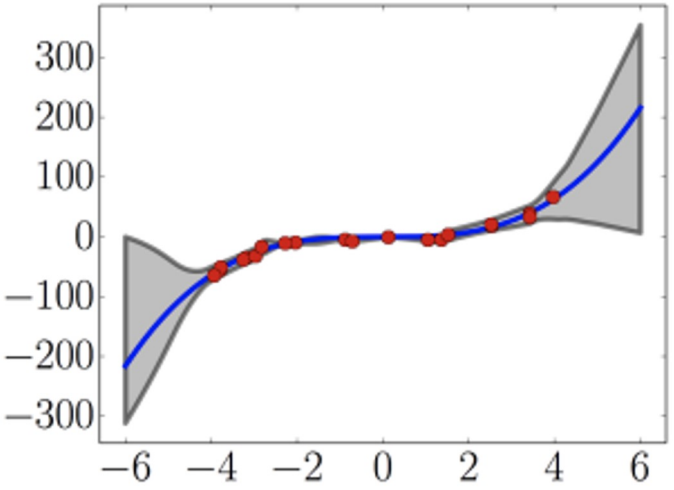
On Calibration of Modern Neural Networks JMLR 2017

Out of distribution problem



output from predicting variance via maximum likelihood





output from ensemble

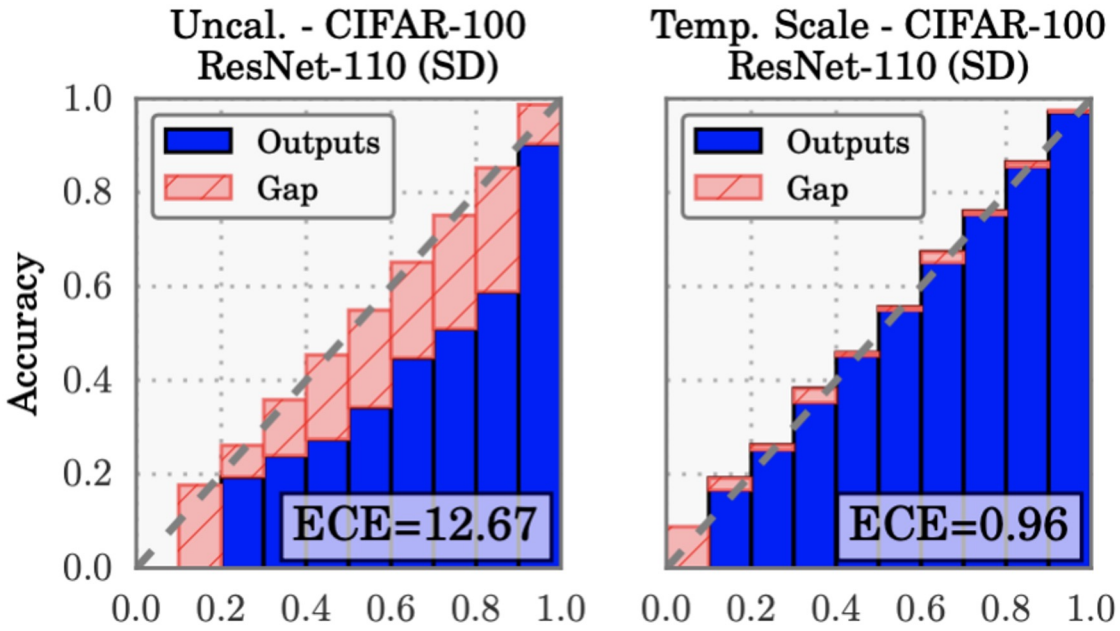
Simple and scalable predictive uncertainty estimation using deep ensembles. 2017

Neural network calibration

Make the confidence output follows the probability of being correct.

How?

Need a seperate training set to train the calibration (calibration set)



Overview of methods for calibration

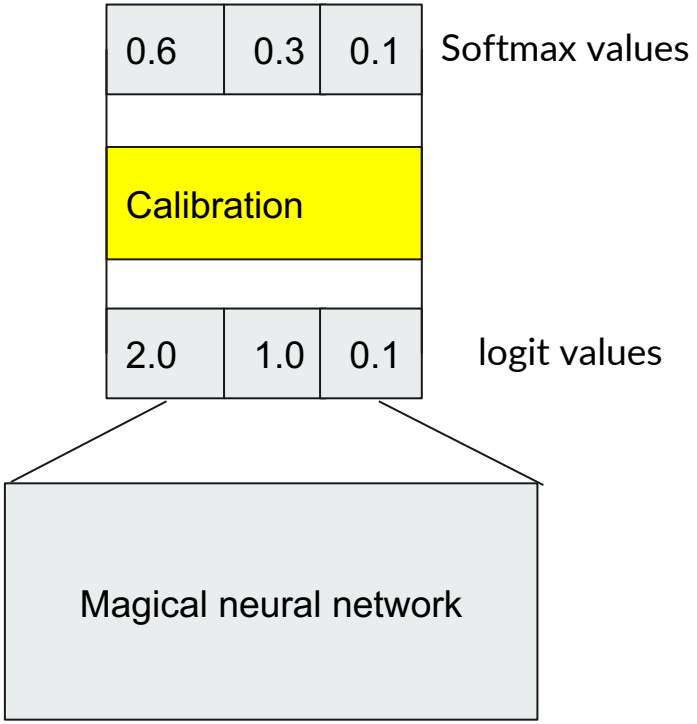
1 Calibration

2 Ensemble

- 2.1 Combining models
- 2.2 Bayesian neural networks

Calibration

Post processing after getting pre-softmax output (logit)



Calibration - Temperature scaling

Add a temperature T to rescale the softmax

$$\hat{q}_i = \max_k \sigma_{\mathrm{SM}}(\mathbf{z}_i / T)$$

T is tuned to maximize log likelihood on the calibration set

Low T

High T

https://arxiv.org/pdf/1611.01144.pdf

Other calibration methods

Histogram binning

Bayesian binning into quartiles (BBQ)

Matrix and vector scaling (model on top of model)

Isotonic regression (model on top of model)

See also https://scikit-learn.org/stable/modules/calibration.html

Try different methods on your dataset. No absolute best.

Combining models

Create multiple models

Calculate mean and variance of the answers!

Multiple models can be just from different initializations

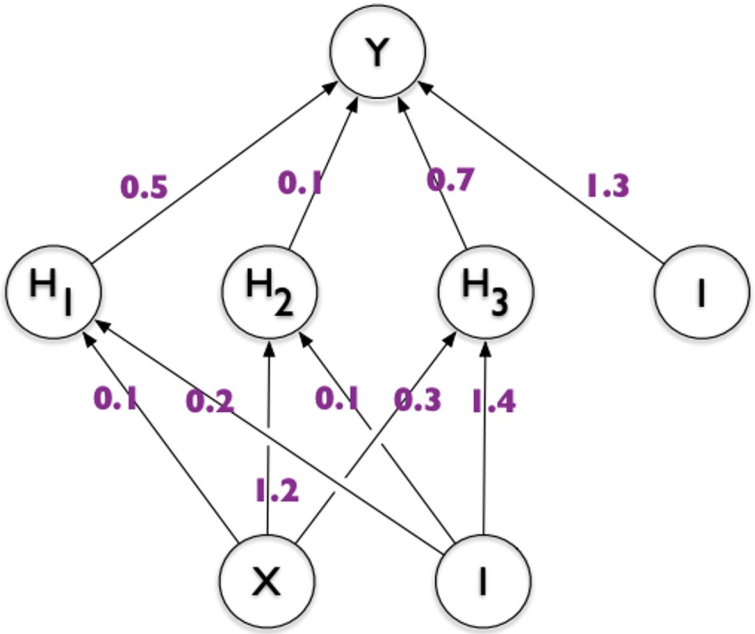
Cons: need to keep a lot of models around

How to get multiple models around?

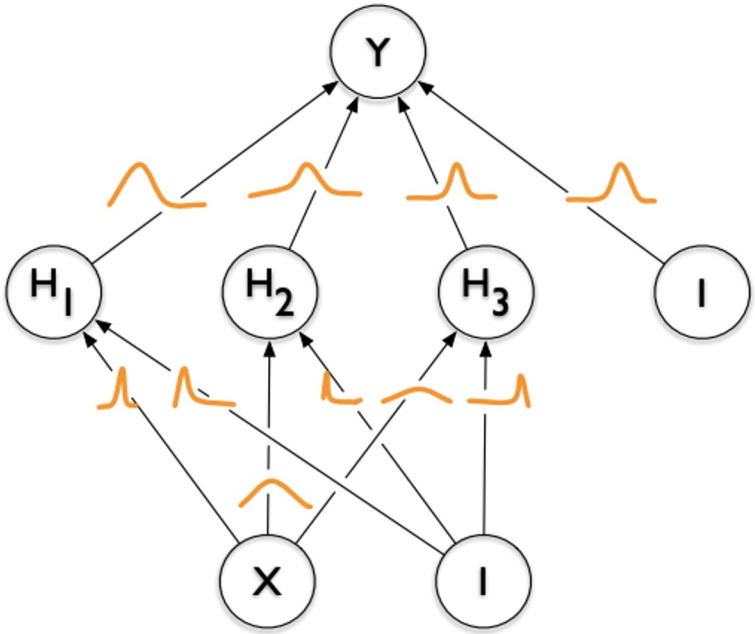
"Simple and scalable predictive uncertainty estimation using deep ensembles." 2017.

Bayesian Neural Network

Normal NN



Bayesian NN



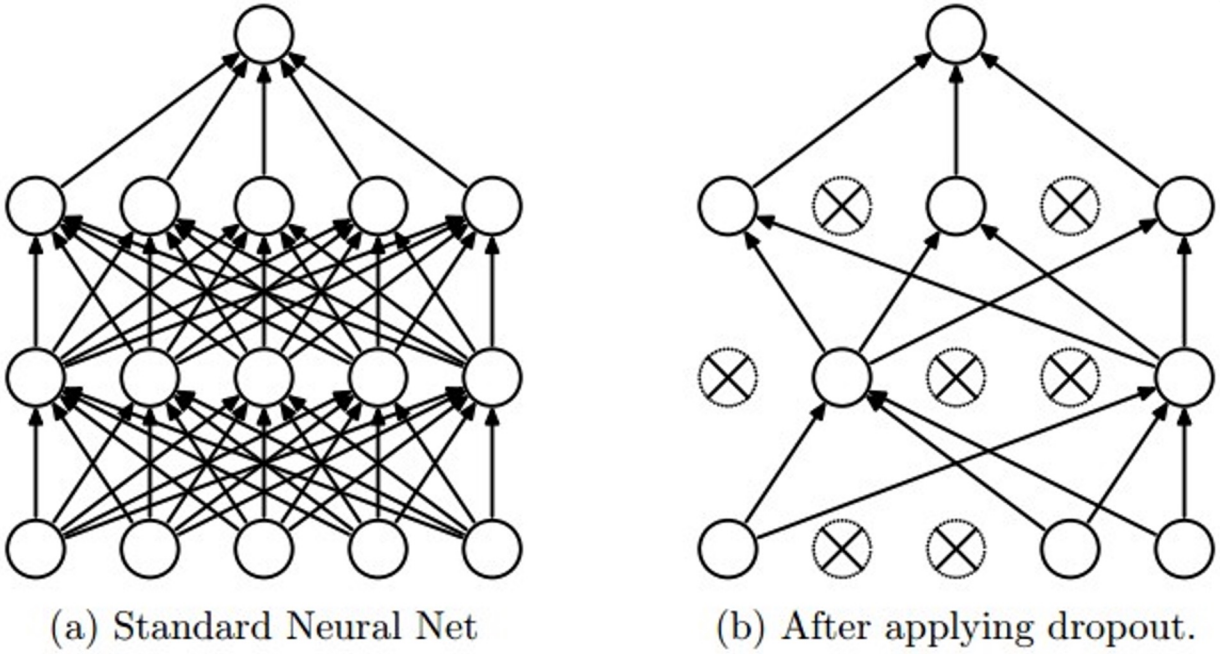
Problem: hard to do inference. Need to find $P(y \mid x, D) = \sum_{\theta} P(y \mid x, \theta) P(\theta \mid D)$

Weight Uncertainty in Neural Network. ICML 2015

Monte Carlo dropout

Use dropout at test time to simulate different models

Sampling of dropout at inference time is the same as sampling weights from Bernoulli distribution



(a) Standard Neural Net

(b) After applying dropout.

https://www.depends-on-the-definition.com/model-uncertainty-in-deep-learning-with-monte-carlo-dropout/

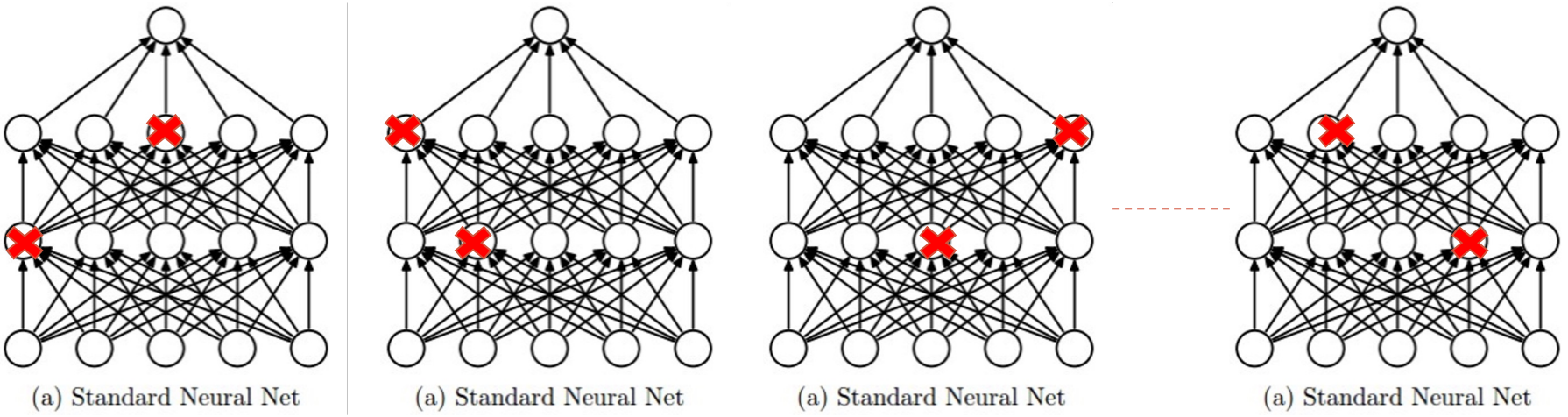
https://arxiv.org/pdf/1506.02142.pdf Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning

Monte Carlo dropout for confidence estimation

compute mean and variance for the answer and confidence

20.2 +- 3.2

25 27 18 15



So we got the confidence, can we say why?

Two levels of understanding

Model level

Describes the model tendency

Talks about the behavior on training data

Output level

Attributes model decision for a given test sample to different features

Model level: Simple example

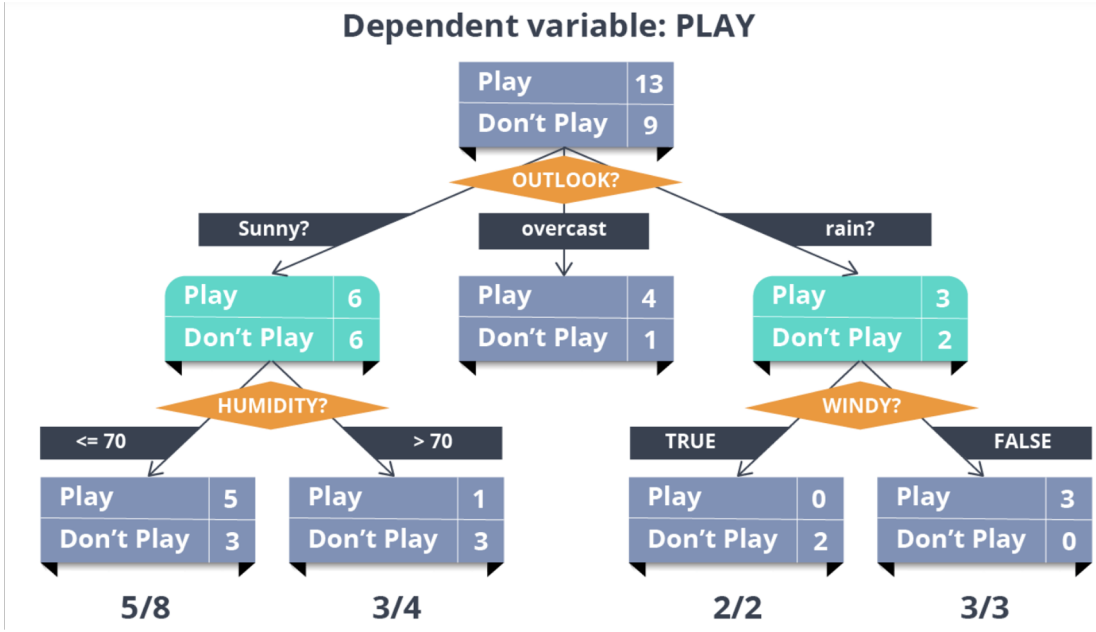
Logistic regression

$$g(z') = \phi_0 + \sum_{i=1}^{M} \phi_i z'_i,$$

You can say which feature is important from size of the coefficients.

Pros: Simple, easy to understand
Cons: Only model linear effects

Model level: tree-based (feature importance in XGBoost)



Assign scores based on how the features are use

A node that splits better (better purity at children): high score

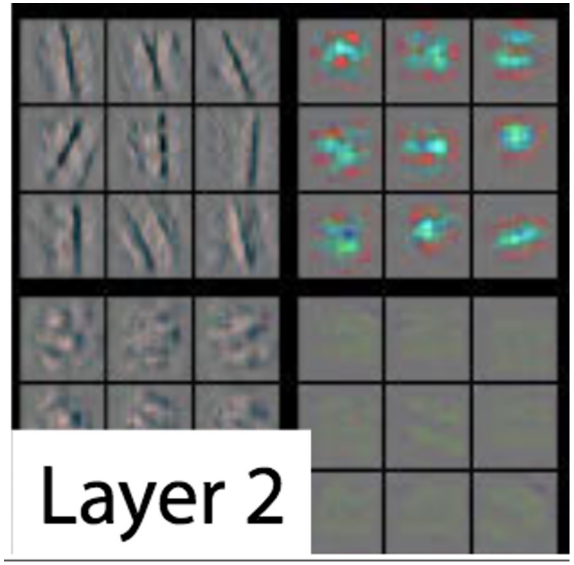
A node with more training samples: high weight

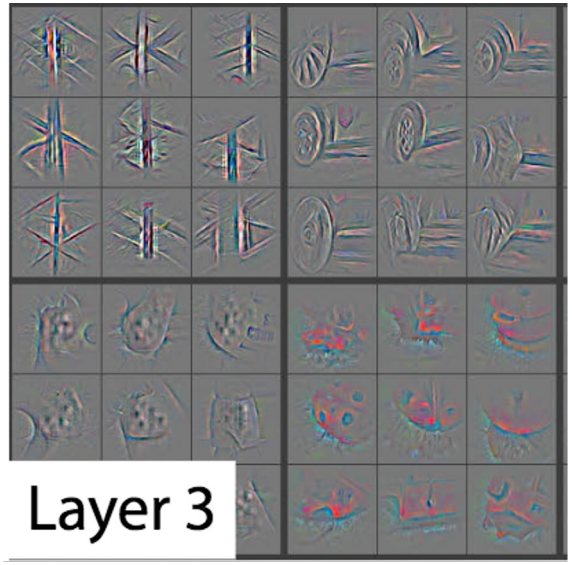
https://www.edureka.co/blog/decision-trees/

Model level: visualizing filters

Plot out the weights of the CNN filters...not so useful in practice

Layer 1





https://arxiv.org/pdf/1311.2901.pdf

Usefulness of model level

Feature selection

Gives you confidence that the model is learning reasonable things

Two levels of understanding

Model level

Describes the model tendency

Talks about the behavior on training data

Output level

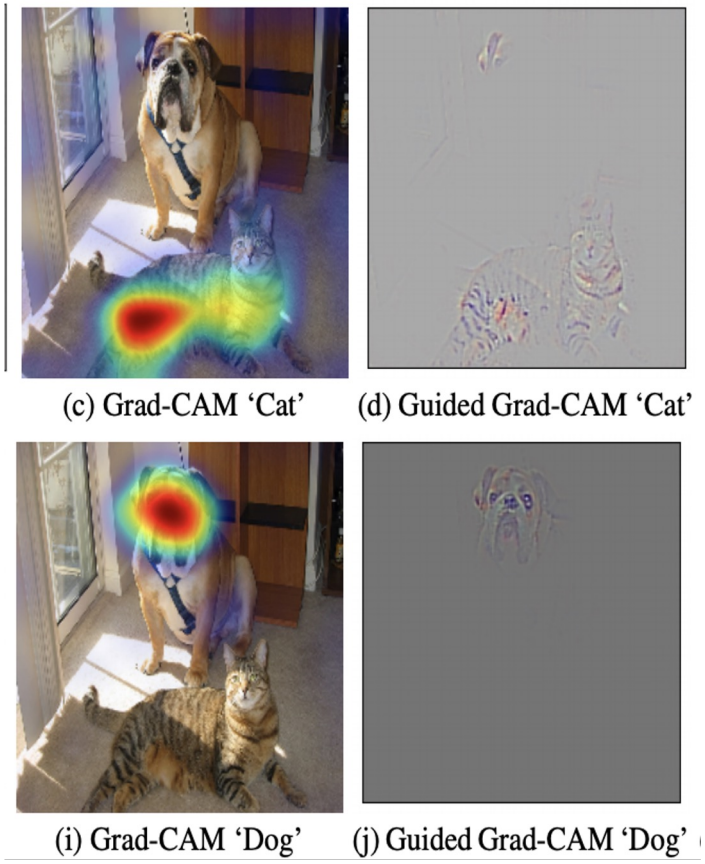
Attributes model decision for a given test sample to different features

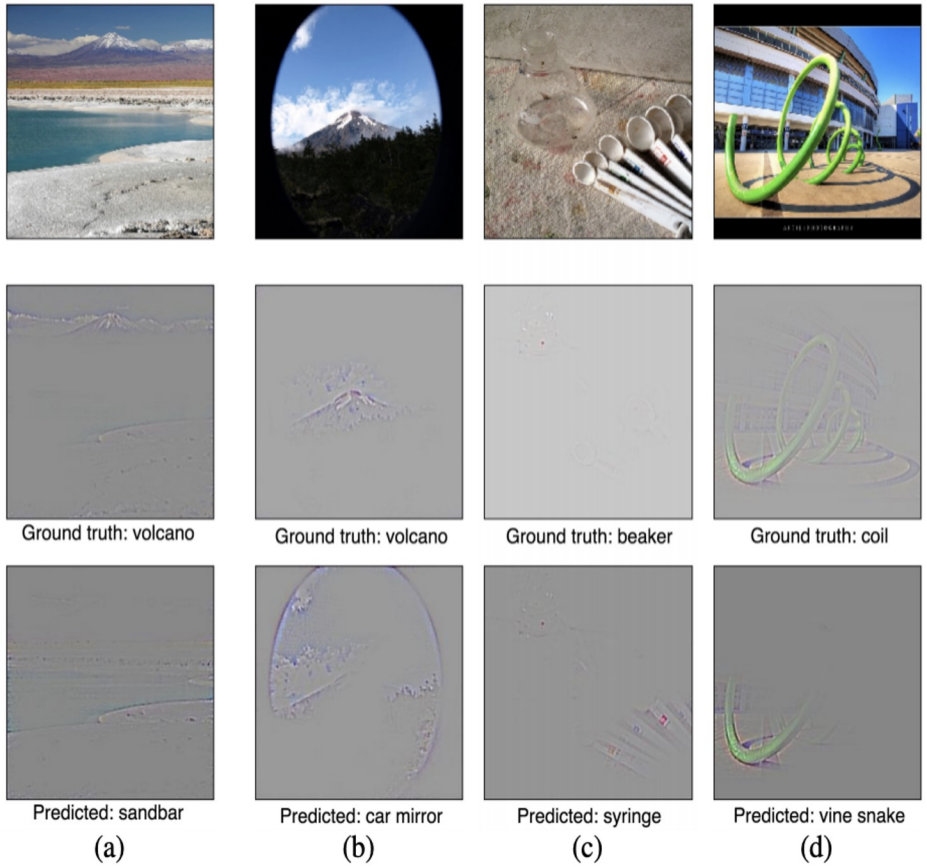
Output level: key ideas

A feature is important if I tweak the feature and the output change a lot.

If I tweak the output, how does the gradient flows

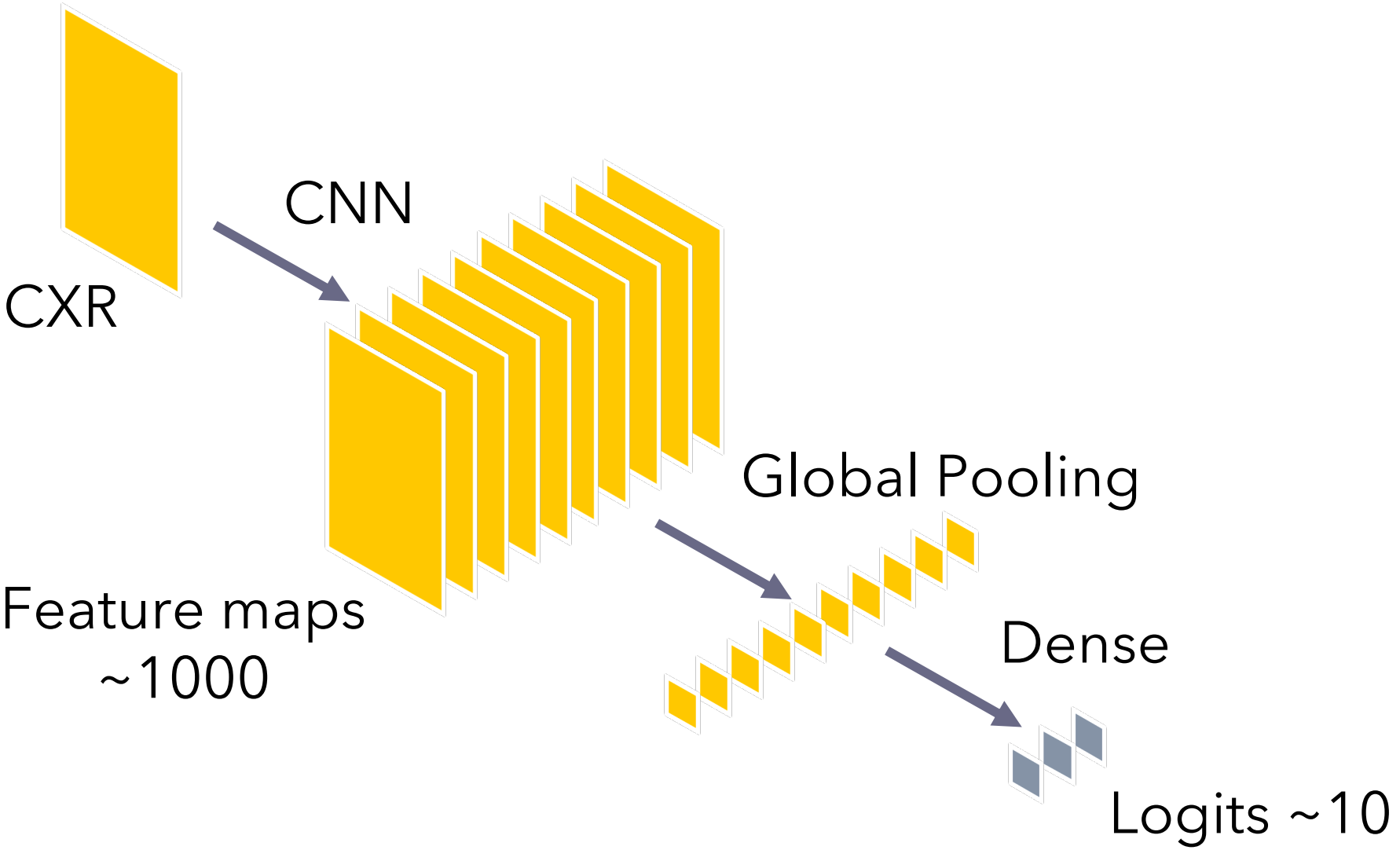
Output level: Gradient-weighted Class Activation Mapping (Grad-CAM)



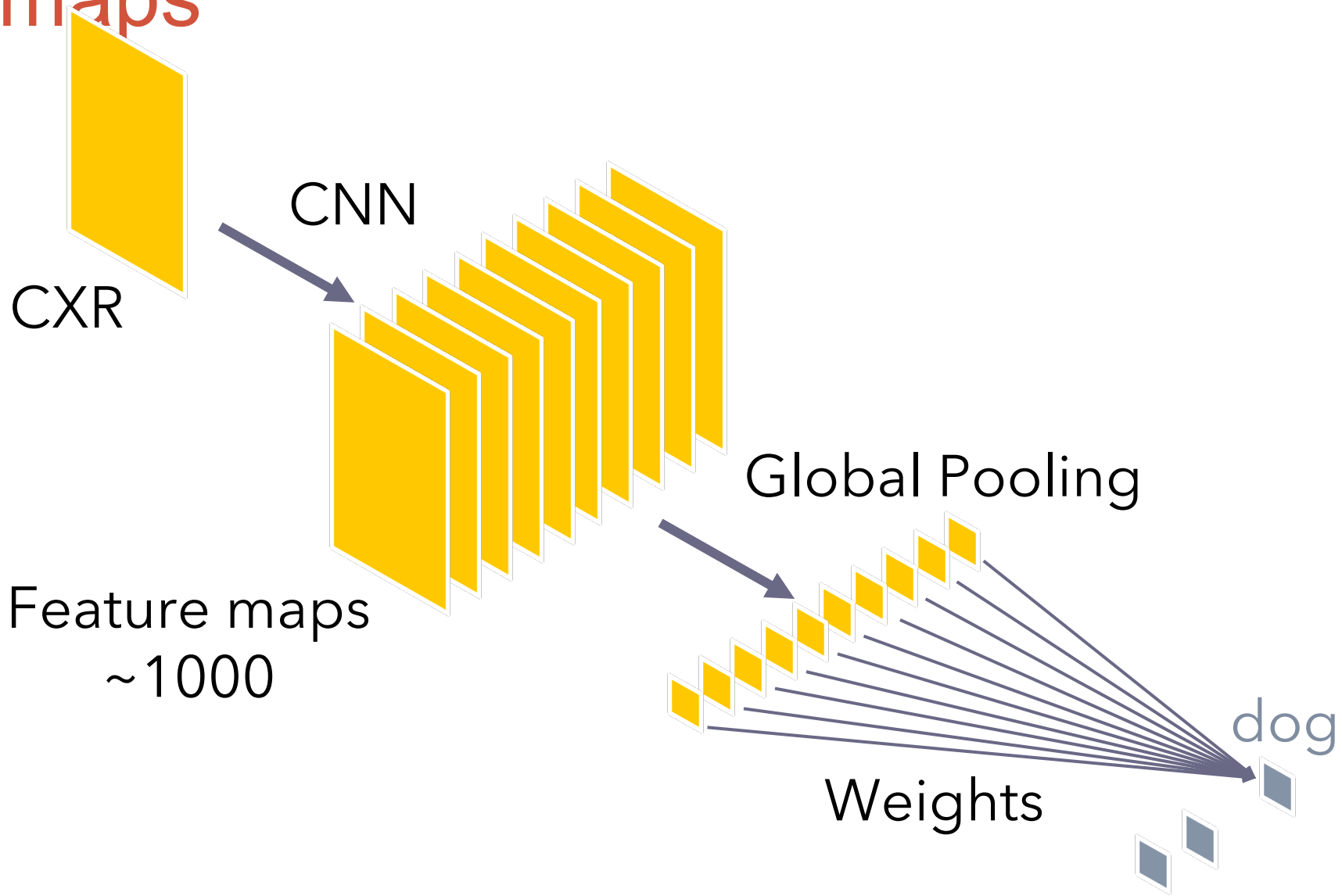


https://arxiv.org/pdf/1610.02391.pdf
https://github.com/jacobgil/keras-grad-cam

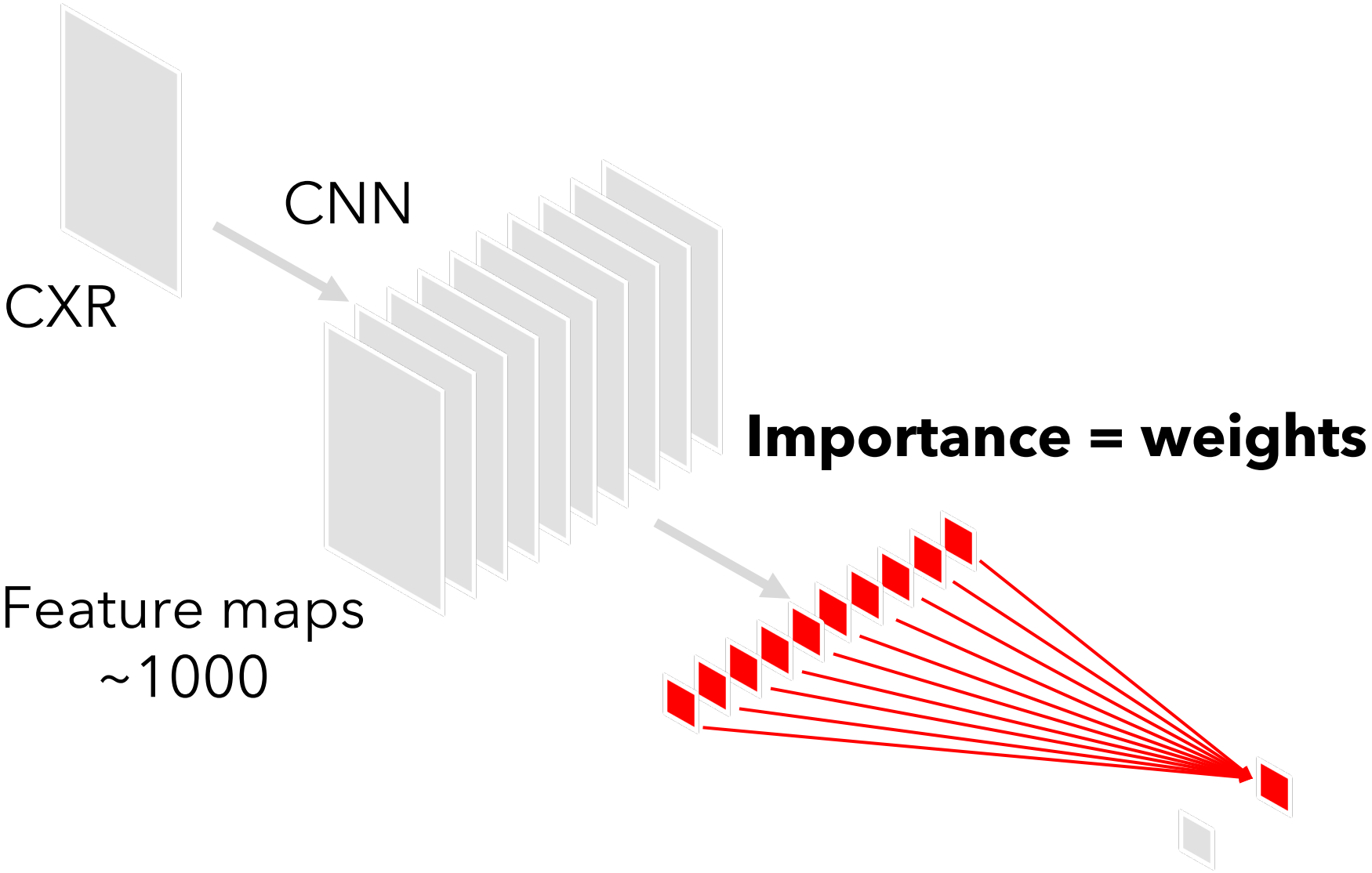
Typical CNN



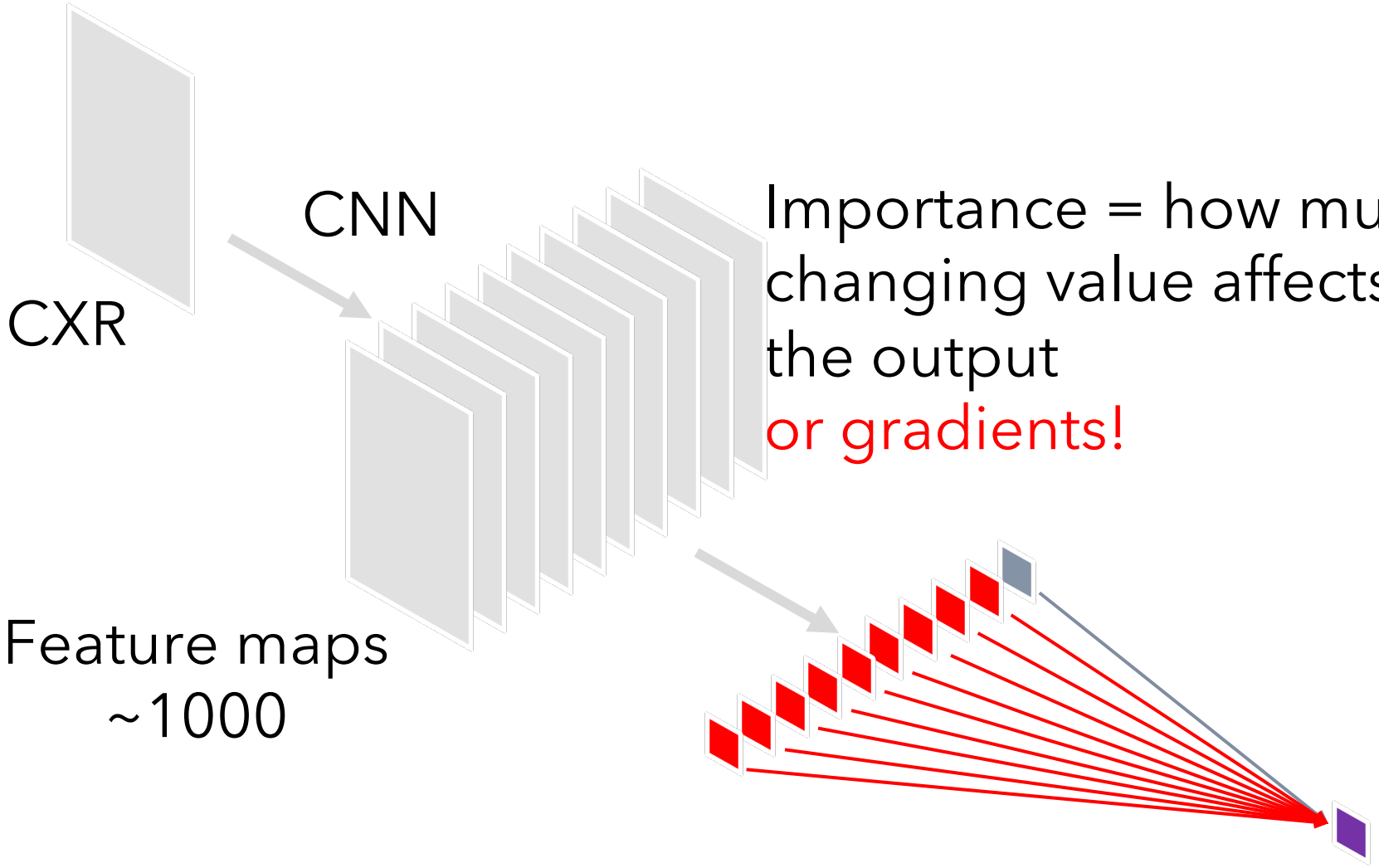
Prediction = weight sum of feat. maps



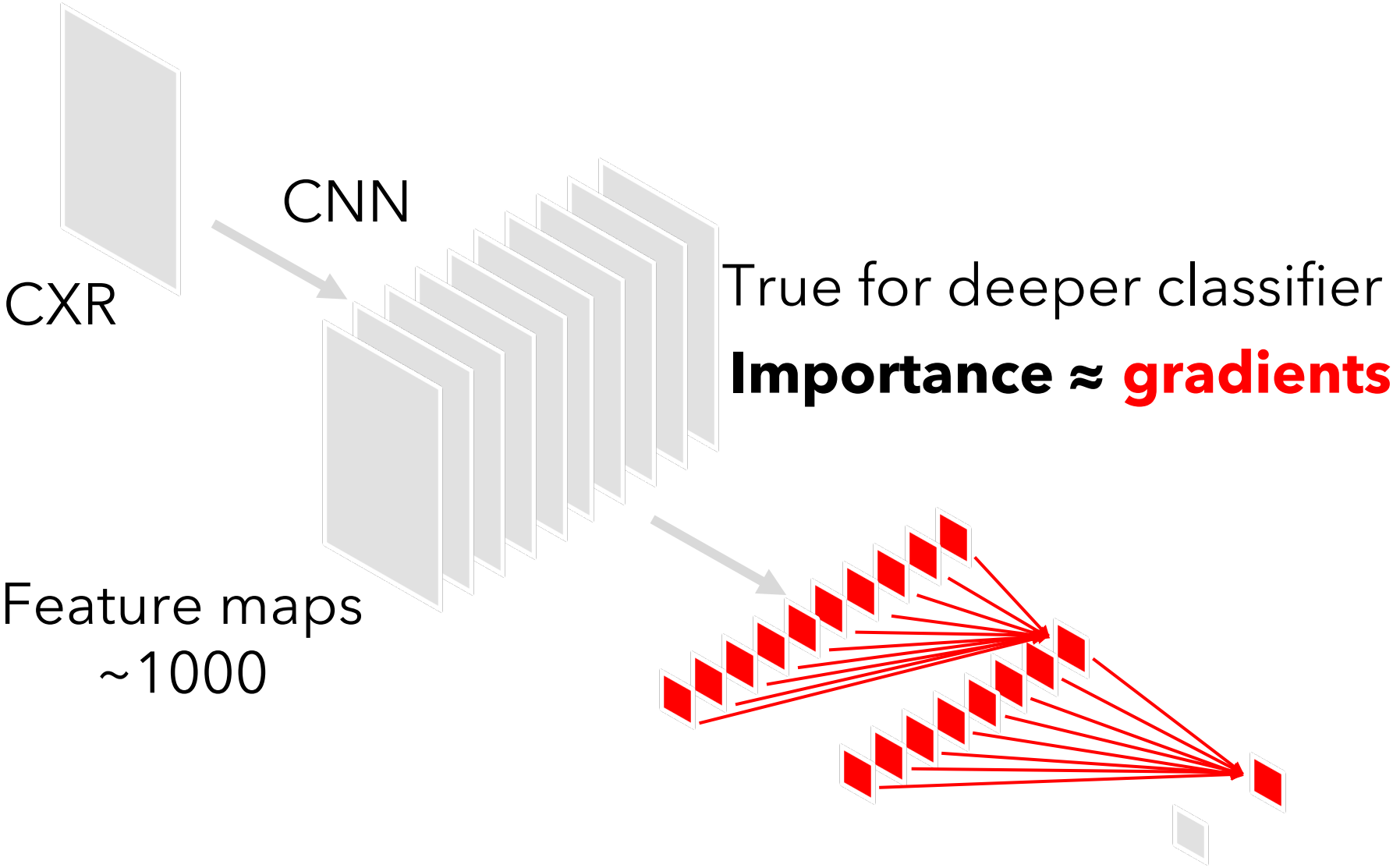
Feature importance = weights



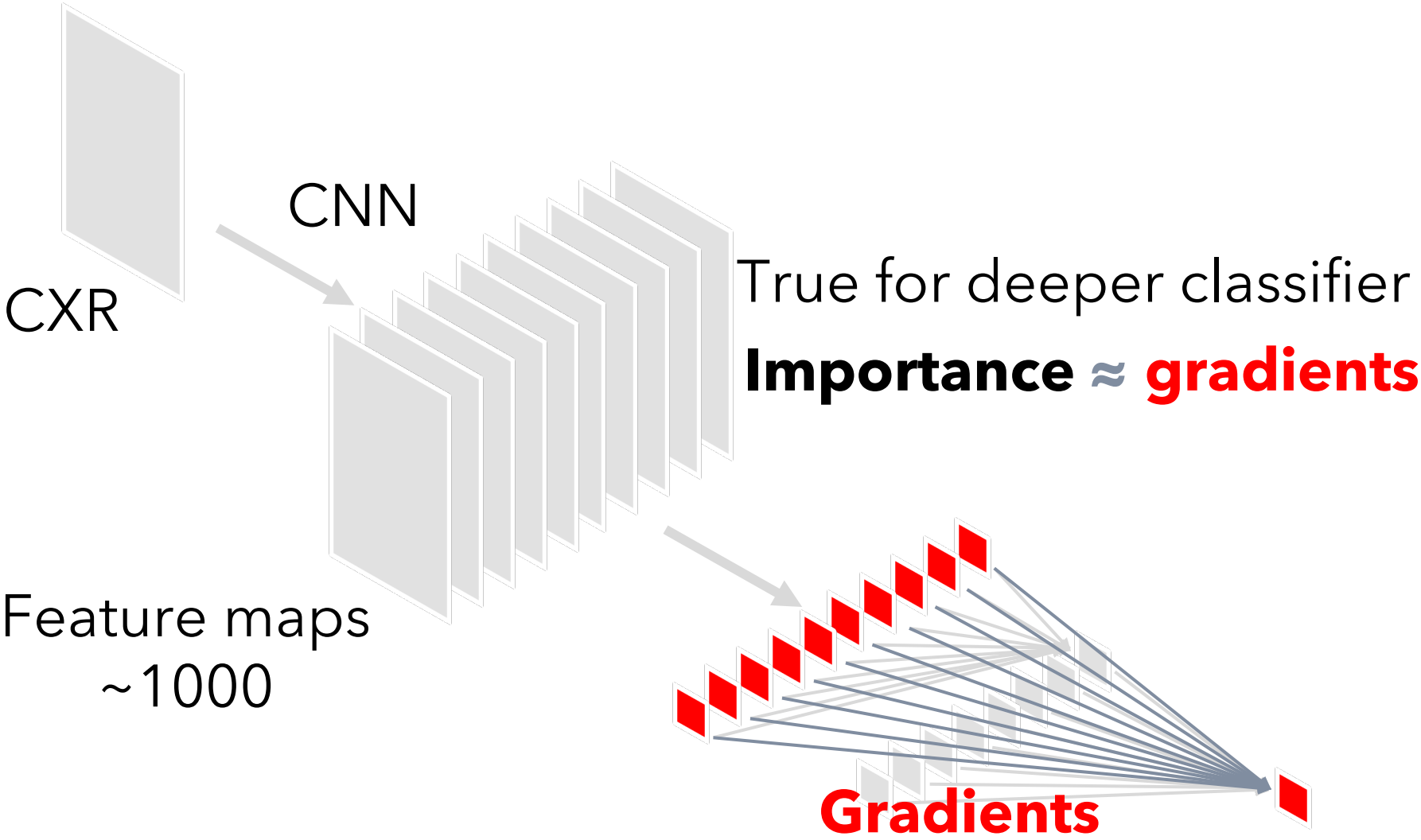
Feature importance ~ gradients



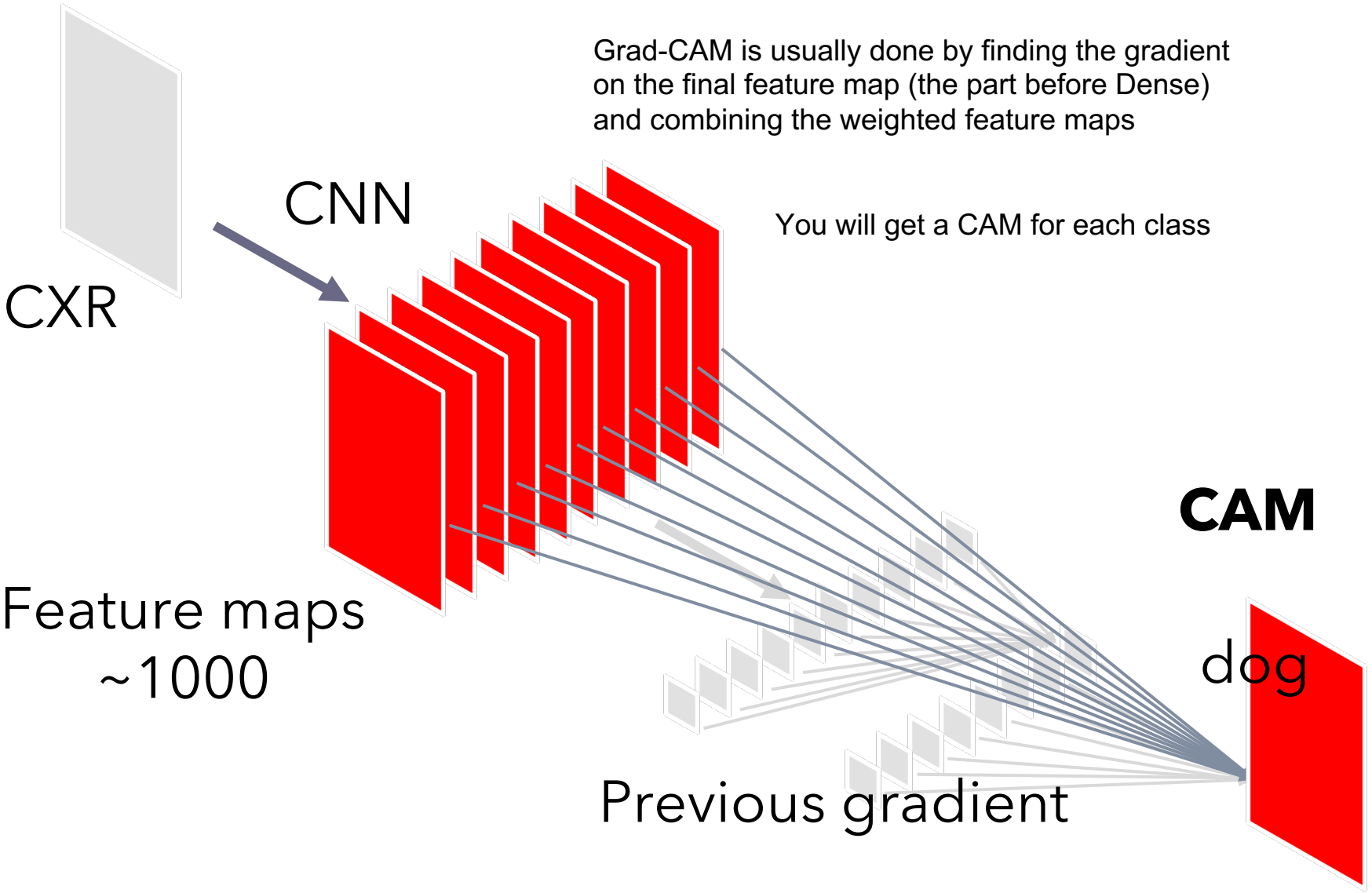
Feature importance ~ gradients



Feature importance ~ gradients



Class activation map (CAM)



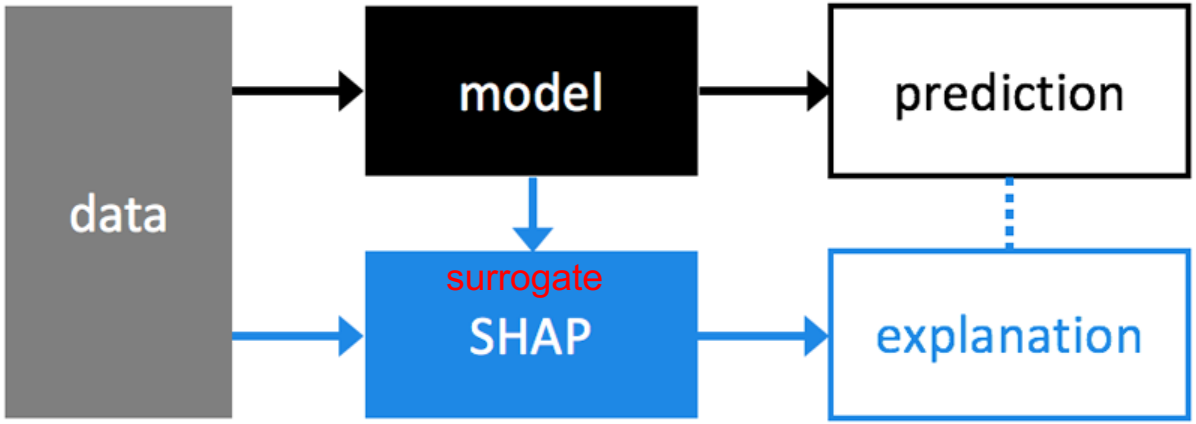
Output level: key ideas

A feature is important if I tweak the feature and the output change a lot. (sensitivity analysis)

Mostly useful for images types

Have a simpler model (**surrogate model**) explains the complicated model.

Converting things to logistic regression



https://github.com/slundberg/shap

Additive feature attribution

$$g(z') = \phi_0 + \sum_{i=1}^{M} \phi_i z'_i,$$

Simplified input features z' have binary values

z' can recover original features x, $x = h_x(z')$ (This mapping depends on x)

Goal: make $g(z') = f(h_x(z'))$. Then we can explain f in terms of simplified features. (**Solved by optimization**)

Example of simplified inputs

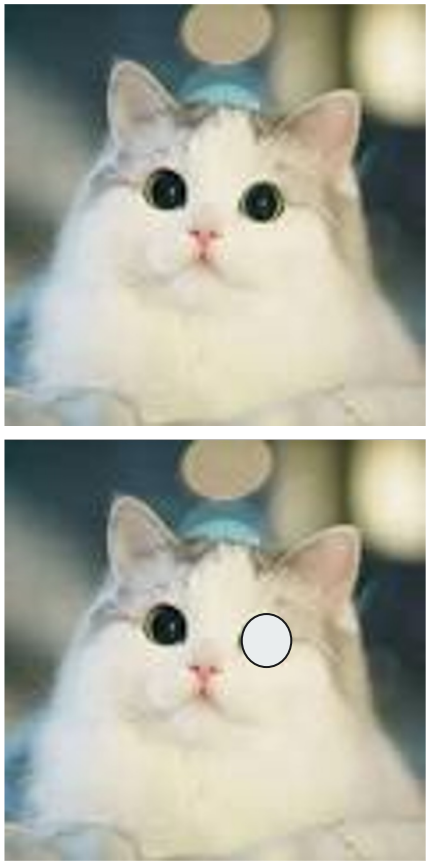
Original x = image

z = 0 if patch is not present

z = 1 if patch is present

$h_x(z) = x$ if, $z=1$

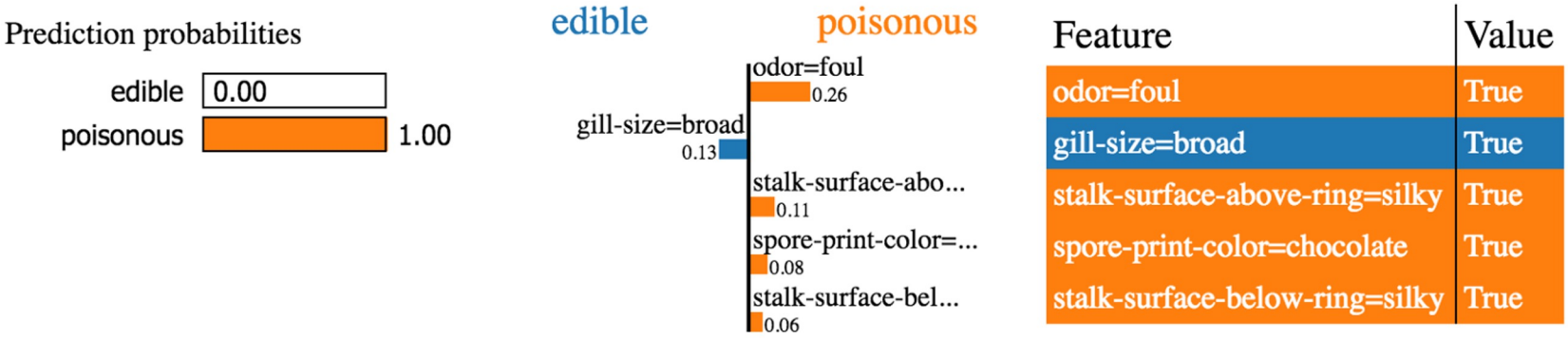
$h_x(z) = x$ with missing patch, if $z=0$



Additive feature attribution

$$g(z') = \phi_0 + \sum_{i=1}^{M} \phi_i z'_i,$$

Many methods fall is additive feature attribution. For example, LIME



Introducing SHAP

SHAP is also an additive feature attribution, but gives credit to expected attribution

Example of simplified inputs

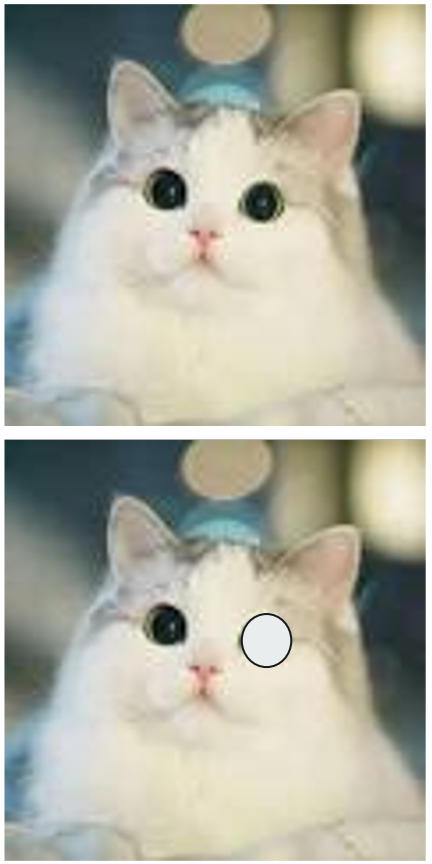
Original x = image

z = 0 if patch is not present

z = 1 if patch is present

$h_x(z)$ = x if, z=1

$h_x(z)$ = x with missing patch, if z=0

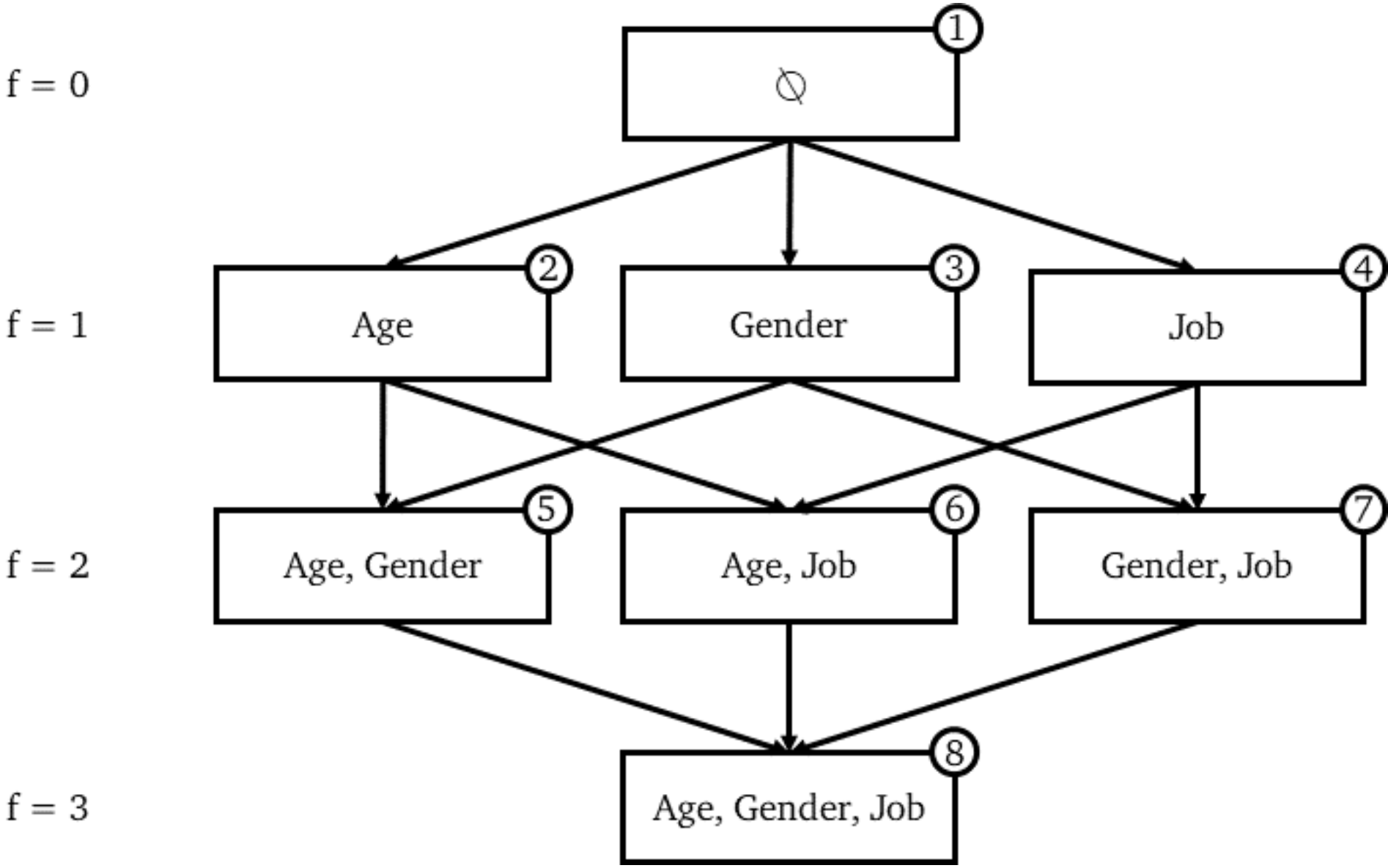


This simplified input talks about this patch.

What happens if we change the other inputs?

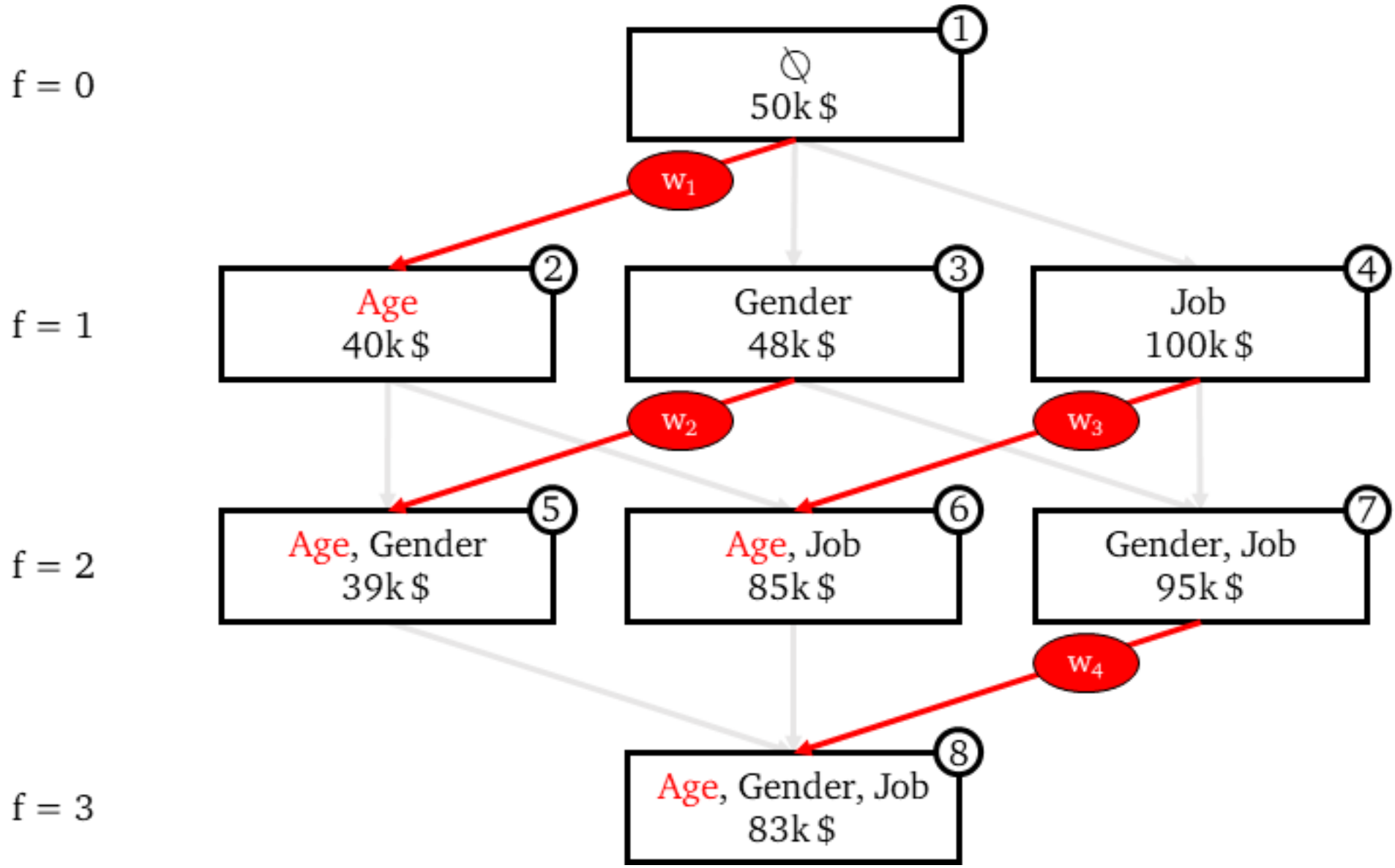
Expected contribution talks about the contribution of the feature regardless of the other contributions

Let's say we have 3 features, and we want to create a model
We can create a power set of features, where we create a model for each set



https://towardsdatascience.com/shap-explained-the-way-i-wish-someone-explained-it-to-me-ab81cc69ef30

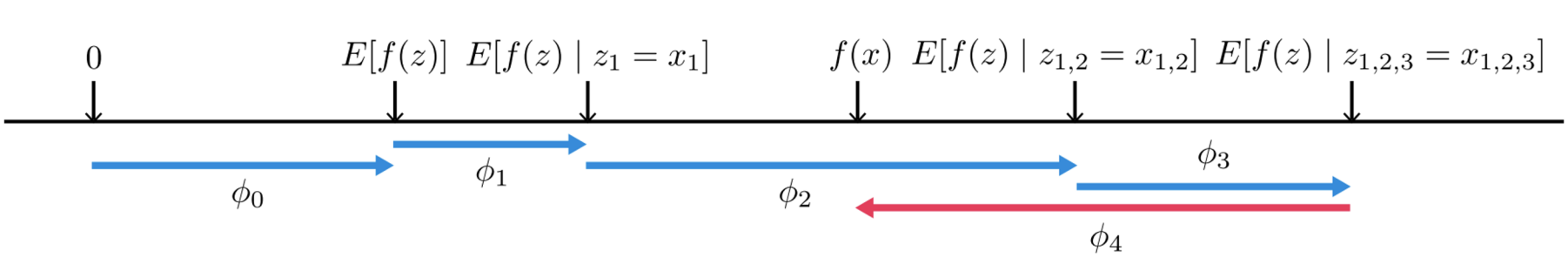
Expected contribution of Age feature is the average of adding age feature



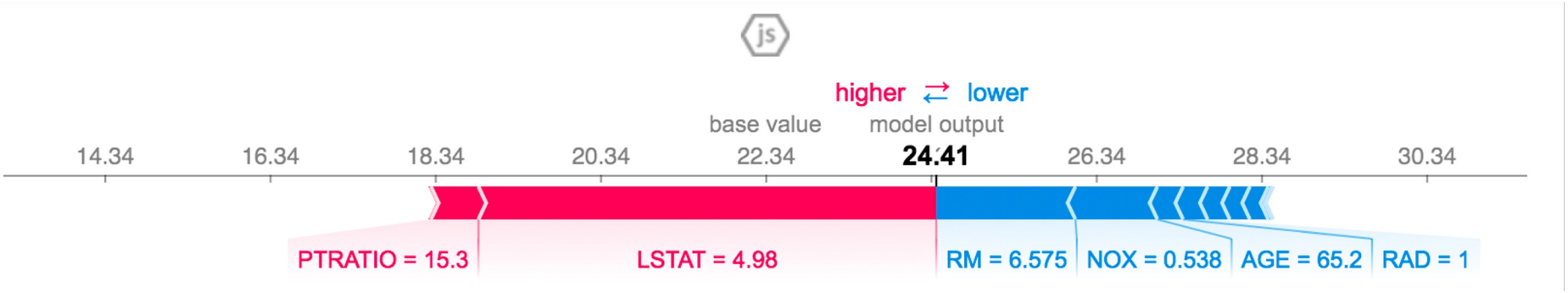
Doing this exactly as shown is intractable in real tasks, SHAP employ tricks and approximations to do this.

https://towardsdatascience.com/shap-explained-the-way-i-wish-someone-explained-it-to-me-ab81cc69ef30

SHAP

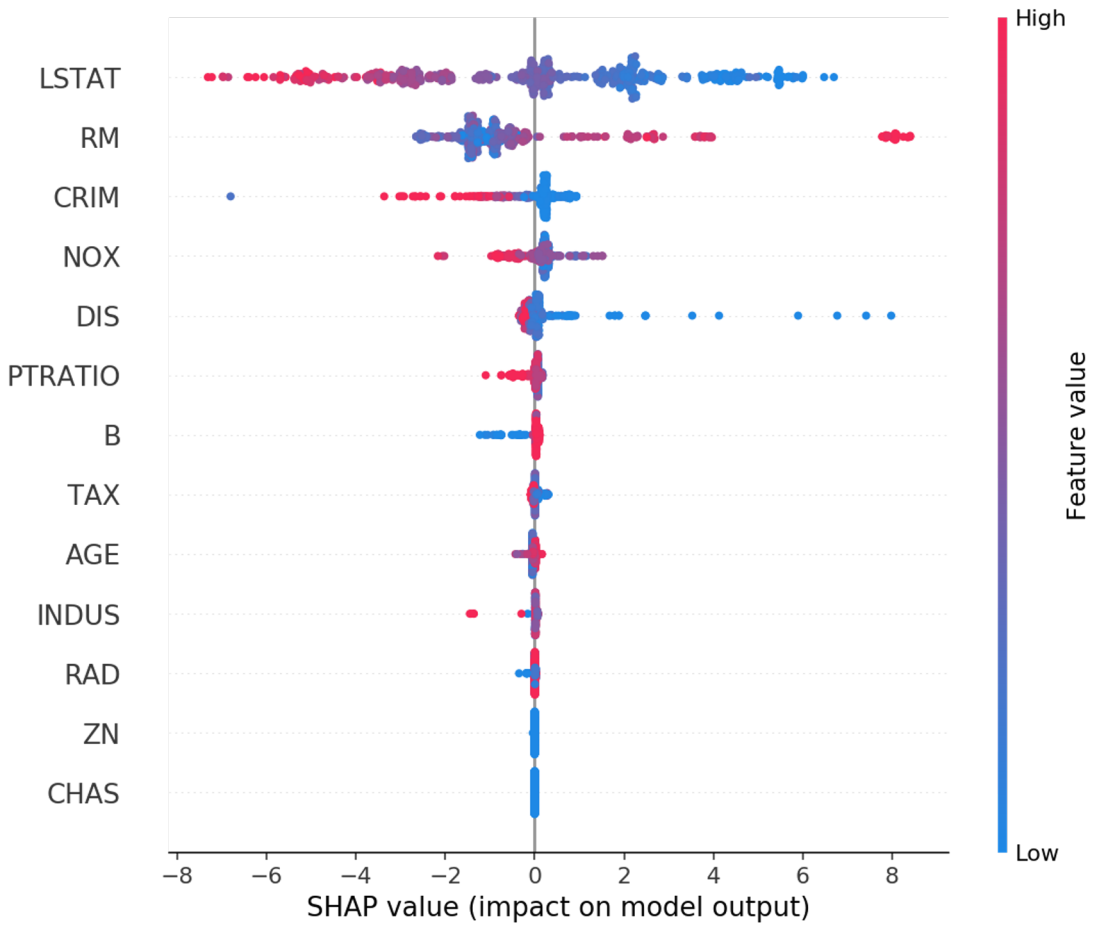

0
$E[f(z)]$
$E[f(z) \mid z_1 = x_1]$
$f(x)$
$E[f(z) \mid z_{1,2} = x_{1,2}]$
$E[f(z) \mid z_{1,2,3} = x_{1,2,3}]$
ϕ_0
ϕ_1
ϕ_2
ϕ_3
ϕ_4

SHAP example



https://github.com/slundberg/shap

SHAP example



Notes on additive attribution

If the features are correlated, it's hard to divide the attribution

Feature 1 = height

Feature 2 = height*2

Notes on attributions

It does not necessary tell how to improve.

If the explainer says the sales is low because of weak marketing

Does not mean increasing marketing will improve sales.

See “Causal Inference”

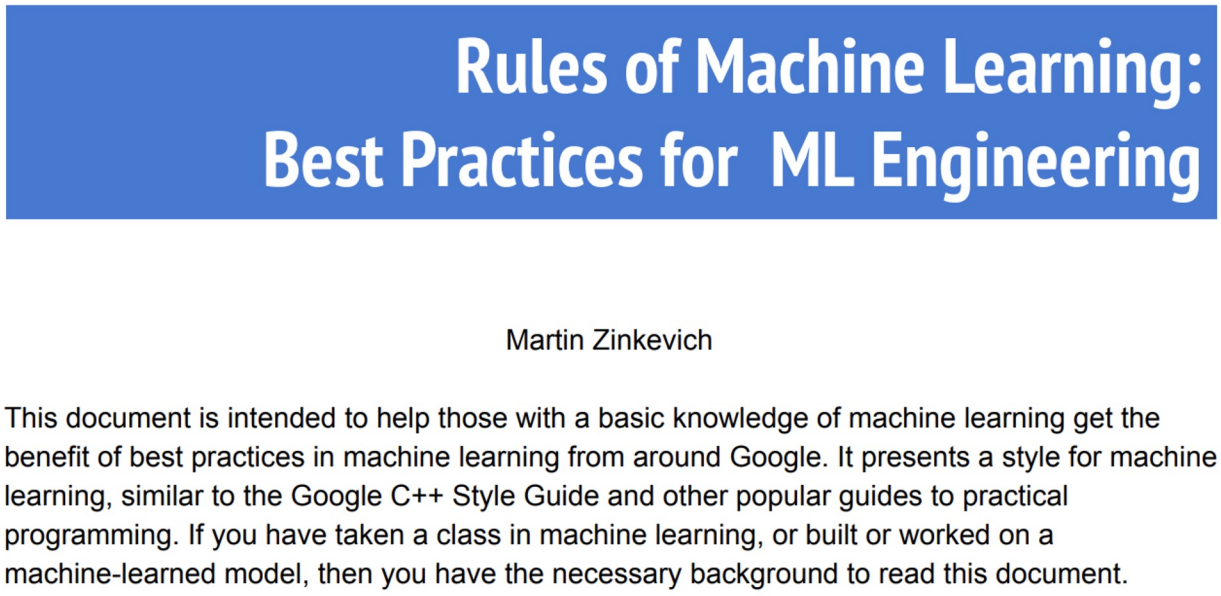
SHAP materials

- https://github.com/slundberg/shap
  - main repo
- https://github.com/linkedin/fasttreeshap
  - Fast SHAP for tree models
- A nice blog that explains SHAP computation
  - https://towardsdatascience.com/shap-explained-the-way-i-wish-someone-explained-it-to-me-ab81cc69ef30
- An improved SHAP: weightedSHAP
  https://github.com/ykwon0407/WeightedSHAP
  - Relaxed the additive assumption of SHAP
  - Can better handle correlated features

Further readings

Rules of Machine Learning: Best Practices for ML Engineering

http://martin.zinkevich.org/rules_of_ml/rules_of_ml.pdf



Rules of Machine Learning:
Best Practices for ML Engineering

Martin Zinkevich

This document is intended to help those with a basic knowledge of machine learning get the benefit of best practices in machine learning from around Google. It presents a style for machine learning, similar to the Google C++ Style Guide and other popular guides to practical programming. If you have taken a class in machine learning, or built or worked on a machine-learned model, then you have the necessary background to read this document.

Rule #1: Don't be afraid to launch a product without machine learning.

Ethics issues

- Abuse
- Bias

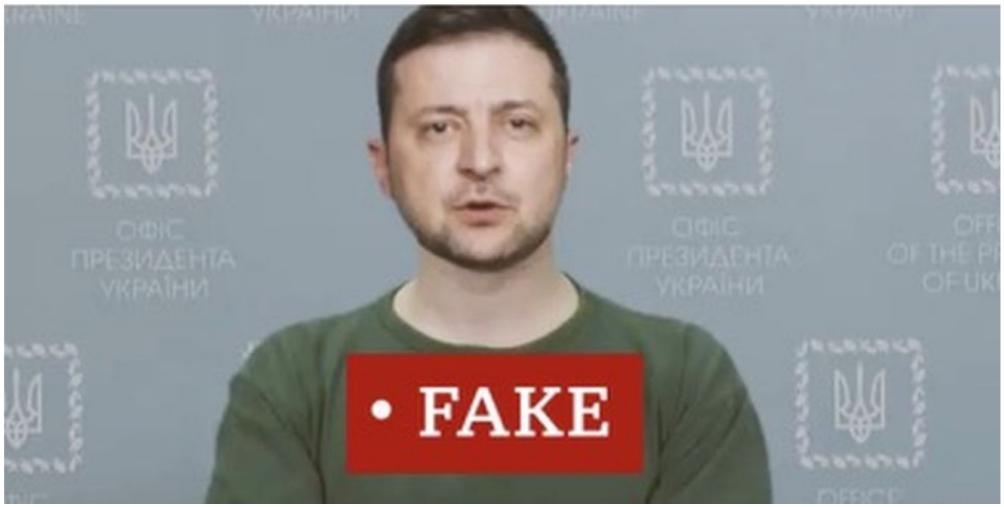
Deep fakes

Deepfake presidents used in Russia-Ukraine war

By Jane Wakefield
BBC Technology

5 days ago

Russia-Ukraine war



https://www.bbc.com/news/technology-60780142

PRO CYBER NEWS

Fraudsters Used AI to Mimic CEO's Voice in Unusual Cybercrime Case

Scams using artificial intelligence are a new challenge for companies

PHOTO: SIMON DAWSON/BLOOMBERG NEWS

By *Catherine Stupp*

Updated Aug. 30, 2019 12:52 pm ET

MOST POPULAR NEWS

1. Where State Abortion Laws Stand if Roe v. Wade Is Overturned 
2. They Kept Paying When Student Loan Debt Paused 
3. NFT Sales Are Flatlining 
4. Cerebral's Preferred Pharmacy Truepill Halts Adderall Prescriptions for All Customers 
5. Immigrants to Get Extension for Expiring or Expired U.S. Work 

https://www.wsj.com/articles/fraudsters-use-ai-to-mimic-ceos-voice-in-unusual-cybercrime-case-11567157402

Tesla is updating Autopilot's 'Hold Steering Wheel' alert after complaints, says Elon Musk

Fred Lambert - Jun. 13th 2018 6:12 am ET @FredericLambert

https://electrek.co/2018/06/13/tesla-autopilot-hold-steering-wheel-alerts-complaints/

What is Project Nimbus, and why are Google workers protesting Israel deal?

Tech workers are protesting against the use of artificial intelligence and other technologies by Israel in its war on Gaza.

By **Al Jazeera Staff**

23 Apr 2024 | Updated: 9 hours ago

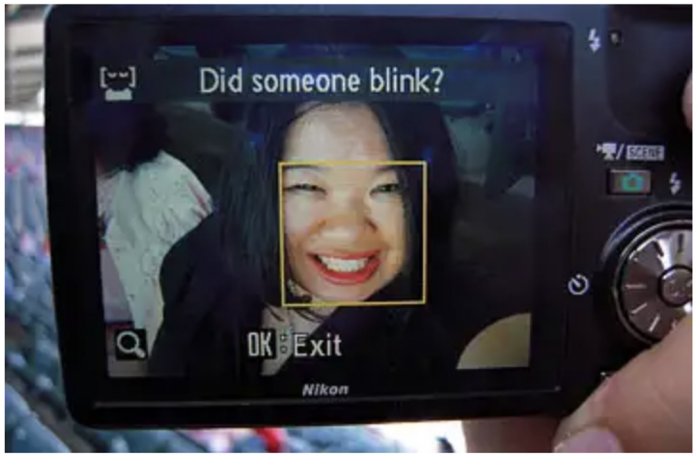
Save articles to read later and create

https://www.aljazeera.com/news/2024/4/23/what-is-project-nimbus-and-why-are-google-workers-protesting-israel-deal

Bias

- Our society is biased which leads to biased data and biased models
  - Understanding how models make prediction help us remove the bias
- Our morality is also biased
  - Understanding the cost-benefit tradeoff

Harms of Bias



Allocation

Models allocate resource to certain groups of people more - credit prediction

Representation

Models have more data of certain demographic and does better

http://content.time.com/time/business/article/0,8599,1954643,00.html

Study finds gender and skin-type bias in commercial artificial-intelligence systems

Examination of facial-analysis software shows error rate of 0.8 percent for light-skinned men, 34.7 percent for dark-skinned women.

Watch Video

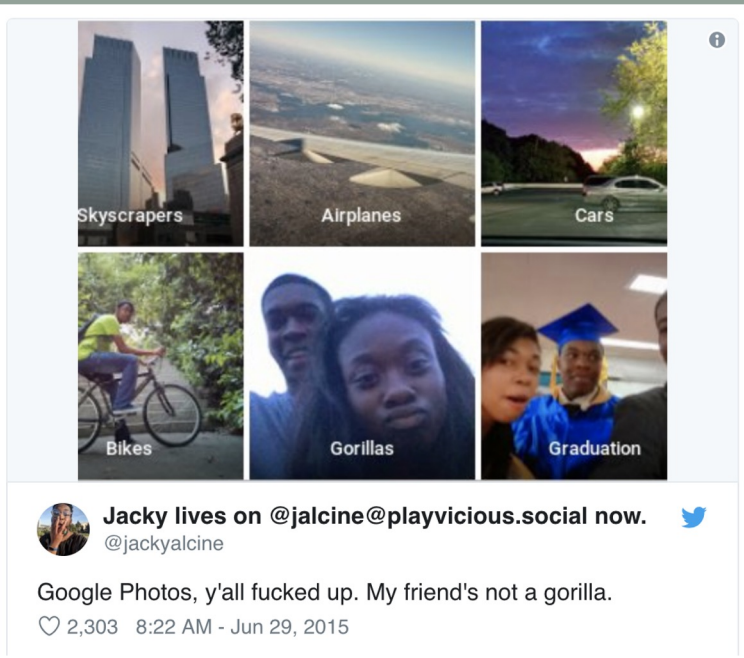
Larry Hardesty | MIT News Office
February 11, 2018

Press Inquiries | PRESS MENTIONS

http://news.mit.edu/2018/study-finds-gender-skin-type-bias-artificial-intelligence-systems-0212

Ways to combat bias

- Improve accuracy <- hard
- Blacklist
- Equal representation
- Awareness
- Test the system on general users
- Try to de-bias <- if known bias, hard. If unknown bias, super hard



Google 'fixed' its racist algorithm by removing gorillas from its image-labeling tech

Nearly three years after the company was called out, it hasn't gone beyond a quick workaround

By James Vincent | @jjvincent | Jan 12, 2018, 10:35am EST

https://www.theverge.com/2018/1/12/16882408/google-racist-gorillas-photo-recognition-algorithm-ai

‘We definitely messed up’: why did Google AI tool make offensive historical images?

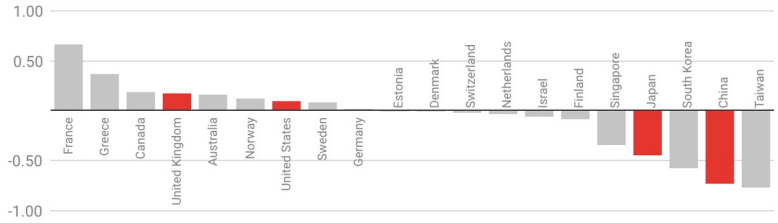
https://www.theguardian.com/technology/2024/mar/08/we-definitely-messed-up-why-did-google-ai-tool-make-offensive-historical-images

Experts say Gemini was not thoroughly tested, after image generator depicted variety of historical figures as people of colour

Google’s Gemini AI illustrations of a 1943 German soldier. Photograph: Gemini AI/Google

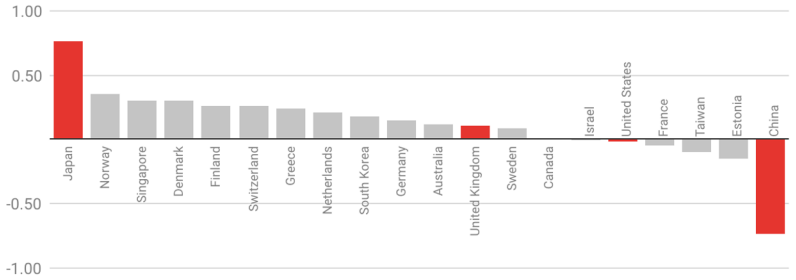
Moral machine

Countries with more individualistic cultures are more likely to spare the young



A comparison of countries piloting self-driving cars: If the bar is closer to 1, respondents placed a greater emphasis on sparing the young; if the bar is closer to -1, respondents placed a greater emphasis on sparing the old; 0 is the global average.

How countries compare in sparing pedestrians over passengers



If the bar is closer to 1, respondents placed a greater emphasis on sparing pedestrians; if the bar is closer to -1, respondents placed a greater emphasis on sparing passengers; 0 is the global average.

Problems with bias is both a social issue and a technological issue

https://www.technologyreview.com/s/612341/a-global-ethics-study-aims-to-help-ai-solve-the-self-driving-trolley-problem/

- “If I were to guess like what our biggest existential threat is, it’s probably that. So we need to be very careful with the artificial intelligence. There should be some regulatory oversight maybe at the national and international level, just to make sure that we don’t do something very foolish.”

- “I think people who are naysayers and try to drum up these doomsday scenarios — I just, I don’t understand it. It’s really negative and in some ways I actually think it is pretty irresponsible”

Poll

Further Learning

- In Chula
  - NLP, ASR – more domain specific, less technical
- https://www.youtube.com/c/YannicKilcher
  - Funny and educational
- Follow your favorite researcher on Twitter
- https://cds.nyu.edu/deep-learning/
  - NYU has a nice advanced grad level deep learning course
  - Graph neural networks, energy-based models
- https://www.bradyneal.com/causal-inference-course
  - Causal inference
  - How to identify bias and de-bias?
- https://stanford-cs329s.github.io/syllabus.html
  - A more deployment-oriented course