Pitney Bowes / Baruch
Data Challenge

Preventive Maintenance of Mailing Meters

Baruch College (4/12/2021)

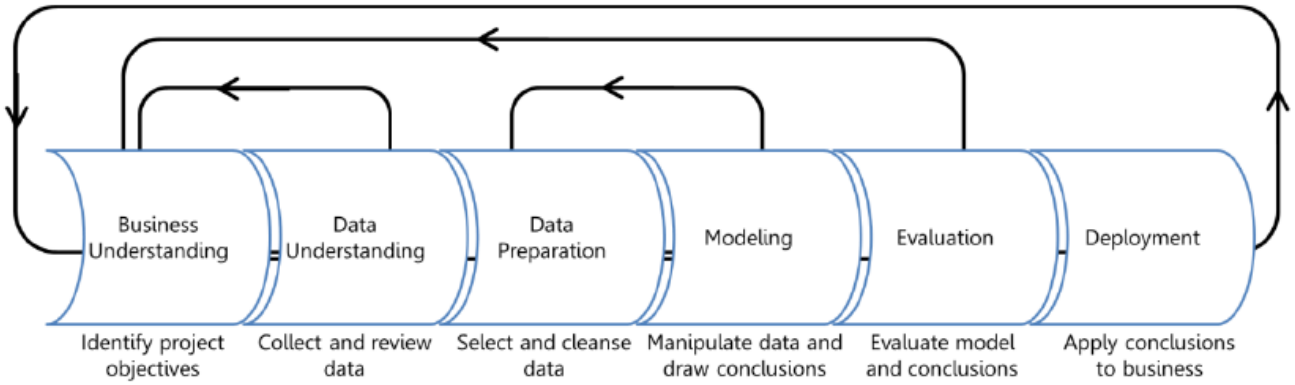
pitney bowes

Baruch COLLEGE CUNY

Traditional ML Lifecycle in Business (AS-IS)

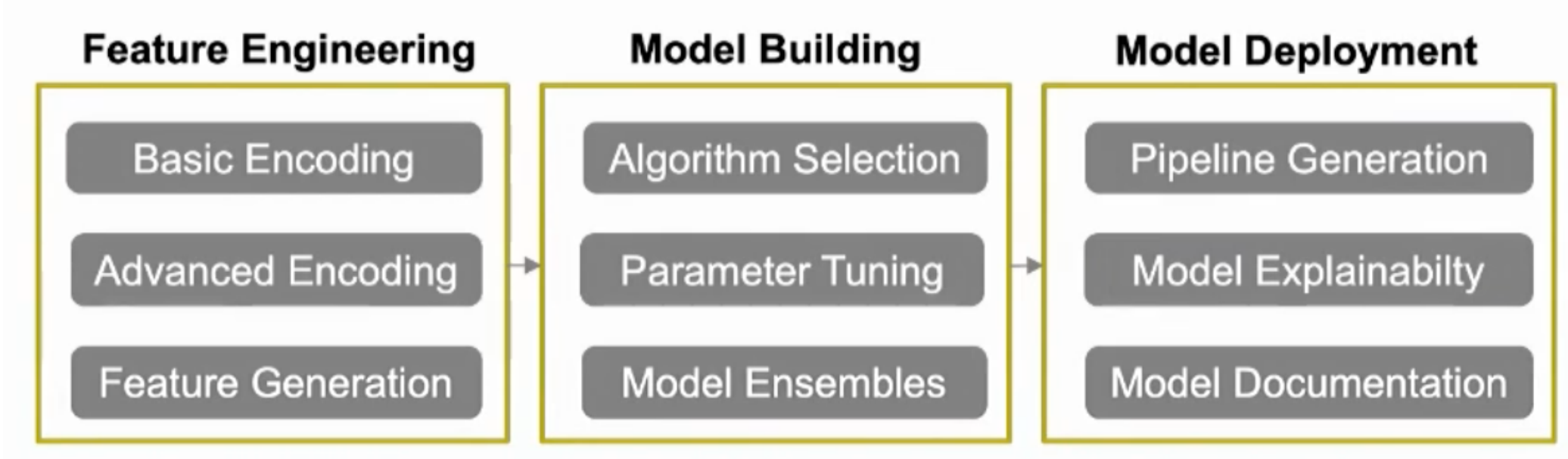
CRISP-DM Method

Cross-industry standard for Data Mining



Source:
https://www.sv-europe.com/crisp-dm-methodology/
https://blogs.msdn.microsoft.com/azuredev/2017/02/19/the-data-science-process-with-azure-machine-learning/


WORKFLOW
Collect Data
Visualize & Understand
Feature Engineering & Transformation
Train
Deploy
Evaluate


Feature Engineering
Basic Encoding
Advanced Encoding
Feature Generation
Model Building
Algorithm Selection
Parameter Tuning
Model Ensembles
Model Deployment
Pipeline Generation
Model Explainabilty
Model Documentation

Dealing with NULL/missing values

Problem: Instances with NULL values are ignored by some ML algorithms
Solution: Impute missing values (**ONLY** in cases where it makes sense to do so and does not violate domain rules

- Ignoring instances with missing values can lead to significant training data lost, particularly in high-dimensional datasets
- In cases where it applies, a better strategy would be to impute the missing value:
  - Mean
  - Median
  - Mode
  - Use a model to predict that value

One type of imputation algorithm is **univariate**, which imputes values in the i-th feature dimension using only non-missing values in that feature dimension (e.g. *impute.SimpleImputer*). By contrast, **multivariate** imputation algorithms use the entire set of available feature dimensions to estimate the missing values (e.g. *impute.IterativeImputer*).

https://scikit-learn.org/stable/modules/impute.html

Dealing with high-dimensional data

Problem: Sparse datasets with many features
Solution: implement dimensionality reduction techniques (i.e., SVD, PCA, Clustering)

```
import pandas as pd
```

```
df = pd.read_csv('train.csv')
```

```
df.head()
```

	deviceid	avg_time_charging_lag1	avg_time_charging_lag2	avg_time_charging_lag3	avg_time_charging_lag7	charging_rate_lag3	charging_rate_lag7	avg_time_discharging_lag1	avg_tim
0	28647	5.12	41.11	6.56	25.39	0.086667	-0.006667	4.37	
1	36175	36.60	5.16	6.23	6.96	0.136667	-1.296667	62.67	
2	16107	5.51	5.04	4.52	5.96	-0.460000	-0.083333	5.13	
3	27362	4.66	39.85	35.76	40.69	0.076667	-0.006667	3.93	
4	19463	5.10	43.24	4.63	5.26	0.040000	-0.153333	4.69	

5 rows × 55 columns

```
df.shape
```

```
(40500, 55)
```

Feature engineering

Problem: Augment feature space
Solution: Feature engineering (SVD, PCA, Interactions, Clustering)

- **Log transformation log(x):** the larger the x, the lower the log(x) increments. Useful to smooth long-tailed data.
- **Scaling:**
  - MinMax scaling: squeezes values into a range of [0,1] [sklearn → MinMaxScaler()]
  - Standard Z-scaling: after standardization, a feature has mean 0 and variance of 1. [sklearn→ scale()]
- **Interaction features:** créate feature combinations (x_1,x_2)->$[x_1,x_2, x_1x_2, x_1^2, x_2^2)$ [sklearn – polynomial features]

Feature engineering: Categorical Encoding

Problem: Many ML algorithms work better with numerical features.
Solution: Perform one-hot encoding, frequency encoding

```
df = pd.get_dummies(customer_info , columns = [attribute1’, attribute2’,
attribute3'], drop_first = True)
```

Feature engineering: temporal features

Problem: Many ML algorithms work better with numerical features.
Solution: apply feature engineering techniques to temporal features

- time binning
- Engineer features: is_weekday, is_weekend, quarter, year, spend_week, spend_year, first_saturdaymonth
- time_differences [between events]: days_since_last_interaction, days_since_last_upgrade

Model Building + Parameter optimization

Problem: what family of models work better with this data?
Solution: model building + parameter optimization

Data Leakage

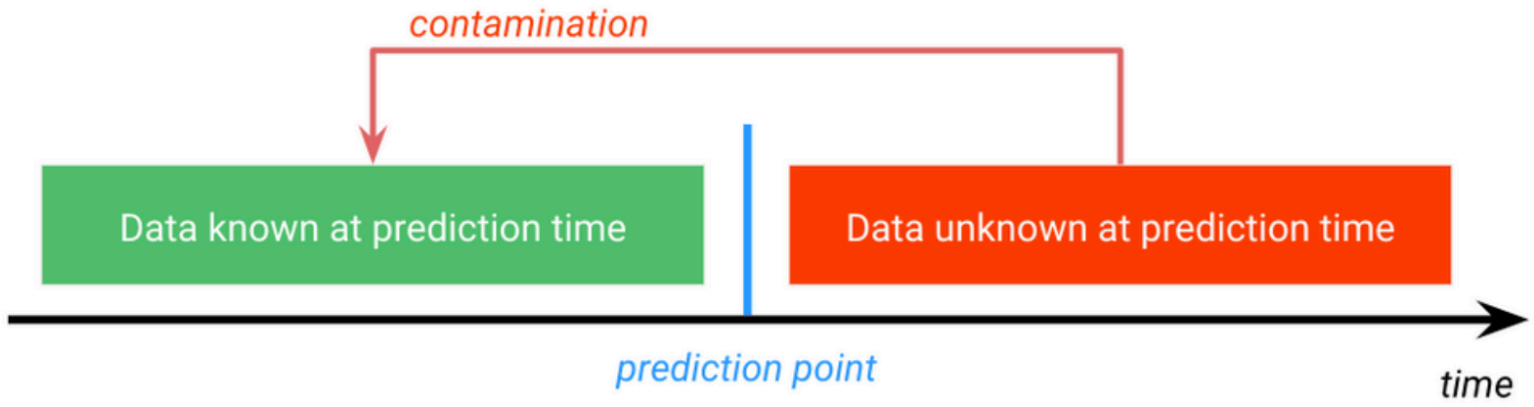
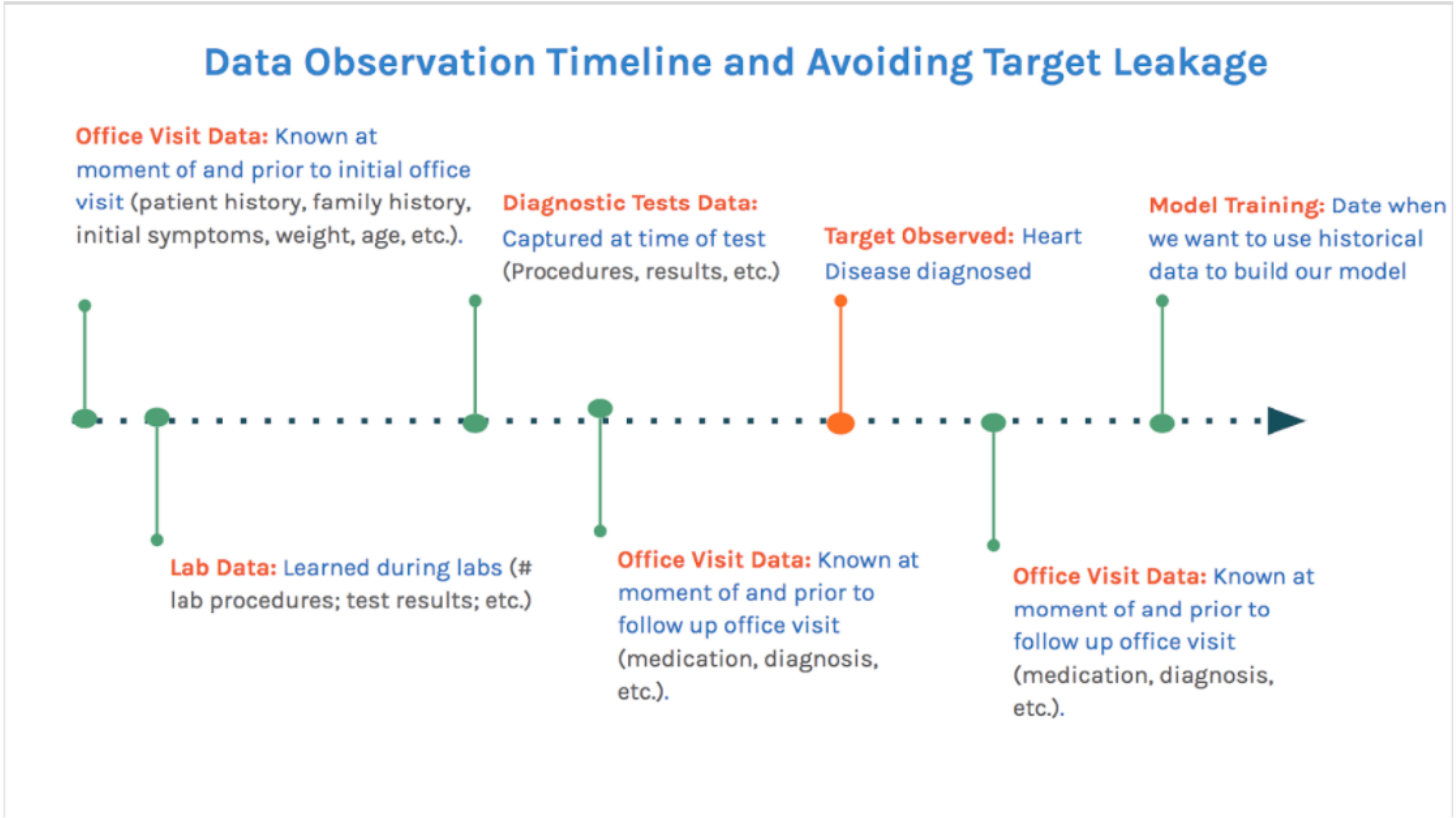


Figure 2. Target leakage visualization diagram (Source: Yuriy Guts)

Image source: https://community.datarobot.com/t5/blog/what-is-target-leakage-and-how-do-i-avoid-it/ba-p/1973

A few killer features have been showing leakage so don't drop that id too soon!



Source: https://www.datarobot.com/wiki/target-leakage/

Cross validation - leakage

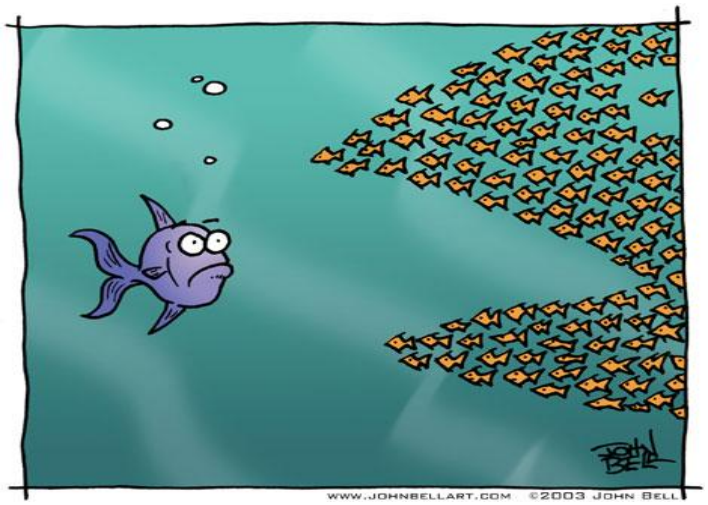
participant	age	diagnosis	score	session	.folds_1	.folds_2	.folds_3
10	32	0	29	1	3	1	1
10	32	0	55	2	3	1	1
10	32	0	81	3	3	1	1
2	23	0	24	1	4	3	4
2	23	0	40	2	4	3	4
2	23	0	67	3	4	3	4
4	21	0	35	1	2	2	2
4	21	0	50	2	2	2	2
4	21	0	78	3	2	2	2
9	34	0	33	1	1	4	3

Ensemble

The premise is that a group of *weak* models is better than one *strong* model.

The idea is to promote diversity

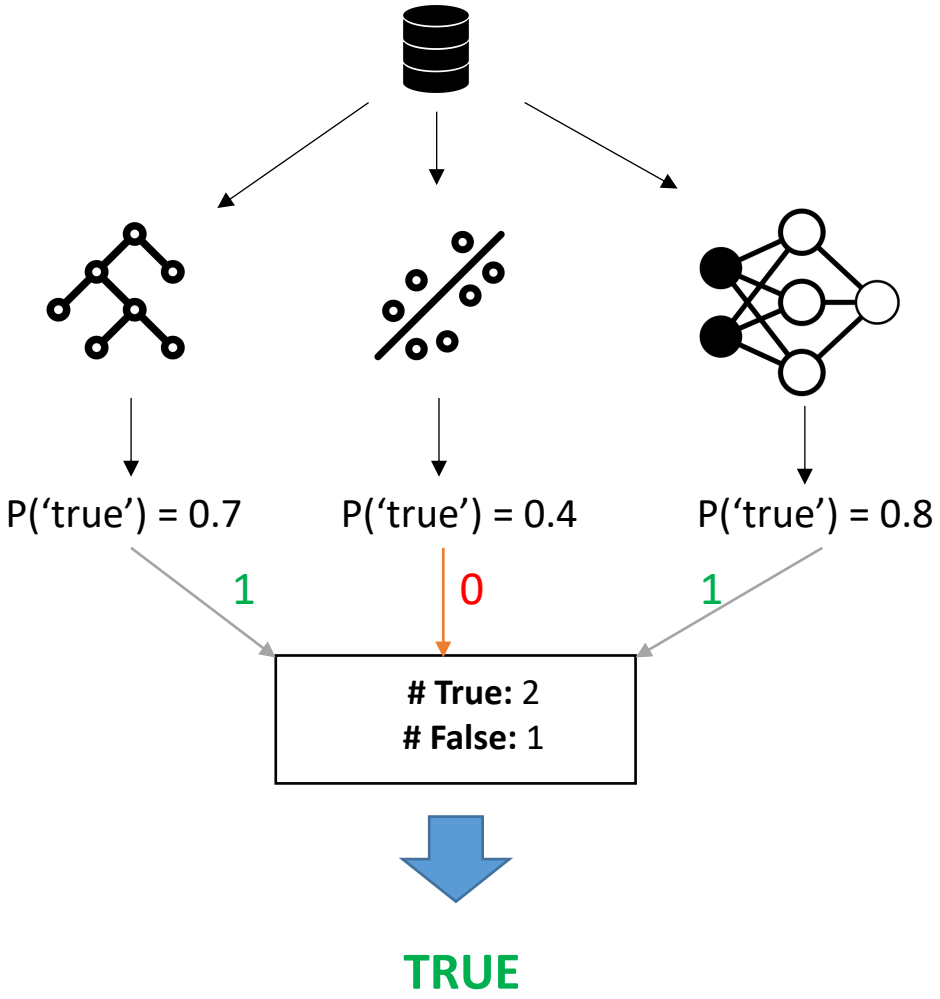
1. Changing the training data (e.g., different models using different data)[
2. Changing the attribute set (e.g., different models use different attributes)
3. Different algorithms
4. Changing the parameters of the algorithms



Each individual model makes a contribution. Individual weaknesses and biases are offset by the strength of the numbers

...

Ensemble: voting



Data

Model

Model output

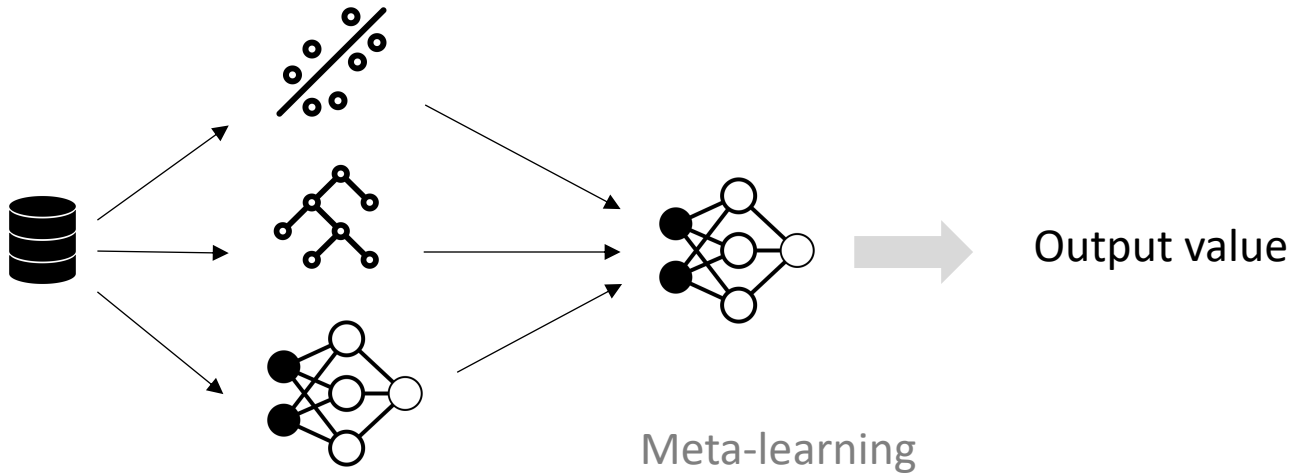
Count Votes

Prediction

Stacking

Stacked generalization (or stacking) is a way of combining multiple models, that introduces the concept of a meta learner.

The predictions, generated by using various machine learning algorithms, are used as inputs in a second-layer learning algorithm. This second-layer algorithm is trained to optimally combine the model predictions to form a new set of predictions.



Stacking Implementation (h2o.ai)

Stacking involves training a second-level "metalearner" to find the optimal combination of the base learners.

Source: http://docs.h2o.ai/h2o/latest-stable/h2o-docs/data-science/stacked-ensembles.html

3 steps:

1. Set up the ensemble
    a. Specify a list of L base algorithms (with a specific set of parameters)
    b. Specify a metalearning algorithm
2. Train the ensemble
    a. Train each of the L base algorithms and collect cross-validated predicted values ("level one" data)
    b. Train the metalearning algorithm on the level-one data.
3. Predict on new data

Parameter Optimization

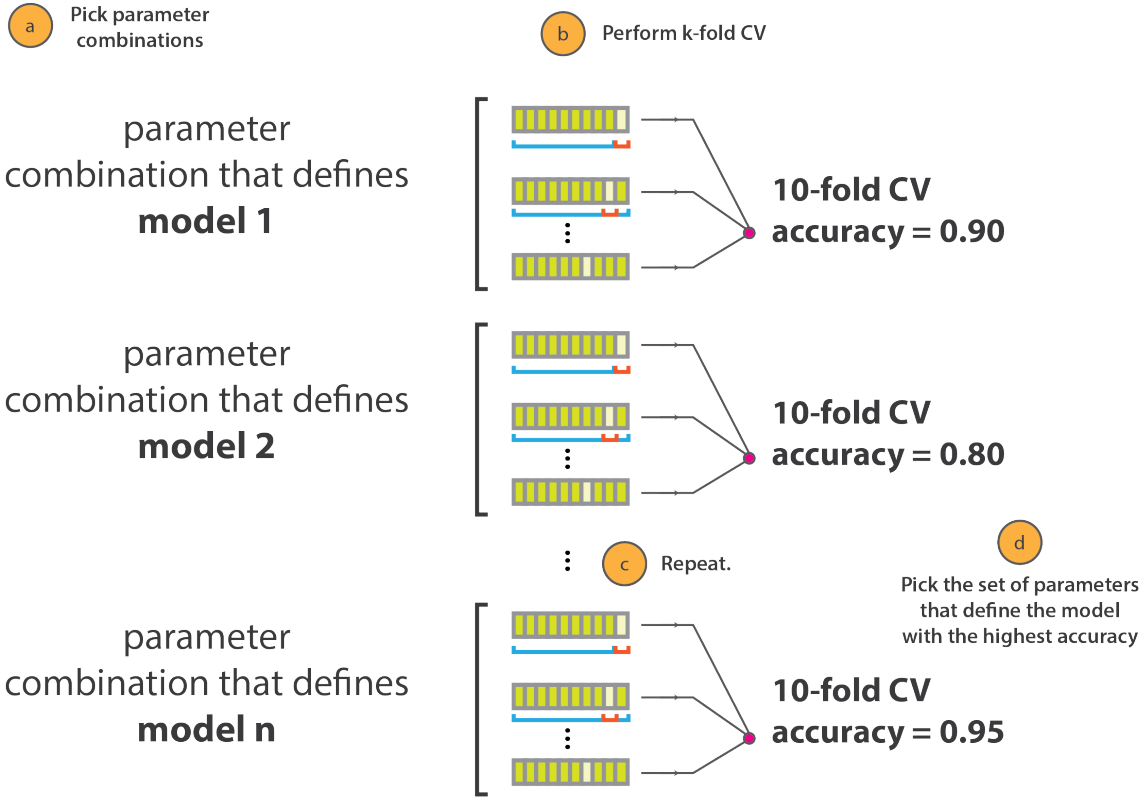
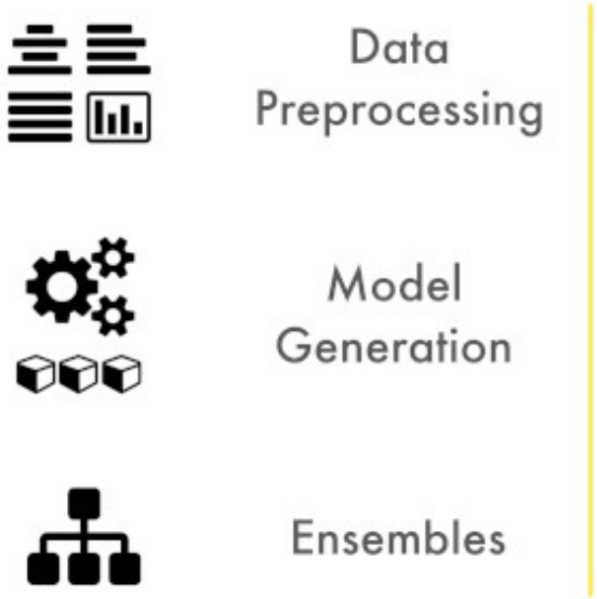


Image source: https://www.kaggle.com/ishivinal/hyperparamters-optimization-gs-rs-boa-tpe-hb-ga

AutoML



Aspects of AutoML

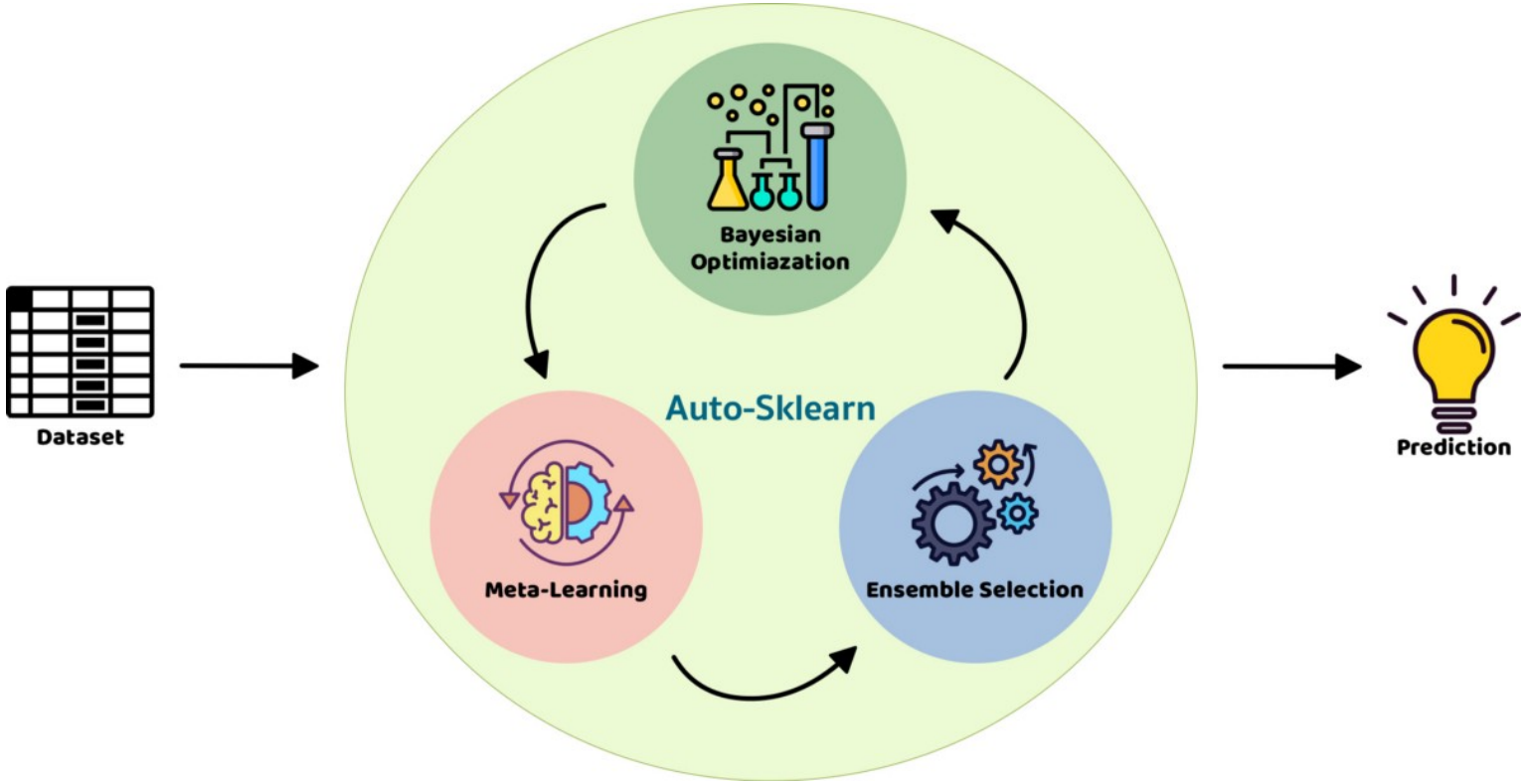
- Imputation, one-hot encoding, standardization
- Feature selection and/or feature extraction (e.g. PCA)
- Count/Label/Target encoding of categorical features

- Cartesian grid search or random grid search
- Bayesian hyperparameter optimization
- Individual models can be tuned using a validation set

- Ensembles often out-perform individual models
- Stacking/Super Learning (Wolpert, Breiman)
- Ensemble Selection (Caruana)

Source: https://www.h2o.ai/products/h2o-automl/
https://automl.github.io/auto-sklearn/master/

Auto-Sklearn



Tips

- Feature engineering is key!
- Sometimes simple and elegant solutions are better than complex models.
- Create ensemble and if you have time build stacked models and do grid parameter optimization.

Jupyter Lab

You could use stacking in the project as an optimization

You can also optimize the parameters using a grid search:
http://docs.h2o.ai/h2o/latest-stable/h2o-docs/grid-search.html

Or scikit learn:
https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html

More resources

https://www.kaggle.com/arthurtok/introduction-to-ensembling-stacking-in-python

https://www.quora.com/What-are-the-differences-between-the-three-commonly-ensemble-learning-techniques-stacking-boosting-and-bagging

https://towardsdatascience.com/ensemble-methods-in-machine-learning-what-are-they-and-why-use-them-68ec3f9fef5f

https://blogs.sas.com/content/subconsciousmusings/2017/05/18/stacked-ensemble-models-win-data-science-competitions/

https://towardsdatascience.com/how-to-create-ensemble-models-using-rapid-miner-72a12160fa51