Machine Learning Odyssey Part 6

Feb 18, 2024

Seungeun Lee

So far…

- Tabular data (data preprocessing, data imbalance, …)
- AutoML
- XGBoost (CNN of machine learning), LightGBM
- TabNet
- Random Forest
- …

① | Tabular Data

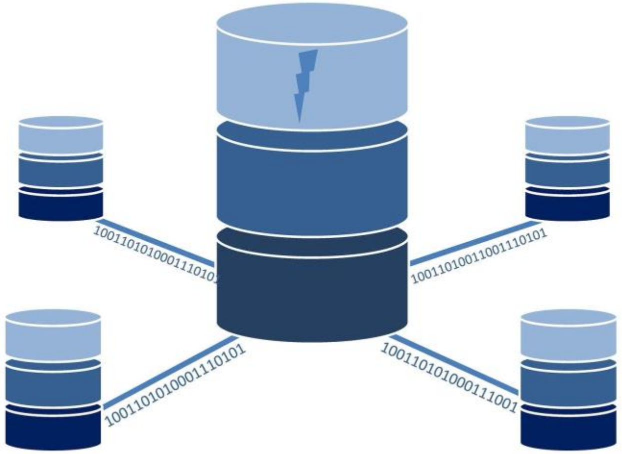
② | SVM

③ | AutoML

④ | Hyperparameter Tuning

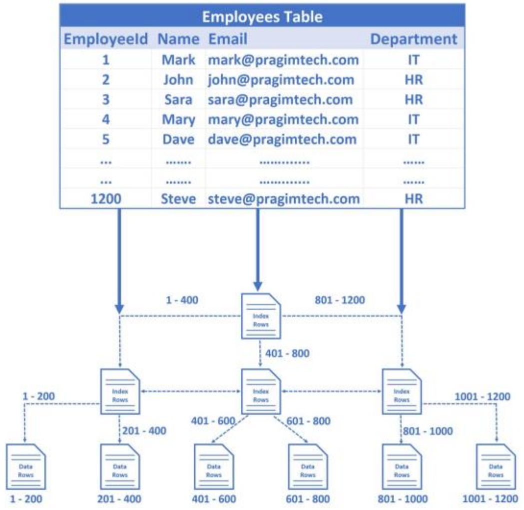
Tabular data

- Tabular data ∈ Structured Data
- Data extracted from a database (DB), represented in a tabular format with rows and columns



DataBase

SQL Query



Employees Table			
EmployeeId	Name	Email	Department
1	Mark	mark@pragimtech.com	IT
2	John	john@pragimtech.com	HR
3	Sara	sara@pragimtech.com	HR
4	Mary	mary@pragimtech.com	IT
5	Dave	dave@pragimtech.com	IT
...	....	....	......
...	....	....	......
1200	Steve	steve@pragimtech.com	HR

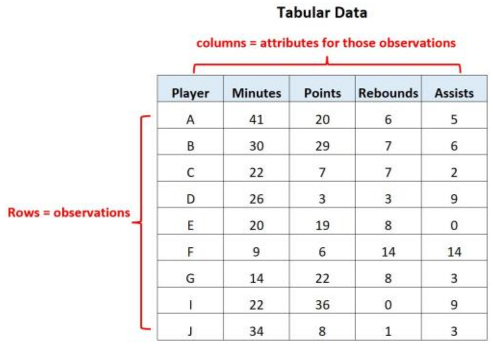
Data

	x	target	o	x	o	o						
	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S
...	...	...	...	...	...	...	...	...	...	...	...	...
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	NaN	S
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	B42	S
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.4500	NaN	S
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C148	C
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	NaN	Q

891 rows × 12 columns

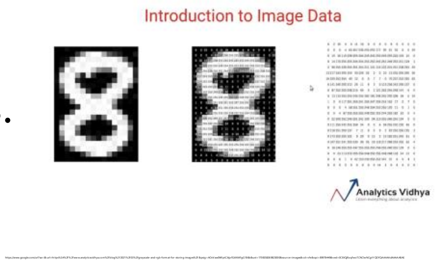
Titanic dataset (https://inhovation97.tistory.com/65)

Tabular vs. Image data



Player	Minutes	Points	Rebounds	Assists
A	41	20	6	5
B	30	29	7	6
C	22	7	7	2
D	26	3	3	9
E	20	19	8	0
F	9	6	14	14
G	14	22	8	3
I	22	36	0	9
J	34	8	1	3

vs.



- Image data $\in$ Unstructured Data
- While there is a lack of overall structural differences btw tabular and image data, what are the distinctions?

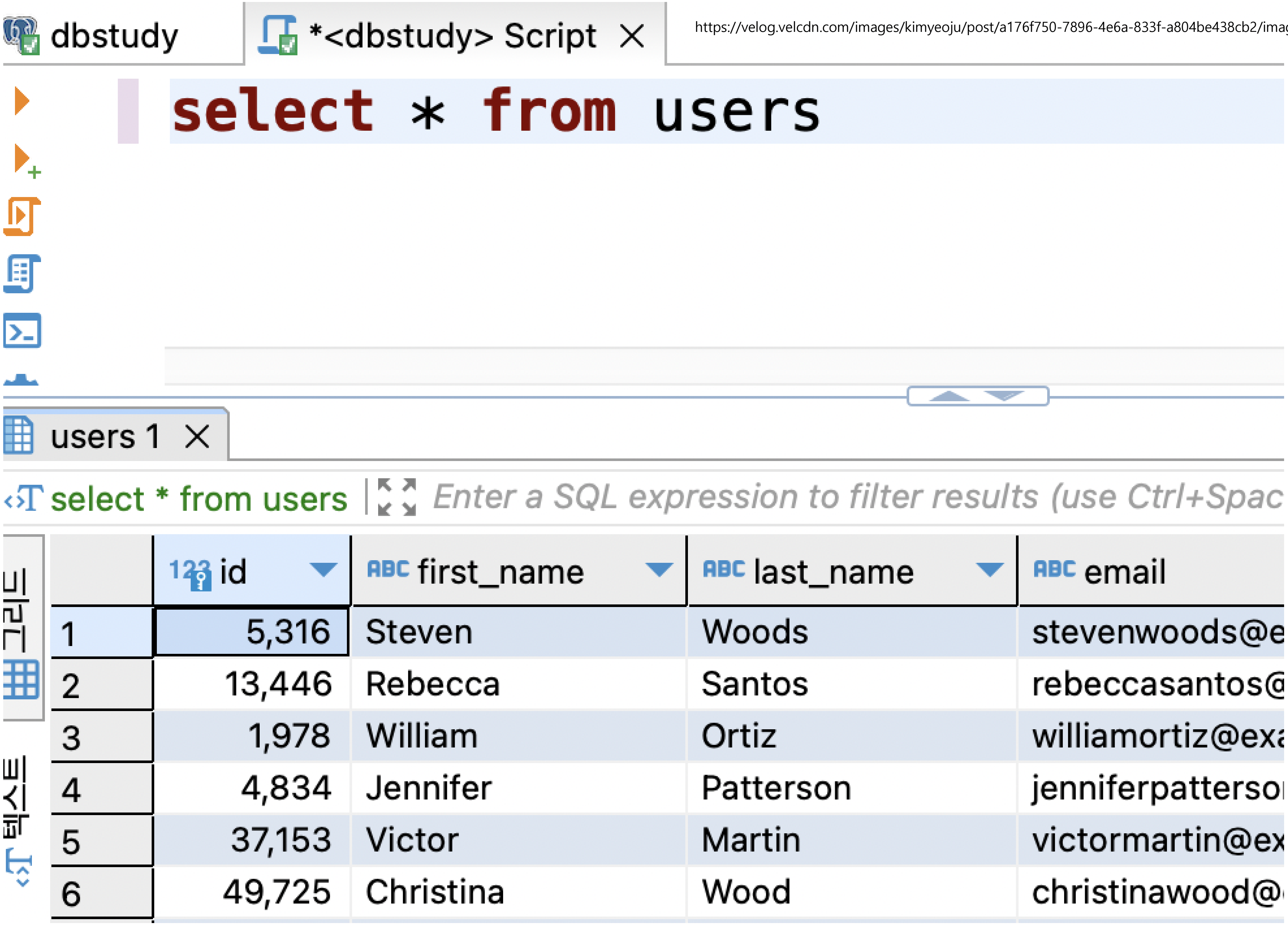
1. Difference in Storage Methods – Database vs. Cloud

2. Level of Feature Extraction – Processed vs. Raw

- The second point reflects the **difference in surrounding values**:

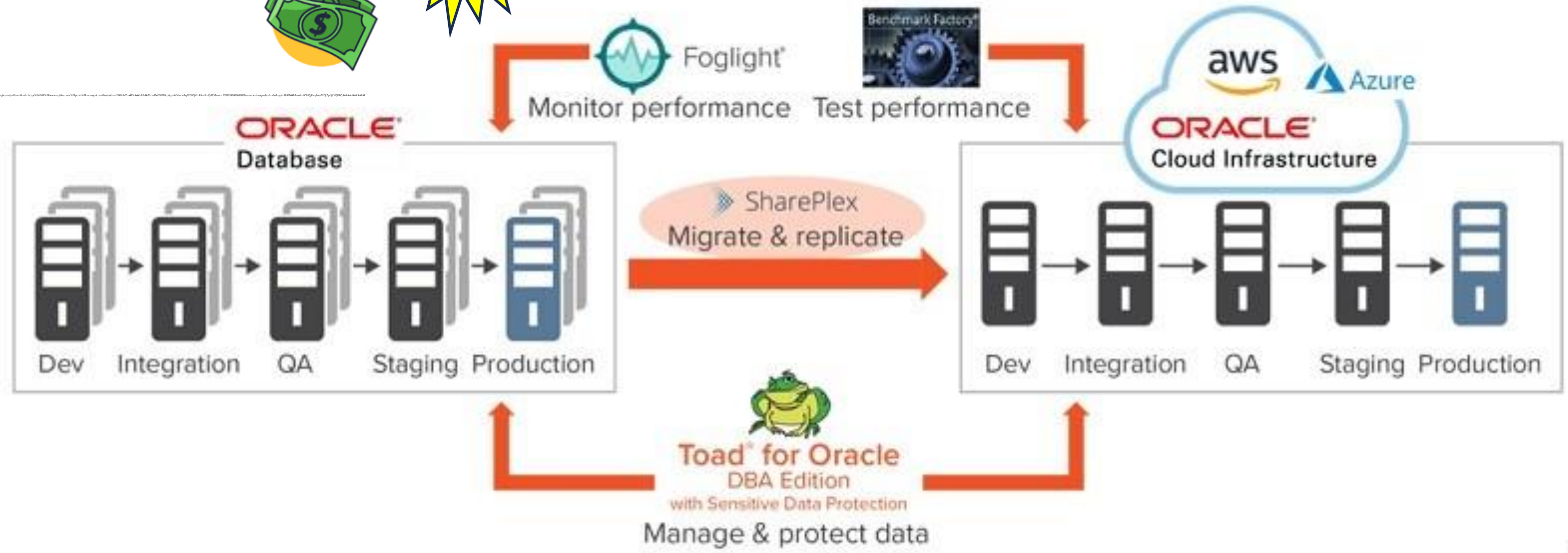
In tabular data, adjacent values are less correlated, while in image data, adjacent pixels are highly correlated (leading to methods like CNN and the concept of the receptive field).

https://velog.velcdn.com/images/kimyeoju/post/a176f750-7896-4e6a-833f-a804be438cb2/image.png



Multimodal Era

Cloud
- Google Cloud Platform (GCP)
- Amazon Web Services (AWS)
- Samsung Cloud Platform (SCP)



Meow 

고양이 (Korean), gato (Español)

Cat

Tabular data related to feed cats

① | Tabular Data

② | SVM

③ | AutoML

④ | Hyperparameter Tuning

Multimodal Era

Image data: CNN, Vision Transformer

Meow 

Audio data: Wav2vec 2.0, AASIST

고양이 (Korean), gato (Español)

Text data (Korean): KoBART, KoBERT / Español

Cat

Text data (English): BERT, GPT, ELECTRA

Tabular data related to feed cats

?????

Machine Learning for Tabular data

- XGBoost (CNN of machine learning), LightGBM
- Random Forest

- **Deep Learning approaches**

- TabNet, STUNT, Trompt, …
- Does not work well, though … (overparametrized, nbhd values, …)

-> ?????

Support Vector Machine [SVM]

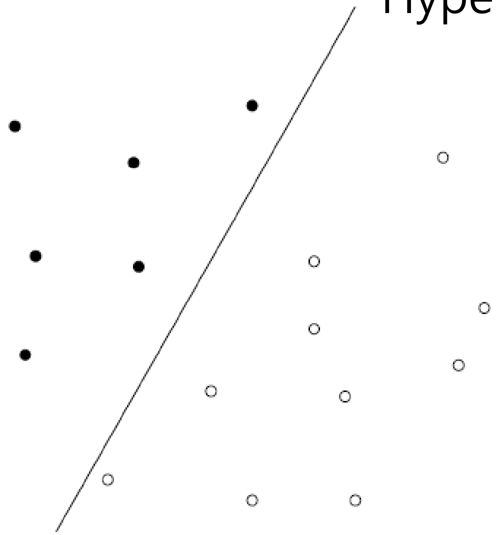
- Finding the optimal hyperplane
- Binary Classification

Linear classifier

separate two sets of points $\{x_1, \ldots, x_N\}$, $\{y_1, \ldots, y_M\}$ by a hyperplane:

$$a^T x_i + b > 0, \quad i = 1, \ldots, N, \qquad a^T y_i + b < 0, \quad i = 1, \ldots, M$$

Hyperplane, linear classifier



homogeneous in a, b, hence equivalent to

For an arbitrarily small $\varepsilon > 0$,

$$a^T x_i + b \geq \overset{\varepsilon}{1}, \quad i = 1, \ldots, N, \qquad a^T y_i + b \leq \overset{-\varepsilon}{-1}, \quad i = 1, \ldots, M$$

a set of linear inequalities in a, b

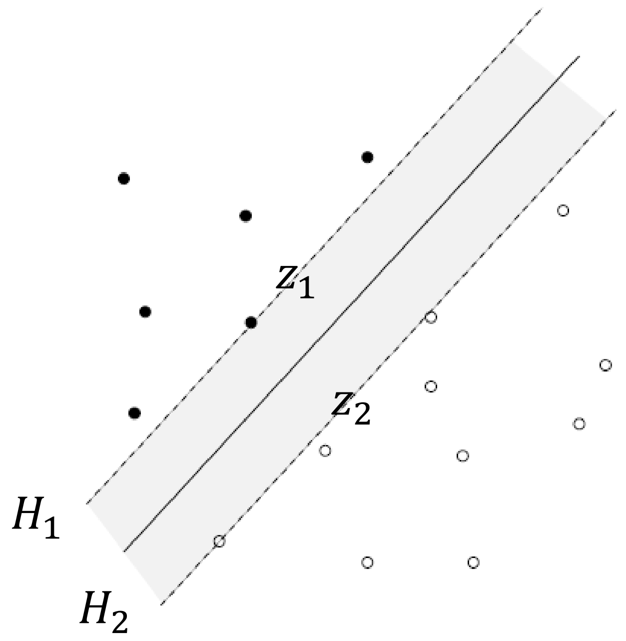
Robust linear classifier

(Euclidean) distance between hyperplanes

$$\mathcal{H}_1 = \{z \mid a^T z + b = 1\}$$
$$\mathcal{H}_2 = \{z \mid a^T z + b = -1\}$$

Maximize the distance btw z_1 and z_2

is $\mathbf{dist}(\mathcal{H}_1, \mathcal{H}_2) = 2/\|a\|_2$



to separate two sets of points by maximum margin,

$$\begin{array}{ll} \text{minimize} & (1/2)\|a\|_2 \\ \text{subject to} & a^T x_i + b \geq 1, \quad i = 1, \ldots, N \\ & a^T y_i + b \leq -1, \quad i = 1, \ldots, M \end{array} \tag{1}$$

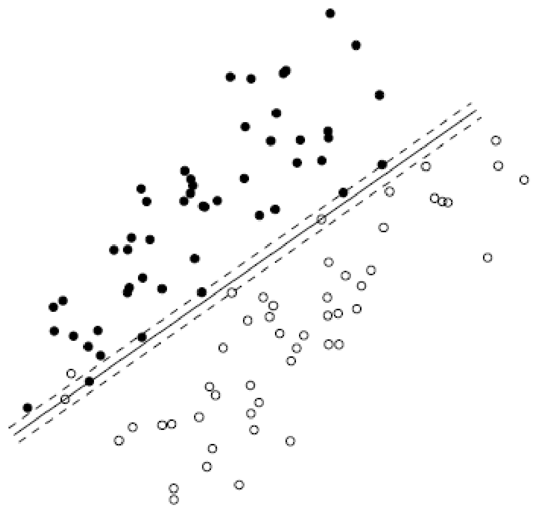
Minimize its reciprocal

(after squaring objective) a QP in a, b

Approximate linear classifier

$$
\begin{array}{ll}
\text{minimize} & \mathbf{1}^T u + \mathbf{1}^T v \\
\text{subject to} & a^T x_i + b \geq 1 - u_i, \quad i = 1, \ldots, N \\
& a^T y_i + b \leq -1 + v_i, \quad i = 1, \ldots, M \\
& u \succeq 0, \quad v \succeq 0
\end{array}
$$

- an LP in a, b, u, v
- at optimum, $u_i = \max\{0, 1 - a^T x_i - b\}$, $v_i = \max\{0, 1 + a^T y_i + b\}$
- can be interpreted as a heuristic for minimizing #misclassified points



- minimizing the number of misclassification is hard, but this approach is good approximation

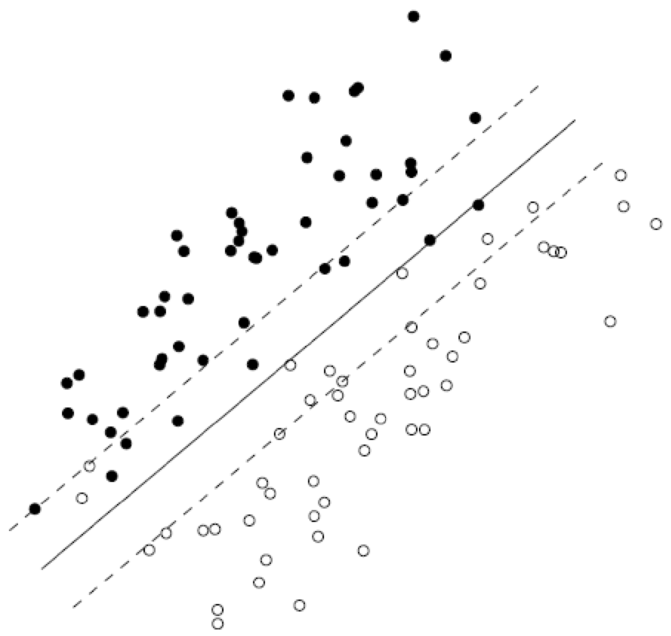
Support vector machine (SVM)

$$\begin{array}{ll} \text{minimize} & \|a\|_2 + \gamma(\mathbf{1}^T u + \mathbf{1}^T v) \\ \text{subject to} & a^T x_i + b \geq 1 - u_i, \quad i = 1, \ldots, N \\ & a^T y_i + b \leq -1 + v_i, \quad i = 1, \ldots, M \\ & u \succeq 0, \quad v \succeq 0 \end{array}$$

γ: *hyperparameter*

produces point on trade-off curve between inverse of margin $2/\|a\|_2$ and classification error, measured by total slack $\mathbf{1}^T u + \mathbf{1}^T v$

same example as previous page,
with $\gamma = 0.1$:



Support Vector Machine [SVM]

- Multi-class classification
- OvO [One vs. One]
- OvR [One vs. Rest]

e.g.) cat vs. dog vs. fig

OvO: cat – dog | cat – fig | dog – fig

OvR: cat – {dog, fig} | dog – {cat, fig} | fig – {cat, dog}

① | Tabular Data

② | SVM

③ | AutoML

④ | Hyperparameter Tuning

AutoML

AutoML-pycaret

- An open-source library that automates the machine learning workflow with simple code, encompassing coding, preprocessing, model selection, and parameter tuning all in one!

- **But… How to Improve Tuning?**

① | Tabular Data

② | SVM

③ | AutoML

④ | Hyperparameter Tuning

Hyperparameter Tuning

A modern nightmare of data scientists…

(1) Grid Search – computes all possible cases

https://dacon.io/codeshare/4646

```
from sklearn.model_selection import GridSearchCV

grid_search = {'criterion': ['entropy', 'gini'],
              'max_depth': [2],
              'max_features': ['auto', 'sqrt'],
              'min_samples_leaf': [4, 6, 8],
              'min_samples_split': [5, 7,10],
              'n_estimators': [20]}

clf = RandomForestClassifier()
model = GridSearchCV(estimator = clf, param_grid = grid_search, cv = 4, verbose= 5, n_jobs = -1)
model.fit(X_Train,Y_Train)
```

Hyperparameter Tuning

(2) Random Search – randomly selects cases

https://dacon.io/codeshare/4646

```
from sklearn.model_selection import RandomizedSearchCV

random_search = {'criterion': ['entropy', 'gini'],
               'max_depth': [2],
               'max_features': ['auto', 'sqrt'],
               'min_samples_leaf': [4, 6, 8],
               'min_samples_split': [5, 7,10],
               'n_estimators': [20]}

clf = RandomForestClassifier()
model = RandomizedSearchCV(estimator = clf, param_distributions = random_search, n_iter = 10,
                          cv = 4, verbose= 1, random_state= 101, n_jobs = -1)
model.fit(X_Train,Y_Train)
```

Hyperparameter Tuning

(3) HyperOpt

https://dacon.io/codeshare/4646

```
from hyperopt import hp, fmin, tpe, STATUS_OK, Trials

space = {'criterion': hp.choice('criterion', ['entropy', 'gini']),
        'max_depth': hp.quniform('max_depth', 10, 12, 10),
        'max_features':  hp.choice('max_features',  ['auto',  'sqrt',  'log2',
None]),
        'min_samples_leaf': hp.uniform ('min_samples_leaf', 0, 0.5),
        'min_samples_split' : hp.uniform ('min_samples_split', 0, 1),
        'n_estimators' : hp.choice('n_estimators', [10, 50])
}

def objective(space):
    hopt = RandomForestClassifier(criterion = space['criterion'],
                          max_depth = space['max_depth'],
                          max_features = space['max_features'],
                          min_samples_leaf = space['min_samples_leaf'],
                          min_samples_split = space['min_samples_split'],
                          n_estimators = space['n_estimators'],
    )
```

```
    accuracy = cross_val_score(hopt, X_Train, Y_Train, cv = 4).mean()
    return {'loss': -accuracy, 'status': STATUS_OK }

trials = Trials()
best = fmin(fn= objective,
            space= space,
            algo= tpe.suggest,
            max_evals = 20,
            trials= trials
)

# optimal solution
best
```

HyperOpt

https://towardsdatascience.com/hyperopt-demystified-3e14006eb6fa

- Tree-based Parzen Esimators (TPE) – optimizes a user-defined objective function

1. Train a model with several sets of randomly-selected hyperparameters, returning objective function values.
2. Split our observed objective function values into "good" and "bad" groups, according to some threshold gamma (γ).
3. Calculate the "promisingness" score, which is just *P(x|good) / P(x|bad)*.
4. Determine the hyperparameters that maximize promisingness score via mixture models.
5. Fit our model using the hyperparameters from step 4.
6. Repeat steps 2–5 until a stopping criteria.

HyperOpt

https://towardsdatascience.com/hyperopt-demystified-3e14006eb6fa

- Tree-based Parzen Estimators (TPE) ∈ Sequential Model-Based Optimization (SMBO) algorithm

```
from hyperopt import hp, fmin, tpe, STATUS_OK, Trials

space = {'criterion': hp.choice('criterion', ['entropy', 'gini']),
       'max_depth': hp.quniform('max_depth', 10, 12, 10),
       'max_features':  hp.choice('max_features',  ['auto',  'sqrt',  'log2', None]),
       'min_samples_leaf': hp.uniform ('min_samples_leaf', 0, 0.5),
       'min_samples_split' : hp.uniform ('min_samples_split', 0, 1),
       'n_estimators' : hp.choice('n_estimators', [10, 50])
}

def objective(space):
    hopt = RandomForestClassifier(criterion = space['criterion'],
                                  max_depth = space['max_depth'],
                                  max_features = space['max_features'],
                                  min_samples_leaf = space['min_samples_leaf'],
                                  min_samples_split = space['min_samples_split'],
                                  n_estimators = space['n_estimators'],
    )
```

constraints

objective function (Accuracy, AUC, RMSE, …)

HyperOpt

https://towardsdatascience.com/hyperopt-demystified-3e14006eb6fa

- Bayesian optimization: a sequential algorithm that finds points in hyperspace with a high probability of being “successful” according to an objective function
- Clever tricks? modeling P(x|y) instead of P(y|x):

$$p(x|y) = \frac{p(y|x)*p(x)}{p(y)}$$

- probability of an objective function value (y), given hyperparameters (x) -> b.o.
- **P(x|y)** :

=> TPE tries to find the best objective function values, then determine the associated hyperparameters

HyperOpt

https://towardsdatascience.com/hyperopt-demystified-3e14006eb6fa

- TPE splits our observed data points into two groups

(1) g(x): good

(2) l(x): bad

$$p(x|y) = \begin{cases} \ell(x) & \text{if } y < y^* \text{ Bad Objective Function Vals} \\ g(x) & \text{if } y \geq y^* \text{ Good Objective Function Vals} \end{cases}$$

HyperOpt

https://towardsdatascience.com/hyperopt-demystified-3e14006eb6fa

- “Promisingness” score

$$P = \frac{g(x)}{l(x)} = \frac{P(x|good)}{P(x|bad)}$$

- Numerator: the probability of observing a set of hyperparameters (x), given the corresponding objective function value is “good.”
- Denominator: the probability of observing a set of hyperparameters (x), given the corresponding objective function value is “bad.”

=> The bigger the “promisingness” value, the more likely that the hyperparameters x will produce a “good” objective function

HyperOpt

https://towardsdatascience.com/hyperopt-demystified-3e14006eb6fa

- “Promisingness” score

$$P = \frac{g(x)}{l(x)} = \frac{P(x|good)}{P(x|bad)}$$

- Numerator: the probability of observing a set of hyperparameters (x), given the corresponding objective function value is “good.”
- Denominator: the probability of observing a set of hyperparameters (x), given the corresponding objective function value is “bad.”

=> The bigger the “promisingness” value, the more likely that the hyperparameters x will produce a “good” objective function

HyperOpt

https://towardsdatascience.com/hyperopt-demystified-3e14006eb6fa

- Bayesian hyperparameter optimization
- develop a probabilistic distribution of the hyperparameter search space, using an acquisition function, such as expected improvement, to transform the hyperspace to be more “searchable.”
- Then leverages optimization algorithm, such as stochastic gradient descent, to find a the hyperparameters that maximize our acquisition function.

A function that suggests what values to try next based on the current probabilistic estimates of the objective function.

=> “Promisingness” score acts as an <u>acquisition function</u> and is proportional to the Expected Improvement (EI)

HyperOpt

https://towardsdatascience.com/hyperopt-demystified-3e14006eb6fa

- "Promisingness" score is measured by the mixture model
- take multiple probability distributions and put them together using a linear combination
- Generally

* TPE also support categorical variables which traditional Bayesian optimization does not.

(1) Distribution types: categorical -> re-weighted categorical distribution / numeric-> Gaussian (i.e. normal) or uniform distribution.

(2) Iterate over each point and insert a distribution at that point.

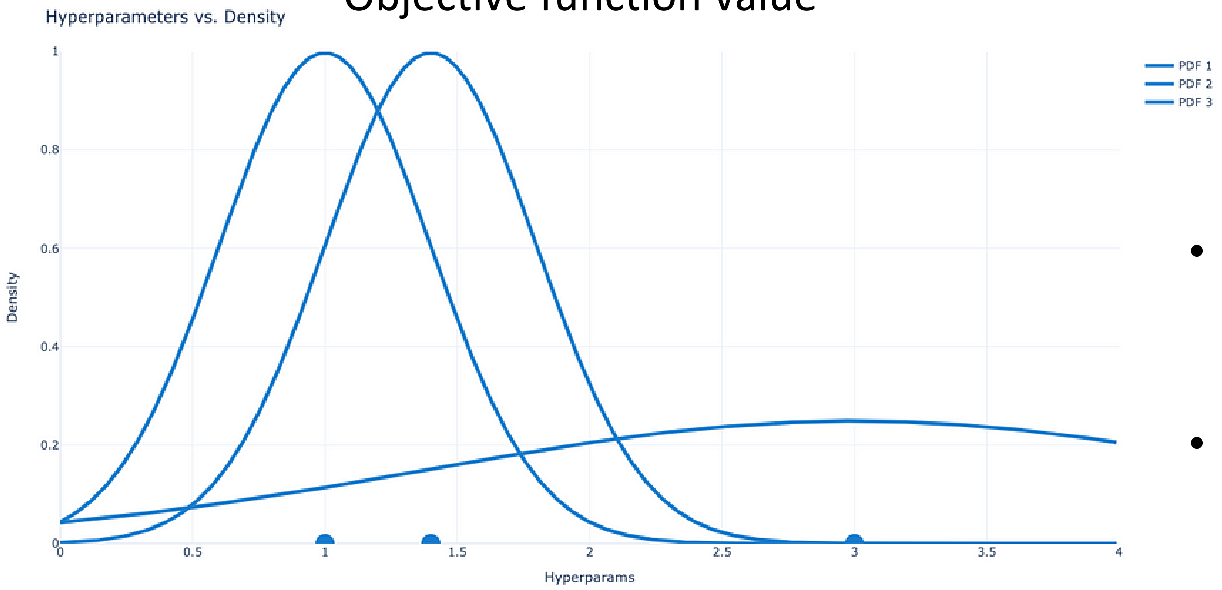
(3) Sum the mass of all distributions to get a probability density estimate.

- Note that this process is run individually for both sets l(x) and g(x).

HyperOpt

https://towardsdatascience.com/hyperopt-demystified-3e14006eb6fa

Objective function value



X

- g(x) = y
- x: hyperparameters, y: objective function value
- **σ**: the standard deviation
- The distance to the closest neighboring point

- **If points are close together -> the standard deviation will be small -> the distribution will be very tall**
- **if points are spread apart, the distribution will be flat**

* Truncated Gaussian

HyperOpt

https://towardsdatascience.com/hyperopt-demystified-3e14006eb6fa

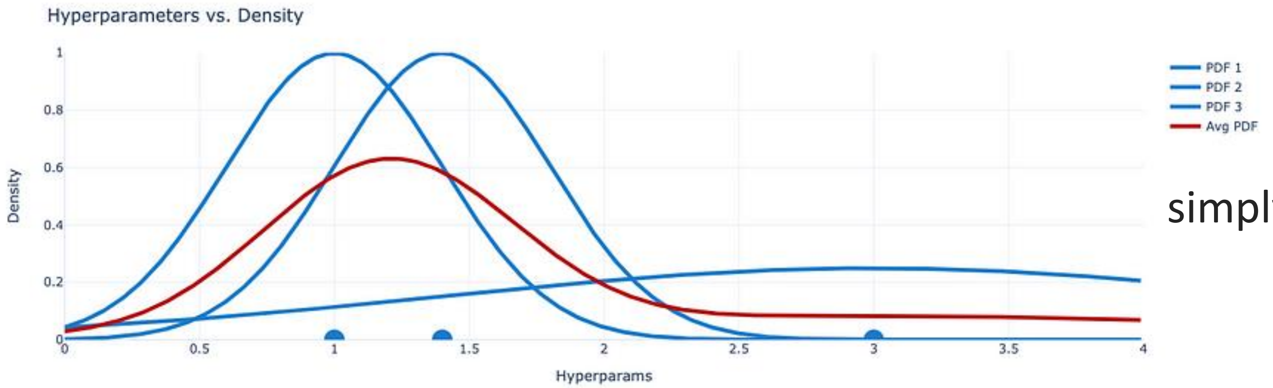
- Finally, let’s determine the next point to explore! – acquisition function

(1) acquired objective function observations

(2) defined our “promisingness” formula

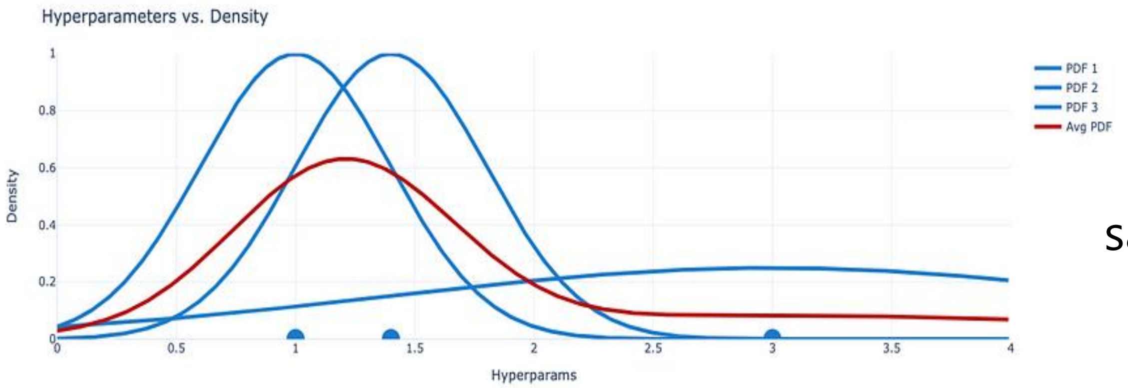
(3) created a probability density estimate via mixture models based on prior values

- average probability density function (PDF) for both *g(x)* and *l(x)*



simply the sum of all PDFs divided by the number of PDFs

HyperOpt



say, *g(x)*

- **Using the average PDF, we can get the probability of any hyperparameter value (*x*) being in *g(x)* or *l(x)*.**
- hyperparameter value (x) of 3.9 or 0.05 -> unlikely to belong to the “good” set
- 1 to 1.2 -> seems to be very likely to belong to the “good” set
- Do the same thing to *l(x)*
- Objective: maximize *g(x) / l(x)*
- **promising points should be located where *g(x)* is high, and *l(x)* is low**
- With these probability distributions, sample cases from tree-structured hyperparameters (TPE..!!) and find the set of hyperparameters that maximize “promisingness”

https://towardsdatascience.com/hyperopt-demystified-3e14006eb6fa

HyperOpt

- Algorithms for Hyper-Parameter Optimization
- https://papers.nips.cc/paper_files/paper/2011/hash/86e8f7ab32cfd12577bc2619bc635690-Abstract.html

Hyperparameter Tuning

Grid Search, Random Search, HyperOpt -> “methods” for hyperparameter tuning

(4) Optuna – computes all possible cases within a given range -> works like AutoML

https://dacon.io/codeshare/4646

```
def objective(trial):
    iris = sklearn.datasets.load_iris()
    x, y = iris.data, iris.target

    classifier_name = trial.suggest_categorical('classifier', ['SVC', 'RandomForest'])

    if classifier_name == 'SVC':
        svc_c = trial.suggest_loguniform('svc_c', 1e-10, 1e10)
        classifier_obj = sklearn.svm.SVC(C=svc_c, gamma='auto')
    else:
        rf_max_depth = int(trial.suggest_loguniform('rf_max_depth', 2, 32))
        classifier_obj = sklearn.ensemble.RandomForestClassifier(max_depth=rf_max_depth, n_estimators=10)

    accuracy = cross_val_score(classifier_obj, x, y, cv = 4).mean()
    return accuracy

study = optuna.create_study(direction='maximize')
study.optimize(objective, n_trials=100)
print(study.best_trial.params)
```

Optuna

https://jhtobigs.oopy.io/optuna_tutorial

SOTA Method

- **Study-Trial method**
- **Study**: A session for optimizing the objective function, composed of multiple trials.
- **Default**: TPE (Tree-structured Parzen Estimator) from HyperOpt; other methods include Random Search and Grid Search.
- The best method is selected based on results.
- **Pruning**: Early termination of trials that yield poor results.