Supervised Learning

Peerapon Vateekul, Ph.D.

peerapon.v@4amconsult.com

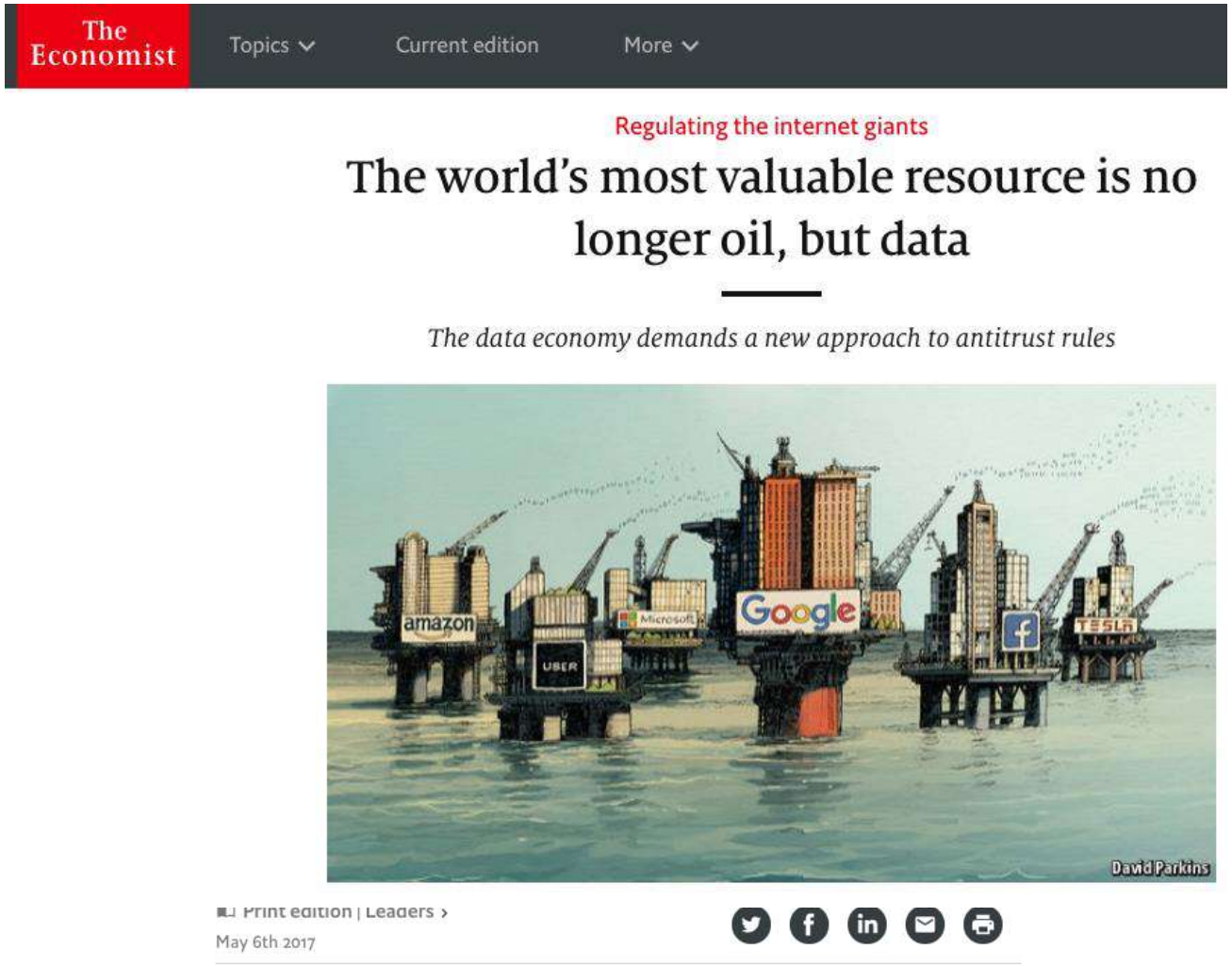
Outlines

- Introduction to Machine Learning
- What is Supervised Learning ?
- Decision Tree
- Regression & Logistic Regression
- Neural Network
- K-Nearest Neighbors
- Support Vector Machine
- Use cases

+

Introduction to Machine Learning

Data is important (in 2017)!



The Economist Topics Current edition More

Regulating the internet giants

The world's most valuable resource is no longer oil, but data

The data economy demands a new approach to antitrust rules

Print edition | Leaders ›

May 6th 2017

- Alphabet (Google's parent company), Amazon, Apple, Facebook and Microsoft
- $25bn in net profit in the first quarter of 2017
- Amazon captures half of all dollars spent online in America.
- Google and Facebook accounted for almost all the revenue growth in digital advertising in America last year

https://www.economist.com/leaders/2017/05/06/the-worlds-most-valuable-resource-is-no-longer-oil-but-data

Data is important (in 2018)! (cont.)

The New Oil

Jennifer Presley Executive Editor, E&P Magazine Hart Energy Thursday, November 1, 2018 - 6:40am

With a number of successful projects under its collective belt, the oil and gas industry is proving Big Data is more than just a buzzword. (Source: Makhnach_S/*Shutterstock.com*; Design by Felicia Hammons)

https://www.epmag.com/new-oil-1720651

Data Science (AI,ML,DM)

The Top 10 Tech Trends In 2023 Everyone Must Be Ready For

Bernard Marr Contributor ⓘ

Follow

3
0

Nov 21, 2022, 01:39am EST

- 1. AI Everywhere
- 2. Parts of the Metaverse Will Become Real
- 3. Progress in Web3
- 4. Bridging the Digital and Physical World
- 5. Increasingly Editable Nature
- 6. Quantum Progress
- 7. Progress in Green Technology
- 8. Robots Will Become More Human
- 9. Progress in Autonomous System
- 10. More Sustainable Technology

https://www.forbes.com/sites/bernardmarr/2022/11/21/the-top-10-tech-trends-in-2023-everyone-must-be-ready-for/?sh=5fb222ad7df0

Top Strategic Technology Trends for 2021

People Centricity	Location Independence	Resilient Delivery
Internet of Behaviors	Distributed Cloud	Intelligent Composable Business
Total Experience	Anywhere Operations	AI Engineering
Privacy-Enhancing Computation	Cybersecurity Mesh	Hyperautomation

Combinatorial Innovation

Gartner

https://www.techtalkthai.com/gartner-technology-trends-2021/

Gartner

Top Strategic Technology Trends for 2022

01 Data Fabric	05 Composable Applications	09 Distributed Enterprise
02 Cybersecurity Mesh	06 Decision Intelligence	10 Total Experience
03 Privacy-Enhancing Computation	07 Hyperautomation	11 Autonomic Systems
04 Cloud-Native Platforms	08 AI Engineering	12 Generative AI

Accelerating Growth | Sculpting Change | Engineering Trust

https://www.gartner.com/en/information-technology/insights/top-technology-trends

Gartner's Top Strategic Technology Trends for 2023

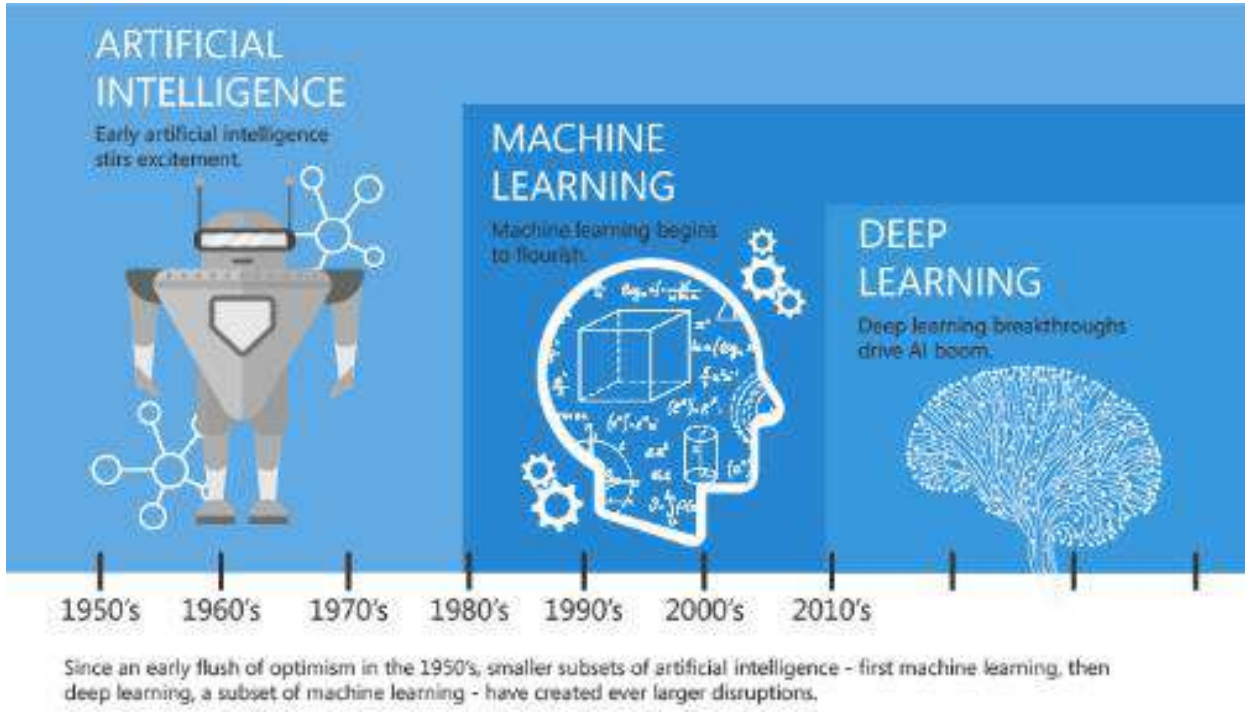
Optimize	Scale	Pioneer
Digital Immune System	Industry Cloud Platforms	Superapps
Applied Observability	Platform Engineering	Adaptive AI
AI TRiSM	Wireless-Value Realization	Metaverse
	Sustainable Technology	

https://www.pcmag.com/news/gartners-top-strategic-technology-trends-for-2023

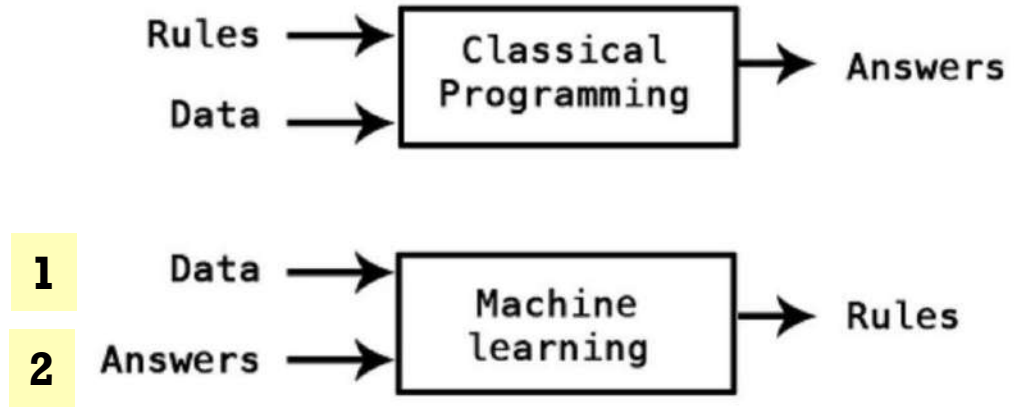
Gartner

AI = Automation

- 1) Rule-based AI
- 2) Machine Learning (ML)



Since an early flush of optimism in the 1950's, smaller subsets of artificial intelligence - first machine learning, then deep learning, a subset of machine learning - have created ever larger disruptions.



https://mc.ai/machine-learning-basics-artificial-intelligence-machine-learning-and-deep-learning/

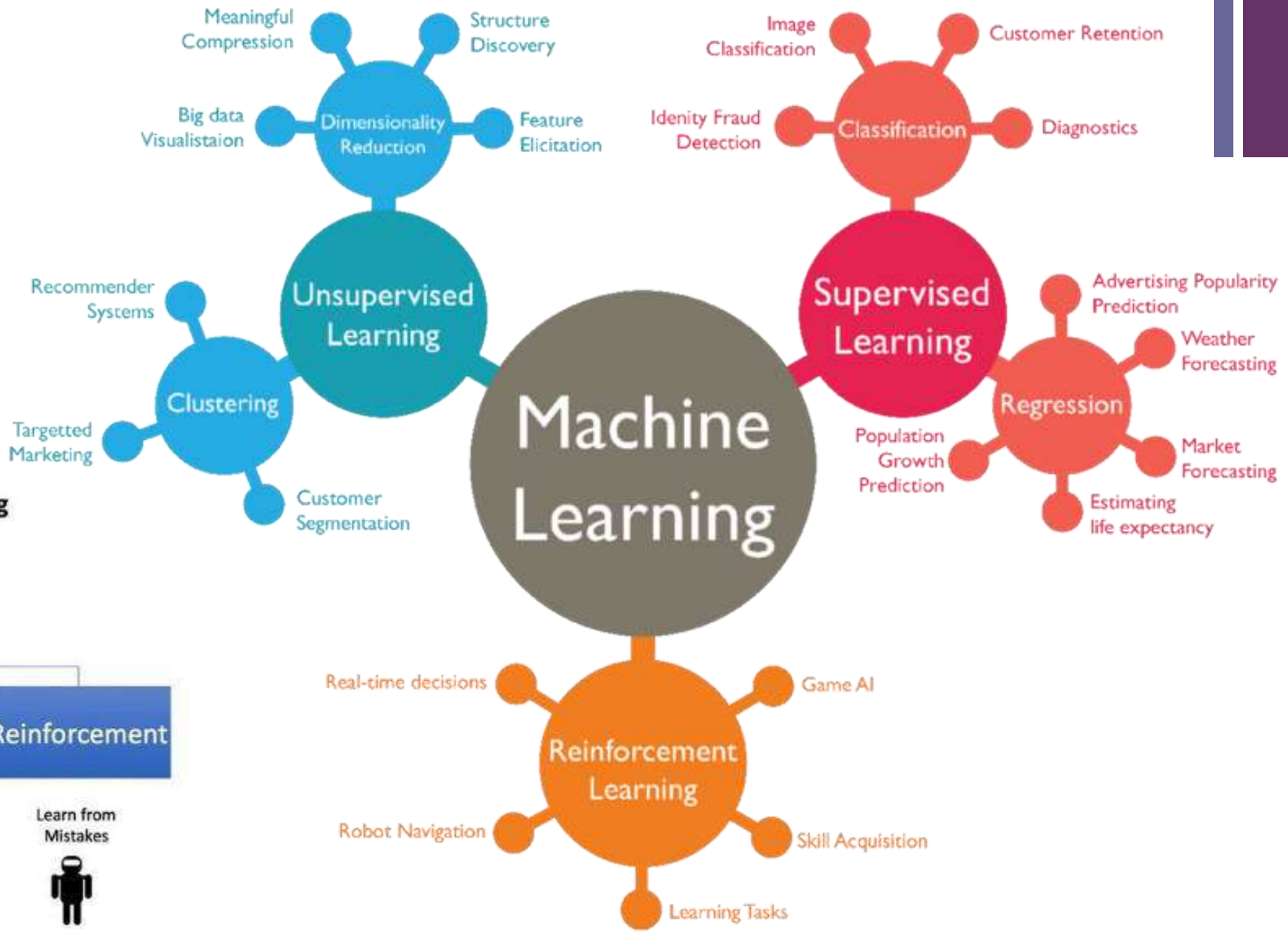
Sample of ML Application: Finding Waldo

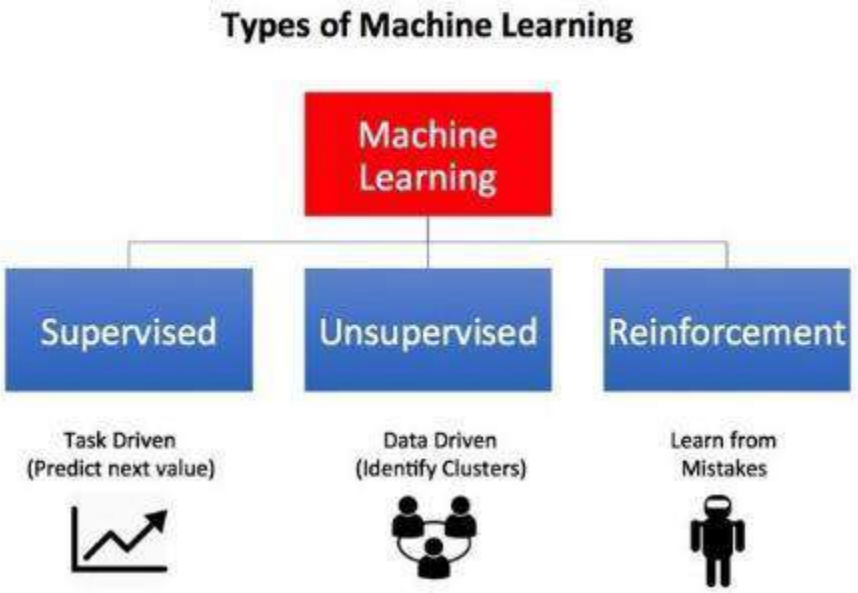
THERE'S WALDO

FINDING WALDOS WITH GOOGLE AUTOML VISION

PROTOTYPE v0.1
redpepper.land/innovation

Machine Learning (cont.)





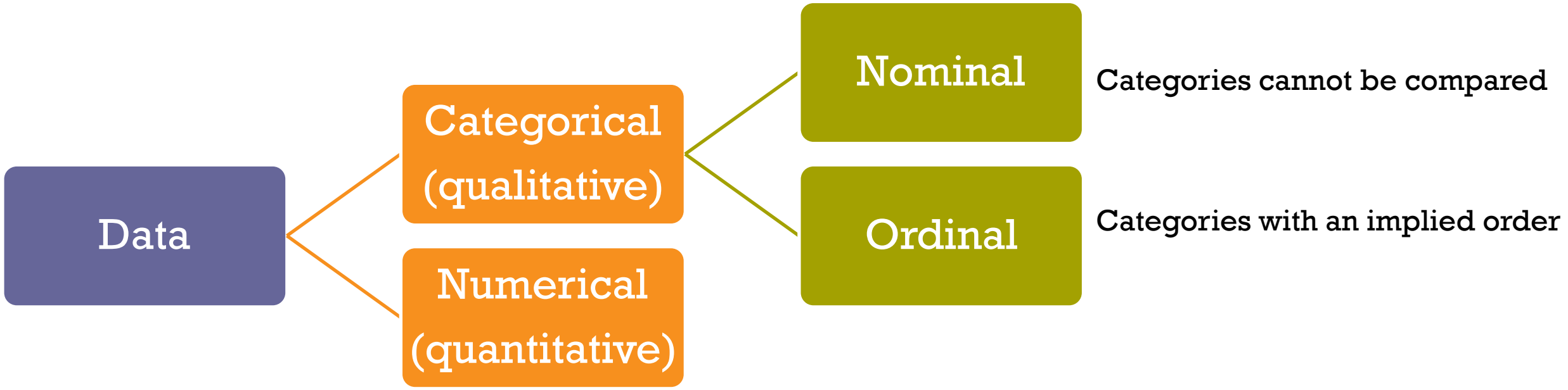
Terminology: Data table

inputs | target

Age	Income	Gender	Province	Purchase
25	25,000	Female	Bangkok	Yes
35	50,000	Female	Nontaburi	Yes
32	35,000	Male	Bangkok	No

- Row
  - Example, instance, case, observation, subject
- Column
  - Feature, variable, attribute
- Input
  - Predictor, independent, explanatory variable
- Target
  - Output, outcome, response, dependent variable

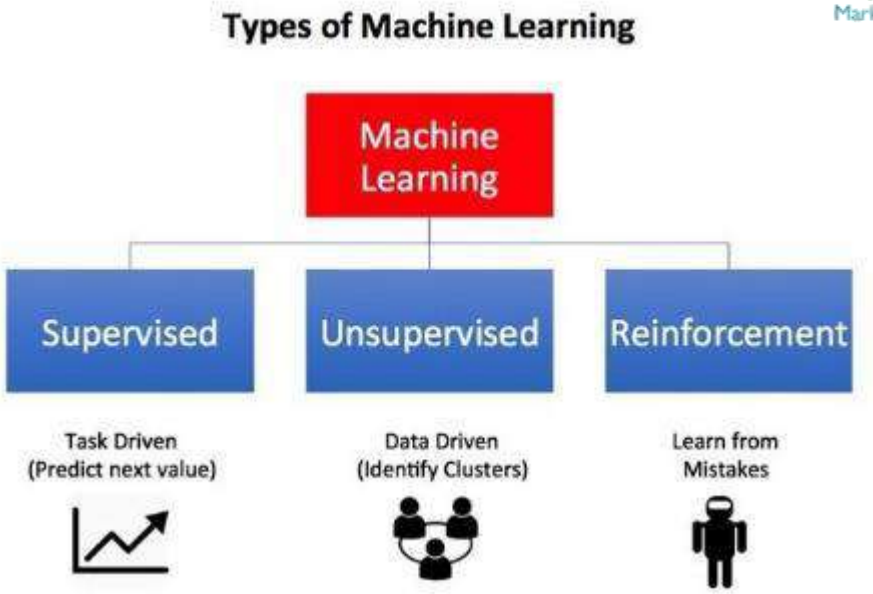
Terminology: Kinds of data

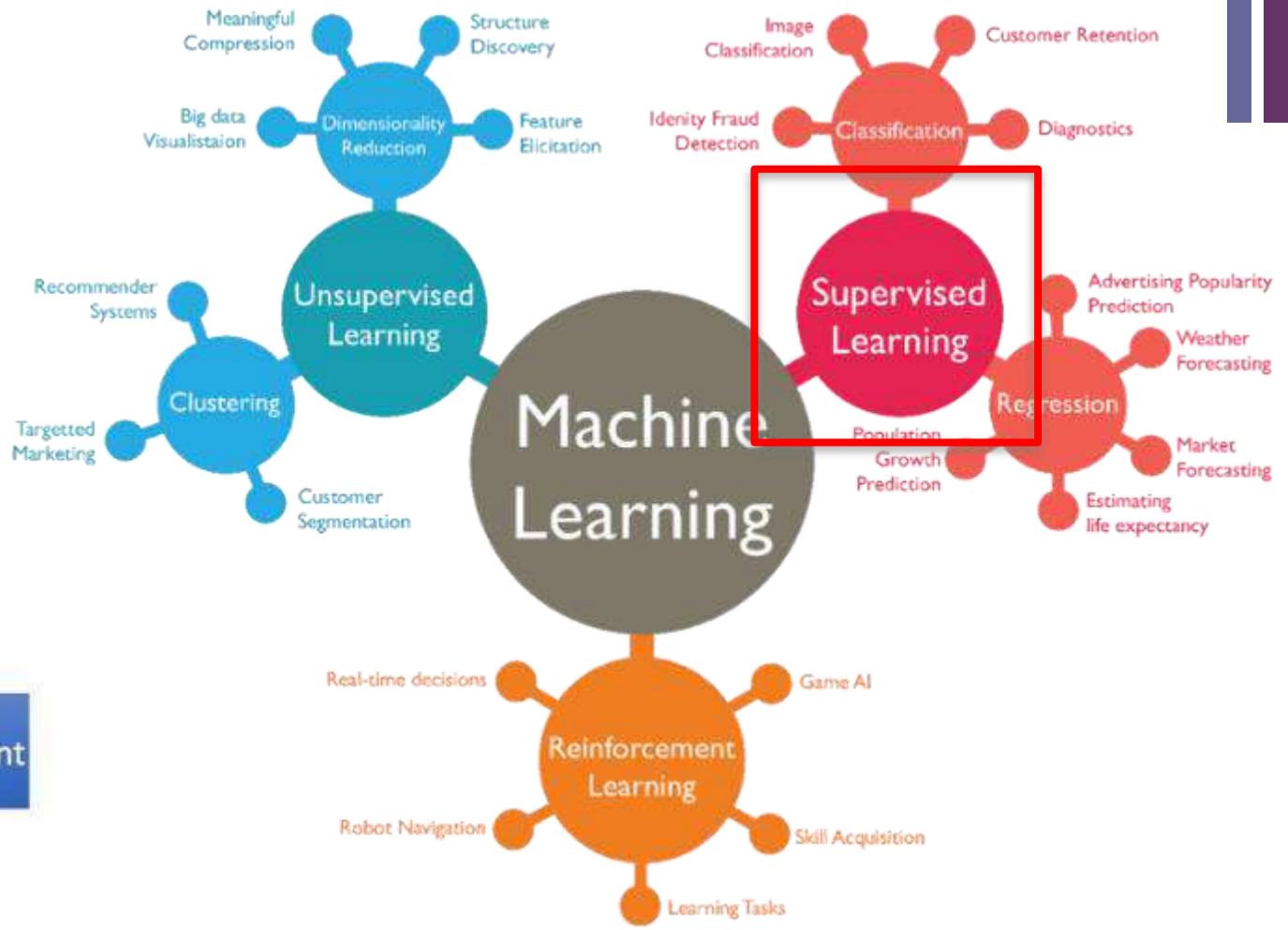


+

What is Supervised Learning ?

Types of Machine Learning





Supervised learning

Handcrafted features

Training Data

inputs | target

Age	Income	Gender	Province	Delinquency
25	25,000	Female	Bangkok	Good
35	50,000	Female	Nontaburi	Bad
32	35,000	Male	Bangkok	Bad

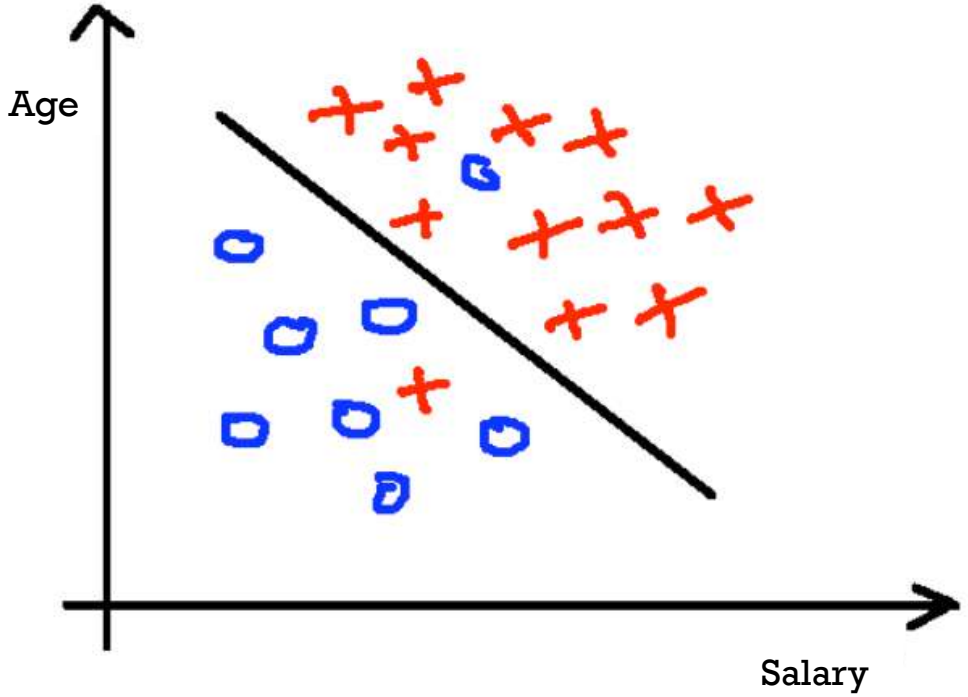
Testing Data

Age	Income	Gender	Province	Delinquency
25	25,000	Female	Bangkok	?

Application: Credit Scoring

Classification: predicting a category

Logistic Regression

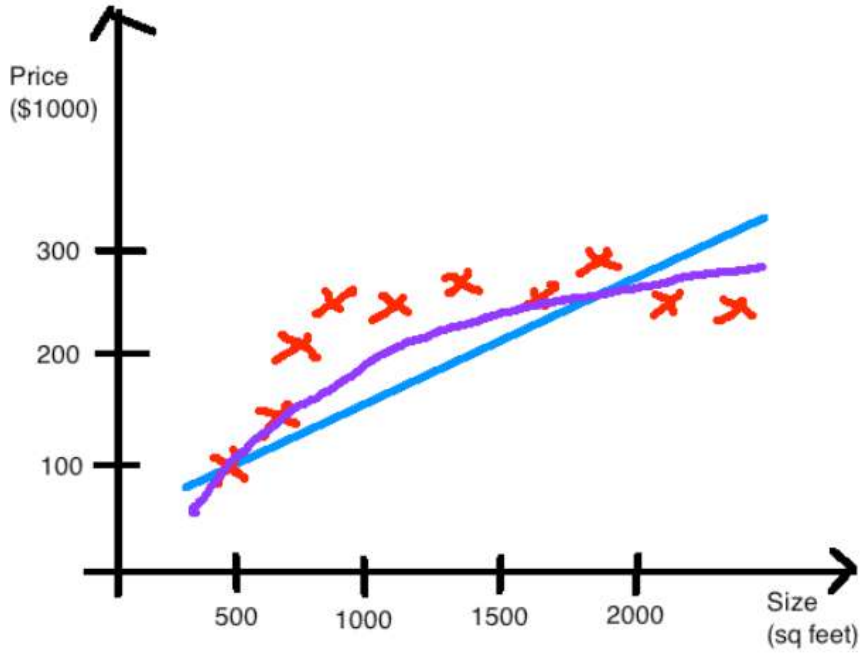


Predict targeted customers who tend to buy our product (yes/no)

- **Some techniques:**
  - Naïve Bayes
  - Decision Tree
  - Logistic Regression
  - Support Vector Machines
  - Neural Network
  - Ensembles
- **Sample Applications**
  - Database marketing
  - Fraud detection
  - Pattern detection
  - Churn customer detection

Regression: predict a continuous value

Linear Regression



Predict a sale price of each house

- **Some techniques:**
  - Linear Regression / GLM
  - Decision Trees
  - Support vector regression
  - Neural Network
  - Ensembles
- **Sample Applications**
  - Financial risk management
  - Revenue forecasting

Scikit-learn: Machine learning library in Python

- Provides many machine learning tools with a common Estimator interface
- Built in helpers for common ML tasks (e.g., metrics, preprocessing)
- Easily combine algorithms to make a complex pipeline
- Relies heavily on numpy and scipy, often used with pandas

How do you pronounce the project name?

sy-kit learn. sci stands for science!

Why scikit?

There are multiple scikits, which are scientific toolboxes built around SciPy. You can find a list at https://scikits.appspot.com/scikits. Apart from scikit-learn, another popular one is scikit-image.

Classification

Identifying to which category an object belongs to.

Applications: Spam detection, Image recognition.
Algorithms: SVM, nearest neighbors, random forest, ... — Examples

Regression

Predicting a continuous-valued attribute associated with an object.

Applications: Drug response, Stock prices.
Algorithms: SVR, ridge regression, Lasso, ... — Examples

Clustering

Automatic grouping of similar objects into sets.

Applications: Customer segmentation, Grouping experiment outcomes
Algorithms: k-Means, spectral clustering, mean-shift, ... — Examples

Version 0.23 (2020/06/12)

Dimensionality reduction

Reducing the number of random variables to consider.

Applications: Visualization, Increased efficiency
Algorithms: PCA, feature selection, non-negative matrix factorization. — Examples

Model selection

Comparing, validating and choosing parameters and models.

Goal: Improved accuracy via parameter tuning
Modules: grid search, cross validation, metrics. — Examples

Preprocessing

Feature extraction and normalization.

Application: Transforming input data such as text for use with machine learning algorithms.
Modules: preprocessing, feature extraction. — Examples

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

Estimator Interface

Decision Trees

We'll start just by training a single decision tree.

```
In [8]: from sklearn.tree import DecisionTreeClassifier

In [9]: dtree = DecisionTreeClassifier(min_samples_leaf=10, criterion='entropy')

In [10]: dtree.fit(X_train,y_train)

Out[10]: DecisionTreeClassifier(class_weight=None, criterion='entropy', max_depth=None,
                                max_features=None, max_leaf_nodes=None,
                                min_impurity_decrease=0.0, min_impurity_split=None,
                                min_samples_leaf=10, min_samples_split=2,
                                min_weight_fraction_leaf=0.0, presort=False, random_state=None,
                                splitter='best')
```

Prediction and Evaluation

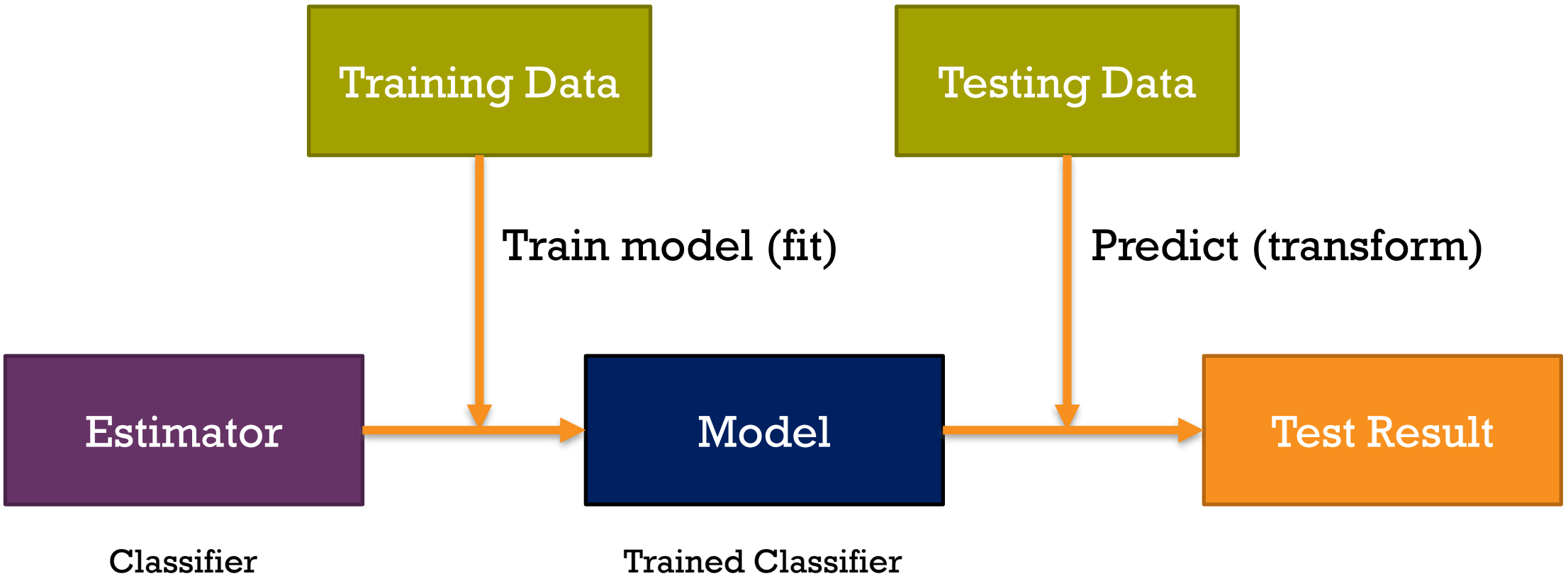
Let's evaluate our decision tree.

```
[11]: predictions = dtree.predict(X_test)

[12]: from sklearn.metrics import classification_report,confusion_matrix

[13]: print(classification_report(y_test,predictions))
```

	precision	recall	f1-score	support
absent	0.85	0.85	0.85	20
present	0.40	0.40	0.40	5
avg / total	0.76	0.76	0.76	25



Training Phase

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split

cancer = load_breast_cancer()      # Get some data
X_train, X_test, y_train, y_test = train_test_split(
        cancer.data, cancer.target,
        stratify=cancer.target, random_state=1337)

tree = DecisionTreeClassifier(random_state=7331)
tree.fit(X_train, y_train)  # Learn a Decision Function
```

Ref: C. Travis Johnson, Mines Linux Users Group on April 27, 2017

Testing Phase (Prediction)

```
y_pred = tree.predict(X_test)
```

```
>>> from sklearn.metrics import classification_report
>>> y_true = [0, 1, 2, 2, 2]
>>> y_pred = [0, 0, 2, 2, 1]
>>> target_names = ['class 0', 'class 1', 'class 2']
>>> print(classification_report(y_true, y_pred, target_names=target_names))
              precision    recall  f1-score   support

     class 0       0.50      1.00      0.67         1
     class 1       0.00      0.00      0.00         1
     class 2       1.00      0.67      0.80         3

   micro avg       0.60      0.60      0.60         5
   macro avg       0.50      0.56      0.49         5
weighted avg       0.70      0.60      0.61         5
```

Prediction Algorithms

- Decision Tree
- (Logistic) Regression
- Neural Networks (NN)
- K-Nearest Neighbors
- Support Vector Machine
- Naïve Bayes

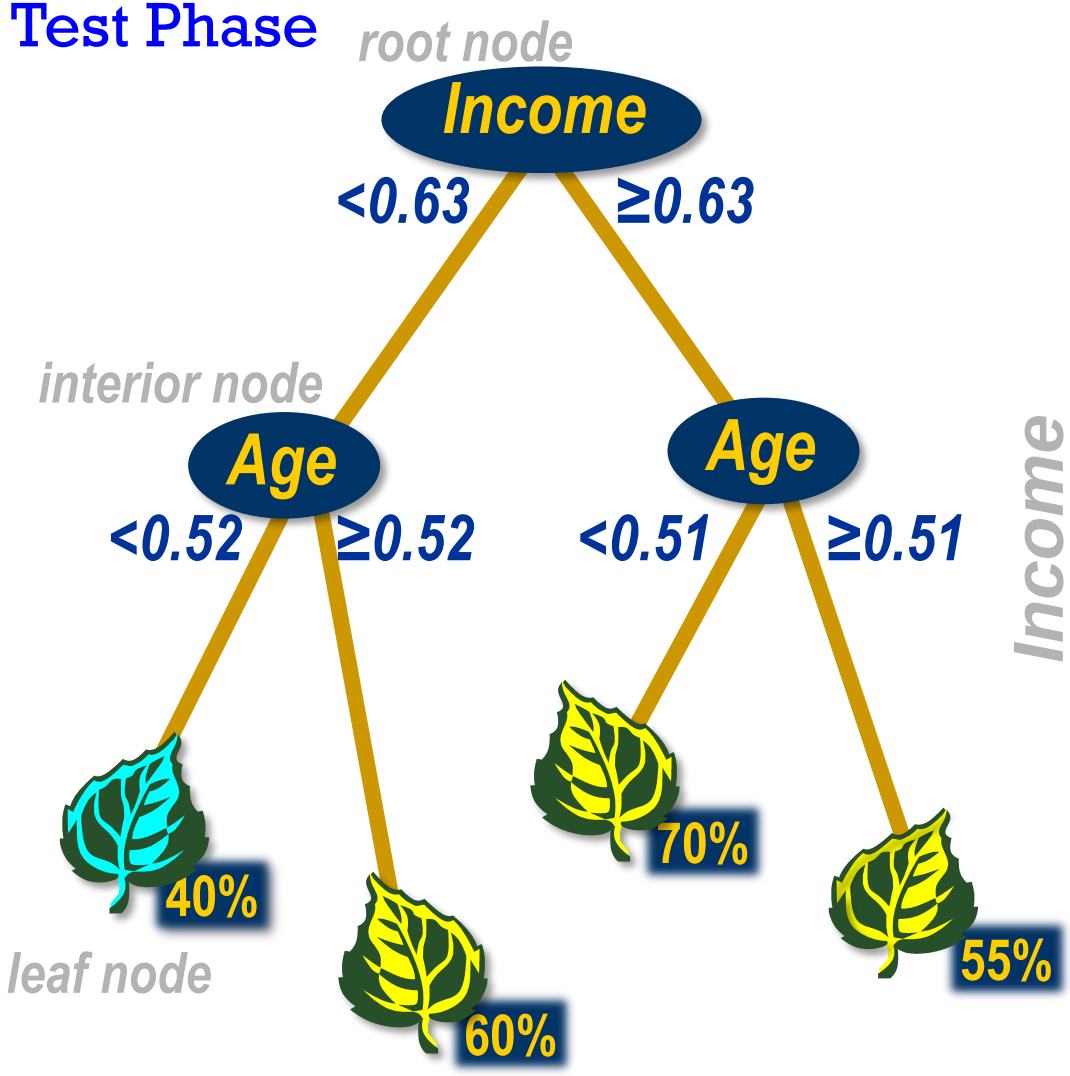
Decision Tree

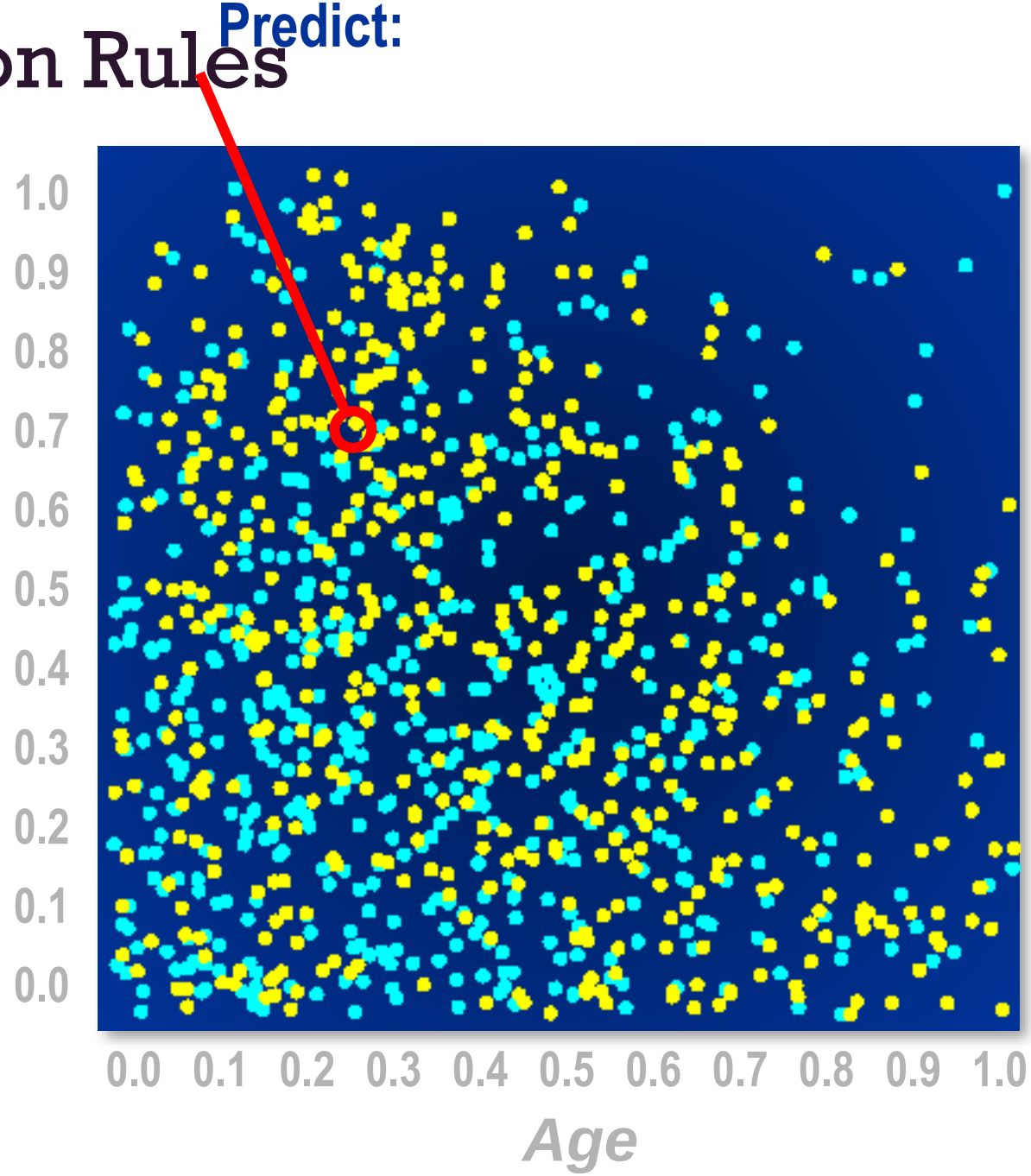
Classification and Regression with

- Decision Tree
- Random Forest
- XGBoost

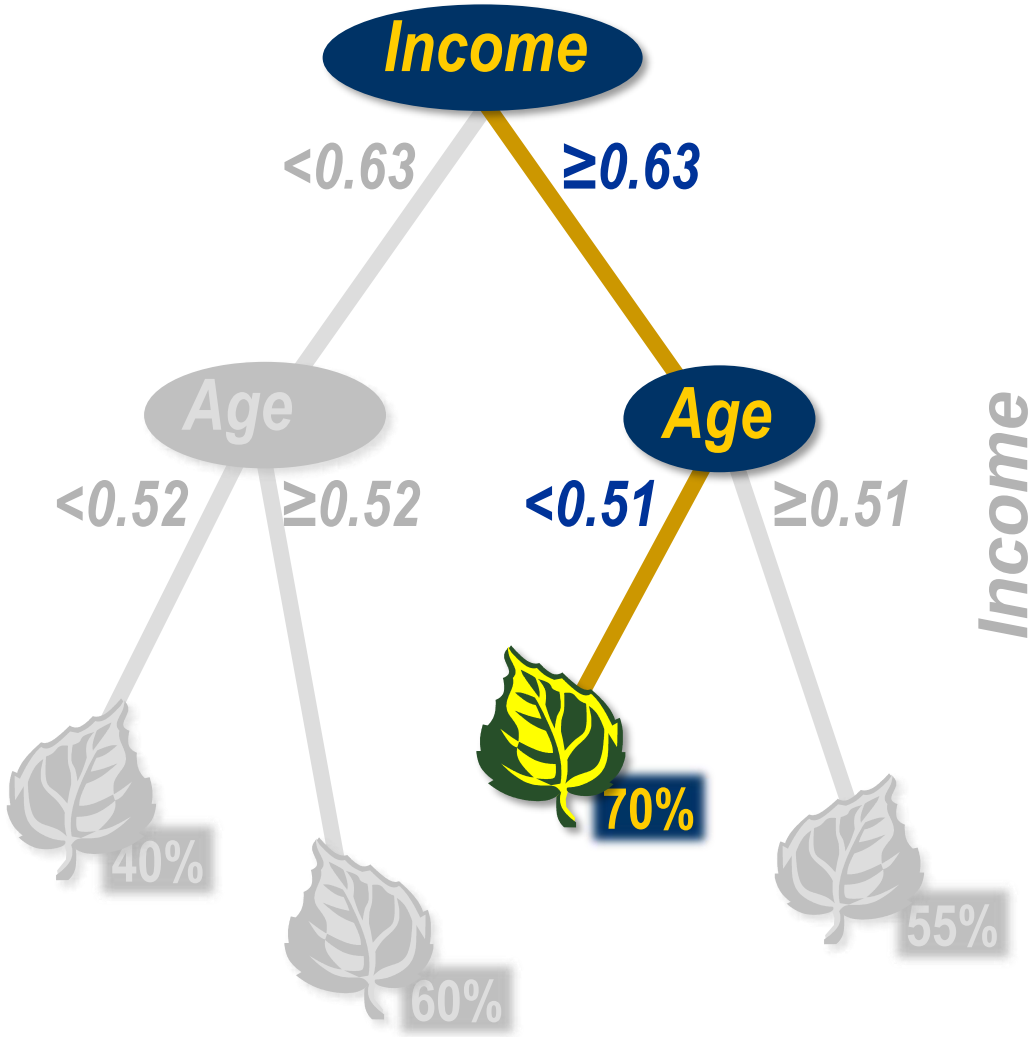
1) Decision Tree Prediction Rules

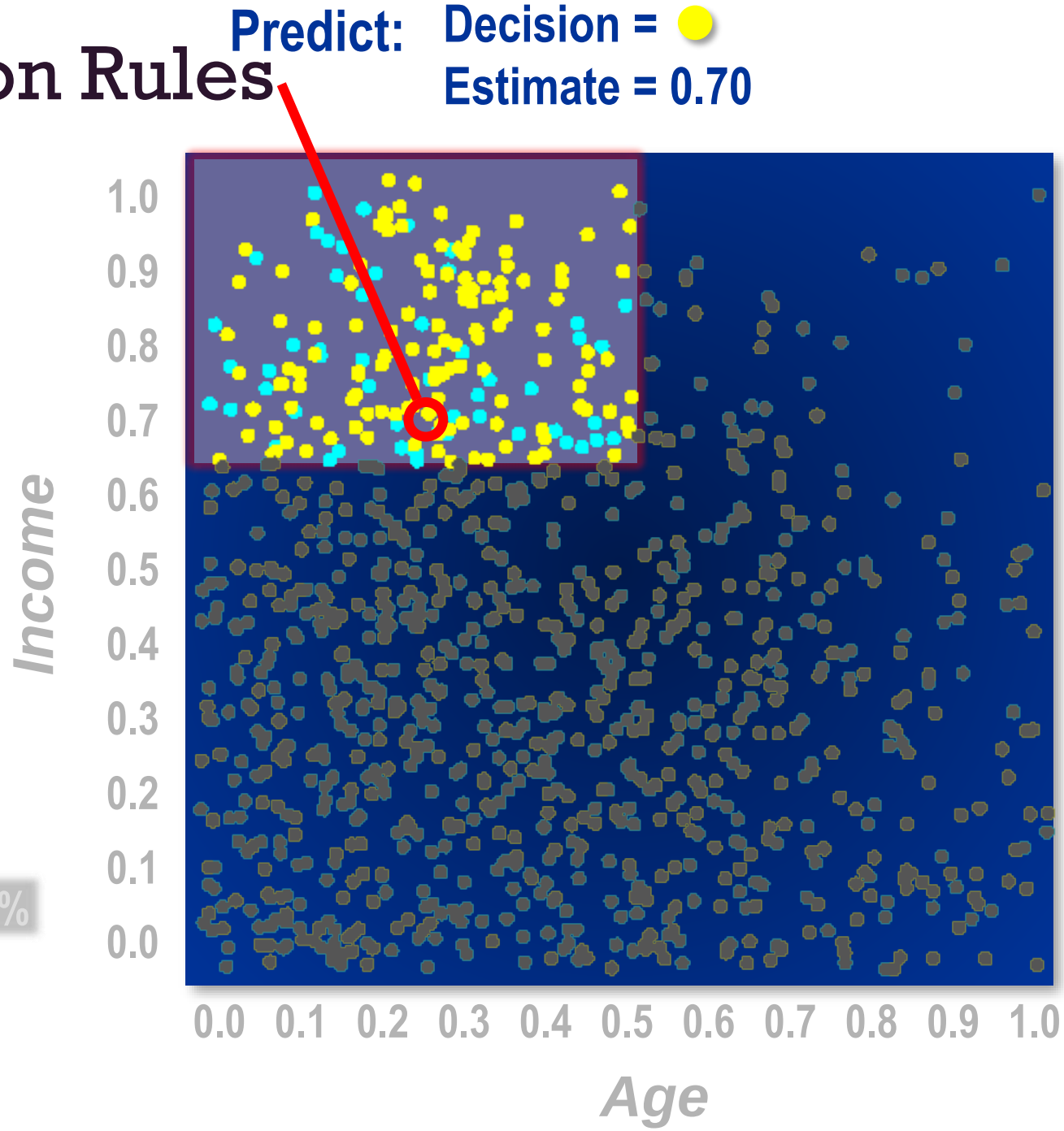
Test Phase





Decision Tree Prediction Rules





1) Entropy (impurity)

- **Entropy** is a measure of disorder or uncertainty and the goal of machine learning models and general is to reduce uncertainty.

$$Entropy = \sum_{i=1}^{n} -p_i log_2 p_i$$

scipy.stats.entropy

scipy.stats.**entropy**(*pk*, *qk=None*, *base=None*, *axis=0*)

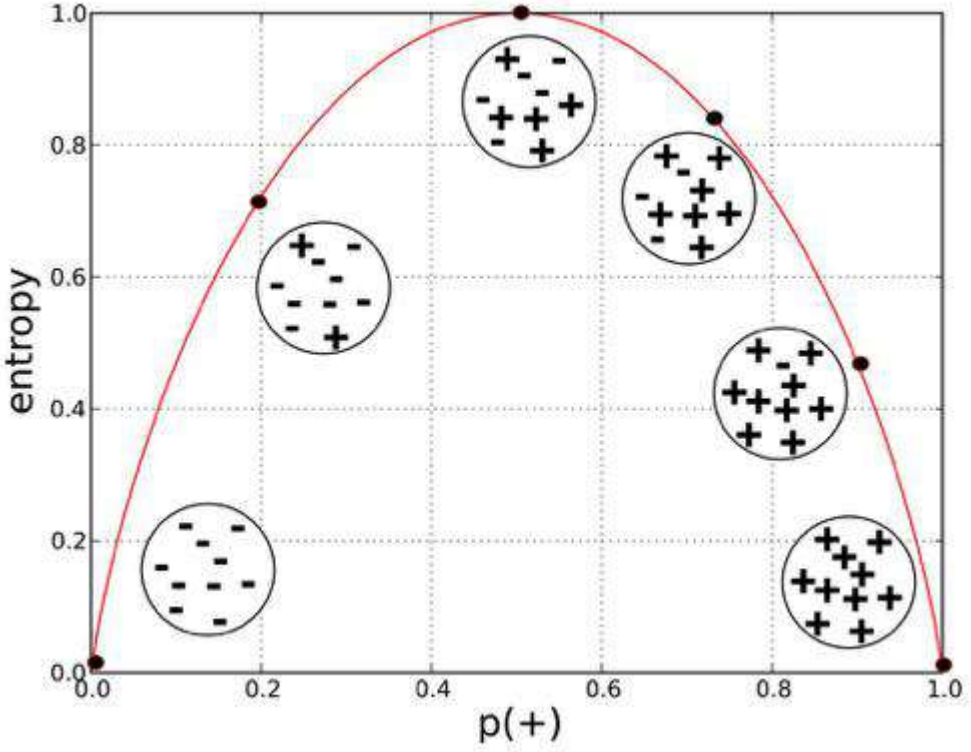
Calculate the entropy of a distribution for given probability values.

If only probabilities *pk* are given, the entropy is calculated as `S = -sum(pk * log(pk), axis=axis)`.

If *qk* is not None, then compute the Kullback-Leibler divergence `S = sum(pk * log(pk / qk), axis=axis)`.

This routine will normalize *pk* and *qk* if they don't sum to 1.

SciPy.org



Source: Data Science for Business: What You Need to Know about Data Mining and Data-Analytic Thinking

Information Gain
Before - After

- Which one is Better ?

Split w/ Age: 70
∑ Entropy: 0.350

Split w/ Age: 50
∑ Entropy: 0.348

- Information Gain: measure the reduction of this disorder in our target variable/class given additional information

$$InformationGain = Entropy(before) - \sum Entropy(after)$$

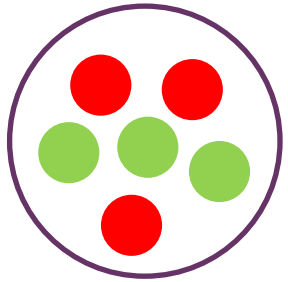
- "Before" = Entropy of Parent Node
  "After" = Entropy of Child Nodes

2) Gini Impurity
Gini Reduction = Before - After

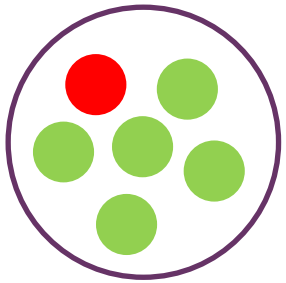
- Another way to measure how well a splitting feature.

$$Gini = 1 - \sum_{i=1}^{n} (P_i)^2$$

When P is the probability of class i in data-set.



$$Gini = 1 - ((3/6)^2 + (3/6)^2)$$
$$= 0.5$$

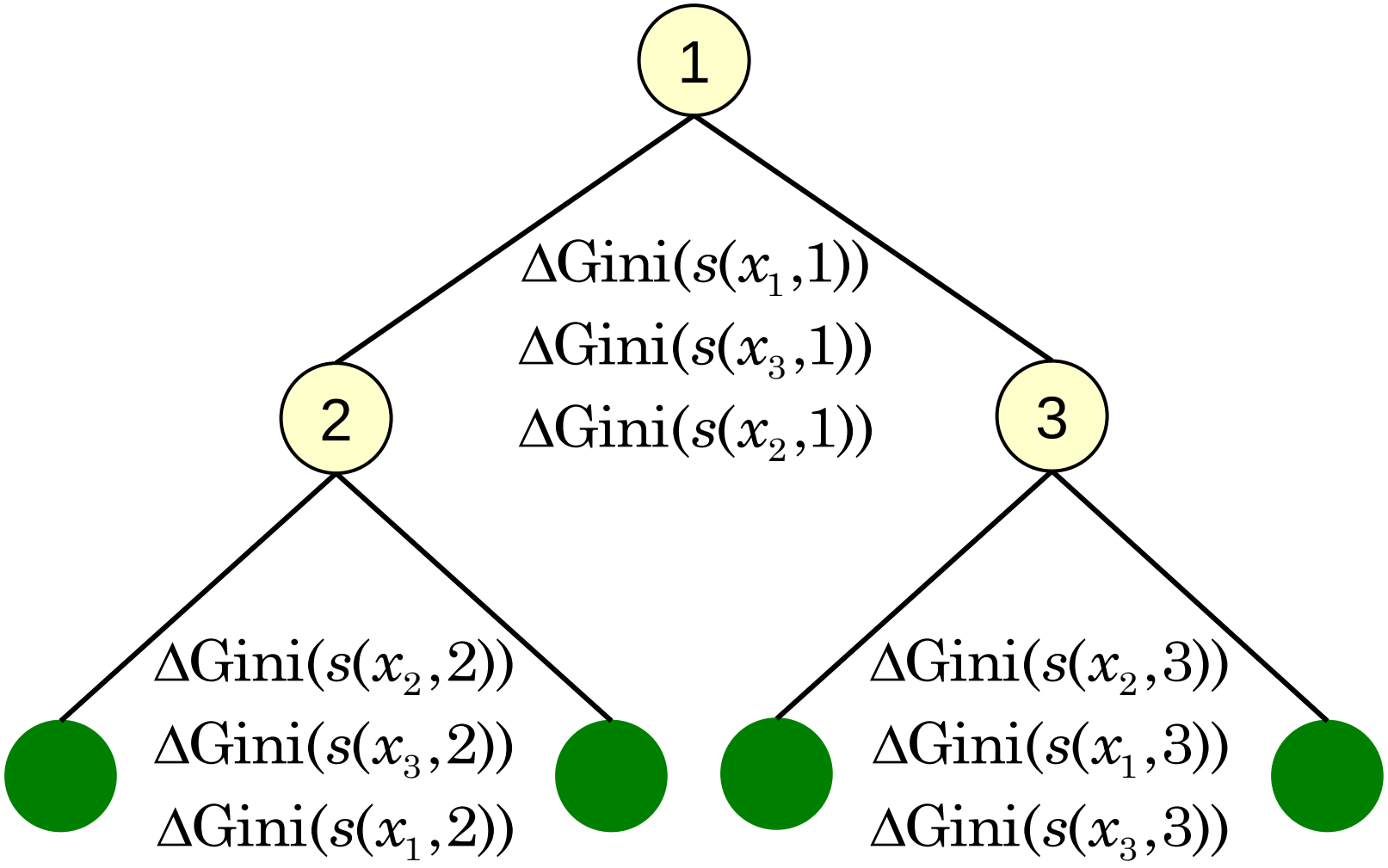


$$Gini = 1 - ((5/6)^2 + (1/6)^2)$$
$$= 0.28$$

- Easy to calculation, may take <u>less time</u> to build in large dataset.

Source: https://towardsdatascience.com/understanding-decision-tree-classification-with-scikit-learn-2ddf272731bd

Variable Importance



Types of Decision Tree

Algorithm	Splitting Measure
ID3	Entropy
C4.5	Gain Ratio
CART	Gini index
CHAID	Chi-squared test

sklearn.tree.DecisionTreeClassifier

class sklearn.tree.**DecisionTreeClassifier**(*criterion='gini', splitter='best', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features=None, random_state=None, max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, class_weight=None, presort='deprecated', ccp_alpha=0.0*) [source]

A decision tree classifier.

Read more in the User Guide.

Parameters:

criterion : *{"gini", "entropy"}, default="gini"*
The function to measure the quality of a split. Supported criteria are "gini" for the Gini impurity and "entropy" for the information gain.

splitter : *{"best", "random"}, default="best"*
The strategy used to choose the split at each node. Supported strategies are "best" to choose the best split and "random" to choose the best random split.

max_depth : *int, default=None*
The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples.

min_samples_split : *int or float, default=2*
The minimum number of samples required to split an internal node:

Decision Tree with Regression

sklearn.tree.DecisionTreeRegressor

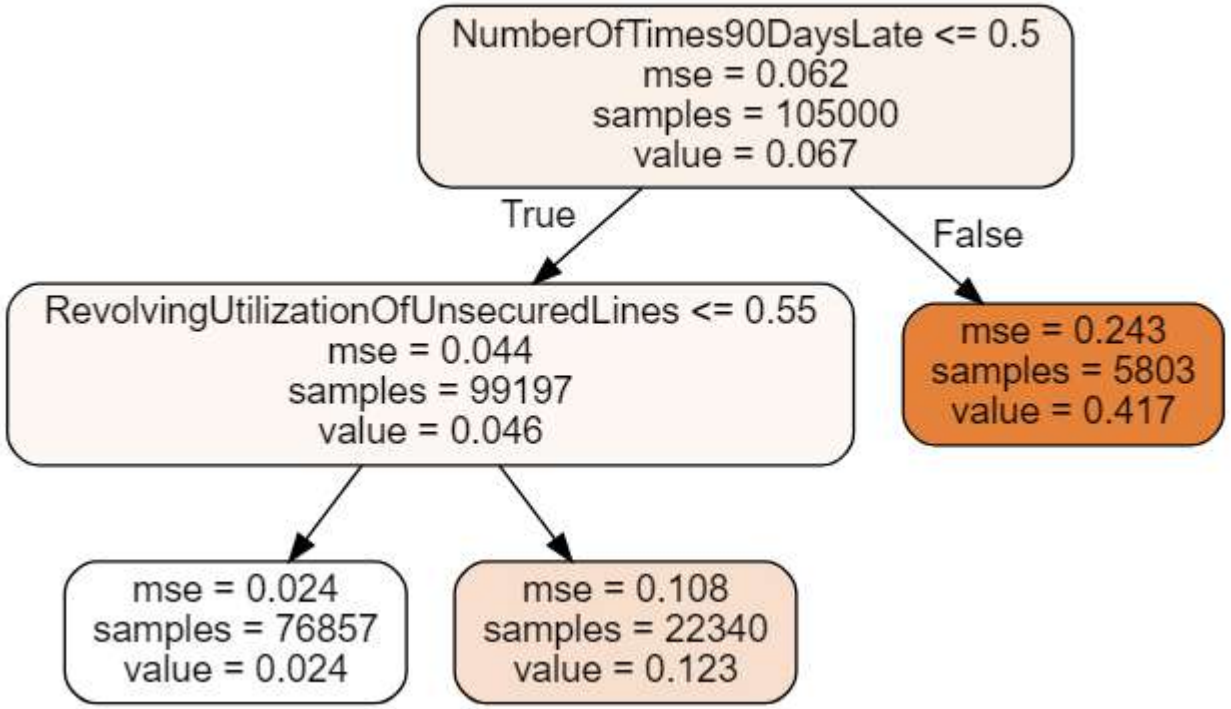
class `sklearn.tree.` **`DecisionTreeRegressor`**(*criterion='mse', splitter='best', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features=None, random_state=None, max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, presort='deprecated', ccp_alpha=0.0*) [source]

A decision tree regressor.

Read more in the User Guide.

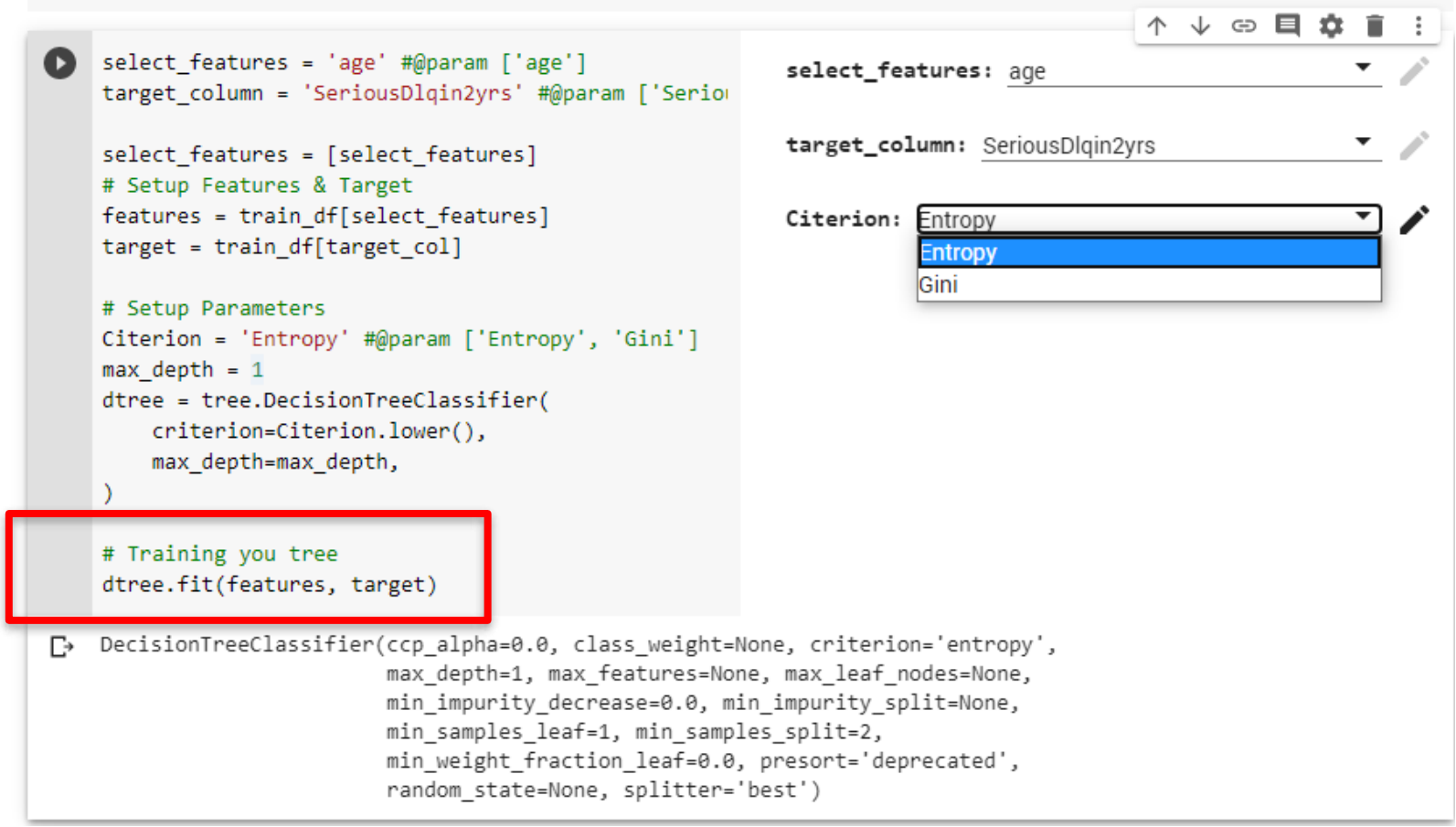
Parameters: criterion : ***{"mse", "friedman_mse", "mae"},***

MSE: Mean Square Error
MAE: Mean Absolute Error



Build your Decision Tree (cont.)

- Code Section 2.4



```
select_features = 'age' #@param ['age']
target_column = 'SeriousDlqin2yrs' #@param ['Serio

select_features = [select_features]
# Setup Features & Target
features = train_df[select_features]
target = train_df[target_col]

# Setup Parameters
Citerion = 'Entropy' #@param ['Entropy', 'Gini']
max_depth = 1
dtree = tree.DecisionTreeClassifier(
    criterion=Citerion.lower(),
    max_depth=max_depth,
)

# Training you tree
dtree.fit(features, target)
```

select_features: age

target_column: SeriousDlqin2yrs

Citerion: Entropy

Entropy
Gini

```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='entropy',
                       max_depth=1, max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')
```

Model Visualization (read the model)

Basic Description

1. get_params : Get all model parameters
2. get_depth : Get depth of trained tree
3. get_n_leaves : Get number of nodes
4. tree_.impurity : Get Gini/Entropy values

1
```
Parameters: {
 "ccp_alpha": 0.0,
 "class_weight": null,
 "criterion": "entropy",
 "max_depth": 1,
 "max_features": null,
 "max_leaf_nodes": null,
 "min_impurity_decrease": 0.0,
 "min_impurity_split": null,
 "min_samples_leaf": 1,
 "min_samples_split": 2,
 "min_weight_fraction_leaf": 0.0,
 "presort": "deprecated",
 "random_state": null,
 "splitter": "best"
}
```
2
```
Tree Depth: 1
```
3
```
Number of Node: 2
```
4
```
Entropy of Nodes: [0.35401168 0.42690946 0.2175608 ]
```

Model Visualization (cont.)

sklearn.tree.plot_tree

sklearn.tree.plot_tree(*decision_tree*, *, *max_depth=None*, *feature_names=None*, *class_names=None*, *label='all'*, *filled=False*, *impurity=True*, *node_ids=False*, *proportion=False*, *rounded=False*, *precision=3*, *ax=None*, *fontsize=None*) [source]

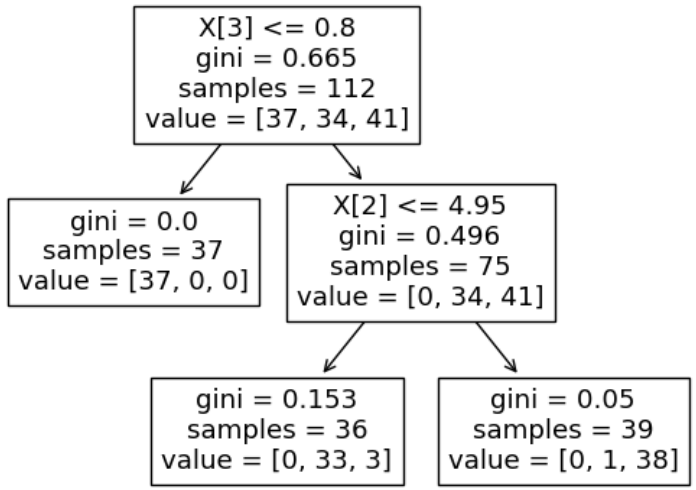
Plot a decision tree.

The sample counts that are shown are weighted with any sample_weights that mig

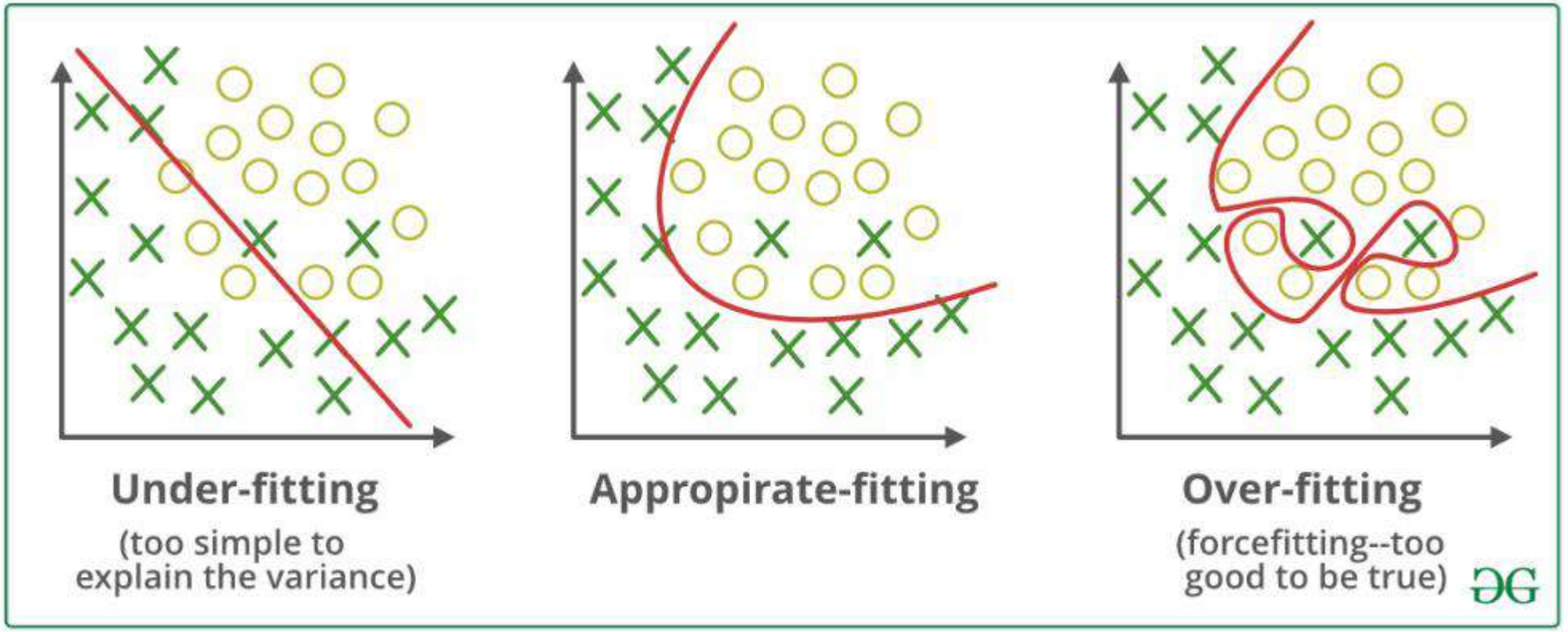
The visualization is fit automatically to the size of the axis. Use the `figsize` or `dp` size of the rendering.

Read more in the User Guide.

New in version 0.21.



Overfitting



Source: https://towardsdatascience.com/underfitting-and-overfitting-in-machine-learning-and-how-to-deal-with-it-6fe4a8a49dbf

Pruning Tree

https://scikit-learn.org/stable/modules/tree.html#minimal-cost-complexity-pruning/

ccp_alpha : *non-negative float, default=0.0*

Complexity parameter used for Minimal Cost-Complexity Pruning. The subtree with the largest cost complexity that is smaller than `ccp_alpha` will be chosen. By default, no pruning is performed. See Minimal Cost-Complexity Pruning for details.

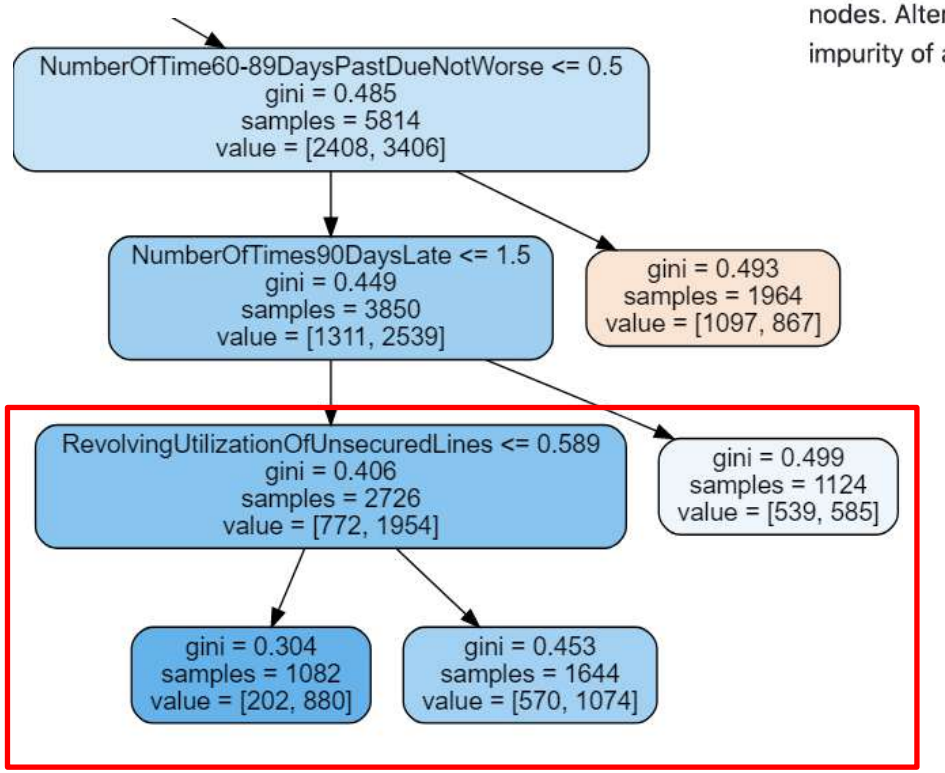
New in version 0.22.

$$R_\alpha(T) = R(T) + \alpha|T|$$

where $|T|$ is the number of terminal nodes in T and $R(T)$ is traditionally defined as the total misclassification rate of the terminal nodes. Alternatively, scikit-learn uses the total sample weighted impurity of the terminal nodes for $R(T)$. As shown above, the impurity of a node depends on the criterion. Minimal cost-complexity pruning finds the subtree of T that minimizes $R_\alpha(T)$.

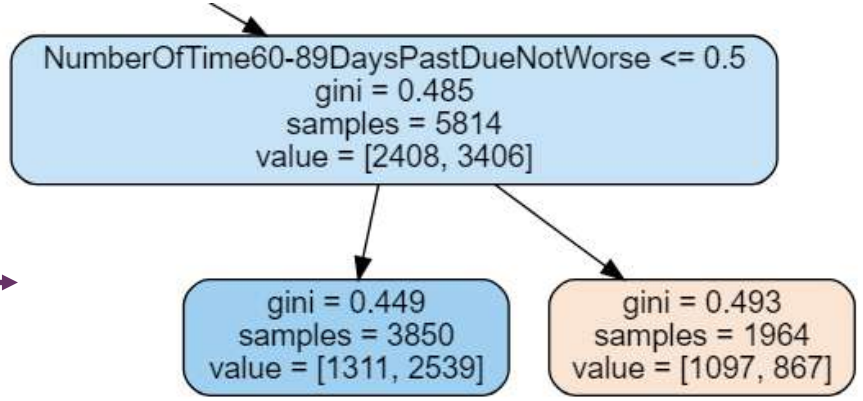
More

Impurity



Less

Prune



When impurity is too low, it cause more overfitting

Test pruning parameter by .cost_complexity_pruning_path

```
dtree.cost_complexity_pruning_path(features, target)
```

```
{'ccp_alphas': array([0.00000000e+00, 8.91619910e-05, 1.17039010e-04, 1.29787452e-04,
       1.94525732e-04, 3.18283592e-04, 4.15157131e-04, 5.84363721e-04,
       6.81710606e-04, 1.05951846e-03, 1.17765555e-03, 1.83295138e-03,
       3.24166605e-03, 1.41432990e-02]),
 'impurities': array([0.10075641, 0.10084557, 0.10096261, 0.1010924 , 0.10128692,
       0.10160521, 0.10202036, 0.10260473, 0.10328644, 0.10434596,
       0.10552361, 0.10735656, 0.11059823, 0.12474153])}
```

ccp_alphas: Values which affect pruning (use lower bound value)
Impurities: Impurity after pruning (default=0; no pruning)

Important Parameters in Decision Tree

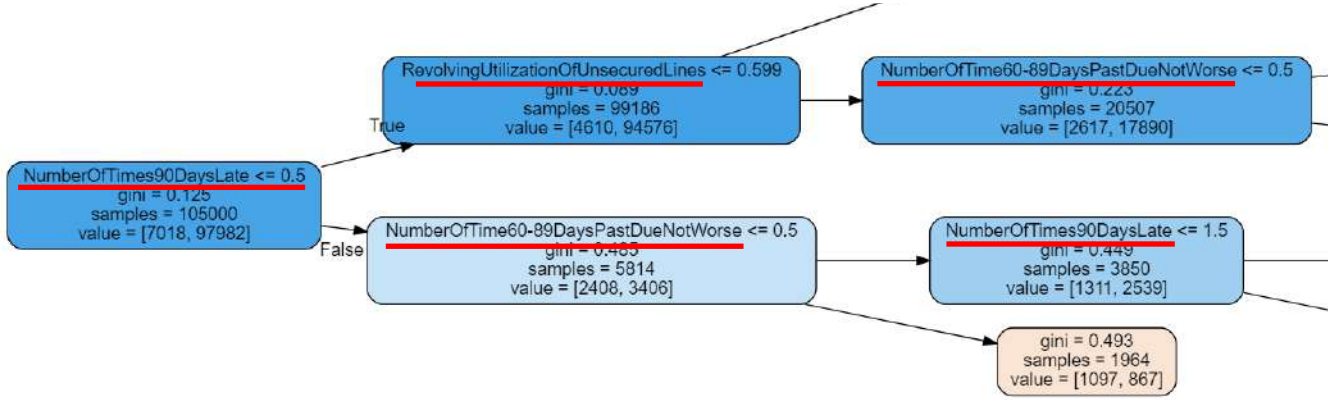
- Splitting measure (criterion) : gini / entropy
- Maximum depth : ~5-10 (depend on number of feature)
- Maximum leaf nodes : depend on number of class (target) and feature
- Minimum sample split : 5 – 20% (depend on number of data)
- Minimum impurity decrease : (default 0)
- Pruning adjustment (ccp_alpha) : 0.001 - 0.01 (depend on size of tree)

Which features are important ?

- Check how important by . feature_importances_
- The importance of a feature is computed as the (normalized) total reduction of the criterion brought by that feature. It is also known as the gini importance.

```
dtree.feature_importances_
```

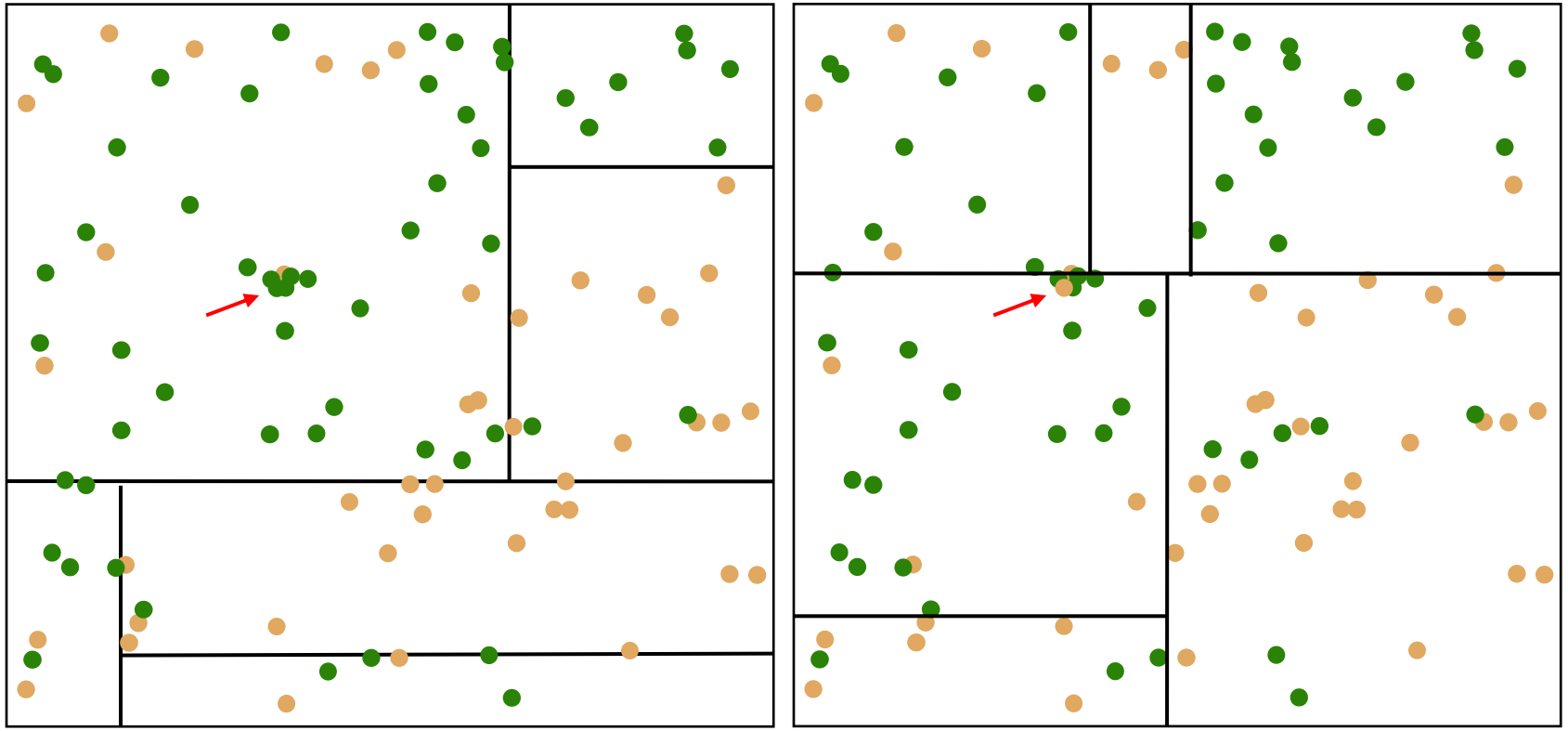
```
array([0.15330291, 0.00371739, 0.07500437, 0.        , 0.        ,
       0.        , 0.61403332, 0.        , 0.15394201, 0.        ])
```




```
======= Features Important =======
0.614 : Numberoftimes90dayslate
0.154 : Numberoftime60-89dayspastduenotworse
0.153 : Revolvingutilizationofunsecuredlines
0.075 : Numberoftime30-59dayspastduenotworse
0.004 : Age
0.000 : Debtratio
0.000 : Monthlyincome
0.000 : Numberofopencreditlinesandloans
0.000 : Numberrealestateloansorlines
0.000 : Numberofdependents
```

Instability

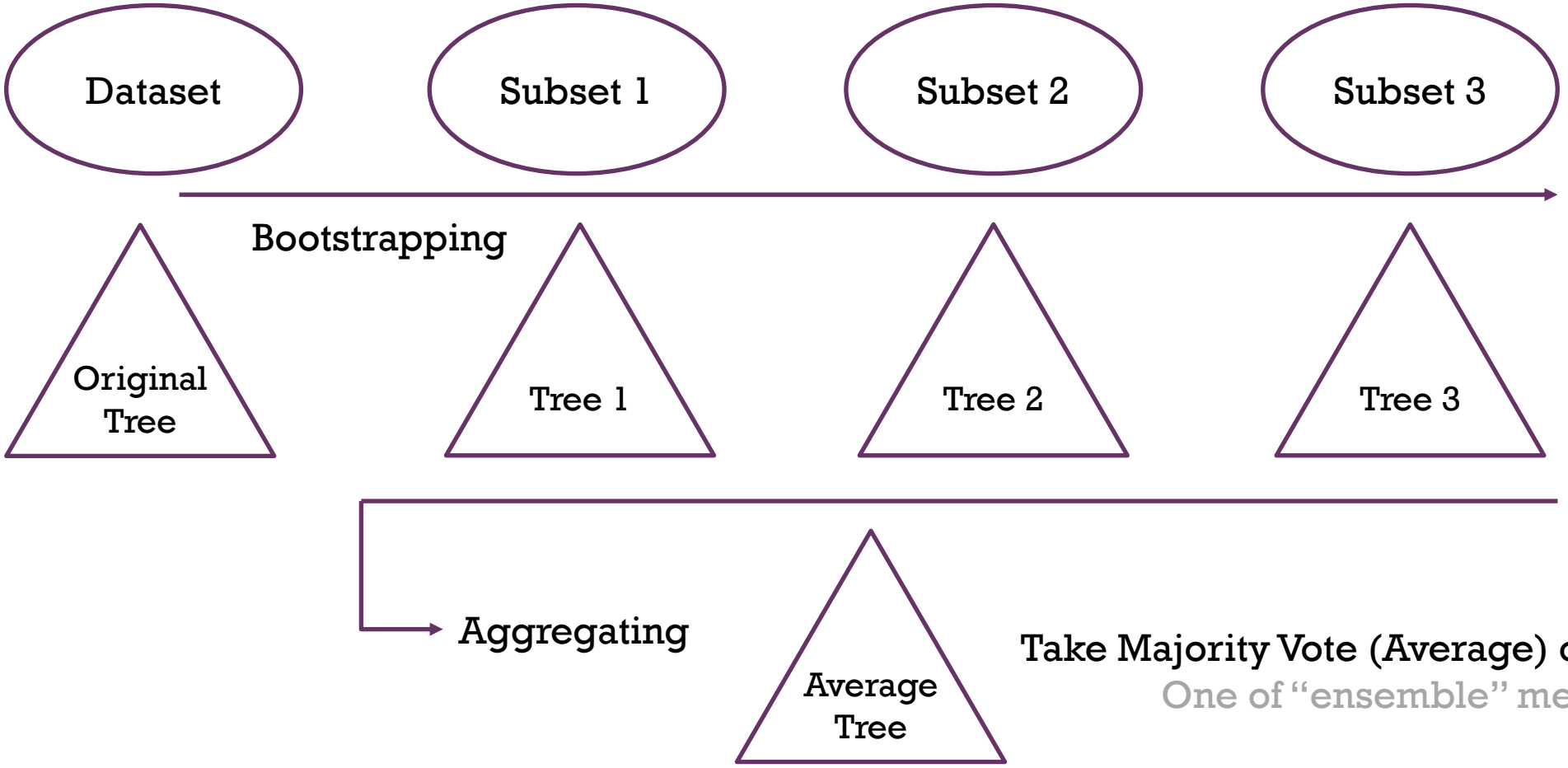
One reversal



Accuracy = 81%

Accuracy = 80%

Bagging (Bootstrap Aggregation)



Take Majority Vote (Average) of Prediction

One of "ensemble" method

Bagging Method in sklearn

sklearn.ensemble.BaggingClassifier

class sklearn.ensemble.**BaggingClassifier**(*base_estimator=None, n_estimators=10, *, max_samples=1.0, max_features=1.0, bootstrap=True, bootstrap_features=False, oob_score=False, warm_start=False, n_jobs=None, random_state=None, verbose=0*)

[source]

sklearn.ensemble.BaggingRegressor¶

class sklearn.ensemble.**BaggingRegressor**(*base_estimator=None, n_estimators=10, *, max_samples=1.0, max_features=1.0, bootstrap=True, bootstrap_features=False, oob_score=False, warm_start=False, n_jobs=None, random_state=None, verbose=0*)

[source]

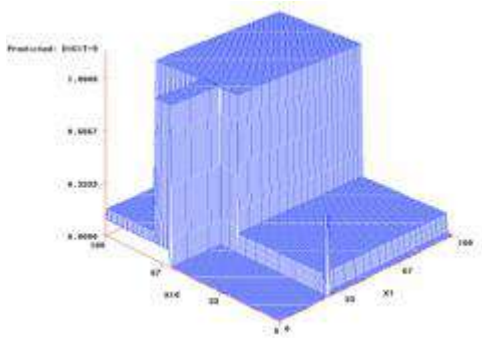
Example

```
>>> from sklearn.ensemble import BaggingClassifier
>>> from sklearn.tree import DecisionTreeClassifier
>>> bagging = BaggingClassifier(DecisionTreeClassifier(),
...                             max_samples=0.5, max_features=0.5)
```

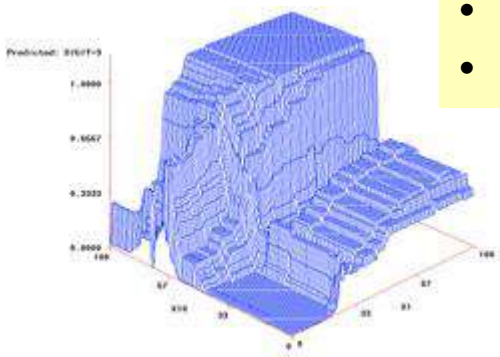
Random Forest Algorithm (RF)

- Forest algorithm samples the rows and the columns at each step.
- Forest takes bootstrap samples of the rows in training data (sampling with replacement)
- At each step, a set of variables (columns) is sampled.
- This increases variation among trees in the **ensemble** often leads to improved prediction accuracy.

Important Params:
- #Tree
- #Columns (variables)
- #Rows (examples)

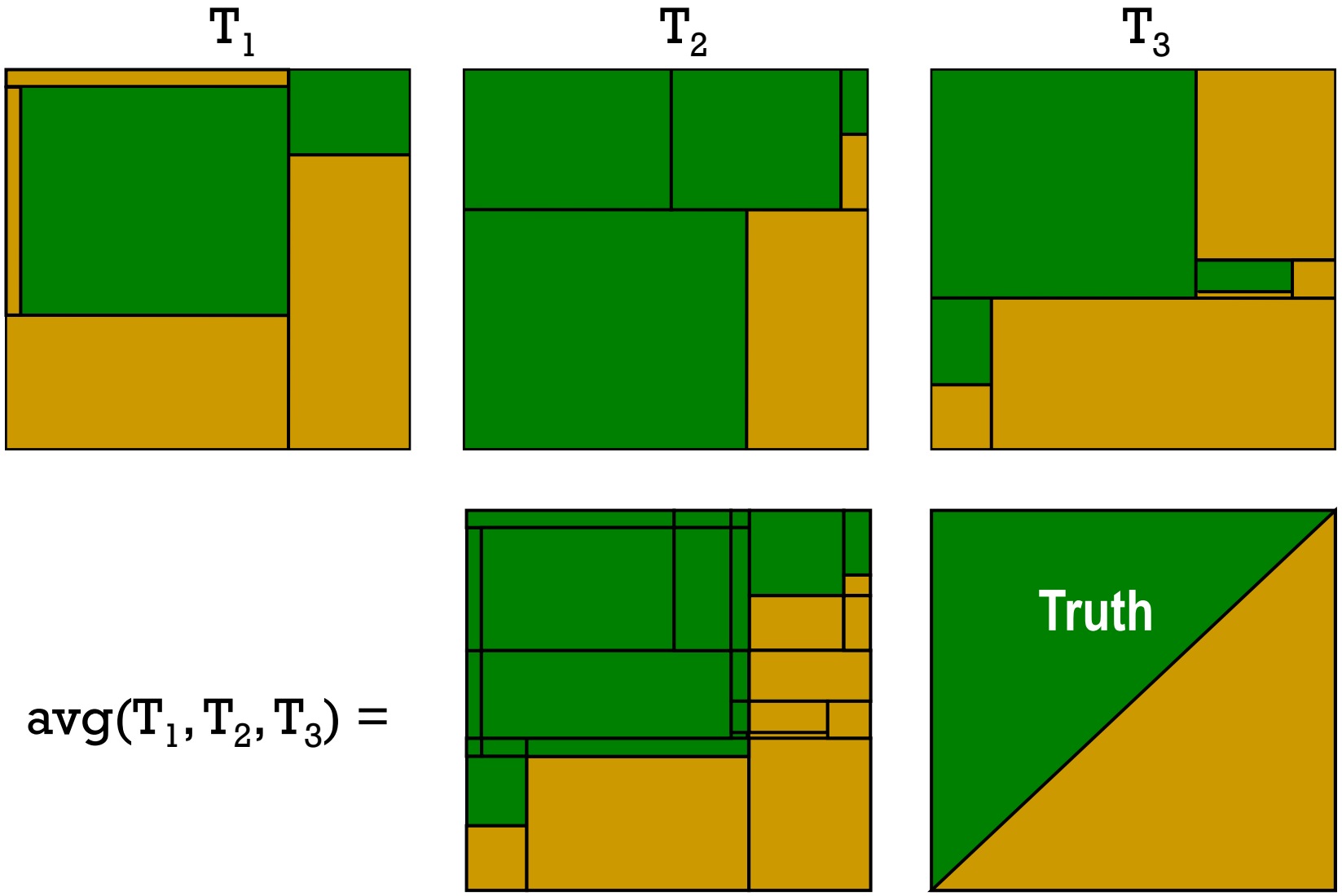


Single Tree



Random Forest

Combine



3.2.4.3.1. sklearn.ensemble.RandomForestClassifier

class `sklearn.ensemble.` **`RandomForestClassifier`**(*n_estimators=100, criterion='gini', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features='auto', max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, bootstrap=True, oob_score=False, n_jobs=None, random_state=None, verbose=0, warm_start=False, class_weight=None, ccp_alpha=0.0, max_samples=None*) [source]

A random forest classifier.

A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is always the same as the original input sample size but the samples are drawn with replacement if `bootstrap=True` (default).

Read more in the User Guide.

Parameters:

n_estimators : *integer, optional (default=100)*
The number of trees in the forest.

Changed in version 0.22: The default value of `n_estimators` changed from 10 to 100 in 0.22.

criterion : *string, optional (default="gini")*
The function to measure the quality of a split. Supported criteria are "gini" for the Gini impurity and "entropy" for the information gain. Note: this parameter is tree-specific.

3.2.4.3.2. sklearn.ensemble.RandomForestRegressor

class sklearn.ensemble.**RandomForestRegressor**(*n_estimators=100*, *criterion='mse'*, *max_depth=None*, *min_samples_split=2*, *min_samples_leaf=1*, *min_weight_fraction_leaf=0.0*, *max_features='auto'*, *max_leaf_nodes=None*, *min_impurity_decrease=0.0*, *min_impurity_split=None*, *bootstrap=True*, *oob_score=False*, *n_jobs=None*, *random_state=None*, *verbose=0*, *warm_start=False*, *ccp_alpha=0.0*, *max_samples=None*) ¶ [source]

A random forest regressor.

A random forest is a meta estimator that fits a number of classifying decision trees on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is always the same as the original input sample size but the samples are drawn with replacement if `bootstrap=True` (default).

Read more in the User Guide.

Parameters:

n_estimators : *integer, optional (default=10)*

The number of trees in the forest.

Changed in version 0.22: The default value of `n_estimators` changed from 10 to 100 in 0.22.

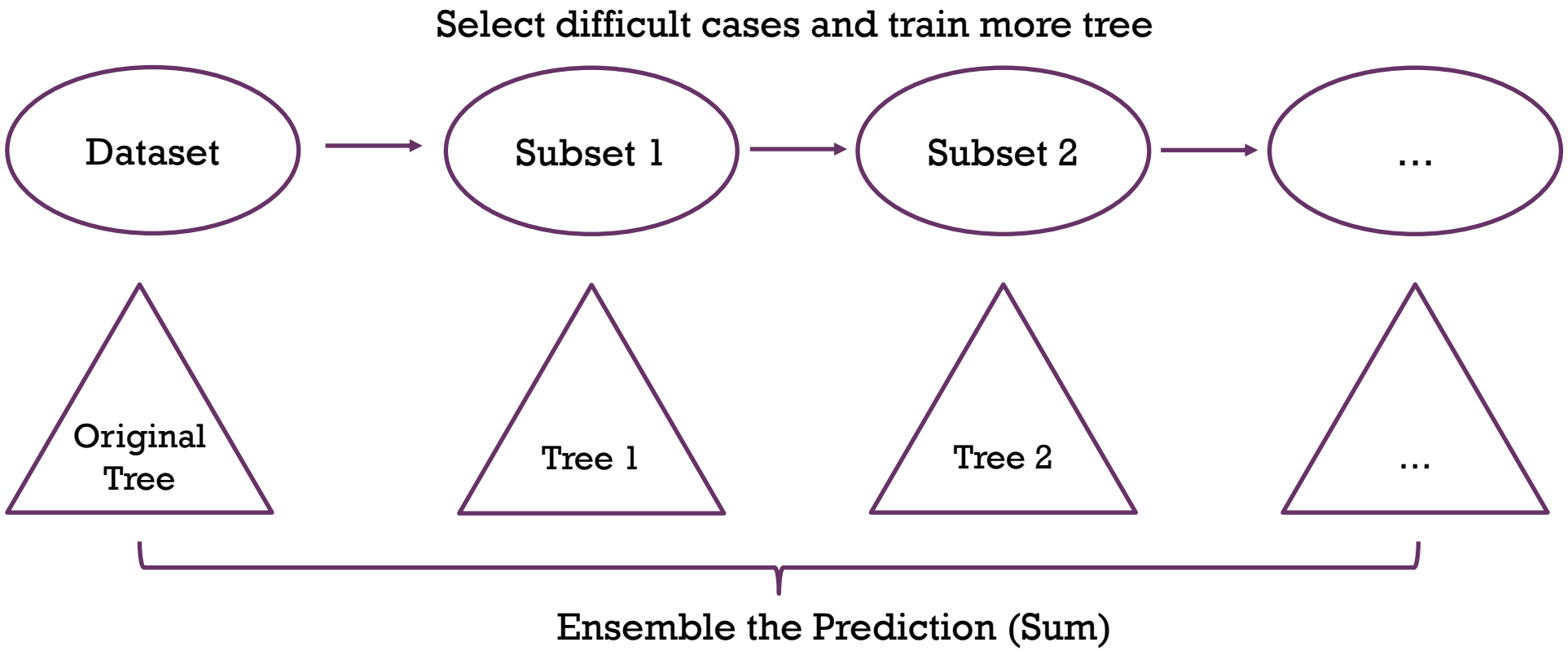
criterion : *string, optional (default="mse")*

The function to measure the quality of a split. Supported criteria are "mse" for the mean squared error, which is equal to variance reduction as feature selection criterion, and "mae" for the mean absolute error.

New in version 0.18: Mean Absolute Error (MAE) criterion.

Boosting

- Boosting is a method of converting weak learners into strong learners.
- In boosting, each new tree is a fit on a modified version of the original data set.



Boosting (cont.)

	k=1		k=2		k=3		k=4 …
case	freq	m	freq	m	freq	m	freq
1	1	1	1.5	1	.5	2	.97
2	1	0	.75	0	.25	0	.06
3	1	1	1.5	2	4.25	3	4.69
4	1	0	.75	1	.5	1	.11
5	1	0	.75	0	.25	0	.06
6	1	0	.75	0	.25	1	.11

*Shown is Arc-x4, one method of boosting

Boosting in sklearn

- AdaBoost, short for "Adaptive Boosting"

sklearn.ensemble.AdaBoostClassifier¶

class sklearn.ensemble.**AdaBoostClassifier**(*base_estimator=None, *, n_estimators=50, learning_rate=1.0, algorithm='SAMME.R', random_state=None*) [source]

sklearn.ensemble.AdaBoostRegressor

class sklearn.ensemble.**AdaBoostRegressor**(*base_estimator=None, *, n_estimators=50, learning_rate=1.0, loss='linear', random_state=None*) [source]

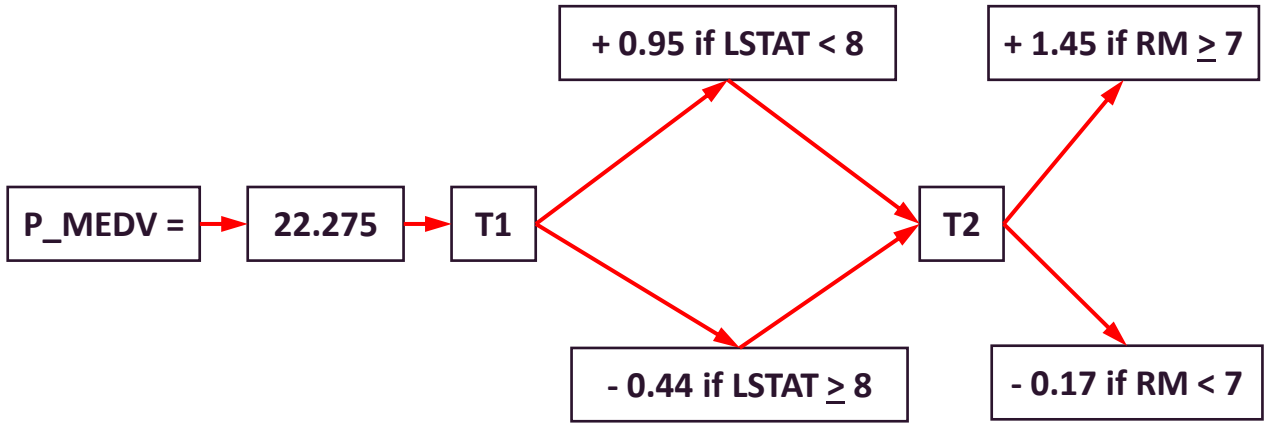
Example

```
>>> from sklearn.ensemble import AdaBoostClassifier
>>> from sklearn.tree import DecisionTreeClassifier
>>> boosting = AdaBoostClassifier(DecisionTreeClassifier(),
...                               max_samples=0.5, max_features=0.5)
```

https://towardsdatascience.com/boosting-algorithm-adaboost-b6737a9ee60c#:~:text=AdaBoost,classification%20can%20be%20represented%20as

Gradient Boosting Tree

- Boosting the performance by using ensemble trees
- Each tree in the next step aims to predict *error*.
- So, the prediction result is the summation of all trees (not average any more).
- In the final step, weights are assigned to all leaf nodes using gradient descent in order to finally reduce errors.

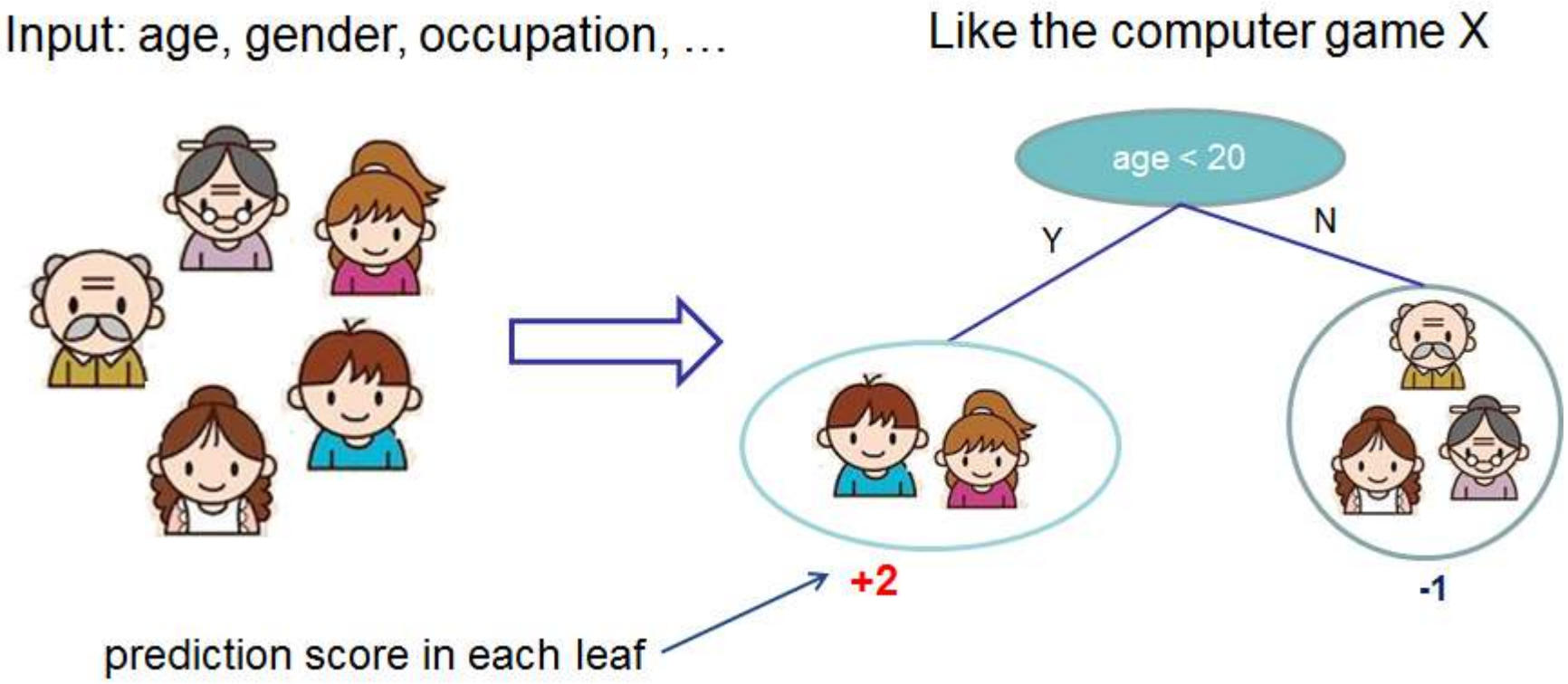


Boston House Price Prediction

- 'RM' is the average number of rooms among homes in the neighborhood.
- 'LSTAT' is the percentage of homeowners in the neighborhood considered "lower class" (working poor).

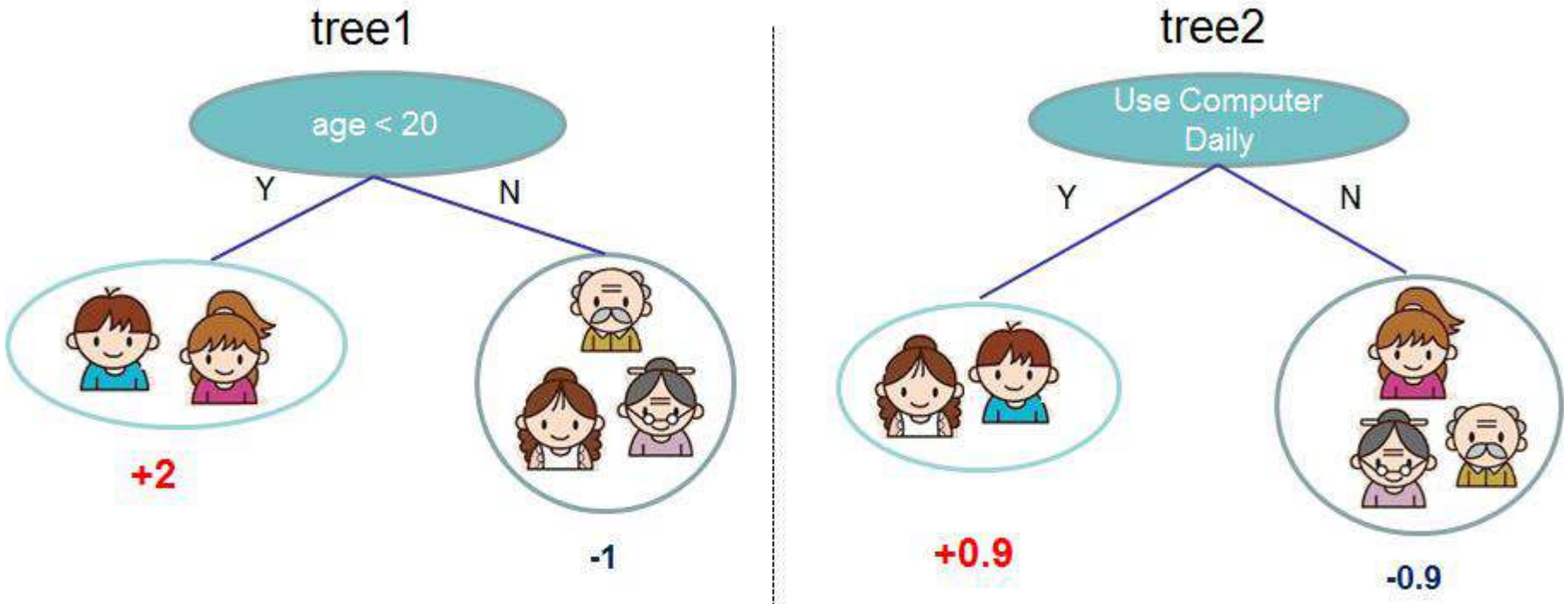
Intro to Gradient Boosting Tree

- Decision Tree



Intro to Gradient Boosting Tree (cont.)

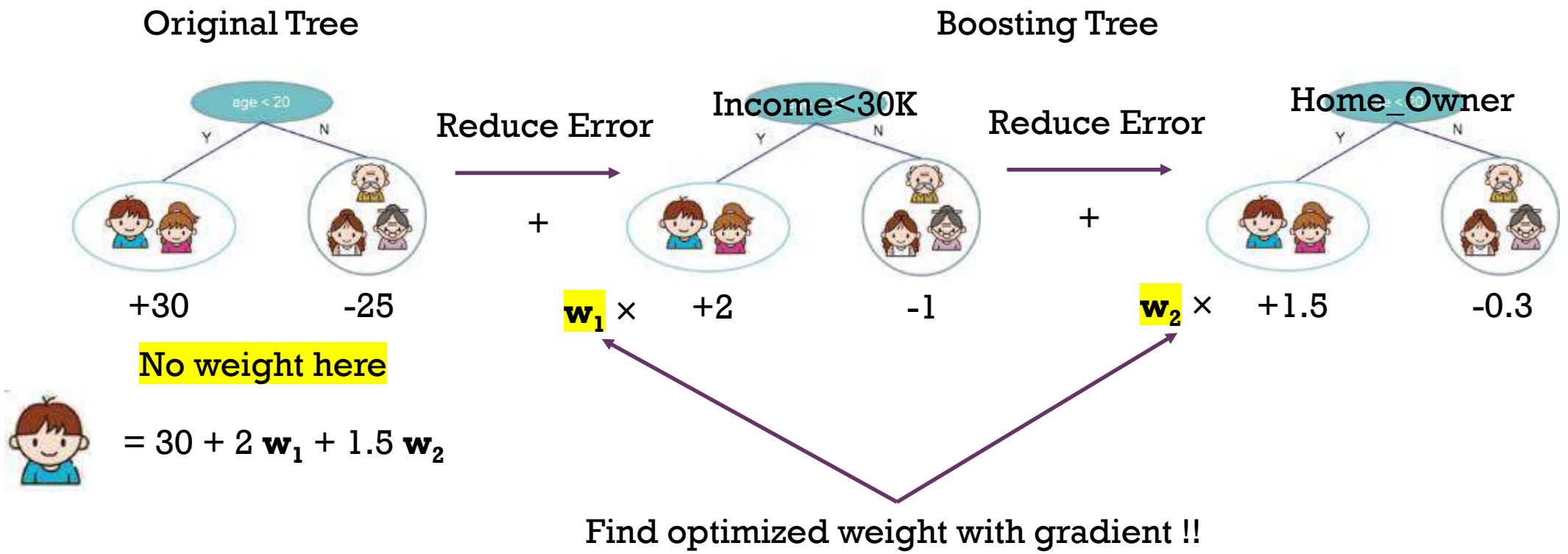
- Ensemble by weight on leaf nodes



f( ) = 2 + 0.9= 2.9 f( )= -1 - 0.9= -1.9

Intro to Gradient Boosting Tree (cont.)

- Optimize weight of leaf nodes by gradient descent



Gradient Boosting Tree in sklearn

3.2.4.3.5. sklearn.ensemble.GradientBoostingClassifier

class sklearn.ensemble.**GradientBoostingClassifier**(**, loss='deviance', learning_rate=0.1, n_estimators=100, subsample=1.0, criterion='friedman_mse', min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_depth=3, min_impurity_decrease=0.0, min_impurity_split=None, init=None, random_state=None, max_features=None, verbose=0, max_leaf_nodes=None, warm_start=False, presort='deprecated', validation_fraction=0.1, n_iter_no_change=None, tol=0.0001, ccp_alpha=0.0*) [source]

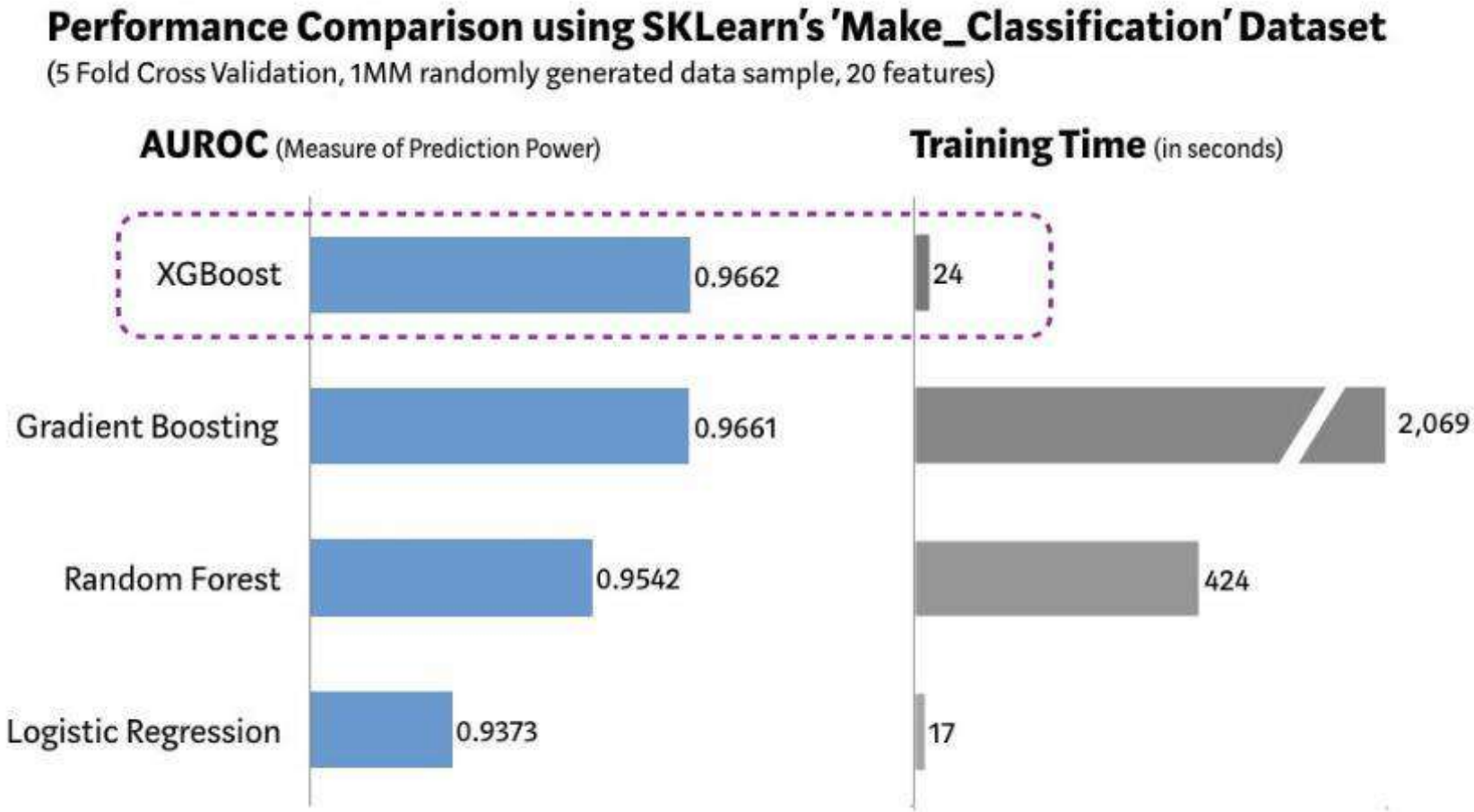
eXtreme Gradient Boosting Tree (XGBoost)

- XGBoost stands for "Extreme Gradient Boosting", where the term "Gradient Boosting" originates from the paper *Greedy Function Approximation: A Gradient Boosting Machine*, by Friedman.

- XGBoost is exactly a tool motivated by the formal principle introduced in this tutorial! More importantly, it is developed with both deep consideration in terms of **systems optimization** and **principles in machine learning**. The goal of this library is to push the extreme of the computation limits of machines to provide a **scalable**, **portable** and **accurate** library

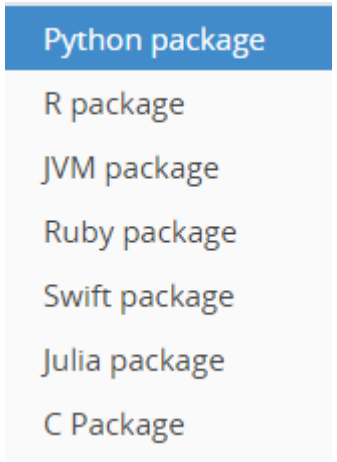
dmlc XGBoost

Models Performance



XGBoost Package

- Multiple Language Support



- Distributed Model on Cloud
  - AWS
  - Kubernetes
  - Spark
  - Dask
- Compatible with sklearn interface

https://xgboost.readthedocs.io/en/latest/parameter.html

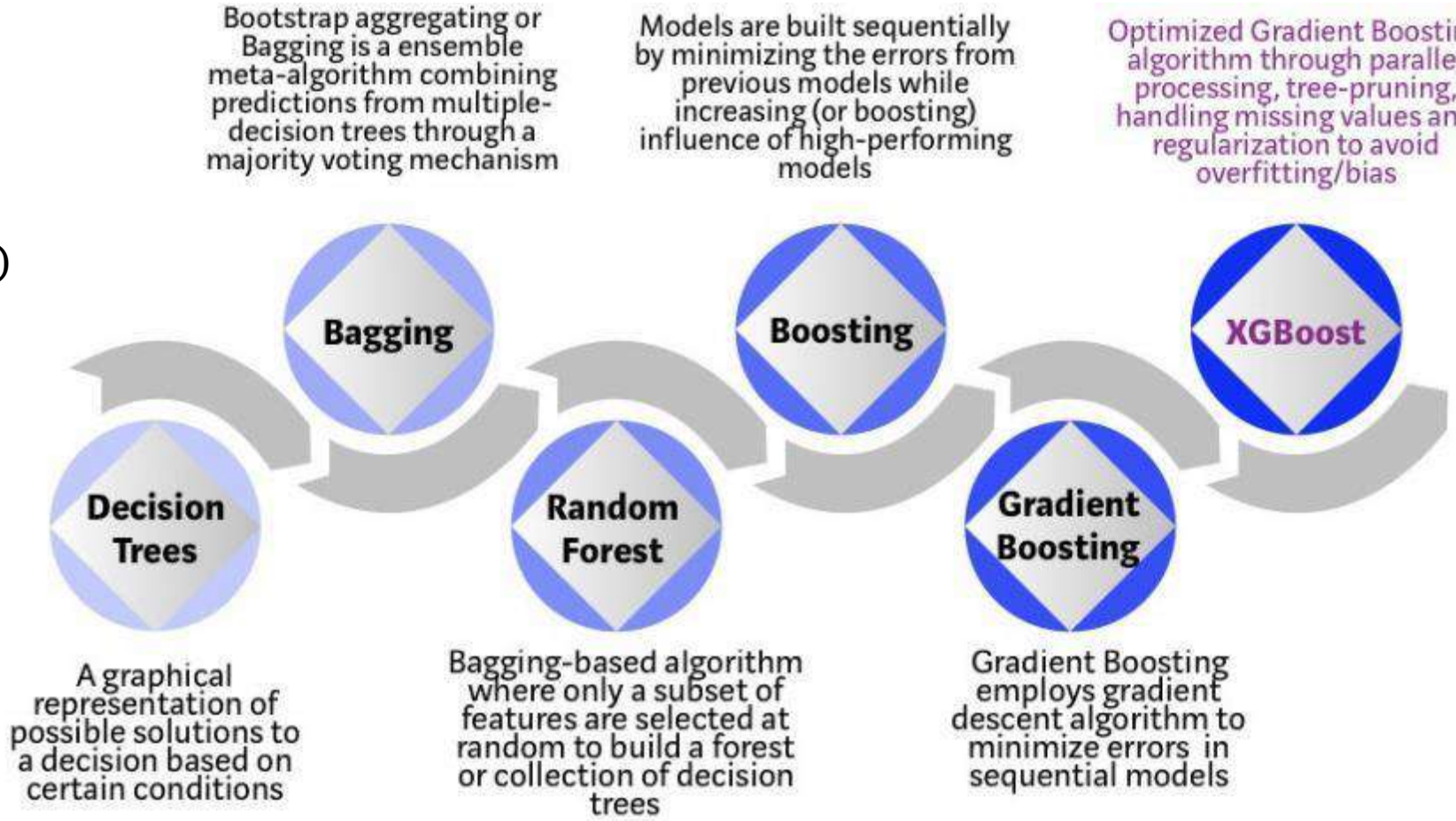
XGBoost Important Parameter

- n_jobs : Number of parallel tasks (-1 = all cores)
- objective : Type of Prediction
  - for Classification `binary:hinge`, `binary:logistic` (prob) or `multi:softmax` (multi-class)
  - for Regression `reg:squarederror`
- learning_rate : Step size used in update weights
- n_estimators : Number of trees
- subsample (0, 1]: Number of training data to growing trees, will prevent overfitting
- importance_type : Get feature importance of each feature.
  - 'weight': the number of times a feature is used to split the data across all trees.
  - 'gain': the average gain across all splits the feature is used in.
  - 'cover': the average coverage across all splits the feature is used in.
  - 'total_gain': the total gain across all splits the feature is used in.
  - 'total_cover': the total coverage across all splits the feature is used in.

Evolution of Decision Trees

Performance (Accuracy, Time)

Method (Based method)

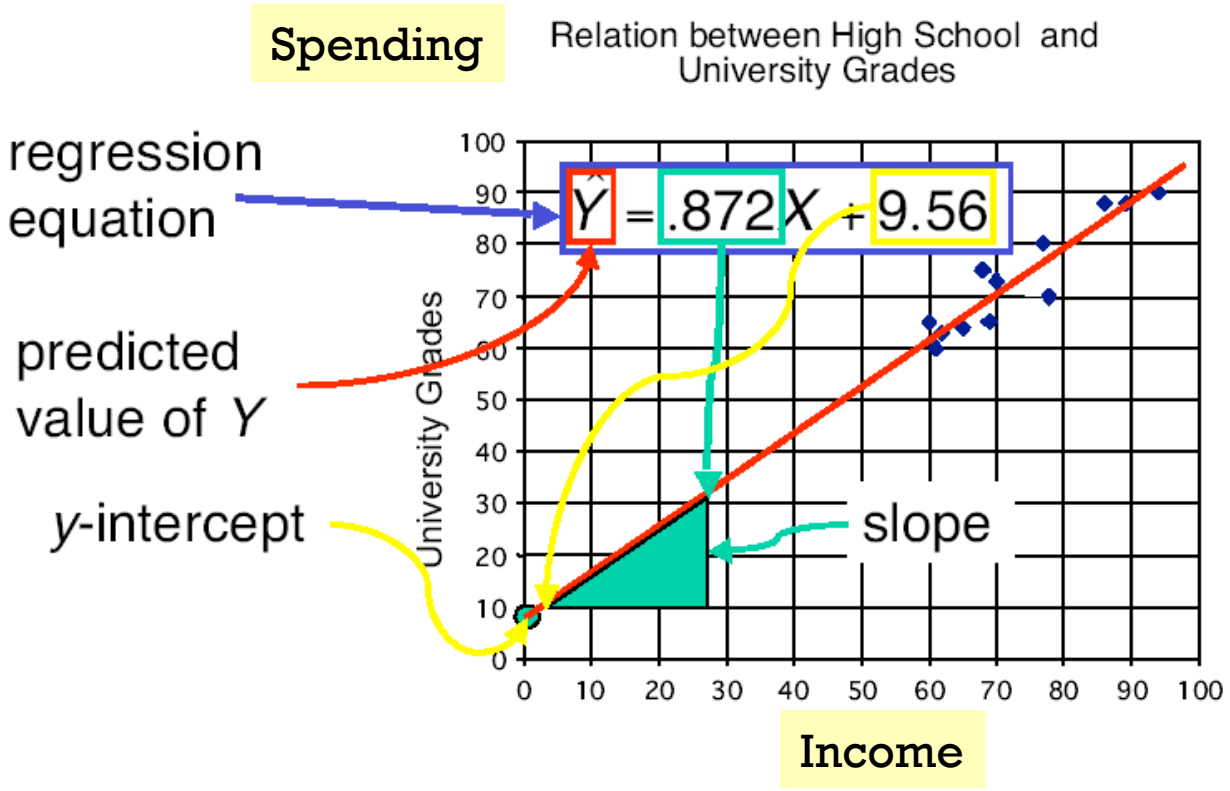


Source: https://towardsdatascience.com/https-medium-com-vishalmorde-xgboost-algorithm-long-she-may-rein-edd9f99be63d

+

Regression, Logistic Regression

2) Regression



weight, coefficient

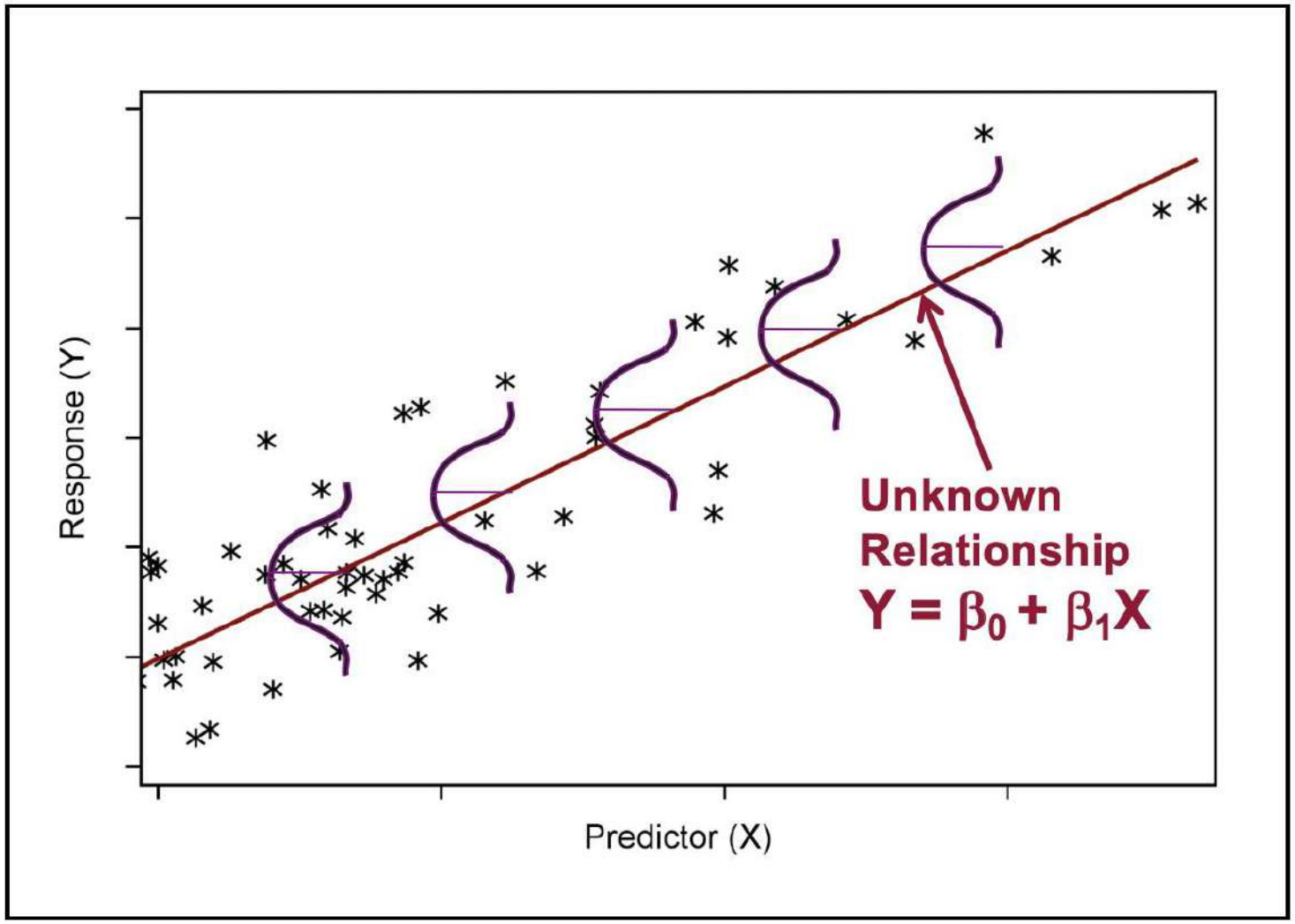
$$\underset{\text{target}}{\hat{y}} = \underset{\text{intercept}}{\hat{w}_0} + \hat{w}_1 x_1 + \hat{w}_2 x_2 \quad \text{input}$$

- The least square method aims to minimize the following term

$$\sum_{\substack{training \\ data}} (y_i - \hat{y}_i)^2$$

Linear Regression Assumption

Spending = 500 + 10*Income10K + 2*Age



- Linear relationship between (y, x_i). [Pearson correlation]
- Error is normal distributed. [remove outliers, log-transformation]
- Error has equal variance (homoscedasticity) [remove outliers, log-transformation]
- Errors are independent from each other. [design new data correction]

sklearn.linear_model.LinearRegression

class sklearn.linear_model. **LinearRegression**(*fit_intercept=True, normalize=False, copy_X=True, n_jobs=None*) [source]

Ordinary least squares Linear Regression.

LinearRegression fits a linear model with coefficients w = (w1, ..., wp) to minimize the residual sum of squares between the observed targets in the dataset, and the targets predicted by the linear approximation.

Parameters:

fit_intercept : *bool, optional, default True*
Whether to calculate the intercept for this model. If set to False, no intercept will be used in calculations (i.e. data is expected to be centered).

normalize : *bool, optional, default False*
This parameter is ignored when `fit_intercept` is set to False. If True, the regressors X will be normalized before regression by subtracting the mean and dividing by the l2-norm. If you wish to standardize, please use

se.

Methods

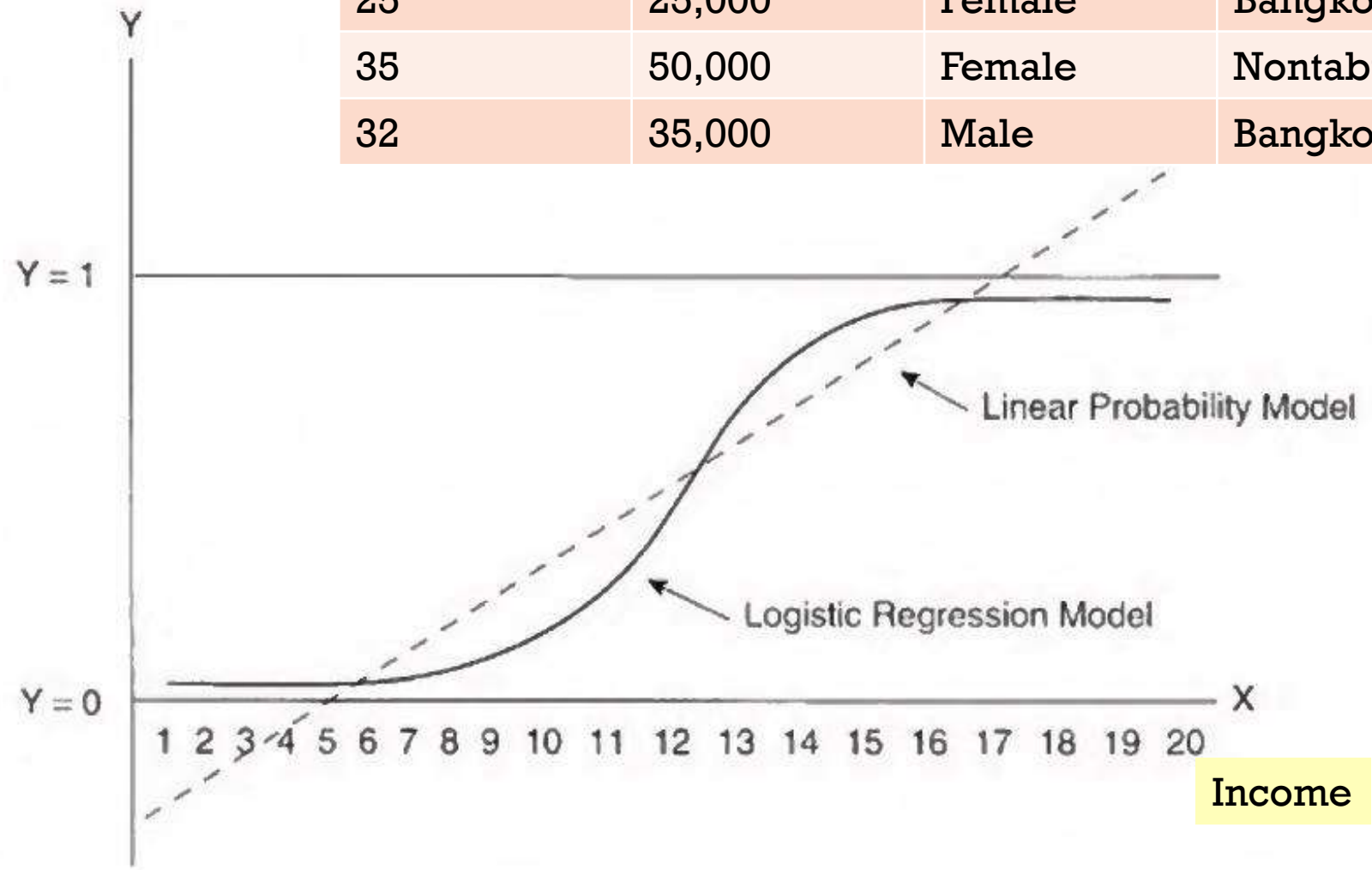
fit(self, X, y[, sample_weight])	Fit linear model.
get_params(self[, deep])	Get parameters for this estimator.
predict(self, X)	Predict using the linear model.
score(self, X, y[, sample_weight])	Return the coefficient of determination R^2 of the prediction.
set_params(self, **params)	Set the parameters of this estimator.

fficient
ocessors.

Classification Problem

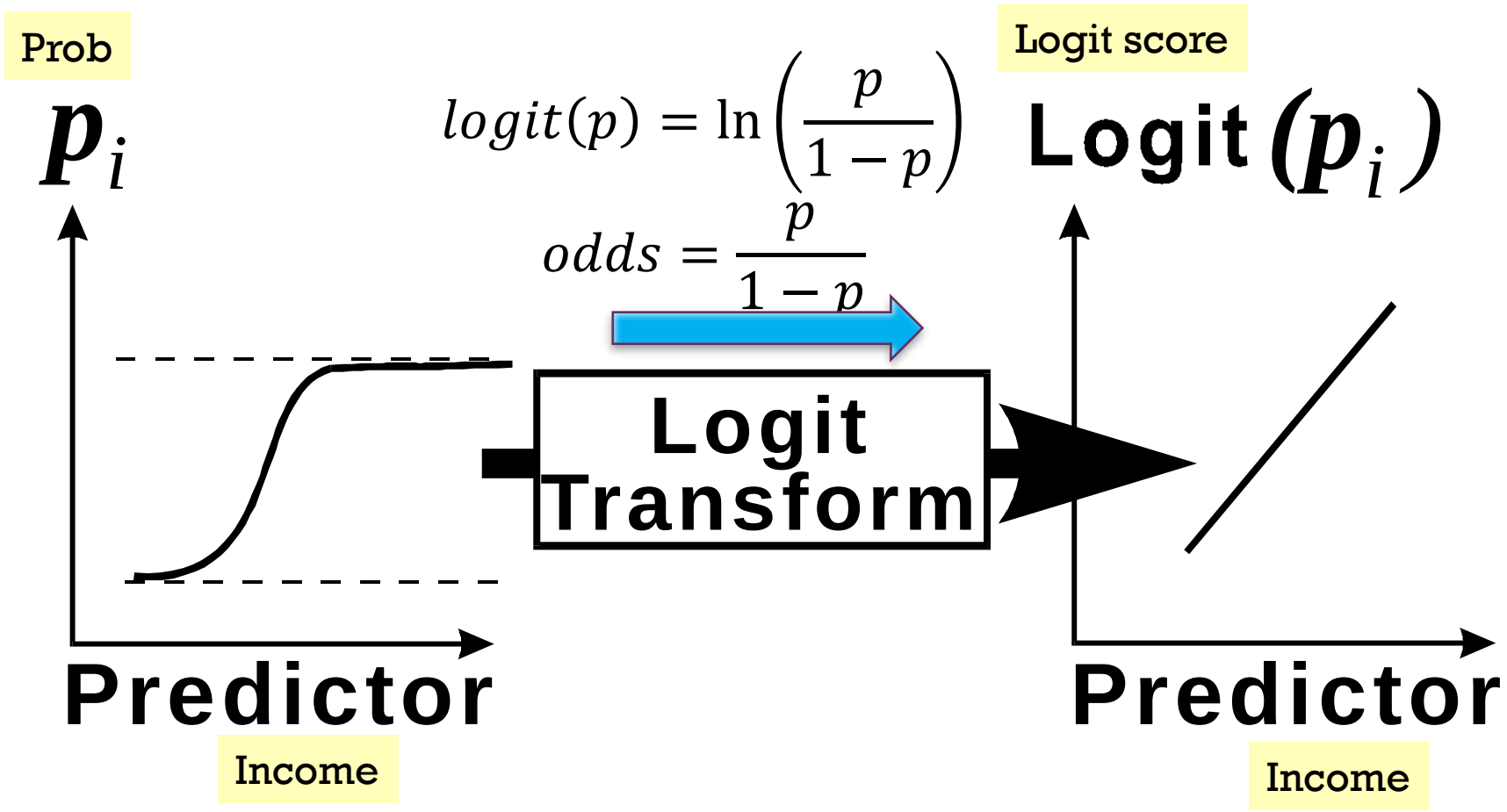
Prob. of purchase

Age	Income	Gender	Province	Purchase
25	25,000	Female	Bangkok	Yes (1)
35	50,000	Female	Nontaburi	Yes (1)
32	35,000	Male	Bangkok	No (0)



Logistic function
Sigmoid function

Logistic Regression (forward pass)
Logit link function: Non-linear to Linear



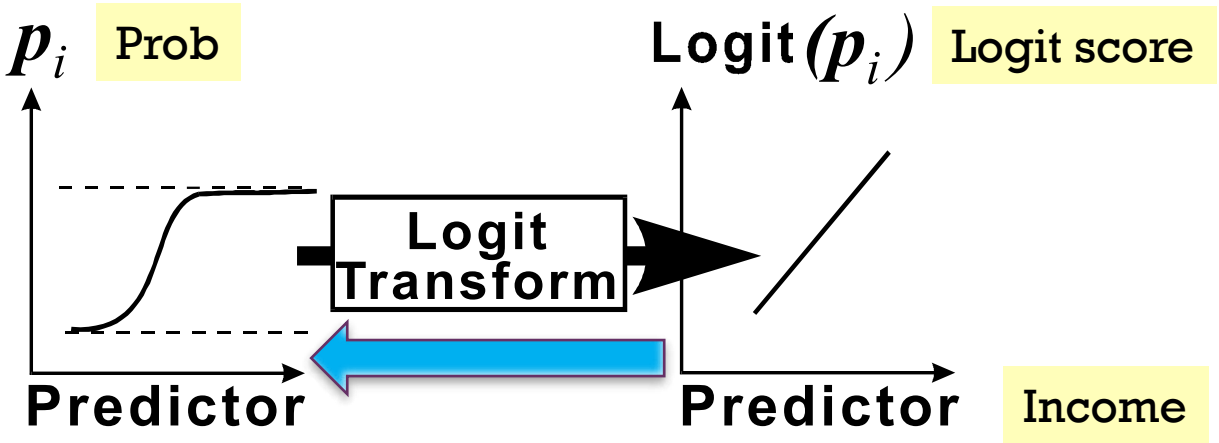
Logit Link Function (backward pass)

$$\log\left(\frac{\hat{p}}{1-\hat{p}}\right) = \hat{w}_0 + \hat{w}_1 x_1 + \hat{w}_2 x_2 = \text{logit}(\hat{p})$$

- Maximum likelihood estimates

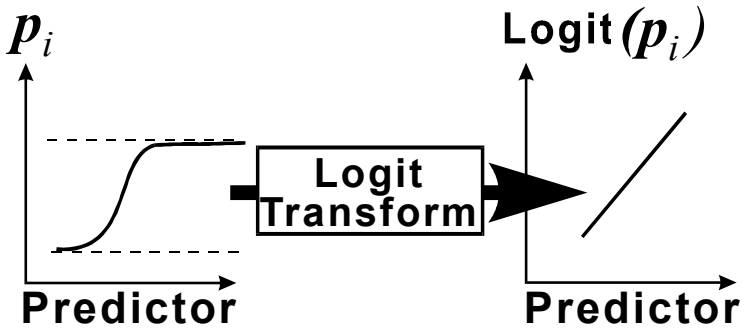
$$\hat{p} = \frac{1}{1 + e^{-\text{logit}(\hat{p})}}$$

To obtain prediction estimates, the logit equation is solved for $\hat{p}$.



...

Quiz

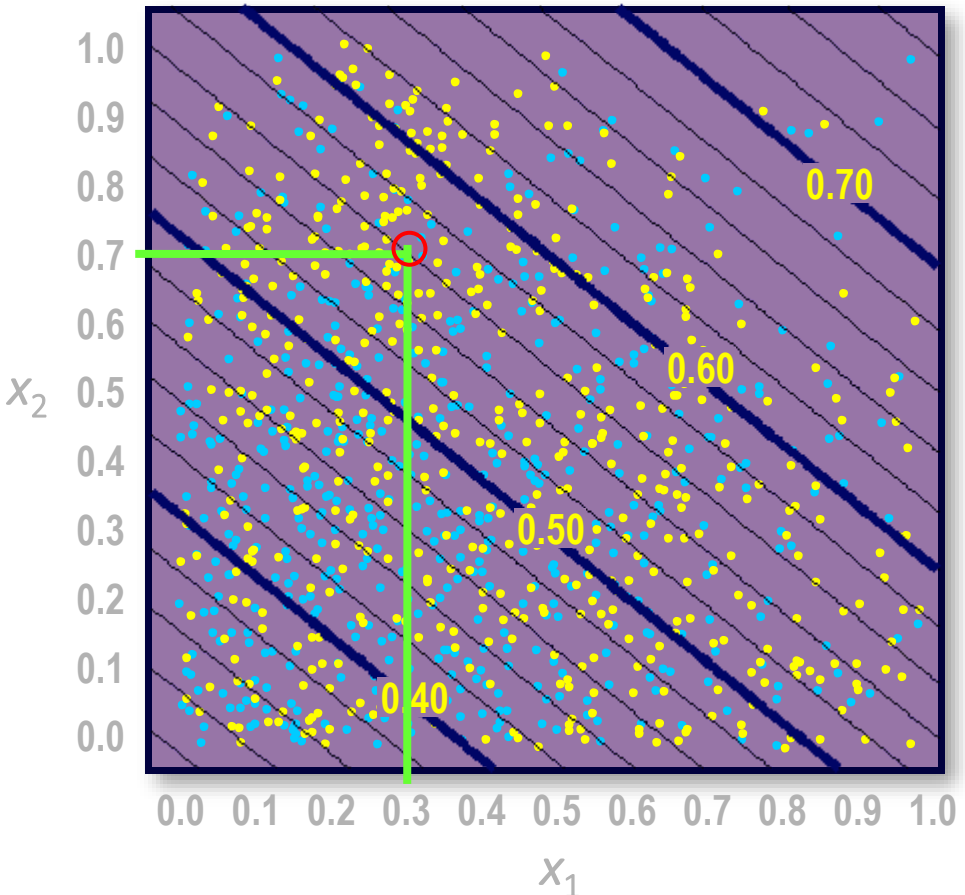


- What is the logistic regression prediction for the indicated point?

a. 0.243
b. 0.56
c. yellow
d. It depends.

$$\text{logit}(\hat{p}) = -0.81 + 0.92\,x_1 + 1.11\,x_2$$

$$\hat{p} = \frac{1}{1 + e^{-\text{logit}(\hat{p})}}$$



sklearn.linear_model.LogisticRegression

class sklearn.linear_model.**LogisticRegression**(*penalty='l2', dual=False, tol=0.0001, C=1.0, fit_intercept=True, intercept_scaling=1, class_weight=None, random_state=None, solver='lbfgs', max_iter=100, multi_class='auto', verbose=0, warm_start=False, n_jobs=None, l1_ratio=None*) ¶ [source]

Logistic Regression (aka logit, MaxEnt) classifier.

In the multiclass case, the training algorithm uses the one-vs-rest (OvR) scheme if the 'multi_class' option is set to 'ovr', and uses the cross-entropy loss if the 'multi_class' option is set to 'multinomial'. (Currently the 'multinomial' option is supported only by the 'lbfgs', 'sag', 'saga' and 'newton-cg' solvers.)

This class implements regularized logistic regression using **regularization is applied by default**. It can handle both c 64-bit floats for optimal performance; any other input forn

The 'newton-cg', 'sag', and 'lbfgs' solvers support only L2 solver supports both L1 and L2 regularization, with a dual supported by the 'saga' solver.

Read more in the User Guide.

Examples

```
>>> from sklearn.datasets import load_iris
>>> from sklearn.linear_model import LogisticRegression
>>> X, y = load_iris(return_X_y=True)
>>> clf = LogisticRegression(random_state=0).fit(X, y)
>>> clf.predict(X[:2, :])
array([0, 0])
>>> clf.predict_proba(X[:2, :])
array([[9.8...e-01, 1.8...e-02, 1.4...e-08],
       [9.7...e-01, 2.8...e-02, ...e-08]])
>>> clf.score(X, y)
0.97...
```

Parameters:

penalty : *{'l1', 'l2', 'elasticnet', 'none'}*, default='l2'

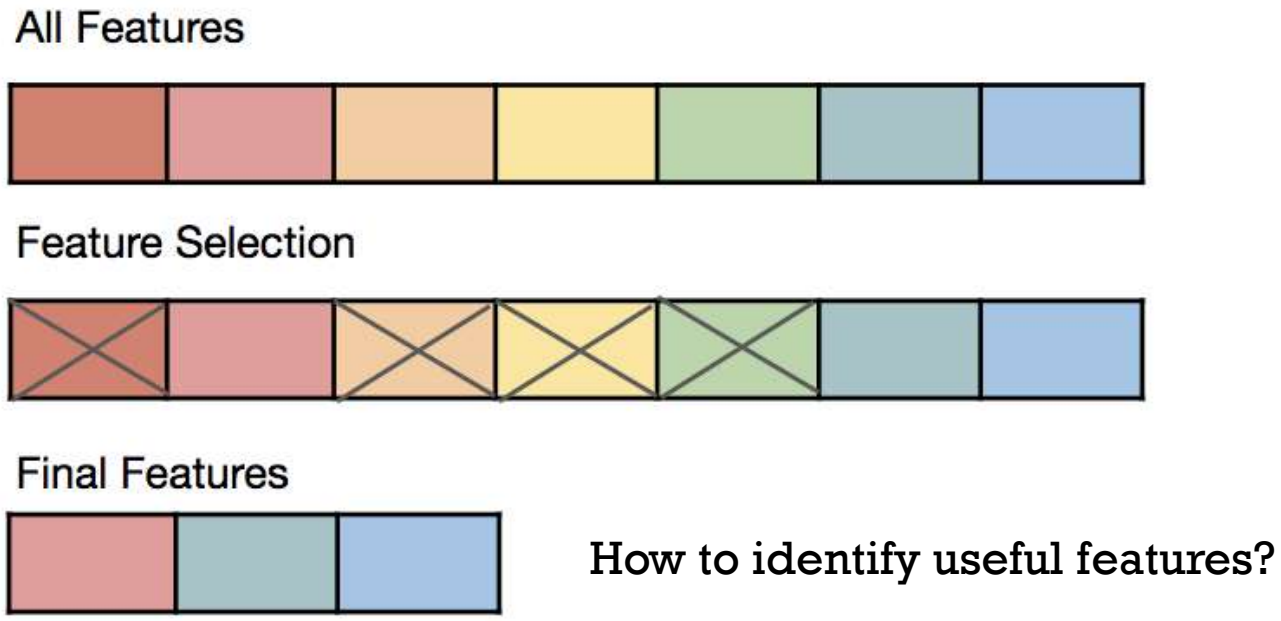
Used to specify the norm used in the penalization. The 'newton-cg', 'sag' and 'lbfgs' solvers support only l2 penalties. 'elasticnet' is only supported by the 'saga' solver. If 'none' (not supported by the liblinear solver), no regularization is applied.

New in version 0.19: l1 penalty with SAGA solver (allowing 'multinomial' + L1)

https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

Features Selection

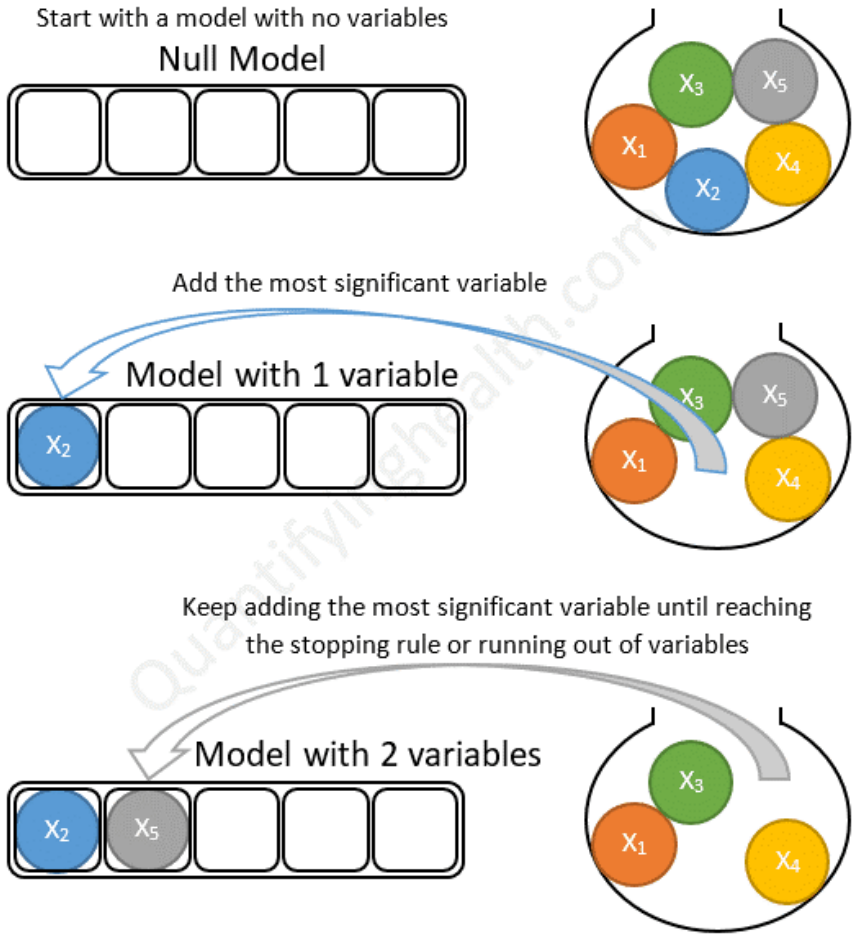
- Feature Selection is the process where you automatically or manually select those features which contribute most to your prediction variable or output in which you are interested in.



How to identify useful features?

Forward Stepwise

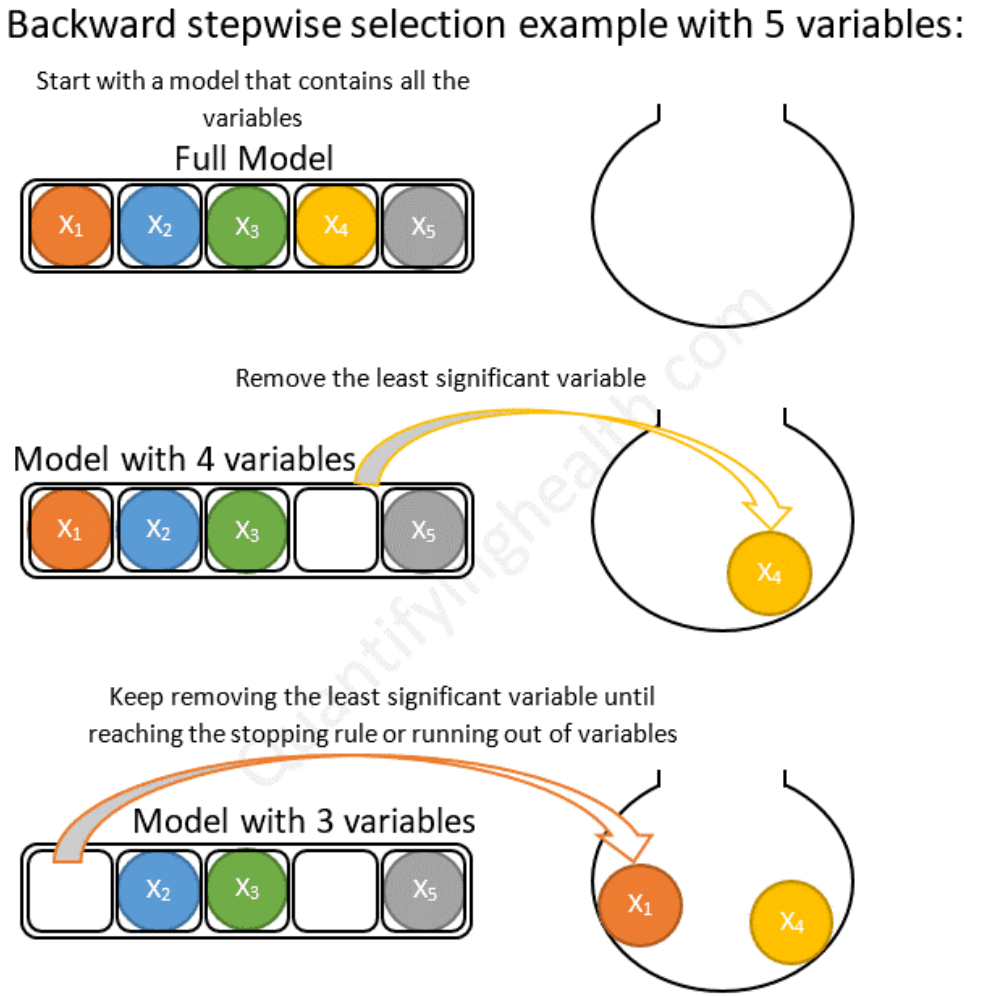
Forward stepwise selection example with 5 variables:



- Work better in large features but small samples.

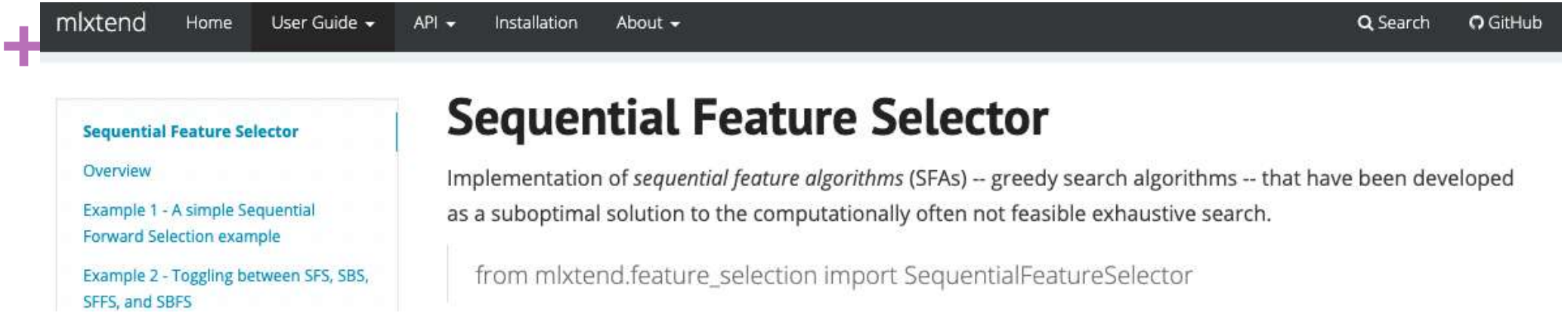
Source: https://quantifyinghealth.com/stepwise-selection/

Backward Stepwise



Source: https://quantifyinghealth.com/stepwise-selection/

- Work better when variables in a model are correlated which each other.



- Sequential feature selection algorithms are a family of greedy search algorithms that are used to reduce an initial d-dimensional feature space to a k-dimensional feature subspace where $k < d$.
- In a nutshell, SFAs remove or add one feature at the time based on the classifier performance (such as, accuracy) until a feature subset of the desired size k is reached.
- Sequential Forward Selection (SFS)
- Sequential Backward Selection (SBS)
- Sequential Forward Floating Selection (SFFS)
- Sequential Backward Floating Selection (SBFS)

- `scoring` : str, callable, or None (default: None)

  If None (default), uses 'accuracy' for sklearn classifiers and 'r2' for sklearn regressors. If str, uses a sklearn scoring metric string identifier, for example {accuracy, f1, precision, recall, roc_auc} for classifiers, {'mean_absolute_error', 'mean_squared_error'/'neg_mean_squared_error', 'median_absolute_error', 'r2'} for regressors. If a callable object or function is provided, it has to be conform with sklearn's signature `scorer(estimator, X, y)`; see http://scikit-learn.org/stable/modules/generated/sklearn.metrics.make_scorer.html for more information.

- `cv` : int (default: 5)

  Integer or iterable yielding train, test splits. If cv is an integer and `estimator` is a classifier (or y consists of integer class labels) stratified k-fold. Otherwise regular k-fold cross-validation is performed. No cross-validation if cv is None, False, or 0.

http://rasbt.github.io/mlxtend/user_guide/feature_selection/SequentialFeatureSelector/

Seqeuntial Feature Selector (cont.)

- The ***floating*** variants, SFFS and SBFS, can be considered as extensions to the simpler SFS and SBS algorithms.
- The dimensionality of the subset during the search can be thought to be "floating" up and down. (somewhat similar to stepwise)
  - Sequential Forward Floating Selection (SFFS)
  - Sequential Backward Floating Selection (SBFS)

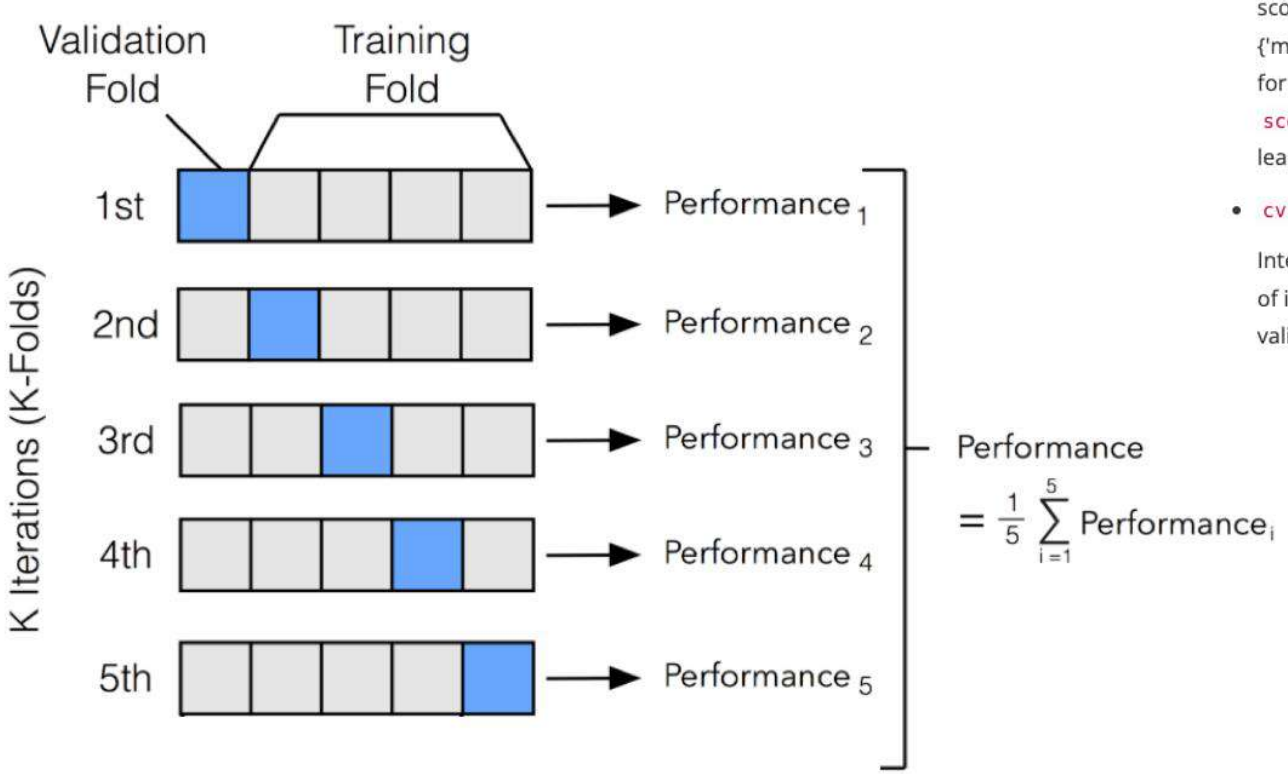
```python
# Build RF classifier to use in feature selection
clf = LogisticRegression()

# Build step forward feature selection
sfs1 = sfs(clf,
           k_features=5,
           forward=True,
           floating=False,
           verbose=2,
           scoring='accuracy',
           cv=5)

# Perform SFFS
sfs1 = sfs1.fit(bank_final, y)
```

```python
# Which features?
feat_cols = list(sfs1.k_feature_idx_)
print(feat_cols)
```

```
[6, 10, 15, 16, 17]
```



- `scoring` : str, callable, or None (default: None)

  If None (default), uses 'accuracy' for sklearn classifiers and 'r2' for sklearn regressors. If str, uses a sklearn scoring metric string identifier, for example {accuracy, f1, precision, recall, roc_auc} for classifiers, {'mean_absolute_error', 'mean_squared_error'/'neg_mean_squared_error', 'median_absolute_error', 'r2'} for regressors. If a callable object or function is provided, it has to be conform with sklearn's signature `scorer(estimator, X, y)`; see http://scikit-learn.org/stable/modules/generated/sklearn.metrics.make_scorer.html for more information.

- `cv` : int (default: 5)

  Integer or iterable yielding train, test splits. If cv is an integer and `estimator` is a classifier (or y consists of integer class labels) stratified k-fold. Otherwise regular k-fold cross-validation is performed. No cross-validation if cv is None, False, or 0.

Now it is available in scikit-learn now.

Prev Up Next
sklearn.feature_selection.RFE
Other versions

Please cite us if you use the software.

sklearn.feature_selection.SequentialFeatureSelector

class `sklearn.feature_selection.`**`SequentialFeatureSelector`**(*estimator*, ***, *n_features_to_select='warn'*, *tol=None*, *direction='forward'*, *scoring=None*, *cv=5*, *n_jobs=None*) [source]

Transformer that performs Sequential Feature Selection.

This Sequential Feature Selector adds (forward selection) or removes (backward selection) features to form a feature subset in a greedy fashion. At each stage, this estimator chooses the best feature to add or remove based on the cross-validation score of an estimator. In the case of unsupervised learning, this Sequential Feature Selector looks only at the features (X), not the desired outputs (y).

Read more in the User Guide.

New in version 0.24.

Parameters:

estimator : *estimator instance*
An unfitted estimator.

n_features_to_select : *"auto", int or float, default='warn'*
If `"auto"`, the behaviour depends on the `tol` parameter:

tol : *float, default=None*
If the score is not incremented by at least `tol` between two consecutive feature additions or removals, stop adding or removing. `tol` is enabled only when `n_features_to_select` is `"auto"`.

New in version 1.1.

direction : *{'forward', 'backward'}, default='forward'*
Whether to perform forward selection or backward selection.

scoring : *str or callable, default=None*
A single str (see The scoring parameter: defining model evaluation rules) or a callable (see Defining your scoring strategy from metric functions) to evaluate the predictions on the test set.

NOTE that when using a custom scorer, it should return a single value.

If None, the estimator's score method is used.

cv : *int, cross-validation generator or an iterable, default=None*
Determines the cross-validation splitting strategy. Possible inputs for cv are:

https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SequentialFeatureSelector.html#sklearn.feature_selection.SequentialFeatureSelector.

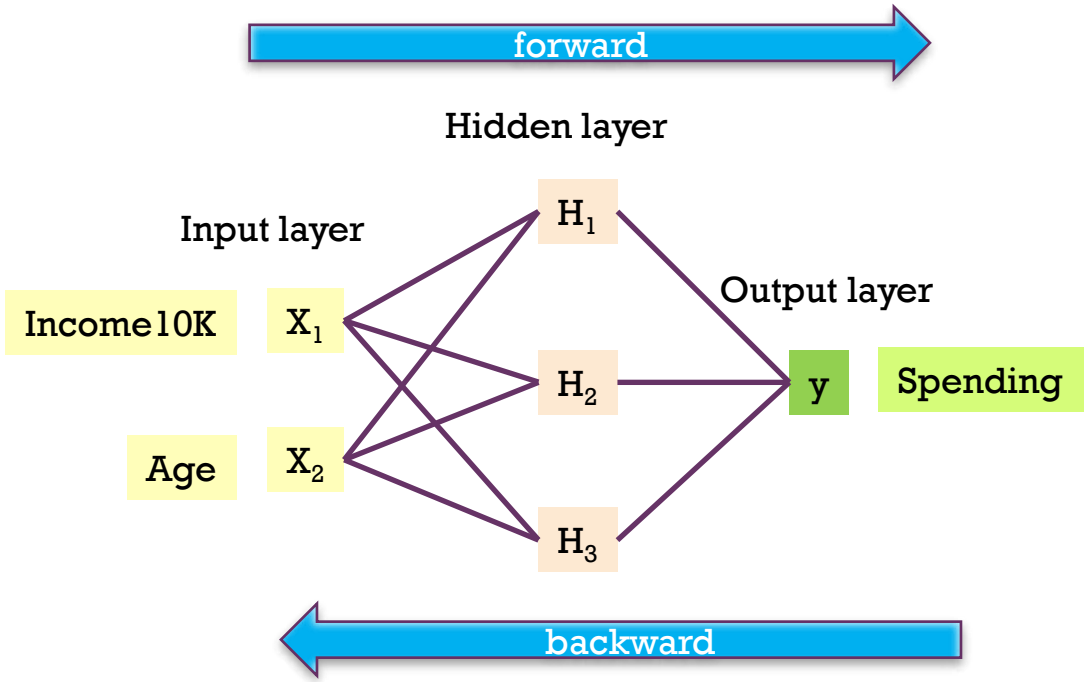
Examples

```
>>> from sklearn.feature_selection import SequentialFeatureSelector
>>> from sklearn.neighbors import KNeighborsClassifier
>>> from sklearn.datasets import load_iris
>>> X, y = load_iris(return_X_y=True)
>>> knn = KNeighborsClassifier(n_neighbors=3)
>>> sfs = SequentialFeatureSelector(knn, n_features_to_select=3)
>>> sfs.fit(X, y)
SequentialFeatureSelector(estimator=KNeighborsClassifier(n_neighbors=3),
                          n_features_to_select=3)
>>> sfs.get_support()
array([ True, False,  True,  True])
>>> sfs.transform(X).shape
(150, 3)
```

+

Neural Networks

Neural Networks

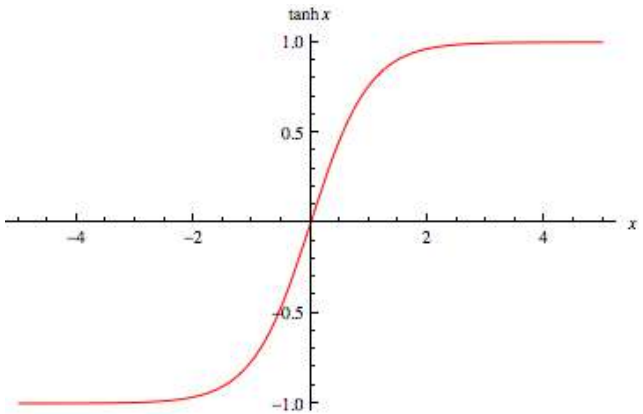


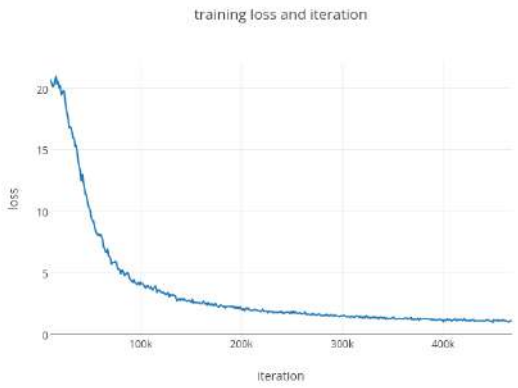
$$log\left(\frac{\hat{p}}{1-\hat{p}}\right) = \hat{w}_0 + \hat{w}_1 H_1 + \hat{w}_2 H_2 + \hat{w}_3 H_3$$

$$H_1 = tanh(\hat{w}_{10} + \hat{w}_{11} x_1 + \hat{w}_{12} x_2)$$

$$H_2 = tanh(\hat{w}_{20} + \hat{w}_{21} x_1 + \hat{w}_{22} x_2)$$

$$H_3 = tanh(\hat{w}_{30} + \hat{w}_{31} x_1 + \hat{w}_{32} x_2)$$





How to update weight
https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/

Prev Up Next

scikit-learn 0.22.2
Other versions

Please cite us if you use the software.

sklearn.neural_network.MLPClassifier
Examples using sklearn.neural_network.MLPClass

sklearn.neural_network.MLPClassifier

class sklearn.neural_network.**MLPClassifier**(*hidden_layer_sizes=(100,), activation='relu', solver='adam', alpha=0.0001, batch_size='auto', learning_rate='constant', learning_rate_init=0.001, power_t=0.5, max_iter=200, shuffle=True, random_state=None, tol=0.0001, verbose=False, warm_start=False, momentum=0.9, nesterovs_momentum=True, early_stopping=False, validation_fraction=0.1, beta_1=0.9, beta_2=0.999, epsilon=1e-08, n_iter_no_change=10, max_fun=15000*) ¶ [source]

Multi-layer Perceptron classifier.

This model optimizes the log-loss function using LBFGS or stochastic gradient descent.

New in version 0.18.

Parameters:

hidden_layer_sizes : *tuple, length = n_layers - 2, default=(100,)*
The ith element represents the number of neurons in the ith hidden layer.

activation : *{'identity', 'logistic', 'tanh', 'relu'}, default='relu'*
Activation function for the hidden layer.

- 'identity', no-op activation, useful to implement linear bottleneck, returns f(x) = x
- 'logistic', the logistic sigmoid function, returns f(x) = 1 / (1 + exp(-x)).
- 'tanh', the hyperbolic tan function, returns f(x) = tanh(x).
- 'relu', the rectified linear unit function, returns f(x) = max(0, x)

solver : *{'lbfgs', 'sgd', 'adam'}, default='adam'*
The solver for weight optimization.

sklearn.neural_network.MLPRegressor

class `sklearn.neural_network.` **`MLPRegressor`**(*hidden_layer_sizes=(100,), activation='relu', solver='adam', alpha=0.0001, batch_size='auto', learning_rate='constant', learning_rate_init=0.001, power_t=0.5, max_iter=200, shuffle=True, random_state=None, tol=0.0001, verbose=False, warm_start=False, momentum=0.9, nesterovs_momentum=True, early_stopping=False, validation_fraction=0.1, beta_1=0.9, beta_2=0.999, epsilon=1e-08, n_iter_no_change=10, max_fun=15000*) ¶ [source]

Multi-layer Perceptron regressor.

This model optimizes the squared-loss using LBFGS or stochastic gradient descent.

New in version 0.18.

Parameters:

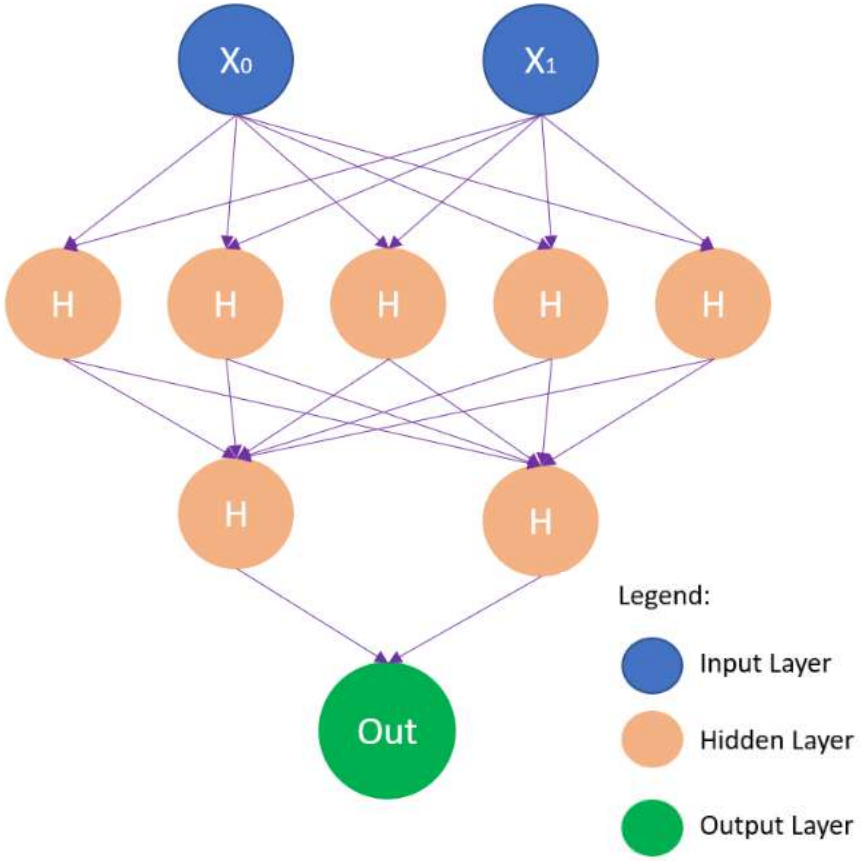
hidden_layer_sizes : *tuple, length = n_layers - 2, default=(100,)*
The ith element represents the number of neurons in the ith hidden layer.

activation : *{'identity', 'logistic', 'tanh', 'relu'}, default='relu'*
Activation function for the hidden layer.

- 'identity', no-op activation, useful to implement linear bottleneck, returns f(x) = x
- 'logistic', the logistic sigmoid function, returns f(x) = 1 / (1 + exp(-x)).
- 'tanh', the hyperbolic tan function, returns f(x) = tanh(x).
- 'relu', the rectified linear unit function, returns f(x) = max(0, x)

solver : *{'lbfgs', 'sgd', 'adam'}, default='adam'*
The solver for weight optimization.

Exmaple



batch_size : *int, default='auto'*

Size of minibatches for stochastic optimizers. If the solver is 'lbfgs', the classifier will not use minibatch. When set to "auto", `batch_size=min(200, n_samples)`.

learning_rate : *{'constant', 'invscaling', 'adaptive'}, default='constant'*

Learning rate schedule for weight updates.

```
mlp_clf = MLPClassifier(hidden_layer_sizes=(5,2),
                        max_iter = 300,activation = 'relu',
                        solver = 'adam')
```

```
param_grid = {
    'hidden_layer_sizes': [(150,100,50), (120,80,40), (100,50
    'max_iter': [50, 100, 150],
    'activation': ['tanh', 'relu'],
    'solver': ['sgd', 'adam'],
    'alpha': [0.0001, 0.05],
    'learning_rate': ['constant','adaptive'],
}
```

```
grid = GridSearchCV(mlp_clf, param_grid, n_jobs= -1, cv=5)
grid.fit(trainX_scaled, trainY)


print(grid.best_params_)
```

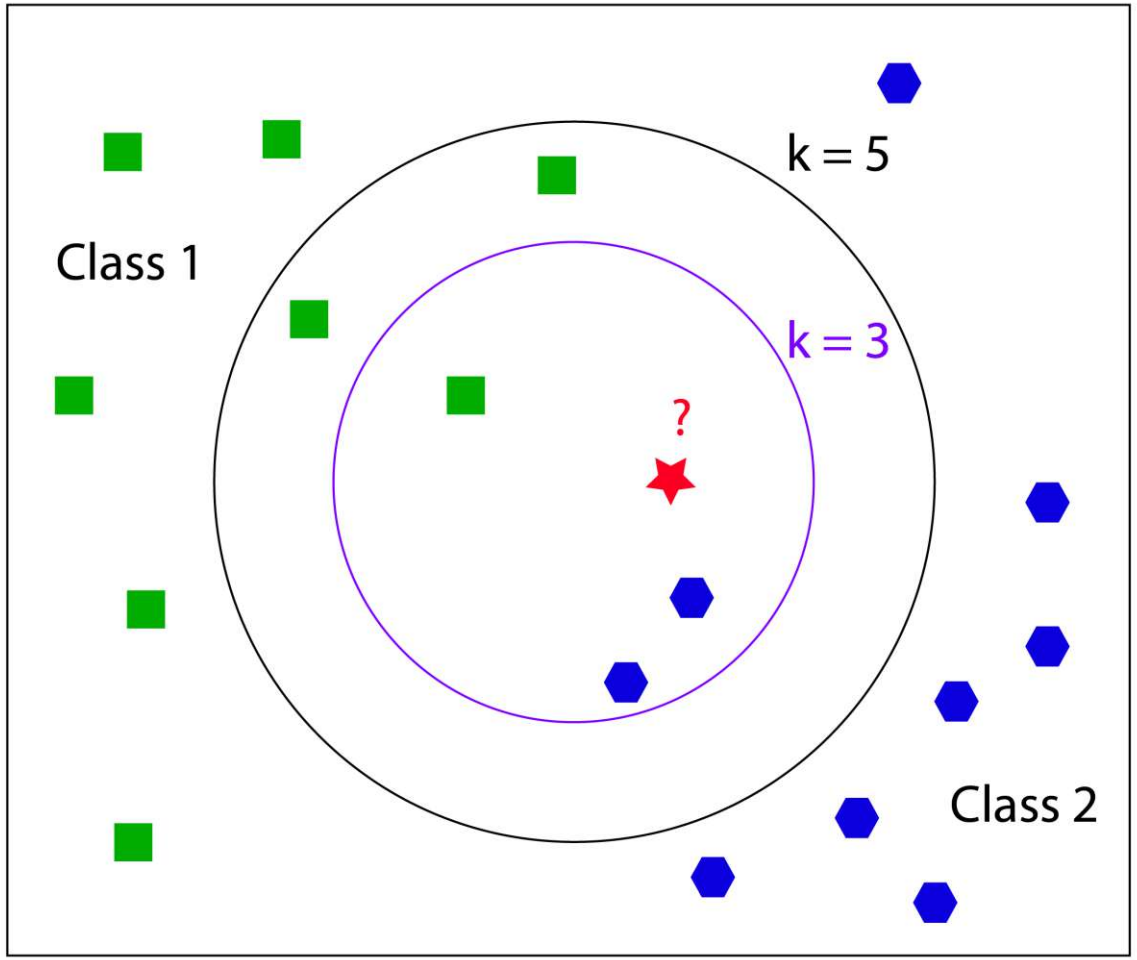
https://michael-fuchs-python.netlify.app/2021/02/03/nn-multi-layer-perceptron-classifier-mlpclassifier/

K-Nearest Neighbors

k-Nearest Neighbors (kNN)

Important Params:
- k
- Distance function

- Memory based learning
- Suitable for small data sets
- Merge
  - Voting
  - Average
  - Maximum prob
- Cautions:
  - Support only numerical variables
  - Need to adjust variable range



Distance Function

- Euclidean Distance

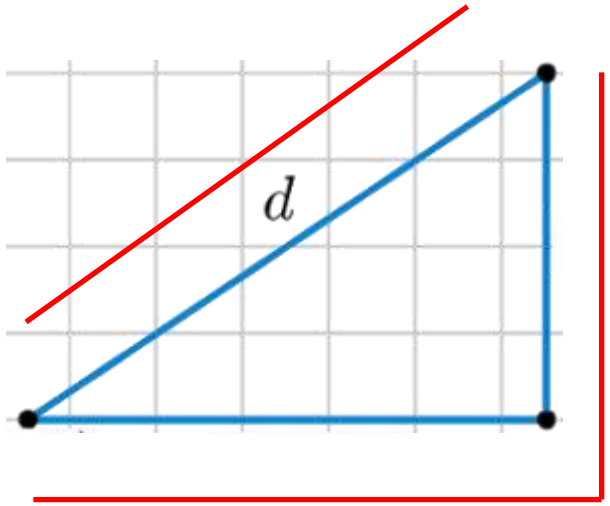
$$\sqrt{\sum_{i=1}^{k}(x_i - y_i)^2}$$

- Manhattan Distance

$$\sum_{i=1}^{k}|x_i - y_i|$$

- Minkowski Distance

$$\left(\sum_{i=1}^{k}(|x_i - y_i|)^q\right)^{1/q}$$



q = 1 is Manhattan Distance
q = 2 is Euclidean Distance

Distance Function (cont.)

- Hamming distance: same (0) vs diff (1)
- It still supports only numeric variable.

$$D_H = \sum_{i=1}^{k} |x_i - y_i|$$

$$x = y \Rightarrow D = 0$$

$$x \neq y \Rightarrow D = 1$$

X	Y	Distance
Male	Male	0
Male	Female	1

Nearest Neighbors

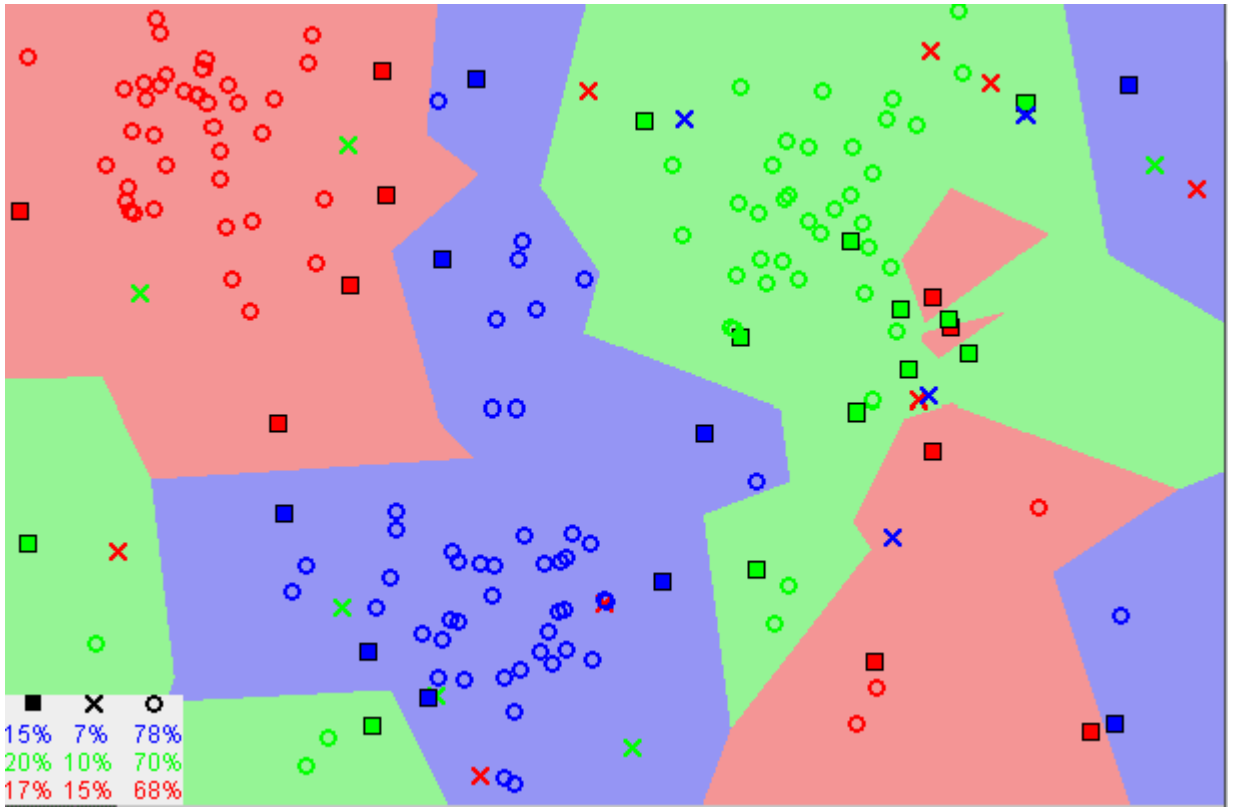


Image showing how similar data points typically exist close to each other

Source : https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761

sklearn.neighbors.KNeighborsClassifier

class `sklearn.neighbors.`**`KNeighborsClassifier`**(*n_neighbors=5, weights='uniform', algorithm='auto', leaf_size=30, p=2, metric='minkowski', metric_params=None, n_jobs=None, **kwargs*) [source]

Classifier implementing the k-nearest neighbors vote.

Read more in the User Guide.

Parameters:

n_neighbors : *int, optional (default = 5)*
Number of neighbors to use by default for `kneighbors` queries.

weights : *str or callable, optional (default = 'uniform')*
weight function used in prediction. Possible values:

- 'uniform' : uniform weights. All points in each neighborhood are weighted equally.
- 'distance' : weight points by the inverse of their distance. in this case, closer neighbors of a query point will have a greater influence than neighbors which are further away.
- [callable] : a user-defined function which accepts an array of distances, and returns an array of the same shape containing the weights.

algorithm : *{'auto', 'ball_tree', 'kd_tree', 'brute'}, optional*
Algorithm used to compute the nearest neighbors:

- 'ball_tree' will use `BallTree`
- 'kd_tree' will use `KDTree`
- 'brute' will use a brute-force search.
- 'auto' will attempt to decide the most appropriate algorithm based on the values passed to `fit` method

sklearn.neighbors.KNeighborsRegressor

class sklearn.neighbors.**KNeighborsRegressor**(*n_neighbors=5*, *weights='uniform'*, *algorithm='auto'*, *leaf_size=30*, *p=2*, *metric='minkowski'*, *metric_params=None*, *n_jobs=None*, ***kwargs*) [sourc

Regression based on k-nearest neighbors.

The target is predicted by local interpolation of the targets associated of the nearest neighbors in the training set.

Read more in the User Guide.

New in version 0.9.

Parameters:

n_neighbors : *int, optional (default = 5)*
Number of neighbors to use by default for kneighbors queries.

weights : *str or callable*
weight function used in prediction. Possible values:

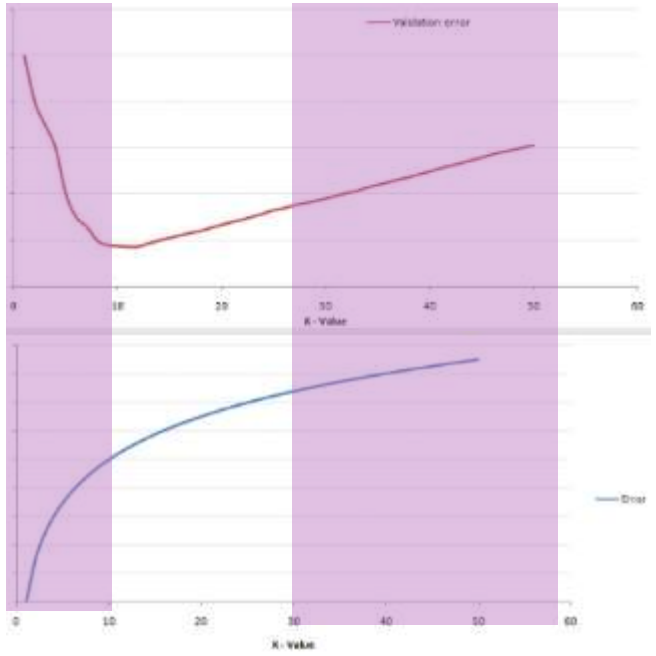
- 'uniform' : uniform weights. All points in each neighborhood are weighted equally.
- 'distance' : weight points by the inverse of their distance. in this case, closer neighbors of a query point will have a greater influence than neighbors which are further away.
- [callable] : a user-defined function which accepts an array of distances, and returns an array of the same shape containing the weights.

Uniform weights are used by default.

Choosing the right value of K

Too Underfitted

Too Overfitted



Search the best parameters with …

sklearn.model_selection.GridSearchCV

class sklearn.model_selection.**GridSearchCV**(*estimator, param_grid, *, scoring=None, n_jobs=None, iid='deprecated', refit=True, cv=None, verbose=0, pre_dispatch='2*n_jobs', error_score=nan, return_train_score=False*) [source]

sklearn.model_selection.RandomizedSearchCV

class sklearn.model_selection.**RandomizedSearchCV**(*estimator, param_distributions, *, n_iter=10, scoring=None, n_jobs=None, iid='deprecated', refit=True, cv=None, verbose=0, pre_dispatch='2*n_jobs', random_state=None, error_score=nan, return_train_score=False*) [source]

```
param_grid=dict(
    n_neighbors=[1,2,3,4,5,6,7,8,9,10],
    weights=['uniform', 'distance'],
),
```

Search with k: 1-10
on uniform/distance weights

+

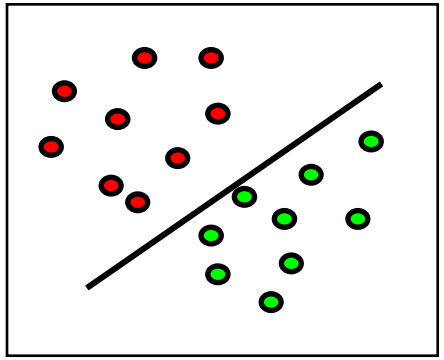
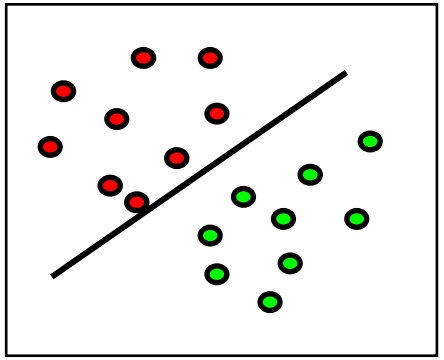
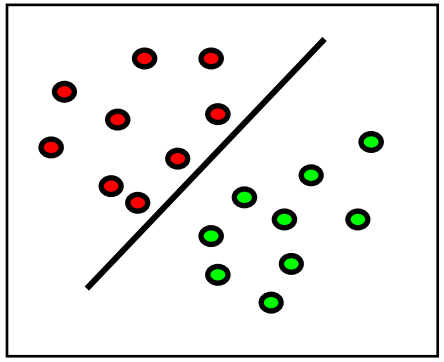
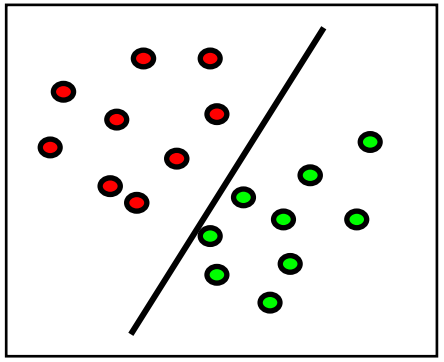
Support Vector Machine

Adapt from ML Class by David Sontag, New York University

https://people.csail.mit.edu/dsontag/courses/ml16/slides/

Linear Separators

- If training data is linearly separable, perceptron is guaranteed to find some linear separator
- Which of these is optimal?

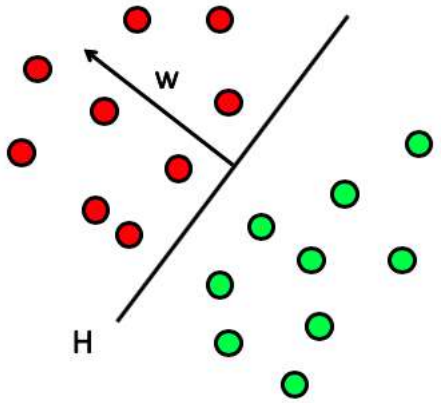





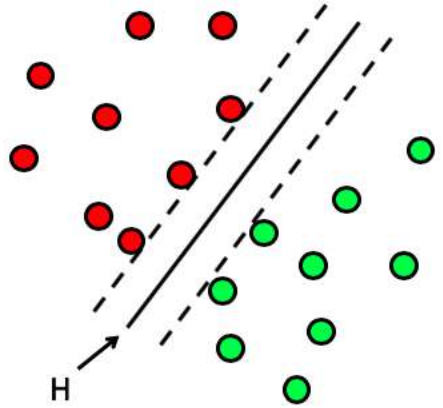
Support Vector Machine (SVM)

- SVMs (Vapnik, 1990's) choose the linear separator with the largest margin

$$H = \{x \mid \langle w, x \rangle + b = 0\}$$

Dot product



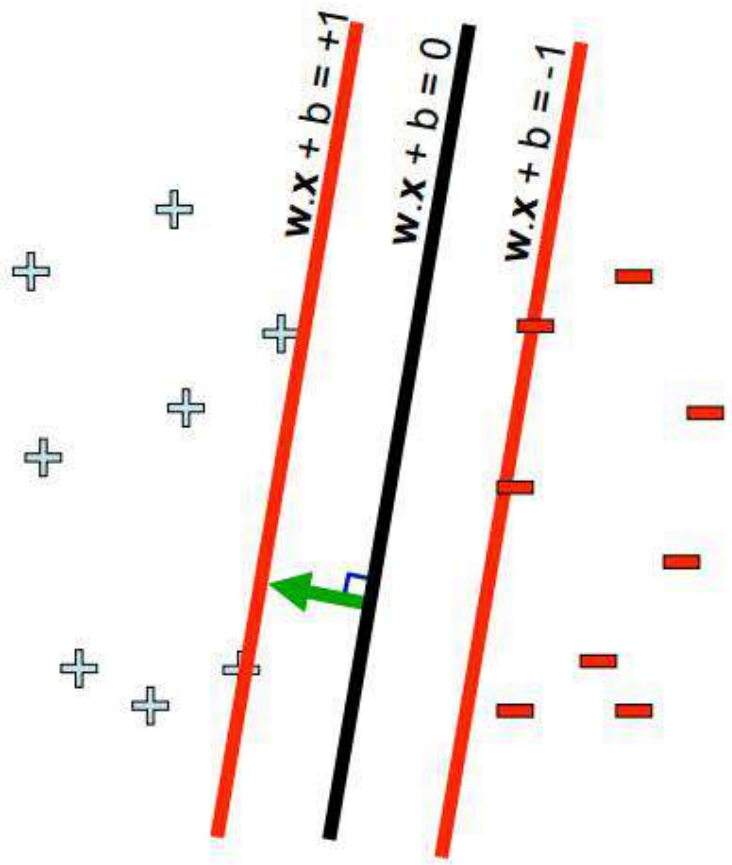


- Good according to intuition, theory, practice
- SVM became famous when, using images as input, it gave accuracy comparable to neural-network with hand-designed features in a handwriting recognition task

Support vector machines: 3 key ideas

1. Use **optimization** to find solution (i.e., a hyperplane) with few errors
2. Seek **large margin** separator to improve generalization
3. Use **kernel trick** to make large feature spaces computationally efficient

Key 1: Optimization



For every data point (x_t, y_t), enforce the constraint

for $y_t = +1$, $w \cdot x_t + b \geq 1$

and for $y_t = -1$, $w \cdot x_t + b \leq -1$

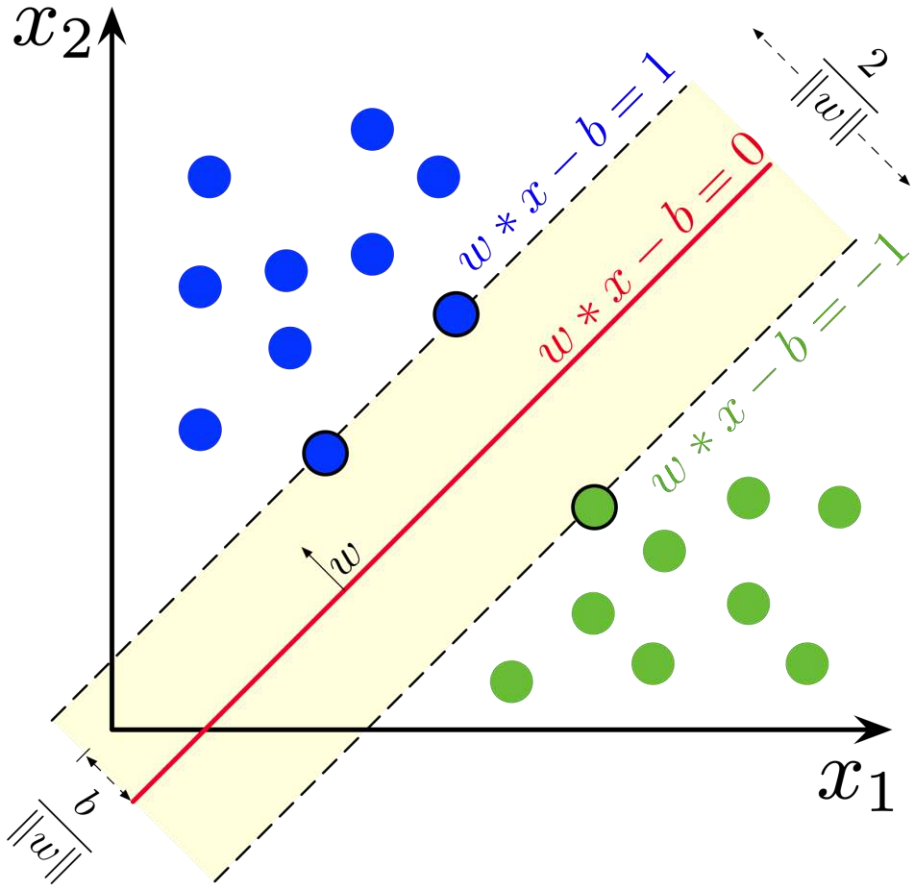
Equivalently, we want to satisfy all of the linear constraints

$$y_t \left(w \cdot x_t + b \right) \geq 1 \quad \forall t$$

This *linear program* can be efficiently solved using algorithms such as simplex, interior point, or ellipsoid

Key 2: Seek Large Margin Hard-margin SVM

- Allow for slack to improve generalization.

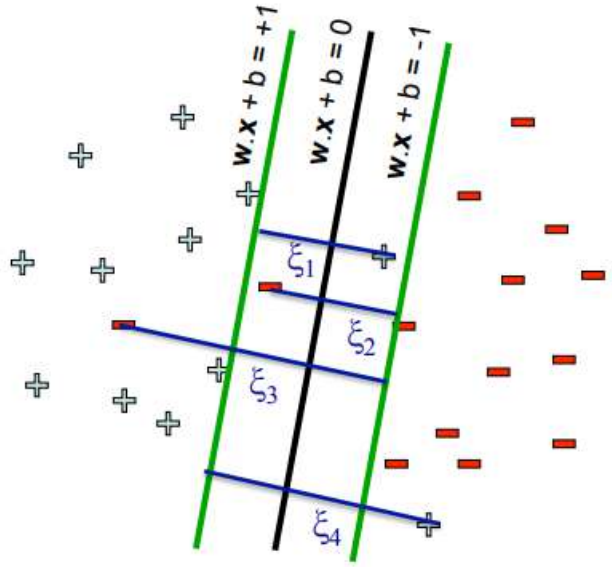


$$y_t \left(w \cdot x_t + b \right) \geq 1 \quad \forall t$$

Minimize $\| w \|^2$

Key 2: Seek Large Margin Soft-margin SVM

- Allow for slack to improve generalization.



$$\text{minimize}_{\mathbf{w},b,\xi} \quad \Sigma_j \xi_j$$

$$\left(\mathbf{w}.\mathbf{x}_j + b\right) y_j \geq 1 - \xi_j \quad , \forall j \quad \xi_j \geq 0$$

↑ "slack variables"

We now have a linear program again, and can efficiently find its optimum

For each data point:

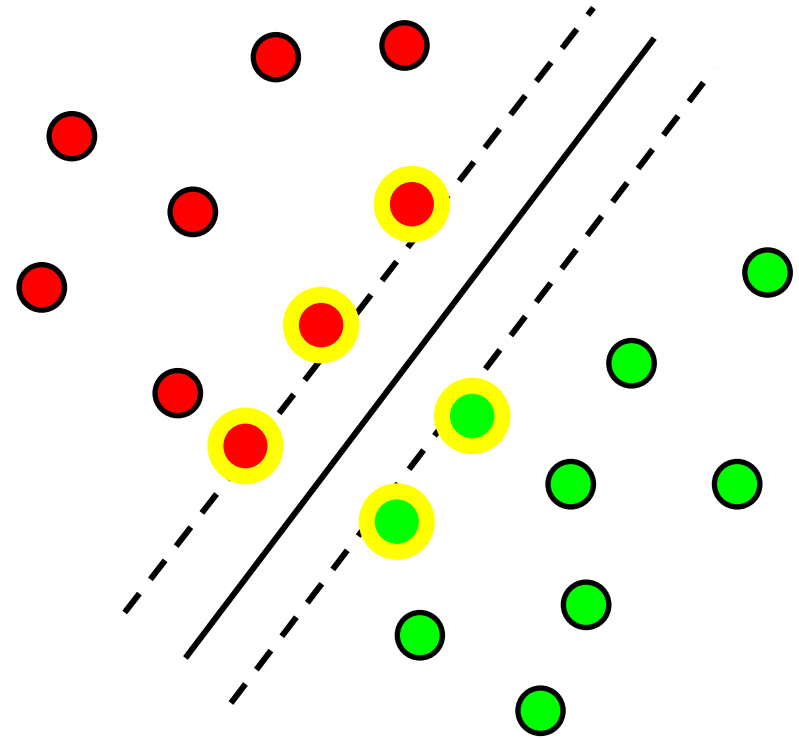
- If functional margin ≥ 1, don't care
- If functional margin < 1, pay linear penalty

Minimize $\| w \|^2 + C \cdot \sum_i \xi_i$

What Are the Support Vectors?

- "Carrying vectors"
- The points, located closest to the hyperplane
- Determining the location of the hyperplane
- All other data points have $\alpha_i = 0$.

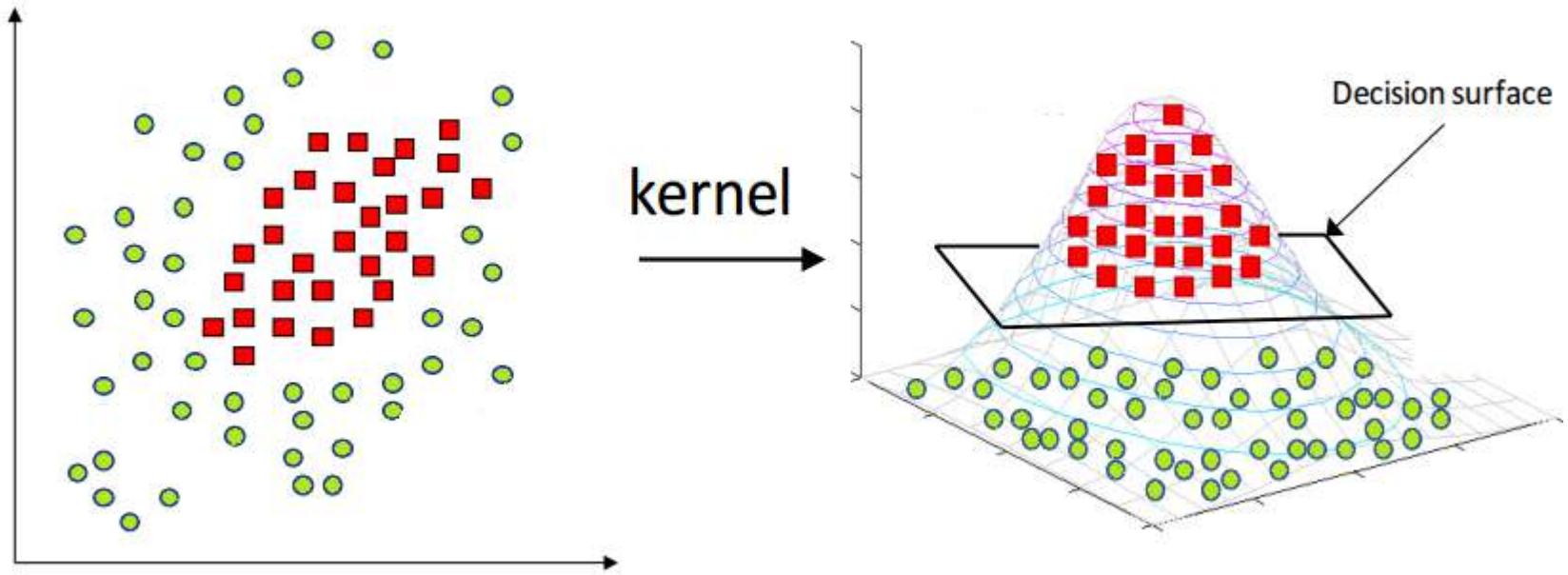
$$w = \sum_{i=1}^{\#sv} \alpha_i y_i x_i^{sv}$$



Prediction = sign(w*x + b)

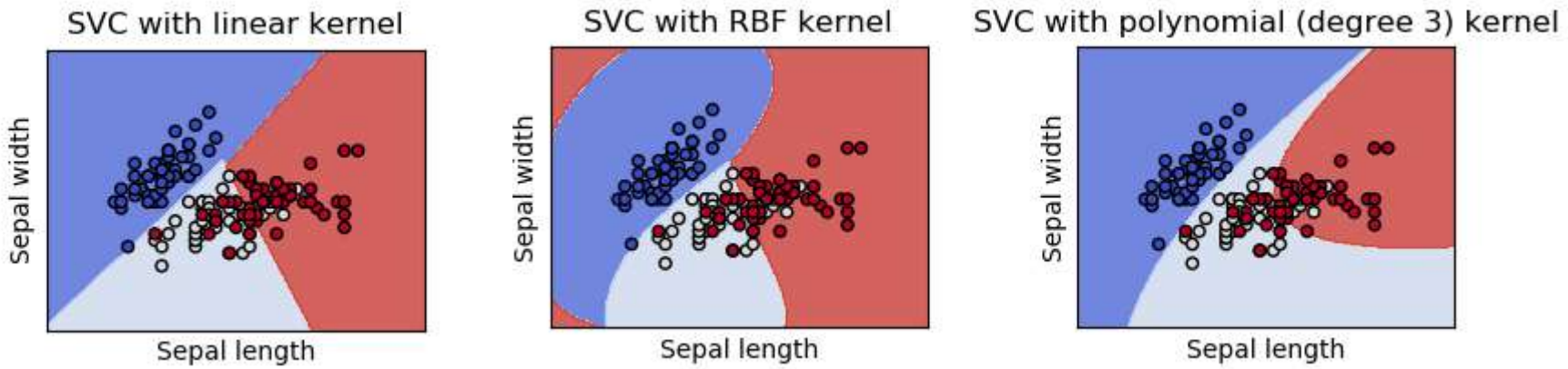
Key 3: Kernel

- Kernel allows us to operate in the original feature space without computing the coordinates of the data in a higher dimensional space.



Source : https://medium.com/@zxr.nju/what-is-the-kernel-trick-why-is-it-important-98a98db0961d

Different Kernel on iris dataset



$$w = \sum_{i=1}^{\#sv} \alpha_i y_i x_i^{sv}$$

RBF: Radial Basis Function

$$K(x_i, x_j) = e^{-\gamma(x_i - x_j)^2}$$

Polynomial

$$K(x_i, x_j) = (x_i \cdot x_j + 1)^p$$

Prediction = sign(w*x + b)

https://scikit-learn.org/stable/auto_examples/svm/plot_iris_svc.html

SVM in sklearn (Classifier)

sklearn.svm.SVC

class sklearn.svm.**SVC**(*, C=1.0, kernel='rbf', degree=3, gamma='scale', coef0=0.0, shrinking=True, probability=False, tol=0.001, cache_size=200, class_weight=None, verbose=False, max_iter=-1, decision_function_shape='ovr', break_ties=False, random_state=None) [source]

C-Support Vector Classification.

- Parameters
  - **kernel** : Can be 'linear', 'poly', 'rbf', 'sigmoid' or callable function (custom).
  - **C** : For regularization. A larger value creates a larger of margin hyperplane.
  - **gamma** : Determine how model fit to training data.

Use cases

ระบบแบบจำลองเพื่อประเมินราคาอสังหาริมทรัพย์ (Land appraisal)

The Treasury Department

ขอนแก่น

สระบุรี

แปลงที่ดิน

แปลงที่ดิน

ปทุมธานี

นนทบุรี

ห้องชุด

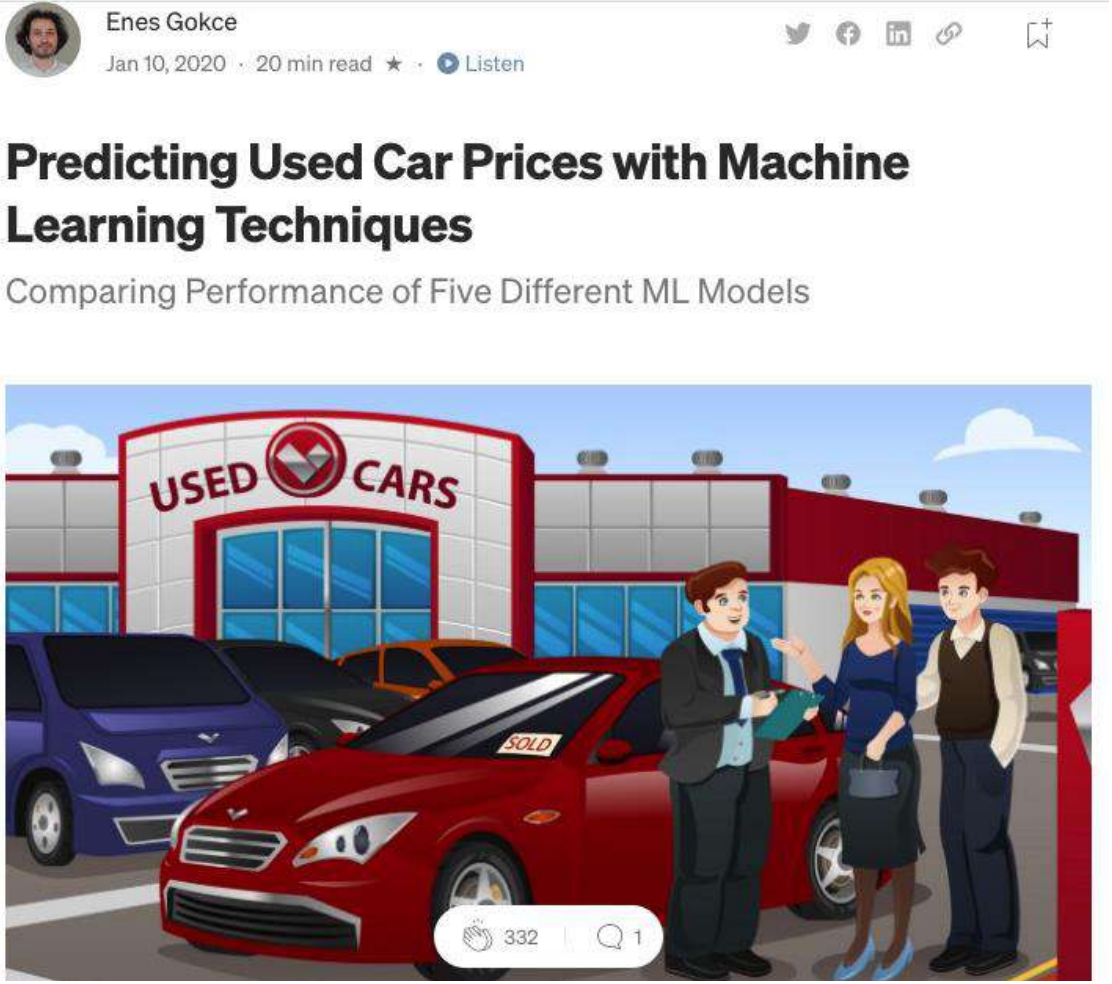


Table 15

Comparison of Model Outcomes

Measure (%) / Model	Random Forest	Ridge	Lasso	KNN	XGBoost
Mean Absolute Error	2047.74	5405.78	5406.47	6257.07	6257.07
Mean Squared Error	15682494.73	57436987.51	57438444.92	94069138.93	94069138.93
RMSE	3960.11	7578.72	7578.82	9698.92	9698.92

Table 1

Missing Values in Car Dataset

Feature	Null Values	Percent
size	366256	67
condition	250074	45
cylinders	218997	40
paint_color	180021	33
drive	165838	30
type	159179	29
odometer	110800	20
manufacturer	26915	5
make	9677	2
fuel	4741	1
title_status	4024	1
transmission	4055	1
year	1487	0
price	0	0
Total	1502064	

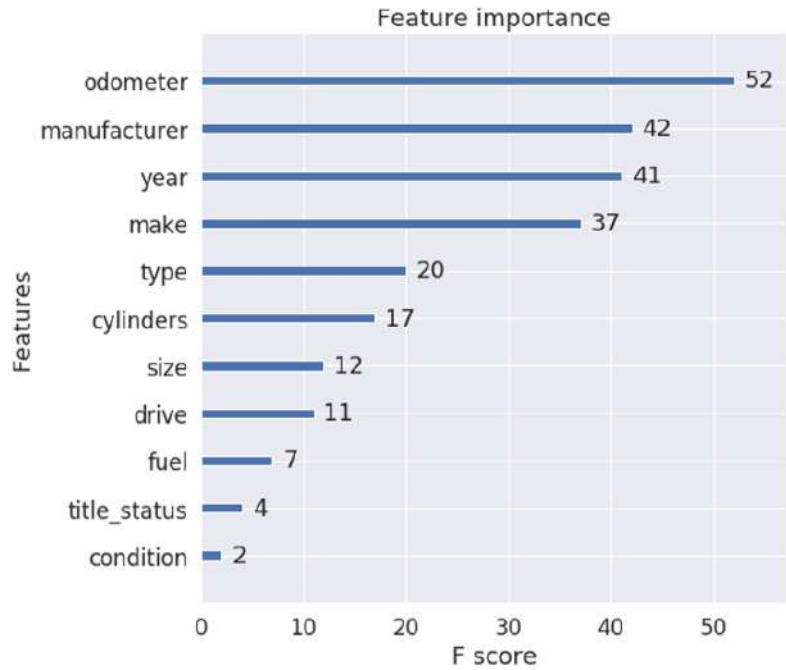


Figure 21: XGBoost Plot for Feature Importance

https://towardsdatascience.com/predicting-used-car-prices-with-machine-learning-techniques-8a9d8313952

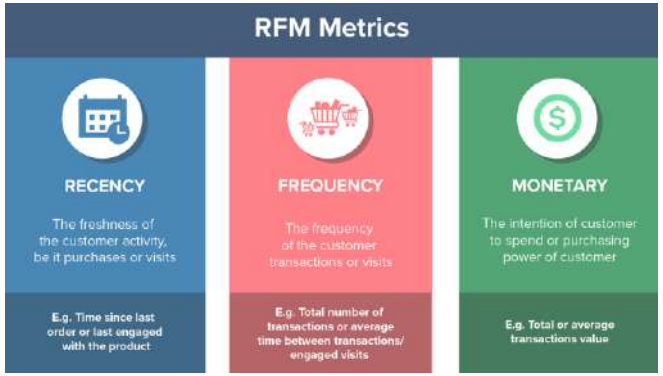
Direct Target Marketing

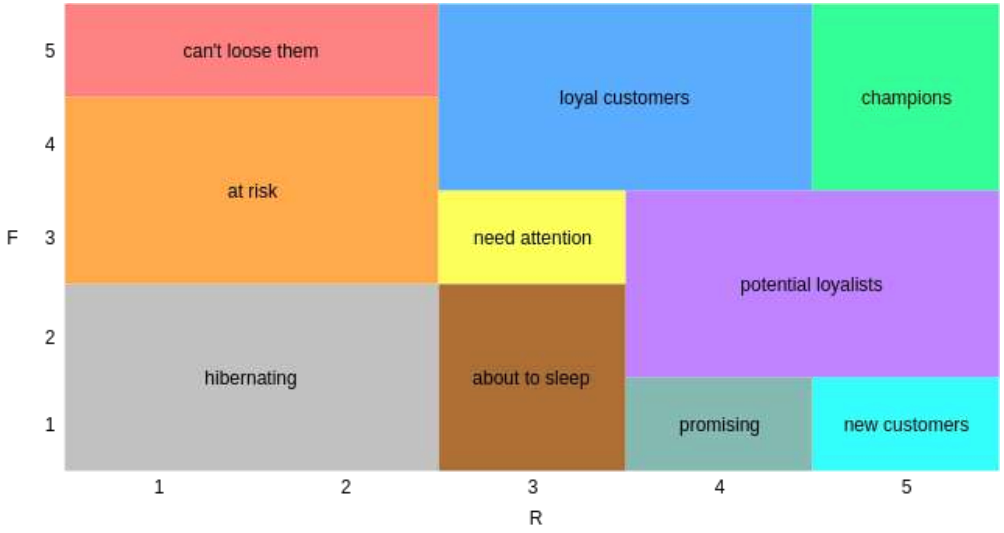
Training Data

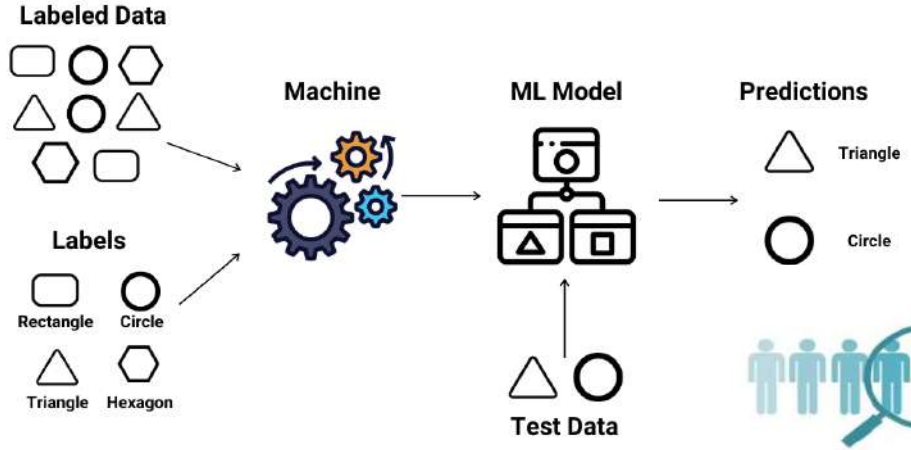
inputs				target
Age	**Income**	**Gender**	**Province**	**Purchase**
25	25,000	Female	Bangkok	Yes
35	50,000	Female	Nontaburi	Yes
32	35,000	Male	Bangkok	No

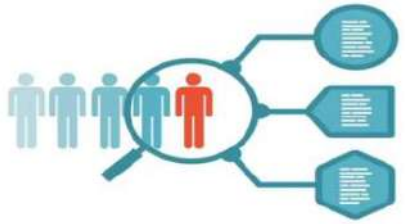
Testing Data

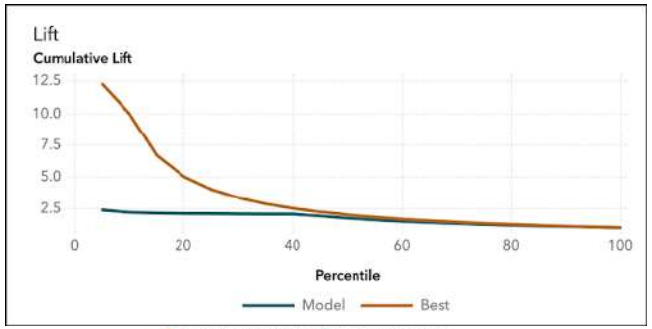
Age	**Income**	**Gender**	**Province**	**Purchase**
25	25,000	Female	Bangkok	?

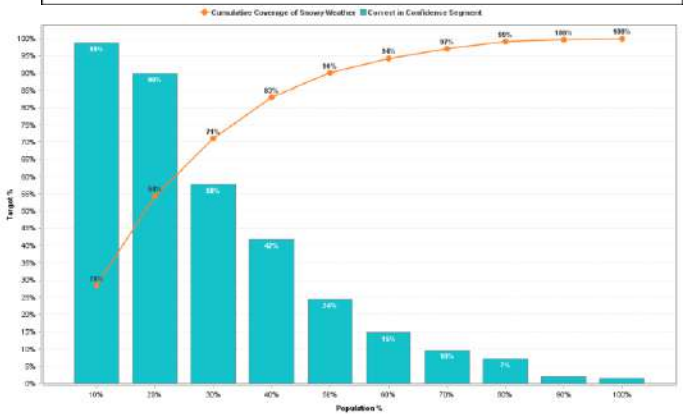


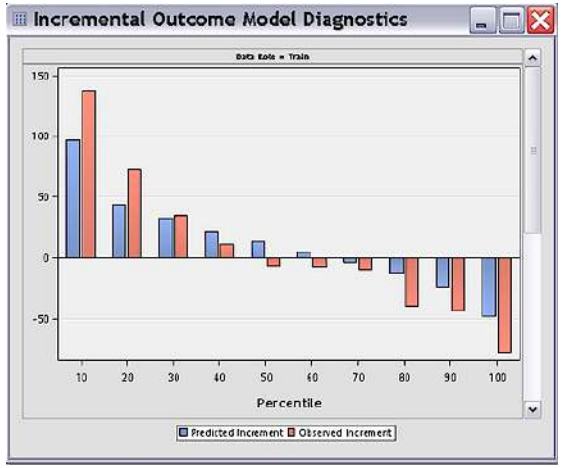












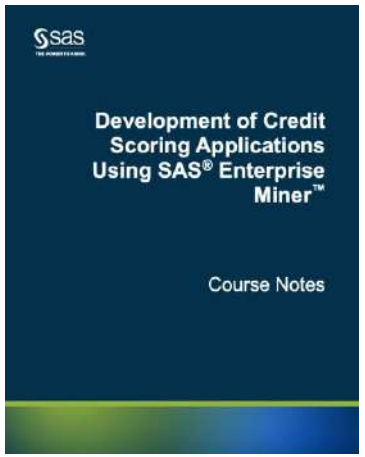
Credit Scoring (Logistic Regression)

- Let cutoff = 500

A new customer applies for credit.

AGE	**32**	**120 points**
HOME	**OWN**	**225 points**
SALARY	**$30,000**	**180 points**
Total		**525 points**

ACCEPT FOR CREDIT



Characteristic	Attribute	Scorecard Points
AGE	<25	100
AGE	25<=AGE<33	120
AGE	33<=AGE<45	185
AGE	>=45	200
HOME	OWN	225
HOME	RENT	110
INCOME	<10000	120
INCOME	10000<=INCOME<25000	140
INCOME	25000<=INCOME<35000	180
INCOME	35000<=INCOME<50000	200
INCOME	>=50000	225