Northwestern Kellogg

Northwestern ENGINEERING

MBAi 448 I Winter 2026

AI Fundamentals

Alex Castrounis

www.whyofai.com | linkedin.com/in/alexcastrounis

Attendance

Enter the “Magic Word” in Canvas

Do Not Share with Absent Students

Image via ChatGPT

Non-Parametric Learning

K-nearest Neighbors

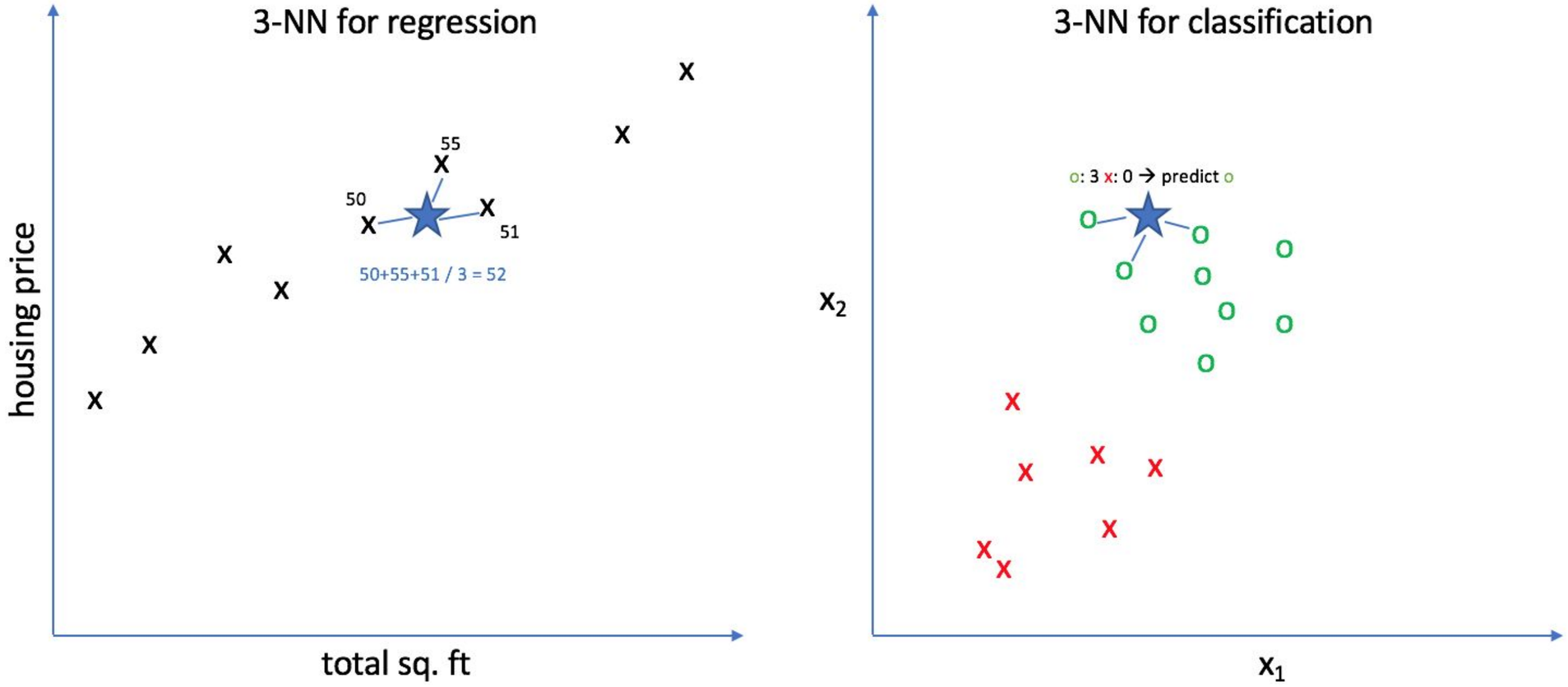
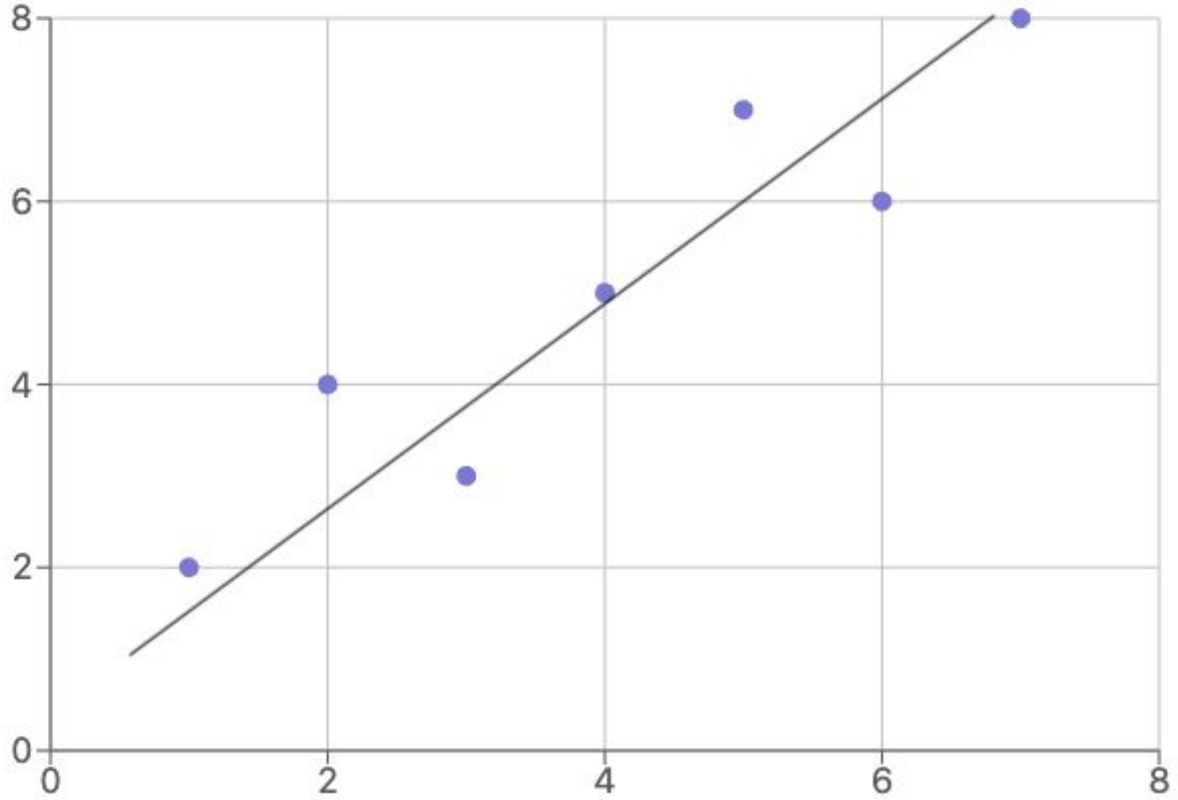


Image credit: Jeremy Jordan

Parametric Learning

$$y = mx + b$$



Machine Learning Types

Supervised | Labeled

Unsupervised | Unlabeled

Semi-supervised | Labeled + Unlabeled

Self-supervised | Unlabeled

Reinforcement (aka post-training) | NA

Transfer (pre-training + fine-tuning) | NA

In-Context | N-shot

The Machine Learning Recipe

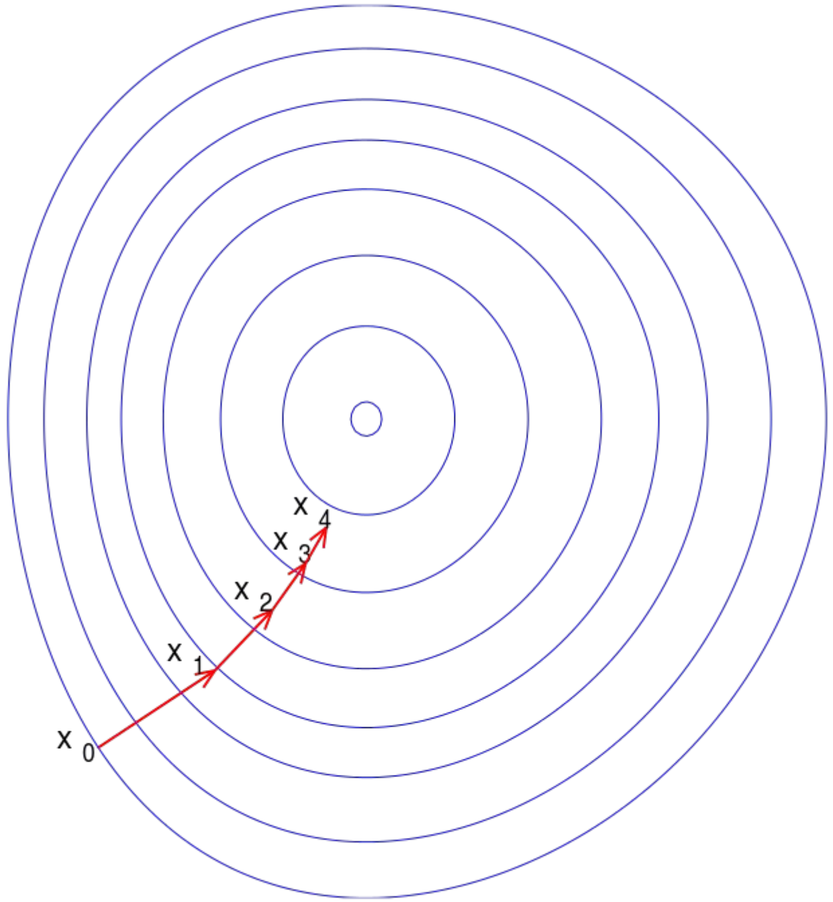
- Choose or define a target model
- Define a loss (error) function
  - Ex: squared error (MSE)
- Define a cost (objective) function to optimize
  - This also defines the optimization criterion
  - Ex: mean squared error (MSE)
- Optimize the cost function
  - Using training data and an optimization routine

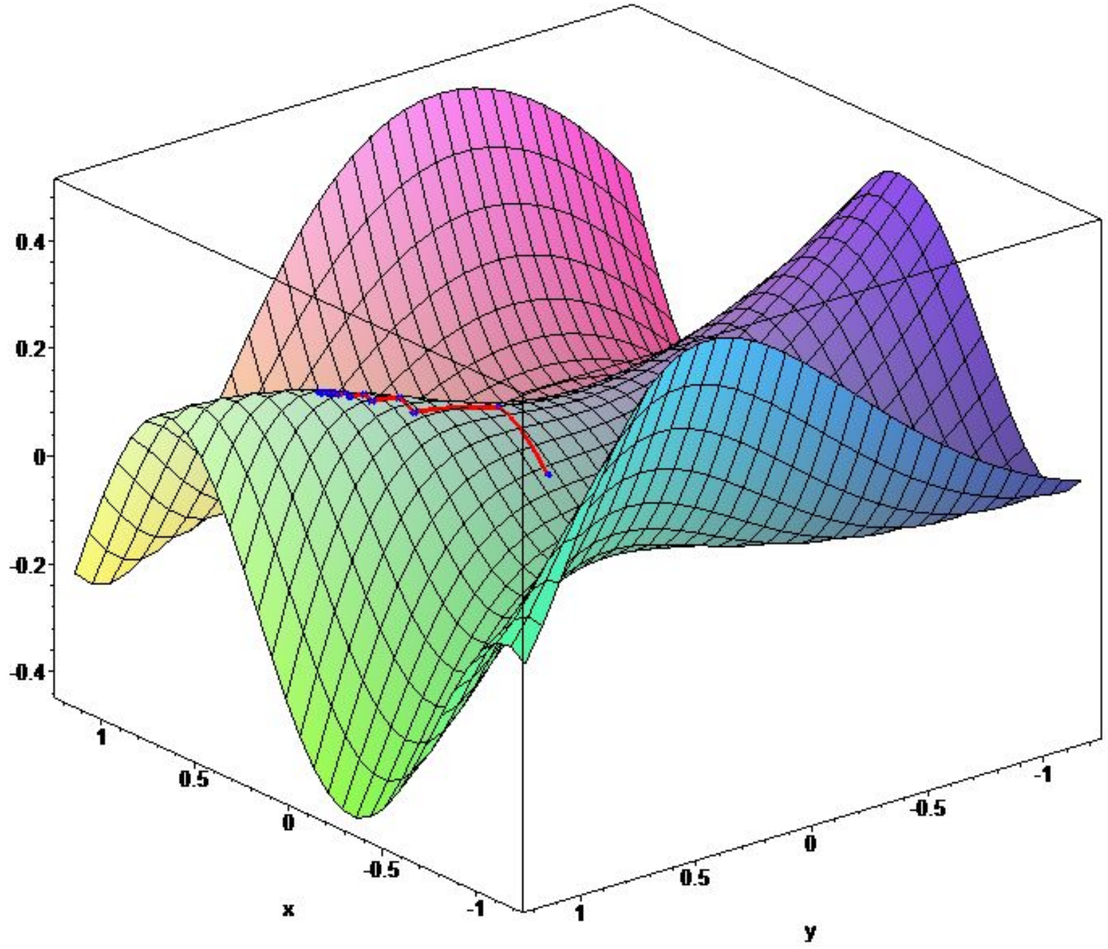
Optimization Goal = Minimize or maximize

*** “Minimize” usually for minimizing average loss (e.g., mean squared error)

*** “Maximize” usually for maximizing the likelihood of the training data for a given model (aka maximum likelihood, or log-likelihood)

What is this?





Images credit: Wikipedia

Target Model

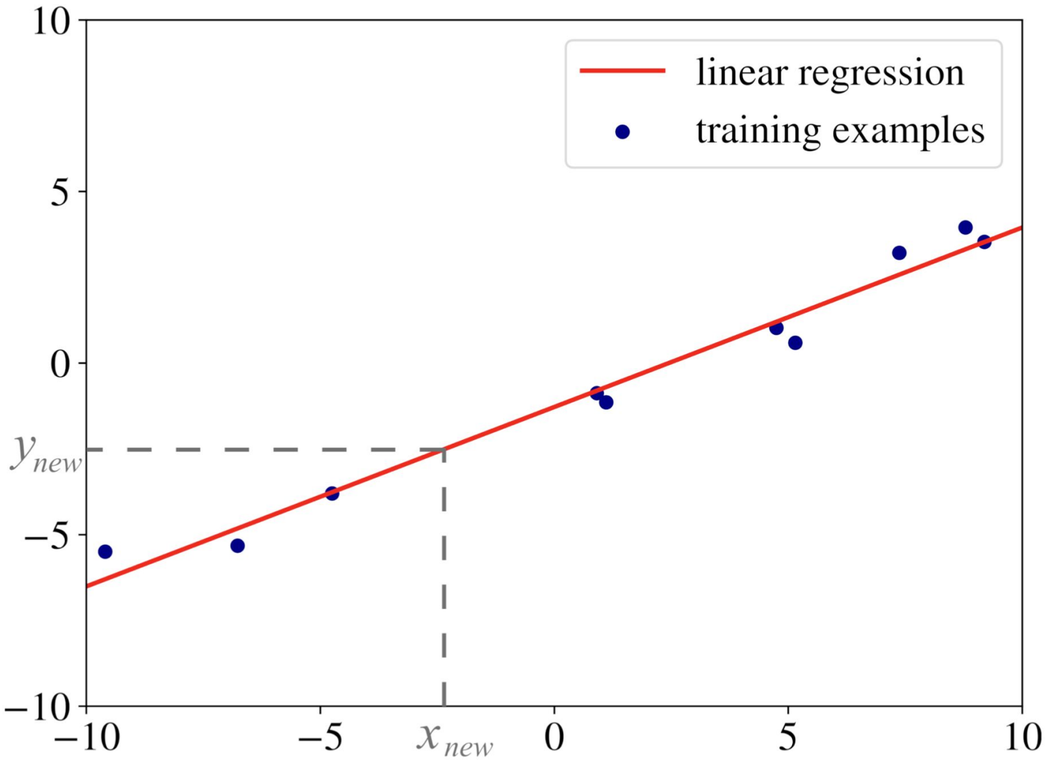


Figure 3.1: Linear Regression for one-dimensional examples.

$$f_{\mathbf{w},b}(\mathbf{x}) = \mathbf{w}\mathbf{x} + b$$

Model Candidate

↓

$$y \leftarrow f_{\mathbf{w},b}(\mathbf{x})$$

Target Model

Images credit: The 100-Page ML Book by Andriy Burkov

Loss / Error Function

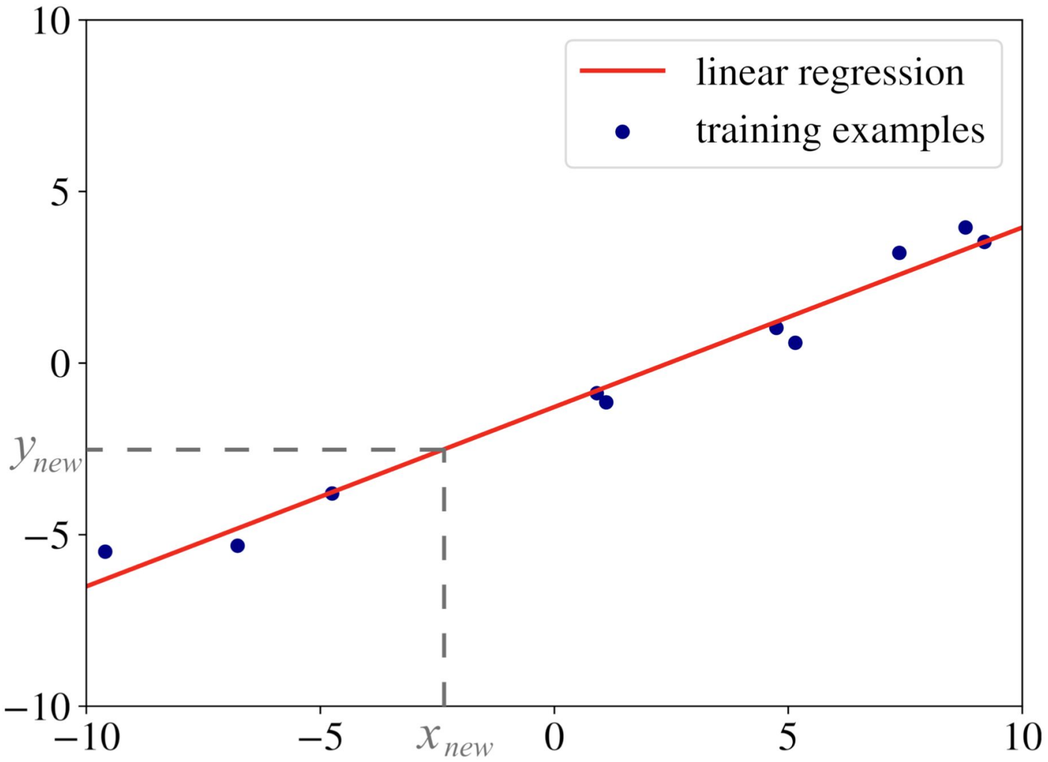


Figure 3.1: Linear Regression for one-dimensional examples.

$$(f_{\mathbf{w},b}(\mathbf{x}_i) - y_i)^2$$

Squared Error Loss

Images credit: The 100-Page ML Book by Andriy Burkov

Cost / Objective Function

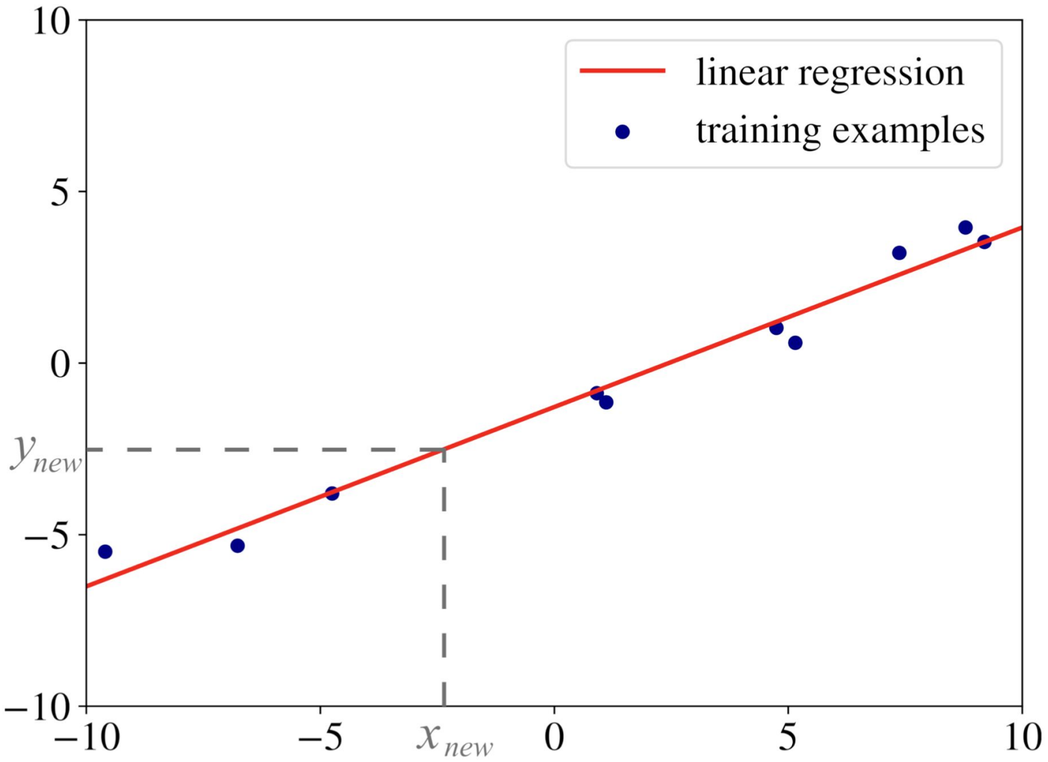


Figure 3.1: Linear Regression for one-dimensional examples.

$$\frac{1}{N}\sum_{i=1\ldots N}\left(f_{\mathbf{w},b}(\mathbf{x}_i)-y_i\right)^2.$$

Cost Function Candidate

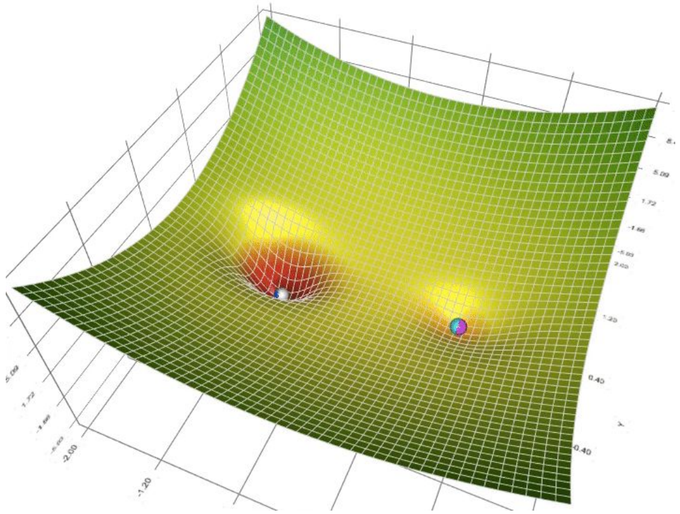
↓

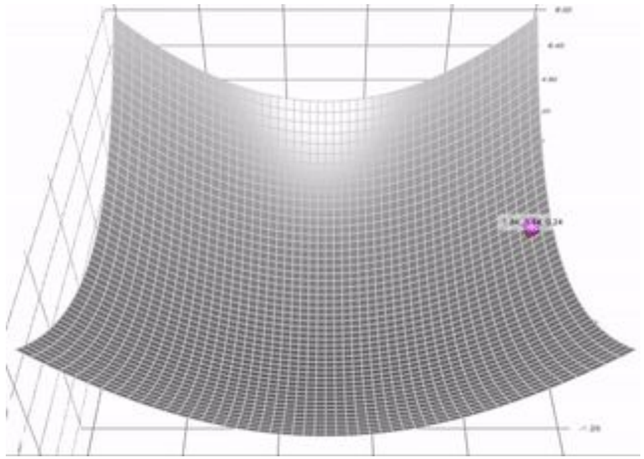
$$l \stackrel{\text{def}}{=} \frac{1}{N}\sum_{i=1}^{N}(y_i-(wx_i+b))^2$$

Cost Function
(aka empirical risk & mean squared error)

Images credit: The 100-Page ML Book by Andriy Burkov

Optimization





$$l \stackrel{\text{def}}{=} \frac{1}{N}\sum_{i=1}^{N}(y_i - (wx_i + b))^2$$

Goal = Minimize MSE (our cost function)

$$\frac{\partial l}{\partial w} = \frac{1}{N}\sum_{i=1}^{N} -2x_i(y_i - (wx_i + b));$$

$$\frac{\partial l}{\partial b} = \frac{1}{N}\sum_{i=1}^{N} -2(y_i - (wx_i + b)).$$

Parameter Partial Derivatives

$$w \leftarrow w - \alpha\frac{\partial l}{\partial w};$$

$$b \leftarrow b - \alpha\frac{\partial l}{\partial b}.$$

Parameter Updates per Epoch

Images credit: Towards Data Science The & 100-Page ML Book by Andriy Burkov

How are model hyper-parameters different from model parameters?

What is the difference between a learning algorithm and a model?

Common Performance Metrics

- Accuracy
- Precision / Recall / F1 Score
- ROC / AUC
- Mean Squared Error (MSE)
- Confusion Matrix – TPs, FPs, TNs, FNs

Confusion Matrix

		Predicted condition			
	Total population = P + N	**Predicted Positive (PP)**	**Predicted Negative (PN)**	Informedness, bookmaker informedness (BM) = TPR + TNR − 1	Prevalence threshold (PT) $= \frac{\sqrt{TPR \times FPR} - FPR}{TPR - FPR}$
Actual condition	**Positive (P)**	**True positive (TP),** hit	**False negative (FN),** type II error, miss, underestimation	True positive rate (TPR), recall, sensitivity (SEN), probability of detection, hit rate, power $= \frac{TP}{P} = 1 - FNR$	False negative rate (FNR), miss rate $= \frac{FN}{P} = 1 - TPR$
	Negative (N)	**False positive (FP),** type I error, false alarm, overestimation	**True negative (TN),** correct rejection	False positive rate (FPR), probability of false alarm, fall-out $= \frac{FP}{N} = 1 - TNR$	True negative rate (TNR), specificity (SPC), selectivity $= \frac{TN}{N} = 1 - FPR$
	Prevalence $= \frac{P}{P + N}$	Positive predictive value (PPV), precision $= \frac{TP}{PP} = 1 - FDR$	False omission rate (FOR) $= \frac{FN}{PN} = 1 - NPV$	Positive likelihood ratio (LR+) $= \frac{TPR}{FPR}$	Negative likelihood ratio (LR−) $= \frac{FNR}{TNR}$
	Accuracy (ACC) $= \frac{TP + TN}{P + N}$	False discovery rate (FDR) $= \frac{FP}{PP} = 1 - PPV$	Negative predictive value (NPV) $= \frac{TN}{PN}$ $= 1 - FOR$	Markedness (MK), deltaP (Δp) = PPV + NPV − 1	Diagnostic odds ratio (DOR) $= \frac{LR+}{LR-}$
	Balanced accuracy (BA) $= \frac{TPR + TNR}{2}$	F_1 score $= \frac{2PPV \times TPR}{PPV + TPR} = \frac{2TP}{2TP + FP + FN}$	Fowlkes–Mallows index (FM) $= \sqrt{PPV \times TPR}$	Matthews correlation coefficient (MCC) $= \sqrt{TPR \times TNR \times PPV \times NPV} - \sqrt{FNR \times FPR \times FOR \times FDR}$	Threat score (TS), critical success index (CSI), Jaccard index $= \frac{TP}{TP + FN + FP}$

Comprehensive Version

		Predicted condition	
	Total population = P + N	**Positive (PP)**	**Negative (PN)**
Actual condition	**Positive (P)**	**True positive (TP)**	**False negative (FN)**
	Negative (N)	**False positive (FP)**	**True negative (TN)**

Simple Version

		Predicted condition	
	Total 8 + 4 = 12	**Cancer**	**Non-cancer**
Actual condition	**Cancer**	6	2
	Non-cancer	1	3

Medical Diagnosis Example

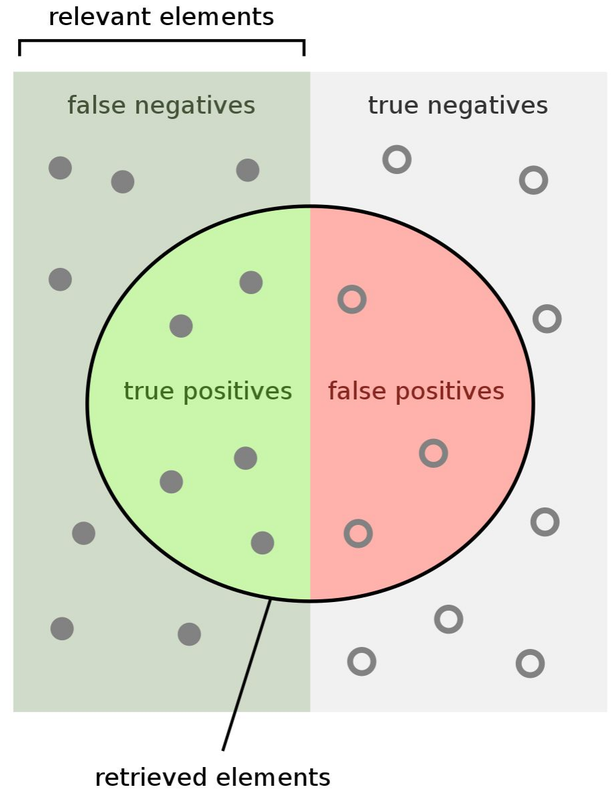
Images credit: Wikipedia

$$\text{precision} \overset{\text{def}}{=} \frac{\text{TP}}{\text{TP} + \text{FP}}$$

Correct Positive Predictions / Total Positive Predictions

$$\text{recall} \overset{\text{def}}{=} \frac{\text{TP}}{\text{TP} + \text{FN}}$$

Correct Positive Predictions / Total Positive Examples



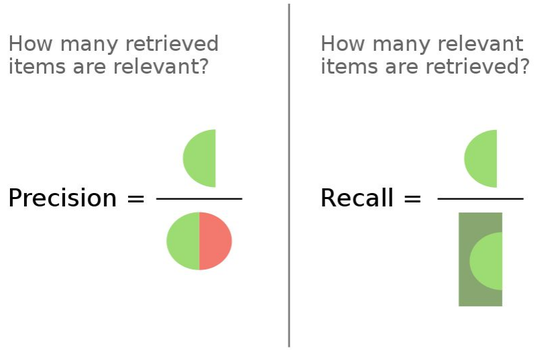


Image credit: Wikipedia

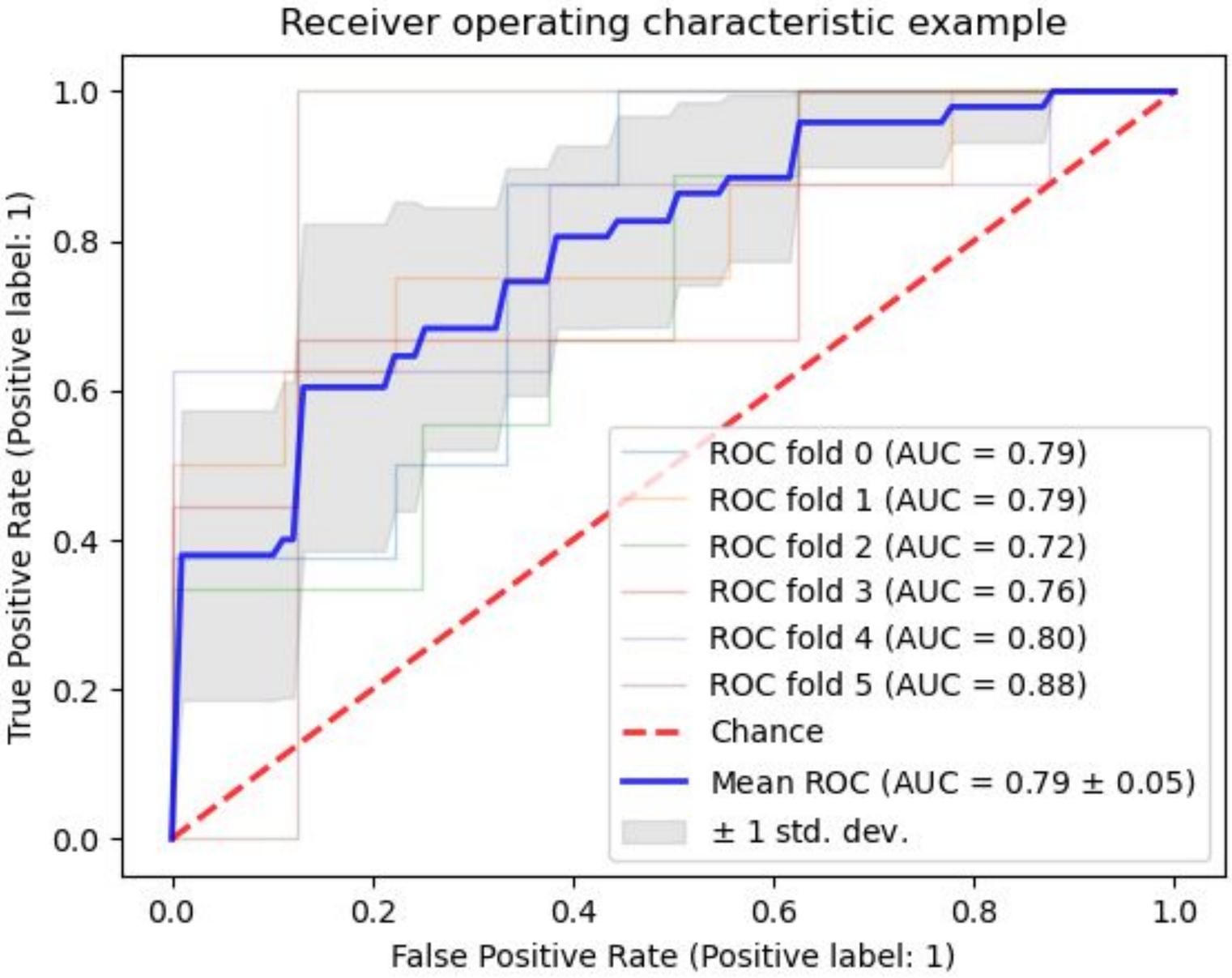


Image credit: Scikit-learn documentation

Always choose a single-number evaluation metric for model selection and optimization

Learn from other metrics as well!

Activity

Model Ready or Not

Use case: You lead an operations analytics team at a national **cold-chain logistics provider** (refrigerated warehouses + trucks). You are building a model to predict **equipment failure risk** (e.g., refrigeration unit failure) so teams can schedule preventive maintenance and avoid spoilage.

Goal: Practice diagnosing **overfitting vs underfitting** and choosing the next best action to improve real-world performance.

Question: For each scenario provided:

- Is the model **overfitting**, **underfitting**, or **well-fit** and what is your **one best next step**?
- What **evidence** would convince you the fix **worked** in the **real world** (not just on the validation set)?

***** Don't use any devices for assistance**
***** Follow guidance on activity steps**

Activity

Model Ready or Not (continued)

Scenario A (results):

Train accuracy: 97%
Validation accuracy: 62%

Scenario B (results):

Train accuracy: 68%
Validation accuracy: 66%

Scenario A (results):

Train accuracy: 82%
Validation accuracy: 80%

***** Don't use any devices for assistance**
***** Follow guidance on activity steps**

Activity

Model Ready or Not (continued)

Preview (Generative AI / Trust):

If you also deploy a GenAI assistant to summarize maintenance logs and propose likely causes, what new risk appears when it outputs a confident explanation that is wrong (“hallucination”)? In that world, what does “accuracy” and “trust” mean operationally?

***** Don’t use any devices for assistance**
***** Follow guidance on activity steps**

What does model tuning mean and how can we tune models?

Data Splitting and Algorithm Validation Methods

- Train/Test or Train/Validate/Test
- Cross validation
  - Can be used in lieu of validation set

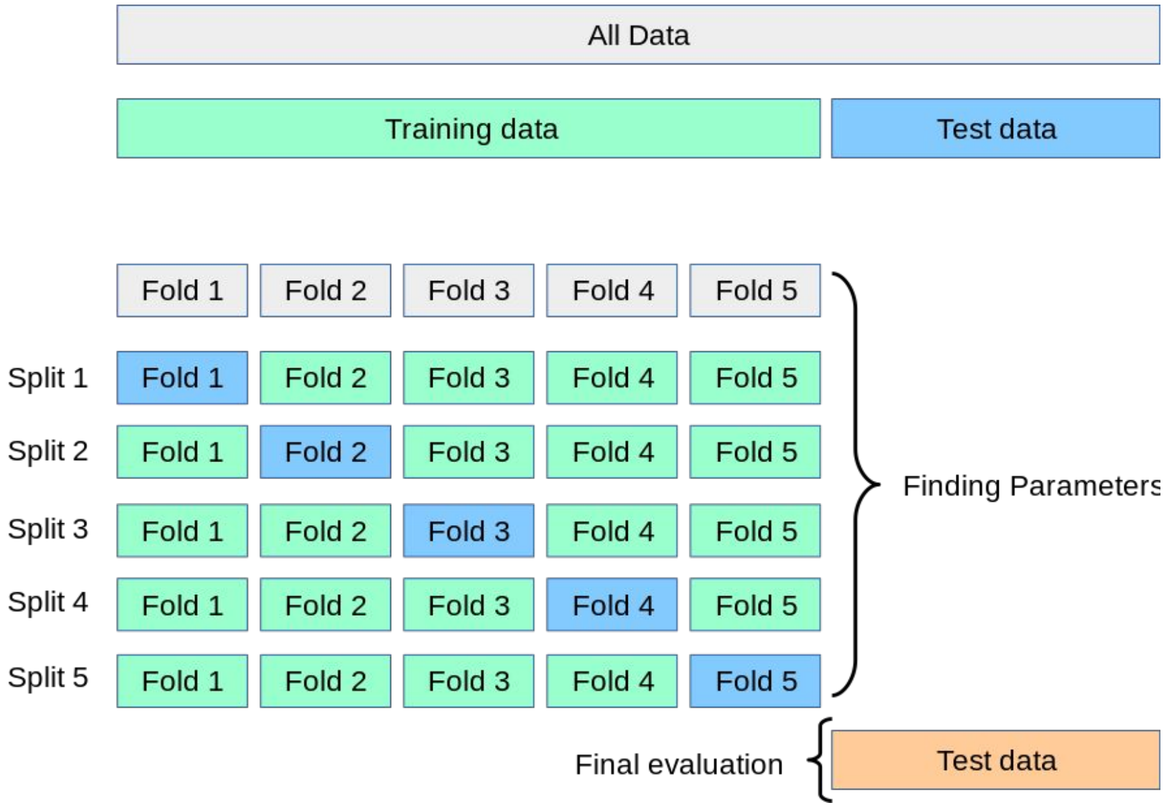
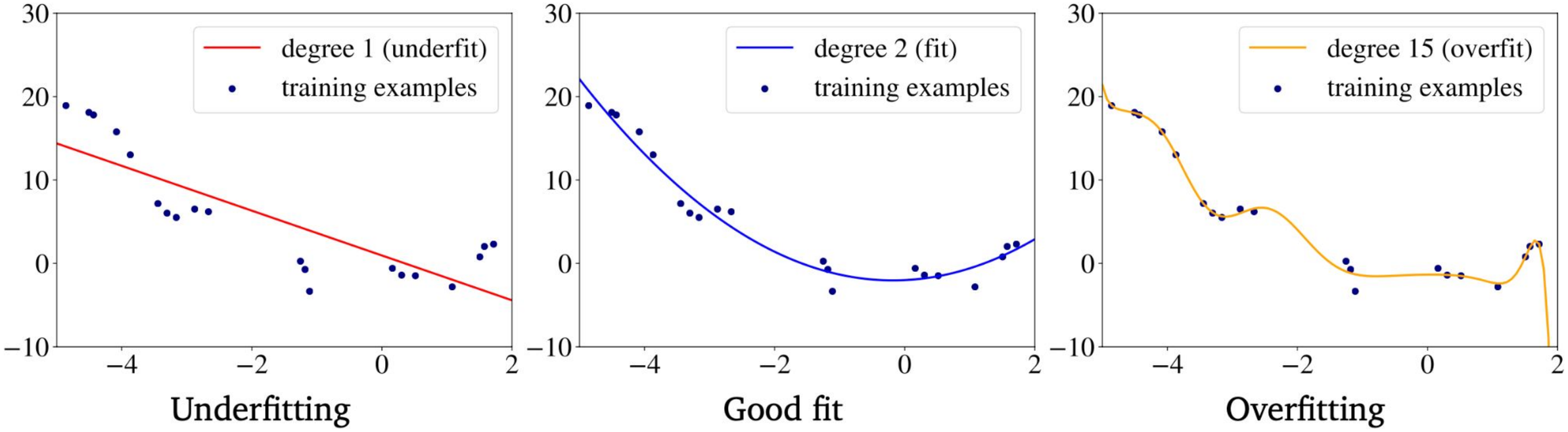


Image credit: Scikit-learn documentation

Results



Images credit: The 100-Page ML Book by Andriy Burkov

What do the following mean?

- Low training score, low test score
- High training score, low test score
- High training score, high test score

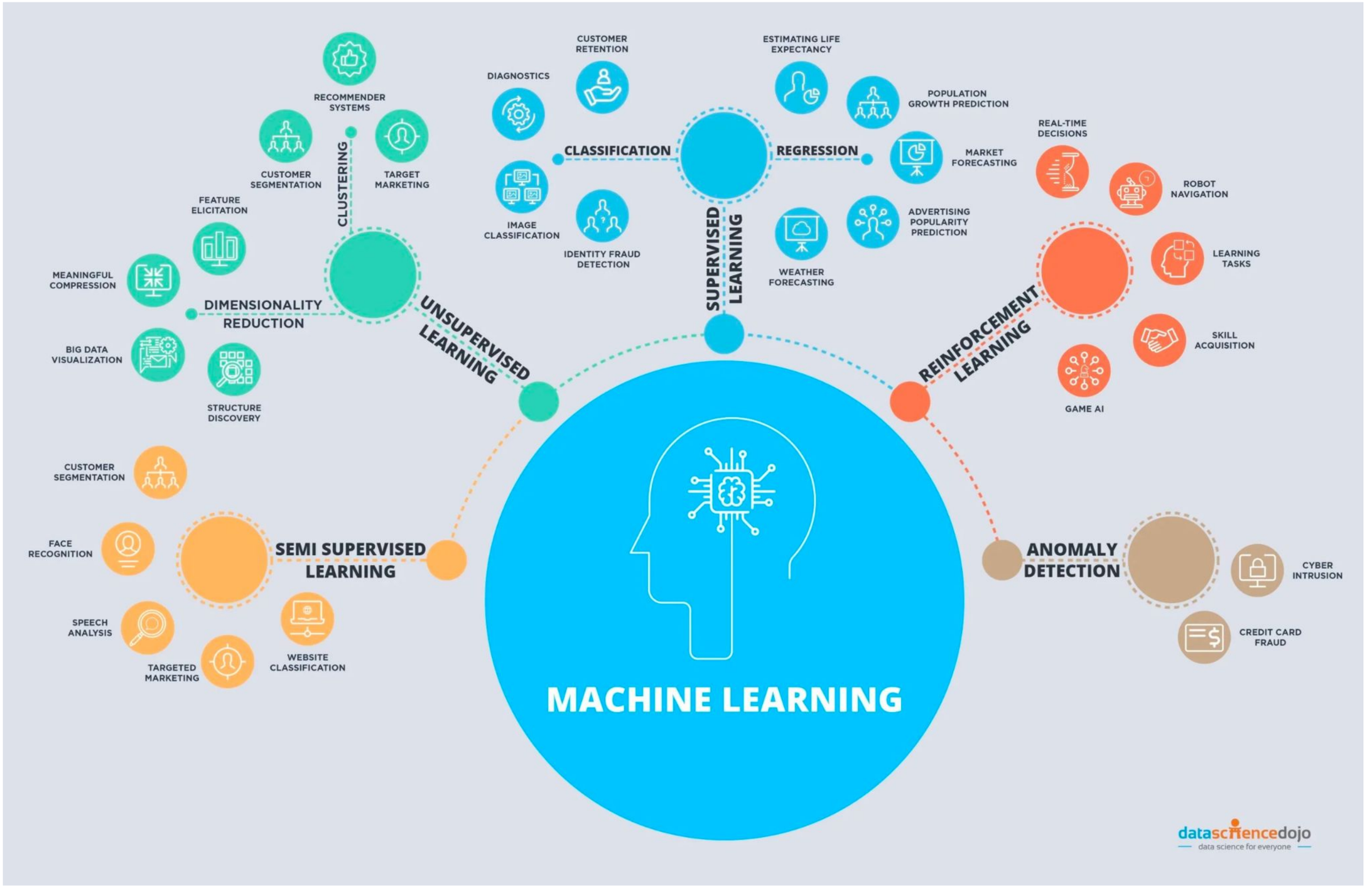
What’s missing and why?

Self-Guided Learning

Model Optimization Techniques

- Regularization (Ridge regression, Lasso, ElasticNet, …)
- Ensembles
  - Bagging (e.g., random forests)
  - Boosting (e.g., GBMs)
- Data
  - Add / remove / change features
  - Add more examples
  - Add more diverse/representative data
  - Add augmented data
- Change learning algorithms
- Hyperparameter tuning
- Oversampling/undersampling
- Train learning algorithm longer
- Change neural network architecture or increase/decrease model complexity

Image via ChatGPT


MACHINE LEARNING
SUPERVISED LEARNING
CLASSIFICATION
DIAGNOSTICS
CUSTOMER RETENTION
IMAGE CLASSIFICATION
IDENTITY FRAUD DETECTION
REGRESSION
ESTIMATING LIFE EXPECTANCY
POPULATION GROWTH PREDICTION
MARKET FORECASTING
ADVERTISING POPULARITY PREDICTION
WEATHER FORECASTING
UNSUPERVISED LEARNING
CLUSTERING
RECOMMENDER SYSTEMS
CUSTOMER SEGMENTATION
TARGET MARKETING
DIMENSIONALITY REDUCTION
FEATURE ELICITATION
MEANINGFUL COMPRESSION
BIG DATA VISUALIZATION
STRUCTURE DISCOVERY
REINFORCEMENT LEARNING
REAL-TIME DECISIONS
ROBOT NAVIGATION
LEARNING TASKS
SKILL ACQUISITION
GAME AI
SEMI SUPERVISED LEARNING
CUSTOMER SEGMENTATION
FACE RECOGNITION
SPEECH ANALYSIS
TARGETED MARKETING
WEBSITE CLASSIFICATION
ANOMALY DETECTION
CYBER INTRUSION
CREDIT CARD FRAUD
datasciencedojo
data science for everyone

Activity

Pick the Metric and the Model

Use case: You are deploying an ML model to **flag potentially fraudulent credit card transactions** for review. Investigators can manually review **only 200 transactions per day**.

Goal: Select a **primary metric** and a **model decision** that best aligns with business risk, capacity constraints, and user experience.

***** Don’t use any devices for assistance**
***** Follow guidance on activity steps**

Activity

Pick the Metric and the Model (cont.)

Question: Which model do you choose and why?

You must decide:

- Which error is worse: false alarm (FP) or missed fraud (FN)
- Your primary metric (accuracy, precision, recall, or F1)
- Your final recommendation: Model A or Model B

Context (assume):

- A missed fraud (FN) costs the company an average of $500 and damages trust.
- A false alarm (FP) creates customer friction and costs $25 in support time.
- Fraud prevalence is low and customers are sensitive to false declines.
- You cannot exceed investigator capacity.

***** Don’t use any devices for assistance**
***** Follow guidance on activity steps**

Pick the Metric and the Model (cont.)

Activity

Candidate models (evaluated on the same 10,000 transactions):

Model A:

Flags for review: 180
True fraud caught (TP): 120
False alarms (FP): 60
Missed fraud (FN): 80
Correct clears (TN): 9,740

Model B:

Flags for review: 260
True fraud caught (TP): 150
False alarms (FP): 110
Missed fraud (FN): 50
Correct clears (TN): 9,690

***** Don’t use any devices for assistance**
***** Follow guidance on activity steps**

Appendix

K-Means

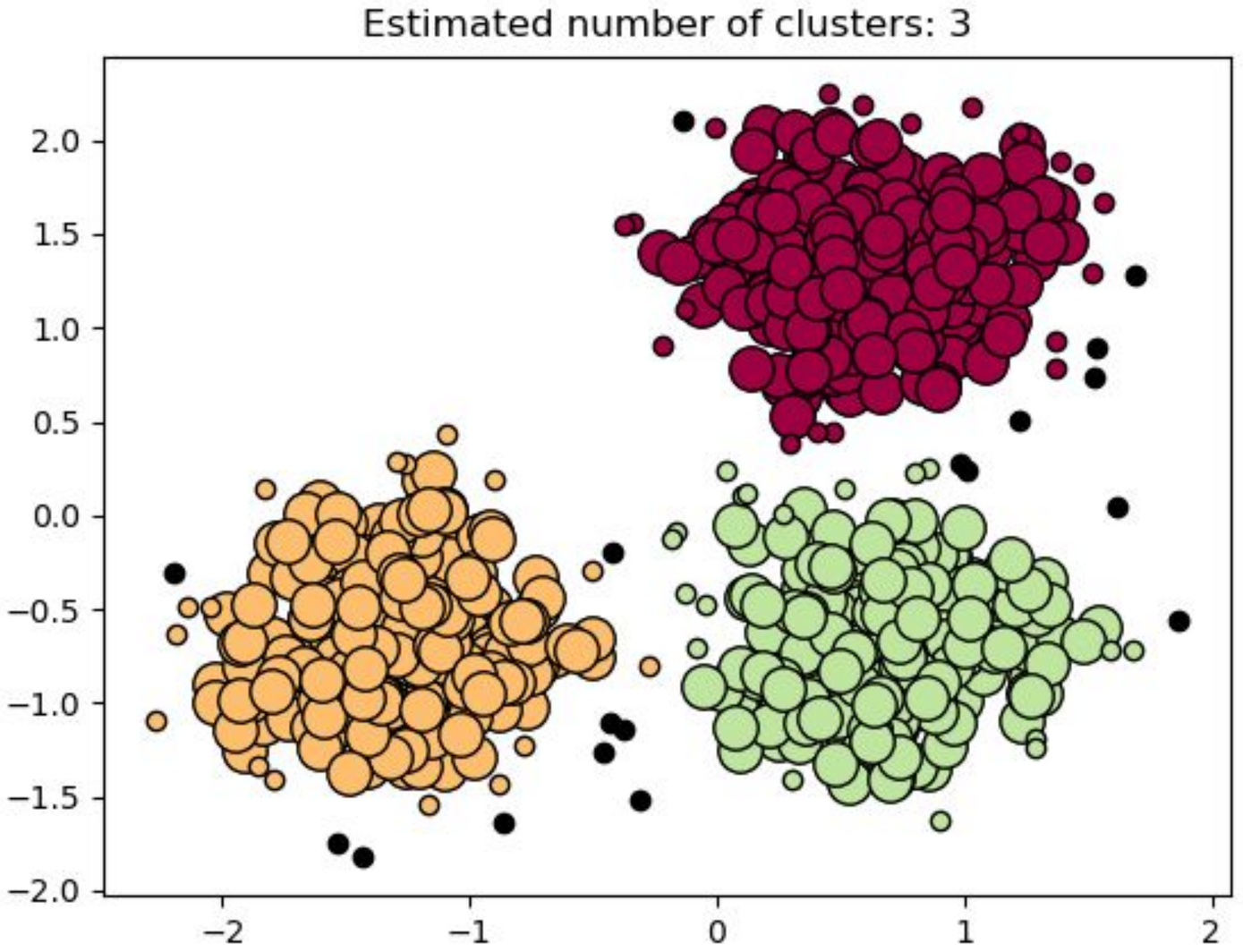
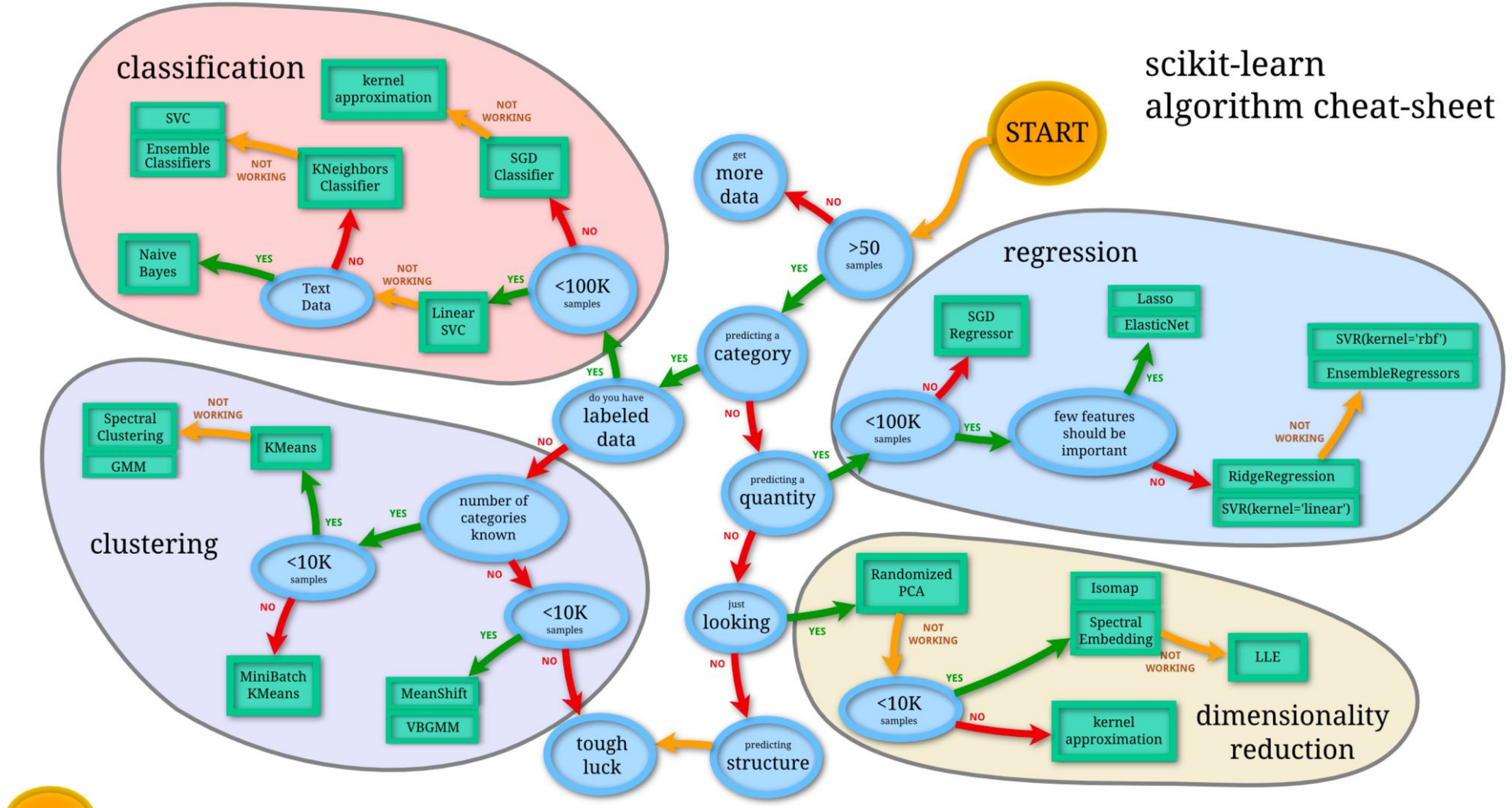


Image credit: Scikit-learn Documentation


scikit-learn
algorithm cheat-sheet
START
classification
kernel approximation
NOT WORKING
SVC
Ensemble Classifiers
NOT WORKING
KNeighbors Classifier
SGD Classifier
NO
Naive Bayes
YES
NO
NOT WORKING
YES
Text Data
NOT WORKING
Linear SVC
<100K samples
NO
get more data
NO
>50 samples
YES
regression
SGD Regressor
Lasso
ElasticNet
SVR(kernel='rbf')
EnsembleRegressors
NO
<100K samples
YES
few features should be important
YES
NOT WORKING
NO
RidgeRegression
SVR(kernel='linear')
YES
predicting a category
YES
do you have labeled data
NO
NO
predicting a quantity
YES
clustering
Spectral Clustering
NOT WORKING
GMM
KMeans
YES
YES
number of categories known
<10K samples
NO
NO
MiniBatch KMeans
NO
<10K samples
YES
MeanShift
VBGMM
NO
just looking
YES
Randomized PCA
NOT WORKING
Isomap
Spectral Embedding
NOT WORKING
LLE
YES
<10K samples
NO
kernel approximation
dimensionality reduction
NO
predicting structure
tough luck
Back
scikit learn

Home > Products > Machine Learning > Glossary

Was this helpful?

Machine Learning Glossary

Send feedback

This glossary defines general machine learning terms, plus terms specific to TensorFlow.

Did You Know?

You can **filter the glossary** by choosing a topic from the Glossary dropdown in the top navigation bar. The hatching bird icon signifies definitions aimed at ML newcomers.

A

ablation

A technique for evaluating the importance of a **feature** or component by temporarily *removing* it from a **model**. You then retrain the model without that feature or component, and if the retrained model performs significantly worse, then the removed feature or component was likely important.

For example, suppose you train a **classification model** on 10 features and achieve 88% **precision** on the **test set**. To check the **importance** of the first feature, you can retrain the model using only the nine other features. If the retrained model performs significantly worse (for instance, 55% precision), then the removed feature was probably important. Conversely, if the retrained model performs equally well, then that feature was probably not that important.

Ablation can also help determine the importance of:

- Larger components, such as an entire subsystem of a larger ML system
- Processes or techniques, such as a data preprocessing step

In both cases, you would observe how the system's performance changes (or doesn't change) after you've removed the component.

Docs » Machine Learning Glossary

Machine Learning Glossary

Brief visual explanations of machine learning concepts resources for learning more.

Warning

If you find errors, please raise an issue or contribute

Basics

- Linear Regression
- Gradient Descent
- Logistic Regression
- Glossary

Math

- Calculus
- Linear Algebra
- Probability (TODO)
- Statistics (TODO)
- Notation

Neural Networks

- Concepts
- Forwardpropagation
- Backpropagation
- Activation Functions
- Layers
- Loss Functions
- Optimizers
- Regularization
- Architectures

datacamp

Python For Data Science
Scikit-Learn Cheat Sheet

Learn Scikit-Learn online at www.DataCamp.com

Scikit-learn

Scikit-learn is an open source Python library that implements a range of machine learning, preprocessing, cross-validation and visualization algorithms using a unified interface.

A Basic Example

```
>>> from sklearn import neighbors, datasets, preprocessing
>>> from sklearn.model_selection import train_test_split
>>> from sklearn.metrics import accuracy_score
>>> iris = datasets.load_iris()
>>> X, y = iris.data[:, :2], iris.target
>>> X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=33)
>>> scaler = preprocessing.StandardScaler().fit(X_train)
>>> X_train = scaler.transform(X_train)
>>> X_test = scaler.transform(X_test)
>>> knn = neighbors.KNeighborsClassifier(n_neighbors=5)
>>> knn.fit(X_train, y_train)
>>> y_pred = knn.predict(X_test)
>>> accuracy_score(y_test, y_pred)
```

> Loading The Data

Also see NumPy & Pandas

Your data needs to be numeric and stored as NumPy arrays or SciPy sparse matrices. Other types that are convertible to numeric arrays, such as Pandas DataFrame, are also acceptable.

```
>>> import numpy as np
>>> X = np.random.random((10,5))
>>> y = np.array(['M','M','F','F','M','F','M','M','F','F','F'])
>>> X[X < 0.7] = 0
```

> Training And Test Data

```
>>> from sklearn.model_selection import train_test_split
>>> X_train, X_test, y_train, y_test = train_test_split(X,
                                                        y,
                                                        random_state=0)
```

> Model Fitting

Supervised learning

```
>>> lr.fit(X, y) #Fit the model to the data
>>> knn.fit(X_train, y_train)
>>> svc.fit(X_train, y_train)
```

Unsupervised Learning

```
>>> k_means.fit(X_train) #Fit the model to the data
>>> pca_model = pca.fit_transform(X_train) #Fit to data, then transform it
```

> Prediction

Supervised Estimators

```
>>> y_pred = svc.predict(np.random.random((2,5))) #Predict labels
>>> y_pred = lr.predict(X_test) #Predict labels
>>> y_pred = knn.predict_proba(X_test) #Estimate probability of a label
```

Unsupervised Estimators

```
>>> y_pred = k_means.predict(X_test) #Predict labels in clustering algos
```

> Preprocessing The Data

Standardization

```
>>> from sklearn.preprocessing import StandardScaler
>>> scaler = StandardScaler().fit(X_train)
>>> standardized_X = scaler.transform(X_train)
>>> standardized_X_test = scaler.transform(X_test)
```

Normalization

```
>>> from sklearn.preprocessing import Normalizer
>>> scaler = Normalizer().fit(X_train)
>>> normalized_X = scaler.transform(X_train)
>>> normalized_X_test = scaler.transform(X_test)
```

Binarization

```
>>> from sklearn.preprocessing import Binarizer
>>> binarizer = Binarizer(threshold=0.0).fit(X)
>>> binary_X = binarizer.transform(X)
```

Encoding Categorical Features

```
>>> from sklearn.preprocessing import LabelEncoder
>>> enc = LabelEncoder()
>>> y = enc.fit_transform(y)
```

Imputing Missing Values

```
>>> from sklearn.preprocessing import Imputer
>>> imp = Imputer(missing_values=0, strategy='mean', axis=0)
>>> imp.fit_transform(X_train)
```

Generating Polynomial Features

```
>>> from sklearn.preprocessing import PolynomialFeatures
>>> poly = PolynomialFeatures(5)
>>> poly.fit_transform(X)
```

> Create Your Model

Supervised Learning Estimators

Linear Regression

```
>>> from sklearn.linear_model import LinearRegression
>>> lr = LinearRegression(normalize=True)
```

Support Vector Machines (SVM)

```
>>> from sklearn.svm import SVC
>>> svc = SVC(kernel='linear')
```

Naive Bayes

```
>>> from sklearn.naive_bayes import GaussianNB
>>> gnb = GaussianNB()
```

KNN

```
>>> from sklearn import neighbors
>>> knn = neighbors.KNeighborsClassifier(n_neighbors=5)
```

Unsupervised Learning Estimators

Principal Component Analysis (PCA)

```
>>> from sklearn.decomposition import PCA
>>> pca = PCA(n_components=0.95)
```

K Means

```
>>> from sklearn.cluster import KMeans
>>> k_means = KMeans(n_clusters=3, random_state=0)
```

> Evaluate Your Model's Performance

Classification Metrics

Accuracy Score

```
>>> knn.score(X_test, y_test) #Estimator score method
>>> from sklearn.metrics import accuracy_score #Metric scoring functions
>>> accuracy_score(y_test, y_pred)
```

Classification Report

```
>>> from sklearn.metrics import classification_report #Precision, recall, f1-score and support
>>> print(classification_report(y_test, y_pred))
```

Confusion Matrix

```
>>> from sklearn.metrics import confusion_matrix
>>> print(confusion_matrix(y_test, y_pred))
```

Regression Metrics

Mean Absolute Error

```
>>> from sklearn.metrics import mean_absolute_error
>>> y_true = [3, -0.5, 2]
>>> mean_absolute_error(y_true, y_pred)
```

Mean Squared Error

```
>>> from sklearn.metrics import mean_squared_error
>>> mean_squared_error(y_test, y_pred)
```

R^2 Score

```
>>> from sklearn.metrics import r2_score
>>> r2_score(y_true, y_pred)
```

Clustering Metrics

Adjusted Rand Index

```
>>> from sklearn.metrics import adjusted_rand_score
>>> adjusted_rand_score(y_true, y_pred)
```

Homogeneity

```
>>> from sklearn.metrics import homogeneity_score
>>> homogeneity_score(y_true, y_pred)
```

V-measure

```
>>> from sklearn.metrics import v_measure_score
>>> metrics.v_measure_score(y_true, y_pred)
```

Cross-Validation

```
>>> from sklearn.cross_validation import cross_val_score
>>> print(cross_val_score(knn, X_train, y_train, cv=4))
>>> print(cross_val_score(lr, X, y, cv=2))
```

> Tune Your Model

Grid Search

```
>>> from sklearn.grid_search import GridSearchCV
>>> params = {"n_neighbors": np.arange(1,3),
              "metric": ["euclidean", "cityblock"]}
>>> grid = GridSearchCV(estimator=knn,
                        param_grid=params)
>>> grid.fit(X_train, y_train)
>>> print(grid.best_score_)
>>> print(grid.best_estimator_.n_neighbors)
```

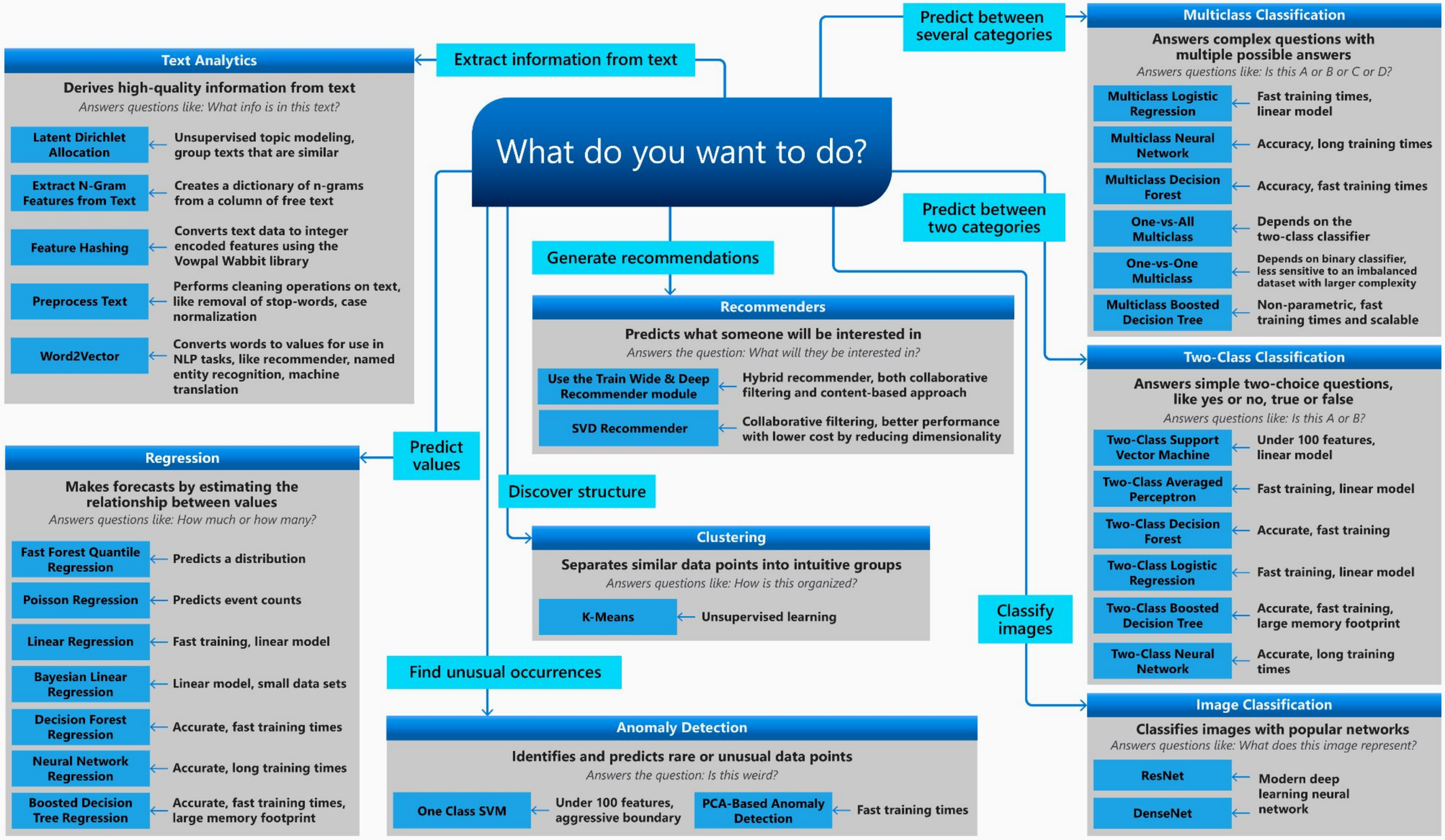
Randomized Parameter Optimization

```
>>> from sklearn.grid_search import RandomizedSearchCV
>>> params = {"n_neighbors": range(1,5), "weights": ["uniform", "distance"]}
>>> rsearch = RandomizedSearchCV(estimator=knn, param_distributions=params,
                                 cv=4, n_iter=8, random_state=5)
>>> rsearch.fit(X_train, y_train)
>>> print(rsearch.best_score_)
```

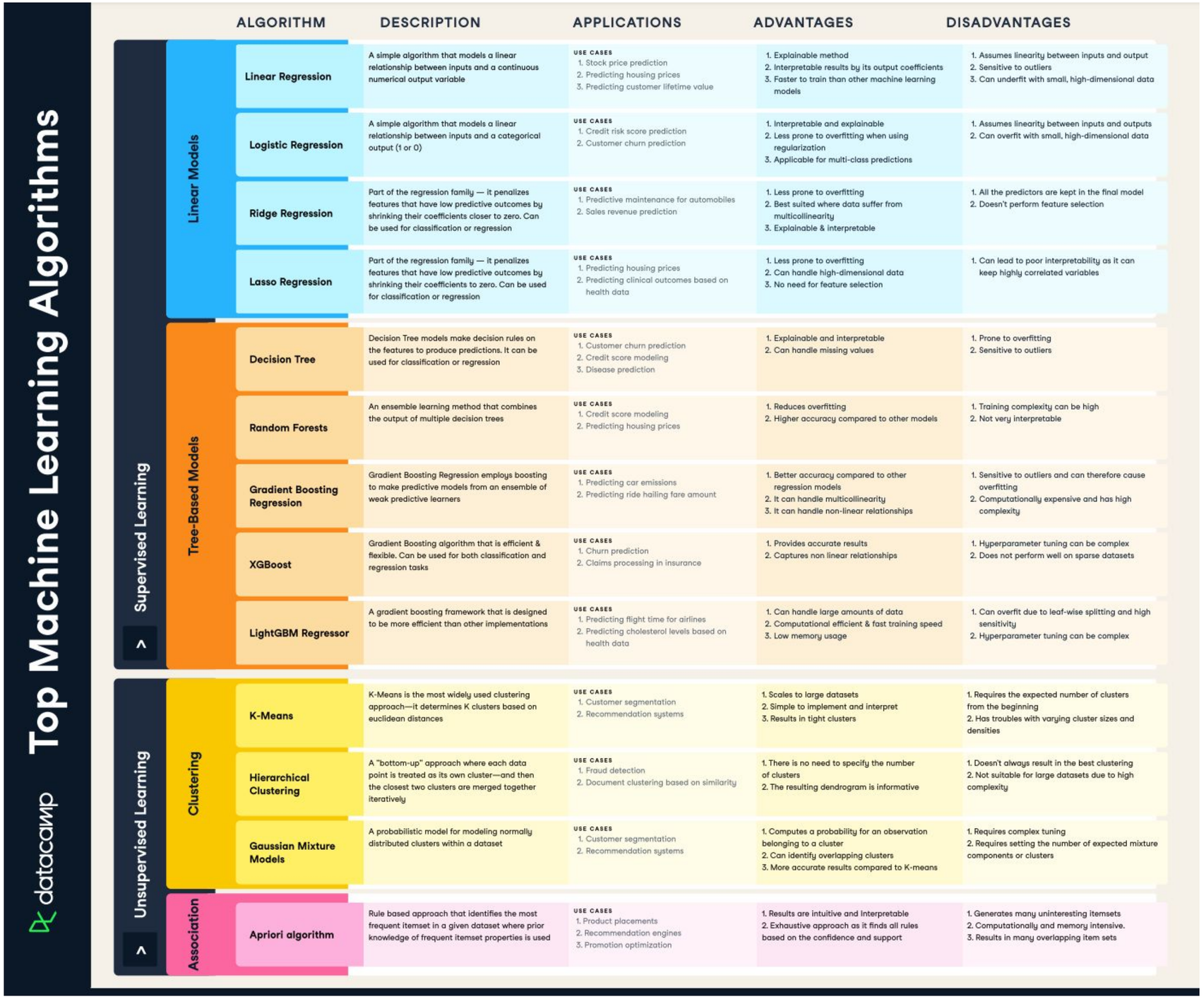
Machine Learning Algorithm Cheat Sheet

This cheat sheet helps you choose the best machine learning algorithm for your predictive analytics solution.
Your decision is driven by both the nature of your data and the goal you want to achieve with your data.

Microsoft Azure



 Share this poster: aka.ms/mlcheatsheet


Top Machine Learning Algorithms
datacamp
ALGORITHM
DESCRIPTION
APPLICATIONS
ADVANTAGES
DISADVANTAGES
Supervised Learning
Linear Models
Linear Regression
A simple algorithm that models a linear relationship between inputs and a continuous numerical output variable
USE CASES
1. Stock price prediction
2. Predicting housing prices
3. Predicting customer lifetime value
1. Explainable method
2. Interpretable results by its output coefficients
3. Faster to train than other machine learning models
1. Assumes linearity between inputs and output
2. Sensitive to outliers
3. Can underfit with small, high-dimensional data
Logistic Regression
A simple algorithm that models a linear relationship between inputs and a categorical output (1 or 0)
USE CASES
1. Credit risk score prediction
2. Customer churn prediction
1. Interpretable and explainable
2. Less prone to overfitting when using regularization
3. Applicable for multi-class predictions
1. Assumes linearity between inputs and outputs
2. Can overfit with small, high-dimensional data
Ridge Regression
Part of the regression family — it penalizes features that have low predictive outcomes by shrinking their coefficients closer to zero. Can be used for classification or regression
USE CASES
1. Predictive maintenance for automobiles
2. Sales revenue prediction
1. Less prone to overfitting
2. Best suited where data suffer from multicollinearity
3. Explainable & interpretable
1. All the predictors are kept in the final model
2. Doesn't perform feature selection
Lasso Regression
Part of the regression family — it penalizes features that have low predictive outcomes by shrinking their coefficients to zero. Can be used for classification or regression
USE CASES
1. Predicting housing prices
2. Predicting clinical outcomes based on health data
1. Less prone to overfitting
2. Can handle high-dimensional data
3. No need for feature selection
1. Can lead to poor interpretability as it can keep highly correlated variables
Tree-Based Models
Decision Tree
Decision Tree models make decision rules on the features to produce predictions. It can be used for classification or regression
USE CASES
1. Customer churn prediction
2. Credit score modeling
3. Disease prediction
1. Explainable and interpretable
2. Can handle missing values
1. Prone to overfitting
2. Sensitive to outliers
Random Forests
An ensemble learning method that combines the output of multiple decision trees
USE CASES
1. Credit score modeling
2. Predicting housing prices
1. Reduces overfitting
2. Higher accuracy compared to other models
1. Training complexity can be high
2. Not very interpretable
Gradient Boosting Regression
Gradient Boosting Regression employs boosting to make predictive models from an ensemble of weak predictive learners
USE CASES
1. Predicting car emissions
2. Predicting ride hailing fare amount
1. Better accuracy compared to other regression models
2. It can handle multicollinearity
3. It can handle non-linear relationships
1. Sensitive to outliers and can therefore cause overfitting
2. Computationally expensive and has high complexity
XGBoost
Gradient Boosting algorithm that is efficient & flexible. Can be used for both classification and regression tasks
USE CASES
1. Churn prediction
2. Claims processing in insurance
1. Provides accurate results
2. Captures non linear relationships
1. Hyperparameter tuning can be complex
2. Does not perform well on sparse datasets
LightGBM Regressor
A gradient boosting framework that is designed to be more efficient than other implementations
USE CASES
1. Predicting flight time for airlines
2. Predicting cholesterol levels based on health data
1. Can handle large amounts of data
2. Computational efficient & fast training speed
3. Low memory usage
1. Can overfit due to leaf-wise splitting and high sensitivity
2. Hyperparameter tuning can be complex
Unsupervised Learning
Clustering
K-Means
K-Means is the most widely used clustering approach—it determines K clusters based on euclidean distances
USE CASES
1. Customer segmentation
2. Recommendation systems
1. Scales to large datasets
2. Simple to implement and interpret
3. Results in tight clusters
1. Requires the expected number of clusters from the beginning
2. Has troubles with varying cluster sizes and densities
Hierarchical Clustering
A "bottom-up" approach where each data point is treated as its own cluster—and then the closest two clusters are merged together iteratively
USE CASES
1. Fraud detection
2. Document clustering based on similarity
1. There is no need to specify the number of clusters
2. The resulting dendrogram is informative
1. Doesn't always result in the best clustering
2. Not suitable for large datasets due to high complexity
Gaussian Mixture Models
A probabilistic model for modeling normally distributed clusters within a dataset
USE CASES
1. Customer segmentation
2. Recommendation systems
1. Computes a probability for an observation belonging to a cluster
2. Can identify overlapping clusters
3. More accurate results compared to K-means
1. Requires complex tuning
2. Requires setting the number of expected mixture components or clusters
Association
Apriori algorithm
Rule based approach that identifies the most frequent itemset in a given dataset where prior knowledge of frequent itemset properties is used
USE CASES
1. Product placements
2. Recommendation engines
3. Promotion optimization
1. Results are intuitive and Interpretable
2. Exhaustive approach as it finds all rules based on the confidence and support
1. Generates many uninteresting itemsets
2. Computationally and memory intensive.
3. Results in many overlapping item sets

Supervised Machine Learning Models

datacamp

	ALGORITHM	DESCRIPTION & APPLICATION	ADVANTAGES	DISADVANTAGES
Regression Only Models	**Linear Regression**	Linear Regression models a linear relationship between input variables and a continuous numerical output variable. The default loss function is the mean square error (MSE).	1. Fast training because there are few parameters. 2. Interpretable/Explainable results by its output coefficients.	1. Assumes a linear relationship between input and output variables. 2. Sensitive to outliers. 3. Typically generalizes worse than ridge or lasso regression.
	Polynomial Regression	Polynomial Regression models nonlinear relationships between the dependent, and independent variable as the n-th degree polynomial.	1. Provides a good approximation of the relationship between the dependent and independent variables. 2. Capable of fitting a wide range of curvature.	1. Poor interpretability of the coefficients since the underlying variables can be highly correlated. 2. The model fit is nonlinear but the regression function is linear. 3. Prone to overfitting.
	Support Vector Regression	Support Vector Regression (SVR) uses the same principle as SVMs but optimizes the cost function to fit the most straight line (or plane) through the data points. With the kernel trick it can efficiently perform a non-linear regression by implicitly mapping their inputs into high-dimensional feature spaces.	1. Robust against outliers. 2. Effective learning and strong generalization performance. 3. Different Kernel functions can be specified for the decision function.	1. Does not perform well with large datasets. 2. Tends to underfit in cases where the number of variables is much smaller than the number of observations.
	Gaussian Process Regression	Gaussian Process Regression (GPR) uses a Bayesian approach that infers a probability distribution over the possible functions that fit the data. The Gaussian process is a prior that is specified as a multivariate Gaussian distribution.	1. Provides uncertainty measures on the predictions. 2. It is a flexible and usable non-linear model which fits many datasets well. 3. Performs well on small datasets as the GP kernel allows to specify a prior on the function space.	1. Poor choice of kernel can make convergence slow. 2. Specifying specific kernels requires deep mathematical understanding.
	Robust Regression	Robust Regression is an alternative to least squares regression when data is contaminated with outliers. The term "robust" refers to the statistical capability to provide useful information even in the face of outliers.	1. Designed to overcome some limitations of traditional parametric and non-parametric methods. 2. Provides better regression coefficient over classical regression methods when outliers are present.	1. More computationally intensive compared to classical regression methods. 2. It is not a cure-all for all violations, such as imbalanced data, poor quality data. 3. If no outliers are present in the data, it may not provide better results than classical regression methods.
Both Regression and Classification Models	**Decision Trees**	Decision Tree models learn on the data by making decision rules on the variables to separate the classes in a flowchart like a tree data structure. They can be used for both regression and classification.	1. Explainable and interpretable. 2. Can handle missing values.	1. Prone to overfitting. 2. Can be unstable with minor data drift. 3. Sensitive to outliers.
	Random Forest	Random Forest classification models learn using an ensemble of decision trees. The output of the random forest is based on a majority vote of the different decision trees.	1. Effective learning and better generalization performance. 2. Can handle moderately large datasets. 3. Less prone to overfit than decision trees.	1. Large number of trees can slow down performance. 2. Predictions are sensitive to outliers. 3. Hyperparameter tuning can be complex.
	Gradient Boosting	An ensemble learning method where weak predictive learners are combined to improve accuracy. Popular techniques include XGBoost, LightGBM and more.	1. Handling of multicollinearity. 2. Handling of non-linear relationships. 3. Effective learning and strong generalization performance. 4. XGBoost is fast and is often used as a benchmark algorithm.	1. Sensitive to outliers and can therefore cause overfitting. 2. High complexity due to hyperparameter tuning. 3. Computationally expensive.
	Ridge Regression	Ridge Regression penalizes variables with low predictive outcomes by shrinking their coefficients towards zero. It can be used for classification and regression.	1. Less prone to overfitting. 2. Best suited when data suffers from multicollinearity. 3. Explainable & Interpretable.	1. All the predictors are kept in the final model. 2. Doesn't perform feature selection.
	Lasso Regression	Lasso Regression penalizes features that have low predictive outcomes by shrinking their coefficients to zero. It can be used for classification and regression.	1. Good generalization performance. 2. Good at handling datasets where the number of variables is much larger than the number of observations. 3. No need for feature selection.	1. Poor interpretability/explainability as it can keep a single variable. from a set of highly correlated variables.
	AdaBoost	Adaptive Boosting uses an ensemble of weak learners that is combined into a weighted sum that represents the final output of the boosted classifier.	1. Explainable & Interpretable. 2. Less need for tweaking parameters. 3. Usually outperforms Random Forest.	1. Less prone to overfitting as the input variables are not jointly optimized. 2. Sensitive to noisy data and outliers.
Classification Only Models	**SVM**	In its simplest form, support vector machine is a linear classifier. But with the kernel trick, it can efficiently perform a non-linear classification by implicitly mapping their inputs into high-dimensional feature spaces. This makes SVM one of the best prediction methods.	1. Effective in cases with a high number of variables. 2. Number of variables can be larger than the number of samples. 3. Different Kernel functions can be specified for the decision function.	1. Sensitive to overfitting, regularization is crucial. 2. Choosing a "good" kernel function can be difficult. 3. Computationally expensive for big data due to high training complexity. 4. Performs poorly if the data is noisy (target classes overlap).
	Nearest Neighbors	Nearest Neighbors predicts the label based on a predefined number of samples closest in distance to the new point.	1. Successful in situations where the decision boundary is irregular. 2. Non-parametric approach as it does not make any assumption on the underlying data.	1. Sensitive to noisy and missing data. 2. Computationally expensive because the entire set of n points for every execution is required.
	Logistic Regression (and its extensions)	The logistic regression models a linear relationship between input variables and the response variable. It models the output as binary values (0 or 1) rather than numeric values.	1. Explainable & Interpretable. 2. Less prone to overfitting using regularization. 3. Applicable for multi-class predictions.	1. Makes a strong assumption about the relationship between input and response variables. 2. Multicollinearity can cause the model to easily overfit without regularization.
	Linear Discriminant Analysis	The linear decision boundary maximizes the separability between the classes by finding a linear combination of features.	1. Explainable & Interpretable. 2. Applicable for multi-class predictions.	1. Multicollinearity can cause the model to overfit. 2. Assuming that all classes share the same covariance matrix. 3. Sensitive to outliers. 4. Doesn't work well with small class sizes.