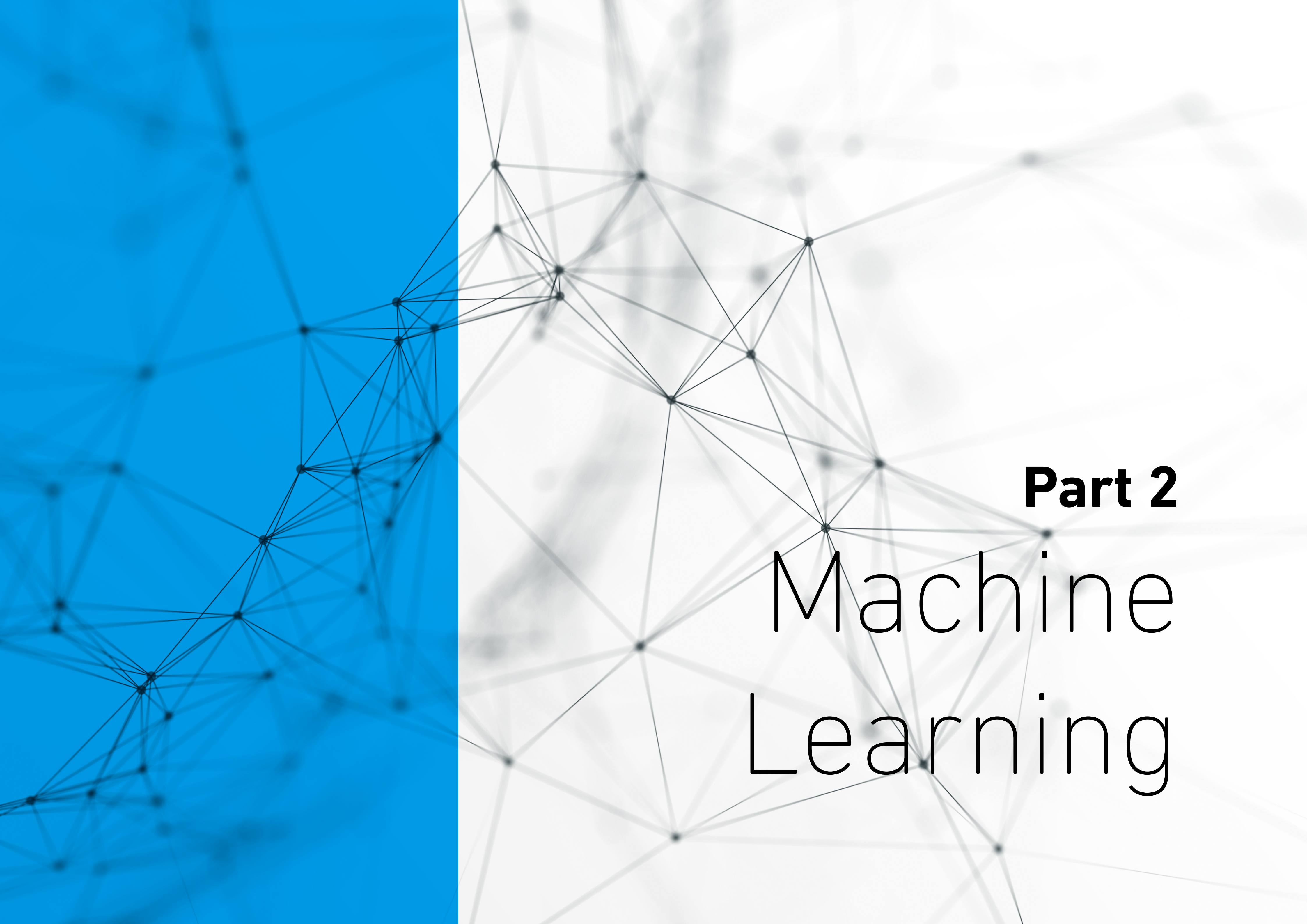


Part 2

Machine Learning

MachineLearning Overview

MACHINE LEARNING IN EMOJI

BecomingHuman.AI

🟩 **SUPERVISED** — human builds model based on input / output

🟪 **UNSUPERVISED** — human input, machine output human utilizes if satisfactory

🟨 **REINFORCEMENT** — human input, machine output human reward/punish, cycle continues

BASIC REGRESSION

🟩 **LINEAR**
linear_model.LinearRegression()
Lots of numerical data

🟩 **LOGISTIC**
linear_model.LogisticRegression()
Target variable is categorical

CLUSTER ANALYSIS

🟩🟪 **K-MEANS**
cluster.KMeans()
Similar datum into groups based on centroids

🟩🟪 **ANOMALY DETECTION**
covariance.EllipticalEnvelope()
Finding outliers through grouping

CLASSIFICATION

🟩🟪🟨 **NEURAL NET**
neural_network.MLPClassifier()
Complex relationships. Prone to overfitting Basically magic.

🟩 **K-NN**
neighbors.KNeighborsClassifier()
Group membership based on proximity

🟩 **DECISION TREE**
tree.DecisionTreeClassifier()
If/then/else. Non-contiguous data. Can also be regression.

🟩🟪 **RANDOM FOREST**
ensemble.RandomForestClassifier()
Find best split randomly Can also be regression

🟩🟪 **SVM**
svm.SVC() svm.LinearSVC()
Maximum margin classifier. Fundamental Data Science algorithm

🟩 **NAIVE BAYES**
GaussianNB() MultinominalNB() BernoulliNB()
Updating knowledge step by step with new info

FEATURE REDUCTION

T-DISTRIB STOCHASTIC NEIB EMBEDDING
manifold.TSNE()
Visual high dimensional data. Convert similarity to joint probabilities

PRINCIPLE COMPONENT ANALYSIS
decomposition.PCA()
Distill feature space into components that describe greatest variance

CANONICAL CORRELATION ANALYSIS
decomposition.CCA()
Making sense of cross-correlation matrices

LINEAR DISCRIMINANT ANALYSIS
lda.LDA()
Linear combination of features that separates classes

OTHER IMPORTANT CONCEPTS

BIAS VARIANCE TRADEOFF

UNDERFITTING / OVERFITTING

INERTIA

ACCURACY FUNCTION
(TP+TN) / (P+N)

PRECISION FUNCTION
manifold.TSNE()

SPECIFICITY FUNCTION
TN / (FP+TN)

SENSITIVITY FUNCTION
TP / (TP+FN)

Originally Created by Emily Barry. **See original here**.

Cheat-Sheet Skicit learn

Phyton For Data Science

BecomingHuman.AI DataCamp

Skicit Learn

Skicit Learn is an open source Phyton library that implements a range if machine learning, processing, cross validation and visualization algorithm using a unified

A basic Example

```
>>> from sklearn import neighbors, datasets, preprocessing
>>> from sklearn.cross validation import train_test_split
>>> from sklearn.metrics import accuracy_score
>>> iris = datasets.load _iris() >>> X, y = iris.data[:, :2], iris.target
>>> Xtrain, X test, y_train, y test = train_test_split (X, y, random stat33)
>>> 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)
```

Prediction

Supervised Estimators

```
>>> y_pred = svc.predict(np.random.radom((2,5)))
>>> y_pred = lr.predict(X_test)
>>> y_pred = knn.predict_proba(X_test)
```

Predict labels
Predict labels
Estimate probability of a label

Unsupervised Estimators

```
>>> y_pred = k_means.predict[X_test]
```

Predict labels in clustering algos

Loading the Data

Your data beeds to be nmueric and stored as NumPy arrays or SciPy sparse matric. other types that they are comvertible to numeric arrays, such as Pandas Dataframe, are also acceptable

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

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

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)
```

Evaluate Your Model's Performance

Classification Metrics

Accuracy Score

```
>>> knn.score(X_test, y_test)
>>> from sklearn.metrics import accuracy_score
>>> accuracy_score(y_test, y_pred)
```

Estimator score method
Metric scoring functions

Classification Report

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

Precision, recall, f1-score and support

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))
```

Model Fitting

Supervised learning

```
>>> lr.fit(X, y)
>>> knn.fit(X_train, y_train)
>>> svc.fit(X_train, y_train)
```

Fit the model to the data

Unsupervised Learning

```
>>> k_means.fit(X_train)
>>> pca_model = pca.fit_transform(X_train)
```

Fit the model to the data
Fit to data, then transform it

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)
```

Training And Test Data

```
>> from sklearn.cross validation import train_test_split
>> X train, X test, y train, y test - train_test_split(X,
                                                       y,
                                                       random state-0)
```

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)
```

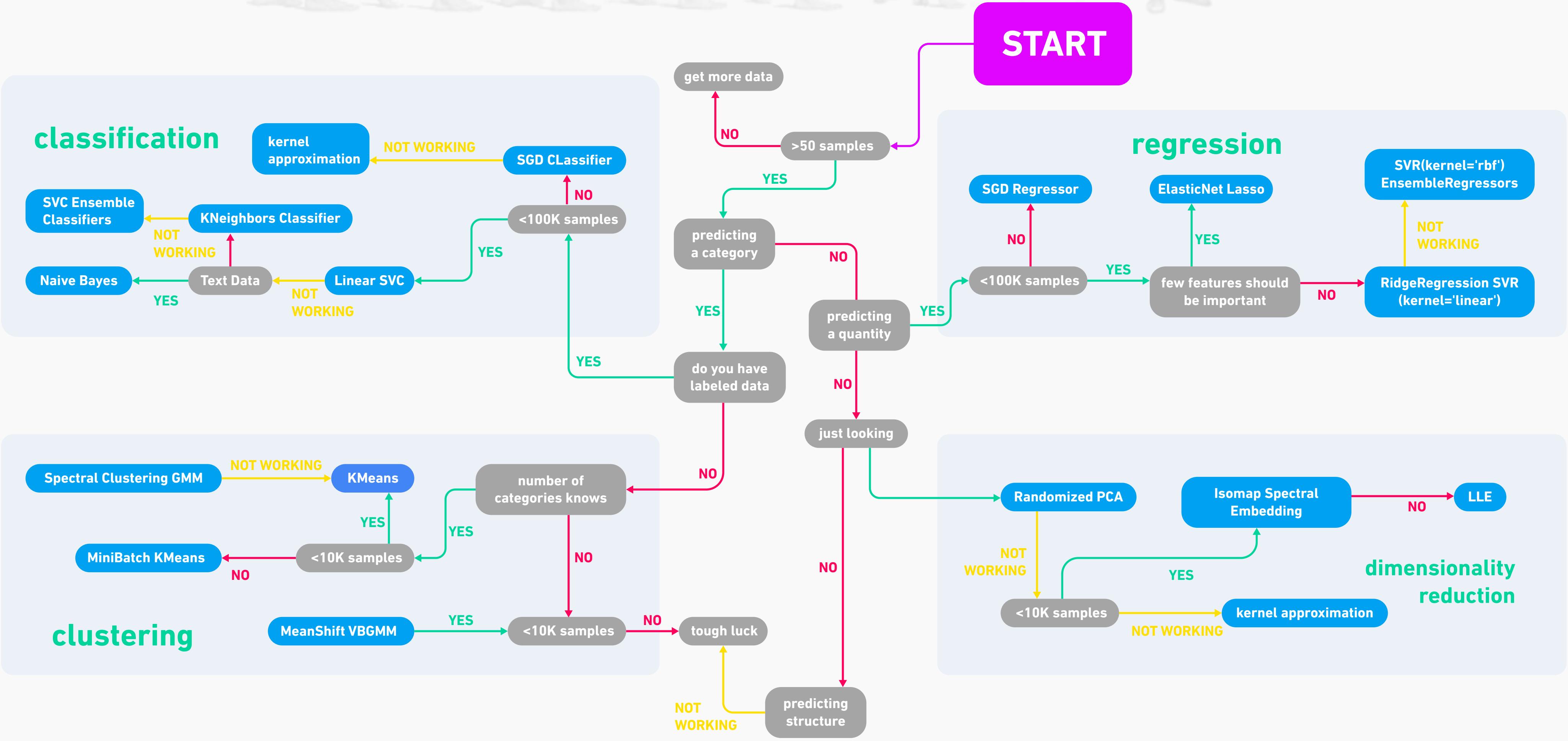
Randomized Parameter Optimization

```
>>> 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_)
```

Content Copyright by DataCamp.com. Design Copyright by BecomingHuman.Ai. **See Original here.**

Skicit-learn Algorithm

BecomingHuman.AI



START

get more data

NO

>50 samples

YES

predicting a category

NO

predicting a quantity

YES

do you have labeled data

YES

NO

just looking

NO

predicting structure

NOT WORKING

tough luck

classification

kernel approximation

NOT WORKING

SGD CLassifier

NO

<100K samples

YES

Linear SVC

NOT WORKING

Text Data

YES

Naive Bayes

KNeighbors Classifier

NOT WORKING

SVC Ensemble Classifiers

regression

SGD Regressor

NO

<100K samples

YES

few features should be important

YES

ElasticNet Lasso

NO

RidgeRegression SVR (kernel='linear')

NOT WORKING

SVR(kernel='rbf') EnsembleRegressors

clustering

number of categories knows

YES

<10K samples

YES

KMeans

NOT WORKING

Spectral Clustering GMM

NO

MiniBatch KMeans

NO

<10K samples

YES

MeanShift VBGMM

NO

tough luck

dimensionality reduction

Randomized PCA

NOT WORKING

<10K samples

YES

Isomap Spectral Embedding

NO

LLE

NOT WORKING

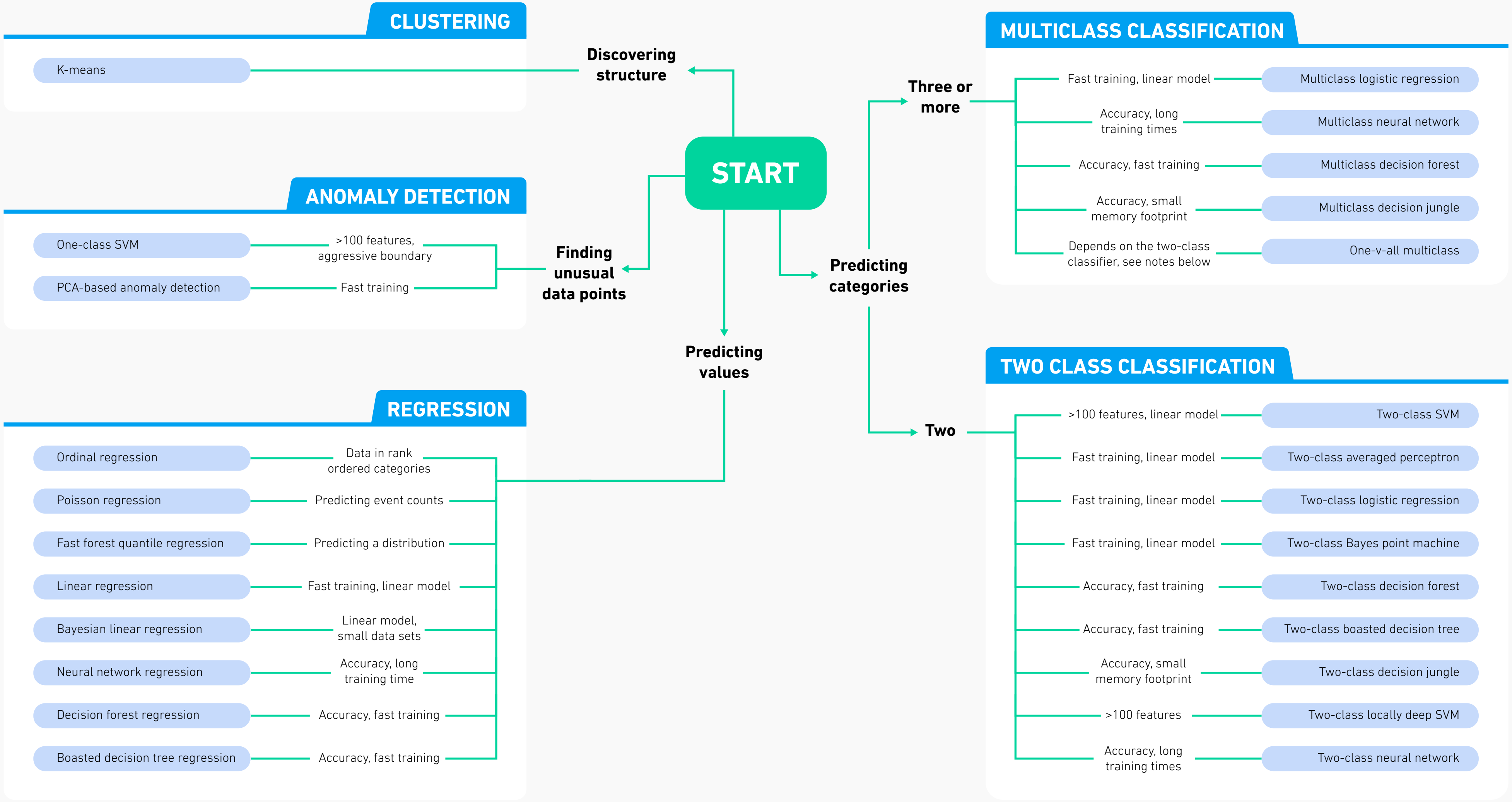
kernel approximation

Created by Skikit-Learn.org BSD Licence. See Original here.

Algorithm Cheat Sheet

BecomingHuman.AI

This cheat sheet helps you choose the best Azure Machine Learning Studio algorithm for your predictive analytics solution. Your decision is driven by both the nature of your data and the question you're trying to answer.



START

Discovering structure

CLUSTERING

- K-means

Finding unusual data points

ANOMALY DETECTION

One-class SVM	>100 features, aggressive boundary
PCA-based anomaly detection	Fast training

Predicting values

REGRESSION

Ordinal regression	Data in rank ordered categories
Poisson regression	Predicting event counts
Fast forest quantile regression	Predicting a distribution
Linear regression	Fast training, linear model
Bayesian linear regression	Linear model, small data sets
Neural network regression	Accuracy, long training time
Decision forest regression	Accuracy, fast training
Boasted decision tree regression	Accuracy, fast training

Predicting categories

Three or more

MULTICLASS CLASSIFICATION

Fast training, linear model	Multiclass logistic regression
Accuracy, long training times	Multiclass neural network
Accuracy, fast training	Multiclass decision forest
Accuracy, small memory footprint	Multiclass decision jungle
Depends on the two-class classifier, see notes below	One-v-all multiclass

Two

TWO CLASS CLASSIFICATION

>100 features, linear model	Two-class SVM
Fast training, linear model	Two-class averaged perceptron
Fast training, linear model	Two-class logistic regression
Fast training, linear model	Two-class Bayes point machine
Accuracy, fast training	Two-class decision forest
Accuracy, fast training	Two-class boasted decision tree
Accuracy, small memory footprint	Two-class decision jungle
>100 features	Two-class locally deep SVM
Accuracy, long training times	Two-class neural network

Creative Commons Microsoft Azure Machine Learning Algorithm by Microsoft is licensed under **CC BY SA 4.0**