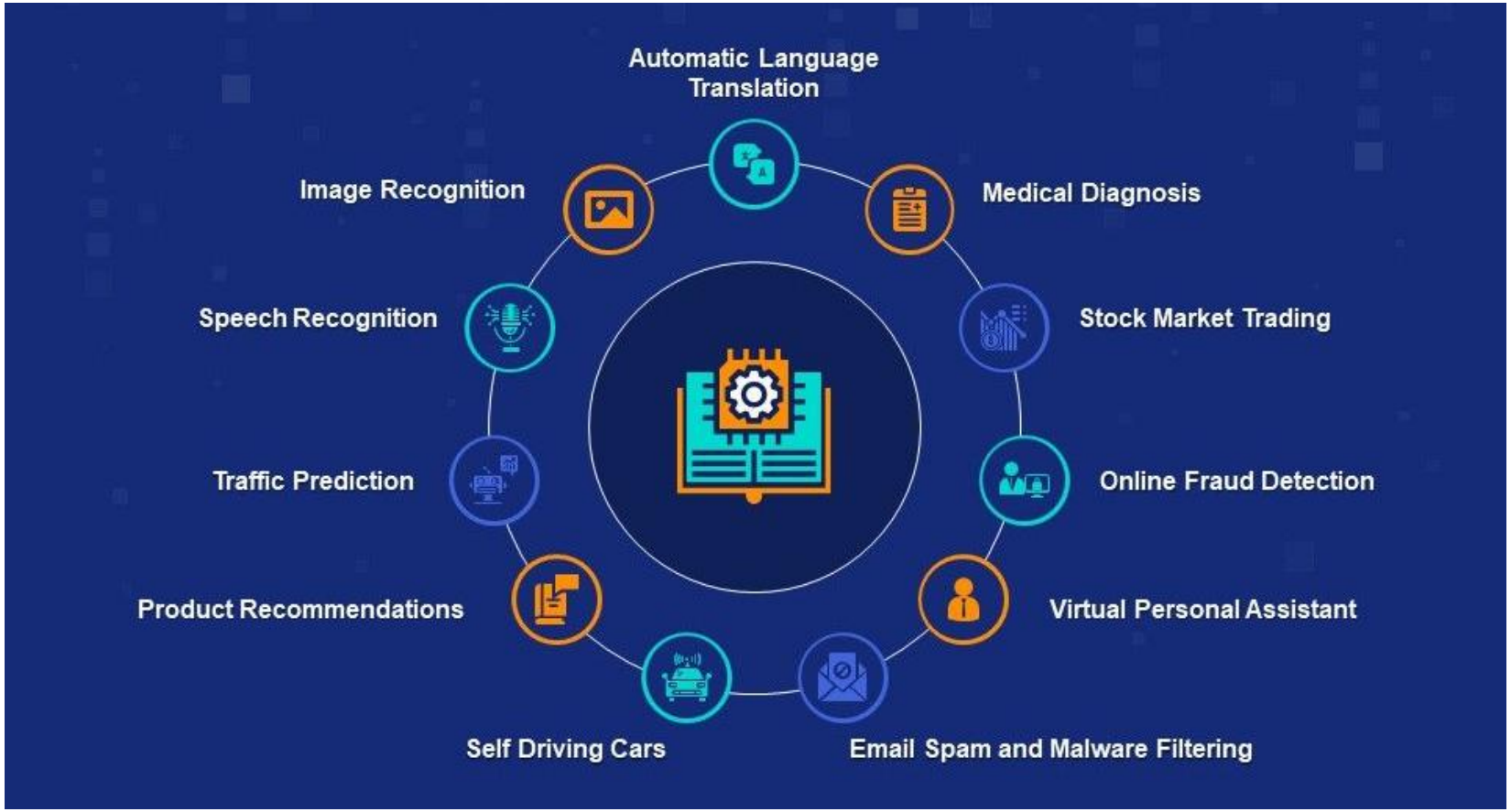
INTRODUCTION TO MACHINE LEARNING ALGORITHMS

odot

OTDMUG, SPRING 2025

SUMMARY

MACHINE LEARNING APPLICATIONS



Source: [Applications of Machine learning - SwissCognitive – The Global AI Hub]

MACHINE LEARNING AT USPS

The United States Postal Service

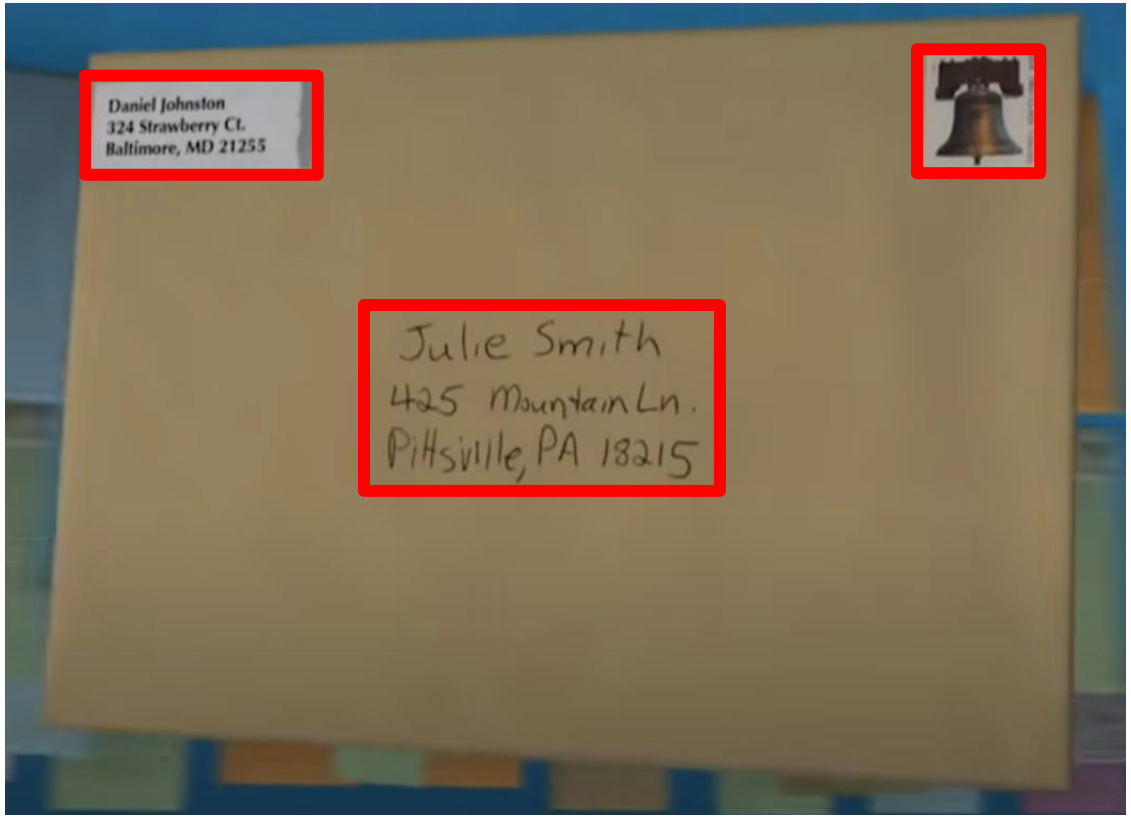
Artificial intelligence technology enables efficiency

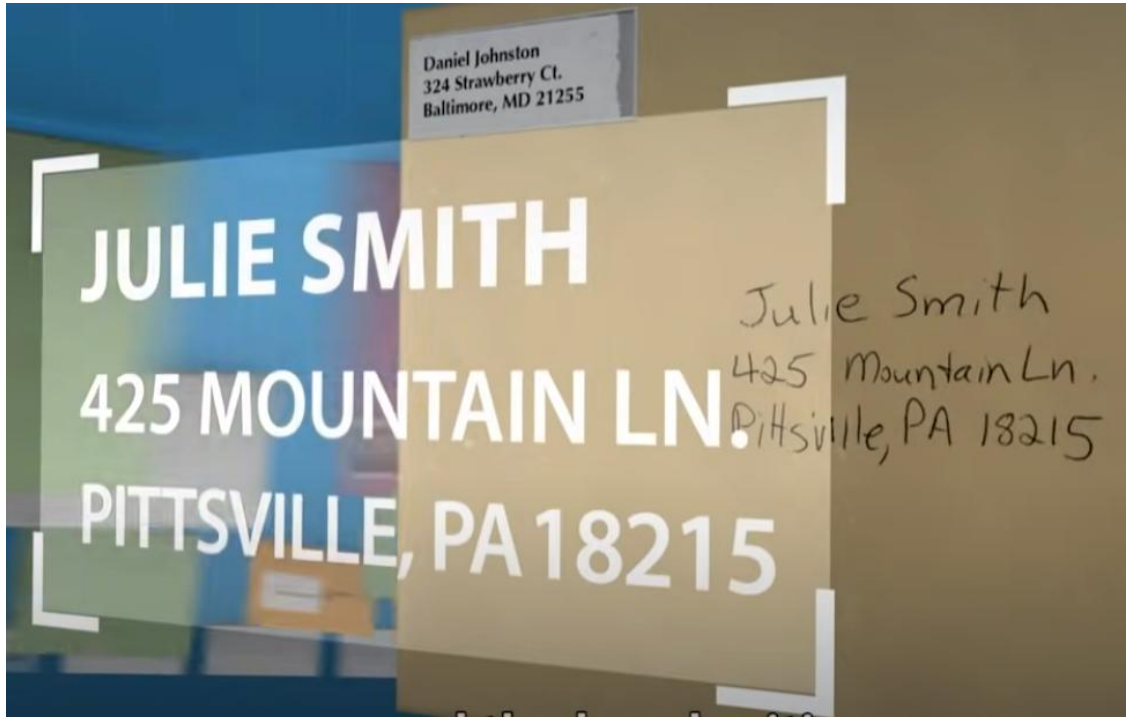
Venu Govindaraju's work in handwriting recognition was at the center of the first handwritten address interpretation system used by the United States Postal Service (USPS). USPS issued a contract to researchers at the University at Buffalo to develop the handwriting recognition technology. One year after implementation it saved the USPS $90 million by automatically processing, and barcoding for precise delivery, more than 25 billion letters. The 2009 Computing Community Consortium dubbed the project as "one of the most successful applications of Machine Learning for developing a real-time engineered system."

This project started in 1996

Source: Case Study - AI at USPS - Vice President Research and Economic Development - University at Buffalo
USPS video presentation: Systems at Work

MACHINE LEARNING AT USPS





USPS video presentation: [Systems at Work]

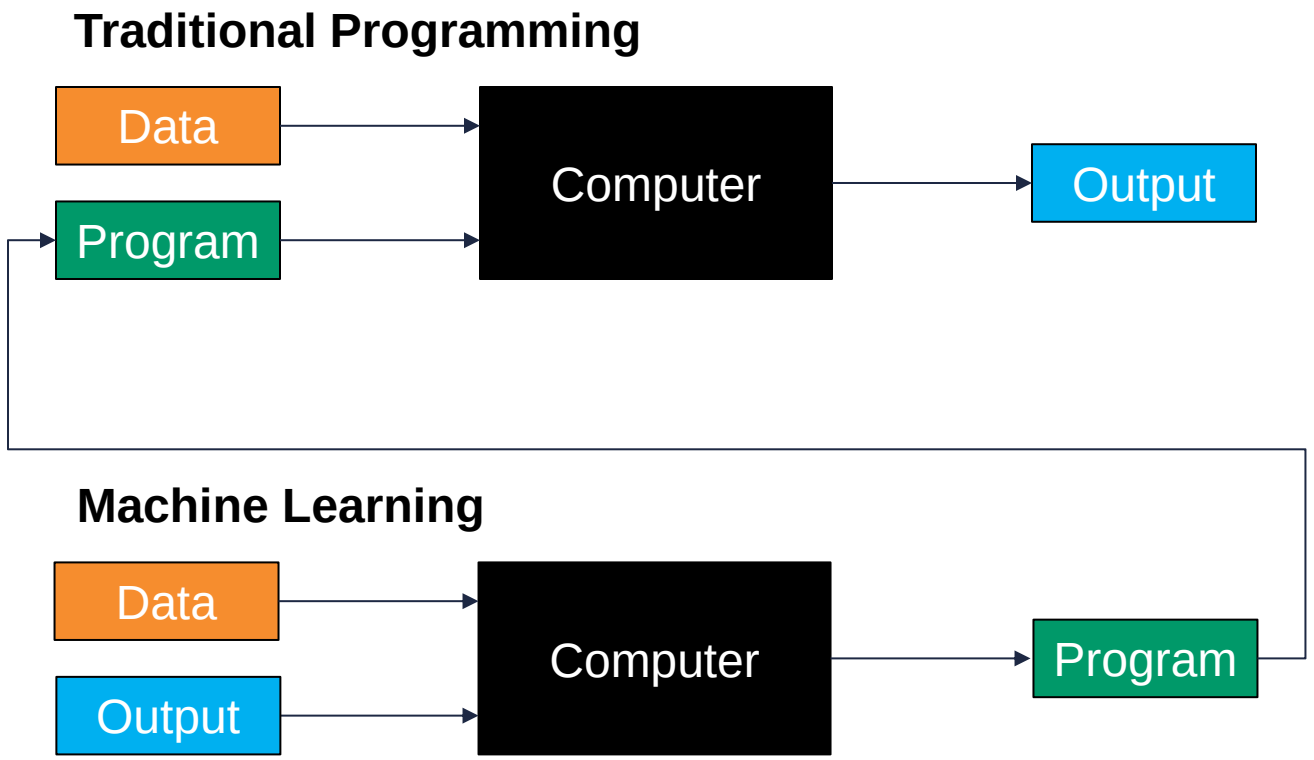
MACHINE LEARNING AT USPS

- Thousands of scanning devices and cameras used in processing packages.
- Processed more than **129 billion pieces of mail** and about **7.3 billion packages** in 2020.
- Generated more than **20TB of image data every day** in 2020.

"They do image classification and object detection on their packages"

Source: How the USPS Is Using AI at the Edge to Improve Mail

TRADITIONAL PROGRAMMING VS MACHINE LEARNING

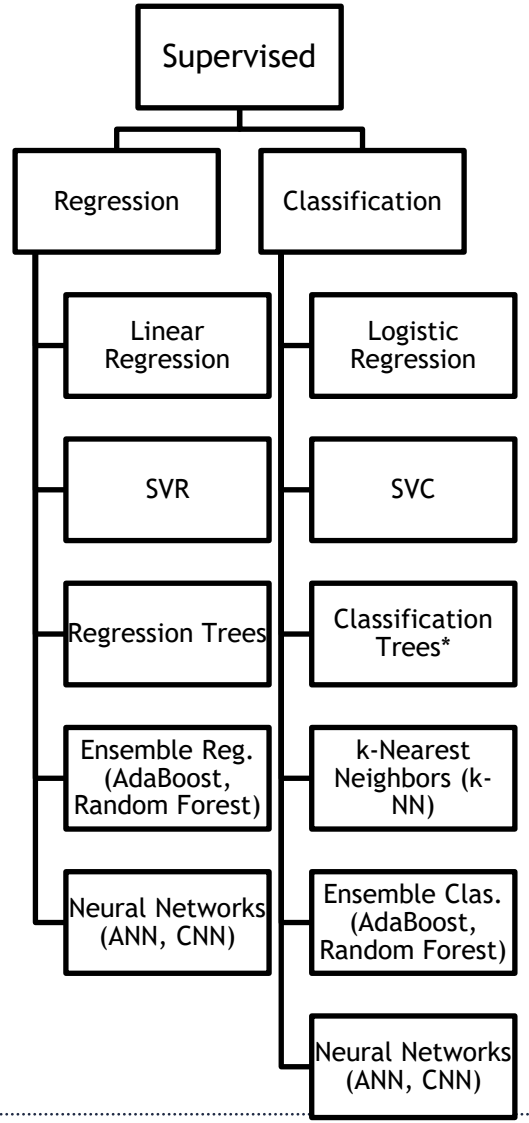


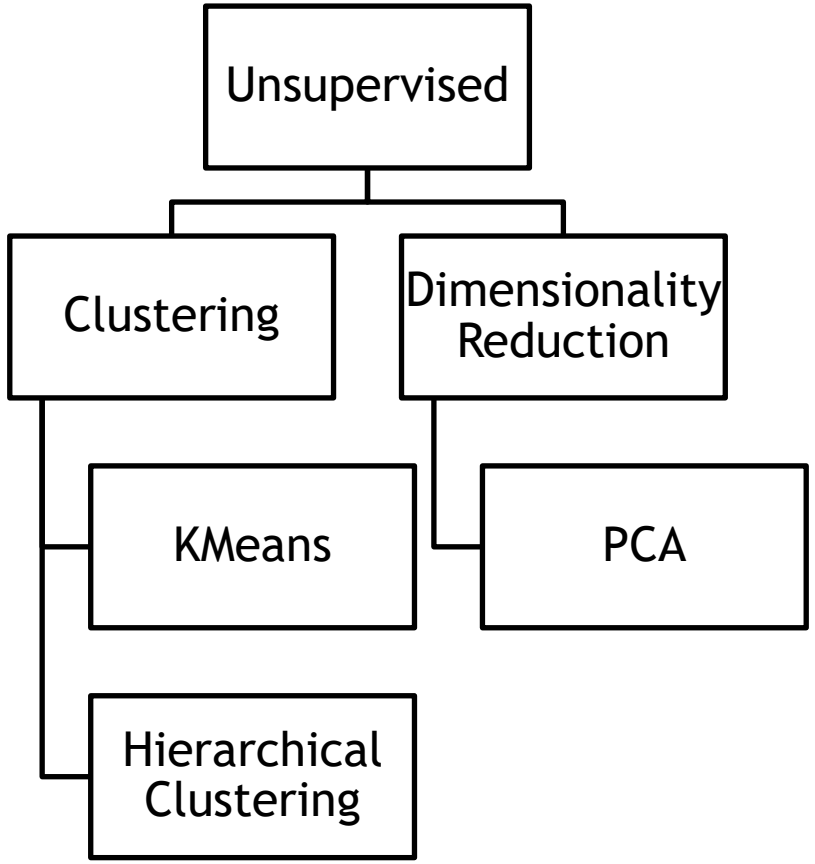
SUPERVISED VS UNSUPERVISED

- Supervised Learning
  - Labeled data (output is known)
  - Fit models that map inputs and outputs
  - Make predictions on unseen data
  - Regression and Classification

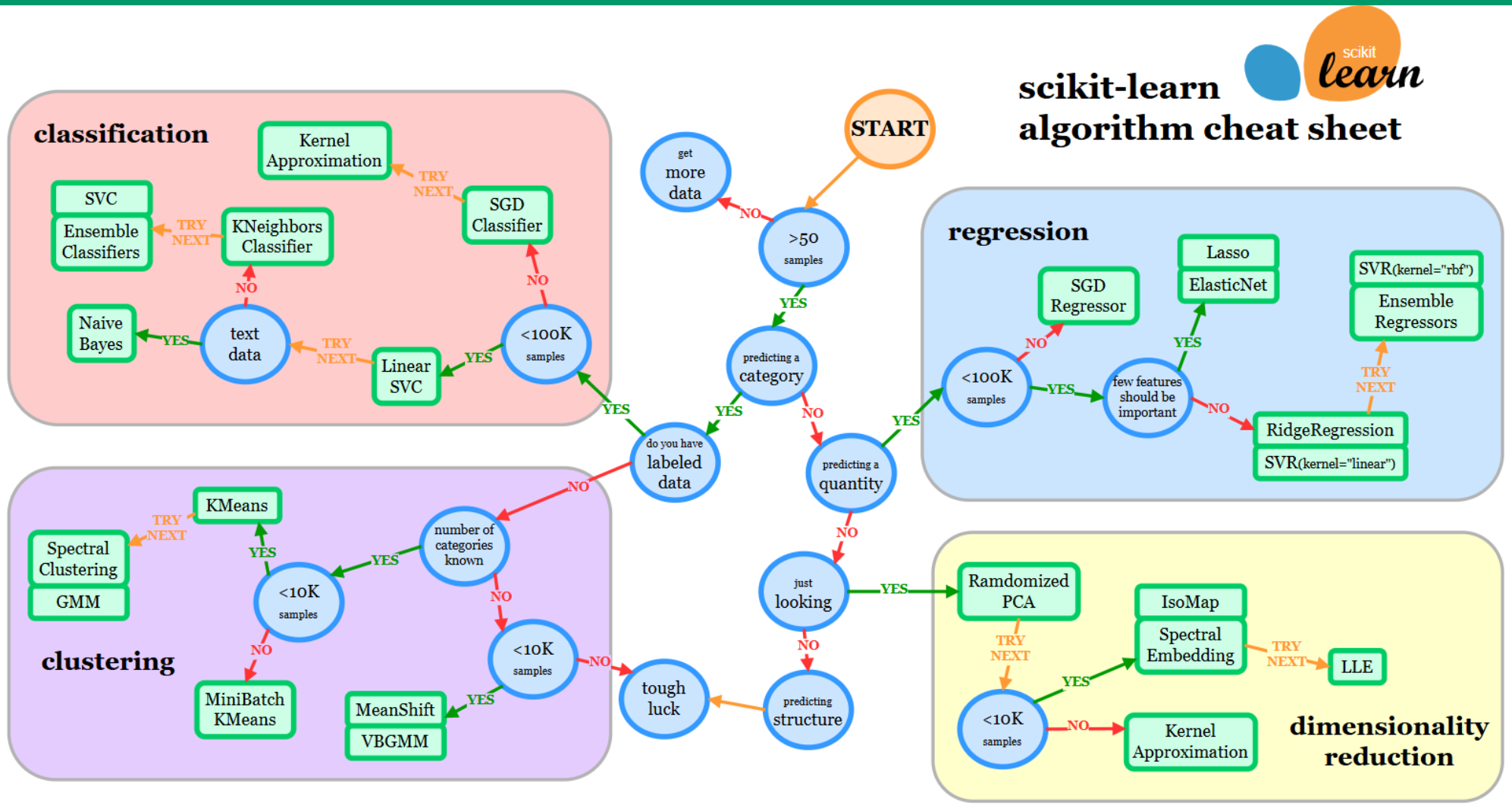
- Unsupervised Learning
  - Unlabeled data (output is unknown)
  - Discover hidden patterns within the data
  - Recognize groups/structures
  - Clustering, Dimensionality Reduction, Feature Extraction, Association, Recomendation

MACHINE LEARNING ALGORITHMS



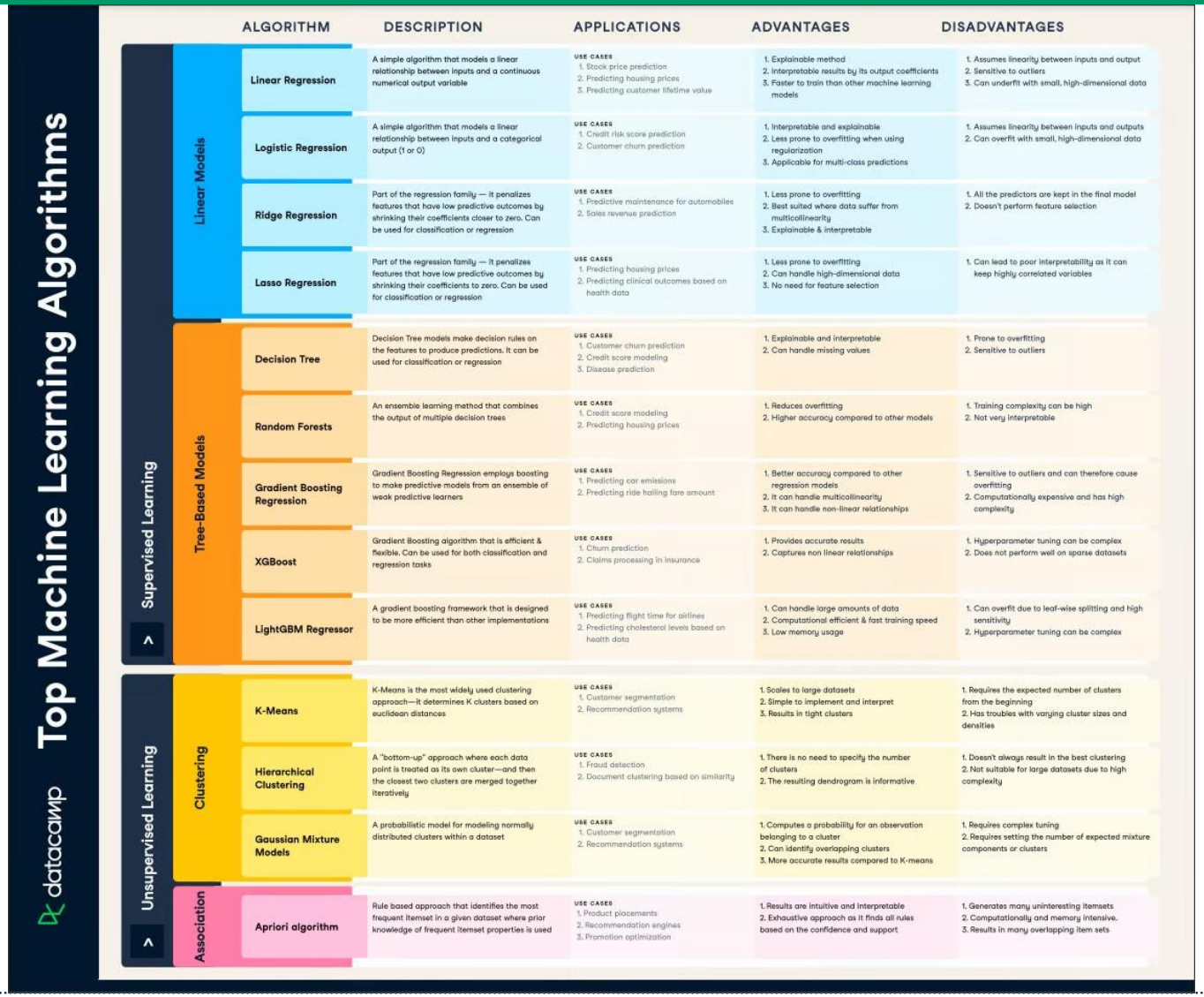


SKLEARN CHEAT SHEET



Source: [12. Choosing the right estimator — scikit-learn 1.6.1 documentation]

DATACAMP CHEAT SHEET



Source: [Machine Learning Cheat Sheet | DataCamp]

DATACAMP CHEAT SHEET

Supervised Learning

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

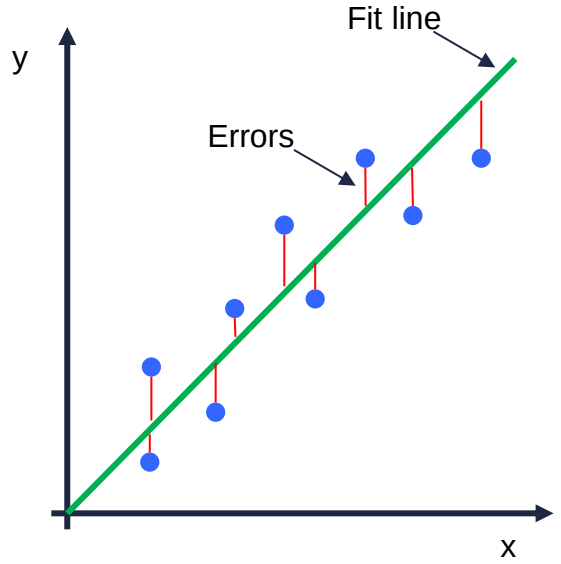
Source: [Machine Learning Cheat Sheet | DataCamp]

dot

SUPERVISED LEARNING

1. Linear Regression

- Assumes a straight-line relationship between dependent and independent variables (MLR when multiple independent variables)
- Minimizes the errors between observed and predicted values (OLS)
- Simple, easy to implement and interpret, base model for comparisons

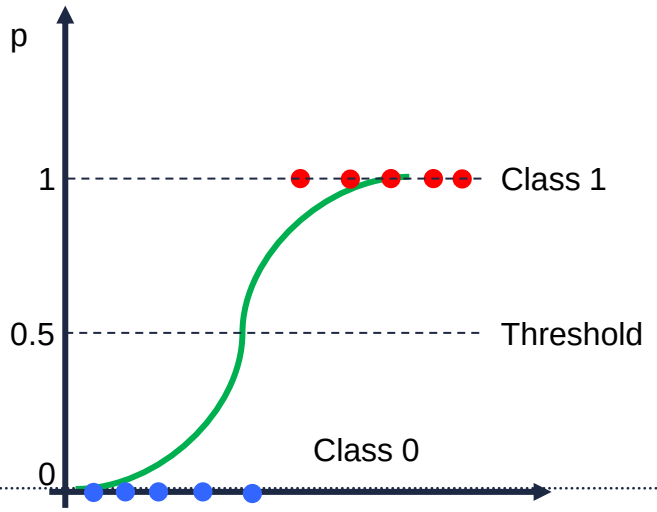



```
from sklearn.linear_model import LinearRegression
reg = LinearRegression().fit(X_train, y_train) # fit/train the model
reg.predict(X_test) # predicts y on new data
reg.score_ # R-squared
reg.coef_ # estimated coefficients
reg.intercept_ # intercept term
```

2. Logistic Regression

- Despite its name, it is used for (binary) classification
- Uses a sigmoid function to calculate probabilities
- Maximizes the likelihood (probabilities of observing the actual outcome)
- Simple, easy to interpret, computationally efficient
- Multinomial Logistic Regression for multiple classes. Softmax function instead.

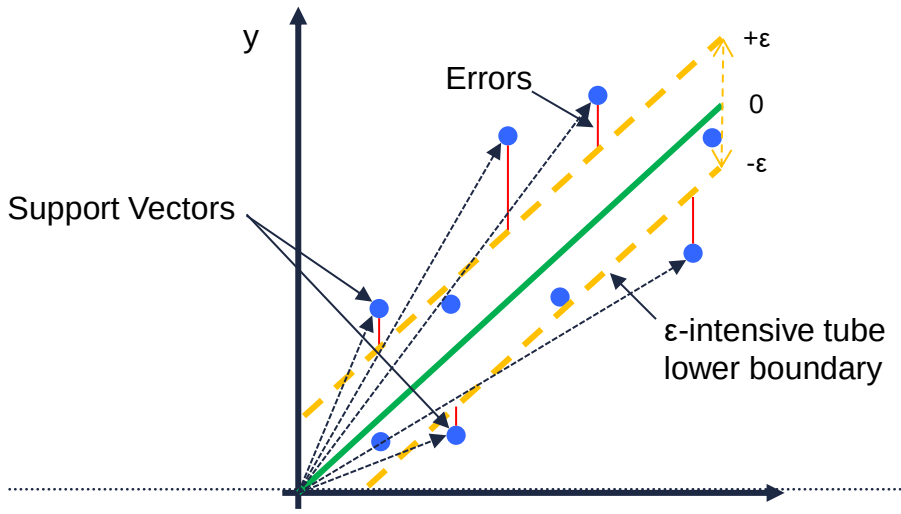
$$S(x) = \frac{1}{1 + e^{-x}}$$




```
from sklearn.linear_model import LogisticRegression
clf = LogisticRegression().fit(X_train, y_train) # fit/train the model
# X is a vector of continuous/discrete variables
# y must be discrete
clf.predict(X_test) # predicts classes on new data
clf.predict_proba(X_test) # predicts probabilities on new data
clf.score_ # mean accuracy
```

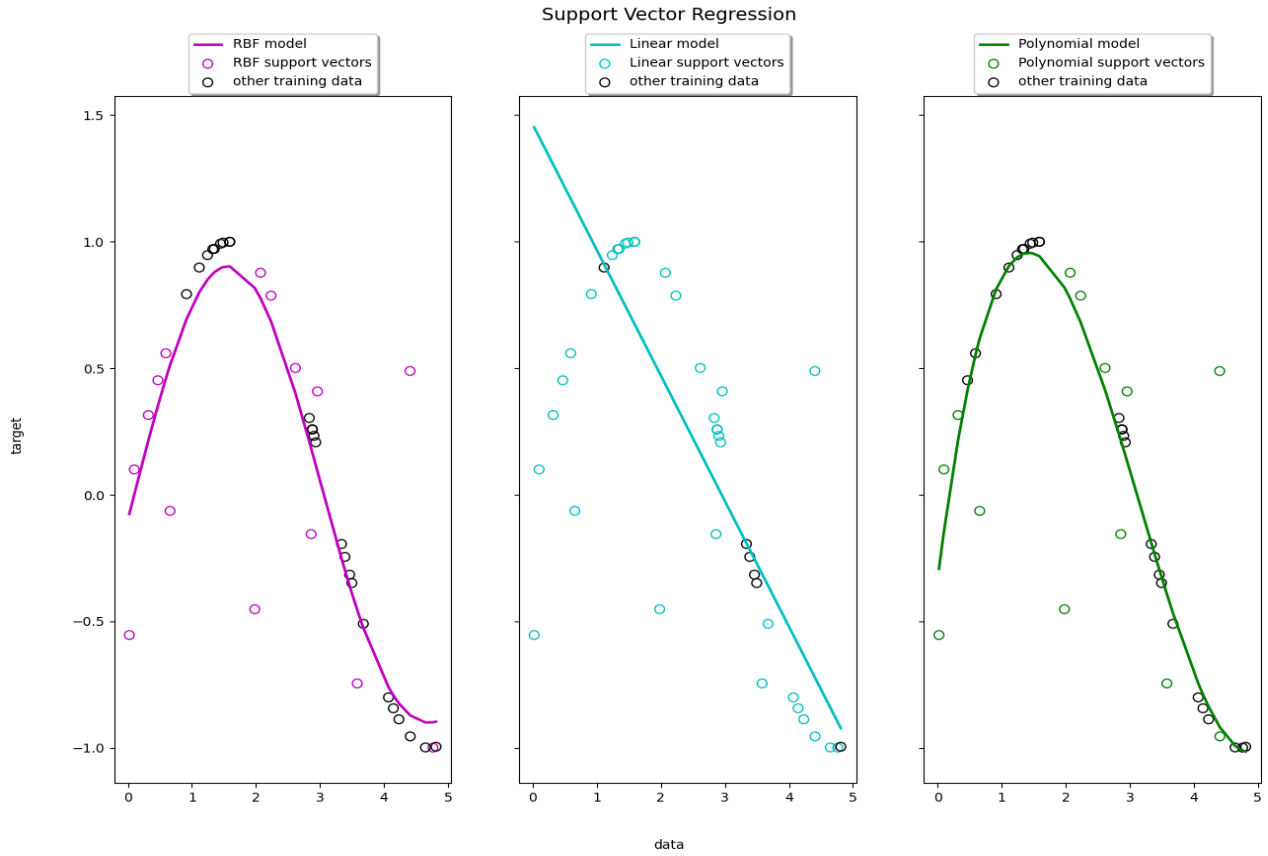
3. Support Vector Machines (SVM)

- Regression or Classification
- Uses Kernels (functions) to map (transform) data to a higher-dimensional space that makes a linear separation possible
- Minimizes the errors between the walls of the ε-intensive tube and the predicted values




```
from sklearn.svm import SVR # regressor
reg = SVR(kernel="linear").fit(X_train, y_train)
# kernel = “linear”, “rbf”, “poly”, “sigmoid”
reg.predict(X_test) # predicts y on new data
reg.score_ # R-squared
reg.coef_ # estimated coefficients
reg.intercept_ # intercept term
```

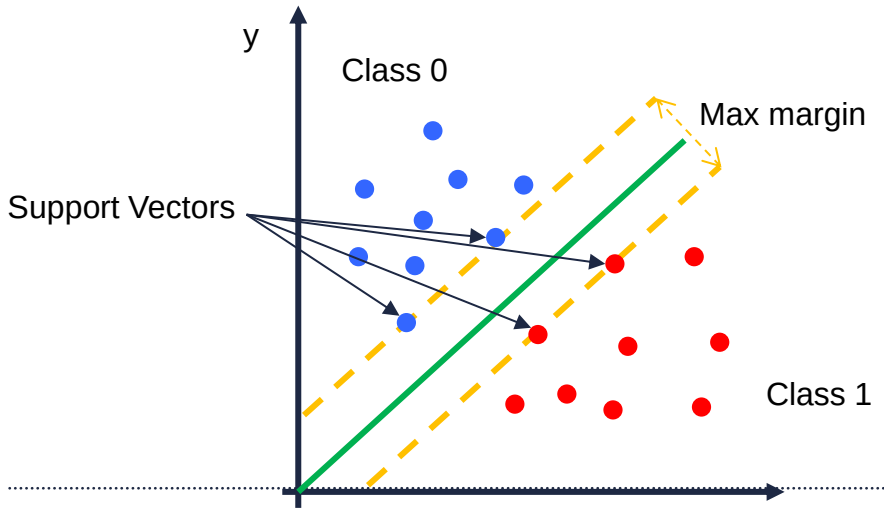
3. Support Vector Machines (SVM)



Source: [Support Vector Regression (SVR) using linear and non-linear kernels — scikit-learn 1.6.1 documentation]

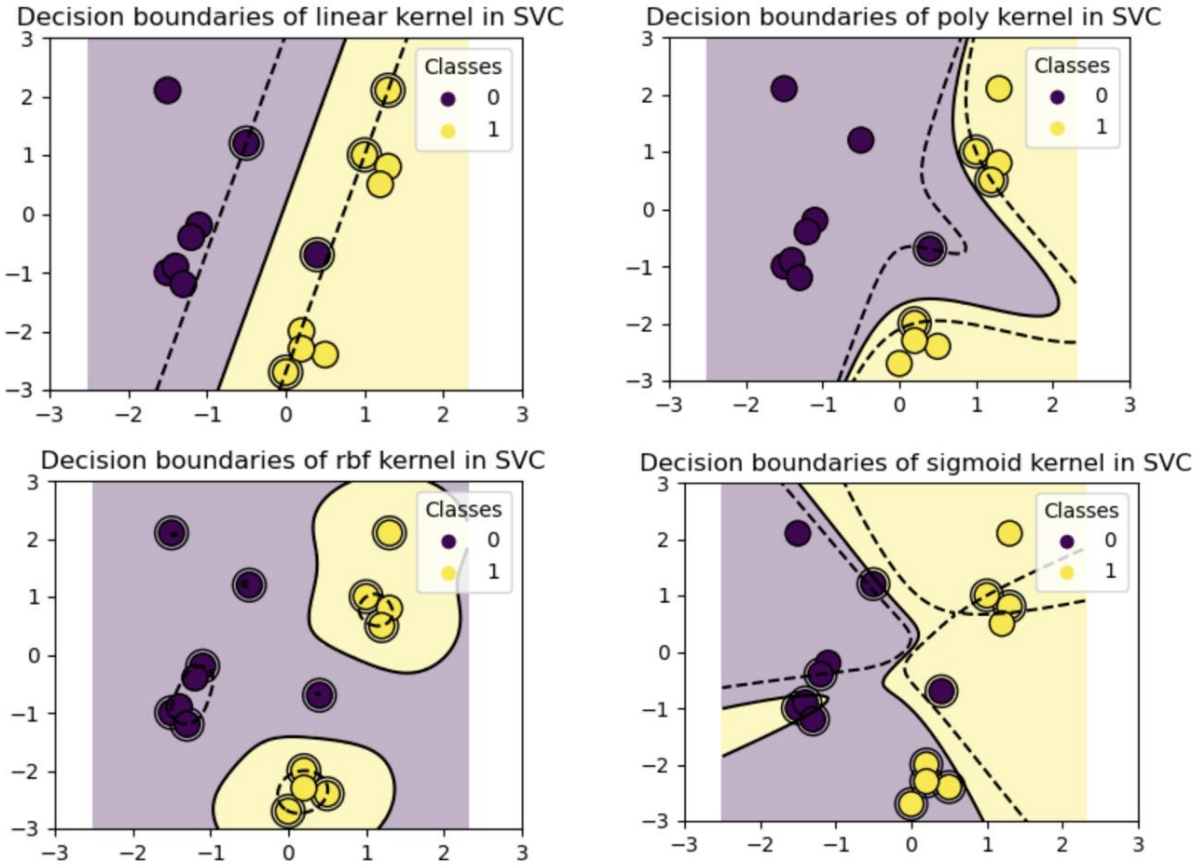
3. Support Vector Machines (SVM)

- Regression or Classification
- Maximizes the margin while minimizing classification errors
- Regularization parameter “C” controls the trade-off between margin and errors (higher C prioritizes minimizing errors)
- Hyperparameter tuning techniques (e.g., cross-validation) help to find a good C




```
from sklearn.svm import SVC # classifier
clf = SVC(kernel="linear“, C=1) .fit(X_train, y_train)
# kernel = “linear”, “rbf”, “poly”, “sigmoid”
clf.predict(X_test) # predicts classes on new data
clf.predict_proba(X_test) # predicts probabilities on new data
clf.score_ # mean accuracy
```

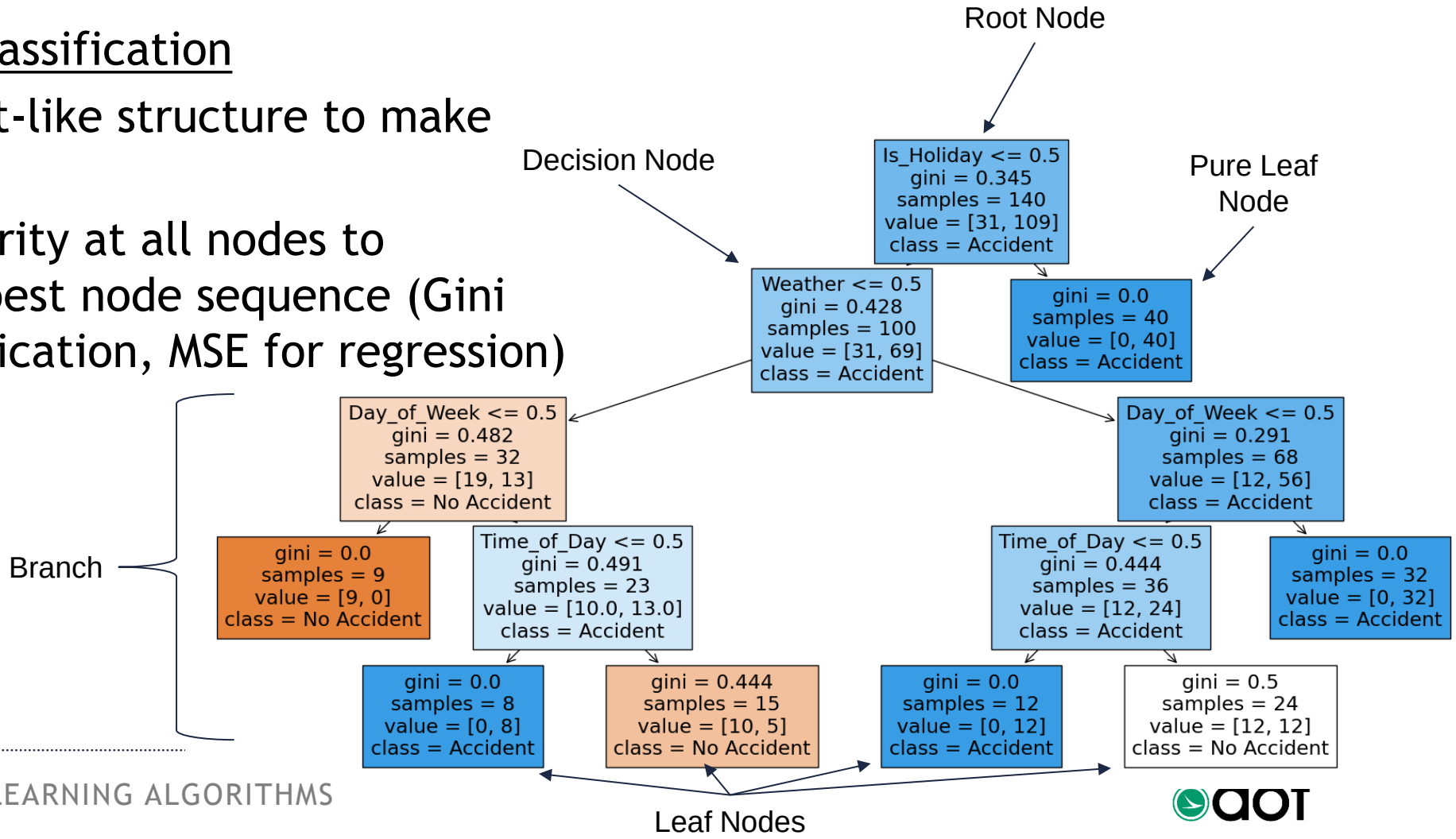
3. Support Vector Machines (SVM)



Source: Demystifying Support Vector Machines: Kernel Machines

4. Decision Trees

- Regression or Classification
- Uses a flowchart-like structure to make decisions
- Calculates impurity at all nodes to determine the best node sequence (Gini index for classification, MSE for regression)



4. Decision Trees

- Dataset: Weather, Time of Day, Day of Week, Is Holiday, and Accident.
- We will try to predict (classify) whether an accident will occur in times of day under certain conditions.

	Weather	Time_of_Day	Day_of_Week	Is_Holiday	Accident
0	Foggy	Morning	Weekend	0	Yes
1	Clear	Afternoon	Weekend	1	Yes
2	Foggy	Morning	Weekend	0	Yes
3	Foggy	Afternoon	Weekend	1	Yes
4	Clear	Afternoon	Weekday	0	No
5	Clear	Evening	Weekday	0	No
6	Foggy	Afternoon	Weekend	0	Yes
7	Rainy	Evening	Weekday	1	Yes
8	Foggy	Morning	Weekday	1	Yes
9	Foggy	Morning	Weekend	0	Yes

```
X = data.drop('Accident', axis=1)
y = data['Accident']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)
```

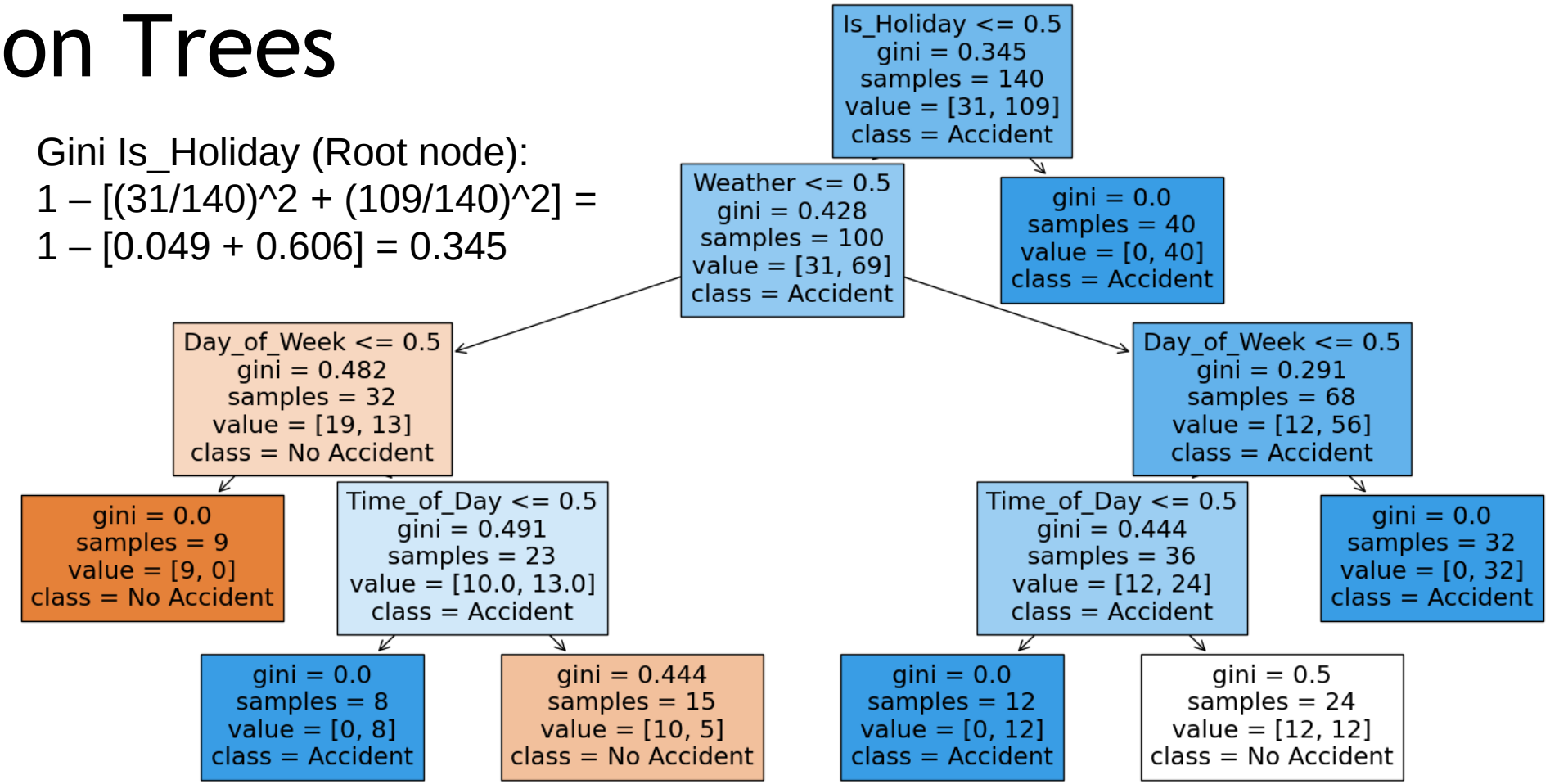
```
clf = DecisionTreeClassifier(random_state=0, max_depth=4)
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
y_predprob = clf.predict_proba(X_test)
y_score = clf.score(X_test, y_test)

print(f"Score Mean Accuracy: {y_score:.3f}")
```

```
Score Mean Accuracy: 0.917
```

4. Decision Trees

$$Gini = 1 - \sum_{i=1}^{C} (p_i)^2$$

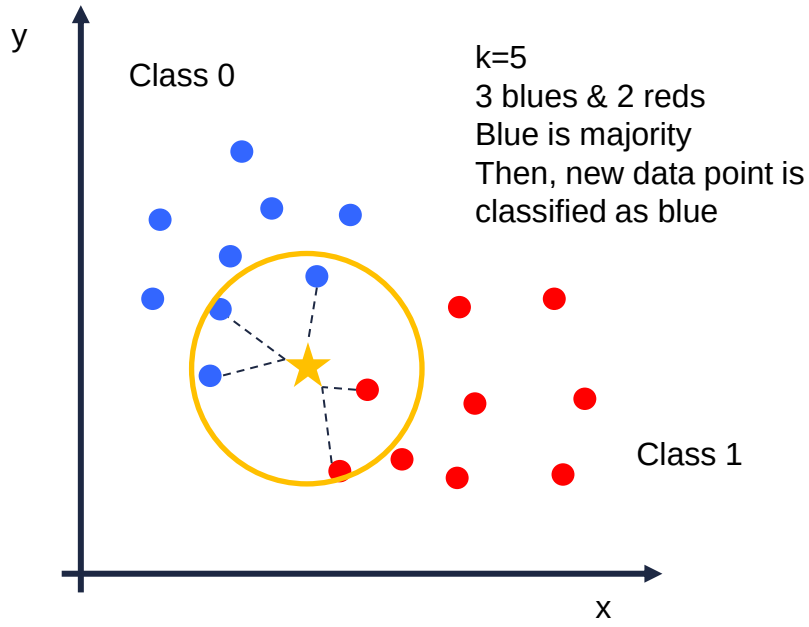
Gini Is_Holiday (Root node):
1 – [(31/140)^2 + (109/140)^2] =
1 – [0.049 + 0.606] = 0.345



Variable with lower Gini is picked

5. k-Nearest Neighbors

- Classification and Regression
- Classifies by finding the k closest data points (neighbors) to a new data point
- Voting system where the majority of the neighbors win
- For regression, the result will be the average of the k neighbors
- Lazy learner. No training, just computes distances (Euclidean)
- Simple, very inefficient with large datasets, curse of dimensionality

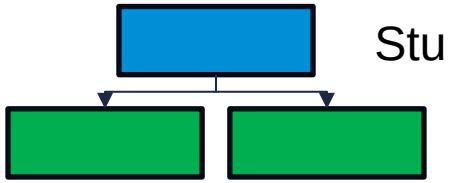


6. Boosting (AdaBoost)

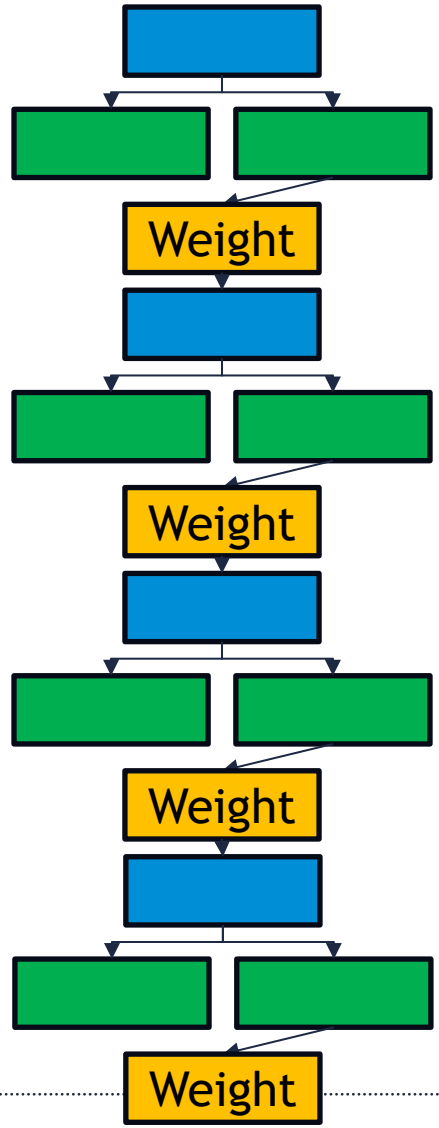
- Regression and Classification
- Ensemble technique (not a model itself)
- Sequentially combines “weak” models (learners) to create a strong model
- Following models focus on correcting the errors of the previous models
- Final prediction is a weighted (performance) sum of the individual predictions
- Prone to overfitting and bias due to weighted samples

6. Boosting (AdaBoost)

- Simple Decision Tree (stump) as “weak” learner
- Calculate Gini to pick 1 variable for the first stump
- The misclassified records in this learner will have their weights increased
- Resample the dataset using weights (more of the misclassified records)
- Calculate Gini to pick…
- Final prediction will be weighted



Stump (root + leaves)



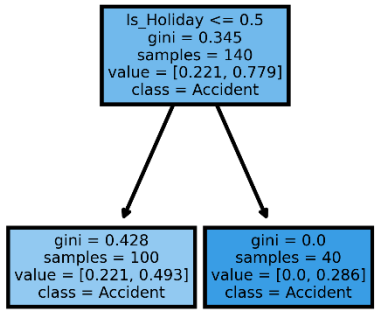
6. Boosting (AdaBoost)

- Need to pay attention to overfitting

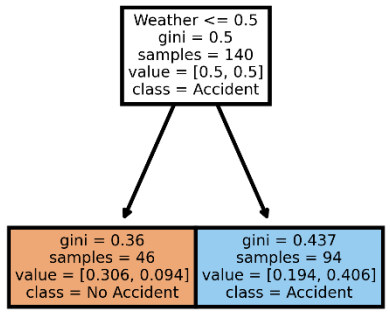
```
clf = AdaBoostClassifier(random_state=0, n_estimators=50)
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
y_predprob = clf.predict_proba(X_test)
y_score = clf.score(X_test, y_test)

print(f"Score Mean Accuracy: {y_score:.3f}")
```

```
Score Mean Accuracy: 1.000
```

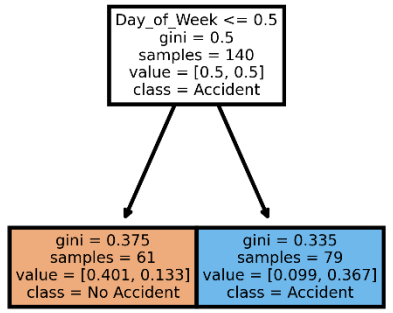
Estimator: 0



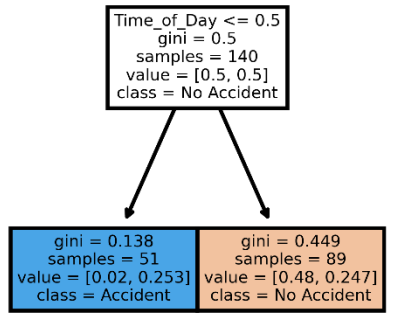
Estimator: 1



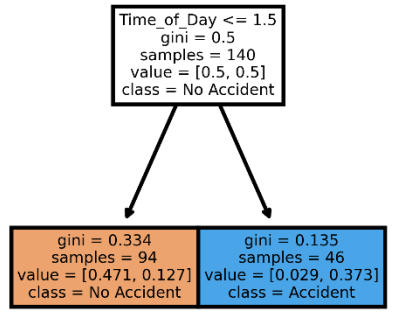
Estimator: 2



Estimator: 3



Estimator: 4



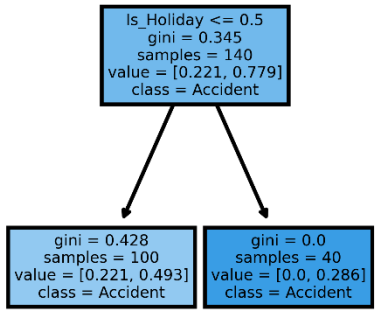
6. Boosting (AdaBoost)

- Need to pay attention to overfitting

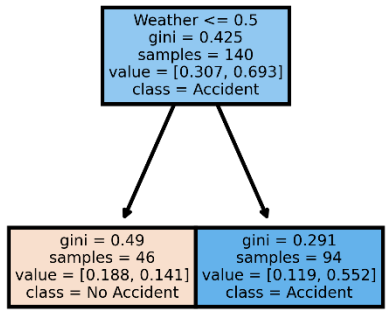
```
clf = AdaBoostClassifier(random_state=0, n_estimators=50, learning_rate=0.1)
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
y_predprob = clf.predict_proba(X_test)
y_score = clf.score(X_test, y_test)

print(f"Score Mean Accuracy: {y_score:.3f}")
```

```
Score Mean Accuracy: 0.967
```

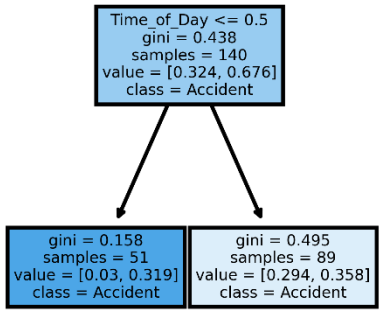
Estimator: 0



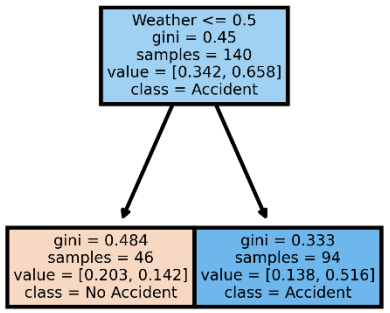
Estimator: 1



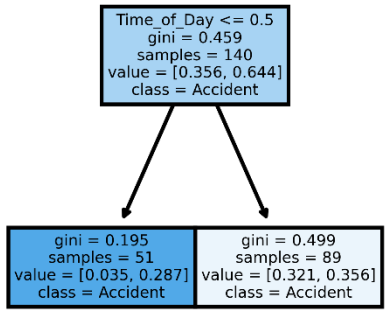
Estimator: 2



Estimator: 3



Estimator: 4

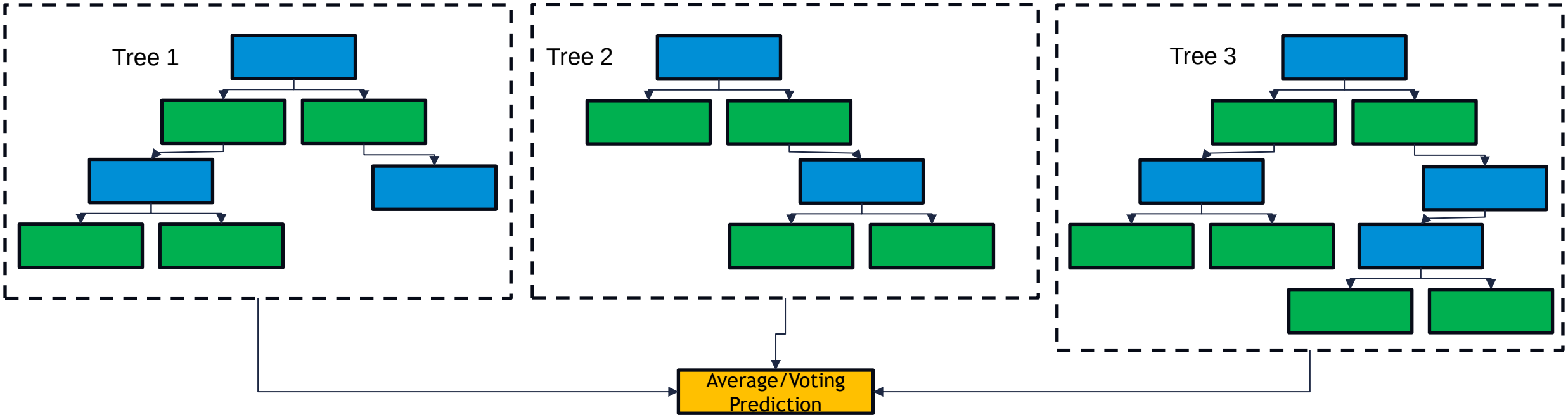


7. Bagging (Random Forest)

- Regression and Classification
- Ensemble technique (not a model itself)
- Considers learners independent and use them in parallel
- Random sampling + Random feature selection
- Final prediction is a simple voting system (class) or average (reg)
- Reduces overfitting and bias

7. Bagging (Random Forest)

- Trees are full decision trees (not stumps as in boosting)
- Each tree is trained on a subset of the dataset (random sampling)
- Each tree has a random set of features (# features is a hyperparameter)

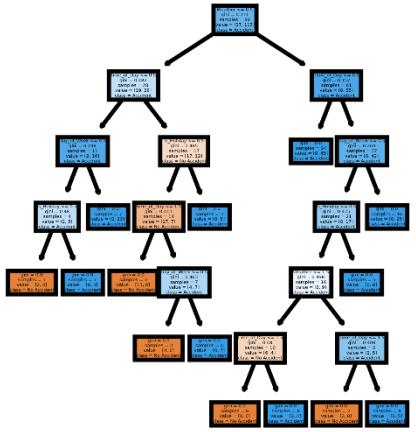


7. Bagging (Random Forest)

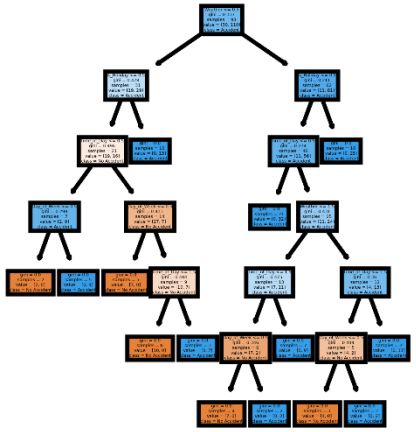
```
clf = RandomForestClassifier(random_state=0, n_estimators=100, max_depth=None, max_features='sqrt')
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
y_predprob = clf.predict_proba(X_test)
y_score = clf.score(X_test, y_test)

print(f"Score Mean Accuracy: {y_score:.3f}")
```

```
Score Mean Accuracy: 0.983
```

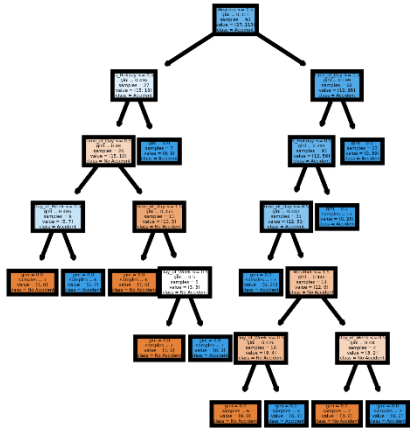
Estimator: 0



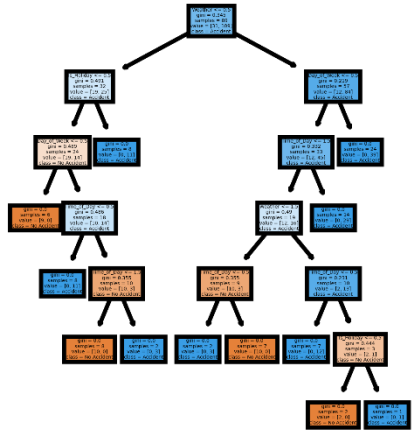
Estimator: 1



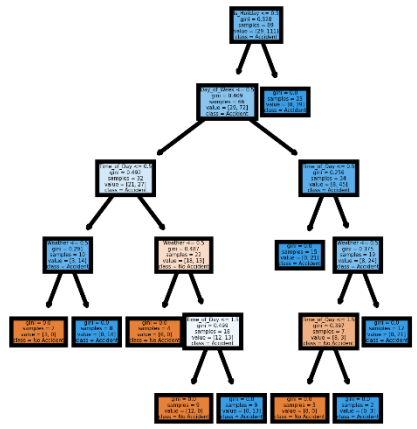
Estimator: 2



Estimator: 3



Estimator: 4

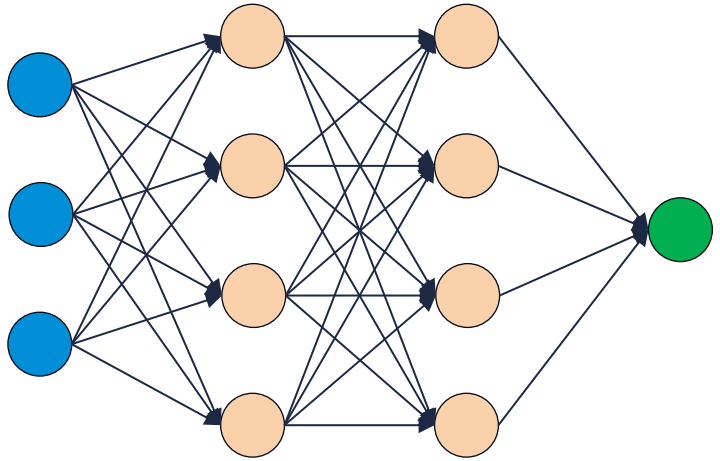


8. Artificial Neural Network (ANN)

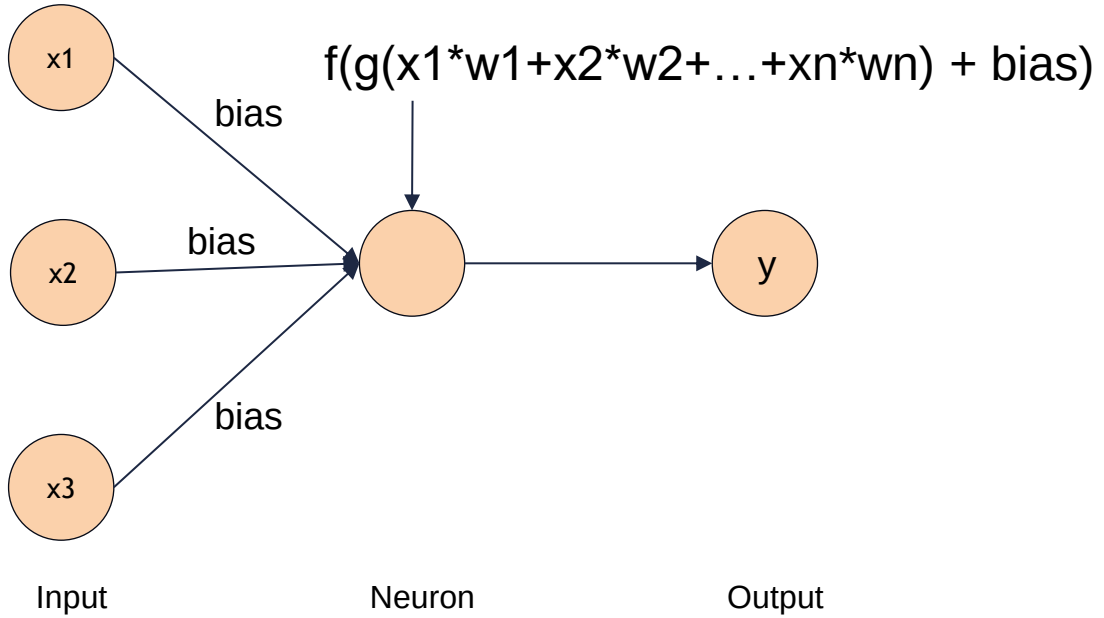
- Mimics how our brains work (node/neuron layers)
- Data is passed forward (feed forward) among nodes
- Uses Linear Regression (!) model at each node, but it introduces non-linearity with activation functions
- Each node output = f(g(x1*w1+x2*w2+…+xn*wn) + bias)
- Weights and biases are initialized randomly
- Activation function f(x) transforms the data

(ReLu, Sigmoid, tanh)

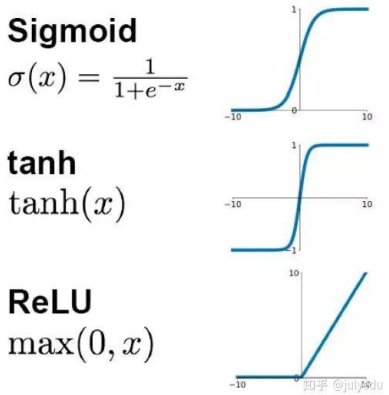
- There are no rules for sizing the network (test!)



8. Artificial Neural Network (ANN)

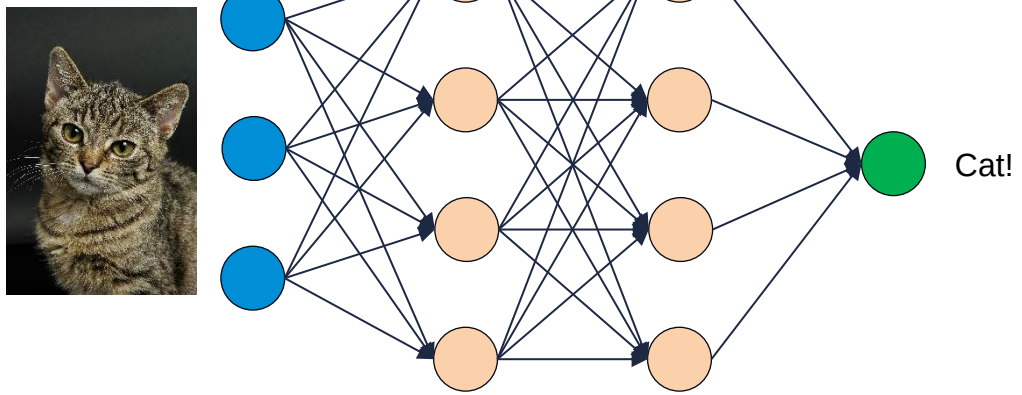


Activation functions (f)

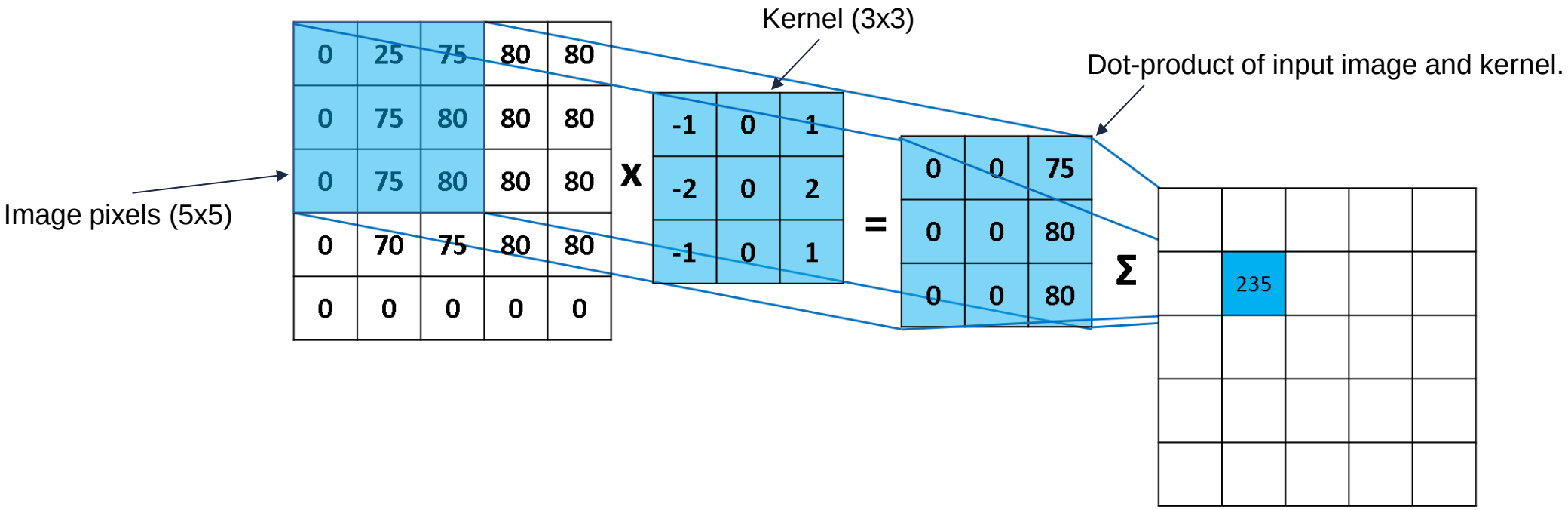


9. Convolutional Neural Network (CNN)

- CNNs are designed to process grid-like data as images
- Very powerful for object detection, image classification, and image segmenting
- The hidden layers are now called convolutional layers (convolution + activation + pooling)
- Convolution = sliding filters. Filters are randomly generated kernels (dot-product functions) that highlight unique characteristics in images (edges, texture, complex shapes)
- Pooling is another type of kernel but to reduce spatial dimensions (image size) while keeping the most important characteristics



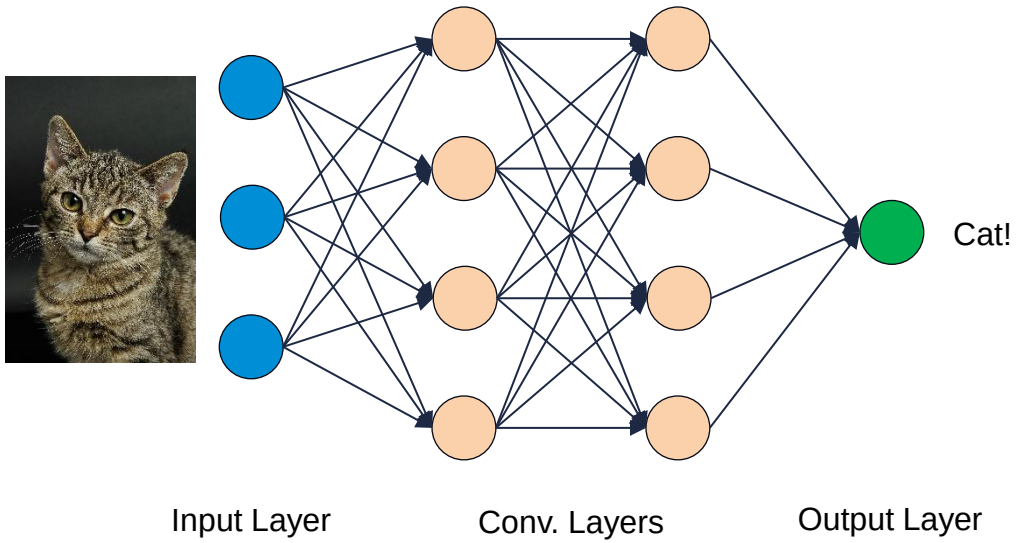
9. Convolutional Neural Network (CNN)



Source: [Convolutional Neural Networks - Basics · Machine Learning Notebook]

9. Convolutional Neural Network (CNN)

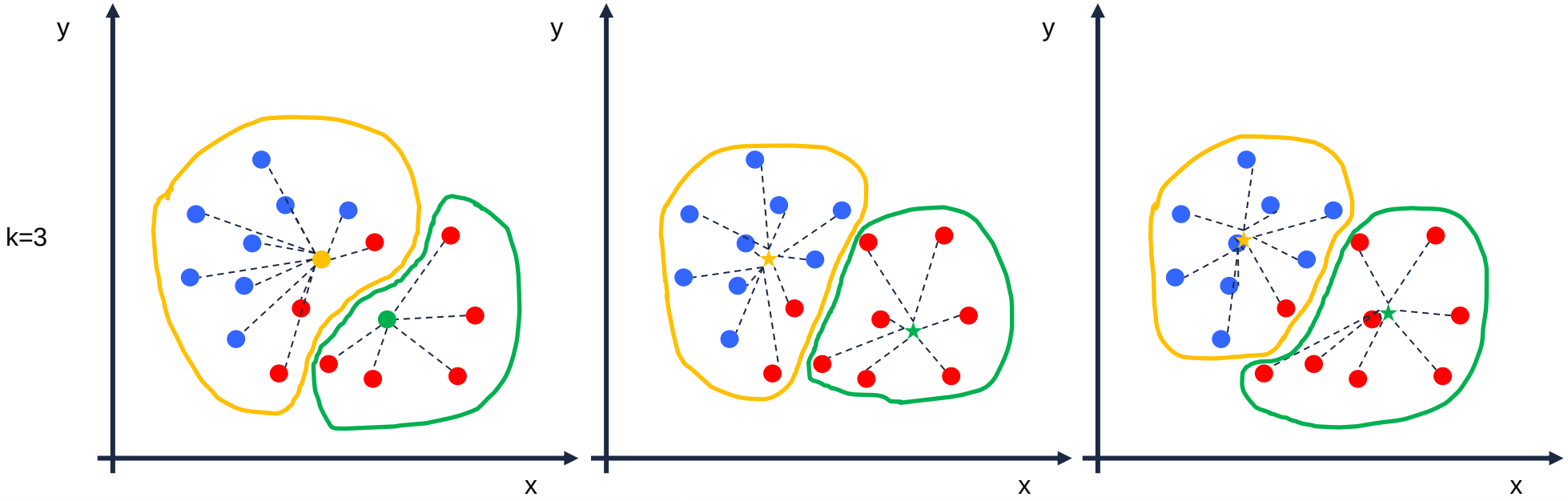
Dynamic illustration of CNNs: [CNN Explainer: Learn CNN in your browser!]



UNSUPERVISED LEARNING

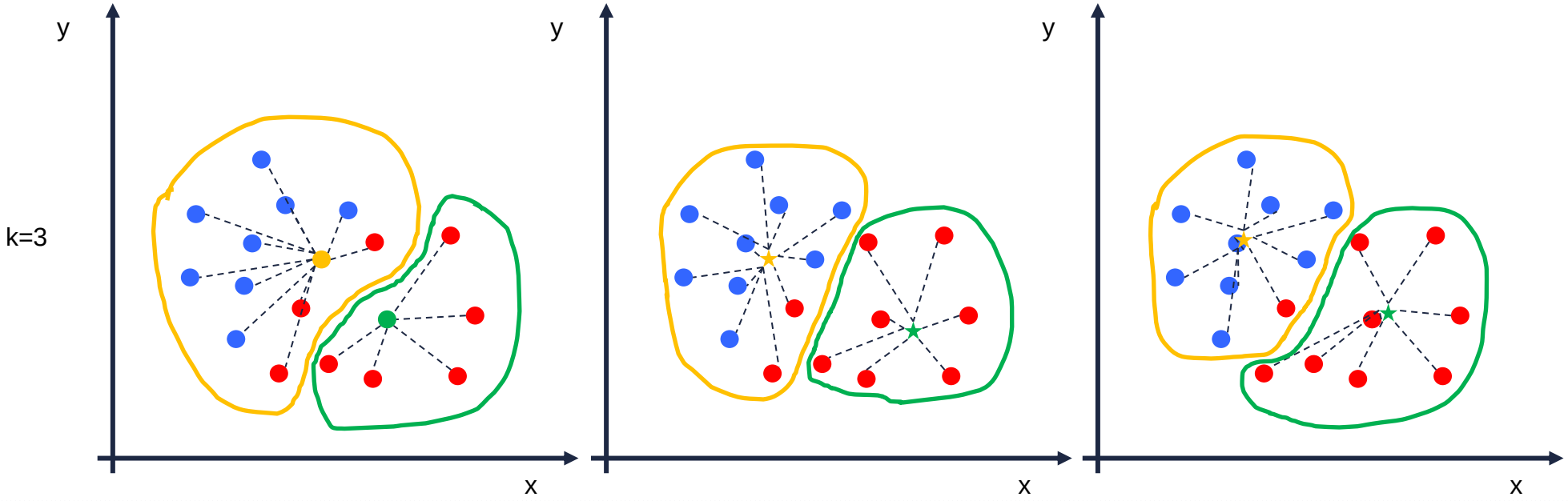
10. KMeans

- Clustering (we don't know the classes)
- Groups data points into clusters (similarity)
- k is the number of clusters and is a hyperparameter
- Minimizes distance between data points and cluster centers



10. KMeans

- Clustering (we don't know the classes)
- Groups data points into clusters (similarity)
- k is the number of clusters and is a hyperparameter
- Minimizes distance between data points and cluster centers



QUESTIONS