Data Science Cheatsheet 2.0

Last Updated March 19, 2021

Statistics

Discrete Distributions

Binomial - number of successes x in n events, each with p probability $\rightarrow \binom{n}{x} p^x q^{n-x}$, with mean = np. If n = 1, this is Bernoulli.
Geometric - first success with p probability on the n^{th} trial $\rightarrow q^{n-1}p$, with mean = $1/p$
Negative Binomial - number of failures before r successes
Hypergeometric - number of successes x in a size N population with n draws, without replacement
Poisson - number of successes x in a fixed time interval, where success occurs at an average rate $\lambda \rightarrow \frac{\lambda^x e^{-\lambda}}{x!}$

Continuous Distributions

Normal/Gaussian $N(\mu, \sigma)$, Standard Normal $Z \sim N(0, 1)$
Central Limit Theorem - sample mean of i.i.d. data approaches normal distribution
Exponential - time between independent events occurring at an average rate $\lambda \rightarrow \lambda e^{-\lambda x}$, with mean = $\frac{1}{\lambda}$
Gamma - time until n independent events occurring at an average rate λ

Hypothesis Testing

Significance Level α - probability of Type 1 error
***p*-value** - probability of getting results at least as extreme as the current test. If p-value $< \alpha$, or if test statistic $>$ critical value, then reject the null.
Type I Error (False Positive) - null true, but reject
Type II Error (False Negative) - null false, but fail to reject
Power - probability of avoiding a Type II Error, and rejecting the null when it is indeed false
Z-Test - tests whether population means/proportions are different. Assumes test statistic is normally distributed and is used when n is large and variances are known. If not, then use a t-test. Paired tests compare the mean at different points in time, and two-sample tests compare means for two groups.
ANOVA - analysis of variance, used to compare 3+ samples with a single test
Chi-Square Test - checks relationship between categorical variables (age vs. income). Or, can check goodness-of-fit between observed data and expected population distribution.

Concepts

Learning

- Supervised - labeled data
- Unsupervised - unlabeled data
- Reinforcement - actions, states, and rewards

Cross Validation - estimate test error with a portion of training data to validate accuracy and model parameters

- k-fold - divide data into k groups, and use one to validate
- leave-p-out - use p samples to validate and the rest to train

Model Evaluation

Prediction Error = Bias^2 + Variance + Irreducible Noise
Bias - wrong assumptions when training $\rightarrow$ can't capture underlying patterns $\rightarrow$ underfit
Variance - sensitive to fluctuations when training$\rightarrow$ can't generalize on unseen data $\rightarrow$ overfit

Regression

Mean Squared Error (MSE) = $\frac{1}{n}\sum(y_i - \hat{y})^2$
Mean Absolute Error (MAE) =$\frac{1}{n}\sum|(y_i - \hat{y})|$
Sum of Squared Error (SSE) = $\sum(y_i - \hat{y})^2$
Total Sum of Squares (SST) = $\sum(y_i - \bar{y})^2$
$\boldsymbol{R^2} = 1 - \frac{SSE}{SST}$

Classification

	Predict Yes	Predict No
Actual Yes	True Positives (TP)	False Negatives (FN)
Actual No	False Positives (FP)	True Negatives (TN)

- Precision = $\frac{TP}{TP+FP}$, percent correct when predict positive
- Recall, Sensitivity = $\frac{TP}{TP+FN}$, percent of actual positives identified correctly (True Positive Rate)
- Specificity = $\frac{TN}{TN+FP}$, percent of actual negatives identified correctly, also 1 - FPR (True Negative Rate)
- $F_1 = 2\frac{precision \cdot recall}{precision+recall}$, useful when classes are imbalanced

ROC Curve - plots TPR vs. FPR for every threshold α. AUC measures how likely the model differentiates positives and negatives. Perfect AUC = 1, Baseline = 0.5
Precision-Recall Curve - focuses on the correct prediction of class 1, useful when data or FP/FN costs are imbalanced

Linear Regression

Models linear relationships between a continuous response and explanatory variables
Ordinary Least Squares - find $\hat{\beta}$ for $\hat{y} = \hat{\beta_0} + \hat{\beta}X + \epsilon$ by solving $\hat{\beta} = (X^T X)^{-1} X^T Y$ which minimizes the SSE
Assumptions

- Linear relationship and independent observations
- Homoscedasticity - error terms have constant variance
- Errors are uncorrelated and normally distributed
- Low multicollinearity

Regularization
Add a penalty for large coefficients to reduce overfitting
Subset (L0): $\lambda||\hat{\beta}||_0 = \lambda(number\ of\ non{-}zero\ variables)$

- Computationally slow, need to fit 2^k models
- Alternatives: forward and backward stepwise selection

LASSO (L1): $\lambda||\hat{\beta}||_1 = \lambda\sum|\hat{\beta}|$

- Coefficients shrunk to zero

Ridge (L2): $\lambda||\hat{\beta}||_2 = \lambda\sum(\hat{\beta})^2$

- Reduces effects of multicollinearity

Combining LASSO and Ridge gives Elastic Net. In all cases, as λ grows, bias increases and variance decreases.
Regularization can also be applied to many other algorithms.

Logistic Regression

Predicts probability that Y belongs to a binary class (1 or 0). Fits a logistic (sigmoid) function to the data that maximizes the likelihood that the observations follow the curve. Regularization can be added in the exponent.

$$P(Y=1) = \frac{1}{1+e^{-(\beta_0+\beta x)}}$$

Odds - output probability can be transformed using $Odds(Y=1) = \frac{P(Y=1)}{1-P(Y=1)}$, where $P(\frac{1}{3}) = 1{:}2$ odds
Assumptions

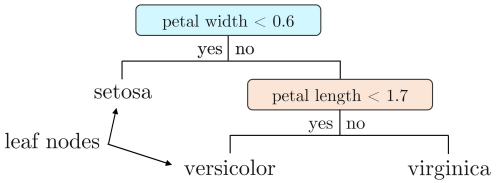
- Linear relationship between X and log-odds of Y
- Independent observations
- Low multicollinearity

Decision Trees

Classification and Regression Tree

CART for regression minimizes SSE by splitting data into sub-regions and predicting the average value at leaf nodes. Trees are prone to high variance, so tune through CV.
Hyperparameters

- Complexity parameter, to only keep splits that improve SSE by at least cp (most influential, small $cp \rightarrow$ deep tree)
- Minimum number of samples at a leaf node
- Minimum number of samples to consider a split



CART for classification minimizes the sum of region impurity, where $\hat{p_i}$ is the probability of a sample being in category i. Possible measures, each with a max impurity of 0.5.

- Gini impurity = $\sum_i(\hat{p_i})^2$
- Entropy = $\sum_i -(\hat{p_i})log_2(\hat{p_i})$

At each leaf node, CART predicts the most frequent category, assuming false negative and false positive costs are the same.

Random Forest

Trains an ensemble of trees that vote for the final prediction
Bootstrapping - sampling with replacement (will contain duplicates), until the sample is as large as the training set
Bagging - training independent models on different subsets of the data, which reduces variance. Each tree is trained on ~63% of the data, so the out-of-bag 37% can estimate prediction error without resorting to CV.
Additional Hyperparameters (no cp):

- Number of trees to build
- Number of variables considered at each split

Deep trees increase accuracy, but at a high computational cost. Model bias is always equal to one of its individual trees.
Variable Importance - RF ranks variables by their ability to minimize error when split upon, averaged across all trees

Aaron Wang

Naive Bayes

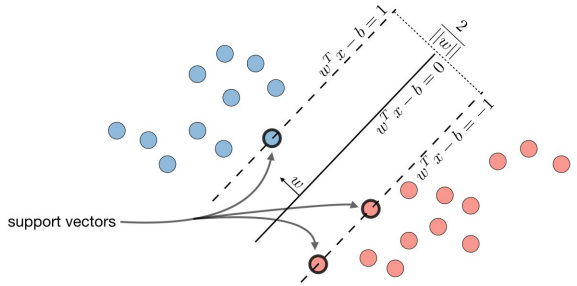
Classifies data using the label with the highest conditional probability, given data a and classes c. Naive because it assumes variables are independent.

Bayes' Theorem $P(c_i|a) = \frac{P(a|c_i)P(c_i)}{P(a)}$

Gaussian Naive Bayes - calculates conditional probability for continuous data by assuming a normal distribution

Support Vector Machines

Separates data between two classes by maximizing the margin between the hyperplane and the nearest data points of any class. Relies on the following:

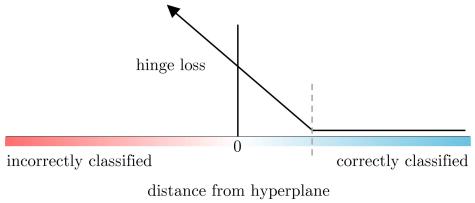


Support Vector Classifiers - account for outliers by allowing misclassifications on the support vectors (points in or on the margin)

Kernel Functions - solve nonlinear problems by computing the similarity between points a, b and mapping the data to a higher dimension. Common functions:

- Polynomial $(ab + r)^d$
- Radial $e^{-\gamma(a-b)^2}$

Hinge Loss - $\max(0, 1 - y_i(w^T x_i - b))$, where w is the margin width, b is the offset bias, and classes are labeled ± 1. Note, even a correct prediction inside the margin gives loss > 0.



k-Nearest Neighbors

Non-parametric method that calculates $\hat{y}$ using the average value or most common class of its k-nearest points. For high-dimensional data, information is lost through equidistant vectors, so dimension reduction is often applied prior to k-NN.

Minkowski Distance $= (\sum |a_i - b_i|^p)^{1/p}$

- p = 1 gives Manhattan distance $\sum |a_i - b_i|$
- p = 2 gives Euclidean distance $\sqrt{\sum (a_i - b_i)^2}$

Hamming Distance - count of the differences between two vectors, often used to compare categorical variables

Clustering

Unsupervised, non-parametric methods that groups similar data points together based on distance

k-Means

Randomly place k centroids across normalized data, and assig observations to the nearest centroid. Recalculate centroids as the mean of assignments and repeat until convergence. Using the median or medoid (actual data point) may be more robust to noise and outliers. k-modes is used for categorical data.

***k*-means++** - improves selection of initial clusters

1. Pick the first center randomly
2. Compute distance between points and the nearest center
3. Choose new center using a weighted probability distribution proportional to distance
4. Repeat until k centers are chosen

Evaluating the number of clusters and performance:

Silhouette Value - measures how similar a data point is to its own cluster compared to other clusters, and ranges from 1 (best) to -1 (worst).

Davies-Bouldin Index - ratio of within cluster scatter to between cluster separation, where lower values are

Hierarchical Clustering

Clusters data into groups using a predominant hierarchy

Agglomerative Approach

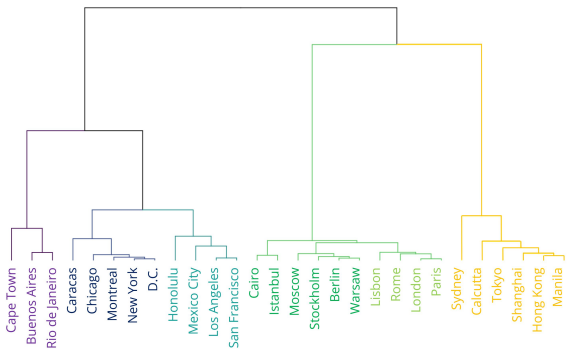
1. Each observation starts in its own cluster
2. Iteratively combine the most similar cluster pairs
3. Continue until all points are in the same cluster

Divisive Approach - all points start in one cluster and splits are performed recursively down the hierarchy

Linkage Metrics - measure dissimilarity between clusters and combines them using the minimum linkage value over all pairwise points in different clusters by comparing:

- Single - the distance between the closest pair of points
- Complete - the distance between the farthest pair of points
- Ward's - the increase in within-cluster SSE if two clusters were to be combined

Dendrogram - plots the full hierarchy of clusters, where the height of a node indicates the dissimilarity between its children



Dimension Reduction

Principal Component Analysis

Projects data onto orthogonal vectors that maximize variance. Remember, given an $n \times n$ matrix A, a nonzero vector $\vec{x}$, and a scaler λ, if $A\vec{x} = \lambda\vec{x}$ then $\vec{x}$ and λ are an eigenvector and eigenvalue of A. In PCA, the eigenvectors are uncorrelated and represent principal components.

1. Start with the covariance matrix of standardized data
2. Calculate eigenvalues and eigenvectors using SVD or eigendecomposition
3. Rank the principal components by their proportion of variance explained $= \frac{\lambda_i}{\sum \lambda}$

For a p-dimensional data, there will be p principal components

Sparse PCA - constrains the number of non-zero values in each component, reducing susceptibility to noise and improving interpretability

Linear Discriminant Analysis

Maximizes separation between classes and minimizes variance within classes for a labeled dataset

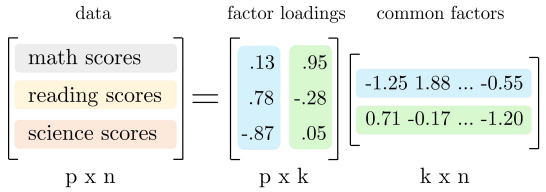
1. Compute the mean and variance of each independent variable for every class
2. Calculate the within-class (σ_w^2) and between-class (σ_b^2) variance
3. Find the matrix $W = (\sigma_w^2)^{-1}(\sigma_b^2)$ that maximizes Fisher's signal-to-noise ratio
4. Rank the discriminant components by their signal-to-noise ratio λ

Assumptions

- Independent variables are normally distributed
- Homoscedasticity - constant variance of error
- Low multicollinearity

Factor Analysis

Describes data using a linear combination of k latent factors. Given a normalized matrix X, it follows the form $X = Lf + \epsilon$, with factor loadings L and hidden factors f.



Assumptions

- $E(X) = E(f) = E(\epsilon) = 0$
- $\text{Cov}(f) = I \rightarrow$ uncorrelated factors
- $\text{Cov}(f, \epsilon) = 0$

Since $\text{Cov}(X) = \text{Cov}(Lf) + \text{Cov}(\epsilon)$, then $\text{Cov}(Lf) = LL'$

Scree Plot - graphs the eigenvalues of factors (or principal components) and is used to determine the number of factors to retain. The 'elbow' where values level off is often used as the cutoff.

Aaron Wang

Natural Language Processing

Transforms human language into machine-usable code
Processing Techniques

- Tokenization - splitting text into individual words (tokens)
- Lemmatization - reduces words to its base form based on dictionary definition (*am, are, is* $\rightarrow$ *be*)
- Stemming - reduces words to its base form without context (*ended* $\rightarrow$ *end*)
- Stop words - remove common and irrelevant words (*the, is*)

Markov Chain - stochastic and memoryless process that predicts future events based only on the current state
***n*-gram** - predicts the next term in a sequence of n terms based on Markov chains
Bag-of-words - represents text using word frequencies, without context or order
tf-idf - measures word importance for a document in a collection (corpus), by multiplying the term frequency (occurrences of a term in a document) with the inverse document frequency (penalizes common terms across a corpus)
Cosine Similarity - measures similarity between vectors, calculated as $\cos(\theta) = \frac{A \cdot B}{||A|| ||B||}$, which ranges from o to 1

Word Embedding

Maps words and phrases to numerical vectors
word2vec - trains iteratively over local word context windows, places similar words close together, and embeds sub-relationships directly into vectors, such that $king - man + woman \approx queen$
Relies on one of the following:

- Continuous bag-of-words (CBOW) - predicts the word given its context
- skip-gram - predicts the context given a word

GloVe - combines both global and local word co-occurence data to learn word similarity
BERT - accounts for word order and trains on subwords, and unlike word2vec and GloVe, BERT outputs different vectors for different uses of words (*cell* phone vs. blood *cell*)

Sentiment Analysis

Extracts the attitudes and emotions from text
Polarity - measures positive, negative, or neutral opinions

- Valence shifters - capture amplifiers or negators such as '*really* fun' or '*hardly* fun'

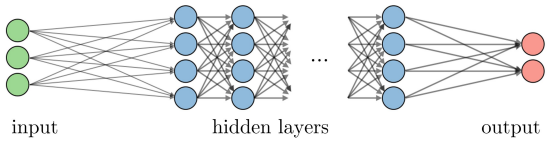
Sentiment - measures emotional states such as happy or sad
Subject-Object Identification - classifies sentences as either subjective or objective

Topic Modelling

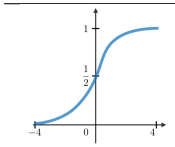
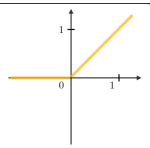
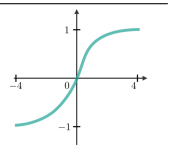
Captures the underlying themes that appear in documents
Latent Dirichlet Allocation (LDA) - generates k topics by first assigning each word to a random topic, then iteratively updating assignments based on parameters α, the mix of topics per document, and β, the distribution of words per topic
Latent Semantic Analysis (LSA) - identifies patterns using tf-idf scores and reduces data to k dimensions through SVD

Neural Network

Feeds inputs through different hidden layers and relies on weights and nonlinear functions to reach an output



Perceptron - the foundation of a neural network that multiplies inputs by weights, adds bias, and feeds the result z to an activation function
Activation Function - defines a node's output

Sigmoid	ReLU	Tanh
$\frac{1}{1+e^{-z}}$	$\max(0, z)$	$\frac{e^z - e^{-z}}{e^z + e^{-z}}$
		

Since a system of linear activation functions can be simplified to a single perceptron, nonlinear functions are commonly used for more accurate tuning and meaningful gradients

Loss Function - measures prediction error using functions such as MSE for regression and binary cross-entropy for probability-based classification

Gradient Descent - minimizes the average loss by moving iteratively in the direction of steepest descent, controlled by the learning rate γ (step size). Note, γ can be updated adaptively for better performance. For neural networks, finding the best set of weights involves:

1. Initialize weights W randomly with near-zero values
2. Loop until convergence:
   - Calculate the average network loss $J(W)$
   - **Backpropagation** - iterate backwards from the last layer, computing the gradient $\frac{\partial J(W)}{\partial W}$ and updating the weight $W \leftarrow W - \gamma \frac{\partial J(W)}{\partial W}$
3. Return the minimum loss weight matrix W

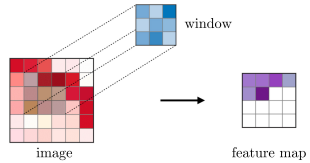
To prevent overfitting, regularization can be applied by:

- Stopping training when validation performance drops
- Dropout - randomly drop some nodes during training to prevent over-reliance on a single node
- Embedding weight penalties into the objective function

Stochastic Gradient Descent - only uses a single point to compute gradients, leading to smoother convergence and faster compute speeds. Alternatively, mini-batch gradient descent trains on small subsets of the data, striking a balance between the approaches.

Convolutional Neural Network

Analyzes structural or visual data by extracting local features
Convolutional Layers - iterate over windows of the image, applying weights, bias, and an activation function to create feature maps. Different weights lead to different features maps.



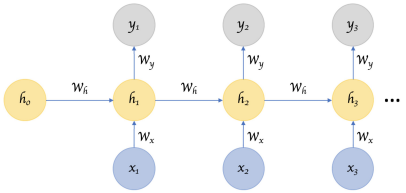
Pooling - downsamples convolution layers to reduce dimensionality and maintain spatial invariance, allowing detection of features even if they have shifted slightly. Common techniques return the max or average value in the pooling window.

The general CNN architecture is as follows:

1. Perform a series of convolution, ReLU, and pooling operations, extracting important features from the data
2. Feed output into a fully-connected layer for classification, object detection, or other structural analyses

Recurrent Neural Network

Predicts sequential data using a temporally connected system that captures both new inputs and previous outputs using hidden states



RNNs can model various input-output scenarios, such as many-to-one, one-to-many, and many-to-many. Relies on parameter (weight) sharing for efficiency. To avoid redundant calculations during backpropagation, downstream gradients are found by chaining previous gradients. However, repeatedly multiplying values greater than or less than 1 leads to:

- Exploding gradients - model instability and overflows
- Vanishing gradients - loss of learning ability

This can be solved using:

- Gradient clipping - cap the maximum value of gradients
- ReLU - its derivative prevents gradient shrinkage for $x > 0$
- Gated cells - regulate the flow of information

Long Short-Term Memory - learns long-term dependencies using gated cells and maintains a separate cell state from what is outputted. Gates in LSTM perform the following:

1. Forget and filter out irrelevant info from previous layers
2. Store relevant info from current input
3. Update the current cell state
4. Output the hidden state, a filtered version of the cell state

LSTMs can be stacked to improve performance.

Aaron Wang

Boosting

Ensemble method that learns by sequentially fitting many simple models. As opposed to bagging, boosting trains on all the data and combines weak models using the learning rate α. Boosting can be applied to many machine learning problems.

AdaBoost - uses sample weighting and decision 'stumps' (one-level decision trees) to classify samples

1. Build decision stumps for every feature, choosing the one with the best classification accuracy
2. Assign more weight to misclassified samples and reward trees that differentiate them, where $\alpha = \frac{1}{2} ln \frac{1-TotalError}{TotalError}$
3. Continue training and weighting decision stumps until convergence

Gradient Boost - trains sequential models by minimizing a given loss function using gradient descent at each step

1. Start by predicting the average value of the response
2. Build a tree on the errors, constrained by depth or the number of leaf nodes
3. Scale decision trees by a constant learning rate α
4. Continue training and weighting decision trees until convergence

XGBoost - fast gradient boosting method that utilizes regularization and parallelization

Recommender Systems

Suggests relevant items to users by predicting ratings and preferences, and is divided into two main types:

- Content Filtering - recommends similar items
- Collaborative Filtering - recommends what similar users like

The latter is more common, and includes methods such as:
Memory-based Approaches - finds neighborhoods by using rating data to compute user and item similarity, measured using correlation or cosine similarity

- User-User - similar users also liked...
  - Leads to more diverse recommendations, as opposed to just recommending popular items
  - Suffers from sparsity, as the number of users who rate items is often low
- Item-Item - similar users who liked this item also liked...
  - Efficient when there are more users than items, since the item neighborhoods update less frequently than users
  - Similarity between items is often more reliable than similarity between users

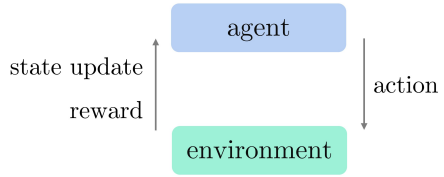
Model-based Approaches - predict ratings of unrated items, through methods such as Bayesian networks, SVD, and clustering. Handles sparse data better than memory-based approaches.

- Matrix Factorization - decomposes the user-item rating matrix into two lower-dimensional matrices representing the users and items, each with k latent factors

Recommender systems can also be combined through ensemble methods to improve performance.

Reinforcement Learning

Maximizes future rewards by learning through state-action pairs. That is, an *agent* performs *actions* in an *environment*, which updates the *state* and provides a *reward*.



Multi-armed Bandit Problem - a gambler plays slot machines with unknown probability distributions and must decide the best strategy to maximize reward. This exemplifies the exploration-exploitation tradeoff, as the best long-term strategy may involve short-term sacrifices.

RL is divided into two types, with the former being more common:

- Model-free - learn through trial and error in the environment
- Model-based - access to the underlying (approximate) state-reward distribution

Q-Value $Q(s, a)$ - captures the expected discounted total future reward given a state and action
Policy - chooses the best actions for an agent at various states $\pi(s) = \arg\max_a Q(s, a)$

Deep RL algorithms can further be divided into two main types, depending on their learning objective
Value Learning - aims to approximate $Q(s, a)$ for all actions the agent can take, but is restricted to discrete action spaces. Can use the ϵ-greedy method, where ϵ measures the probability of exploration. If chosen, the next action is selected uniformly at random.

- Q-Learning - simple value iteration model that maximizes the Q-value using a table on states and actions
- Deep Q Network - finds the best action to take by minimizing the Q-loss, the squared error between the target Q-value and the prediction

Policy Gradient Learning - directly optimize the the policy $\pi(s)$ through a probability distribution of actions, without the need for a value function, allowing for continuous action spaces.

Actor-Critic Model - hybrid algorithm that relies on two neural networks, an actor $\pi(s, a, \theta)$ which controls agent behavior and a critic $Q(s, a, w)$ that measures how good an action is. Both run in parallel to find the optimal weights θ, w to maximize expected reward. At each step:

1. Pass the current state into the actor and critic
2. The critic evaluates the action's Q-value, and the actor updates its weight θ
3. The actor takes the next action leading to a new state, and the critic updates its weight w

Anomaly Detection

Identifies unusual patterns that differ from the majority of the data, and can be applied in supervised, unsupervised, and semi-supervised scenarios. Assumes that anomalies are:

- Rare - the minority class that occurs rarely in the data
- Different - have feature values that are very different from normal observations

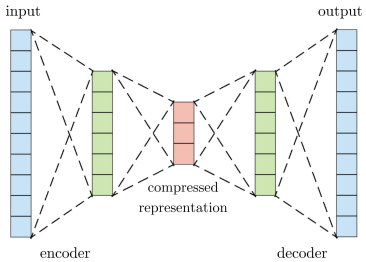
Anomaly detection techniques spans a wide range, including methods based on:
Statistics - relies on various statistical methods to identify outliers, such as Z-tests, boxplots, interquartile ranges, and variance comparisons
Density - useful when data is grouped around dense neighborhoods, measured by distance. Methods include k-nearest neighbors, local outlier factor, and isolation forest.

- Isolation Forest - tree-based model that labels outliers based on an anomaly score
  1. Select a random feature and split value, dividing the dataset in two
  2. Continue splitting randomly until every point is isolated
  3. Calculate the anomaly score for each observation, based on how many iterations it took to isolate that point.
  4. If the anomaly score is greater than a threshold, mark it as an outlier

  Intuitively, outliers are easier to isolate and should have shorter path lengths in the tree

Clusters - data points outside of clusters could potentially be marked as anomalies

Autoencoders - unsupervised neural networks that compress data and reconstructs it. The network has two parts: an encoder that embeds data to a lower dimension, and a decoder that produces a reconstruction. Autoencoders do not reconstruct the data perfectly, but rather focus on capturing important features in the data.



Upon decoding, the model will accurately reconstruct normal patterns but struggle with anomalous data. The reconstruction error is used as an anomaly score to detect outliers.

Autoencoders are applied to many problems, including image processing, dimension reduction, and information retrieval.