Bias-Variance Tradeoff

- **3 Errors**: Bias² + Var + Irr² **Bias**: error due to oversimplified assumptions **Variance**: change in estimate as train data changes **Irreducible**: noise
- **Underfitting**: model has high bias & low variance → high training & high test error, typical of linear models
- **Overfitting**: model has low bias & high variance → low training error & high test error, typical of nonlinear models
- **Non-Parametric Model**: no fixed num of model params, no assumptions about underlying data distribution (e.g. tree, knn, randfor)

Gradient Descent

- Optimization algo, minimize cost func by iteratively tweaking model params, feats must be scaled, LR η must be tuned, $\vec{\theta}_1 = \vec{\theta}_0 - \eta \cdot \nabla J(\vec{\theta}_0)$
- **SGD**: 1 rand sample each step, faster than GD, use if m huge, bounces out of local min, simulated annealing prevents bouncing out of global min

K-Fold Cross Validation (CV)

- Split data into train/test, perform k-fold CV on train, each train example is trained on k-1 times and evaluated on once
- **(+)** Has lower variance than train-val-test evaluation, obtains avg performance and stdev of performances, produces more generalizable model

Categorical Features, Embeddings, Missing Data

- One-Hot (<10 cats), Ordinal (if ordered), Hash (hash into n buckets), Binary (convert to n digit binary num), Target (cat's avg target val)
- **Embedding**: Trainable dense vector for each cat, high-dim sparse cat feat → low dim dense representation, train using NN, e.g. zip codes
- **Missing Data**: del feat, del samples, impute by mean, impute by KNN, use ML to pred missing vals, use trees method (can handle missing vals)

Feature Scaling & Distance Measures

- **When to Scale**: GD (smoother), distance algo (KNN, KMeans, SVM), PCA (maxes variance), linreg (interpretable coeffs), l1 or l2 reg
- **Min-Max Normalizing**: scale to [0,1], affected by outliers, $\vec{x}' = \frac{\vec{x}-\min(\vec{x})}{\max(\vec{x})-\min(\vec{x})}$ **Standardizing**: scale to μ=0, σ=1, robust to outliers, $\vec{x}' = \frac{\vec{x}-\bar{x}}{\sigma}$
- **Minkowski**: $\left(\sum_{i=1}^{n} |x_i^{(1)} - x_i^{(2)}|^p\right)^{1/p}$ p=1 manhatten, p=2 euclidean, higher p → more focus on outliers **Cosine**: cos(θ)=$\frac{\vec{a}\cdot\vec{b}}{\|\vec{a}\|\|\vec{b}\|}$ angle sim of 2 vecs

Classification

- **Precision** = $\frac{\mathrm{TP}}{\mathrm{TP+FP}}$ e.g. Youtube vids: risk FNs to min FPs **Recall** = $\frac{\mathrm{TP}}{\mathrm{TP+FN}}$ e.g. Covid tests: risk FPs to min FNs **Tradeoff**: min FPs or FNs
- **F1** = $2 \times \frac{\mathrm{prec}\times\mathrm{recall}}{\mathrm{prec}+\mathrm{recall}}$ harmonic mean **ROC Curve** = plot FPR (1-TNR) vs. TPR for all thresholds pick top left pt, tradeoff: higher TPR → higher FPR
- **Choose Decision Threshold**: 1. threshold that maxes F1 2. pt on recall vs. precision for all thresholds 3. top left ROC curve pt
- **Class Imbalance**: 1. recall/precision/f1/AUC 2. apply class weight to loss func 3. under & oversample (dupes or SMOTE) 4. tree-based algo
- **Multiclass**: softmax, knn, tree, nn, if binary classifier: 1vAll (train one classifier per class), 1v1 (train one classifier per pair of classes)

Ensembling

- **Ensemble**: Train high bias weak learners on subsets of train data &/or subsets of feats for low bias low variance model
- **Classification**: Hard Vote (plurality/majority/weighted voting) Soft Vote (mean/sum/weighted probas) **Regression**: mean or median (if non-normal)
- **Bootstrapping**: repeat sampling with replacement **Bagging**: aggregate models trained on separate bootstrapped samples → out-of-bag samples
- **Boosting**: train models sequentially on their residuals **Stacking**: parent model that trains on outputs of child models

Dimensionality Reduction

- **Curse of Dims**: dims grow → feat space vol grows → data sparsity → exponentially more data to stop overfitting **Eval**: 1. Reconstruction Error
- **(+)** faster training, improve accuracy (less noise/sparsity/collinearity), viz in 2D/3D (t-SNE) **(-)** feats harder to interpret, worse accuracy (lose info)
- **Ways**: 1. Feat Select (lasso, imprtnce scores) 2. Feat Extract a. Project: collapse unimport dims (PCA/LDA), b. Manifold: model data manifold (LLE)

Anomaly Detection

- **Isolation Forest**: unsupervised randfor, split on rand feat & rand threshold, split until 1 sample leafs, outlier leafs have smaller depths than inliers
- **One-Class SVM**: unsupervised, use RBF kernel for nonlinear boundary to circle inliers, outside boundary is anomaly, ν=% of outliers we expect

Clustering

- **Types**: centroid, density **Uses**: customer segmentation, semi-supervised label propagation, anomaly detection (anything not in a cluster)
- **Eval**: 1. Inertia (avg sqrd dist of samp to its centroid, assumes convex clusters) 2. Dunn Idx (lowest intercluster dist / highest intracluster dist)
  - 3. Silhouettes (silhouette coeff for a sample $\frac{b-a}{\max(a,b)}$ a = avg intracluster dist, b = avg dist to nearest clust, [-1, 1] -1: wrong clust, 1: own clust)
- **DBSCAN**: density, cnt # of neighs each samp has within ϵ-dist, high cnts = core instance & their neighborhood = a clust, (+) non-convex clusters

Time Series Forecasting

- **Autocorrelation**: corr at curr t with prev time steps (ACF plot) **Partial Autocorrelation**: corr at lag k after removing any correlations at shorter lags
- **Stationary**: values not a func of time (constant mean/var/autocorr)   **Differencing**: subtract consecutive t-steps to remove temporal dependence
- **MAPE**: mean absolute percent error = $\frac{100\%}{m}\sum_{i=1}^{m}|\frac{y_i-\hat{y}_i}{y_i}|$, unitless, interpretable, loss func & metric, better to weight e.g. by recency
- **SARIMA**: Seasonal, AutoRegress (p=num of lags to use), Integrated (d=num of times to difference), Moving Avg (q=size of moving avg window)

Natural Language Processing

- **Preprocess**: tokenize (ngrams), rm stop words (prepositions), stemming (rm prefix/suffix), lemmatize (standardize words with similar meaning)
- **Text Vectorization**: Bag of Words (rows=docs, cols=ngrams, vals=cnts), Pre-Trained Embeddings (encode concept similarity, e.g. Words2Vec)
    - TF*IDF (emphasizes word's relevance to a doc, term freq = $\frac{\text{term cnt in doc}}{\text{tot terms in doc}}$, inv doc freq = $\log(\frac{\text{tot docs}}{\text{tot docs containing term}})$), POS Tags, NER
- **Topic Modeling**: LDA, assign topic probas to docs, calc p(word w in topic t) = p(topic t | doc d) * p(word w | topic t) for each word