Linear & Polynomial Regression

- **LinReg**: model linear data: $\hat{y} = \vec{\theta}^T \vec{x}$ where x_0=1, linear in params $\vec{\theta}$, OLS normal equation: $\hat{\vec{\theta}} = (\boldsymbol{X}^T \boldsymbol{X})^{-1} \boldsymbol{X}^T \vec{y}$ (slower than GD when m large)
- **(+)** models linear data well, white-box, feat importance **(-)** strict assumptions, outliers can skew results
- **Assumptions**: 1. Linearity - feats indep linear wrt targ (pair plot, ypred vs resids) 2. No Multicollinearity - hurts coeff interpretability (correlations, VIF)
  - 3. Normal Residuals (histogram, QQ Plot, stat test) 4. Homoskedastic - residual variance equal wrt to ypred (ypred vs residuals, stat test)
  - 5. No Auto-Correlation - resids indep of each other, stationarize data by differencing (ACF plot, Durbin-Watson Test)
- **Coeff**: mean Δ in depen var for each unit of Δ in indep var, can interpret as feat importance, p-val = prob coeff should be 0
- **Cost Func**: MSE penalizes outliers more → fits them better than MAE, $\text{MSE} = \frac{1}{m} \sum_{i=1}^{m} (\hat{y}^{(i)} - y^{(i)})^2$, $\nabla \text{MSE} = \frac{2}{m} \boldsymbol{X}^T (\boldsymbol{X} \vec{\theta} - \vec{y})$
- $\boldsymbol{R^2}$: % of variance explained, compared to model that predicts mean, $R^2 = 1 - \frac{\text{SSE}}{\text{SST}}$, R^2_{adj} penalizes unnecessary feats to avoid overfitting
- **Regularization**: limit coeffs to limit overfitting, feats must be scaled, apply by adding term to cost func, α hyperparam controls strength
  - L1 Lasso: diamond loss contours → feat selection $\alpha \sum_{i=1}^{n} |\theta_i|$, L2 Ridge, circle loss contours: $\alpha \frac{1}{2} \sum_{i=1}^{n} \theta_i^2$, Elastic Net: weighted mix
- **Polynomial Regression**: model nonlinear data, params still linear, add interaction & higher power terms
- **Bayesian LinReg**: use prior to get posterior distrib for model params: $P(\boldsymbol{\theta}|\boldsymbol{y}, \boldsymbol{X}) = \frac{P(\boldsymbol{y}|\boldsymbol{\theta}, \boldsymbol{X}) \cdot P(\boldsymbol{\theta}|\boldsymbol{X})}{P(\boldsymbol{y}|\boldsymbol{X})}$, preds made from distrib: $\hat{\boldsymbol{y}} \sim \mathcal{N}(\boldsymbol{\theta}^T \boldsymbol{x}, \sigma^2 \boldsymbol{I})$

Logistic & Softmax Regression

- **LogReg**: linear/hyperplane decision boundary, $\hat{p} = \sigma(\vec{\theta}^T \vec{x}) = \frac{1}{1 + e^{-\vec{\theta}^T \vec{x}}}$ sigmoid func takes linreg logit (-∞,∞) and outputs prob [0,1]
- **(+)** models linearly separable data well, white-box, calcs class proba, feat importance (1 unit Δ in x_i → depend var Δ by e^{θ_i}) **(-)** strict assumptions
- **Assumptions**: 1. No Multicollinearity - hurts coeff interpretability (correlations, VIF) 2. Linearity: feats indep linear wrt logit of preds i.e. linreg output
  - 3. No Influential Outliers (Cook's Dist: change in model when outlier removed) 4. Requires larger m when more feats
- **Odds** [0,∞) = $\frac{\pi}{1-\pi}$ **Log-Odds** (-∞,∞) = $\log\left(\frac{\pi}{1-\pi}\right) = \text{logit}(\pi) = \eta$ (logit) **Sigmoid** [0,1] = $\sigma(\eta) = \text{logit}^{-1}(\eta) = \frac{1}{1+e^{-\eta}} = \pi$ (probability)
- **Cost Func**: Log Loss $J(\vec{\theta}) = -\frac{1}{m} \sum_{i=1}^{m} \left(y^{(i)} \log(\hat{p}^{(i)}) + (1 - y^{(i)}) \log(1 - \hat{p}^{(i)}) \right)$ **Pred**: $\hat{y}^{(i)}$=1 if $\hat{p}^{(i)}$≥0.5, else $\hat{y}^{(i)}$=0
- **Regularization**: L1, L2, & Elastic Net **Polynomial LogReg**: nonlinear boundary **SoftmaxReg**: multiclassifier logreg, piecewise linear boundaries
- 1. Calc logit for each class k: $s_k(\vec{x}) = \vec{x}^T \vec{\theta}^{(k)}$, 2. Softmax calcs prob of each logit: $\hat{p}_k = \sigma(\vec{s}(\vec{x}))_k = \frac{e^{s_k(\vec{x})}}{\sum_{j=1}^{K} e^{s_j(\vec{x})}}$, 3. Pred class /w highest prob
- **Softmax Cost Func**: Cross Entropy, how well class probs match target probs, $J(\boldsymbol{\Theta}) = -\frac{1}{m} \sum_{i=1}^{m} \sum_{k=1}^{K} y_k^{(i)} \log(\hat{p}_k^{(i)})$

Support Vector Machines (SVM)

- **SVM**: lin/nonlin regressor/binary classifier/outlier detection, finds n dim decision boundary hyperplane maxes margin between 2 classes
- **(+)** high-dim data, inference memory efficient (only needs SVs) **(-)** don't scale with data, slow to train, no class probas
- **Decision Func**: n+1 dim, $h(\vec{x}) = \vec{w}^T \vec{x} + b$, model params are $\vec{w}$ & b **Decision Boundary**: n dim, where h=0 **Margins**: n dim, where h=-1, 1
- **Support Vectors**: closest samples from each class, determine margin boundaries **Classify**: plug pts into h, $\hat{y}$=1 if h≥0 else $\hat{y}$=-1
- **Soft-Margin**: wide margin + limit margin violations **C Hyperparam**: controls tradeoff, ↓C = wider margin = more violations = more generalizeable
- **Cost Func**: Hinge Loss = $\max\left(0,\ 1 - \hat{y}(\vec{w}^T \vec{x} + b)\right)$, is 0 if $\hat{y}(\vec{w}^T \vec{x} + b) \geq 1$ (correct class + outside margin), else linear penalty $1 - \hat{y} \cdot h(\vec{x})$
- **Kernel Trick**: map nonlinearly separable data into higher dim to find hyperplane boundary, kernel func computes dot prod of instances mapped in
  - higher dim without transforming them to higher dim, allows to use SVM on transformed feats without transforming them (e.g. polynomial, RBF)

Decision Tree

- **DecTree**: nonparametric, lin/nonlin regressor/multiclassifier, stochastic if max_feats set, use stop criteria to stop splitting
- **(+)** white-box, handle outliers/noise/missing, no scaling, inference is log(m), feat importance **(-)** 90° boundaries, high variance, overfit
- **CART**: each split is greedy search to find feat k & threshold t_k that results in lowest cost: $J(k, t_k) = \frac{m_{\text{left}}}{m_{\text{node}}} \text{Impurity}_{\text{left}} + \frac{m_{\text{right}}}{m_{\text{node}}} \text{Impurity}_{\text{right}}$
- **Impurity**: 0 if node's instances all 1 class, >0 if multiple classes **Information Gain**: $\text{Entropy}_{\text{parent}} - \left(\frac{m_{\text{left}}}{m_{\text{parent}}} \text{Entropy}_{\text{left}} + \frac{m_{\text{right}}}{m_{\text{parent}}} \text{Entropy}_{\text{right}}\right)$
- **Gini**: $G_i = 1 - \sum_{k=1}^{K} p_{i,k}^2$ **Entropy**: $E_i = -\sum_{k=1}^{K} p_{i,k} \log_2(p_{i,k})$ $p_{i,k}$ = ratio of class k instances among all train instances in ith node
- **Classify**: leaf node classifies whatever the majority represented label is **Class Proba** = ratio of train instances of class k in instance's leaf node
- **Regression**: each split mins MSE, leaf node preds avg targ value of its train instances
- **Feat Importance**: how much tree nodes that use that feat reduce impurity on avg, each node weighted by num of train samples associated with it

- **Regularization**: 1. pre-pruning (max_depth, max_feats, max_leafs, min_samples_per_leaf), 2. post-pruning (delete unnecessary nodes)

Random Forest & Gradient Boosting

- **RandFor**: nonparametric, stochastic, use bagging to train trees, each tree trained on random subset of feats & bootstrapped sample of data
- **(+)** low variance, handle outliers/noise/missing, no scaling, feat importance **(-)** slow training (but parallelizeable unlike boosting), black box
- **Gradient Boosting**: train models sequentially, each model fit on resids of prev model, if tree → shallow <10 splits (shallower than randfor trees)
- **Hyperparams**: n_trees (too few → underfit, too many → overfit, use early stopping), subsample (% of train data to train each tree → stochastic)
- **Regression**: pred by summing all tree preds in the ensemble

K-Nearest Neighbors (KNN)

- **KNN**: supervised, nonparametric, nonlinear regressor/multiclassifier, predict by finding k nearest train samples by dist (instance-based learning)
- **(+)** handle nonlinear data well, white-box, no training **(-)** more train data or feats → slower inference, sensitive to noise & outliers, curse of dims
- **Assumptions**: 1. similar data points are close together in feat space 2. feats are scaled since distance-based
- **Inference**: compare n feats with all m train samples = $\mathcal{O}(mn)$, then sort calced distances and return k nearest = $\mathcal{O}(m \log m)$
- **k Tradeoff**: small k (high variance + might overfit + jagged boundary), large k (low variance + high bias + might underfit + smooth boundary)
- **Classification**: majority label of k nearest neighbors **Regression**: avg of k nearest neighbors **Choose k**: either use CV or approx as $\sqrt{m}$

Naive Bayes

- **NBayes**: supervised, multiclassifier, probabilistic, estimates posterior $P(y_i|\boldsymbol{X})$ via naive assumption = feats conditionally independent given y
- **(+)** works well in high dims (e.g. text), requires less data (doesn't need data demonstrating feat interactions) **Assumption**: little multicollinearity
- **Classification**: $P(y_i|\boldsymbol{X})$ = $\frac{P(\boldsymbol{X}\,|\,y_i)\cdot P(y_i)}{P(\boldsymbol{X})}$ = $\frac{P(\vec{x}_1,\vec{x}_2,\ldots,\vec{x}_n\,|\,y_i)\cdot P(y_i)}{P(\vec{x}_1,\vec{x}_2,\ldots,\vec{x}_n)}$ → naive assumption → = $P(\vec{x}_1\,|\,y_i)\cdot\ldots\cdot P(\vec{x}_n\,|\,y_i)\cdot P(y_i)$
- $P(y_i)$: prior proba of y_i class = $\frac{\text{examples labeled } y_i}{\text{total examples}}$, $P(\vec{x}_j|y_i)$: class conditional = PDF for each $\vec{x}_j$ feat with y_i class (k classes, n feats = kn PDFs)
- **PDFs**: fit PDF to each feat (e.g. binary → binomial, numeric → gaussian, unknown → kernel density estimation), update prior & PDFs when new data
- **MLE**: nbayes estimates priors from data instead of making assumptions about their distributions, it is not bayesian, simply uses MLE to find PDFs

KMeans Clustering

- **KMeans**: unsupervised, clustering, centroid-based, assigns instance to cluster based on distance from centroid **Assumptions**: feats are scaled
- **(+)** guaranteed convergence, relatively fast clustering ($\mathcal{O}(nmk)$ for each centroid update), cluster probas based on distance from centroids
- **(-)** bad if clusts vary in diameter/density or non-convex, sensitive to cluster init → rerun, k hard to choose, sensitive to outliers & curse of dims
- **Algo**: 0. rand init k samples as centroids, repeat: 1a. update cluster labels to nearest centroid, 1b. update centroids to center of new clusters
- **Convergence**: when pts reclassified to same centroid, local or global optima depends on centroid init, rerun and keep model with lowest inertia
- **Choose k**: 1. Elbow: as k↑ inertia↓, use elbow on k vs inertia graph 2. Silhouettes: compare avg silhouette coeff or cluster knives for each k

Principal Component Analysis (PCA)

- **PCA**: unsupervised, dim reduction, project data onto max variance orthogonal linear combos of old correlated dims = SVD on covariance matrix
- **(+)** removes correlated feats **(-)** new dims are linear combos → less interpretable, not scale/rotation/translation invariant, negative loading scores
- **Assumptions**: linear data (PCs are linear combos), standardized data, high variance is structure & low variance is noise, no significant outliers
- **Principal Axes**: orthonormal eigvecs **PCs**: data projected onto principal axes scale to 1 stdev **Loadings**: eigvec components = weight of old dims
- **Reduce**: standardize data → find orthogonal dims of highest variance → keep k PCs based on total variance to preserve (scree plot, keep 95%)