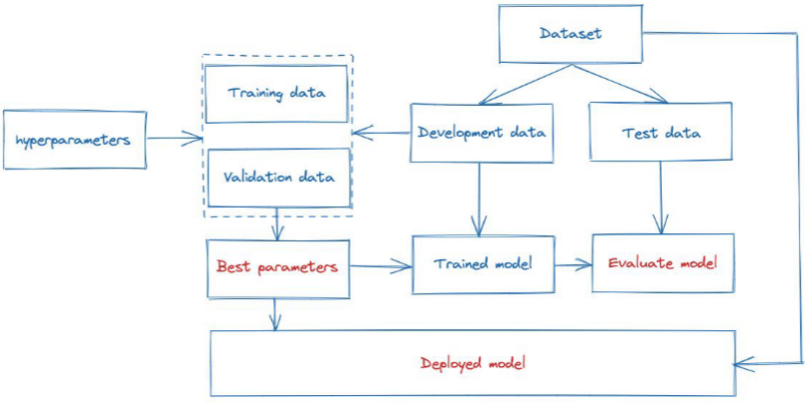
Midterm Preparation



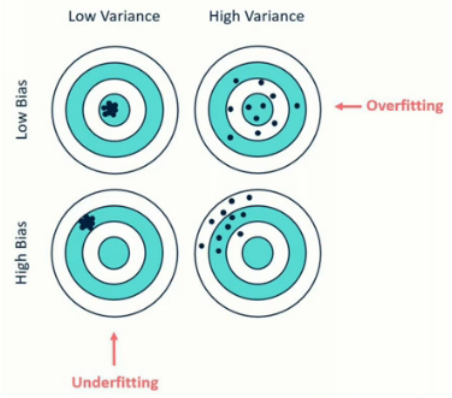
Supervised Learning Framework

▼ **Development-Test Split**

- Dataset is split into **development** and **test**.
- Splitting Strategies:
  1. **Random Splitting** – randomly
  2. **Stratified Splitting** – for highly imbalanced datasets, highly skewed.
  3. **Structured Splitting** – prevent data leakage

▼ **Hyperparameter Tuning**

- Model Complexity vs Model Performance
- Bias-Variance Tradeoff (high error → low error; high error → good error → high error)
  - **High Bias:** underfitting, models too simple, performs poorly on both training and testing data
  - **High Variance:** overfitting, models too strict, performs well on training data but not on test data



- **Hyper Parameter vs Model Parameters**
  - Model Parameters – learn from the data (typically with optimization function)
  - Hyperparameters - you choose!
    - **Decision Tree:** depth, criterion, min samples, split.
    - **Support Vector Machines:** kernel width, penalty.
    - **Neural Networks:** network topology, learning rate.
- **Hyperparameter Search**
  1. **Grid search** – search all in grid layout. – uninformed search strategies.

2. **Random search** – search random parameters
    a. performs better than gridsearch because there are unimportant parameters that gridsearch might consider and with random you have a chance to explore optimal hyper parameters.
    b. you can combine both → grid search after random search.
3. **Bayesian optimization** – informed search strategies.
    a. sequential, determined at how you perform at each points
4. Evolutionary optimization - not mentioned

How do we evaluate the hyper-parameters? We could use the test set but we use **validation** because we tune hyper-parameters on the test set and we only want to use it for the final evaluation. Split development dataset.

Why do you have a validation set? to test the performance of hyper-parameters, as we want to use the test set for the final evaluation.

- **Model Selection Strategies for Hyperparameter Search**
    - **Three-way holdout**
        - Development → Train and Validation → Model Selection is the best performance on validation data
        - Can get away with this for large data sets.
    - **Leave-one-out CV**
        - Set k = to the number of samples, when you have less data (n data points = n times)
    - **K-fold CV**
        - Split development data into k-folds (validation typically 3/5/10)
        - Train on k-1 folds, evaluate on last fold (repeat for each k fold) → estimate average model performance → choose best hyperparameter.
        - **Repeated** means you do this multiple times (don't recommend).
    - **Stratified K-fold CV** - each fold and test data is stratified.

Why do K-folds? How do you choose one of these?

▼ **Data Preprocessing**

- **Numerical Data**
    - Typically they have different scales →
        - **StandardScaler** → (X - μ) / σ → centered.
        - **MinMax** → map min and max to 0, 1.
        - **RobustScaler** → when data has outlier to not get influenced (standardscaler but with 95th percentile)
        - **MaxAbsoluteScaler** → preserve the sign of the data by dividing by the max absolute value.
        - **Normalizer** → Normalize by the columns, if you have a row, you divide by the norm of the vector → row scaling
    - fit_transform → figures out the μ σ with development data
    - transform → applies it to test data.
    - pipeline → give sequence of steps to apply same transformation on test data.
    - scaling with cross validation → scale → validate.

When does it help? Can set this as a hyperx-parameter with different techniques.

- **Missing Data**
    - Can be informative → make it an indicator.
    - Not informative
        - Few data → drop rows
        - Lots of data → drop col
        - Impute using mean or median
        - Label with KNN (nan_euclidean_distance metric), regression models, matrix factorization.
- **Categorical Data**
    - **Ordinal Encoding** - there is an order
    - **One-Hot Encoding** - no order

        - introduces multi-collinearity → drop them (interpretation is very important so might not be good) → instead apply regularization
        - some techniques can still take categorical data (tree-based, naive bayes)
        - leads to high-dimensional datasets
    - **Target Encoding** - map to one column instead of many → map to relationship with average target for that category.
- Combine Numerical and Categorical (make_column_transformer)
    - **Why might an NaN appear?** Categories that were not fitted before (e.g., OneHotEncoder) are introduced in test/new data. Use 'raise error' and 'handle_unknown' to deal with it.

▼ **Models – Post Processing**

▼ Linear Models for Regression

- **Linear Regression:** assumptions are ***linearity***, ***independence***, ***homoscedasticity*** (noise are randomly distributed around the mean), ***normality*** (of residuals).
    - transform and observe to see if assumptions are met
    - solve an unconstrained optimization problem → typically there is a **closed-form solution.**
        - **What is the closed form solution?** Given X = n (samples)*m (features), w = X transpose X over X transpose y.
        - Typically use gradient descent for unconstrained optimization where there is no closed-form solution.
        - **Why do you still need to do an iterative solution even with a closed-form sometimes?**
            - If X transpose is large, it can take a long time to invert the matrix. Instead you can do gradient descent. Size of X variables.
            - Inverse may not exist for X (e.g., determinant is 0)
        - **Why does it matter that the model is convex?** There exists a unique solution and you can use an iterative algorithm to find a solution. Performance of the model is stable.
    - find model **parameters** using **loss function.**
        - **What is the loss functioned used for linear regression?** least squares minimization. How does the loss function change with different regularization?
    - What are the problems?
        - *Too many predictors* (X is not a full rank matrix when m > n → XTX = m*m because the rank of the matrix is n)
        - *Multi-collinearity* - one feature is a linear combination of everything else. Check for high condition number for XTX.
- **Ridge Regression (L2 norm)**
    - When m (features) is large compared to n (samples), you must drop features (feature selection and preprocessing.)
    - Closed form solution exists!
        - w = (XTX + alpha I) over XTy
            - LSM - constrained optimization
            - **L2 norm:** constrain sum(w^2) ≤ c - always lies within a sphere
        - problem is still convex → use lagrangian multipler (unconstrained optimization).
        - **alpha** (hyperparameter) is inversely proportional to **c**.
        - **I** (identity matrix) is m * m.
    - What is the issue? biased (original assumptions of linear regression may not hold).
- **Lasso Regression (L1 Norm)**
    - Applying an sum(abs(w)) < c constraint
    - No closed-form solution, must use an iteration to find optimum (Least Angle Regression to solve)
    - Some of the weights are 0 (non-important) - Lasso does feature selection.
        - **alpha** decreases → features increases (becomes active) → coefficient increases
- **Elastic-net Regression:** Combine Lasso and Ridge with Convex Combination (**alpha** and **L1 ratio**)

What is an L2 vs L1 norm, when do you use one over the other? L2 has closed-form solution. L1 does feature selection.

- Contour hits a corner for Ridge (L2), for Lasso (L1) it hits a corner.

▼ Linear Models for Classification

Binary Classification

Logistic Regression

- Assume there is a linear relationship - y_prediction = sign(wTx + b)
  - find **w** and **b** by minimizing a **loss function.**
    - **0-1 loss function** → not easy to minimize (not differentiable, to make this problem easier define the following...)
    - **Hinge Loss** max(0, 1-y*f)
    - **Log Loss** (Logistic Regression – get a ***probability p(y|x)*** for a label)
    - **What is the functional form of a logistic regression?** S shape.
    - Maximize the **Log Likelihood** (LL) → convex so you can solve!

  → variance of Regularized Logistic Regressions

  - Lasso, Ridge, Elastic-net

SVM

- Maximize the margin classifier (Convex Optimization problem)
  - s.t.) 2 parallel linear boundaries
  - s.t.) No points fall between
- Constrained minimization problem: mimize w^2/2 s.t. y(wtx + b) ≥ 1
  - no closed-form solution → can use SMO or other methods.
- **Hard-margin** vs **Soft-margin**
  - Tradeoff between how much you can push (**margin**) and allowing no points (**error**).
  - Soft-margin includes a forcing function in the objective function → ζ → **Minimize Hinge Loss** → introduces upper bound **C** on alpha in the dual.
- Primal and Dual: the primal is a minimization → dual is maximization.
  - Primal is solved in **w** and **b** (m dimension vector)
  - Dual is solved on the **α** (n dimension vector) - both has same solution.

How do you choose between primal and dual? make a concious choice based on the size of your dataset (features vs samples).

What is the support vector? When you are making a transition from primal to dual, α(1 - y(wtx + b)) = 0, points when α is non-zero and closest to the hyperplane is the support vectors, think holding up the hyperplanes.

- Easily becomes non-linear → **Kernel Trick.**
  - Assume function **φ(x)** projects data to high dimensional space
  - **Kernel function K(x • x)** estimates inner product between two points in the projected space (gives you a number)
  - Do SVM but instead ***x = K(x • x) = φ(x)Tφ(x)***
    - If K is symmetric and positive definite, you can assume φ exists without finding it (the *trick!*).
    - **(RBF) Gaussian kernel** shows that it can be eventually projected to infinite dimensional space → at some values of gamma, training error = 0.

Multi-class Classification

- **OVO & OVR**
  - OVR → set of binary classification problems → rank order → choose best.
    - Non-balanced datasets, rarely a region of uncertainty
  - OVO → building one against the other → train nC2 classifiers → use classifiers and majority voting to say it belongs to a certain spot.
    - Balanced datasets, region of uncertainty

How do you predict using OVR vs OVO?

- **Multinomial Logistic Regression**
  - Another way to do multi-class, find the probability that p(y = class | x) and then find the maximum likelihood function

▼ Decision Trees

- **Greedy** algorithm to train
  - choose impure node → estimate reduction in **impurity** due to every feature → choose feature with maximum reduction → create two child nodes → repeat until stop (can become 100% pure with enough depth).
  - Splitting for continuous value vs categorical features:
    - Going through all values → measure IG → for numerical check all values; for categorical order by **MRR (mean response rate)** – target encoding, and decide best split.
  - Hyper-parameter to **prune** or **early stop** to avoid overfitting.
    - Pruning: **reduce error** and **cost complexity**.
    - Early Stop: **max depth, nodes, sample_split, IG**
- Non-linear, minimal preprocessing, invariant to scale
- **Exhaustive** → will always end up at a leaf node

▼ **Classification**

- **Majority Voting** by probability.
- Measure Impurity via **Gini Index** and **Entropy**.
  - Probability of split
  - **Information Gain** takes a ***weighted average*** to maximize the reduction in impurity → split.

▼ **Regression**

- **Sample Mean**
- Measure Impurity via **MSE** and **MAE**
- **SSE** to maximize reduction in impurity → split.

What is the difference between classification and regression in terms of prediction, measure impurity, and how to decide split?

- **Feature Importance**: which feature has most impact on IG, its depth and number of samples impacted.

How is the feature importance measured? It is the sum of a probability of a point reaching that node multiplied by its IG at that node.

Can a feature be used more than once to split? Yes, it cycles through all features to determine the optimal split at each node.

▼ Ensemble Methods

- Trees are highly unstable, limited capacity of regression values → build **ensembles**! → Instead of building 1 tree, build multiple → reduce model variance.
- You have bagging and boosting, difference is the **training dataset** and **how you get the output.**

▼ **Bagging (Bootstrap Aggregation)**

- **Sampling w replacement** to create training set of same size → build tree with each → average to make final prediction.
- **RF**
  - Hyper-parameters: # of trees, # of features (sqrt(m) for classification, m for regression)
  - **Out of Bag (OOB) error** for model selection, average error of a data point, calculated using prediction from trees that do not contain it in their respective bootstrap sample. **comes for free!**
  - You already have an amount of trees, and you can add more.

▼ **Boosting**

- ML algorithms to improve weak learners to stronger ones sequentially.
- Early learners fitting simple models to the data then analyzing data for errors.
- Each tree corrects previous tree.
- Tree finds prediction y → calculates error (y - y_prediction) → **inputs residual into next tree** → repeat until error is close to 0 (check with validation set).
- **Adaboost** (depth 1 (decision **stump**) → assume average **weight** → calculate error by summing up the weights → calculate alpha (high means low error) → set weight based on alpha.
- **Gradient Boosting Algorithm:** ALWAYS building **regression trees** → Initialize **F0 Classifier** (model predicts the mean of your target) and **calculate residual** → for iterations, compute pseudo residual →

    - Gradient is the residual
  - Hyper-parameters - **Number of trees**, **learning rate, depth**, **regularization** decision tree parameters, row sampling, column sampling.

  What is the gradient in gradient boosted trees? The gradient, derivative of error with respect to F(m), becomes the residual

  - **GradientBoostingClassifier** - slow on large data
  - **HistGradientBoosting** - faster on large datasets, iterates through all values of histogram.
  - **XGBoost** - Fast approximate split finding based on histograms, supports GPU training, sparse data & missing values, adds L1 and L2 penalty on leaf weights, monotonicity
  - **LightGBM** - Microsoft version, faster than XGBoost on CPUs, Distributed Training, CLI.
  - **CatBoost** - Optimized for categorical features

▼ **Model Evaluation**

- Evaluation methods for Classification
  - **Threshold-based Metrics** ← use score to assign label (lower threshold → recall; higher threshold → precision)
    - **Classification Accuracy -** how accurate is your classification - misleading for imbalanced datasets - **Accuracy Paradox**
    - **Precision, Recall & F1-score** - good for imbalanced datasets - **positive** class is the **minority class**.
      - **Macro** (direct average) **vs Weighted** (average by size) for each class.
      - **Weighted Recall = Accuracy**

  What is the confusion matrix? matrix of actual and predicted positives and negatives.

  - **Ranking-based Metrics** ← use score to rank
    - **Average Precision (AP)**
      - **PR Curve** to compare models
    - **AUROC** - determines relationship between FPR and TPR (Recall).
      - can say whether the model is good or bad.
- Evaluation metric for Regresstion
  - R^2, MSE, MAE, MAPE (Percentage Error)

▼ Multi-Fidelity Optimization

- **Successive Halving** reduce hyper-parameters by halves as you approach the budget
- **Hyperband** takes care of hyper-parameters by training at different budgets.