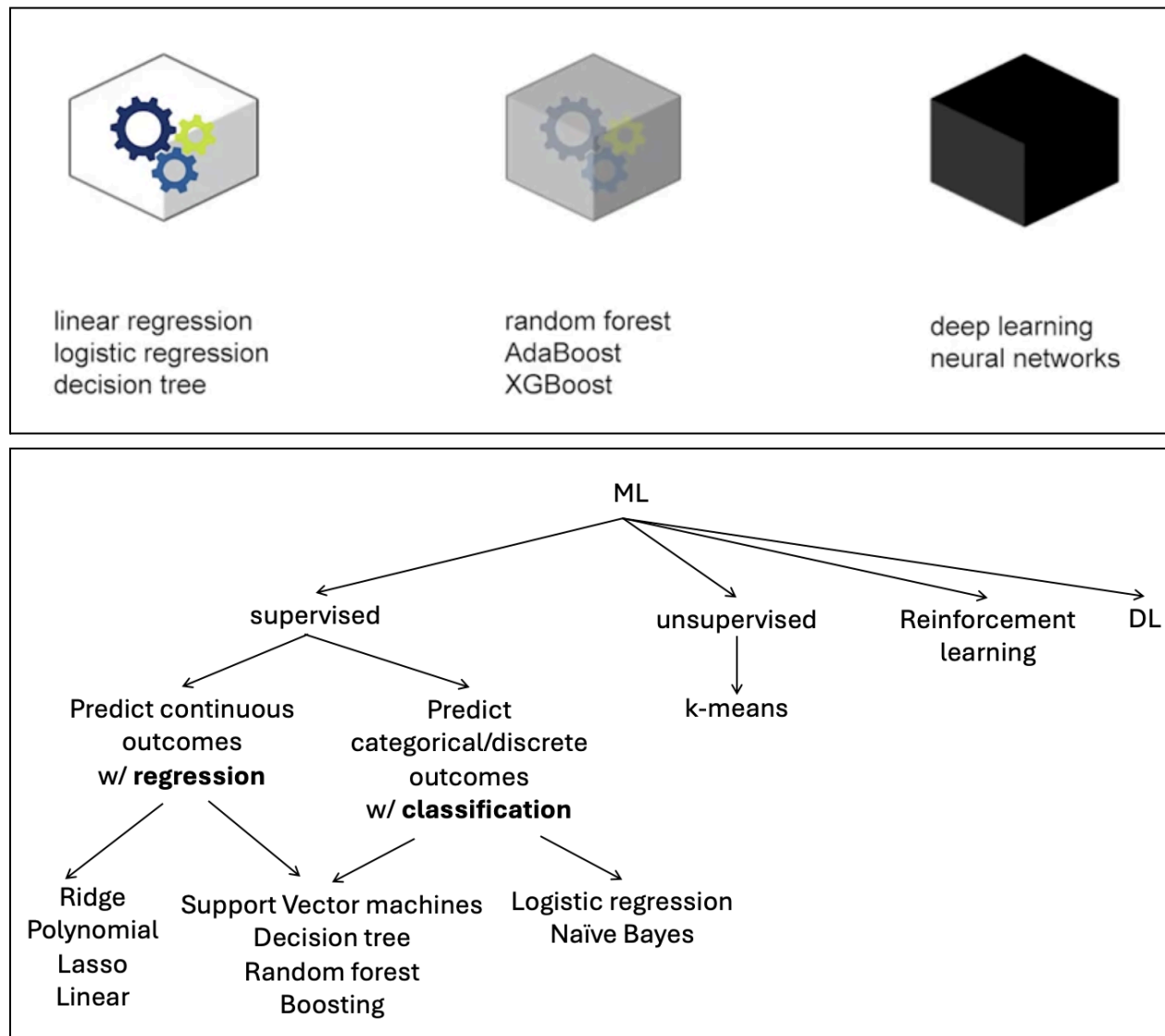
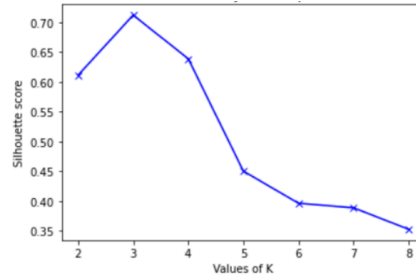


My notes & understanding from the course:



~ k-means

- Picks centroid datapoints (the number depends on k) that are furthest apart, and changes them iteratively until all the other data points are closest to a particular centroid and clustered by k number of groups
- Decide k with elbow method / silhouette coefficient

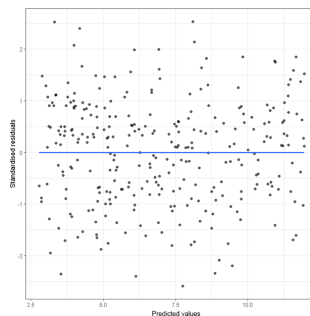


Feature engineering (for supervised ML)

- Feature selection
- Feature transformation
 - Log: for skewed distributions
 - Scaling
 - Normalization (eg MinMaxScaler (0-1))
 - Standardization (eg StandardScaler)
 - One-hot encoding = for converting any categorical IVs to Boolean
- Feature extraction = make new IVs using other ones
- Class imbalance: of the target variable, eg 90 vs 10
 - Tx: downsampling (for large ds's) / upsampling (for small ds's)

~ Linear regression

- Estimates mean
- Correlation-based
- Minimizes loss function; calculated with OLS (ordinary least squares)
- Assumptions: linearity, normality of residuals (also called errors = predicted value - real value), homoscedasticity, independent observations



- Homoscedasticity graph:
- Formula: $y = B_0 + B_1X_1$

~ Multiple Linear regression

- For: 2/+ independent variables (aka features)

- Formula: $y = B_0 + B_1X_1 + B_2X_2 \dots$
- Assumptions: linearity, normality of residuals (also called errors = predicted value - real value), homoscedasticity, independent observations + no multicollinearity (e.g. X_1 & X_2 cannot be linearly related)
 - Deal with multicollinearity: forward selection, reverse elimination, Lasso regression, PCA
- Use adjusted R^2 which accounts for adding more IVs

~ Logistic regression (can be binomial or multinomial)

- For: binary target variables
- Link function-based
- Estimates probability of an outcome
- For binomial logistic regression:
 - Assumptions: linearity (logit = $\log(P/1-P)$), independent observations, no multicollinearity, no extreme outliers

~ Naive Bayes

- Calculates posterior probability
- Assumptions: independent predictors
- Types:
 - Gaussian: for continuous, normally distributed features
 - Multinomial: for discrete features
 - Bernoulli: for Boolean features
 - Categorical: for categorical features

~ Decision trees

- Chooses splits with Gini impurity (closer to 0 the better), entropy, info gain, log loss

Ensemble learning

ML technique where 2+ models (called base learners) are combined

Each base learner has different set of training data

Types:

- Bagging = bootstrap (sampling with replacement) aggregation (predictions of all base learners (all the individual models) are aggregated)

- Each base learner samples from the data with replacement, so various base learners sample some of the same observations
 - I.e. a copy of the data is made to train each base learner, but each base learner's copy is a bit different
- Output:
 - average the predictions (for regression models)
 - random forests (for decision tree learners)
 - mode of predictions (for classification models)
- Boosting
 - XGBoost
- Stacking
- Voting

~ Random forests

= bagging + random feature sampling (no single tree sees all features used for the model)

- Additional hyperparameters:
 - Max_features = number of features each tree can select
 - N_estimators = how many trees

~ Boosting

= builds ensemble of weak learners sequentially (not in parallel like random forests), with each consecutive learner trying to correct errors of the one that preceded it

Adaboost

Assigns weight to incorrect observation

Gradient boosting

Each base learner built to predict residual errors of the model preceding it

Works with missing data

Has many hyperparameters

Can be hard to interpret

XGBoost

Optimized GBM (gradient boosting machine)

- Max_depth: 2-10 usually
- N_estimators: 50-500 usually
 - number of base learners the ensemble will grow
- Learning rate (shrinkage): 0.01-0.3 usually
 - weight given to each consecutive tree's prediction in final ensemble
 - less weight for each subsequent tree

- Min_child_weight: 0-1 (interpreted as percentage)
 - tree won't split a node if any child node will have less than this weight

Hyperparameter tuning (for decision trees, random forests, boosting)

- GridSearchCV
 - For a list of hyperparameters to be tuned, it tests all combos of those hyperparameters
 - Possible hyperparameters:
 - Max_depth = number of levels
 - Min_samples_split
 - Min_samples_leaf
 - Outputs (related to performance evaluation section): F1 score, precision, recall, accuracy

Performance evaluation (for Decision tree, Random forest, Boosting, Logistic regression, Naive bayes)

- Confusion matrix (TP, TN, FP, FN)
- Precision
- Recall
- Accuracy
- F1 score: higher the better
- FB score
- ROC curve (and AUC): TP rate vs FP rate

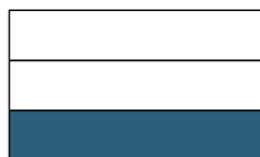
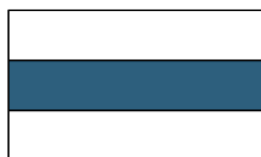
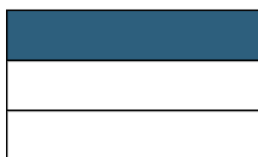
Validation



Use validation dataset as the test data set to compare your models, and then use the test data as final test with your chosen/champion model

k-fold cross validation

Training data is split into k folds, one of the folds is the validation fold, the model is trained on the other folds, and tested on the validation fold, repeat this process so that each fold becomes the validation fold, then average the k-scores. Lastly, the model is refit to all of the training data.

k = 3



 Validation fold
 Train fold

Overall steps:

		k-means	Linear regression	Multiple linear regression	Logistic regression	Naive Bayes	Decision trees	Random forest	Boosting
Clean up data (remove NAs)									
Feature engineering	Feature selection								
	No multicollinearity	n/a	n/a						
	Log transformation for skewed								
	Scaling - Normalization (MinMaxScaler)								
	Scaling - Standardization (StandardScaler)								
	One hot encoding for categorical features								
	Class imbalance					Depends on type			
Separate target variable from rest of data									
Split into train, test data									
Instantiate the model									
Fit the model									
Hyperparameter tuning									
Test the model									
Evaluate	elbow method / silhouette coefficient								
	MSE								
	RMSE								
	R^2								
	Adj R^2								
	Confusion matrix								
	Precision								
	Recall								
	Accuracy								
	F1 score								
	ROC curve								
Bonus: validation data used as test data, and test data used at very end with champion model									

Pseudo-code outline of gradient boosting for an ensemble of just three trees:

```

learner1.fit(X, y)           # Fit the data
ŷ1 = learner1.predict(X)     # Predict on X
error1 = y - ŷ1              # Calculate the error → (actual - predicted)
learner2.fit(X, error1)      # Fit tree 2, but target = error from tree 1
ŷ2 = learner2.predict(X)     # Predict on X
error2 = ŷ2 - error1         # Calculate the new error
learner3.fit(X, error2)      # Fit tree 3, but target = error from tree 2
ŷ3 = learner3.predict(X)     # Predict on X
error3 = ŷ3 - error2         # Calculate the new error

```

Final prediction = learner1.predict(X) + learner2.predict(X) + learner3.predict(X)