QSS20: Modern Statistical Computing

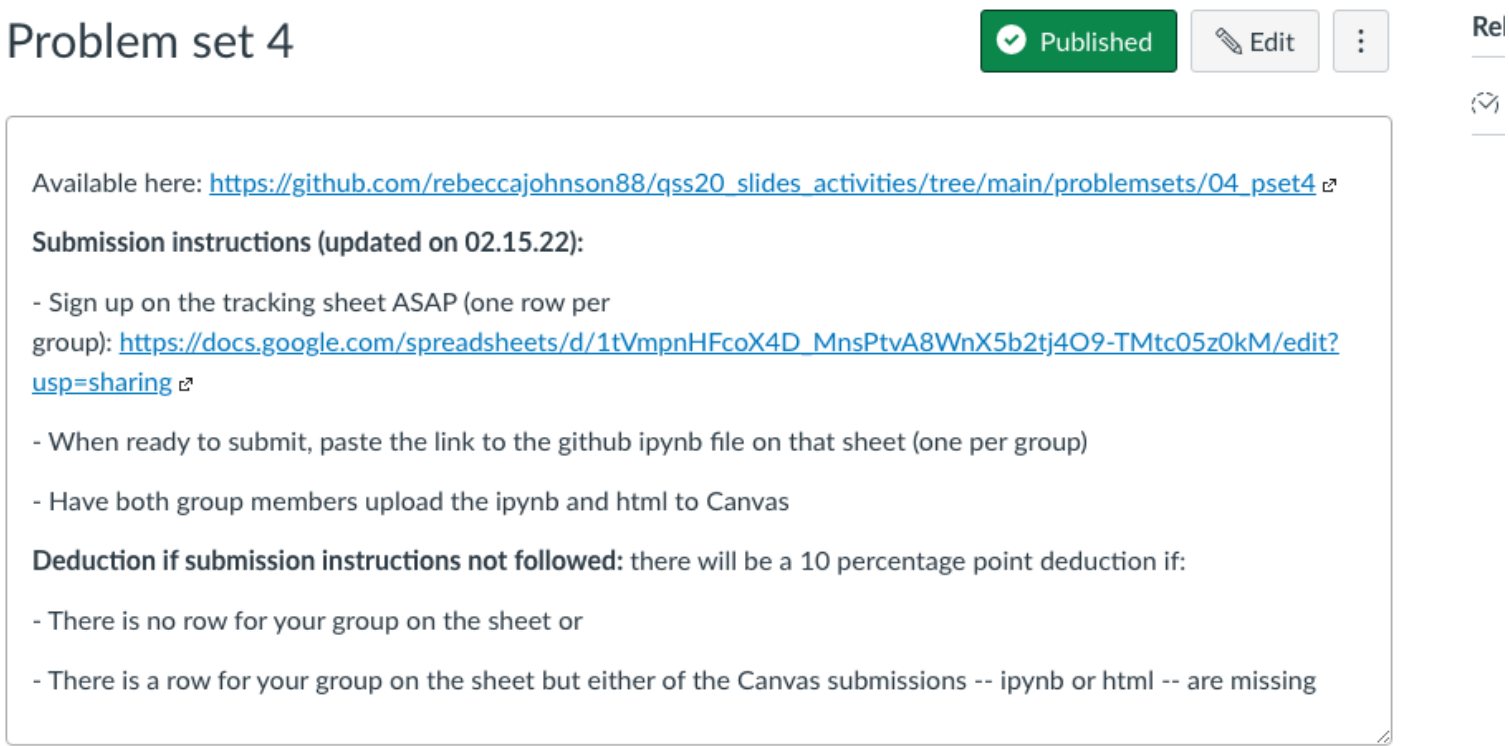
Unit 10: Intro to Supervised Machine Learning (SML)

Plan for class

- **Housekeeping**
- Supervised machine learning: why and two major types of classifiers
- Supervised machine learning: how in Python

Housekeeping

- Pset 3 grades and feedback: tonight/tomorrow
- Pset 4 (due Friday 02.25): 10 percentage point deduction if not submitted following instructions



- Final project milestone 2 (due Sunday 02.27):

Plan for class

- Housekeeping
- **Supervised machine learning: why and two major types of classifiers**
- Supervised machine learning: how in Python

Where has machine learning come up already?

- **Sentiment classification:** `VADER` scores sentiment using a dictionary, but dictionaries work more or less well depending on the context
  - **Solution:** supervised machine learning where we:
    1. Take data labeled with positive/negative sentiment (binary) or sentiment score (continuous)
    2. Create a document-term matrix
    3. Predict sentiment using the terms as predictors
    4. Use that to score the sentiment of unlabeled data
- **Topic modeling:** LDA is a form of unsupervised machine learning that takes an unlabeled input—a document-term matrix—and returns high probability topics and words
- **Record linkage:** k-means is a form of unsupervised machine learning where we feed the algorithm a vector of 1/0 reflecting a pair's matches or not on different fuzzy variables and we cluster the pairs into "likely match" and "likely not match"

Terminology

- **Unsupervised versus supervised learning:** unsupervised learning is a form of dimension reduction/clustering on unlabeled data; supervised learning is used for prediction on labeled data
- **What do we mean by "labeled":** in social science, we use the terms "outcome variable" or "dependent variable"; in ML, we use the terms "label" or "target"
- **Two main types of supervised machine learning:**
  1. Regression: predict numeric values
  2. **Classification:** predict categories
     - **Binary classification:** predict yes/no or True/false category (e.g., fraudulent or not; experiencing homelessness or not; positive sentiment or not)
     - **Multiclass** > 2 categories
- **In supervised machine learning, what do we use to predict the label?** features/predictors (in social science terms: covariates)
- **Main goal:** generalization, or a model that predicts the response on novel inputs/new data it hasn't yet seen

When is SML particularly helpful?

- Consider a traditional linear or logistic regression with two predictors:

$$\text{positive sentiment}_i = \alpha + \beta_1 \times \text{uses word great } (1 = \text{yes})_i + \beta_2 \times \text{uses word terrible } (1 = \text{yes})_i + \epsilon_i$$

- What if we want to predict sentiment using many terms, with the number of terms possible exceeding the number of observations? need principled way to:
  - Predict sentiment, or optimize predictions, while....
  - Avoiding overfitting, or making the model too sensitive to idiosyncratic noise in the data
- Two tools help us with that task:
  1. More flexible classifiers than ordinary least squares: $y = f(x)$, where f is a function mapping some input (e.g., a document-term matrix) to a label (y); go beyond OLS or generalized linear models (GLM) for the mapping function
  2. Sample splitting: we estimate/train the model in one random split of the data; we evaluate its performance in a separate split of the data

First main approach: regularization

- **Bias-variance tradeoff:** ordinary least squares (OLS) is guaranteed to minimize `bias` in the coefficients, but at the cost of variance (small changes in the input data can lead to large changes in $\hat{\beta}$)
- **Regularization:** add a penalty term that decreases the complexity of the model by penalizing $\hat{\beta}$ based on distance from zero
- **OLS loss function:**, where i indexes an observation and j indexes a predictor/covariate:

$$\hat{\beta}_{LS} = \sum_{i=1}^{n}(y_i - \sum_{j=1}^{p} x_{ij}\hat{\beta}_j)^2$$

- **LASSO/L_1 regularization**: absolute value; tends to shrink to zero

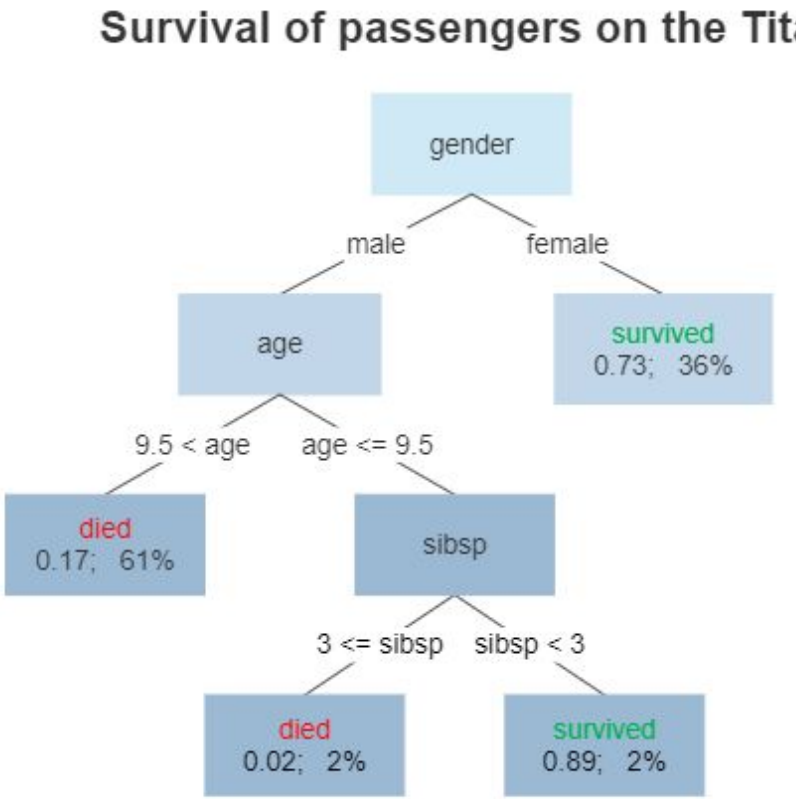
$$\hat{\beta}_{Lasso} = \sum_{i=1}^{n}(y_i - \sum_{j=1}^{p} x_{ij}\hat{\beta})^2 + \lambda \sum_{j=1}^{p}|\hat{\beta}_j|$$

- **Ridge/L_2 regularization**: squared; shrinks smaller but less likely towards zero

$$\hat{\beta}_{Ridge} = \sum_{i=1}^{n}(y_i - \sum_{j=1}^{p} x_{ij}\hat{\beta})^2 + \lambda \sum_{j=1}^{p}(\hat{\beta}_j)^2$$

Second main approach: tree-based methods

- Regularized regression helps w/ overfitting but still maps features to the label in a linear way (e.g., survival = -2.5 $\times$ age + 0.25 $\times$ female)
- Tree-based models help us learn non-linear relationships- eg



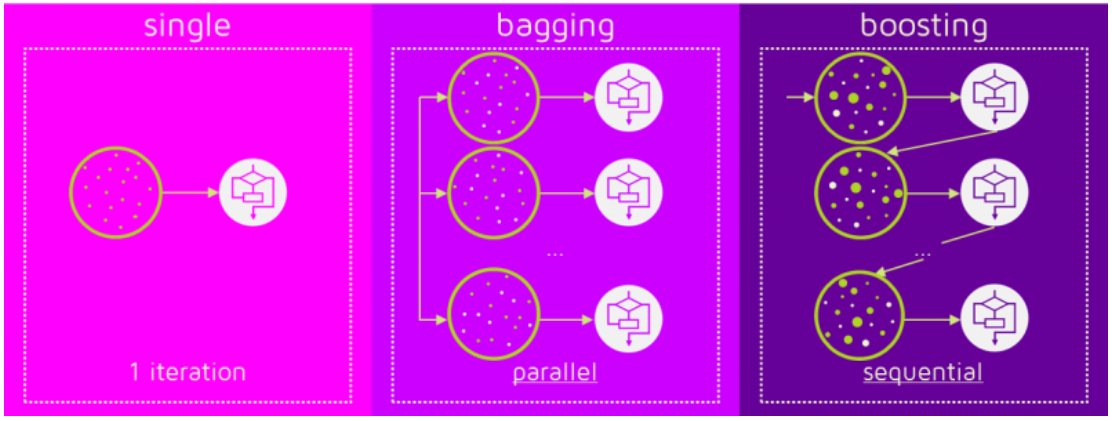
Graphics source

Logic behind classification trees/random forests

- **Classification and regression trees (CART):** take an input space—in the case of the Titanic, one comprised of gender; age; number of family on ship—and recursively partition that space into regions with different local models (in the Titanic case, binary prediction of survival or mortality)
- **Growing a tree:**
  - Categorical variables (e.g., male or female): consider split on each category
  - Continuous variables (e.g., age): find sorted/unique values and consider thresholds (τ)
  - What do we use to evaluate the splits? evaluate how much a split reduces our loss/cost function (e.g., gain in "information" from split); evaluate splits against max. tree depth; ensure sufficient observations in terminal leafs
- **Random forests:** randomly sample from **both** the rows (observations) and columns (inputs/predictors), fit many trees, and take average over predictions (bagging) to aggregate over the trees; longer computational time but less overfitting

Other extensions of trees: "boosted" trees

- Random forest is an example of the bagging approach to ensemble learning, where we fit many trees in parallel with subsets of observations/predictors



- Another form of ensemble learning is boosting, or a sequential process where we start with a shallow or "weak learner" (basically, something only slightly better than random guess) to predict the outcome, examine the predictions or residuals, target misclassification in next iteration
  - **AdaBoost:** upweight misclassified observations from previous iteration
  - **GradientBoosting:** more general framework for boosting; calculates residuals from previous iteration $(y - \hat{y})$ and predicts those residuals in next iteration

Diagram source QuantDare

Plan for class

- Housekeeping
- Supervised machine learning: why and two major types of classifiers
- **Supervised machine learning: how in Python**

Four DataCamp chapters

1. **Classification**: uses `k-nearest neighbors`; in activity, we'll shift to three other classifiers (decision tree; ridge; LASSO)
2. **Regression**: covers K-fold cross-validation and ridge/lasso
3. **Fine-tuning your model:** ROC curve with different $\hat{y}$ thresholds for continuous logistic regression predictions; hyperparameter tuning with `GridSearchCV` and `RandomizedSearchCV`
4. **Preprocessing and pipelines:** transforming categorical features into dummy indicators; imputation; standardizing inputs

DataCamp: pros/cons of their approach

- **What I liked:**
  - Emphasis on important of preprocessing for model accuracy
  - Start with a model where the hyperparameters (e.g., α/λ in LASSO/ridge are fixed) and move to models where we iterate over a grid of hyperparameters and choose the one that maximizes eval metric
- **What I liked less:**
  - **Reliance on black-box functions**: rather than using functions like `train_test_split` and `score`, personally believe it's better (since we're already black-boxing the algorithms) to do these steps ourselves:
    - **Splitting:** can use `random` module; increased flexibility to (1) balance features/outcomes across the splits; (2) account for temporal or row-based clustering (eg if we observe a person or employer multiple times in the data, want all their rows in one split)
    - **Evaluation:** calculate metrics using $\hat{y}$ (predicted response; binary or continuous 0-1) versus y
  - **Rationale for imputation**: less about "losing data"; more about biases if rows missing some values differ systematically from those w/ complete data

Example classification task

Predicting positive sentiment (`1 = positive`) using text of Yelp reviews

metadata_label	raw_text
0	Unfortunately, the frustration of being Dr. Goldberg's patient is a repeat of the experience I've had with so many other doctors in NYC -- good doctor, terrible staff. It seems that his staff simply never answers the phone. It usually takes 2 hours of repeated calling to get an answer. Who has time for that or wants to deal with it? I have run into this problem with many other doctors and I just don't get it. You have office workers, you have patients with medical needs, why isn't anyone answering the phone? It's incomprehensible and not work the aggravation. It's with regret that I feel that I have to give Dr. Goldberg 2 stars.
1	Been going to Dr. Goldberg for over 10 years. I think I was one of his 1st patients when he started at MHMG. He's been great over the years and is really all about the big picture. It is because of him, not my now former gyn Dr. Markoff, that I found out I have fibroids. He explores all options with you and is very patient and understanding. He doesn't judge and asks all the right questions. Very thorough and wants to be kept in the loop on every aspect of your medical health and your life.
0	I don't know what Dr. Goldberg was like before moving to Arizona, but let me tell you, STAY AWAY from this doctor and this office. I was going to Dr. Johnson before he left and Goldberg took over when Johnson left. He is not a caring doctor. He is only interested in the co-pay and having you come in for medication refills every month. He will not give refills and could less about patients's financial situations. Trying to get your 90 days mail away pharmacy prescriptions through this guy is a joke. And to make matters even worse, his office staff is incompetent. 90% of the time when you call the office, they'll put you through to a voice mail, that NO ONE ever answers or returns your call. Both my adult

Labeled Yelp data (subsample of 15k): Zhang, Zhan, Lecun, 2015. "Character-level Convolutional Networks for Text Classification"

Notebook to follow along

09_binaryclassification_activity_blank.ipynb

Step 1: preprocess the data

See slides here for process of document-term matrix creation

Step 1: terms versus negative label

Uses word “frustrat” and negative label:

	metadata_label	raw_text	frustrat
0	0	Say \"BITES.\" Say \"WINGS.\"\n\nApparently, my waiter did not know the difference between \"WINGS\" and \"BITES,\" because when my roommate and I ordered buffalo BITES, we got WINGS instead. We hate wings and love bonless buffalo BITES. It was even more frustrating when our buddy came to join us at happy hour and the waiter spoke for two minutes about how he prefers the BITES over the WINGS.\n\nI know I probably sound dramatic, but I will NOT be back...	1
1	0	This used to be my \"go to\" place for groceries. However, lately I'm becoming more and more frustrated by the store. If you go between 4pm and 6pm on a weekday or at any time on Saturday or Sunday, be prepared to navigate a mess of a parking lot, an over crowded store, and a 20 minute wait in a checkout lane. \n\nOn a Sunday, it's impossible to find parking even though there is outdoor parking as well as a 2 story garage. Once you get in the store, you have to navigate through an obstacle course of a produce section and around overwhelmed first-timers who are trying to figure out where everything is. It's entirely too small for a place that markets itself towards the \"fresh foods\" end of things (as opposed to a regular Giant Eagle). Then you have to figure out where everything is in the aisles. I get the impression that things were just randomly arranged around the store. Once you actually have your groceries, the waiting begins	1

Step 1: terms versus positive label

Uses word “frustrat” and positive label:

	metadata_label	raw_text	frustrat
	1	This review is based on this location's truck delivery alone. I jumped on their 50% off patio furniture sale last month and I took advantage of their delivery for $56. As long as you order everything online, they will deliver it all for that one fee...even if they have to make multiple trips. Score!\n\nIf that isn't fantastic enough, when the delivery guys came, they even took extra time to put together my furniture. Seriously, that saved me hours of work and frustration. They were so nice to do it and they took the cardboard boxes with them. \n\nThe managers that called to schedule the deliveries or confirm a return were so courteous! They didn't waste any time getting me my items. t could not have asked for a more smooth process and enjoyable online purchase!	1
	1	Perry has been cutting my hair for about 4 to 5 years. Whenever I walk out of there I cannot believe how my hair looks, and I oftern get complete strangers complimenting my cut. Perry does a great job of taking my preferences and developing cuts that flatter me. What more can I ask?\n\nThe massage before the shampoo is standard and more than once I've fallen asleep. Once a couple years ago, in the late afernoon of a particularly frustrating day at work, they asked if I wanted anything to drink. I said, only if you have vodka. Next thing I know, a clear drink on ice shows up...and tasted suspiciously like vodka. No that's	1

Step 2: split into train and test set (more manual; notebook also has `train_test_split`)

```
## define the number of rows in training set (20\% test)
## and sample
nrows_train = round(X.shape[0]*0.8)
nrows_test = X.shape[0] - nrows_train
random.seed(221)
train_ids = random.sample(set(X['metadata_rowid']),
                          nrows_train)
## function fed ids and name of id col
def my_split(train_ids, id_col):
    test_ids = set(X[id_col]).difference(train_ids)
    X_train_man = X[X[id_col].isin(train_ids)].copy()
    X_test_man = X[X[id_col].isin(test_ids)].copy()
    y_train_man = y[y.index.isin(train_ids)].iloc[:,
                    0].to_numpy()
    y_test_man = y[y.index.isin(test_ids)].iloc[:,
                    0].to_numpy()
    return(X_train_man, X_test_man, y_train_man, y_test_man)
```

Step 3: estimate model in training data (here, we're hardcoding the hyperparameters)

```
## list of nonfeatures
non_feat = ['metadata_rowid', 'raw_text', 'process_text']

## initialize classifier
dt = DecisionTreeClassifier(random_state=0, max_depth = 10)

## estimate the model on training data/training label
dt.fit(X_train_man[[col for col in X_train.columns
                    if col not in non_feat]],
                    y_train_man)
```

Breaking things down:

- `non_feat`: want to shield things like our `id` variable from being treated as a feature; keep it in matrix for later post-modeling dx
- `dt`: initialize the decision-tree classifier; this is a `DecisionTreeClassifier` class within sklearn; telling it max tree depth is 10 features
- `dt.fit()`: use the `fit()` method from class; feed it training data and training label
- **Note:** skipped standardization bc features already on common scale

Step 4: apply the model to test set to generate binary and continuous predictions

```
y_pred = dt.predict(X_test_man[[col for col
                 in X_test_man.columns if "metadata_rowid"
                 not in col]])
y_predprob = dt.predict_proba(X_test_man[[col for col
                 in X_test_man.columns if "metadata_rowid"
                 not in col]])
```

Breaking things down:

- `predict`: predicted binary class (default cutoff $\sim$ 0.5)
- `predict_proba`: continuous predicted probability (0-1 range for classification)
- **To discuss:** why don't I need `y_test` at this stage?
- **To discuss:** what needs to be the same about the training and test set in order for me to generate these predictions? (e.g., # of rows; columns?)

Output of step 4: one or two-D array with predicted labels or probabilities

```
## print the results
y_pred[0:5]
y_predprob[0:5]
```

```
array([0, 0, 1, 0, 0])
```

```
array([[0.70544662, 0.29455338],
       [0.70544662, 0.29455338],
       [0.1918429 , 0.8081571 ],
       [0.70544662, 0.29455338],
       [0.70544662, 0.29455338]])
```

Step 5: use the $\hat{y}$ from step 4 to evaluate accuracy

```
## dataframe of binary, continuous, and true labels
y_pred_df = pd.DataFrame({'y_pred_binary': y_pred,
                          'y_pred_continuous':
                          [one_prob[1]  for one_prob in y_predprob],
                          'y_true': y_test_man})

## use np.select to code each obs to its error cat
error_cond = [  (y_pred_df['y_true'] == 1) &
                (y_pred_df['y_pred_binary'] == 1),
                 (y_pred_df['y_true'] == 1) &
                 (y_pred_df['y_pred_binary'] == 0),
                (y_pred_df['y_true'] == 0) &
                (y_pred_df['y_pred_binary'] == 0)]
error_codeto = ["TP", "FN", "TN"]

y_pred_df['error_cat'] = np.select(error_cond,
                           error_codeto, default = "FP")

## see notebook for calculations using the error_cat column
```

Step 6: interpret the model

```
feat_imp = pd.DataFrame({'feature_imp': dt.feature_importances_,
                         'feature_name':
                        [col for col in X_train.columns
                        if col not in non_feat]})
```

Breaking it down:

- `feature_importances_` is an attribute of the estimated decision tree
- **Important note:** tree-based feature importances just have magnitude and not direction; in this case, an "important" feature might be either highly predictive of positive sentiment (1) or highly predictive of negative sentiment (0)
- `coef_` in regularized logistic regressions have both magnitude and direction

Can then generalize this basic setup in many ways:

- **Within the same model, parametrize the hyperparameters and iterate over a grid**: code has example of more manual via putting the `ridge` model into a function and parameterizing the α/λ parameter
- **Iterate over models and hyperparameters:** can create a function that takes in a sklearn classifier and then use list comprehension to iterate over classifiers
- **Parametrize the evaluation metrics:** create a function that returns different eval metrics; use that to simultaneously evaluate many models
- **Parametrize the feature sets used for prediction:** e.g., 6A group project on predicting Bitcoin pricing; historical price data versus tweet sentiment versus both combined

Break for activity (section 4 in notebook)

Notebook: https://github.com/rebeccajohnson88/qss20_slides_activities/blob/main/activities/w22_activities/09_binaryclassification_activity_blank.ipynb

1. Read the documentation here to initialize a ridge regression (l2 penalty)- you can use the same cost parameter (C) and number of iterations as in the lasso example above: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html
2. Fit the model using `X_train_man, y_train_main`
3. Generate binary and continuous predictions
4. Create a function that takes in a dataframe of binary predictions and true labels and calculates the F_1 score using this formula:

$$F_1 = 2 * \frac{precision * recall}{precision + recall} = \frac{TP}{TP + 0.5(FP + FN)}$$

5. Apply that function to calculate the F1 score for the decision tree and lasso (from above), and ridge regression (from the activity)
6. **Challenge exercise**: parametrize the model fitting with a function that takes in a classifier as an argument and returns coefficients or feature importances and certain eval metrics (eg precision, recall, and F1)