

# QSS20: Modern Statistical Computing

Unit 11: Supervised Machine Learning, Part I (Intro and regularization)

# Goals for this week

## **Today: Introduction to supervised machine learning**

- ▶ Recap of Twitter API
- ▶ Lecture on basics of supervised ML
- ▶ Walkthrough of supervised ML in Python
- ▶ Activity on basic ML

## **Wednesday: Decision Trees and optimizing supervised ML models**

- ▶ Recap of basic supervised ML
- ▶ Lecture on Decision Trees & optimization
- ▶ Activity on DTs & optimization

# Goals for today

## **Today: Introduction to supervised machine learning**

- ▶ **Recap of Twitter API**
- ▶ Lecture on basics of supervised ML
- ▶ Walkthrough of supervised ML in Python
- ▶ Activity on basic ML

## Wednesday: Decision Trees and optimizing supervised ML models

- ▶ Recap of basic supervised ML
- ▶ Lecture on Decision Trees & optimization
- ▶ Activity on DTs & optimization

# Recap of Twitter API

What do you remember?

# Recap of Twitter API

## **Tips:**

- ▶ Words

## **Useful commands:**

code

# Goals for today

## **Today: Introduction to supervised machine learning**

- ▶ Recap of Twitter API
- ▶ **Lecture on basics of supervised ML**
- ▶ Walkthrough of supervised ML in Python
- ▶ Activity on basic ML

## Wednesday: Decision Trees and optimizing supervised ML models

- ▶ Recap of basic supervised ML
- ▶ Lecture on Decision Trees & optimization
- ▶ Activity on DTs & optimization

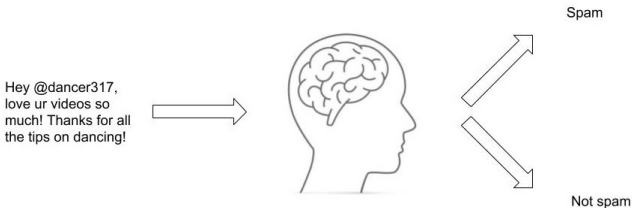
## Example of supervised ML: classifying spam emails

**Which of these are spam, and which aren't?** What do you notice that can help make this distinction?

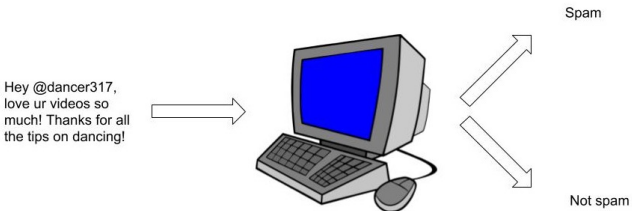
- ▶ *Hey @dancer317, love ur videos so much! Thanks for all the tips on dancing!*
- ▶ *OUR LASER PRINTER/FAX/COPIER TONER CARTRIDGE PRICES NOW AS LOW AS 39 DOLLARS. SPECIALS WEEKLY ON ALL LASER PRINTER SUPPLIES. WE CARRY MOST ALL LASER PRINTER CARTRIDGES, FAX SUPPLIES AND COPIER TONERS AT WAREHOUSE PRICES*
- ▶ *I'm not sold on your first point about crossing national boundaries, but I see what you mean about non-economic alternatives.*
- ▶ *Some of the most beautiful women in the world bare it all for you. Denise Richards, Britney Spears, Jessica Simpson, and many more. [CLICK HERE FOR NUDE CELEBS](#)*

# Example of supervised ML: classifying spam emails

What you just learned to do is text classification!



We want our machines to learn to do this, like so:





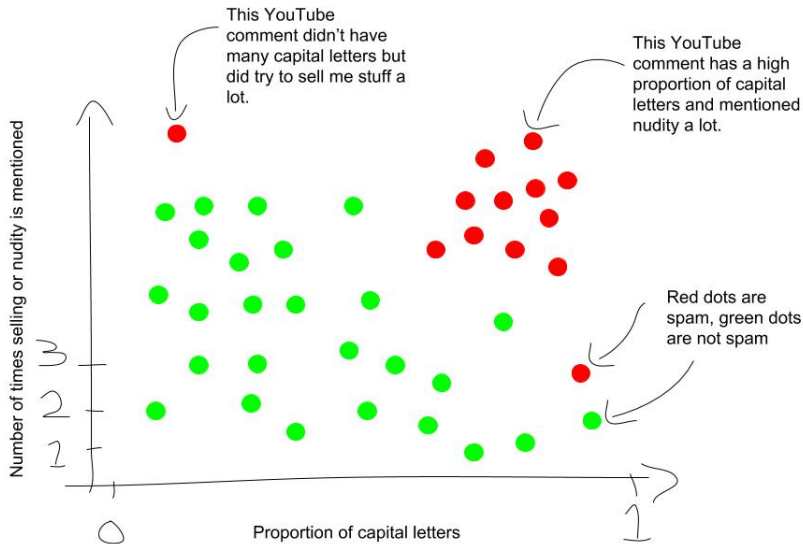
# Example of supervised ML: classifying spam emails

How our machines see the data:

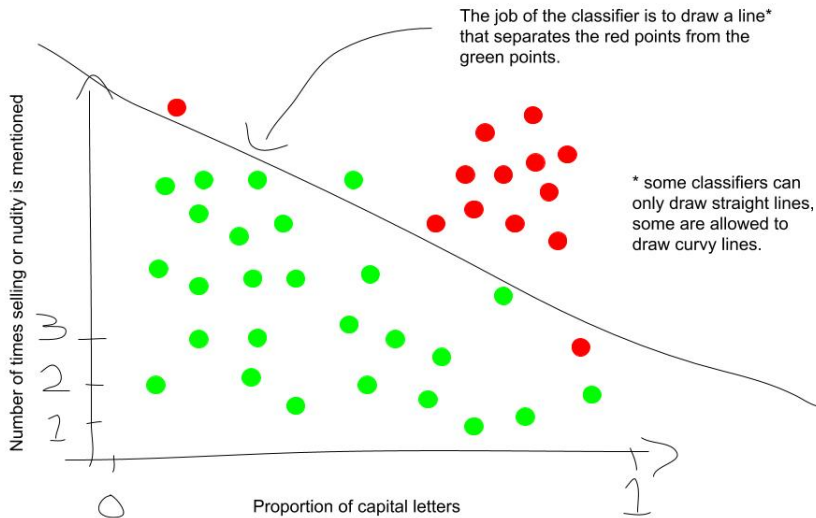
Comment	Selling or nudity	Proportion capital letters	Is it spam?
Hey @dancer317, love ur videos so much! Thanks for ...	0	0.1	No
OUR LASER PRINTER/FAX/COPIER TONER CARTRIDGE PRICES ...	4	1.0	Yes
I'm not sold on your first point ...	1	0.05	No
Some of the most beautiful women in the world ...	3	0.15	Yes

(1) Whether the comment mentions selling or nudity and (2) the proportion of capital letters here are the *features* we want to train a model to recognize when trying to recognize spam. The last column is our human-judged *label*.

## Example cont'd: Plotting our features



## Example cont'd: Plotting our features



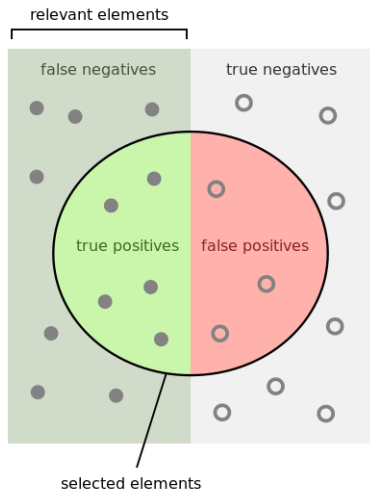
# Where has machine learning come up already?

- ▶ **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”

# What is 'supervised machine learning?': Introductory terminology

- ▶ **Supervised machine learning:** Building a model to infer classifications based on features. Learning a function that maps features to outputs.
- ▶ **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**
- ▶ **Labeled data:** is associated with outcome/result we are interested in, also called the 'target' (in CS) or the 'dependent variable' (social science). For ex: pos/neg sentiment, spam/not spam, voting yes/no,
- ▶ **Features/predictors:** what we use to predict the label, also called 'covariates' or 'independent variables' (in social science). For images, features are often pixels. For text, features could be word counts or weighted word counts, parts of speech, proportion of capitals, etc.
- ▶ **Text vectorization:** Converting natural language documents into numbers that can be fed into a classifier. Usually this means converting a corpus into a document-term matrix (which is?).

# Evaluating ML models: precision and recall



How many selected items are relevant?

$$\text{Precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}}$$

How many relevant items are selected?

$$\text{Recall} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}}$$

# Evaluating ML models: how to compute metrics

**Accuracy:**

$$\frac{\sum \text{true positives} + \sum \text{true negatives}}{\sum \text{total population}}$$

**Precision:**

$$\frac{\sum \text{true positives}}{\sum \text{predicted positives}}$$

**Recall::**

$$\frac{\sum \text{true positives}}{\sum \text{condition positives}}$$

**F1-score:**

$$\frac{\sum \text{true positives}}{\sum \text{true positives} + 1/2(\sum \text{false positives} + \sum \text{false negatives})}$$

# Evaluating ML: terminology

- ▶ **Train-test split:** Dividing labeled data into a training set and a test set. Often done once in supervised machine learning model training, but can be done several times independently as in cross-validation (see Wednesday)
- ▶ **Training set:** A selection of labeled data that is used to train the machine learning algorithm
- ▶ **Test set:** A selection of labeled data that is used to test the accuracy of the machine learning algorithm
- ▶ **Loss function:** Machine learning models seek to minimize this function, which measures model effectiveness at estimating the relationship between an input (features) and output variable/label. This is typically expressed as a difference between the predicted value and the actual value.
- ▶ **Accuracy:** The ratio of true positives and true negatives (the numerator) to the total number of cases (the denominator)
- ▶ **Precision:** How close a models' guesses are to the mark; the ratio of true positives to predicted positives
- ▶ **Recall:** How many actual positives a model guessed right; the ratio of true positives to condition positives



# Why do supervised ML?

- ▶ Once we have a good number of labeled texts (say 200-500 cases), can use supervised ML to train a model to predict the labels
- ▶ **Two benefits:**
  1. Identifies what features (e.g., words) are most defining of each category
  2. Can scale up almost indefinitely to predict labels for unobserved cases
- ▶ **Main goal:** *generalization*, or a model that predicts the response on novel inputs/new data it hasn't yet seen (e.g., new or future cases)
- ▶ **Two main types:**
  1. **Regression:** predict numeric values (not the focus this week)
  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 (focus this week)
    - ▶ **Multiclass** > 2 categories

# Problems with machine learning

Consider a traditional linear or logistic regression with two predictors trying to predict a sentiment label:

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

What would happen if we tried to scale this to a large corpus of text data? How many terms might we end up including? How might that number compare to the number of observations?

# Problems with machine learning

**The problem:** When a model gets overly complex (e.g., the number of terms modeled exceeds the number of observations), we risk **overfitting** because the model would pick up on language use or other idiosyncratic trends/noise particular to some observed cases but meaningless outside the observed data

For example, if you trained a linear model on reviews of fantasy movies, it might decide that the word 'hilarious' associates with negative reviews while the phrase 'hobbit\_feet' indicates a positive review.

Would these words still be meaningful if you used the model you just trained to make predictions on a *broader corpus of movies*?

# How to make classification better

- ▶ Two tools help us establish robust patterns/avoid overfitting:
  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
    - ▶ One way to do this: **Regularization** to prevent models from overfitting on training data/decrease their sensitivity to noise (by adding a penalty term for model complexity; read more [here from scikit-learn](#) or from [this blog](#))
    - ▶ Another way: Tree-based methods to learn non-linear relationships (we'll talk about this on Wednesday)
  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

# More about 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

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

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

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

# Goals for today

## Today: Introduction to supervised machine learning

- ▶ Recap of Twitter API
- ▶ Lecture on basics of supervised ML
- ▶ **Walkthrough of supervised ML in Python**
- ▶ Activity on basic ML

## Wednesday: Decision Trees and optimizing supervised ML models

- ▶ Recap of basic supervised ML
- ▶ Lecture on Decision Trees & optimization
- ▶ Activity on DTs & optimization

## Four DataCamp chapters

1. **Classification:** uses k-nearest neighbors; in activities, we'll shift to other classifiers (ridge; LASSO; Decision Tree; Random Forest)
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 importance of preprocessing for model accuracy
- Start with a model where the hyperparameters (e.g.,  $\alpha/\lambda$  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	<p>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.</p>
1	<p>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.</p>
0	<p>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</p>

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

# Notebook to follow along

09\_ML\_intro\_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	Say \"BITES.\" Say \"WINGS.\" Apparently, 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 boneless 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. I know I probably sound dramatic, but I will NOT be back...	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. On 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	<p>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!</p>	1
1	<p>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 often 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 afternoon 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</p>	1

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

```
1 ## define the number of rows in training set (20\% test)
2 ## and sample
3 nrows_train = round(X.shape[0]*0.8)
4 nrows_test = X.shape[0] - nrows_train
5 random.seed(221)
6 train_ids = random.sample(set(X['metadata_rowid']),
7                             nrows_train)
8 ## function fed ids and name of id col
9 def my_split(train_ids, id_col):
10     test_ids = set(X[id_col]).difference(train_ids)
11     X_train_man = X[X[id_col].isin(train_ids)].copy()
12     X_test_man = X[X[id_col].isin(test_ids)].copy()
13     y_train_man = y[y.index.isin(train_ids)].iloc[:,
14                                                         0].to_numpy()
15     y_test_man = y[y.index.isin(test_ids)].iloc[:,
16                                                         0].to_numpy()
17     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)

```
1 ## list of nonfeatures
2 non_feat = ['metadata_rowid', 'raw_text', 'process_text']
3
4 ## initialize classifier
5 dt = DecisionTreeClassifier(random_state=0, max_depth = 10)
6
7 ## estimate the model on training data/training label
8 dt.fit(X_train_man[[col for col in X_train.columns
9                     if col not in non_feat]],
10        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

```
1 y_pred = dt.predict(X_test_man[[col for col
2                               in X_test_man.columns if "metadata_rowid"
3                               not in col]])
4 y_predprob = dt.predict_proba(X_test_man[[col for col
5                                         in X_test_man.columns if "metadata_rowid"
6                                         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

```
1 ## dataframe of binary, continuous, and true labels
2 y_pred_df = pd.DataFrame({'y_pred_binary': y_pred,
3                             'y_pred_continuous':
4                                 [one_prob[1] for one_prob in y_predprob],
5                                 'y_true': y_test_man})
6
7 ## use np.select to code each obs to its error cat
8 error_cond = [ (y_pred_df['y_true'] == 1) &
9                 (y_pred_df['y_pred_binary'] == 1),
10                (y_pred_df['y_true'] == 1) &
11                (y_pred_df['y_pred_binary'] == 0),
12                (y_pred_df['y_true'] == 0) &
13                (y_pred_df['y_pred_binary'] == 0)]
14 error_codeto = ["TP", "FN", "TN"]
15
16 y_pred_df['error_cat'] = np.select(error_cond,
17                                     error_codeto, default = "FP")
18
19 ## see notebook for calculations using the error_cat column
```

## Step 6: interpret the model

```
1 feat_imp = pd.DataFrame({'feature_imp': dt.feature_importances_,  
2                           'feature_name':  
3                           [col for col in X_train.columns  
4                           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  $\alpha/\lambda$  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., predicting product pricing using (1) historical price data versus (2) tweet sentiment versus (3) both combined

# Goals for today

## **Today: Introduction to supervised machine learning**

- ▶ Recap of Twitter API
- ▶ Lecture on basics of supervised ML
- ▶ Walkthrough of supervised ML in Python
- ▶ **Activity on basic ML**

## Wednesday: Decision Trees and optimizing supervised ML models

- ▶ Recap of basic supervised ML
- ▶ Lecture on Decision Trees & optimization
- ▶ Activity on DTs & optimization

# Break for activity (section 4 in notebook)

Notebook:

[https://github.com/jhaber-zz/QSS20\\_public/blob/main/activities/09\\_binaryclassification\\_activity\\_blank.ipynb](https://github.com/jhaber-zz/QSS20_public/blob/main/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](https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html)
2. Fit the model using `X_train_main`, `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:

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

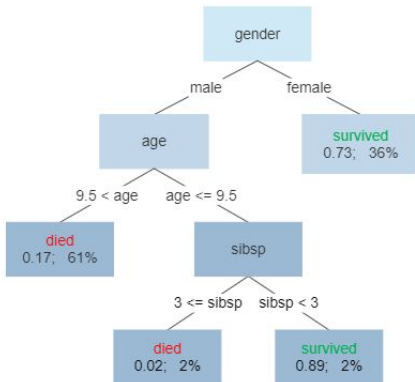
5. Apply that function to calculate the F1 score for the decision tree and lasso (from above), and ridge regression (from the activity)
6. **Challenge exercise:** parametrize the model fitting with a function that takes in a classifier as an argument and returns coefficients or feature importances and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

# QSS20: Modern Statistical Computing

## Unit 11: Supervised Machine Learning, Part II (Decision Trees & Optimization)

# Tree-based methods to learn non-linear relationships

Survival of passengers on the Titanic



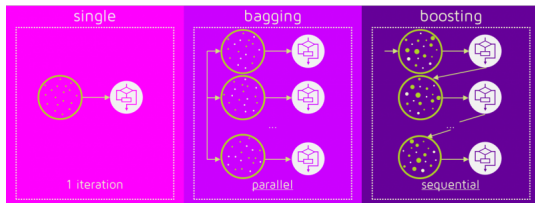


# Logic behind classification trees/random forests

- ▶ **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 ( $\tau$ )
  - ▶ **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

# Optimizing ML: terminology

- ▶ **overfitting:** When a model is trained on too little data to generalize to new data, usually when trained and tested on the same sample. In this case, the model just repeats the labels of the samples that it has just seen, and is evaluated to have a perfect score—but it fails to predict anything useful on yet-unseen data.
- ▶ **underfitting:** When a model fails to adequately capture the underlying structure of the data, usually when trained on too little data
- ▶ **cross-validation:** A way to assess the performance of an algorithm on an unseen data set. Essentially this repeats a train-test split several times and averages the result of these independent slices, giving a superior estimation of model accuracy compared to a single train-test split.
- ▶ **parameter:** An algorithm setting that may be selected for performance or substantive reasons. If a parameter is not learned directly within an estimator but instead must be set explicitly (passed as an argument), then it's called a *hyper-parameter*.
- ▶ **optimization:** Selecting the best element from some set of alternatives such that the result maximizes some criterion (e.g., model accuracy)
- ▶ **grid-search:** An exhaustive search over model parameters to discover the optimal configuration, maximizing model performance

# Optimization

(Slides in progress)