,

Introduction to Supervised Machine Learning (SML)

Rebecca Johnson, Assistant Professor, McCourt School of Public Policy (teach in MS in Data Science for Public Policy);
www.rebeccajohnson.io

Penn Methodology Working Group: 02.13.2023

Plan for session

- **Broad overview/terminology**
- Supervised machine learning: types of classifiers and example sociological applications
- Supervised machine learning: how to using `scikit-learn` in Python

Machine learning and computational social science

- **Sentiment classification:** `VADER` in Python and other popular packages score 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:** latent dirichlet allocation (LDA) or structural topic models (STM) are forms of unsupervised machine learning that take an unlabeled input—a document-term matrix—and returns high probability topics and words
- **Probabilistic record linkage:** many packages use 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 variables (e.g., name; address) and we cluster the pairs into "likely match" and "likely not match"

Broad reference

Machine Learning for Sociology

Annual Review of Sociology

Vol. 45:27-45 (Volume publication date July 2019)
First published as a Review in Advance on May 13, 2019
https://doi.org/10.1146/annurev-soc-073117-041106

Mario Molina and Filiz Garip

Department of Sociology, Cornell University, Ithaca, New York 14853, USA; email: mm2535@cornell.edu, fgarip@cornell.edu

Full Text HTML | Download PDF | Article Metrics | Permissions | Reprints | Download Citation | Citation Alerts

Sections

ABSTRACT
KEYWORDS
INTRODUCTION
SUPERVISED MACHINE LEARNING
SUPERVISED MACHINE LEARNING FOR POLICY PREDICTIONS, CAUSAL INFERENCE, AND DATA AUGMENTATION
UNSUPERVISED MACHINE LEARNING
MACHINE LEARNING: NEW

Abstract

Machine learning is a field at the intersection of statistics and computer science that uses algorithms to extract information and knowledge from data. Its applications increasingly find their way into economics, political science, and sociology. We offer a brief introduction to this vast toolbox and illustrate its current uses in the social sciences, including distilling measures from new data sources, such as text and images; characterizing population heterogeneity; improving causal inference; and offering predictions to aid policy decisions and theory development. We argue that, in addition to serving similar purposes in sociology, machine learning tools can speak to long-standing questions on the limitations of the linear modeling framework, the criteria for evaluating empirical findings, transparency around the context of discovery, and the epistemological core of the discipline.

Keywords

supervised learning, unsupervised learning, causal inference, prediction, heterogeneity, discovery

Molina and Garip, Machine Learning for Sociology, *Annual Review of Sociology*; shorthand MG

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 (SML):**
  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

Where we are

- Broad overview/terminology
- **Supervised machine learning: types of classifiers and example sociological applications**
- Supervised machine learning: how to using `scikit-learn` in Python

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 — as Molina + Garip note, we can also do three splits (more on that later)

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

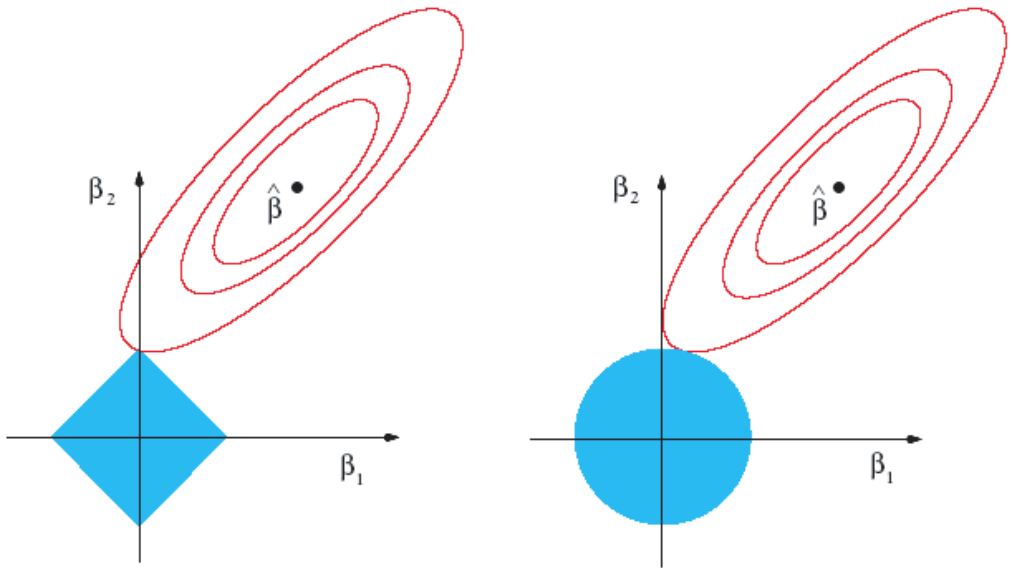
$$\hat{\beta}_{Lasso} = \sum_{i=1}^{n}(y_i - \sum_{j=1}^{p} x_{ij}\hat{\beta})^2 + \lambda \sum_{j=1}^{p} |\hat{\beta}_j|$$

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

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

Building intuition: L_1 versus L_2 regularization

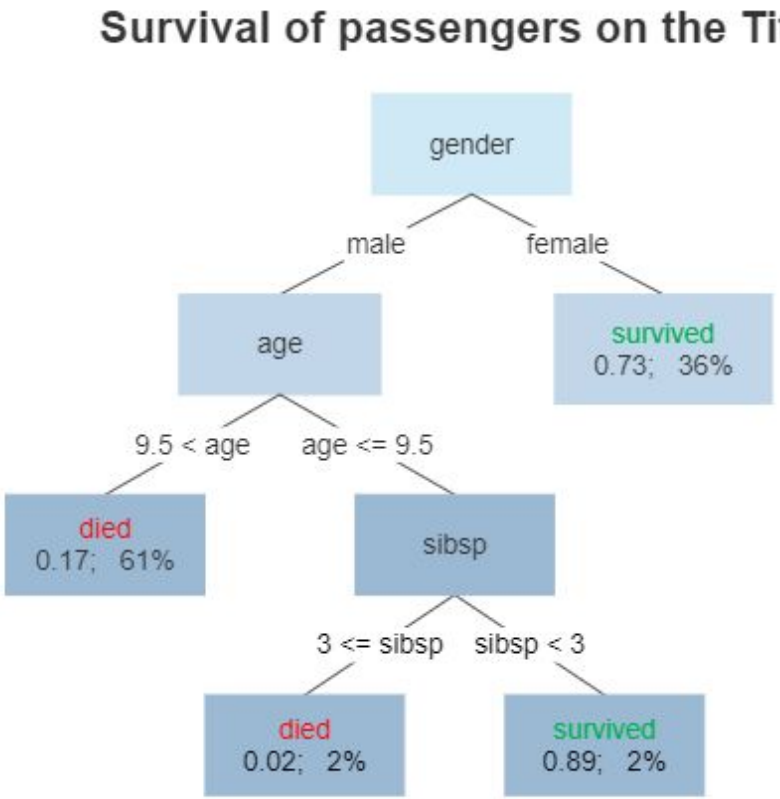
Visual that helps some people: L_1 (left; sharp quadratic) versus L_2 (right; smooth); source



What helps me more: estimate both on same data; merge feature importances at the attribute level (eg β on continuous household income for each data); for same attribute, see pattern of regularization towards $\beta = 0$ from lasso (sparsity) versus regularization towards non-zero small β

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



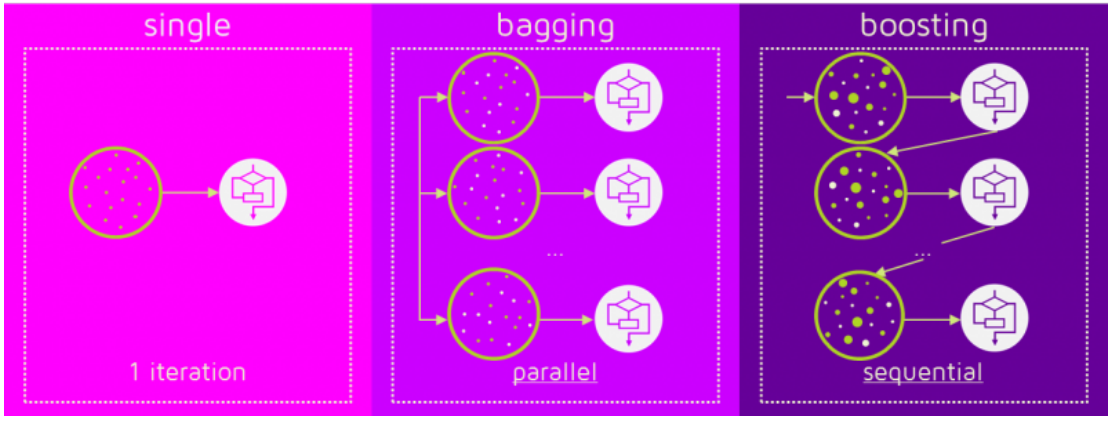
Graphics source

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

Across classifiers: sample splitting

- **More rigorous approach:** three-part split
  1. **Training data:** used to estimate the model, often iterating through different classifiers, and within each classifier, different hyperparameters (next slide)
  2. **Validation data:** used to evaluate accuracy for a particular constellation of training data + classifier + hyperparameters for that classifier; select model that scores best on your choice of accuracy statistic
  3. **Test or hold-out data:** data "kept in the vault" (MG) — used to calculate the "real"* generalization error of the selected model
- **Other approaches:**
  - **Training data + validation data** only (tradeoffs bc want as high n as possible to estimate the model)
  - **k-fold cross validation (CV):** rather than "pure" partition, k-fold CV recycles each observation so it serves as part of the validation set in one fold; part of the training data in $k-1$ folds

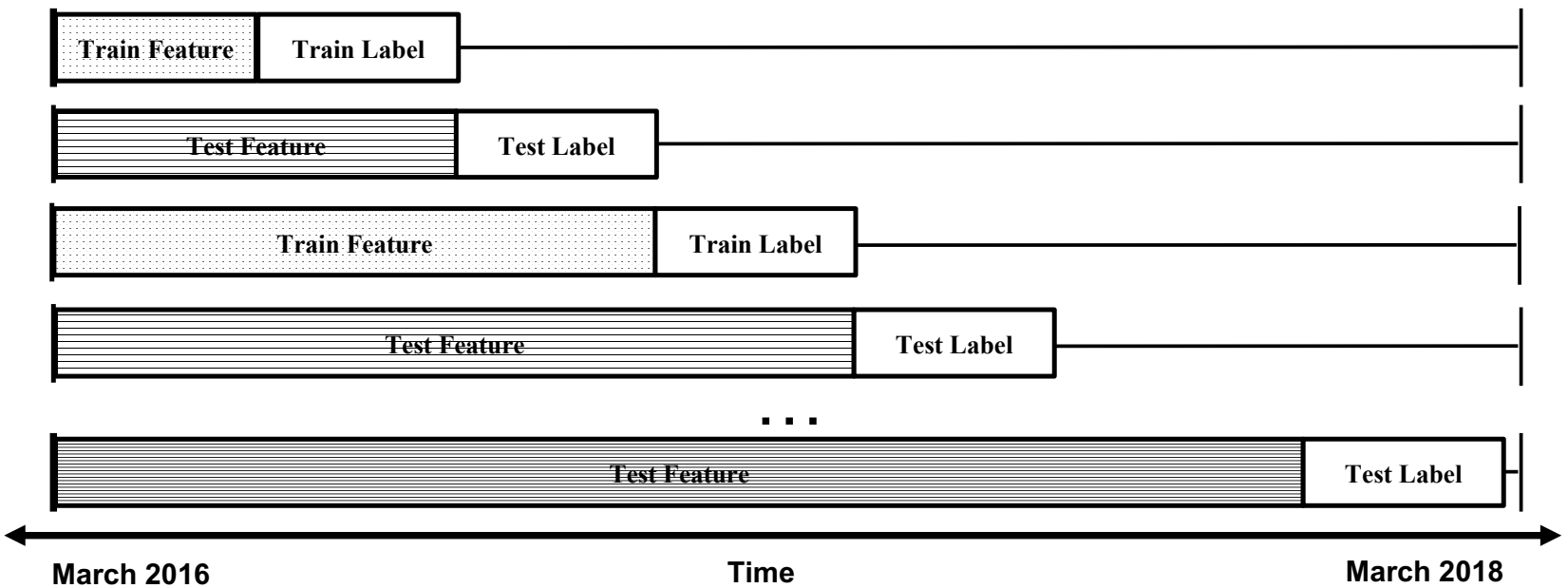
*More on model decay later...

Toy example: 3-fold CV with 6 observations

person	**fold 1**	**fold 2**	**fold 3**
p1	Training	Validation	Training
p2	Training	Validation	Training
p3	Training	Training	Validation
p4	Validation	Training	Training
p5	Training	Training	Validation
p6	Validation	Training	Training
⋮			

Especially relevant for social scientists: temporal (time series) cross-validation

Visual illustration:



First challenge: leakage in the model-building phase

- **Rough definition of leakage**: inadvertently using information from the future to predict the past
- **Examples:**
  - One of your predictors is a highly-correlated proxy for the outcome (e.g., Y = yes or no apply to Penn; one feature is a Penn-specific expected family contribution field that is only calculated/observed for students who apply)
  - Use mean (or model-based) imputation to impute missing values; use the full sample for that imputation rather than only observations in the "pre-label" time window; leakage since information from future time period appears in the imputed values for present time period
- **Resources:**
  - Ghani et al: blog post
  - Kapoor and Narayanan: Leakge and the Reproducibility Crisis in ML-based Science

Second challenge: model drift at deployment phase

- **Model drift—decay in predictive performance over time—as underlying social world changes:**
  - **Example:** Chicago Public Schools trains a model on rising seniors (17-19 year olds from particular birth cohort; seniors in fall of 2006) to predict who applies to college
  - Wants to then use the same model for several other classes of juniors/seniors to offer extra supports
- **Different types of drift (non exhaustive):**
  - **Concept drift:** change in conditional probability $P(Y|X)$
    - **Example:** City of Chicago newly funds means-tested free community college; changes relationship between family income (one feature) and the label
  - **Data or feature drift:** input features change — $p(X)$ changes but $P(Y|X)$ stays fixed
    - **Example:** K-12 schools newly adopt Community Eligibility Provision so meaning of "Yes receiving Free or Reduced Price Lunch (FRPL)" changes
  - **Label drift:** change in probability of Y without change in conditional relationship – difficult for me to think of examples!

Across classifiers: tuning hyperparameters

- Different classifiers have different *available hyperparameters* that will lead to different accuracy in the validation set; how do we select combinations of hyperparameters to iterate over/compare results for?
- **Grid search:** define a series of discrete hyperparameter values; test all combinations; .py with different grids

```
small_grid = {
'RF':{'n_estimators': [100, 10000], 'max_depth': [5,50], 'max_features': ['sqrt','log2'],
'LR': { 'penalty': ['l1','l2'], 'C': [0.00001,0.001,0.1,1,10]},
'SGD': { 'loss': ['hinge','log','perceptron'], 'penalty': ['l2','l1','elasticnet']},
'ET': { 'n_estimators': [100, 10000], 'criterion' : ['gini', 'entropy'] ,'max_depth': [5,
'AB': { 'algorithm': ['SAMME', 'SAMME.R'], 'n_estimators': [1,10,100,1000,10000]},
'GB': {'n_estimators': [100, 10000], 'learning_rate' : [0.001,0.1,0.5],'subsample' : [0.1
'NB' : {},
'DT': {'criterion': ['gini', 'entropy'], 'max_depth': [1,5,10,20,50,100], 'max_features':
'SVM' :{'C' :[0.00001,0.0001,0.001,0.01,0.1,1,10],'kernel':['linear']},
'KNN' :{'n_neighbors': [1,5,10,25,50,100],'weights': ['uniform','distance'],'algorithm':
      }
```

- **Random search:** define bounds for hyperparameter ranges to search within; randomly sample — e.g, instead of max tree depth 5 and 50, would define range of 5-50 and sample within

Across classifiers: evaluating model accuracy

- Example output in validation set

person	$\hat{y}_{continuous}$	$\hat{y}_{binary}$	y	Category
p1	0.74	1	1	TP
p2	0.24	0	1	FN
p3	0.57	1	0	FP
p4	0.08	0	0	TN

- For binary classifiers, widely-used metrics include precision (more weight on minimizing false positives), recall (more weight on minimizing false negatives), and F1 score (harmonic mean of two that balances the two goals); see code for formulas
- Key role for social scientists: context shapes preferred metric:

  "For the purposes of our field experiment, we selected the best model using `recall`. Our rationale is that, for the purpose of targeting incentives, we want to focus more on minimization of false negatives—respondents we fail to provide incentives to but who might be moved by that incentive to respond—than on wasted incentives on false positives—units that would have responded anyways."

What about neural networks and other tools within "deep learning"?

- Most useful for very unstructured high-dimensional data (e.g., image data; audio data)
- From Prof. Han Zhang's presentation on Deep Learning/Convolutional Neural Networks
  - **Previous classifiers:** explicitly engineer features that you then use for predicting the label (e.g., in student data, might have FRPL status Y/N; race/ethnicity encoded as binary variables; and counts of absences over different historical time ranges)
  - **Representation learning:** rather than us "hand" engineering features, model learns a representation of the features (e.g., in the case of images, a tensor (stacked matrices) representing Red-Green-Blue values at specific pixels) predictive of the label
- Implementation via off-the-shelf Python packages: enjoyed workbook from session last fall covering TensorFlow and Keras; syntax similar to `scikit-learn`

Using the MG typology, (biased)‡ set of published examples

1. **Policy predictions:**†
   - Ye, **Rebecca Johnson**...Ghani. "Using Machine Learning to Help Tenants in New York City"
   - **Han Zhang** and Pan. "CASM: A Deep-Learning Approach for Identifying Collective Action Events with Text and Image Data from Social Media"
2. **Causal inference: Ian Lundberg**. "The Gap-Closing Estimand: A Causal Approach to Study Interventions That Close Disparities Across Social Categories"; can use classifiers like ridge regression or random forest as the estimation approach for the treatment algorithm
3. **Data augmentation and imputation:** Bucca and **Daniela Urbina**. "Lasso Regularization for Selection of Log-linear Models: An Application to Educational Assortative Mating"

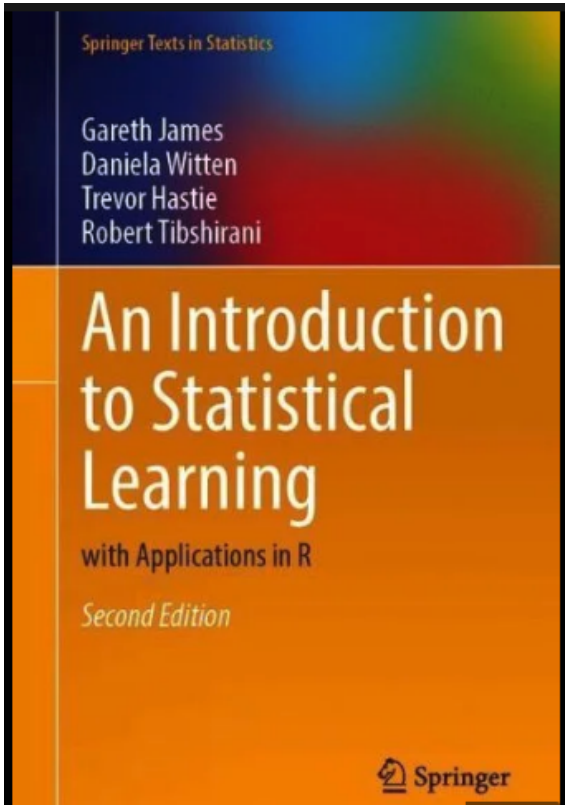
---

†Though I'd argue that the estimand policymakers care about is conditional average treatment effects from a policy intervention (e.g., a housing voucher) rather than $\hat{Y}$

‡Participants in similar reading group at Princeton: `https://bstewart.scholar.princeton.edu/sociology-statistics-reading-group`

What if I hate Python...can I do this in R?

Yes! Good resource:



More generally, packages: `tidymodels` (meant to provide more of a unifying framework across classifiers); `ranger` for random forests; `glmnet` for lasso/ridge; `gbm` for gradient boosting; and so on

Where we are

- Broad overview/terminology
- Supervised machine learning: types of classifiers and example sociological applications
- **Supervised machine learning: how to using `scikit-learn` in Python**

Working example of binary classification

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

	metadata_label	raw_text
0	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	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.
2	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

Repo: https://github.com/rebeccajohnson88/pennmethods_wg_SML

`code/`

- Example code; no activity solutions: 00_binaryclassification_activity_blank.ipynb
- Example code + activity solutions: 01_binaryclassification_activity_solutions.ipynb

`data/`

- Raw labeled reviews: `yelp_forML.pkl`
- Preprocessed reviews: `yelp_forML_preprocessed.csv`

Step 1: preprocess the data

See old teaching slides here for process of document-term matrix creation

Data exploration: terms versus negative label

Uses word “frustrat” and negative label:

metadata_label	raw_text	frustrat
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
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

Data exploration: 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)

```
## create list of columns present in the training matrix
## that are not actually features
non_feat = ['metadata_rowid', 'raw_text', 'process_text']

## initialize the classifier - this is step where we feed it
    hyperparameters
dt = DecisionTreeClassifier(random_state=0, max_depth = 10)

## use the initialized classifier to fit the model
X_features = X_train_man[[col for col in X_train_man.columns
            if col not in non_feat]]
dt.fit(X_features, 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

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**

Activity (section 4 in notebook)

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 the same training features and label
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)