Sequencing Legal DNA

NLP for Law and Political Economy

4. Supervised Learning with Text

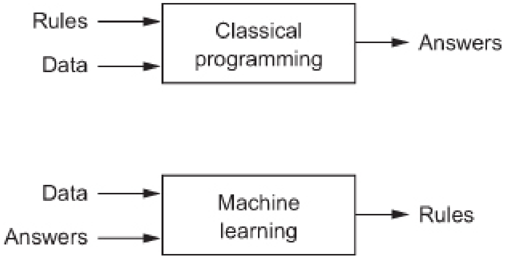
Reminder: Assignment 1 Due by Wednesday

- Two reading response essays, one code replication exercise, a problem set solution, or the hybrid assignment (one essay and answers to four problem set questions).
- Upload to assignment dropbox by 23:59 on Wednesday.

Reminder: Assignment 1 Due by Wednesday

- Two reading response essays, one code replication exercise, a problem set solution, or the hybrid assignment (one essay and answers to four problem set questions).
- Upload to assignment dropbox by 23:59 on Wednesday.
- Essays will be posted (anonymously) on the forum:
  - please vote on some of your classmates' essays, and comment on at least one (before next week's class).

What is machine learning?



- In classical computer programming, humans input the rules and the data, and the computer provides answers.
- In machine learning, humans input the data and the answers, and the computer learns the rules.

Machine Learning with Text Data

- We have a corpus (or dataset) D of $n_D \geq 1$ documents (or data points), whose features can be represented as a matrix of vectors $\boldsymbol{x}$ with $n_x \geq 1$ features.

Machine Learning with Text Data

- We have a corpus (or dataset) D of $n_D \geq 1$ documents (or data points), whose features can be represented as a matrix of vectors $\boldsymbol{x}$ with $n_x \geq 1$ features.
- Each document has an associated outcome or label $\boldsymbol{y}$ with dimensions $n_y \geq 1$

Machine Learning with Text Data

- We have a corpus (or dataset) D of $n_D \geq 1$ documents (or data points), whose features can be represented as a matrix of vectors $\boldsymbol{x}$ with $n_x \geq 1$ features.
- Each document has an associated outcome or label $\boldsymbol{y}$ with dimensions $n_y \geq 1$
- Some documents are unlabeled $\rightarrow$ we would like to train a model to machine-classify them.

Econometrics

- y is one-dimensional ($n_y = 1$), x is low-dimensional ($n_x \ll n_D$).

Econometrics

- y is one-dimensional ($n_y = 1$), x is low-dimensional ($n_x \ll n_D$).

Machine learning

- $\boldsymbol{y}$ can be multi-dimensional ($n_y \geq 1$), $\boldsymbol{x}$ can be high-dimensional ($n_x \gtrless n_D$).

Econometrics

- y is one-dimensional ($n_y = 1$), x is low-dimensional ($n_x \ll n_D$).
- estimate a low-dimensional **causal parameter** ρ using

$$y_i = \alpha_i + x_i \cdot \rho + \epsilon_i$$

where i indexes over documents, α_i includes control variables (and fixed effects), $\cdot$ is dot product, and ϵ_i is the error residual.

Machine learning

- $\boldsymbol{y}$ can be multi-dimensional ($n_y \geq 1$), $\boldsymbol{x}$ can be high-dimensional ($n_x \gtrless n_D$).

Econometrics

- y is one-dimensional ($n_y = 1$), x is low-dimensional ($n_x \ll n_D$).
- estimate a low-dimensional **causal parameter** ρ using

$$y_i = \alpha_i + x_i \cdot \rho + \epsilon_i$$

where i indexes over documents, α_i includes control variables (and fixed effects), $\cdot$ is dot product, and ϵ_i is the error residual.

Machine learning

- $\boldsymbol{y}$ can be multi-dimensional ($n_y \geq 1$), $\boldsymbol{x}$ can be high-dimensional ($n_x \gtrless n_D$).
- learn a high-dimensional vector (or matrix) of parameters θ to approximate a (potentially non-linear) function

$$\boldsymbol{y}_i = f(\boldsymbol{x}_i; \theta)$$

that predicts $\boldsymbol{y}$ given covariates $\boldsymbol{x}$.

Econometrics

- y is one-dimensional ($n_y = 1$), x is low-dimensional ($n_x \ll n_D$).
- estimate a low-dimensional **causal parameter** ρ using

  $$y_i = \alpha_i + x_i \cdot \rho + \epsilon_i$$

  where i indexes over documents, α_i includes control variables (and fixed effects), $\cdot$ is dot product, and ϵ_i is the error residual.
- ρ gives a prediction how outcome y would change if treatment variable x were **exogenously shifted**.
  - useful for policy evaluation.

Machine learning

- $\boldsymbol{y}$ can be multi-dimensional ($n_y \geq 1$), $\boldsymbol{x}$ can be high-dimensional ($n_x \gtrsim n_D$).
- learn a high-dimensional vector (or matrix) of parameters θ to approximate a (potentially non-linear) function

  $$\boldsymbol{y}_i = f(\boldsymbol{x}_i; \theta)$$

  that predicts $\boldsymbol{y}$ given covariates $\boldsymbol{x}$.

Econometrics

- y is one-dimensional ($n_y = 1$), x is low-dimensional ($n_x \ll n_D$).
- estimate a low-dimensional **causal parameter** ρ using

$$y_i = \alpha_i + x_i \cdot \rho + \epsilon_i$$

  where i indexes over documents, α_i includes control variables (and fixed effects), $\cdot$ is dot product, and ϵ_i is the error residual.
- ρ gives a prediction how outcome y would change if treatment variable x were **exogenously shifted**.
  - useful for policy evaluation.

Machine learning

- $\boldsymbol{y}$ can be multi-dimensional ($n_y \geq 1$), $\boldsymbol{x}$ can be high-dimensional ($n_x \gtrsim n_D$).
- learn a high-dimensional vector (or matrix) of parameters θ to approximate a (potentially non-linear) function

$$\boldsymbol{y}_i = f(\boldsymbol{x}_i; \theta)$$

  that predicts $\boldsymbol{y}$ given covariates $\boldsymbol{x}$.
- if we collected more data on $\boldsymbol{x}$, we could predict the associated $\hat{y}$.
- but $h(\cdot)$ does not provide a *counterfactual prediction* – that is, how the outcome would change if $\boldsymbol{x}$'s were exogenously shifted.

Objectives

1. **What is the research question?**

Objectives

1. **What is the research question?**
2. Corpus and Data:
   - obtain, clean, preprocess, and link.
   - Produce descriptive visuals and statistics on the text and metadata

Objectives

1. **What is the research question?**
2. Corpus and Data:
   - obtain, clean, preprocess, and link.
   - Produce descriptive visuals and statistics on the text and metadata
3. Machine learning:
   - Select a model and train it.
   - Fine-tune hyperparameters for out-of-sample fit.
   - Interpret predictions using model explanation methods.

Objectives

1. **What is the research question?**
2. Corpus and Data:
   - obtain, clean, preprocess, and link.
   - Produce descriptive visuals and statistics on the text and metadata
3. Machine learning:
   - Select a model and train it.
   - Fine-tune hyperparameters for out-of-sample fit.
   - Interpret predictions using model explanation methods.
4. Empirical analysis
   - Produce statistics or predictions with the trained model.
   - **Answer the research question.**

What do ML Algorithms do? Minimize a cost function

- A typical cost function for regression problems is Mean Squared Error (MSE):

$$\mathrm{MSE}(x, f) = \frac{1}{n_D} \sum_{i=1}^{n_D} (f(x_i; \theta) - y_i)^2$$

  - n_D, the number of rows/observations
  - x, the feature set, with row x_i
  - y, the set of outcomes, with item y_i
  - $f(x_i; \theta) = \hat{y}$ the model prediction (hypothesis)

Loss functions, more generally

- The loss function $L(\hat{\boldsymbol{y}}, \boldsymbol{y})$ assigns a score based on prediction and truth:
  - Should be bounded from below, with the minimum attained only for cases where the prediction is correct.
- The corpus-wide average loss is

$$\mathcal{L}(\theta) = \frac{1}{n_D} \sum_{i=1}^{n_D} L(f(\boldsymbol{x}_i; \theta), \boldsymbol{y}_i)$$

  - $\rightarrow$ treats the data as constants; parameters determine the loss.
- The estimated parameter matrix θ solves

$$\hat{\theta} = \arg\min_{\theta} \mathcal{L}(\theta)$$

OLS Regression is Machine Learning

- OLS assumes the functional form $f(x;\theta) = x_i'\theta$ and minimizes the MSE

$$\min_{\hat{\theta}} \frac{1}{n_D} \sum_{i=1}^{n_D} (x_i'\hat{\theta} - y_i)^2$$

OLS Regression is Machine Learning

- OLS assumes the functional form $f(x;\theta) = x_i'\theta$ and minimizes the MSE

$$\min_{\hat{\theta}} \frac{1}{n_D} \sum_{i=1}^{n_D} (x_i'\hat{\theta} - y_i)^2$$

- This minimand has a closed form solution

$$\hat{\theta} = (\boldsymbol{x}'\boldsymbol{x})^{-1}\boldsymbol{x}'\boldsymbol{y}$$

- But most machine learning models do **not** have a closed form solution.

How do ML Algorithms Work? Gradient Descent

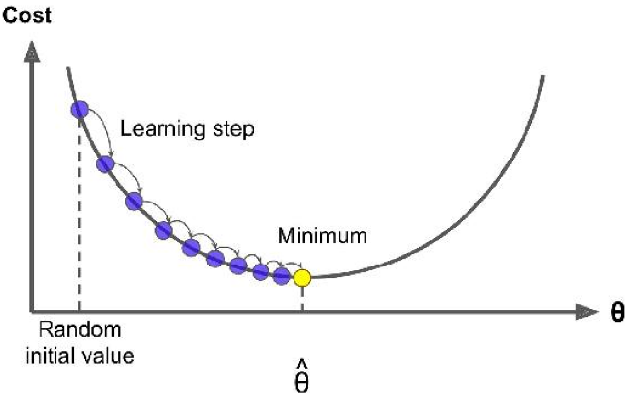


Figure 4-3. Gradient Descent

- Gradient descent measures the local gradient of the error function, and then steps in that direction.
  - Once the gradient equals zero, you have reached a minimum.

$$\text{MSE}(\theta) = \frac{1}{n_D} \sum_{i=1}^{n_D} (\boldsymbol{x}_i' \hat{\theta} - y_i)^2$$

- The partial derivative of MSE for feature j is

$$\frac{\partial \text{MSE}}{\partial \theta_j} = \frac{2}{m} \sum_{i=1}^{n} (\boldsymbol{x}_i' \hat{\theta} - y_i) x_i^j$$

  - $\rightarrow$ estimates the amount that changing θ_j would reduce the error.

$$\mathrm{MSE}(\theta) = \frac{1}{n_D} \sum_{i=1}^{n_D} (\boldsymbol{x}_i' \hat{\theta} - y_i)^2$$

- The partial derivative of MSE for feature j is

$$\frac{\partial \mathrm{MSE}}{\partial \theta_j} = \frac{2}{m} \sum_{i=1}^{n} (\boldsymbol{x}_i' \hat{\theta} - y_i) x_i^j$$

  - $\rightarrow$ estimates the amount that changing θ_j would reduce the error.
- The *gradient* ∇ gives the vector of these partial derivatives:

$$\nabla_\theta \mathrm{MSE} = \begin{bmatrix} \frac{\partial \mathrm{MSE}}{\partial \theta_0} \\ \frac{\partial \mathrm{MSE}}{\partial \theta_0} \\ \vdots \\ \frac{\partial \mathrm{MSE}}{\partial \theta_j} \end{bmatrix} = \frac{2}{m} \boldsymbol{X}' (\boldsymbol{X}' \theta - \boldsymbol{y})$$

$$\text{MSE}(\theta) = \frac{1}{n_D}\sum_{i=1}^{n_D}(\boldsymbol{x}_i'\hat{\theta} - y_i)^2$$

- The partial derivative of MSE for feature j is

$$\frac{\partial \text{MSE}}{\partial \theta_j} = \frac{2}{m}\sum_{i=1}^{n}(\boldsymbol{x}_i'\hat{\theta} - y_i)x_i^j$$

  - $\rightarrow$ estimates the amount that changing θ_j would reduce the error.
- The *gradient* ∇ gives the vector of these partial derivatives:

$$\nabla_\theta \text{MSE} = \begin{bmatrix} \frac{\partial \text{MSE}}{\partial \theta_0} \\ \frac{\partial \text{MSE}}{\partial \theta_0} \\ \vdots \\ \frac{\partial \text{MSE}}{\partial \theta_j} \end{bmatrix} = \frac{2}{m}\boldsymbol{X}'(\boldsymbol{X}'\theta - \boldsymbol{y})$$

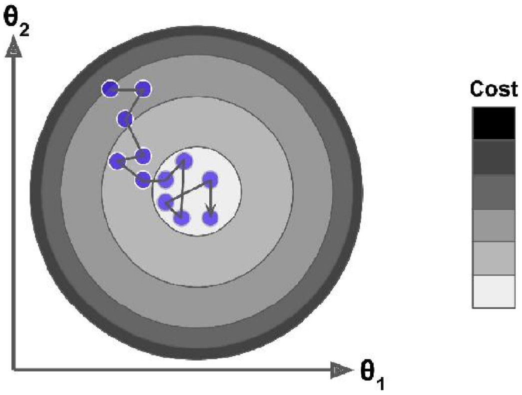
- Gradient descent nudges θ against the gradient (the direction that reduces MSE):

$$\theta_{t+1} = \theta_t - \eta\nabla_\theta \text{MSE}$$

  - η = learning rate

Stochastic Gradient Descent

- SGD picks a random instance in the training set and computes the gradient only for that single instance.



- Much faster to train, but bounces around even after it is close to the minimum.
  - Compromise: mini-batch gradient descent, selects a sample of rows (a "mini-batch") for gradient compute.

Training and Testing

- Machine learning models can achieve arbitrarily high accuracy in-sample.
  - performance should be evaluated out-of-sample.

Training and Testing

- Machine learning models can achieve arbitrarily high accuracy in-sample.
  - performance should be evaluated out-of-sample.
- standard approach:
  - randomly sample 80% training dataset to learn parameters
  - form predictions in 20% testing dataset for evaluating performance.

Data Prep for Machine Learning

- See Geron Chapter 2 for pandas and sklearn syntax:
  - imputing missing values.
  - feature scaling (often helpful/necessary for ML models to work well)
  - encoding categorical variables.

Data Prep for Machine Learning

- See Geron Chapter 2 for pandas and sklearn syntax:
  - imputing missing values.
  - feature scaling (often helpful/necessary for ML models to work well)
  - encoding categorical variables.
- Best practice:
  - **reproducible data pipeline**.

Data Prep for Machine Learning

- See Geron Chapter 2 for pandas and sklearn syntax:
  - imputing missing values.
  - feature scaling (often helpful/necessary for ML models to work well)
  - encoding categorical variables.
- Best practice:
  - **reproducible data pipeline**.
  - if you want a "clean" evaluation in the test set, you have to **do data prep in the training set**.

Scikit-Learn Design Overview

- *Estimator:* An object that can estimate parameters.
  - Estimation is performed by `fit()` method.
  - Exogenous parameters (provided by the researcher) are called *hyperparameters*.

Scikit-Learn Design Overview

- *Estimator:* An object that can estimate parameters.
  - Estimation is performed by `fit()` method.
  - Exogenous parameters (provided by the researcher) are called *hyperparameters*.
- *Transformer*: An object that transforms a data set.
  - E.g. **TfidfVectorizer** to get tf-idf frequencies from documents
  - Transformation is performed by the `transform()` method.
  - The convenience method `fit_transform()` both fits an estimator and returns the transformed input data set.

Scikit-Learn Design Overview

- *Estimator:* An object that can estimate parameters.
  - Estimation is performed by **`fit()`** method.
  - Exogenous parameters (provided by the researcher) are called *hyperparameters*.
- *Transformer*: An object that transforms a data set.
  - E.g. **TfidfVectorizer** to get tf-idf frequencies from documents
  - Transformation is performed by the **`transform()`** method.
  - The convenience method **`fit_transform()`** both fits an estimator and returns the transformed input data set.
- *Predictor*: An object that forms a prediction from an input data set.
  - The **`predict()`** method forms the predictions.
  - It also has a **`score()`** method that measures the quality of the predictions given a test set.

Scikit-Learn Design Overview

- *Estimator:* An object that can estimate parameters.
  - Estimation is performed by **`fit()`** method.
  - Exogenous parameters (provided by the researcher) are called *hyperparameters*.
- *Transformer*: An object that transforms a data set.
  - E.g. **TfidfVectorizer** to get tf-idf frequencies from documents
  - Transformation is performed by the **`transform()`** method.
  - The convenience method **`fit_transform()`** both fits an estimator and returns the transformed input data set.
- *Predictor*: An object that forms a prediction from an input data set.
  - The **`predict()`** method forms the predictions.
  - It also has a **`score()`** method that measures the quality of the predictions given a test set.
- **Miscellaneous:**
  - **Inspection:** Hyperparameters and parameters are accessible. Learned parameters have an underscore suffix (e.g. `lin_reg.coef_`)

Scikit-Learn Design Overview

- *Estimator:* An object that can estimate parameters.
  - Estimation is performed by **`fit()`** method.
  - Exogenous parameters (provided by the researcher) are called *hyperparameters*.
- *Transformer*: An object that transforms a data set.
  - E.g. **TfidfVectorizer** to get tf-idf frequencies from documents
  - Transformation is performed by the **`transform()`** method.
  - The convenience method **`fit_transform()`** both fits an estimator and returns the transformed input data set.
- *Predictor*: An object that forms a prediction from an input data set.
  - The **`predict()`** method forms the predictions.
  - It also has a **`score()`** method that measures the quality of the predictions given a test set.
- **Miscellaneous:**
  - **Inspection:** Hyperparameters and parameters are accessible. Learned parameters have an underscore suffix (e.g. `lin_reg.coef_`)
  - **Non-proliferation of classes:** Use native Python data types; existing building blocks are used as much as possible.

Scikit-Learn Design Overview

- *Estimator:* An object that can estimate parameters.
  - Estimation is performed by `fit()` method.
  - Exogenous parameters (provided by the researcher) are called *hyperparameters*.
- *Transformer*: An object that transforms a data set.
  - E.g. **TfidfVectorizer** to get tf-idf frequencies from documents
  - Transformation is performed by the `transform()` method.
  - The convenience method `fit_transform()` both fits an estimator and returns the transformed input data set.
- *Predictor*: An object that forms a prediction from an input data set.
  - The `predict()` method forms the predictions.
  - It also has a `score()` method that measures the quality of the predictions given a test set.
- **Miscellaneous:**
  - **Inspection:** Hyperparameters and parameters are accessible. Learned parameters have an underscore suffix (e.g. `lin_reg.coef_`)
  - **Non-proliferation of classes:** Use native Python data types; existing building blocks are used as much as possible.
  - **Sensible defaults:** Provides reasonable default values for hyperparameters – easy to get a good baseline up and running.

Use Cross-Validation During Model Training

- Within the training set:
  - Use `cross_val_score` method to get model performance across subsets of data.
  - Use `GridSearchCV` or `RandomizedSearchCV to automate search over parameter space.`
- Find the best hyperparameters for out-of-fold prediction in the training set.
  - then evaluate model performance in the test set.

Regression models ↔ Continuous outcome

- If the outcome is continuous (e.g., Y = tax revenues collected, or criminal sentence imposed in months of prison):

- Need a regression model. Problems with OLS:
  - tends to over-fit training data.
  - cannot handle multicollinearity.

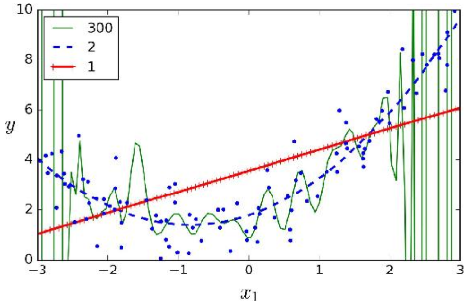


Figure 4-14. High-degree Polynomial Regression

Regression models $\leftrightarrow$ Continuous outcome

- If the outcome is continuous (e.g., Y = tax revenues collected, or criminal sentence imposed in months of prison):

- Need a regression model. Problems with OLS:
  - tends to over-fit training data.
  - cannot handle multicollinearity.

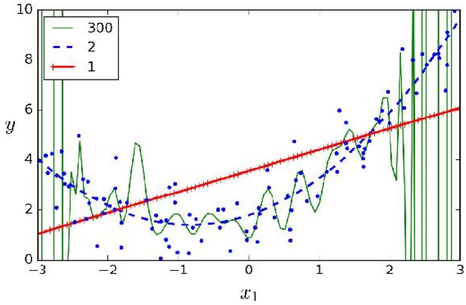


Figure 4-14. High-degree Polynomial Regression

- Machine learning models are evaluated by the fit in held out data (the test set)
  - "Regularization" refers to ML model training methods designed to reduce/prevent over-fitting of the training set
  - (and hopefully better fit in the test set).

Regularization

- Minimizing the loss L directly usually results in over-fitting. It is standard to add regularization:

$$\hat{\boldsymbol{\theta}} = \arg\min_{\theta} \frac{1}{n_D} \sum_{i=1}^{n_D} L(f(\boldsymbol{x}_i; \theta), \boldsymbol{y}_i) + \lambda R(\theta)$$

  - $R(\theta)$ is a "regularization function" or "regularizer", designed to reduce over-fitting.
  - λ is a hyperparameter where higher values increase regularization.

Regularization

- Minimizing the loss L directly usually results in over-fitting. It is standard to add regularization:
$$\hat{\boldsymbol{\theta}} = \arg\min_{\theta} \frac{1}{n_D} \sum_{i=1}^{n_D} L(f(\boldsymbol{x}_i;\theta), \boldsymbol{y}_i) + \lambda R(\theta)$$
  - $R(\theta)$ is a "regularization function" or "regularizer", designed to reduce over-fitting.
  - λ is a hyperparameter where higher values increase regularization.
- For example:
  - "Ridge" penalty:
$$R_2 = \|\theta\|_2^2 = \sum_{j=1}^{n_x} (\theta_j)^2$$
  - "Lasso" penalty:
$$R_1 = \|\theta\|_1 = \sum_{j=1}^{n_x} |\theta_{j]}|$$

Elastic Net = Lasso + Ridge

- The Elastic Net cost function is:

$$\begin{aligned} J(\theta) &= \text{MSE}(\theta) + \lambda_1 R_1 + \lambda_2 R_2 \\ &= \text{MSE}(\theta) + \lambda_1 \sum_{j=1}^{n_x} |\theta_{j]}| + \lambda_2 \sum_{j=1}^{n_x} (\theta_j)^2 \end{aligned}$$

  - λ_1 is a hyperparameter setting the strength of the L1 (Lasso) penalty – automatically performs feature selection and outputs a sparse model.

Elastic Net = Lasso + Ridge

- The Elastic Net cost function is:

$$
\begin{aligned}
J(\theta) &= \mathrm{MSE}(\theta) + \lambda_1 R_1 + \lambda_2 R_2 \\
&= \mathrm{MSE}(\theta) + \lambda_1 \sum_{j=1}^{n_x} |\theta_{j]}| + \lambda_2 \sum_{j=1}^{n_x} (\theta_j)^2
\end{aligned}
$$

  - λ_1 is a hyperparameter setting the strength of the L1 (Lasso) penalty – automatically performs feature selection and outputs a sparse model.
  - λ_2 is a hyperparameter setting the strength of the L2 (Ridge) penalty – shrinks coefficients toward zero and helps select between collinear predictors.

Scaling while maintaining sparsity

- Regularization penalties are designed to work with scaled data.
  - An important feature of text data is sparsity, which is lost when taking out the mean:

$$\tilde{x}_i = \frac{x_i - \bar{\boldsymbol{x}}}{\mathrm{SD}[\boldsymbol{x}]}$$

- Solution:

$$\tilde{x}_i = \frac{x_i}{\mathrm{SD}[\boldsymbol{x}]}$$

  - in sklearn, use **`StandardScaler(with_mean=False)`**.
  - *Note: should not include validation/test observations in calculating SD.*

Selecting Elastic Net Hyperparameters

- Elastic net hyperparameters should be selected to optimize out-of-sample fit.
- "Grid search" scans over the hyperparameter space ($\lambda_1 \geq 0$,$\lambda_2 \geq 0$), computes out-of-sample MSE for all pairs (λ_1, λ_2), and selects the MSE-minimizing model.

Selecting Elastic Net Hyperparameters

- Elastic net hyperparameters should be selected to optimize out-of-sample fit.
- "Grid search" scans over the hyperparameter space ($\lambda_1 \geq 0$,$\lambda_2 \geq 0$), computes out-of-sample MSE for all pairs (λ_1, λ_2), and selects the MSE-minimizing model.
- In general, mean squared error (MSE) should be used as the metric for selecting regression models.

Evaluating Regression Models: R^2

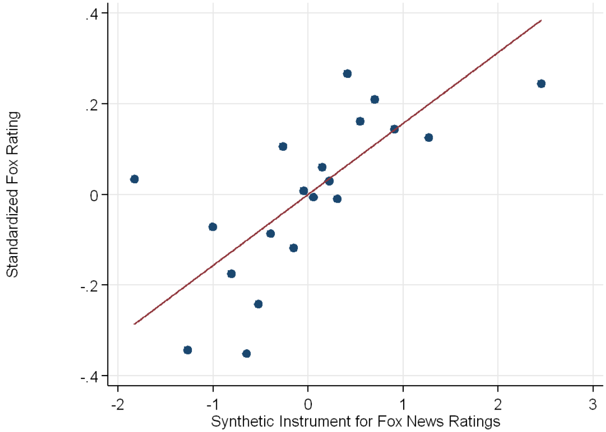
- Mean squared error is good for comparing regression models, but the units depend on the outcome variable and therefore are not interpretable.
  - Better to use R^2 in the test set.
  - MSE and R^2 provide the same ranking of models.

Evaluating Regression Models: R^2

- Mean squared error is good for comparing regression models, but the units depend on the outcome variable and therefore are not interpretable.
  - Better to use R^2 in the test set.
  - MSE and R^2 provide the same ranking of models.
- If you do not want to lose data, you can use cross-validation to get evaluation metrics for every fold in the data:
  - split data into K folds, e.g. 5.
  - for each fold $k \in \{1, 2, ..., K\}$, train and tune model in rest of data $-k$, and evaluate R^2 in k.
  - Report mean and s.d. of R^2 across folds.

Evaluating Regression Models: Binscatter Plots

- Binscatters (e.g. seaborn.regplot) provide visual evidence of regression model performance: Plot y against $\hat{y}$ in the test set:



Sample from Ash and Labzina (2019). $R^2 = .03$.

Binary Outcome $\leftrightarrow$ Binary Classification

- Binary classifiers try to match a boolean outcome $y \in \{0, 1\}$.
- Simplest linear model is

$$\hat{y} = \text{sign}(\boldsymbol{x} \cdot \theta)$$

  that is, choose one if positive and zero if negative.

Binary Outcome $\leftrightarrow$ Binary Classification

- Binary classifiers try to match a boolean outcome $y \in \{0, 1\}$.
- Simplest linear model is
$$\hat{y} = \text{sign}(\boldsymbol{x} \cdot \theta)$$
that is, choose one if positive and zero if negative.
- Better option: use a sigmoid transformation
$$\hat{y} = \text{sigmoid}(\boldsymbol{x} \cdot \theta)$$
with
$$\text{sigmoid}(a) = \frac{1}{1 + \exp(-a)}$$
  - Then $\hat{y} \in [0, 1]$ can be interpreted as the predicted probability of class 1 given $\boldsymbol{x}$.

Training Classifiers

- Better classifiers $\leftrightarrow$ more accurate in held-out test set.

Training Classifiers

- Better classifiers $\leftrightarrow$ more accurate in held-out test set.
  - as above, tune a model's hyperparameters in the training set, then evaluate in the test set.
  - Can use MSE as cost function in binary models, but other cost functions have better performance.

Binary cross-entropy (logistic loss)

- Assume $y \in \{0, 1\}$ and $\hat{y} \in [0, 1]$ (e.g., by sigmoid transformation).
- Prediction rule is 0 for $\hat{y} < .5$ and 1 otherwise.

Binary cross-entropy (logistic loss)

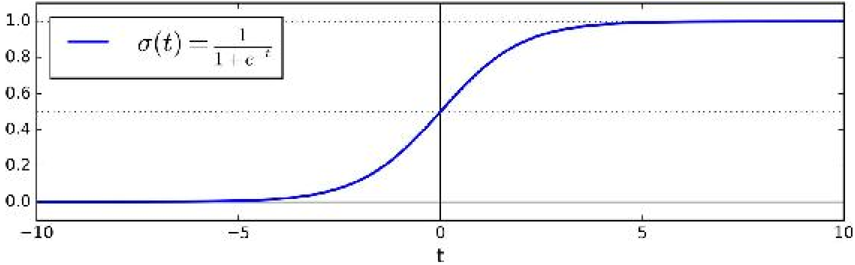
- Assume $y \in \{0, 1\}$ and $\hat{y} \in [0, 1]$ (e.g., by sigmoid transformation).
- Prediction rule is 0 for $\hat{y} < .5$ and 1 otherwise.
- Logistic loss:

$$L(\theta) = -y \log \hat{y}(x, \theta) - [1 - y] \log(1 - \hat{y}(x, \theta))$$

Logistic Regression is for Classification

- Logistic "regression" computes a weighted sum of the input features to predict the output.
  - But rather than output the sum directly, it transforms the sum using the logistic function:

$$\hat{y} = \Pr(y_i = 1) = \frac{1}{1 + \exp(-\theta' x)})$$



- The prediction $\hat{Y} \in \{0, 1\}$ is determined by whether $\hat{p} \gtrless .5$.
- Can be regularized with L1 or L2 penalties.

Logistic Regression Cost Function

- The log loss

$$L(\theta) = \underbrace{-\frac{1}{n_D}}_{\text{negative}} \sum_{i=1}^{n_D} [\underbrace{y_i}_{y_i=1} \underbrace{\log(\hat{y}_i)}_{\text{log prob} y_i=1} + \underbrace{(1-y_i)}_{y_i=0} \underbrace{\log(1-\hat{y}_i)}_{\text{log prob} y_i=0}]$$

  does not have a closed form solution
- but it is convex, so gradient descent will find the global minimum.

Logistic Regression Cost Function

- The log loss

$$L(\theta) = \underbrace{-\frac{1}{n_D}}_{\text{negative}} \sum_{i=1}^{n_D} [\underbrace{y_i}_{y_i=1} \underbrace{\log(\hat{y}_i)}_{\text{log prob} y_i=1} + \underbrace{(1-y_i)}_{y_i=0} \underbrace{\log(1-\hat{y}_i)}_{\text{log prob} y_i=0}]$$

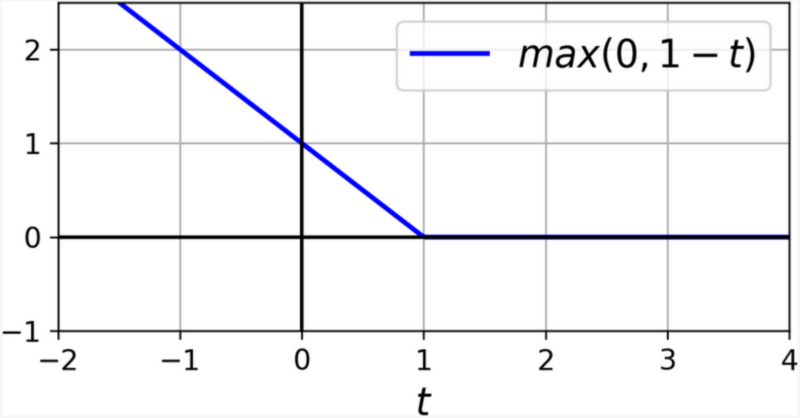
  does not have a closed form solution
- but it is convex, so gradient descent will find the global minimum.
- Just like linear models, logistic can be regularized with L1 or L2 penalties, e.g.:

$$L_2(\theta) = \mathrm{L}(\theta) + \lambda_2 \sum_{j=1}^{n_x} \theta_j^2$$

Hinge Loss (used in SVM)

HINGE LOSS

The function *max*(0, 1 – *t*) is called the *hinge loss* function (see the following image). It is equal to 0 when $t \geq 1$. Its derivative (slope) is equal to –1 if $t < 1$ and 0 if $t > 1$. It is not differentiable at $t = 1$, but just like for Lasso Regression (see "Lasso Regression"), you can still use Gradient Descent using any *subderivative* at $t = 1$ (i.e., any value between –1 and 0).



Confusion Matrix

- A confusion matrix is a nice way to visualize classifier performance:

		True Class	
		Positive	Negative
Predicted Class	Positive	# True Positives	# False Positives
	Negative	# False Negatives	# True Negatives

- The values in the table give counts in the evaluation set.

Precision and Recall

		True Class	
		Positive	Negative
Predicted Class	Positive	# True Positives	# False Positives
	Negative	# False Negatives	# True Negatives

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} = \text{Row Mean}$$

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} = \text{Column Mean}$$

Precision and Recall

		True Class	
		Positive	Negative
Predicted Class	Positive	# True Positives	# False Positives
	Negative	# False Negatives	# True Negatives

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} = \text{Row Mean}$$

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} = \text{Column Mean}$$

- **Precision decreases with false positives.**
- **Recall decreases with false negatives.**

Precision and Recall

		True Class	
		Positive	Negative
Predicted Class	Positive	# True Positives	# False Positives
	Negative	# False Negatives	# True Negatives

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} = \text{Row Mean}$$

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} = \text{Column Mean}$$

- **Precision decreases with false positives.**
- **Recall decreases with false negatives.**
- **Note:** here, we computed precision and recall for the positive class. Can do the same for the negative class (symmetrically).

Accuracy vs. F1 Score

- If labels are (almost) balanced, then accuracy (share correct predictions) is a decent metric.
  - If not (say 90% in one category), accuracy will be uninformative/misleading.

Accuracy vs. F1 Score

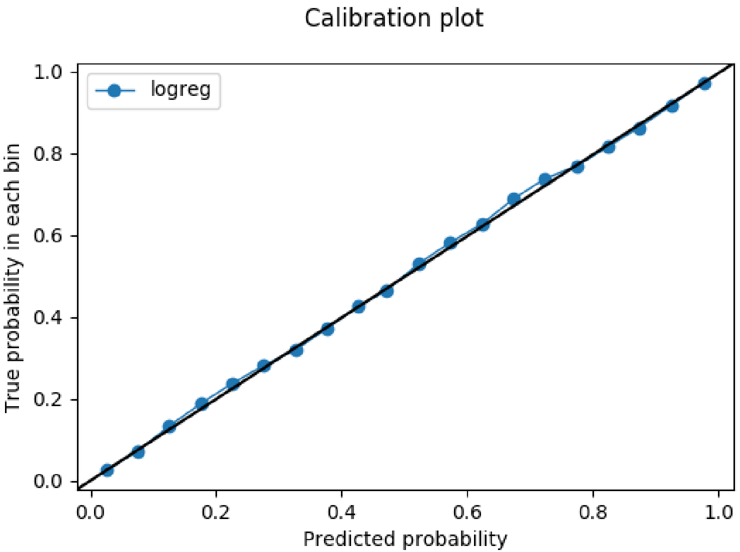
- If labels are (almost) balanced, then accuracy (share correct predictions) is a decent metric.
  - If not (say 90% in one category), accuracy will be uninformative/misleading.
- F_1 score provides a combined metric for use with imbalanced data, the harmonic mean of precision and recall:

$$F_1 = \frac{2}{\frac{1}{\text{precision}} + \frac{1}{\text{recall}}} = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

  - requires decent accuracy for both classes.

Evaluating Classification Models: Calibration Curves

- Plotting the binned fraction in a category (Y axis) against the predicted probability in a category (X axis):



Application: Predicting Political Party from Text

Andrew Peterson and Arthur Spirling, "Classification accuracy as a substantive quantity of interest: Measuring polarization in Westminster systems," ***Political Analysis*** **(2018).**

Application: Predicting Political Party from Text

Andrew Peterson and Arthur Spirling, "Classification accuracy as a substantive quantity of interest: Measuring polarization in Westminster systems," *Political Analysis* (2018).

- Machine Learning Problem:
  - Corpus $D = 3.5$M U.K. parliament speeches, 1935-2013.

Application: Predicting Political Party from Text

Andrew Peterson and Arthur Spirling, "Classification accuracy as a substantive quantity of interest: Measuring polarization in Westminster systems," ***Political Analysis*** **(2018).**

- Machine Learning Problem:
  - Corpus D = 3.5M U.K. parliament speeches, 1935-2013.
  - Label Y = party of speaker (Conservative or Labour)

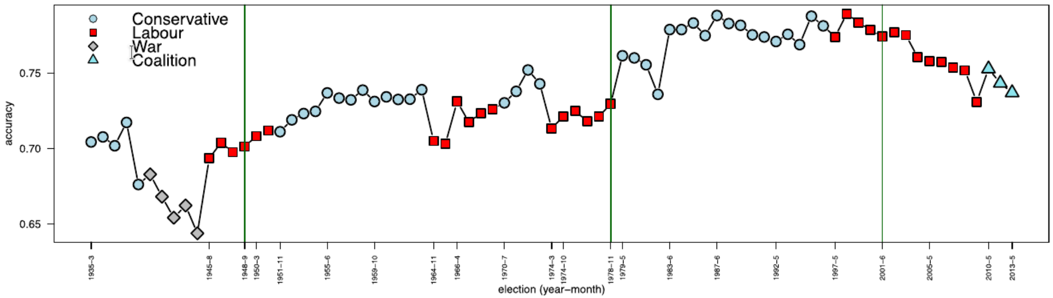
Application: Predicting Political Party from Text

Andrew Peterson and Arthur Spirling, "Classification accuracy as a substantive quantity of interest: Measuring polarization in Westminster systems," *Political Analysis* (2018).

- Machine Learning Problem:
  - Corpus D = 3.5M U.K. parliament speeches, 1935-2013.
  - Label Y = party of speaker (Conservative or Labour)
- Analysis:
  - In years that classifier is more accurate, speech is more polarized.

Polarization in U.K. Parliament

Peterson and Spirling (*Political Analysis* 2018)



- Accuracy of party prediction over time.

Polynomial Features don't work in NLP

- There could be important pair-wise interactions or non-linear effects of the features.
- But computing polynomial features in NLP problems is (usually) computationally intractable:
  - for n_x features, a degree-2 polynomial transformation gives $2n_x + (n_x - 1)^2$ features.
  - e.g., with 10K features, transformation produces 100M polynomial features.

The Kernel Trick

- Consider two vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ with dot product $\boldsymbol{a} \cdot \boldsymbol{b}$.
- Let $\phi(\cdot)$ be a transformation of the vectors which increases dimensionality, satisfying certain properties (Mercer 1909).
  - E.g. a degree-2 polynomial transformation:

$$\phi(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}) = \begin{bmatrix} x_1^2 \\ \sqrt{2}x_1x_2 \\ x_2^2 \end{bmatrix}$$

The Kernel Trick

- Consider two vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ with dot product $\boldsymbol{a} \cdot \boldsymbol{b}$.
- Let $\phi(\cdot)$ be a transformation of the vectors which increases dimensionality, satisfying certain properties (Mercer 1909).
  - E.g. a degree-2 polynomial transformation:

$$\phi\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) = \begin{bmatrix} x_1^2 \\ \sqrt{2}x_1x_2 \\ x_2^2 \end{bmatrix}$$

- Then the dot product between a transformation of two vectors, $\phi(\boldsymbol{a}) \cdot \phi(\boldsymbol{b})$, can be computed as a transformation of the dot product of the vectors $\boldsymbol{a} \cdot \boldsymbol{b}$.
  - e.g., for the quadratic transformation, $\phi(\boldsymbol{a}) \cdot \phi(\boldsymbol{b}) = (\boldsymbol{a} \cdot \boldsymbol{b})^2$.

The Kernel Trick

- Consider two vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ with dot product $\boldsymbol{a} \cdot \boldsymbol{b}$.
- Let $\phi(\cdot)$ be a transformation of the vectors which increases dimensionality, satisfying certain properties (Mercer 1909).
  - E.g. a degree-2 polynomial transformation:

$$\phi\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) = \begin{bmatrix} x_1^2 \\ \sqrt{2}x_1x_2 \\ x_2^2 \end{bmatrix}$$

- Then the dot product between a transformation of two vectors, $\phi(\boldsymbol{a}) \cdot \phi(\boldsymbol{b})$, can be computed as a transformation of the dot product of the vectors $\boldsymbol{a} \cdot \boldsymbol{b}$.
  - e.g., for the quadratic transformation, $\phi(\boldsymbol{a}) \cdot \phi(\boldsymbol{b}) = (\boldsymbol{a} \cdot \boldsymbol{b})^2$.
- This "kernel trick" means that some classifiers (notably SVM) can use the predictive information from feature interactions without increasing dimensionality.

Multi-Class Models

- Many interesting machine learning problems involve multiple un-ordered categories:
  - categorizing a case by area of law.
  - predicting the political party of a speaker in a multi-party system.
  - predicting topic labels in speeches or articles

Multiple Classes: Setup

- The outcome $y_i \in \{1, ..., k, ..., n_y\}$ output classes, with one-hot vector representation
$$\boldsymbol{y}_i = \{\mathbf{1}[y_i = 1], ..., \mathbf{1}[y_i = n_y]\}$$

Multiple Classes: Setup

- The outcome $y_i \in \{1, ..., k, ..., n_y\}$ output classes, with one-hot vector representation
$$\boldsymbol{y}_i = \{\mathbf{1}[y_i = 1], ..., \mathbf{1}[y_i = n_y]\}$$
- We want to learn a vector function
$$\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x}, \theta)$$
with items $\{f_1(\boldsymbol{x}, \theta), ..., f_{n_y}(\boldsymbol{x}, \theta)\}$ and where θ is the set of learnable parameters.

Multiple Classes: Setup

- The outcome $y_i \in \{1, ..., k, ..., n_y\}$ output classes, with one-hot vector representation
$$\boldsymbol{y}_i = \{\mathbf{1}[y_i = 1], ..., \mathbf{1}[y_i = n_y]\}$$
- We want to learn a vector function
$$\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x}, \theta)$$
with items $\{f_1(\boldsymbol{x}, \theta), ..., f_{n_y}(\boldsymbol{x}, \theta)\}$ and where θ is the set of learnable parameters.
- Our machine classifier outputs a vector of probabilities
$$\hat{\boldsymbol{y}} = \{\hat{y}^1, ..., \hat{y}^{n_y}\}, \hat{y}^k \in [0, 1] \; \forall k$$
and for prediction/decision, selects the highest-probability class:
$$\tilde{y} = \arg\max_k \hat{y}_{[k]}$$

Softmax Transformation

- Since $\hat{y}$ are probabilities, we normally want to ensure they are non-negative and sum to one.
- Standard approach: transform the score vector using the softmax function:

$$\mathrm{softmax}(\boldsymbol{a})_{[k]} = \frac{\exp(\boldsymbol{a}_{[k]})}{\sum_l \exp(\boldsymbol{a}_{[l]})}$$

  the multi-class generalization of sigmoid$(\cdot)$.
- In our notation:

$$\hat{y}_{[k]}(\boldsymbol{x}, \theta) = \frac{\exp(f_k(\boldsymbol{x}, \theta))}{\sum_l \exp(f_l(\boldsymbol{x}, \theta))}$$

Categorical Cross Entropy

- The standard loss function in multinomial classification is **categorical cross entropy**:

$$L(\theta) = -\sum_{k=1}^{n_y} \boldsymbol{y}_{[k]} \log(\hat{\boldsymbol{y}}_{[k]}(\boldsymbol{x}, \theta))$$

  - measures the dissimilarity between the true label distribution $\boldsymbol{y}$ and the predicted label distribution $\hat{\boldsymbol{y}}$.
  - convex, so gradient descent will find the global minimum.

Categorical Cross Entropy

- The standard loss function in multinomial classification is **categorical cross entropy**:

$$L(\theta) = -\sum_{k=1}^{n_y} \boldsymbol{y}_{[k]} \log(\hat{\boldsymbol{y}}_{[k]}(\boldsymbol{x}, \theta))$$

  - measures the dissimilarity between the true label distribution $\boldsymbol{y}$ and the predicted label distribution $\hat{\boldsymbol{y}}$.
  - convex, so gradient descent will find the global minimum.
- Usually there is just one true class ($y = 1$ for one class, and zero for others). So this simplifies to

$$L(\theta) = -\log(\hat{\boldsymbol{y}}_{[k^*]}(\boldsymbol{x}, \theta))$$

  where k^* is the true class.
  - Rewards putting higher probability on the true class, ignores distribution of probabilities on other classes.

Multinomial Logistic Regression

- Logistic can be generalized to multiple classes.
  - For a given data point i, multinomial logistic computes a score $f_k(\boldsymbol{x}_i)$ for each class k,
$$f_k(\boldsymbol{x}_i) = \theta_k' \boldsymbol{x}_i$$
  - With n_x features and n_y output classes, there is a $n_y \times n_x$ parameter matrix Θ, where the parameters for each class are stored as rows.

Multinomial Logistic Regression

- Logistic can be generalized to multiple classes.
  - For a given data point i, multinomial logistic computes a score $f_k(\boldsymbol{x}_i)$ for each class k,
$$f_k(\boldsymbol{x}_i) = \theta'_k \boldsymbol{x}_i$$
  - With n_x features and n_y output classes, there is a $n_y \times n_x$ parameter matrix Θ, where the parameters for each class are stored as rows.
- Using the scores, probabilities for each class are computed using the softmax function
$$\hat{y}_k(\boldsymbol{x}_i) = \Pr(y_i = k) = \frac{\exp(\theta'_k \boldsymbol{x}_i)}{\sum_{l=1}^{n_y} \exp(\theta'_l \boldsymbol{x}_i)}$$
  - And the prediction $y_i \in \{1, ..., n_y\}$ is determined by the highest-probability category.

Regularized Multinomial Logistic

- The corpus-wide categorical cross entropy is

$$\mathcal{L}(\theta) = \underbrace{-\frac{1}{n_D}}_{\text{negative}} \sum_{i=1}^{n_D} \sum_{k=1}^{n_y} \underbrace{\mathbf{1}[y_i = k]}_{y_i=k} \underbrace{\log(\hat{\boldsymbol{y}}_k(\boldsymbol{x}_i))}_{\text{log prob} y_i=k}$$

Regularized Multinomial Logistic

- The corpus-wide categorical cross entropy is

$$\mathcal{L}(\theta) = \underbrace{-\frac{1}{n_D}}_{\text{negative}} \sum_{i=1}^{n_D} \sum_{k=1}^{n_y} \underbrace{\mathbf{1}[y_i = k]}_{y_i = k} \underbrace{\log(\hat{\boldsymbol{y}}_k(\boldsymbol{x}_i))}_{\text{log prob} y_i = k}$$

- The L2-penalized logistic regression loss is

$$\mathcal{L}(\theta) = -\frac{1}{n_D} \sum_{i=1}^{n_D} \sum_{k=1}^{n_y} \mathbf{1}[y_i = k] \log \frac{\exp(\theta'_k \boldsymbol{x}_i)}{\sum_{l=1}^{n_y} \exp(\theta'_l \boldsymbol{x}_i)} + \lambda \sum_{j=1}^{n_x} \sum_{k=1}^{n_y} (\theta_{[j,k]})^2$$

    - λ = strength of L2 penalty (could also add lasso penalty)
    - as before, predictors should be scaled to the same variance.

Multi-Class Confusion Matrix

		Predicted Class		
		Class A	Class B	Class C
True Class	Class A	Correct A	A, classed as B	A, classed as C
	Class B	B, classed as A	Correct B	B, classed as C
	Class C	C, classed as A	C, classed as B	Correct C

- More generally, can have a confusion matrix M with items M_{ij} (row i, column j).

Multi-Class Performance Metrics

Confusion matrix M with items M_{ij} (row i, column j).

$$\text{Precision for } k = \frac{\text{True Positives for } k}{\text{True Positives for k} + \text{False Positives for } k} = \frac{M_{kk}}{\sum_l M_{lk}}$$

$$\text{Recall for } k = \frac{\text{True Positives for } k}{\text{True Positives for } k \ + \text{ False Negatives for } k} = \frac{M_{kk}}{\sum_l M_{kl}}$$

$$F_1(k) = 2 \times \frac{\text{precision}(k) \times \text{recall}(k)}{\text{precision}(k) + \text{recall}(k)}$$

Metrics for whole model

- **Macro-averaging**: average of the per-class precision, recall, and F1, e.g.

$$F_1 = \frac{1}{n_y} \sum_{k=1}^{n_y} F_1(k)$$

(weights all classes equally)

Metrics for whole model

- **Macro-averaging**: average of the per-class precision, recall, and F1, e.g.

$$F_1 = \frac{1}{n_y} \sum_{k=1}^{n_y} F_1(k)$$

  (weights all classes equally)
- **Micro-averaging**: Compute model-level sums for true positives, false positives, and false negatives; compute precision/recall from model sums.

$$\text{Precision} = \frac{\text{Total TP}}{\text{Total TP + Total FP}}, \text{Recall} = \frac{\text{Total TP}}{\text{Total TP + Total FN}}, F_1 = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

  (up-weights more frequent classes)

Metrics for whole model

- **Macro-averaging**: average of the per-class precision, recall, and F1, e.g.

$$F_1 = \frac{1}{n_y}\sum_{k=1}^{n_y} F_1(k)$$

(weights all classes equally)

- **Micro-averaging**: Compute model-level sums for true positives, false positives, and false negatives; compute precision/recall from model sums.

$$\text{Precision} = \frac{\text{Total TP}}{\text{Total TP} + \text{Total FP}}, \text{Recall} = \frac{\text{Total TP}}{\text{Total TP} + \text{Total FN}}, F_1 = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

(up-weights more frequent classes)

- "**Weighted**": computed like macro-averaging, but classes are weighted by true frequency.

Multiple Continuous Outcomes

- What if we want to predict multiple continuous (non-categorical) outcomes?
  - e.g., want to predict the share of court case text across three categories: facts, law, and verdict.
  - or the share of speaking time used by each judge in oral argument

Multiple Continuous Outcomes

- What if we want to predict multiple continuous (non-categorical) outcomes?
  - e.g., want to predict the share of court case text across three categories: facts, law, and verdict.
  - or the share of speaking time used by each judge in oral argument
- Regression models (in particular, many of the regression models in scikit-learn) work fine in this case.
  - costs (e.g. MSE) are added up across outcome classes for each observation.

Cross-Domain (Transfer) Learning

- A recent but now widespread approach to machine learning is **transfer learning**:
  - train a model in a big labeled dataset
  - apply in a smaller (mostly) unlabeled dataset

Cross-Domain (Transfer) Learning

- A recent but now widespread approach to machine learning is **transfer learning**:
  - train a model in a big labeled dataset
  - apply in a smaller (mostly) unlabeled dataset
- In NLP:
  - transfer learning is intuitive because NLP tasks share common knowledge about language.
  - labeled data is scarce/expensive, so learn tasks on tons of unlabeled data.

Cross-Domain (Transfer) Learning

- A recent but now widespread approach to machine learning is **transfer learning**:
  - train a model in a big labeled dataset
  - apply in a smaller (mostly) unlabeled dataset
- In NLP:
  - transfer learning is intuitive because NLP tasks share common knowledge about language.
  - labeled data is scarce/expensive, so learn tasks on tons of unlabeled data.
  - BERT and GPT-2 are the big success stories in transfer learning.

Osnabruegge, Ash, and Morelli 2020

This paper takes the idea of transfer learning to the political science context.

- Learn to predict political topics from text in a labeled corpus (party manifestos from Comparative Manifesto Project)

Osnabruegge, Ash, and Morelli 2020

This paper takes the idea of transfer learning to the political science context.

- Learn to predict political topics from text in a labeled corpus (party manifestos from Comparative Manifesto Project)
- Apply model to classify topics in unlabeled corpus (parliamentary speeches).

Osnabruegge, Ash, and Morelli 2020

This paper takes the idea of transfer learning to the political science context.

- Learn to predict political topics from text in a labeled corpus (party manifestos from Comparative Manifesto Project)
- Apply model to classify topics in unlabeled corpus (parliamentary speeches).
- Use for empirical analysis of electoral institutions and speech content.

Overview of Text Analysis Methods

	Dictionaries (Custom)	Dictionaries (Generic)	Topic Modeling	Within-Domain Supervised Learning	Cross-Domain Supervised Learning
Design/Annotation Costs	High	Low	Low	High	Moderate
Specificity	High	Moderate	Low	High	Moderate
Interpretability	High	High	Moderate	High	High
Validatability	Low	Low	Low	High	High

	Dictionaries (Custom)	Dictionaries (Generic)	Topic Modeling	Within-Domain Supervised Learning	Cross-Domain Supervised Learning
Design/Annotation Costs	High	Low	Low	High	Moderate
Specificity	High	Moderate	Low	High	Moderate
Interpretability	High	High	Moderate	High	High
Validatability	Low	Low	Low	High	High

- **Design/Annotation Costs**: How significant are the classifier's up-front costs?
  - designing a (custom) dictionary of terms and labeling data (for within-domain machine learning) are costly in the design process.

	Dictionaries (Custom)	Dictionaries (Generic)	Topic Modeling	Within-Domain Supervised Learning	Cross-Domain Supervised Learning
Design/Annotation Costs	High	Low	Low	High	Moderate
Specificity	High	Moderate	Low	High	Moderate
Interpretability	High	High	Moderate	High	High
Validatability	Low	Low	Low	High	High

- **Design/Annotation Costs**: How significant are the classifier's up-front costs?
  - designing a (custom) dictionary of terms and labeling data (for within-domain machine learning) are costly in the design process.
  - using generic dictionaries (e.g. LIWC) or running a topic model have low up-front costs.

	Dictionaries (Custom)	Dictionaries (Generic)	Topic Modeling	Within-Domain Supervised Learning	Cross-Domain Supervised Learning
Design/Annotation Costs	High	Low	Low	High	Moderate
Specificity	High	Moderate	Low	High	Moderate
Interpretability	High	High	Moderate	High	High
Validatability	Low	Low	Low	High	High

- **Design/Annotation Costs**: How significant are the classifier's up-front costs?
  - designing a (custom) dictionary of terms and labeling data (for within-domain machine learning) are costly in the design process.
  - using generic dictionaries (e.g. LIWC) or running a topic model have low up-front costs.
  - Costs of using a pre-trained model (cross-domain learning) are moderate, assuming that one annotates a subset of documents for validation.

	Dictionaries (Custom)	Dictionaries (Generic)	Topic Modeling	Within-Domain Supervised Learning	Cross-Domain Supervised Learning
Design/Annotation Costs	High	Low	Low	High	Moderate
Specificity	High	Moderate	Low	High	Moderate
Interpretability	High	High	Moderate	High	High
Validatability	Low	Low	Low	High	High

- **Specificity**: Can the method be targeted to particular dimensions of language?
  - custom dictionaries and standard machine learning (with custom annotations) are the highest.
  - topic models are the lowest

	Dictionaries (Custom)	Dictionaries (Generic)	Topic Modeling	Within-Domain Supervised Learning	Cross-Domain Supervised Learning
Design/Annotation Costs	High	Low	Low	High	Moderate
Specificity	High	Moderate	Low	High	Moderate
Interpretability	High	High	Moderate	High	High
Validatability	Low	Low	Low	High	High

- **Specificity**: Can the method be targeted to particular dimensions of language?
  - custom dictionaries and standard machine learning (with custom annotations) are the highest.
  - topic models are the lowest
  - generic dictionaries and cross-domain learning are moderate, as it requires that a dictionary/model already exists for the dimension of interest.

	Dictionaries (Custom)	Dictionaries (Generic)	Topic Modeling	Within-Domain Supervised Learning	Cross-Domain Supervised Learning
Design/Annotation Costs	High	Low	Low	High	Moderate
Specificity	High	Moderate	Low	High	Moderate
Interpretability	High	High	Moderate	High	High
Validatability	Low	Low	Low	High	High

- **Interpretability**: Do you understand what the model is doing?
  - dictionaries and annotation codebooks are easy to inspect.
  - topics produced by LDA are sometimes interpretable, sometimes not.

	Dictionaries (Custom)	Dictionaries (Generic)	Topic Modeling	Within-Domain Supervised Learning	Cross-Domain Supervised Learning
Design/Annotation Costs	High	Low	Low	High	Moderate
Specificity	High	Moderate	Low	High	Moderate
Interpretability	High	High	Moderate	High	High
Validatability	Low	Low	Low	High	High

- **Validatability**: Is the model classifying documents correctly?
  - dictionaries and topic models cannot be validated without hand-labeling of documents, which defeats the purpose.
  - supervised learning models on annotated data can be validated with machine learning metrics.

Comparative Manifesto Project Corpus

- 115,410 annotated English-language political statements
  - hundreds of political party platforms from English-speaking countries.

Comparative Manifesto Project Corpus

- 115,410 annotated English-language political statements
  - hundreds of political party platforms from English-speaking countries.
- Each statement gets a CMP code $\boldsymbol{y}_i$, e.g. “decentralization”, “education”
  - $n_y = 44$ topics
    - some topics are somewhat esoteric, such as “marxist analysis”

Comparative Manifesto Project Corpus

- 115,410 annotated English-language political statements
  - hundreds of political party platforms from English-speaking countries.
- Each statement gets a CMP code $\boldsymbol{y}_i$, e.g. “decentralization”, “education”
  - $n_y = 44$ topics
    - some topics are somewhat esoteric, such as “marxist analysis”
  - also: 8 broader “topic domains” (external relations, freedom and democracy, political system, economy, welfare and quality of life, fabric of society, social groups, and no topic)

Featurizing the Statements

- Standard featurization steps:
  - remove capitalization, punctuation, stopwords
  - construct n-grams up to length 3
  - remove n-grams appearing in less than 10 statements or more than 40 percent of statements

Featurizing the Statements

- Standard featurization steps:
  - remove capitalization, punctuation, stopwords
  - construct n-grams up to length 3
  - remove n-grams appearing in less than 10 statements or more than 40 percent of statements
- $n_x = 19,734$ features
  - compute tf-idf-weighted n-gram frequencies

Prediction Model

- Trained L2-Penalized multinomial logistic model ($\lambda = .5$ selected by cross-validation in training set) to predict classes from n-grams.

Prediction Model

- Trained L2-Penalized multinomial logistic model ($\lambda = .5$ selected by cross-validation in training set) to predict classes from n-grams.

Predict the CMP code in a held-out sample of manifesto corpus statements:

- 44-topic specification:
  - test-sample accuracy = 0.54
  - training-sample accuracy = .70
  - choosing randomly would be correct 2% of the time; choosing most-frequent category (other topic) would be correct 15% of the time.

Prediction Model

- Trained L2-Penalized multinomial logistic model ($\lambda = .5$ selected by cross-validation in training set) to predict classes from n-grams.

Predict the CMP code in a held-out sample of manifesto corpus statements:

- 44-topic specification:
  - test-sample accuracy $= 0.54$
  - training-sample accuracy $= .70$
  - choosing randomly would be correct 2% of the time; choosing most-frequent category (other topic) would be correct 15% of the time.
- 8-topic specification:
  - test-sample accuracy $= 0.64$
  - training-sample accuracy $= 0.76$
  - choosing randomly $==$ 0.125 accuracy, choosing most-frequent category would be correct 30% of the time.

	Economy	External Relations	Fabric of society	Freedom & Democracy	Political system	Social groups	Welfare & quality of life	No topic / Other	Total true
Economy	5270	93	131	40	301	254	1108	0	**7197**
External Relations	175	1207	137	83	85	49	209	1	**1946**
Fabric of society	269	107	1785	90	204	115	618	1	**3189**
Freedom and Democracy	102	60	135	631	219	35	177	0	**1359**
Political system	608	71	186	137	1255	65	542	1	**2865**
Social groups	493	51	185	29	111	1230	818	0	**2917**
Welfare and quality of life	1033	66	316	58	267	293	7138	0	**9171**
No topic / Other	58	6	37	9	34	7	55	3	**209**
Total predicted	**8008**	**1661**	**2912**	**1077**	**2476**	**2048**	**10665**	**6**	
Total predicted / Total true	**1.11**	**0.85**	**0.91**	**0.79**	**0.86**	**0.70**	**1.16**	**0.03**	

New Zealand Parliamentary Speeches

- All parliament speeches, 1987-2002.
  - $437\mathrm{K}$ speeches in total, removed procedural remarks, short speeches, and foreign-language speeches, to get 290K for analysis.
  - speeches linked to speaker and parliamentary debate type.

New Zealand Parliamentary Speeches

- All parliament speeches, 1987-2002.
  - $437\mathrm{K}$ speeches in total, removed procedural remarks, short speeches, and foreign-language speeches, to get 290K for analysis.
  - speeches linked to speaker and parliamentary debate type.
- Apply featurization pipeline and logistic model to get predicted topic probabilities for each speech.

(a) economy

(b) external relations

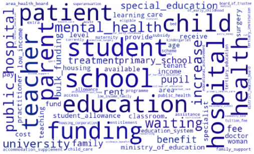
(e) political system

(f) social groups

(c) fabric of society

(d) freedom and democracy

(g) welfare and quality of life



(h) no topic

Validation with Target-Corpus Annotation

Table 3: Human Coding vs. Predicted Manifesto Topics

	Top 1	Top 3	Top 5	N
8 topics				
Welfare and quality of life	62	91	98	796
Political system	57	90	98	1,069
External relations	56	84	91	94
Fabric of society	55	87	97	433
Economy	54	85	95	721
Social groups	37	71	88	325
Freedom and democracy	37	71	88	545
no topic	1	2	12	192
Total	51	82	92	4,175

- We sampled 4,175 NZ speeches and a manifesto coder annotated them.
- In 44-topic spec, overall top-1 accuracy is 41% and top-3 accuracy is 65%

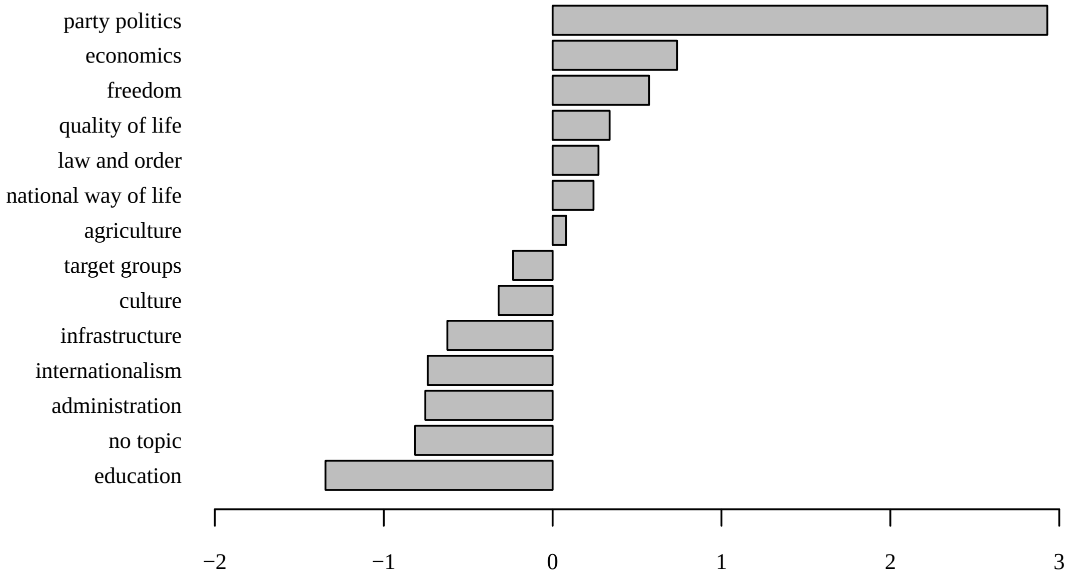
Experiment: Electoral Reform in New Zealand

- A 1993 reform in New Zealand moved from majoritarian to proportional representation:
  - **Majoritarian (first past the post)**: two parties, single party controls parliament.
  - **Proportional representation**: many minority parties, coalition governments.

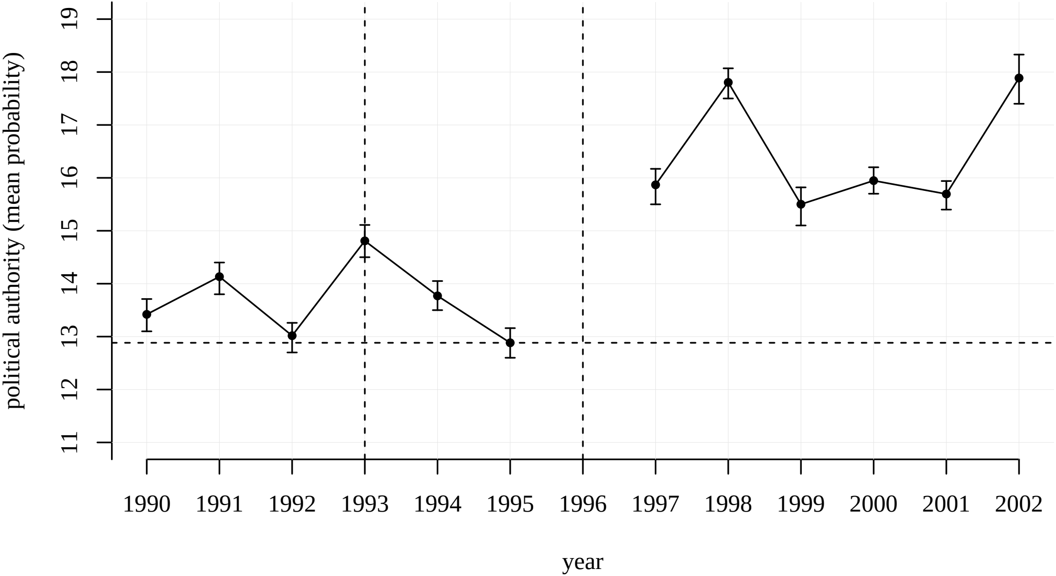
Experiment: Electoral Reform in New Zealand

- A 1993 reform in New Zealand moved from majoritarian to proportional representation:
  - **Majoritarian (first past the post)**: two parties, single party controls parliament.
  - **Proportional representation**: many minority parties, coalition governments.
- How did it affect speech topics in the New Zealand Parliament?

Change in Parliament Attention due to Reform



Change in Party Politics Topic



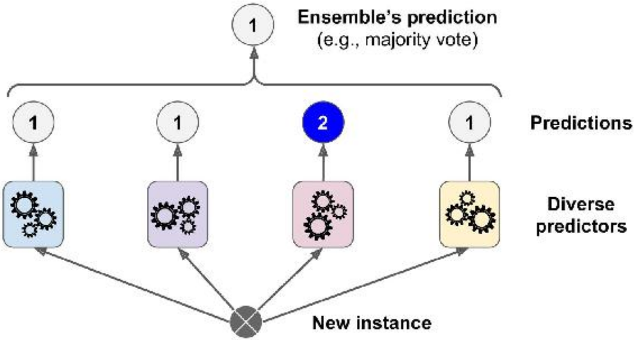
Example "Party Politics" Speech

Osnabruegge, Ash, Morelli (2019)

"I have seen seven Opposition leaders in my time, but I have never seen a leader as relentlessly negative as Helen Clark. . . . How could anybody be so negative, day in, day out? It could get into the Guinness Book of Records. She does not have a positive word to say about anything. It is all negative, negative, negative."

- Parliamentarian Richard Prebble, 15 Feb 1999

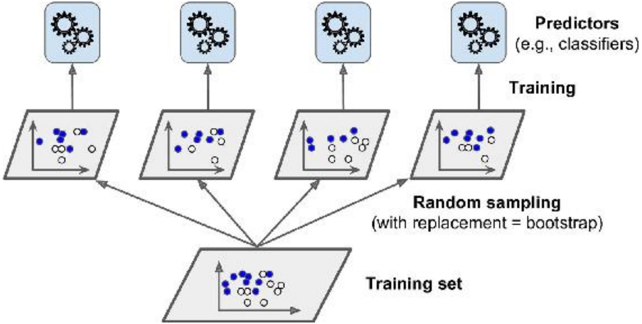
Voting Classifiers



- voting classifiers (ensembles of different models that vote on the prediction) generally out-perform the best classifier in the ensemble.
  - more diverse algorithms will make different types of errors, and improve your ensemble's robustness.

Bagging (Bootstrapping)

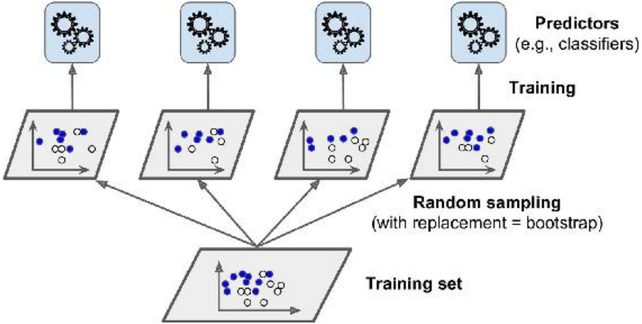
- Rather than use the same data on different classifiers, one can use different subsets of the data on the same classifier:



  - This is called **bagging** (bootsrap aggregating, when sampling with replacement) or **pasting** (when sampling without replacement).
  - can also use different subsets of features across subclassifiers.

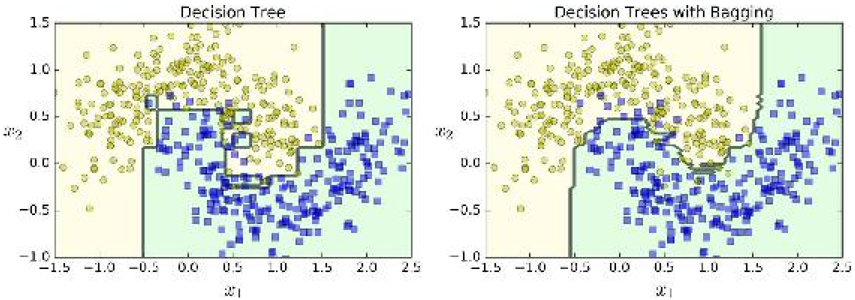
Bagging (Bootstrapping)

- Rather than use the same data on different classifiers, one can use different subsets of the data on the same classifier:



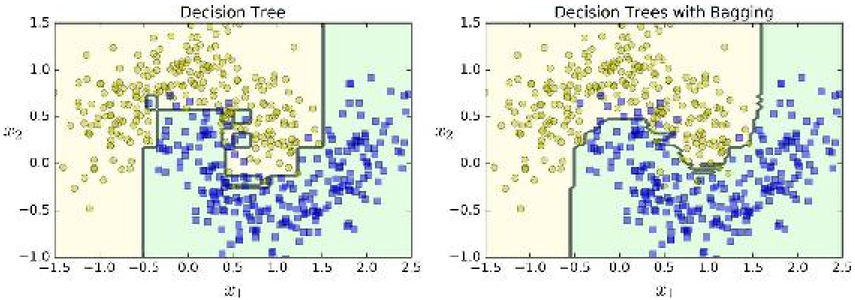
  - This is called **bagging** (bootsrap aggregating, when sampling with replacement) or **pasting** (when sampling without replacement).
  - can also use different subsets of features across subclassifiers.
- The ensemble predicts by aggregating the predictions:
  - for regression, use the mean/median output
  - for classification, can use the mean/median predicted probability or most frequent prediction

Bagging Benefits



- While the individual predictors have a higher bias than a predictor trained on all the data, aggregation reduces both bias and variance.
  - Generally, the ensemble has a similar bias but lower variance than a single predictor trained on all the data.
- Predictors can be trained in parallel using separate CPU cores.

Bagging Benefits



- While the individual predictors have a higher bias than a predictor trained on all the data, aggregation reduces both bias and variance.
  - Generally, the ensemble has a similar bias but lower variance than a single predictor trained on all the data.
- Predictors can be trained in parallel using separate CPU cores.
- Also gives you a distribution of predictions, which useful when using predictions for downstream empirical analysis.

Random Forests and Gradient Boosting

- Random Forests:
  - optimized voting ensembles of decision trees.
  - Good prediction performance – due to out-of-sample validation being baked into the training process.

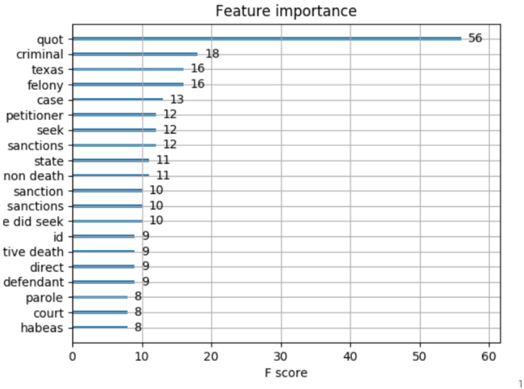
Random Forests and Gradient Boosting

- Random Forests:
  - optimized voting ensembles of decision trees.
  - Good prediction performance – due to out-of-sample validation being baked into the training process.
- Gradient boosting machines:
  - works by sequentially adding predictors to an ensemble – fits the new predictor to the residual errors made by the previous predictor to gradually improve the model.
  - give state-of-the-art performance except (sometimes) deep neural nets.

Feature Importance

```
from xgboost import plot_importance
plot_importance(xgb_reg, max_num_features=20)
```

```
<IPython.core.display.Javascript object>
```



- Random forests and boosted trees provide a metric of feature importance that summarizes how well each feature contributes to predictive accuracy.

A baseline for machine learning using text

1. Take POS-filtered bigrams as inputs X.

A baseline for machine learning using text

1. Take POS-filtered bigrams as inputs X.
2. Select a machine learning model for predicting outcome y:
   - For classification (y is discrete), L2-penalized logistic regression or gradient boosted classifier.
   - For regression (y is one-dimensional and continuous), elastic net or gradient boosted regressor.

A baseline for machine learning using text

1. Take POS-filtered bigrams as inputs X.
2. Select a machine learning model for predicting outcome y:
   - For classification (y is discrete), L2-penalized logistic regression or gradient boosted classifier.
   - For regression (y is one-dimensional and continuous), elastic net or gradient boosted regressor.
   - *(If y is more complicated, e.g. a sequence of words, we use deep learning.)*

A baseline for machine learning using text

1. Take POS-filtered bigrams as inputs X.
2. Select a machine learning model for predicting outcome y:
   - For classification (y is discrete), L2-penalized logistic regression or gradient boosted classifier.
   - For regression (y is one-dimensional and continuous), elastic net or gradient boosted regressor.
   - *(If y is more complicated, e.g. a sequence of words, we use deep learning.)*
3. Use cross-validation grid search in training set to select model hyperparameters.
   - For classification, use categorical cross entropy; for regression, use mean squared error

A baseline for machine learning using text

1. Take POS-filtered bigrams as inputs X.
2. Select a machine learning model for predicting outcome y:
    - For classification (y is discrete), L2-penalized logistic regression or gradient boosted classifier.
    - For regression (y is one-dimensional and continuous), elastic net or gradient boosted regressor.
    - *(If y is more complicated, e.g. a sequence of words, we use deep learning.)*
3. Use cross-validation grid search in training set to select model hyperparameters.
    - For classification, use categorical cross entropy; for regression, use mean squared error
4. Evaluate model in held-out test set:
    - For classification, use F1 score, confusion matrix, and calibration plot.
    - For regression, use R squared and binscatter plot.

A baseline for machine learning using text

1. Take POS-filtered bigrams as inputs X.
2. Select a machine learning model for predicting outcome y:
   - For classification (y is discrete), L2-penalized logistic regression or gradient boosted classifier.
   - For regression (y is one-dimensional and continuous), elastic net or gradient boosted regressor.
   - *(If y is more complicated, e.g. a sequence of words, we use deep learning.)*
3. Use cross-validation grid search in training set to select model hyperparameters.
   - For classification, use categorical cross entropy; for regression, use mean squared error
4. Evaluate model in held-out test set:
   - For classification, use F1 score, confusion matrix, and calibration plot.
   - For regression, use R squared and binscatter plot.
5. Interpret the model predictions:
   - for gradient boosting, use feature importance ranking.
   - for linear models, examine coefficients
   - look at highest and lowest ranked documents for $\hat{y}$

A baseline for machine learning using text

1. Take POS-filtered bigrams as inputs X.
2. Select a machine learning model for predicting outcome y:
   - For classification (y is discrete), L2-penalized logistic regression or gradient boosted classifier.
   - For regression (y is one-dimensional and continuous), elastic net or gradient boosted regressor.
   - *(If y is more complicated, e.g. a sequence of words, we use deep learning.)*
3. Use cross-validation grid search in training set to select model hyperparameters.
   - For classification, use categorical cross entropy; for regression, use mean squared error
4. Evaluate model in held-out test set:
   - For classification, use F1 score, confusion matrix, and calibration plot.
   - For regression, use R squared and binscatter plot.
5. Interpret the model predictions:
   - for gradient boosting, use feature importance ranking.
   - for linear models, examine coefficients
   - look at highest and lowest ranked documents for $\hat{y}$
6. Answer the research question!