

Evidence and Probabilities

1

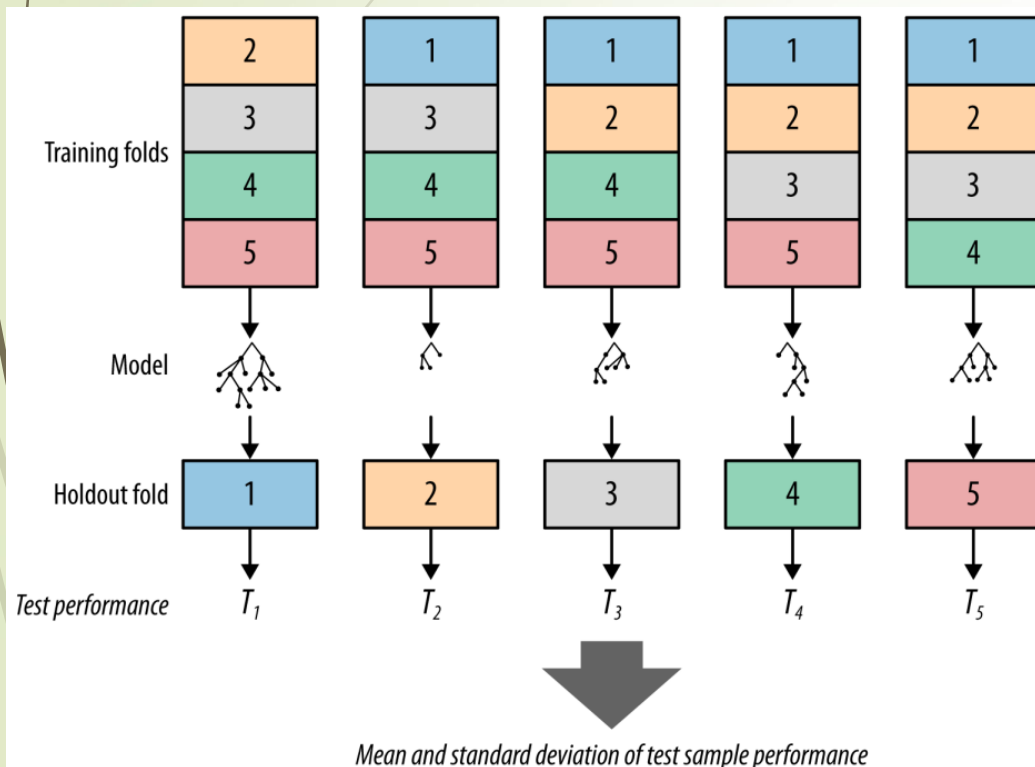
Dr. Yi Long (Neal)

Most contents (text or images) of course slides are from the following textbook
Provost, Foster, and Tom Fawcett. Data Science for Business: What you need to
know about data mining and data-analytic thinking. " O'Reilly Media, Inc.", 2013

Outline

- ▶ Ensemble Modeling
- ▶ Issues in Business Understanding
- ▶ Evidence and Naive Bayes Model

GridSearchCV



```
clf = SVC()
# Select best C for SVC by AUC score of ROC curve
best_clf = GridSearchCV(clf, scoring='roc_auc', cv=5, n_jobs=-1,
                        param_grid={'C': [0.01, 0.1, 1, 10, 100, 500, 1000]})
best_clf.fit(X, y)
print("Select best SVC model with C = {} with average score during 5-fold CV :{}".format(
    best_clf.best_params_['C'],
    best_clf.best_score_))
y_true = y
y_true[y_true == 'yes'] = 1
y_true[y_true == 'no'] = 0
print("The training performance the best SVC selected as above=",
      roc_auc_score(y_true.tolist(), best_clf.decision_function(X).tolist()))
```

Select best SVC model with C = 100 with best average roc_auc score during 5-foldCV :0.9489766981488318
The training performance the best SVC selected as above= 0.9983324827328093

Ensemble Modeling

- **Ensemble modeling**: building lots of different data mining models and combining them into one super model (an ensemble model)
 - ✓ An ensemble as a collection of experts while each model as a sort of “expert” on a target prediction task
 - ✓ A k-nearest-neighbor model is a simple ensemble method
 - ✓ Main approaches: bagging, boosting and stacking
 - ✓ Voting/Averaging
 - ✓ Stacking/Blending
 - ✓ Bagging (base/weak learner)
 - ✓ Boosting (base/weak learner)

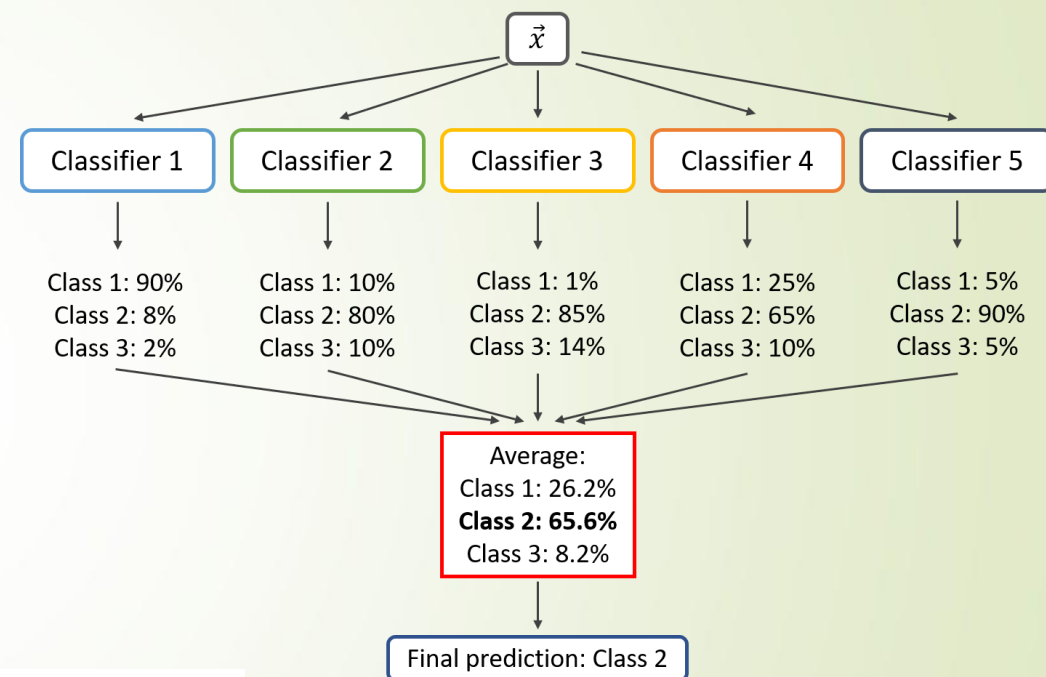
Voting/Averaging

- Max Voting/Averaging (hard)
 - Class 2 receives 4 total votes
- Weighted Voting Averaging (soft)
 - Class 2 receives 0.656 average votes

voting : str, {'hard', 'soft'} (default='hard')

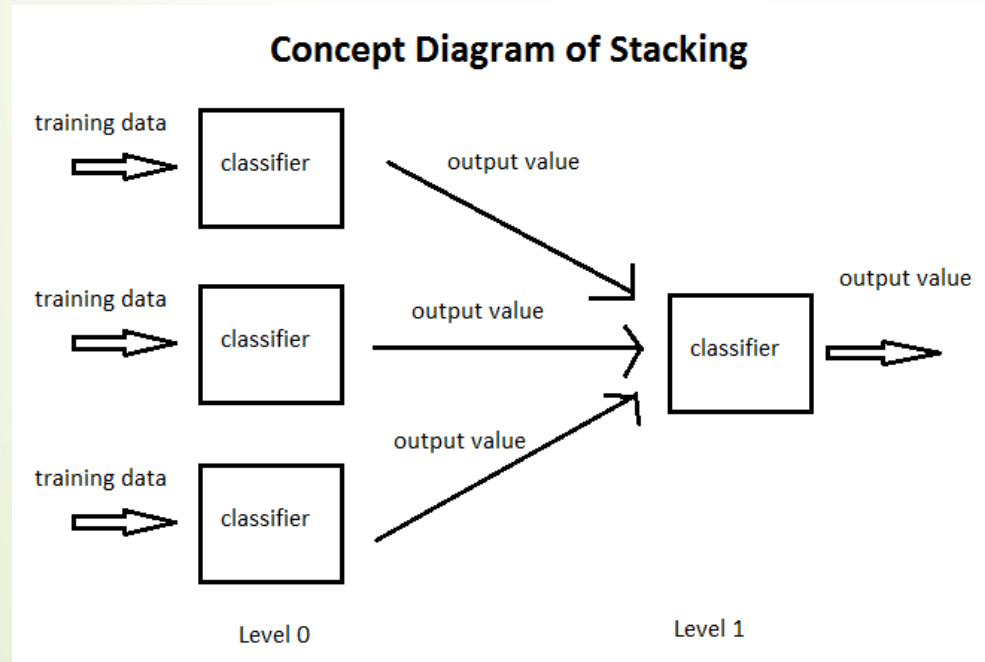
If 'hard', uses predicted class labels for majority rule voting. Else if 'soft', predicts the class label based on the argmax of the sums of the predicted probabilities, which is recommended for an ensemble of well-calibrated classifiers.

<https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.VotingClassifier.html>



Stacking/Blending

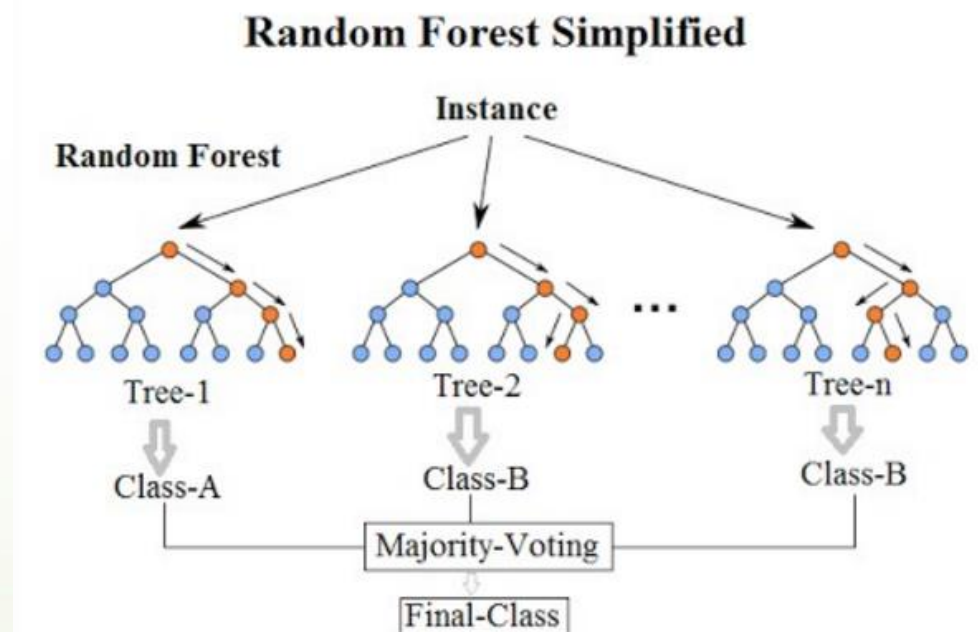
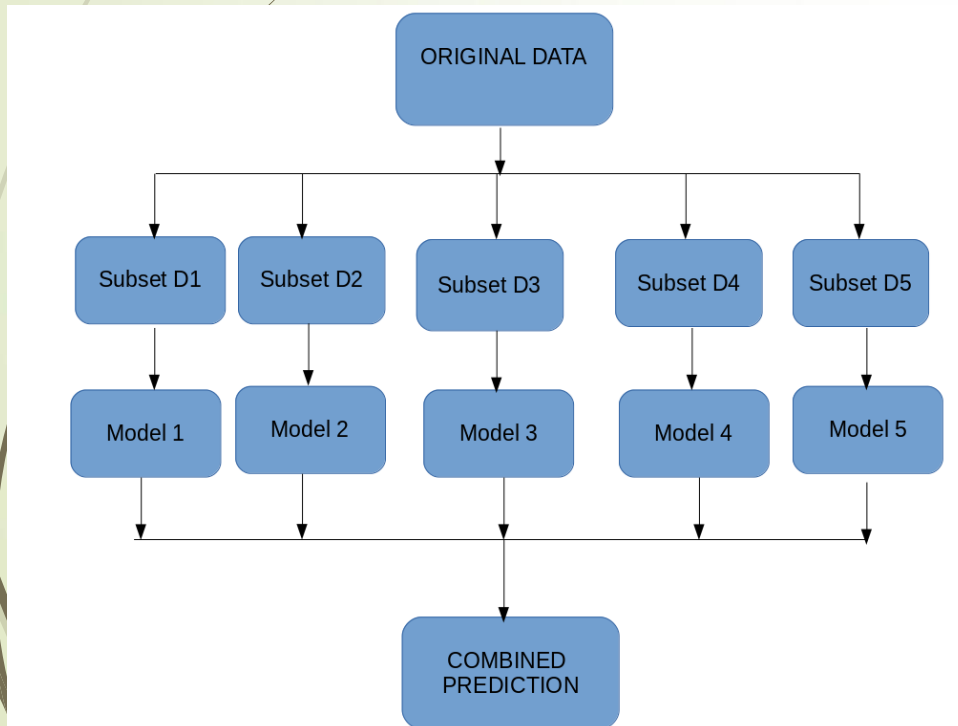
- **Stacking/blending** involves training a learning algorithm to combine the predictions of several other learning algorithms
 - ✓ Can be multiple layers



Bagging

➤ Bootstrap **agg**regating (bagging)

- ✓ Training M models on M independent and **random subset** of the whole dataset
- ✓ Then averaging the output (for regression) or voting (for classification)
- ✓ Example: Random Forest



Boosting

- **Boosting** referred to the process of turning a weak learner into a strong learner
 - ✓ A **weak/base learner** is defined to be a classifier which is only slightly correlated with the true classification (it can label examples better than random guessing)
 - ✓ **A decision stump** (decision tree with depth =1)can be viewed as a weak learner
 - ✓ Example: Adaboost, Gradient Boost Model (GBM)

Step 1: The base learner takes all the distributions and assign equal weight or attention to each observation.

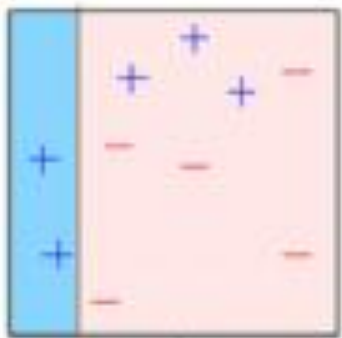
Step 2: If there is any prediction error caused by first base learner, then we pay higher attention to observations having prediction error. Then, we build the next learner

Step 3: Iterate Step 2 till the limit of base learning algorithm is reached or higher accuracy is achieved

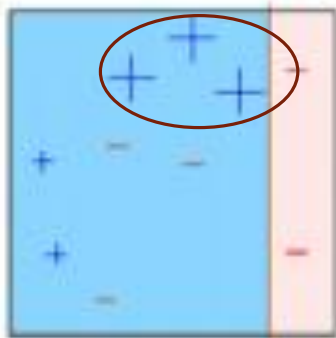
Output: The final model (strong learner) is the weighted mean of all the models (weak learners)

Boosting Examples

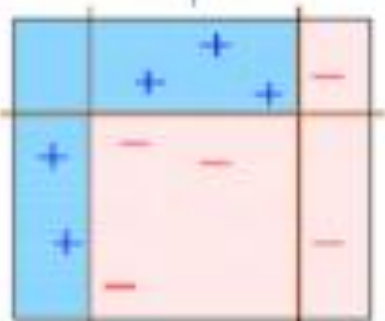
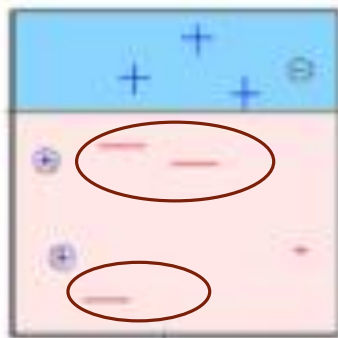
1st Base Learner



2nd Base Learner



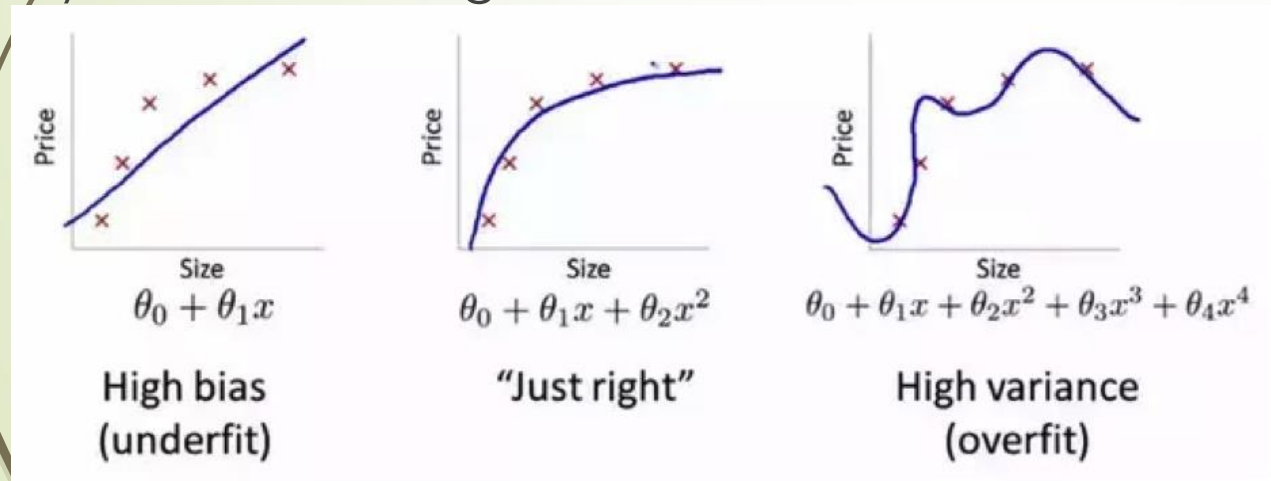
3rd Base Learner



三个臭皮匠，顶个诸葛亮

Bias–variance Tradeoff

- The errors a model makes can be characterized by three factors:
 - ✓ Inherent randomness: different customers with same attribute do not always buy
 - ✓ Bias : fitting nonlinear relationships with linear models raises high bias (**under-fitted**)
 - ✓ Variance: *high variance* usually means more likely to be **overfired**
- Models with higher bias tend to be relatively simple (low-order or even linear regression polynomials), but may produce lower variance predictions when applied beyond the training set.



Target:

Low Variance+ Low Bias

Why Ensemble Learning?

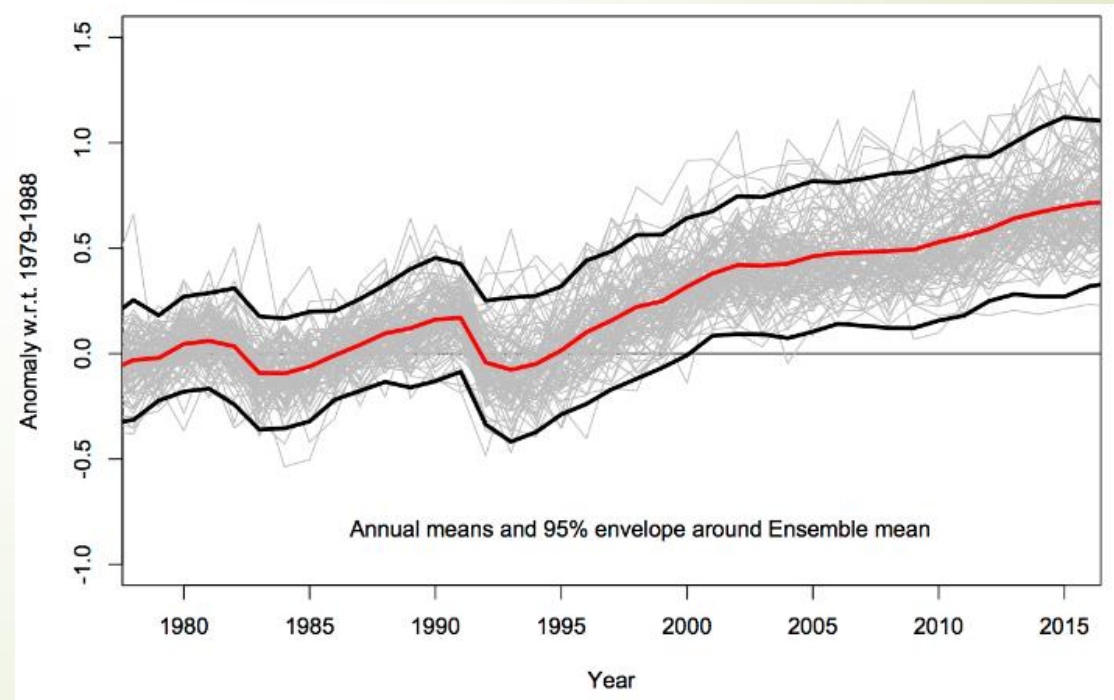
- Averaging over multiple predictions reduces the variance in the predictions.
 - ✓ Tend to improve the predictive ability **more for higher-variance methods**
 - ✓ Often used with tree induction, since trees tend to have **high variance but low bias**

$$E(X_i) = \mu \quad \text{Var}(X_i) = \sigma^2 \quad \text{for all } i = 1, 2, \dots, n$$

$$\bar{X} = \frac{X_1 + X_2 + \dots + X_n}{n} = \left(\frac{1}{n}\right) (X_1 + X_2 + \dots + X_n)$$

$$\text{Var}(\bar{X}) = \left(\frac{1}{n}\right)^2 \text{Var}(X_1 + X_2 + \dots + X_n) = \left(\frac{1}{n}\right)^2 (n\sigma^2) = \frac{\sigma^2}{n}$$

1



Ensemble Learning in Sklearn

➡ <https://scikit-learn.org/stable/modules/ensemble.html>

sklearn.ensemble: Ensemble Methods

The `sklearn.ensemble` module includes ensemble-based methods for classification, regression and anomaly detection.

User guide: See the [Ensemble methods](#) section for further details.

<code>ensemble.AdaBoostClassifier(...)</code>	An AdaBoost classifier.
<code>ensemble.AdaBoostRegressor([base_estimator, ...])</code>	An AdaBoost regressor.
<code>ensemble.BaggingClassifier([base_estimator, ...])</code>	A Bagging classifier.
<code>ensemble.BaggingRegressor([base_estimator, ...])</code>	A Bagging regressor.
<code>ensemble.ExtraTreesClassifier(...)</code>	An extra-trees classifier.
<code>ensemble.ExtraTreesRegressor([n_estimators, ...])</code>	An extra-trees regressor.
<code>ensemble.GradientBoostingClassifier([loss, ...])</code>	Gradient Boosting for classification.
<code>ensemble.GradientBoostingRegressor([loss, ...])</code>	Gradient Boosting for regression.
<code>ensemble.IsolationForest([n_estimators, ...])</code>	Isolation Forest Algorithm.
<code>ensemble.RandomForestClassifier(...)</code>	A random forest classifier.
<code>ensemble.RandomForestRegressor(...)</code>	A random forest regressor.
<code>ensemble.RandomTreesEmbedding(...)</code>	An ensemble of totally random trees.
<code>ensemble.StackingClassifier(estimators[, ...])</code>	Stack of estimators with a final classifier.
<code>ensemble.StackingRegressor(estimators[, ...])</code>	Stack of estimators with a final regressor.
<code>ensemble.VotingClassifier(estimators[, ...])</code>	Soft Voting/Majority Rule classifier for unfitted estimators.
<code>ensemble.VotingRegressor(estimators[, ...])</code>	Prediction voting regressor for unfitted estimators.
<code>ensemble.HistGradientBoostingRegressor(...)</code>	Histogram-based Gradient Boosting Regression Tree.
<code>ensemble.HistGradientBoostingClassifier(...)</code>	Histogram-based Gradient Boosting Classification Tree.

XGBoost

- XGBoost (extreme Gradient Boosting) is an advanced implementation of the gradient boosting (and random forest) algorithm .
 - XGBoost has proved to be a highly effective ML algorithm, extensively used in machine learning competitions like Kaggle.
 - High predictive power and is almost 10 times faster than the other gradient boosting techniques.

How to deal with Missing Value

XGBoost supports missing value by default. In tree algorithms, branch directions for missing values are learned during training. Note that the gblinear booster treats missing values as zeros.

<https://xgboost.readthedocs.io/en/latest/faq.html>






LightGBM

- LightGBM is a gradient boosting framework that uses tree based learning algorithms.
 - Maintained by Microsoft Corporation
 - Faster training speed and higher efficiency.
 - Lower memory usage.
 - Better accuracy.
 - Support of parallel and GPU learning.
 - Capable of handling large-scale data.

<https://lightgbm.readthedocs.io/en/latest/>

CatBoost

- CatBoost can automatically deal with **categorical variables** and does not require extensive data preprocessing like other machine learning algorithms.

	CatBoost		LightGBM		XGBoost		H2O	
	Tuned	Default	Tuned	Default	Tuned	Default	Tuned	Default
 Adult	0.26974	0.27298 +1.21%	0.27602 +2.33%	0.28716 +6.46%	0.27542 +2.11%	0.28009 +3.84%	0.27510 +1.99%	0.27607 +2.35%
 Amazon	0.13772	0.13811 +0.29%	0.16360 +18.80%	0.16716 +21.38%	0.16327 +18.56%	0.16536 +20.07%	0.16264 +18.10%	0.16950 +23.08%
 Click prediction	0.39090	0.39112 +0.06%	0.39633 +1.39%	0.39749 +1.69%	0.39624 +1.37%	0.39764 +1.73%	0.39759 +1.72%	0.39785 +1.78%
 KDD appetency	0.07151	0.07138 -0.19%	0.07179 +0.40%	0.07482 +4.63%	0.07176 +0.35%	0.07466 +4.41%	0.07246 +1.33%	0.07355 +2.86%
 KDD churn	0.23129	0.23193 +0.28%	0.23205 +0.33%	0.23565 +1.89%	0.23312 +0.80%	0.23369 +1.04%	0.23275 +0.64%	0.23287 +0.69%

<https://catboost.ai/docs/>

Application is Easy (调包侠)

- Most packages provide Scikit-Learn-Like API.
 - Can also use similar functions in Sklearn for cross validation, GridSearchCV ...

```
fit (X, y, sample_weight=None, base_margin=None, eval_set=None, eval_metric=None,  
early_stopping_rounds=None, verbose=True, xgb_model=None, sample_weight_eval_set=None,  
callbacks=None)
```

Fit gradient boosting classifier

Parameters

- **X** (*array_like*) – Feature matrix
- **y** (*array_like*) – Labels
- **sample_weight** (*array_like*) – instance weights

```
predict (data, output_margin=False, ntree_limit=None, validate_features=True, base_margin=None)
```

Predict with data.

```
predict_proba (data, ntree_limit=None, validate_features=True, base_margin=None)
```

Predict the probability of each data example being of a given class.

https://catboost.ai/docs/concepts/python-reference_catboostclassifier.html

https://xgboost.readthedocs.io/en/latest/python/python_api.html#xgboost.XGBClassifier.apply

Outline

- ▶ Ensemble Modeling
- ▶ Issues in Business Understanding
- ▶ Evidence and Naive Bayes Model

Expected Value

- The expected value computation provides a framework that is useful in organizing thinking about data-analytic problems
- The general form of an expected value calculation:

$$EV = p(o_1) \cdot v(o_1) + p(o_2) \cdot v(o_2) + p(o_3) \cdot v(o_3) \dots$$

- Examples for marketing revenue:

- ✓ Product Price: \$200; Product Cost: \$100; Targeting Cost: \$1
- ✓ The probability of response for consumer \mathbf{x} in general is $P_R(\mathbf{x})$
- ✓ Then $V_R = 200 - 100 - 1 = \$99$, $V_{NR} = -\$1$

$$\begin{aligned} \text{Expected benefit of targeting} &= p_R(\mathbf{x}) \cdot v_R + [1 - p_R(\mathbf{x})] \cdot v_{NR} \\ &= \\ &= p_R(\mathbf{x}) \cdot \$99 - [1 - p_R(\mathbf{x})] \cdot \$1 \end{aligned}$$

➔ Value of Target = $P_R(X) \cdot 100 - 1 > 0$

Different Marketing Applications

- Fundraising organizations need to select a “good” subset of the potential donors to target for a charity mailing
- It seems that fundraising targeting is similar to traditional market targeting, but is actually quite different
 - ✓ The response of fund raising marketing can vary — some people might donate \$100 while others might donate \$1
 - ✓ $v_R(\mathbf{x})$ is the value we get from a response from consumer \mathbf{x} and $v_{NR}(\mathbf{x})$ is the value we get if consumer \mathbf{x} does not respond

~~Expected benefit of targeting = $p_R(\mathbf{x}) \cdot v_R + [1 - p_R(\mathbf{x})] \cdot v_{NR}$~~



~~Expected benefit of targeting = $p(R | \mathbf{x}) \cdot v_R + [1 - p(R | \mathbf{x})] \cdot v_{NR}$~~



$$\text{Expected benefit of targeting} = p(R | \mathbf{x}) \cdot v_R(\mathbf{x}) + [1 - p(R | \mathbf{x})] \cdot v_{NR}(\mathbf{x})$$

Decomposing the Business Problem

- The expected value framework provides a helpful decomposition of possibly complicated business problems into sub-problems that we understand better
 - ✓ Looking at historical data on consumers who have been targeted, we can use regression modeling to estimate how much a consumer will respond/donate using regression model, i.e., $d_R(\mathbf{x})$
 - ✓ We can model the probability of response for consumer \mathbf{x} , i.e., $p(R | \mathbf{x})$, by mining historical data using one of the many techniques we have discussed
 - ✓ Let c be the mailing/targeting cost

$$\text{Expected benefit of targeting} = p(R | \mathbf{x}) \cdot (d_R(\mathbf{x}) - c) + [1 - p(R | \mathbf{x})] \cdot (-c)$$

- ✓ If we want to get positive benefit, then

$$p(R | \mathbf{x}) \cdot (d_R(\mathbf{x}) - c) + [1 - p(R | \mathbf{x})] \cdot (-c) > 0$$

$$\Rightarrow p(R | \mathbf{x}) \cdot d_R(\mathbf{x}) - p(R | \mathbf{x}) \cdot c - c + p(R | \mathbf{x}) \cdot c > 0$$

$$\Rightarrow p(R | \mathbf{x}) \cdot d_R(\mathbf{x}) > c$$

The Expected Value framework for Churn Prediction

- We use our data to determine which customers would be the most likely to defect shortly after their contracts expire, and target the special retention offer to some appropriate subset of our customer base
- ✓ As with the donation case study, we might represent the expected benefit of targeting a customer with the special offer as

$$\text{Expected benefit of targeting} = p(S \mid \mathbf{x}) \cdot v_S(\mathbf{x}) + [1 - p(S \mid \mathbf{x})] \cdot v_{NS}(\mathbf{x})$$

where,

$p(S \mid \mathbf{x})$ is the probability that the customer will **S**tay with the company after being targeted

$v_S(\mathbf{x})$ is the value we get if consumer \mathbf{x} stays with the company

$v_{NS}(\mathbf{x})$ is the value we get if consumer \mathbf{x} does not stay (defects or churns).

- ✓ If we do not target, some customer stays anyway
- ✓ In fact, we don't want to target those with the highest value if they were to stay. We want to target those where we would lose the most value if they were to leave

Assessing the Influence of the Incentive

- ▶ We indicate by conditioning the probability of staying on the two possibilities (target, T , or not target, not T), i.e., $p(S \mid \mathbf{x}, T)$ and $p(S \mid \mathbf{x}, \text{not}T)$

- ✓ The expected benefit of targeting is:

$$EB_T(\mathbf{x}) = p(S \mid \mathbf{x}, T) \cdot (u_s(\mathbf{x}) - c) + [1 - p(S \mid \mathbf{x}, T)] \cdot (u_{NS}(\mathbf{x}) - c)$$

- ✓ The expected benefit of not targeting is:

$$EB_{\text{not}T}(\mathbf{x}) = p(S \mid \mathbf{x}, \text{not}T) \cdot (u_s(\mathbf{x}) - c) + [1 - p(S \mid \mathbf{x}, \text{not}T)] \cdot (u_{NS}(\mathbf{x}) - c)$$

where,

$u_s(\mathbf{x})$ is the profit from customer \mathbf{x} if she stays, not including the incentive cost c

$u_{NS}(\mathbf{x})$ is the profit from customer \mathbf{x} if she leaves, not including the incentive cost c

Value of Targeting

- ▶ We try to maximize value of targeting (VT) as

$$\begin{aligned} VT &= EB_T(\mathbf{x}) - EB_{notT}(\mathbf{x}) \\ &= p(S \mid \mathbf{x}, T) \cdot u_s(\mathbf{x}) - p(S \mid \mathbf{x}, notT) \cdot u_s(\mathbf{x}) - c \\ &= [p(S \mid \mathbf{x}, T) - p(S \mid \mathbf{x}, notT)] \cdot u_s(\mathbf{x}) - c \\ &= \Delta(p) \cdot u_s(\mathbf{x}) - c \end{aligned}$$

where $\Delta(p)$ is the difference in the predicted probabilities of staying, depending on whether the customer is targeted or not

- ▶ We want to target those customers with the greatest change in their probability of staying, moderated by their value if they were to stay!

Data Preparation

$$VT = p(S \mid \mathbf{x}, T) \cdot u_s(\mathbf{x}) - p(S \mid \mathbf{x}, \text{not}T) \cdot u_s(\mathbf{x}) - c$$

- We can easily estimate the probability of staying on not targeting $p(S \mid \mathbf{x}, \text{not}T)$
 - ✓ Assuming that nothing substantial has changed in the business environment
 - ✓ But the coming of iPhone would be an event changing the business
- We usually do not have data for estimating $p(S \mid \mathbf{x}, T)$ for a new offer
 - ✓ Simplification: $p(S \mid \mathbf{x}, T) = 1?$
 - ✓ Using a “proxy” for the target label of interest: data collected from a similar, but not identical, offer in the past
 - ✓ “Data and Data Science Capability as a Strategic Asset”: Invest in data on how people will respond to this offer

Selection Bias

- Selection bias—the data were not selected randomly from the population to which you intend to apply the model, but instead were biased in some way
 - ✓ When predicting donation, data are usually from past donations—from the individuals who did respond in the past
 - ✓ Modeling creditworthiness based on the experience with past credit customers: those are likely the people whom you had deemed to be creditworthy in the past!
 - ✓ However, we need to apply the model to the general population

Outline

- ▶ Ensemble Modeling
- ▶ Issues in Business Understanding
- ▶ Evidence and Naive Bayes Model

Example: Online Ads (1)

- Advertising campaign for upscale hotel chain
 - ✓ We want to run a campaign getting more bookings per dollar spent on ad impressions
- Target variable: Whether the consumer booked room within one week after having seen the advertisement
- Prediction: class probability estimation
 - ✓ The probability that a consumer will book a room after seeing an ad
- Targeting: target some subset of the highest probability consumers as our budget allows
- Features: the set of content pieces that he has viewed during the last month
 - ✓ Thousands of pages of different content: financial news, tour, entertainment ...
 - ✓ Based on browser cookie

Example: Online Ads (2)

- ▶ We treat the features of instances as **evidence** for or against different values of the **target**

- ✓ Represent the strength by probability

- ✓ Given 100 people who have bought beer before, 80 are male and 20 are female

A consumer bought beer ➡ This consumer is male since $p(\text{male} \mid \text{beer}) = 0.8 > p(\text{female} \mid \text{beer}) = 0.2$

- ▶ A user's view patterns are the evidence for/against whether he will response to the online advertisement of upscale hotel

- ✓ What is the chance that in our training data we have seen a consumer with exactly the same visiting patterns as a consumer we will see in the future?

$p(\text{target} \mid \text{web}_1, \text{web}_6, \text{web}_{13}, \dots, \text{web}_{7098}, \dots)$

- ✓ We will consider the different pieces of evidence separately, and then combine the evidence

$p(\text{target} \mid \text{web}_1), p(\text{target} \mid \text{web}_6), p(\text{target} \mid \text{web}_{13}) \rightarrow p(\text{target} \mid \text{web}_1, \text{web}_6, \text{web}_{13}, \dots, \text{web}_{7098}, \dots)$

Example: Play Golf Or Not

Whether or not play the Golf?



#	Outlook	Temp.	Humidity	Windy	Play
1	Sunny	Hot	High	False	No
2	Sunny	Hot	High	True	No
3	Overcast	Hot	High	False	Yes
4	Rainy	Mild	High	False	Yes
5	Rainy	Cool	Normal	False	Yes
6	Rainy	Cool	Normal	True	No
7	Overcast	Cool	Normal	True	Yes
8	Sunny	Mild	High	False	No
9	Sunny	Cool	Normal	False	Yes
10	Rainy	Mild	Normal	False	Yes
11	Sunny	Mild	Normal	True	Yes
12	Overcast	Mild	High	True	Yes
13	Overcast	Hot	Normal	False	Yes
14	Rainy	Mild	High	True	No

Joint Probability and Independence

➤ Given that

$$p(AB) = p(A) \cdot (B|A) = p(B) \cdot (A|B)$$

✓ Example: **A** = fire alarm rings, **B** = Jack calls 119

➤ Joint probability when **A** and **B** are independent with each other

$$p(AB) = p(A) \cdot (B)$$

✓ **A** and **B** are independent: $p(A|B) = p(A)$ and $p(B|A) = p(B)$

✓ Example: throw one equal dice, **A** = get 6 for the first time, **B** = get 6 for the second time



Bayes' Rule

- ➡ Given that $p(B|A) = \frac{p(AB)}{p(A)}$ and $p(AB) = p(A) \cdot (B|A) = p(B) \cdot (A|B)$,

$$p(B | A) = \frac{p(A | B) \cdot p(B)}{p(A)} \quad (\text{Bayes' Rule})$$

- ➡ Renaming **B** with **H** for hypothesis, and **A** with **E** for evidence

$$p(H | E) = \frac{p(E | H) \cdot p(H)}{p(E)}$$

- ✓ Hypothesis **H** could be “this customer needs (not) to be targeted ”
- ✓ Evidence **E** = $\langle e_1, e_2, \dots, e_k \rangle$ could be “view pattern of the customer” ,
i.e., e_1 = the user visited web₁

Bayes Rule for Classification

$$p(C = c \mid \mathbf{E}) = \frac{p(\mathbf{E} \mid C = c) \cdot p(C = c)}{p(\mathbf{E})}$$

- $p(C = c \mid \mathbf{E})$ is the **posterior probability**
 - ✓ The probability that the target variable C takes on the class of interest c after taking the evidence E
- $p(C = c)$ is the **prior probability** of the class c
 - ✓ The probability we would assign to the class before seeing any evidence (1. expertise; 2. use class priors from data)
- $p(\mathbf{E} \mid C = c)$ is the likelihood of seeing the evidence \mathbf{E} when the class $C = c$
- $p(\mathbf{E})$ is the likelihood of the evidence

Simplified Bayes Rule for Classification

$$p(C = c \mid \mathbf{E}) = \frac{p(\mathbf{E} \mid C = c) \cdot p(C = c)}{p(\mathbf{E})}$$

- We can firstly just focus on the calculation of $p(\mathbf{E} \mid C = c)$
 - ✓ Both $p(\mathbf{E} \mid C = c)$ and $p(\mathbf{E})$ are difficult to measure
 - ✓ Probability assignment: $p(\mathbf{E})$ is required to calculate a probability, which can also rank instances (e.g., estimating those that are most likely to respond to our advertisement)
 - ✓ Ranking/Classification only : Select the class c that can maximize $p(\mathbf{E} \mid C = c) \cdot p(C = c)$ /rank by $p(\mathbf{E} \mid C = c) \cdot p(C = c)$ and **ignore** $p(\mathbf{E})$
- However, it is hard to compute $p(\mathbf{E} \mid C = c)$ from training data
 - ✓ We seldom see a specific example in the training data that exactly matches a given $\mathbf{E} = \langle e_1, e_2, \dots, e_k \rangle$ in our testing data

Difficulties of Applying Bayes Rule*

- Compute the $P(E | C=c)$ for different c

$$p(E|C = c) = p(e_1 \wedge e_2 \wedge \dots \wedge e_k|c)$$

- Each observation in practice is a **complex** combination of **many** individual events
 - Not enough examples for estimating probability of observation

$E_i = (e_1, e_2, e_3, e_4)$ with :

- e_1 : Outlook = Sunny
- e_2 : Temp. = Hot
- e_3 : Humidity = High
- e_4 : Windy = False

$$P(E | C='No') = 1,$$

$$P(E | C='Yes') = ?$$

Whether or not play the Golf?

#	Outlook	Temp.	Humidity	Windy	Play
1	Sunny	Hot	High	False	No
2	Sunny	Hot	High	True	No
3	Overcast	Hot	High	False	Yes
4	Rainy	Mild	High	False	Yes
5	Rainy	Cool	Normal	False	Yes
6	Rainy	Cool	Normal	True	No
7	Overcast	Cool	Normal	True	Yes
8	Sunny	Mild	High	False	No
9	Sunny	Cool	Normal	False	Yes
10	Rainy	Mild	Normal	False	Yes
11	Sunny	Mild	Normal	True	Yes
12	Overcast	Mild	High	True	Yes
13	Overcast	Hot	Normal	False	Yes
14	Rainy	Mild	High	True	No

Difficulties of Applying Bayes Rule*

➤ In general,

$$p(E|C = c) = p(e_1 \wedge e_2 \wedge \dots \wedge e_k | c)$$

$$= p(e_1 | c) \cdot p(e_2 | e_1 \wedge c) \cdot \dots \cdot p(e_k | e_1 \wedge e_2 \wedge \dots \wedge e_{k-1} \wedge c)$$

➤ Each observation in practice is a **complex** combination of **many** individual events

➤ Not enough examples for estimating probability of observation. E.g., E=(Sunny, Hot, High, Not strong)

$E_i = (e_1, e_2, e_3, e_4)$ with :

- e_1 : Outlook = Sunny
- e_2 : Temp.= Hot
- e_3 : Humidity= High
- e_4 : Windy= False (Not strong)

$$P(e_1 | C='No') = 3/5$$

$$P(e_1 | C='Yes') = 2/9$$

$$P(E | C='No') = 1, P(E | C='Yes') = ?$$

Whether or not play the Golf?

#	Outlook	Temp.	Humidity	Windy	Play
1	Sunny	Hot	High	False	No
2	Sunny	Hot	High	True	No
3	Overcast	Hot	High	False	Yes
4	Rainy	Mild	High	False	Yes
5	Rainy	Cool	Normal	False	Yes
6	Rainy	Cool	Normal	True	No
7	Overcast	Cool	Normal	True	Yes
8	Sunny	Mild	High	False	No
9	Sunny	Cool	Normal	False	Yes
10	Rainy	Mild	Normal	False	Yes
11	Sunny	Mild	Normal	True	Yes
12	Overcast	Mild	High	True	Yes
13	Overcast	Hot	Normal	False	Yes
14	Rainy	Mild	High	True	No

“Naive” Assumptions *

- In general,

$$\begin{aligned} p(E|C = c) &= p(e_1 \wedge e_2 \wedge \cdots \wedge e_k | c) \\ &= p(e_1 | c) \cdot p(e_2 | e_1 \wedge c) \cdots p(e_k | e_1 \wedge e_2 \wedge \cdots \wedge e_{k-1} \wedge c) \end{aligned}$$

- Conditionally independent

- ✓ Example: following **A** and **B** are conditionally independent given **C**

A = Pony calls 119, **B** = Jack calls 119, **C** = Fire alarm rings,

then, $p(A|BC) = p(A|C)$, $p(B|AC) = p(B|C)$ and $p(AB|C) = p(A|C) \cdot p(B|C)$

- We make a “**Naive**” **assumptions** that attributes are conditionally independent, given the class label, such that $p(e_2 | e_1 \wedge c) = p(e_2 | c)$, ..., $p(e_k | e_1 \wedge e_2 \wedge \cdots \wedge e_{k-1} \wedge c) = p(e_k | c)$

$$\begin{aligned} p(\mathbf{E} | c) &= p(e_1 \wedge e_2 \wedge \cdots \wedge e_k | c) \\ &= p(e_1 | c) \cdot p(e_2 | c) \cdots p(e_k | c) \end{aligned}$$

Naive Bayes Classifier

- Basis of Naive Bayes classifier is as follows,

$$p(c \mid \mathbf{E}) = \frac{p(e_1 \mid c) \cdot p(e_2 \mid c) \cdots p(e_k \mid c) \cdot p(c)}{p(\mathbf{E})}$$

- We can further compute the likelihood of the evidence $p(\mathbf{E})$ based on as well

$$\begin{aligned} p(\mathbf{E}) &= p(\mathbf{E} \wedge c_0) + p(\mathbf{E} \wedge c_1) \\ &= p(\mathbf{E} \mid c_0) \cdot p(c_0) + p(\mathbf{E} \mid c_1) \cdot p(c_1) = p(e_1 \mid c_0) \cdot p(e_2 \mid c_0) \cdots p(e_k \mid c_0) \cdot p(c_0) \\ &\quad + p(e_1 \mid c_1) \cdot p(e_2 \mid c_1) \cdots p(e_k \mid c_1) \cdot p(c_1) \end{aligned}$$

- Then the complete form is

$$p(c_0 \mid \mathbf{E}) = \frac{p(e_1 \mid c_0) \cdot p(e_2 \mid c_0) \cdots p(e_k \mid c_0) \cdot p(c_0)}{p(e_1 \mid c_0) \cdot p(e_2 \mid c_0) \cdots p(e_k \mid c_0) \cdot p(c_0) + p(e_1 \mid c_1) \cdot p(e_2 \mid c_1) \cdots p(e_k \mid c_1) \cdot p(c_1)}$$

Example: Naive Bayes classifier

$E = (\text{Sunny, Cool, High, Strong?} - \text{True})$

Whether or not play the Golf?

Probability that we can play the Game!

$P(\text{Outlook} = \text{Sunny} \mid \text{Play} = \text{Yes})$	2 / 9
$P(\text{Temperature} = \text{Cool} \mid \text{Play} = \text{Yes})$	3 / 9
$P(\text{Humidity} = \text{High} \mid \text{Play} = \text{Yes})$	3 / 9
$P(\text{Wind} = \text{Strong} \mid \text{Play} = \text{Yes})$	3 / 9
$P(\text{Play} = \text{Yes})$	9 / 14

Probability that we avoid to Play Golf!

$P(\text{Outlook} = \text{Sunny} \mid \text{Play} = \text{No})$	3 / 5
$P(\text{Temperature} = \text{Cool} \mid \text{Play} = \text{No})$	1 / 5
$P(\text{Humidity} = \text{High} \mid \text{Play} = \text{No})$	4 / 5
$P(\text{Wind} = \text{Strong} \mid \text{Play} = \text{No})$	3 / 5
$P(\text{Play} = \text{No})$	5 / 14

$$p(c \mid \mathbf{E}) = \frac{p(e_1 \mid c) \cdot p(e_2 \mid c) \cdots p(e_k \mid c) \cdot p(c)}{p(\mathbf{E})}$$

$P(E \mid \text{Play} = \text{Yes}) P(\text{Play} = \text{Yes}) = [(2/9) * (3/9) * (3/9) * (3/9)] * (9/14) = 0.0053$

$P(E \mid \text{Play} = \text{No}) P(\text{Play} = \text{No}) = [(3/5) * (1/5) * (4/5) * (3/5)] * (5/14) = 0.0206$

#	Outlook	Temp.	Humidity	Windy	Play
1	Sunny	Hot	High	False	No
2	Sunny	Hot	High	True	No
3	Overcast	Hot	High	False	Yes
4	Rainy	Mild	High	False	Yes
5	Rainy	Cool	Normal	False	Yes
6	Rainy	Cool	Normal	True	No
7	Overcast	Cool	Normal	True	Yes
8	Sunny	Mild	High	False	No
9	Sunny	Cool	Normal	False	Yes
10	Rainy	Mild	Normal	False	Yes
11	Sunny	Mild	Normal	True	Yes
12	Overcast	Mild	High	True	Yes
13	Overcast	Hot	Normal	False	Yes
14	Rainy	Mild	High	True	No

(Dis)Advantages of Naive Bayes

$$p(c_0 | \mathbf{E}) = \frac{p(e_1 | c_0) \cdot p(e_2 | c_0) \cdots p(e_k | c_0) \cdot p(c_0)}{p(e_1 | c_0) \cdot p(e_2 | c_0) \cdots p(e_k | c_0) \cdot p(c_0) + p(e_1 | c_1) \cdot p(e_2 | c_1) \cdots p(e_k | c_1) \cdot p(c_1)}$$

- Naive Bayes classifier
 - ✓ is a simple classifier
 - ✓ is very efficient in terms of storage space and computation time
 - ✓ performs surprisingly well for text classification
 - ✓ is an “**incremental learner**” (Another one we learned is KNN)
- Naive Bayes classifier provide **non-accurate** class probability estimation
 - ✓ Due to the naïve assumption
 - ✓ Usually used for classification or ranking

Incremental Learner

- Incremental learner models can be used to tackle large scale classification problems for which the full training set might not fit in memory.

<code>naive_bayes.BernoulliNB([alpha, binarize, ...])</code>	Naive Bayes classifier for multivariate Bernoulli models.
<code>naive_bayes.CategoricalNB([alpha, ...])</code>	Naive Bayes classifier for categorical features
<code>naive_bayes.ComplementNB([alpha, fit_prior, ...])</code>	The Complement Naive Bayes classifier described in Rennie et al.
<code>naive_bayes.GaussianNB([priors, var_smoothing])</code>	Gaussian Naive Bayes (GaussianNB)
<code>naive_bayes.MultinomialNB([alpha, ...])</code>	Naive Bayes classifier for multinomial models

<code>fit(self, X, y[, sample_weight])</code>	Fit Naive Bayes classifier according to X, y
<code>get_params(self[, deep])</code>	Get parameters for this estimator.
<code>partial_fit(self, X, y[, classes, sample_weight])</code>	Incremental fit on a batch of samples.
<code>predict(self, X)</code>	Perform classification on an array of test vectors X.
<code>predict_log_proba(self, X)</code>	Return log-probability estimates for the test vector X.
<code>predict_proba(self, X)</code>	Return probability estimates for the test vector X.
<code>score(self, X, y[, sample_weight])</code>	Return the mean accuracy on the given test data and labels.
<code>set_params(self, **params)</code>	Set the parameters of this estimator.

5% Lab Quiz + 2% Bonus Quiz

- **Deadline:** 17:59 p.m., April. 24, 2020
- **Upload** the **answer worksheet** and the accomplished **Python files** to the **Blackboard**
- You may submit **unlimited times** but only the **LAST** submission will be considered
- **Only the answers in answer sheet** will be referred for grading
- Note: **MUST attach ALL** the required files in every submission/resubmission, otherwise other files will be missing.

Extended Deadline for Individual Project

- Electronic submission will be required on following materials with on/before ~~Sat. May 2~~ **Sun. May 10, 2020 at 23:59**
- Top 3 students will be invited to share their ~~experience during the last lecture on May 8~~ **at 10: 00 AM on Fri. May 15, 2020**
- The invitation for sharing will be sent before ~~Tue. May 5~~ **Tue. May 12, 2020** at 23:59 and the acceptance of invitation should be confirmed before ~~Wed. May 6~~ **Wed. May 13, 2020**

Extended Deadline for Group Project

- Report (Electronic Copy Due: ~~Sunday May 3~~ **Sunday May 10, 2020** at 23:59)
- Presentation (Due: ~~Friday May 8~~ **Friday May 15, 2020, Last lecture**)
- *Slides, Dataset & Code* (**Due: ~~Friday May 8~~ Friday May 15, 2020 at 23:59**)