

Enhance Model Performance

1

Dr. Yi Long (Neal)

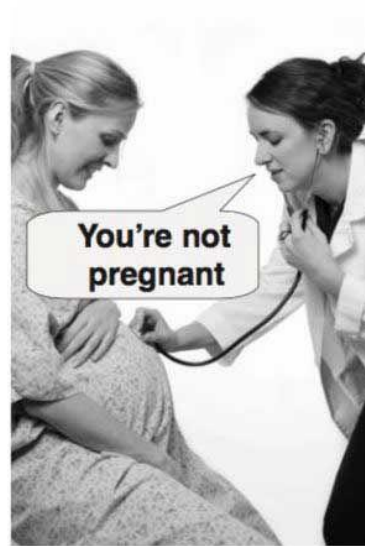
Most contents (text or images) of course slides are from the following textbook
Provost, Foster, and Tom Fawcett. Data Science for Business: What you need to know about data mining and data-analytic thinking. " O'Reilly Media, Inc.", 2013

Confusion Matrix and Errors

Type I error
(false positive)



Type II error
(false negative)



		Reality	
		True	False
Measured or Perceived	True	Correct 😊	Type 1 error False Positive
	False	Type 2 error False Negative	Correct 😊

Outline

- Data Clean
 - Data Quality Issues
 - Imbalance
- Feature Engineering
- Model selection & Parameter Tuning
- Evaluation Methodology
- Ensembles of Models

Outline

- Data Clean
 - Data Quality Issues
 - Imbalance
- Feature Engineering
- Ensembles of Models
- Quiz

Data Quality – Dimensions



- **Completeness:** are values/records missing?
- **Conformity:** does the data match the rules?
- **Consistency:** is the data consistent across various data stores?
- **Uniqueness:** is there duplicated data?
- **Timeliness:** does the data represent reality from the required point in time?
- **Accuracy:** the degree to which the data represents reality

Conformity

- **Data-Type Constraints:** values in a particular column must be of a particular datatype, e.g., boolean, numeric, date, etc.
 - Example: a string '24' in column – 'age'
- **Range Constraints:** number or date should fall within a certain range. Example: age, date of birth, human weight/height
 - A customer with column - 'age' = 201 or -20
- **Mandatory Constraints:** certain columns cannot be empty or date column must follow MM/dd/yyyy or dd/MM/yyyy in text files
 - "Hahaha" in column - contact: contact communication type (categorical: "cellular","telephone")
- **Consistency Constraints:** different attributes about the same information
 - Birth information and age, retired information and age ...

Identify/Fix Structural Errors

► Obvious errors/typos

- ✓ Change the error value of column-"marital" from "sungle" to "single"
- ✓ Check the data type: '38' to 38 for column 'age'

► Inferred from of auxiliary information (consistency)

- ✓ Bank-marketing: Column: previous, poutcome, and Pday

► Unsolvable errors

- ✓ Treat as missing value
- ✓ E.g. age with value -10 or 208

M	N	O	
pdays	previous	poutcome	p.
999	0	nonexistent	
999	0	nonexistent	
999	0	nonexistent	
999	0	nonexistent	
999	0	nonexistent	
999	2	failure	
999	0	nonexistent	
999	0	nonexistent	
999	1	failure	
999	0	nonexistent	

Outliers

- **Detection:** in statistics, an outlier is a data point that differs significantly from other observations.
 - Tukey's fences: if Q_1 and Q_3 are the lower and upper quartiles respectively, then an outlier are any observation outside the range ($k=1.5$ or 3):

$$[Q_1 - k(Q_3 - Q_1), Q_3 + k(Q_3 - Q_1)]$$

- Mean and Standard Deviation Method , Modified Thompson Tau test ...
- **How to handle**
 - Drop outliers
 - Treat as missing

Handle Missing Value

- **Fill with domain knowledge**

- ✓ For the missing 'job' for customers older than 60, just fill "retired"

- **Fill by statistics**

- ✓ Using Mean/Median/Most Frequent/Most Recent Values

- **Fill by regression/classification (e.g., KNN)**

- ✓ Infer gender from name, infer height from weight, Infering jobs from age/education

- **Fill with flag (NULL)**

- ✓ Flag missing data for future process (numpy.nan)

- **Drop rows/columns**

Handle Missing Value in Python

<https://scikit-learn.org/stable/modules/classes.html#module-sklearn.impute>

sklearn.impute: Impute

Transformers for missing value imputation

User guide: See the [Imputation of missing values](#) section for further details.

<code>impute.SimpleImputer</code> ([missing_values, ...])	Imputation transformer for completing missing values.
<code>impute.IterativeImputer</code> ([estimator, ...])	Multivariate imputer that estimates each feature from all the others.
<code>impute.MissingIndicator</code> ([missing_values, ...])	Binary indicators for missing values.
<code>impute.KNNImputer</code> ([missing_values, ...])	Imputation for completing missing values using k-Nearest Neighbors.

<https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.interpolate.html>

pandas.DataFrame.interpolate

`DataFrame.interpolate`(method='linear', axis=0, limit=None, inplace=False, limit_direction='forward', limit_area=None, downcast=None, **kwargs)

[\[source\]](#)

Interpolate values according to different methods.

Please note that only `method='linear'` is supported for DataFrames/Series with a MultiIndex.

method : {'linear', 'time', 'index', 'values', 'nearest', 'zero', 'slinear', 'quadratic', 'cubic', 'barycentric', 'krogh', 'polynomial', 'spline', 'piecewise_polynomial', 'from_derivatives', 'pchip', 'akima'}

<https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.fillna.html>

pandas.DataFrame.fillna

`DataFrame.fillna`(self, value=None, method=None, axis=None, inplace=None, limit=None, downcast=None)

Fill NA/NaN values using the specified method.

Drop Columns or Records

➤ Drop/filter rows (records)

- ✓ Missing records (missing values in a column rarely happen and occur at random)
- ✓ Records with certain conditions

➤ Drop columns (features)

- ✓ Columns that are unnecessary for the target (favorite color for issuing credit card)
- ✓ Columns with too many random missing value
- ✓ Columns with almost the same value for all records

pandas.DataFrame.dropna

`DataFrame.dropna(self, axis=0, how='any', thresh=None, subset=None, inplace=False)`

Remove missing values.

See the [User Guide](#) for more on which values are considered missing, and how to work with

Parameters: `axis` : {0 or 'index', 1 or 'columns'}, default 0

Determine if rows or columns which contain missing values are removed

- 0, or 'index' : Drop rows which contain missing values.
- 1, or 'columns' : Drop columns which contain missing value.

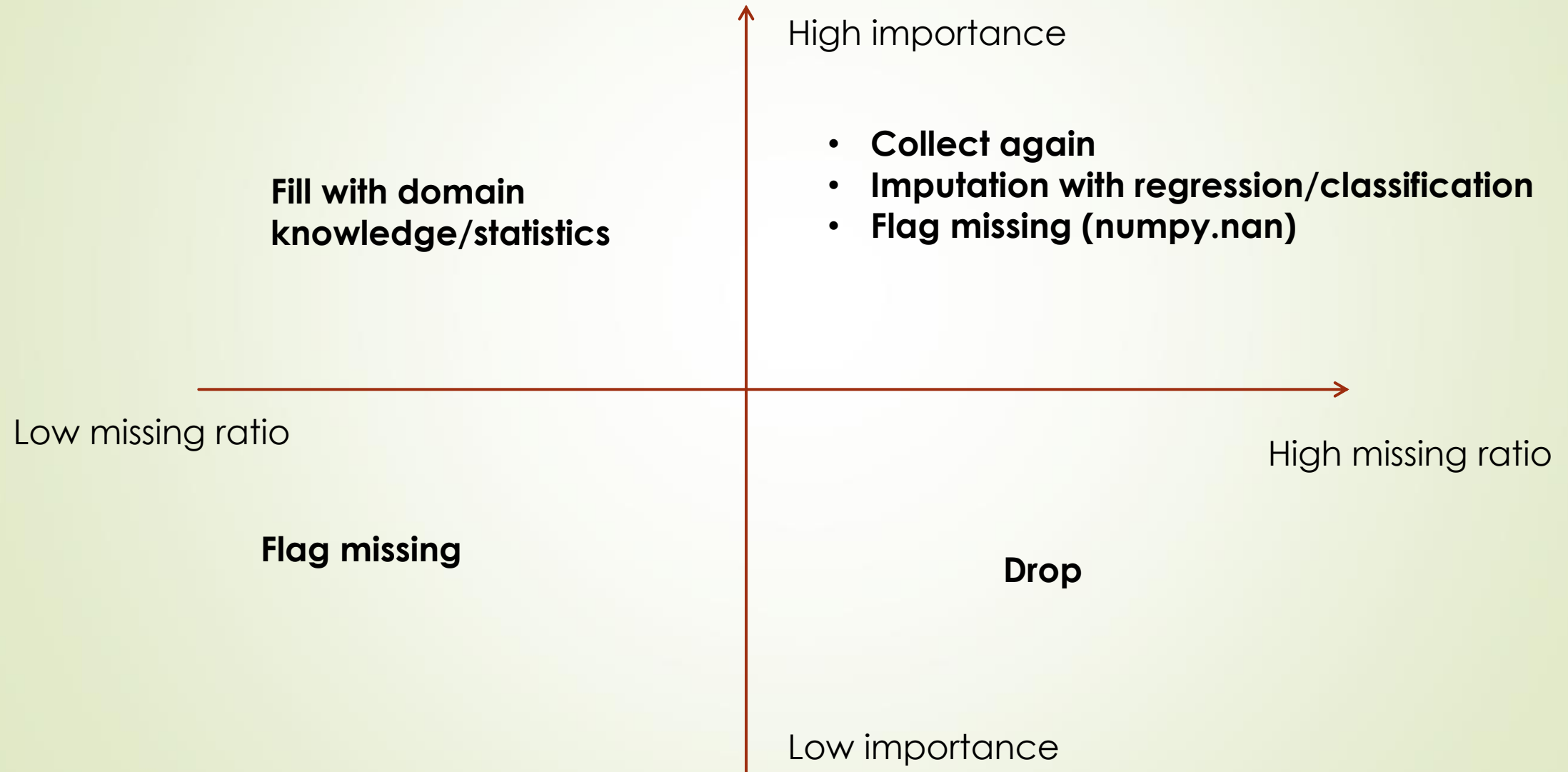
Changed in version 1.0.0: Pass tuple or list to drop on multiple axes. Only

`how` : {'any', 'all'}, default 'any'

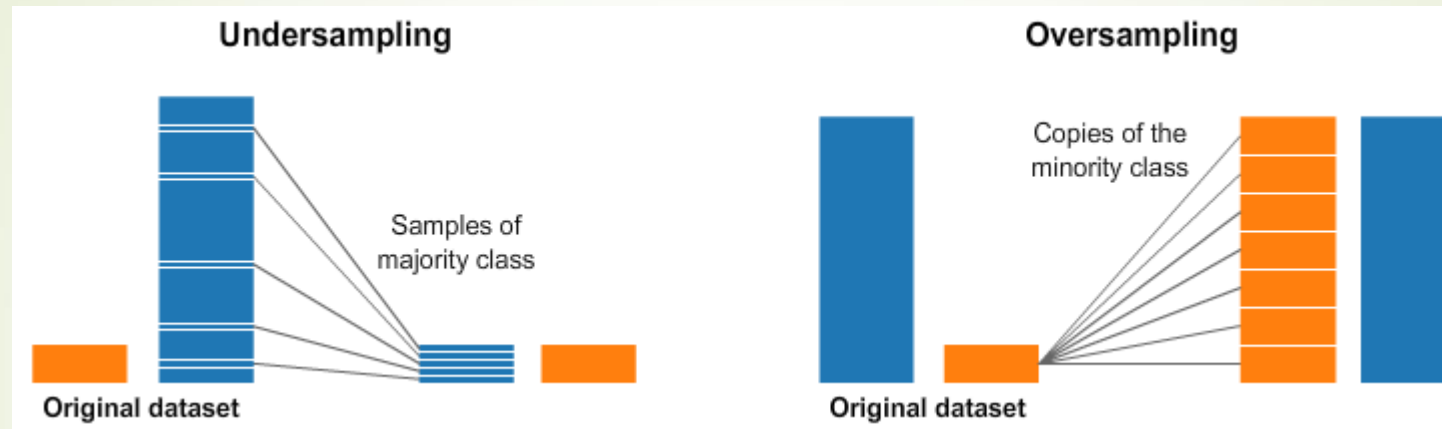
Determine if row or column is removed from DataFrame, when we have

- 'any' : If any NA values are present, drop that row or column.
- 'all' : If all values are NA, drop that row or column.

Imputation or Drop



Imbalanced Data



 imbalanced-learn

☰ User Guide

1. Introduction
2. Over-sampling
3. Under-sampling
4. Combination of over- and under-sampling
5. Ensemble of samplers

- 2.1. A practical guide
 - 2.1.1. Naive random over-sampling
 - 2.1.2. From random over-sampling to SMOTE and ADASYN
 - 2.1.3. Ill-posed examples
 - 2.1.4. SMOTE variants
- 2.2. Mathematical formulation
 - 2.2.1. Sample generation
 - 2.2.2. Multi-class management

Class Weight in Loss Function

https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

class_weight : dict or 'balanced', default=None

Weights associated with classes in the form `{class_label: weight}`. If not given, all classes are supposed to have weight one.

The "balanced" mode uses the values of `y` to automatically adjust weights inversely proportional to class frequencies in the input data as `n_samples / (n_classes * np.bincount(y))`.

Note that these weights will be multiplied with `sample_weight` (passed through the fit method) if `sample_weight` is specified.

$$\text{TrainLoss}(\mathbf{w}) = \text{loss}(f, (\mathbf{x}_1, y_1)) + \text{loss}(f, (\mathbf{x}_2, y_2)) + \dots + \text{loss}(f, (\mathbf{x}_n, y_n))$$



$$\text{TrainLoss}(\mathbf{w}) = c_1 * \text{loss}(f, (\mathbf{x}_1, y_1)) + c_2 * \text{loss}(f, (\mathbf{x}_2, y_2)) + \dots + c_n * \text{loss}(f, (\mathbf{x}_n, y_n))$$

Outline

- Data Preparation
 - Data Quality Issues
 - Imbalance
- Feature Engineering
- Ensembles of Models
- Quiz

Approaches for Feature Engineering

- **Feature scaling**
- **Feature encoding (categorical)**
- **Feature discretization (continuous)**
- **Feature generation**
- **Feature transformation**
- **Feature selection**
- Feature reduction (recommendation)
- Feature extraction (text mining)

Feature Scaling

- Feature scaling is a method used to normalize the range of independent features of data (also known as data normalization)
 - Since the range of values of raw data varies widely, in some machine learning algorithms, objective functions will not work properly without normalization.
 - E.g.: many models built on Euclidean distance between two examples



$\text{Dist}(A,B) = 0.3$

$\text{Dist}(A,C) = 5$

A(1.7m, 160 pounds)

B(1.4m, 160 pounds)

C(1.7m, 165 pounds)

$\text{max}(\text{height}) = 2 \text{ m}$
 $\text{min}(\text{height}) = 0.8 \text{ m}$

$\text{max}(\text{weight}) = 200 \text{ pounds}$
 $\text{min}(\text{weight}) = 50 \text{ pounds}$

Feature Scaling in Sklearn

- Standardization (StandardScaler) : zero mean and unit variance

$$x_{scaled} = \frac{x - \text{mean}(X)}{\text{std}(X)}$$

- Max-min scaler (MinMaxScaler/ MaxAbsScaler): scaling features to a range

$$x_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}} \quad / \quad x_{scaled} = \frac{x}{\max(\text{abs}(X))}$$

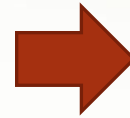
- Normalization (Normalizer): scaling to unit length (p=1 or 2, for L1/L2-norm)

$$x_{scaled} = \frac{x}{\|X\|_p}$$

Feature Engineering in Sklearn

fit(X)

learn parameters like mean,max,min, std



transform(X)

scaling with learned parameters

fit_transform(X): Fit to data, then transform it.

```
In [9]: X=[[2,200],
...:      [0.8,50],
...:      [1.7,160],
...:      [1.4,160],
...:      [1.7,165]]

In [10]: scaler = preprocessing.MinMaxScaler()

In [11]: scaler.fit_transform(X)
Out[11]:
array([[1.         , 1.         ],
       [0.         , 0.         ],
       [0.75        , 0.73333333],
       [0.5         , 0.73333333],
       [0.75        , 0.76666667]])
```

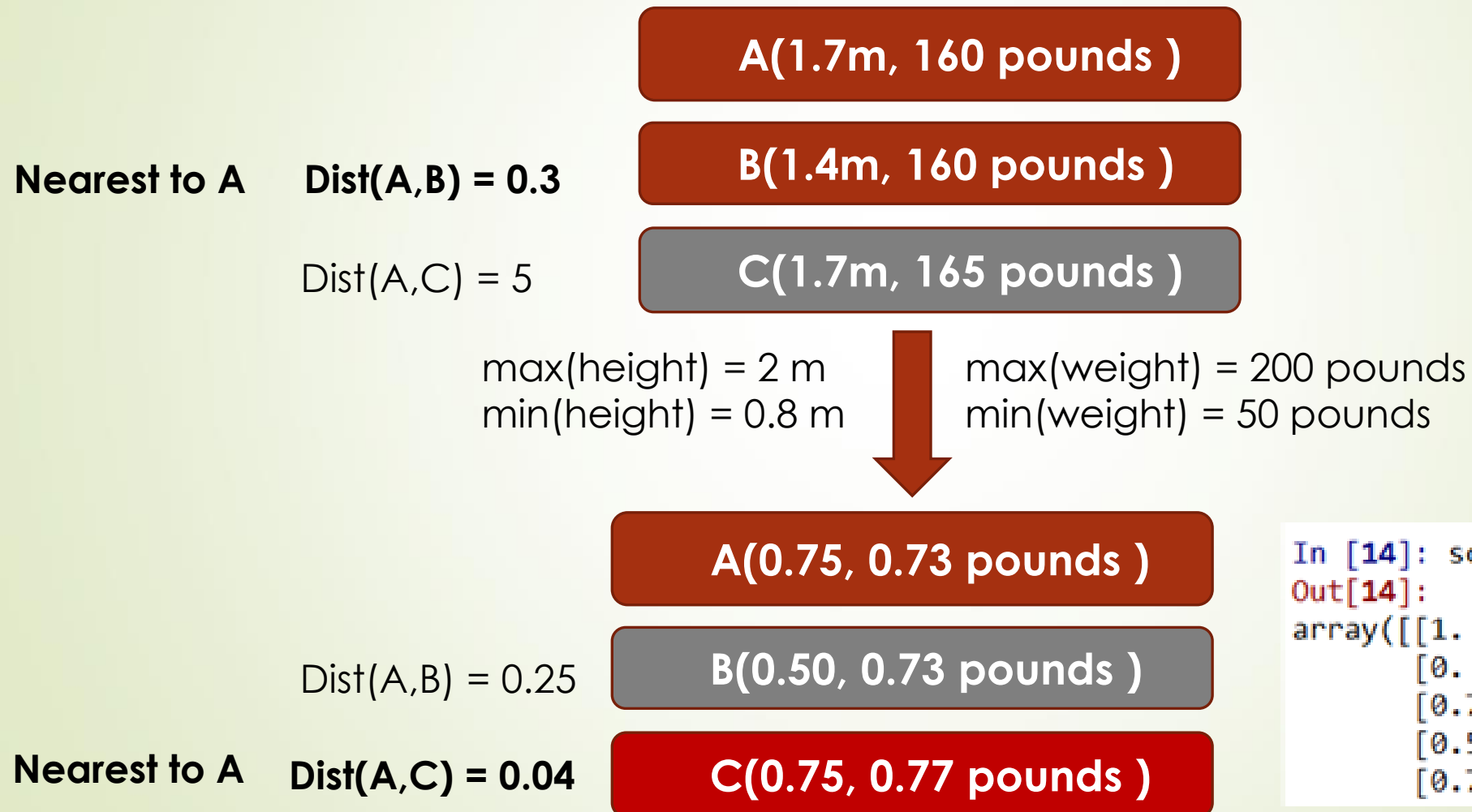
=

```
In [12]: scaler = preprocessing.MinMaxScaler()

In [13]: scaler.fit(X)
Out[13]: MinMaxScaler(copy=True, feature_range=(0, 1))

In [14]: scaler.transform(X)
Out[14]:
array([[1.         , 1.         ],
       [0.         , 0.         ],
       [0.75        , 0.73333333],
       [0.5         , 0.73333333],
       [0.75        , 0.76666667]])
```

Distance with Min-max Scaling



```
In [14]: scaler.transform(X)
Out[14]:
array([[1.         , 1.         ],
       [0.         , 0.         ],
       [0.75        , 0.73333333],
       [0.5        , 0.73333333],
       [0.75        , 0.76666667]])
```

Feature Encoding

- Many models cannot handle categorical features directly
 - Decision tree vs. SVM, Logistic Regression, KNN, K-means ...

▶ Example: Similar Whiskey

- How can we find the most similar whiskey of a given type of whiskey as a data scientist

- ✓ Construct 5 attributes that can describe the general whiskey as follows

Color: <i>yellow, very pale, pale, pale gold, gold, old gold, full gold, amber, etc.</i>	(14 values)
Nose: <i>aromatic, peaty, sweet, light, fresh, dry, grassy, etc.</i>	(12 values)
Body: <i>soft, medium, full, round, smooth, light, firm, oily.</i>	(8 values)
Palate: <i>full, dry, sherry, big, fruity, grassy, smoky, salty, etc.</i>	(15 values)
Finish: <i>full, dry, warm, light, smooth, clean, fruity, grassy, smoky, etc.</i>	(19 values)

- ✓ The values of attributes are **NOT** mutually exclusive (e.g., Aberlour's palate is described as medium, full, soft, round and smooth).
- ✓ Use a feature vector of 68 (=14+12+8+15+19) binary (0/1) attributes to reprint a type of whiskey as [0,1,1,0,...,1] (with 68 entries of 0/1)

OrdinalEncoder

- Transforms each categorical feature to one new feature of integers (0 to $n_categories - 1$)

```
In [10]: enc = preprocessing.OrdinalEncoder()

In [11]: users = [{"male", "Shanghai"}, {"male", "Hong Kong"}, {"female", "Shanghai"},
                  {"female", "New York"}]
                  1           2           0           2

In [12]: enc.fit_transform(users)
Out[12]:
array([[1., 2.],
       [1., 0.],
       [0., 2.],
       [0., 1.]])

In [13]: enc.transform([{"male", "New York"}])
Out[13]: array([[1., 1.]])
```

- Not used directly in general, as interpreted the categories as being **ordered**, which is often not desired

OneHotEncoder

- Transforms each categorical feature with n categories possible values into n categories binary features, with one of them 1, and all others 0

```
In [16]: enc = preprocessing.OneHotEncoder(handle_unknown='ignore')
```

```
In [17]: enc.fit_transform(users).toarray()
```

```
Out[17]:
```

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

Gender

City

```
In [19]: enc.transform([["male", "Beijing"]]).toarray()
```

```
Out[19]: array([[0., 1., 0., 0., 0.]])
```

pandas.get_dummies

`pandas.get_dummies(data, prefix=None, prefix_sep='_', dummy_na=False, columns=None, sparse=False, drop_first=False, dtype=None) → 'DataFrame'`

Convert categorical variable into dummy/indicator variables.

Feature Discretization

- Discretization (also known as quantization or binning) provides a way to partition continuous features into discrete/ordinal features
 - `KBinsDiscretizer` discretizes features into k bins

encode : {'onehot', 'onehot-dense', 'ordinal'}, (default='onehot')
Method used to encode the transformed result.

strategy : {'uniform', 'quantile', 'kmeans'}, (default='quantile')
Strategy used to define the widths of the bins.

uniform

All bins in each feature have identical widths.

quantile

All bins in each feature have the same number of points.

kmeans

Values in each bin have the same nearest center of a 1D k-means cluster.

```
In [28]: X = [[-2, 1, -4, -1],
...:         [-1, 2, -3, -0.5],
...:         [ 0, 3, -2, 0.5],
...:         [ 1, 4, -1, 2]]
```

```
In [29]: est =
preprocessing.KBinsDiscretizer(n_bins=3,
encode='ordinal', strategy='uniform')
```

```
In [30]: est.fit_transform(X)
```

```
Out[30]:
array([[0., 0., 0., 0.],
       [1., 1., 1., 0.],
       [2., 2., 2., 1.],
       [2., 2., 2., 2.]])
```


Feature Binarization

- Feature binarization is the process of thresholding numerical features to get boolean values.
- Binarizer: Values greater than the threshold map to 1, while values less than or equal to the threshold map to 0. With the default threshold of 0

Parameters:	threshold : <i>float, optional (0.0 by default)</i> Feature values below or equal to this are replaced with 0 in all operations on sparse matrices.
	copy : <i>boolean, optional, default True</i> set to False to perform inplace binarization on scipy.sparse CSR matrix).

```
In [35]: X = [[-2, 1, -4, -1],  
...:         [-1, 2, -3, -0.5],  
...:         [ 0, 3, -2, 0.5],  
...:         [ 1, 4, -1, 2]]
```

```
In [36]: trans = preprocessing.Binarizer()
```

```
In [37]: trans.fit_transform(X)
```

```
Out[37]:  
array([[0., 1., 0., 0.],  
       [0., 1., 0., 0.],  
       [0., 1., 0., 1.],  
       [1., 1., 0., 1.]])
```

Feature Generation

- It's useful to add complexity to the model by considering nonlinear features of the input data.

- We can extend 2-d feature vector of each sample (both training and new data) by adding quadratic combinations to a 5-d feature vector

$$(x_1, x_2) \Rightarrow (x_1, x_2, x_1^2, x_2^2, x_1x_2) \Rightarrow (x'_1, x'_2, x'_3, x'_4, x'_5)$$

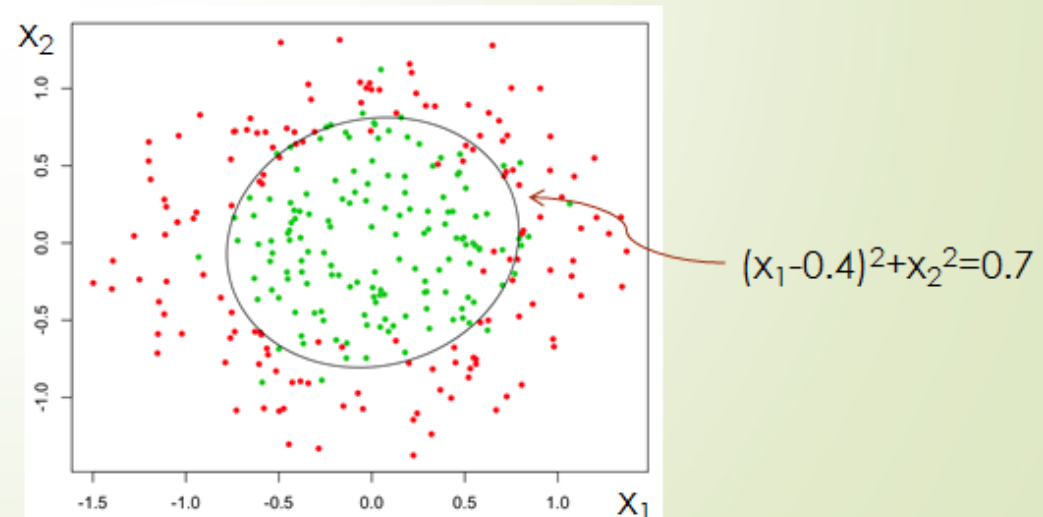
- Accordingly, the linear discriminant function in 5-d space changes to

$$f(\mathbf{x}) = w_0 + w_1x'_1 + w_2x'_2 + w_3x'_3 + w_4x'_4 + w_5x'_5$$

- Then we can create a hyperplane in the 5-d space as follows to separate the data as good as the nonlinear function in 2-d space with

$$w_0 = -0.54, w_1 = -0.8, w_2 = 0$$

$$w_3 = 1, w_4 = 1, w_5 = 0$$



Feature Generation in Sklearn

- ➡ PolynomialFeatures can get features' high-order and interaction terms

degree : integer

The degree of the polynomial features. Default = 2.

interaction_only : boolean, default = False

If true, only interaction features are produced: features that are products of features (so not $x[1]**2$, $x[0]*x[2]**3$, etc.).

include_bias : boolean

If True (default), then include a bias column, the feature in which all values are ones - acts as an intercept term in a linear model).

```
In [3]: X=np.array([[0, 1],
...:                [2, 3],
...:                [4, 5]])
```

```
In [4]: poly = preprocessing.PolynomialFeatures(2)
```

```
In [5]: poly.fit_transform(X)
```

```
Out[5]:
array([[ 1.,  0.,  1.,  0.,  0.,  1.],
       [ 1.,  2.,  3.,  4.,  6.,  9.],
       [ 1.,  4.,  5., 16., 20., 25.]])
```

```
In [6]: poly =
preprocessing.PolynomialFeatures(2,interaction_
n_only=True)
```

```
In [7]: poly.fit_transform(X)
```

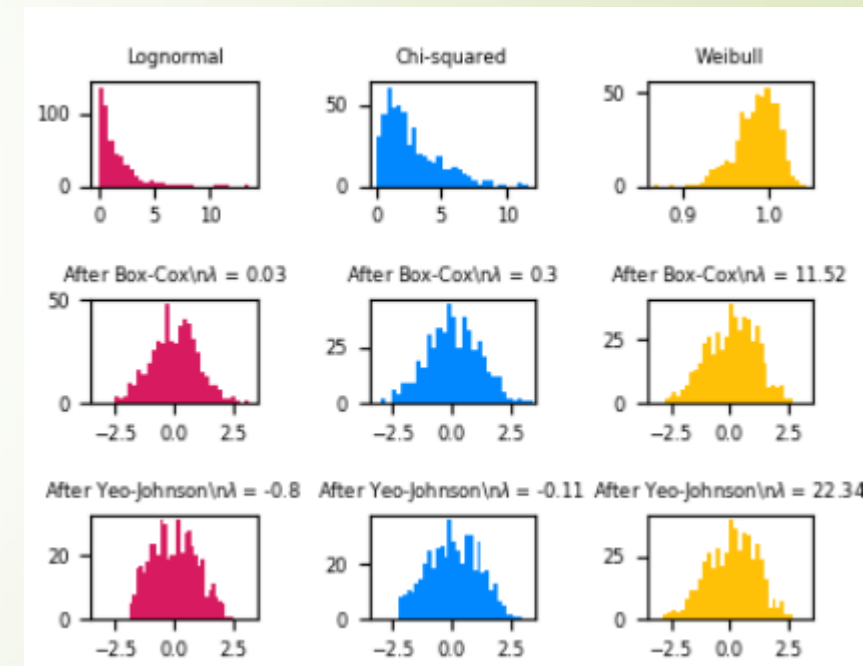
```
Out[7]:
array([[ 1.,  0.,  1.,  0.],
       [ 1.,  2.,  3.,  6.],
       [ 1.,  4.,  5., 20.]])
```

The features of X have been transformed from (X_1, X_2) to $(1, X_1, X_2, X_1^2, X_1X_2, X_2^2)$.

Interaction terms

Feature Transformation

- Feature transformation refers to family of algorithms that create new features using the existing features
 - Monotonic transformations of the features, preserving the ranks
 - Map the data to a desired distribution
- QuantileTransformer: map the data to a **uniform** distribution with values between 0 and 1
- PowerTransformer: mapping to a **Gaussian** distribution



Feature Selection in Sklearn

https://scikit-learn.org/stable/modules/feature_selection.html

- Selection by pre-defined measures like variance, Chi-squared stats
 - Univariate feature selection: [SelectKBest](#)
- Recursive feature elimination([RFE](#))
 - Repeated on the pruned set until the desired number of features to select is eventually reached.
- [SelectFromModel](#)
 - L1-based feature selection
 - Tree-based feature selection

Feature Engineering in Practice

- In general, **ALL** the **same features** engineering steps taken in training data should also be applied to new data in the **same order except**
 - Records dropping (not feature dropping)
 - Imbalanced records resampling

```
data_train[cond_1, 'col'] = np.nan
imp = SimpleImputer(strategy='most_frequent')
data_train= imp.fit_transform(data_train)
scaler = StandardScaler()
data_train= scaler.fit_transform(data_train)
model = SVC()
model.fit(data_train,label)
```



```
data_new[cond_1, 'col'] = np.nan
data_new= imp.transform(data_new)
data_new= scaler.transform(data_new)
pred_labels = model.predict(data_new)
```

- Pipelines and composite estimators

<https://www.youtube.com/watch?v=BFaadlqWIAg>

Pipelines And Composite Estimators

- Transformers are usually combined with classifiers, regressors or other estimators to build a composite estimator.
 - The most common tool is a Pipeline.
 - Pipeline is often used in combination with FeatureUnion which concatenates the output of transformers into a composite feature space.
 - The ColumnTransformer helps performing different transformations for different columns of the data. It works on pandas DataFrames.
 - The make_column_selector can select columns based on datatype or the columns name with a regex.

6.1. Pipelines and composite estimators

6.1.1. Pipeline: chaining estimators

6.1.2. Transforming target in regression

6.1.3. FeatureUnion: composite feature spaces

6.1.4. ColumnTransformer for heterogeneous data

Version of Sklearn

Available documentation for Scikit-learn ¶

Web-based documentation is available for versions listed below:

- [Scikit-learn 0.23.dev0 \(dev\) documentation \(PDF 49.5 MB\)](#)
- [Scikit-learn 0.22.2 \(stable\) documentation \(PDF 48.5 MB\)](#)
- [Scikit-learn 0.21.3 documentation \(PDF 46.7 MB\)](#)
- [Scikit-learn 0.20.4 documentation \(PDF 45.2 MB\)](#)
- [Scikit-learn 0.19.2 documentation \(PDF 42.2 MB\)](#)
- [Scikit-learn 0.18.2 documentation \(PDF 46.5 MB\)](#)
- [Scikit-learn 0.17.1 documentation \(PDF 46.0 MB\)](#)
- [Scikit-learn 0.16.1 documentation \(PDF 56.8 MB\)](#)
- [Scikit-learn 0.15-git documentation](#)

```
In [38]: import sklearn
```

```
In [39]: sklearn.__version__
```

```
Out[39]: '0.21.3'
```

```
conda install -c conda-forge scikit-learn==0.22.2
```

OR

```
conda update -c conda-forge scikit-learn
```

OR

```
conda update -all -c conda-forge
```

OR

<https://www.anaconda.com/distribution/>

Outline

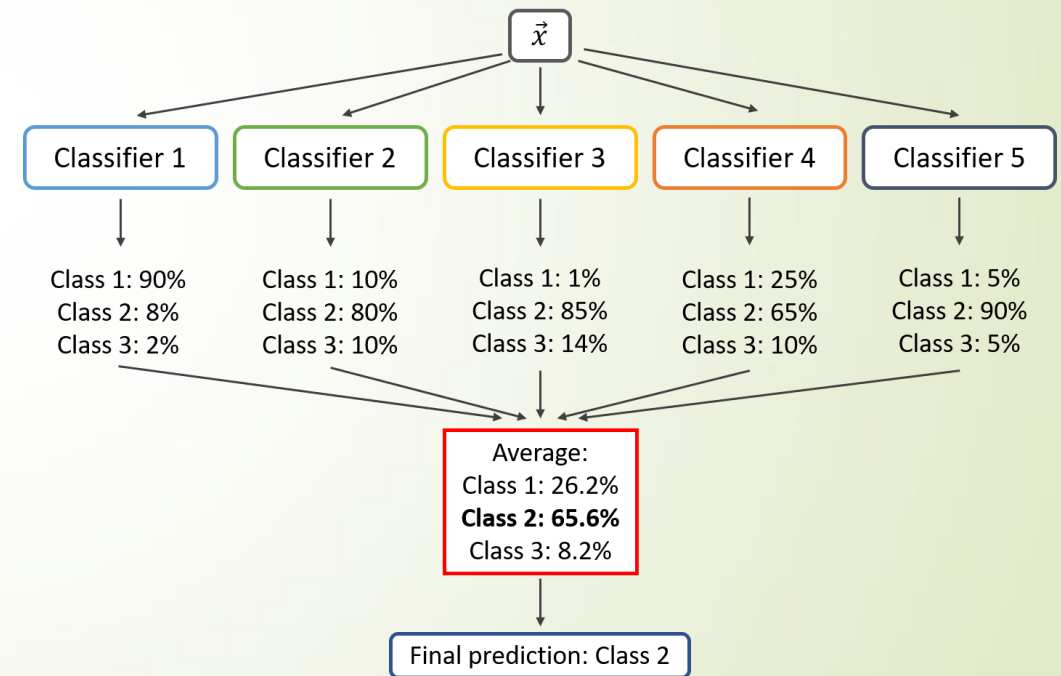
- Data Preparation
 - Data Quality Issues
 - Imbalance
- Feature Engineering
- Ensembles of Models
- Quiz

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 learner)
 - ✓ Boosting (base/weaklearner)

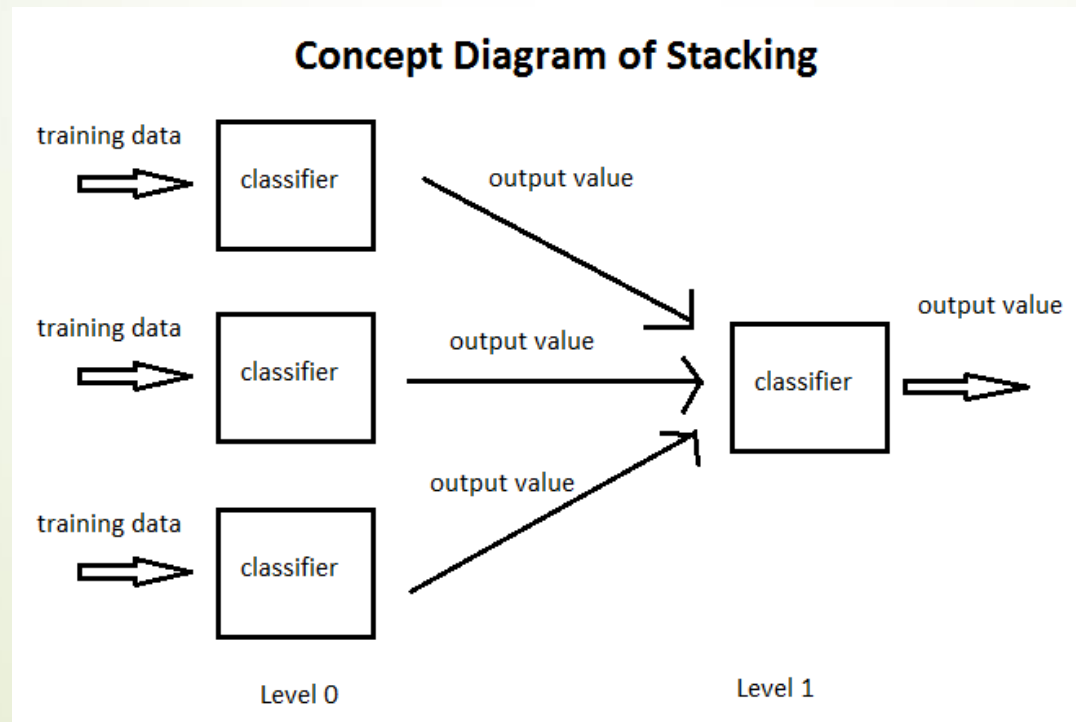
Voting/Averaging

- Max Voting/Averaging
 - Class 2: 4 total votes
- Weighted Voting Averaging
 - Class 2: 0.656 average votes



Stacking/Blending

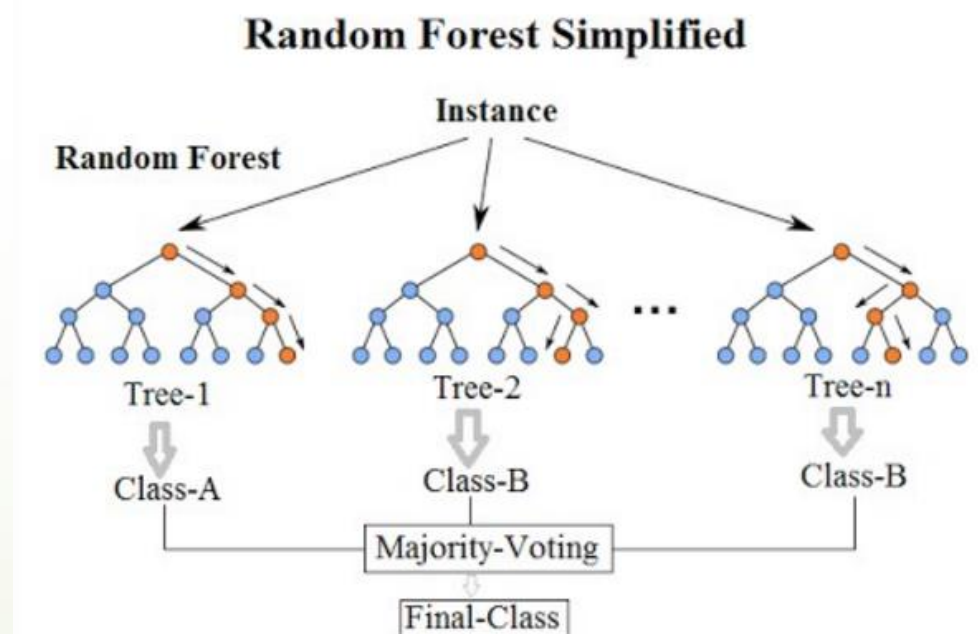
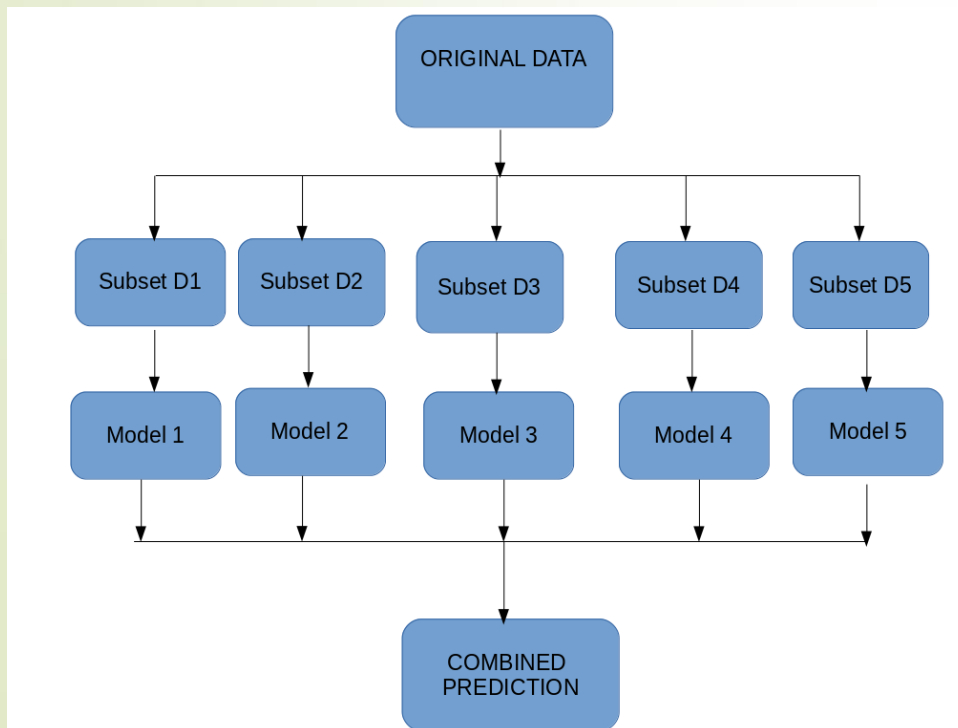
- **Stacking/blending** involves training a learning algorithm to combine the predictions of several other learning algorithms
 - ✓ Can be multiple layers



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.

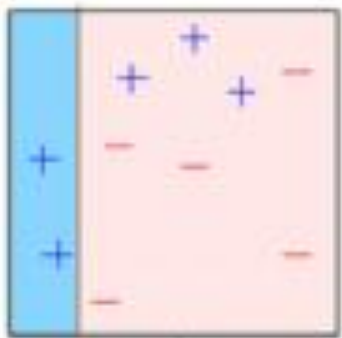
Step 2: If there is any prediction error caused by first base learner, then we pay higher attention to observations having prediction error. Then, we build the next learner

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

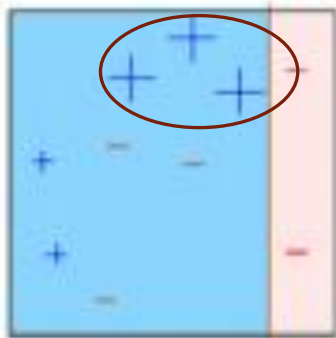
Output: The final model (strong learner) is the weighted mean of all the models (weak learners)

Boosting Examples

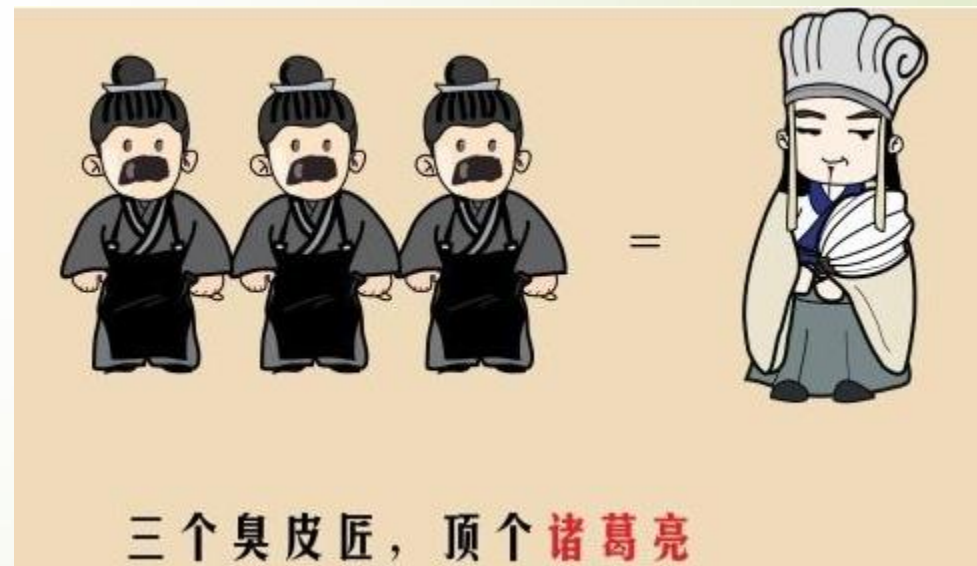
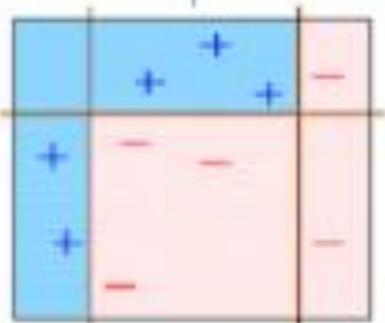
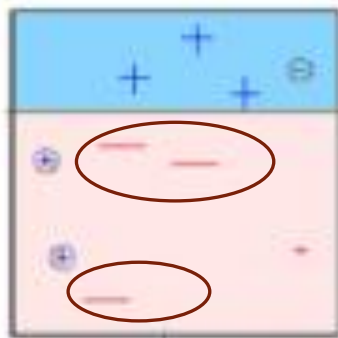
1st Base Learner



2nd Base Learner

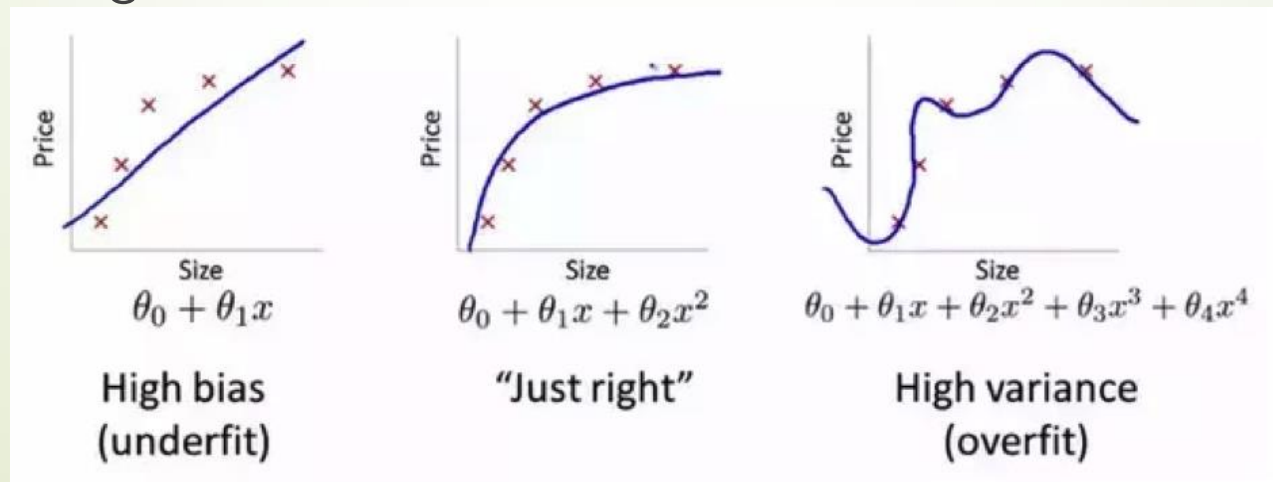


3rd Base Learner



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 : fit nonlinear relationships with linear models (**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.



Why Ensemble Learning?

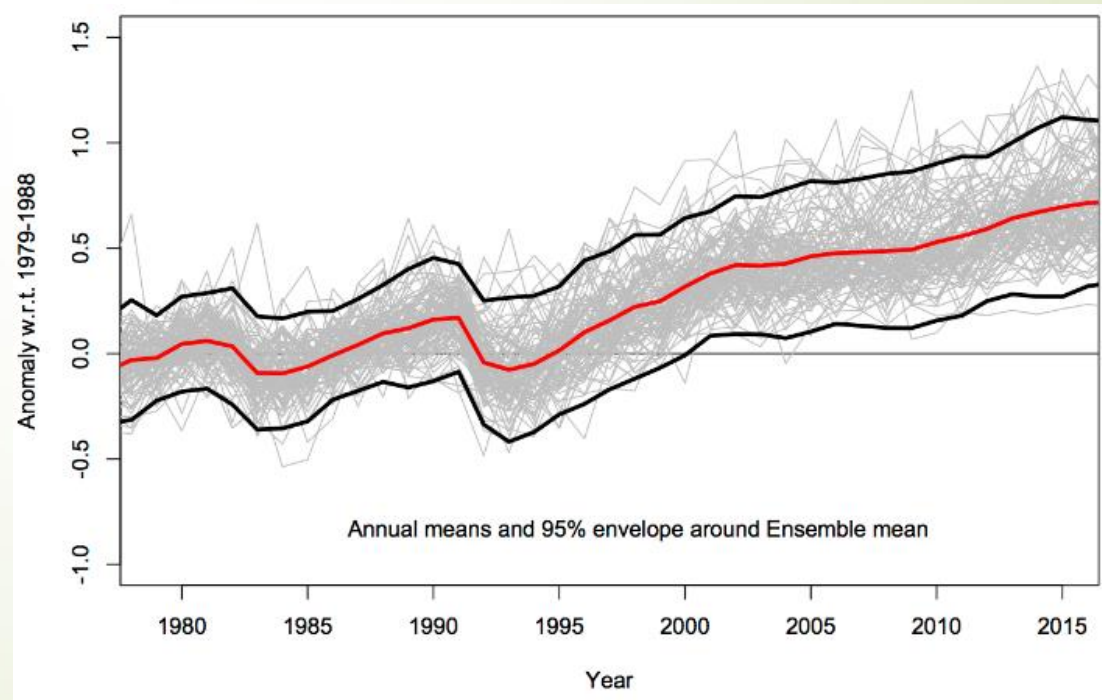
- Averaging over multiple predictions reduces the variance in the predictions.
 - ✓ Tend to improve the predictive ability more for higher-variance methods
 - ✓ Often used with tree induction, since trees tend to have high variance

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

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

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

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

- Data Preparation
 - Data Quality Issues
 - Imbalance
- Feature Engineering
- Ensembles of Models
- Quiz

Lab Quiz

- **Deadline:** 17:59 p.m., April. 17, 2020
- Two questions accounting for **5%** of overall score
- **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.

Hint on Individual Project

- Check/transform other columns with similar approaches, such as
 - 999 in pdays (manual discretization?)
 - Missing values in job, education, age
 - Normalization/standard scaler/max-min scaler ?
- With or without imbalance class adjustment
- Different single models or ensemble models
- Train-validation-test split CV or even nested CV for evaluation
- Decision function or predict_proba for ranking
- Make sure the ranking score is for **being positive**
- Make sure there are **8,000 ranking scores** in submitted file with **same order** as test dataset