MADS-ML – Machine Learning

Classification II

Prof. Dr. Stephan Doerfel

FACHHOCHSCHULE KIEL
University of Applied Sciences

Moodle (WiSe 2024/25)

Outline

Problem

- some classifiers only distinguish between two classes – e.g. SVMs

Problem

- some classifiers only distinguish between two classes – e.g. SVMs
- in real-life: often multiple classes

Problem

- some classifiers only distinguish between two classes – e.g. SVMs
- in real-life: often multiple classes
- approach:

Problem

- some classifiers only distinguish between two classes – e.g. SVMs
- in real-life: often multiple classes
- approach:
  - split the multiclass problem into a set of binary taks

Problem

- some classifiers only distinguish between two classes – e.g. SVMs
- in real-life: often multiple classes
- approach:
  - split the multiclass problem into a set of binary taks
  - train a binary classifier for each new task

Problem

- some classifiers only distinguish between two classes – e.g. SVMs
- in real-life: often multiple classes
- approach:
  - split the multiclass problem into a set of binary taks
  - train a binary classifier for each new task
  - combine the predictions of each single classifier into one final prediction

Problem

- some classifiers only distinguish between two classes – e.g. SVMs
- in real-life: often multiple classes
- approach:
  - split the multiclass problem into a set of binary taks
  - train a binary classifier for each new task
  - combine the predictions of each single classifier into one final prediction
- two ways:

Problem

- some classifiers only distinguish between two classes – e.g. SVMs
- in real-life: often multiple classes
- approach:
  - split the multiclass problem into a set of binary taks
  - train a binary classifier for each new task
  - combine the predictions of each single classifier into one final prediction
- two ways:
  - One vs. One

Problem

- some classifiers only distinguish between two classes – e.g. SVMs
- in real-life: often multiple classes
- approach:
  - split the multiclass problem into a set of binary taks
  - train a binary classifier for each new task
  - combine the predictions of each single classifier into one final prediction
- two ways:
  - One vs. One
  - One vs. Rest

One vs. One (OVO)

Given: m classes.

Split:

For each two different classes i and j, learn a classifier that predicts either i or j.

One vs. One (OVO)

Given: m classes.

Split:

For each two different classes i and j, learn a classifier that predicts either i or j.

Training:

For each possible combinations of two classes i and j, train a classifier.

One vs. One (OVO)

Given: m classes.

Split:

For each two different classes i and j, learn a classifier that predicts either i or j.

Training:

For each possible combinations of two classes i and j, train a classifier.

- $\frac{m \cdot (m-1)}{2}$ classifiers

One vs. One (OVO)

Given: m classes.

Split:

For each two different classes i and j, learn a classifier that predicts either i or j.

Training:

For each possible combinations of two classes i and j, train a classifier.

- $\frac{m \cdot (m-1)}{2}$ classifiers
- training data per classifier: subset containing the instances of classes i and j

One vs. One (OVO)

Given: m classes.

Split:

For each two different classes i and j, learn a classifier that predicts either i or j.

Training:

For each possible combinations of two classes i and j, train a classifier.

- $\frac{m \cdot (m-1)}{2}$ classifiers
- training data per classifier: subset containing the instances of classes i and j

Classification:

To classify x, use alle the classifiers on x and let them vote. The class with the most votes is predicted.

One vs. One (OVO)

Given: m classes.

Split:

For each two different classes i and j, learn a classifier that predicts either i or j.

Training:

For each possible combinations of two classes i and j, train a classifier.

- $\frac{m \cdot (m-1)}{2}$ classifiers
- training data per classifier: subset containing the instances of classes i and j

Classification:

To classify x, use alle the classifiers on x and let them vote. The class with the most votes is predicted.

In Python:

`sklearn.multiclass.OneVsOneClassifier`

One vs. Rest (OVR)

Given: m classes.

Split:

For each class i, learn a classifier for the subproblem C_i vs. $\bigcup_{j \neq i} C_j$.

One vs. Rest (OVR)

Given: m classes.

Split:

For each class i, learn a classifier for the subproblem C_i vs. $\bigcup_{j \neq i} C_j$.

Training:

Pick each class C_i and

One vs. Rest (OVR)

Given: m classes.

Split:

For each class i, learn a classifier for the subproblem C_i vs. $\bigcup_{j \neq i} C_j$.

Training:

Pick each class C_i and

- replace all other classes by 'other'

One vs. Rest (OVR)

Given: m classes.

Split:

For each class i, learn a classifier for the subproblem C_i vs. $\bigcup_{j \neq i} C_j$.

Training:

Pick each class C_i and

- replace all other classes by 'other'
- learn a classifier for the subproblem on the full (modified) dataset

One vs. Rest (OVR)

Given: m classes.

Split:

For each class i, learn a classifier for the subproblem C_i vs. $\bigcup_{j \neq i} C_j$.

Training:

Pick each class C_i and

- replace all other classes by 'other'
- learn a classifier for the subproblem on the full (modified) dataset

Classification:

To classify x, use all classifiers and let them compute a probability for their class. The class with the highest probability is predicted.

One vs. Rest (OVR)

Given: m classes.

Split:

For each class i, learn a classifier for the subproblem C_i vs. $\bigcup_{j \neq i} C_j$.

Training:

Pick each class C_i and

- replace all other classes by 'other'
- learn a classifier for the subproblem on the full (modified) dataset

Classification:

To classify x, use all classifiers and let them compute a probability for their class. The class with the highest probability is predicted.

In Python:

`sklearn.multiclass.OneVsRestClassifier`

Discussion

Which?

Discussion

Which?

- Tradeoff between learning fewer classifiers (OVR) vs. learning on smaller datasets (OVO).

Discussion

Which?

- Tradeoff between learning fewer classifiers (OVR) vs. learning on smaller datasets (OVO).
- Often OVR is faster.

Discussion

Which?

- Tradeoff between learning fewer classifiers (OVR) vs. learning on smaller datasets (OVO).
- Often OVR is faster.
- Depends on algorithm's scalability w.r.t. the dataset size.

Discussion

Which?

- Tradeoff between learning fewer classifiers (OVR) vs. learning on smaller datasets (OVO).
- Often OVR is faster.
- Depends on algorithm's scalability w.r.t. the dataset size.
- For OVR, the algorithm must provide probabilities.

Discussion

Which?

- Tradeoff between learning fewer classifiers (OVR) vs. learning on smaller datasets (OVO).
- Often OVR is faster.
- Depends on algorithm's scalability w.r.t. the dataset size.
- For OVR, the algorithm must provide probabilities.
- For OVR, the probabilities are confidences of the individual classifiers. They are not necessarily on the same scale.

Discussion

Which?

- Tradeoff between learning fewer classifiers (OVR) vs. learning on smaller datasets (OVO).
- Often OVR is faster.
- Depends on algorithm's scalability w.r.t. the dataset size.
- For OVR, the algorithm must provide probabilities.
- For OVR, the probabilities are confidences of the individual classifiers. They are not necessarily on the same scale.
- For OVR, even if the dataset is balanced, the binary problems are (heavily) unbalanced.

Discussion

Which?

- Tradeoff between learning fewer classifiers (OVR) vs. learning on smaller datasets (OVO).
- Often OVR is faster.
- Depends on algorithm's scalability w.r.t. the dataset size.
- For OVR, the algorithm must provide probabilities.
- For OVR, the probabilities are confidences of the individual classifiers. They are not necessarily on the same scale.
- For OVR, even if the dataset is balanced, the binary problems are (heavily) unbalanced.

Probabilities:

For OVR, the probabilities usually do not sum up to 1! The are confidences of the individual classifiers and may be on different scales.

Discussion

Which?

- Tradeoff between learning fewer classifiers (OVR) vs. learning on smaller datasets (OVO).
- Often OVR is faster.
- Depends on algorithm's scalability w.r.t. the dataset size.
- For OVR, the algorithm must provide probabilities.
- For OVR, the probabilities are confidences of the individual classifiers. They are not necessarily on the same scale.
- For OVR, even if the dataset is balanced, the binary problems are (heavily) unbalanced.

Probabilities:

For OVR, the probabilities usually do not sum up to 1! The are confidences of the individual classifiers and may be on different scales.

Note: Implementations like SVC already implement such strategies (SVC implements OVO).

Outline

"All models are wrong, but some are useful."

"For optimization problems, the performance of two algorithms, averaged over all problems is identical."

Very rough summary of the "No-free-lunch" theorem by David Wolpert.

Selection of Suitable Algorithms

- check that data conforms to an algorithms requirements

Selection of Suitable Algorithms

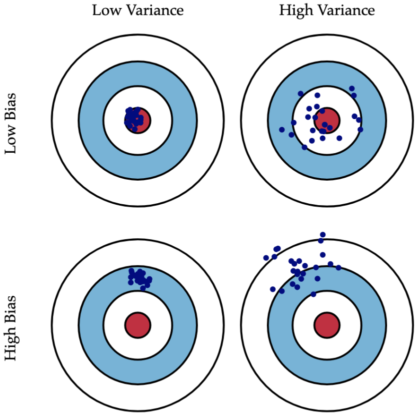
- check that data conforms to an algorithms requirements
- some algorithms assume certain distributions for the features

Selection of Suitable Algorithms

- check that data conforms to an algorithms requirements
- some algorithms assume certain distributions for the features
- test and compare algorithms on your dataset

The Problem

Repeatedly learn model using different sets of training data. Each time, predict for the same, new data instance:



Source: [1]

Bias and Variance

Bias: Errors due to bias show in the training data: difference between actual and predicted value.

Bias and Variance

Bias: Errors due to bias show in the training data: difference between actual and predicted value.

- high bias means, the model is underfitting (low quality on training data)

Bias and Variance

Bias: Errors due to bias show in the training data: difference between actual and predicted value.

- high bias means, the model is underfitting (low quality on training data)
- potential causes:

Bias and Variance

Bias: Errors due to bias show in the training data: difference between actual and predicted value.

- high bias means, the model is underfitting (low quality on training data)
- potential causes:
  - model too simple

Bias and Variance

Bias: Errors due to bias show in the training data: difference between actual and predicted value.

- high bias means, the model is underfitting (low quality on training data)
- potential causes:
  - model too simple
  - data does not conform to assumptions of algorithm

Bias and Variance

Bias: Errors due to bias show in the training data: difference between actual and predicted value.

- high bias means, the model is underfitting (low quality on training data)
- potential causes:
  - model too simple
  - data does not conform to assumptions of algorithm

Variance: The predicted value varies a lot for different choices of training datasets.

Bias and Variance

Bias: Errors due to bias show in the training data: difference between actual and predicted value.

- high bias means, the model is underfitting (low quality on training data)
- potential causes:
  - model too simple
  - data does not conform to assumptions of algorithm

Variance: The predicted value varies a lot for different choices of training datasets.

- model is specific to the training

Bias and Variance

Bias: Errors due to bias show in the training data: difference between actual and predicted value.

- high bias means, the model is underfitting (low quality on training data)
- potential causes:
  - model too simple
  - data does not conform to assumptions of algorithm

Variance: The predicted value varies a lot for different choices of training datasets.

- model is specific to the training
- high variance leads to overfitting (high quality on the training data but low quality on the test data)

Bias and Variance

Bias: Errors due to bias show in the training data: difference between actual and predicted value.

- high bias means, the model is underfitting (low quality on training data)
- potential causes:
  - model too simple
  - data does not conform to assumptions of algorithm

Variance: The predicted value varies a lot for different choices of training datasets.

- model is specific to the training
- high variance leads to overfitting (high quality on the training data but low quality on the test data)
- potential causes:

Bias and Variance

Bias: Errors due to bias show in the training data: difference between actual and predicted value.

- high bias means, the model is underfitting (low quality on training data)
- potential causes:
  - model too simple
  - data does not conform to assumptions of algorithm

Variance: The predicted value varies a lot for different choices of training datasets.

- model is specific to the training
- high variance leads to overfitting (high quality on the training data but low quality on the test data)
- potential causes:
  - model picks up on random patterns in the training data

Bias and Variance

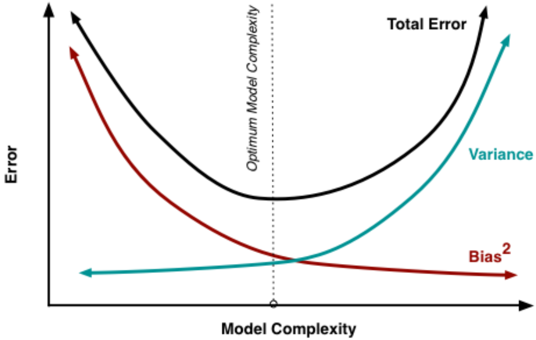
Bias: Errors due to bias show in the training data: difference between actual and predicted value.

- high bias means, the model is underfitting (low quality on training data)
- potential causes:
  - model too simple
  - data does not conform to assumptions of algorithm

Variance: The predicted value varies a lot for different choices of training datasets.

- model is specific to the training
- high variance leads to overfitting (high quality on the training data but low quality on the test data)
- potential causes:
  - model picks up on random patterns in the training data
  - model is too complex, has too many degrees of freedom (parameters)

Bias-Variance-Tradeoff



Source: [1]

Combatting overfitting is one of the main tasks in ML.

Train - Test Splits

Method

Train - Test Splits

Method

- Split the data into two sets

Train - Test Splits

Method

- Split the data into two sets
- train on the training set

Train - Test Splits

Method

- Split the data into two sets
- train on the training set
- test on the test set

Train - Test Splits

Method

- Split the data into two sets
- train on the training set
- test on the test set
- quality score on training data evaluates the fit of the model

Train - Test Splits

Method

- Split the data into two sets
- train on the training set
- test on the test set
- quality score on training data evaluates the fit of the model
- quality score on test set indicates generalizability of the model

Train - Test Splits

Method

- Split the data into two sets
- train on the training set
- test on the test set
- quality score on training data evaluates the fit of the model
- quality score on test set indicates generalizability of the model

Problems

Train - Test Splits

Method

- Split the data into two sets
- train on the training set
- test on the test set
- quality score on training data evaluates the fit of the model
- quality score on test set indicates generalizability of the model

Problems

- Only one dataset / one split for evaluation

Train - Test Splits

Method

- Split the data into two sets
- train on the training set
- test on the test set
- quality score on training data evaluates the fit of the model
- quality score on test set indicates generalizability of the model

Problems

- Only one dataset / one split for evaluation
- Hyperparameters are optimized on the test data!

Train - Test Splits

Method

- Split the data into two sets
- train on the training set
- test on the test set
- quality score on training data evaluates the fit of the model
- quality score on test set indicates generalizability of the model

Problems

- Only one dataset / one split for evaluation
- Hyperparameters are optimized on the test data!
- Lucky split → overestimate performance

Train - Test Splits

Method

- Split the data into two sets
- train on the training set
- test on the test set
- quality score on training data evaluates the fit of the model
- quality score on test set indicates generalizability of the model

Problems

- Only one dataset / one split for evaluation
- Hyperparameters are optimized on the test data!
- Lucky split → overestimate performance
- Using the same dataset for tuning parameters over and over leads to overfitting

Train - Test Splits

Method

- Split the data into two sets
- train on the training set
- test on the test set
- quality score on training data evaluates the fit of the model
- quality score on test set indicates generalizability of the model

Problems

- Only one dataset / one split for evaluation
- Hyperparameters are optimized on the test data!
- Lucky split ➔ overestimate performance
- Using the same dataset for tuning parameters over and over leads to overfitting

Solutions

Train - Test Splits

Method

- Split the data into two sets
- train on the training set
- test on the test set
- quality score on training data evaluates the fit of the model
- quality score on test set indicates generalizability of the model

Problems

- Only one dataset / one split for evaluation
- Hyperparameters are optimized on the test data!
- Lucky split → overestimate performance
- Using the same dataset for tuning parameters over and over leads to overfitting

Solutions

- Use several splits of the data → cross validation

k-Fold Cross Validation

- Split the training data into k folds (parts).

k-Fold Cross Validation

- Split the training data into k folds (parts).
- Use $k - 1$ folds for training, the remaining fold for testing.

k-Fold Cross Validation

- Split the training data into k folds (parts).
- Use $k - 1$ folds for training, the remaining fold for testing.
- Thus k different splits.

k-Fold Cross Validation

- Split the training data into k folds (parts).
- Use $k-1$ folds for training, the remaining fold for testing.
- Thus k different splits.
- Overall quality = mean of the k quality scores.

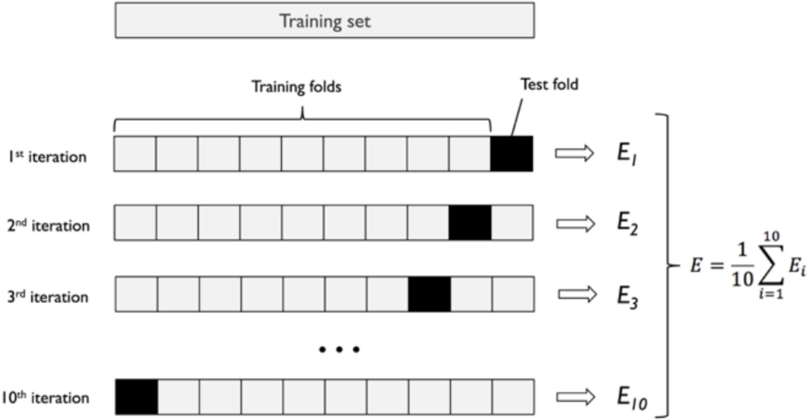
k-Fold Cross Validation

- Split the training data into k folds (parts).
- Use $k - 1$ folds for training, the remaining fold for testing.
- Thus k different splits.
- Overall quality = mean of the k quality scores.
- Each data instance belongs to the test set once (minimizes variance of the overall result).

k-Fold Cross Validation

- Split the training data into k folds (parts).
- Use $k-1$ folds for training, the remaining fold for testing.
- Thus k different splits.
- Overall quality = mean of the k quality scores.
- Each data instance belongs to the test set once (minimizes variance of the overall result).
- 10-fold CV has been suggested to yield a good balance between bias and variance.

k-Fold Cross Validation



Source: [2], p. 192

Cross Validation Usecases

Cross Validation can be used for

Hyperparameter Optimization For different settings of the hyperparameters, models are trained on the $k-1$ training folds and evaluated and compared on the remaining fold.

Cross Validation Usecases

Cross Validation can be used for

Hyperparameter Optimization For different settings of the hyperparameters, models are trained on the $k - 1$ training folds and evaluated and compared on the remaining fold.

Comparison of Different Algorithms For comparing different algorithms on "a couple of datasets", we can use cross validation.

Cross Validation Usecases

Cross Validation can be used for

Hyperparameter Optimization For different settings of the hyperparameters, models are trained on the $k - 1$ training folds and evaluated and compared on the remaining fold.

Comparison of Different Algorithms For comparing different algorithms on "a couple of datasets", we can use cross validation.

Both In a **nested cross validation**, an inner loop is used for parameter optimization (splitting in training and validation data) and an outer loop is used for evaluating and comparing the respective best versions).

Cross Validation Usecases

Cross Validation can be used for

Hyperparameter Optimization For different settings of the hyperparameters, models are trained on the $k-1$ training folds and evaluated and compared on the remaining fold.

Comparison of Different Algorithms For comparing different algorithms on "a couple of datasets", we can use cross validation.

Both In a **nested cross validation**, an inner loop is used for parameter optimization (splitting in training and validation data) and an outer loop is used for evaluating and comparing the respective best versions).

In Python, k-fold and stratified k-fold setups are available.

Choosing Hyperparameters

Grid Search:

Choosing Hyperparameters

Grid Search:

- Define a grid of hyperparameters.

Choosing Hyperparameters

Grid Search:

- Define a grid of hyperparameters.
- Train and Evaluate the algorithm on **each** point of the grid (exhaustive search).

Choosing Hyperparameters

Grid Search:

- Define a grid of hyperparameters.
- Train and Evaluate the algorithm on **each** point of the grid (exhaustive search).
- Use cross validation to compute a stable estimates, less prone to the influcence of random splits.

Choosing Hyperparameters

Grid Search:

- Define a grid of hyperparameters.
- Train and Evaluate the algorithm on **each** point of the grid (exhaustive search).
- Use cross validation to compute a stable estimates, less prone to the influcence of random splits.
- return the hyperparameters with the highest classification quality (averaged over the folds)

Choosing Hyperparameters

Grid Search:

- Define a grid of hyperparameters.
- Train and Evaluate the algorithm on **each** point of the grid (exhaustive search).
- Use cross validation to compute a stable estimates, less prone to the influcence of random splits.
- return the hyperparameters with the highest classification quality (averaged over the folds)
- In Python: `sklearn.model_selection.GridSearchCV`

Alternatives to Grid Search:

Choosing Hyperparameters

Grid Search:

- Define a grid of hyperparameters.
- Train and Evaluate the algorithm on **each** point of the grid (exhaustive search).
- Use cross validation to compute a stable estimates, less prone to the influcence of random splits.
- return the hyperparameters with the highest classification quality (averaged over the folds)
- In Python: `sklearn.model_selection.GridSearchCV`

Alternatives to Grid Search:

- Randomized Search: randomized sampling of parameters (from distributions) instead of exhaustive search on a grid (e.g. `sklearn.model_selection.RandomizedSearchCV`)

Choosing Hyperparameters

Grid Search:

- Define a grid of hyperparameters.
- Train and Evaluate the algorithm on **each** point of the grid (exhaustive search).
- Use cross validation to compute a stable estimates, less prone to the influcence of random splits.
- return the hyperparameters with the highest classification quality (averaged over the folds)
- In Python: `sklearn.model_selection.GridSearchCV`

Alternatives to Grid Search:

- Randomized Search: randomized sampling of parameters (from distributions) instead of exhaustive search on a grid (e.g. `sklearn.model_selection.RandomizedSearchCV`)
- Simulated Annealing: borrows from physics, starts with random point, then checks neighbors, gets out of local optima by allowing moves to worse solutions with decreasing probabilities (e.g. module `simulated_annealing`)

How much training data, how many parameters?

Compute **learning curves**. Use different sizes subsets of the data for training. Plot training and test quality against training set size.

How much training data, how many parameters?

Compute learning curves. Use different sizes subsets of the data for training. Plot training and test quality against training set size.

- With more training data, training quality usually decreses, test quality usually rises.

How much training data, how many parameters?

Compute learning curves. Use different sizes subsets of the data for training. Plot training and test quality against training set size.

- With more training data, training quality usually decreses, test quality usually rises.
- Large differences between the two scores indicate overfitting.

How much training data, how many parameters?

Compute learning curves. Use different sizes subsets of the data for training. Plot training and test quality against training set size.

- With more training data, training quality usually decreses, test quality usually rises.
- Large differences between the two scores indicate overfitting.
- Low training quality indicates underfitting.

How much training data, how many parameters?

Compute learning curves. Use different sizes subsets of the data for training. Plot training and test quality against training set size.

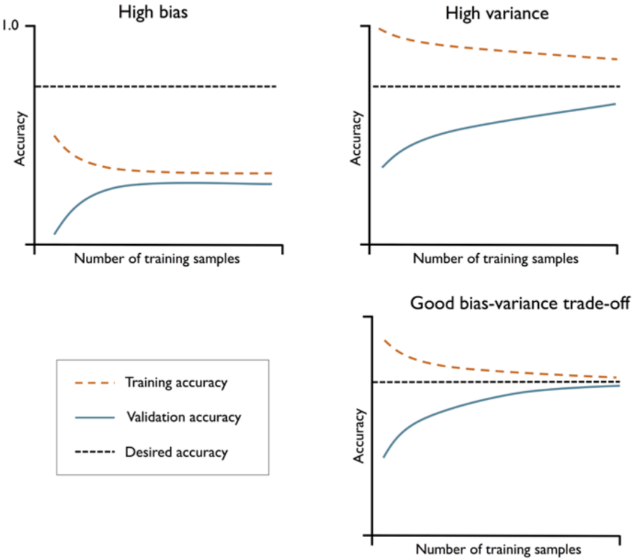
- With more training data, training quality usually decreses, test quality usually rises.
- Large differences between the two scores indicate overfitting.
- Low training quality indicates underfitting.
- Usually, more training data reduces overfitting. When training and test scores converge towards the same value, enough training data is used.

How much training data, how many parameters?

Compute learning curves. Use different sizes subsets of the data for training. Plot training and test quality against training set size.

- With more training data, training quality usually decreses, test quality usually rises.
- Large differences between the two scores indicate overfitting.
- Low training quality indicates underfitting.
- Usually, more training data reduces overfitting. When training and test scores converge towards the same value, enough training data is used.
- Use cross validation to compute a stable curve, less prone to the influcence of random splits.

Learning Curves



Source: [2], p. 196

Validation Curves

Plot training and test scores against different settings of hyper parameters.

- Find the point where the test scores are best.

Validation Curves

Plot training and test scores against different settings of hyper parameters.

- Find the point where the test scores are best.
- If the algorithm does not overfit, in that point, the test and training score should be close.

Validation Curves

Plot training and test scores against different settings of hyper parameters.

- Find the point where the test scores are best.
- If the algorithm does not overfit, in that point, the test and training score should be close.
- Use cross validation to compute a stable curve, less prone to the influcence of random splits.

Outline

Evaluation with Imbalanced Data

Problem: With imbalanced data, accuracy might be very high, even though we miss all the elements of the small class.
Solutions:

- evaluation measures like balanced accuracy, weighted accuracy, recall/precision/f1, area under the roc curve
- penalize missclassification during training harder for smaller (more expensive) classes
- upsampling: copy instances of the smaller class to make it larger
- downsampling: drop instances of the larger class to make it smaller
- create artificial instances of the smaller class

Class Distribution and Sampling

Problem: When splitting datasets (e.g. train/test split, cross validations), the resulting subsets might have a different class distribution than the original dataset.

Class Distribution and Sampling

Problem: When splitting datasets (e.g. train/test split, cross validations), the resulting subsets might have a different class distribution than the original dataset.

Solution: **Stratification** – enforce the same class distribution in all subsets (as far as possible)

Outline

Evaluation Measures

Notebook 06_1_evaluation_metrics

Evaluation Measures

Notebook 06_1_evaluation_metrics

- many measures can be extended using weights (some examples count more than others) to include different costs for missclassifications

Evaluation Measures

Notebook 06_1_evaluation_metrics

- many measures can be extended using weights (some examples count more than others) to include different costs for missclassifications
- the choice of the evaluation measure should be domain-driven

Evaluation Measures

Notebook 06_1_evaluation_metrics

- many measures can be extended using weights (some examples count more than others) to include different costs for missclassifications
- the choice of the evaluation measure should be domain-driven
- often it is reasonable to compute and visualize more than one measure

Outline

Pipelines

Idea Collect multiple steps of the ML process into one pipeline.

Components E.g.:

- imputing missing values
- preprocessing steps
- scaling
- encoding of values

Exercises 1–2

Outline

"Shit in – Shit out."

Missing Values

Detection (in tabular data): In pandas: `df.isnull().sum()` yields missing values per column

Missing Values

Detection (in tabular data): In pandas: `df.isnull().sum()` yields missing values per column

Elimination `df.dropna(axis=x)`

Can also specify subset of features among missing values are unacceptable.

Missing Values

Detection (in tabular data): In pandas: `df.isnull().sum()` yields missing values per column

Elimination `df.dropna(axis=x)`

- x=0: drop rows with missing values

Can also specify subset of features among missing values are unacceptable.

Missing Values

Detection (in tabular data): In pandas: `df.isnull().sum()` yields missing values per column

Elimination `df.dropna(axis=x)`

- x=0: drop rows with missing values
- x=1: drop columns with missing values

Can also specify subset of features among missing values are unacceptable.

Missing Values

Detection (in tabular data): In pandas: `df.isnull().sum()` yields missing values per column

Elimination `df.dropna(axis=x)`

- x=0: drop rows with missing values
- x=1: drop columns with missing values

Can also specify subset of features among missing values are unacceptable.

Imputation Fill in the gap. Most common: Mean imputation. Replace missing value by the features mean (total or per class). Alternatively, use a fix constant (zero, one, ...)

Missing Values

Detection (in tabular data): In pandas: `df.isnull().sum()` yields missing values per column

Elimination `df.dropna(axis=x)`

- x=0: drop rows with missing values
- x=1: drop columns with missing values

Can also specify subset of features among missing values are unacceptable.

Imputation Fill in the gap. Most common: Mean imputation. Replace missing value by the features mean (total or per class). Alternatively, use a fix constant (zero, one, ...)

Flagging For categorical values, impute using a new category "Missing". For continous values, impute and add a feature with the values missing.

Missing Values

Detection (in tabular data): In pandas: `df.isnull().sum()` yields missing values per column

Elimination `df.dropna(axis=x)`

- x=0: drop rows with missing values
- x=1: drop columns with missing values

Can also specify subset of features among missing values are unacceptable.

Imputation Fill in the gap. Most common: Mean imputation. Replace missing value by the features mean (total or per class). Alternatively, use a fix constant (zero, one, ...)

Flagging For categorical values, impute using a new category "Missing". For continous values, impute and add a feature with the values missing.

Cleanup Data Extraction Investigate and debug the data creation process, remove errors that cause the missing values!

Cleaning the Data

- Missing values.

Cleaning the Data

- Missing values.
- Compatibility checks (e.g. for data from different sources, or collected over longer periods of time).

Cleaning the Data

- Missing values.
- Compatibility checks (e.g. for data from different sources, or collected over longer periods of time).
  - Time: different time zones, different time formats, leap years, domain-specific pauses (e.g. EOB), . . .

Cleaning the Data

- Missing values.
- Compatibility checks (e.g. for data from different sources, or collected over longer periods of time).
  - Time: different time zones, different time formats, leap years, domain-specific pauses (e.g. EOB), ...
  - Units:

Cleaning the Data

- Missing values.
- Compatibility checks (e.g. for data from different sources, or collected over longer periods of time).
  - Time: different time zones, different time formats, leap years, domain-specific pauses (e.g. EOB), ...
  - Units:
    - metric system, imperial system, US customary system, ...

Cleaning the Data

- Missing values.
- Compatibility checks (e.g. for data from different sources, or collected over longer periods of time).
  - Time: different time zones, different time formats, leap years, domain-specific pauses (e.g. EOB), ...
  - Units:
    - metric system, imperial system, US customary system, ...
    - unit scales: cm vs m, vs km, ...

Cleaning the Data

- Missing values.
- Compatibility checks (e.g. for data from different sources, or collected over longer periods of time).
  - Time: different time zones, different time formats, leap years, domain-specific pauses (e.g. EOB), ...
  - Units:
    - metric system, imperial system, US customary system, ...
    - unit scales: cm vs m, vs km, ...
    - Category/Feature names, spelling

Cleaning the Data

- Missing values.
- Compatibility checks (e.g. for data from different sources, or collected over longer periods of time).
  - Time: different time zones, different time formats, leap years, domain-specific pauses (e.g. EOB), ...
  - Units:
    - metric system, imperial system, US customary system, ...
    - unit scales: cm vs m, vs km, ...
    - Category/Feature names, spelling
- Outliers: Could be interesting / Could be nonsense

Cleaning the Data

- Missing values.
- Compatibility checks (e.g. for data from different sources, or collected over longer periods of time).
  - Time: different time zones, different time formats, leap years, domain-specific pauses (e.g. EOB), ...
  - Units:
    - metric system, imperial system, US customary system, ...
    - unit scales: cm vs m, vs km, ...
    - Category/Feature names, spelling
- Outliers: Could be interesting / Could be nonsense
- Duplicates and contradictory data.

Cleaning the Data

- Missing values.
- Compatibility checks (e.g. for data from different sources, or collected over longer periods of time).
  - Time: different time zones, different time formats, leap years, domain-specific pauses (e.g. EOB), ...
  - Units:
    - metric system, imperial system, US customary system, ...
    - unit scales: cm vs m, vs km, ...
    - Category/Feature names, spelling
- Outliers: Could be interesting / Could be nonsense
- Duplicates and contradictory data.
- Trade-off between

Cleaning the Data

- Missing values.
- Compatibility checks (e.g. for data from different sources, or collected over longer periods of time).
  - Time: different time zones, different time formats, leap years, domain-specific pauses (e.g. EOB), ...
  - Units:
    - metric system, imperial system, US customary system, ...
    - unit scales: cm vs m, vs km, ...
    - Category/Feature names, spelling
- Outliers: Could be interesting / Could be nonsense
- Duplicates and contradictory data.
- Trade-off between
  - overly pragmatic (just remove anything that pops up)

Cleaning the Data

- Missing values.
- Compatibility checks (e.g. for data from different sources, or collected over longer periods of time).
  - Time: different time zones, different time formats, leap years, domain-specific pauses (e.g. EOB), ...
  - Units:
    - metric system, imperial system, US customary system, ...
    - unit scales: cm vs m, vs km, ...
    - Category/Feature names, spelling
- Outliers: Could be interesting / Could be nonsense
- Duplicates and contradictory data.
- Trade-off between
  - overly pragmatic (just remove anything that pops up)
  - expensive investigation of causes

Cleaning the Data

- Missing values.
- Compatibility checks (e.g. for data from different sources, or collected over longer periods of time).
  - Time: different time zones, different time formats, leap years, domain-specific pauses (e.g. EOB), ...
  - Units:
    - metric system, imperial system, US customary system, ...
    - unit scales: cm vs m, vs km, ...
    - Category/Feature names, spelling
- Outliers: Could be interesting / Could be nonsense
- Duplicates and contradictory data.
- Trade-off between
  - overly pragmatic (just remove anything that pops up)
  - expensive investigation of causes
- **Always make notes of what and why you clean!**

Cleaning the Data

- Missing values.
- Compatibility checks (e.g. for data from different sources, or collected over longer periods of time).
  - Time: different time zones, different time formats, leap years, domain-specific pauses (e.g. EOB), ...
  - Units:
    - metric system, imperial system, US customary system, ...
    - unit scales: cm vs m, vs km, ...
    - Category/Feature names, spelling
- Outliers: Could be interesting / Could be nonsense
- Duplicates and contradictory data.
- Trade-off between
  - overly pragmatic (just remove anything that pops up)
  - expensive investigation of causes
- **Always make notes of what and why you clean!**
- **Make cleaning reproducible!**

Categorical to Numerical

Sometimes, data contains categorical values, but the algorithms handle only continouus data. ➔ Turn them into numerical variables, using dummies.

Categorical to Numerical

Sometimes, data contains categorical values, but the algorithms handle only continouus data. ➔ Turn them into numerical variables, using dummies.

- For a binary variable, we just turn the categories into 1 and 0.

Categorical to Numerical

Sometimes, data contains categorical values, but the algorithms handle only continouus data. → Turn them into numerical variables, using dummies.

- For a binary variable, we just turn the categories into 1 and 0.
    - e.g. variable smoker with values yes and no becomes 1 and 0

Categorical to Numerical

Sometimes, data contains categorical values, but the algorithms handle only continouus data. → Turn them into numerical variables, using dummies.

- For a binary variable, we just turn the categories into 1 and 0.
  - e.g. variable smoker with values yes and no becomes 1 and 0
- Variables with C categories can be transformed using **one hot encoding** / **dummy encoding**:

Categorical to Numerical

Sometimes, data contains categorical values, but the algorithms handle only continouus data. ➔ Turn them into numerical variables, using dummies.

- For a binary variable, we just turn the categories into 1 and 0.
  - e.g. variable smoker with values yes and no becomes 1 and 0
- Variables with C categories can be transformed using **one hot encoding** / **dummy encoding**:
  - The variable is split into C / $C - 1$ dummy variables.

Categorical to Numerical

Sometimes, data contains categorical values, but the algorithms handle only continouus data. → Turn them into numerical variables, using dummies.

- For a binary variable, we just turn the categories into 1 and 0.
  - e.g. variable smoker with values yes and no becomes 1 and 0
- Variables with C categories can be transformed using **one hot encoding** / **dummy encoding**:
  - The variable is split into C / $C-1$ dummy variables.
  - One dummy variable corresponds to one of the C values. It is 1 if an instance has the value and 0 else.

One Hot vs. Dummy

Example: feature color, three possible values: red, green, blue.

One Hot vs. Dummy

Example: feature color, three possible values: red, green, blue.

One hot encoding: color → color_red, color_green, color_blue, each with values 1 and 0

One Hot vs. Dummy

Example: feature color, three possible values: red, green, blue.

One hot encoding: color → color_red, color_green, color_blue, each with values 1 and 0

Dummy encoding: color → color_red, color_green, each with values 1 and 0

Observations: In one hot encoding, one variable is dependent on the others. In the example: color_blue $= 1 -$ (color_red $+$ color_green)

Dependent variables are usually undesirable among the features.

One Hot vs. Dummy

Example: feature color, three possible values: red, green, blue.

One hot encoding: color → color_red, color_green, color_blue, each with values 1 and 0

Dummy encoding: color → color_red, color_green, each with values 1 and 0

Observations: In one hot encoding, one variable is dependent on the others. In the example: color_blue $= 1-$ (color_red $+$ color_green)

Dependent variables are usually undesirable among the features.

Before dummyfication: Check the number of categories!

One Hot vs. Dummy

Example: feature color, three possible values: red, green, blue.

One hot encoding: color → color_red, color_green, color_blue, each with values 1 and 0

Dummy encoding: color → color_red, color_green, each with values 1 and 0

Observations: In one hot encoding, one variable is dependent on the others. In the example: color_blue $= 1-$ (color_red $+$ color_green)

Dependent variables are usually undesirable among the features.

💡 Before dummyfication: Check the number of categories!

💡 Warning: With dummy encoding, different distances! E.g.

- the manhattan distance between something blue and something green (all else the same) would be 1
- the distance between something green and something red would be 2.

One Hot vs. Dummy

Example: feature color, three possible values: red, green, blue.

One hot encoding: color → color_red, color_green, color_blue, each with values 1 and 0

Dummy encoding: color → color_red, color_green, each with values 1 and 0

Observations: In one hot encoding, one variable is dependent on the others. In the example: color_blue = 1− (color_red + color_green)

Dependent variables are usually undesirable among the features.

💡 Before dummyfication: Check the number of categories!

💡 Warning: With dummy encoding, different distances! E.g.

- the manhattan distance between something blue and something green (all else the same) would be 1
- the distance between something green and something red would be 2.

💡 When you present data, make sure to include all values!

Scaling

Reasons: Make data comparable, suitable for particular algorithms (e.g. neural networks).

Scaling

Reasons: Make data comparable, suitable for particular algorithms (e.g. neural networks).

Means: Normalization and Standardization

Scaling

Reasons: Make data comparable, suitable for particular algorithms (e.g. neural networks).

Means: Normalization and Standardization

Normalization: Map data into interval $[0, 1]$ (`MinMaxScaler`).

Scaling

Reasons: Make data comparable, suitable for particular algorithms (e.g. neural networks).

Means: Normalization and Standardization

Normalization: Map data into interval $[0, 1]$ (`MinMaxScaler`).

- data in comparable range

Scaling

Reasons: Make data comparable, suitable for particular algorithms (e.g. neural networks).

Means: Normalization and Standardization

Normalization: Map data into interval $[0, 1]$ (`MinMaxScaler`).

- data in comparable range
- preserves differences in variance and mean between features

Scaling

Reasons: Make data comparable, suitable for particular algorithms (e.g. neural networks).

Means: Normalization and Standardization

Normalization: Map data into interval $[0, 1]$ (`MinMaxScaler`).

- data in comparable range
- preserves differences in variance and mean between features

Standardization (z-score transformation): Set mean to 0 and standard deviation to 1

Scaling

Reasons: Make data comparable, suitable for particular algorithms (e.g. neural networks).

Means: Normalization and Standardization

Normalization: Map data into interval $[0, 1]$ (`MinMaxScaler`).

- data in comparable range
- preserves differences in variance and mean between features

Standardization (z-score transformation): Set mean to 0 and standard deviation to 1

Rule of Thumb:

Scaling

Reasons: Make data comparable, suitable for particular algorithms (e.g. neural networks).

Means: Normalization and Standardization

Normalization: Map data into interval $[0, 1]$ (`MinMaxScaler`).

- data in comparable range
- preserves differences in variance and mean between features

Standardization (z-score transformation): Set mean to 0 and standard deviation to 1

Rule of Thumb:

- Use standardization when the data follows a Gaussian distribution.

Scaling

Reasons: Make data comparable, suitable for particular algorithms (e.g. neural networks).

Means: Normalization and Standardization

Normalization: Map data into interval $[0,1]$ (`MinMaxScaler`).

- data in comparable range
- preserves differences in variance and mean between features

Standardization (z-score transformation): Set mean to 0 and standard deviation to 1

Rule of Thumb:

- Use standardization when the data follows a Gaussian distribution.
- Use normalization when differences in variance are relevant.

Evaluation of Data Suitability

- Rather a non-technical process, requires domain expertise

Evaluation of Data Suitability

- Rather a non-technical process, requires domain expertise
- Features might not be as clear as they seem to be, e.g. gender

Evaluation of Data Suitability

- Rather a non-technical process, requires domain expertise
- Features might not be as clear as they seem to be, e.g. gender
- Remove features or values of features (e.g. for ethical reasons).

Evaluation of Data Suitability

- Rather a non-technical process, requires domain expertise
- Features might not be as clear as they seem to be, e.g. gender
- Remove features or values of features (e.g. for ethical reasons).
- Estimate biases from data collection – biased data means biased results.

Evaluation of Data Suitability

- Rather a non-technical process, requires domain expertise
- Features might not be as clear as they seem to be, e.g. gender
- Remove features or values of features (e.g. for ethical reasons).
- Estimate biases from data collection – biased data means biased results.
- Exclusion of unethical features does not mean, the algorithms will make ethical choices!

Evaluation of Data Suitability

- Rather a non-technical process, requires domain expertise
- Features might not be as clear as they seem to be, e.g. gender
- Remove features or values of features (e.g. for ethical reasons).
- Estimate biases from data collection – biased data means biased results.
- Exclusion of unethical features does not mean, the algorithms will make ethical choices!
- When learning on historic datasets, consider the conditions under which that data was gathered. **Especially, when humans created the data!**

Evaluation of Data Suitability

- Rather a non-technical process, requires domain expertise
- Features might not be as clear as they seem to be, e.g. gender
- Remove features or values of features (e.g. for ethical reasons).
- Estimate biases from data collection – biased data means biased results.
- Exclusion of unethical features does not mean, the algorithms will make ethical choices!
- When learning on historic datasets, consider the conditions under which that data was gathered. **Especially, when humans created the data!**
- The same feature can be considered illegitimate (unethical) or legitimate, depending on the context of the experiment. E.g. country of origin or gender:

Evaluation of Data Suitability

- Rather a non-technical process, requires domain expertise
- Features might not be as clear as they seem to be, e.g. gender
- Remove features or values of features (e.g. for ethical reasons).
- Estimate biases from data collection – biased data means biased results.
- Exclusion of unethical features does not mean, the algorithms will make ethical choices!
- When learning on historic datasets, consider the conditions under which that data was gathered. **Especially, when humans created the data!**
- The same feature can be considered illegitimate (unethical) or legitimate, depending on the context of the experiment. E.g. country of origin or gender:
  - not acceptable for most applications

Evaluation of Data Suitability

- Rather a non-technical process, requires domain expertise
- Features might not be as clear as they seem to be, e.g. gender
- Remove features or values of features (e.g. for ethical reasons).
- Estimate biases from data collection – biased data means biased results.
- Exclusion of unethical features does not mean, the algorithms will make ethical choices!
- When learning on historic datasets, consider the conditions under which that data was gathered. **Especially, when humans created the data!**
- The same feature can be considered illegitimate (unethical) or legitimate, depending on the context of the experiment. E.g. country of origin or gender:
  - not acceptable for most applications
  - might be acceptable e.g. for medical conditions

"Biased data
leads to
biased predictions!"

General rule

References

S. Fortmann-Roe.
Understanding the bias-variance tradeoff, 2012.

S. Raschka and V. Mirjalili.
Python Machine Learning.
Packt Publishing Ltd., Livery Place 35 Livery Street Birmingham B3 2PB, UK, second edition, September 2017.