

Mudcard

- **In which situation we should use RMSE over MSE?**
 - Either works
 - I have a slight preference for RMSE over MSE because RMSE has the same unit as the target variable
 - Non-technical audiences have an easier time understanding RMSE in my experience
- **When to use ROC vs PRC**
 - Use the precision recall curve if your classification problem is imbalanced
- **Should we memorize these equations too? I am still trying to understand these metrics but the equations/math = muddy**
 - Yes :)
 - The idea is that once you understand these metrics, you will remember the equations because they will make sense, so you won't just simply memorize them.
- **what should we take care when apply The logloss metric**
 - I'm not sure I understand the question.
 - Could you please expand on this and post on the course forum?
- **just wanted to clarify what it means to have a bad classifier, either in the continuous or categorical case? would good and bad classifiers be symmetric in the sense that if you classify all points wrong you classify all points right, just with some kind of inversion applied?**
 - classification has a categorical target variable
 - regression has a continuous target variable
 - No, if a classifier gives worse than baseline predictions, it is not a good idea to invert the predictions
 - worse than baseline classifiers usually mean there is a bug in your code
 - focus your effort on finding and fixing that bug
 - you never want to deploy a buggy model

Overview of what we covered so far and what's coming

Complications in the ML pipeline:

- if your dataset is non-iid, address it in splitting
 - time series data: use VAR(p), and sklearn's TimeSeriesSplit
 - group structure: use the group based splitters

- if there are categorical features in your dataset, it does not mean that there is group structure!
 - look for ID columns instead
- if a classification problem is imbalanced, address it in splitting
 - use stratified splitting methods
- if your dataset has missing values in categorical and ordinal features, address it in preprocessing
 - treat missing values as another category
 - do not impute, do not drop columns
- if dataset is imbalanced, use evaluation metrics that do not rely on the True Negative element of the confusion matrix
 - no accuracy, no ROC curve for example
 - use precision, recall, f score, p-r curve, etc instead
- if your dataset is imbalanced, be careful about oversampling and undersampling methods - Lecture 17
- missing values in continuous features: Lecture 18-19
- interpretability: Lecture 20-21

The supervised ML pipeline

0. Data collection/manipulation: you might have multiple data sources and/or you might have more data than you need

- you need to be able to read in datasets from various sources (like csv, excel, SQL, parquet, etc)
- you need to be able to filter the columns/rows you need for your ML model
- you need to be able to combine the datasets into one dataframe

1. Exploratory Data Analysis (EDA): you need to understand your data and verify that it doesn't contain errors

- do as much EDA as you can!

2. Split the data into different sets: most often the sets are train, validation, and test (or holdout)

- practitioners often make errors in this step!
- you can split the data randomly, based on groups, based on time, or any other non-standard way if necessary to answer your ML question

3. Preprocess the data: ML models only work if X and Y are numbers! Some ML models additionally require each feature to have 0 mean and 1 standard deviation (standardized features)

- often the original features you get contain strings (for example a gender feature would contain 'male', 'female', 'non-binary', 'unknown') which needs to be transformed into numbers
- often the features are not standardized (e.g., age is between 0 and 100) but it needs to be standardized

4. Choose an evaluation metric: depends on the priorities of the stakeholders

- often requires quite a bit of thinking and ethical considerations

5. Choose one or more ML techniques: it is highly recommended that you try multiple models

- start with simple models like linear or logistic regression
- try also more complex models like nearest neighbors, support vector machines, random forest, etc.

6. Tune the hyperparameters of your ML models (aka cross-validation or hyperparameter tuning)

- ML techniques have hyperparameters that you need to optimize to achieve best performance
- for each ML model, decide which parameters to tune and what values to try
- loop through each parameter combination
 - train one model for each parameter combination
 - evaluate how well the model performs on the validation set
- take the parameter combo that gives the best validation score
- evaluate that model on the test set to report how well the model is expected to perform on previously unseen data

7. Interpret your model: black boxes are often not useful

- check if your model uses features that make sense (excellent tool for debugging)
- often model predictions are not enough, you need to be able to explain how the model arrived to a particular prediction (e.g., in health care)

Non-linear ML algorithms

By the end of this lecture, you will be able to

- Summarize the no free lunch theorem
- Describe how decision trees work
- Describe how a random forest works

Non-linear ML algorithms

By the end of this lecture, you will be able to

- **Summarize the no free lunch theorem**
- Describe how decision trees work
- Describe how a random forest works

Which ML algorithm to try on your dataset?

- **No supervised ML algorithm is universally better than all others** - no free lunch theorem
- **For every learning algorithm, there is a dataset/task on which it fails. Even though another algorithm can successfully learn on it.**
- If we had a universally good algorithm, we all just would use that one algorithm.
- you need to try as many as you can to find the one that performs best
- you will see during the final project presentations that different models perform best for different datasets

Quiz 1

Non-linear ML algorithms

By the end of this lecture, you will be able to

- Summarize the no free lunch theorem
- **Describe how decision trees work**
- Describe how a random forest works

Decision trees

- no free lunch theorem in action!
- deep learning tools are all the rage nowadays but tree-based methods outperform deep learning tools on tabular data
- see NeurIPS paper [here](#)
- we discuss decision trees and random forest today, gradient boosting and XGBoost on Monday
- Decision tree: the data is split according to certain features
- Here is an example tree fitted to data:



No description has been provided for this image

- Trees have nodes and leaves.
 - Nodes are where the split happens
 - Leaves are where we predict the target variable
- The depth of the decision tree is the number of nodes along the longest path from the root node to a leaf.
 - The tree above has a depth of 2.
- Notice that each split is an if-else statement (binary split), and only one feature is split on each node
 - splits with three branches or complex conditions (i.e., split over multiple features in one node) are not done.
- The critical values and features in the nodes are determined automatically by minimizing a cost function.


Non-linear ML algorithms

By the end of this lecture, you will be able to

- Summarize the no free lunch theorem
- Describe how decision trees work
- **Describe how a random forest works**

Random forest

- Random forest: ensemble of independent random decision trees
- Each tree sees a random subset of the training data, that's why the forest is random.
 - each tree is trained on a random subset of features or a random subset of points, or both
- Majority voting among the trees determines the final predicted class
- the fraction of votes is used to calculate predicted probabilities

 No description has been provided for this image

Quiz 2

- **Use the dataset below and create a decision tree with `max_depth = 2` to predict the target variable!**
- **What is your tree's prediction for each person?**
- Remember, you might not be able to find a tree that predicts all points perfectly.
- It just needs to get as many points as possible right.

```
In [20]: # how many features should your team use?
nr_fts = np.random.randint(1,4)
# which features should your team use?
fts_to_use = np.random.randint(0,6,size=nr_fts)
print('use columns with 0-based indices', fts_to_use, 'to create your decis
```

use columns with 0-based indices [0 2 3] to create your decision tree.

X	age	gender (M=0, F=1)	is student?	is parent?	uses computer for work?	nr. of hours on c.	Like computer games?
person 0	5	0	1	0	0	0.0	1
person 1	48	1	0	1	0	1.8	1
person 2	62	0	0	1	0	0.2	0
person 3	10	1	1	0	0	2.4	1
person 4	23	1	1	0	1	4.2	0
person 5	36	0	0	0	1	3.1	1
person 6	12	0	1	0	0	3.1	1
person 7	85	0	0	0	1	1.0	0
person 8	33	1	1	1	0	1.5	0
person 9	56	0	0	0	1	0.1	1

Mud card

In []: