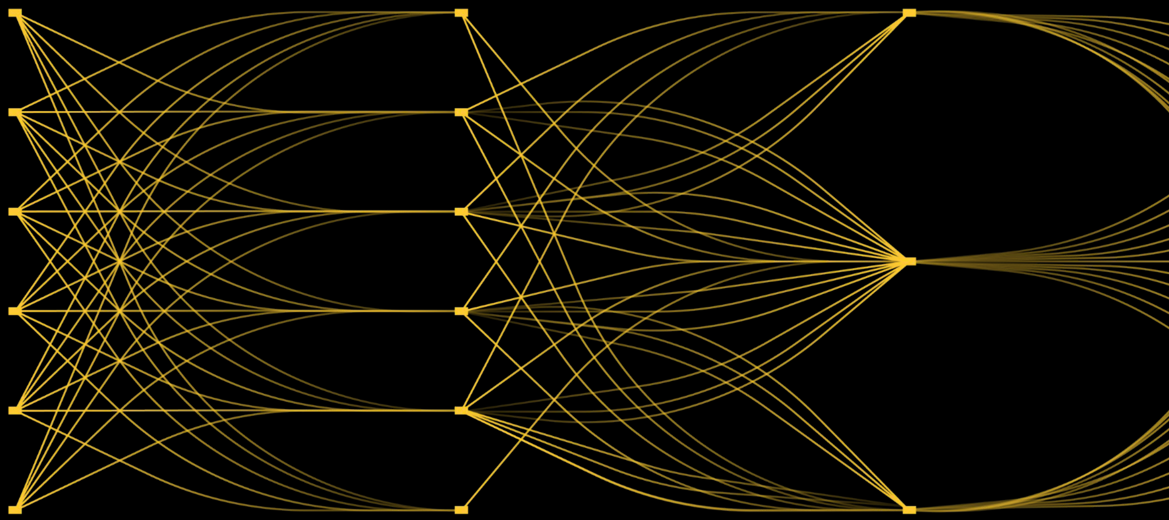
Weights & Biases

Effective MLOps
Model Development



Lesson 3 - Model Evaluation

July 2022

RECAP LESSON 1

Building an End-to-End Prototype

- Understand the Business Context
- Frame the Data Science Problem
- Explore & Understand Your Data
- Establish Baseline Metrics & Models
- Communicate Your Results

Tables

Artifacts

Experiments

Reports

RECAP LESSON 2

Hyperparameter Optimization and Collaborative Model Training

Organize code for experimentation

Conduct experiments

Analyze results of your experiments

Share results and brainstorm with your team

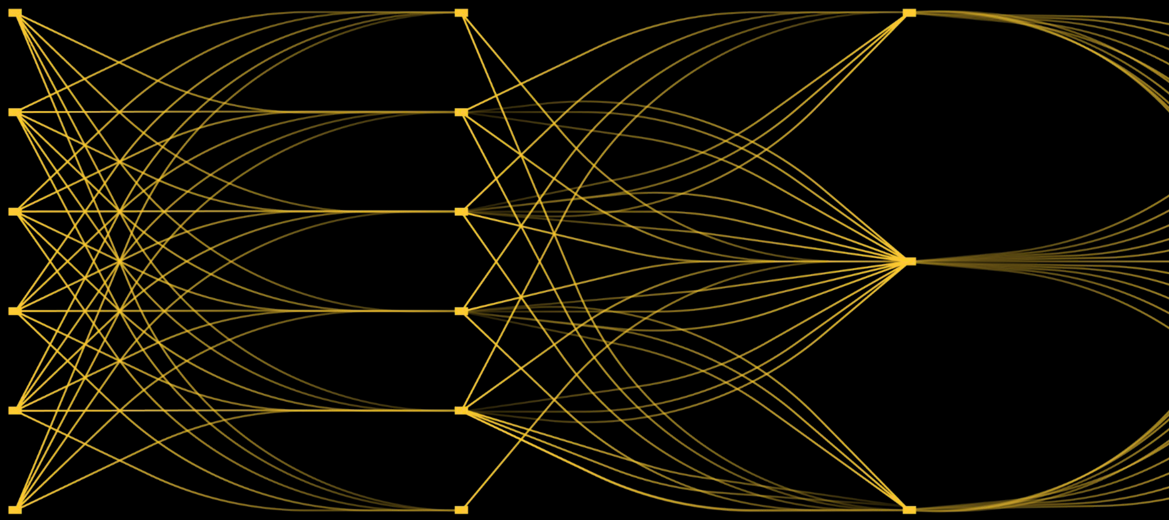
Experiments

Reports

Sweeps

Weights & Biases

Effective MLOps
Model Development



Lesson 3 - Model Evaluation

July 2022

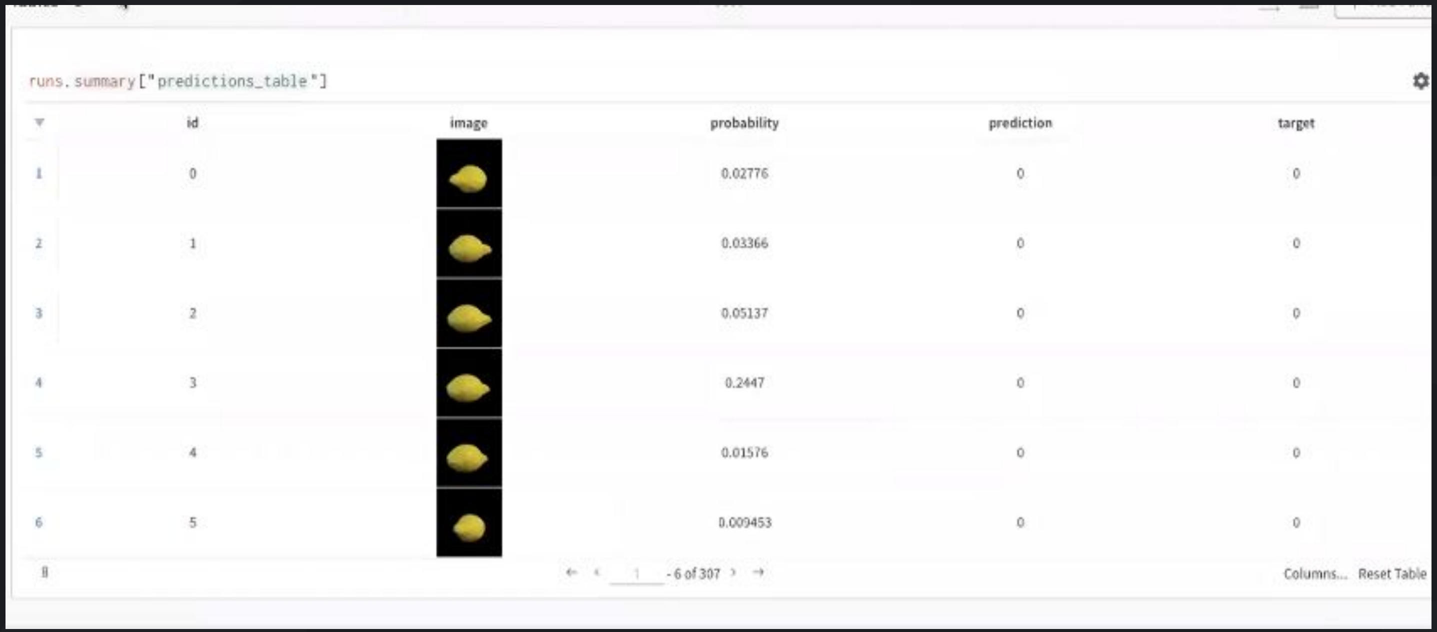
Agenda

We briefly touched on model evaluation in earlier lessons.
However, there are many important aspects of evaluation for you to consider:

01 Partitioning your data

02 Choose an evaluation metric & error analysis

03 Assess business value



runs.summary["predictions_table"]

	id	image	probability	prediction	target
1	0		0.02776	0	0
2	1		0.03366	0	0
3	2		0.05137	0	0
4	3		0.2447	0	0
5	4		0.01576	0	0
6	5		0.009453	0	0

- 6 of 307 Columns... Reset Table

Error Analysis with W&B Reports

Data Partitioning

As a general rule we want to partition our data into three segments

- Training
- Validation (in some cases cross-validation)
- Holdout

However! There are many traps to avoid here. You have to make sure that:

- The partitions are drawn from the same distribution but really validation/test should be like production
- There isn't any data leakage in between partitions

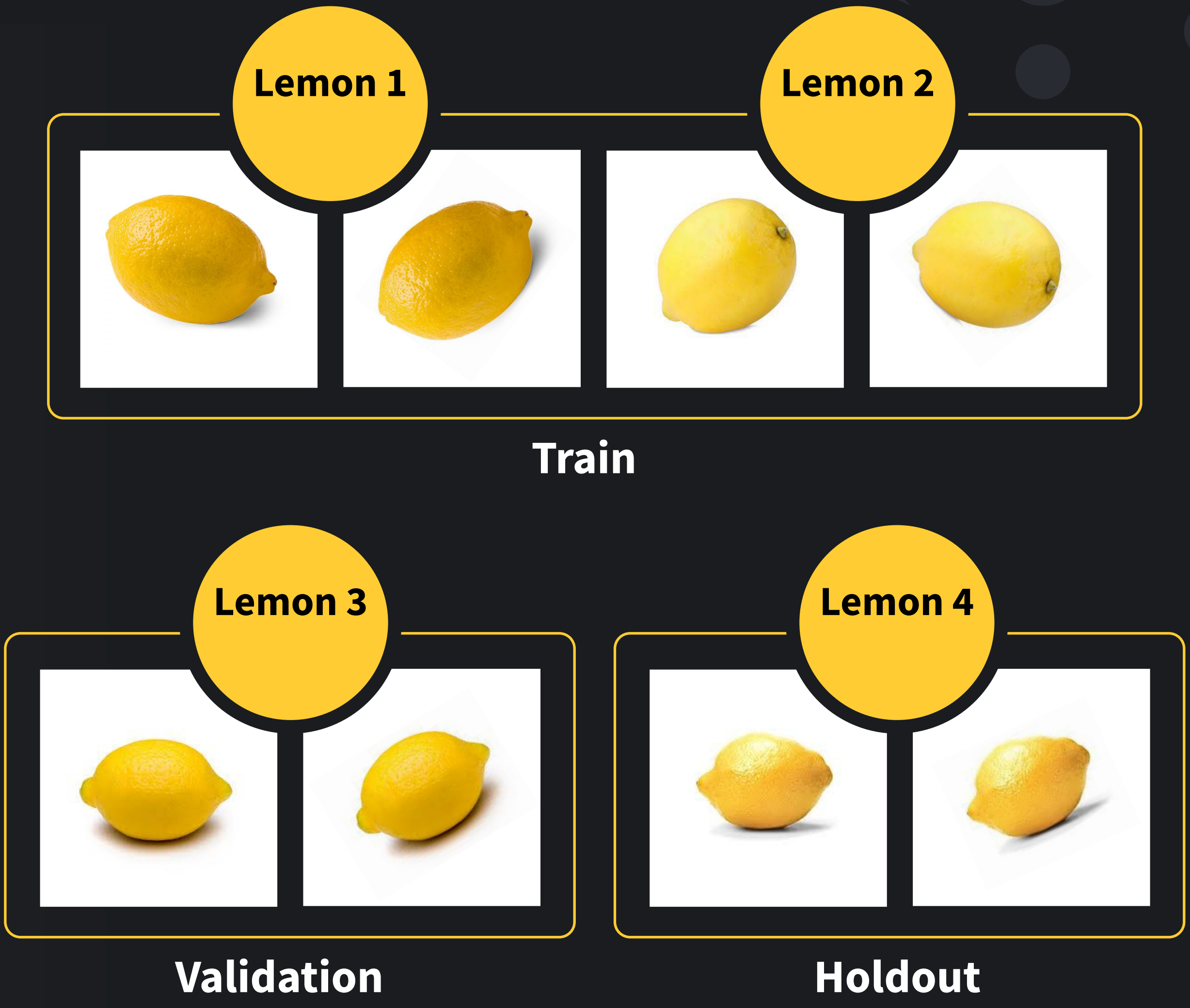
Let's look at some examples.

Data Partitioning: Group Partition

Many times data are not truly independent.

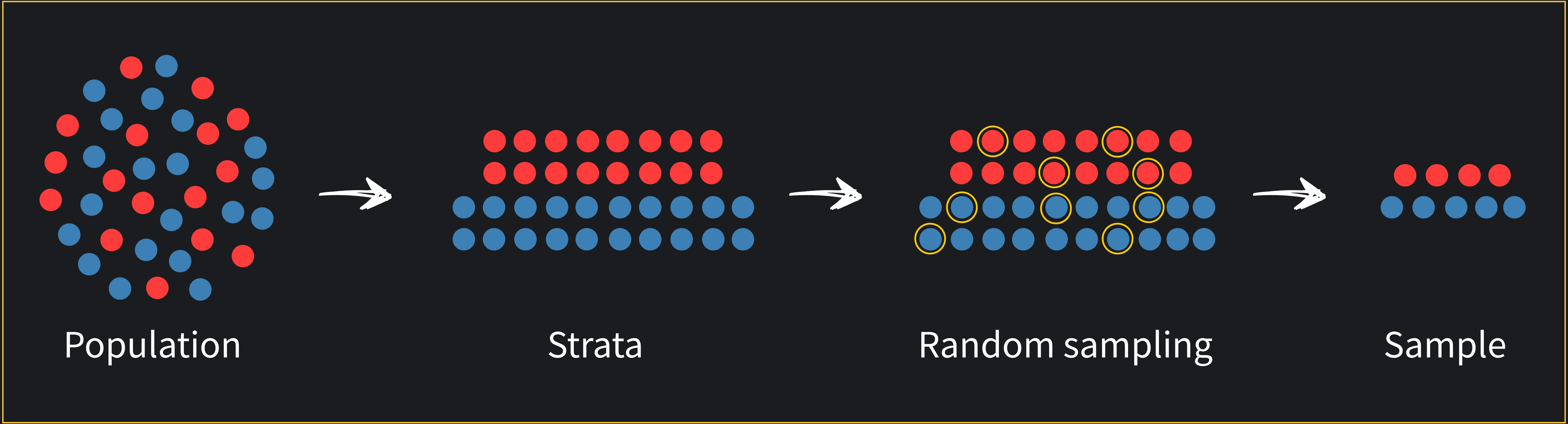
Ex: dataset of lemons, every lemon has multiple pictures w/different angles.

Again, we cannot randomly split our data!



Data Partitioning: Stratified Partition

Often helpful when you have rare labels: Mold vs. Not Mold

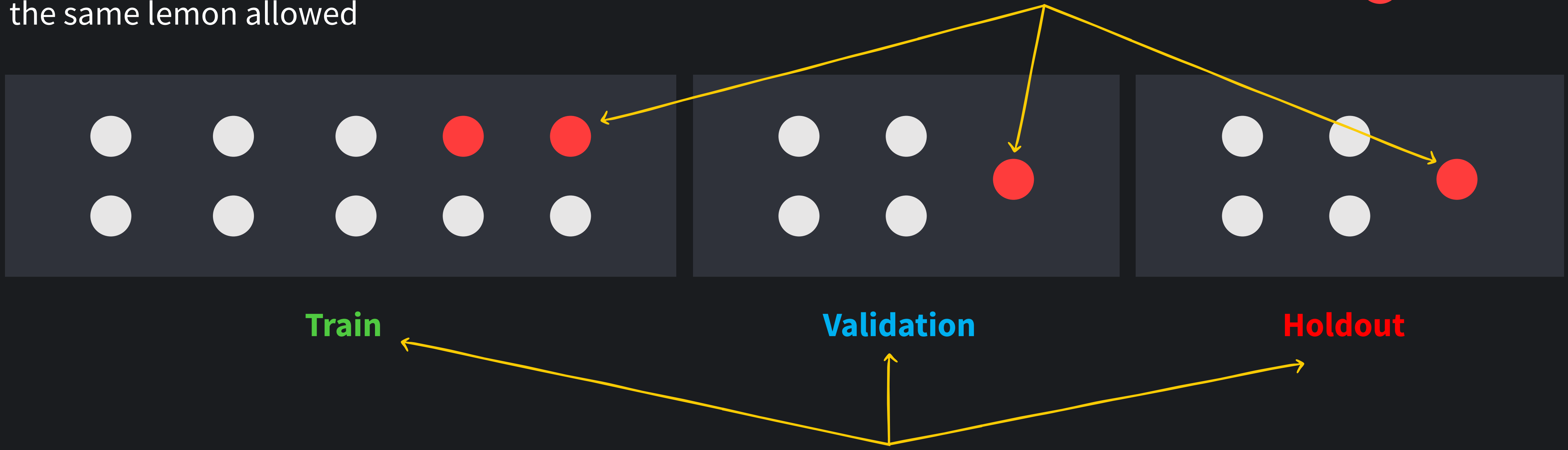


Combining Concepts: Stratified + Group Partition

Prevent data leakage + Enforce that consistent representation of class exist in each fold.

Within a partition multiple pictures of the same lemon allowed

Each partition has same % of moldly lemons



Multiple pictures of the same lemon cannot cross partitions

Code: Stratified + Group Partitioning

prepare_data.py

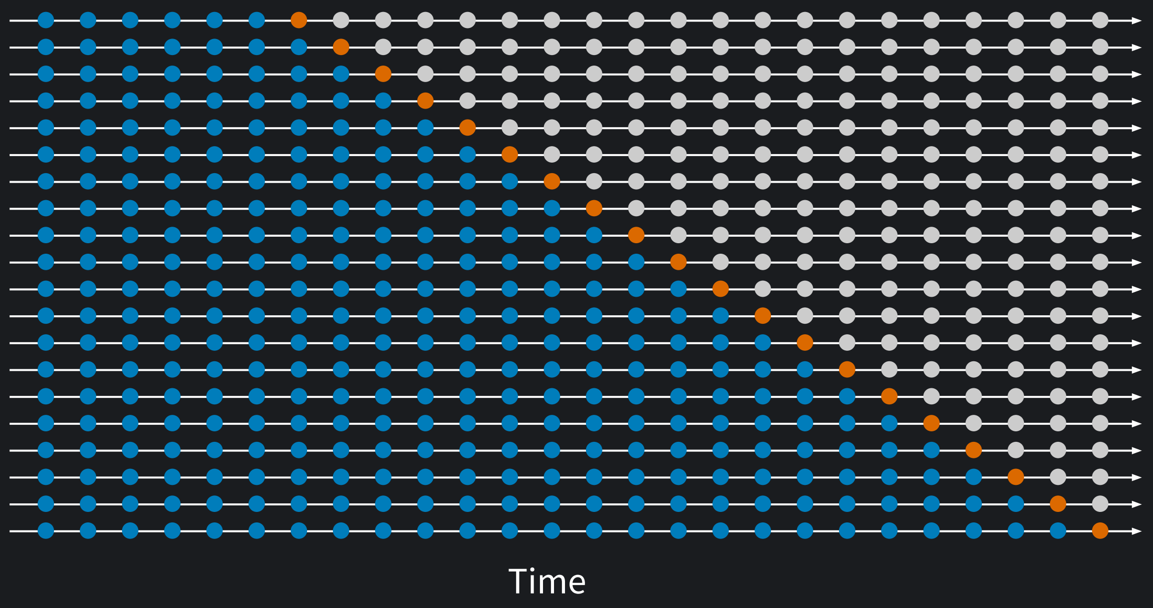
```python
from sklearn.model_selection import StratifiedGroupKFold

X = df.index.values
y = df.mold.values
groups = df.fruit_id.values

cv = StratifiedGroupKFold(n_splits=10, random_state=42, shuffle=True)
for i, (train_idxs, test_idxs) in enumerate(cv.split(X, y, groups)):
    df['fold'].iloc[test_idxs] = i

def split(x): return 'test' if x == 0 else ('valid' if x == 1 else 'train')
df['stage'] = df['fold'].apply(split)
```

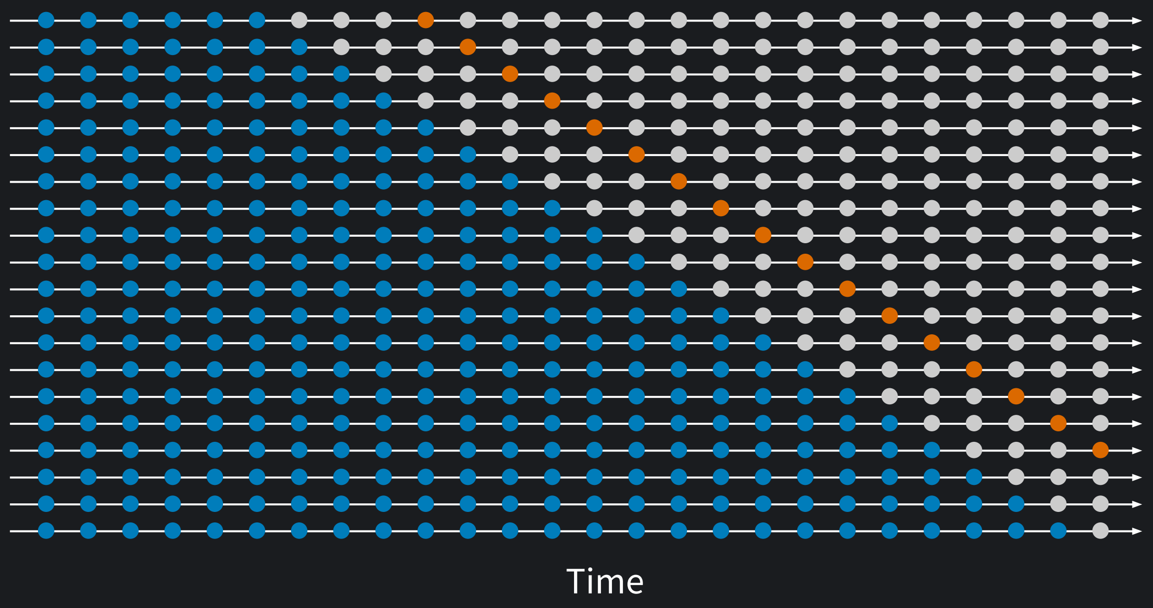
We don't want 10 splits, so we make 2 splits "test" and "valid" and the rest "train"

Data Partitioning: Time Series

You should NOT split your data randomly in time series data. Instead, you want to do time based cross validation. Here's what it looks like:

Each Row is a "fold", split by time. Blue: training, Red: validation Set aside time periods on the right hand side for "holdout"

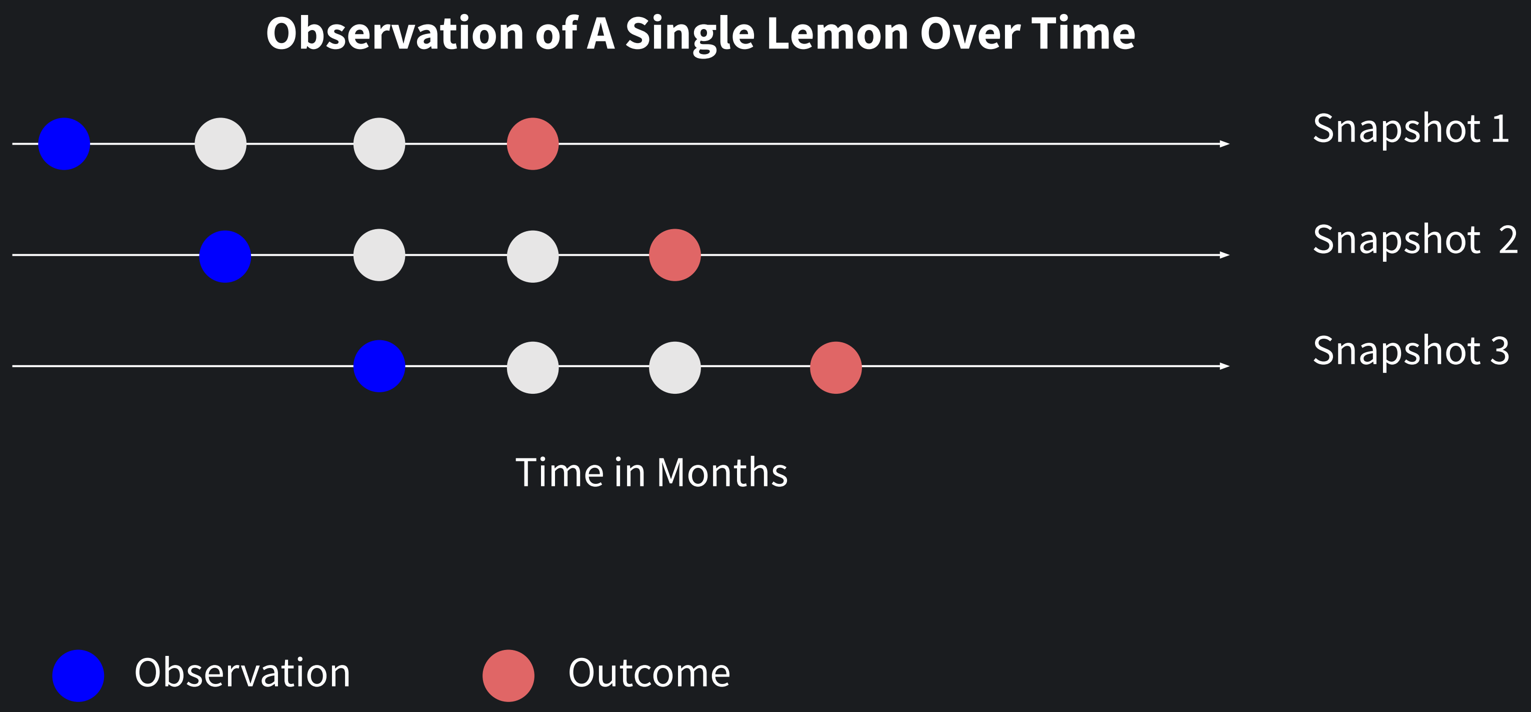




Source: Robert Hyndman, Forecasting: Principles and Practice

Combining Concepts: Time + Group Partition

Be careful to simulate actual conditions when partitioning your data. For example:
Will a lemon develop mold after 2 months? Will look like this



Make sure you think about two things:

- Respecting time
- Information not leaking

Other Types of Data Partitioning

K-Fold Cross Validation:

Generally useful when you don't have as much data, can be more computationally intensive

Random Train/ Validation/Holdout Split

This is common on datasets where there is no time element and now information leakage across examples

Data Partitioning and W&B

Indicate your data partition with a field in your data, to allow you to easily filter and create reports.

Don't peek at the holdout until after you have selected your model!

Code: Logging Validation Predictions in W&B

We can log a table with all the predictions on the validation dataset using `learn.get_preds`

```
inp,preds,targs,out = learn.get_preds(with_input=True, with_decoded=True)
inp.shape, preds.shape, targs.shape, out.shape
```

We will create a Table with 4 columns: (Images, probabilities, targets, predictions)

```
imgs = [wandb.Image(t.permute(1,2,0)) for t in inp] # we need to put as channels last for wandb.Image
pred_proba = preds[:,1].numpy().tolist()
targets = targs.numpy().tolist()
predictions = out.numpy().tolist()
```

we create an intermediate `pd.DataFrame` to then create a Table.

```
preds_df = pd.DataFrame(list(zip(imgs, pred_proba, predictions, targets)),
                        columns =['image', 'probability', 'prediction', 'target'])

run.log({'predictions_table': wandb.Table(dataframe=preds_df)})
run.finish()
```

https://github.com/wandb/edu/blob/main/model-dev-course/lesson1/baseline_classifier.ipynb

Choosing An Evaluation Metric

You want to pick a metric that is related to business outcomes. Some rules of thumb:

Pick a single-number evaluation metric.

Precision and Recall is not a single number, ex: use F1 score instead

Try not to combine many different metrics. Where possible, determine a minimum threshold for other metrics, and choose one metric to optimize.

Ex: Latency (min threshold) vs Accuracy (optimize)

Model Selection Metric For Lemon Mold: Log Loss

We also log Precision, Recall and F1 Score as those provide useful context

Robust to imbalanced classes: (mold vs not mold)

Incentivizes model to produce probabilities closer to the ground truth, which is useful for model calibration.

Is *somewhat* human interpretable, but that's ok. (Lower the better)

How To Log Metrics in W&B

We use convenient W&B callbacks

fastai.py

```python
learn = vision_learner(dls,
                       cfg.arch,
                       metrics=[accuracy, Precision(), Recall(), F1Score()],
                       cbs=[WandbCallback(log_preds=False, log_model=True),
                            SaveModelCallback(monitor='f1_score')]
                      )


learn.fine_tune(2)
```

WandbCallBack logs all metrics

https://github.com/wandb/edu/blob/main/model-dev-course/lesson1/baseline_classifier.ipynb

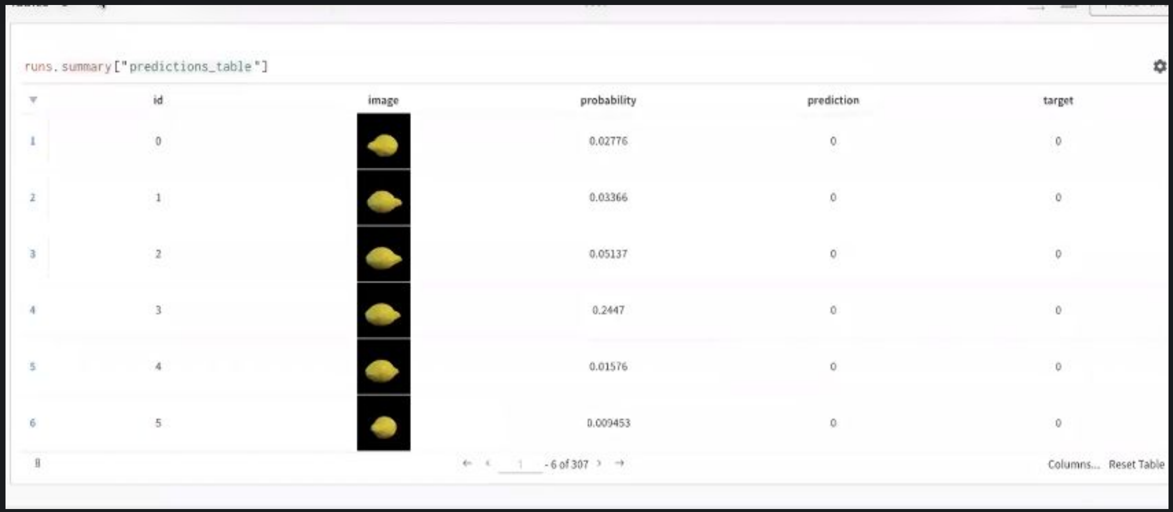
Error Analysis

Look at your validation errors to gain intuition on where your model is failing. W&B Tables can facilitate this process.

01 Look at ~ 50 examples where model is confident but wrong (the error is high) and ~ 50 examples where model is wrong but not as confident.

02 Try to create categories of why the model is wrong and bucket these items. Example: poor lighting, obstruction, etc.   This can help you zone in on the biggest areas of opportunity.

03 Fix incorrect labels or remedy issues. You will often find that many times there are just incorrect labels in your training set.



Walkthrough: Error Analysis Using W&B Tables

Look at your validation errors to gain intuition on where your model is failing. W&B Tables can facilitate this process.

```
runs.summary["predictions_table"]
```

	image	probability	prediction	target	prediction != target	target - probability ↓
102		0.0076	0	1	True	0.9924
179		0.02049	0	1	True	0.9795
189		0.05372	0	1	True	0.9463
183		0.07028	0	1	True	0.9297
184		0.3094	0	1	True	0.6906
187		0.5265	1	1	False	0.4735

← ‹ 1 - 6 of 210 › → Columns... Reset Table

https://wandb.ai/wandb_course/lemon-project/runs/zxum4ef0

Assesing Your Model’s Business Value

1. Human Baseline
2. ML Baseline:
    a. What are the costs when model
        i. False Positives (Predict mold when no mold)
        ii. False Negatives (Predict no mold when there is mold)

Since this model’s function is risk mitigation (identifying mold), we can express business value in terms of reducing costs.

Q: Does our model save the company money by reducing labor and/or being able to identify mold w/higher accuracy than a human?

Human Baseline Total Cost	ML Baseline Total Cost
= Labor (Lemon Inspectors) + Mistakes	= Model Maintenance/Tooling + Mistakes

Using the W&B API to download predictions

W&B entity

Project name

get_val_preds.py

```python
import wandb
import pandas as pd
from fastcore.all import Path

def val_pred_table(run_id, entity='wandb_course', proj='lemon-project'):
    "Get prediction table on the validation set for the lemon project."
    api = wandb.Api()
    path = api.artifact(f'{entity}/{proj}/run-{run_id}-predictions_table:v0').download()
    preds = (Path(path)/'predictions_table.table.json').read_json()
    return pd.DataFrame([{'pred':p, 'label':t}  for _,p,_,t in preds['data']])


if __name__ == '__main__':
    preds = val_pred_table('zxum4ef0')
    preds.to_excel(Path('Validation_Predictions.xlsx'))
```

Run id

https://gist.github.com/hamelsmu/f1989fb35fd5c1f2935dcefe41cbfe64

Model Calibration & Business Value

Threshold	0.3	Num Lemons		1,500,000
False Positive Rate	1.64%	FP Cost / Lemon	$	0.25
False Negative Rate	14.8%	FN Cost / Lemon	$	1.50
% Mold	12.9%			

Cost Benefit Analysis

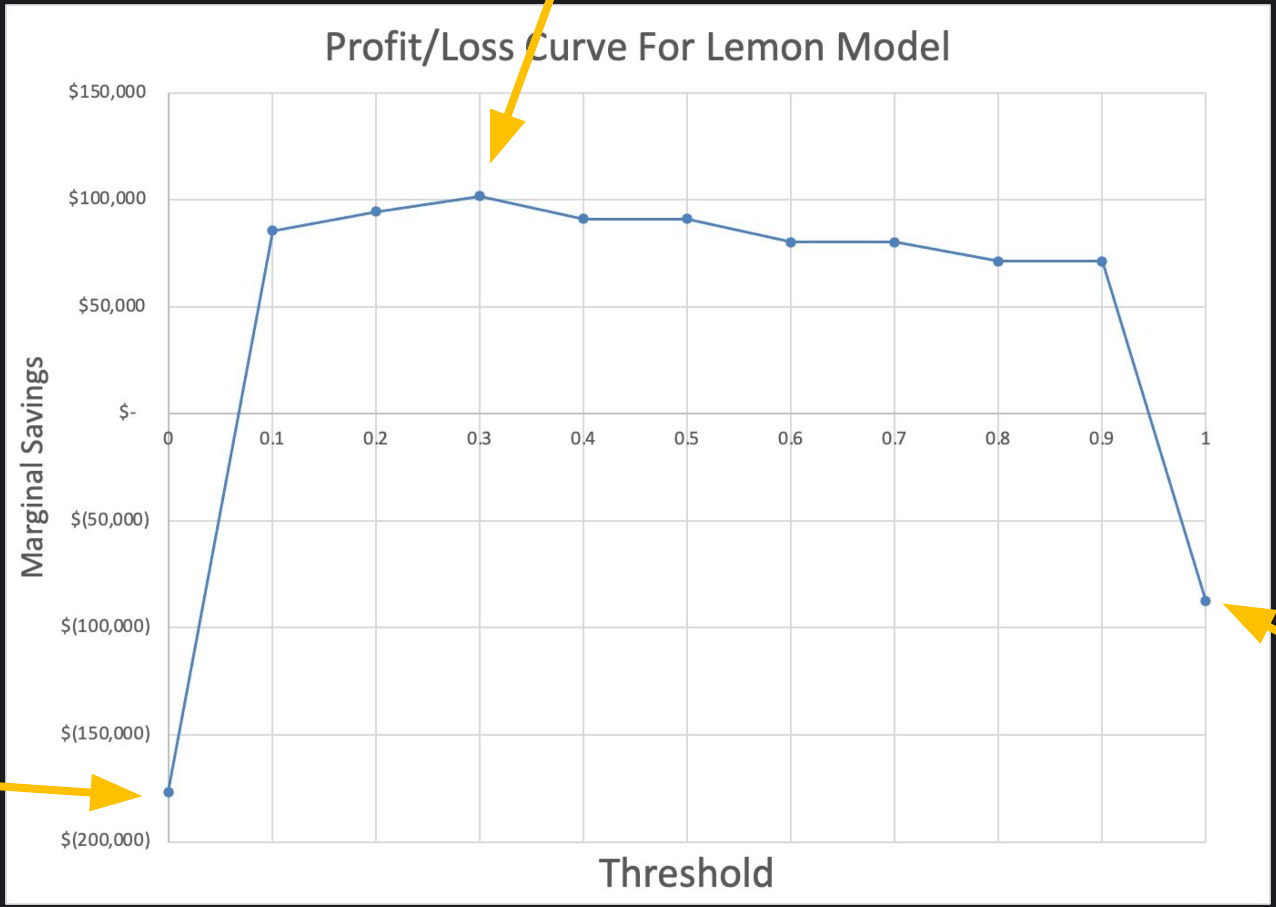
New Avg Cost Per Lemon	**$**	**0.032**
Old Avg Cost Per Lemon	**$**	**0.100**
Savings Per Lemon	$	0.07
Total Savings	**$**	**101,786**

Pred Probability	Label	Decision	False Positive	False Negatives	Cost
0.020070542	0	0	0	0	$ -
0.034065075	0	0	0	0	$ -
0.006420959	0	0	0	0	$ -
0.056603111	0	0	0	0	$ -
0.024350833	0	0	0	0	$ -
0.059544567	0	0	0	0	$ -
0.006659458	0	0	0	0	$ -
0.016839102	0	0	0	0	$ -
0.020861411	0	0	0	0	$ -
0.004080259	0	0	0	0	$ -
0.01576495	0	0	0	0	$ -
0.051844362	0	0	0	0	$ -
0.016569775	0	0	0	0	$ -
0.004659445	0	0	0	0	$ -
0.024644258	0	0	0	0	$ -
0.011145318	0	0	0	0	$ -
0.005516733	0	0	0	0	$ -
0.007749026	0	0	0	0	$ -
0.01884055	0	0	0	0	$ -
0.006060779	0	0	0	0	$ -
0.020482956	0	0	0	0	$ -
0.006485218	0	0	0	0	$ -

https://www.dropbox.com/s/apox0e22e8lk9wu/Validtation_Predictions_Simulation.xlsx?dl=0

Profit Curve: Threshold vs. Business Value

Optimal threshold is ~ .3: its okay to throw away some lemons at the expense of catching more mold.



Extreme case where you decide every lemon has mold

Extreme case where you decide NO lemon has mold

Model Registry & Evaluating On Test Set

1. Get the model from the W&B model registry

2. Get the parameters from the associated run

3. Setup Data Loader & Model

```
run = wandb.init(project=cfg.PROJECT_NAME, entity=cfg.ENTITY, job_type="evaluation", tags=['staging'])

artifact = run.use_artifact('wandb_course/model-registry/Lemon Mold Detector:v0', type='model')
artifact_dir = artifact.download()
model_path = Path(artifact_dir).absolute() / 'model'

producer_run = artifact.logged_by()
cfg.img_size = producer_run.config['img_size']
cfg.bs = producer_run.config['bs']
cfg.arch = producer_run.config['arch']

wandb.config.update(cfg)

df, path = prepare_data(cfg.PROCESSED_DATA_AT)

dls = ImageDataLoaders.from_df(df, path=path,
                               fn_col='file_name',
                               label_col=cfg.target_column,
                               valid_col='valid',
                               item_tfms=Resize(cfg.img_size),
                               bs=cfg.bs
                              )
learn = vision_learner(dls,
                       cfg.arch,
                       metrics=[accuracy, Precision(), Recall(), F1Score()])

learn.load(model_path)

_, val_accuracy, val_precision, val_recall, val_f1 = learn.validate(ds_idx=1)
_, tst_accuracy, tst_precision, tst_recall, tst_f1 = learn.validate(ds_idx=0)
```

4. Perform Inference

Model Management

Manage the model lifecycle from training to production

W&B is a central system of record for model development, enabling you and your team to collaboratively manage the lifecycle of machine learning models, covering three key use cases: Model Versioning, Model Lineage, and Model Lifecycle.

Model Registry

Browse & discover all the models you have access to across all teams and organizations.

- Create new Model Collections for your team to collaborate.
- Search across all teams and organizations.
- View Action History audit log to see membership and status history.

Model Versioning

Iterate to get the best model version for a task, and catalog all the changes along the way.

- Track every model version in a central repository
- Browse and compare model versions
- Capture training metrics and hyperparameters

Model Lineage

Document and reproduce the complete pipeline of model training and evaluation.

We will do a demo, but also see the docs: https://docs.wandb.ai/guides/models

What to do when metrics on Test set are very different?

1. Look for data leakage: did you split your data in the right way?

2. Is your holdout set different in some way than other partitions?   Advanced: ML can help you identify how -> Adversarial Validation

3. Noise: is there lots of natural variation in your data? Might want to get a revised error estimate using 5+ fold CV to get a range of errors instead of a point estimate. Now is test within that range?

If you do find an issue, you must re-partition your data and start the modeling process over again.

Demo:
Model Registry & Test Set Eval

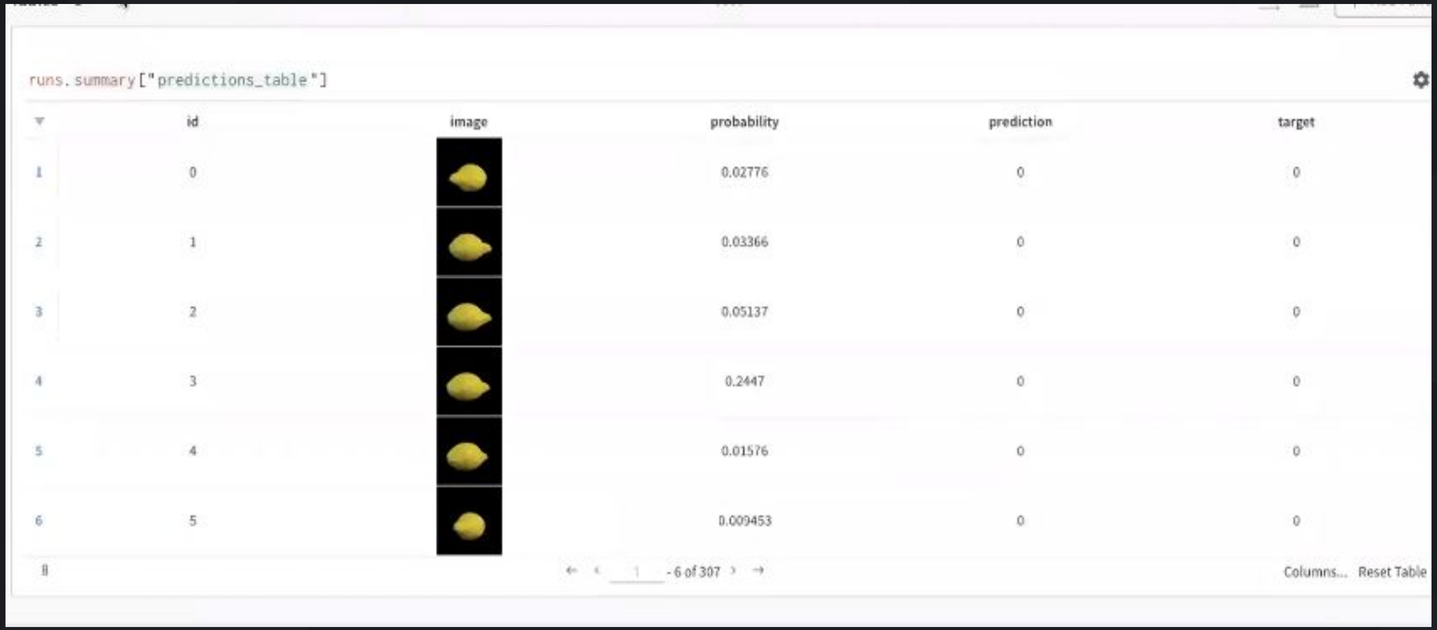
Recap

Model evaluation starts with careful attention to partitioning your data, followed by error analysis. Many people forget to consider business value, but this is also important.

01 Partitioning your data

02 Choose an evaluation metric & error analysis

03 Assess business value



runs.summary["predictions_table"]

	id	image	probability	prediction	target
1	0		0.02776	0	0
2	1		0.03366	0	0
3	2		0.05137	0	0
4	3		0.2447	0	0
5	4		0.01576	0	0
6	5		0.009453	0	0

1 - 6 of 307   Columns...   Reset Table

Error Analysis with W&B Reports

Final Assignment

- Decide an on the model evaluation approach for your project
- Prepare final project report, including:
    - Description of the problem
    - Summary of experiments and findings
    - Model evaluation approach
- Post link to report in course discord channel by Monday, July 18th
- Next week: Presentation of selected reports