DATA SCIENCE FESTIVAL

10 Data Science Uberhacks to Turbocharge your Workflow!!

Codenode, London
20/05/2023

Tom Ewing
Head of AI & Data Engineering
Station10

tom-s10

@TomEwingS10

Station10

Firstly… sorry about the Clickbaity title.

Things this presentation was nearly called...

Big Tech **HATES** these 10 Data Science Hacks!

Things this presentation was nearly called…

10 Reasons why your Data Science Workflow **SUCKS**!

Things this presentation was nearly called…

Multi-zillionaire tech-ninja open sources the 10 secrets that made him his **FORTUNE**!

About Me.

Tom Ewing
Head of AI & Data Engineering
Station10

+ 20 years experience in Data
+ 8 years experience in Data Science, Data Engineering & Machine Learning Engineering
+ Hands-on practitioner
+ I love “hacks”
+ Contributor to DSF since 2016

What Station10 do.

Digital Analytics

Implementing tried & tested tools and strategies to maximise the earning potential of your website.

Strategy & Insight

Revealing patterns and insights to better support data-driven decision making.

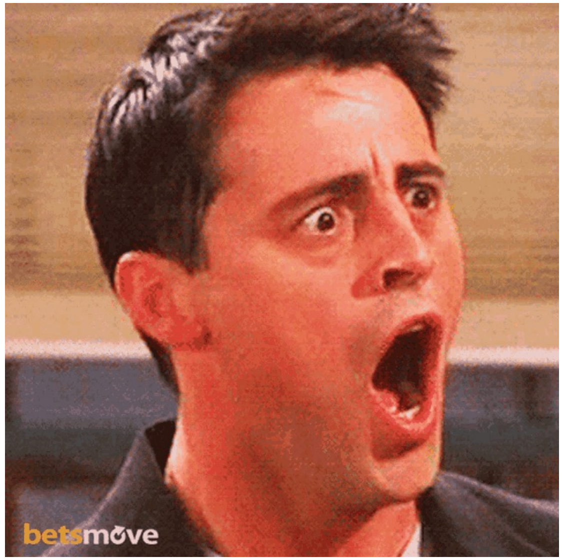
AI & Data Engineering

Creating and embedding AI & data services and analysis to drive change, efficiency and value.

"Creating value from data"

What this presentation is.

+ Gave a talk last year on Dask…
+ A lot of people there didn't know about PySpark's game-changing pandas-style API!



What this presentation is.

- This got me thinking… What else might people not know about?
- Tech moves quickly!
- There's things I like and use, but I can't fill 30 mins talking about one of them…
  So, maybe talk about all of them?!

The rules.

I’ve come up with 10 “things” that:

+ Can make specific jobs or tasks around AI easier
+ You might not have heard of, or not be using
+ Are easy to pick up and integrate.
+ Are (mostly) Python-based.
+ Can showcased in 2 minutes or less.

What this presentation is NOT.

- A “deep dive” into anything.
- About “hacking” outcomes or circumventing established AI processes.
- Anything to do with Chat-GPT.

Introducing the Uberhacks!

1. Github Awesomeness
Awesome curated content.

2. D-Tale
Eyeball data easily.

3. Ydata-profiling
EDA as a Service.

4. thefuzz
Fuzzy string matching.

5. UK Open Data
So. Much. Data.

6. Yellowbrick
Easy AI Visualisation.

7. Shap
AI explainability.

8. Fairlearn
Non-discriminatory AI.

9. Metaflow
Easy Pipelines.

10. Make
CLI maker-easier.

1. Github Awesomeness

Awesome curated content.

1. Github Awesomeness Intro.

Because Google is always trying to sell you stuff.

- Community curated content around a subject area (e.g. Machine Learning, Python, Data Engineering etc.) on Github.
- Contain a list of links to packages, repos, sites, research papers… resources!
- Wide ranging
- Example: awesome-production-machine-learning

1. Github Awesomeness pros & cons.

The Good…

- ✓ A great starting point for research
- ✓ Open Source centric
- ✓ Tend to be exhaustive (if updated)
- ✓ Quicker than Googling
- ✓ Opportunity to find more Uberhacks!

The not so Good…

- ✗ Some topics are repeated across different lists.
- ✗ Some lists aren’t updated regularly, so be sure to check.

2. D-Tale

Eyeball data easily.

2. D-Tale Intro.

Excel but way better.

+ Python package that generates an interactive dashboard to explore your data from your Notebook.
+ Has a range of exploration, visualisation and analysis options, tailored for ML use cases.
+ Viewed in the browser or a notebook.

Example: [Live D-Tale demo](#)(House Prices)

2. D-Tale example.

Installation (CLI)

```
conda install dtale
```

Execution (Python)

```python
import dtale

d = dtale.show(df)
d.open_browser()
```

2. D-Tale pros & cons.

The Good…

- ✓ Blazing fast as it loads what it needs dynamically
- ✓ View the whole dataframe not just the .head( )
- ✓ Great breadth of functionality
- ✓ Does charting & dashboarding
- ✓ Very easy to use

The not so Good…

- ✗ Doesn't work well in the cloud.
- ✗ Non-Python syntax for filtering which takes some getting used to.
- ✗ Crashes sometimes, particularly on larger datasets.
- ✗ Lots of dependencies

3. Ydata-profiling

EDA as a Service.

3. Ydata-profiling Intro.

EDA done for you

+ The package formerly known as pandas-profiling
+ D-Tale = low level, Ydata-profiling = high level.
+ Shows data on duplicates, missing vlaues, aggregations, correlations, statistics etc.

Example: Live Ydata-profiling example (Titanic)

3. Ydata-Profiling example.

Installation (CLI)

```
conda install ydata-profiling
```

Execution (Python)

```
From ydata_profiling import ProfileReport

profile = ProfileReport(df)
profile.to_file(“my_report.html”)
```

3. Ydata-profiling pros & cons.

The Good…

- ✓ Also, two lines of code!
- ✓ Saves time plotting and correlating
- ✓ Simple & explorable with a sharable output
- ✓ Has modes for dataset comparison and time series too.
- ✓ Makes it look like you've done more work than you actually have 😎

The not so Good…

- ✗ Slow particularly on larger datasets (but can now run on Spark)
- ✗ Limited scope

4. thefuzz

Fuzzy string matching.

4. thefuzz Intro.

Easy fuzzy joins in Pandas (or anything)

- Python package that measures and matches similarity between two string elements.
- Wide range of matching options including comparison, partial and token (sentence) matching
- Can be used to join two dataframes with similar but non-identical keys

4. thefuzz examples.

Installation (CLI)

```
conda install thefuzz
```

Execution (Python)

```python
from thefuzz import fuzz
from thefuzz import process



>>> fuzz.ratio("this is a test", "this is a test!")
97

>>> fuzz.partial_ratio("this is a test", "this is a test!")
100

>>> fuzz.token_sort_ratio("fuzzy wuzzy was a bear", "wuzzy fuzzy was a bear")
100

>>> choices = ["Atlanta Falcons", "New York Jets", "New York Giants", "Dallas Cowboys"]
>>> process.extract("new york jets", choices, limit=2)
[('New York Jets', 100), ('New York Giants', 78)]
```

4. thefuzz pros & cons.

The Good…

- Simple & effective
- Quick (considering…)
- Makes “dirty joins” possible with a degree of control
- Versatile and can be used outside of pandas

The not so Good…

- Slow if your data is large
- Need to be careful and check the joins

5. UK Open Data

So. Much. Data.

5. UK Open Data Intro.

Open data can make your stuff better!

+ UK has a wide range of open data particularly from central statistical authorities and government (ONS, Gov.Scot, NIRSA, Gov.uk etc.)

+ A multitude of areas (economic, demographic, housing, deprivation, transport etc.) that can enhance outcomes.

+ 2021 / 2022 Census data is presently "fresh"

5. ONS Open Data pros & cons.

Office for National Statistics

The Good…

- ✓ Enables per-capita comparisons between areas
- ✓ Huge breadth of data
- ✓ Deep granularity
- ✓ A range of geographic breakdowns
- ✓ Well supported by the ONS Geoportal

The not so Good…

- ✗ ONS website isn't user friendly
- ✗ Need some geographic knowledge to merge to existing data
- ✗ Time consuming to collate and maintain
- ✗ Disparate sources across England & Wales, Scotland and Northern Ireland

6. Yellowbrick

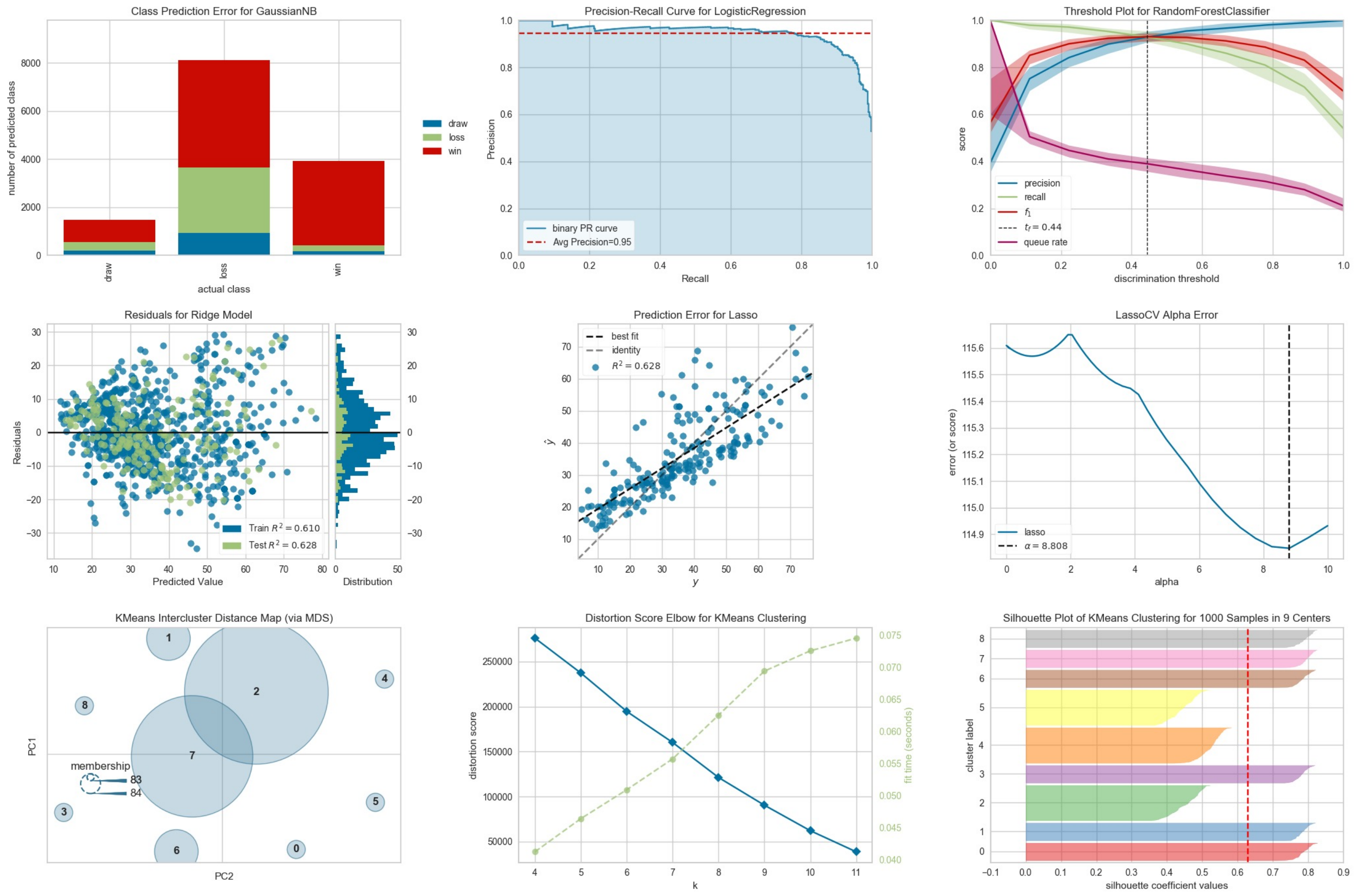
Easy AI Visualisation.

6. Yellowbrick Intro.

Because C+P matplotlib is so 2017…

- Python package that visualises AI algorithms.
- Extends Scikit-Learn
- Covers regression, classification, clustering and much more.

6. Yellowbrick gallery.



6. Yellowbrick examples.

Installation (CLI)

```
conda install -c districtdatalabs yellowbrick
```

Execution (Python)

```python
from yellowbrick.classifier import ClassificationReport
from sklearn.linear_model import LogisticRegression

model = LogisticRegression()
visualizer = ClassificationReport(model)

visualizer.fit(X_train, y_train)
visualizer.score(X_test, y_test)
visualizer.show()
```

6. Yellowbrick pros & cons.

The Good…

- ✓ Beautiful
- ✓ Wide range of visualisers
- ✓ Simple & works out of the box
- ✓ Excellent prototyping tool
- ✓ Excellent documentation & examples gallery

The not so Good…

- ✗ SKL only
- ✗ “Wrapper” may cause issues if using in conjunction with other wrappers
- ✗ Sometimes you have to train twice
- ✗ Visualisers aren’t easily customisable

7. Shap

AI Explainability.

7. Shap Intro.

Explainability isn't just for stakeholders…

+ Uses game theory to explain the predictions of any model.

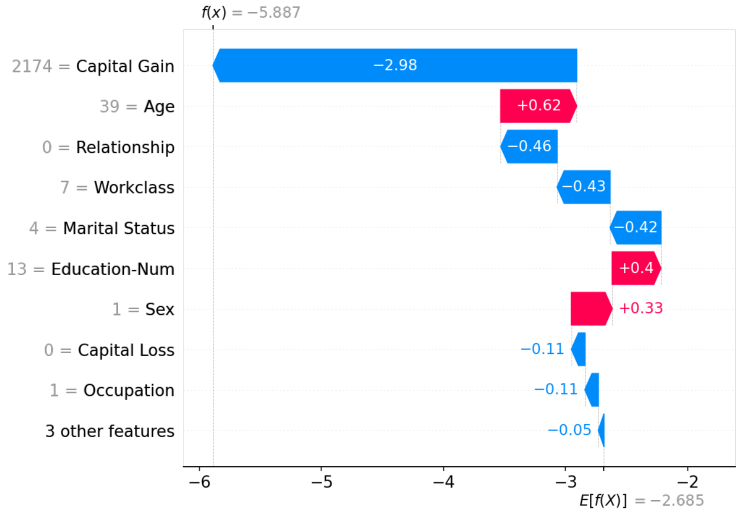
+ Calculates the contribution of each feature for each record in the overall prediction data.

+ This uncovers complex relationships between the features and the target and uncovers how models are working "under the hood".

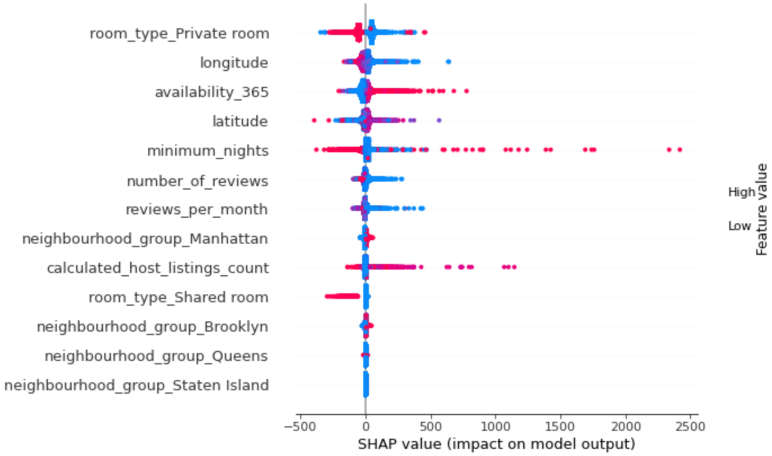
+ Reveals where features do well and badly in the data.

7. Shap gallery.

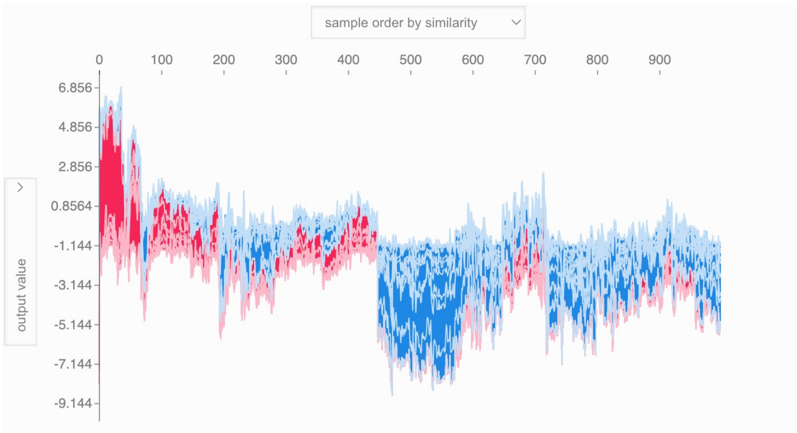
Waterfall Plot



Summary Plot



Force Plot



Red = Positive contribution

Blue = Negative contribution

7. Shap examples.

Installation (CLI)

```
conda install shap
```

Execution (Python)

```
# Initialise the explainer & Calculate Shap values
explainer = shap.Explainer(model, X_train)
shap_values = explainer(X_test)

# Waterfall plot on a single record
shap.plots.waterfall(shap_values[0])

# Summary plot for all records
shap.summary_plot(shap_values, X_test)

# Force plot
shap.force_plot(explainer.expected_value, shap_values, X_test)
```

7. Shap pros & cons.

The Good…

- ✓ Simple to use…
- ✓ Works with most major AI libraries
- ✓ Interactive plotting in your notebook
- ✓ Beautiful
- ✓ Excellent for images

The not so Good…

- ✗ Complex to master!
- ✗ Highly mathematical
- ✗ Computationally expensive

8. Fairlearn
Non-discriminatory AI.

8. Fairlearn Intro.

AI Regulation is coming!

+ Python package that helps identify and mitigate unfairness in AI models.
+ Has two components:
    + **Metrics** for identifying which groups are negatively impacted by a model
    + **Algorithms** for mitigating any unfairness

8. Fairlearn examples.

Installation (CLI)

```
conda install fairlearn
```

Execution (Python)

```python
# Metric (Gender)

from fairlearn.metrics import MetricFrame
from sklearn.metrics import accuracy_score
from sklearn.tree import DecisionTreeClassifier

classifier = DecisionTreeClassifier(min_samples_leaf=10, max_depth=4).fit(X, y_true)

y_pred = classifier.predict(X)
mf = MetricFrame(
    metrics=accuracy_score,
    y_true=y_true,
    y_pred=y_pred,
    sensitive_features=sex
)

mf.overall
mf.by_group
```

8. Fairlearn examples.

Execution (Python)

```python
# Algorithm (Gender)

from fairlearn.reductions import (
    ExponentiatedGradient,
    DemographicParity
)

constraint = DemographicParity()
classifier = DecisionTreeClassifier(min_samples_leaf=10, max_depth=4)

mitigator = ExponentiatedGradient(classifier, constraint).fit(X, y_true, sensitive_features=sex)

y_pred_mitigated = mitigator.predict(X)

sr_mitigated = MetricFrame(metrics=selection_rate, y_true=y_true, y_pred=y_pred_mitigated, sensitive_features=sex)

sr_mitigated.overall
sr_mitigated.by_group
```

8. Fairlearn pros & cons.

The Good…

- ✓ Integrates with major AI packages (SKL, PyTorch etc.)
- ✓ Wide range of fairness metrics & algorithms
- ✓ Great documentation
- ✓ Integrated into Azure
- ✓ It’s ethical and the right thing to do

The not so Good…

- ✗ Another “Wrapper”
- ✗ Highly mathematical
- ✗ Computationally expensive
- ✗ Algorithms need data on gender, ethnicity, religion, immigration etc.
- ✗ Be prepared to take a hit on model performance

9. Metaflow

Easy Pipelines.

9. Metaflow Intro.

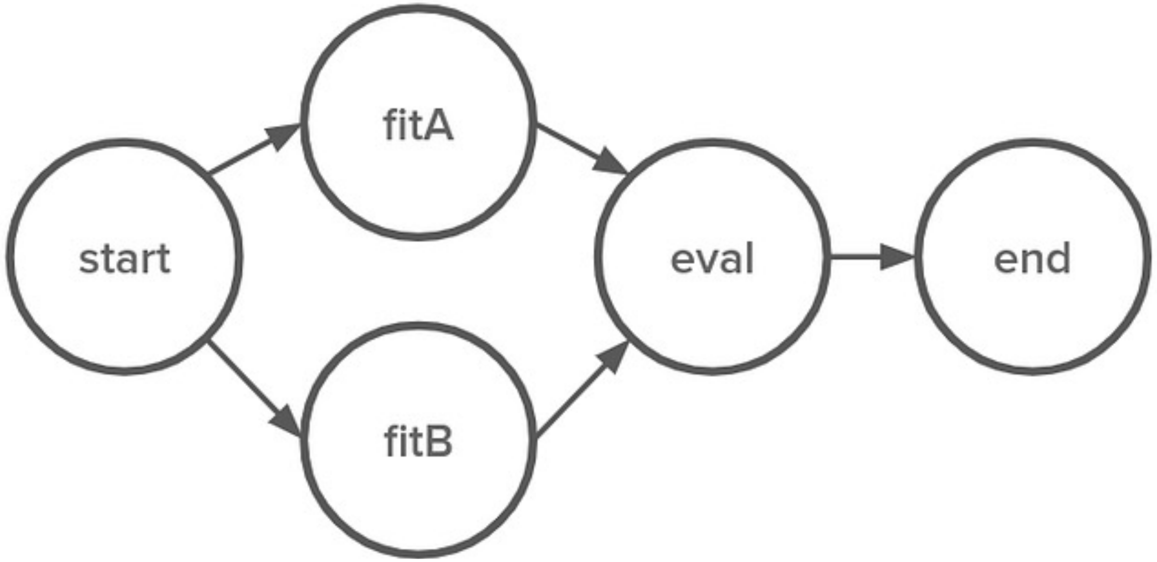
METAFLOW

Because everyone loves pipelines!

- Netflix produced "Data Science Project Manager"
- Versatile and easy orchestration tool:
  - In-project pipelines
  - End-to-end data workflows
  - Environment managment

9. Metaflow examples.

The DAG (Directed Acyclic Graph):



9. Metaflow examples.

Installation (CLI)

```
conda install metaflow
```

Execution (Python)

```
from metaflow import FlowSpec, step

class MyFlow(FlowSpec):

@step
def start(self):
self.data = load_your_data()
self.next(self.step_a, self.step_b)

@step
def fitA(self):
self.model= model_a()
self.next(self.eval)

@step
def fitB(self):
self.model=model_b()
self.next(self.eval)

@step
def eval(self, inputs):
self.best = pick_best(inputs)
self.next(self.end)

@step
def end(self):
logger.success("All done!")
```

9. Metaflow pros & cons.

METAFLOW

The Good…

- ✓ Simple, versatile & scalable
- ✓ Easy parallelisation of workflows
- ✓ Extensible “microframework”
- ✓ GUI option

The not so Good…

- ✗ Verbose
- ✗ Limited Cloud support for non-AWS

10. Make

CLI maker-easier.

10. Make Intro.

CLI maker-easier

- Automates the process of building programs through managing dependencies (Think data pipelines but for programs!)

- It also enables creation of repeatable CLI jobs (what I use it for)

- Specify complex tasks in the Makefile and run these with `make <command>`

10. Make examples.

Make installation

Mac: Included in Xcode

Windows: Via chocolate package manager

Linux: Included already!

Makefile example

```
CONDA_ENVIRONMENT_NAME = my_env
PYTHON_VERSION = 3.8

ACTIVATE = source $$(conda info --base)/etc/profile.d/conda.sh ; conda activate ; conda activate
DEACTIVATE = source $$(conda info --base)/etc/profile.d/conda.sh ; conda deactivate ; conda deactivate

# Environment Management
.PHONY: create-environment
create-environment: ## Create the conda environment & jupyter kernel
conda create --name $(CONDA_ENVIRONMENT_NAME) --channel conda-forge --yes python=$(PYTHON_VERSION)
$(ACTIVATE) $(CONDA_ENVIRONMENT_NAME) && \
    conda config --add channels conda-forge && \
    conda config --set channel_priority strict && \
    conda install --yes --file requirements-conda.txt && \
    conda update pip && \
    pip install -r requirements-pip.txt && \
    conda env export --name $(CONDA_ENVIRONMENT_NAME) -no-builds > environment.yaml && \
    ipython kernel install --user --name $(CONDA_ENVIRONMENT_NAME) && \
    nbstripout --install --attributes .gitattributes
    $(DEACTIVATE)

.PHONY: remove-environment
remove-environment: ## Remove the environment and any relevant files
    conda env remove --name $(CONDA_ENVIRONMENT_NAME)

.PHONY: run
run: ## Run the application
    $(ACTIVATE) $(CONDA_ENVIRONMENT_NAME) && \
    python -m main --env=.env
    $(DEACTIVATE)
```

10. Make pros & cons.

The Good…

- ✓ Simplifies complex CLI jobs
- ✓ Pre-installed on most VMs
- ✓ Easy model serving via REST API
- ✓ “Hackable” to an extent
- ✓ Made in 1976… Stood the test of time

The not so Good…

- ✗ Verbose & complex syntax
- ✗ Unforgiving (e.g. requires tabs not spaces)
- ✗ Difficult to install on windows machines

11. Bonus Round!

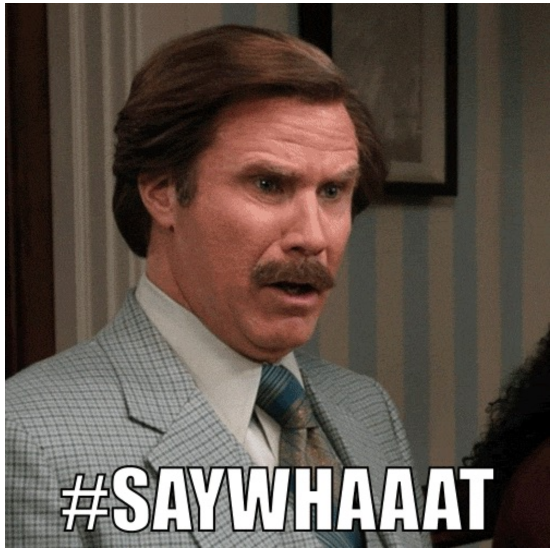
The honourable mentions:

- **polars**: High Performance Computing (HPC) in Python
- **dask**: Like Spark but on your machine
- **splink**: Data linking at scale
- **ydata-synthetic**: Synthetic data generator
- **python-dp**: Python data privacy
- **scikit-plot**: AI explanation & visualisation
- **lime**: Quicker explainable AI
- **pyLDAvis**: Interactive topic model visualisation
- **mlflow:** AI tracking & serving
- **nbstripout**: Remove Jupyter output cells from git
- **loguru**: Easy, colourful logging
- **Excalidraw**: Online collaborative whiteboarding
- **Github copilot ($)**: Predictive coding AI
- **DuckDB**: SQLite for analytics
- **Streamlit**: R Shiny! In Python
- **SparseSC**: Synthetic A/B testing

12. Bonus Round #2.

The not-so honourable mentions from Reddit:

- Domain knowledge
- Communication skills
- Sense of humour
- Economics
- Psychometrics
- My data engineer
- Pandas
- Excel
- Powerpoint
- The side scroller on my mouse



Those 10 Uberhacks again.

1. Github Awesomeness

https://github.com

2. D-Tale

https://github.com/man-group/**dtale**

3. Ydata-profiling

https://github.com/ydataai/**ydata-profiling**

4. Thefuzz

https://github.com/seatgeek/**thefuzz**

5. UK Open Data

https://www.ons.gov.uk/
https://opendata.scot/
https://www.nisra.gov.uk/

6. Yellowbrick

https://github.com/DistrictDataLabs/**yellowbrick**

7. Shap

https://github.com/slundberg/**shap**

8. Fairlearn

https://github.com/fairlearn/**fairlearn**

9. Metaflow

https://**metaflow**.org

10. Make

https://www.gnu.org/software/**make**

Thanks!

Slides, links, posts & mo' shizzle on my socials from Monday!

tom-s10

@TomEwingS10

Station10

www.station10.co.uk

?

Station10

www.station10.co.uk