MANGO SOLUTIONS

The Case for Testing in Data Science

Chris Musselle

Senior Data Scientist

cmusselle@mango-solutions.com

Introduction

- Work
  - Senior Data Science Consultant
  - Half Scientist / Half Engineer
  - Resident Python specialist

- Self
  - Life long learning
  - Iterate, refine, automate

Data Science?

Ultimate goal is to make valuable use of data for the business:

- By deriving insights from it to help drive decisions.
- By solving problems with it to provide new or improved services and products.

Different Needs at Different Times

EDA
Stats / ML
Visualisation

Programming
Prototyping

Software
Development
Delivery

Fast
Feedback

Automation

Reliability

Data Science Vs Development

Data Science	Development
Iterative/Agile	Iterative/Agile
Self Taught (tool)	Formally Trained (craft)
Mostly Greenfield projects, many smaller codebases and shorter projects	Mostly Brownfield projects, fewer but larger codebases and longer project timescales
Uncertainty around feasibility	Uncertainty around time and effort to build
Many data integrity issues	Fewer data integrity issues
Testing often ad hoc	Automated testing as Standard

What Do I Mean By Testing?

An automated set of checks that when run, prove a specific feature or function of your software works as you expected.

- Big topic, many types of testing.
- Focus mostly on “Unit Testing” today.

Why are We Disinclined to Test?

- Testing is time consuming and takes effort
- Feels like low value work, a chore
- Not as interesting as analytics
- Harder to do after the fact
- Pressure to be “done”

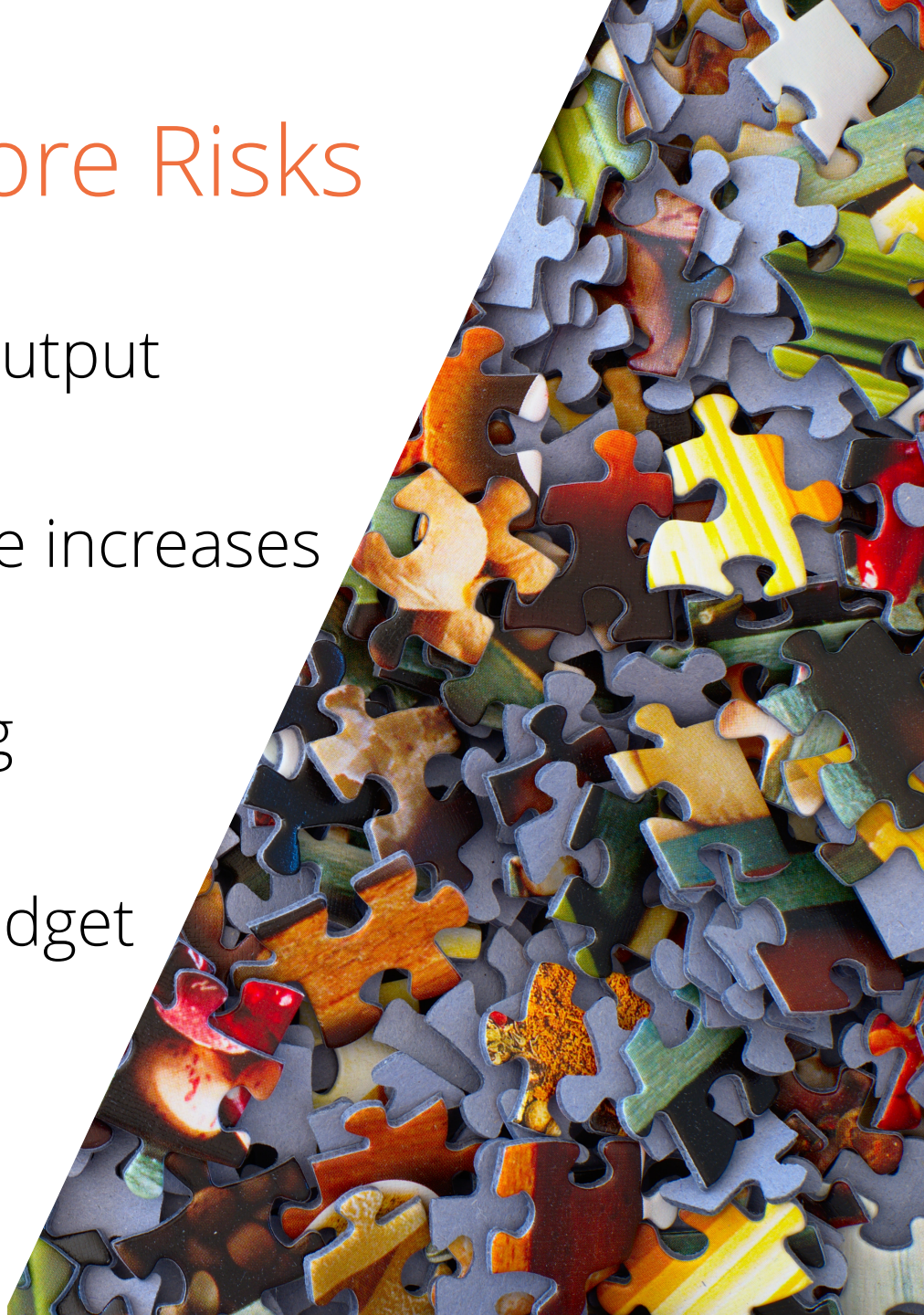
Realities of Software Development

Programming is hard and
people are pretty bad at it!

- All code has bugs
- How can you prove it works?
- Testing provides confidence and feedback

Fewer Tests = More Risks

- More defects, wrong output
- Slower progress as size increases
- Time lost to debugging
- Over time and over budget
- Unhappy customers



More Tests = More Benefits

- Validate assumptions
- Safety net for library upgrades
- Safety net for refactoring, adding new features
- Forced to think of design
- Documentation

When to Test?

Testing as You Develop is Easier

- You have as much knowledge as possible
- Forced to think about design
- Quick feedback of usability and utility

- If the code is hard to test, it probably has design issues.

TDD for Data Science?

- Test Driven Development (TDD) - Regarded as the “best” way in Software Development
  - Assumes you know what you are trying to build, and how it will be used.
  - Data Science is more disorganized!
  - More uncertainty, need time to prototype and experiment first

Great Opportunities for Tests

- After manually testing in the REPL
- After scripting a set of operations
- After finding a bug in the code (regression testing)
- After exploration in the Jupyter Notebook

What to Test?

Anything You Have Written

In the Analysis:

- Data Cleaning Pipelines
- Model Preprocessing Steps
- Summary Reports / Tables

- Don’t test what has already been tested.

Anything You Have Written

In constructing the Deliverable / Service:

- User facing APIs
- Data validation
- Utility functions
- Configuration Setup
- Logging

How to Test?

Libraries

A few to choose from, but *pytest* is a great choice.

- Minimal boilerplate
- Detailed failing test output
- Active community, many plugins

Given, When, Then

A good structure to follow is:

- **Given** a starting set of conditions
- **When** I perform these actions
- **Then** I expect these results

Pytest Example

```
# test_pipeline.py
import my_analysis_pkg as lib


def test_pipeline_output_exists():

    # Given
    config = lib.get_default_config()
    data = lib.load_data('path/to/test/data’)

    # When
    pipeline = lib.Pipeline(data, config)
    pipeline.run()

    # Then
    assert pipeline.output_report
```

Pytest Example

Run all tests from the command line with:

```
pytest path/to/test/directory
```

- Test *discovery*, *execution* and *reporting*.
- See the docs at: https://docs.pytest.org

Test Driven Data Analysis?

Test Driven Data Analysis (TDDA)

- Combining TDD approaches with Data Analysis
- Nick Radcliffe
  - Talk at PyData Berlin 2017 – YouTube
  - Tutorial at PyData London 2017 – YouTube
  - Blog at http://tdda.info

```
pip install tdda
```

TDDA: Core Features

- Provides ways to test your code ***and*** data

- Reference Tests:
  - Against "known to be correct" reference results.

- Constraint Tests:
  - Does the data look similar to what you have seen before?

Test Driven Data Analysis

- Supports pytest integration
- Has cli tool

```
tdda discover path/to/data.csv
```

- Run checks with

```
tdda verify path/to/data.csv
```

- Get started with

```
tdda examples
```

Summary

Testing Guidelines:

- Test alongside development of the code
- Focus on your code over libraries
- Need to check our code (pytest) and our data (tdda)

Questions?

Chris Musselle
Senior Data Scientist
cmusselle@mango-solutions.com

Resources

- Clean Code - Robert Martin
- Python 3 Object Oriented Programming - Dusty Phillips
- Python Testing with pytest - Brian Okken