Balancing Speed and Accuracy in Model Development

By Ivan Popov

Data Scientist at Abound & Render, London, UK

- 3 years of Machine Learning experience in FinTech and Computer Vision
- 4 years of experience as a Data Analyst
- DevOps and Data Engineering Expert
- Co-founded a business and grew it from 0 to 80,000 registered users with 1.5M monthly site traffic

Today’s talk

- The Essence of Speed and Accuracy Balance
- Identifying Your Model’s Needs
- Optimisation Strategies

The Essence of Balance: Speed vs Accuracy

Factors impacting accuracy and speed

- Complexity of the model architecture
- Amount and quality of input data
- Hardware

The Business Impact of Speed and Accuracy

Context is the key

Real-World Examples

- Loan industry: speed is prioritised
- E-commerce: equal balance between the speed and accuracy
- Medical diagnostics: accuracy is prioritised

Balancing Act: Speed, Accuracy, and Cost

Is money always the answer?

Strategic Importance of the Balance

Mastering the balance between model speed and accuracy can serve as a significant competitive advantage

Identifying Your Model's Needs

How to Understand Business Objectives

- What is the purpose of the model?
- Is it for internal use or customer-facing?
- What are the desired outcomes of the model?
- What are the KPIs?
- Who are the end-users of the model?

Scenarios for ML-models

- Customer-facing applications: speed is often more critical than accuracy
- Internal analytics: accuracy is more important than speed

Optimisation Strategies

Training data quality and quantity

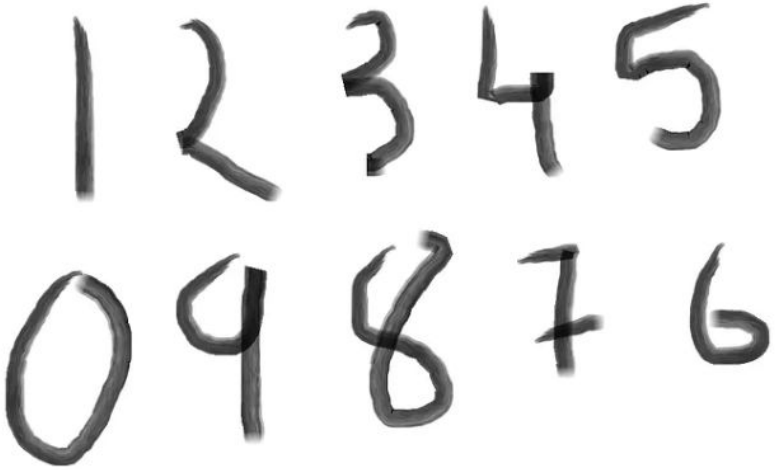
Get yourself a dataset with good quality data and good labels

What is a good dataset?

- Includes samples from multiple writers and different writing styles
- Has a balanced distribution of digits
- All images have a clear label associated with them

What is a bad dataset?

- Only includes handwritten digits from a single writer
- Misses certain digits or labels for the images.



Data Pre-processing

- Data cleaning
- Data normalisation
- Feature generation

How to find inefficiencies in data pre-processing?

`time()`

cPython

yappi

yappi

```
import yappi

yappi.set_clock_type("wall")
yappi.start()
foo()
yappi.get_func_stats().print_all()
yappi.get_thread_stats().print_all()
'''
Clock type: WALL
Ordered by: totaltime, desc
name                                  ncall  tsub      ttot
test.py:6 my_func                     1      0.000007  4.004159
'''
```

Most Common Inefficiencies

1. Extensive use of `.apply()` in pandas

   If you use `.apply()` to a single column, try to find simpler execution, e.g., `df['radius']*2`

   If you use `.apply()` to multiple columns, avoid the `.apply(,axis=1)` format. Write a standalone function, and then use it on the `.values` of the Pandas Series

2. Performing calculations more times than needed

   If you have metadata, perform the calculation once per group

Use **numpy** instead of pandas where it is possible

Feature Selection

Select the most important features and remove irrelevant ones to simplify your model and reduce the risk of overfitting

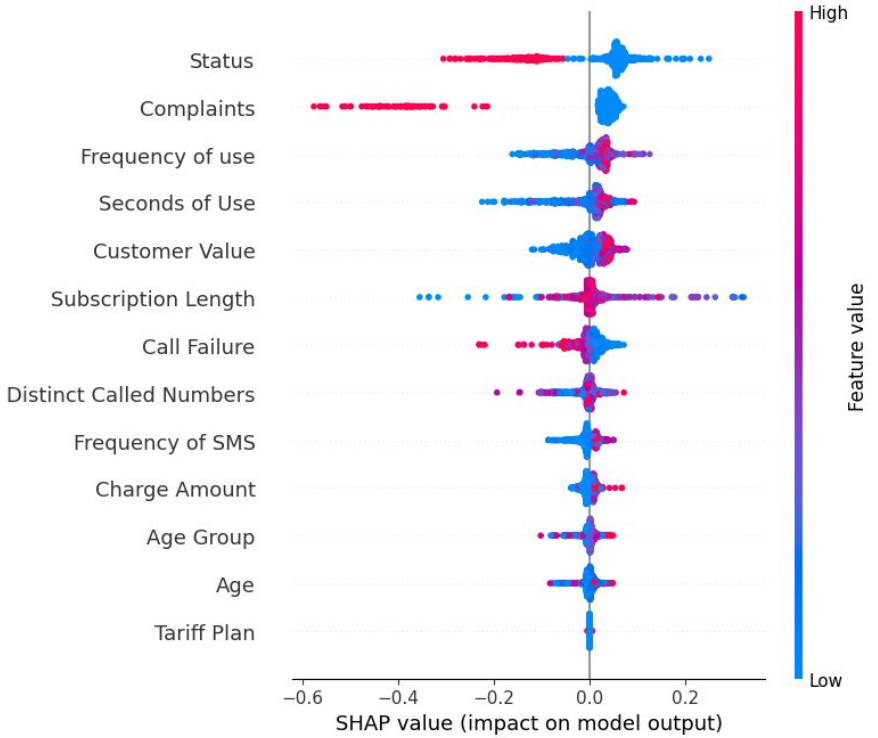
SHAP Values for Feature Selection

Can be used to interpret any machine learning model:

- Linear regression
- Decision trees
- Random forests
- Gradient boosting models
- Neural networks

SHAP Values for Feature Selection

Useful when dealing with high-dimensional, complex datasets



Model selection

Try using simpler models like

XGBoost and LightGBM

XGBoost

- Widespread usage: employed since 2014.

- Flexibility

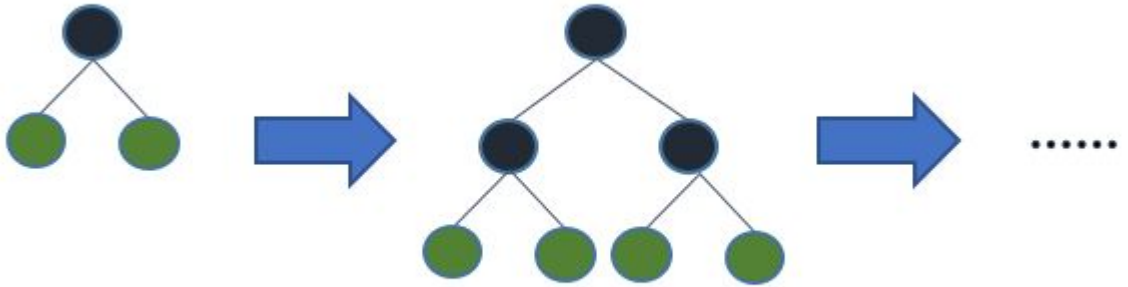
- DMatrix

- Regularisation

- Depth-first order

- Computational considerations: may be slower with large datasets

XGBoost

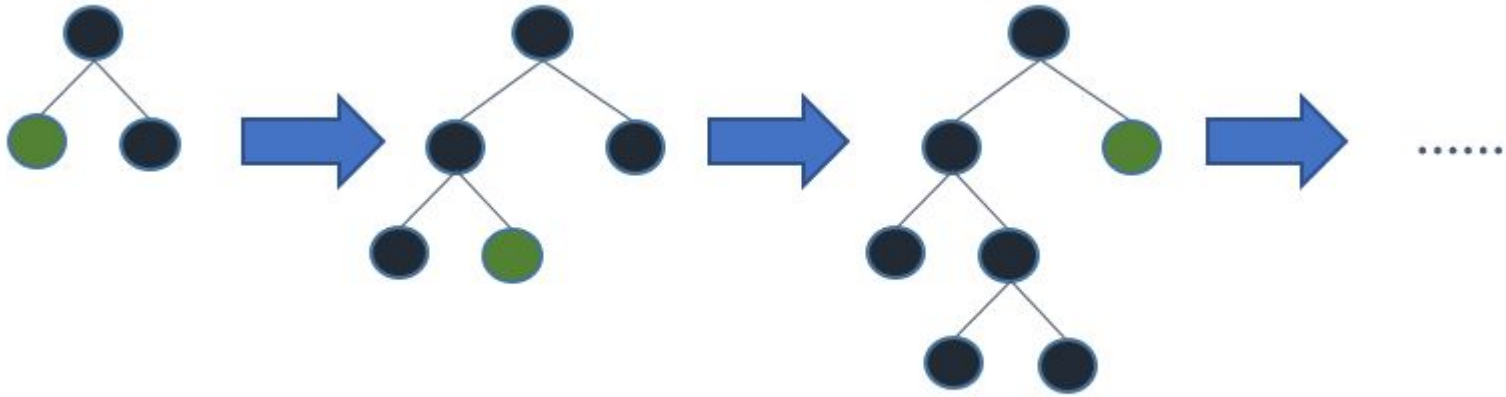


Level-wise tree growth

LightGBM

- Lightning-fast performance
- Efficient with large datasets
- Categorical data handling
- Regularisation
- Best-first order
- Computational considerations: memory intensive

LightGBM



Leaf-wise tree growth

How to Choose the Best Option

Experiment and decide

A Quick Recap

- Balancing speed and accuracy often depends on the context or the field of application.
- To identify your model's needs, align its purpose with business goals and tailor the model to meet user expectations.
- Make sure you acquire a robust dataset with quality data and accurate labels.
- Experiment with simpler models like XGBoost and LightGBM.
- Use profilers such as cProfile and yappi to find bottlenecks in your code.
- The data pre-processing step is often the most frequent place for the bottlenecks.

Thank you for your time!

References

- LightGBM documentation and installation guides - https://lightgbm.readthedocs.io/en/stable/
- XGBoost documentation and installation guides - https://xgboost.readthedocs.io/en/stable/
- cProfile documentation - https://docs.python.org/3/library/profile.html
- yappi documentation and installation guides - https://pypi.org/project/yappi/
- shap documentation and installation guides - https://shap.readthedocs.io/en/latest/