Your best bet

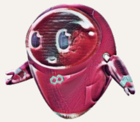
Effortless MLOps with dbt Python models

Dorian Van den Heede - PyData Eindhoven - 30/11/2023

:~$ whoami

datarॐts

11 monthly listeners



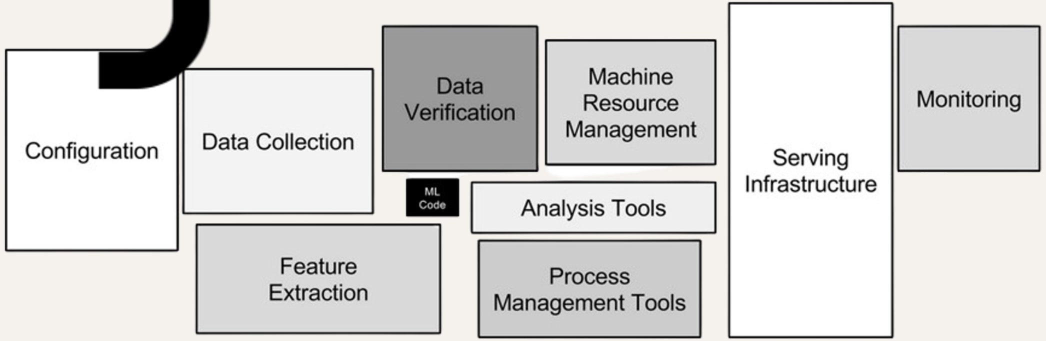
Beatroots AI

@devdnhee

Table of contents

MLOps complexity

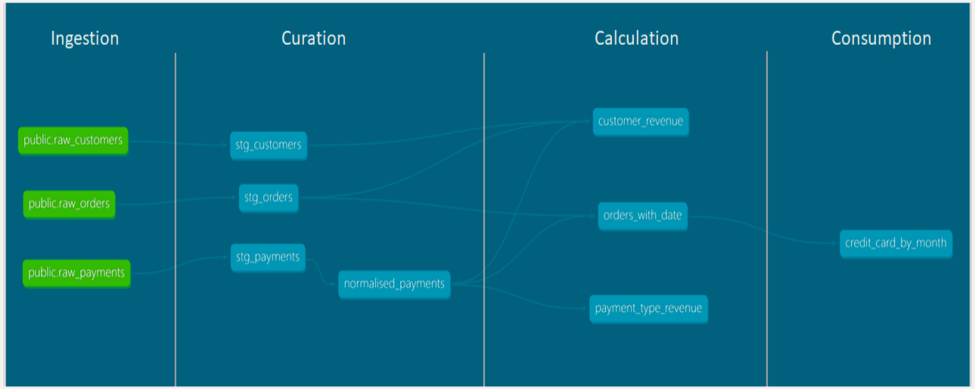
- Data preparation + maintenance
- Performance tracking
- Continuous experimentation
- Model maintenance
- Avoid training / serving skews
- Reproducibility
- Infrastructure
- Team collaboration
- ...



Hidden technical debt in ML systems, 2015, D Sculley

What is dbt? SQL first

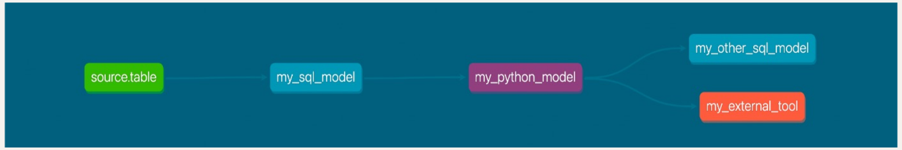
- data transformation workflow
- Orchestration + optimisation in DAGs
- Connects to data platforms
- Configure tests + documentation
- Enables DRY SQL code
- DE, DS, DA, MLE



```
select * from {{ref('really_big_table')}}

{% if incremental and target.schema == 'prod' %}
  where timestamp >= (select max(timestamp) from {{this}})
{% else %}
  where timestamp >= dateadd(day, -3, current_date)
{% endif %}
```

What is dbt? SQL first, Python second

- data transformation workflow
- Orchestration + optimisation in DAGs
- Connects to data platforms
- Configure tests + documentation
- Enables DRY SQL code
- DE, DS, DA, MLE



```python
import ...

def model(dbt, session):
    my_sql_model_df = dbt.ref("my_sql_model")
    final_df = ...
    return final_df
```

models/python_model.py

02

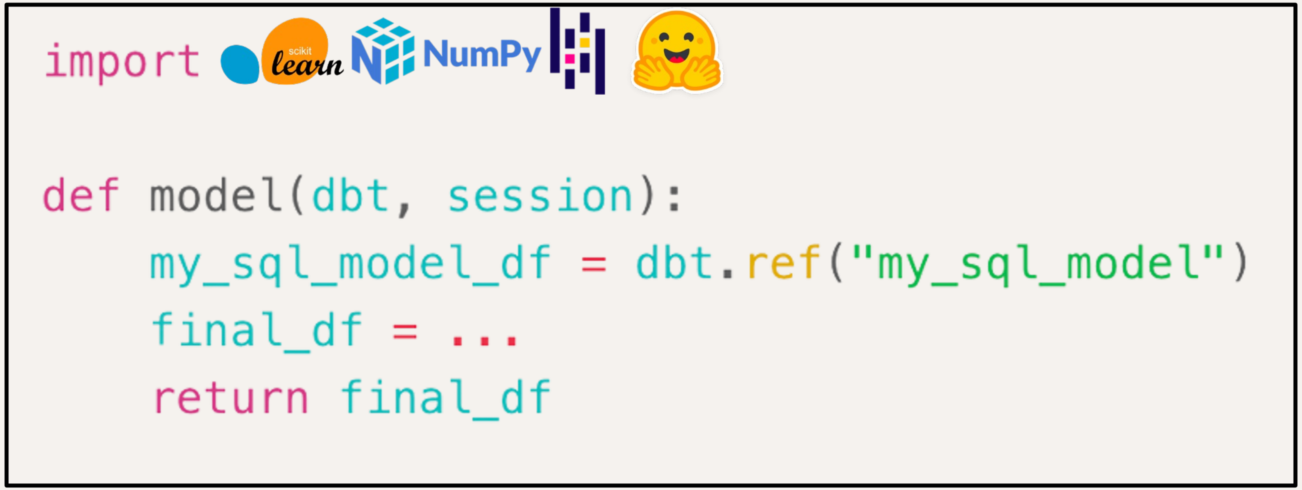
🏁 Quickstart

HOWTO: python in dbt

```python
import ...

def model(dbt, session):
    my_sql_model_df = dbt.ref("my_sql_model")
    final_df = ...
    return final_df
```

models/python_model.py

HOWTO: python in dbt



```python
import

def model(dbt, session):
    my_sql_model_df = dbt.ref("my_sql_model")
    final_df = ...
    return final_df
```

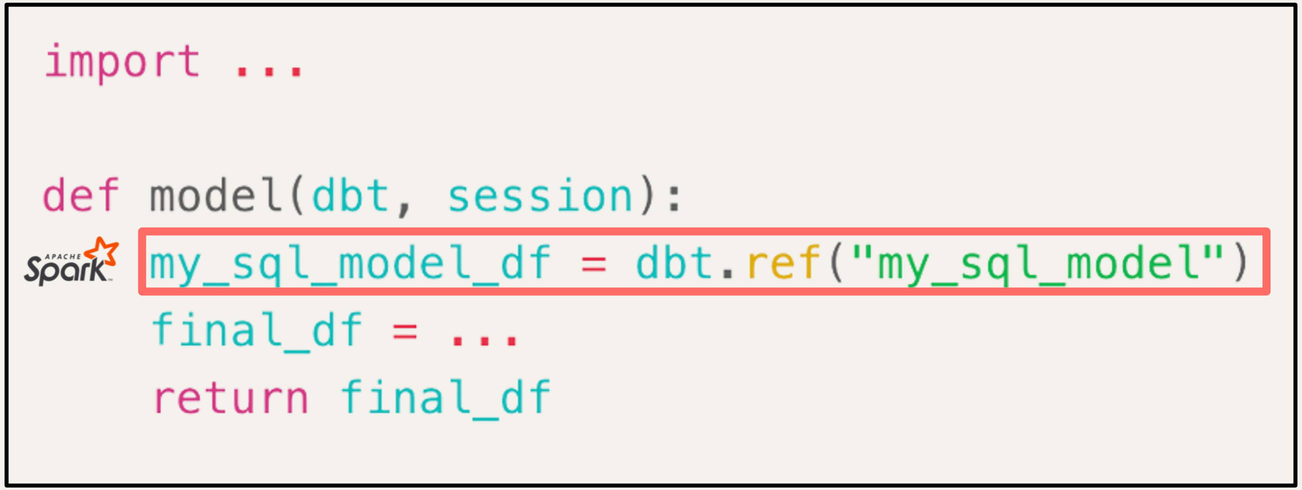
models/python_model.py

HOWTO: python in dbt

```python
import ...

def model(dbt, session):
    my_sql_model_df = dbt.ref("my_sql_model")
    final_df = ...
    return final_df
```

models/python_model.py

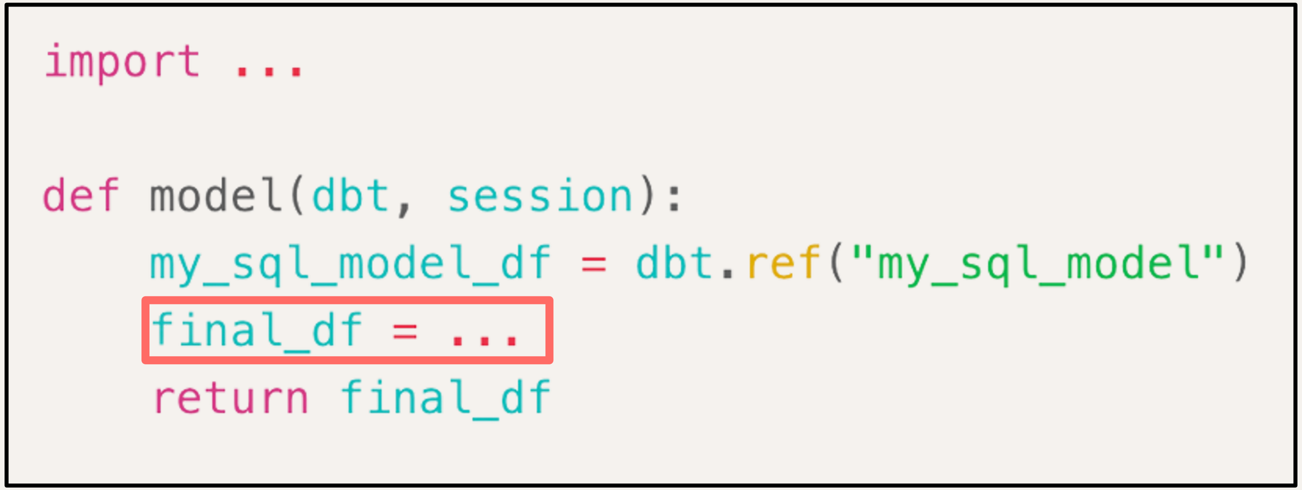
HOWTO: python in dbt



```
import ...

def model(dbt, session):
    my_sql_model_df = dbt.ref("my_sql_model")
    final_df = ...
    return final_df
```

models/python_model.py

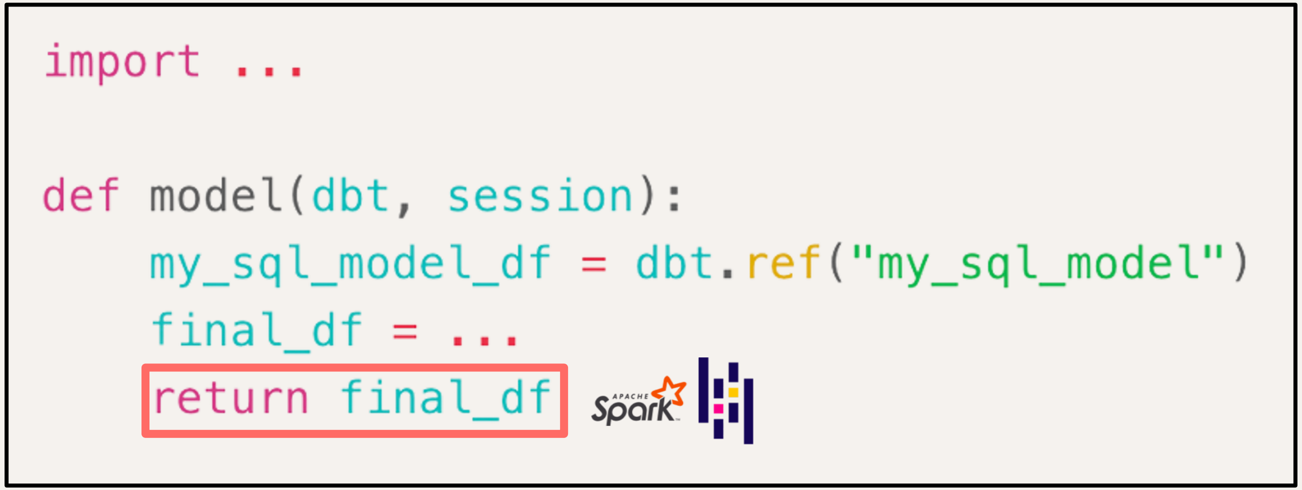
HOWTO: python in dbt



```python
import ...

def model(dbt, session):
    my_sql_model_df = dbt.ref("my_sql_model")
    final_df = ...
    return final_df
```

models/python_model.py

HOWTO: python in dbt



```python
import ...

def model(dbt, session):
    my_sql_model_df = dbt.ref("my_sql_model")
    final_df = ...
    return final_df
```

models/python_model.py

03

⚽ Toy DWH

Can we beat the bookies?

🏟️ European soccer database

- Open source football database
- 11 countries
- 25k match results, 2008 -> 2016
- Weekly fifa stats players + teams
- Betting odds from 10 bookies
    - Implied probability: 1 / odds
- SQLite

Win → +1.17
Loss → -1

E[💶] = 44% x 1.17 + 56% x -1
= -0.045

Result	Home win	Draw	Away win
Odd	2.17	3.60	3.30
Implied probability	46% → 44%	28% → 27%	30% → 29%

Σ = 104%

🎲 Use case: beating bookies with ML

Predict the **winner** of the match given:

- Team stats (Fifa)
- Player stats from the previous game (Fifa)
- Team form (last 5 matches)
- Odds bookies

As **probabilities**, Then **transform to odds.**

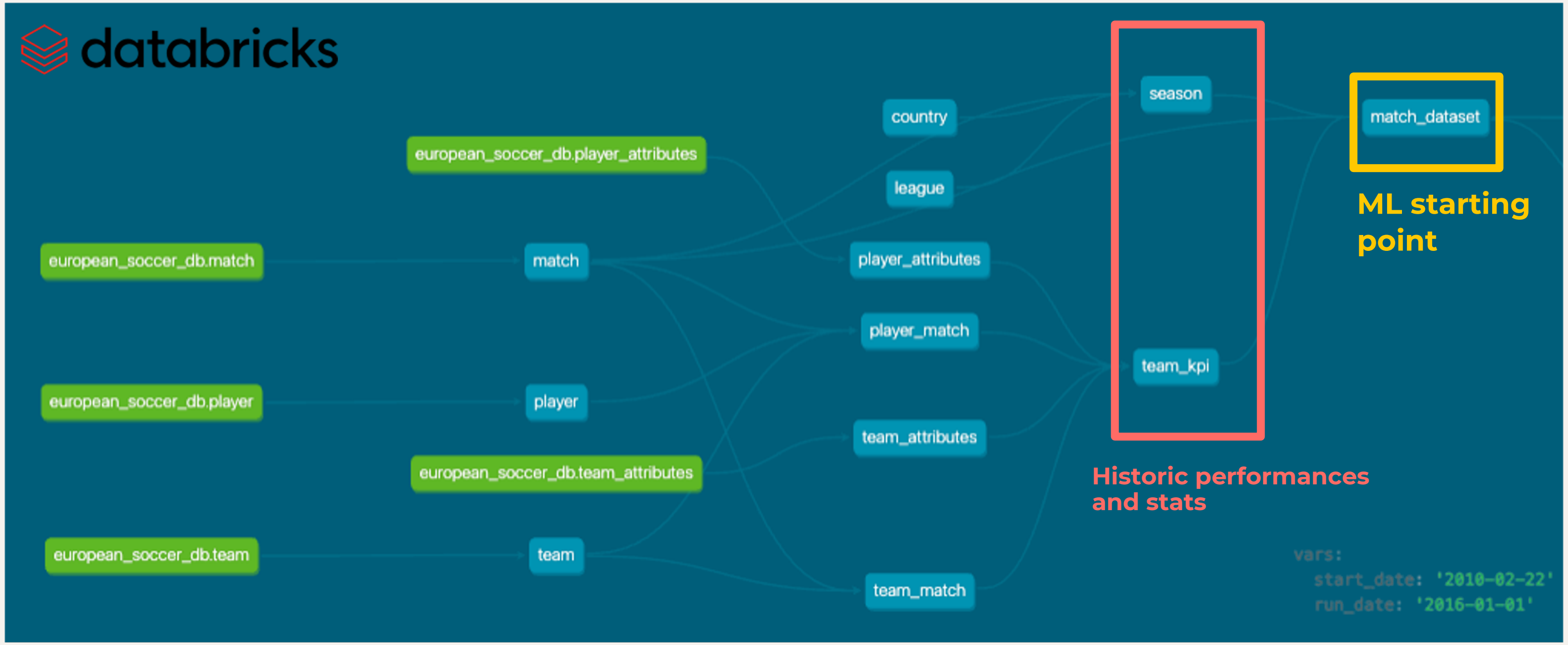
We place our bet on the **bookie + winner** where our model prospects **lower odds**

Result	Home win	Draw	Away win
Odd	2.17	3.60	3.30
Implied probability	46% → 44%	28% → 27%	30% → 29%
Model probability	50%	25%	25%
Model Odds	2	4	4

Win → +1.17
Loss → -1

E[💶] = 50% x 1.17 + 50% x -1
= +0.085

🏗️ Architecture: Feature engineering all SQL



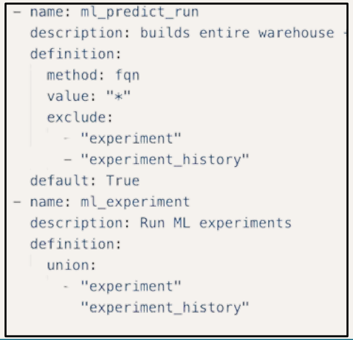
```
vars:
  start_date: '2010-02-22'
  run_date: '2016-01-01'
```

```
dbt build -selector gold
```

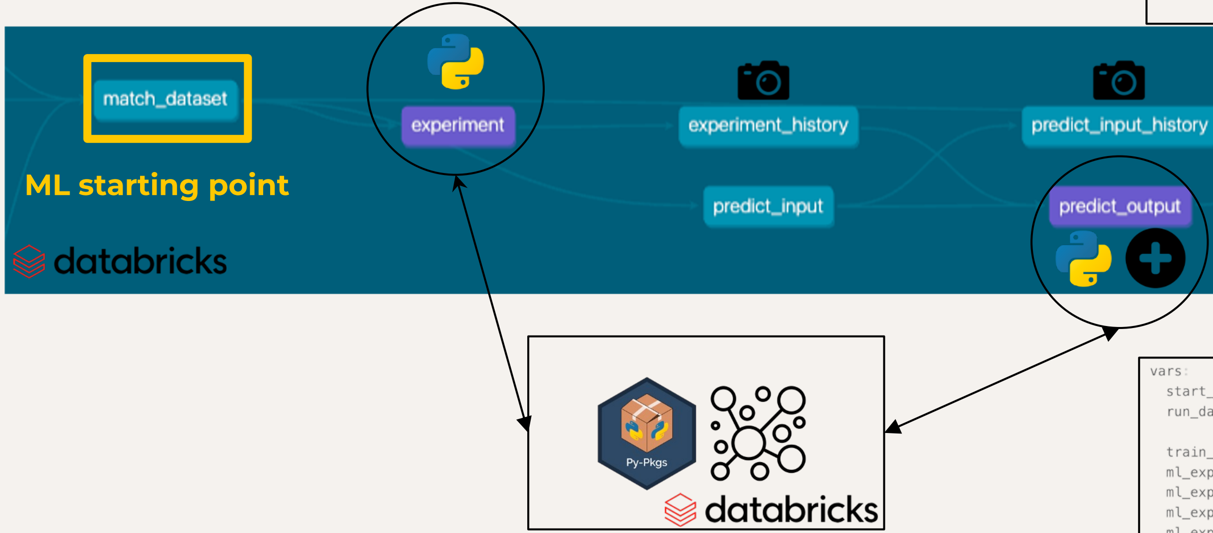
```
selectors:
    name: gold
    description: Run entire warehouse without ML, so also preprocessing
    definition: "+match_dataset"
  - name: ml_predict_run
```

🚰 Architecture: ML pipeline

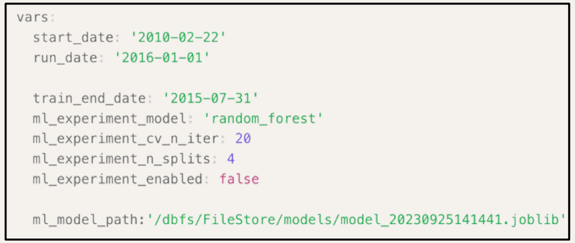
```
dbt build -selector ml_experiment
dbt build -selector ml_predict_run
```



```
- name: ml_predict_run
  description: builds entire warehouse
  definition:
    method: fqn
    value: "*"
    exclude:
      - "experiment"
      - "experiment_history"
  default: True
- name: ml_experiment
  description: Run ML experiments
  definition:
    union:
      - "experiment"
        "experiment_history"
```



Compute cluster



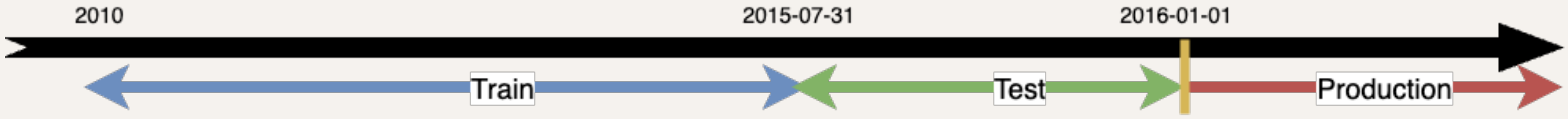
```
vars:
  start_date: '2010-02-22'
  run_date: '2016-01-01'

  train_end_date: '2015-07-31'
  ml_experiment_model: 'random_forest'
  ml_experiment_cv_n_iter: 20
  ml_experiment_n_splits: 4
  ml_experiment_enabled: false

  ml_model_path:'/dbfs/FileStore/models/model_20230925141441.joblib'
```

04

🤞Demo

https://github.com/datarootsio/your-best-bet

Simulation start

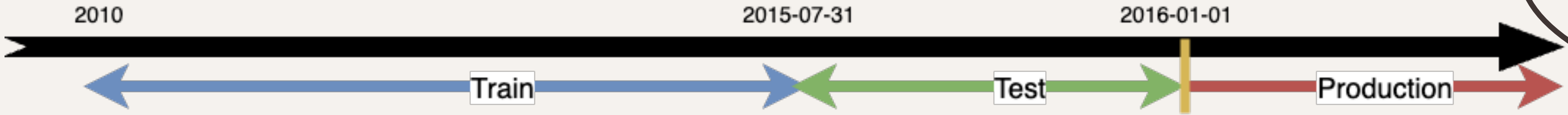


```
dbt build -selector gold -vars '{"run_date": "2016-01-01"}'
```

#	day	home	away	b365h	b365d	b365a	home_form_5m	away_from_5m	winner
1	2015-12-29	Leicester City	Manchester City	4.33	4.00	1.85	10	9	draw
2	2015-12-30	Sunderland	Liverpool	5.50	3.90	1.70	3	7	away
3	2016-01-02	Arsenal	Newcastle United	1.29	6.50	11.00	12	7	NULL
4	2016-01-02	Sunderland	Aston Villa	2.50	3.25	3.20	0	3	NULL
5	2016-01-02	Norwich City	Southampton	3.20	3.40	2.40	7	4	NULL
6	2016-01-02	Watford	Manchester City	5.00	4.00	1.75	10	7	NULL

match_dataset

Simulation start

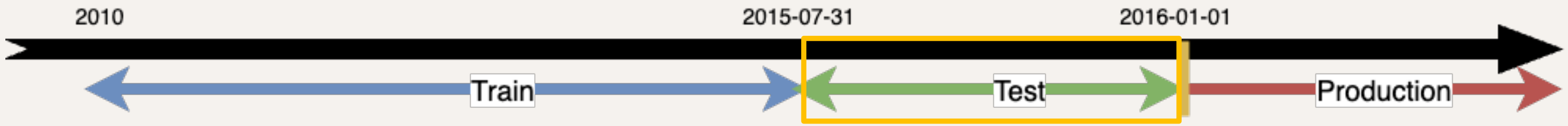


```
dbt build --selector ml_experiment --vars '{"ml_experiment_model": "xgboost”, “train_end_date”: “2015-07-31”}’
dbt build --selector ml_experiment --vars '{"ml_experiment_model": ”random_forest”, “train_end_date”: “2015-07-31”}’
dbt build --selector ml_experiment --vars '{"ml_experiment_model": ”logistic_regression”, “train_end_date”: “2015-07-31”}'
```

train_end_date	experiment_id	Rank_logloss	params	mean_test_auc	best_estimator_path
2015-07-31	3eb087ab	1	{"model__estimator__C": 4.778811494000367, "selector__k": 29}	0,650532	model_20231125101357.joblib
2015-07-31	761450c7	2	{"model__estimator__C": 211.57716003613038, "selector__k": 29}	0,650162	model_20231125101357.joblib
2015-07-31	1febe59f	3	{"model__estimator__C": 0.34893756446012314, "selector__k": 51}	0,64808	model_20231125101357.joblib
2015-07-31	633cab37	4	{"model__estimator__C": 3.573772971230787, "selector__k": 49}	0,648862	model_20231125101357.joblib
2015-07-31	f86dec04	5	{"model__estimator__C": 422.3597270604446, "selector__k": 41}	0,648024	model_20231125101357.joblib
2015-07-31	e819bff4	6	{"model__estimator__C": 30.651121446436083, "selector__k": 54}	0,646431	model_20231125101357.joblib
2015-07-31	4c8785da	7	{"model__estimator__C": 32.05437819428035, "selector__k": 67}	0,644743	model_20231125101357.joblib
2015-07-31	86efed9e	8	{"model__estimator__C": 72.55488650213594, "selector__k": 81}	0,643301	model_20231125101357.joblib
2015-07-31	01740a74	9	{"model__estimator__C": 0.5569818837953282, "selector__k": 89}	0,641938	model_20231125101357.joblib
2015-07-31	a9305230	10	{"model__estimator__C": 363.02551035572196, "selector__k": 90}	0,641471	model_20231125101357.jo

experiment

Simulation start



```
dbt build -selector ml_predict_run -vars '{"run_date": "2016-01-01"}'
```

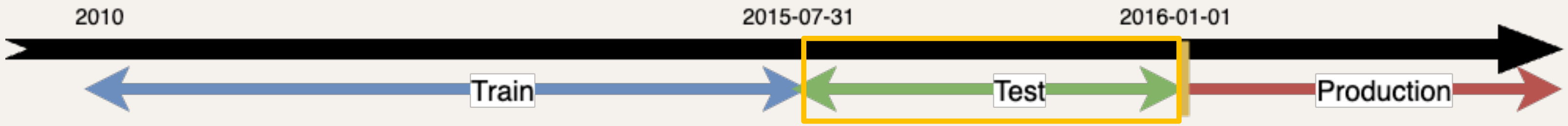
prediction_id	match_date	home_team	away_team	bet	xOdd	bookie	odd_bookie	xProfit
ea5e3797	2015-12-29	Leicester City	Manchester City	draw	2,202137	odds_psd	4,1	0.88
002e9918	2015-12-30	Sunderland	Liverpool	draw	1,794264	odds_psd	3,91	1.15
03d506f6	2016-01-02	Arsenal	Newcastle United	away	5,067033	odds_psa	12,81	1.42
d5f058bc	2016-01-02	Leicester City	Bournemouth	away	3,978723	odds_b365a	4,33	0.05
fdd429bc	2016-01-02	Manchester United	Swansea City	away	4,230447	odds_psa	7,66	0.98
b4621cb9	2016-01-02	Norwich City	Southampton	draw	2,567488	odds_psd	3,49	0.33

prediction_output

prediction_id	match_date	home_team	away_team	bet	xOdd	bookie	odd_bookie	winner	bet_correct	profit
ea5e3797	2015-12-29	Leicester City	Manchester City	draw	2,18033	odds_psd	4,1	draw	TRUE	3,1
002e9918	2015-12-30	Sunderland	Liverpool	draw	1,82006	odds_psd	3,91	away	FALSE	-1

evaluation

Simulation start

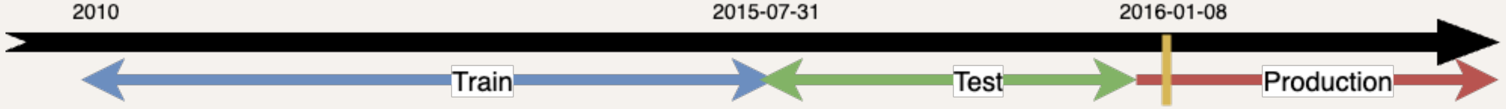


```
dbt build -selector ml_predict_run -vars '{"run_date": "2016-01-01"}'
```

model_path	model	total_profit	avg_profit_per_bet	nbr_matches_predicted
model_20231125101357 .joblib	LogisticRegression	**24,94**	0,015033	1659
model_20231124155618 .joblib	LogisticRegression	8,04	0,004846	1659
model_20231124144252 .joblib	XGBClassifier	0,96	0,000579	1659
model_20231124134416 .joblib	RandomForestClassifier	-3,36	-0,00203	1659

compare_model_profit.sql

Simulation: 1 week

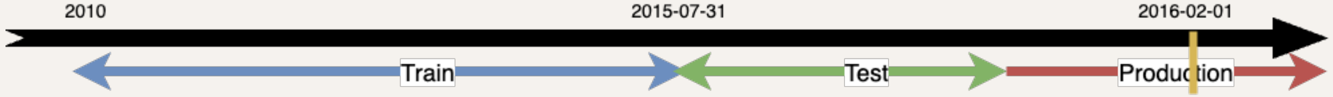


```
dbt build '{"run_date": "2016-01-08"}'
```

match_date	model	day_profit	avg_profit	nbr_predicted	nbr_bets	total_model_profit	total_nbr_predictions
2016-01-06	model_20231125101357.joblib	-1,5	-0,08333	18	18	**38,72**	1712
2016-01-05	model_20231125101357.joblib	2,6	2,6	1	1	40,22	1694
2016-01-04	model_20231125101357.joblib	0	0	1	0	37,62	1693
2016-01-03	model_20231125101357.joblib	9,72	1,08	9	9	37,62	1692
2016-01-02	model_20231125101357.joblib	2,96	0,123333	24	23	27,9	1683
2015-12-31	model_20231125101357.joblib	-1	-1	1	1	24,94	1659
2015-12-30	model_20231125101357.joblib	-10,78	-0,77	14	14	25,94	1658
2015-12-29	model_20231125101357.joblib	2,1	1,05	2	2	36,72	1644
2015-12-28	model_20231125101357.joblib	-3,67	-0,45875	8	8	34,62	1642
2015-12-27	model_20231125101357.joblib	0,59	0,1475	4	4	38,29	1634

model_profit_per_day.sql

Simulation: 1 month



```
dbt build '{"run_date": "2016-02-01"}'
```

match_date	model	day_profit	avg_profit	nbr_predicted	nbr_bets	total_profit	total_nbr_predictions
2016-01-31	model_20231125101357.joblib	8,86	0,316429	28	28	**27,37**	1990
2016-01-30	model_20231125101357.joblib	-10,04	-0,31375	32	32	18,51	1962
2016-01-29	model_20231125101357.joblib	-1,27	-0,254	5	5	28,55	1930
2016-01-28	model_20231125101357.joblib	9	4,5	2	2	29,82	1925
2016-01-27	model_20231125101357.joblib	-5	-1	5	5	20,82	1923
2016-01-26	model_20231125101357.joblib	1,71	0,855	2	2	25,82	1918
2016-01-25	model_20231125101357.joblib	-2	-1	2	2	24,11	1916
2016-01-24	model_20231125101357.joblib	5,55	0,185	30	30	26,11	1914
2016-01-23	model_20231125101357.joblib	8,78	0,209048	42	41	20,56	1884
2016-01-22	model_20231125101357.joblib	-3,39	-0,48429	7	7	11,78	1842

model_profit_per_day.sql

05

🏭 Design Patterns

📸 dbt snapshot

= detects row changes in a table and implements type-2 SCD to create full history

Will automatically stack records with fresh reruns of your DAG

- Applied on model input ≣ Feature store ⇒ **Reproducibility**
- Applied on model output ≣ Prediction store ⇒ **Auditability**
- Applied on experiments ≣ Model Registry ⇒ **Reproducibility**

```
{{
    config(
      ...
     unique_key='id',
     strategy='timestamp',
     updated_at='last_match_date',
    )
}}
```

team_id	form	last_match_date
1	12	2023-09-24

team_id	form	last_match_date
1	9	2023-09-27

team_id	form	last_match_date	dbt_valid_from	dbt_valid_to
1	12	2023-09-24	2023-09-24	2023-09-27
1	9	2023-09-27	2023-09-27	null

🔑 Surrogate keys

💡Generate surrogate keys with dbt_utils

```
select {{ dbt_utils.generate_surrogate_key(match_columns) }} as surrogate_key
```

id	home_team_form_5m	away_team_form_5m	prediction_id
1609	8	10	b47de063628941d5bb2e059b24d813c9
1609	8	9	ee920b456ad5879ab10428c3ec8566a6
1610	7	7	04a275ea6d15a9fef0cc7cf10a72feb2
1610	10	10	bdeeb6d774cf167344c155ed394e48a9

🐍 Python modules + functions

Current Limitations

- No macros / cross-model UDFs supported
- No Jinja support
- No local modules

```
from riskrover import pipeline
```

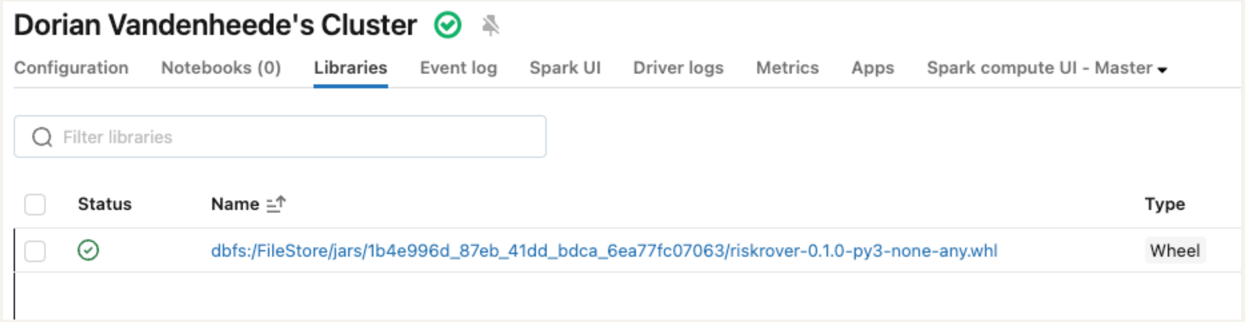
models/experiment.py

```
from riskrover.model import RiskRover
from riskrover.pipeline import preprocess_match_dataset
```

models/predict_input.py

Workaround

- Package yourself
- Install on cluster

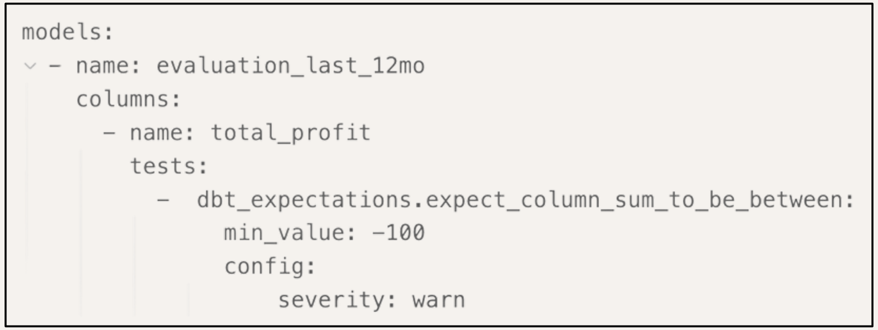


Dorian Vandenheede's Cluster

Status	Name	Type
✓	dbfs:/FileStore/jars/1b4e996d_87eb_41dd_bdca_6ea77fc07063/riskrover-0.1.0-py3-none-any.whl	Wheel

🧪 Testing + Documentation

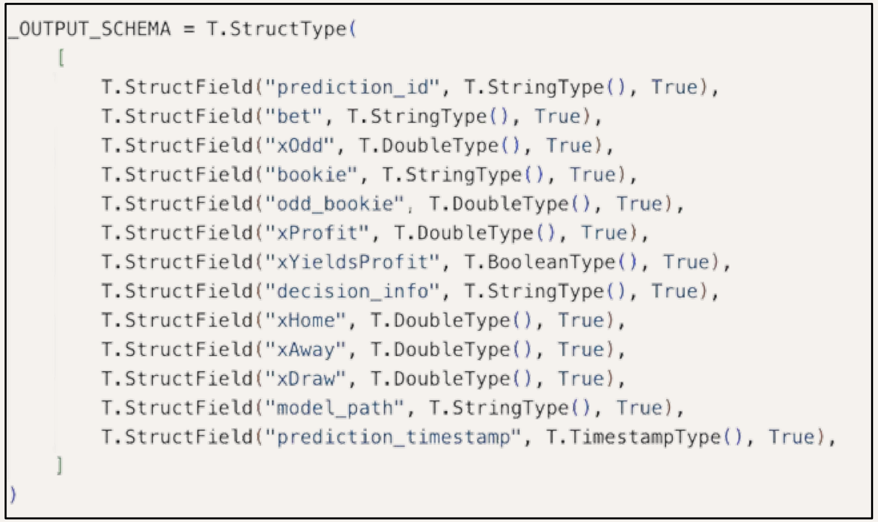
- Python package
  - Functional unit tests
  - Documentation of functions
- dbt
  - Test data, columns, tables, ...
  - Documentation of data, columns, tables
  - Alerts for model degradation



```yaml
models:
  - name: evaluation_last_12mo
    columns:
      - name: total_profit
        tests:
          -  dbt_expectations.expect_column_sum_to_be_between:
               min_value: -100
               config:
                   severity: warn
```

Other tips

- Prepare for empty outputs -> define schema

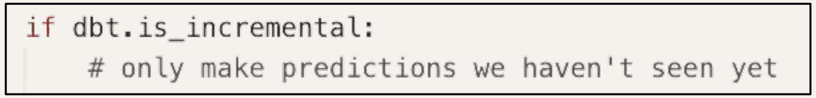


```
_OUTPUT_SCHEMA = T.StructType(
    [
        T.StructField("prediction_id", T.StringType(), True),
        T.StructField("bet", T.StringType(), True),
        T.StructField("xOdd", T.DoubleType(), True),
        T.StructField("bookie", T.StringType(), True),
        T.StructField("odd_bookie", T.DoubleType(), True),
        T.StructField("xProfit", T.DoubleType(), True),
        T.StructField("xYieldsProfit", T.BooleanType(), True),
        T.StructField("decision_info", T.StringType(), True),
        T.StructField("xHome", T.DoubleType(), True),
        T.StructField("xAway", T.DoubleType(), True),
        T.StructField("xDraw", T.DoubleType(), True),
        T.StructField("model_path", T.StringType(), True),
        T.StructField("prediction_timestamp", T.TimestampType(), True),
    ]
)
```

Models/predict_output.py

Other tips

- Prepare for empty outputs -> define schema
- Incremental python models for predictions

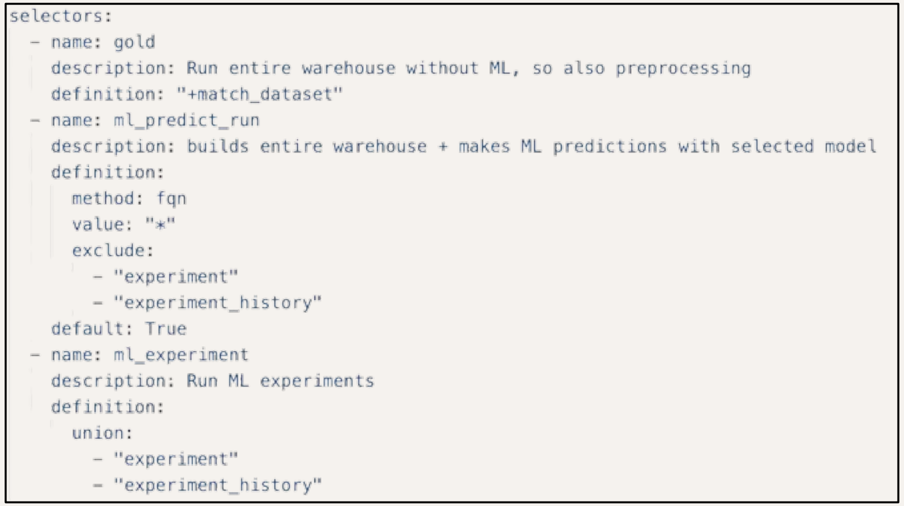


```python
if dbt.is_incremental:
    # only make predictions we haven't seen yet
```

models/predict_output.py

Other tips

- Prepare for empty outputs -> define schema
- Incremental python models to avoid redundant work: batch prediction
- Complex selection criteria with selectors



```yaml
selectors:
  - name: gold
    description: Run entire warehouse without ML, so also preprocessing
    definition: "+match_dataset"
  - name: ml_predict_run
    description: builds entire warehouse + makes ML predictions with selected model
    definition:
      method: fqn
      value: "*"
      exclude:
        - "experiment"
        - "experiment_history"
    default: True
  - name: ml_experiment
    description: Run ML experiments
    definition:
      union:
        - "experiment"
        - "experiment_history"
```

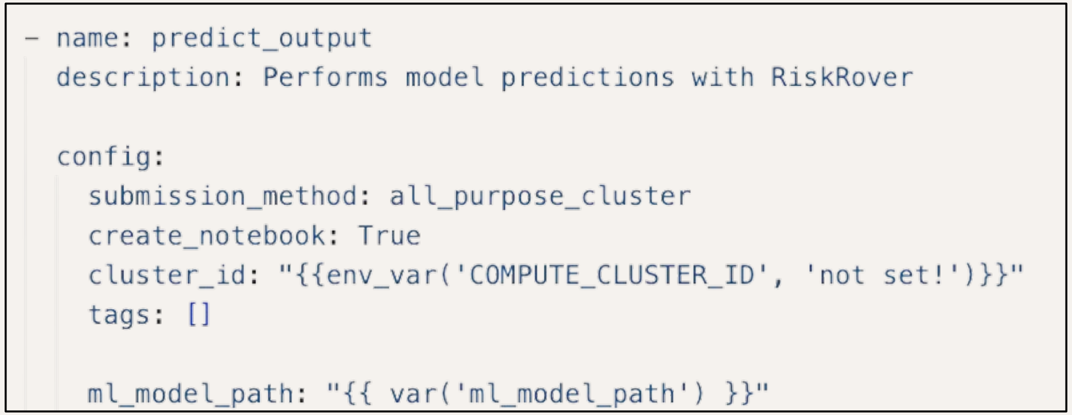
selectors.yaml

Other tips

- Prepare for empty outputs -> define schema
- Incremental python models to avoid redundant work: batch prediction
- Complex selection criteria with selectors
- Spark / Spark on Pandas API / Pandas

Other tips

- Prepare for empty outputs -> define schema
- Incremental python models to avoid redundant work: batch prediction
- Complex selection criteria with selectors
- Spark / Spark on Pandas API / Pandas
- Use of vars / env vars



```yaml
- name: predict_output
  description: Performs model predictions with RiskRover

  config:
    submission_method: all_purpose_cluster
    create_notebook: True
    cluster_id: "{{env_var('COMPUTE_CLUSTER_ID', 'not set!')}}"
    tags: []

    ml_model_path: "{{ var('ml_model_path') }}"
```

models.yaml

Other tips

- Prepare for empty outputs -> define schema
- Incremental python models to avoid redundant work: batch prediction
- Complex selection criteria with selectors
- Spark / Spark on Pandas API / Pandas
- Use of vars / env vars
- Don’t perform (too many api) calls
- …

06

🚪 Conclusion

⚖️ or ?

+⇒	−⇒
Development time + cost	Computation Time + cost
Richer than SQL for data transformations	Overwhelming options
Lingua franca for DS + MLE	Less accessible to analysts
Full (batch) MLOps capabilities with python + SQL	Microservices are easier to troubleshoot, dbt single source of failure

Thanks

❓ Do you have any questions?

✉️ devdnhee@gmail.com / dorian@dataroots.io

🔗

https://github.com/datarootsio/your-best-bet

Resources

dbt

- Original discussion
- Python models
- github repo
- European soccer database