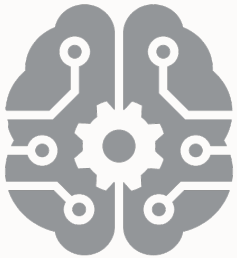
ORACLE



Oracle Machine Learning: Python for the Enterprise

Move the algorithms, not the data

Vili Tajnić
Principal Solutions Engineer
OD Prime Tech EE
June 2022

Safe harbor statement

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, timing, and pricing of any features or functionality described for Oracle's products may change and remains at the sole discretion of Oracle Corporation.

Why data scientists and data analysts use Python

Powerful

Extensible

Graphical

Extensive statistics

Ease of installation and use

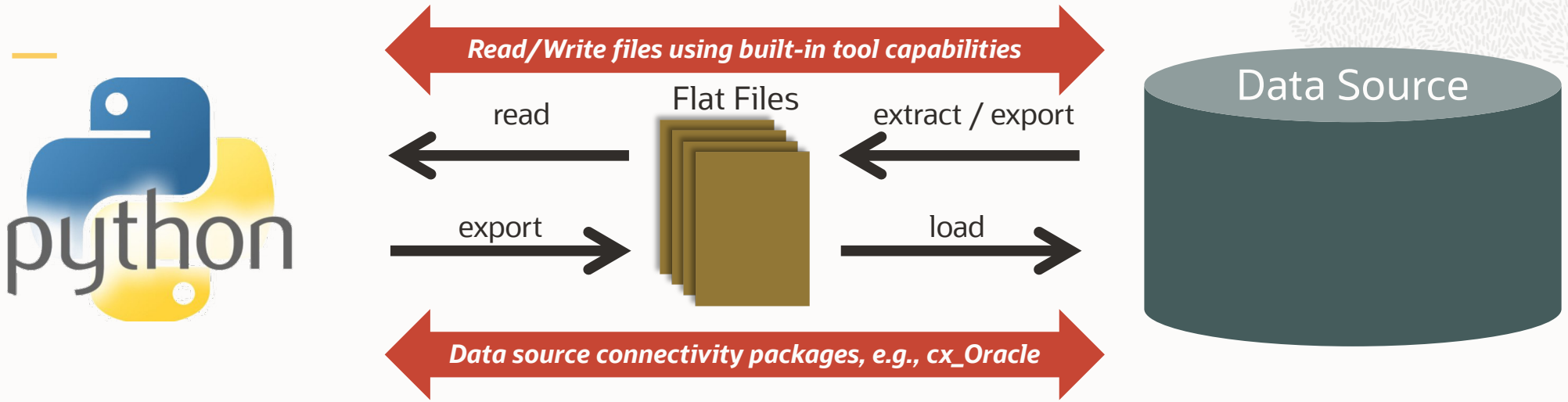
Rich ecosystem

- 1000s of open-source packages (pypi.org)
- Millions of users worldwide

Heavily used by data scientists

Free

Traditional Analytics and Data Source Interaction



Deployment
Ad hoc
cron job

Access latency
Paradigm shift: Python → *Data Access Language* → Python
Memory limitation – data size, in-memory processing
Single threaded
Issues for backup, recovery, security
Ad hoc production deployment

Oracle Machine Learning

OML4SQL

OML Notebooks

Collaborative notebook environment based on Apache Zeppelin with Autonomous Database

Interfaces for 3 popular data science languages: SQL, R, and Python

OML4R

Oracle Data Miner

SQL Developer extension to create, schedule, and deploy ML solutions through a drag-and-drop interface

OML4Py

OML4Spark

ML for the big data environment from R with scalable algorithms

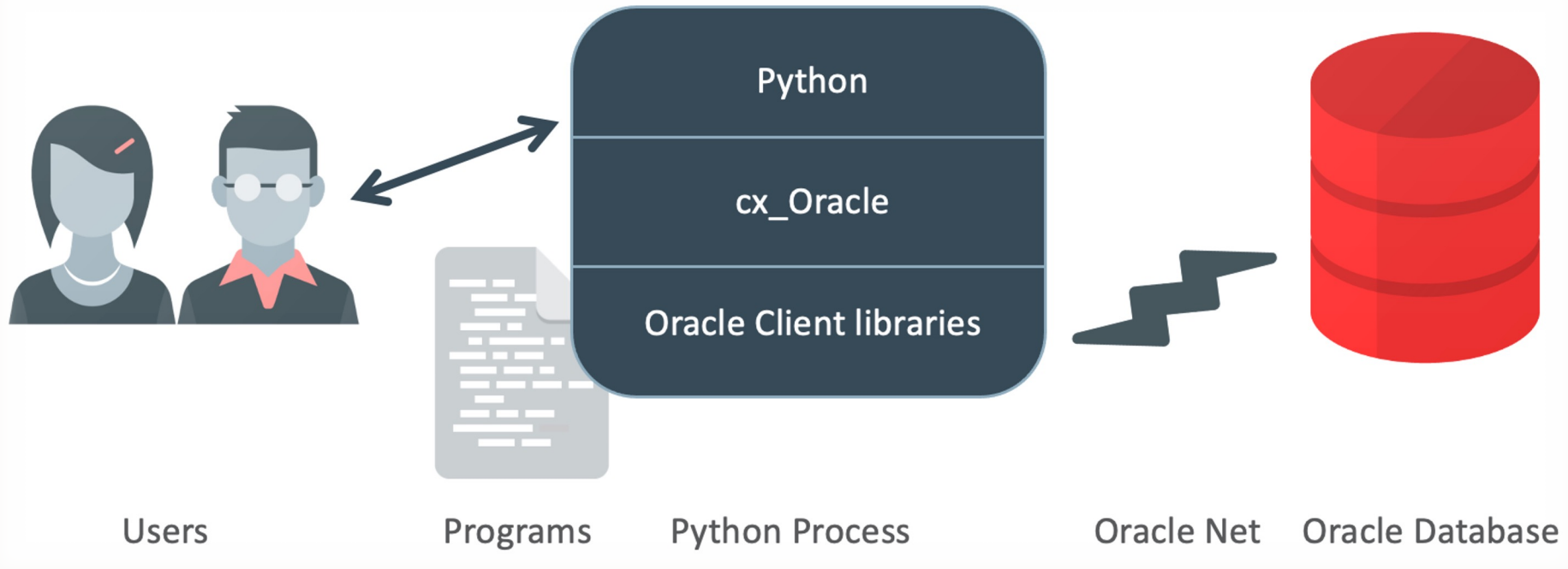
No-code AutoML interface on Autonomous Database

OML AutoML UI

Model Deployment and Management, Cognitive Text

OML Services

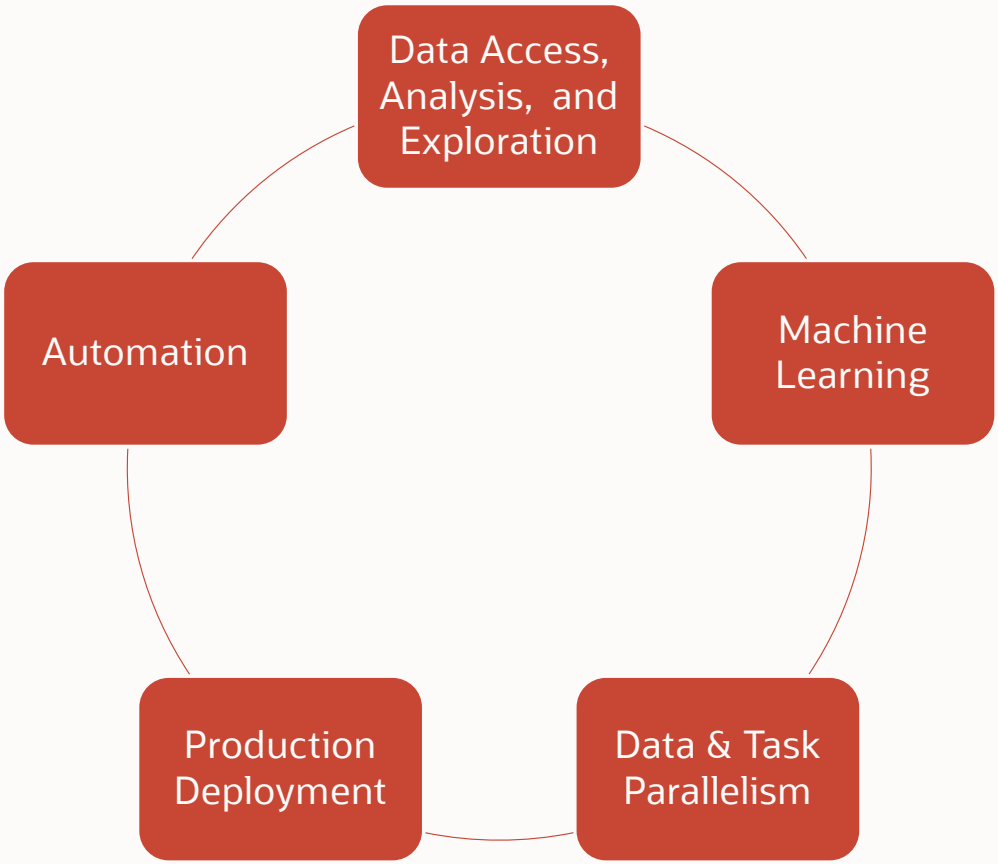
OML4Py Architecture (cx_Oracle + Oracle Client Libraries)



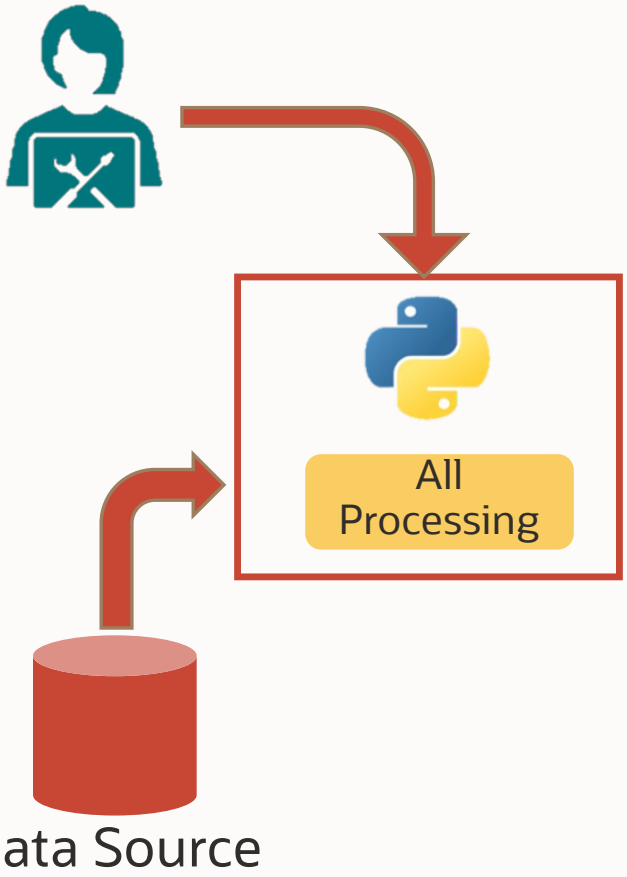
Major Features of OML4Py

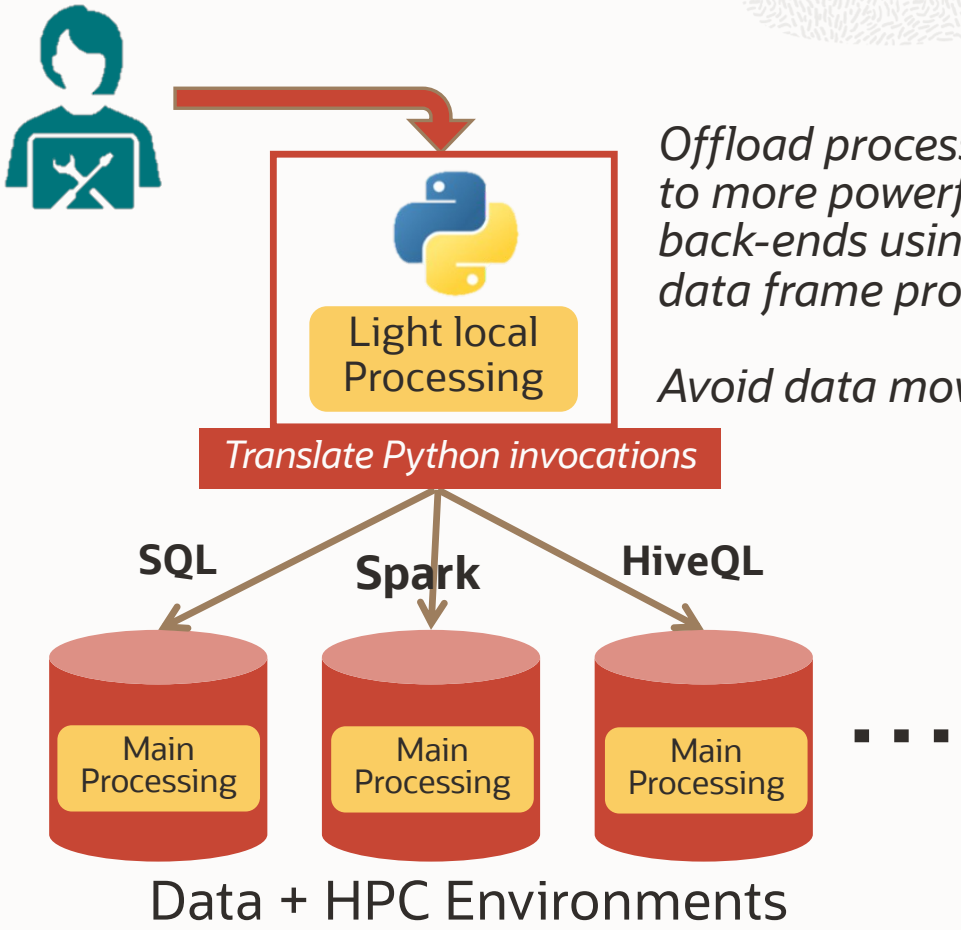
Transparency Layer	With the transparency layer classes, you can convert select **Python objects to Oracle database objects** and also invoke a range of familiar Python functions that are **overloaded to invoke** the corresponding SQL on tables in the database.
In-Database Machine Learning	Classes provide access to in-database machine learning algorithms.
Embedded Python Execution	You may choose to run your functions in a **data-parallel** or **task-parallel** manner in one or more of these Python engines.
Automated Machine Learning	Automated Machine Learning (AutoML) provides built-in data science expertise about data analytics and modeling that you can employ to build machine learning models.

Elements affecting enterprise scalability for Python



Data access, analysis, and exploration





Offload processing to more powerful back-ends using data frame proxies

Avoid data movement

Open-Source Language access to Oracle Database
OML4Py

```
import oml
import pandas as pd
import matplotlib.pyplot as plt
oml.connect("oml_user", ...)
```

```
t = oml.sync(table="*",regex_match=True)
t.keys()
oml.dir()
ONTIME_S = t.get("ONTIME_S","none")
ONTIME_S.head()
```

```
ONTIME_S.shape()
ONTIME_S.describe()
ONTIME_S[['ARRDELAY','DEPDELAY']].corr()
```


Open-Source Language access to Oracle Database
OML4Py

```
letters = list(map(chr, range(97, 123)))
df1 = pd.DataFrame({'x1':range(1,6),
                    'x2':letters[1:6]})
df2 = pd.DataFrame({'x1':range(1,6),
                    'x2':letters[11:16]})
oml.drop("TEST_DF1")
oml.drop("TEST_DF2")

TEST_DF1 = oml.create(df1, table="TEST_DF1")
TEST_DF2 = oml.create(df2, table="TEST_DF2")
```

```
df2.merge(df1,on='x1')
```


Data transfer-related functions

oml.create(x, table[, oranumber, dbtypes, . . .])

- **Creates a table** in Oracle Database from a Pandas DataFrame returning a proxy object

oml.push(x[, oranumber, dbtypes])

- Pushes data to Oracle Database **creating a temporary** table returning a proxy object

oml.sync(schema=None, regex_match=False, table=None, view=None, query=None)

- Creates a **DataFrame proxy object** in Python that represents an Oracle Database table

oml.drop([table, view])

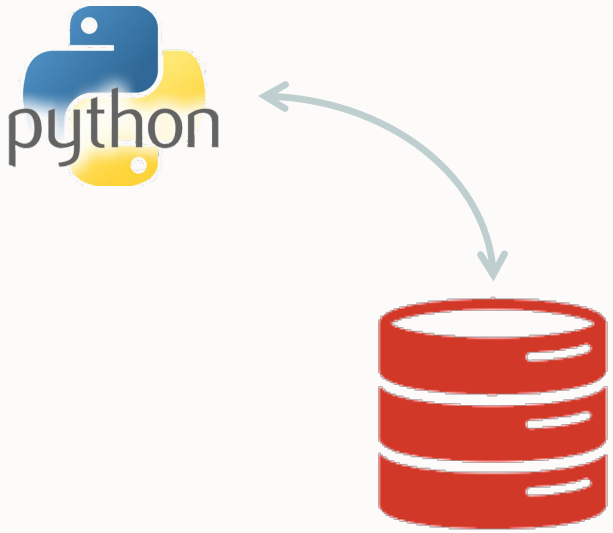
- **Drops** the named database table or view

oml.dir()

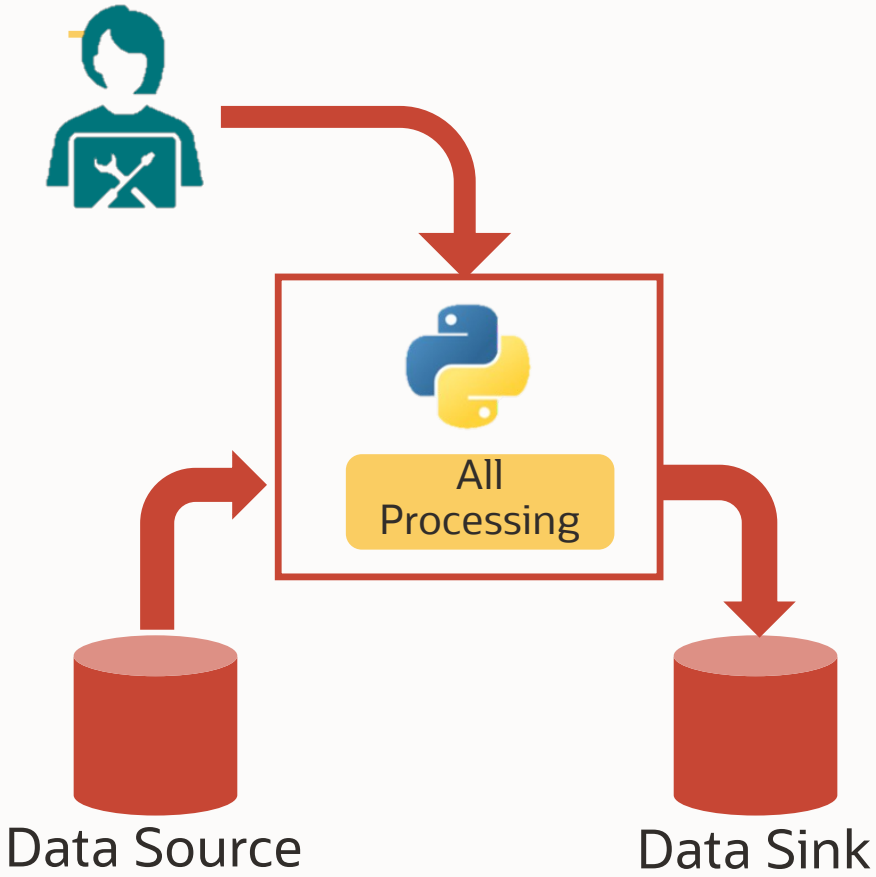
- **Returns the names** of OML objects in the workspace

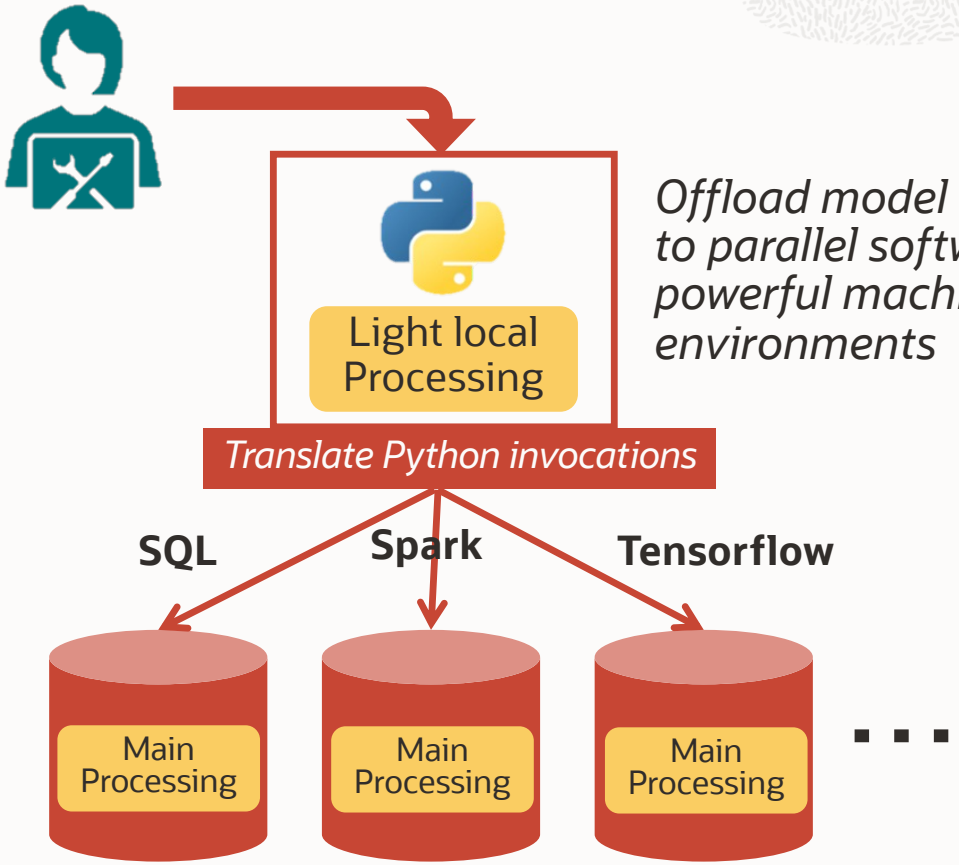
oml.cursor()

- Returns a **cx_Oracle cursor** object of the current OML database connection



Scalable Machine Learning





Offload model building to parallel software and powerful machines and environments

Scalable Machine Learning

Maintain open-source machine learning interface

- OML4Py – familiar Python predictors/target interface with fit() and predict()

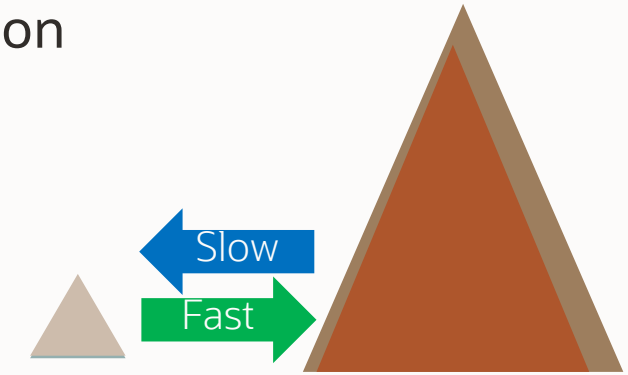
Scalable Machine Learning

Maintain open-source machine learning interface

- OML4Py – familiar Python predictors/target interface with fit() and predict()

Bring the algorithm to the data

- Eliminate or minimize data movement
- Leverage proxy objects to reference data from Python



Scalable Machine Learning

Maintain open-source machine learning interface

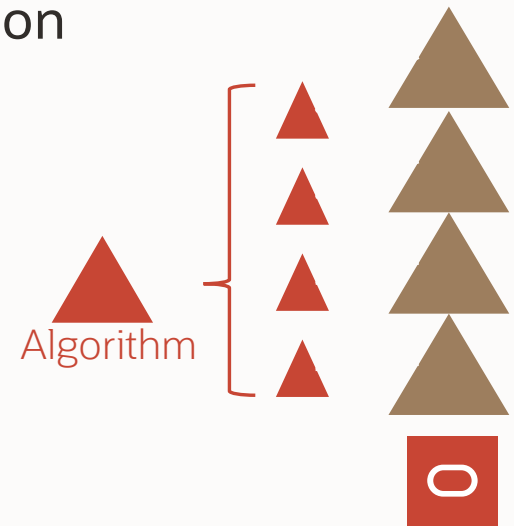
- OML4Py – familiar Python predictors/target interface with fit() and predict()

Bring the algorithm to the data

- Eliminate or minimize data movement
- Leverage proxy objects to reference data from R/Python

Parallel, distributed algorithm implementations

- Custom state-of-the-art integrated implementations
- Supplement with open-source packages and toolkits



OML4Py list

Machine Learning

Supports automatic

Classification

Predict target variable containing 2 (binary) or more (multi-class) category values

Decision Tree	Generates human-interpretable rules, can be used for segmentation
Explicit Semantic Analysis	Text categorization suitable for large text corpora
Extreme Gradient Boosting	Scalable implementation of popular XGBoost algorithm; supports tree and linear models
Logistic Regression / Generalized Linear Model	Predict binary (0/1, Yes/No) target attributes with attribute coefficients and model statistics; narrow, wide, sparse data; enables ridge, feature selection/generation; row diagnostics
Naïve Bayes	Computes conditional probabilities and yields interpretable probabilities; assumes predictor attribute independence
Neural Network	Well-suited to noisy and complex data, supports many hidden layers
Random Forest	Tree-based ensemble method that relies on bagging and feature randomness
Support Vector Machine	Solves linear and non-linear problems; multiple solvers; sparsity optimizations; supports multi-target classification (a list of targets per row)

Regression

Predict numeric target variable

Extreme Gradient Boosting	Scalable implementation of popular XGBoost algorithm; supports tree and linear models
Generalized Linear Model	Predict numeric target attributes with attribute coefficients and model statistics; narrow, wide, sparse data; enables ridge, feature selection/generation; row diagnostics
Neural Network	Well-suited to noisy and complex data, supports many hidden layers
Stepwise Regression	Selects "best" set of predictors for linear model; supports forward, backward, both, and alternate direction
Support Vector Machine	Solves linear and non-linear problems; multiple solvers; sparsity optimizations

Attribute Importance

Supervised and unsupervised ranking of variables to improve model quality

CUR Decomposition	Supports a low-rank SVD-based approach for ranking attribute importance as unsupervised method
Expectation Maximization	Supports unsupervised variable ranking and pairwise dependency estimates
Minimum Description Length	Select most important variables for classification and regression;

Ranking

Supervised prediction probability of one item ranking over other items

Extreme Gradient Boosting	Supports pairwise and list-wise ranking

Clustering

Group or segment cases into hierarchical clusters producing probabilities, rules, and statistics

Expectation Maximization	Automated model search; protection against overfitting; numeric and multinomial distributions; high quality probability estimates
K-Means	Produces specified number, k, of clusters; Euclidean and cosine distance functions; sparsity optimizations
Orthogonal Partitioning	Discovers natural clusters up to maximum number specified; density-based

Feature Extraction

Derive new values where all Input variables considered to generate reduced set of variables

Explicit Semantic Analysis	Text categorization with human-readable topic labels derived from corpus; semantic similarity estimates among documents
Non-negative Matrix Factorization	Derives features based on non-negative linear combinations for greater feature interpretability
Principal Component Analysis	Uses SVD to obtain a set of uncorrelated variables that contain the maximum amount of variance from dataset
Singular Value Decomposition	Narrow data via tall and skinny solvers; wide data via stochastic solvers

Anomaly Detection

Identify cases as normal or anomalous by learning patterns of normal data

One-Class SVM	Special case of SVM classification that does not use a target; Solves linear and non-linear problems; multiple solvers; sparsity optimizations
MSET-SPRT	Process monitoring to detect anomalies with non-linear, non-parametric patterns in IoT sensor data; "Multivariate State Estimation Technique"

Time Series

Forecast or predict sequential numeric data using series order column with either Number or Date/Timestamp types

Exponential Smoothing	Single, double, and triple exponential smoothing for regular and irregular series, with and without trend and seasonality; multiple methods supported, including Holt-Winters

Association Rules

Market basket analysis using transactional or 2D data representation to extract frequently occurring patterns and rules

Apriori	Finds frequent itemsets and generates human-interpretable rules; computes support, confidence, lift, and aggregate measures associated with rules

Row Importance

Unsupervised ranking of rows

CUR Decomposition	Supports low-rank SVD-based approach for ranking row importance as unsupervised method

In-database scalable aggregation

Example using the crosstab function

```
ONTIME_S = oml.sync(table="ONTIME_S")
res = ONTIME_S.crosstab('DEST')
type(res)
res.head()
```

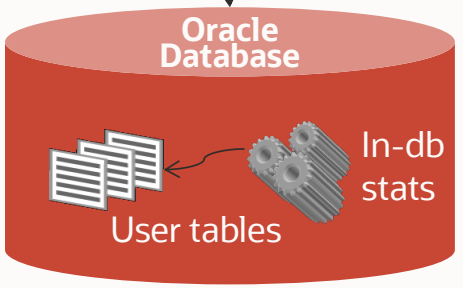
```
>>> ONTIME_S = oml.sync(table="ONTIME_S")
>>> res=ONTIME_S.crosstab('DEST')
>>> type(res)
<class 'oml.core.frame.DataFrame'>
>>> res.head()
  DEST  count
0  ABE    237
1  ABI     34
2  ABQ   1357
3  ABY     10
4  ACK      3
```

Source data is a DataFrame, ONTIME_S, which is an Oracle Database table

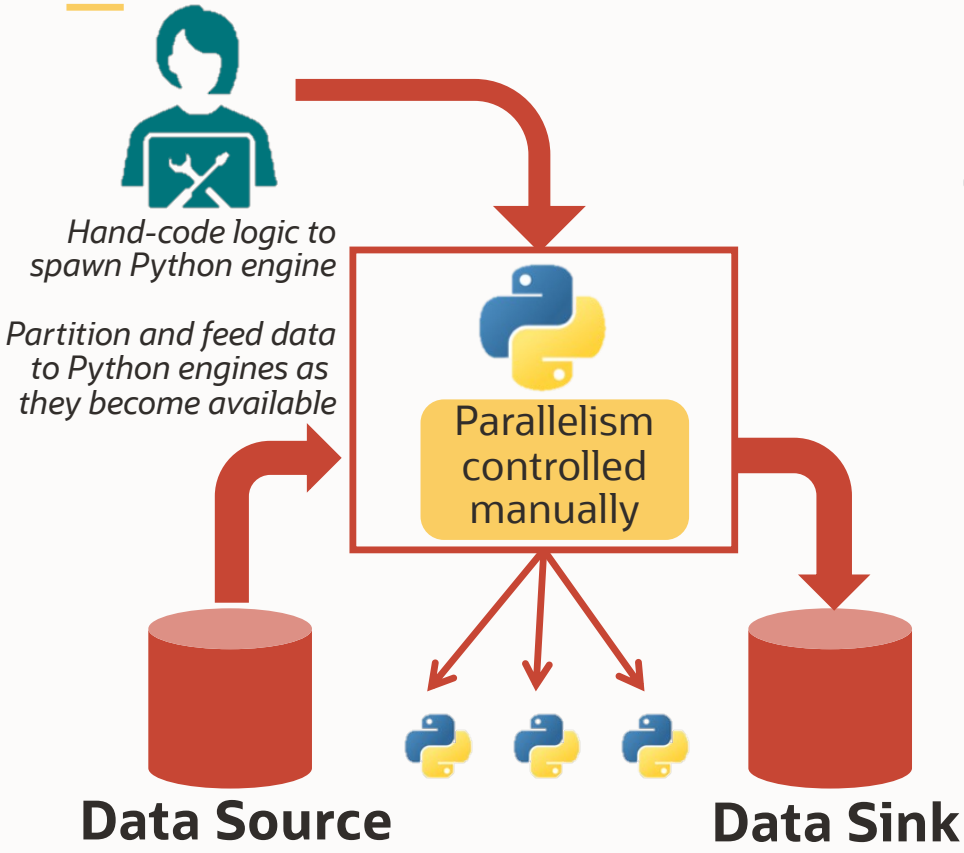
crosstab() function overloaded to accept OML DataFrame objects and transparently generates SQL for scalable processing in Oracle Database

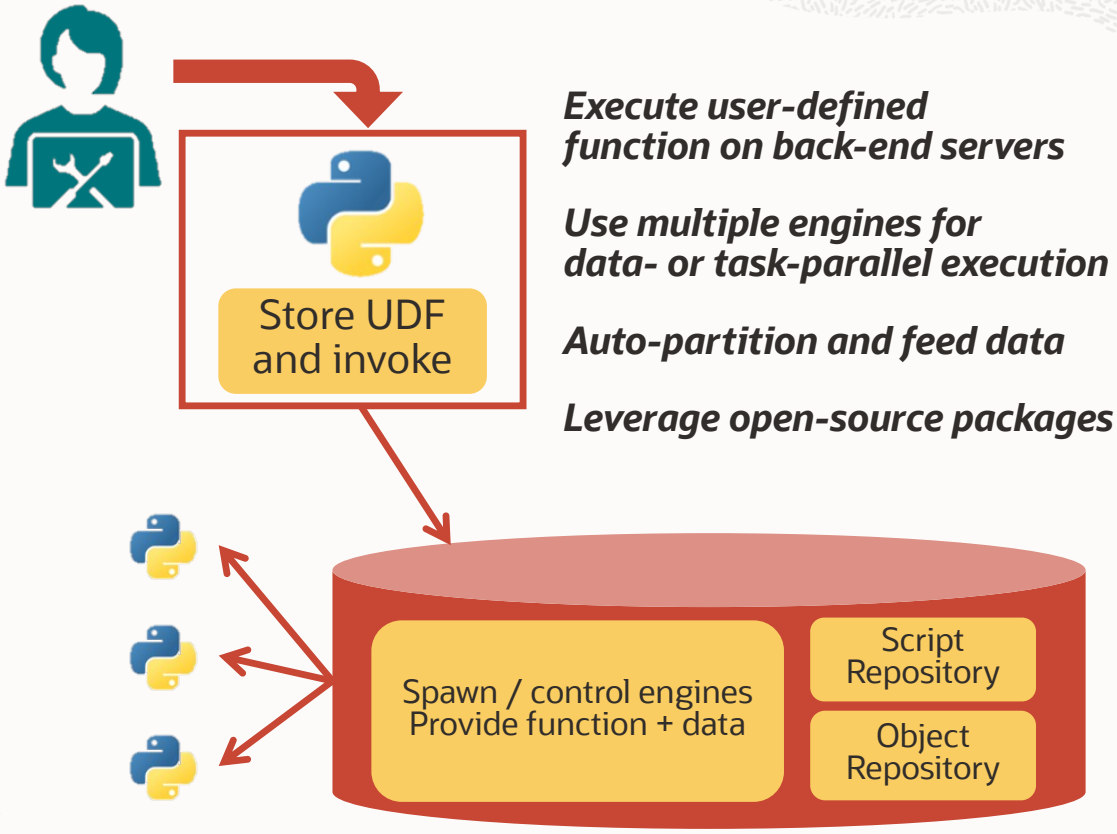
Returns an 'oml.core.frame.DataFrame' object


```
select DEST, count(*)
from ONTIME_S
group by DEST
```



Data and Task Parallel Execution





Execute user-defined function on back-end servers

Use multiple engines for data- or task-parallel execution

Auto-partition and feed data

Leverage open-source packages

Data and Task Parallel Execution

Easily specify parallelism and data partitioning

- Simplified API – all-in-one
- Build and score with large number of open-source models
- Partition data by value or count
- Invoke user-defined functions in parallel with index input

Automated management of Python engines

- Insulation from hardware details
- Limit memory and compute resources, as possible

Automated load of data and user-defined functions into Python engines

Leverage packages from open-source ecosystems

Example of OML4Py – Table Apply
OML4Py

```
def build_nb(dat, dsname):
    import oml
    from sklearn.naive_bayes import GaussianNB
    from sklearn import preprocessing
    le = preprocessing.LabelEncoder()
    raw_labels = dat[["Species"]].values
    le.fit(raw_labels)
    y = le.transform(raw_labels)
    X = dat[["SEPAL_LENGTH","SEPAL_WIDTH",  "PETAL_LENGTH","PETAL_WIDTH"]].values
    mod = GaussianNB().fit(X, y)
    oml.ds.save(objs={'mod':mod},name=dsname,
                overwrite=True)
```

```
mod = oml.table_apply(IRIS, build_nb,
                      dsname = 'NB_Model-1',
                      oml_connect=True)
```

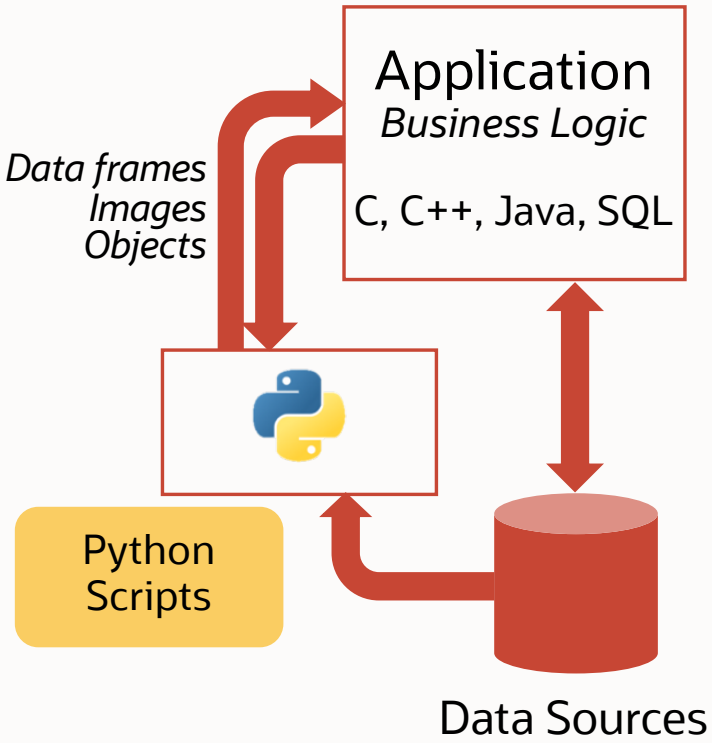

Example of OML4Py – Row Apply

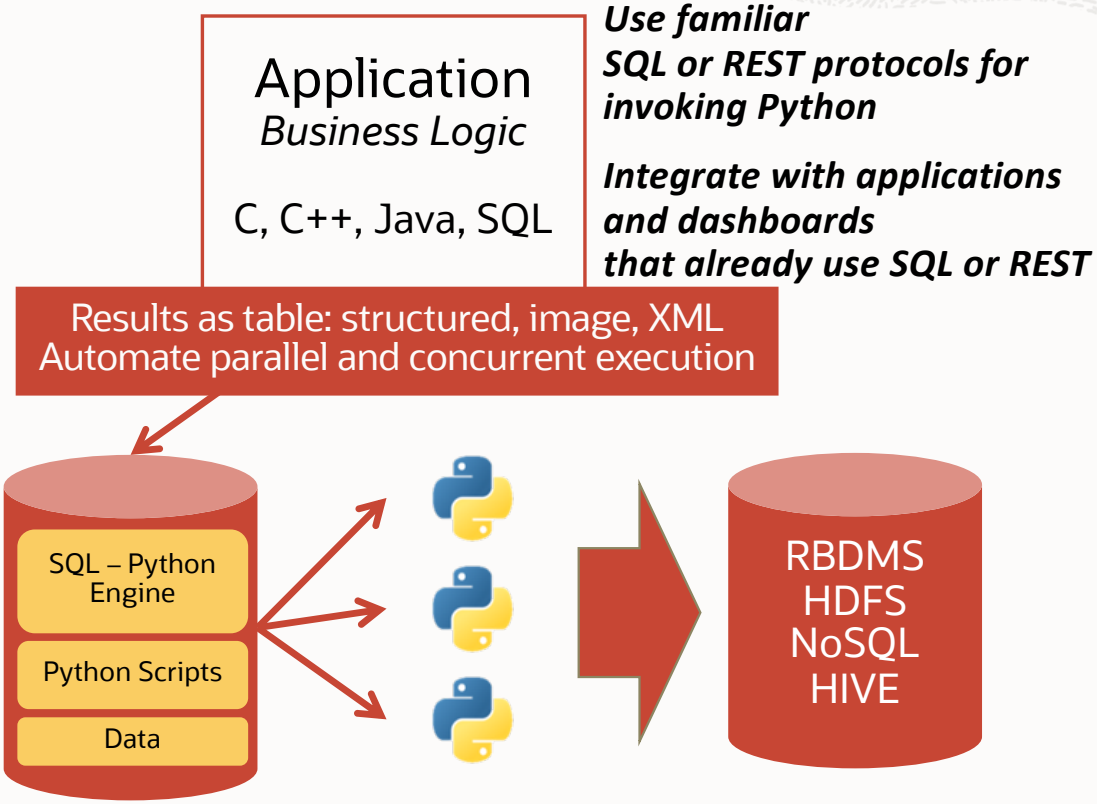
OML4Py

```
def score_nb_mod(dat, dsname):
    import oml
    from sklearn.naive_bayes import GaussianNB
    objs = oml.ds.load(dsname,to_globals=False)
    mod = objs['mod']
    dat['PREDICTION'] =
       mod.predict(dat.drop('Species',axis=1))
    return dat
```

```
IRIS_PRED = pd.DataFrame([(1,1,1,1,'a',1)],
   columns=["SEPAL_LENGTH","SEPAL_WIDTH",
            "PETAL_LENGTH","PETAL_WIDTH",
            "Species","PREDICTION"]))
res = oml.row_apply(IRIS,score_nb_mod,
                    dsname = 'NB_Model-1',
                    parallel = 4, rows = 10
                    func_value = IRIS_PRED,
                    oml_connect = True)
```

Deployment





Create user-defined functions from SQL (or use from Python)

OML4Py

```
BEGIN
   sys.pyqScriptDrop('RandomRedDots')
   sys.pyqScriptCreate('RandomRedDots')
 def RandomRedDots ():
    import numpy as np
    import pandas as pd
    import matplotlib.pyplot as plt

    d = {''id'': range(1,10),
         ''val'': [x/100 for x in range(1,10)]}
    df = pd.DataFrame(data=d)
    fig = plt.figure(1)
    ax = fig.add_subplot(111)
    ax.scatter(range(0,100),
               np.random.rand(100),c=''r'')
    fig.suptitle("Random Red Dots")
    return df', NULL, TRUE)
END;
```


Invoke user-defined functions from SQL
OML4Py

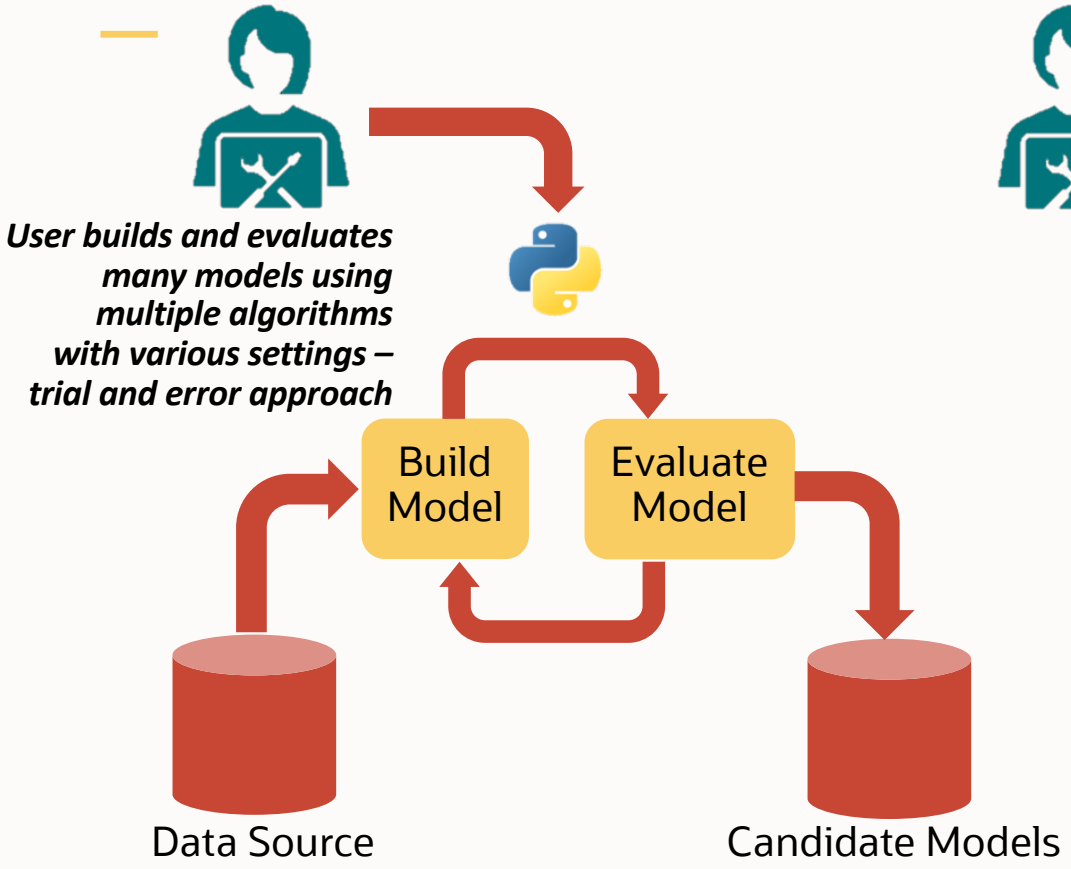
```
select ID, IMAGE from table(pyqEval(NULL,'PNG','RandomRedDots'))
```

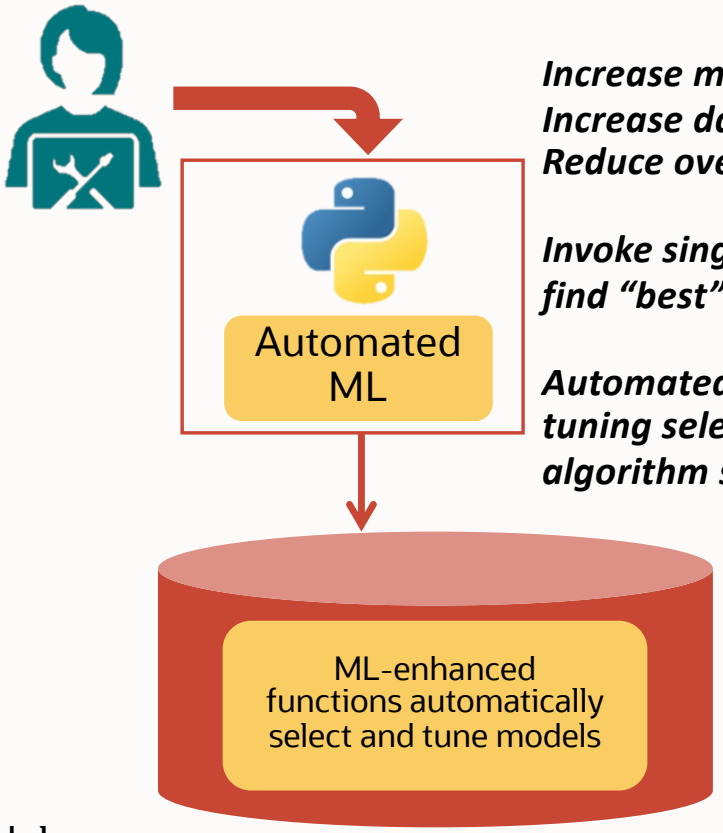
```
select ID, VAL from
  table(pyqEval(NULL,
                'select 1 id, 1 val from dual',
                'RandomRedDots'))
```

```
select dbms_lob.substr(VALUE,4000,1) from
  table(pyqEval(NULL,'XML','RandomRedDots'))
```

```
# In Python, invoke same function by name
res = oml.do_eval(func='RandomRedDots')
```


Automation





Increase model quality
Increase data scientist productivity
Reduce overall compute time

Invoke single function to find "best" algorithm and model

Automated hyperparameter tuning selects "best" algorithm settings

AutoML – with OML4Py

Increase data scientist productivity – reduce overall compute time

Auto Algorithm Selection

- Identify in-database algorithm that achieves highest model quality
- Find best model faster than with exhaustive search

Auto Feature Selection

- Reduce # of features by identifying most predictive
- Improve performance and accuracy

Auto Tune Hyperparameters

- Significantly improve model accuracy
- Avoid manual or exhaustive search techniques

Enables non-expert users to leverage Machine Learning

Auto Tune

Significantly improve model accuracy guided by ML

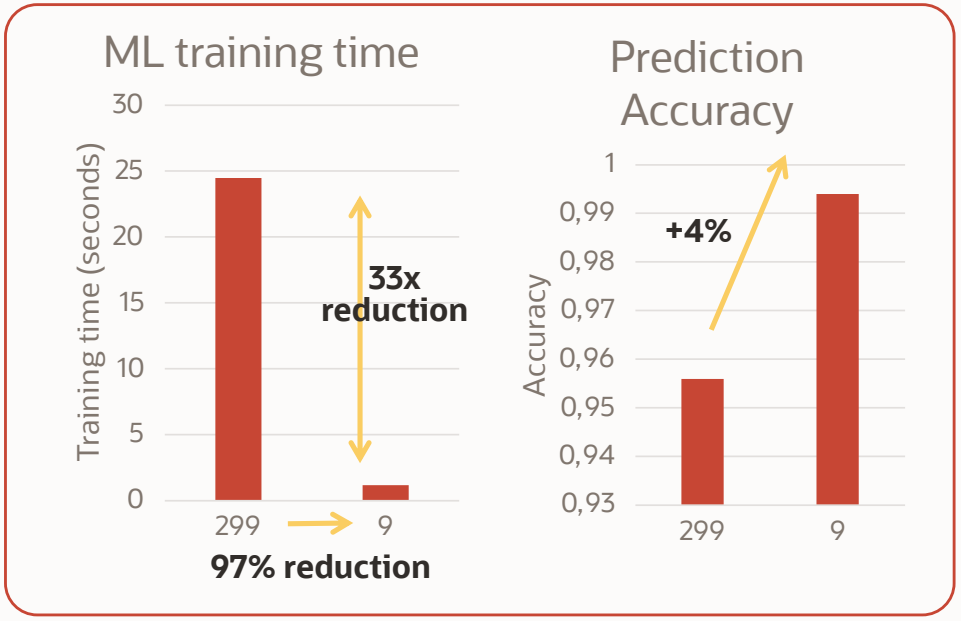
Avoid manual or exhaustive search techniques

OML4Py

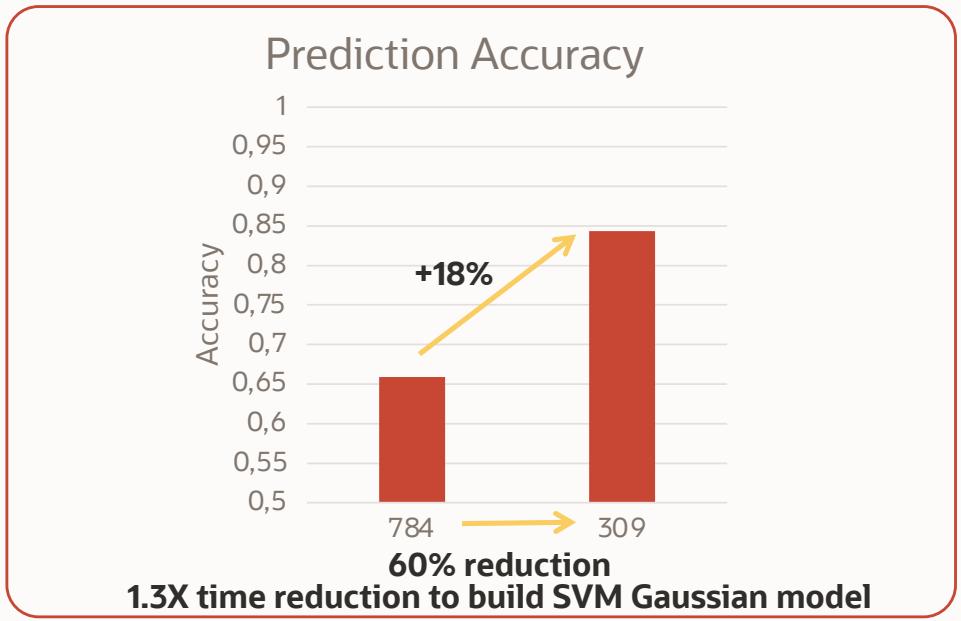
```
at = Autotune(mining_function = 'classification',
              score_metric = 'accuracy',
                   parallel = 4)

results = at.tune('dt', X_train, y_train)

best_mod = results['best_model']
all_evals = results['all_evals']
```


OML4Py Auto Feature Selection: examples

Reduce # of features by identifying most relevant, improving performance and accuracy



Open dataset 312 with 1925 rows, 299 columns



OpenML dataset 40996 with 56K rows, 784 columns

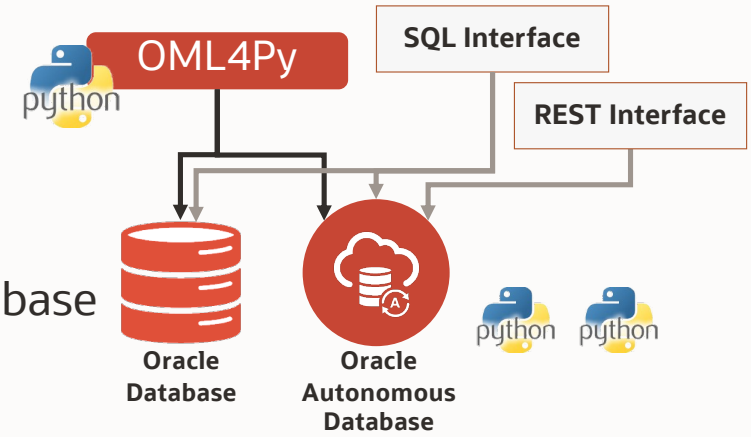
Oracle Machine Learning for Python advantages

Use Oracle Database as HPC environment

- Explore, transform, and analyze data faster and at scale

Use in-database parallelized and distributed ML algorithms

- Build more models on more data, and score large volume data – faster
- Use in-database algorithms from OML4SQL via natural Python API
- Increased productivity from automatic data preparation, partitioned models, and integrated text mining capabilities



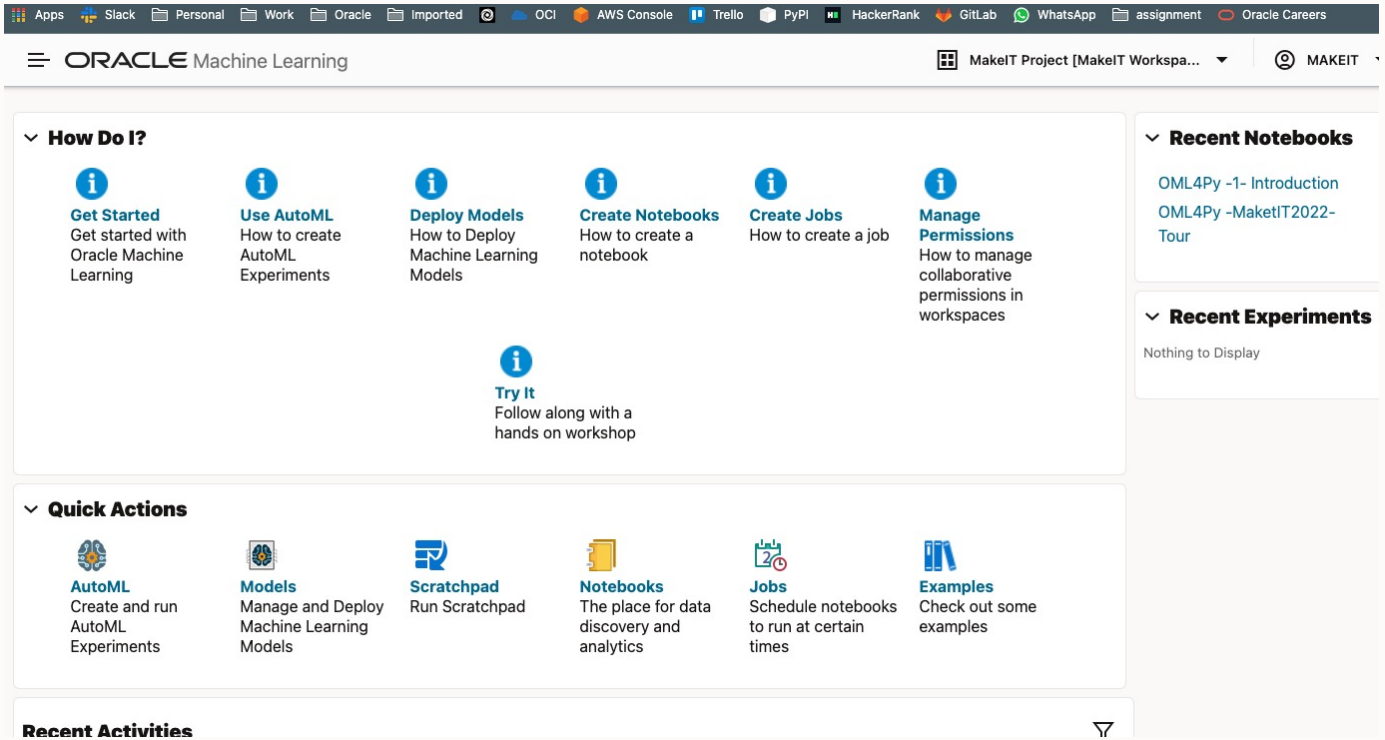
Run user-defined Python functions in database-spawned and controlled Python engines and manage Python objects in-database

- Collaborate: hand-off data science products from data scientist to developers easily
- Run user-defined functions in data-parallel, task-parallel, and non-parallel fashion
- Return structured and image results in Python and REST API

New automated machine learning (AutoML) and model explainability (MLX)

- Enhance data scientist productivity and enable non-experts to use and benefit from machine learning
- Algorithm selection, feature selection, hyperparameter tuning, model selection
- Model-agnostic identification of important features that impact model predictions

Demo OCI Machine Learning - max 10 min.



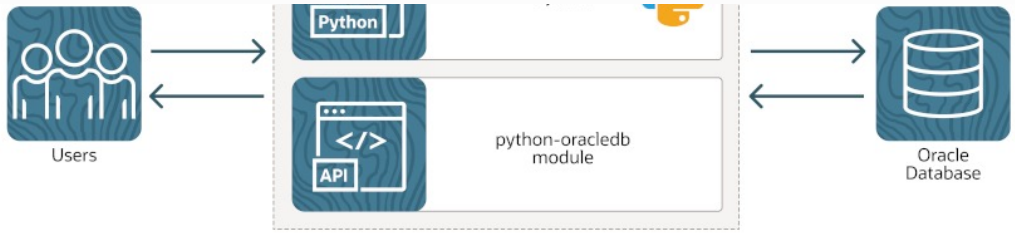
ORACLE

Python python-oracledb Driver

Installation Documentation Release Notes Help Source code

```
python -m pip install oracledb
```

```
import oracledb as cx_Oracle
```



The module conforms to the Python Database API 2.0 specification with a considerable number of additions and a couple of minor exclusions, see the feature list.

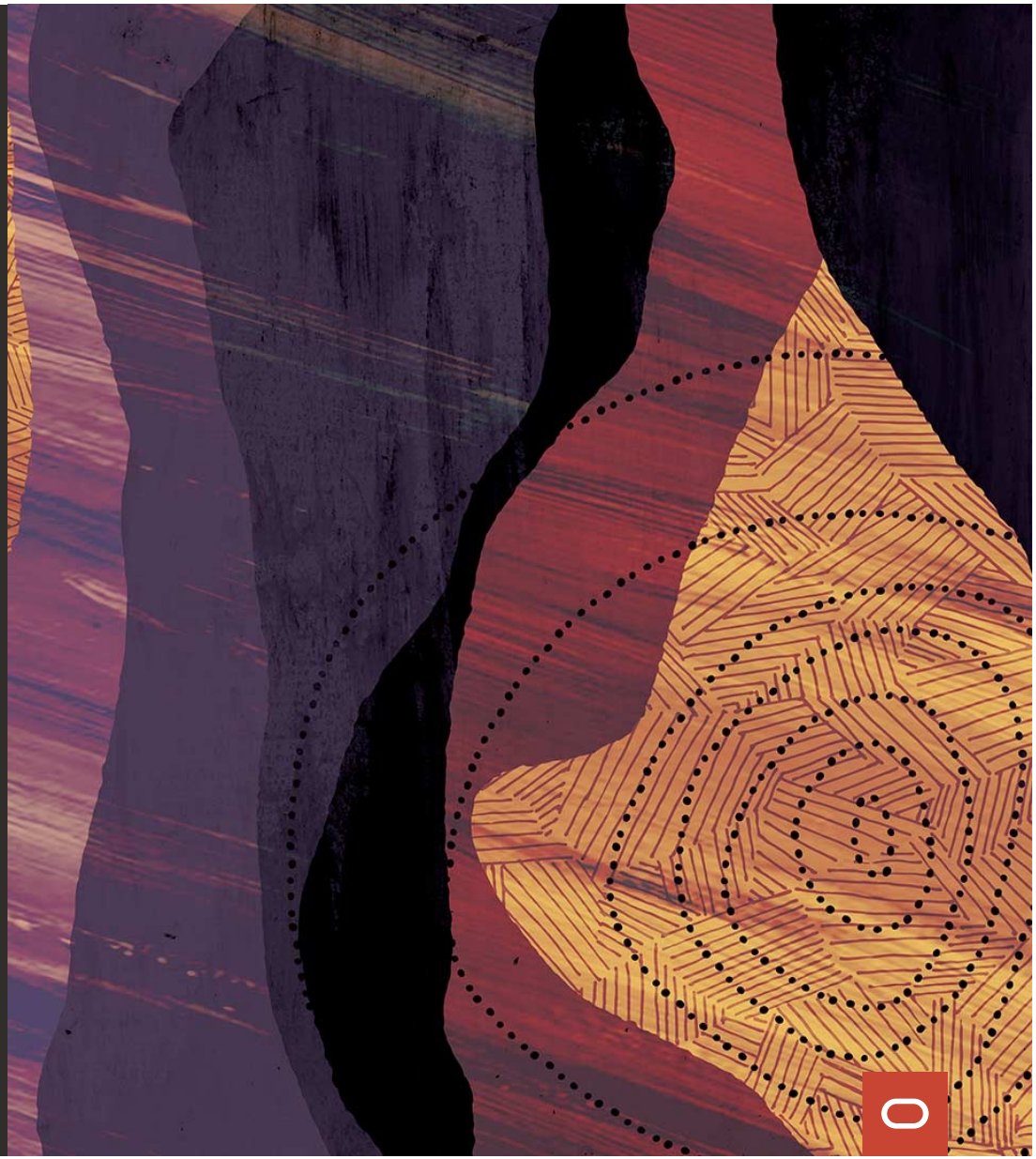
https://medium.com/@sharad-chandran/upgrade-your-python-apps-from-cx-oracle-8-to-python-oracledb-2bfeaee7aac9

Next Steps

https://apexapps.oracle.com/pls/apex/dbpm/r/livelabs/view-workshop?wid=922

Thank you

Vili Tajnić
@vilita
vili.tajnic@oracle.com



Our mission is to help people
see data in new ways, discover insights,
unlock endless possibilities.


ORACLE