



Python Machine Learning in Oracle Autonomous ADB

Natwest Developer Session – HOL Details

July 2021

Introduction



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

Code-free AutoML interface on Autonomous Database

Model Deployment and Management, Cognitive Text

Collaborative notebook environment based on Apache Zeppelin with Autonomous Database

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

ML for the big data environment from R with scalable algorithms

Oracle Machine Learning

OML4SQL

OML Notebooks

OML4R

Oracle Data Miner

OML4Py

OML4Spark

OML AutoML UI

OML Services





Oracle Machine Learning Notebooks

Autonomous Database as a Data Science Platform



Collaborative UI

- Based on Apache Zeppelin
- Supports data scientists, data analysts, application developers, DBAs with SQL and Python
- Easy notebook sharing
- Permissions, versioning, and scheduling of notebooks

Included with Autonomous Database

- Automatically provisioned and managed
- In-database algorithms and analytics functions
- Explore and prepare, build and evaluate models, score data, deploy solutions

The screenshot shows the Oracle Machine Learning Notebooks interface. At the top, there's a navigation bar with the Oracle logo, a search bar, and user information. Below the header, the title "Oracle Machine Learning for Python" is displayed.

The first code cell, titled "Overloaded data visualization functions", contains Python code to generate boxplots for the IRIS dataset. The resulting boxplot visualizes the distribution of four attributes: Sepal Length, Sepal Width, Petal Length, and Petal Width. The x-axis labels are "SEPAL_LENGTH", "SEPAL_WIDTH", "PETAL_LENGTH", and "PETAL_WIDTH". The y-axis ranges from 0 to 8 for Sepal attributes and 0 to 7 for Petal attributes.

The second code cell, titled "Sepal Length variation in IRIS data set", uses the `hist` function to create a histogram of Sepal Length. The x-axis is labeled "Sepal Length" and ranges from 4.5 to 8.0. The y-axis is labeled "# of iris instances" and ranges from 0 to 25. The histogram bars are red.

The third code cell, titled "Create derived variables", contains a single line of Python code: `is_large_petal = (IRIS['PETAL_LENGTH'] > 5.0) & (IRIS['PETAL_WIDTH'] > 2.0)`. This creates a new variable based on petal dimensions.



Oracle Machine Learning for Python

Supported in Oracle Autonomous Database with OML Notebooks

Use Oracle Database as HPC environment

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

Use in-database parallelized and distributed ML algorithms

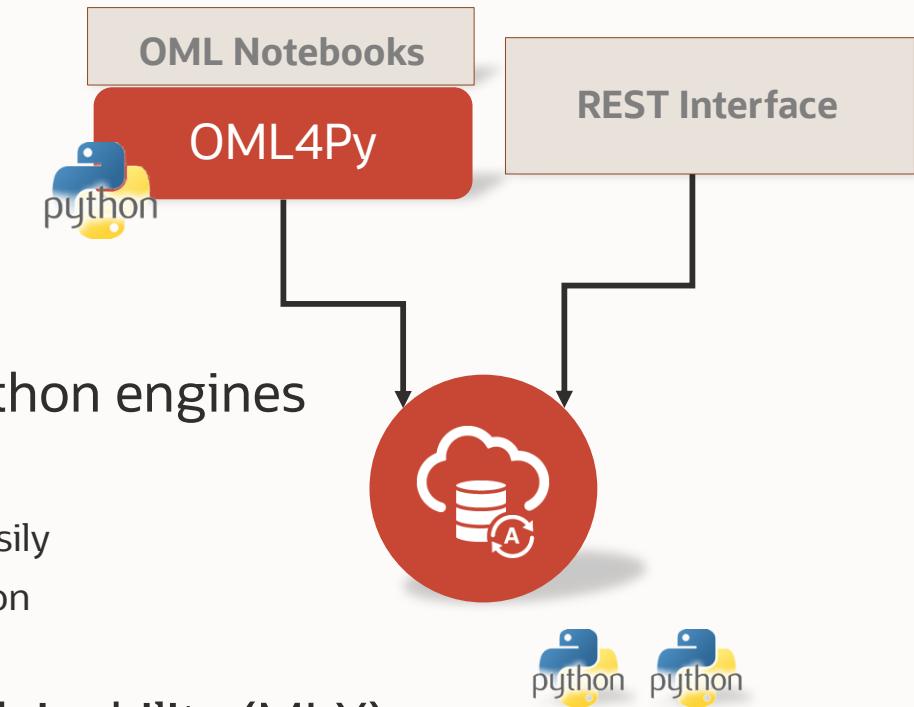
- Build more models on more data, and score large volume data – faster
- Use in-database algorithms from OML4SQL via well-integrated Python API
- Increased productivity from automatic data preparation, partitioned models, and integrated text mining capabilities

Run Python scripts 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



Labs overview



Autonomous Data Warehouse

Overview

Activity

Administration

Development

1

Download Client Credentials (Wallet)

i

Connections to Autonomous Data Warehouse use a secure connection. Your existing tools and applications will need to use this wallet file to connect to your Autonomous Data Warehouse instance. If you are familiar with using an Oracle Database within your own data center, you may not have previously used these secure connections.

Set Administrator Password

2

Set or reset your database administrator user's (ADMIN) password and when locked unlock your administrator user account on Autonomous Data Warehouse.

Send Feedback to Oracle

Use our Cloud Customer Connect forum to provide feedback about the service to Oracle, post questions, connect with experts, and share your thoughts and ideas. Click here to link to the forum.

Set Resource Management Rules

i

Set resource management rules to allocate CPU/IO shares to consumer groups and to cancel SQL statements based on their runtime and amount of IO.

Manage Oracle ML Users

i

Create new Oracle Machine Learning user accounts and manage the credentials for existing Oracle Machine Learning users.

DATABASE
OLIADB20210729

ORACLE® Machine Learning User Administration



Users

 Create Delete Show All Users

Search...



User Name	Full Name	Role	Email	Created On	Status
ADMIN		System Administrator		1/27/20 11:34 PM	Open

Autonomous Database | Oracle X Autonomous Data Warehouse | Oracle X Oracle Machine Learning User X Oracle Machine Learning Login X +

https://adb.eu-frankfurt-1.oraclecloud.com/omlusers/index.html?root=useredit&userid=create-new-user

Most Visited Getting Started

ORACLE® Machine Learning User Administration

Create User

1

2

3

4

Username: OMLUSER

Email Address:

Password: Welcome1#Welcome1#

Confirm Password:

Generate password and email account details to user. User will be required to reset the password on first sign in.

Create Cancel

* Username: OMLUSER

First Name:

Last Name:

* Email Address:

Generate password and email account details to user. User will be required to reset the password on first sign in.

Password: Welcome1#Welcome1#

Confirm Password:

Create User

Generate password and email account details to user. User will be required to reset the password on first sign in.

User Created

Users

Create	Delete	Show All Users	Search...			
User Name	Full Name	Role	Email	Created On	Status	
ADMIN		System Administrator		1/27/20 11:34 PM	Open	
OMLUSER		Developer	operard@yahoo.com	7/29/21 6:01 AM	Open	

Lab high-level outline

Lab 1: Getting started with OML4Py

Lab 2: Select and manipulate data using the Transparency Layer

Lab 3: Using in-database algorithms and models

Lab 4: Use datastores to store Python objects

Lab 5: Run user-defined functions using Embedded Python Execution

Lab 6: Use AutoML

Lab 6: Bonus Round - Use OML AutoML UI

Lab 1: Getting started with OML4Py

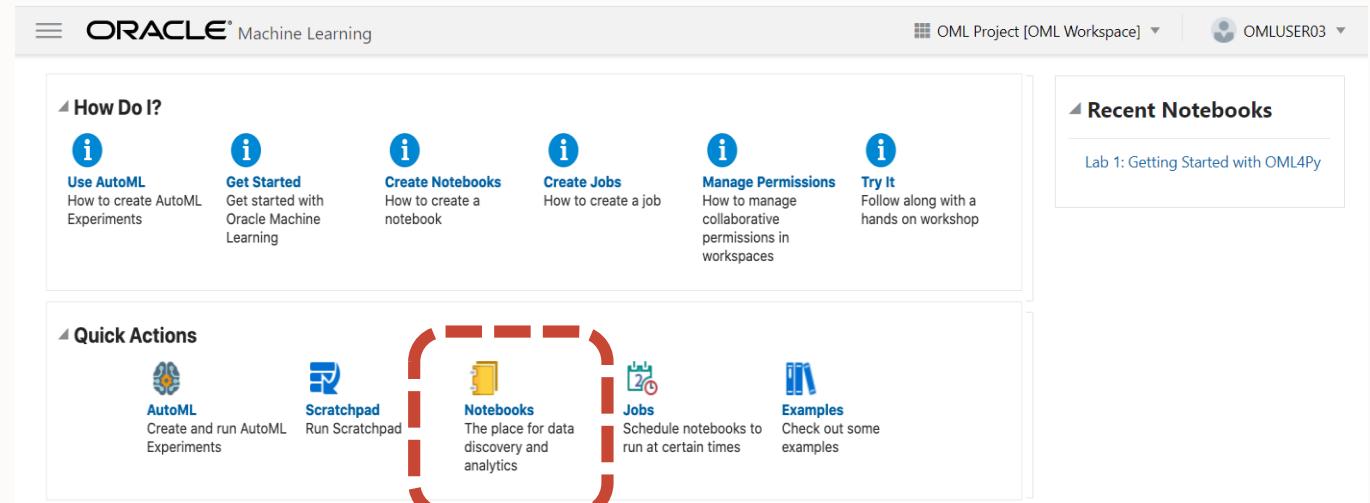


Lab 1 Getting Started with OML4Py

Step 1: In your browser (Chrome or Firefox), go to the provided URL

Step 2: Log in using provided username and password

Step 3: Click “Notebooks”



Step 4: Click the notebook for
"Lab 1: Getting Started with OML4Py"

The screenshot shows the 'Notebooks' list in the Oracle Machine Learning interface. The table has columns for Name, Comment, Last Update, Updated By, and Connection Group. There are two rows visible:

Name	Comment	Last Update	Updated By	Connection Group
Lab 1: Getting Started with OML4Py		1/17/21 2:15 AM	OMLUSER03	Global
Lab 2: Data Selection and Manipulation using the...		1/17/21 12:18 AM	OMLUSER03	Global

Lab 1: Getting Started with OML4Py

The screenshot displays the Oracle Machine Learning interface with several toolbars and status elements:

- Top Toolbar:** Includes "Show/Hide Output" and "Clear Notebook" buttons.
- Second Toolbar:** Includes "Run All", "Show/Hide Code", "Clear Output", "Export Notebook", and "Search Code" buttons.
- Third Toolbar:** Includes icons for Run, Stop, Refresh, Notebook, Cell, and Cell Type.
- User Status:** Shows "OML Project [OML Workspace]" and "OMLUSER03" with a "Connected" status.
- Bottom Status Bar:** Shows "FINISHED" and various status indicators (Width, Font size, Move down, Insert new, Run all below, Clone paragraph, Show title, Show line numbers, Disable run, Clear output, Remove).
- Bottom Buttons:** Includes "Paragraph Status", "Run", "Hide", "ParagraphOutput", "Show Editor", and "More Features".
- Right Side:** A red arrow points from the "Run" button towards a list of keyboard shortcuts on the right.
- Shortcuts List:** A table of keyboard shortcuts with their descriptions and hotkeys.

Lab 1: Getting Started with OML4Py

Oracle Machine Learning for Python (OML4Py)

Oracle Machine Learning for Python (OML4Py) makes the open source Python scripting language and environment ready for the enterprise and big data. Designed for problems involving both large and small data volumes, OML4Py integrates Python with Oracle Autonomous Database, allowing users to run Python commands and scripts for statistical, machine learning, and visualization analyses on database tables and views using Python syntax. Many familiar Python functions are overloaded that translate Python behavior into SQL for running in-database, as well as new automatic machine learning capabilities.

This example shows you how to:

- Set OML notebook bindings and interpreters
- Enable the Python interpreter to run Python commands
- Import the OML4Py `oml` package and verify the database connection

Copyright (c) 2020 Oracle Corporation

The Universal Permissive License (UPL), Version 1.0

Took 0 secs. Last updated by OMLUSER2 at January 15 2021, 12:40:41 PM.

Lab 2: Select and manipulate data using the Transparency Layer



Transparency Layer

In-database performance – indexes, query optimization, parallelism, partitioning

Leverages proxy objects for database data: *oml.DataFrame*

```
# Create table from Pandas DataFrame data
DATA = oml.create(data, table = 'BOSTON')

# Get proxy object to DB table boston
DATA = oml.sync(table = 'BOSTON')
```

Uses familiar Python syntax to manipulate database data
Overloads Python functions translating functionality to SQL

```
DATA.shape
DATA.head()
DATA.describe()
DATA.std()
DATA.skew()
```

```
TRAIN, TEST =
    DATA.split()
TRAIN.shape
TEST.shape
```

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])

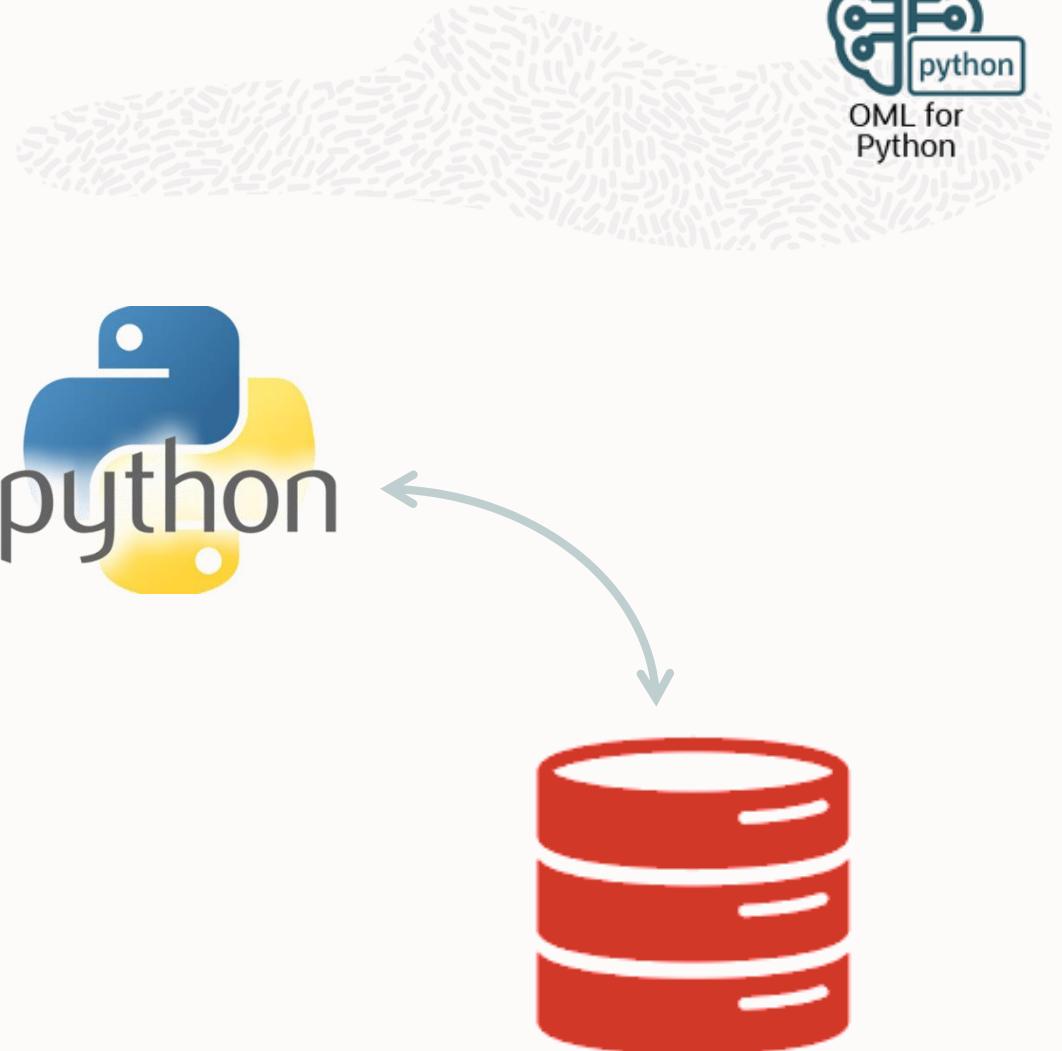
- Drops the named database table or view

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



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()
```

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

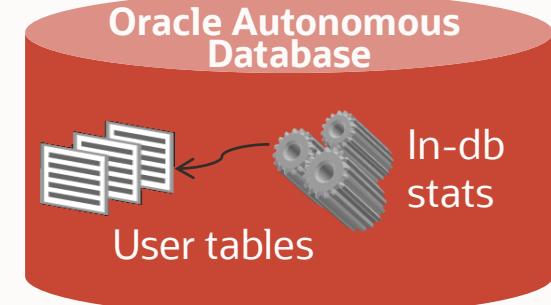
Returns an 'oml.core.frame.DataFrame' object



OML4Py

```
>>> 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
```

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



Functions on OML DataFrame executed in-database

KFold
append
columns
concat
corr
count
create_view
crosstab
cumsum
describe
drop
drop_duplicates

rename
round
select_types
shape
skew
sort_values
split
std
sum
t_dot
tail
types

dropna
head
kurtosis
materialize
max
mean
median
merge
min
nunique
pivot_table
pull

Lab 2: Select and manipulate data using the Transparency Layer

Goal: Become familiar with creating and using DataFrame proxy objects for exploring and transforming database tables and views

Step 1: Import libraries and create OML DataFrame proxy object

Step 2: Select table columns

Step 3: Select table rows

Step 4: Use Pandas DataFrame objects + TIY

Step 5: Use the split and KFold functions + TIY

Step 6: Use the crosstab and pivot_table functions

Step 7: Use oml.boxplot and oml.hist

Step 8: Create a persistent database table



In the Sign-In screen of OML Notebooks, use **OMLUSER** and the Password **AAbbcc123456**. Once you are in, click on the OML Notebooks link

ORACLE Cloud Infrastructure

SIGN IN

Database name: ADW3900

Sign in with your Oracle Machine Learning Database User credentials

USERNAME: OMLUSER
PASSWORD: AAbbcc123456
Sign In

Sign in with
OMLUSER
and the Password
AAbbcc123456

ORACLE Machine Learning

OMLUSER Project [OMLUSER Works...]

OMLUSER

Recent Notebooks
Nothing to Display

How Do I?

- Use AutoML
- Get Started
- Create Notebooks
- Create Jobs
- Manage Permissions
- Try It

Quick Actions

- AutoML
- Scratchpad
- Notebooks
- Jobs
- Examples

Recent Activities
No items to display.

Click on OML Notebooks

Back in the LiveLabs Main page (where all links and passwords are), scroll down to the bottom of the page and click "Open the workshop instructions in a new tab"

The screenshot shows the Oracle LiveLabs main page. At the top, there's a navigation bar with 'LiveLabs' and a search bar. Below it, a banner for the 'Launch Oracle Machine Learning for Python on Autonomous Database Workshop' is displayed. The banner includes a 'Workshop Details' section with various login credentials and a 'Launch Console' button. A red box highlights the 'Let's Get Started - Log in to Oracle Cloud' link. Another red box highlights the 'Click here to open the next part of the workshop' link. A third red box highlights the 'Open the workshop instructions in a new tab' link at the bottom of the page.

New TAB with the Instructions

The right side of the screenshot shows the 'Get Started with OML4Py on Autonomous Database' instructions page. It has a sidebar with a table of contents for 'Lab 1: Getting Started with OML4Py'. The main content area starts with an 'Introduction' section, followed by a bulleted list of objectives. Below that is a 'Step 1: Create an OML User' section with sub-steps. Further down are sections for 'Step 2: Access Oracle Machine Learning Notebooks', 'Step 3: (Optional) Download and View the Notebook File', 'Step 4: Create an OML Notebook', 'Step 5: Connect to the Python Interpreter', 'Step 6: Verify Connection to the Autonomous Database', 'Step 7: View Help Files', 'Learn More', and 'Acknowledgements'. At the bottom, there's a note about downloading the notebook file and a 'Step 4: Create an OML Notebook' section.

Let's import the Notebooks for the Labs



Option 1: Download All Labs Notebooks in a single ZIP file from:

<http://oracle.com/a/tech/docs/otn-batch1/oml4py-hol-notebooks.zip>

Option 2: At each Lab, under the Optional Section, you can download a copy of each of the Notebooks to import them into your own. Another way is to copy and paste each paragraph on a new Notebook

The screenshot shows the 'Get Started with OML4Py on Autonomous Database' page. On the left, there's a sidebar with various lab steps. The main content area has a heading 'Get Started with OML4Py on Autonomous Database'. Below it, there's an 'Introduction' section, 'Objectives', and a list of tasks. A red box highlights the 'Step 3: (Optional) Download and View the Notebook File' section, which contains instructions and a note.

Step 3: (Optional) Download and View the Notebook File

To download the notebook version of this lab (without screenshots), click [here](#).

To view the downloaded notebook file:

1. In the Notebooks page, click **Import** and upload the notebook file.
2. After the notebook is successfully imported, click the notebook to view it.

Note: You can now follow the instructions in the notebook for performing the steps below.

Step 3: (Optional) Download and View the Notebook File

To download the notebook version of this lab (without screenshots), click [here](#).

To view the downloaded notebook file:

1. In the Notebooks page, click **Import** and upload the notebook file.
2. After the notebook is successfully imported, click the notebook to view it.

Note: You can now follow the instructions in the notebook for performing the steps below.

Import the Notebooks into your OML Notebooks Session

It should say 6 out of 6 notebooks imported successfully

The screenshot shows the Oracle Machine Learning interface with two main windows. The top window is titled 'Notebooks' and has a toolbar with 'Edit', '+ Create', 'Duplicate', 'Save as Template', 'Delete', 'Import' (which is highlighted with a red box), 'Version', and a search bar. Below the toolbar is a table header with columns: Name, Comment, Last Update, Updated By, and Connection Group. A message 'No data to display.' is shown. The bottom window shows a file explorer with a folder named 'INSYNC21' containing six JSON files: lab1_get_st..._livelabs.json, lab2_select_te_data.json, lab3_in-db.algo.json, lab4_datastores.json, lab5_embed_python.json, and lab6_automl.json. A red box highlights the 'Open' button at the bottom of this window. A yellow callout box with red text states: 'You can import multiple by selecting the 6 Notebook JSON files on your machine'. A red arrow points from the 'Import' button in the top window to the 'Open' button in the bottom window. Another red arrow points from the text in the yellow box to the 'Open' button.

You can import multiple by selecting the 6 Notebook JSON files on your machine

Name	Com...	Last Update	Updated By	Connection Group
Lab 1: Get Started with OML4Py on Autonomous Database (LiveLabs)_1		3/30/21 3:41 PM	OMLUSER	Global
Lab 5: Run user-defined functions using Embedded Python Execution		3/30/21 3:41 PM	OMLUSER	Global
Lab 2: Select and manipulate data using the Transparency Layer		3/30/21 3:41 PM	OMLUSER	Global
Lab 3: Use in-database algorithms and models		3/30/21 3:41 PM	OMLUSER	Global
Lab 4: Use Datastores to store Python objects		3/30/21 3:41 PM	OMLUSER	Global
Lab 6: Use AutoML		3/30/21 3:41 PM	OMLUSER	Global

*Work on lab 2
10 Minutes...*

Lab 3: Using in-database algorithms and models



OML4Py 1.0

Machine Learning in-database algorithms

Classification

- Decision Tree
- Naïve Bayes
- Generalized Linear Model
- Support Vector Machine
- Random Forest
- Neural Network

Regression

- Generalized Linear Model
- Neural Network
- Support Vector Machine

Clustering

- Expectation Maximization
- Hierarchical k-Means

Attribute Importance

- Minimum Description Length

Anomaly Detection

- 1 Class Support Vector Machine

Association Rules

- Apriori – Association Rules

Feature Extraction

- Singular Value Decomposition
- Explicit Semantic Analysis
- Principal Component Analysis via SVD

Supports automatic data preparation, partitioned model ensembles, integrated text mining

Scalable in-database algorithms

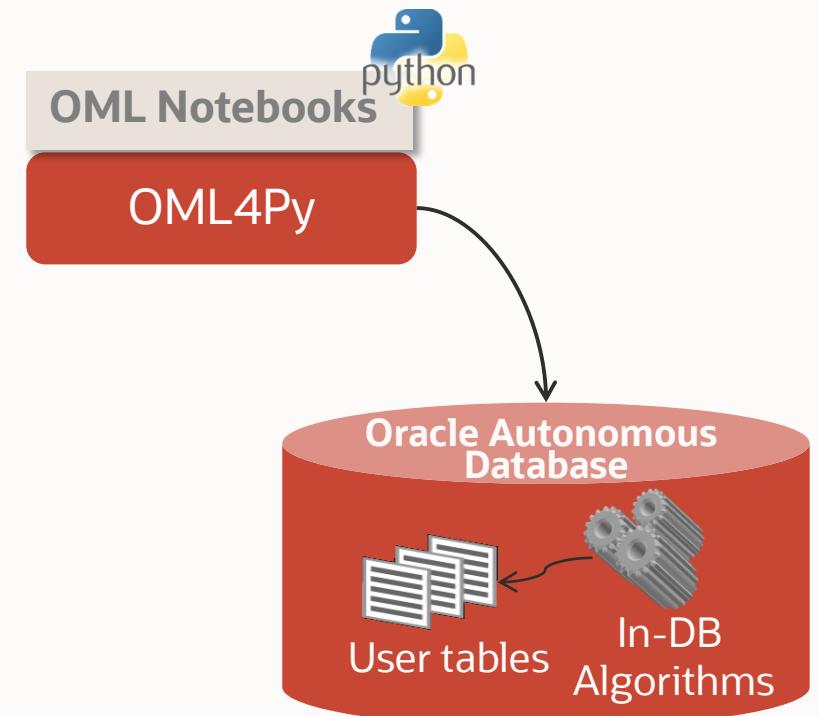
Example using Support Vector Machine for anomaly detection

```
from oml import svm

# create proxy object
ONTIME = oml.sync(table='ONTIME')

# define model object
settings = {'svms_outlier_rate' : 0.01}
svm_mod = svm('anomaly_detection',
              svms_kernel_function =
                  'dbms_data_mining.svms_linear',
              **settings)

# build anomaly detection model
svm_mod = svm_mod.fit(x=ONTIME, y=None)
# view model object
svm_mod
```



Partitioned Models

Automating a typical machine learning task

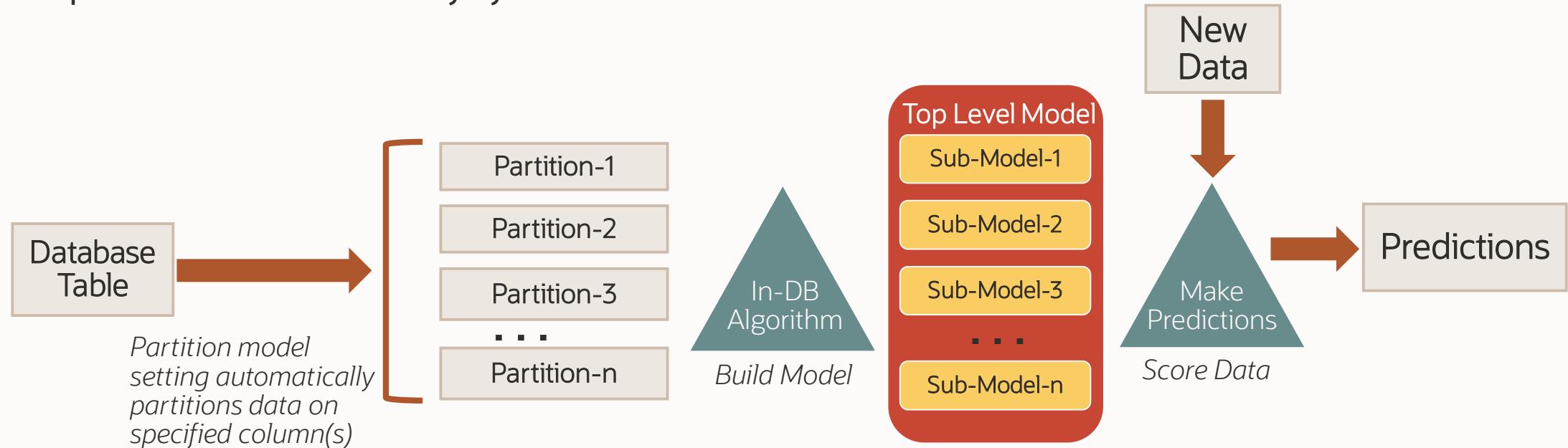


Builds ensemble model with multiple sub-models, one for each data partition

- Potentially achieve better accuracy through multiple targeted models
- Sub-models automatically managed and used as one model

Simplified scoring using top-level model only

- Proper sub-model chosen by system based on row of data to be scored



Lab 3: Using in-database algorithms and models

Goal: Learn how to build and score using in-database machine learning algorithms, as well as some of the other model-related functionality

Step 1: Import libraries

Step 2: Regression using GLM

Step 3: Clustering using KMeans

Step 4: Partitioned Models

Step 5: Rank attribute importance using Model Explainability



*Work on lab 3
10 Minutes...*

Lab 4: Use datastores to store Python objects



Datastore for Python object persistence

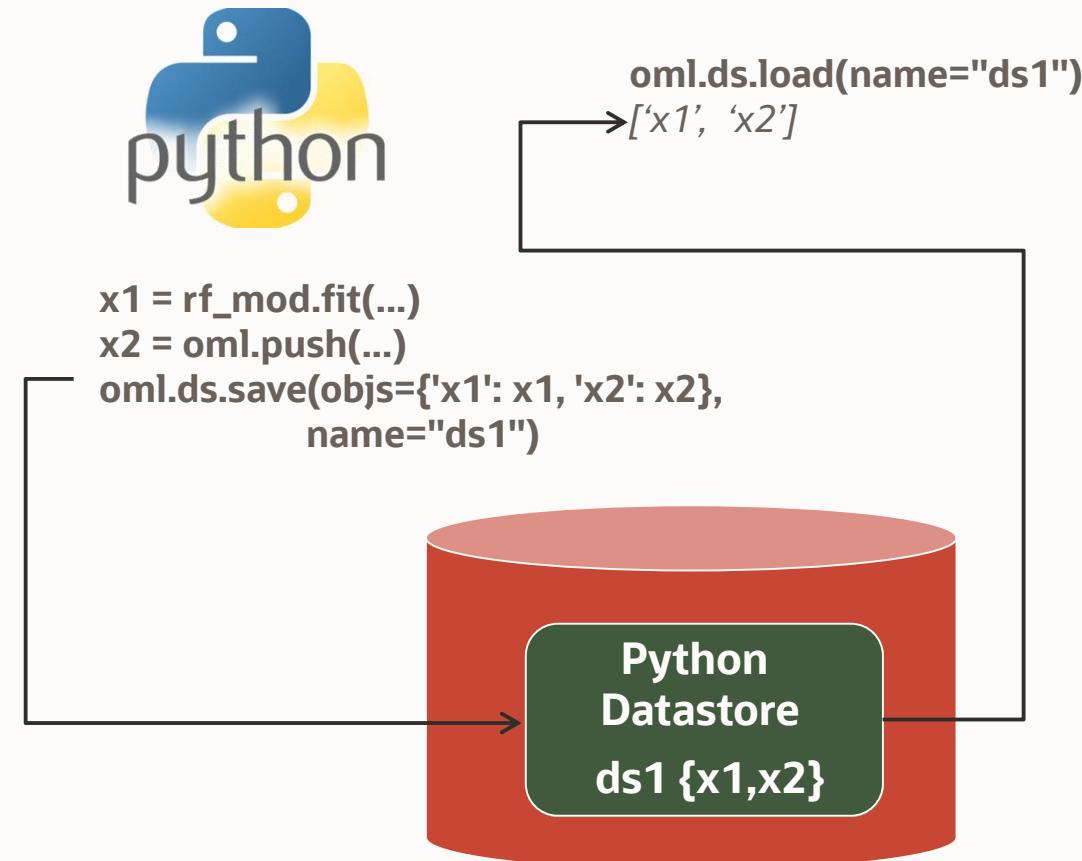


`oml.ds.save()` and `oml.ds.load()`

Provide database storage to save/restore
Python and OML4Py objects across
Python sessions

Use cases

- Preserve OML4Py objects across Python sessions
- Passing arguments to Python functions with embedded Python execution, especially when non-scalar for REST invocation, such as native Python ML models



Datastore functions for storing Python objects



oml.ds.save(objs, name[, description, . . .])

- Saves Python objects to a datastore in the user's schema

oml.ds.dir([name, regex_match, dstype])

- Lists existing datastores available to the current session user

oml.ds.describe(name[, owner])

- Describes the contents of the named datastore available to the current session user

oml.ds.load(name[, objs, owner, to_globals])

- Loads Python objects from a datastore in the user's schema

oml.ds.delete(name[, objs, regex_match])

- Deletes one or more datastores from the user's schema
- Deletes specific objects within a named datastore

oml.grant(name[, typ, user])

- Grants read privilege for a Python datastore

oml.revoke(name[, typ, user])

- Revokes read privilege for a Python datastore

Lab 4: Use datastores to store Python objects

Goal: Learn how to store and manage Python objects, both native and from OML, in the database using Datastore

Step 1: Import libraries supporting OML4Py

Step 2: Create Pandas DataFrames and load them into Autonomous Database

Step 3: Save Python objects to datastore

Step 4: Save model objects in a datastore

Step 5: Load datastore objects into memory

Step 6: View datastores and other details

Step 7: View contents of a datastore

Step 8: Manage datastore privileges

Step 9: Delete datastore content



Lab 4: Create OMLUSER2 for managing privileges to OML4Py Data Store

You will see the users ADMIN and OMLUSER, but we want to create a new one named OMLUSER2. We can give it just a random e-mail, and use the same Initial Password

The screenshot shows two windows of the Oracle Machine Learning User Administration application.

Left Window (Users List):

- Header: ORACLE® Machine Learning User Administration
- Section: Users
- Buttons: + Create, Delete, Show All Users
- Data:

User Name	Full Name	Role
ADMIN		System Administrator
OMLUSER		Developer

A red box highlights the "+ Create" button with the instruction "Click on the "+Create". A yellow box contains the following instructions:

- Fill it in with:
- OMLUSER2
- Any e-mail address
- Uncheck Generate Pass
- Paste the Initial Password twice

Right Window (Create User):

- Header: ORACLE® Machine Learning User Administration
- Title: Create User
- Buttons: Create, Cancel
- Form fields:

* Username	OMLUSER2
First Name	[empty]
Last Name	[empty]
* Email Address	any.email@anyserver.com
<input type="checkbox"/> Generate password and email account details to user. User will be required to reset the password on first sign in.	
* Password	[redacted]
* Confirm Password	[redacted]

Dotted arrows point from the yellow box on the left to the corresponding fields in the right window, indicating the mapping of the listed steps to the UI elements.

Work on lab 4
10 Minutes...

Lab 5: Run user-defined functions using Embedded Python Execution



Embedded Python Execution – features on Autonomous Database

- Database environment controls and manages spawning of Python engines
- Return results as Oracle tables and views, accessible via DataFrame proxy objects
- Return image and structured content from user-defined functions
- Use provided third-party packages in user-defined functions
- Run user-defined Python functions using data- and task-parallelism
- Store and manage user-defined Python functions in database script repository
- Invoke user-defined Python functions using REST for application integration



Embedded Python execution functions for invoking user-defined Python functions



oml.do_eval(func[, func_owner, graphics])

Runs the user-defined Python function using a Python engine spawned and controlled by the database environment

oml.table_apply(data, func[, func_owner, ...])

Runs the user-defined Python function with data pulled from a database table or view using a Python engine spawned and controlled by the database environment

oml.row_apply(data, func[, func_owner, ...])

Partitions database data into chunks of rows and runs the user-defined Python function on each chunk using Python engines spawned and controlled by the database environment

oml.group_apply(data, index, func[, ...])

Partitions database data by the column(s) specified in index and runs the user-defined Python function on each partition using Python engines spawned and controlled by the database environment

oml.index_apply(times, func[, func_owner, ...])

Runs the user-defined Python function multiple times, passing the run index as first argument, using Python engines spawned and controlled by the database environment

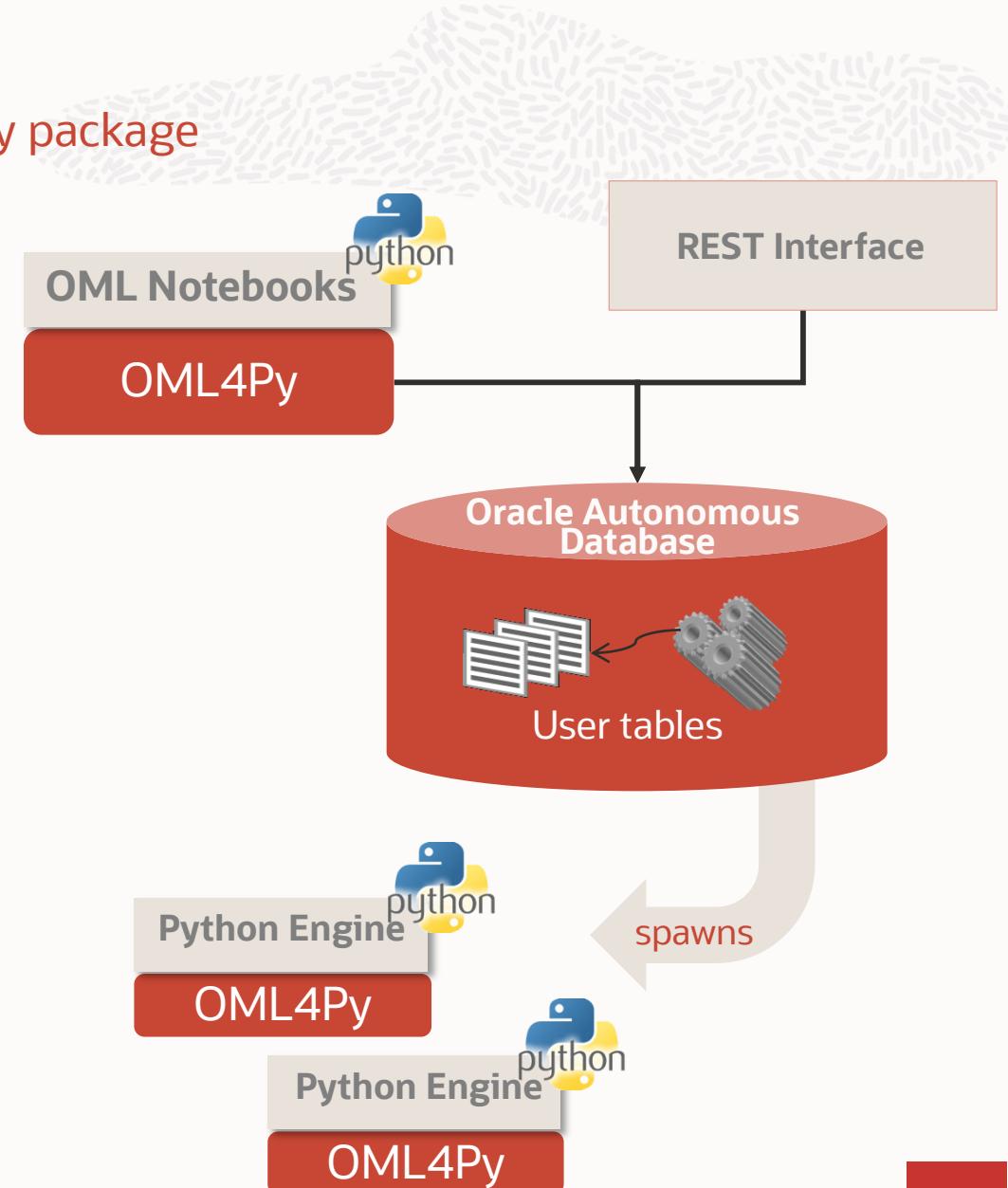
Embedded Python Execution

Example of parallel partitioned data flow using third party package

```
# user-defined function using sklearn
def build_lm(dat):
    from sklearn import linear_model
    lm = linear_model.LinearRegression()
    X = dat[['PETAL_WIDTH']]
    y = dat[['PETAL_LENGTH']]
    lm.fit(X, y)
    return lm

# select column(s) for partitioning data
index = oml.DataFrame(iris['SPECIES'])
# invoke function in parallel on IRIS table
mods = oml.groupby_parallel(iris, index,
                            func=build_lm,
                            parallel=2)

mods.pull().items()
```



OML4Py script repository



Use the script repository to...

- Create and store user-defined Python functions as scripts in Oracle Database
- Grant or revoke the read privilege to a script
- List available scripts
- Load a script function into the Python environment
- Drop a script from the script repository

Script repository functions for managing Python scripts



oml.script.create(name, func[, is_global, ...])

- Creates a Python script, which contains a single function definition, in the Oracle Database Python script repository

oml.script.dir([name, regex_match, sctype])

- Lists the scripts present in the Oracle Database Python script repository

oml.script.load(name[, owner])

- Loads the named script from the Oracle Database Python script repository as a callable object

oml.script.drop(name[, is_global, silent])

- Drops the named script from the Oracle Database Python script repository

oml.grant(name[, typ, user])

- Grants read privilege for a Python script (or datastore)

oml.revoke(name[, typ, user])

- Revokes read privilege for a Python script (or datastore)

Lab 5: Run user-defined functions using Embedded Python Execution

Goal: Learn how to run user-defined Python functions in database-controlled Python engines and work with the Python script repository

Step 1: Import the OML4Py library

Step 2: Build and score a Linear Model

Step 3: Build the model using embedded Python execution

Step 4: Build one model per Species using the group_apply function

Step 5: Invoke a function N times

Step 6: Return multiple images from embedded Python execution

Step 7: Using the Python script repository

Step 8: Create scripts in repository

Step 9: Store a function as a global function and invoke with table_apply

Step 10: Drop scripts from repository

Work on lab 5
10 Minutes...

Lab 6: Use AutoML



OML4Py AutoML objectives

Alleviating pain points

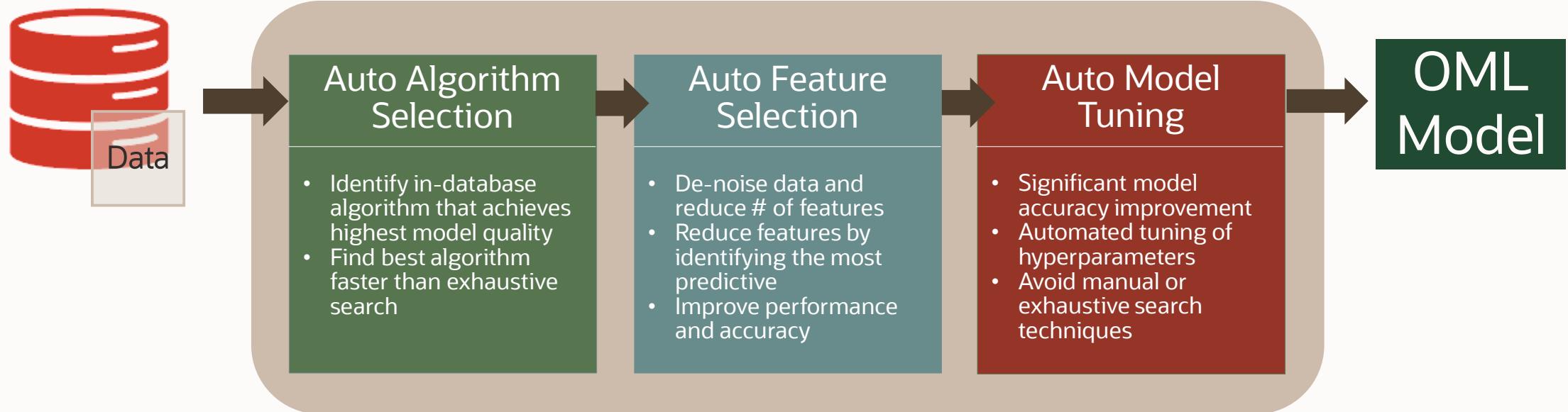
- Eliminate repetitive tasks of model building / evaluation to increase user productivity
- **Apply ML to the ML process** to reduce algorithm and hyperparameters search space and reduce compute time and cost
- Enable non-expert users to leverage machine learning





AutoML – *new* with OML4Py

Increase data scientist productivity – reduce overall compute time



Enables non-expert users to leverage Machine Learning

Lab 6: Use AutoML

Goal: Become familiar with the AutoML workflow and related functions

Step 1: Import libraries supporting OML4Py

Step 2: Automated Algorithm Selection

Step 3: Automated Feature Selection

Step 4: Automated Model Tuning

Step 5: Automated Model Selection

*Note: Some AutoML function invocations can take a few minutes to complete.
A lot of going on behind the scenes. Please be patient. ☺*

Work on lab 6
10 Minutes...

Lab 6: Bonus Round - Use OML AutoML UI



If you are not yet at the OML Notebooks screen, Sign-In again with **OMLUSER** and the Password **AAbbcc123456**. Once you are in, click on the **AutoML** link

The image shows two screenshots of the Oracle Cloud Infrastructure interface. The top screenshot is the 'SIGN IN' page, featuring a large cloud icon, the text 'Database name: ADW3900', and a 'SIGN IN' button. A yellow callout box on the left says 'Sign in with OMLUSER and the Password AAbbcc123456'. Red arrows point from this box to the 'USERNAME' field containing 'OMLUSER', the 'PASSWORD' field, and the 'Sign In' button. The bottom screenshot is the 'ORACLE Machine Learning' dashboard. It includes sections for 'How Do I?' (with links to AutoML, Get Started, Create Notebooks, Create Jobs, Manage Permissions, and Try It), 'Quick Actions' (with links to AutoML, Scratchpad, Notebooks, Jobs, and Examples), and 'Recent Activities' (which is currently empty). A red box highlights the 'AutoML' link under 'Quick Actions', and a red arrow points from the 'Click on AutoML' callout box at the bottom right towards it.

Sign in with
OMLUSER
and the Password
AAbbcc123456

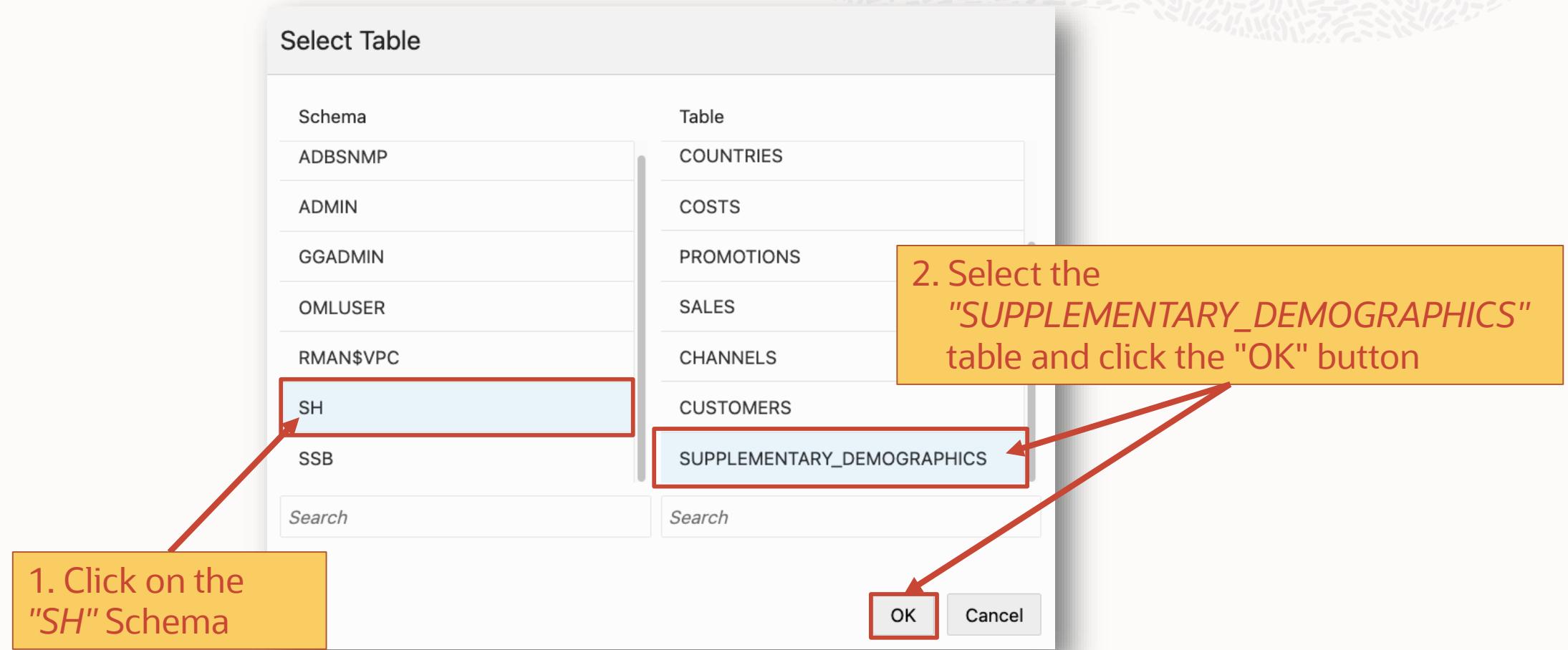
Click on AutoML

In the AutoML Experiments screen, click "Create" to create a new one

The screenshot shows the Oracle Machine Learning interface. At the top, it displays 'OML Project [OML Workspace]' and 'OMLUSER'. Below this, the 'AutoML Experiments' section is shown. A red box highlights the '+ Create' button in the toolbar. A yellow callout box with a red arrow points to this button, containing the text '1. Click on +Create'. The main table area shows 'No data to display.' and includes columns for Name, Comment, Created On, and Created By. Below the table are navigation controls for pages and items. To the right, a modal window titled 'Create Experiment' is open. It has fields for 'Name *' (containing 'First AutoML Experiment'), 'Comments', 'Data Source *' (with a search icon highlighted by a red box and arrow), and 'Prediction Type *'. A yellow callout box with a red arrow points to the 'Name' field, containing the text '2. Give the Experiment a name'. Another yellow callout box with a red arrow points to the search icon in the 'Data Source' field, containing the text '3. Click on the Loupe to search for a Data Source'.

In the menu that appears, select:

Schema = *SH* & Table = *SUPPLEMENTARY_DEMOGRAPHICS*



In the Create Experiments screen, enter the *Predict*, *Case ID*, and adjust performance settings

The screenshot shows the 'Create Experiment' dialog box from Oracle Machine Learning. The 'Predict' field is set to 'AFFINITY_CARD' and the 'Case ID' field is set to 'CUST_ID'. In the 'Additional Settings' section, the 'Maximum Top Models' is set to 2, 'Maximum Run Duration (Hours)' is set to 8, 'Database Service Level' is set to 'Medium', and 'Model Metric' is set to 'Balanced Accuracy'.

1. For the "Predict" attribute, select AFFINITY_CARD from the pull-down menu
2. For the "Case ID" attribute, select CUST_ID from the pull-down menu
3. Expand the "Additional Settings" and reduce the Maximum Top Models to "2" to save time in this exercise.
4. Change the "Database Service Level" to Medium to get increased parallelism

We are ready to start the Experiment

Click on the Start -> Faster Results

The screenshot shows the Oracle Machine Learning interface for creating an experiment. The main window is titled "Create Experiment". It includes fields for "Name *" (set to "First AutoML Experiment"), "Comments", "Data Source *" (set to "SH.SUPPLEMENTARY_DEMOGRAPHICS"), "Prediction Type *" (set to "Classification"), and "Additional Settings" which include "Maximum Top Models *" (set to 2), "Maximum Run Duration (Hours) *" (set to 8), and "Database Service Level" (set to "Medium"). On the right, there's a "Session" tab with "Predict *" set to "AFFINITY_CARD" and "Case ID" set to "CUST_ID". A modal window titled "OMLUSER Project [OMLUSER Works...]" is open at the top right, showing a "Start" button with a dropdown menu. The menu has two options: "Faster Results" and "Better Accuracy", with "Faster Results" being highlighted. Three yellow callout boxes with red arrows point from the text instructions to the "Start" button, the "Faster Results" option in the dropdown, and the progress bar in the background modal respectively.

1. Click on Start
2. Click on Faster Results to make sure we have a quick result for this Session
3. Experiment will start

Click on the three dots (...) to see the Progress

The screenshot shows the Oracle Machine Learning interface. At the top, there's a navigation bar with the Oracle logo, the text "Machine Learning", and a dropdown for "OMLUSER Project [OMLUSER Works...]" and "OMLUSER". Below the navigation, the main content area is titled "First AutoML Experiment" and displays a chart titled "Balanced Accuracy" with a single data point at 0.500. Below the chart is a "Leader Board" table.

A yellow callout box with a red arrow points to the three-dot menu icon on the right side of the "Running" status bar. This menu is expanded to show a "Progress" report window.

The "Progress" window lists the following tasks:

Task	Status
Algorithm Selection	Completed
Adaptive Sampling	Completed
Feature Selection	Running
Model Tuning	Queued
Naive Bayes	Queued
Generalized Linear Model	Queued
Feature Prediction Impact	Queued

Completed vs Running Steps

The screenshot shows the Oracle Machine Learning interface. At the top, it says "OMLUSER Project [OMLUSER Works...]" and "OMLUSER". On the left, there's a sidebar with "Experiment Settings" and "Edit" options. Below that is a chart titled "Balanced Accuracy" with a single data point at 0.500. Further down is a "Leader Board" section with "Deploy", "Create Notebook", and "Metrics" buttons, and a table showing two entries: "Naive Bayes" and "Generalized Linear Model". A central modal window is titled "Progress" and lists several steps: "Algorithm Selection" (Completed), "Adaptive Sampling" (Completed), "Feature Selection" (Running, indicated by a progress bar and a circular loading icon), "Model Tuning" (Queued), "Naive Bayes" (Queued), "Generalized Linear Model" (Queued), and "Feature Prediction Impact" (Queued). A red arrow points from a callout box to the "Feature Selection" step.

Checkboxes are shown for completed steps, while running steps show an animation

Top algorithms selected by AutoML

The screenshot shows the Oracle Machine Learning interface for a project named "OMLUSER Project [OMLUSER Works...]".

Leader Board: A chart titled "Balanced Accuracy" showing accuracy values from 0.496 to 0.504. Below the chart is a table:

Algorithm	Model Name	Balanced Accuracy
Naive Bayes	nb_5894c519c9	0.502
Generalized Linear Model	glm_15baeb3f99	0.498

A red box highlights the "Algorithm" column, and a red arrow points from it to a yellow callout box containing the text: "Only the Top algorithms identified in the \"Algorithm Selection\" phase are shown in the Leader Board".

Progress Dialog: A modal window titled "Progress" lists the following tasks:

- Algorithm Selection: Completed (checkmark)
- Adaptive Sampling: Completed (checkmark)
- Feature Selection: Running (loading icon)
- Model Tuning: Queued
- Naive Bayes: Queued
- Generalized Linear Model: Queued
- Feature Prediction Impact: Queued

At the top of the interface, there are navigation links: "Experiment Settings" and "Edit", and a status bar showing "Running" with a "Stop" button.

A completed Experiment looks like this

The screenshot shows the Oracle Machine Learning interface with a completed experiment. The top navigation bar includes the Oracle logo, 'Machine Learning', 'OMLUSER Project [OMLUSER Works...]', and 'OMLUSER'. A red callout box on the left says: 'The Experiment finishes when you see the "Completed" message at the top' with an arrow pointing to the 'Completed' status indicator. Another red callout box on the right says: 'Checkboxes are shown for all completed steps' with an arrow pointing to the 'Run Summary' section. The main content area displays the 'First AutoML Experiment' with a 'Balanced Accuracy' chart and a 'Leader Board' table.

ORACLE® Machine Learning

OMLUSER Project [OMLUSER Works...]

OMLUSER

<- Experiments

First AutoML Experiment

Experiment Settings Edit

Balanced Accuracy

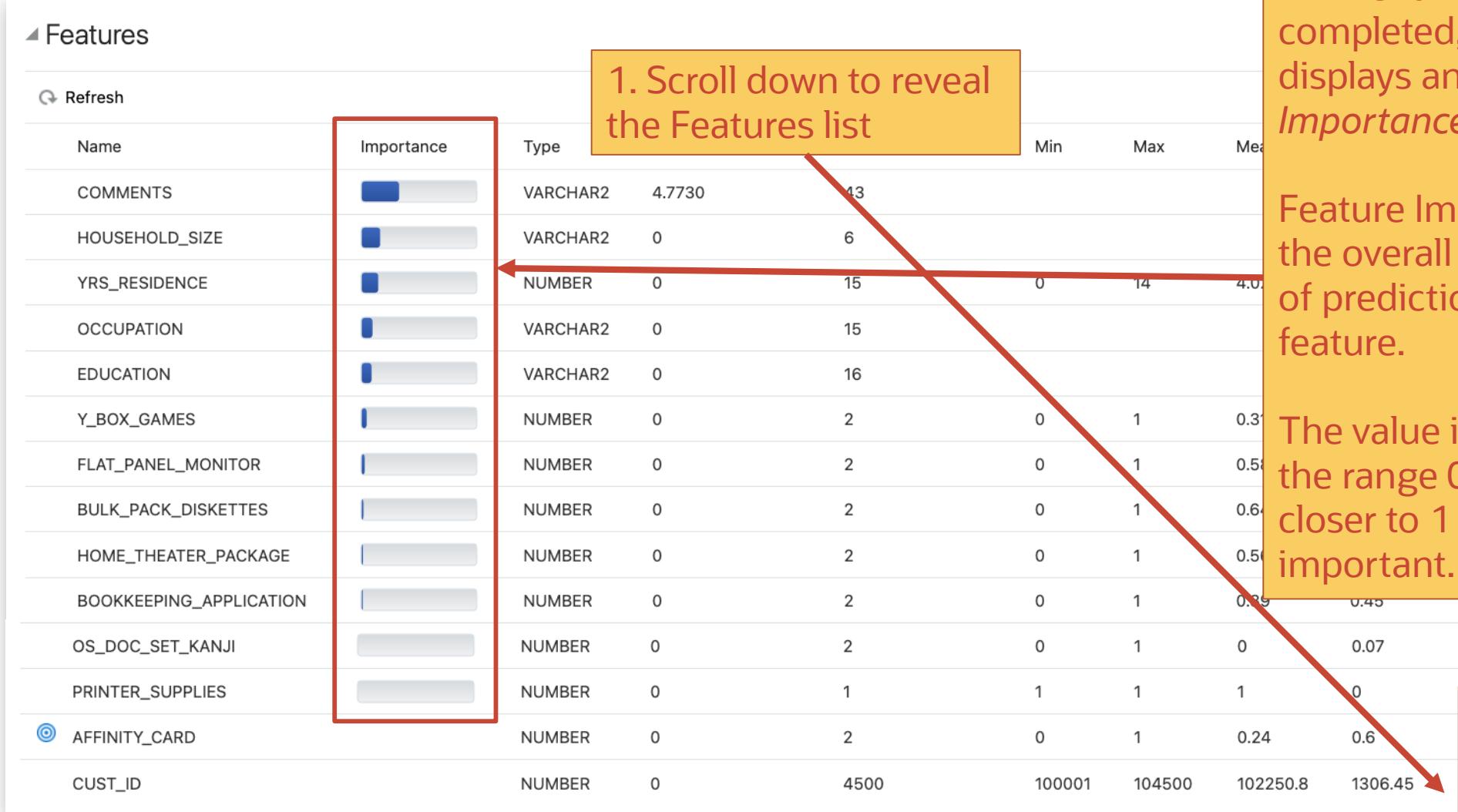
Leader Board

Deploy	Create Notebook	Metrics
Algorithm	Model Name	Balanced Accuracy
Naive Bayes	nb_5894c519c9	0.8868
Generalized Linear Model	glm_15baeb3f99	0.8697

Run Summary

Algorithm Selection	Completed
Adaptive Sampling	Completed
Feature Selection	Completed
Model Tuning	Completed
Naive Bayes	Completed
Generalized Linear Model	Completed
Feature Prediction Impact	Completed

Let's check the global Feature Importance



Let's open the Model Detail to inspect the Model

The screenshot shows the Oracle Machine Learning interface. At the top, it says "OMLUSER Project [OMLUSER Works...]" and "OMLUSER". Below that, it says "First AutoML Experiment". On the left, there's a "Balanced Accuracy" chart and a "Leader Board" table. A red callout box with the text "Click on the Naïve Bayes model name in blue" points to the "nb_5894c519c9" entry in the Leader Board. To the right, a modal window titled "Run Summary" lists completed steps: Algorithm Selection, Adaptive Sampling, Feature Selection, Model Tuning, Naive Bayes, Generalized Linear Model, and Feature Prediction Impact.

ORACLE® Machine Learning

OMLUSER Project [OMLUSER Works...]

OMLUSER

<- Experiments

First AutoML Experiment

Experiment Settings Edit

Balanced Accuracy

Leader Board

Algorithm	Model Name	Balanced Accuracy
Naive Bayes	nb_5894c519c9	0.8868
Generalized Linear Model	glm_15baeb3f99	0.8697

Metrics

Deploy Create Notebook Metrics

Click on the Naïve Bayes model name in blue

Run Summary

- Algorithm Selection Completed
- Adaptive Sampling Completed
- Feature Selection Completed
- Model Tuning Completed
- Naive Bayes Completed
- Generalized Linear Model Completed
- Feature Prediction Impact Completed

Review the Model Detail – Prediction Impacts

The screenshot shows the Oracle Machine Learning interface. In the center, a modal window titled "Model Detail - nb_5894c519c9" is displayed. The window has two tabs: "Prediction Impacts" (which is selected) and "Confusion Matrix". The "Prediction Impacts" tab contains a table with columns "Name" and "Prediction Impact". The table lists several attributes with their corresponding impact values represented by horizontal bars. A red box highlights this table. To the right of the modal, a large orange callout box contains two pieces of text: "A new window with the Model Details show the Prediction Impacts of the Attributes." and "It uses OML's Machine Learning Explainability module to provide model-agnostic functionality to identify the important features that impact a trained model's predictions." An arrow points from the text in the callout box to the "Prediction Impacts" table in the modal.

Name	Prediction Impact
COMMENTS	0.70
HOUSEHOLD_SIZE	0.55
OCCUPATION	0.45
YRS_RESIDENCE	0.40
Y_BOX_GAMES	0.40
HOME_THEATER_PACKAGE	0.35
EDUCATION	0.30
BOOKKEEPING_APPLICATION	0.30
BULK_PACK_DISKETTES	0.30

A new window with the Model Details show the Prediction Impacts of the Attributes.

It uses OML's Machine Learning Explainability module to provide model-agnostic functionality to identify the important features that impact a trained model's predictions.

Review the Model Detail – Confusion Matrix

The screenshot shows the Oracle Machine Learning interface. At the top, it says "OMLUSER Project [OMLUSER Works...]" and "OMLUSER". Below that, there's a "Completed" status indicator and a "Start" button. On the left, there's a link "[-> Experiments](#)". The main title is "First AutoML Experiment". On the left, there are links for "Experiment Settings" and "Balanced Accuracy". A graph for "Balanced Accuracy" is shown, starting at 0.50 and rising to approximately 0.85. Below the graph is a "Leader Board" table. A red box highlights the "Model Detail - nb_5894c519c9" window. This window has tabs for "Prediction Impacts" and "Confusion Matrix", with "Confusion Matrix" being the active tab. It displays the following confusion matrix data:

	Predicted: 0	Predicted: 1
Actual: 0	598	63
Actual: 1	32	212

The Confusion Matrix shows an evaluation of the Model on the Validation Data selected by AutoML at the end of the Process

What have we accomplished...

Lab 1: Getting started with OML4Py

Lab 2: Select and manipulate data using the Transparency Layer

Lab 3: Using in-database algorithms and models

Lab 4: Use datastores to store Python objects

Lab 5: Run user-defined functions using Embedded Python Execution

Lab 6: Use AutoML

Lab 6: Bonus Round - Use OML AutoML UI

Where to go from here?



Helpful Links

ORACLE MACHINE LEARNING ON O.COM

<https://www.oracle.com/machine-learning>



OML TUTORIALS

OML LiveLab: https://apexapps.oracle.com/pls/apex/dbpm/r/livelabs/view-workshop?p180_id=560

OML4Py LiveLab: <https://apexapps.oracle.com/pls/apex/dbpm/r/livelabs/view-workshop?wid=786>

Interactive tour: <https://docs.oracle.com/en/cloud/paas/autonomous-database/oml-tour>



OML OFFICE HOURS

<https://asktom.oracle.com/pls/apex/asktom.search?office=6801#sessions>



ORACLE ANALYTICS CLOUD

<https://www.oracle.com/solutions/business-analytics/data-visualization/examples.html>



OML4PY

[OML4Py \(2m video\)](#)

[OML4Py Introduction \(17m video\)](#)

[OML4Py Technical Brief](#)

[OML4Py User's Guide](#)

[Blog: Introducing OML4Py](#)

[GitHub Repository with Python notebooks](#)

ORACLE AUTOML UI

[Oracle Machine Learning AutoML UI \(2m video\)](#)

[Oracle Machine Learning Demonstration \(6m video\)](#)

[OML AutoML UI Technical Brief](#)

[Blog: Introducing Oracle Machine Learning AutoML UI](#)

OML SERVICES

[Oracle Machine Learning Services \(2m video\)](#)

[OML Services Technical Brief](#)

[Oracle Machine Learning Services Documentation](#)

[Blog: Introducing Oracle Machine Learning Services](#)

[GitHub Repository with OML Services examples](#)

Oracle Machine Learning AskTom Office Hours sessions

<https://asktom.oracle.com/pls/apex/asktom.search?office=6801#sessions>



Sessions every Tuesday at 8AM Pacific Time

June 15 2021	OML usage highlight: Machine Learning Recommendations for Maintenance and Repair	
June 8 2021	Weekly Office Hours: OML on Autonomous Database - Ask & Learn	
June 1 2021	Hands-On Lab using Oracle Machine Learning for Python on Autonomous Database	
May 18 2021	Hands-On Lab using Oracle Machine Learning Services on Autonomous Database	
May 11 2021	OML usage highlight: Oracle Process Automation with Real-time OML Services scoring	

Machine Learning 101 sessions

Previous Sessions

April 20 2021	OML usage highlight: Oracle Stream Analytics with Real-time OML Services scoring	
April 13 2021	OML usage highlight: Making Oracle Digital Assistant smarter with OML Services	
March 30 2021	OML feature highlight: OML AutoML UI for Automated Model Building	
March 23 2021	Weekly Office Hours: OML on Autonomous Database - Ask & Learn	
March 11 2021	OML feature highlight: OML Services on Autonomous for Model Deployment	
March 2 2021	Weekly Office Hours: OML on Autonomous Database - Ask & Learn	
February 23 2021	Hands-On Lab using Oracle Machine Learning for Python on Autonomous Database	
February 18 2021	Machine Learning 102 - Feature Extraction	
December 17 2020	Machine Learning 101: Feature Extraction	
November 5 2020	Machine Learning 102: Clustering	
October 28 2020	Oracle Machine Learning for R: An Introduction	
September 29 2020	Machine Learning 101: Clustering	
August 4 2020	Machine Learning 102: Regression	
July 21 2020	Machine Learning 101: Regression	
June 9 2020	Machine Learning 102: Classification	
May 19 2020	Machine Learning 101: Classification	
March 11 2020	Oracle Machine Learning Notebooks - Deep Dive	
February 18 2020	Oracle Machine Learning for Spark	
January 8 2020	Oracle Machine Learning for Python, with Demos	
December 11 2019	Oracle's Comprehensive Machine Learning Platform	

Thank you !



ORACLE

