ORACLE

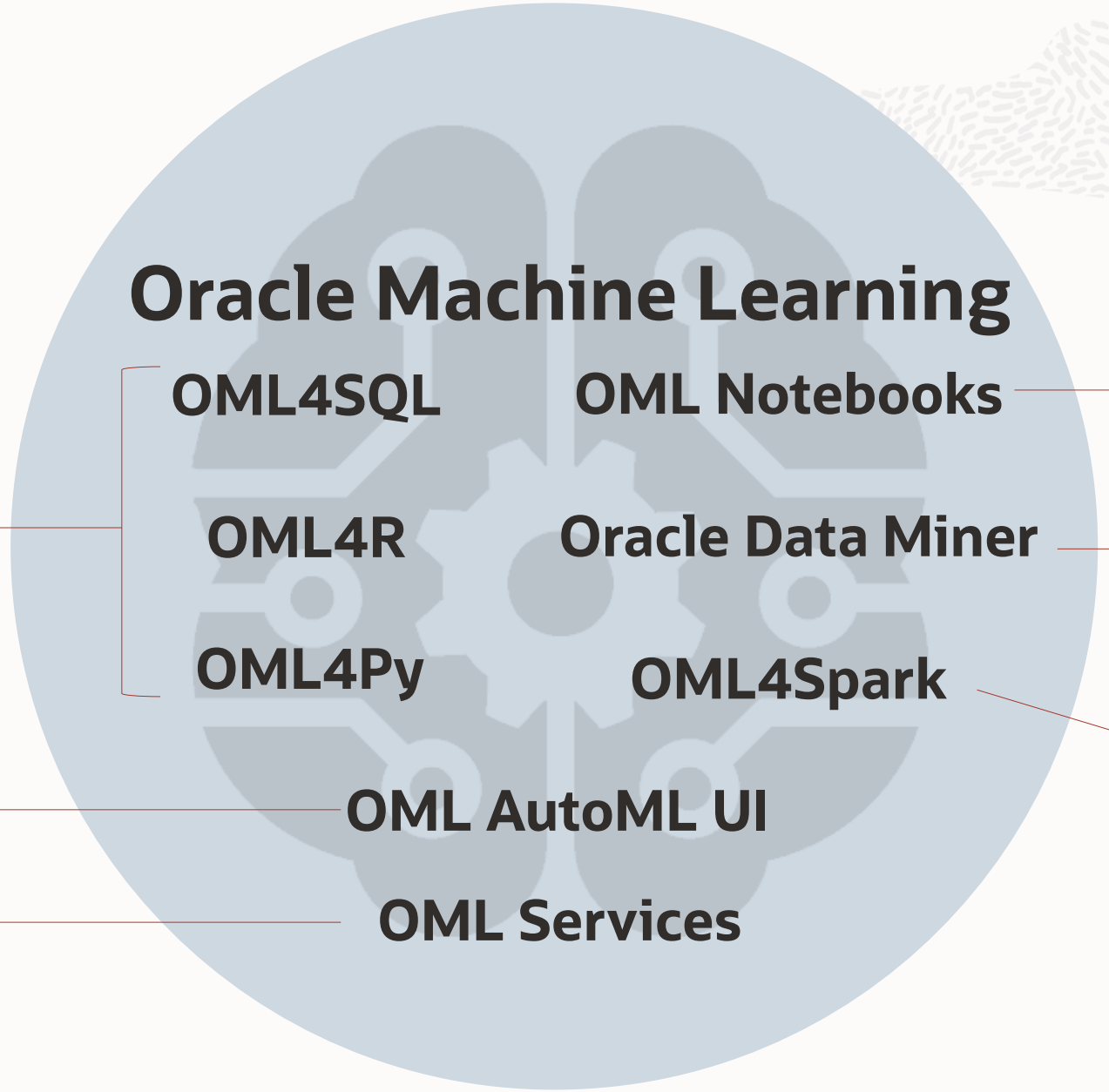
Python Machine Learning in Oracle Autonomous ADB

Natwest Developer Session – HOL Details

July 2021

Introduction

Oracle Machine Learning



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

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

Code-free AutoML interface on Autonomous Database

Model Deployment and Management, Cognitive Text

Oracle Machine Learning Notebooks

Autonomous Database as a Data Science Platform

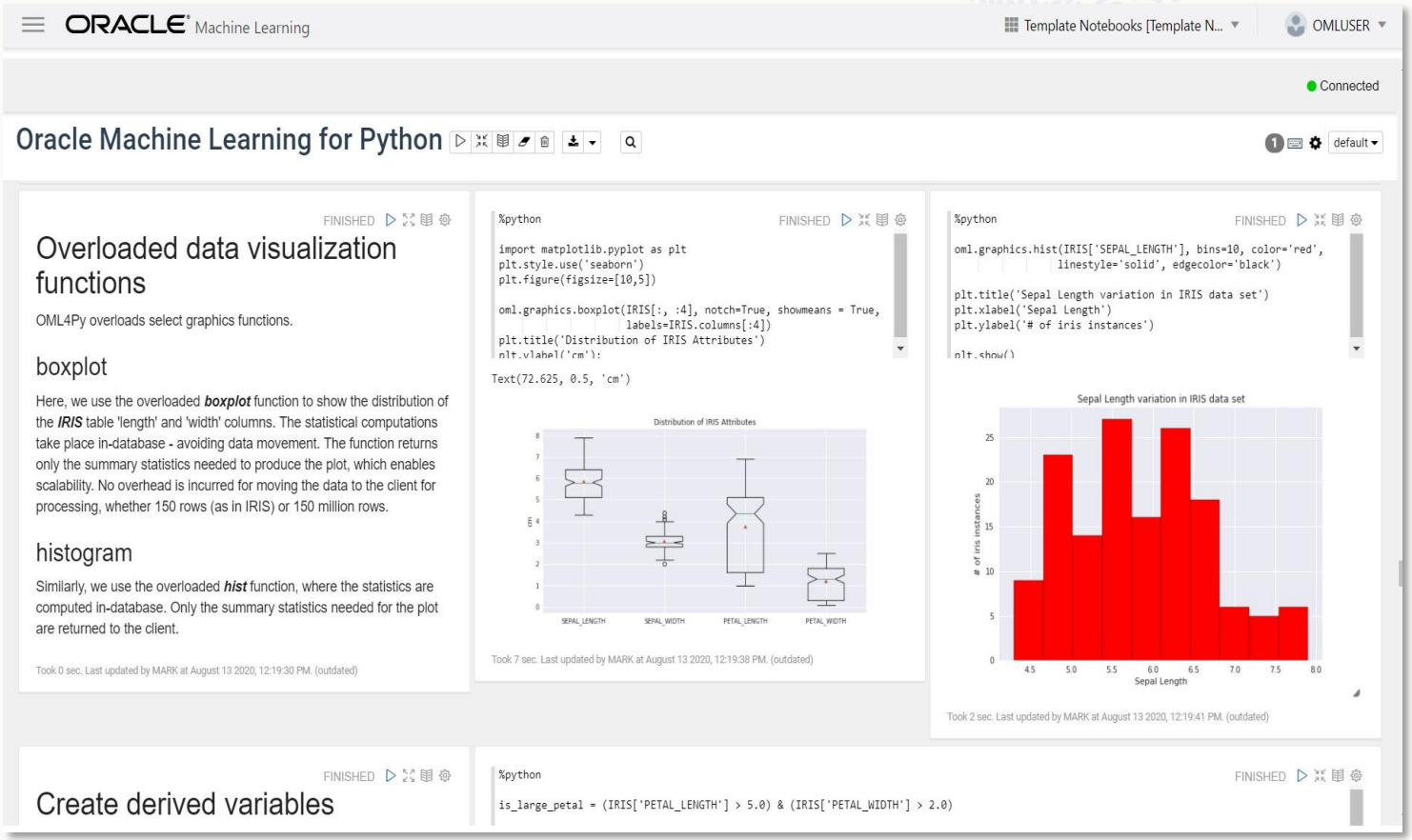
OML Notebooks

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



Oracle Machine Learning for Python

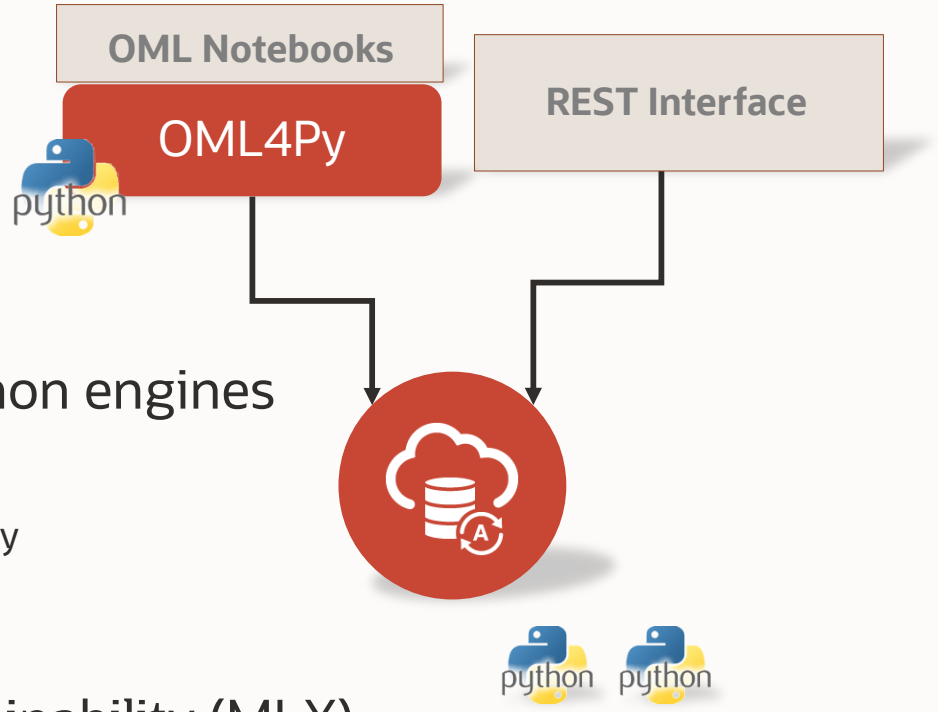
Supported in Oracle Autonomous Database with OML Notebooks

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

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

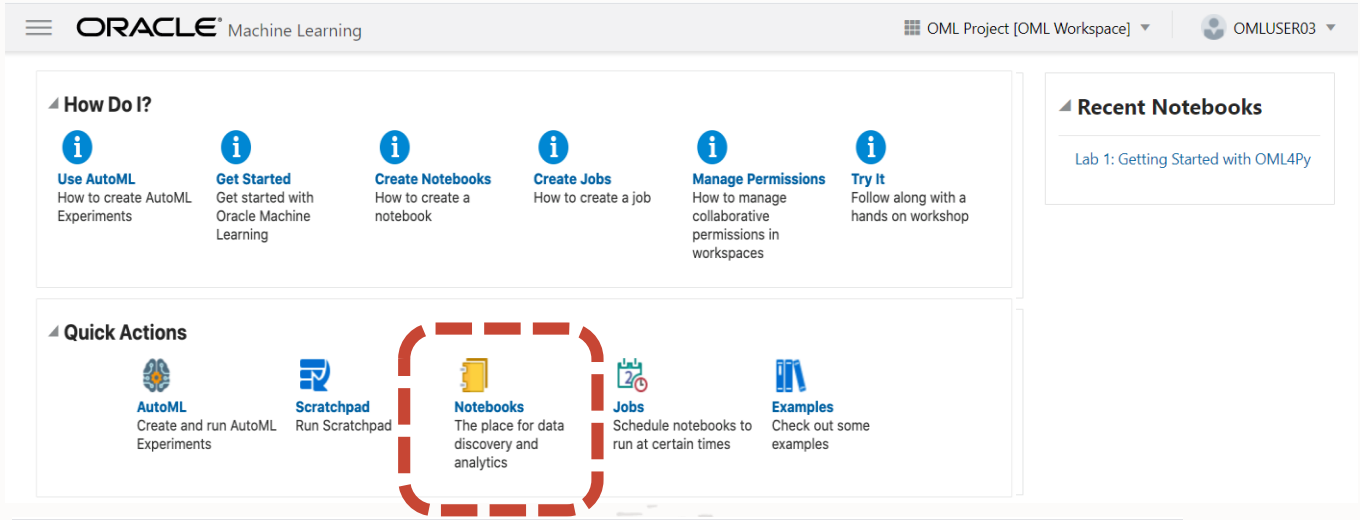
Lab 1: Getting started with OML4Py

Lab 1 Getting Started with OML4Py

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

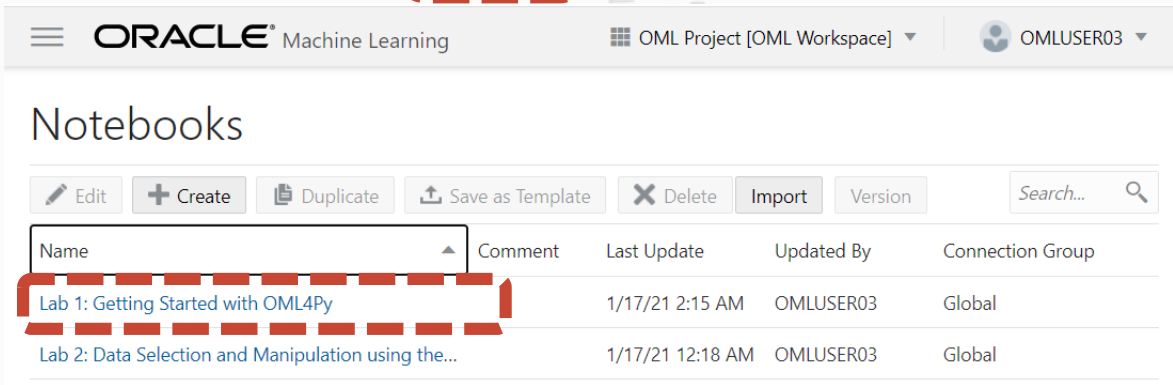
Step 2: Log in using provided username and password

Step 3: Click “Notebooks”

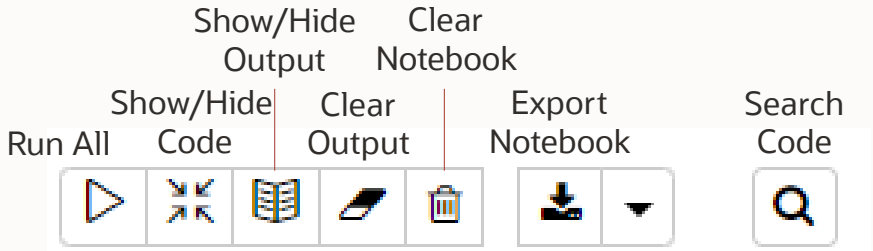


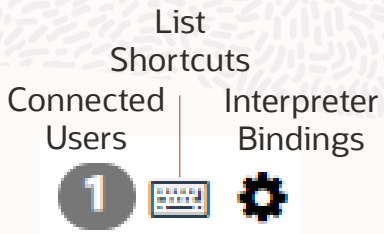
Step 4: Click the notebook for

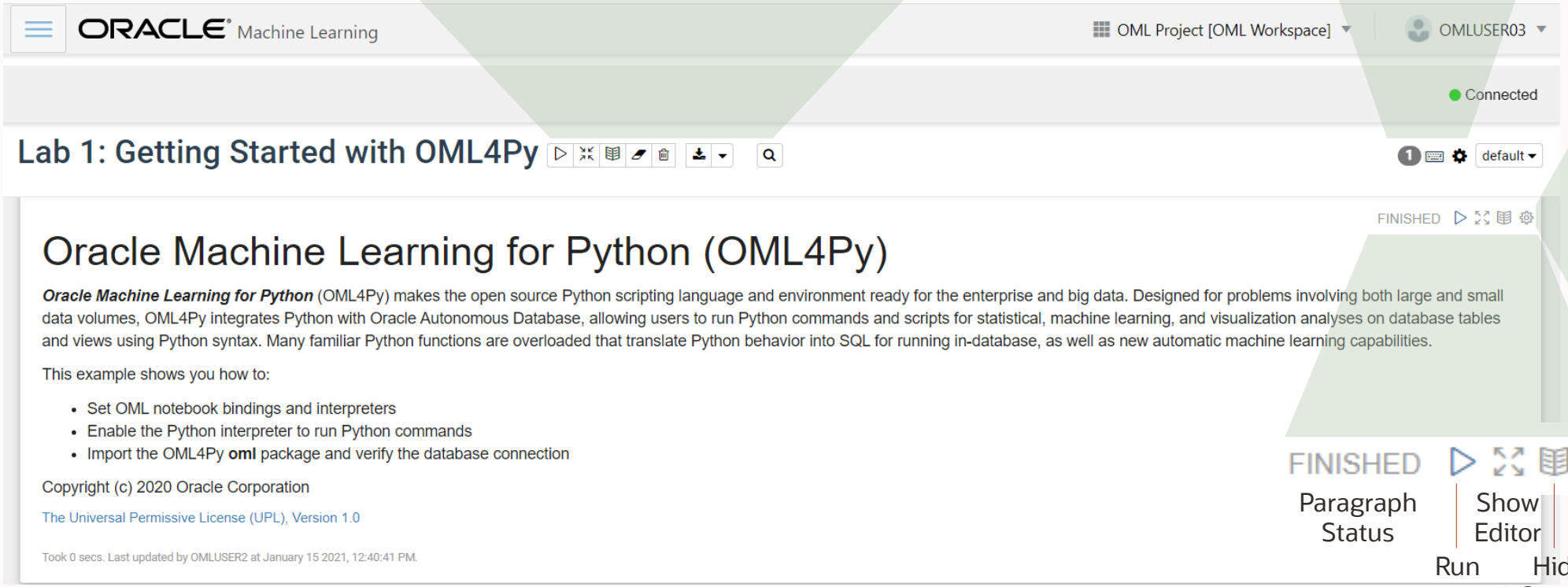
"Lab 1: Getting Started with OML4Py"

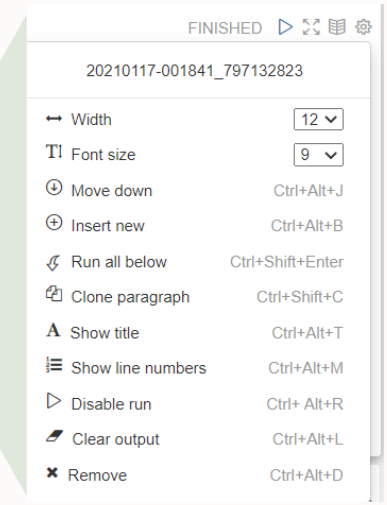


Lab 1: Getting Started with OML4Py









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])

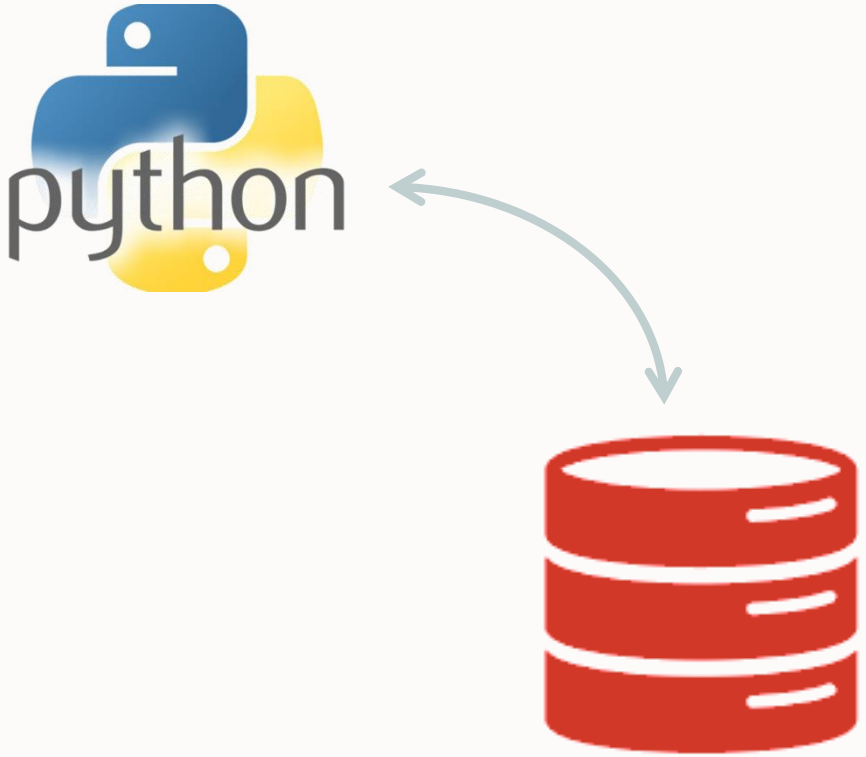
- Drops the named database table or view

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

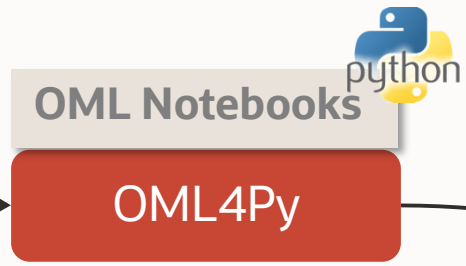
Example using the crosstab function

```
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

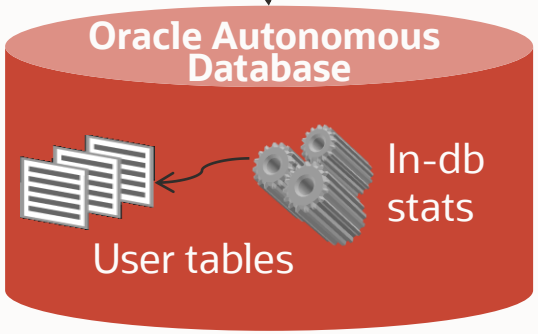
crosstab() function overloaded to accept OML DataFrame objects and transparently generates SQL for scalable processing in Oracle Database

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




```
>>> 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

Step1: 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

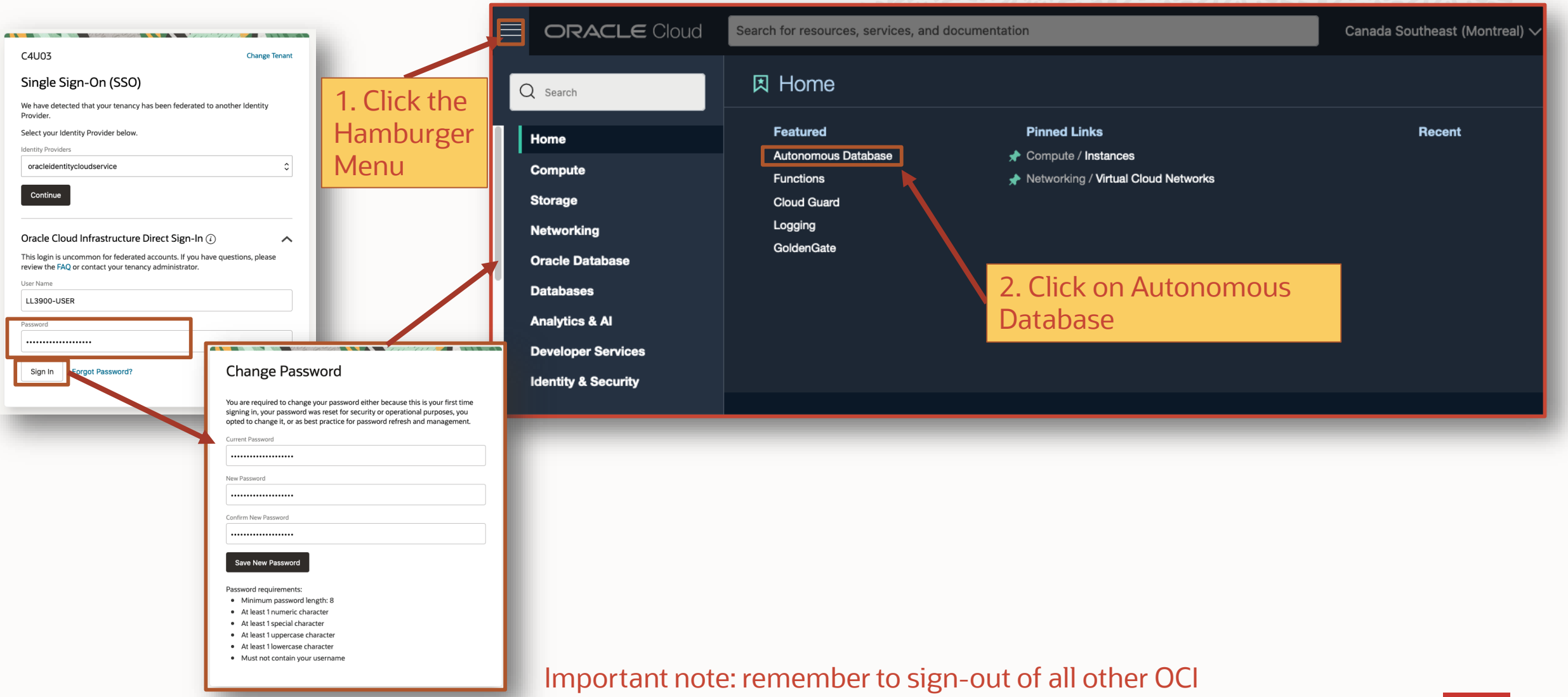
Step 8: Create a persistent database table

Quick check on the environment readiness

Contact us via **"Chat to the Panelists"** if your environment is not ready yet, of if you have problems

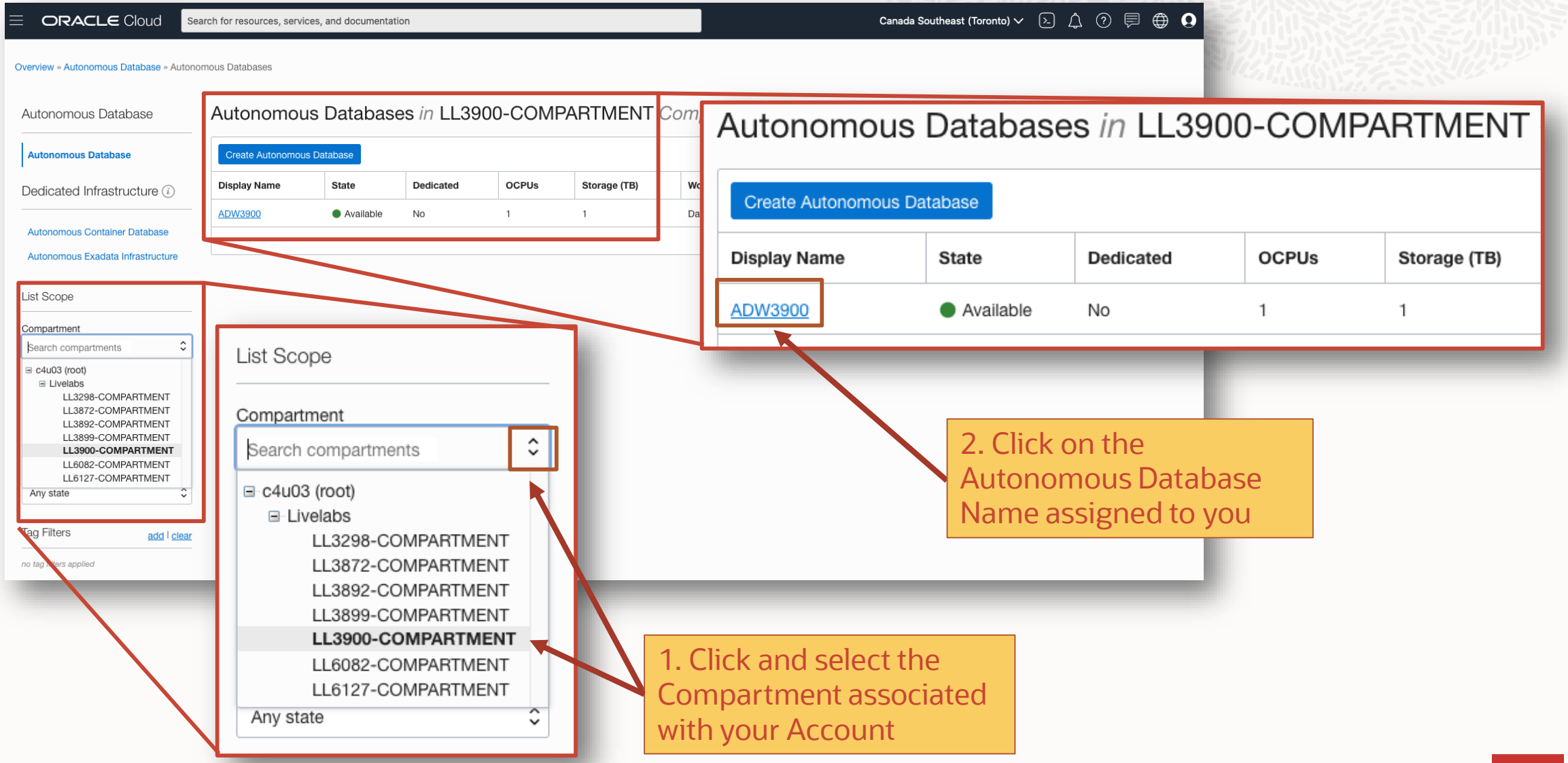
Have you received the email indicating that your LiveLabs environment is ready to use?

In the Single Sign-On screen, enter the Password given in the Workshop screen for your user LL####-USER. You will need to change the Password (you can use the same), then click on the Hamburger Menu on the Top left of OCI, then Autonomous Database.

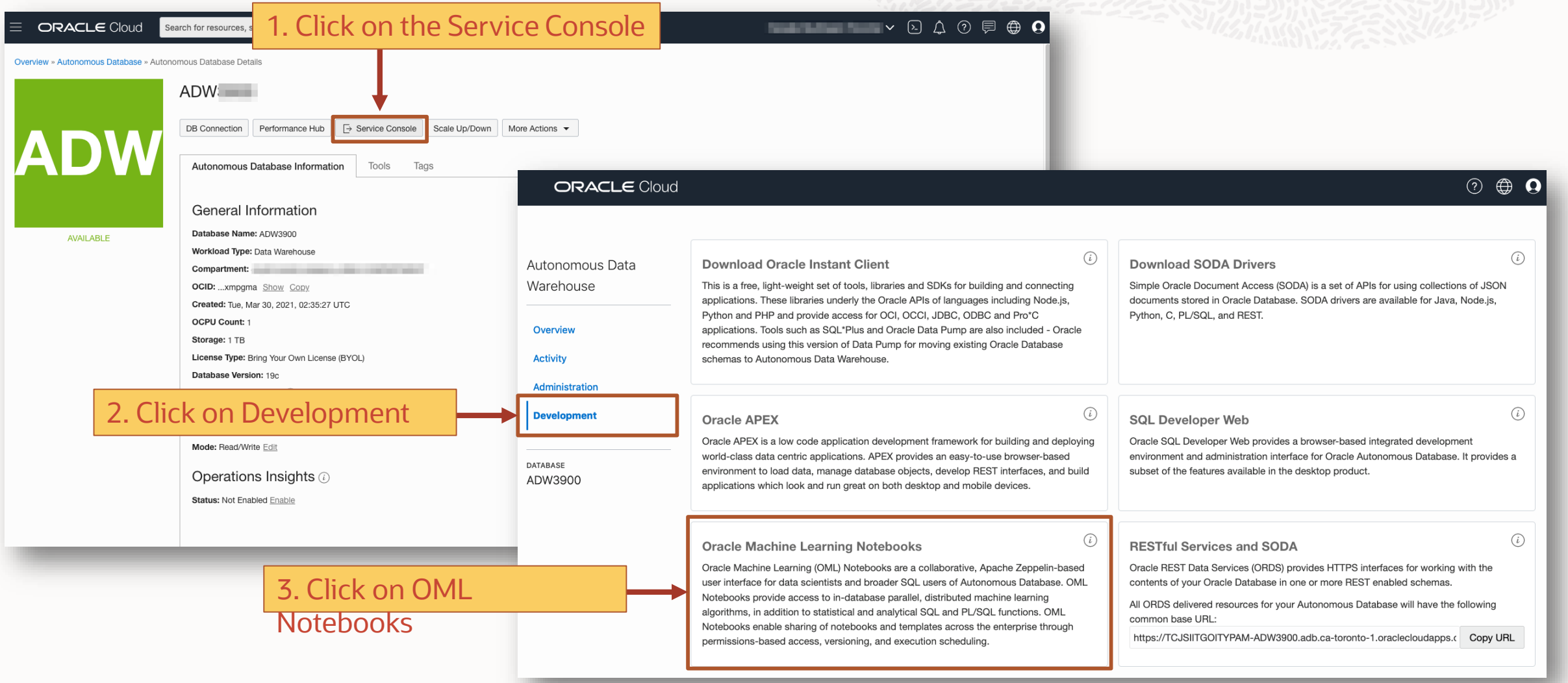


Important note: remember to sign-out of all other OCI Account Tabs you might have open

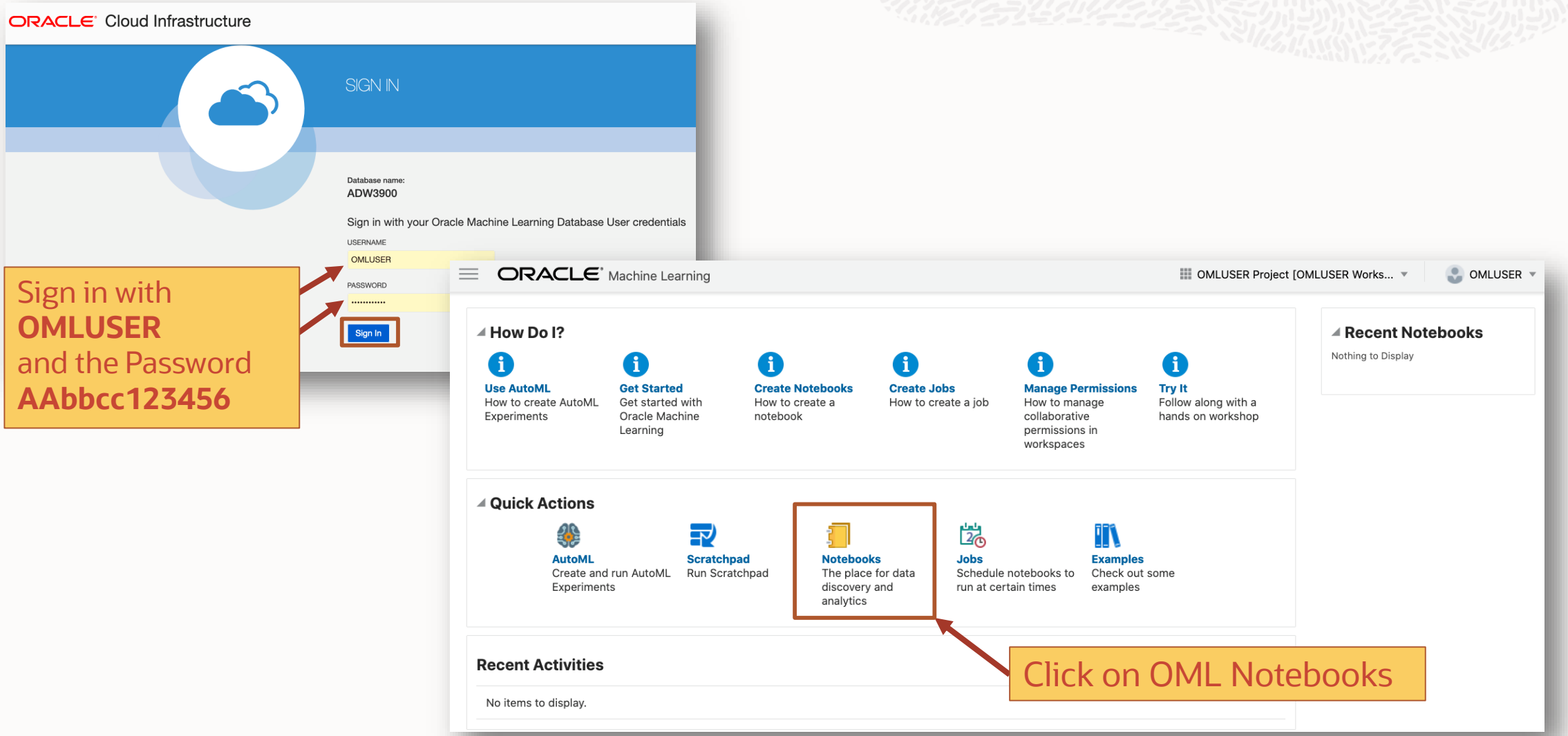
Select the specific ADW Compartment indicated in the Workshop Main screen, and then click on the Autonomous Database



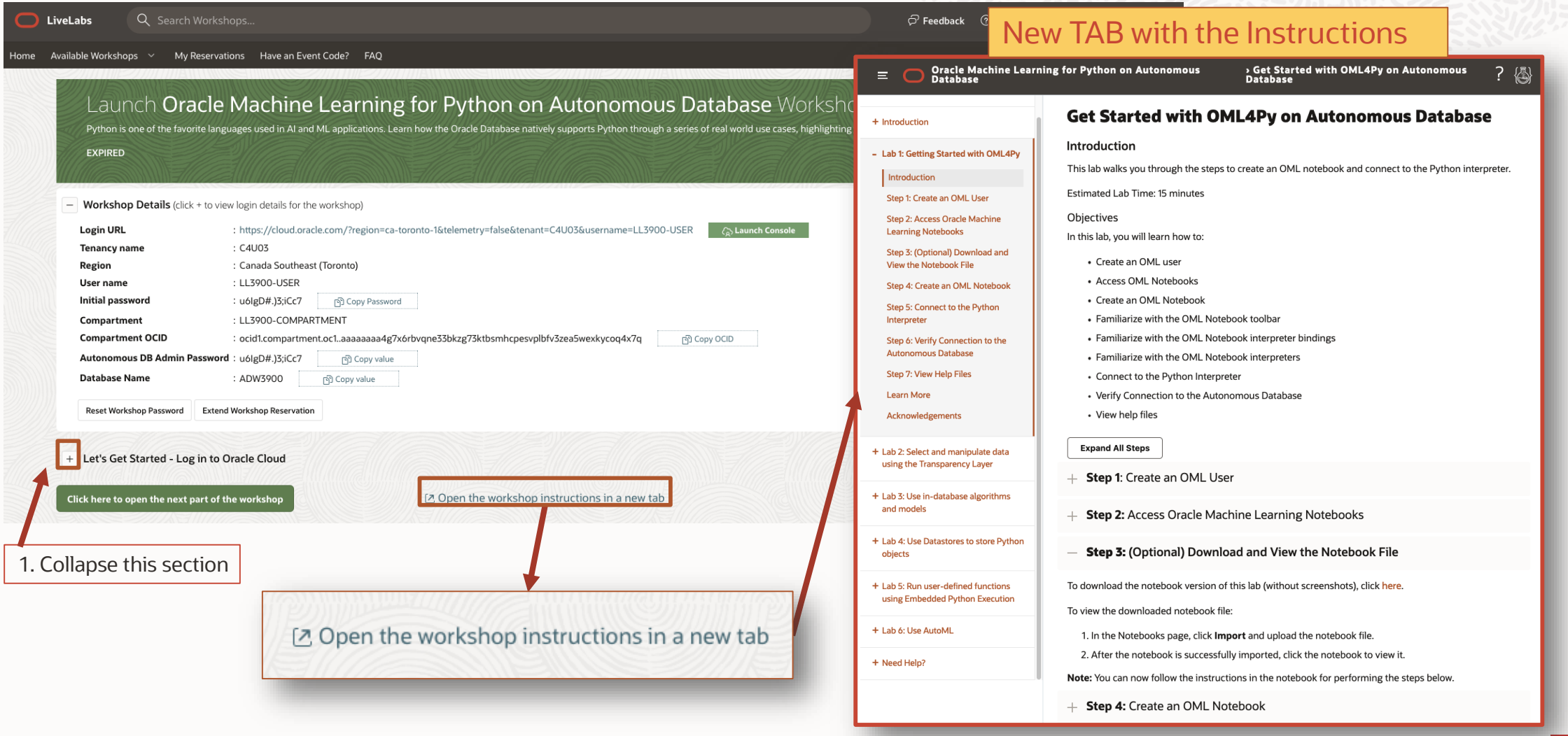
Once inside the Autonomous Database, click on "Service Console". In there, select "Development" and then click on "Oracle Machine Learning Notebooks"



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



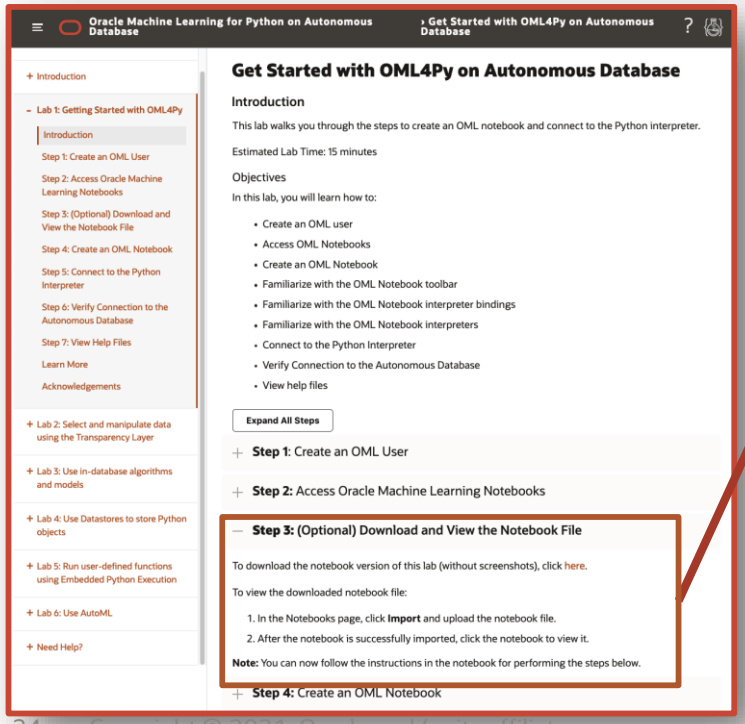
Back in the LiveLabs Main page (where all links and passwords are), scroll down to the bottom of the page and click "Open the workshop instructions in a new tab"

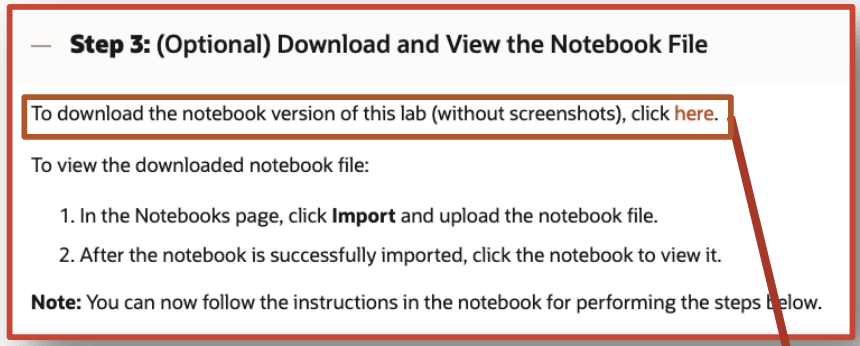


Let's import the Notebooks for the Labs

Option 1: Download All Labs Notebooks in a single ZIP file from:
http://oracle.com/a/tech/docs/otn-batch1/oml4py-hol-notebooks.zip

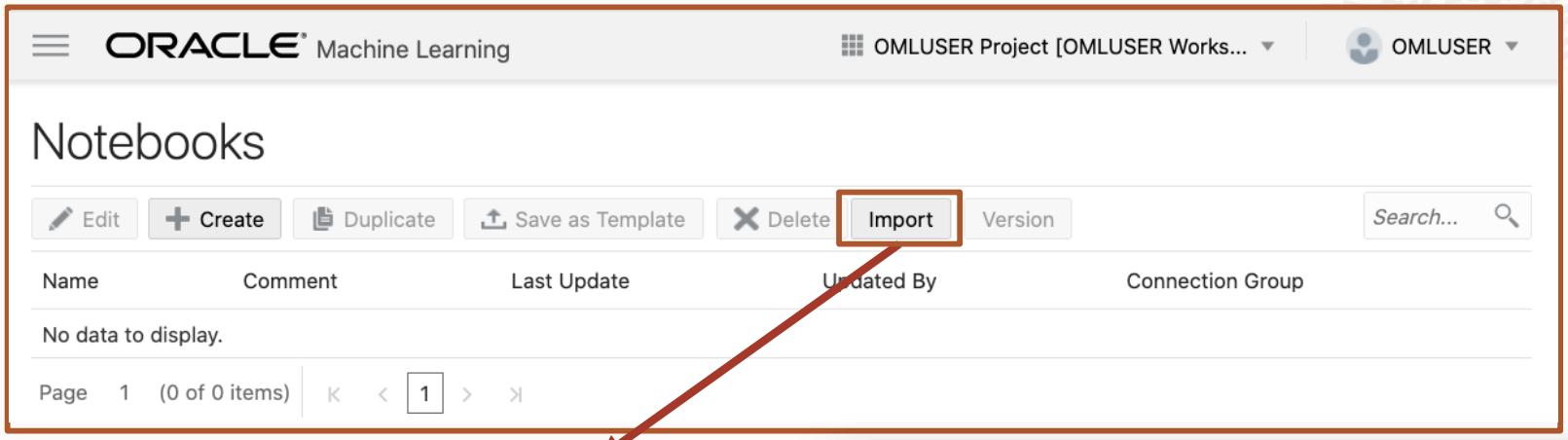
Option 2: At each Lab, under the Optional Section, you can download a copy of each of the Notebooks to import them into your own. Another way is to copy and paste each paragraph on a new Notebook

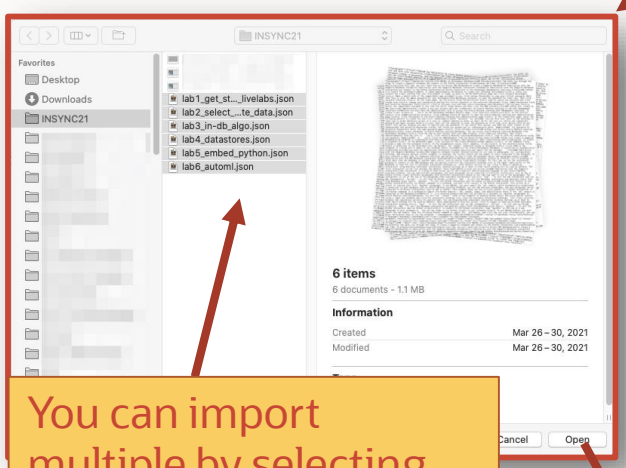




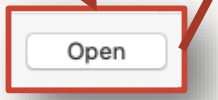
Import the Notebooks into your OML Notebooks Session

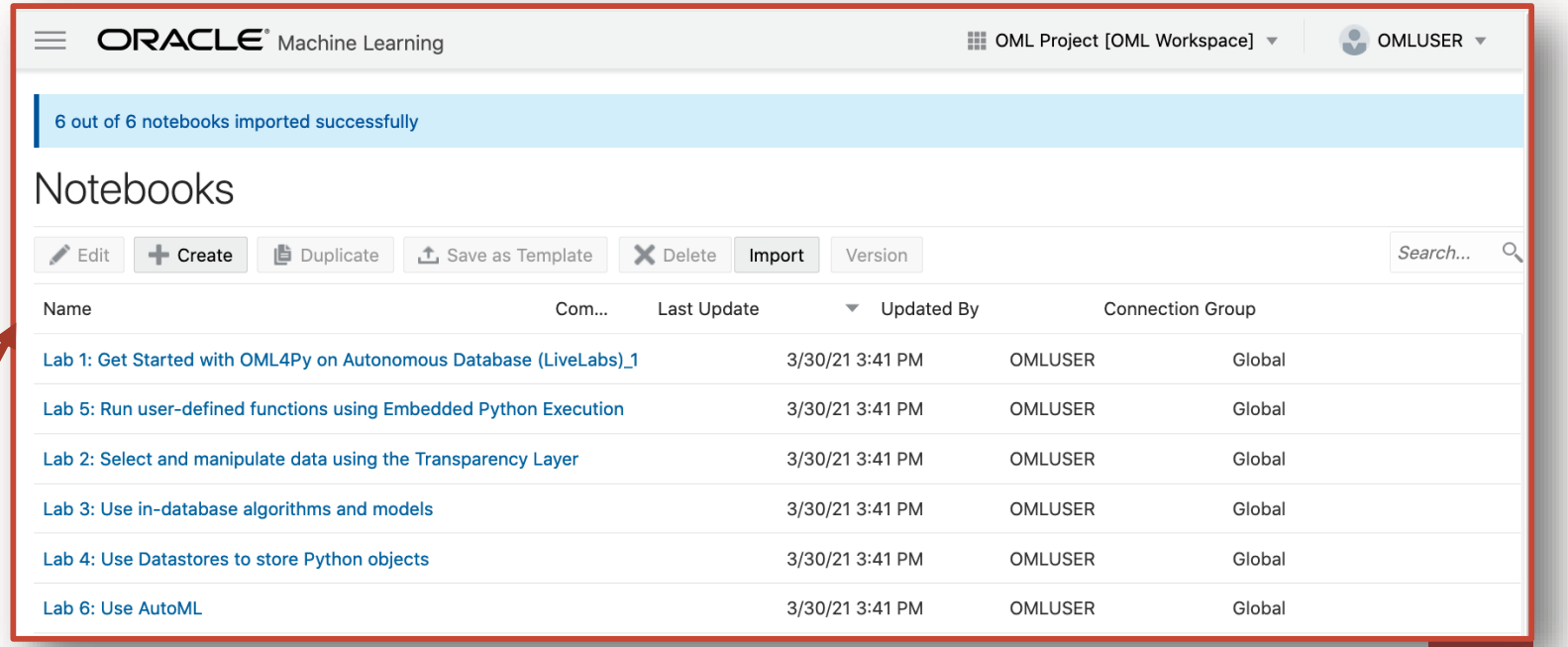
It should say 6 out of 6 notebooks imported successfully





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





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

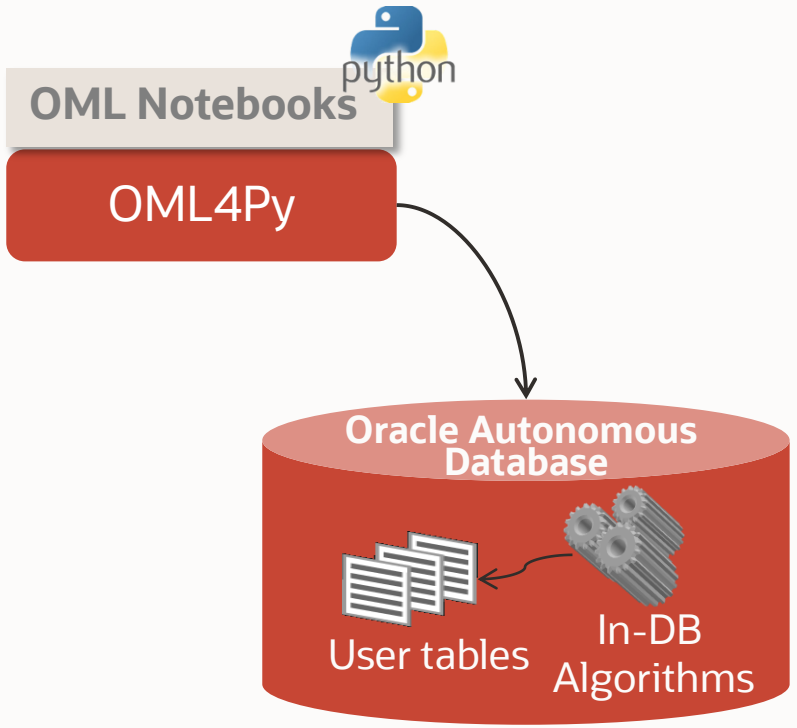
Scalable in-database algorithms

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

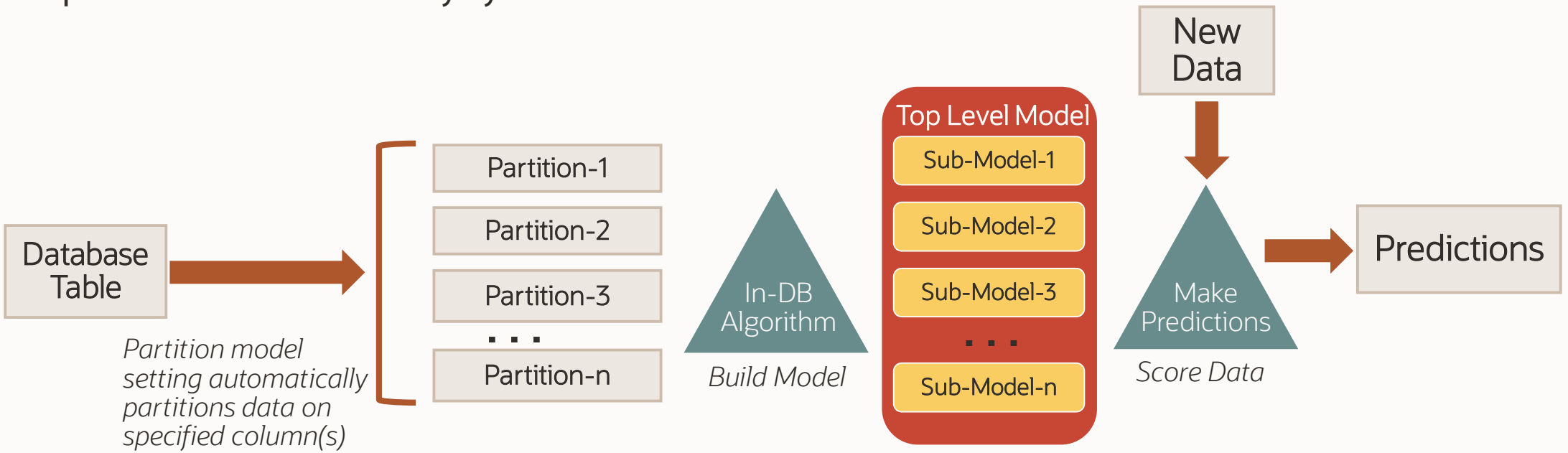
Automating a typical machine learning task

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

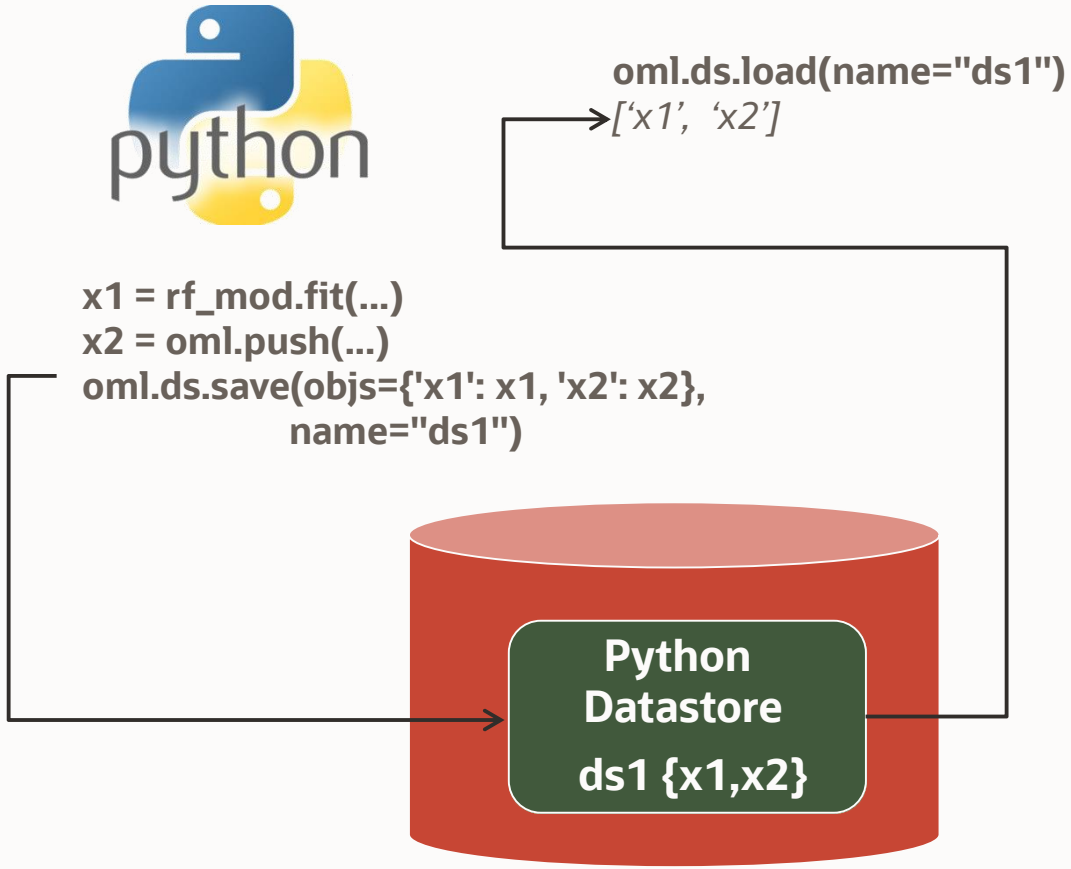
Datastore for Python object persistence

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

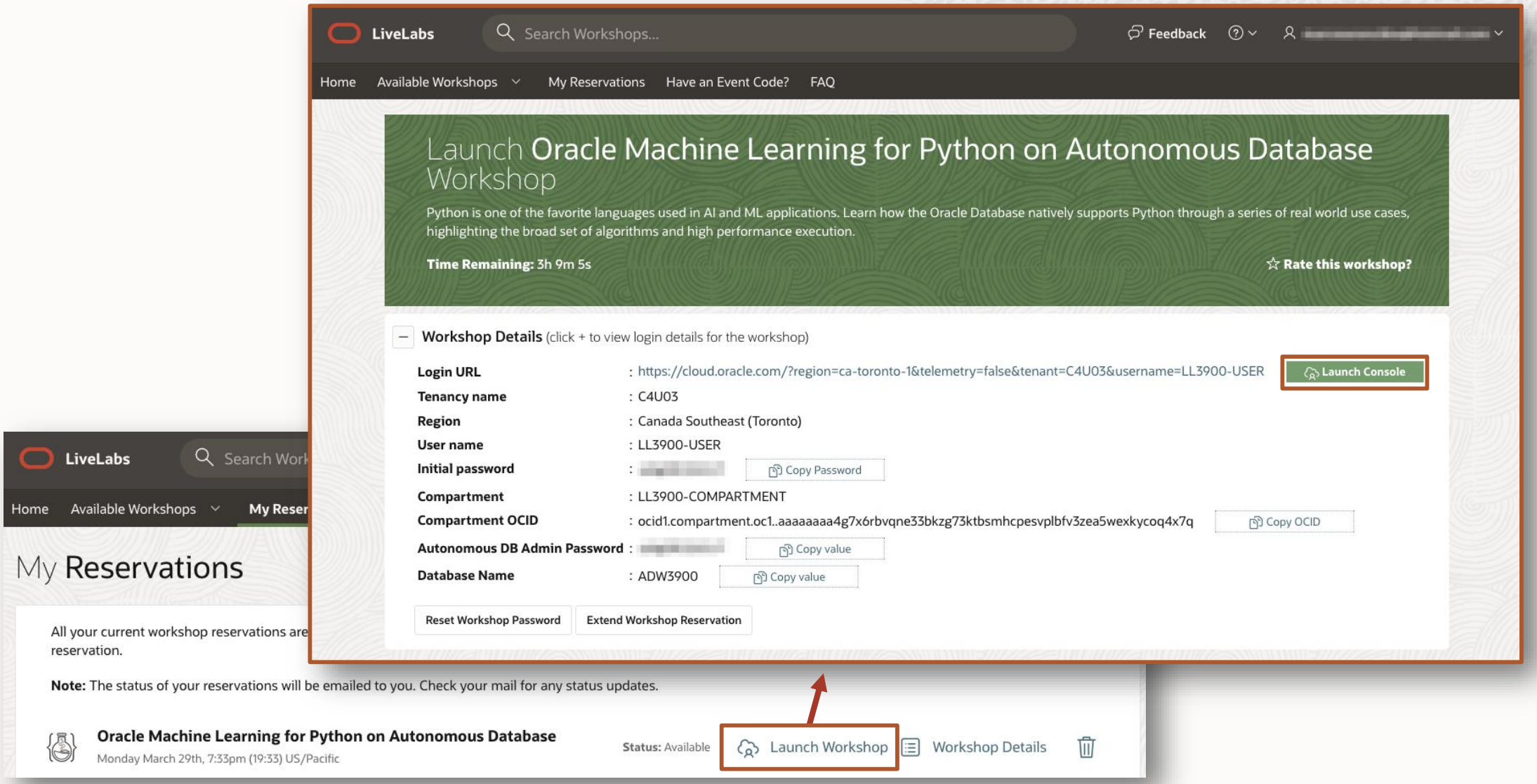
Step 7: View contents of a datastore

Step 8: Manage datastore privileges

Step 9: Delete datastore content

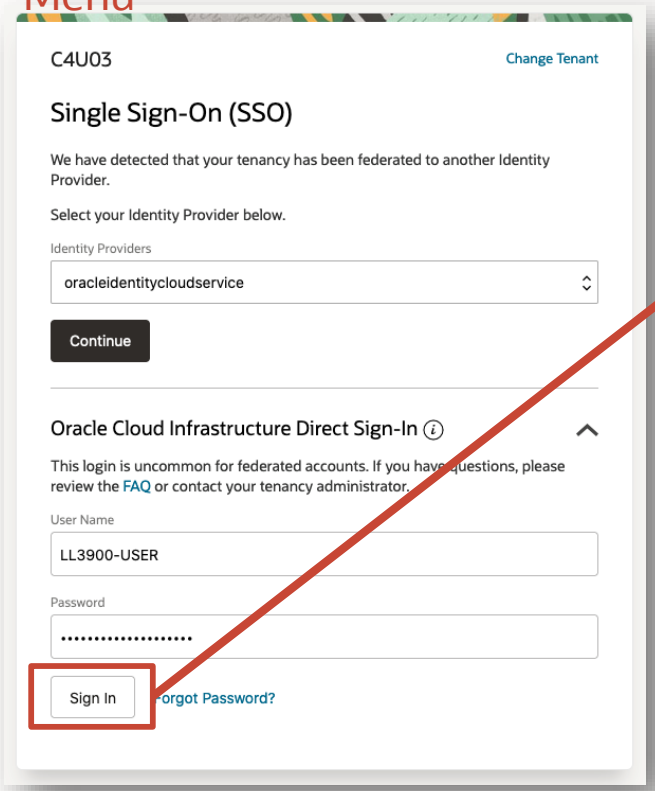
Lab 4: Create OMLUSER2 for managing privileges to OML4Py Data Store

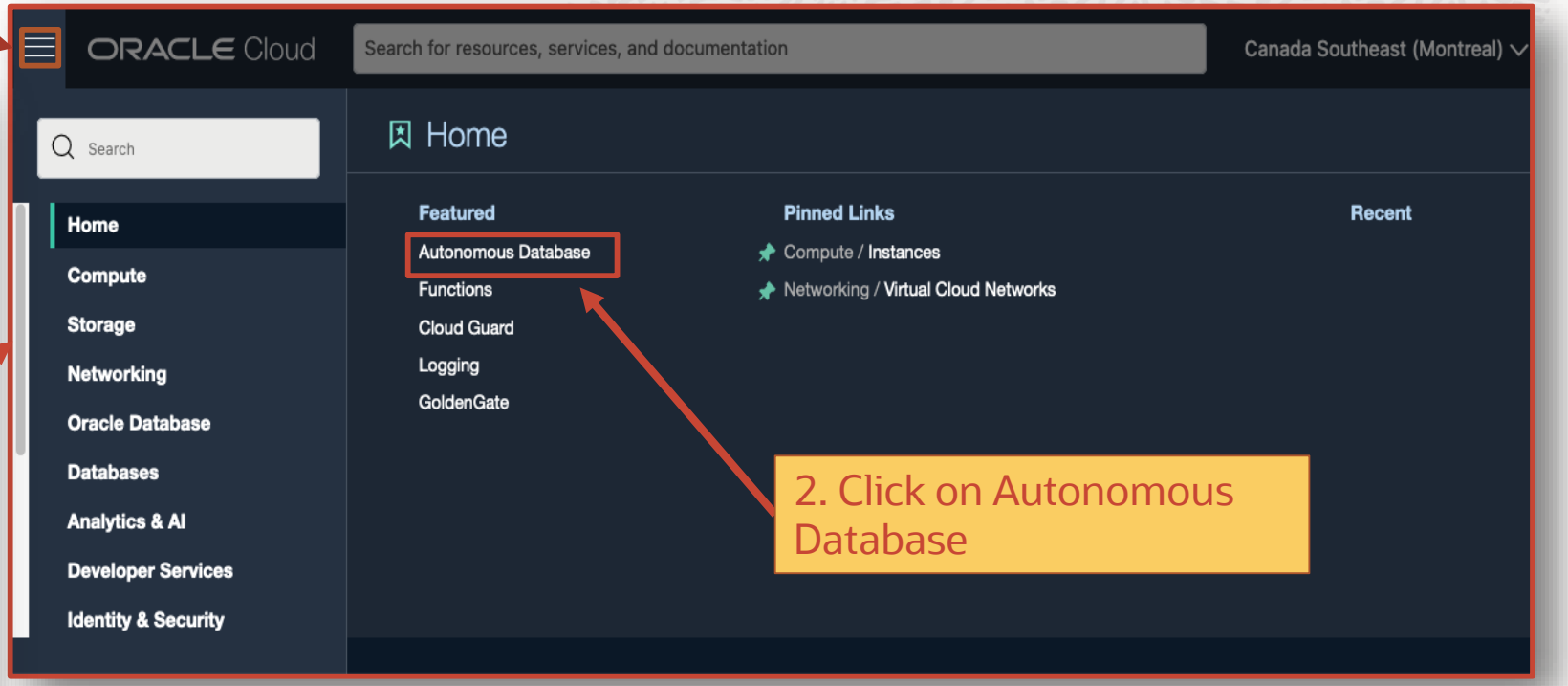
If you don't have the Autonomous Database already open, we need to get there:
In the "My Reservations" window at: https://apexapps.oracle.com/pls/apex/dbpm/r/livelabs/my-reservations
Click on "Launch Workshop", then "Launch Console"



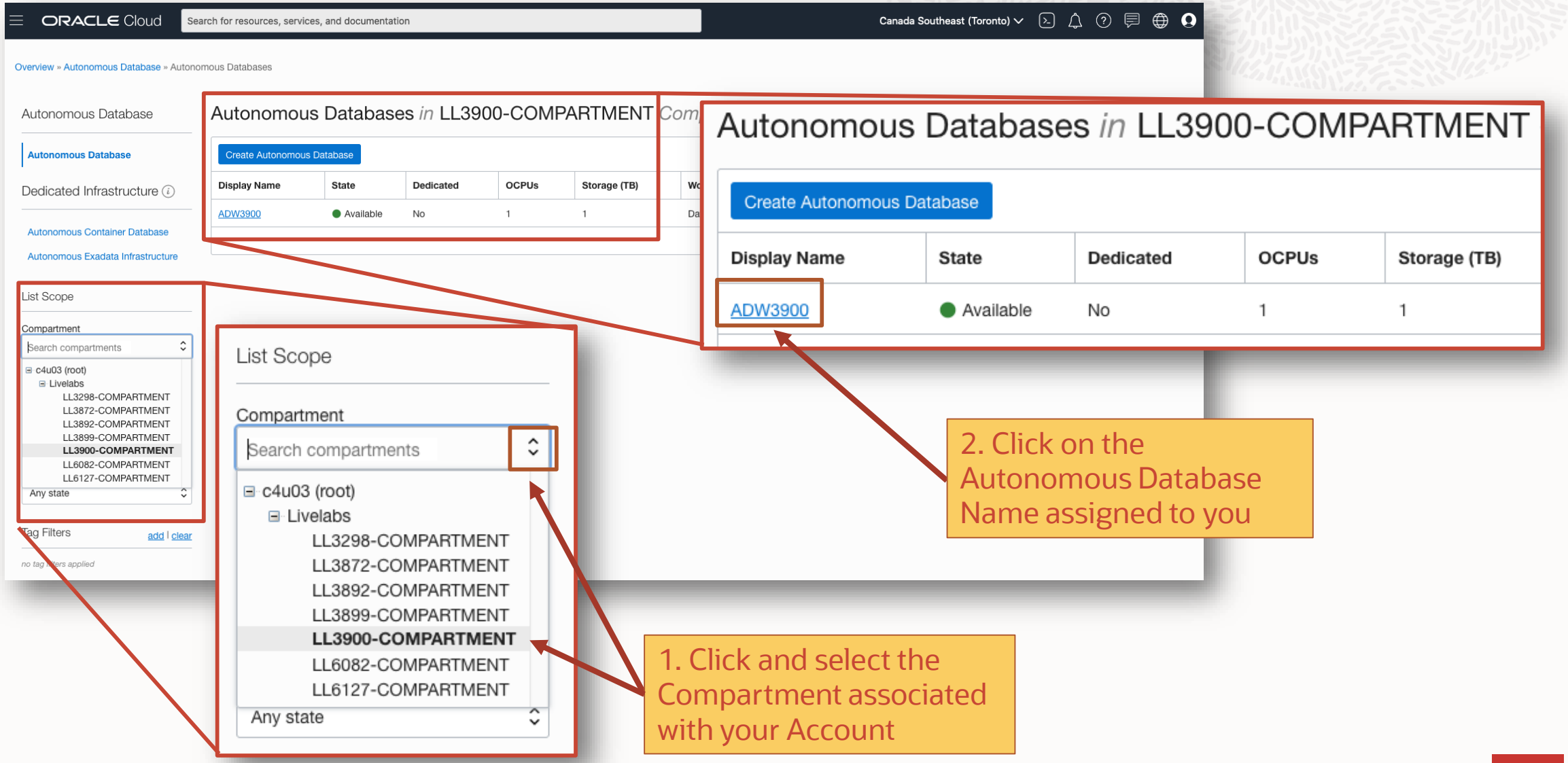
In the Single Sign-On screen, enter the Password given in the Workshop screen for your user LL####-USER (or whatever password you changed it to, if you did change it). Click on the Hamburger Menu on the Top left of OCI, then Autonomous Database.

1. Click the Hamburger Menu

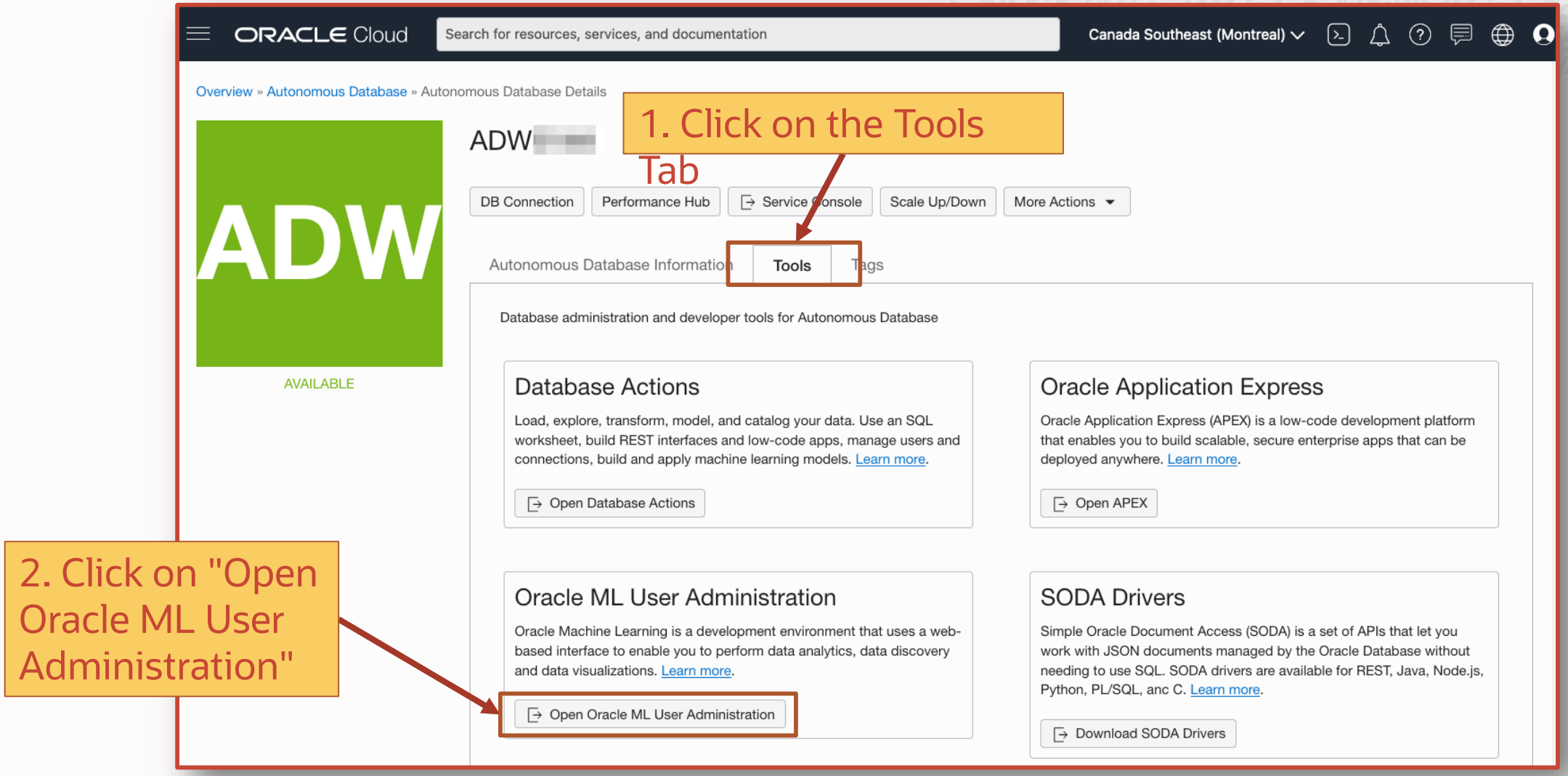




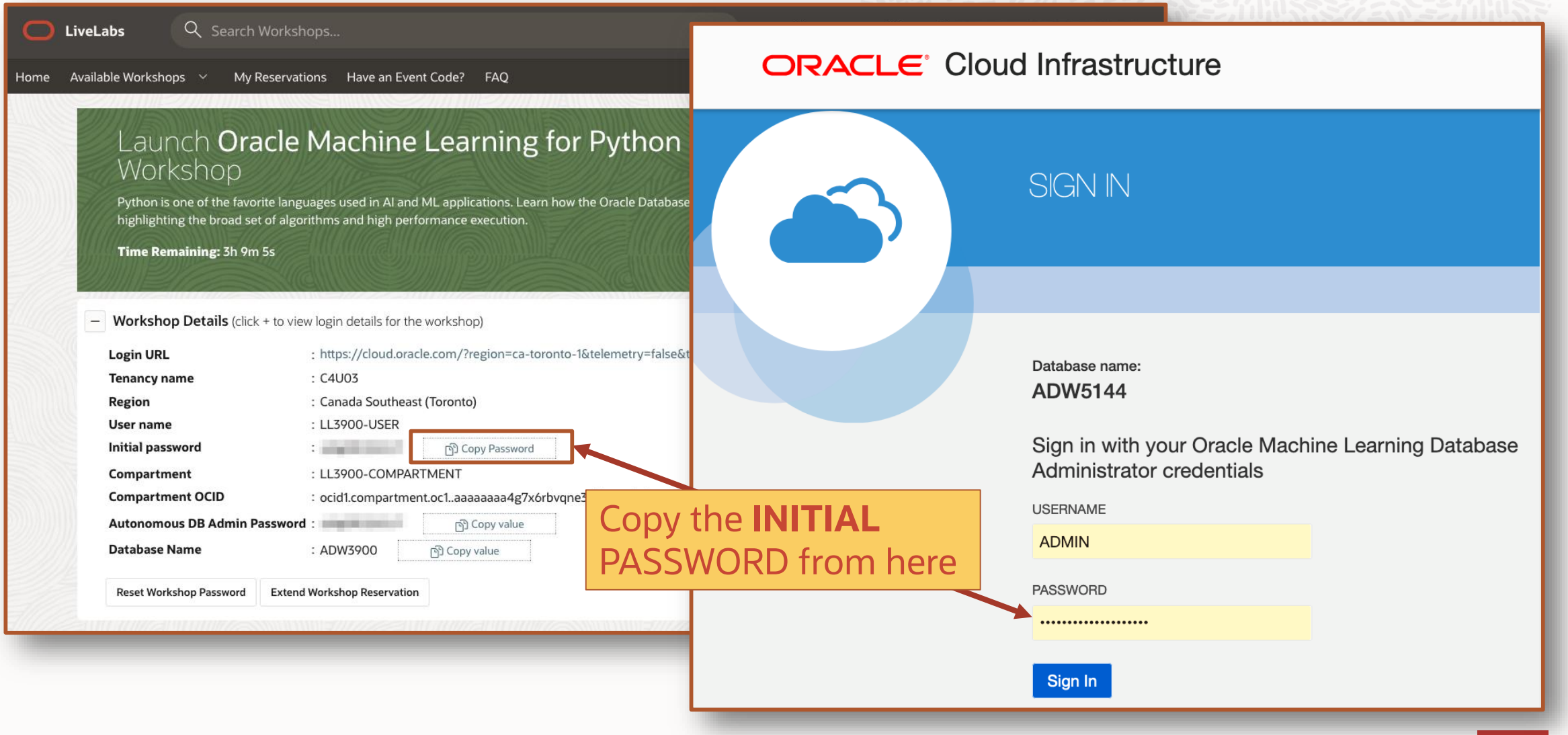
Select the specific ADW Compartment indicated in the Workshop Main screen, and then click on the Autonomous Database



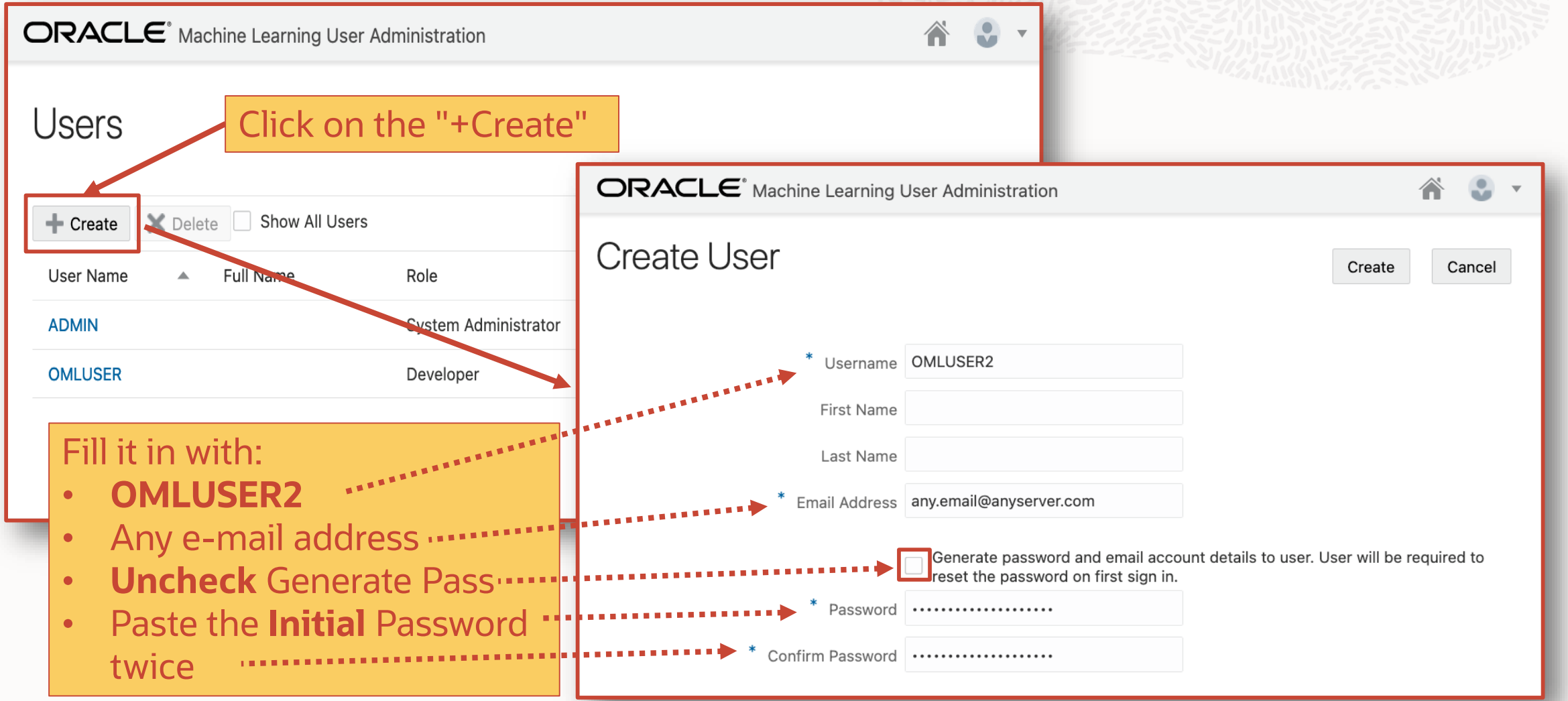
Once inside the Autonomous Database, click on the "Tools" tab.
In there, click on the "Open Oracle ML User Administration" under the Oracle ML Tile



You will be asked to Login, this time as the Administrator of the Oracle Database Make sure to use the user ADMIN and the Initial Password indicated in the Console



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



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

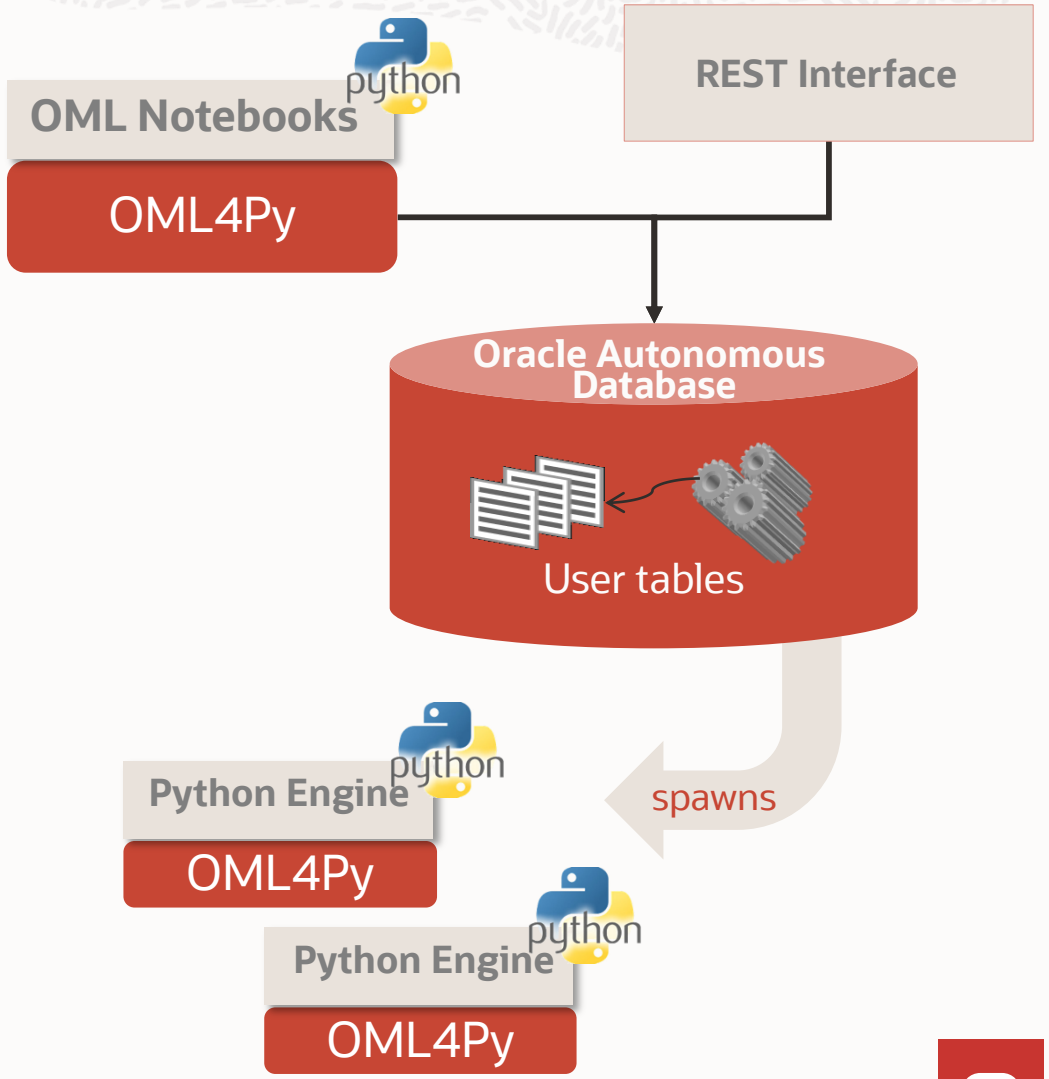
oml.index_apply(times, func[, func_owner, . . .])

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.group_apply(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

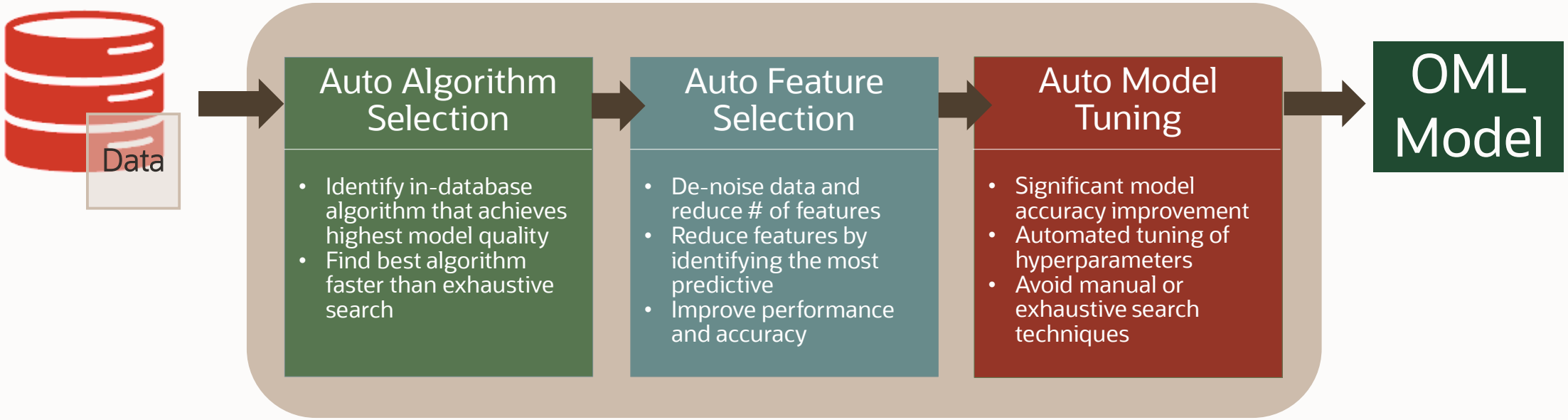
OML4Py AutoML objectives

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

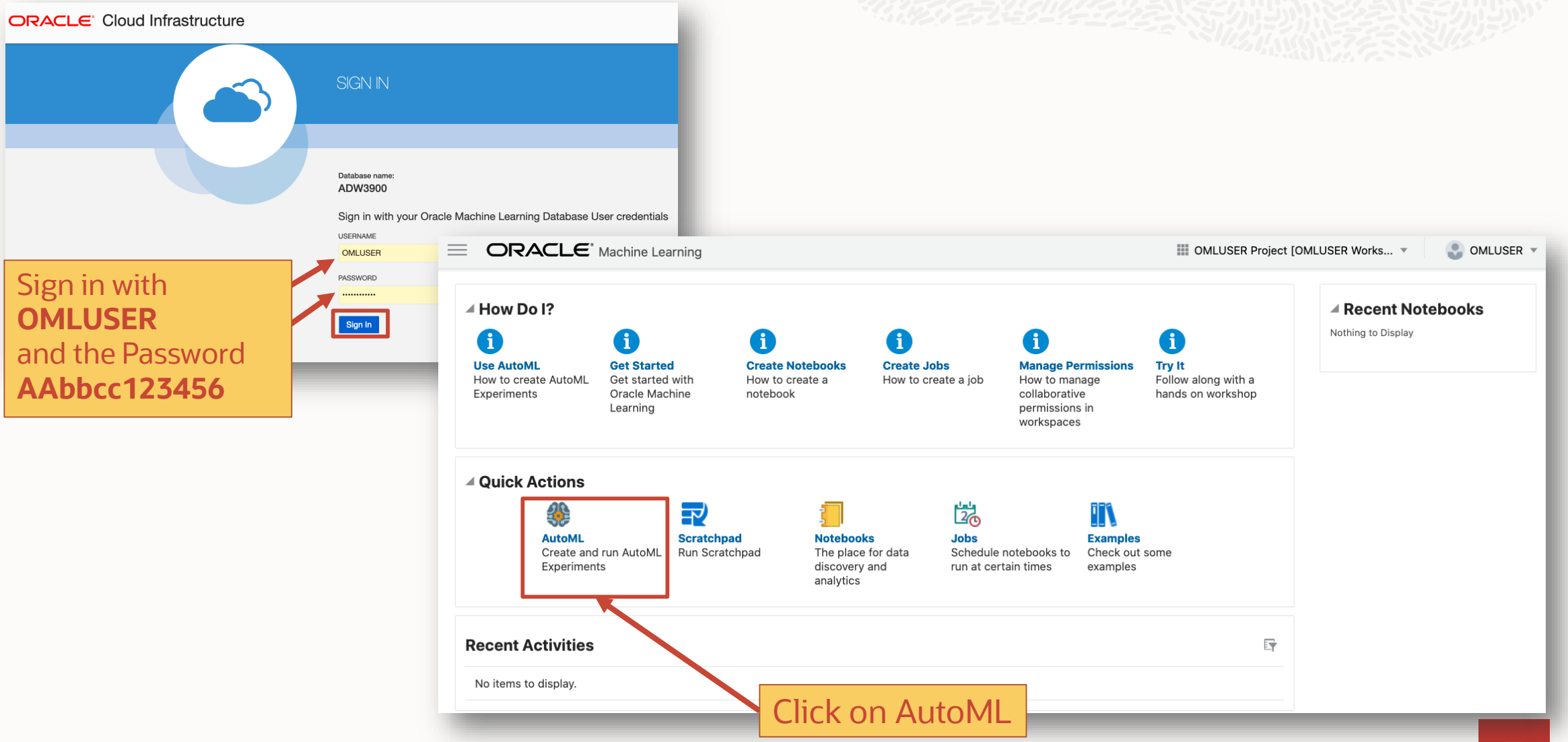
Note: Some AutoML function invocations can take a few minutes to complete.
A lot of going on behind the scenes. Please be patient. ☺

Work on lab 6

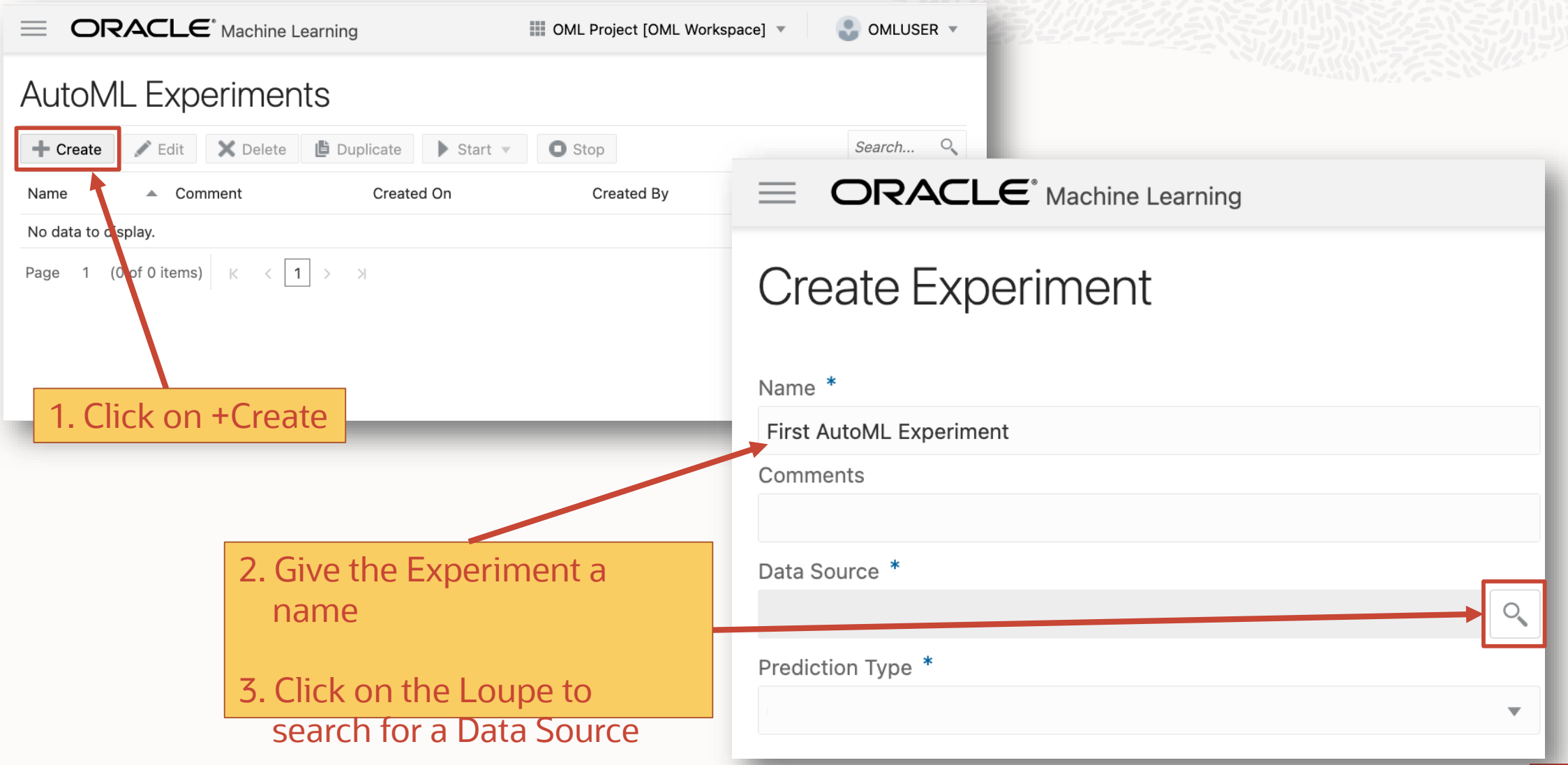
10 Minutes…

Lab 6: Bonus Round - Use OML AutoML UI

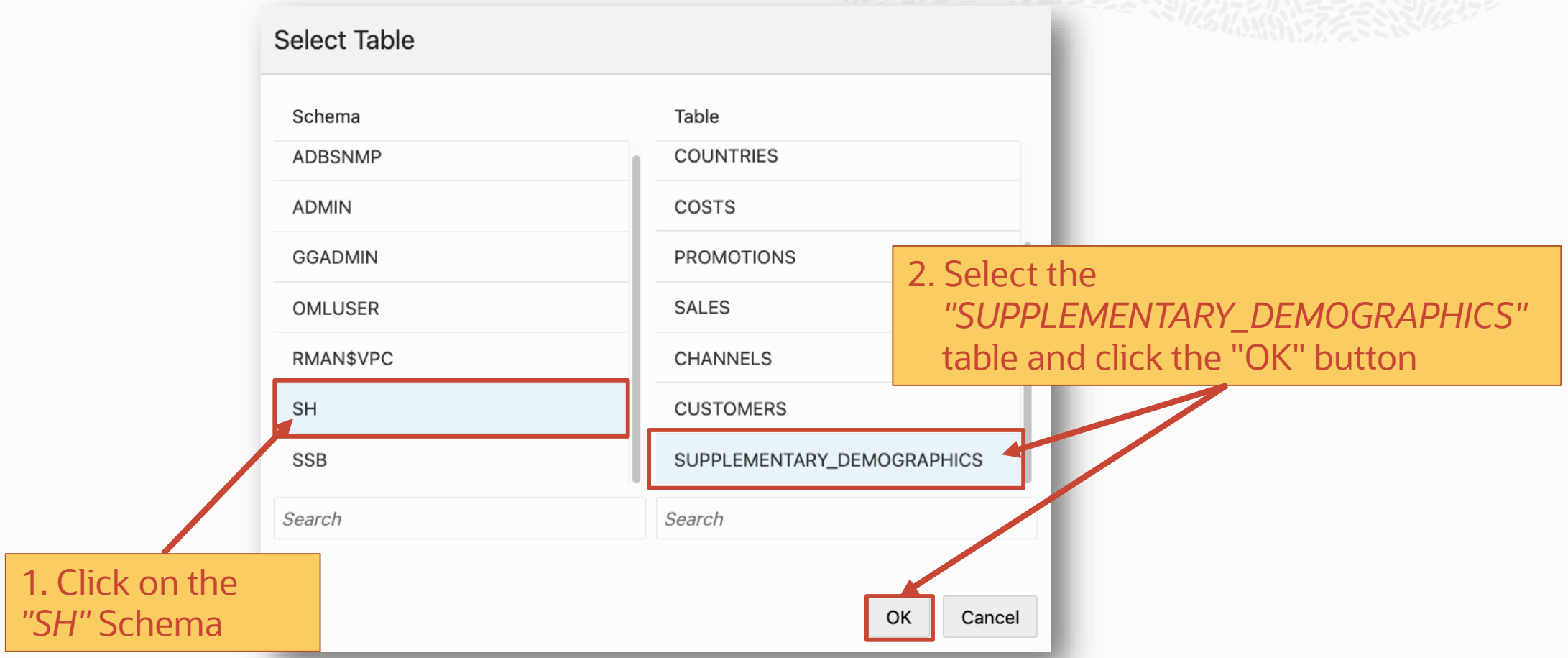
If you are not yet at the OML Notebooks screen, Sign-In again with OMLUSER and the Password AAbbcc123456. Once you are in, click on the AutoML link



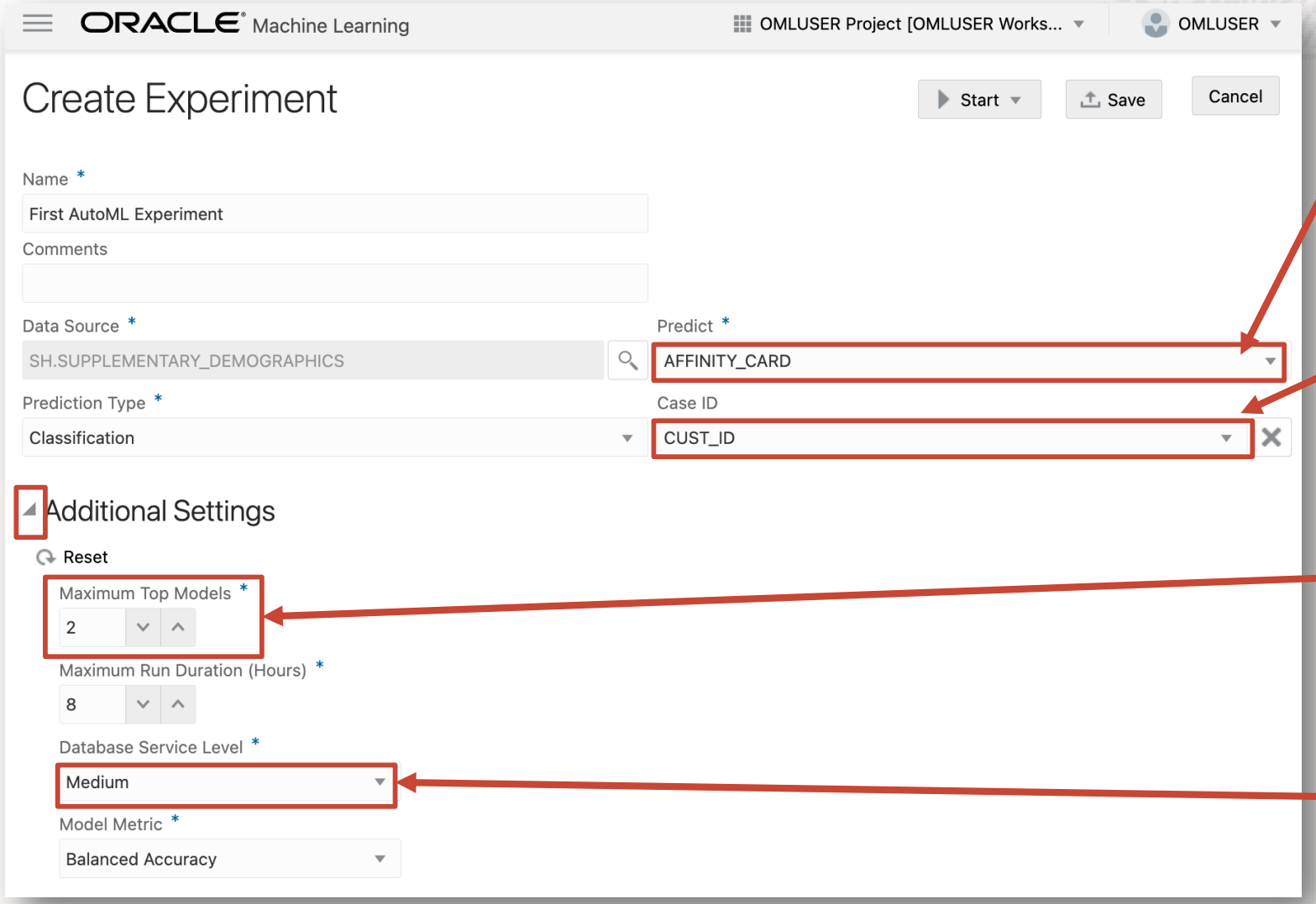
In the AutoML Experiments screen, click "Create" to create a new one



# 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



1. For the "Predict" attribute, select AFFINITY_CARD from the pull-down menu

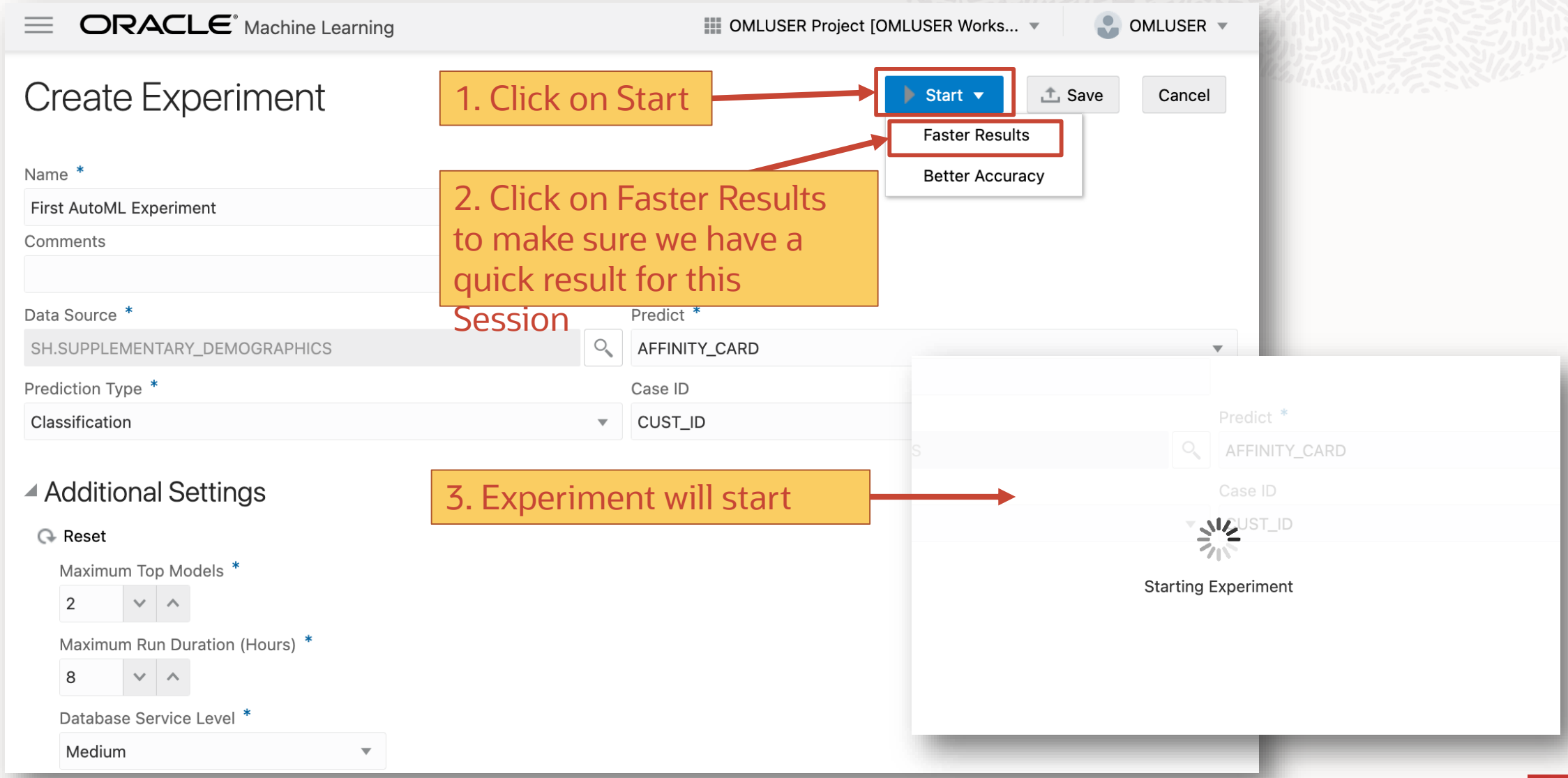
2. For the "Case ID" attribute, select CUST_ID from the pull-down menu

3. Expand the "Additional Settings" and reduce the Maximum Top Models to "2" to save time in this exercise.

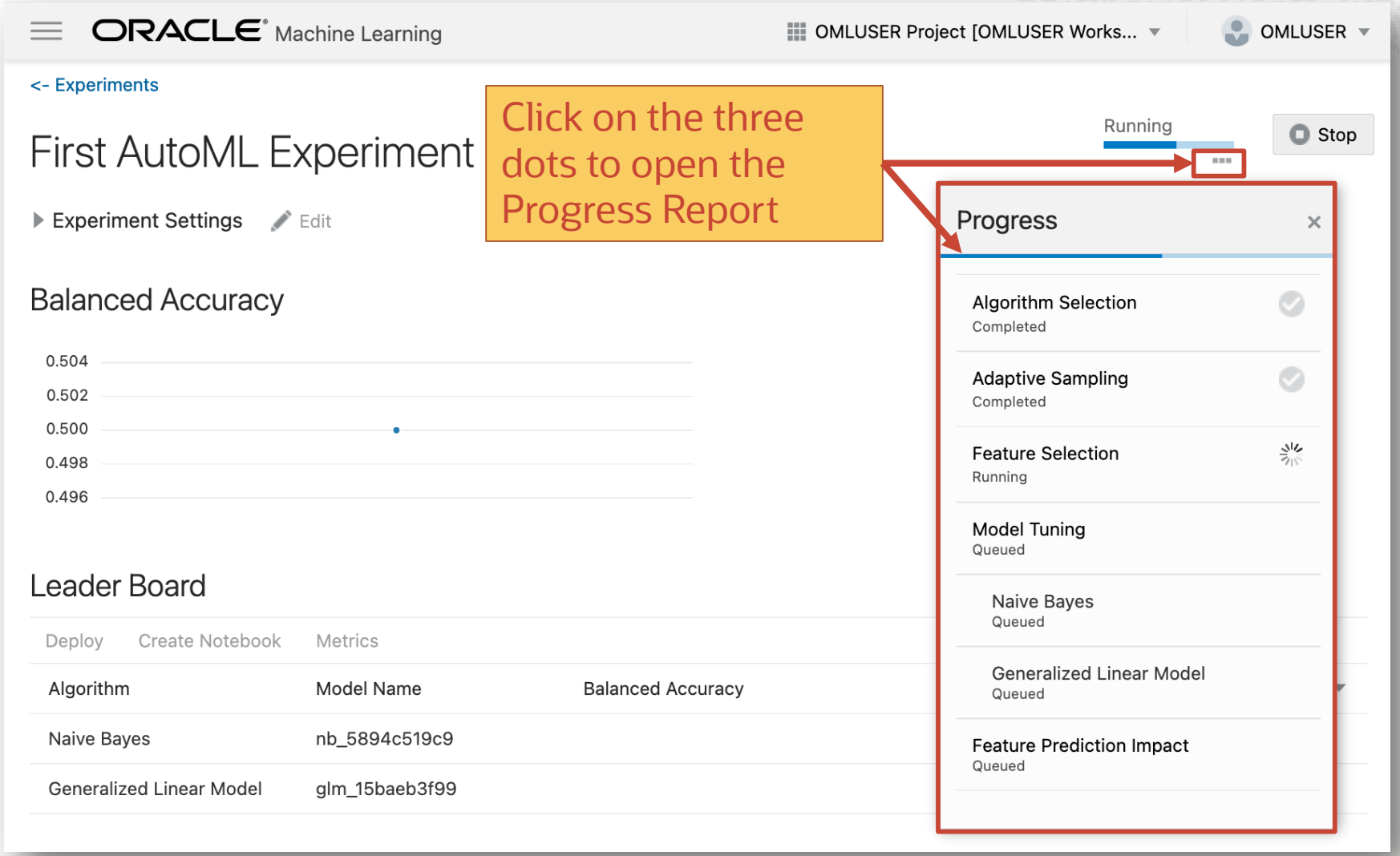
4. Change the *"Database Service Level"* to Medium to get increased parallelism

We are ready to start the Experiment

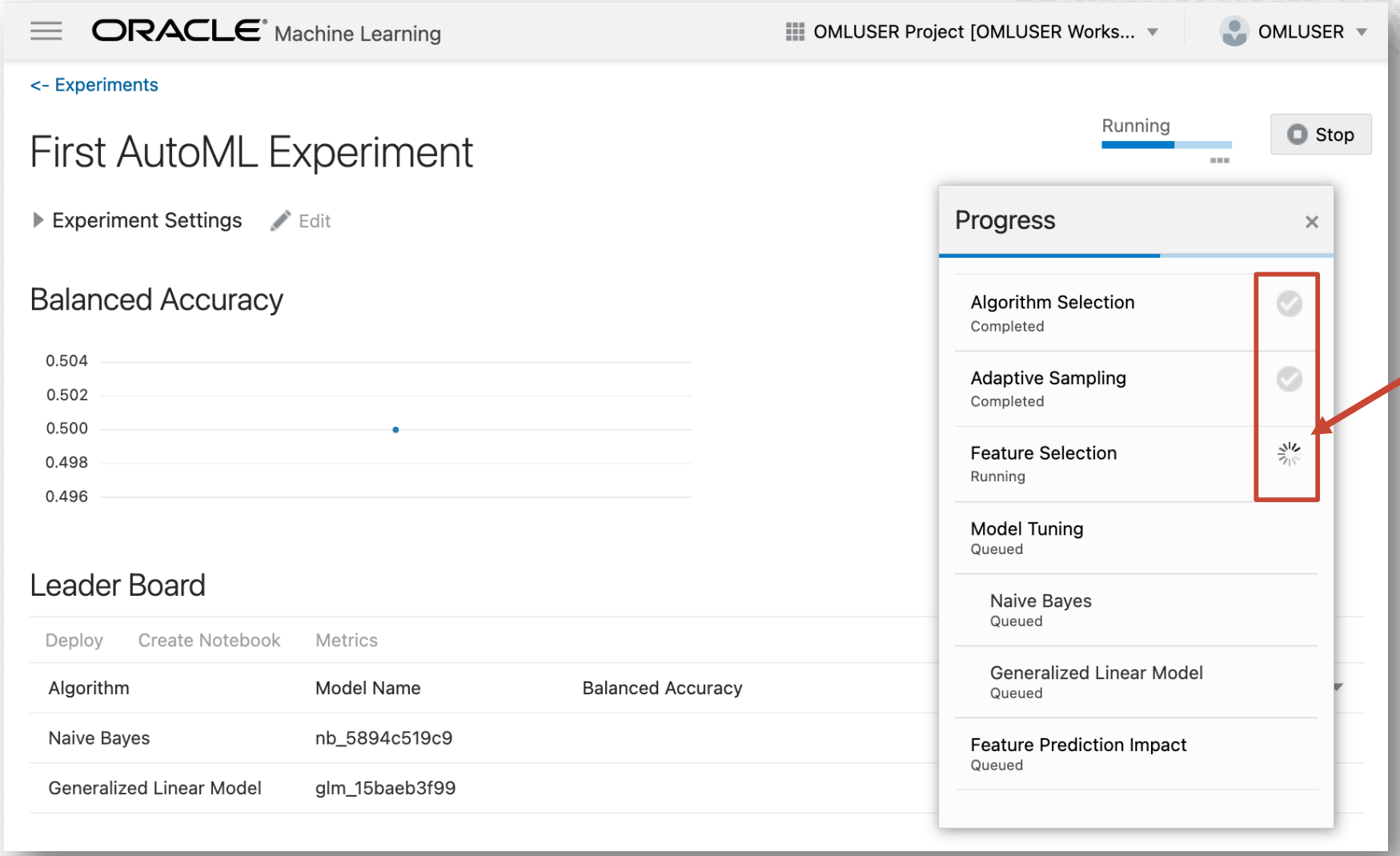
Click on the Start -> Faster Results



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

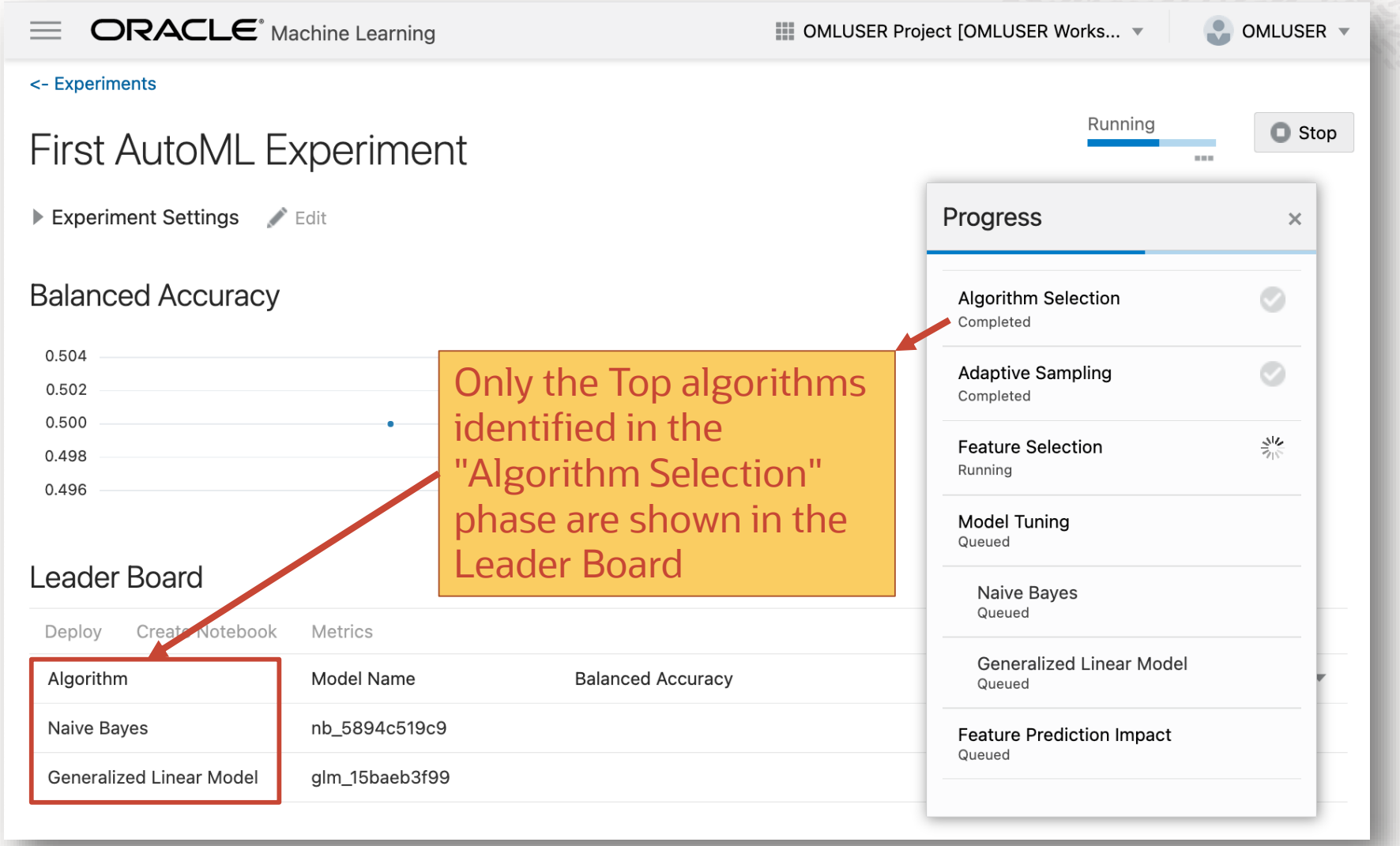


Completed vs Running Steps

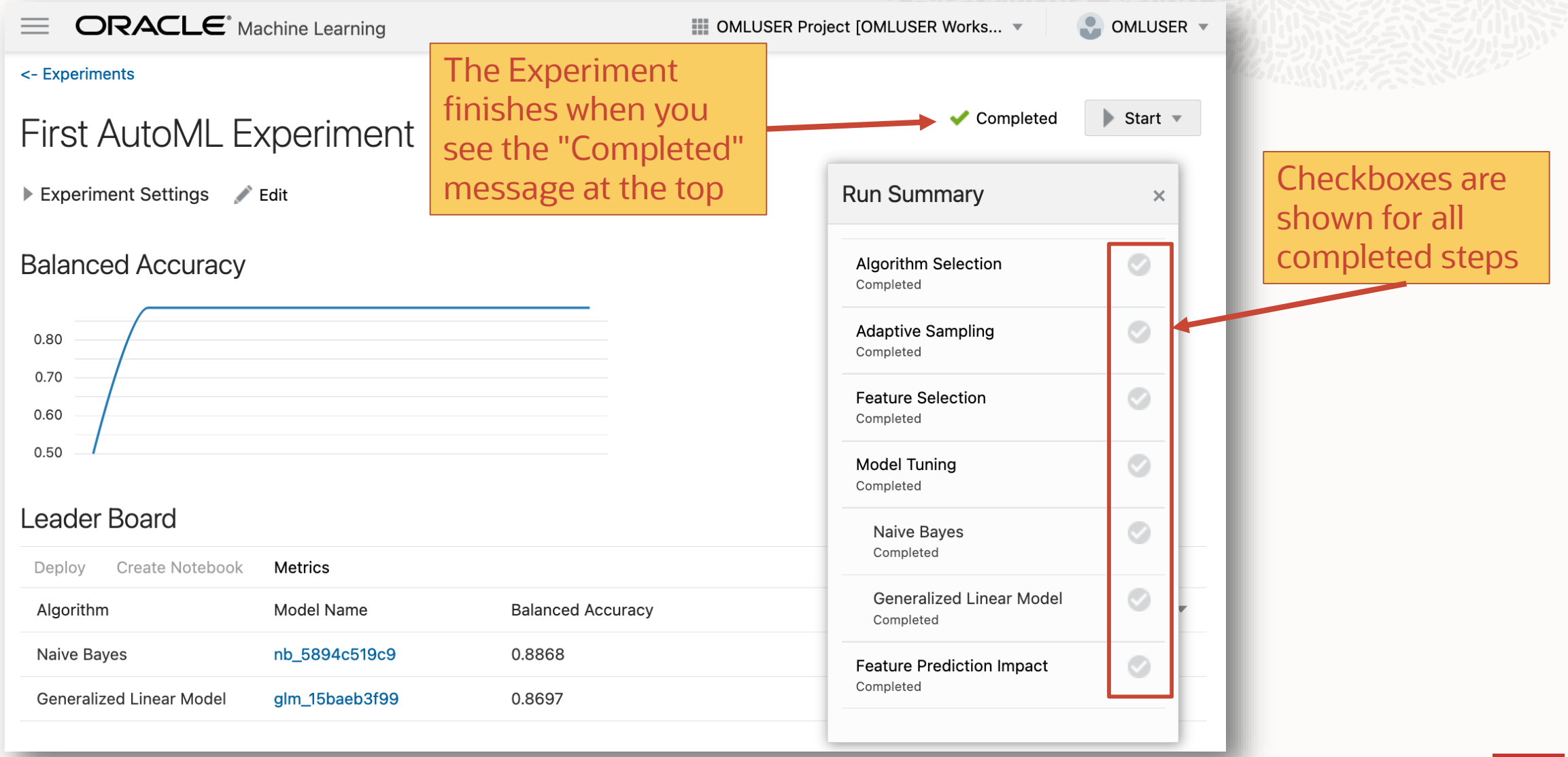


Checkboxes are shown for completed steps, while running steps show an animation

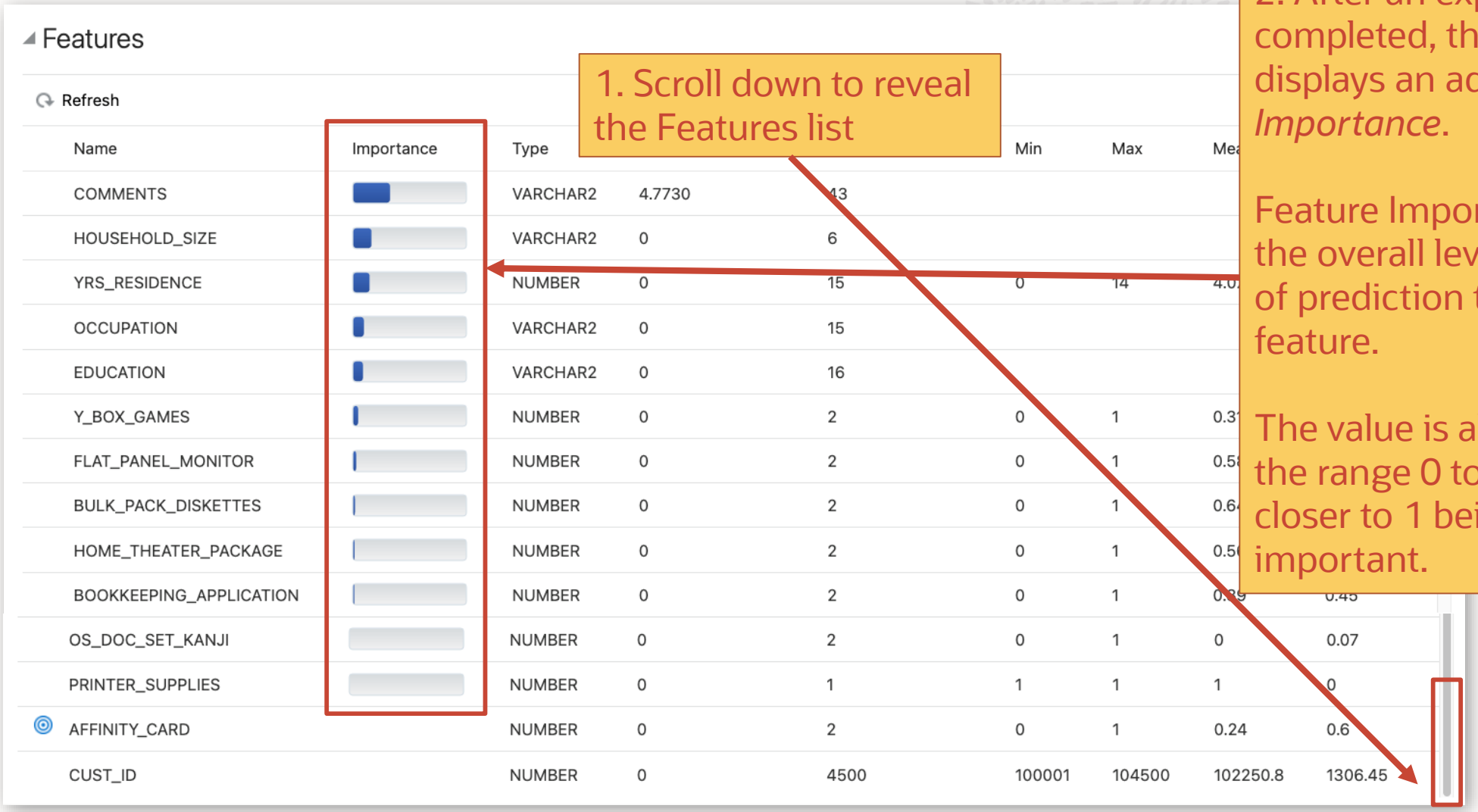
Top algorithms selected by AutoML



A completed Experiment looks like this



Let's check the global Feature Importance



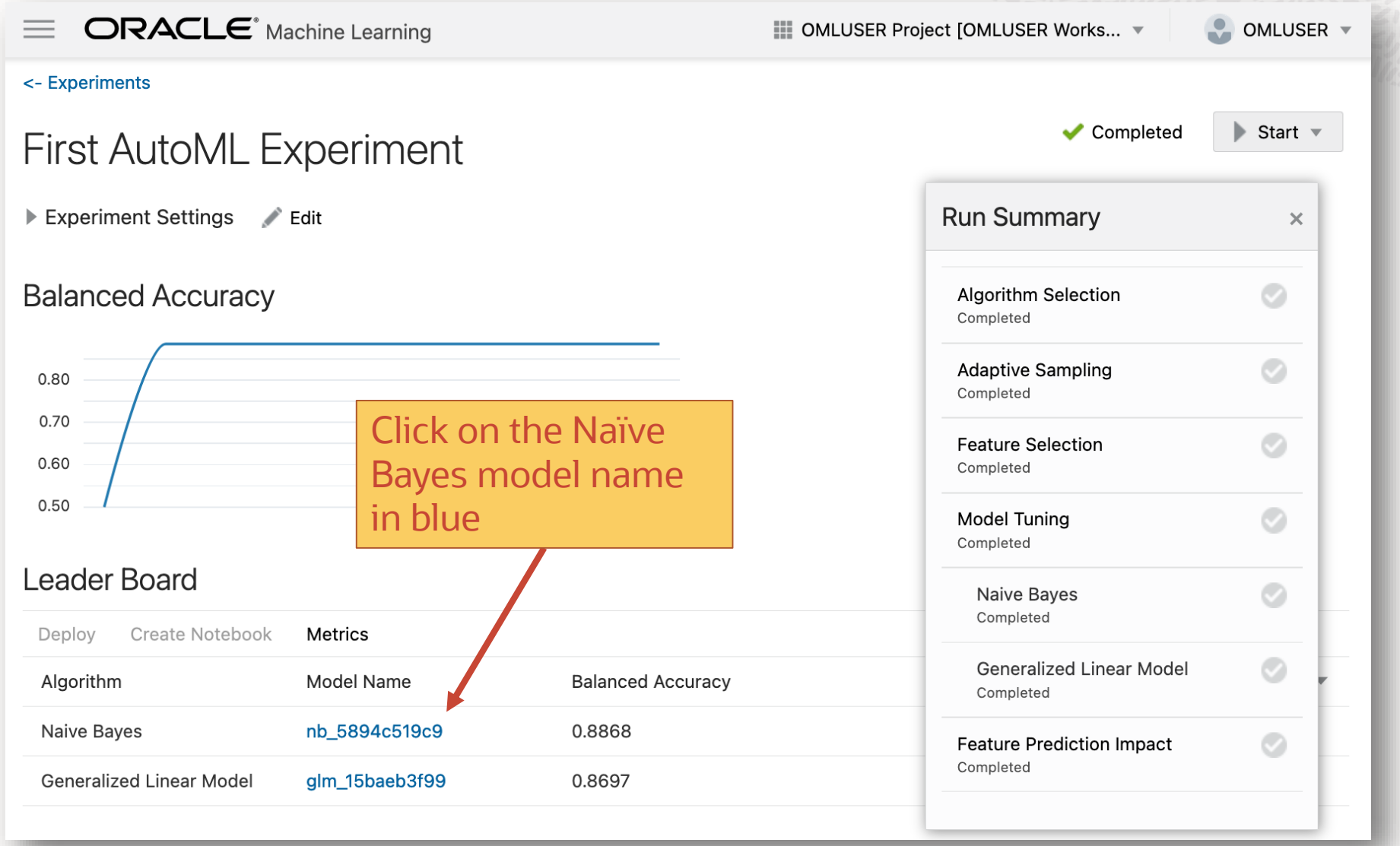
1. Scroll down to reveal the Features list

2. After an experiment run is completed, the Features grid displays an additional column *Importance*.

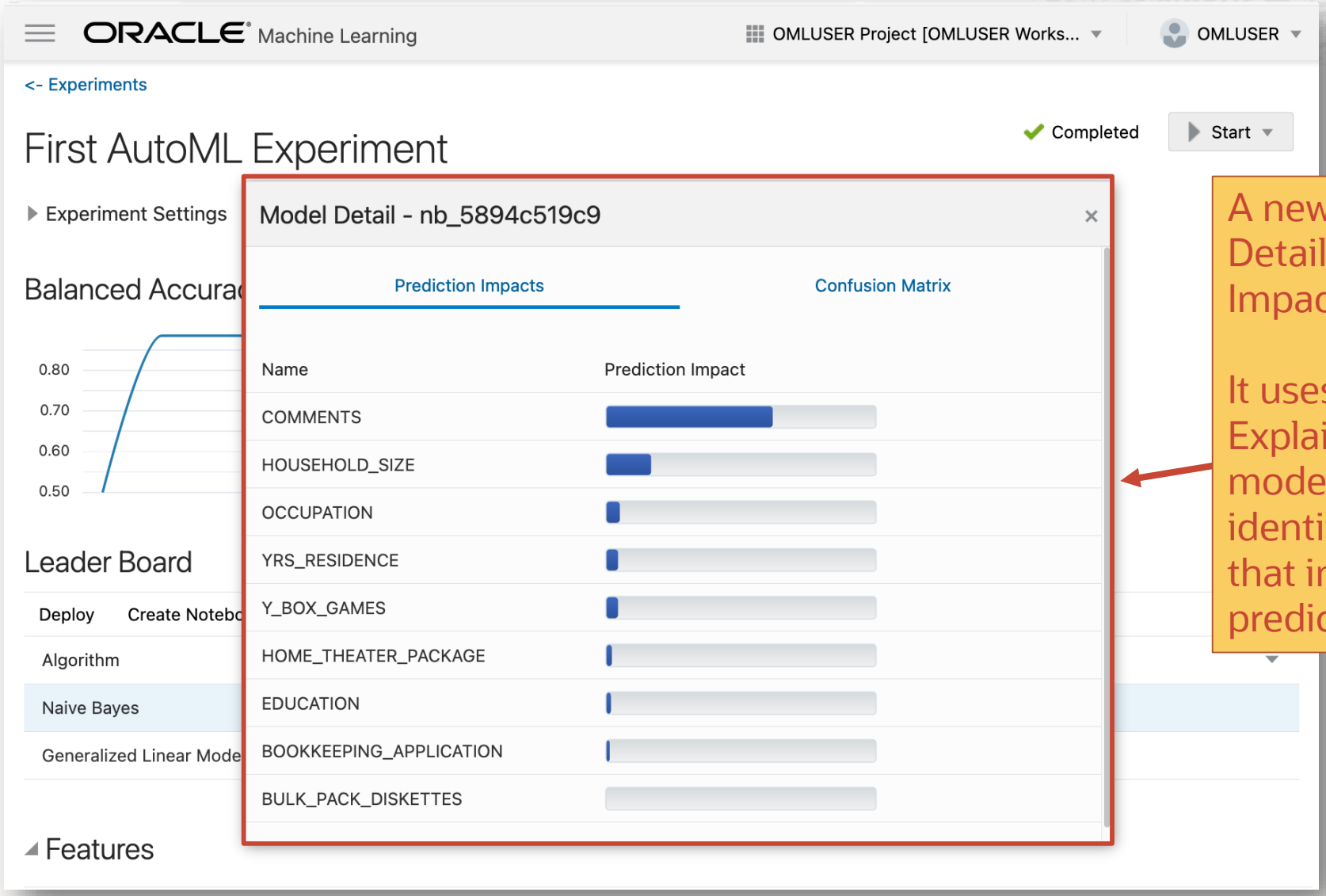
Feature Importance indicates the overall level of sensitivity of prediction to a particular feature.

The value is always depicted in the range 0 to 1, with values closer to 1 being more important.

Let's open the Model Detail to inspect the Model



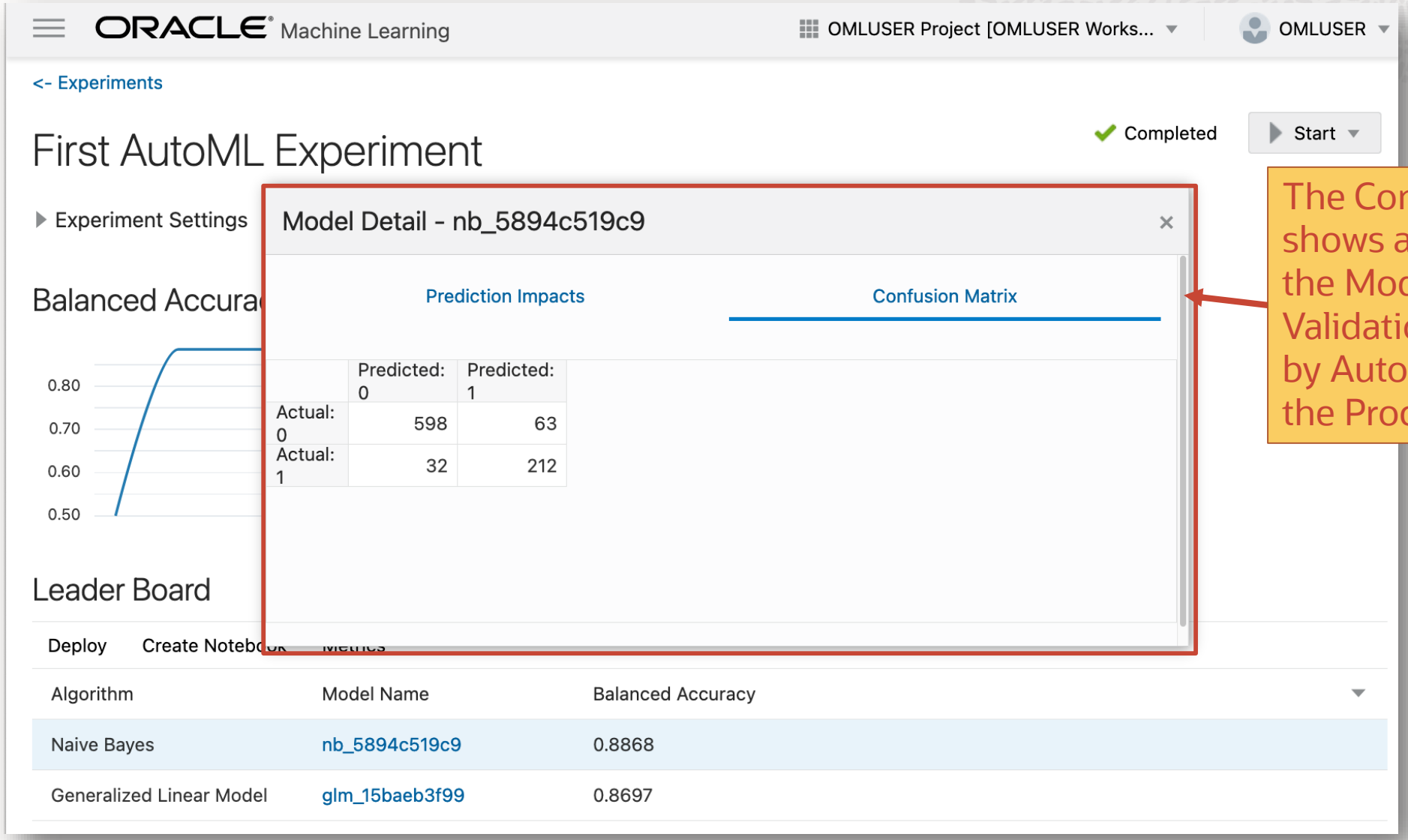
Review the Model Detail – Prediction Impacts



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 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

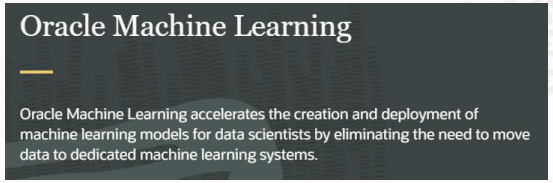
Lab 6: Bonus Round - Use OML AutoML UI

Where to go from here?

Helpful Links

ORACLE MACHINE LEARNING ON O.COM

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



OML TUTORIALS

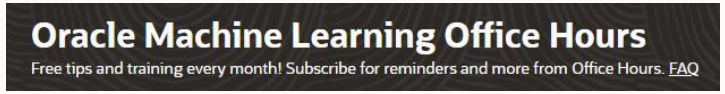
OML LiveLab: https://apexapps.oracle.com/pls/apex/dbpm/r/livelabs/view-workshop?p180_id=560

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

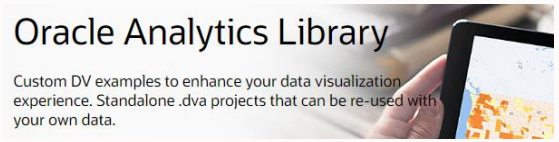
OML OFFICE HOURS

https://asktom.oracle.com/pls/apex/asktom.search?office=6801#sessionss



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#sessionss

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