République Algérienne démocratique et populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
École nationale supérieure d'informatique (ESI ex. INI)

BDA PROJECT #2

2nd Cycle Software Engineering (2CS, SIL)
2024-2025

Technical Report on Oracle Machine Learning (OML) with PL/SQL

Prepared by

- **Mahdia Toubal**
- **Dina Keddour**
- **LOUNI IMENE**
- **Guitoun Djihen**

Group SIL2

Table of content

1. Introduction

Oracle Machine Learning (OML) enables data scientists and developers to build and deploy machine learning models directly within the Oracle Database. This approach offers significant advantages by eliminating data movement, reducing latency, and leveraging the database's computational power. In this project, we demonstrate how to build a diabetes prediction model using PL/SQL and OML capabilities.

2. Environment Setup

Oracle Cloud Infrastructure

To begin our project, we set up the following environment:

1. **Oracle Cloud Account:**
   Created an oracle cloud free tier account which provides access to Oracle Autonomous Database services with Oracle Machine Learning (OML) capabilities (cloud.oracle.com).
2. **Autonomous Database Provisioning**
   - selected "Autonomous Data Warehouse" for our analytical workload.
   - Configured with 2 OCPU and 1TB storage.
   - Enabled Oracle Machine Learning services during provisioning.
3. **Accessing Development Environment**
   - Navigated to OML Notebooks interface.
   - Created a new notebook for PL/SQL development.

3. Data Preparation

a. **Creating the Diabetes Dataset Table**
   We started by creating a table to store our diabetes dataset with relevant medical features:

```
%script
DROP TABLE diabetes_data PURGE;

CREATE TABLE diabetes_data (
    Pregnancies  NUMBER,
    Glucose      NUMBER,
    BloodPressure NUMBER,
    SkinThickness NUMBER,
    Insulin      NUMBER,
    BMI          NUMBER,
    DiabetesPedigreeFunction NUMBER,
    Age          NUMBER,
    Outcome      NUMBER
);
```

b. **Inserting Sample Data**
We populated the table with sample records representing patient data and diabetes outcomes (1 = diabetic, 0 = non-diabetic):

```
%script
INSERT ALL
  INTO diabetes_data (Pregnancies, Glucose, BloodPressure, SkinThickness, Insulin, BMI, DiabetesPedigreeFunction, Age, Outcome) VALUES (6, 148, 72, 35, 0, 33.6, 0.627,
  INTO diabetes_data (Pregnancies, Glucose, BloodPressure, SkinThickness, Insulin, BMI, DiabetesPedigreeFunction, Age, Outcome) VALUES (1, 85, 66, 29, 0, 26.6, 0.351,
  INTO diabetes_data (Pregnancies, Glucose, BloodPressure, SkinThickness, Insulin, BMI, DiabetesPedigreeFunction, Age, Outcome) VALUES (8, 183, 64, 0, 0, 23.3, 0.672,
  INTO diabetes_data (Pregnancies, Glucose, BloodPressure, SkinThickness, Insulin, BMI, DiabetesPedigreeFunction, Age, Outcome) VALUES (1, 89, 66, 23, 94, 28.1, 0.167,
  INTO diabetes_data (Pregnancies, Glucose, BloodPressure, SkinThickness, Insulin, BMI, DiabetesPedigreeFunction, Age, Outcome) VALUES (0, 137, 40, 35, 168, 43.1, 2.28
  INTO diabetes_data (Pregnancies, Glucose, BloodPressure, SkinThickness, Insulin, BMI, DiabetesPedigreeFunction, Age, Outcome) VALUES (5, 116, 74, 0, 0, 25.6, 0.201,
  INTO diabetes_data (Pregnancies, Glucose, BloodPressure, SkinThickness, Insulin, BMI, DiabetesPedigreeFunction, Age, Outcome) VALUES (3, 78, 50, 32, 88, 31, 0.248,
  INTO diabetes_data (Pregnancies, Glucose, BloodPressure, SkinThickness, Insulin, BMI, DiabetesPedigreeFunction, Age, Outcome) VALUES (10, 115, 0, 0, 0, 35.3, 0.134,
  INTO diabetes_data (Pregnancies, Glucose, BloodPressure, SkinThickness, Insulin, BMI, DiabetesPedigreeFunction, Age, Outcome) VALUES (2, 197, 70, 45, 543, 30.5, 0.15
  INTO diabetes_data (Pregnancies, Glucose, BloodPressure, SkinThickness, Insulin, BMI, DiabetesPedigreeFunction, Age, Outcome) VALUES (8, 125, 96, 0, 0, 0, 0.232, 54,
  INTO diabetes_data (Pregnancies, Glucose, BloodPressure, SkinThickness, Insulin, BMI, DiabetesPedigreeFunction, Age, Outcome) VALUES (4, 110, 92, 0, 0, 37.6, 0.191,
  INTO diabetes_data (Pregnancies, Glucose, BloodPressure, SkinThickness, Insulin, BMI, DiabetesPedigreeFunction, Age, Outcome) VALUES (10, 168, 74, 0, 0, 38, 0.537, 3
  INTO diabetes_data (Pregnancies, Glucose, BloodPressure, SkinThickness, Insulin, BMI, DiabetesPedigreeFunction, Age, Outcome) VALUES (10, 139, 80, 0, 0, 27.1, 1.441,
```

c. **Splitting Data into Training and Test Sets**
For proper model evaluation, we splitted the dataset into training data (80%) and test data (20%):

```
%script
ALTER TABLE diabetes_data ADD (data_set VARCHAR2(10));

UPDATE diabetes_data
SET data_set = CASE WHEN DBMS_RANDOM.VALUE < 0.8 THEN 'TRAIN' ELSE 'TEST' END;

SELECT data_set, COUNT(*) FROM diabetes_data GROUP BY data_set;
```

4. Model Creation

With the training and test datasets prepared in the diabetes_data table, we proceeded to build a machine learning classification model using PL/SQL and Oracle's in-database mining capabilities. The process includes:

1. Dropping Previous Models and Settings (PL/SQL)

To ensure a clean setup, we first dropped any existing model name 'DIABETES_MODEL' and the associated settings table using PL/SQL anonymous blocks:

```
%script
BEGIN
  DBMS_DATA_MINING.DROP_MODEL('DIABETES_MODEL');
EXCEPTION
  WHEN OTHERS THEN NULL;
END;
/

BEGIN
  EXECUTE IMMEDIATE 'DROP TABLE diabetes_settings';
EXCEPTION
  WHEN OTHERS THEN NULL;
END;
/
```

2. Specifying Model Parameters

We defined model settings in a dedicated table using standard SQL. The key parameter was the algorithm:

```
CREATE TABLE diabetes_settings (
    setting_name  VARCHAR2(30),
    setting_value VARCHAR2(30)
);

INSERT INTO diabetes_settings (setting_name, setting_value) VALUES
  ('ALGO_NAME', 'ALGO_DECISION_TREE');
```

Here we use the ALGO_DECISION_TREE, a built-in Oracle algorithm well-suited for classification tasks.

3. Training the Model with PL/SQL

Using the DBMS_DATA_MINING.CREATE_MODEL procedure, we built the model directly in PL/SQL:

```
BEGIN
  DBMS_DATA_MINING.CREATE_MODEL(
    model_name          => 'DIABETES_MODEL',
    mining_function     => DBMS_DATA_MINING.CLASSIFICATION,
    data_table_name     => 'diabetes_data',
    case_id_column_name => 'ID',
    target_column_name  => 'Outcome',
    settings_table_name => 'diabetes_settings',
    data_schema_name    => USER
  );
END;
/
```

This step creates a model stored within the Oracle database, no external tools required.

5. Model Evaluation

Once the model was trained, we evaluated its predictive performance on the test dataset using PL/SQL and SQL queries.

1. Generating Predictions with SQL

Using the PREDICTION SQL function, we compared predicted vs actual outcomes:

```
%script
SELECT
    Pregnancies, Glucose, BloodPressure, SkinThickness, Insulin, BMI, DiabetesPedigreeFunction, Age, Outcome AS actual,
    PREDICTION(DIABETES_MODEL USING
        Pregnancies AS Pregnancies,
        Glucose AS Glucose,
        BloodPressure AS BloodPressure,
        SkinThickness AS SkinThickness,
        Insulin AS Insulin,
        BMI AS BMI,
        DiabetesPedigreeFunction AS DiabetesPedigreeFunction,
        Age AS Age
    ) AS predicted
FROM diabetes_data
WHERE data_set = 'TEST';
```

This in-database scoring process eliminates the need to move data outside the Oracle DB for prediction.

Results:

PREGNANCIES	GLUCOSE	BLOODPRESSURE	SKINTHICKNESS	INSULIN	BMI	DIABETESPEDIGREEFUNCTION	AGE	ACTUAL	PREDICTED
2	197	70	45	543	30.5	0.158	53	1	1
4	110	92	0	0	37.6	0.191	30	0	0
7	100	0	0	0	30	0.484	32	1	0
9	102	76	37	0	32.9	0.665	46	1	0
7	159	64	0	0	27.4	0.294	40	0	1
0	146	82	0	0	40.5	1.781	44	0	1
7	114	66	0	0	32.8	0.258	42	1	0
0	109	88	30	0	32.5	0.855	38	1	0
13	126	90	0	0	43.4	0.583	42	1	0
4	129	86	20	270	35.1	0.231	23	0	0
7	83	78	26	71	29.3	0.767	36	0	0
8	112	72	0	0	23.6	0.84	58	0	0
5	77	82	41	42	35.8	0.156	35	0	0
7	114	64	0	0	27.4	0.732	34	1	0

PREGNANCIES	GLUCOSE	BLOODPRESSURE	SKINTHICKNESS	INSULIN	BMI	DIABETESPEDIGREEFUNCTION	AGE	ACTUAL	PREDICTED
1	167	74	17	144	23.4	0.447	33	1	1
10	111	70	27	0	27.5	0.141	40	1	0
6	165	68	26	168	33.6	0.631	49	0	1
1	144	82	46	180	46.1	0.335	46	1	1
10	94	72	18	0	23.1	0.595	56	0	0
3	102	44	20	94	30.8	0.4	26	0	0
0	179	50	36	159	37.8	0.455	22	1	1
10	162	84	0	0	27.7	0.182	54	0	1
1	99	58	10	0	25.4	0.551	21	0	0
6	195	70	0	0	30.9	0.328	31	1	1
2	56	56	28	45	24.2	0.332	22	0	0
0	95	64	39	105	44.6	0.366	22	0	0
2	121	70	32	95	39.1	0.886	23	0	0
7	142	90	24	480	30.4	0.128	43	1	1

PREGNANCIES	GLUCOSE	BLOODPRESSURE	SKINTHICKNESS	INSULIN	BMI	DIABETESPEDIGREEFUNCTION	AGE	ACTUAL	PREDICTED
4	127	88	11	155	34.5	0.598	28	0	0
4	83	86	19	0	29.3	0.317	34	0	0
1	106	76	0	0	37.5	0.197	26	0	0
3	180	64	35	70	34	0.271	26	0	1
1	103	80	11	82	19.4	0.491	22	0	0
5	99	74	27	0	29	0.203	32	0	0
1	128	48	34	194	40.5	0.613	24	1	0
9	112	82	24	0	28.2	1.282	50	1	0
10	129	62	36	0	41.2	0.441	38	1	0
0	104	64	23	116	27.8	0.454	23	0	0
0	146	70	0	0	37.9	0.334	28	1	1
7	161	86	0	0	30.4	0.165	47	1	1
2	108	80	0	0	27	0.259	52	1	0
11	138	74	26	144	36.1	0.557	50	1	0

PREGNANCIES	GLUCOSE	BLOODPRESSURE	SKINTHICKNESS	INSULIN	BMI	DIABETESPEDIGREEFUNCTION	AGE	ACTUAL	PREDICTED
0	119	0	0	0	32.4	0.141	24	1	0
2	108	64	0	0	30.8	0.158	21	0	0
7	97	76	32	91	40.9	0.871	32	1	0
4	112	78	40	0	39.4	0.236	38	0	0
2	174	88	37	120	44.5	0.646	24	1	1
11	120	80	37	150	42.3	0.785	48	1	0

2. Accuracy Calculation (SQL)

We measured accuracy directly in SQL:

```
%script
SELECT
    COUNT(*) AS total,
    SUM(CASE WHEN Outcome = PREDICTION(DIABETES_MODEL USING
        Pregnancies AS Pregnancies,
        Glucose AS Glucose,
        BloodPressure AS BloodPressure,
        SkinThickness AS SkinThickness,
        Insulin AS Insulin,
        BMI AS BMI,
        DiabetesPedigreeFunction AS DiabetesPedigreeFunction,
        Age AS Age
    ) THEN 1 ELSE 0 END) AS correct,
    ROUND(100 * SUM(CASE WHEN Outcome = PREDICTION(DIABETES_MODEL USING
        Pregnancies AS Pregnancies,
        Glucose AS Glucose,
        BloodPressure AS BloodPressure,
        SkinThickness AS SkinThickness,
        Insulin AS Insulin,
        BMI AS BMI,
        DiabetesPedigreeFunction AS DiabetesPedigreeFunction,
        Age AS Age
    ) THEN 1 ELSE 0 END) / COUNT(*), 2) AS accuracy_percent
FROM diabetes_data
WHERE data_set = 'TEST';
```

TOTAL	CORRECT	ACCURACY_PERCENT
48	28	58.33

This provides a clear measure of model effectiveness based solely on SQL logic.

6. Conclusion

This project demonstrates that PL/SQL, when used with Oracle Machine Learning (OML), is fully capable of handling real-world machine learning workflows:

- Data preprocessing
- Model training
- Evaluation
- Prediction

All operations are executed in-database, which optimizes performance, minimizes data leakage risk, and integrates naturally with existing Oracle-based applications. PL/SQL proves itself as a viable, modern tool even in the era of cloud AI services and microservices.

7. AI Tools Utilization

AI tools were used throughout the project to support different tasks:

- Understanding Concepts: Helped understand key ideas related to Oracle Machine Learning (OML) and how to use PL/SQL for in-database machine learning.
- Code Writing: Assisted in generating and improving PL/SQL code for data preparation, model creation, and evaluation.
- Error Handling: Helped interpret Oracle error messages and suggested fixes during development.
- Report Writing: Supported the structuring of this report and corrected language.