take-home-exercise-4 (Python)

Import notebook

```
# Import libraries
import pandas as pd
import mlflow
import mlflow.sklearn

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
```

```
f1_data = pd.read_csv("/dbfs/FileStore/tables/results-1.csv")
```

f1_data: pandas.core.frame.DataFrame = [resultId: int64, raceId: int64 ... 16 more fields]

```
# Select features and target for prediction
features = ['grid', 'constructorId']  # Features we use for prediction
target = 'positionOrder'              # What we want to predict

X = f1_data[features]
y = f1_data[target]

# Display the first few rows of features and target
X.head(), y.head()
```

X: pandas.core.frame.DataFrame = [grid: int64, constructorId: int64]

```
(   grid  constructorId
 0     1             1
 1     5             2
 2     7             3
 3    11             4
 4     3             1,
 0    1
 1    2
 2    3
 3    4
 4    5
 Name: positionOrder, dtype: int64)
```

```
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.25, random_state=42
)

# Display the shape of the resulting sets
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

X_test: pandas.core.frame.DataFrame = [grid: int64, constructorId: int64]

X_train: pandas.core.frame.DataFrame = [grid: int64, constructorId: int64]

```
((20069, 2), (6690, 2), (20069,), (6690,))
```

```
# Train a Logistic Regression model
with mlflow.start_run(run_name="Logistic Regression"):
    model1 = LogisticRegression(max_iter=1000)
    model1.fit(X_train, y_train)

    # Make predictions
    y_pred1 = model1.predict(X_test)

    # Calculate metrics
    acc1 = accuracy_score(y_test, y_pred1)
    prec1 = precision_score(y_test, y_pred1, average='weighted', zero_division=0)
    rec1 = recall_score(y_test, y_pred1, average='weighted', zero_division=0)
    f1_1 = f1_score(y_test, y_pred1, average='weighted', zero_division=0)

    # Log metrics
    mlflow.log_metric("accuracy", acc1)
    mlflow.log_metric("precision", prec1)
    mlflow.log_metric("recall", rec1)
    mlflow.log_metric("f1_score", f1_1)

    # Log model
    mlflow.sklearn.log_model(model1, "logistic_regression_model")
```

```
/databricks/python/lib/python3.12/site-packages/sklearn/linear_model/_logistic.py:469: ConvergenceWarning: lbfgs faile
d to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://scikit-learn.org/stable/modules/preprocessing.
html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression (https://scikit-learn.org/stable/mod
ules/linear_model.html#logistic-regression)
  n_iter_i = _check_optimize_result(
2025/04/29 01:49:35 WARNING mlflow.models.model: Model logged without a signature. Signatures will be required for upc
oming model registry features as they validate model inputs and denote the expected schema of model outputs. Please vi
sit https://www.mlflow.org/docs/2.15.1/models.html#set-signature-on-logged-model (https://www.mlflow.org/docs/2.15.1/m
odels.html#set-signature-on-logged-model) for instructions on setting a model signature on your logged model.


2025/04/29 01:49:36 WARNING mlflow.models.model: Input example should be provided to infer model signature if the mode
l signature is not provided when logging the model.
2025/04/29 01:49:36 INFO mlflow.tracking._tracking_service.client: 🏃 View run Logistic Regression at: columbiau-gr506
9.cloud.databricks.com/ml/experiments/743188476171287/runs/a5dc4cdbc16b4492ad6f0ea766b88b3d.
2025/04/29 01:49:36 INFO mlflow.tracking._tracking_service.client: 🧪 View experiment at: columbiau-gr5069.cloud.datab
ricks.com/ml/experiments/743188476171287.
```

```
# Train a Decision Tree Classifier
with mlflow.start_run(run_name="Decision Tree"):
    model2 = DecisionTreeClassifier(max_depth=5, random_state=42)
    model2.fit(X_train, y_train)

    # Make predictions
    y_pred2 = model2.predict(X_test)

    # Calculate metrics
    acc2 = accuracy_score(y_test, y_pred2)
    prec2 = precision_score(y_test, y_pred2, average='weighted', zero_division=0)
    rec2 = recall_score(y_test, y_pred2, average='weighted', zero_division=0)
    f1_2 = f1_score(y_test, y_pred2, average='weighted', zero_division=0)

    # Log metrics
    mlflow.log_metric("accuracy", acc2)
    mlflow.log_metric("precision", prec2)
    mlflow.log_metric("recall", rec2)
    mlflow.log_metric("f1_score", f1_2)

    # Log model
    mlflow.sklearn.log_model(model2, "decision_tree_model")
```

```
2025/04/29 01:50:35 WARNING mlflow.models.model: Model logged without a signature. Signatures will be required for upc
oming model registry features as they validate model inputs and denote the expected schema of model outputs. Please vi
sit https://www.mlflow.org/docs/2.15.1/models.html#set-signature-on-logged-model (https://www.mlflow.org/docs/2.15.1/m
odels.html#set-signature-on-logged-model) for instructions on setting a model signature on your logged model.


2025/04/29 01:50:36 WARNING mlflow.models.model: Input example should be provided to infer model signature if the mode
l signature is not provided when logging the model.
2025/04/29 01:50:36 INFO mlflow.tracking._tracking_service.client: 🏃 View run Decision Tree at: columbiau-gr5069.clou
d.databricks.com/ml/experiments/743188476171287/runs/66d622ab35f84e3b805bf245d98f5081.
2025/04/29 01:50:36 INFO mlflow.tracking._tracking_service.client: 🧪 View experiment at: columbiau-gr5069.cloud.datab
ricks.com/ml/experiments/743188476171287.
```

```
# Create a database if not already exist
spark.sql("CREATE DATABASE IF NOT EXISTS student_db")
```

```
DataFrame[]
```

```
# Use the database
spark.sql("USE student_db")
```

```
DataFrame[]
```

```
spark.sql("""
CREATE TABLE IF NOT EXISTS model1_predictions (
    id INT,
    prediction DOUBLE
)
""")
```

```
DataFrame[]
```

```
spark.sql("""
CREATE TABLE IF NOT EXISTS model2_predictions (
    id INT,
    prediction DOUBLE
)
""")
```

```
DataFrame[]
```

```
# Prepare predictions from Logistic Regression model
import numpy as np

predictions_df1 = pd.DataFrame({
    'id': np.arange(len(y_pred)),  # <-- fix id to be 0,1,2,...
    'prediction': y_pred
})

# Convert to Spark DataFrame
spark_df1 = spark.createDataFrame(predictions_df1)

# Save into the first table
spark_df1.write.mode("overwrite").saveAsTable("model1_predictions")
```

- predictions_df1: pandas.core.frame.DataFrame = [id: int64, prediction: int64]
- spark_df1: pyspark.sql.dataframe.DataFrame = [id: long, prediction: long]

```
# Prepare predictions from Decision Tree model
predictions_df2 = pd.DataFrame({
    'id': np.arange(len(y_pred2)),  # <-- generate id from 0,1,2,...
    'prediction': y_pred2
})

# Convert to Spark DataFrame
spark_df2 = spark.createDataFrame(predictions_df2)

# Save into the second table
spark_df2.write.mode("overwrite").saveAsTable("model2_predictions")
```

- predictions_df2: pandas.core.frame.DataFrame = [id: int64, prediction: int64]
- spark_df2: pyspark.sql.dataframe.DataFrame = [id: long, prediction: long]

```
+---+----------+
| id|prediction|
+---+----------+
|  0|         0|
|  1|         0|
+---+----------+

+---+----------+
| id|prediction|
+---+----------+
|  0|         3|
|  1|         1|
|  2|        15|
|  3|         5|
```

```
|  4|         15|
+---+----------+
```