Cmd 1

Cmd 2

Logistic Regression Classifier

Module 4 Assignment

This final assignment is broken up into 2 parts:

1. Completing this Logistic Regression Classifier notebook
   - Submitting question answers to Coursera
   - Uploading notebook to Coursera for peer reviewing
2. Answering 3 free response questions on Coursera platform

☆ In this notebook you:

- Preprocess data for use in a machine learning model
- Step through creating a sklearn logistic regression model for classification
- Predict the `Call_Type_Group` for incidents in a SQL table

For each **bold** question, input its answer in Coursera.

Cmd 3

```
%run ../Includes/Classroom-Setup
```

```
Command took 4.37 minutes -- by tjamesbu@gmail.com at 5/25/2021, 6:25:13 PM on My Cluster
```

```
WARNING: This curriculum was written for DBR 5.5 ML. Please create a new cluster that uses the runtime DBR 5.5 ML
```

OK

Cmd 4

Load the `/mnt/davis/fire-calls/fire-calls-clean.parquet` data as `fireCallsClean` table.

Cmd 5

```
-- TODO
USE DATABRICKS;

CREATE TABLE IF NOT EXISTS fireCallsClean
USING parquet
OPTIONS (
    path "/mnt/davis/fire-calls/fire-calls-clean.parquet"
)
-- FILL IN
```

▸ (1) Spark Jobs

OK

```
Command took 6.38 seconds -- by tjamesbu@gmail.com at 5/25/2021, 6:29:46 PM on My Cluster
```

Cmd 6

Check that your data is loaded in properly.

Cmd 7

```
SELECT * FROM fireCallsClean LIMIT 10
```

▸ (1) Spark Jobs

	Call_Number	Unit_ID	Incident_Number	Call_Type	Call_Date	Watch_Date	Received_DtTm	Entry_DtTm	Dispatch_DtTm	Response_DtTm
1	141600888	65	14055109	Traffic Collision	06/09/2014	06/09/2014	06/09/2014 09:35:33 AM	06/09/2014 09:36:46 AM	06/09/2014 09:37:43 AM	06/09/2014 09:37:55 AM
2	162743687	E01	16108733	Medical Incident	09/30/2016	09/30/2016	09/30/2016 08:05:57 PM	09/30/2016 08:07:53 PM	09/30/2016 08:08:14 PM	09/30/2016 08:08:23 PM
3	102210202	75	10069623	Medical Incident	08/09/2010	08/09/2010	08/09/2010 01:28:40 PM	08/09/2010 01:30:47 PM	08/09/2010 01:31:53 PM	08/09/2010 01:32:29 PM
4	160681260	E42	16027085	Medical Incident	03/08/2016	03/08/2016	03/08/2016 10:42:26 AM	03/08/2016 10:42:57 AM	03/08/2016 10:43:20 AM	03/08/2016 10:50:11 AM
5	113200298	E18	11106370	Structure Fire	11/16/2011	11/16/2011	11/16/2011 05:13:01 PM	11/16/2011 05:13:01 PM	11/16/2011 05:13:09 PM	11/16/2011 05:13:54 PM
6	162584030	KM07	16101787	Medical Incident	09/14/2016	09/14/2016	09/14/2016 08:41:50 PM	09/14/2016 08:41:50 PM	09/14/2016 08:43:07 PM	09/14/2016 08:43:34 PM
7	133150239	KM09	13107112	Medical Incident	11/11/2013	11/11/2013	11/11/2013 01:41:24 PM	11/11/2013 01:44:26 PM	11/11/2013 01:44:41 PM	11/11/2013 01:45:10 PM

Showing all 10 rows.

```
Command took 9.44 seconds -- by tjamesbu@gmail.com at 5/25/2021, 6:30:00 PM on My Cluster
```

Cmd 8

By the end of this assignment, we would like to train a logistic regression model to predict 2 of the most common `Call_Type_Group` given information from the rest of the table.

Cmd 9

Write a query to see what the different `Call_Type_Group` values are and their respective counts.

Question 1

How many calls of `Call_Type_Group` "Fire"?

Cmd 10

```
-- TODO
SELECT
  Call_Type,
  COUNT(*) AS `num`
FROM
  fireCallsClean
GROUP BY
  Call_Type
ORDER BY num DESC
```

(2) Spark Jobs

	Call_Type	num
1	Medical Incident	300334
2	Alarms	33875
3	Structure Fire	29446
4	Traffic Collision	13470
5	Citizen Assist / Service Call	11221
6	Other	8932
7	Outside Fire	7569

Showing all 31 rows.

Command took 2.69 seconds -- by tjamesbu@gmail.com at 5/25/2021, 6:40:41 PM on My Cluster

Cmd 11

Let's drop all the rows where `Call_Type_Group = null`. Since we don't have a lot of `Call_Type_Group` with the value `Alarm` and `Fire`, we will also drop these calls from the table. Call this new temporary view `fireCallsGroupCleaned`.

Cmd 12

```
-- TODO
CREATE OR REPLACE VIEW fireCallsGroupCleaned
AS (
  SELECT *
  FROM fireCallsClean
  WHERE Call_Type_Group IS NOT NULL AND Call_Type_Group <> "Alarm" AND Call_Type_Group <> "Fire"
)
```

OK

Command took 0.45 seconds -- by tjamesbu@gmail.com at 5/25/2021, 6:44:48 PM on My Cluster

Cmd 13

Check that every entry in `fireCallsGroupCleaned` has a `Call_Type_Group` of either `Potentially Life-Threatening` or `Non Life-threatening`.

Cmd 14

```
-- TODO
SELECT
  Call_Type_Group
FROM fireCallsGroupCleaned
GROUP BY
  Call_Type_Group
```

(2) Spark Jobs

	Call_Type_Group
1	Potentially Life-Threatening
2	Non Life-threatening

Showing all 2 rows.

Command took 3.54 seconds -- by tjamesbu@gmail.com at 5/25/2021, 6:45:30 PM on My Cluster

Cmd 15

Question 2

How many rows are in `fireCallsGroupCleaned`?

Cmd 16

We probably don't need all the columns of `fireCallsGroupCleaned` to make our prediction. Select the following columns from `fireCallsGroupCleaned` and create a view called `fireCallsDF` so we can access this table in Python:

- "Call_Type"
- "Fire_Prevention_District"
- "Neighborhooods_-_Analysis_Boundaries"
- "Number_of_Alarms"
- "Original_Priority"
- "Unit_Type"
- "Battalion"
- "Call_Type_Group"

Cmd 17

```
-- TODO
CREATE OR REPLACE VIEW fireCallsDF
AS (
  SELECT
    Call_Type
    , Fire_Prevention_District
```

```
    , `Neighborhooods_-_Analysis_Boundaries`
    , Number_of_Alarms
    , Original_Priority
    , Unit_Type
    , Battalion
    , Call_Type_Group
  FROM fireCallsGroupCleaned
)
```

OK

Command took 0.28 seconds -- by tjamesbu@gmail.com at 5/25/2021, 6:46:11 PM on My Cluster

Cmd 18

Fill in the string SQL statement to load the `fireCallsDF` table you just created into python.

Cmd 19

```
%python
# TODO
df = sql("SELECT * FROM fireCallsDF")
display(df)
```

(1) Spark Jobs

df: pyspark.sql.dataframe.DataFrame = [Call_Type: string, Fire_Prevention_District: string ... 6 more fields]

	Call_Type	Fire_Prevention_District	Neighborhooods_-_Analysis_Boundaries	Number_of_Alarms	Original_Priority	Unit_Type	Battalion	Call_Type_Group
1	Traffic Collision	10	Bayview Hunters Point	1	2	MEDIC	B10	Non Life-threatening
2	Medical Incident	2	South of Market	1	2	ENGINE	B02	Non Life-threatening
3	Medical Incident	10	Portola	1	3	ENGINE	B10	Potentially Life-Threatening
4	Medical Incident	3	South of Market	1	2	PRIVATE	B03	Non Life-threatening
5	Medical Incident	2	Mission	1	1	PRIVATE	B02	Non Life-threatening
6	Medical Incident	8	Sunset/Parkside	1	2	MEDIC	B08	Potentially Life-Threatening
7	Medical Incident	6	Castro/Upper Market	1	2	ENGINE	B05	Potentially Life-Threatening

Showing the first 1000 rows.

Command took 1.68 seconds -- by tjamesbu@gmail.com at 5/25/2021, 7:08:05 PM on My Cluster

Cmd 20

Creating a Logistic Regression Model in Sklearn

Cmd 21

First we will convert the Spark DataFrame to pandas so we can use sklearn to preprocess the data into numbers so that it is compatible with the logistic regression algorithm with a LabelEncoder.

Then we'll perform a train test split on our pandas DataFrame. Remember that the column we are trying to predict is the `Call_Type_Group`.

Cmd 22

```
%python
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder

pdDF = df.toPandas()
le = LabelEncoder()
numerical_pdDF = pdDF.apply(le.fit_transform)

X = numerical_pdDF.drop("Call_Type_Group", axis=1)
y = numerical_pdDF["Call_Type_Group"].values
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

(1) Spark Jobs

1

Command took 7.03 seconds -- by tjamesbu@gmail.com at 5/25/2021, 6:47:32 PM on My Cluster

Cmd 23

Look at our training data `X_train` which should only have numerical values now.

Cmd 24

```
%python
display(X_train)
```

(1) Spark Jobs

	Call_Type	Fire_Prevention_District	Neighborhooods_-_Analysis_Boundaries	Number_of_Alarms	Original_Priority	Unit_Type	Battalion
1	0	7	10	0	1	2	6
2	0	2	9	0	1	5	1
3	0	2	29	0	2	2	9
4	0	3	5	0	1	4	0
5	0	1	1	0	1	2	5
6	0	4	36	0	3	5	3
7	2	10	37	0	2	2	2

Showing the first 1000 rows.

Command took 0.50 seconds -- by tjamesbu@gmail.com at 5/25/2021, 6:47:47 PM on My Cluster

Cmd 25

We'll create a pipeline with 2 steps.

1. One Hot Encoding: Converts our features into vectorized features by creating a dummy column for each value in that category.
2. Logistic Regression model: Although the name includes "regression", it is used for classification by predicting the probability that the `Call Type Group` is one label and not the other.

Cmd 26

```python
%python
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import OneHotEncoder
from sklearn.pipeline import Pipeline

ohe = ("ohe", OneHotEncoder(handle_unknown="ignore"))
lr = ("lr", LogisticRegression())

pipeline = Pipeline(steps = [ohe, lr]).fit(X_train, y_train)
y_pred = pipeline.predict(X_test)
```

```
/databricks/python/lib/python3.8/site-packages/sklearn/linear_model/_logistic.py:762: ConvergenceWarning: lbfgs failed 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
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

Command took 1.57 seconds -- by tjamesbu@gmail.com at 5/25/2021, 6:47:58 PM on My Cluster

Cmd 27

Run the following cell to see how well our model performed on test data (data that wasn't used to train the model)!

Cmd 28

```python
%python
from sklearn.metrics import accuracy_score
print(f"Accuracy of model: {accuracy_score(y_pred, y_test)}")
```

```
Accuracy of model: 0.8177347242921014
```

Command took 0.04 seconds -- by tjamesbu@gmail.com at 5/25/2021, 6:49:01 PM on My Cluster

Cmd 29

Question 3

What is the accuracy of our model on test data? Round to the nearest percent.

Cmd 30

Save pipeline (with both stages) to disk.

Cmd 31

```python
%python
%pip install mlflow
%dbutils.library.installPyPI("mlflow")

```

```
Python interpreter will be restarted.
Requirement already satisfied: mlflow in /local_disk0/.ephemeral_nfs/envs/pythonEnv-726d3116-0907-40c4-b738-6a1b4bdde8b5/lib/python3.8/site-packages (1.17.0)
Requirement already satisfied: cloudpickle in /local_disk0/.ephemeral_nfs/envs/pythonEnv-726d3116-0907-40c4-b738-6a1b4bdde8b5/lib/python3.8/site-packages (from mlflow) (1.6.0)
Requirement already satisfied: querystring-parser in /local_disk0/.ephemeral_nfs/envs/pythonEnv-726d3116-0907-40c4-b738-6a1b4bdde8b5/lib/python3.8/site-packages (from mlflow) (1.2.4)
Requirement already satisfied: pandas in /databricks/python3/lib/python3.8/site-packages (from mlflow) (1.1.3)
Requirement already satisfied: sqlparse>=0.3.1 in /local_disk0/.ephemeral_nfs/envs/pythonEnv-726d3116-0907-40c4-b738-6a1b4bdde8b5/lib/python3.8/site-packages (from mlflow) (0.4.1)
Requirement already satisfied: requests>=2.17.3 in /databricks/python3/lib/python3.8/site-packages (from mlflow) (2.24.0)
Requirement already satisfied: Flask in /local_disk0/.ephemeral_nfs/envs/pythonEnv-726d3116-0907-40c4-b738-6a1b4bdde8b5/lib/python3.8/site-packages (from mlflow) (2.0.1)
Requirement already satisfied: sqlalchemy in /local_disk0/.ephemeral_nfs/envs/pythonEnv-726d3116-0907-40c4-b738-6a1b4bdde8b5/lib/python3.8/site-packages (from mlflow) (1.4.15)
Requirement already satisfied: gunicorn; platform_system != "Windows" in /local_disk0/.ephemeral_nfs/envs/pythonEnv-726d3116-0907-40c4-b738-6a1b4bdde8b5/lib/python3.8/site-packages (f
rom mlflow) (20.1.0)
Requirement already satisfied: gitpython>=2.1.0 in /local_disk0/.ephemeral_nfs/envs/pythonEnv-726d3116-0907-40c4-b738-6a1b4bdde8b5/lib/python3.8/site-packages (from mlflow) (3.1.17)
Requirement already satisfied: docker>=4.0.0 in /local_disk0/.ephemeral_nfs/envs/pythonEnv-726d3116-0907-40c4-b738-6a1b4bdde8b5/lib/python3.8/site-packages (from mlflow) (5.0.0)
Requirement already satisfied: entrypoints in /databricks/python3/lib/python3.8/site-packages (from mlflow) (0.3)
Requirement already satisfied: protobuf>=3.6.0 in /local_disk0/.ephemeral_nfs/envs/pythonEnv-726d3116-0907-40c4-b738-6a1b4bdde8b5/lib/python3.8/site-packages (from mlflow) (3.17.1)
Requirement already satisfied: numpy in /databricks/python3/lib/python3.8/site-packages (from mlflow) (1.19.2)
Requirement already satisfied: databricks-cli>=0.8.7 in /local_disk0/.ephemeral_nfs/envs/pythonEnv-726d3116-0907-40c4-b738-6a1b4bdde8b5/lib/python3.8/site-packages (from mlflow) (0.1
4.3)
Requirement already satisfied: click>=7.0 in /local_disk0/.ephemeral_nfs/envs/pythonEnv-726d3116-0907-40c4-b738-6a1b4bdde8b5/lib/python3.8/site-packages (from mlflow) (8.0.1)
Requirement already satisfied: pytz in /databricks/python3/lib/python3.8/site-packages (from mlflow) (2020.5)
Requirement already satisfied: prometheus-flask-exporter in /local_disk0/.ephemeral_nfs/envs/pythonEnv-726d3116-0907-40c4-b738-6a1b4bdde8b5/lib/python3.8/site-packages (from mlflow)
```

Command took 4.54 seconds -- by tjamesbu@gmail.com at 5/25/2021, 6:54:15 PM on My Cluster

Cmd 32

```python
import mlflow
from mlflow.sklearn import save_model

model_path = "/dbfs/" + username + "/Call_Type_Group_lr"
dbutils.fs.rm(username + "/Call_Type_Group_lr", recurse=True)
save_model(pipeline, model_path)
```

```
Error in SQL statement: ParseException:
missing 'TABLE' at 'mlflow'(line 1, pos 7)

== SQL ==
import mlflow
-------^^^
from mlflow.sklearn import save_model

model_path = "/dbfs/" + username + "/Call_Type_Group_lr"
dbutils.fs.rm(username + "/Call_Type_Group_lr", recurse=True)
save_model(pipeline, model_path)
```

Command took 0.06 seconds -- by tjamesbu@gmail.com at 5/25/2021, 6:54:24 PM on My Cluster

Cmd 33

UDF

Cmd 34

Now that we have created and trained a machine learning pipeline, we will use MLflow to register the `.predict` function of the sklearn pipeline as a UDF which we can use later to apply in parallel. Now we can refer to this with the name `predictUDF` in SQL.

Cmd 35

```python
%python
import mlflow
from mlflow.pyfunc import spark_udf

predict = spark_udf(spark, model_path, result_type="int")
spark.udf.register("predictUDF", predict)
```

```
NameError: name 'model_path' is not defined
```

Command took 0.84 seconds -- by tjamesbu@gmail.com at 5/25/2021, 6:52:38 PM on My Cluster

Cmd 36

Create a view called `testTable` of our test data `X_test` so that we can see this table in SQL.

Cmd 37

```python
%python
spark_df = spark.createDataFrame(X_test)
spark_df.createOrReplaceTempView("testTable")
```

Cmd 38

Create a table called `predictions` using the `predictUDF` function we registered beforehand. Apply the `predictUDF` to every row of `testTable` in parallel so that each row of `testTable` has a `Call_Type_Group` prediction.

Cmd 39

```sql
-- TODO
USE DATABRICKS;
DROP TABLE IF EXISTS predictions;

CREATE TEMPORARY VIEW predictions
AS (
  SELECT
    CAST(predictUDF(Call_Type
                    , Fire_Prevention_District
                    , `Neighborhooods_-_Analysis_Boundaries`
                    , Number_of_Alarms
                    , Original_Priority
                    , Unit_Type
                    , BattalioN) AS double) AS predictions
     , *
   FROM testTable
);
-- FILL_IN
```

Cmd 40

Now take a look at the table and see what your model predicted for each call entry!

Cmd 41

```sql
SELECT * FROM predictions LIMIT 10
```

Cmd 42

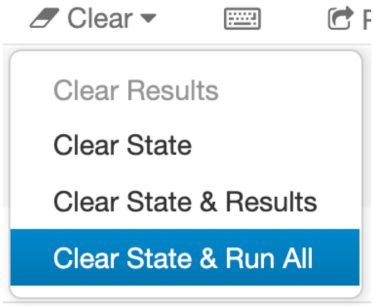
Question 4:

What 2 values are in the `prediction` column?

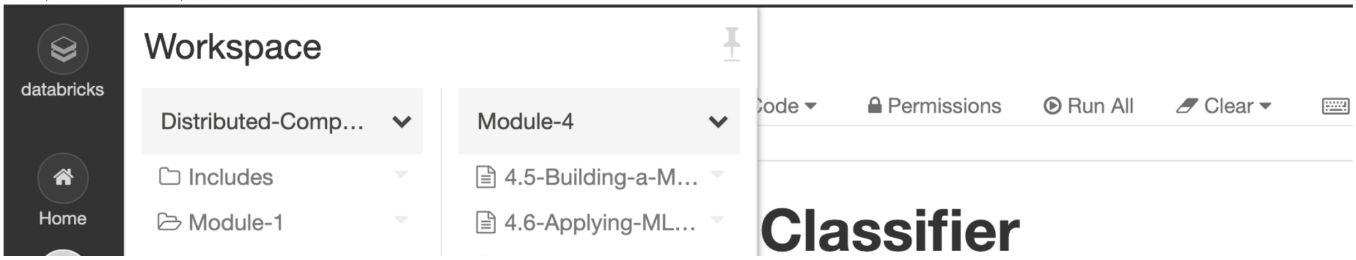
Cmd 43

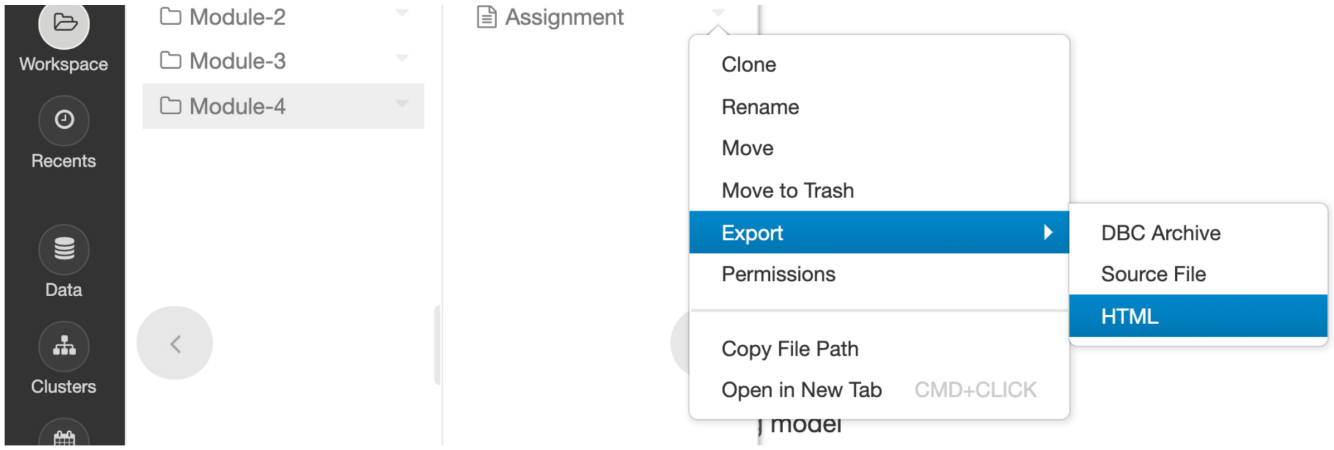
Congrats on finishing your last assignment notebook!

Now you will have to upload this notebook to Coursera for peer reviewing.

1. Make sure that all your code will run without errors
   - Check this by clicking the "Clear State & Run All" dropdown option at the top of your notebook
   - 

2. Click on the "Workspace" icon on the side bar
3. Next to the notebook you're working in right now, click on the dropdown arrow
4. In the dropdown, click on "Export" then "HTML"





5. On the Coursera platform, upload this HTML file to Week 4's Peer Review Assignment

Go back onto the Coursera platform for the free response portion of this assignment and for instructions on how to review your peer's work.

Cmd 44

Shift+Enter to run