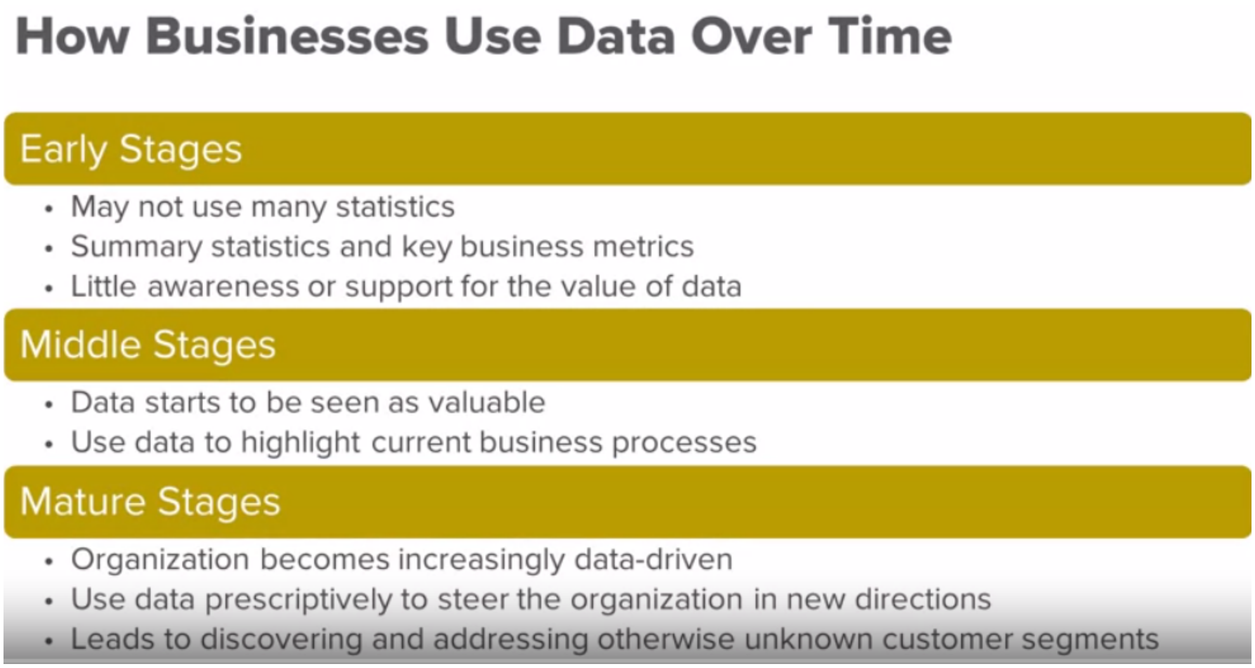
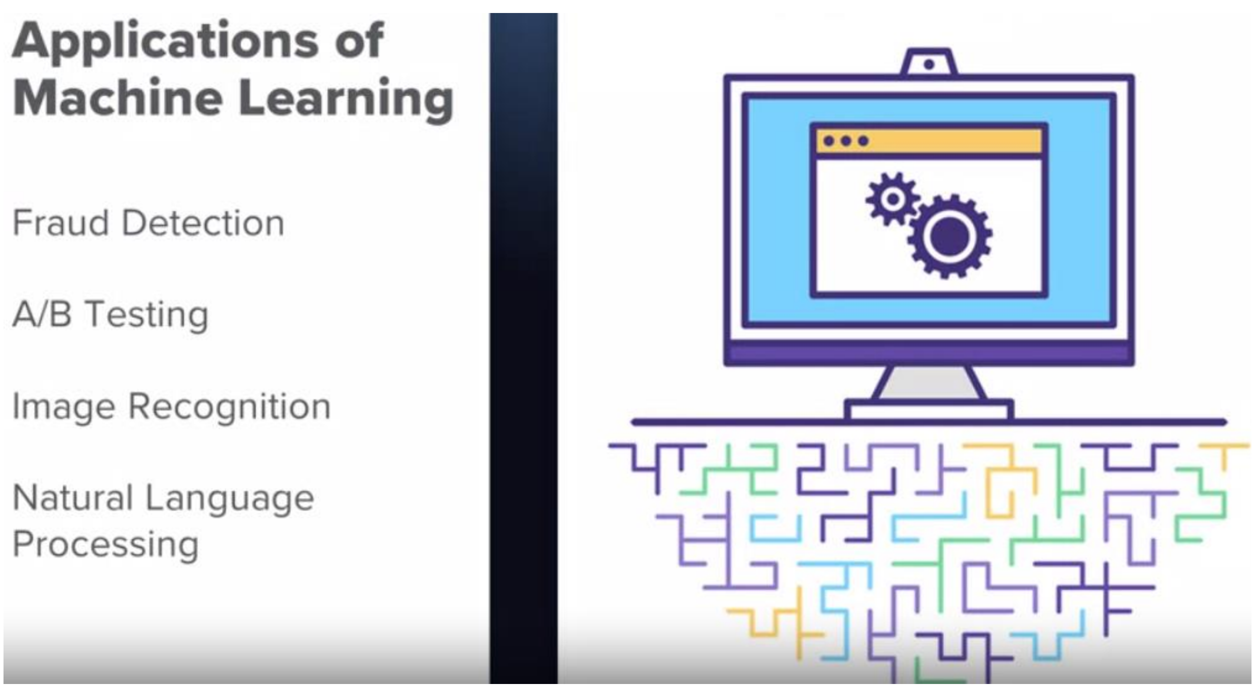
University of California Davis and Databricks collaboration

Distributed Computing with Spark SQL

104 Machine Learning Applications of Spark and Logistic Regression Classifier





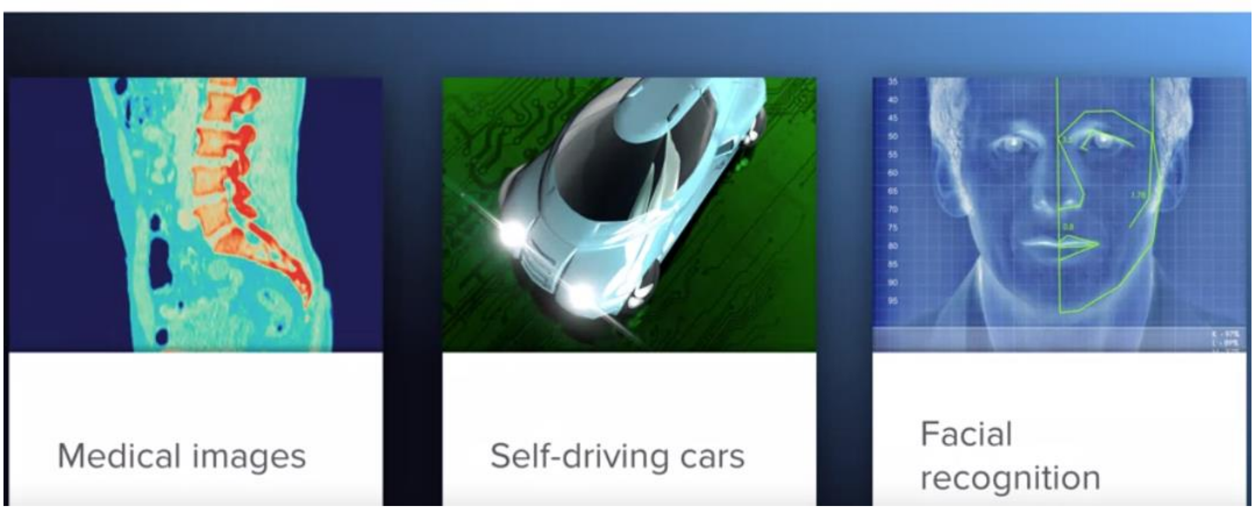
Fraud Detection in Real-Time

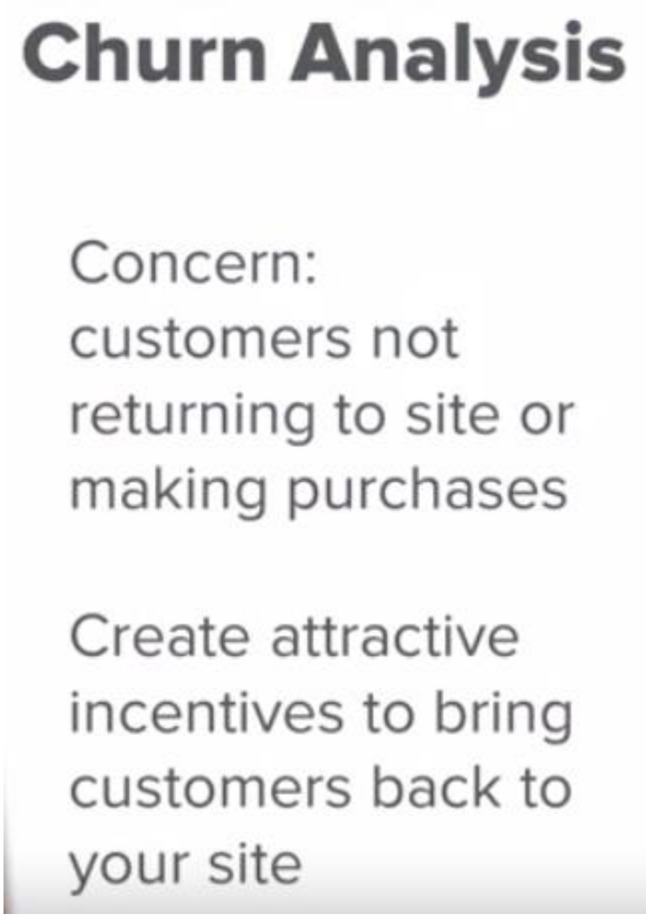
Natural Language Processing

Classifying medical records

Chatbot sentiment analysis

Image Processing




Churn Analysis
Concern:
customers not
returning to site or
making purchases
Create attractive
incentives to bring
customers back to
your site

How Can We Approach Churn Analysis as a Data Problem?

Define the Problem: What is Churn?

No purchases within a time-frame

No web visits for a certain period of time

Asking Predictive Questions

More visits to web site, the less likelihood of churn

Long-term customers, less likely to churn

Churn Analysis

Machine learning would use past user data to find patterns between variables

Translate Business Problems Into Data Problems

Can we predict future user activity based on past user activity?

Look for Strategically Significant Correlations

Types of Machine Learning

Supervised

Unsupervised

Reinforcement

Semi-supervised

Supervised Machine Learning

Labeled data points

Task is to predict the label

Classification Tasks

Predicts a discrete set of categories

- Binary classification
- Multiclass classification

Regression Tasks

Predict a continuous value

Financial forecasting

Unbounded number rather than a category

Applying Machine Learning – Fire Call Dataset

Predict response times using various input features

- Type of call
- Location of the call

Supervised machine learning – regression problem

- Predicting a continuous variable: response time delay

Calculating Error

- Predict response times
- Look at the difference between predicted and true values

$$Error = (y_i - \hat{y}_i)$$

Root Mean Squared Error

The lower the RMSE, the better

Compute the Sum of the Squared Error

The lower the RMSE, the better

$$SE = (y_i - \hat{y}_i)^2$$

$$SSE = \sum_{i=1}^{n} (y_i - \widehat{y_i})^2$$

$$MSE = \frac{1}{n} \sum_{i=1}^{n} (y_i - \widehat{y_i})^2$$

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (y_i - \widehat{y}_i)^2}$$

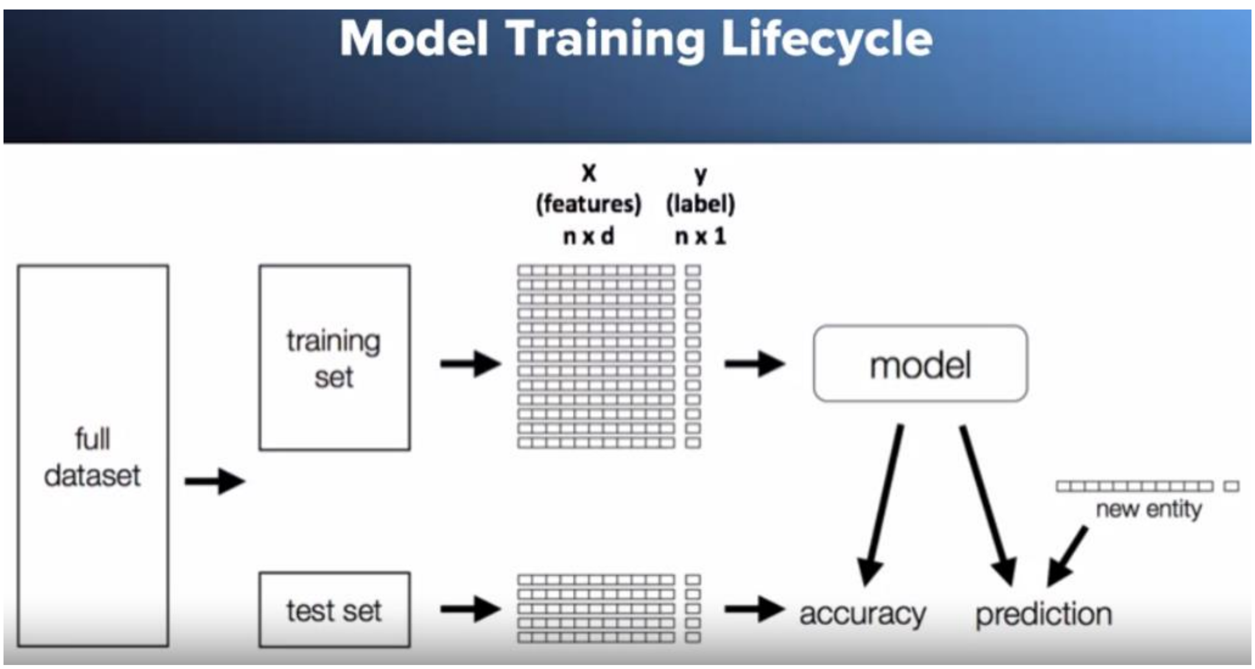
Applying Machine Learning – Fire Call Dataset

What would an RSME of 10 minutes mean?

Our predictions are off by 10 minutes in either direction from the true value

RMSE is dependent on the scale of your data

If we change our unit of measure from minutes to seconds, our RMSE would be much larger



Opting for Interpretable Models Over Accurate Models

Algorithm predicts 80% success

Procedure fails, don't blame the algorithm

You must understand why the algorithm made the prediction

Interpretable

Linear regression

Decision trees

Accurate

Neural networks

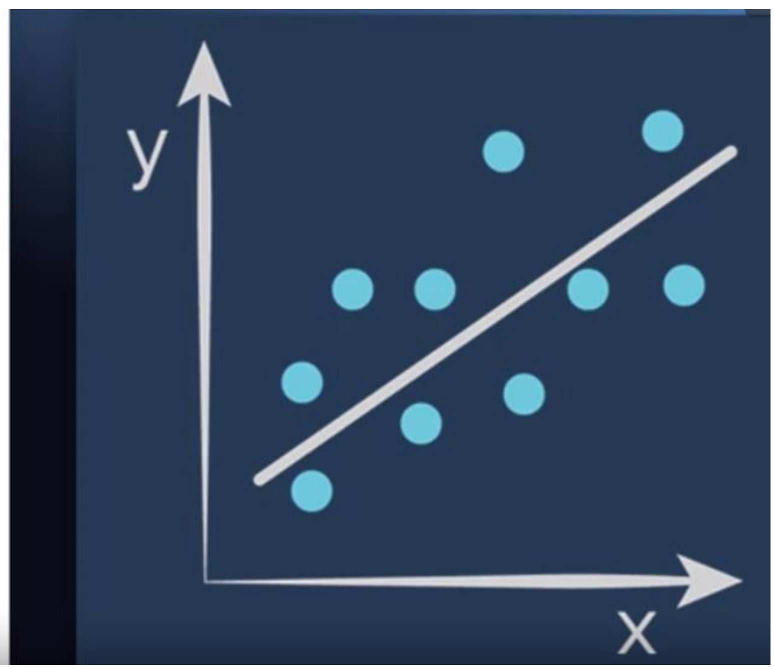
Linear Regression

Goal: Find line of best fit

$$y \approx \hat{y} = w_0 + w_1 x + \in$$

X: feature

y: label



Assumptions of Linear Regression

There is a linear relationship between input features and the output

Multivariate Regression

Use of linear regression with multiple variables

$$\hat{y} = \beta_0 + \beta_1 X_1 + \beta_2 X_2 \dots + \beta_p X_p$$

p = total number of features in the dataset

Interpreting Linear Regression

A highly interpretable model

Examine coefficients to see how a prediction is made

E.g.: If the coefficient for number of calls is -0.5 – then the response time decreases by half a minute for every additional call received

4.5-Building-a-Machine-Learning-Model (SQL)

My First Cluster | File | Edit | View: Standard | Permissions | Run All | Clear

Building a Machine Learning Model

Module 4, Lesson 5

In this lesson you:

- Build a Machine Learning model using scikit-learn
- Predict the response time to an incident given different features

Cmd 3

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

Command took 50.01 seconds -- by SWCPROPERTY@GMAIL.COM at 6/30/2021, 9:46:12 PM on My First 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 Data

We are going to build a model on a subset of our data.

Cmd 5

```sql
USE DATABRICKS;

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

(1) Spark Jobs

OK

Command took 1.62 seconds -- by SWCPROPERTY@GMAIL.COM at 6/30/2021, 9:47:10 PM on My First Cluster

Timestamp

Let's convert our `Response_DtTm` and `Received_DtTm` into timestamp types.

Cmd 7

```
CREATE OR REPLACE VIEW time AS (
  SELECT *, unix_timestamp(Response_DtTm, "MM/dd/yyyy hh:mm:ss a") as ResponseTime,
            unix_timestamp(Received_DtTm, "MM/dd/yyyy hh:mm:ss a") as ReceivedTime
  FROM fireCallsClean
)
```

OK

Command took 1.57 seconds -- by SWCPROPERTY@GMAIL.COM at 6/30/2021, 9:47:19 PM on My First Cluster

Cmd 8

Time Delay

Now that we have our `Response_DtTm` and `Received_DtTm` as timestamp types, we can compute the difference in minutes between the two.

Cmd 9

```
CREATE OR REPLACE VIEW timeDelay AS (
  SELECT *, (ResponseTime - ReceivedTime)/60 as timeDelay
  FROM time
)
```

OK

Command took 0.66 seconds -- by SWCPROPERTY@GMAIL.COM at 6/30/2021, 9:47:24 PM on My First Cluster

Cmd 10

Uh oh! We have some records with a negative time delay, and some with very extreme values. We will filter out those records.

Cmd 11

```
SELECT timeDelay, Call_Type, Fire_Prevention_District, `Neighborhooods_-_Analysis_Boundaries`, Number_of_Alarms, Original_Priority, Unit_Type
FROM timeDelay
WHERE timeDelay < 0
```

	timeDelay	Call_Type	Fire_Prevention_District	Neighborhooods_-_Analysis_Boundaries	Number_of_Alarms	Original_Priority	Unit_Type
1	-58.233333333333334	Structure Fire	5	Hayes Valley	1	3	ENGINE
2	-49.333333333333336	Medical Incident	1	Financial District/South Beach	1	3	TRUCK
3	-56.016666666666666	Medical Incident	2	Potrero Hill	1	3	ENGINE
4	-57.416666666666664	Structure Fire	4	Tenderloin	1	3	TRUCK
5	-56.06666666666667	Medical Incident	2	Tenderloin	1	1	MEDIC
6	-1.2	Alarms	3	Mission Bay	1	3	ENGINE

Showing all 6 rows.

1

Command took 14.88 seconds -- by SWCPROPERTY@GMAIL.COM at 6/30/2021, 9:47:28 PM on My First Cluster

Cmd 12

Great! Our data is prepped and ready to be used to build a model!

Cmd 13

```
SELECT timeDelay, Call_Type, Fire_Prevention_District, `Neighborhooods_-_Analysis_Boundaries`, Number_of_Alarms, Original_Priority, Unit_Type
FROM timeDelay
WHERE timeDelay < 15 AND timeDelay > 0
```

(1) Spark Jobs

	timeDelay	Call_Type	Fire_Prevention_District	Neighborhooods_-_Analysis_Boundaries	Number_of_Alarms	Original_Priority	Unit_Type
1	2.3666666666666667	Traffic Collision	10	Bayview Hunters Point	1	2	MEDIC
2	2.433333333333333	Medical Incident	2	South of Market	1	2	ENGINE
3	3.816666666666667	Medical Incident	2	Mission	1	1	MEDIC
4	7.75	Medical Incident	10	Portola	1	3	ENGINE
5	0.8833333333333333	Structure Fire	8	Sunset/Parkside	1	3	ENGINE
6	1.7333333333333334	Medical Incident	3	South of Market	1	2	PRIVATE
7	3.7666666666666666	Medical Incident	2	Mission	1	1	PRIVATE

Truncated results, showing first 1000 rows.

Command took 2.29 seconds -- by SWCPROPERTY@GMAIL.COM at 6/30/2021, 9:47:45 PM on My First Cluster

Convert to Pandas DataFrame

We are going to convert our Spark DataFrame to a Pandas DataFrame to build a scikit-learn model. Although we could use SparkML to train models, a lot of data scientists start by building their models using Pandas and Scikit-Learn.

We will also enable Apache Arrow for faster transfer of data from Spark DataFrames to Pandas DataFrames.

Cmd 15

```
%python
spark.conf.set("spark.sql.execution.arrow.enabled", "true")

pdDF = sql("""SELECT timeDelay, Call_Type, Fire_Prevention_District, `Neighborhoods_-_Analysis_Boundaries`, Number_of_Alarms, Original_Priority, Unit_Type
            FROM timeDelay
            WHERE timeDelay < 15 AND timeDelay > 0""").toPandas()
```

▸ (1) Spark Jobs

Command took 12.52 seconds -- by SWCPROPERTY@GMAIL.COM at 6/30/2021, 9:47:58 PM on My First Cluster

Cmd 16

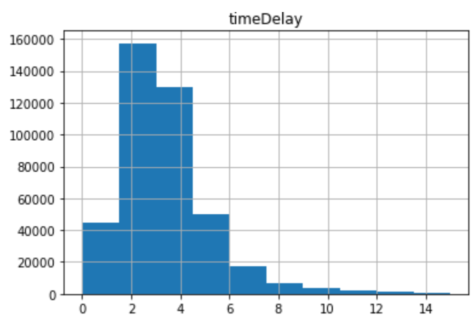
Visualize

Let's visualize the distribution of our time delay.

Cmd 17

```
%python
import pandas as pd
import numpy as np

fig = pdDF.hist(column="timeDelay")[0][0]
display(fig.figure)
```



Command took 1.15 seconds -- by SWCPROPERTY@GMAIL.COM at 6/30/2021, 9:48:13 PM on My First Cluster

Cmd 18

Train-Test Split

In this notebook we are going to use 80% of our data to train our model, and 20% to test our model. We set a random_state for reproducibility.

Cmd 19

```
%python
from sklearn.model_selection import train_test_split

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

Command took 1.63 seconds -- by SWCPROPERTY@GMAIL.COM at 6/30/2021, 9:48:39 PM on My First Cluster

Baseline Model

Before we get started building our linear regression model, let's establish our baseline RMSE on our test dataset by always predicting the average value. Here, we are going to take the square root of the MSE.

Cmd 21

```
%python
from sklearn.metrics import mean_squared_error
import numpy as np

avgDelay = np.full(y_test.shape, np.mean(y_train), dtype=float)

print("RMSE is {0}".format(np.sqrt(mean_squared_error(y_test, avgDelay))))
```

```
RMSE is 1.8348773381102907
```

Command took 0.04 seconds -- by SWCPROPERTY@GMAIL.COM at 6/30/2021, 9:48:50 PM on My First Cluster

Cmd 22

Build Linear Regression Model

Great! Now that we have established a baseline, let's use scikit-learn's pipeline API to build a linear regression model.

Our pipeline will have two steps:

1. One Hot Encoder: this converts our categorical features into numeric features by creating a dummy column for each value in that category.
   - For example, if we had a column called `Animal` with the values `Dog`, `Cat`, and `Bear`, the corresponding one hot encoding representation for Dog would be: `[1, 0, 0]`, Cat: `[0, 1, 0]`, and Bear: `[0, 0, 1]`
2. Linear Regression model: find the line of best fit for our training data

Cmd 23

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

ohe = ("ohe", OneHotEncoder(handle_unknown="ignore"))
lr = ("lr", LinearRegression(fit_intercept=True, normalize=True))

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

▾ (1) MLflow run

Logged 1 run to an MLflow experiment. Learn more

Command took 6.06 seconds -- by SWCPROPERTY@GMAIL.COM at 6/30/2021, 9:49:08 PM on My First Cluster

Cmd 24

You can see the corresponding one hot encoded feature names below.

Cmd 25

```
%python
print(pipeline.steps[0][1].get_feature_names())
```

```
['x0_Administrative' 'x0_Aircraft Emergency' 'x0_Alarms'
 'x0_Assist Police' 'x0_Citizen Assist / Service Call'
 'x0_Confined Space / Structure Collapse' 'x0_Electrical Hazard'
 'x0_Elevator / Escalator Rescue' 'x0_Explosion'
 'x0_Extrication / Entrapped (Machinery, Vehicle)' 'x0_Fuel Spill'
 'x0_Gas Leak (Natural and LP Gases)' 'x0_HazMat' 'x0_High Angle Rescue'
 'x0_Industrial Accidents' 'x0_Marine Fire' 'x0_Medical Incident'
 'x0_Mutual Aid / Assist Outside Agency' 'x0_Odor (Strange / Unknown)'
 'x0_Oil Spill' 'x0_Other' 'x0_Outside Fire'
 'x0_Smoke Investigation (Outside)' 'x0_Structure Fire'
 'x0_Suspicious Package' 'x0_Traffic Collision' 'x0_Train / Rail Incident'
 'x0_Vehicle Fire' 'x0_Water Rescue' 'x0_Watercraft in Distress' 'x1_1'
 'x1_10' 'x1_2' 'x1_3' 'x1_4' 'x1_5' 'x1_6' 'x1_7' 'x1_8' 'x1_9' 'x1_None'
 'x2_Bayview Hunters Point' 'x2_Bernal Heights' 'x2_Castro/Upper Market'
 'x2_Chinatown' 'x2_Excelsior' 'x2_Financial District/South Beach'
 'x2_Glen Park' 'x2_Golden Gate Park' 'x2_Haight Ashbury'
 'x2_Hayes Valley' 'x2_Inner Richmond' 'x2_Inner Sunset' 'x2_Japantown'
 'x2_Lakeshore' 'x2_Lincoln Park' 'x2_Lone Mountain/USF' 'x2_Marina'
 'x2_McLaren Park' 'x2_Mission' 'x2_Mission Bay' 'x2_Nob Hill'
 'x2_Noe Valley' 'x2_None' 'x2_North Beach'
 'x2_Oceanview/Merced/Ingleside' 'x2_Outer Mission' 'x2_Outer Richmond'
```

Command took 0.03 seconds -- by SWCPROPERTY@GMAIL.COM at 6/30/2021, 9:49:21 PM on My First Cluster

Evaluate on Test Data

Let's take a look at our RMSE.

Cmd 27

```
%python
from sklearn.metrics import mean_squared_error

print("RMSE is {0}".format(np.sqrt(mean_squared_error(y_test, y_pred))))
```

RMSE is 1.724841956799332

Command took 0.03 seconds -- by SWCPROPERTY@GMAIL.COM at 6/30/2021, 9:49:31 PM on My First Cluster

Cmd 28

Save Model

Not bad! We did a bit better than our baseline model.

Let's save this model using MLflow. MLflow is an open-source project created by Databricks to help simplify the Machine Learning life cycle.

While MLflow is out of the scope of this class, it has a nice function to generate Spark User-Defined Function (UDF) to apply this model in parallel to the rows in our dataset. We will see this in the next notebook.

Cmd 29

```
%python
try:
  import mlflow
  from mlflow.sklearn import save_model

  model_path = "/dbfs/" + username + "/firecalls_pipeline"
  dbutils.fs.rm(username + "/firecalls_pipeline", recurse=True)
  save_model(pipeline, model_path)
except:
  print("ERROR: This cell did not run, likely because you're not running the correct version of software. Please use a cluster with `DBR 5.5 ML` rather than `DBR 5.5` or a different cluster version.")

```

Command took 0.14 seconds -- by SWCPROPERTY@GMAIL.COM at 6/30/2021, 9:49:44 PM on My First Cluster

Cmd 30

Applying ML with UDFs

Module 4, Lesson 6

In this notebook you:

- Apply a pre-trained Linear Regression model to predict response times
- Identify which types of calls or neighborhoods are anticipated to have the longest response time

Cmd 3

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

Command took 56.01 seconds -- by SWCPROPERTY@GMAIL.COM at 6/30/2021, 11:26:59 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

Create UDF

MLflow can create a User Defined Function for us to use in PySpark or SQL. This allows for custom code (that is, functionality not in core Spark) to be run on Spark.

You can use `spark.udf.register` to register this Python UDF in the SQL namespace and call it `predictUDF`.

Cmd 5

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

  model_path = "/dbfs/mnt/davis/fire-calls/models/firecalls_pipeline"
  predict = spark_udf(spark, model_path, result_type="string")

  spark.udf.register("predictUDF", predict)
except:
  print("ERROR: This cell did not run, likely because you're not running the correct version of software. Please use a cluster with `DBR 5.5 ML` rather than `DBR 5.5` or a different cluster version.")
```

Import the Data

Create a temporary view called `fireCallsParquet`

Cmd 7

```
CREATE OR REPLACE TEMPORARY VIEW fireCallsParquet
USING Parquet
OPTIONS (
    path "/mnt/davis/fire-calls/fire-calls-1p.parquet"
  )
```

▸ (1) Spark Jobs

OK

Command took 1.75 seconds -- by SWCPROPERTY@GMAIL.COM at 6/30/2021, 11:14:32 PM on My Cluster

Cmd 8

Save Predictions

We are going to save our predictions to a table called `predictions`.

Cmd 9

```
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) as double) as prediction, *
  FROM fireCallsParquet
  LIMIT 10000)
```

Save Predictions

We are going to write out our predictions to a table called `Predictions`.

Cmd 9

```
USE DATABRICKS;
DROP TABLE IF EXISTS predictions;

CREATE TABLE predictions AS (
  SELECT *, cast(predictUDF(Call_Type, Fire_Prevention_District, `Neighborhooods_-_Analysis_Boundaries`,
                Number_of_Alarms, Original_Priority, Unit_Type) as double) as prediction
  FROM fireCallsParquet
  LIMIT 10000)
```

▸ (1) Spark Jobs

OK

Command took 21.94 seconds -- by conor.murphy@databricks.com at 2/21/2019, 5:24:44 PM on myfirstcluster

Cmd 10

```
SELECT * FROM predictions
```

▸ (1) Spark Jobs

ber_of_Alarms	Unit_Type	Unit_sequence_in_call_dispatch	Fire_Prevention_District	Supervisor_District	Neighborhooods_-_Analysis_Boundaries	Location	RowID	prediction
	TRUCK	3	5	8	Castro/Upper Market	(37.7658679882367, -122.431025473299)	131020115-T06	3.6939779036086473
	TRUCK	2	7	1	Outer Richmond	(37.7798905753776, -122.489013490407)	061150482-T14	2.235490212419418
	CHIEF	3	4	2	Marina	(37.7979514351842, -122.443292158435)	093060021-B04	3.2860609500543996
	CHIEF	3	2	6	South of Market	(37.7744195015925, -122.410885266244)	130850246-B03	3.2368085291594144

Showing the first 1000 rows.

Average Prediction by Neighborhood

Let's see which district in San Francisco has the highest predicted average response time! Do you remember why we are setting the shuffle partitions here?

Cmd 12

```
SET spark.sql.shuffle.partitions=8;

SELECT avg(prediction) as avgPrediction, `Neighborhooods_-_Analysis_Boundaries`
FROM predictions
GROUP BY `Neighborhooods_-_Analysis_Boundaries`
ORDER BY avgPrediction DESC
```

▸ (1) Spark Jobs

avgPrediction	Neighborhooods_-_Analysis_Boundaries
4.318644445623701	Treasure Island
4.124829115555952	None
3.8228950535873256	Presidio
3.757215087520699	Seacliff
3.4948636986869794	Twin Peaks
3.4557893921610208	Lakeshore
3.363489626430374	Visitacion Valley
3.332032827703922	Glen Park

Average Prediction by Neighborhood

Let's see which district in San Francisco has the highest predicted average response time! Do you remember why we are setting the shuffle partitions here?

Cmd 11

```
SET spark.sql.shuffle.partitions=8;
```

	key	value
1	spark.sql.shuffle.partitions	8

Showing all 1 rows.

Command took 0.18 seconds -- by SWCPROPERTY@GMAIL.COM at 6/30/2021, 11:32:30 PM on My Cluster

Cmd 12

```
SELECT avg(prediction) as avgPrediction, `Neighborhooods_-_Analysis_Boundaries`
FROM predictions
GROUP BY `Neighborhooods_-_Analysis_Boundaries`
ORDER BY avgPrediction DESC
```

San Francisco Districts

Cmd 14

Standard Deviation on Prediction by Neighborhood

Cmd 15

```sql
SELECT stddev(prediction) as stddevPrediction, `Neighborhooods_-_Analysis_Boundaries`
FROM predictions
GROUP BY `Neighborhooods_-_Analysis_Boundaries`
ORDER BY stddevPrediction DESC
```

Standard Deviation on Prediction by Neighborhood

Cmd 15

```sql
SELECT stddev(prediction) as stddevPrediction, `Neighborhooods_-_Analysis_Boundaries`
FROM predictions
GROUP BY `Neighborhooods_-_Analysis_Boundaries`
ORDER BY stddevPrediction DESC
```

▸ (1) Spark Jobs

stddevPrediction	Neighborhooods_-_Analysis_Boundaries
0.987371486099583	Seacliff
0.9094264632450011	Treasure Island
0.7625440938035575	Glen Park
0.7544633196039048	Haight Ashbury
0.7321867220613195	Pacific Heights
0.7271908009114048	Marina
0.7023229238863061	Lakeshore
0.6978604542458299	West of Twin Peaks

Average Prediction by Call Type

Cmd 17

```sql
SELECT avg(prediction) as avgPrediction, Call_Type
FROM predictions
GROUP BY Call_Type
ORDER BY avgPrediction DESC
```

▸ (1) Spark Jobs

avgPrediction	Call_Type
7.404982442266542	Watercraft in Distress
6.51103247220119	Extrication / Entrapped (Machinery, Vehicle)
5.526132635689161	HazMat
5.433352438566127	High Angle Rescue
5.034354588668969	Marine Fire
4.851198153802674	Train / Rail Incident
4.368151081000986	Odor (Strange / Unknown)
4.367899273276612	Water Rescue

Average Prediction by Call Type

Cmd 17

```sql
SELECT avg(prediction) as avgPrediction, Call_Type
FROM predictions
GROUP BY Call_Type
ORDER BY avgPrediction DESC

```

Cmd 18

Assignment (SQL)

My Cluster | File | Edit | View: Standard | Permissions | Run All | Clear

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 10.49 seconds -- by SWCPROPERTY@GMAIL.COM at 6/30/2021, 11:44:48 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

```sql
-- TODO
USE DATABRICKS;
Create table if not exists fireCallsClean
using Parquet
Options (
  path "/mnt/davis/fire-calls/fire-calls-clean.parquet"
)
```

▸ (1) Spark Jobs

OK

Command took 2.34 seconds -- by SWCPROPERTY@GMAIL.COM at 6/30/2021, 11:56:24 PM on My Cluster

Cmd 6

Check that your data is loaded in properly.

Cmd 7

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

▸ (1) Spark Jobs

	_Unit	Call_Type_Group	Number_of_Alarms	Unit_Type	Unit_sequence_in_call_dispatch	Fire_Prevention_District	Supervisor_District	Neighborhooods_-_Analysis_Boundaries	Location
1		Non Life-threatening	1	MEDIC	1	10	10	Bayview Hunters Point	(37.740961928907, -122.401555700705)
2		null	1	MEDIC	1	2	8	Mission	(37.7692677111289, -122.423396856968)
3		Non Life-threatening	1	PRIVATE	1	2	9	Mission	(37.7650513381945, -122.419668973861)
4		null	1	MEDIC	1	9	11	Outer Mission	(37.7258249736518, -122.442324422614)
5		Potentially Life-Threatening	1	ENGINE	1	10	9	Portola	(37.7318198889718, -122.405412091734)
6		Non Life-threatening	1	ENGINE	1	2	6	South of Market	(37.7777124404316, -122.412736707425)
7		Non Life-threatening	1	PRIVATE	1	3	6	South of Market	(37.7746534767072, -122.405118197249)

Showing all 10 rows.

Command took 8.65 seconds -- by SWCPROPERTY@GMAIL.COM at 6/30/2021, 11:56:36 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

```sql
-- TODO
select `Call_Type_Group`, count(*) as cnt
from fireCallsClean
group by `Call_Type_Group`
order by cnt
```

▸ (2) Spark Jobs

	Call_Type_Group	cnt
1	Fire	4196
2	Alarm	32566
3	Non Life-threatening	56168
4	Potentially Life-Threatening	78030
5	null	246459

Showing all 5 rows.

Command took 3.34 seconds -- by SWCPROPERTY@GMAIL.COM at 6/30/2021, 11:58:28 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

```sql
-- TODO
Create or Replace temporary View fireCallsGroupCleaned as (
select *
from fireCallsClean
where `Call_Type_Group` IN ('Non Life-threatening','Potentially Life-Threatening')
)
```

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`, count(*) as cnt
from fireCallsGroupCleaned
group by `Call_Type_Group`
order by cnt
```

▸ (2) Spark Jobs

	Call_Type_Group	cnt
1	Non Life-threatening	56168
2	Potentially Life-Threatening	78030

Showing all 2 rows.

Command took 3.05 seconds -- by SWCPROPERTY@GMAIL.COM at 7/1/2021, 12:02:50 AM 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 temp 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
)
;
select count(*) from fireCallsDF
```

▸ (2) Spark Jobs

	count(1)
1	134198

Showing all 1 rows.

Command took 2.70 seconds -- by SWCPROPERTY@GMAIL.COM at 7/1/2021, 12:13:04 AM 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

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

Command took 6.93 seconds -- by SWCPROPERTY@GMAIL.COM at 7/1/2021, 12:13:43 AM 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

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
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)
```

▾ (1) MLflow run

Logged 1 run to an MLflow experiment. Learn more

```
/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 6.89 seconds -- by SWCPROPERTY@GMAIL.COM at 7/1/2021, 12:15:06 AM 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
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 SWCPROPERTY@GMAIL.COM at 7/1/2021, 12:15:32 AM 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
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 0.14 seconds -- by SWCPROPERTY@GMAIL.COM at 7/1/2021, 12:16:20 AM on My Cluster

Cmd 32

UDF

Cmd 33

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 34

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

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

```
Out[11]: <function mlflow.pyfunc.spark_udf.<locals>.predict(*args)>
```

Command took 0.16 seconds -- by SWCPROPERTY@GMAIL.COM at 7/1/2021, 12:33:35 AM on My Cluster

Cmd 35

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

Cmd 36

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

spark_df: pyspark.sql.dataframe.DataFrame

```
Call_Type: long
Fire_Prevention_District: long
Neighborhooods_-_Analysis_Boundaries: long
Number_of_Alarms: long
Original_Priority: long
Unit_Type: long
Battalion: long
```

Command took 0.14 seconds -- by SWCPROPERTY@GMAIL.COM at 7/1/2021, 12:33:39 AM on My Cluster

Cmd 37

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 38

```
-- 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 prediction, *
  from testTable )
```

OK

Command took 0.17 seconds -- by SWCPROPERTY@GMAIL.COM at 7/1/2021, 12:33:48 AM on My Cluster

Cmd 39

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

Cmd 40

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

▸ (1) Spark Jobs

	prediction	Call_Type	Fire_Prevention_District	Neighborhooods_-_Analysis_Boundaries	Number_of_Alarms	Original_Priority	Unit_Type	Battalion
1	1	0	3	34	0	2	2	2
2	0	2	1	0	0	1	2	9
3	0	0	9	25	0	1	4	8
4	0	0	3	34	0	1	5	2
5	1	0	2	18	0	2	4	1
6	0	0	3	34	0	1	4	2
7	0	0	8	35	0	1	4	7

Showing all 10 rows.

Command took 2.47 seconds -- by SWCPROPERTY@GMAIL.COM at 7/1/2021, 12:33:52 AM on My Cluster

Cmd 41

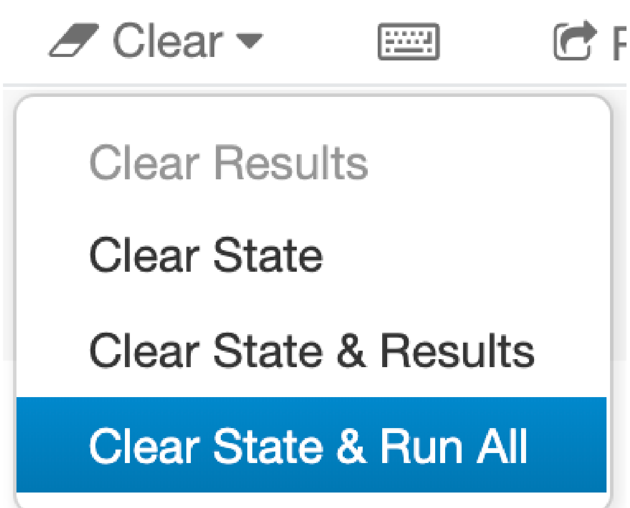
Question 4:

What 2 values are in the `prediction` column?

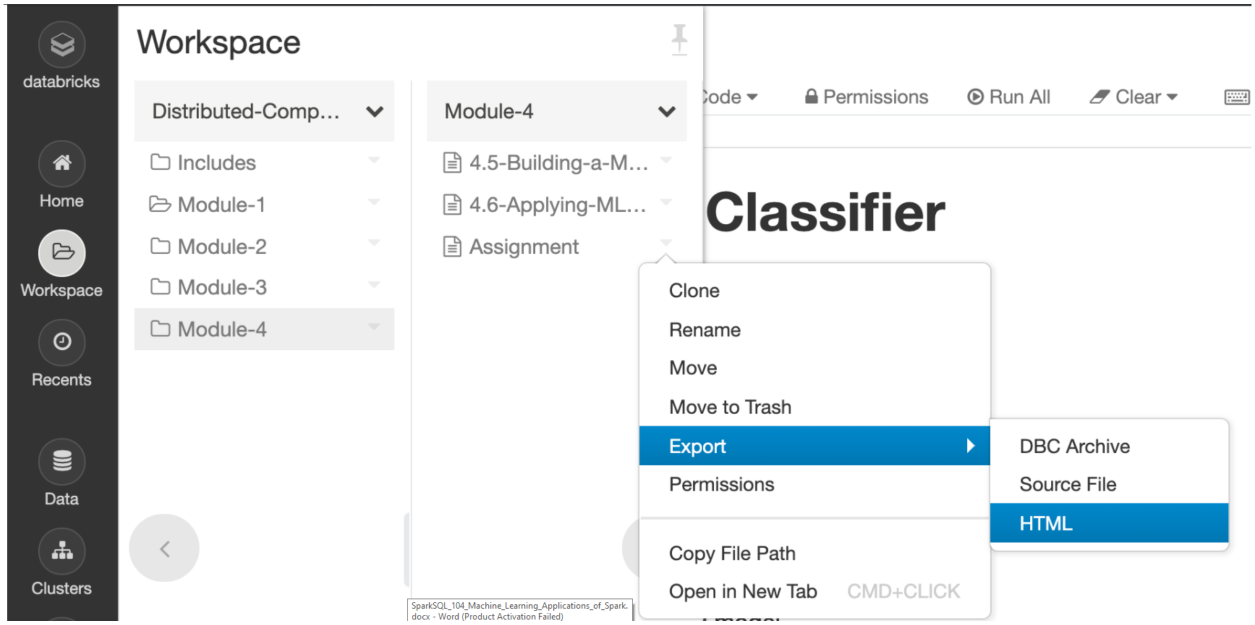
Cmd 42

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 43

Shift+Enter to run