

Watson Studio: Machine Learning with SparkML

Introduction

In this lab, we will explore machine learning using Spark ML. We will exploit Spark ML's high-level APIs built on top of DataFrames to create and tune machine learning pipelines. We will utilize Spark ML's feature transformers to convert, modify and scale the features that will be used to develop the machine learning model. Finally, we will evaluate and cross validate our model to demonstrate the process of determining a best fit model, load the results in the database, and save the model to the model repository.

We are using machine learning to try to predict records that a human has not seen or vetted before. We will use these predictions to sort the highest priority records for a human to look at. We will use as a training set for the algorithm simulated data that has been vetted by an analyst as high, medium or low.

End-to-End Data Science

The general flow of the End to End Data Science PoT will be guided by the activities shown in Figure 1- End to End Flow. This lab spans the Prepare Data, Build Model, and Save and Deploy activities.

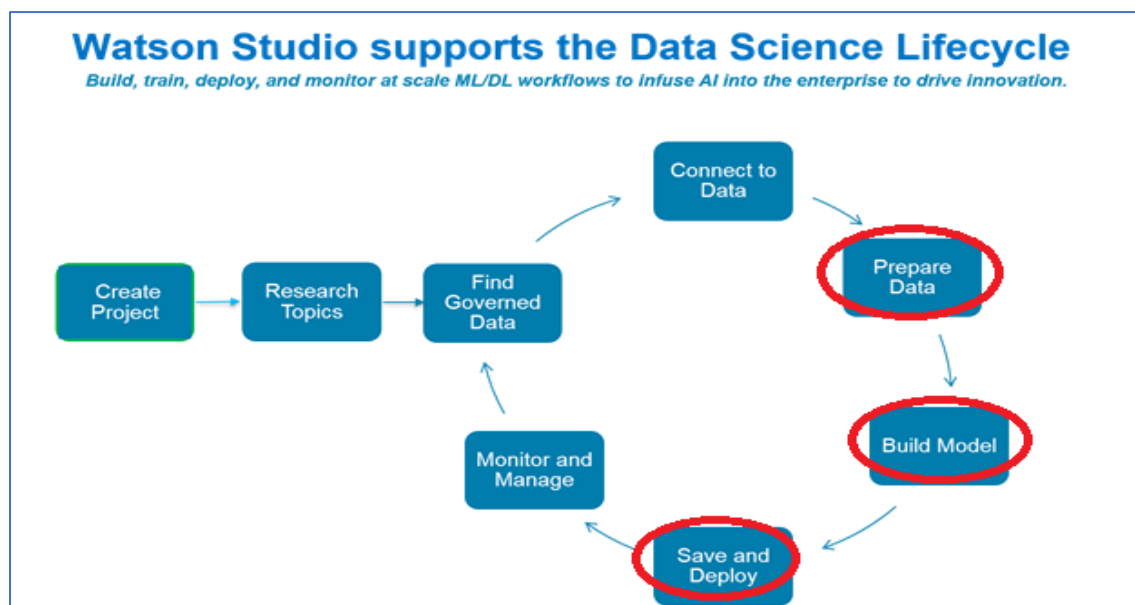


Figure 1- End to End Flow

Objectives

Upon completing the lab, you will know how to:

- Join data from three sources.
- Identify labels and transform data.
- Conduct feature engineering for algorithm data.

- Declare a machine learning model.
- Setup the Pipeline for data transforms and training.
- Train the model.
- Evaluate and show model results
- Automatically tune model
- Score data and load into a new DB2 table.
- Save the model to the model repository.

Female Human Trafficking Data

The data sets used for this lab consist of **simulated** travel itinerary data. The use case corresponds to an analyst reviewing the travel data to assign a risk of trafficking. The risk is recorded as the VETTING_LEVEL column in the dataset. Some of the records have already been analyzed and have a VETTING_LEVEL of low (value is 30), medium (value is 20), or high risk (value is 10). Others have not yet been vetted (value is 100). We will use the data that has been vetted to train a model to predict the risk for the unvetted records. This can be used to automate the process and augment the analyst. For example, one option would be to send the predicted high-risk persons to the analyst for further investigation.

The OCCUPATION data included in the travel data is very granular. For modeling purposes, it was decided to categorize the OCCUPATION data. Two additional datasets are used for this purpose. The occupation.csv dataset maps the granular occupation data to a category code. The categories dataset maps a category code to a category description. These datasets will be joined to the main dataset to prepare the data for modeling.

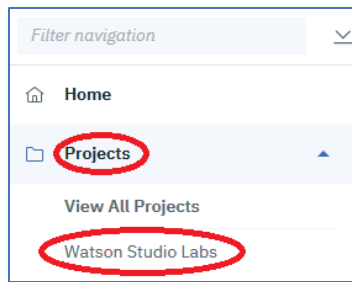
Other columns in the dataset are similarly very granular and could also be categorized for modeling purposes. This lab does not include steps to accomplish this, but it would be similar to what was done for the occupation column.

Lab Steps

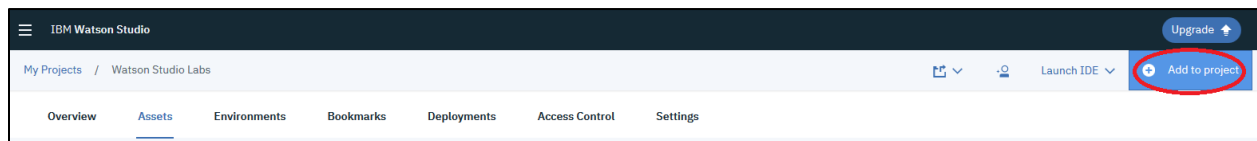
Step 1 - Create a Jupyter Notebook

1. Click on the hamburger icon , then click on **Projects**, and then **Watson Studio Labs** (or whatever you named the project)

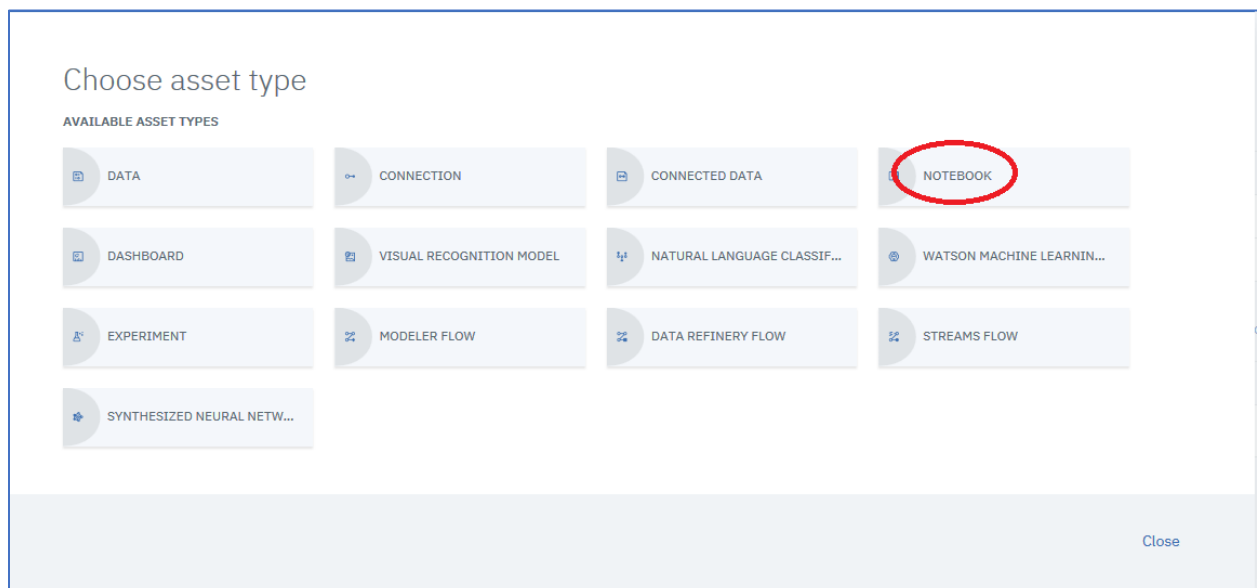




2. We are now going to create a notebook in our project. This notebook will be created from a url that points to the Machine Learning with SparkML notebook in the github repository. Click the **Add to project** link.



3. Click on **NOTEBOOK**



4. Click on **From URL** under **New Notebook**, enter **Machine Learning with SparkML** for the **Name**, optionally enter a **Description**, cut and paste the following url into the **Notebook URL** field.

https://github.com/bleonardb3/DS_POT_06-03/blob/master/Lab-3/Machine%20Learning%20with%20SparkML.ipynb

Select the Runtime.

MAKE SURE TO SELECT Default Spark Python 3.5 XS (Driver with 1vCPU ...

Click **Create Notebook**.

New notebook

Blank From file **From URI**

Name*
Machine Learning with SparkML 21 Characters Remaining

Description
Type your Description here

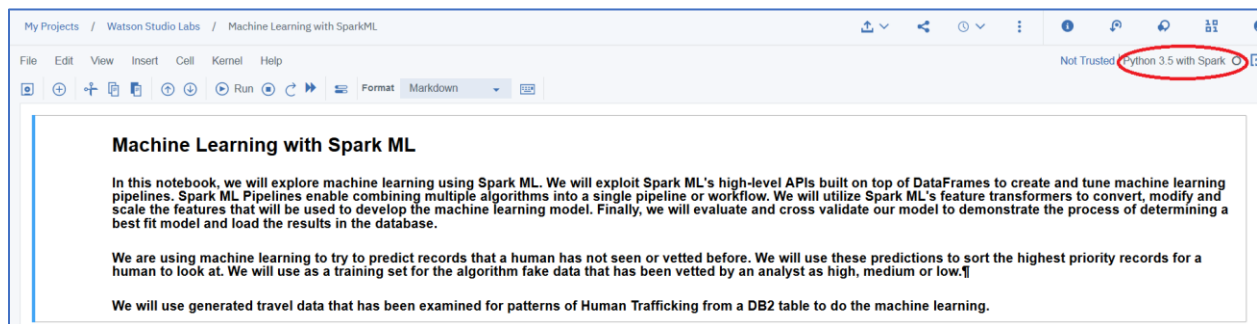
Notebook URL*
https://github.com/bleonardb3/D5_POT_06-03/blob/master/Lab-3/Machine%20Learnin

Select runtime* Includes notebook environments ⓘ
Default Spark Python 3.5 XS (Driver with 1 vCPU and 4 GB RAM, 2 executors with 1 v...

The selected runtime uses one driver with 1 vCPU and 4 GB RAM, and 2 executors each with 1 vCPU and 4 GB RAM.
This runtime consumes 1.5 capacity units per hour.
[Learn more about capacity unit hours and Watson Studio pricing plans.](#)

Cancel **Create Notebook**

5. Please make sure the notebook has Python 3.5 with Spark in the top right corner.



6. A Jupyter notebook consists of a series of cells. These cells are of 2 types (1) documentation cells containing markdown, and (2) code cells (denoted by a bracket on the left of the cell) where you write Python code, R, or Scala code depending on the type of notebook. Code cells can be run by putting the cursor in the code cell and pressing **<Shift><Enter>** on the keyboard. Alternatively, you can execute the cells by clicking on **Run icon** on the menu bar that will run the current cell (where the cursor is located) and then select the cell below. In this way, repeatedly clicking on **Run** executes all the cells in the notebook. When a code cell is executed the brackets on the left change to an asterisk '*' to indicate the code cell is executing. When completed, a sequence number appears. The output, if any, is displayed below the code cell.

Step 2: Insert Generated Code

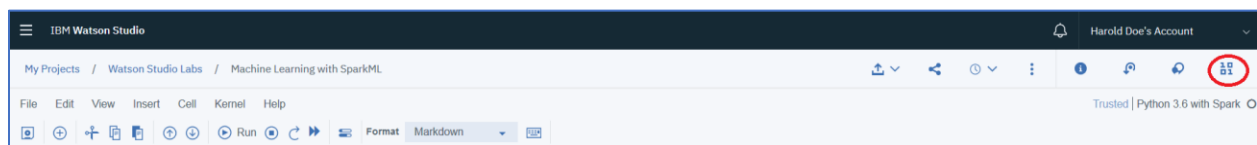
1. Before executing the cells in the notebook, we are going to use the IBM value-add code generator to insert code in 3 code cells that will read in the 3 input files and code in 1

code cell that will specify the database credentials. Scroll down in the notebook until you see the **Read Data Asset- female_human_trafficking – See Lab Instructions** and put the cursor in the code cell underneath the comment lines. (Comments begin with the # sign).

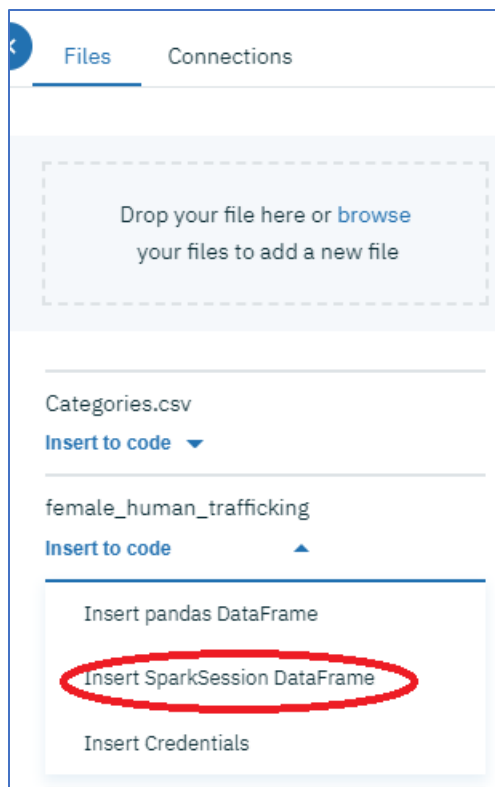
Read Data Asset - female_human_trafficking - See Lab Instructions

```
In [ ]: # Insert SparkSession DataFrame code in this cell after the comments.  
# make CERTAIN to rename the default dataframe name (df_data_1 or df_data_2 or df_data_3, etc) to trafficking_df  
# Put cursor on the next line to Insert to code.
```

- Click on the 1/0 icon.  at the top right.



- Click on the insert to code down arrow  below **female human trafficking** and click on **insertSparkSession DataFrame**.



- Locate the variable `df_data_n` (n is a number). This is a generated variable. We need to change this variable name to **trafficking_df**.

Read Data Asset - female_human_trafficking - See Lab Instructions

```
In [ ]: # Insert SparkSession DataFrame code in this cell after the comments.
# make CERTAIN to rename the default dataframe name (df_data_1 or df_data_2 or df_data_3, etc) to trafficking_df
# Put cursor on the next line to Insert to code.
from pyspark.sql import SparkSession
spark = SparkSession.builder.getOrCreate()

#@hidden_cell
# The following code is used to access your data and contains your credentials.
# You might want to remove those credentials before you share your notebook.

properties_b13f4a659ddb4b3aa53ffd2f142e4912 = {
    'jdbcurl': 'jdbc:db2://dashdb-entry-yp-dal09-08.services.dal.bluemix.net:50000/BLUDB',
    'user': 'dash100316',
    'password': 'GvEI{uLxgr4n'
}

data_df_1 = spark.read.jdbc(properties_b13f4a659ddb4b3aa53ffd2f142e4912['jdbcurl'], table='DASH100316.FEMALE_HUMAN_TRAFFICKING', properties=properties_b13f4a659ddb4b3aa53ffd2f142e4912)
data_df_1.head()
```

Change to.

Read Data Asset - female_human_trafficking - See Lab Instructions

```
In [ ]: # Insert SparkSession DataFrame code in this cell after the comments.
# make CERTAIN to rename the default dataframe name (df_data_1 or df_data_2 or df_data_3, etc) to trafficking_df
# Put cursor on the next line to Insert to code.
from pyspark.sql import SparkSession
spark = SparkSession.builder.getOrCreate()

#@hidden_cell
# The following code is used to access your data and contains your credentials.
# You might want to remove those credentials before you share your notebook.

properties_b13f4a659ddb4b3aa53ffd2f142e4912 = {
    'jdbcurl': 'jdbc:db2://dashdb-entry-yp-dal09-08.services.dal.bluemix.net:50000/BLUDB',
    'user': 'dash100316',
    'password': 'GvEI{uLxgr4n'
}

trafficking_df = spark.read.jdbc(properties_b13f4a659ddb4b3aa53ffd2f142e4912['jdbcurl'], table='DASH100316.FEMALE_HUMAN_TRAFFICKING', properties=properties_b13f4a659ddb4b3aa53ffd2f142e4912)
trafficking_df.head()
```


5. Scroll down to **Read Data Asset – Occupations – See Lab Instructions**. Click cursor underneath the commented lines in the code cell.

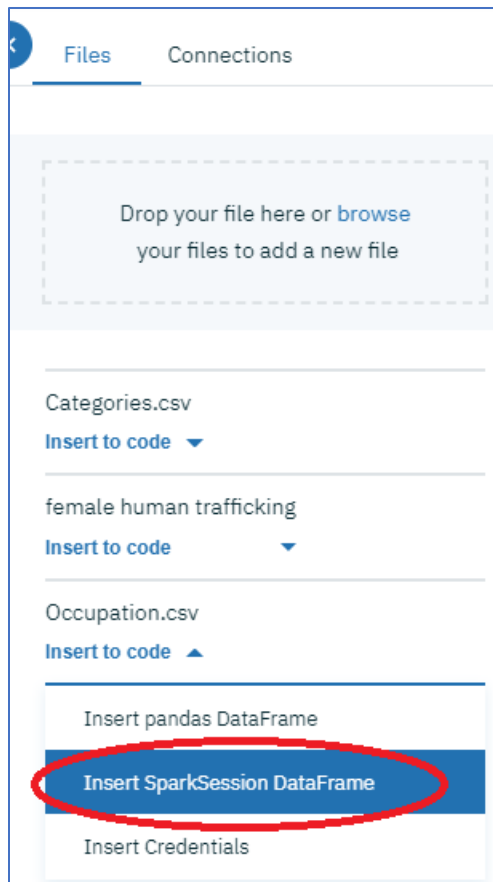
Read Data Asset - Occupations - See Lab Instructions

The occupations listed in the female human trafficking file are too numerous to use as input to a machine learning model. We will categorize these occupations into 15 categories by joining with two other files. The Occupation.csv file contains a mapping of the occupations in the female human trafficking table to a category code. The Categories.csv file contains each code followed by the category name. This information needs to be joined to the female human trafficking table.

Follow the same procedure as above to insert a SparkDataFrame for Occupations

```
In [ ]: # Insert SparkSession DataFrame code in this cell after the comments
# make CERTAIN to rename the default dataframe name (df_data_1 or df_data_2 or df_data_3, etc) to occupations
# Put cursor on the next line to Insert to code
```

6. Click on  down arrow underneath **Occupation.csv**. Click on **Insert Spark Session DataFrame** (Note if it says **Insert SparkSession Setup** choose that option)



7. Locate the variable `df_data_n` (n is a number). This is a generated variable. We need to change this variable name to **occupations**.

Read Data Asset - Occupations - See Lab Instructions

The occupations listed in the female human trafficking file are too numerous to use as input to a machine learning model. We will categorize these occupations into 15 categories by joining with two other files. The Occupation.csv file contains a mapping of the occupations in the female human trafficking table to a category code. The Categories.csv file contains each code followed by the category name. This information needs to be joined to the female human trafficking table.

Follow the same procedure as above to insert a SparkDataFrame for Occupations

```
In [ ]: # Insert SparkSession DataFrame code in this cell after the comments
# make CERTAIN to rename the default dataframe name (df_data_1 or df_data_2 or df_data_3, etc) to occupations
# Put cursor on the next line to Insert to code
import ibmos2spark
# @hidden_cell
credentials = {
    'endpoint': 'https://s3.us-south.objectstorage.service.networklayer.com',
    'service_id': '521638cb-5af6-4e63-93cb-7bdb93aee863',
    'iam_service_endpoint': 'https://iam.ng.bluemix.net/oidc/token',
    'api_key': 'XwVvkqalVGhyEKYh939--uGpC_XZ9Ktvpv5NS_4nj1VN'
}

configuration_name = 'os_8f79652d51d1439eba1d3b38a0253b8c_configs'
cos = ibmos2spark.CloudObjectStorage(sc, credentials, configuration_name, 'bluemix_cos')

from pyspark.sql import SparkSession
spark = SparkSession.builder.getOrCreate()
df_data_4 = spark.read\
    .format('org.apache.spark.sql.execution.datasources.csv.CSVFileFormat')\
    .option('header', 'true')\
    .load(fros.url('data_asset/Occupation_S3cYutWTE.csv', 'watsonstudiolabs-datacatalog-rkvbotbp4'))
df_data_4.take(5)
```

Change to:

Read Data Asset - Occupations - See Lab Instructions

The occupations listed in the female human trafficking file are too numerous to use as input to a machine learning model. We will categorize these occupations into 15 categories by joining with two other files. The Occupation.csv file contains a mapping of the occupations in the female human trafficking table to a category code. The Categories.csv file contains each code followed by the category name. This information needs to be joined to the female human trafficking table.

Follow the same procedure as above to insert a SparkDataFrame for Occupations

```
In [ ]: # Insert SparkSession DataFrame code in this cell after the comments
# make CERTAIN to rename the default dataframe name (df_data_1 or df_data_2 or df_data_3, etc) to occupations
# Put cursor on the next line to insert to code
import ibmos2spark
# @hidden_cell
credentials = {
    'endpoint': 'https://s3.us-south.objectstorage.service.networklayer.com',
    'service_id': '521638cb-5af6-4e63-93cb-7bdb93aee063',
    'iam_service_endpoint': 'https://iam.ng.bluemix.net/oidc/token',
    'api_key': 'XwVvkq1VGlyEKYh939--uGpC_XZ9KtVpVSNS_4nj1VN'
}

configuration_name = 'os_8f79652d51d1439ebaid3b38a0253b8c_configs'
cos = ibmos2spark.CloudObjectStorage(sc, credentials, configuration_name, 'bluemix_cos')

from pyspark.sql import SparkSession
spark = SparkSession.builder.getOrCreate()
occupations = spark.read
    .format('org.apache.spark.sql.execution.datasources.csv.CSVFileFormat')\
    .option('header', 'true')\
    .load(cos.url('data_asset/Occupation_5JcYutMTE.csv', 'watsonstudiolabs-datacatalog-rkvbotbp4'))
occupations.take(5)
```

8. Scroll down to **Read Data Asset – Categories – See Lab Instructions**. Click cursor underneath the commented lines in the code cell.

Read Data Asset - Categories - See Lab Instructions

Follow the same procedure as above to insert a SparkDataFrame for Categories

```
In [ ]: # Insert SparkSession DataFrame code in this cell after the comments
# make CERTAIN to rename the default dataframe name (df_data_1 or df_data_2 or df_data_3, etc) to categories
# Put cursor on the next line to insert to code
```

Join the Occupation mapping file content with the Category file content

9. Click on **Insert to code**  down arrow underneath **Categories.csv**. Click on **Insert Spark Session DataFrame**. (Note if it says **Insert SparkSession Setup** choose that option)

Files

Connections

Drop your file here or [browse](#) your files to add a new file

Categories.csv

Insert to code

Insert pandas DataFrame

Insert SparkSession DataFrame

Insert Credentials

10. Locate the variable `df_data_n` (n is a number). This is a generated variable. We need to change this variable name to **categories**.

Read Data Asset - Categories - See Lab Instructions
Follow the same procedure as above to insert a SparkDataFrame for Categories

```
In [ ]: # Insert SparkSession DataFrame code in this cell after the comments
# make CERTAIN to rename the default dataframe name (df_data_1 or df_data_2 or df_data_3, etc) to categories
# Put cursor on the next line to Insert to code
import ibmos2spark
# @hidden_cell
credentials = {
    'endpoint': 'https://s3.us-south.objectstorage.service.networklayer.com',
    'service_id': 'S21638cb-5af6-4e63-93cb-7bdb93aee063',
    'iam_service_endpoint': 'https://iam.ng.bluemix.net/oidc/token',
    'api_key': 'XwYvqalVGnyEKYh939--uGpC_XZ9KtVpV5N5_4njiVN'
}

configuration_name = 'os_8f79652d51d1439eba1d3b38a0253b8c_configs'
cos = ibmos2spark.CloudObjectStorage(sc, credentials, configuration_name, 'bluemix_cos')

from pyspark.sql import SparkSession
spark = SparkSession.builder.getOrCreate()
df_data_4 = spark.read\
    .format('org.apache.spark.sql.execution.datasources.csv.CSVFileFormat')\
    .option('header', 'true')\
    .load(cos.url('data_asset/Categories_rJOS0tWTE.csv', 'watsonstudiolabs-datacatalog-rkvbotbp4'))
df_data_4.take(5)
```

Change to:

Read Data Asset - Categories - See Lab Instructions
Follow the same procedure as above to insert a SparkDataFrame for Categories

```
In [ ]: # Insert SparkSession DataFrame code in this cell after the comments
# make CERTAIN to rename the default dataframe name (df_data_1 or df_data_2 or df_data_3, etc) to categories
# Put cursor on the next line to Insert to code
import ibmos2spark
# @hidden_cell
credentials = {
    'endpoint': 'https://s3.us-south.objectstorage.service.networklayer.com',
    'service_id': 'S21638cb-5af6-4e63-93cb-7bdb93aee063',
    'iam_service_endpoint': 'https://iam.ng.bluemix.net/oidc/token',
    'api_key': 'XwYvqalVGnyEKYh939--uGpC_XZ9KtVpV5N5_4njiVN'
}

configuration_name = 'os_8f79652d51d1439eba1d3b38a0253b8c_configs'
cos = ibmos2spark.CloudObjectStorage(sc, credentials, configuration_name, 'bluemix_cos')

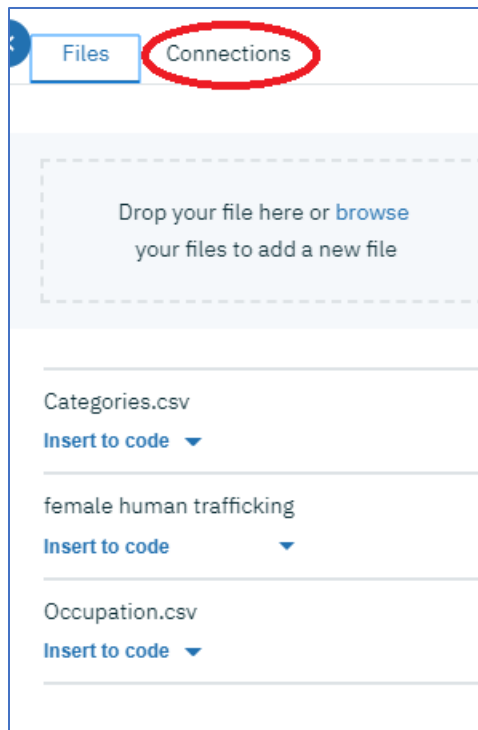
from pyspark.sql import SparkSession
spark = SparkSession.builder.getOrCreate()
categories = spark.read\
    .format('org.apache.spark.sql.execution.datasources.csv.CSVFileFormat')\
    .option('header', 'true')\
    .load(cos.url('data_asset/Categories_rJOS0tWTE.csv', 'watsonstudiolabs-datacatalog-rkvbotbp4'))
categories.take(5)
```

11. Scroll down further towards the middle of the notebook until you see **Insert the database credentials – see Lab Instructions**. Click cursor underneath the commented lines in the code cell.

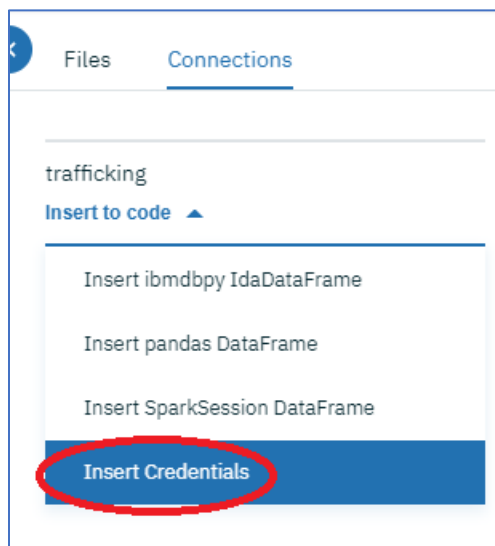
Insert the database credentials - see Lab Instructions

```
In [ ]: # Insert database connection credentials below
# Make sure the name that is used is credentials. If credentials_1 is shown, please change to credentials.
```

12. Click on **Connections**.



13. Click on [Insert to code](#) ▼ down arrow underneath **trafficking**. Click on **Insert Credentials**.



14. Locate the variable `credentials_n` (n is a number). This is a generated variable. We need to change this variable name to **credentials**.

Insert the database credentials - see Lab Instructions

```
In [ ]: # Insert database connection credentials below
# Make sure the name that is used is credentials. If credentials_1 is shown, please change to credentials.
# @hidden_cell
# The following code contains the credentials for a connection in your Project.
# You might want to remove those credentials before you share your notebook.
credentials = {
    'username': 'dash100316',
    'password': '""GvEI{uLxgr4r""',
    'sg_service_url': 'https://sgmanager.ng.bluemix.net',
    'database': 'BLUDB',
    'host': 'dashdb-entry-yp-dal09-08.services.dal.bluemix.net',
    'port': '50000',
    'url': 'https://undefined'
}
```

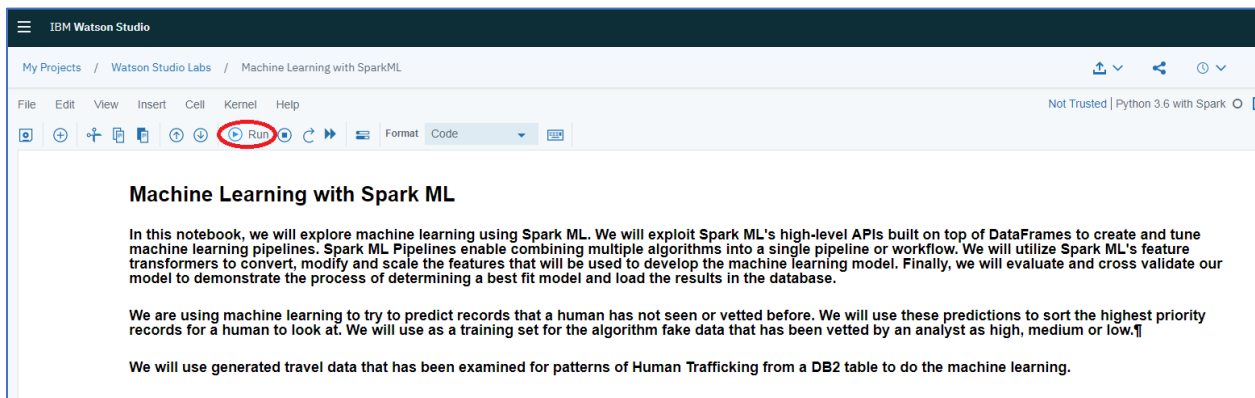
Change to:

Insert the database credentials - see Lab Instructions

```
In [ ]: # Insert database connection credentials below
# Make sure the name that is used is credentials. If credentials_1 is shown, please change to credentials.
# @hidden_cell
# The following code contains the credentials for a connection in your Project.
# You might want to remove those credentials before you share your notebook.
credentials = {
    'username': 'dash100316',
    'password': '""GvEI{uLxgr4r""',
    'sg_service_url': 'https://sgmanager.ng.bluemix.net',
    'database': 'BLUDB',
    'host': 'dashdb-entry-yp-dal09-08.services.dal.bluemix.net',
    'port': '50000',
    'url': 'https://undefined'
}
```

Step 3: Execute the code cells in the notebook

1. Scroll back to the top of the notebook. Execute each of the code cells in order by clicking into each code cell starting at the top and pressing the <Shift><Enter> keys or by clicking into the first code cell and using the Run icon in the menu bar at the top. Read the documentation to gain an understanding of the code that is executing.

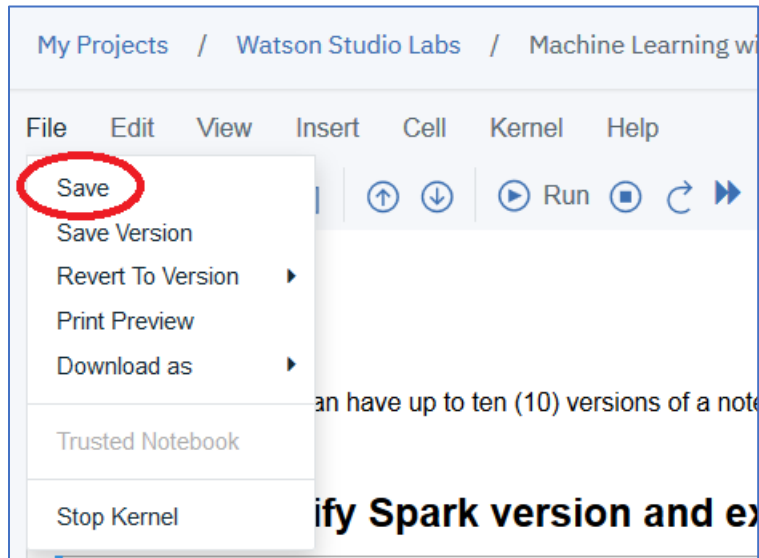


The screenshot shows the IBM Watson Studio interface. At the top, there's a header with 'IBM Watson Studio' and a breadcrumb trail: 'My Projects / Watson Studio Labs / Machine Learning with SparkML'. Below this is a menu bar with 'File', 'Edit', 'View', 'Insert', 'Cell', 'Kernel', and 'Help'. To the right of the menu bar, it says 'Not Trusted | Python 3.6 with Spark'. Below the menu bar is a toolbar with various icons. The 'Run' icon, which is a blue circle with a white play button, is circled in red. Below the toolbar, the notebook content is displayed. The title is 'Machine Learning with Spark ML'. The text describes the notebook's purpose: to explore machine learning using Spark ML, exploit its high-level APIs, and create and tune machine learning pipelines. It mentions using Spark ML Pipelines to combine multiple algorithms into a single pipeline or workflow. The text also states that they will use these predictions to sort the highest priority records for a human to look at, and that they will use as a training set for the algorithm fake data that has been vetted by an analyst as high, medium or low. Finally, it mentions that they will evaluate and cross validate their model to demonstrate the process of determining a best fit model and load the results in the database. The text concludes by stating that they will use generated travel data that has been examined for patterns of Human Trafficking from a DB2 table to do the machine learning.

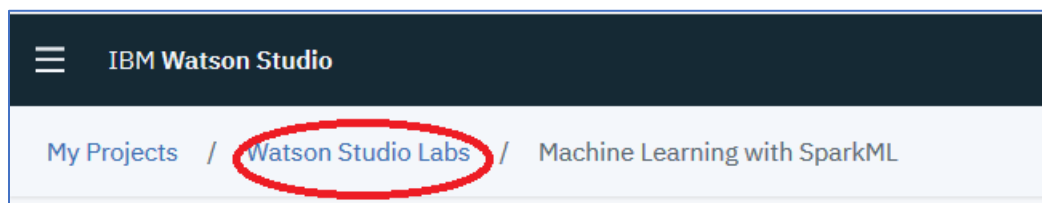
Step 4: Deploy the Model

After the notebook code cells have been executed, the **Female Human Trafficking** model should be saved in the model repository. We are now going to deploy that model on the IBM Cloud.

1. Click on **File** and **Save** to save the notebook.



2. Click on **Watson Studio Labs**.



3. Scroll down to the **Models** section and click on **Female Human Trafficking** under the **Watson Machine Learning models**.

Models

Watson Machine Learning models [New Watson Machine Learning model](#)

NAME	STATUS	TYPE	RUNTIME	LAST MODIFIED	ACTIONS
Female Human Trafficking	trained	ml-lib-2.3	spark-2.3	2 Jun 2019	

4. The model **Overview** page is displayed. The **Overview** page shows the label column and the input schema (scroll down) among other information. Click on **Evaluation**.

MODEL

Female Human Trafficking

[Overview](#) **Evaluation** [Deployments](#) [Lineage](#)

Summary

Machine learning service	WatsonMachineLearning
Model Type	mllib-2.3
Runtime environment	spark-2.3
Training date	2 Jun 2019, 10:41 AM
Label column	VETTING_LEVEL
Latest version	1d5786f0-ac1a-4324-a526-6ea8c2c2d3f6

5. The Evaluation page is displayed. The area under the ROC result is shown. Click on **Deployments**.

MODEL

Female Human Trafficking

[Overview](#) [Evaluation](#) **Deployments** [Lineage](#)

Last Evaluation Result


Version	1d5786f0-ac1a-4324-a526-6ea8c2c2d3f6
Phase	setup
areaUnderROC	0.818

6. Click on **Add Deployment**.

MODEL

Female Human Trafficking

[Overview](#) [Evaluation](#) [Deployments](#) [Lineage](#)

Add Deployment 

NAME	STATUS	DEPLOYMENT TYPE	ACTIONS
Your model is not deployed.			

7. Enter the **Female Human Trafficking Deployed** for the **Name** of the deployment, keep the default **Web Service** for the Deployment type, and click **Save**.

Create Deployment

Define deployment details

Name

Female Human Trafficking Deployed

Description

Deployment description

Deployment type

☒ Web service
 ☐ Batch prediction

Cancel

Save

8. The deployment should complete successfully and when it does you should see **DEPLOY_SUCCESS** for the **STATUS**. Sometimes you need to refresh the browser to update the STATUS. Click on **Female Human Trafficking Deployed** once the deployment successfully completes.

MODEL			
Female Human Trafficking			
Overview	Evaluation	Deployments	Lineage
			Add Deployment
NAME	STATUS	DEPLOYMENT TYPE	ACTIONS
Female Human Trafficking Deployed	DEPLOY_SUCCESS	Web Service	

9. The deployment Overview page is displayed. Click on **Implementation**.

Female Human Trafficking Deployed

Overview

Implementation

Test

Deployment

Name	Female Human Trafficking Deployed
Type	Web Service
Deployment ID	43c56ecc-fa11-4b52-907d-3e4e34ae95d2
Status	DEPLOY_SUCCESS
Asset type	model
Asset name	Female Human Trafficking
Machine learning service	WatsonMachineLearning
Created	02 Jun 2019 11:27am
Last modified	02 Jun 2019 11:28am

10. The Implementation page presents useful information for application developers to call the deployed model. Various client language implementations are presented as code snippets. The scoring endpoint is also shown. Click on **Test**.

Female Human Trafficking Deployed

Overview Implementation **Test**

Implementation [View API Specification](#)

Scoring End-point	https://us-south.ml.cloud.ibm.com/v3/wml_instances/1e0bc606-04b5-43a8-8113-816d171a2c22/deployments/43c56ecc-fa11-4b52-907d-3e4e34ae95d2/online
Authorization: Bearer <token>	Review the WML authentication documentation for details about generating IAM tokens.
ML Instance-ID	The "ML Instance-ID" HTTP header must be populated with the WML instance id, which can be obtained as described here .
Content-type: application/json	Required if the request body is sent in JSON format.

Code Snippets

cURL Java JavaScript Python Scala

```
import urllib3, requests, json

# NOTE: generate iam_token and retrieve ml_instance_id based on provided documentation
header = {'Content-Type': 'application/json', 'Authorization': 'Bearer ' + iam_token, 'ML-Instance-ID': ml_instance_id}

# NOTE: manually define and pass the array(s) of values to be scored in the next line
```

11. The deployment can be tested from this page. Enter **Government** for the **Category**, **5** for **COUNTRIES_VISITED_COUNT**, **GH** for **PASSPORT_COUNTRY_CODE**, and **2000** for **BIRTH_YEAR**, and then click **Predict**.

Female Human Trafficking Deployed

Overview Implementation **Test**

Enter input data


Category
Government

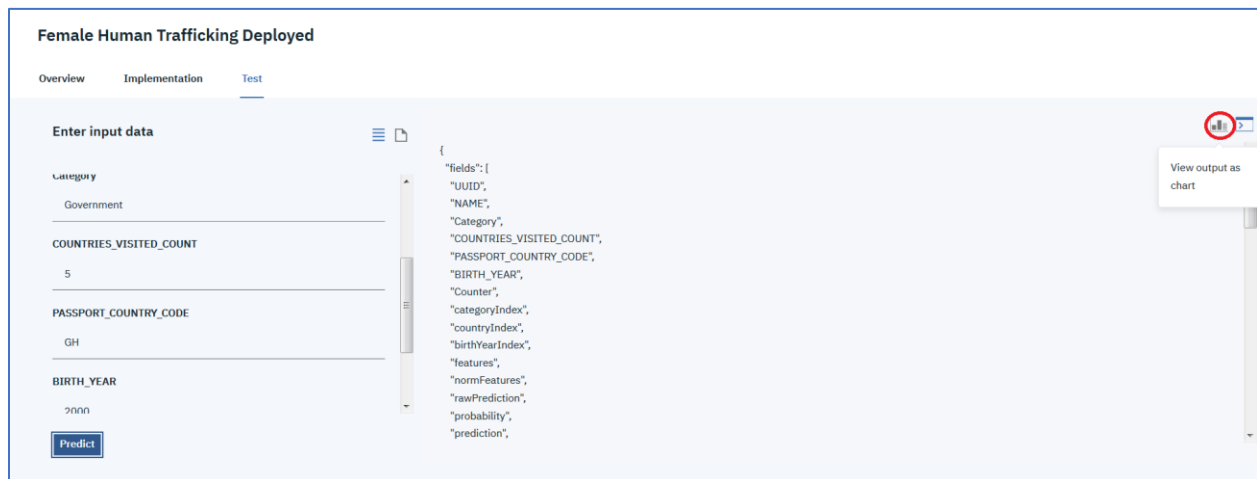
COUNTRIES_VISITED_COUNT
5

PASSPORT_COUNTRY_CODE
GH

BIRTH_YEAR
2000

Predict

12. The results should be displayed either in a chart or in JSON. If displayed in JSON, click on the chart icon .



Female Human Trafficking Deployed

Overview Implementation **Test**

Enter input data

Category
Government

COUNTRIES_VISITED_COUNT
5

PASSPORT_COUNTRY_CODE
GH

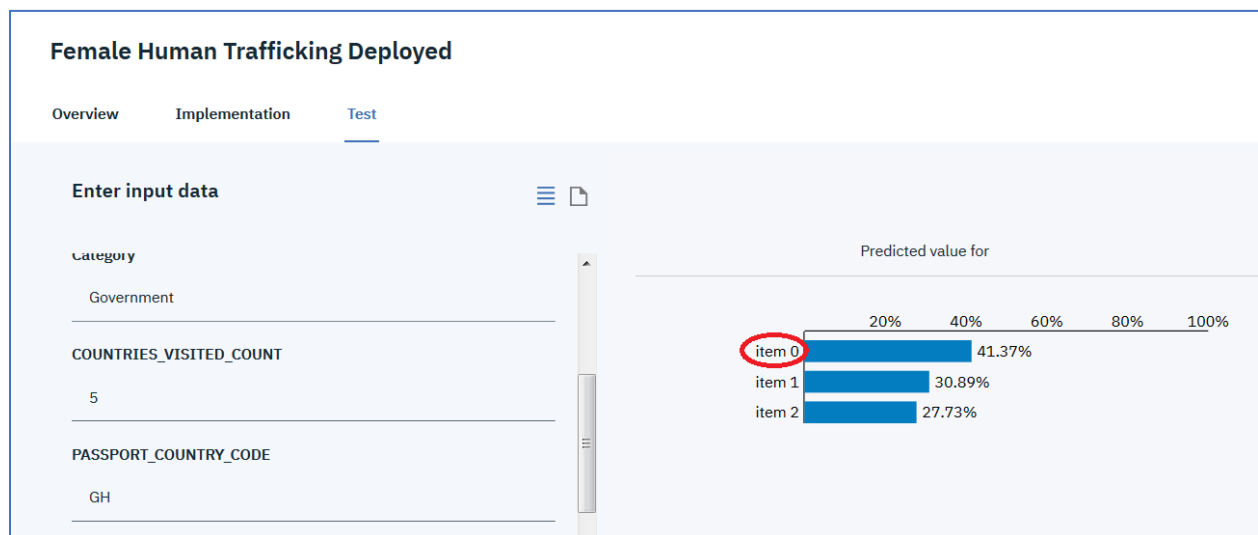
BIRTH_YEAR
2000

Predict

```
{
  "fields": [
    "UUID",
    "NAME",
    "Category",
    "COUNTRIES_VISITED_COUNT",
    "PASSPORT_COUNTRY_CODE",
    "BIRTH_YEAR",
    "Counter",
    "categoryIndex",
    "countryIndex",
    "birthYearIndex",
    "features",
    "normFeatures",
    "rawPrediction",
    "probability",
    "prediction"
  ]
}
```

View output as chart

13. The results are displayed in chart form. For this example, the deployed model predicts item 0 (which refers to “30” or Low probability).



14. Click on **Watson Studio Labs** to return to the project Assets page.

You have completed Lab-5!

- ✓ Joined data from three sources.
- ✓ Identified labels and transformed data.
- ✓ Conducted feature engineering for algorithm data.
- ✓ Declared a machine learning model.
- ✓ Created the Pipeline for data transforms and training.
- ✓ Trained the model.
- ✓ Evaluated and showed model results.

- ✓ Automatically tuned model.
- ✓ Scored data and loaded into a new DB2 table.
- ✓ Saved the model to the model repository.
- ✓ Deployed and tested the model using the UI