

# Watson Studio: Heart Disease modeling with Jupyter Notebooks

## Introduction

In this lab, you use a Jupyter Notebook to train a model using the XGBoost library to classify whether a person has heart disease or not. In addition to training a model, the notebook also explains how to persist a trained model to the IBM Watson Machine Learning repository, and deploy the model as a REST service.

To train and test the heart disease prediction model, you are using an open source data set published in the University of California, Irvine (UCI) Machine Learning Repository.

The notebook environment includes Python 3.5 runtime, XGBoost 0.6 and Scikit-Learn 0.17.

## Objectives

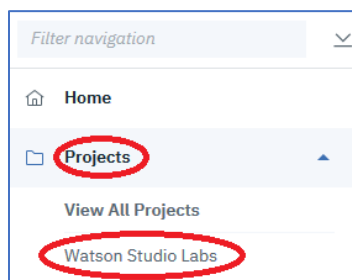
Upon completing the lab, you will know how to:

1. Load a CSV file into Pandas DataFrame.
2. Data exploration using Pixiedust
3. Prepare data for training and evaluation.
4. Create, train, and evaluate a XGBoost model.
5. Visualize the importance of features that were used to train the model.
6. Use cross validation to select optimal model hyperparameters based on a parameter grid
7. Persist best model in Watson Machine Learning repository using Python client library.
8. Deploy the model for online scoring using the Watson Machine Learning's REST APIs

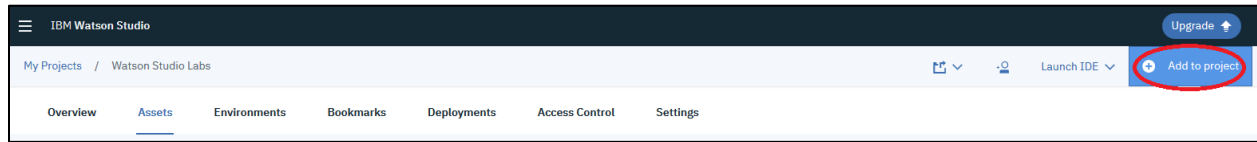
## Lab Steps

### Step 1 - Create a Jupyter Notebook

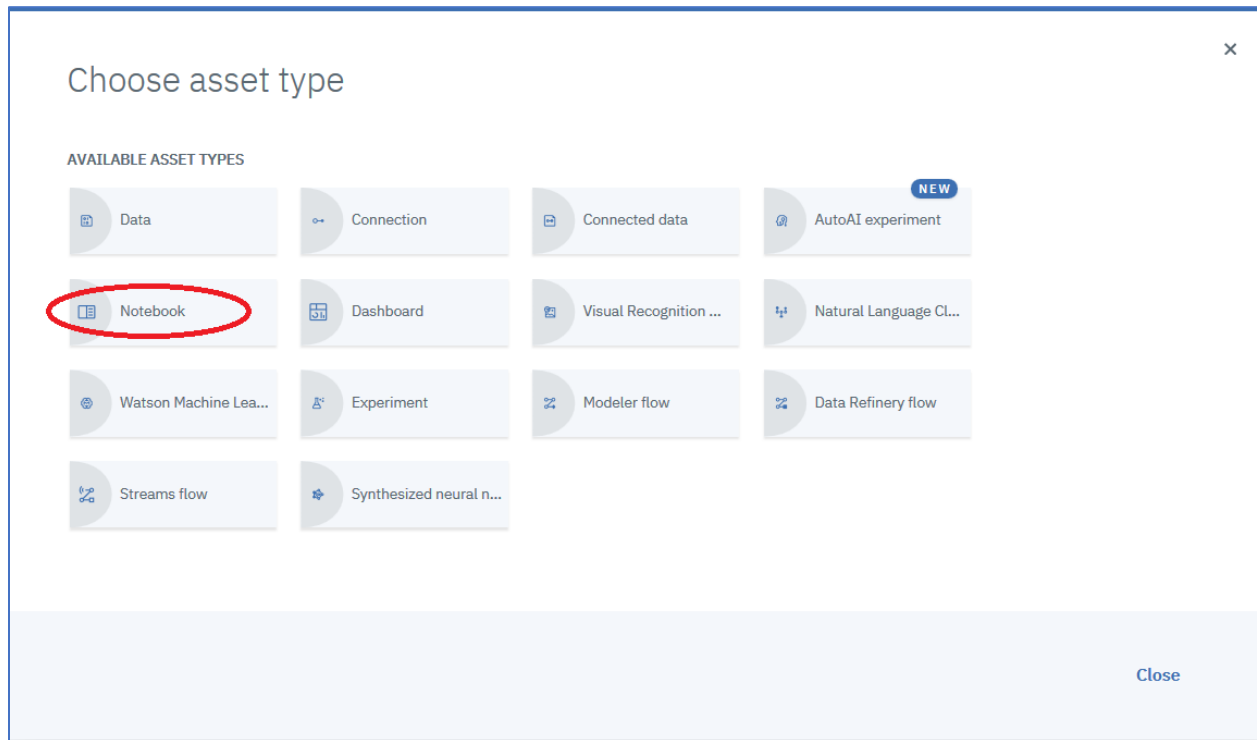
1. Click on the hamburger icon , then click on **Projects**, and then **Watson Studio Labs** (or whatever you named the project)



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



3. Click on **Notebook**



4. Click on **From URL** under **New Notebook**, enter **Heart Disease** for the **Name**, and optionally enter a **Description**. Close the deprecation panel.

**Select the Runtime. You will need to change the default selection. MAKE SURE TO SELECT Default Spark Python 3.5 XS (Driver with 1vCPU ...**

Cut and paste the following url into the **Notebook URL** field.

[https://github.com/bleonardb3/AA\\_POT\\_07-30/blob/master/Lab-2/Heart%20Disease.ipynb](https://github.com/bleonardb3/AA_POT_07-30/blob/master/Lab-2/Heart%20Disease.ipynb)

Click **Create Notebook**.

New notebook

Blank From file **From URL**

Name  
**Heart Disease** 27 characters remaining

Description (optional)  
Type your description here 500 characters remaining

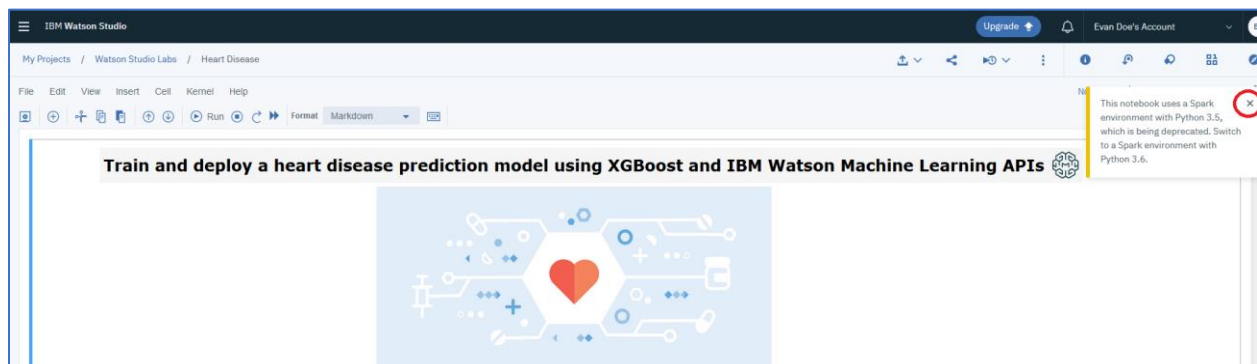
Select runtime  
**Default Spark Python 3.5 XS (Driver with 1 vCPU and 4 GB RAM, 2 executors with 1 vCPU and 4 GB RAM)**  
The selected runtime uses one driver with 1 vCPU and 4 GB RAM, and 2 executors each with 1 vCPU and 4 GB RAM.  
This runtime consumes 1.5 capacity units per hour.  
[Learn more](#) about capacity unit hours and Watson Studio pricing plans.

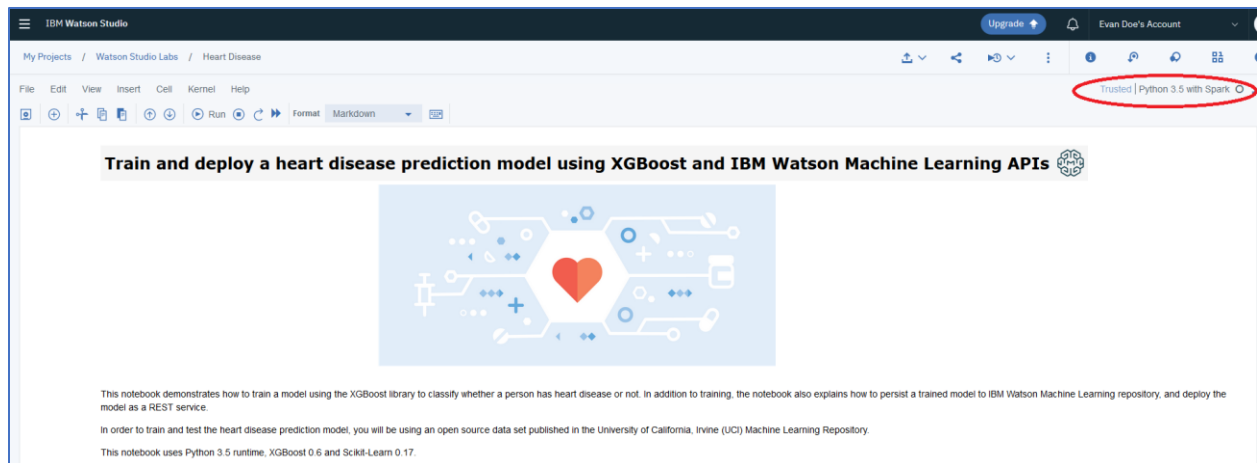
Notebook URL  
**https://github.com/bleonard3/AA\_POT\_07-30/blob/master/Lab-2/Heart%20Disease.ipynb**

Cancel **Create Notebook**

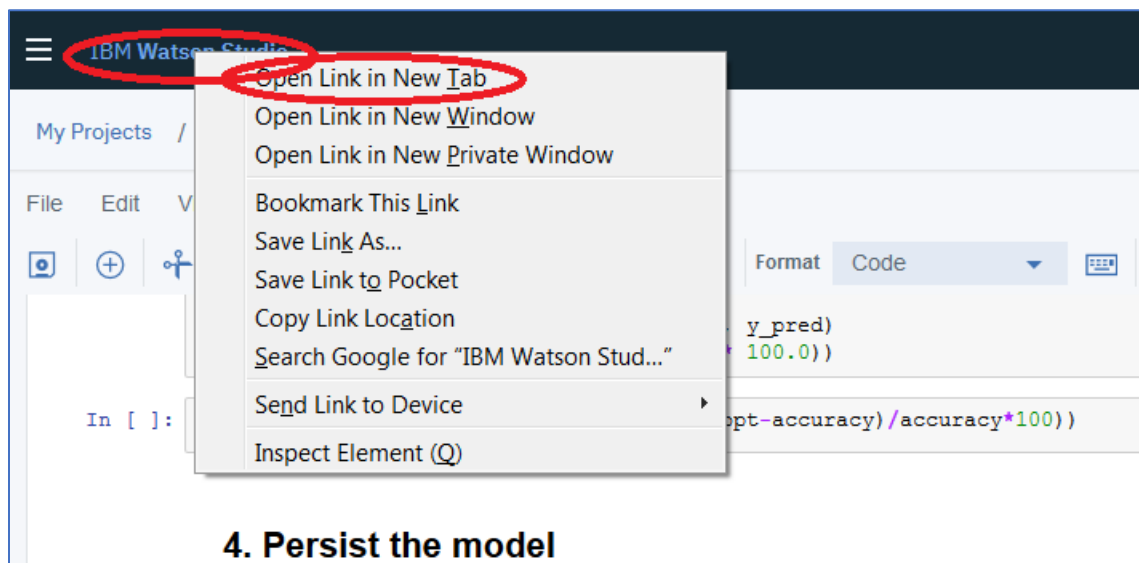
Spark Environments with Python 3.5 are being deprecated. Choose a Spark environment with Python 3.6 instead.

5. Before executing the code in the notebook, we need to do the following:
  - a. Close the deprecation panel (See below)
  - b. Please make sure the notebook has **Python 3.5 with Spark** in the top right corner (See below).
  - c. Obtain the credentials of the Watson Machine Learning service and copy those credentials into a designated notebook cell. We will need these credentials to work with the Watson Machine Learning API. The procedure to obtain the credentials will be described below.





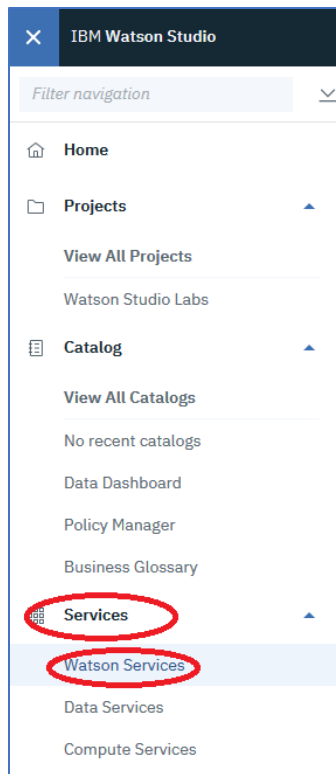
6. Right-click on **IBM Watson Studio**, and then click on **Open Link in New Tab**.




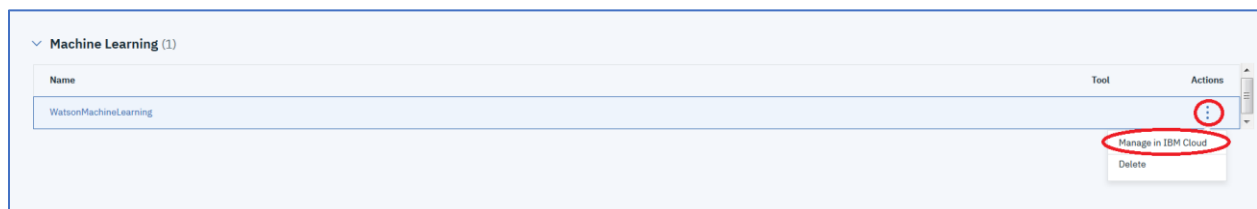
7. Click on the new **Watson Studio** browser tab.



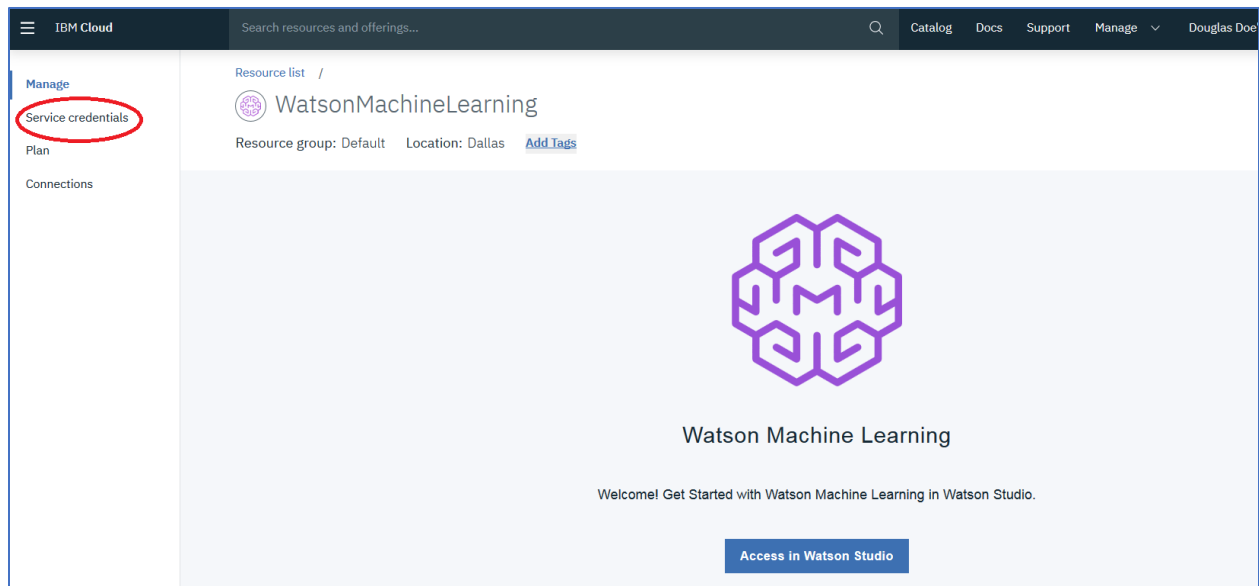
8. Click on the hamburger icon , and then **Services**, and **Watson Services**.



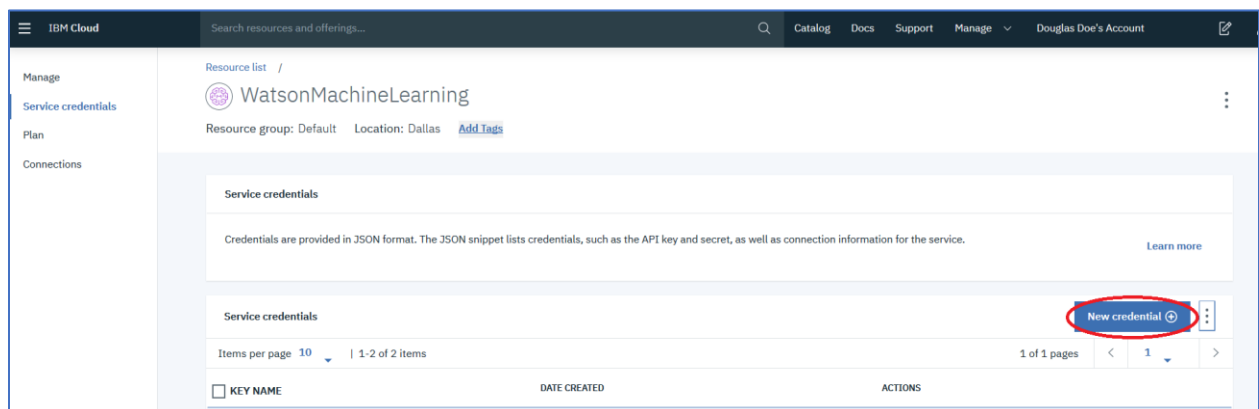
9. Hover over WatsonMachineLearning, click on the vertical ellipse on the right. , and click on **Manage in IBM Cloud**.



10. A new browser tab will be created titled **Service Details – IBM Cloud**. This browser tab will be interfacing with the IBM Cloud user interface. Click on **Service credentials**.



11. Click on **New Credentials+**.



12. Click on **Add**.

×

## Add new credential

**Name:**

Service credentials-1

**Role:**

Writer

**Select Service ID (Optional)**

Select Service ID...

**Add Inline Configuration Parameters (Optional):**

Cancel

Add

13. Click on ▼ next to View Credentials in the Service Credentials-1 row.

Service credentials

Credentials are provided in JSON format. The JSON snippet lists credentials, such as the API key and secret, as well as connection information for the service. [Learn more](#)

Service credentials

New credential

Items per page 10 | 1-2 of 2 items 1 of 1 pages < 1 >

<input type="checkbox"/> KEY NAME	DATE CREATED	ACTIONS
<input type="checkbox"/> wdp-writer	JUN 25, 2019 - 02:42:46 PM	View credentials <div></div> <div></div>
<input type="checkbox"/> Service credentials-1	JUL 5, 2019 - 06:00:17 PM	View credentials <div></div> <div></div>

14. Click on the copy icon



Service credentials

Items per page 10 | 1-2 of 2 items 1 of 1 pages < 1 >

KEY NAME	DATE CREATED	ACTIONS
<input type="checkbox"/> wdp-writer	JUN 25, 2019 - 02:42:46 PM	View credentials View credentials View credentials
<input type="checkbox"/> Service credentials-1	JUL 5, 2019 - 06:00:17 PM	View credentials View credentials View credentials

```
{
  "apikey": "pk3_Y0ZmsMQQFNK1ua189gn9rABBLenhAhk9-TjX7GL0",
  "iam_apikey_description": "Auto-generated for key f4f88830-8943-4f2c-a987-a9597d3455c7",
  "iam_apikey_name": "Service credentials-1",
  "iam_role_crn": "crn:v1:bluemix:public:iam::::serviceRole:Writer",
  "iam_serviceid_crn": "crn:v1:bluemix:public:iam-identity::a/ae951831c9a94c28bb9e44efe8e66ac8::serviceid:ServiceId-84c2cec8-d92c-40a2-86b9-b81cc61ba897",
  "instance_id": "2ad707c8-a75e-469c-8cb1-97dd8b302d81",
  "password": "47ee3c0f-ad6a-47ae-a771-1486c5c76cd",
  "url": "https://us-south.ml.cloud.ibm.com",
  "username": "f4f88830-8943-4f2c-a987-a9597d3455c7"
}
```

15. Click on the Heart Disease browser tab to go back to the notebook. Should be two browser tabs to the left of the **Service Details – IBM Cloud**.

IBM Watson Studio X IBM Watson Studio X Service Details - IBM Cloud X

16. Type Ctrl-F and type in Paste to find the notebook cell to paste in the credentials.

#### 4. Persist the model

In this section store the XGBoost model in the Watson Machine Learning repository by using Watson Machine Learning repository service Python client libraries.

```
In [ ]: from repository.mlrepository import MetaNames
from repository.mlrepository import MetaProps
from repository.mlrepositoryclient import MLRepositoryClient
from repository.mlrepositoryartifact import MLRepositoryArtifact
```

Authenticate to Watson Machine Learning service on the IBM Cloud.

Action: **PASTE** CREDENTIALS FROM YOUR INSTANCE OF WATSON MACHINE LEARNING INTO THE FOLLOWING CELL.

```
In [ ]: # Need to insert credentials obtained from above steps
wml_credentials = {
    "instance_id": "",
    "password": "",
    "url": "",
    "username": ""
}
```

```
In [ ]: ml_repository_client = MLRepositoryClient(wml_credentials['url'])
ml_repository_client.authorize(wml_credentials['username'], wml_credentials['password'])
```

#### 4.1: Save a XGBoost model in the Machine Learning Repository

In this subsection you will learn how to save a model artifact to your Watson Machine Learning instance by using the Watson Machine Learning repository Python client package.

Paste

17. Highlight the text from the starting curly brace { to the ending curly brace }.



#### 4. Persist the model

In this section store the XGBoost model in the Watson Machine Learning repository by using Watson Machine Learning repository service Python client libraries.

```
In [ ]: from repository.mlrepository import MetaNames
from repository.mlrepository import MetaProps
from repository.mlrepositoryclient import MLRepositoryClient
from repository.mlrepositoryartifact import MLRepositoryArtifact
```

Authenticate to Watson Machine Learning service on the IBM Cloud.

Action: PASTE CREDENTIALS FROM YOUR INSTANCE OF WATSON MACHINE LEARNING INTO THE FOLLOWING CELL.

```
In [ ]: # Need to insert credentials obtained from above steps
wml_credentials = {
    "instance_id": "",
    "password": "",
    "url": "",
    "username": ""
}
```

18. Right-click on the highlighted text and click **Paste**.

#### 4. Persist the model

In this section store the XGBoost model in the Watson Machine Learning repository by using Watson Machine Learning repository service Python client libraries.

```
In [ ]: from repository.mlrepository import MetaNames
from repository.mlrepository import MetaProps
from repository.mlrepositoryclient import MLRepositoryClient
from repository.mlrepositoryartifact import MLRepositoryArtifact
```

Authenticate to Watson Machine Learning service on the IBM Cloud.

Action: PASTE CREDENTIALS FROM YOUR INSTANCE OF WATSON MACHINE LEARNING INTO THE FOLLOWING CELL.

```
In [ ]: # Need to insert credentials obtained from above steps
wml_credentials = {
    "instance_id": "",
    "password": "",
    "url": "",
    "username": ""
}
```

```
In [ ]: ml_repository_client = MLRepositoryClient(wml_credentials['instance_id'],
ml_repository_client.authenticate(wml_credentials['password'])
```

#### 4.1: Save a XGBoost model

In this subsection you will learn how to save a trained XGBoost model to the Watson Machine Learning repository.

Paste

19. The credentials should appear similar to below with different values.

#### 4. Persist the model

In this section store the XGBoost model in the Watson Machine Learning repository by using Watson Machine Learning repository service Python client libraries.

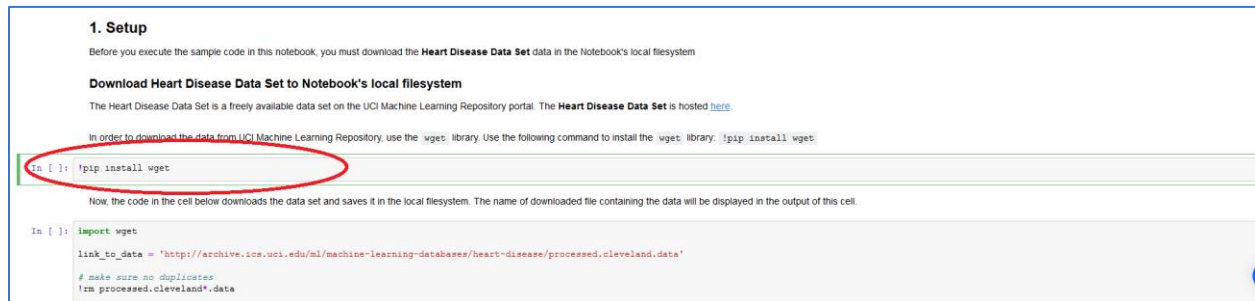
```
In [ ]: from repository.mlrepository import MetaNames
from repository.mlrepository import MetaProps
from repository.mlrepositoryclient import MLRepositoryClient
from repository.mlrepositoryartifact import MLRepositoryArtifact
```

Authenticate to Watson Machine Learning service on the IBM Cloud.

Action: PASTE CREDENTIALS FROM YOUR INSTANCE OF WATSON MACHINE LEARNING INTO THE FOLLOWING CELL.

```
In [ ]: # Need to insert credentials obtained from above steps
wml_credentials = {
    "apikey": "pk3_YOZmaMQPFNKlual09gn9rABBLenhAhk9-TjX7GLO",
    "iam_apikey_description": "Auto-generated for key f4f88830-8943-4f2c-a987-a9597d3455c7",
    "iam_apikey_name": "Service credentials-1",
    "iam_role_crn": "crn:v1:bluemix:public:iam:::serviceRole:Writer",
    "iam_serviceid_crn": "crn:v1:bluemix:public:iam-identity::a/ae951031c9a94c20bb9e44efe8e66ac0::serviceid:ServiceId-84c2cec8-d92c-40a2-86b9-b81cc61ba897",
    "instance_id": "2ad707c8-a75e-460c-8cb1-97dd8b302d81",
    "password": "47ee3c0f-ad6a-47ae-a771-1406c5cf76cd",
    "url": "https://us-south.ml.cloud.ibm.com",
    "username": "f4f88830-8943-4f2c-a987-a9597d3455c7"
}
```

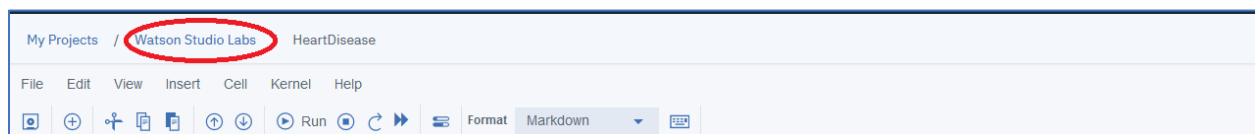
20. Return to the top of the notebook, read through the documentation in the beginning and then select the first code cell.



For those not familiar with Jupyter notebooks, read below.

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.

21. Execute each of the notebook cells in order (either by typing in **<Shift><Enter>** or using the **Run** menu option). Read the notebook documentation to gain an understanding of the code that is executing. **When all the cells in the notebook have been successfully executed, please return to this document, and continue with Step 22.**
22. The notebook built, trained, saved, and deployed a machine learning model. Click Watson Studio Labs exit out of the notebook.



23. Scroll down the **Assets** page until you see the **Models** heading. The model listed was generated programmatically from the notebook using the Watson Machine Learning APIs.

Models

Watson Machine Learning models New Watson Machine Learning model

NAME	STATUS	TYPE	RUNTIME	LAST MODIFIED	ACTIONS
XGB_Heart_Disease_Detection	trained	scikit-learn-0.19	python-3.5	30 Jun 2019	

24. Scroll back to the top of the **Assets** page and click **Deployments**.

My Projects / Watson Studio Labs

Overview	Assets	Environments	Bookmarks	Deployments	Access Control	Settings
----------	--------	--------------	-----------	-------------	----------------	----------

25. The deployment listed was generated programmatically from the notebook using the Watson Machine Learning APIs. It is a Web service deployment.

My Projects / Watson Studio Labs

Overview Assets Environments Bookmarks Deployments Access Control Settings

Deployments

NAME	TYPE	STATUS	ACTIONS
xgb_heart_disease_v1	Web service	ready	

## You have completed Lab-2!

- ✓ Loaded a CSV file into Pandas DataFrame.
- ✓ Explored data using Pixiedust
- ✓ Prepared data for training and evaluation.
- ✓ Created, trained, and evaluated a XGBoost model.
- ✓ Visualized the importance of features that were used to train the model.
- ✓ Used cross validation to select optimal model hyperparameters based on a parameter grid
- ✓ Persisted the best model in Watson Machine Learning repository using Python client library.
- ✓ Deployed the model for online scoring using the Watson Machine Learning's REST APIs
- ✓ Visualized the model and the deployment in the Watson Studio UI.