

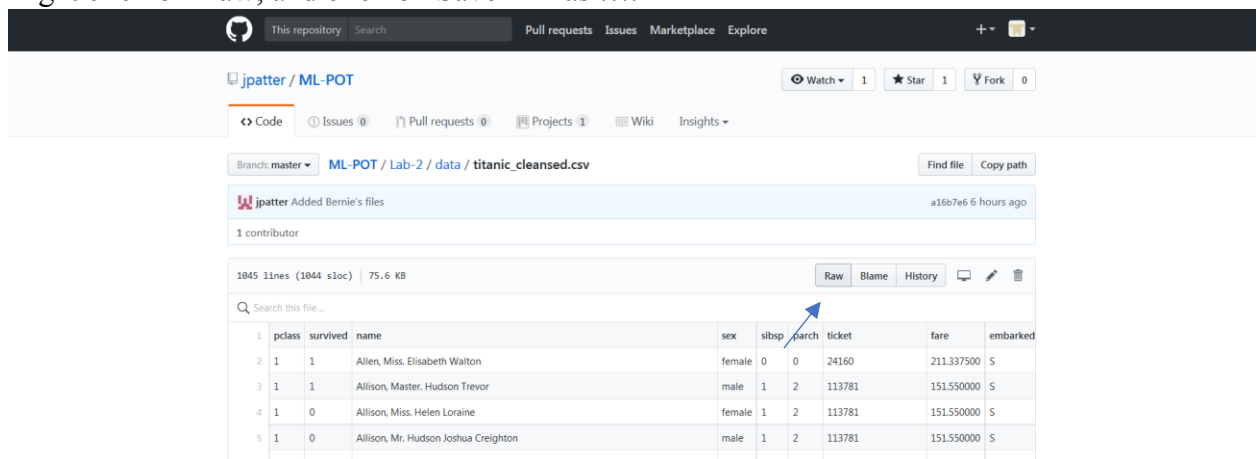
Watson Machine Learning Overview

This lab will introduce the Watson Machine Learning capability using the Titanic dataset. The lab will consist of the following steps:

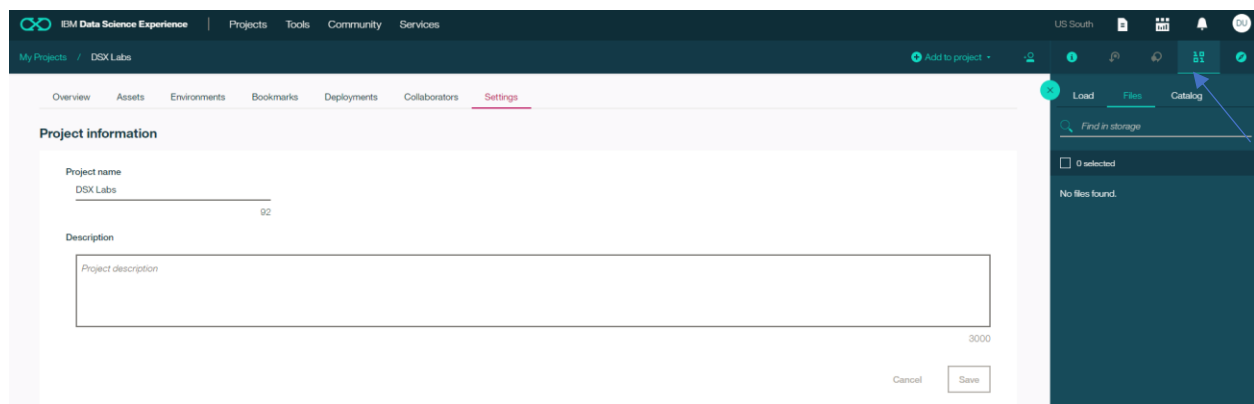
1. Adding a data asset to the Watson Studio Labs project
2. Creating a Model to predict whether a passenger would survive
3. Deploying and Testing the Model
4. Deploying a simple web front-end and connecting it to the Titanic deployed model.

Step 1: Adding a Data Asset to the project

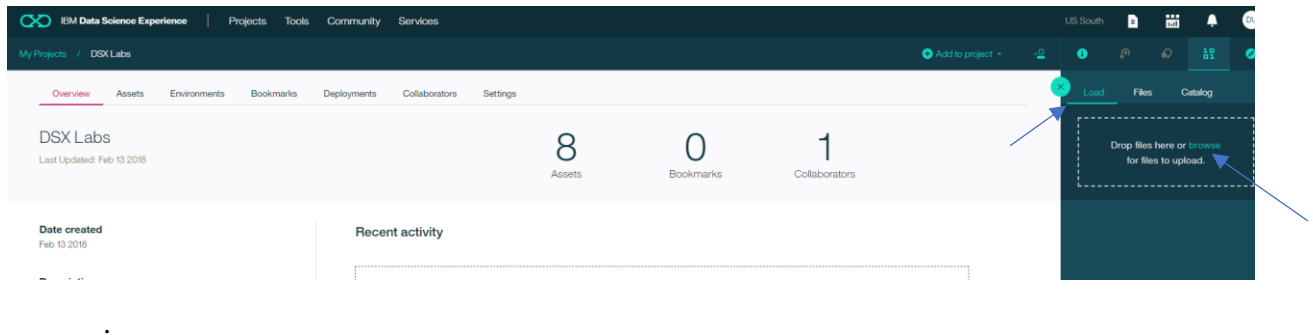
1. Download the Titanic data file from the following location by clicking on the link [Cleansed Titanic Data Set](#) and following the instructions below.
2. Right click on Raw, and click on Save link as



3. Go back to your Watson Studio Labs project. Click on the  icon.

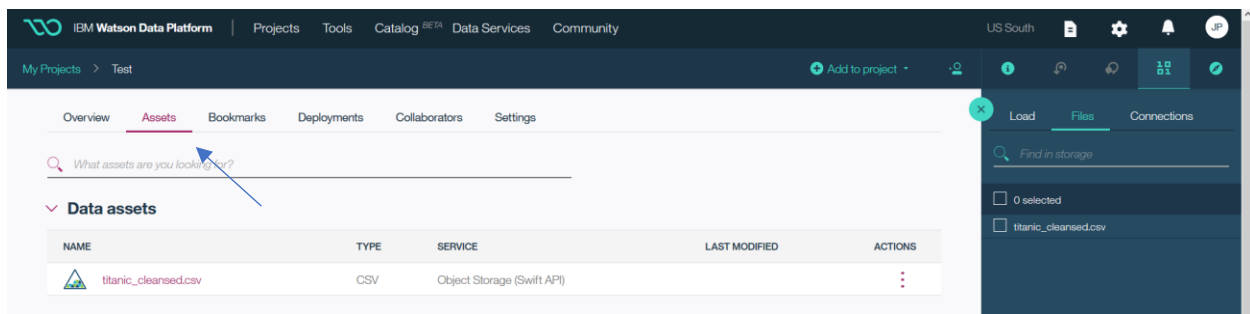


- Click on **Load** and then **browse** and then go to the folder where the titanic_cleansed.csv is stored. Select titanic_cleansed.csv and then click Open.

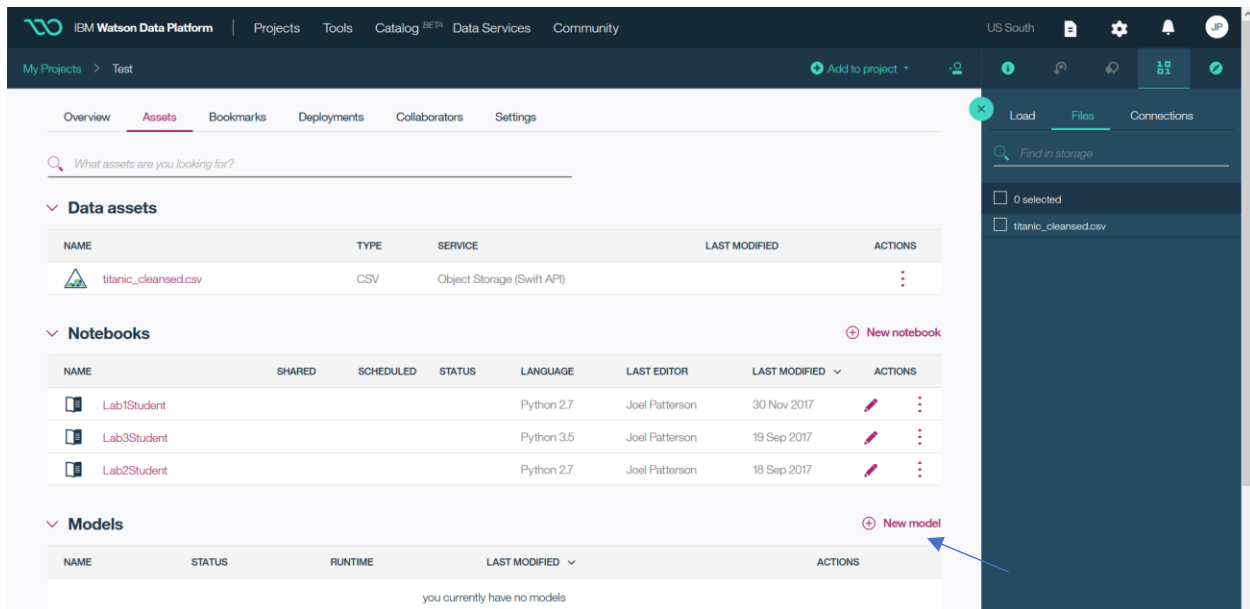


Step 2: Create a Model to predict survival

- Click on the **Assets** Tab



- Click on **New model**.



3. Enter a **Model Name** (eg Titanic), optionally a **Description**, Select **Manual**, and click on **Create**.

IBM Data Science Experience | Projects Tools Community Services

US South

New model BETA

Define model details

Name

Model name

Description

Model description

Machine Learning Service

Machine Learning-bb

Select model type

☒ Model builder ☐ From sample

Spark Service

Spark

Automatic
Prepare my data and create a model automatically

Manual
Let me prepare my data and select which models to train

Need something more flexible? Create a [notebook](#) or design an [SPSS Modeler flow](#).

Cancel Create

4. Click on the `titanic_cleansed.csv` and click on **Next**

IBM Watson Data Platform | Projects Tools Catalog BETA Data Services Community

US South

My Projects > Test > Titanic-ML

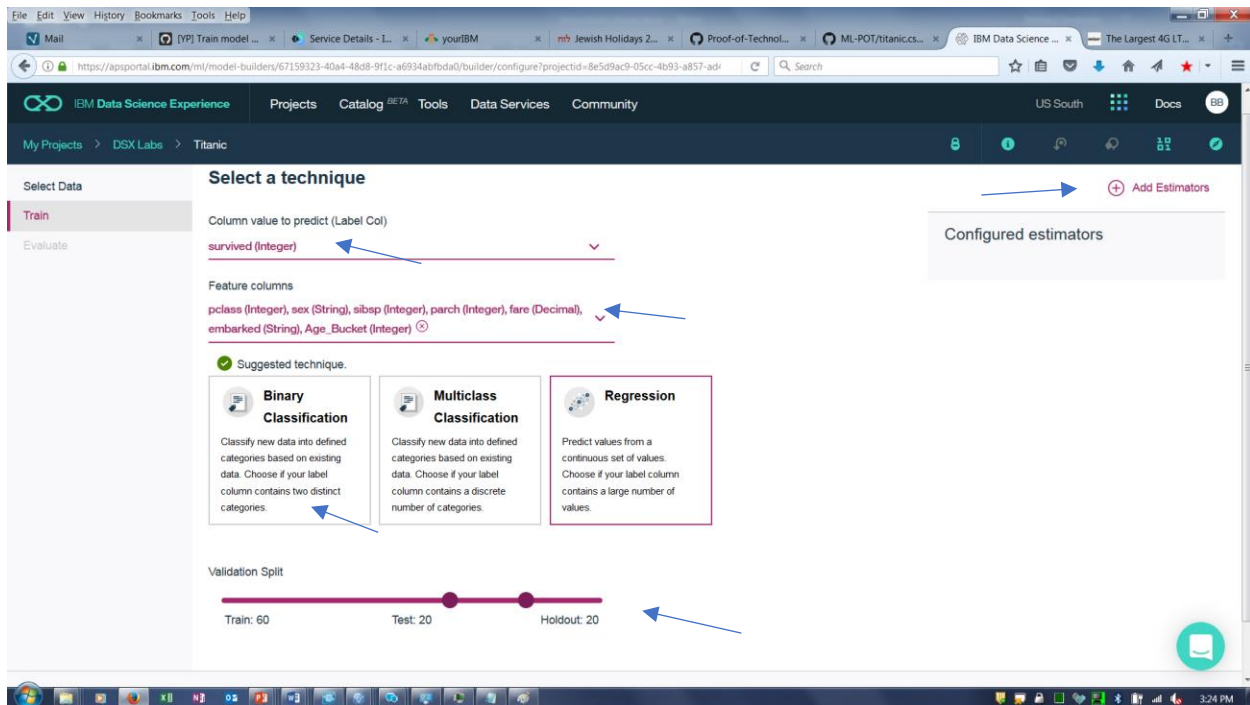
Select data asset

The model builder currently supports CSV files and IBM Db2 Warehouse on Cloud data assets.

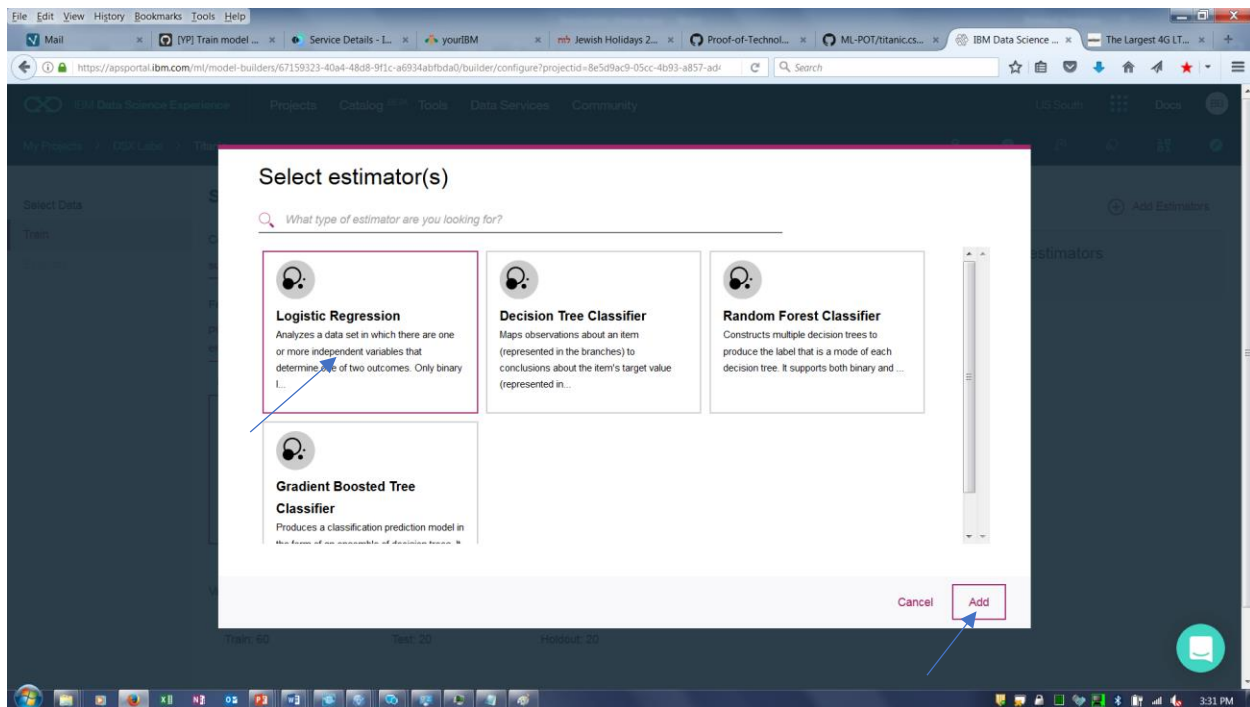
NAME	TYPE	SERVICE
titanic_cleansed.csv	CSV	Object Storage (Swift API)

Close Next

5. For **Column value to predict (Label Col)** select **survived**. For **Feature columns** select the following features (**pclass, sex, sibsp, parch, fare, embarked, Age_Bucket**) . Click on the **Binary Classification** Box (which is suggested by the service). Adjust the **Validation Split** as desired. Click on **Add Estimators** to add the specific models to use.



6. Select **Logistic Regression**. You can select more if you wish to see the results of multiple models. Select **Add**.



7. Select the **Next** button.

Select a technique

You cannot change label column, feature columns, model type, or validation split after adding an estimator. You must first delete all estimators in order to make changes to these attributes.

Column value to predict (Label Col)
survived (Integer)

Feature columns
pclass (Integer), sex (String), sibsp (Integer), parch (Integer), fare (Decimal), emb

✓ Suggested technique.

Binary Classification

Classify new data into defined categories based on existing data. Choose if your label column contains two distinct categories.

Multiclass Classification

Classify new data into defined categories based on existing data. Choose if your label column contains a discrete number of categories.

Regression

Predict values from a continuous set of values. Choose if your label column contains a large number of values.

Validation Split

Train: 60 Test: 20 Holdout: 20

Configured estimators

- Logistic Regression (Not Yet Trained)
- Decision Tree Classifier (Not Yet Trained)
- Random Forest Classifier (Not Yet Trained)
- Gradient Boosted Tree Classifier (Not Yet Trained)

Close Previous **Next**

- The system trains and evaluates each model. If more than one model was selected, the models would be listed in descending order of quality with the best result at the top. Note: if a model fails to run (rare, but happens), select Previous, delete that model and re-add it. Then run again. Click on **Logistic Regression** (if it is the best) and then click **Save**.

Select model

	ESTIMATOR TYPE	STATUS	PERFORMANCE	AREA UNDER ROC CURVE	AREA UNDER PR CURVE	LAST EVALUATION	ACTIONS
<input checked="" type="radio"/>	LogisticRegression	Trained & Evaluated	Good	0.88036	0.87011	30 Nov 2017, 4:31 PM	⋮
<input type="radio"/>	RandomForestClassifier	Trained & Evaluated	Good	0.87617	0.8728	30 Nov 2017, 4:32 PM	⋮
<input type="radio"/>	GBClassifier	Trained & Evaluated	Good	0.8003	0.81388	30 Nov 2017, 4:33 PM	⋮
<input type="radio"/>	DecisionTreeClassifier	Trained & Evaluated	Fail	0.4506	0.49969	30 Nov 2017, 4:32 PM	⋮

Close Previous **Save**

- Click **Save** again on the next screen.

! Save model?

Are you sure that you want to save this model?

Cancel

Save

10. The system displays the model training summary. Click on Evaluation.

The screenshot shows the IBM Watson Data Platform interface. The top navigation bar includes 'Projects', 'Tools', 'Catalog', 'Data Services', and 'Community'. The breadcrumb trail is 'My Projects > Test > Titanic'. The 'Titanic' page has three tabs: 'Overview', 'Evaluation', and 'Deployments'. The 'Evaluation' tab is selected, and a blue arrow points to it. Below the tabs is a 'Summary' section with a table of model details.

Property	Value
Machine learning service	ML-113017
Runtime environment	spark-2.0
Training date	30 Nov 2017, 4:39 PM
Label column	survived
Latest version	e00e9080-f42c-4e68-b45f-3430c7b86a90
Model builder details	View

11. The system displays the recorded evaluation statistics for the run. You can also set up Continuous Learning (Performance Monitoring) on this screen. We will not do this now.

The screenshot shows the IBM Data Science Experience interface. The top navigation bar includes 'Projects', 'Tools', 'Community', and 'Services'. The breadcrumb trail is 'My Projects / DSX Labs / Titanic'. The 'Titanic' page has three tabs: 'Overview', 'Evaluation', and 'Deployments'. The 'Evaluation' tab is selected. Below the tabs is a 'Last Evaluation Result' section with a table of evaluation statistics.

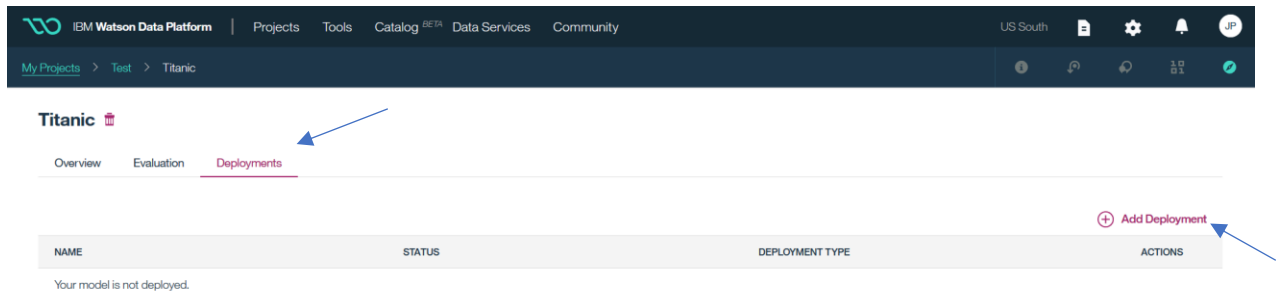
Property	Value
Version	2090ebba-6d3d-4b7e-bed8-dfbd90b870ba
Phase	setup
AreaUnderPR	0.823
AreaUnderROC	0.864

Below the table is a 'Performance Monitoring' section with a text box containing instructions on how to configure performance monitoring.

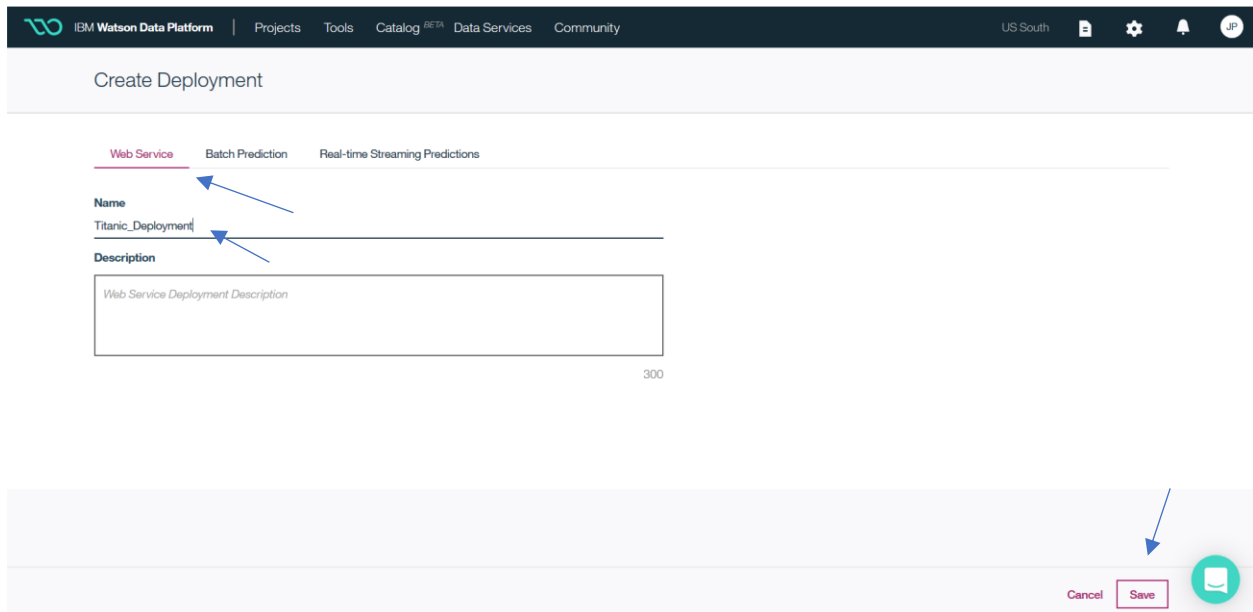
Step 3: Deploying a Model

We can deploy the model to enable applications to invoke it via an API call. This is called a Web Service deployment or Online deployment.

1. Select the **Deployments** Tab
2. Click on **Add Deployment**



3. Three options for deployment are available. For this lab, we are going to embed the predictive model scoring in a web application. Therefore, select the **Web Service** tab, enter **Titanic_Deployment** for **Name**, and click on **Save**.



4. The system responds with an acknowledgement that the model was successfully deployed. Click on **Titanic_Deployment** to test the deployed API.

Titanic

Overview Evaluation **Deployments**

+ Add Deployment			
NAME	STATUS	DEPLOYMENT TYPE	ACTIONS
Titanic_Deployment	ACTIVE	Web Service	

5. The system displays information about the deployed service. Click on **Test** to test out the API.

Titanic_Deployment

Overview Implementation **Test**

Deployment

Name	Titanic_Deployment
Type	Web Service
Deployment ID	2e4ee472-49db-4a21-8156-d098116d278b
Status	ACTIVE
Machine learning service	Machine Learning-bb
Created	14 Feb 2018 10:22am
Last modified	14 Feb 2018 02:00pm

6. Enter values for the input fields or just use the defaults, and then click on **Predict**. Note that the values inputted for any of the fields not included in the model parameters (e.g. name) will not affect the prediction.

Titanic_Deployment

Overview Implementation **Test**

Input data

pclass
1

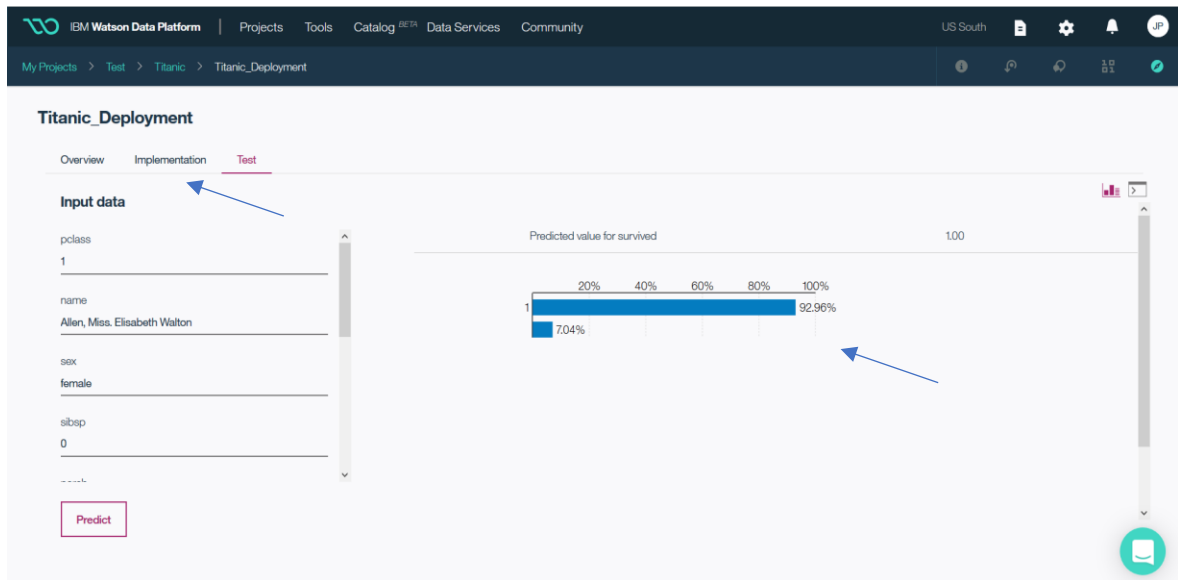
name
Allen, Miss. Elisabeth Walton

sex
female

sibsp
0

Predict

7. The predicted result is returned. Now click on the Implementation tab.



8. The Implementation panel provides information for the application developers to invoke the deployed model. It includes sample code in various programming languages and the scoring endpoint to be used when invoking the web service. Open Windows Notepad to copy and paste the scoring endpoint. We will be using this endpoint in Step 4.

The screenshot shows the IBM Data Science Experience interface for the 'Titanic_Deployment' project. The 'Implementation' tab is selected, displaying the scoring endpoint and code snippets.

Implementation

Field	Value
Scoring End-point	https://ibm-watson-ml.mybluemix.net/v3/wml_instances/c4802b9a-5cb3-4d8e-9884-5255e55484e5/published_models/f1485ef-810a-4239-8db6-37312d8d32a7/deployments/2e4ee472-49db-4a21-8156-d00110d278b/online
Authorization: Bearer <token>	See code snippets below for information on how to retrieve the WML Authorization Token to be passed with scoring requests.
Content-type: application/json	Required if the request body is sent in JSON format.

Code Snippets

cURL Java JavaScript Python Scala

```
# retrieve your $WML_SERVICE_CREDENTIALS_USERNAME, $WML_SERVICE_CREDENTIALS_PASSWORD, and $WML_SERVICE_CREDENTIALS_URL from the
# Service credentials associated with your IBM Cloud Watson Machine Learning Service Instance

curl --basic --user $WML_SERVICE_CREDENTIALS_USERNAME:$WML_SERVICE_CREDENTIALS_PASSWORD $WML_SERVICE_CREDENTIALS_URL/v3/identity/token

# the above CURL request will return an auth token that you will use as $WML_AUTH_TOKEN in the scoring request below
# TODO: manually define and pass values to be scored below
curl -X POST --header 'Content-Type: application/json' --header 'Accept: application/json' --header 'Authorization: Bearer $WML_AUTH_TOKEN' -d '{"fields": [{"pclass", "name", "sex", "sibsp", "parch", "ticket", "fare", "embarked", "Age"
```

Step 4: Deploy a simple web front-end to invoke the Watson Machine Learning service

This section will provide an example of a simple Python Flask front-end that invokes the Titanic scoring API demonstrating embedding machine learning in a web app. You will click on a link below that will deploy the sample Python web application into your IBM Cloud account. A toolchain will be set up for continuous delivery of the application. The application code will be cloned from a public Git repository into a private Git repo in your account that will be set up as part of the toolchain. Each time you commit changes to the repo, the app will be built and deployed.

The toolchain uses tools that are part of the Continuous Delivery service. If an instance of that service isn't already in your organization, when you click **Deploy**, it is automatically added with the free [Lite](#) plan selected.

You will need to customize the application to provide the credentials for your Watson Machine Learning service, and to provide the scoring endpoint.

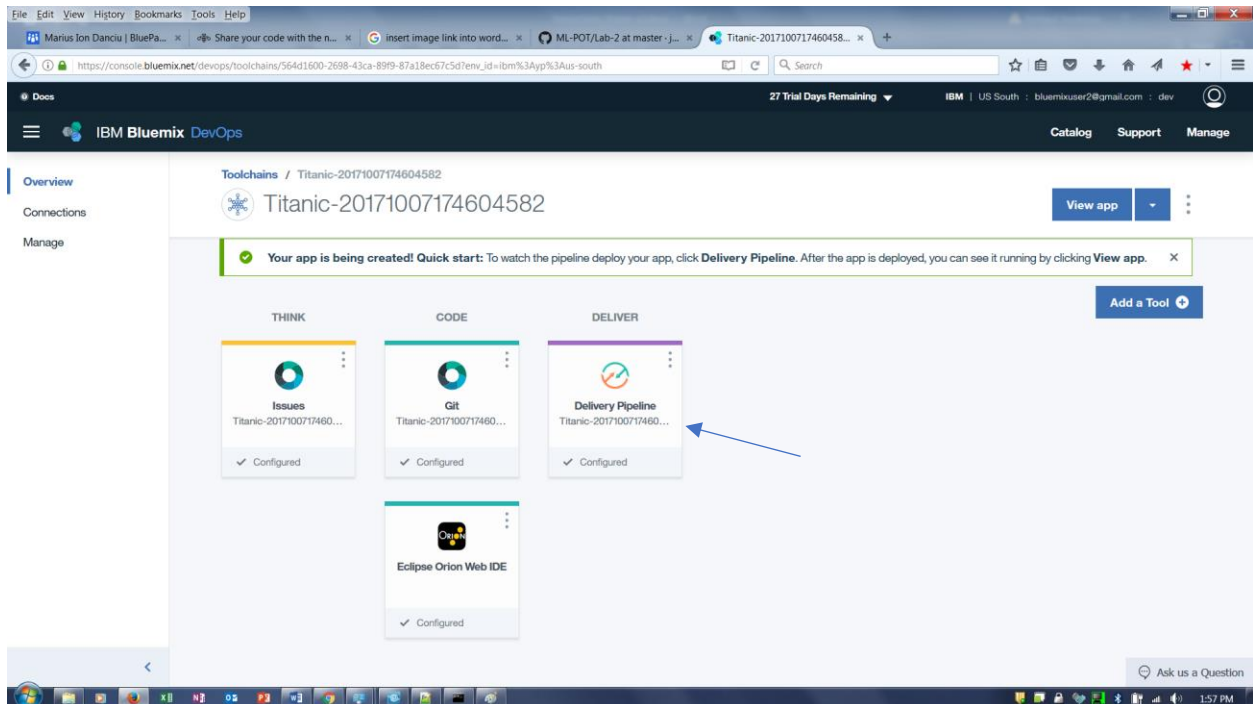
1. Click on the **Deploy to Bluemix** link below to deploy a sample Python Flask web application into your IBM Cloud account. Note you may get this message – “An IBM Cloud account is required. To get started, click Log In or Sign Up at the top of this page”. If you get this message, click on **Log In**.

[Deploy to Bluemix](#)

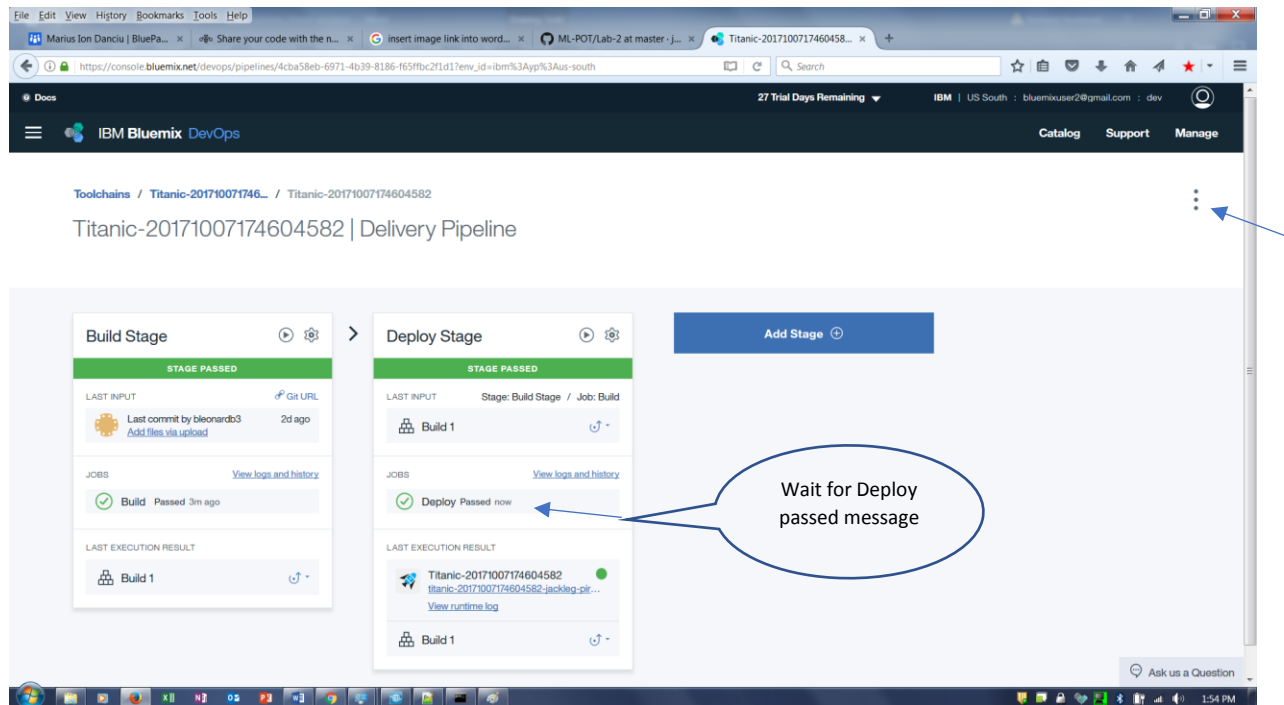
2. **SCROLL DOWN TO THE BOTTOM. MAKE SURE THAT YOU UPDATE THE “SPACE” (CLICKING ON DOWN ARROW) TO BE THE SAME WHERE THE WATSON MACHINE LEARNING SERVICE IS DEPLOYED. CHANGE THE DEFAULT (“DEV”) TO THE APPROPRIATE SPACE, IF NEEDED.** Click on the **Deploy** button.

The screenshot displays the IBM Cloud deployment interface. On the left, there is a 'TEMPLATE INFO' section with a 'GIT URL' field containing the link <https://github.com/open-to-...>. The main area is titled 'Toolchain Name:' and shows 'Titanic-20171130220547025'. Below this, 'Select Region:' is set to 'US South' and 'Choose an organization:' is set to 'jpatter@us.ibm.com'. The 'Tool Integrations' section shows three icons: 'Git Repos and Issue Tracking', 'Eclipse Orion Web IDE', and 'Delivery Pipeline'. A section titled 'The Delivery Pipeline automates continuous deployment.' contains an 'App name:' field with the value 'Titanic-20171130220547025'. At the bottom, there are three dropdown menus: 'Region' (US South (Production)), 'Organization' (jpatter@us.ibm.com), and 'Space' (dev). A blue 'Deploy' button is located at the bottom right. Two blue arrows point to the 'Space' dropdown and the 'Deploy' button, indicating the required actions.

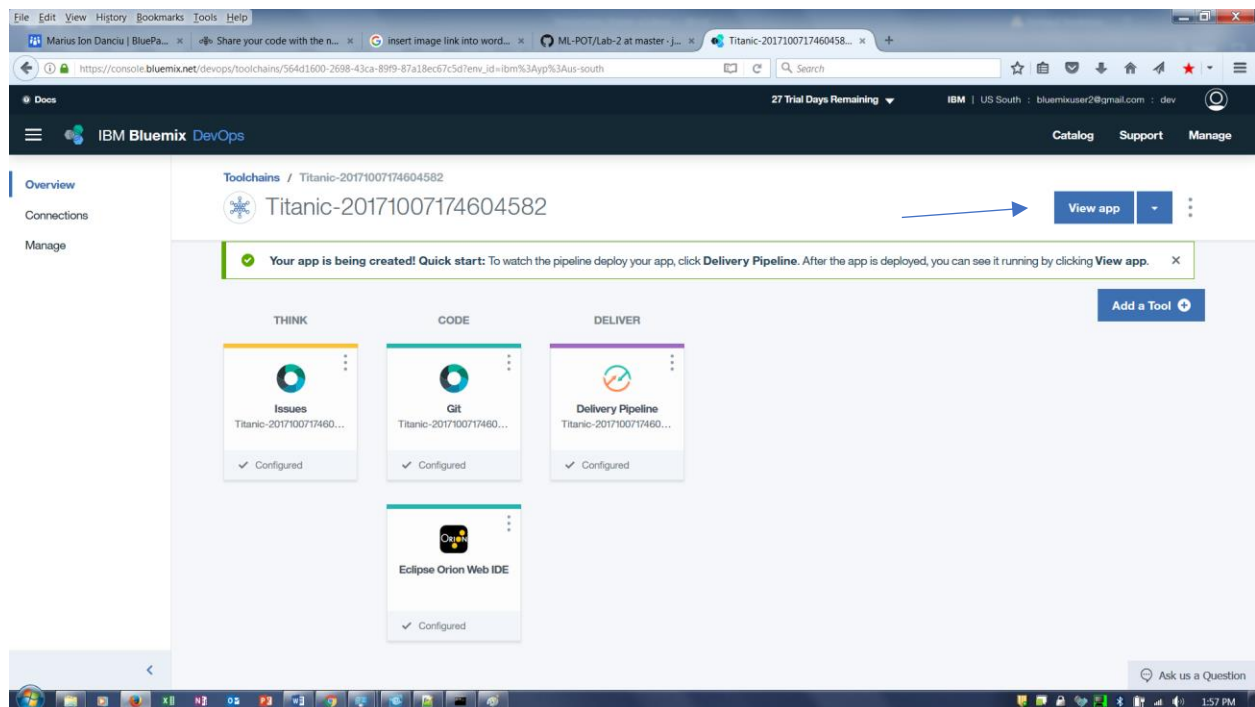
3. Your app is being created! To watch the pipeline deploy your app, click **Delivery Pipeline**.




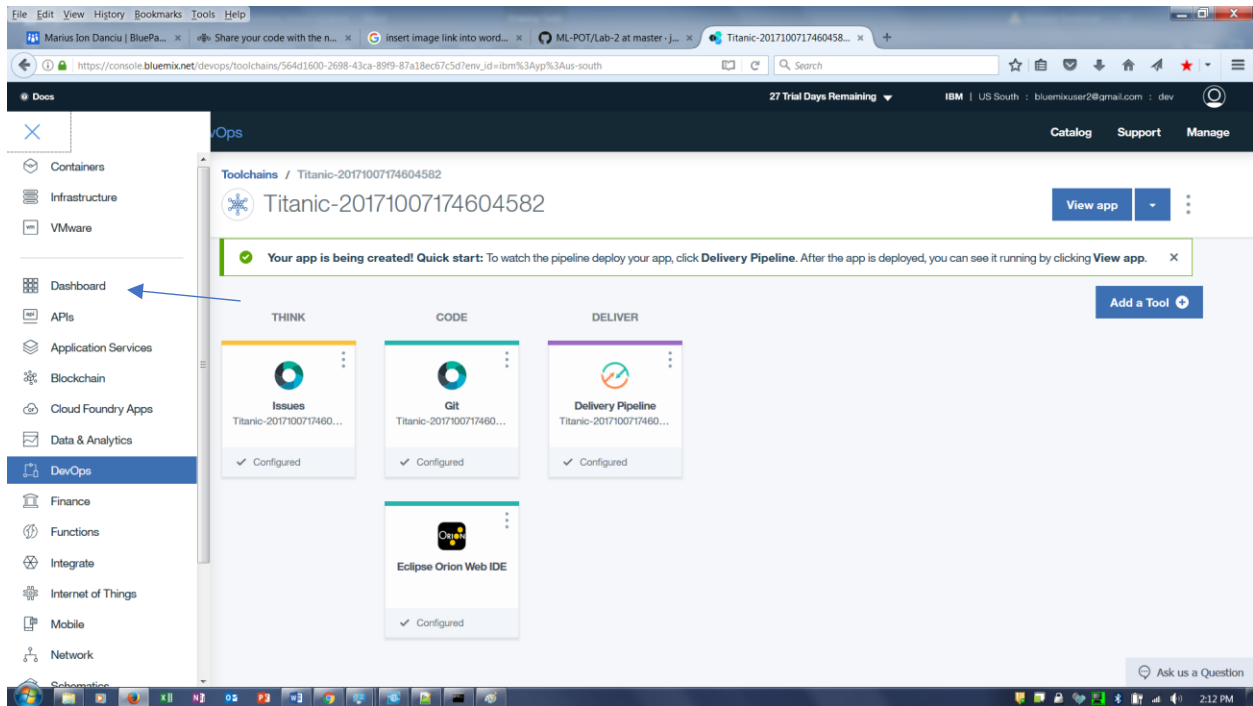
4. After the app is deployed successfully (should say Deploy Passed in the Deploy stage), return to the Delivery Pipeline by clicking on the vertical ellipse and click on **View Toolchain**.



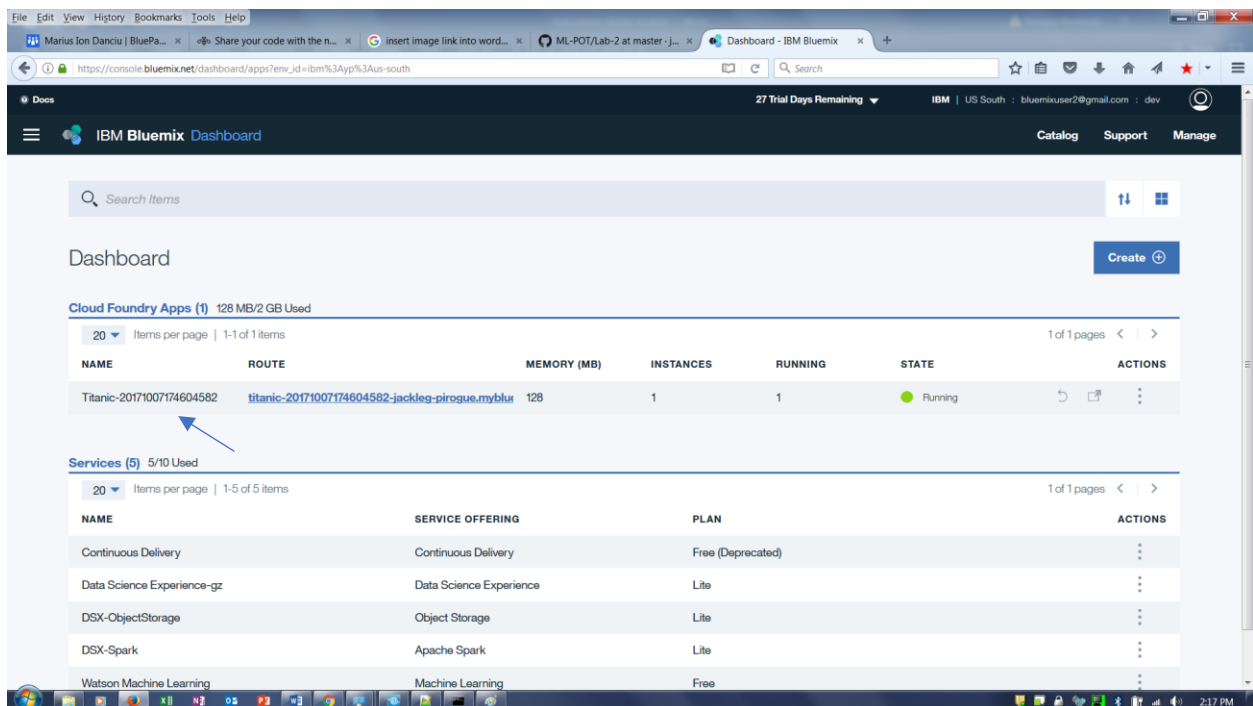
5. You can see the running app by clicking **View app**. The web form collecting the Titanic passenger data should appear. Note that the application is not functional until we connect it to the Watson Machine Learning service so if you Submit you will get an error!



6. Close the Titanic prediction app tab, and click on the  icon and Dashboard in the pulldown to navigate to the Dashboard where the running application should be listed.



7. We are now going to connect the application to the Watson Machine Learning service that was created earlier. Click on the application name.



8. Scroll down until you see the Connections panel. Click on **Create Connection**.

The screenshot shows the IBM Cloud dashboard for a Cloud Foundry application. The application is named 'Titanic-20171130220547025' and is in a 'Running' state. The dashboard displays the following information:

- Org:** jpatter@us.ibm.com
- Location:** US South
- Space:** dev
- Buildpack:** Python
- Instances:** 1 (All instances are running, Health is 100%)
- MB Memory per Instance:** 128
- Total MB Allocation:** 128 (7.875 GB still available)
- Connections:** No services are connected to this app. You can bind a service. A button labeled 'Create connection' is visible.
- Runtime cost:** \$0.00 (Current charges for billing period) and \$0.00 (Estimated total for billing period Nov 1, 2017 - Nov 30, 2017).

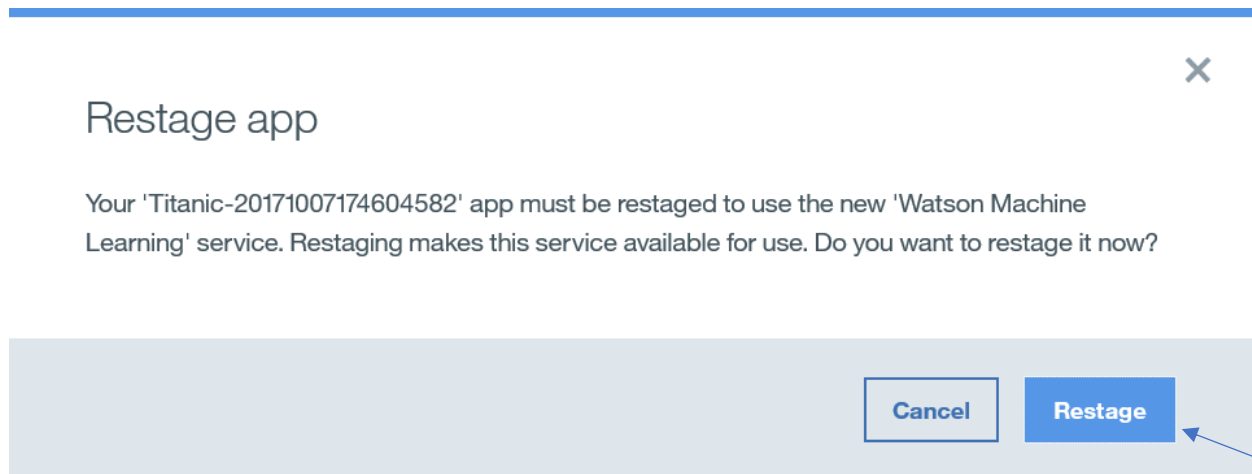
- You should see at least 3 services listed, a Cloud Object Storage service, a Spark service, and a Watson Machine Learning service. Click on the **Watson Machine Learning** service for your application, and then click on **Connect**.


The screenshot shows the 'Connect Existing Compatible Service' dialog in the IBM Cloud dashboard. It displays a table of available services that can be connected to the application. The table has the following columns: SERVICES, RESOURCE GROUP, PLAN, and SERVICE OFFERING.

SERVICES	RESOURCE GROUP	PLAN	SERVICE OFFERING
Apache Spark-Weather	--	Lite	Apache Spark
Apache Spark-Weather_objectstore	--	Lite	Object Storage
cloud-object-storage-joel	default	Lite	Cloud Object Storage
Decision Optimization-hz	--	Beta - 10 core / 60 GB (ODSTRAL)	Decision Optimization
DSE-Spark	--	Lite	Apache Spark
DSE-Spark20	--	Lite	Apache Spark
JoelDashDB	--	Entry	Db2 Warehouse on Cloud
ML	--	Lite	Machine Learning
ML-113017	--	Lite	Machine Learning
ML-POT	--	Lite	Apache Spark


A blue arrow points to the 'ML' service row in the table.

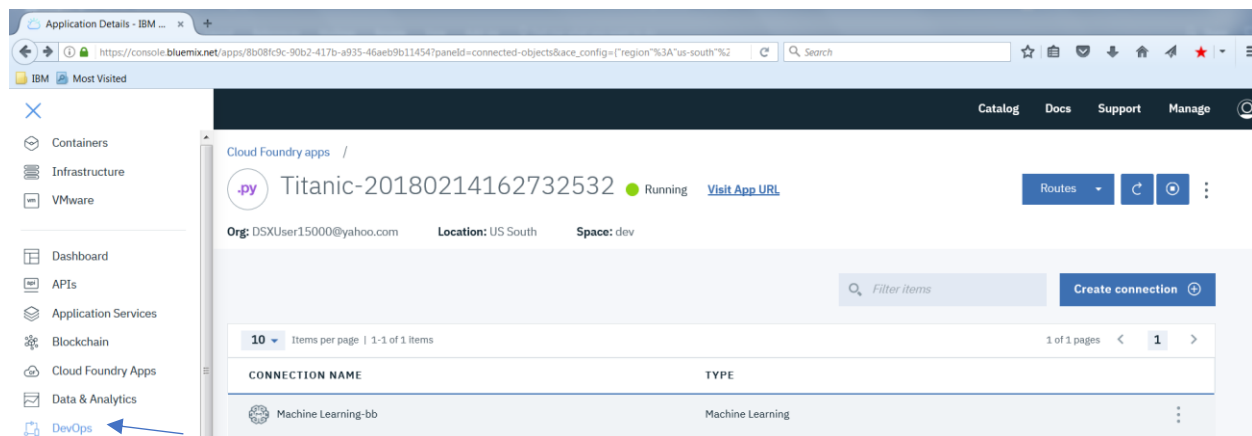
- You will get a pop up that asks to Restage the application. Click on **Restage**.



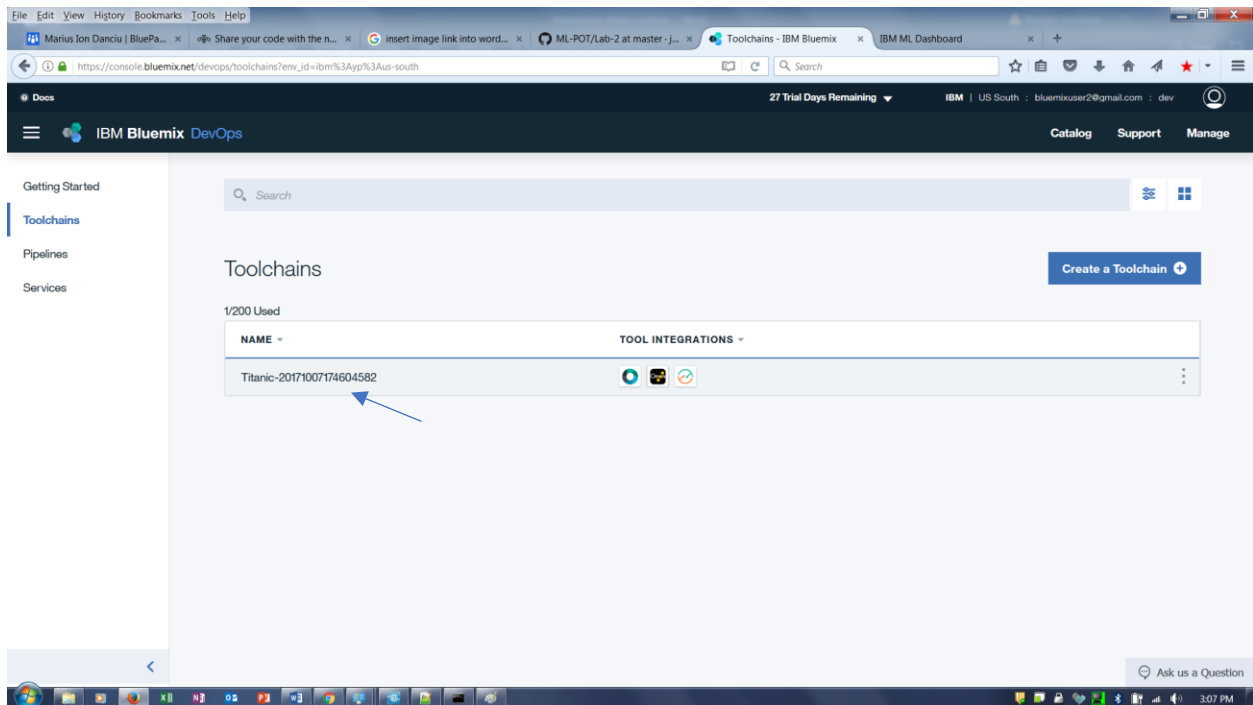
11. Wait for the application status to indicate  Running

12. We now have tied the web application to the Watson Machine Learning service. Note that the Watson Machine Learning service could have more than one deployed model available to select and then embed in the web application. In our case, we have only one deployed model. We now need to copy the scoring endpoint of that deployed model (previously copy and pasted into Notepad) and paste it in the web application code.

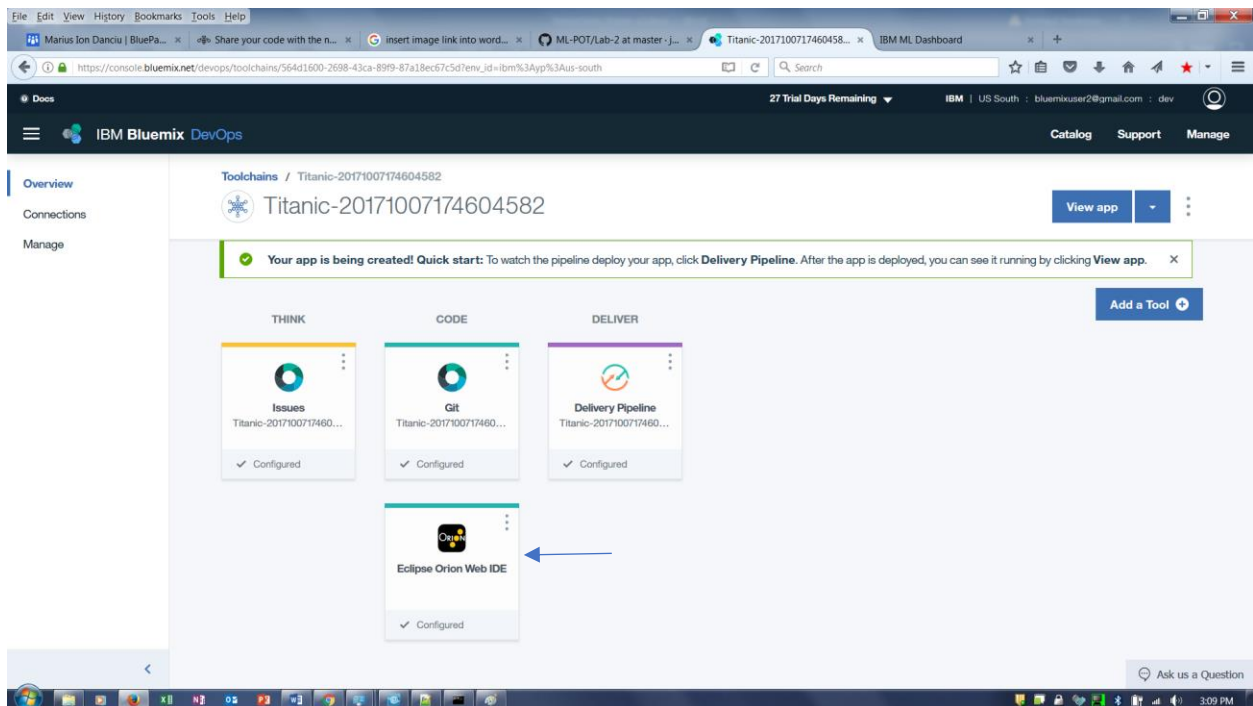
Click on the  icon, and click on DevOps in the pulldown to navigate to the Toolchain.



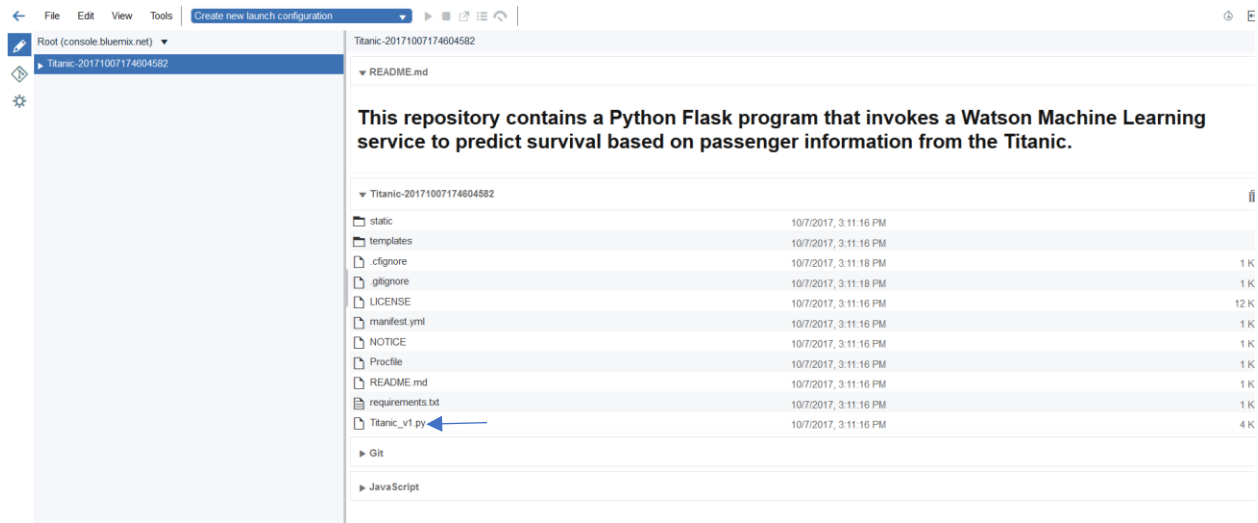
13. We are now going to paste the scoring endpoint into the application code. Click on the Toolchain (Titanic-2017xxxxx below).




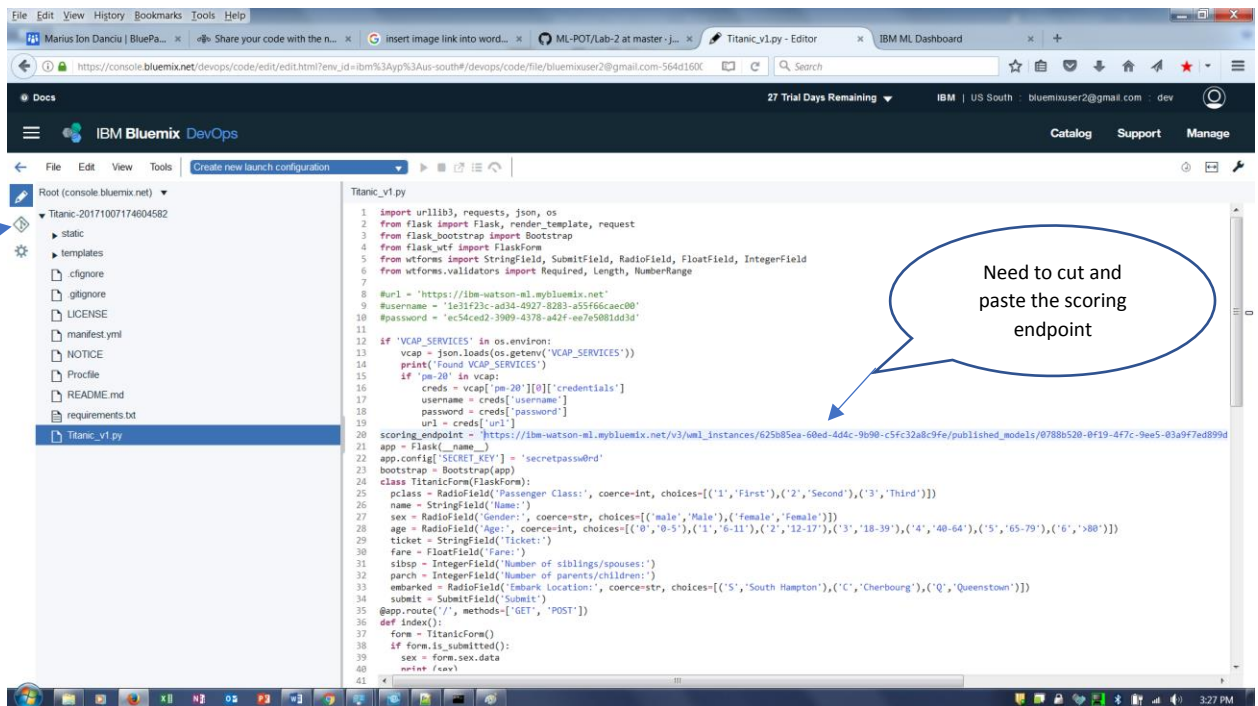
14. Click on the Eclipse Orion Web IDE.



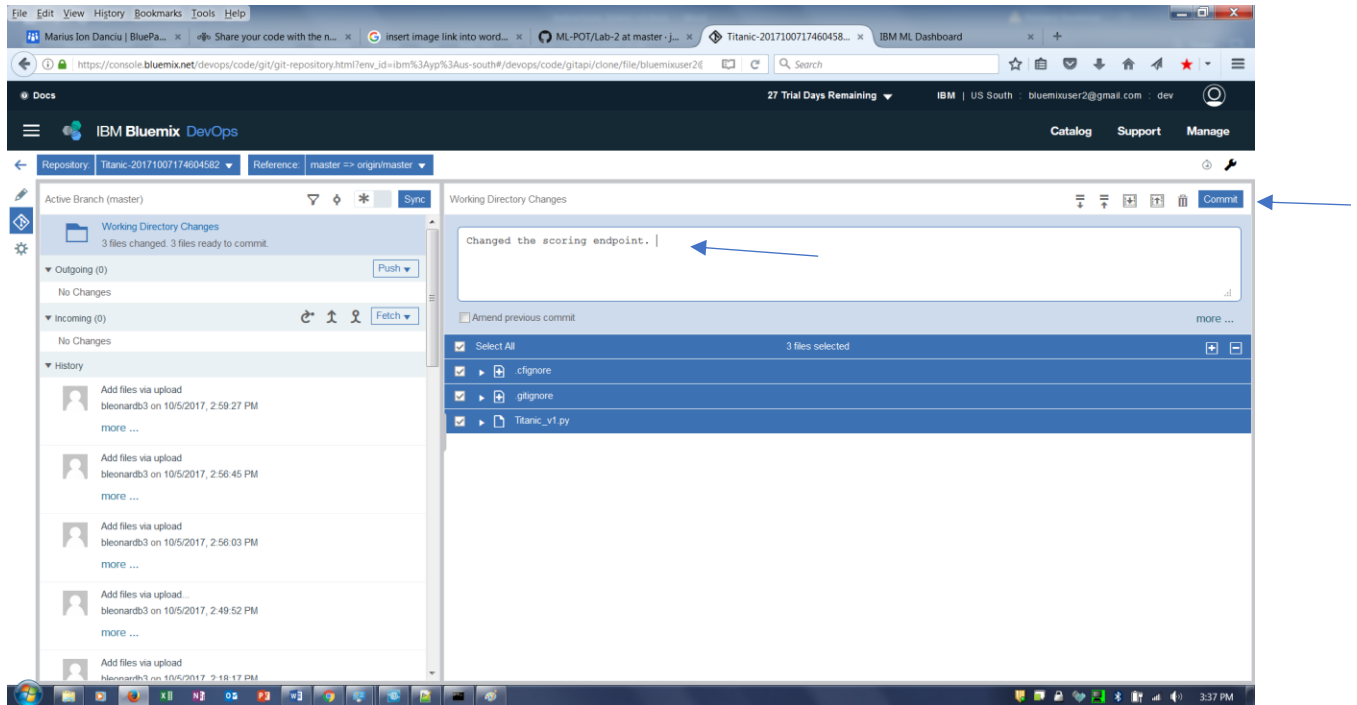
15. Click on the Titanic_v1.py file. This is a python source file.



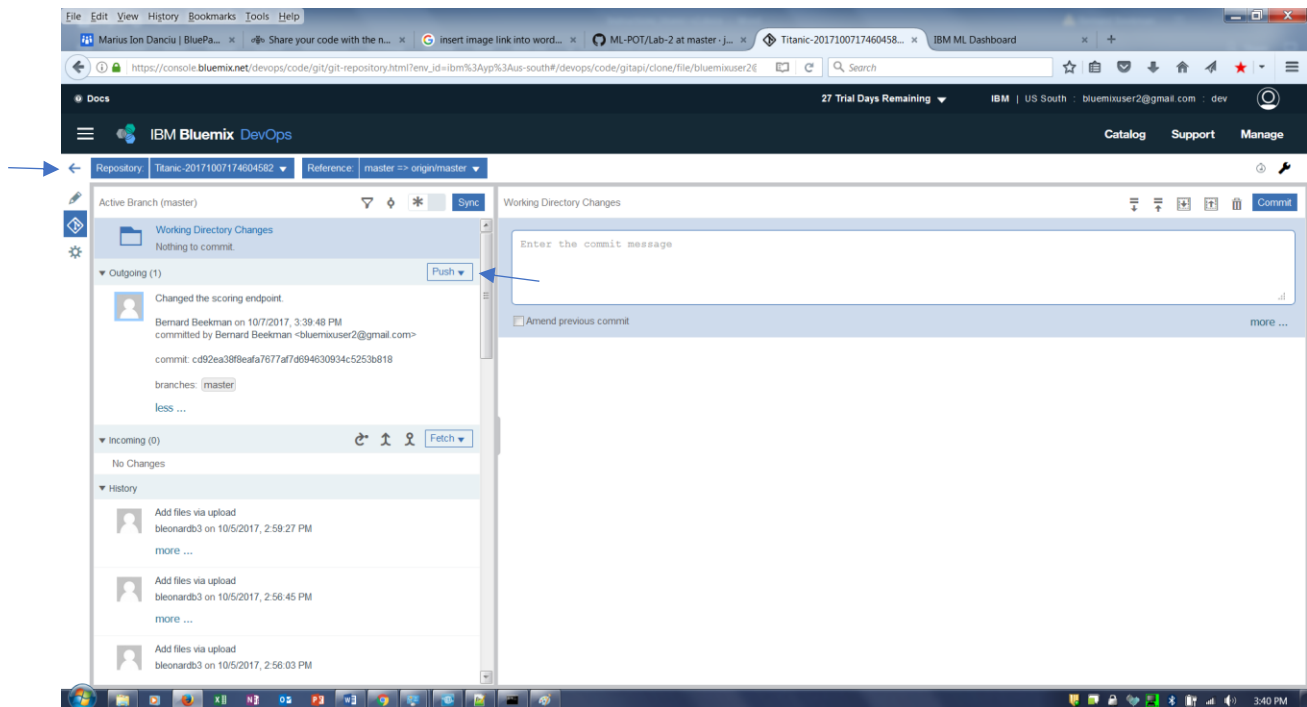
16. Go back to the Notepad file, and copy the scoring endpoint to the clipboard. Look around line 20 in the Titanic_v1.py file for the “scoring endpoint =”. Select the scoring endpoint value in line 20 (starting with https:// may want to use Shift-End to get to the end of the line, and then back up one space to not select the endpoint quote – if you do just make sure to put it back in). Enter Ctrl-V to paste the new scoring endpoint from your Notepad file. Enter Ctrl-S or File > Save to save the file. Then click on the  icon on the top left.



17. The next step is to commit the change to the git repository. Enter “Changed the Scoring Endpoint” in the Enter Commit Message field, and then click on **Commit**.



18. Then click on **Push** to push the changes to the central Git repo which will start the build and deploy of the application. Click on the left arrow to return to the Toolchain.



19. Click on the **Delivery Pipeline** to view status of the deployment as before. Once the Deployment status shows **Deploy passed now** (note it shouldn't take long so click

Reload in case the UI didn't update), click on the vertical ellipse and then click on the **View Toolchain** option to return to the Toolchain screen. Click on the **View Apps** button. (see Steps 3,4,5 above as a reminder if necessary). The web form should appear. Enter data in all the fields and click on the **Submit** button. (the submit button is located at the bottom of the web form – you may need to scroll).

To determine the survival prediction, please enter the following:

Passenger Class:

- ☒ First
☐ Second
☐ Third

Name: Bernie Beekman

Gender:

- ☒ Male
☐ Female

Number of siblings/spouses: 1

Number of parents/children: 1

Ticket: 1234

Fare: 23

Embark Location:

- ☒ South Hampton
☐ Cherbourg
☐ Queenstown

Age:

- ☒ 0-5
☐ 6-11
☐ 12-17

20. You should see something similar to the following depending on the values of the input fields that you entered. Click on the **Try Again!**, if you want to experiment with different inputs.

Titanic Prediction

prediction:survived
probability: 0.827966430684

[Try Again!](#)