Watson Machine Learning Overview

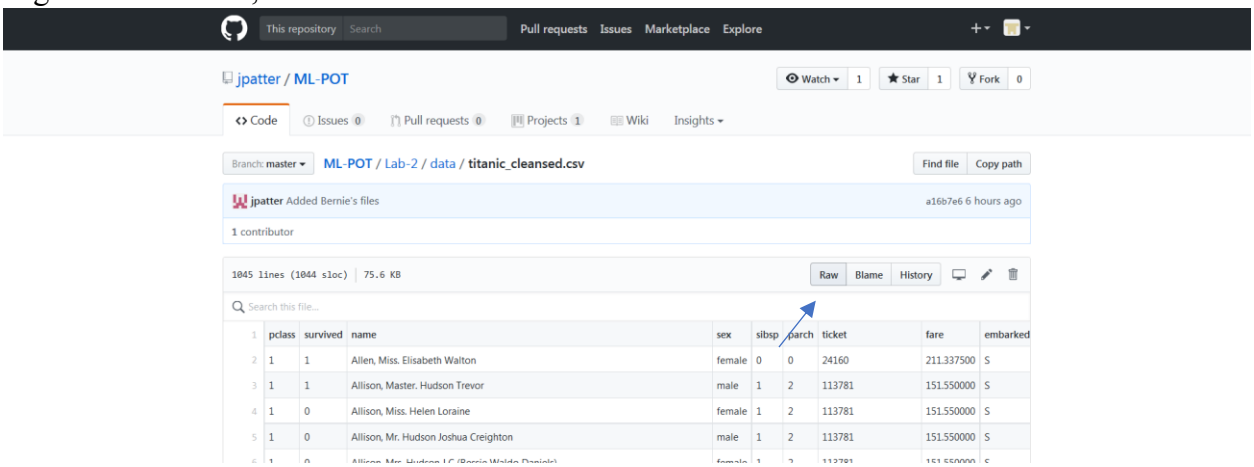
This lab will introduce the Watson Machine Learning capability using the Titanic dataset. The lab will consist of the following steps:

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.

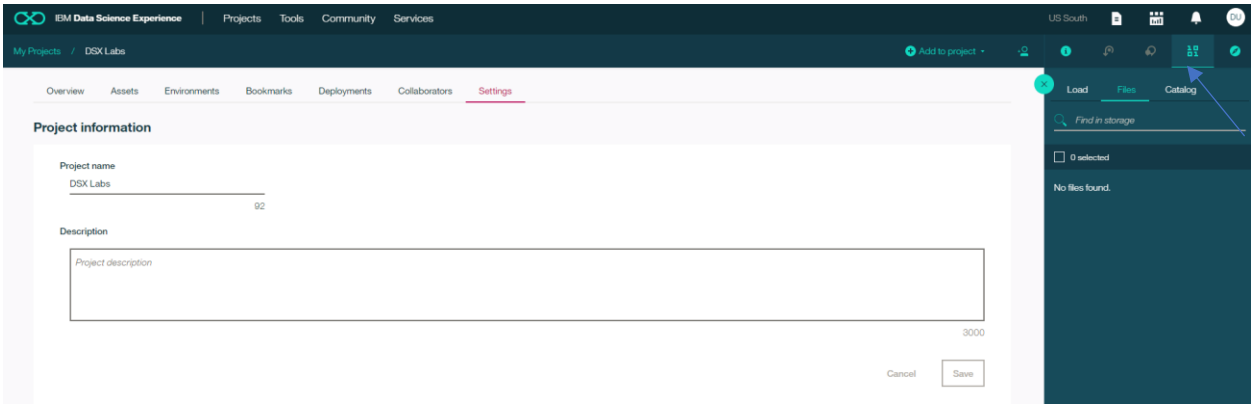
Step 1: Adding a Data Asset to the project

1. Download the Titanic data file from the following location by clicking on the link Cleansed Titanic Data Set and following the instructions below.

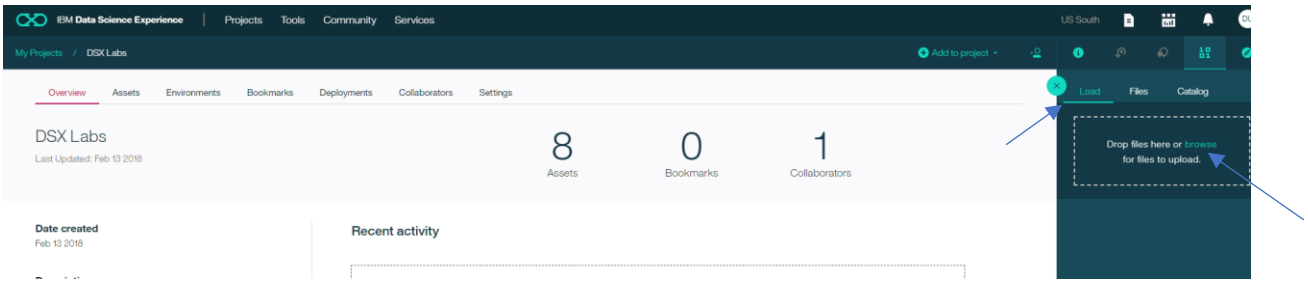
2. Right click on Raw, and click on Save link as ….



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



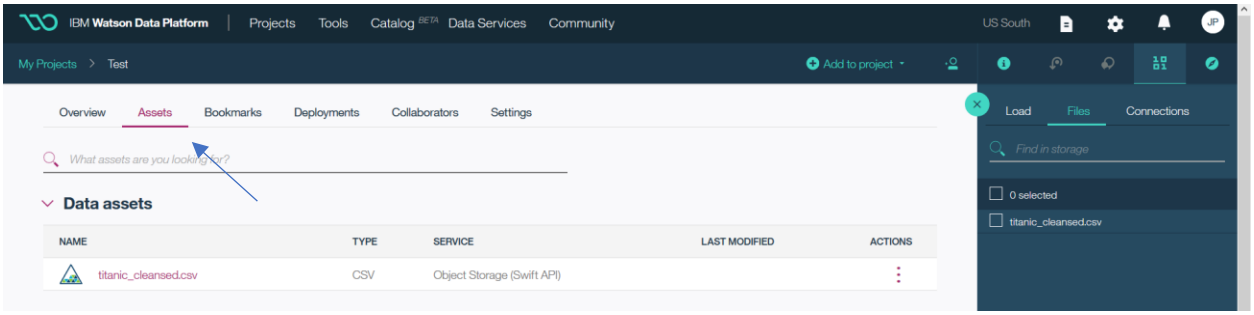
4. Click on **Load** and then **browse** and then go to the folder where the titanic_cleansed.csv is stored. Select titanic_cleansed.csv and then click Open.



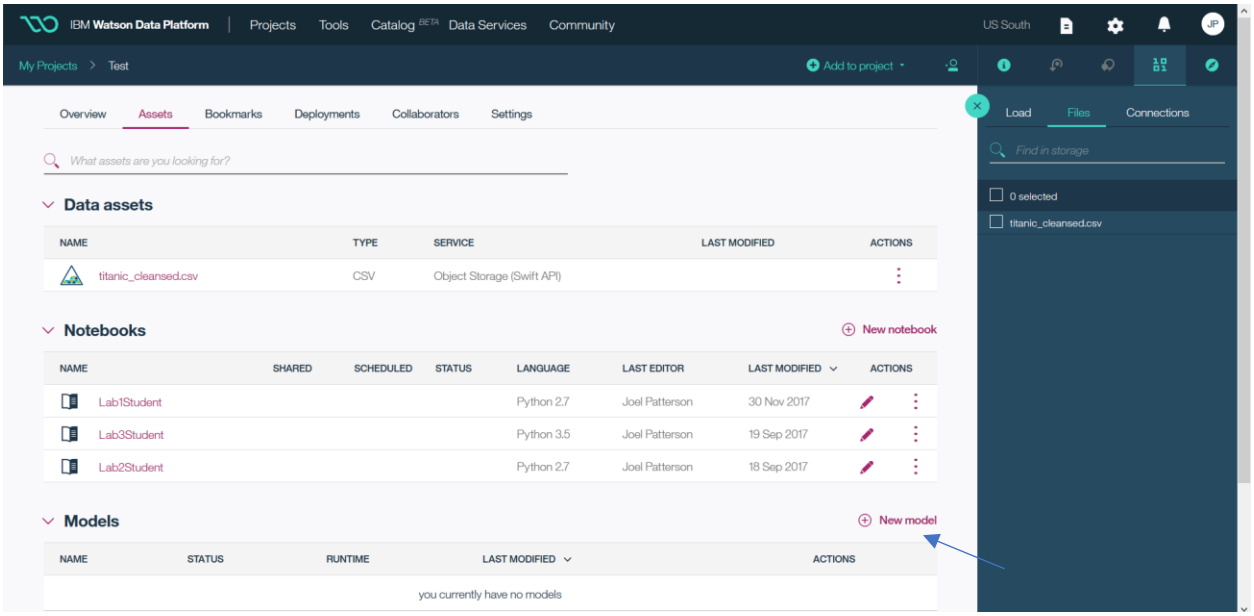
.

Step 2: Create a Model to predict survival

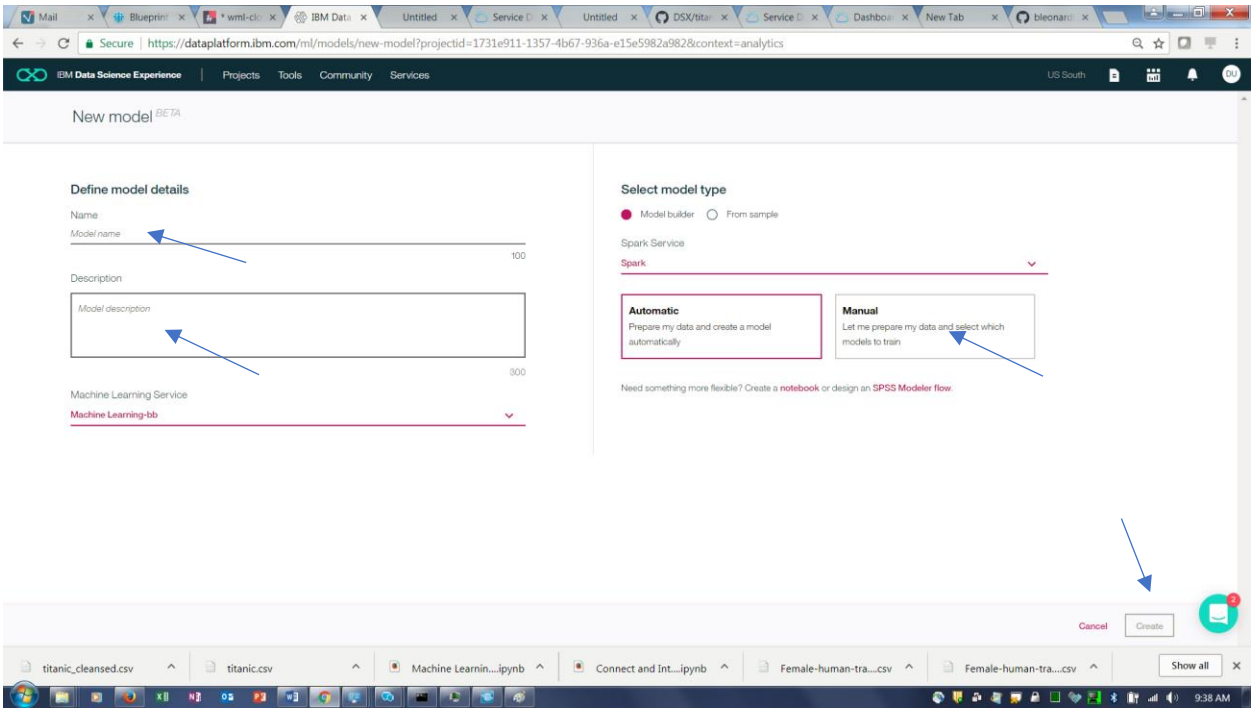
1. Click on the **Assets** Tab



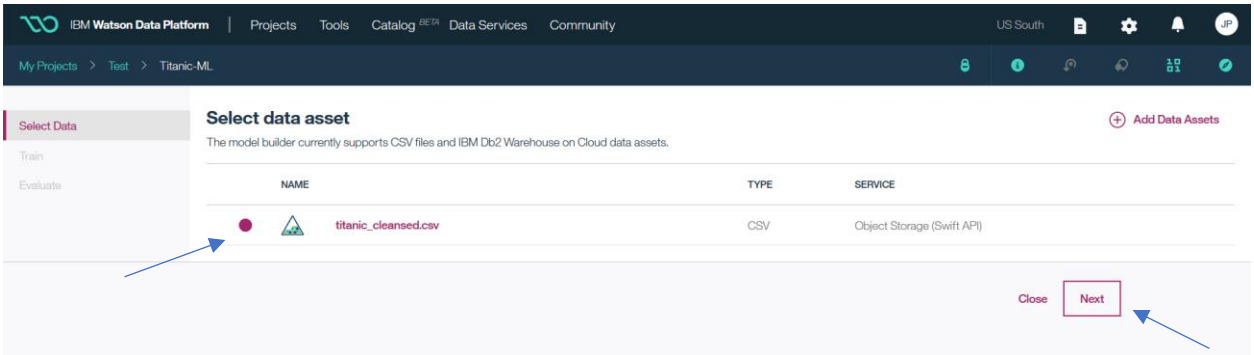
2. Click on **New model**.



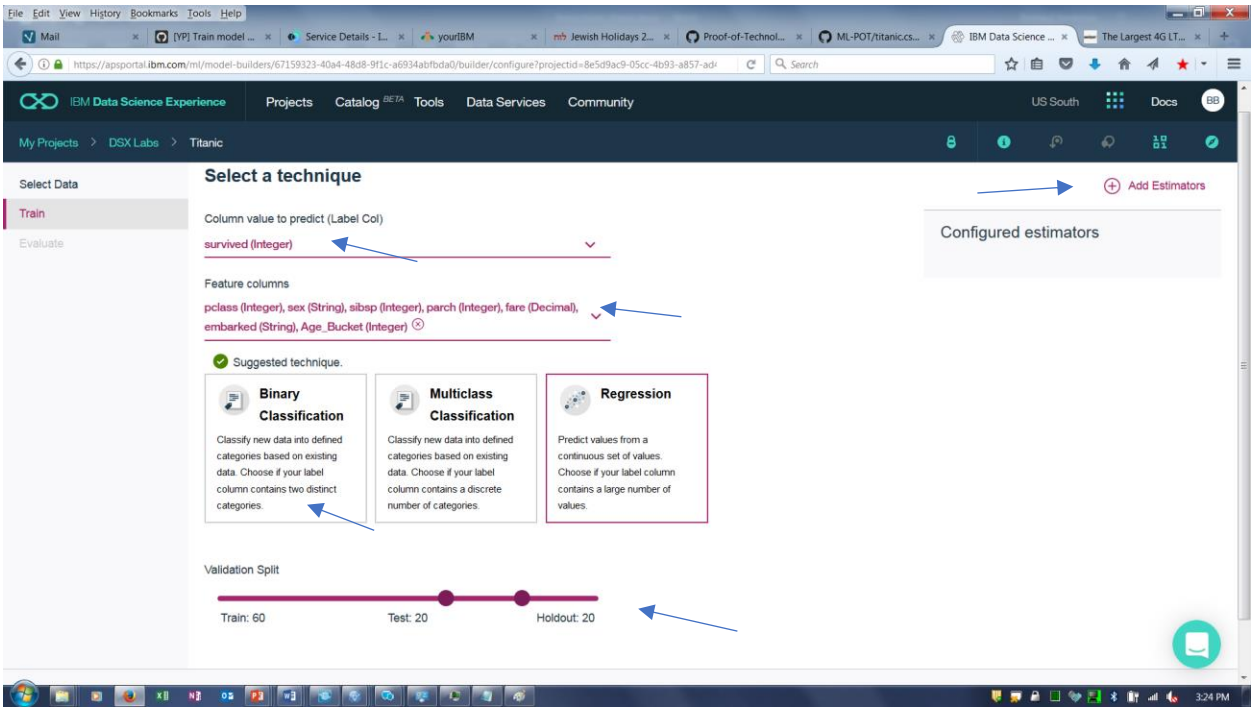
3. Enter a Model **Name** (eg Titanic), optionally a **Description**, Select **Manual**, and click on **Create**.



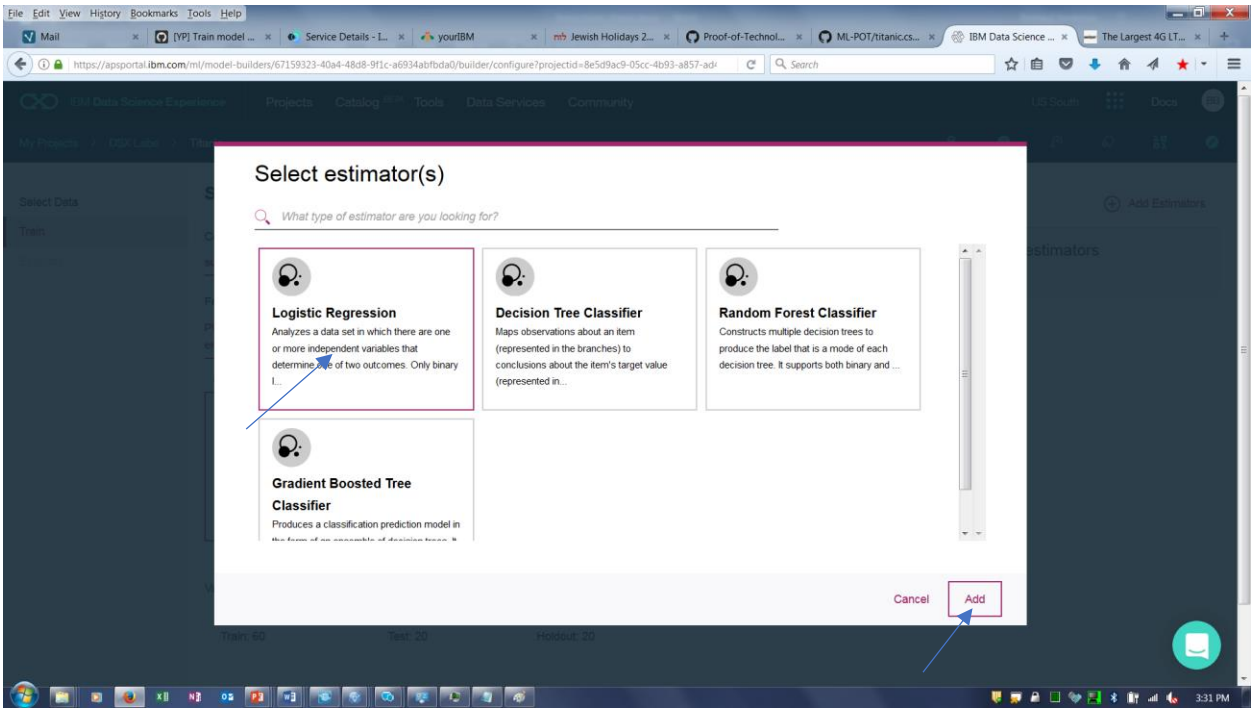
4. Click on the titanic_cleansed**.csv** and click on **Next**



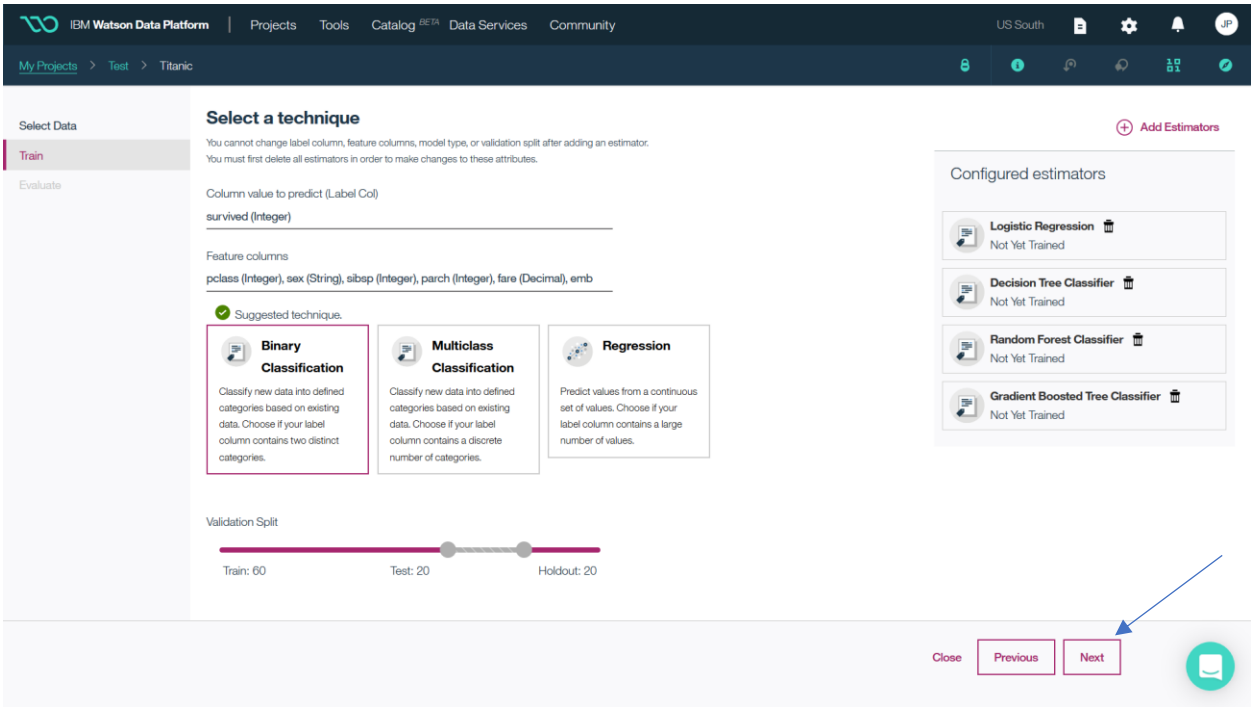
5. For **Column value to predict (Label Col)** select **survived**. For **Feature columns** select the following features (**pclass, sex, sibsp, parch, fare, embarked, Age_Bucket**) . Click on the **Binary Classification** Box (which is suggested by the service). Adjust the **Validation Split** as desired.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



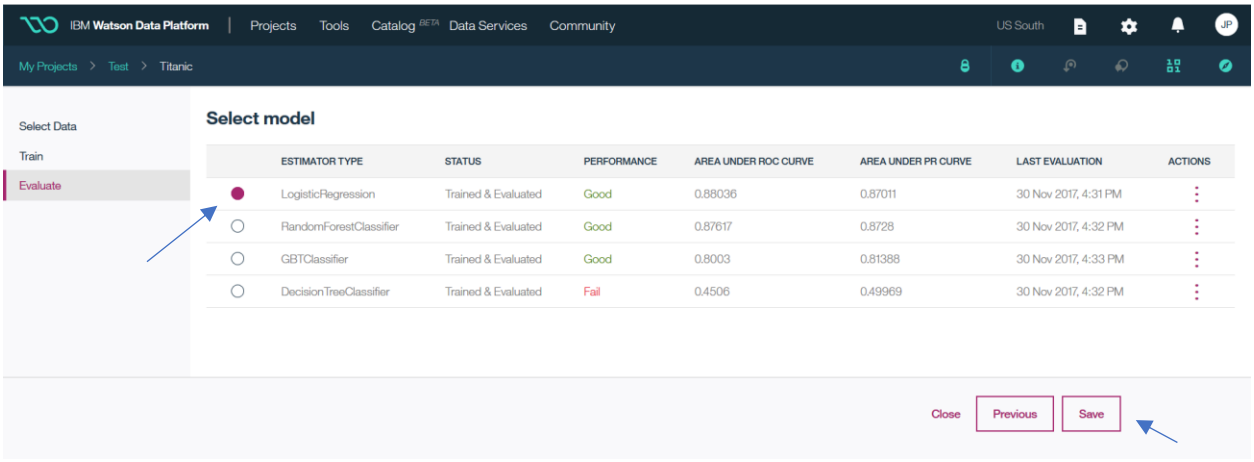
6. Select **Logistic Regression**. You can select more if you wish to see the results of multiple models. Select **Add**.



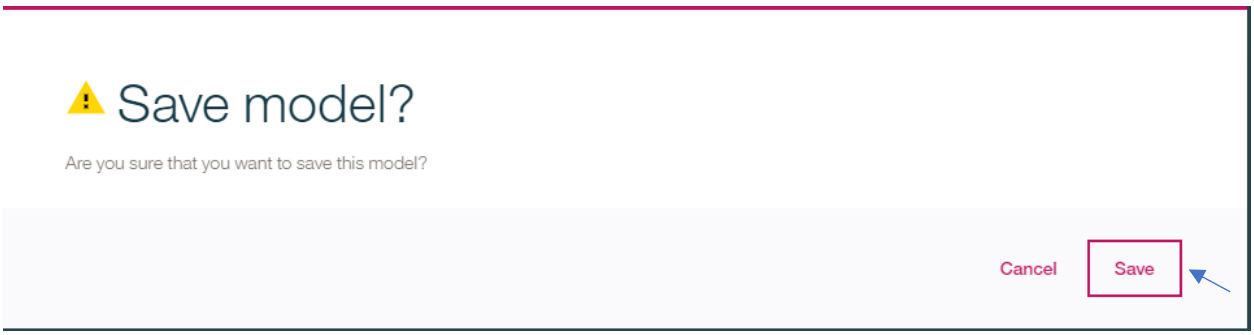
7. Select the **Next** button.



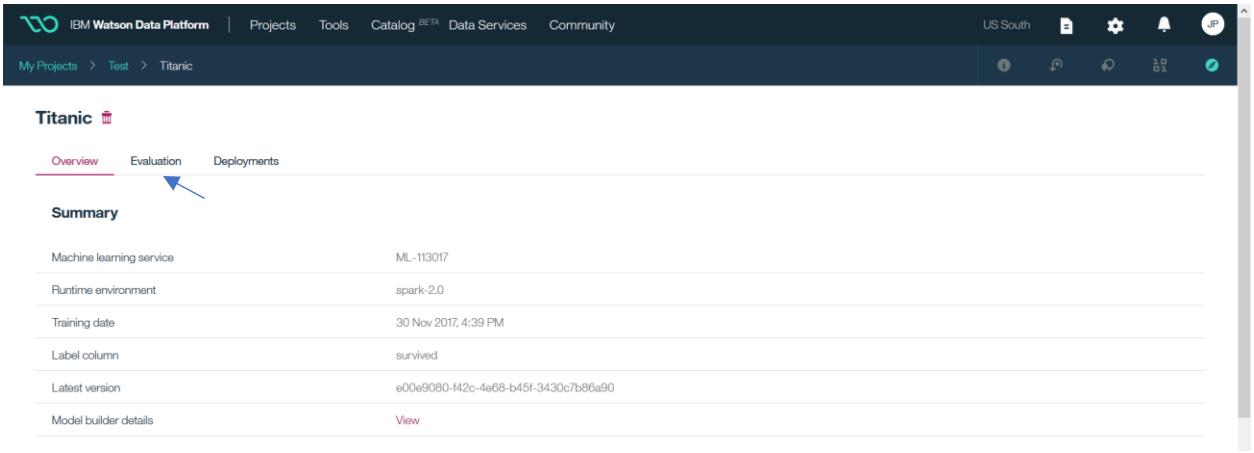
8. The system trains and evaluates each model. If more than one model was selected, the models would be listed in descending order of quality with the best result at the top. Note: if a model fails to run (rare, but happens), select Previous, delete that model and re-add it. Then run again. Click on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



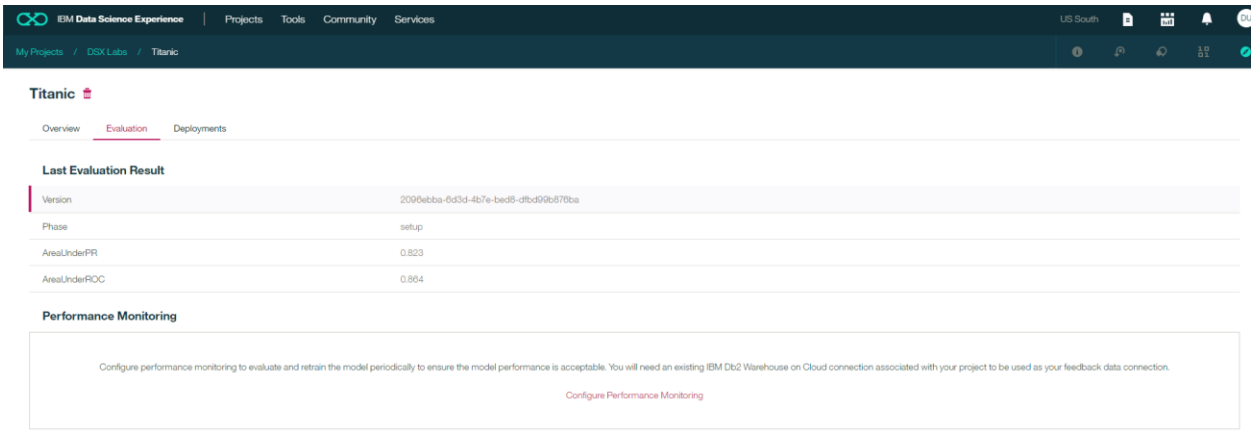
9. Click **Save** again on the next screen.



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



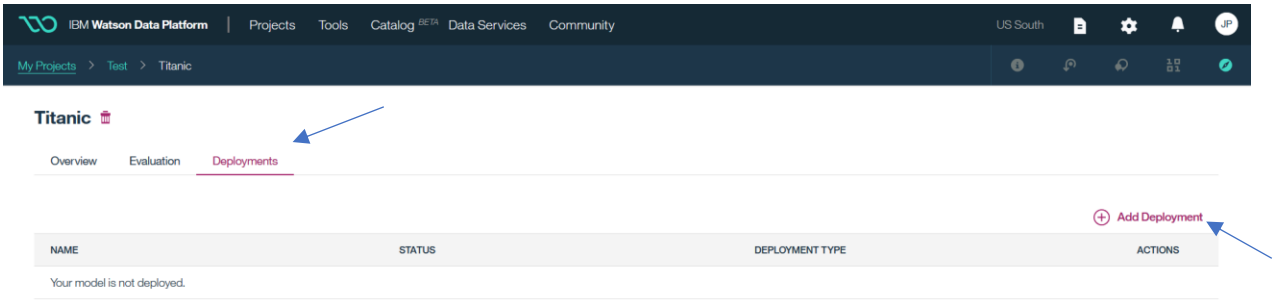
11. The system displays the recorded evaluation statistics for the run. You can also set up Continuous Learning (Performance Monitoring) on this screen. We will not do this now.



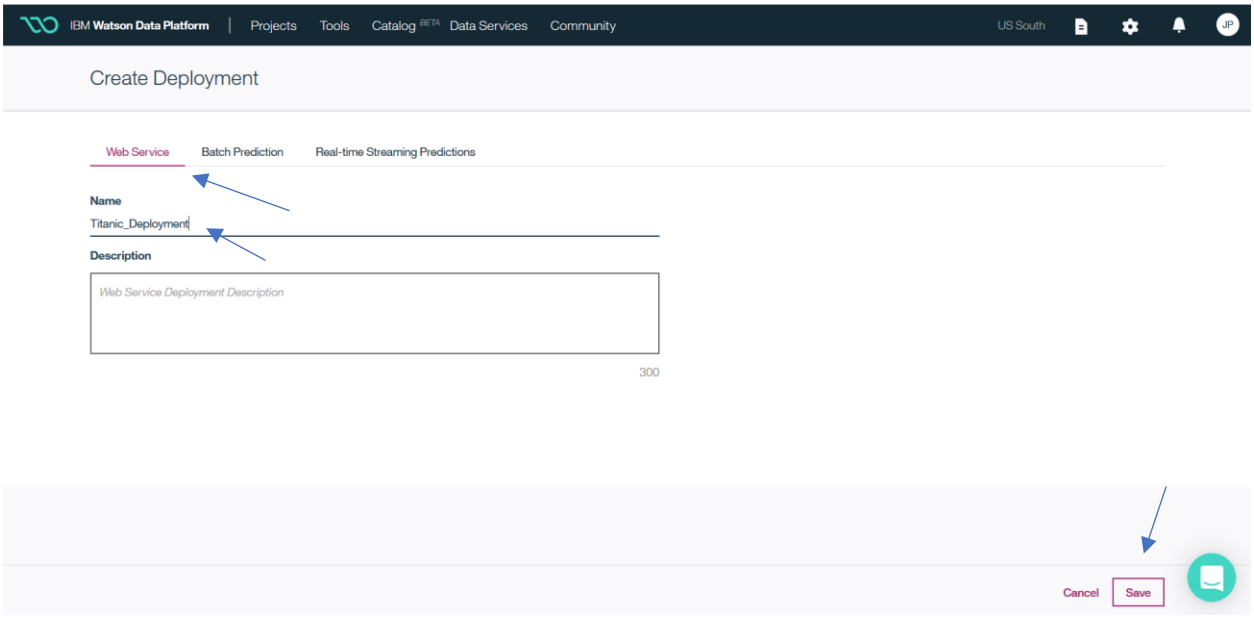
Step 3: Deploying a Model

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

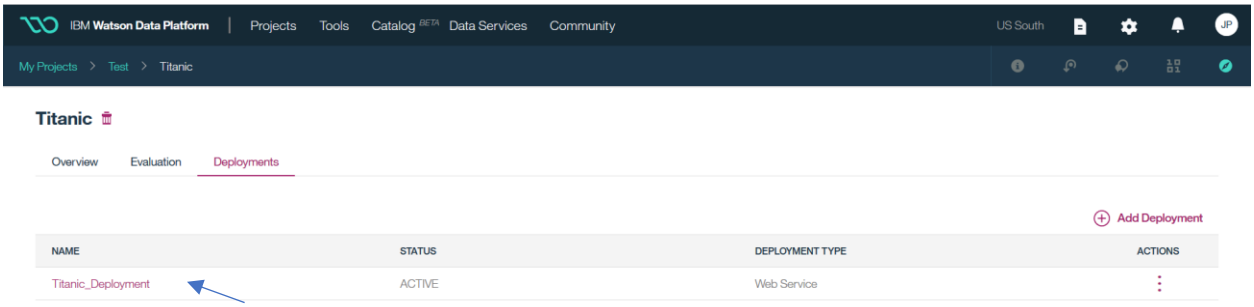
1. Select the **Deployments** Tab
2. Click on **Add Deployment**



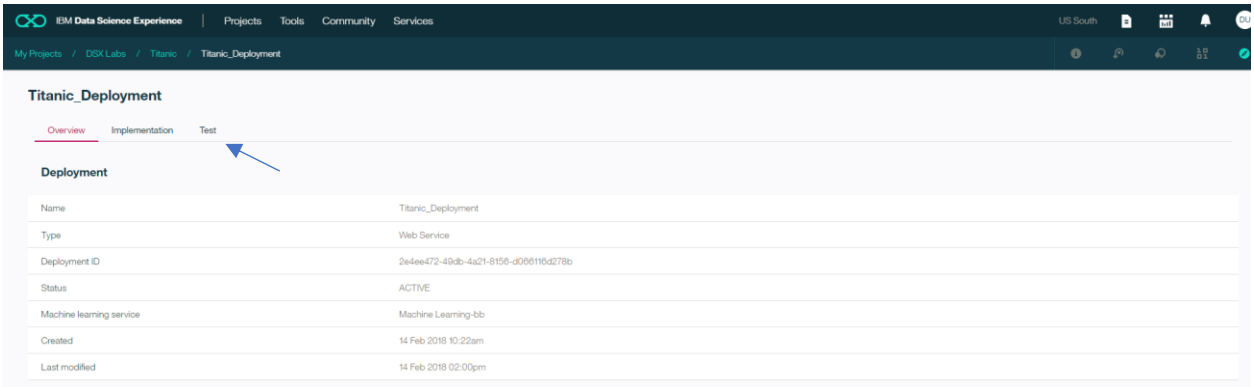
3. Three options for deployment are available. For this lab, we are going to embed the predictive model scoring in a web application. Therefore, select the **Web Service** tab, enter Titanic_Deployment for **Name**, and click on **Save**.



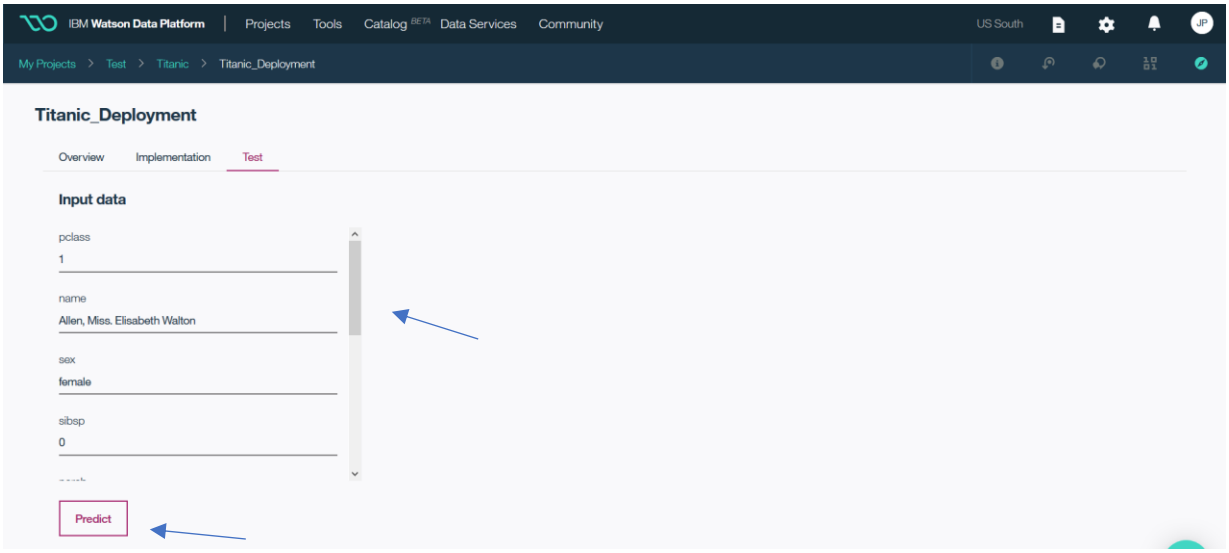
4. The system responds with an acknowledgement that the model was successfully deployed. Click on **Titanic_Deployment** to test the deployed API.



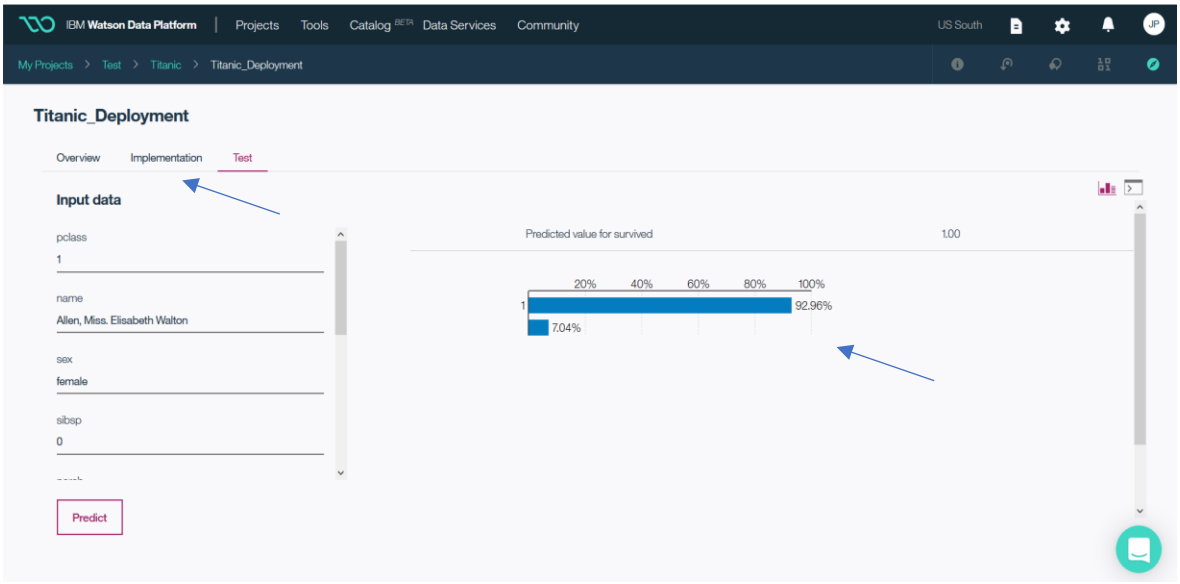
5. The system displays information about the deployed service. Click on **Test** to test out the API.



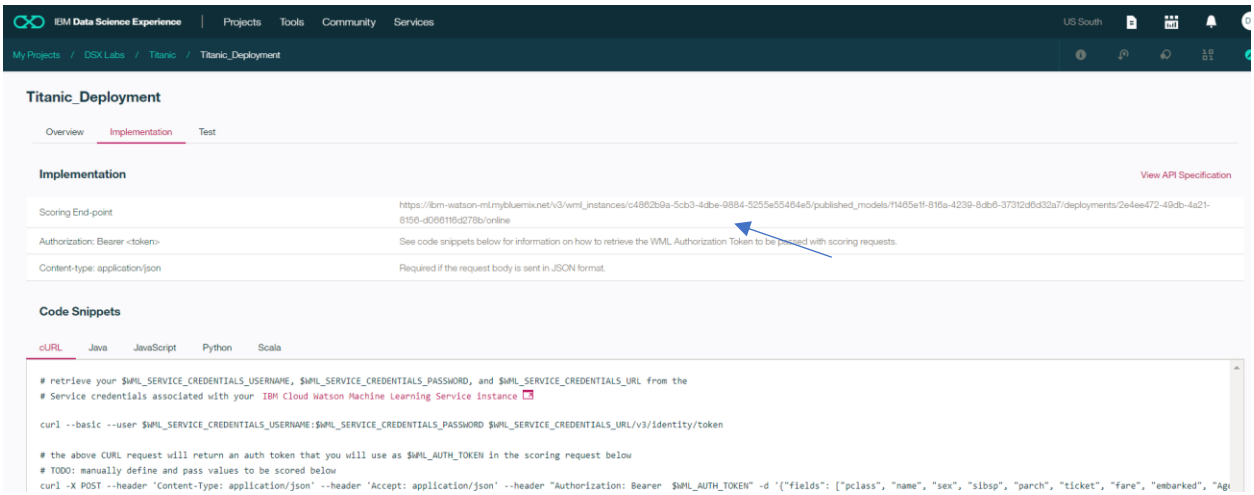
6. Enter values for the input fields or just use the defaults, and then click on **Predict.** Note that the values inputted for any of the fields not included in the model parameters (e.g. name) will not affect the prediction.



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.



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.

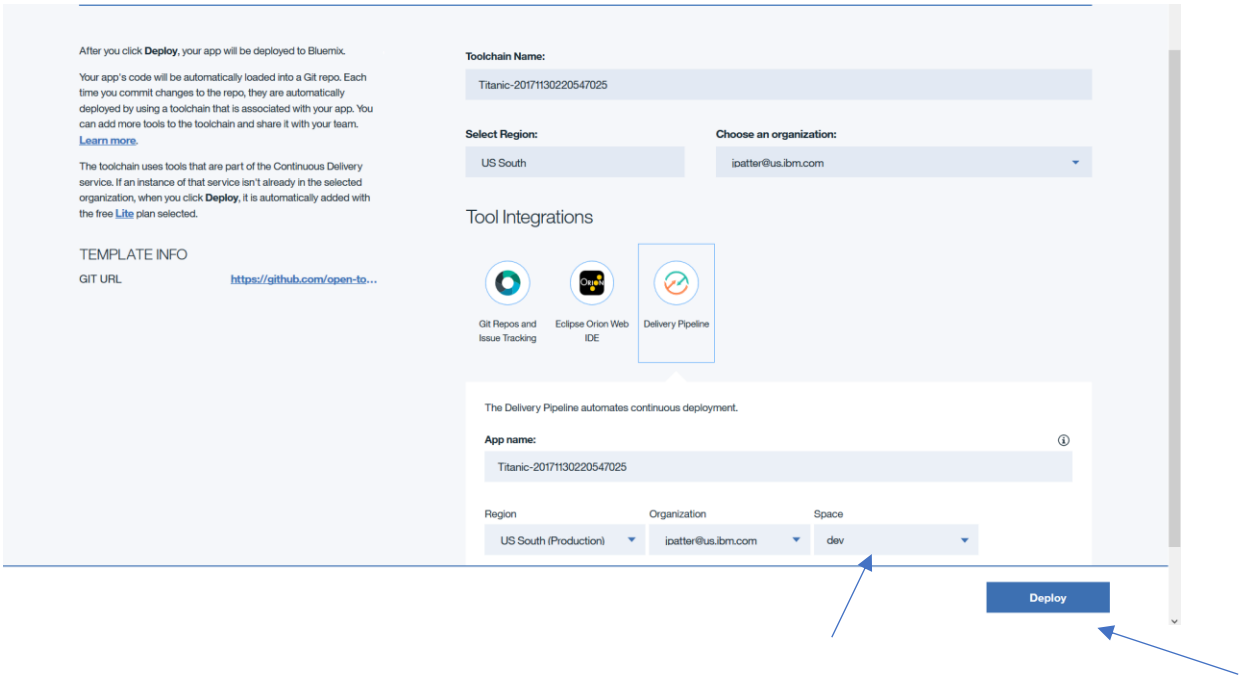
The toolchain uses tools that are part of the Continuous Delivery service. If an instance of that service isn't already in your organization, when you click **Deploy**, it is automatically added with the free Lite plan selected.

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

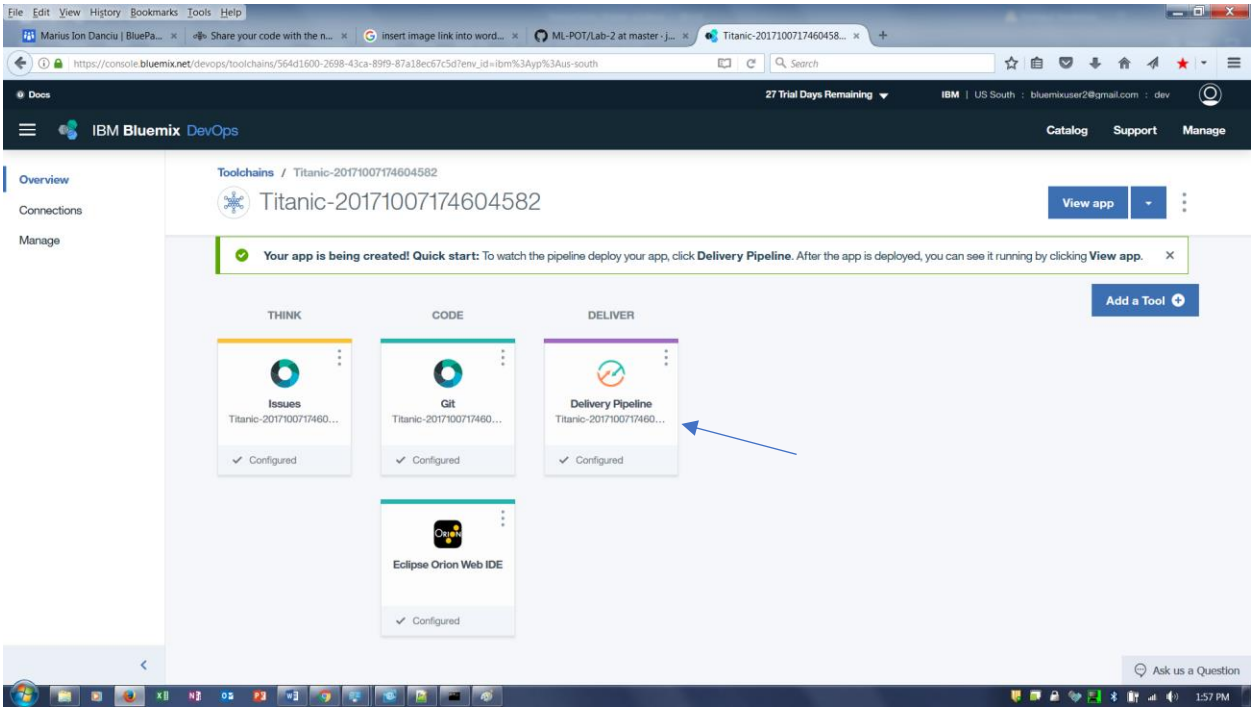
1. Click on the **Deploy to Bluemix** link below to deploy a sample Python Flash web application into your IBM Cloud account. Note you may get this message – "*An IBM Cloud account is required. To get started, click Log In or Sign Up at the top of this page*". If you get this message, click on **Log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

   Deploy to Bluemix

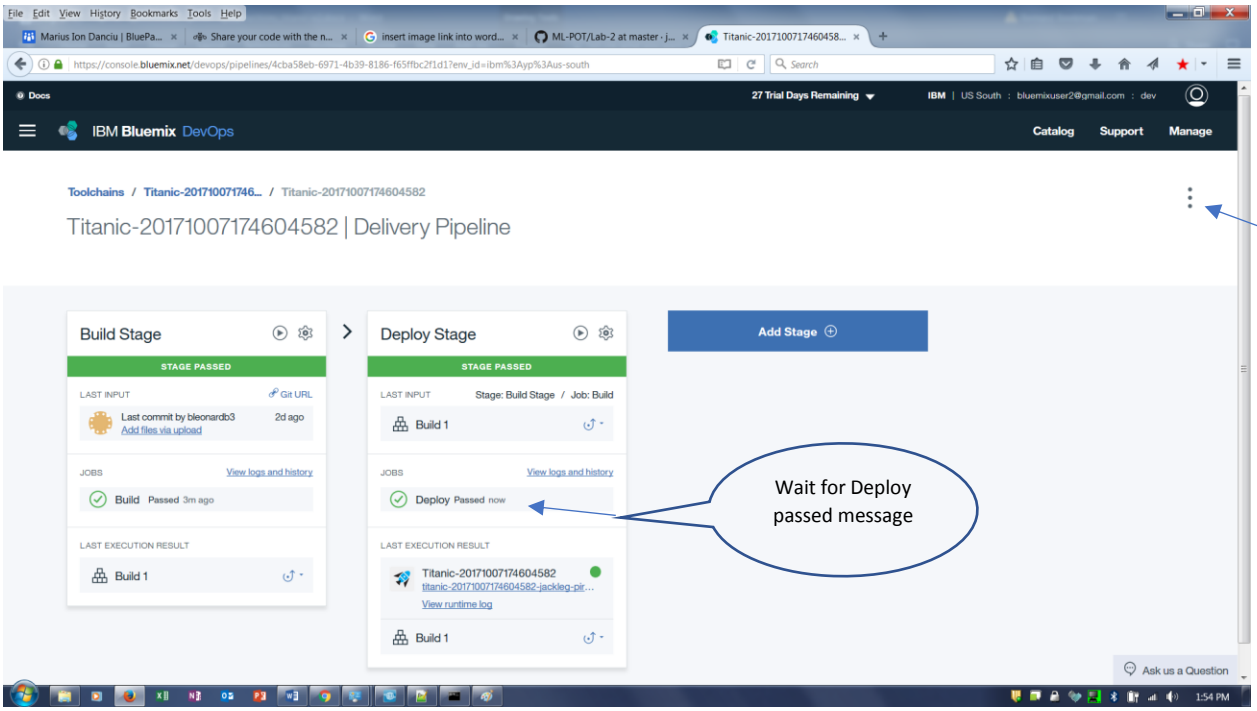
2. **SCROLL DOWN TO THE BOTTOM. MAKE SURE THAT YOU UPDATE THE "SPACE" (CLICKING ON DOWN ARROW) TO BE THE SAME WHERE THE WATSON MACHINE LEARNING SERVICE IS DEPLOYED. CHANGE THE DEFAULT ("DEV") TO THE APPROPRIATE SPACE, IF NEEDED.** Click on the **Deploy** button**.**



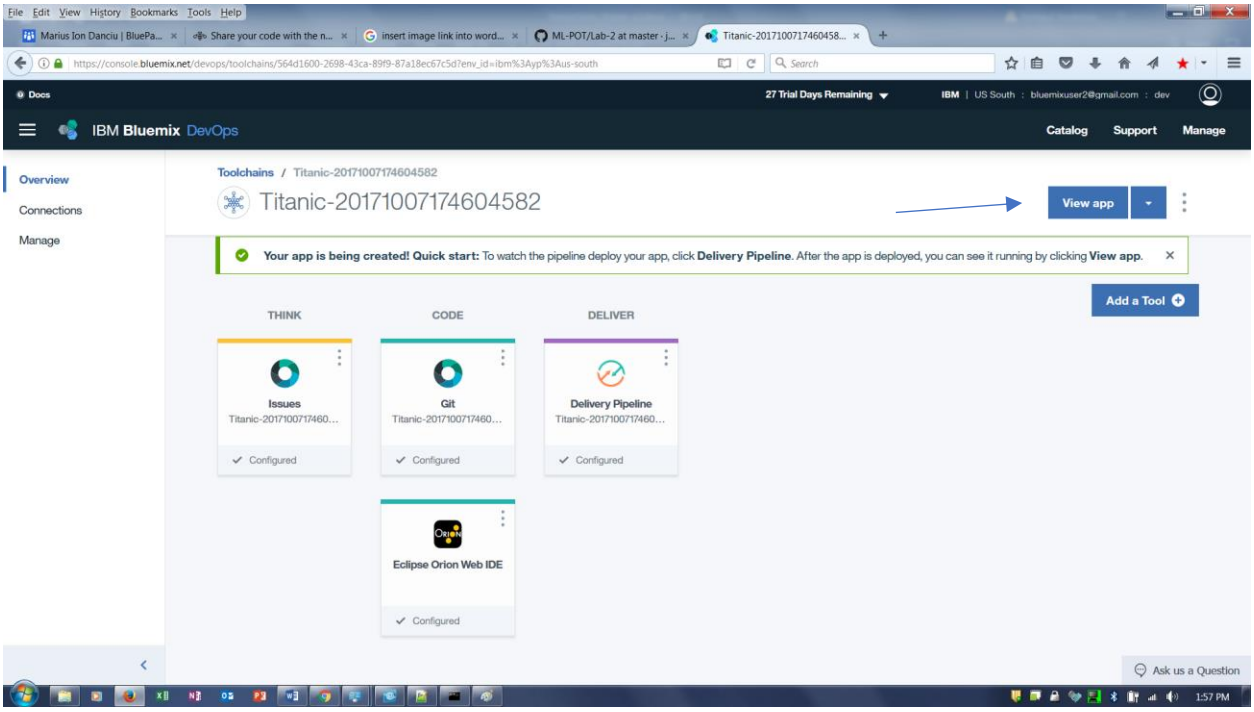
3. Your app is being created! To watch the pipeline deploy your app, click **Delivery Pipeline**.



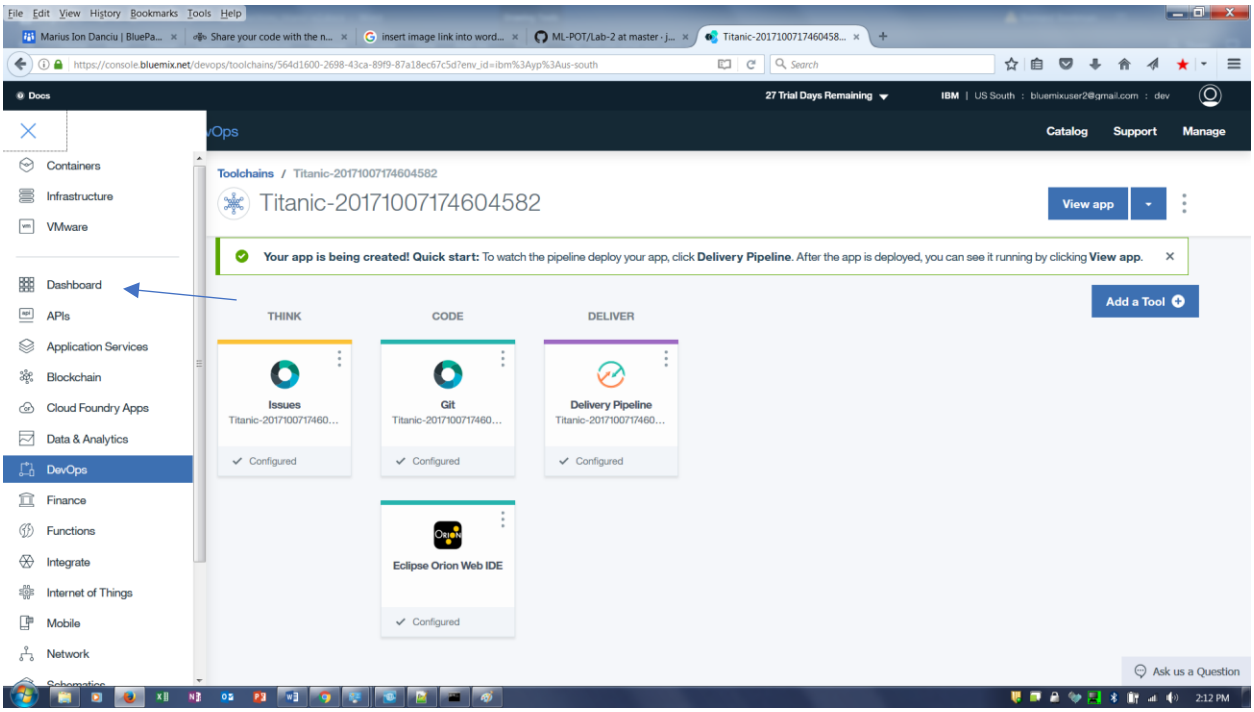
4. After the app is deployed successfully (should say Deploy Passed in the Deploy stage), return to the Delivery Pipeline by clicking on the vertical ellipse and click on **View Toolchain.**



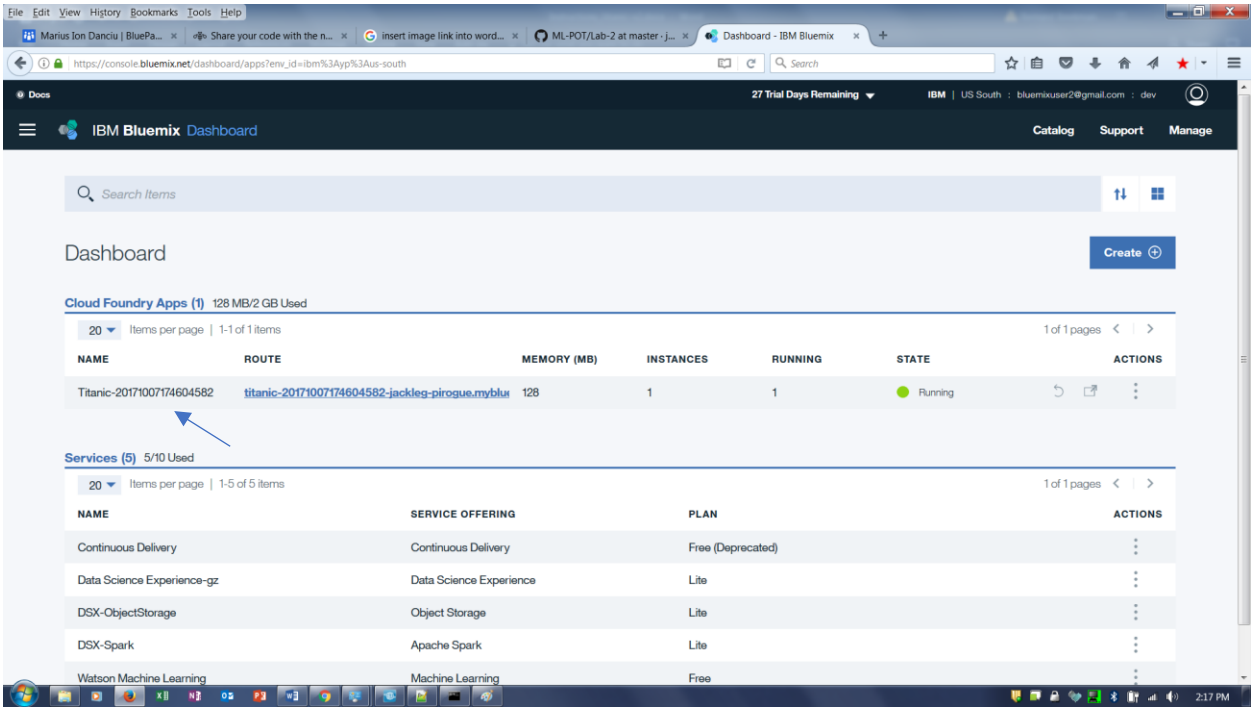
5. You can see the running app by clicking **View app**. The web form collecting the Titanic passenger data should appear. Note that the application is not functional until we connect it to the Watson Machine Learning service so if you Submit you will get an error!



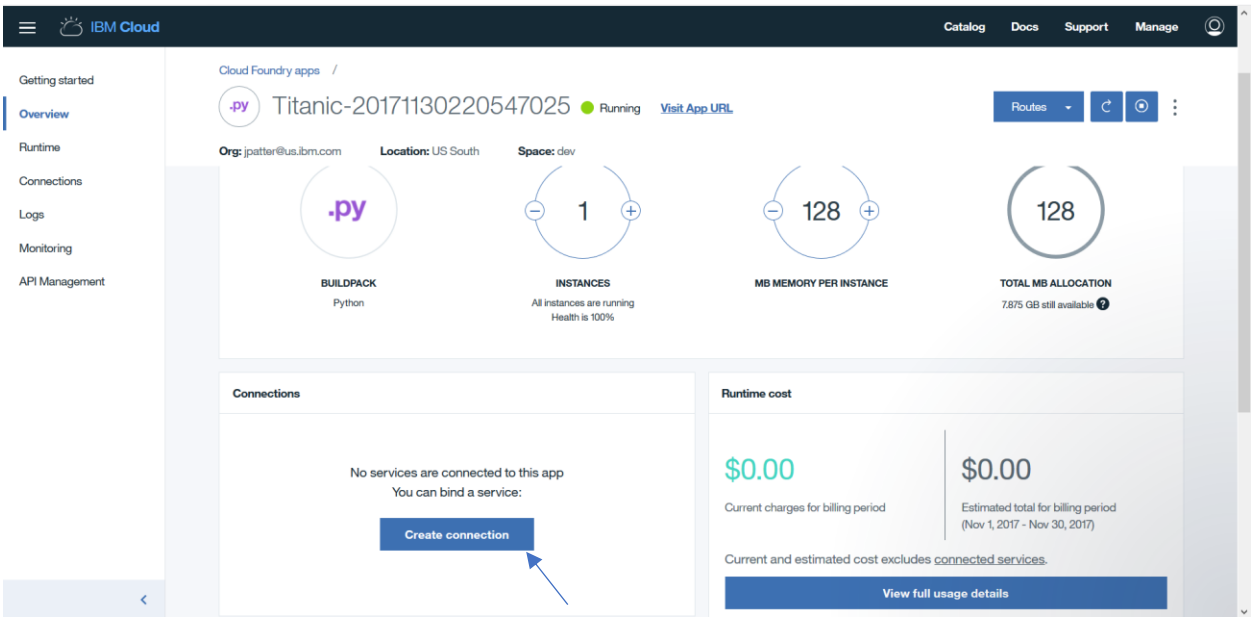
6. Close the Titanic prediction app tab, and click on the ☰ icon and Dashboard in the pulldown to navigate to the Dashboard where the running application should be listed.



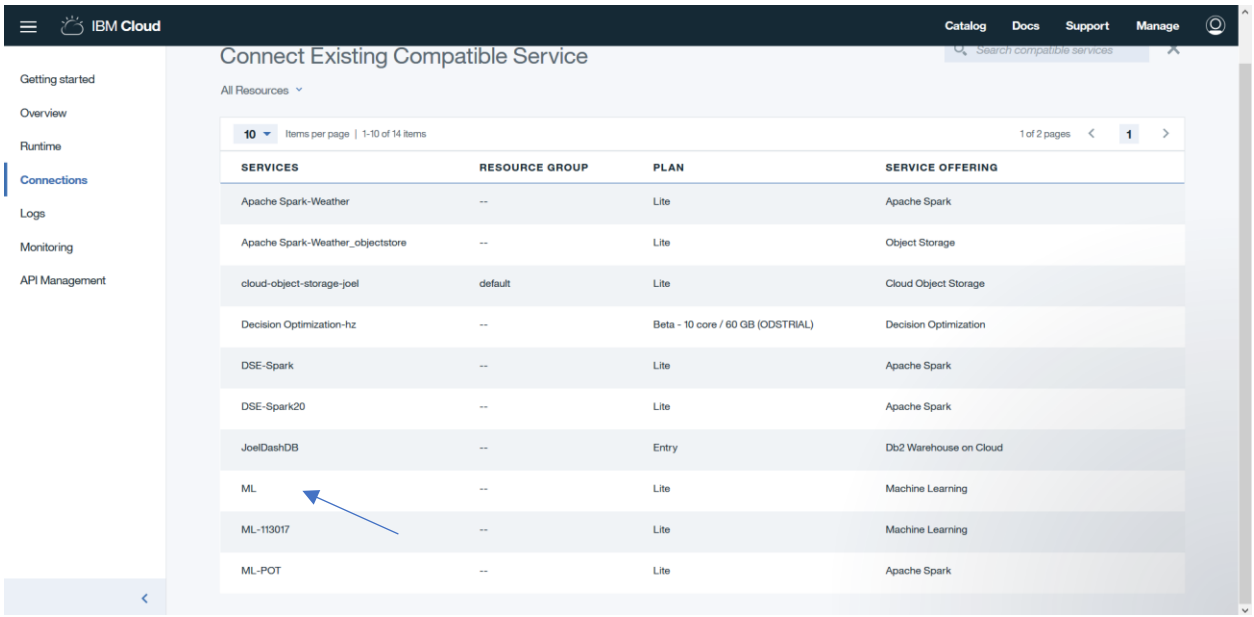
7. We are now going to connect the application to the Watson Machine Learning service that was created earlier. Click on the application name.



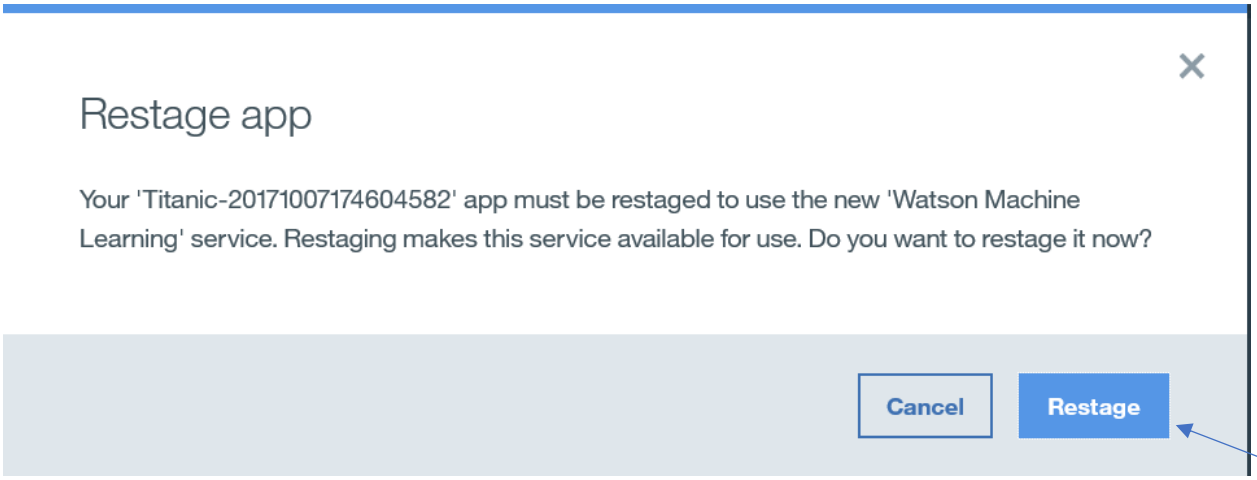
8. Scroll down until you see the Connections panel. Click on **Create Connection**.



9. 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**.

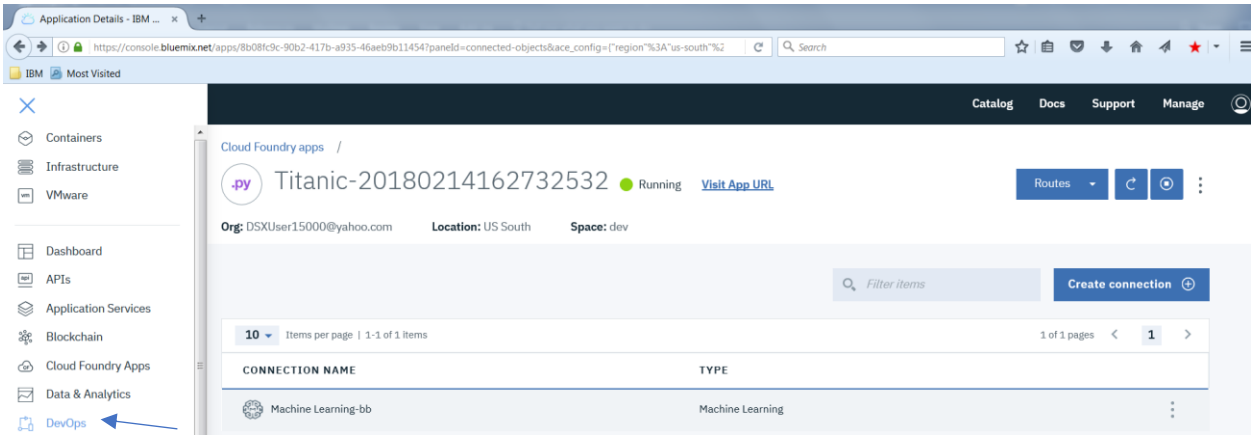


10. You will get a pop up that asks to Restage the application. Click on **Restage.**

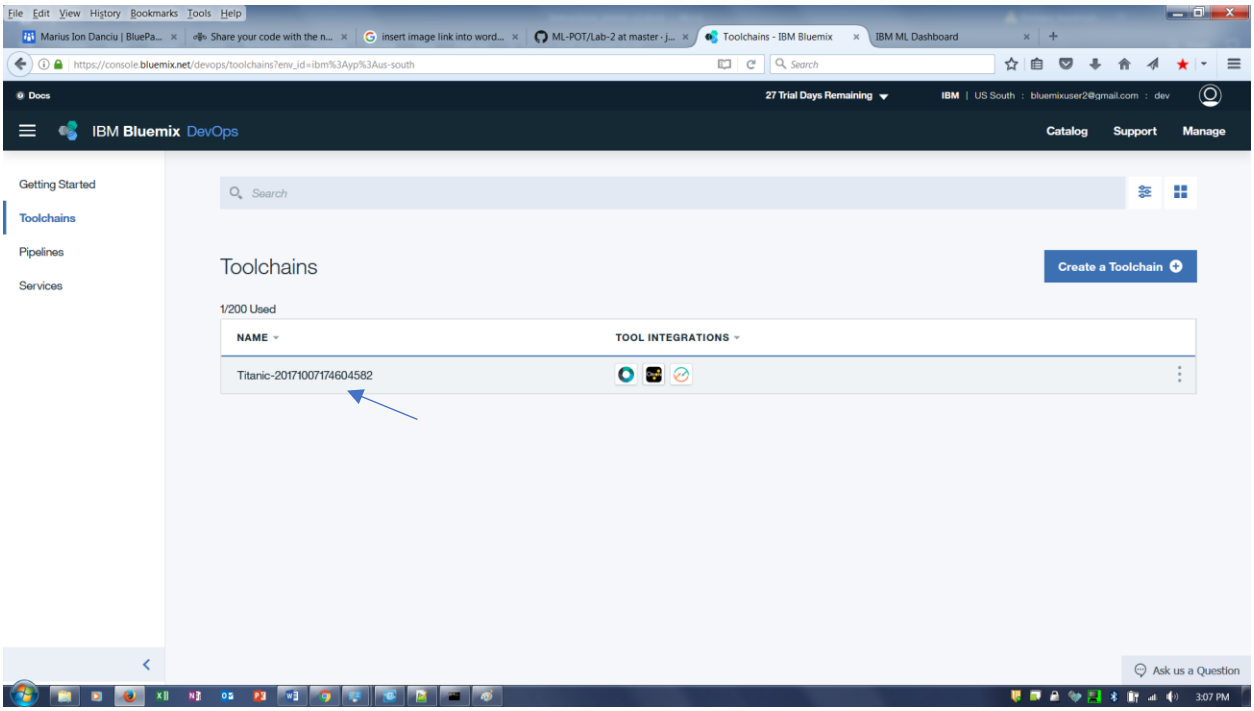


11. Wait for the application status to indicate Running

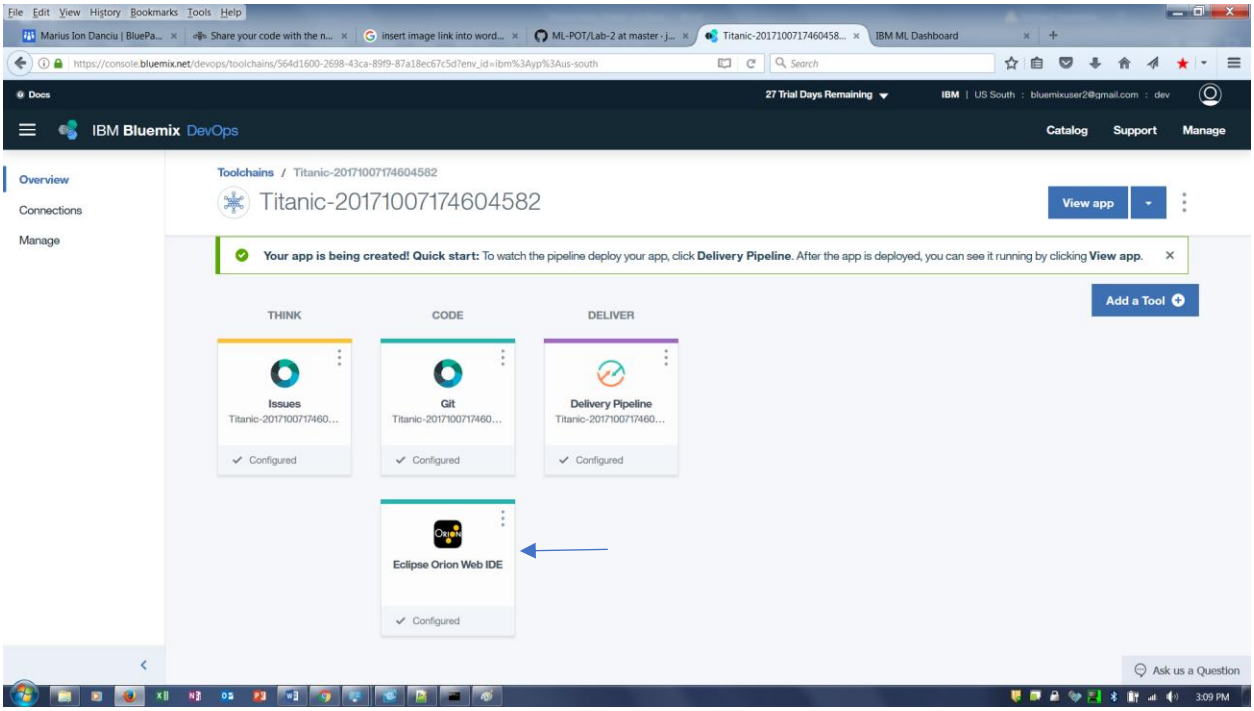
12. We now have tied the web application to the Watson Machine Learning service. Note that the Watson Machine Learning service could have more than one deployed model available to select and then embed in the web application. In our case, we have only one deployed model. We now need to copy the scoring 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.



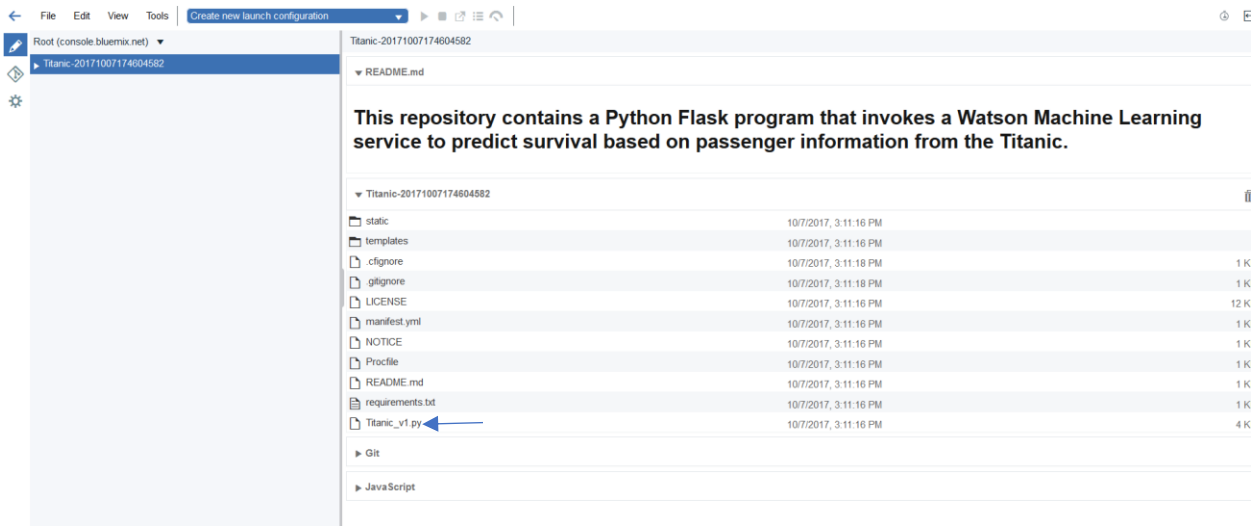
13. We are now going to paste the scoring endpoint into the application code. Click on the Toolchain (Titanic-2017xxxxx below).



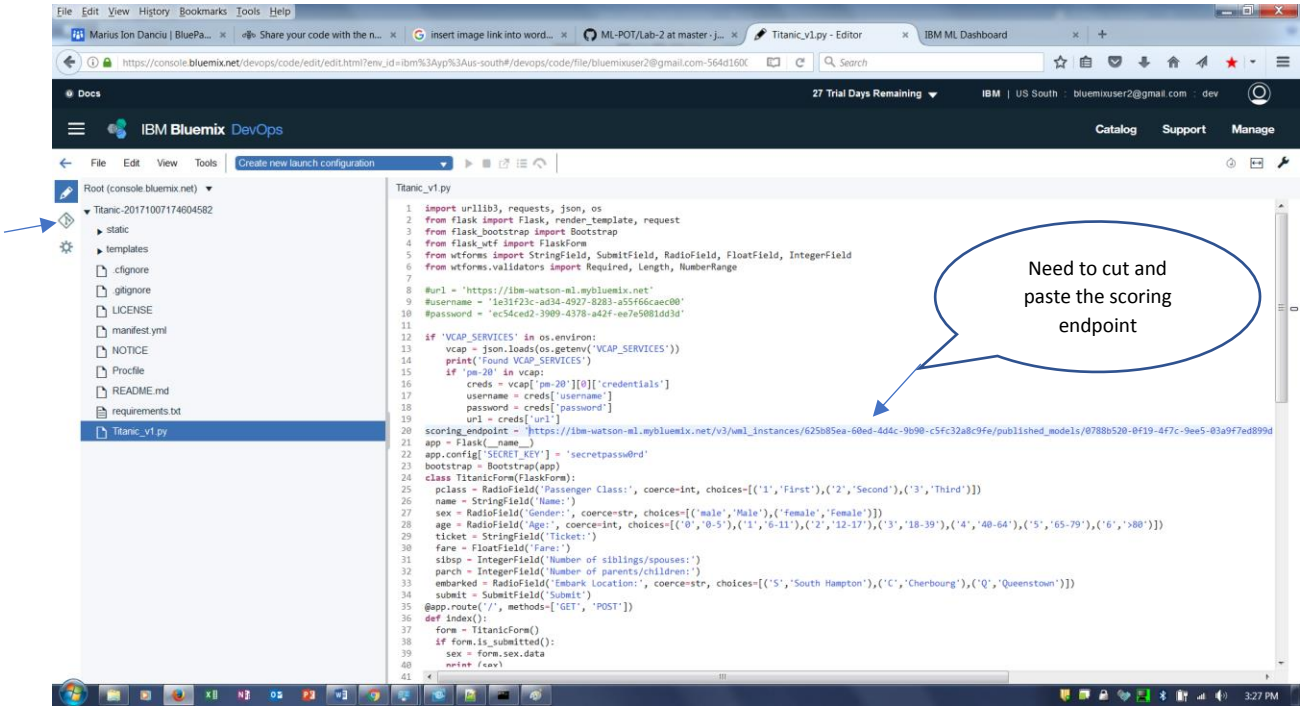
14. Click on the Eclipse Orion Web IDE.



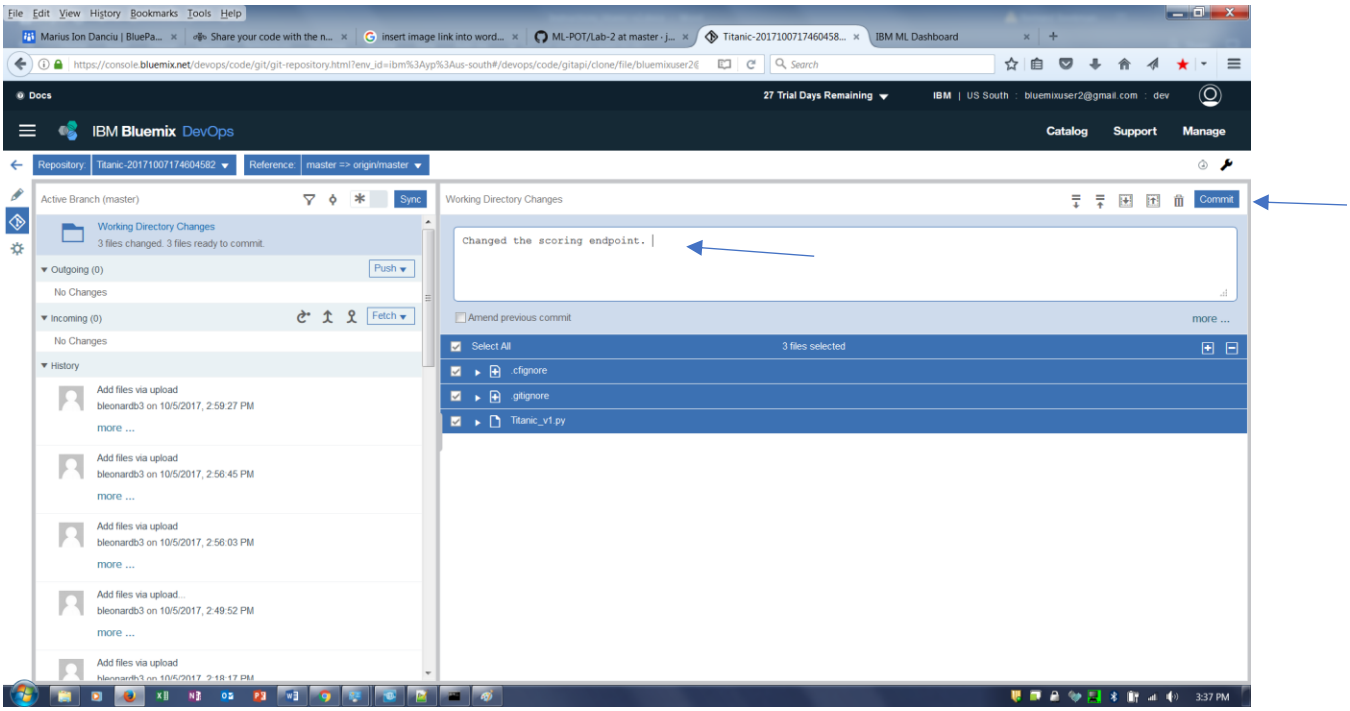
15. Click on the Titanic_v1.py file. This is a python source file.



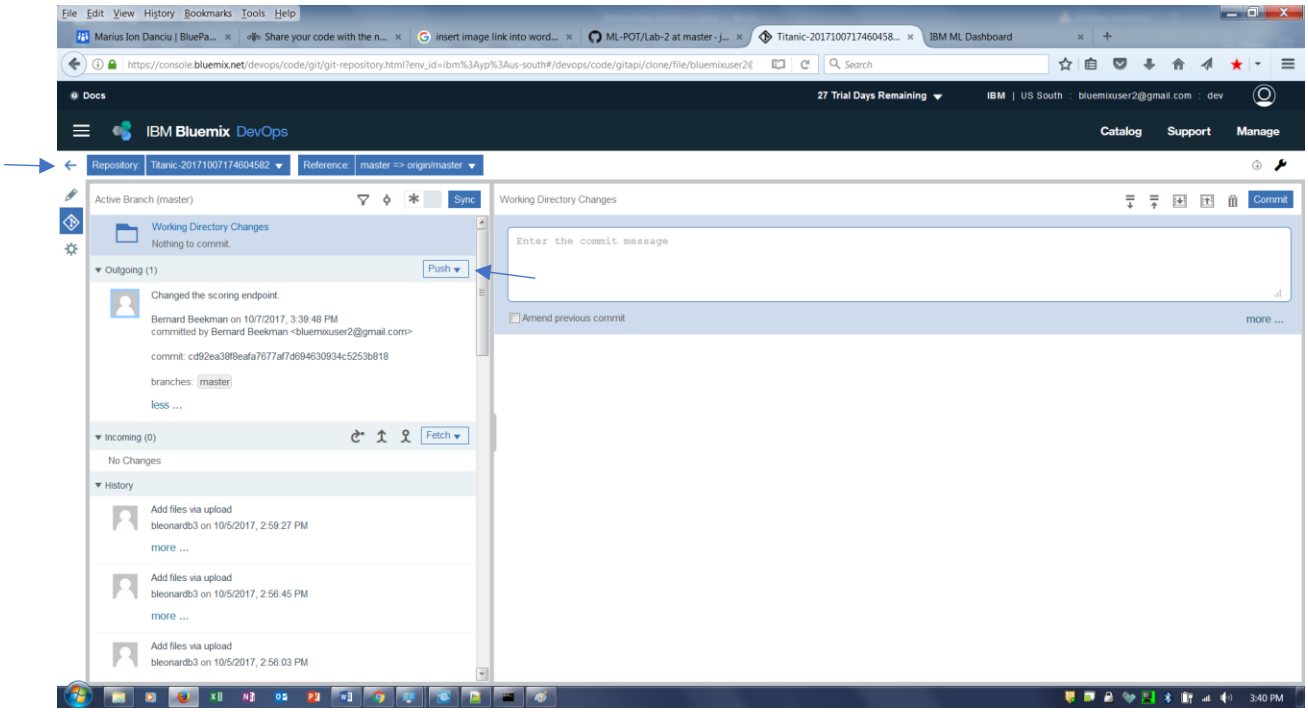
16. Go back to the Notepad file, and copy the scoring endpoint to the clipboard. Look around line 20 in the Titanic_v1.py file for the "scoring endpoint =". Select the scoring endpoint value in line 20 (starting with https:// may want to use Shift-End to get to the end of the line, and then back up one 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**.

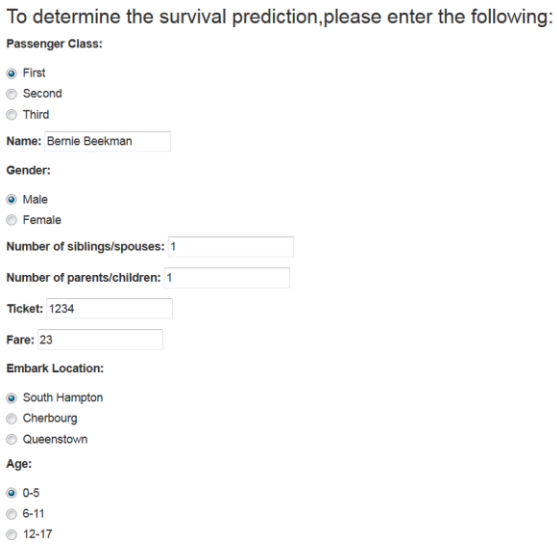


18. Then click on **Push** to push the changes to the central Git repo which will start the build and deploy of the application. Click on the left arrow to return to the Toolchain.



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).



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.