# AutoAI Lab + DevOps

This lab consists of two parts. The first part will demonstrate the new and exciting AutoAI capability to build and deploy an optimized model based on the Titanic data set. The second part will deploy an application using the IBM Cloud DevOps toolchain that will invoke the deployed model to predict whether a passenger would have survived.

AutoAI in Watson Studio automatically analyzes your data and generates candidate model pipelines customized for your predictive modeling problem. AutoAI algorithms analyze your dataset to discover data transformations, estimator algorithms, and parameter settings that work best for your problem setting. Results are displayed on a leaderboard, showing the automatically generated model pipelines ranked according to your problem optimization objective.

Using AutoAI, you can build and deploy a machine learning model with sophisticated training features and no coding. The tool does most of the work for you.
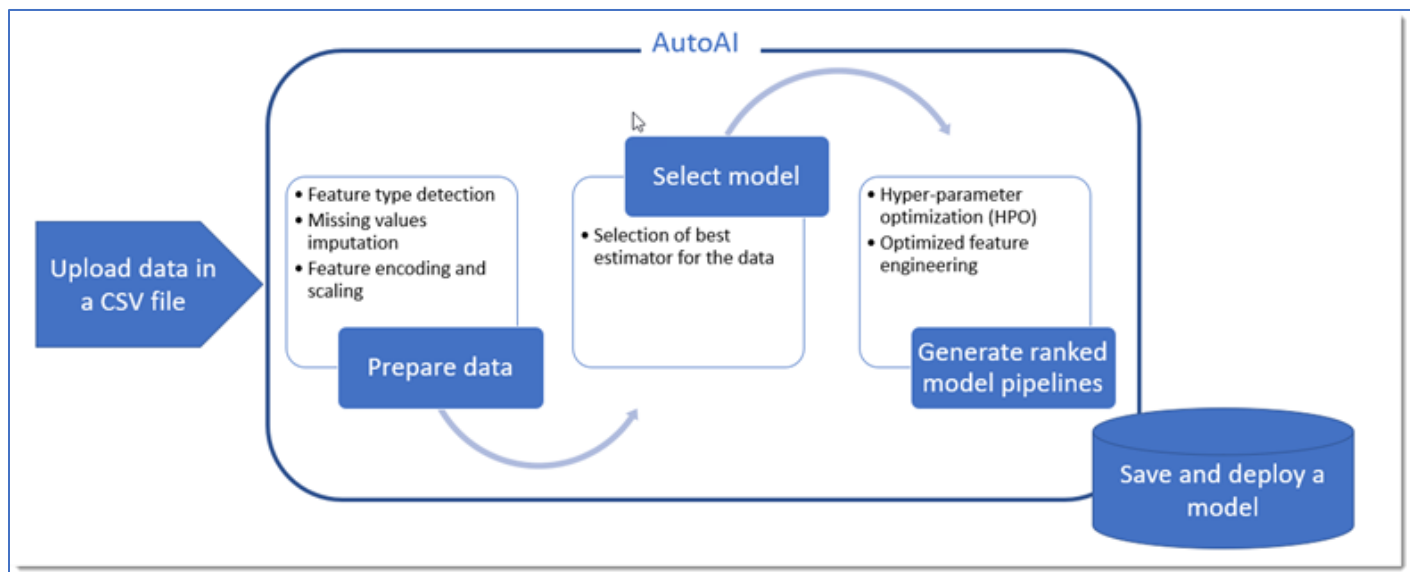
The AutoAI process follows this sequence to build candidate pipelines:

- Data pre-processing
- Automated model selection
- Automated feature engineering
- Hyperparameter optimization

We will perform the following steps in this lab:

1. Download a Titanic cleansed data set
2. Add an Auto AI Experiment
3. Save and Deploy the selected model.

4. Deploy a simple web front-end application and connect it to the deployed model using an IBM Cloud toolchain.
5. Make a coding change and re-build the application using the DevOps toolchain
6. Run the application, input passenger data, and receive the survivability prediction.

Step 1: Download the titanic_cleansed.csv data set

1. Download the **titanic_cleansed.csv** data file from the following location by clicking on the link here. Note this is a different file than used in the previous labs.

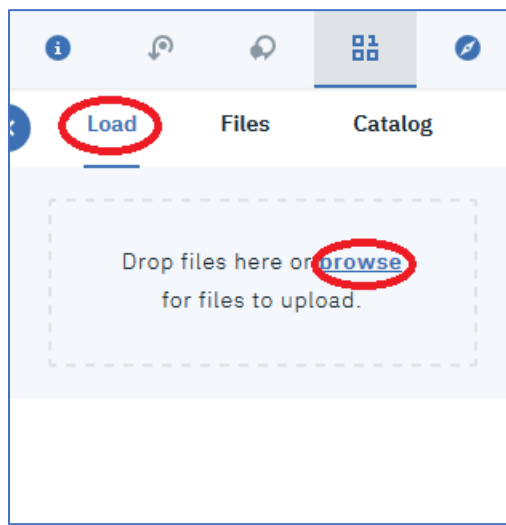2. Right-click on the window, and click **Save Page As…**
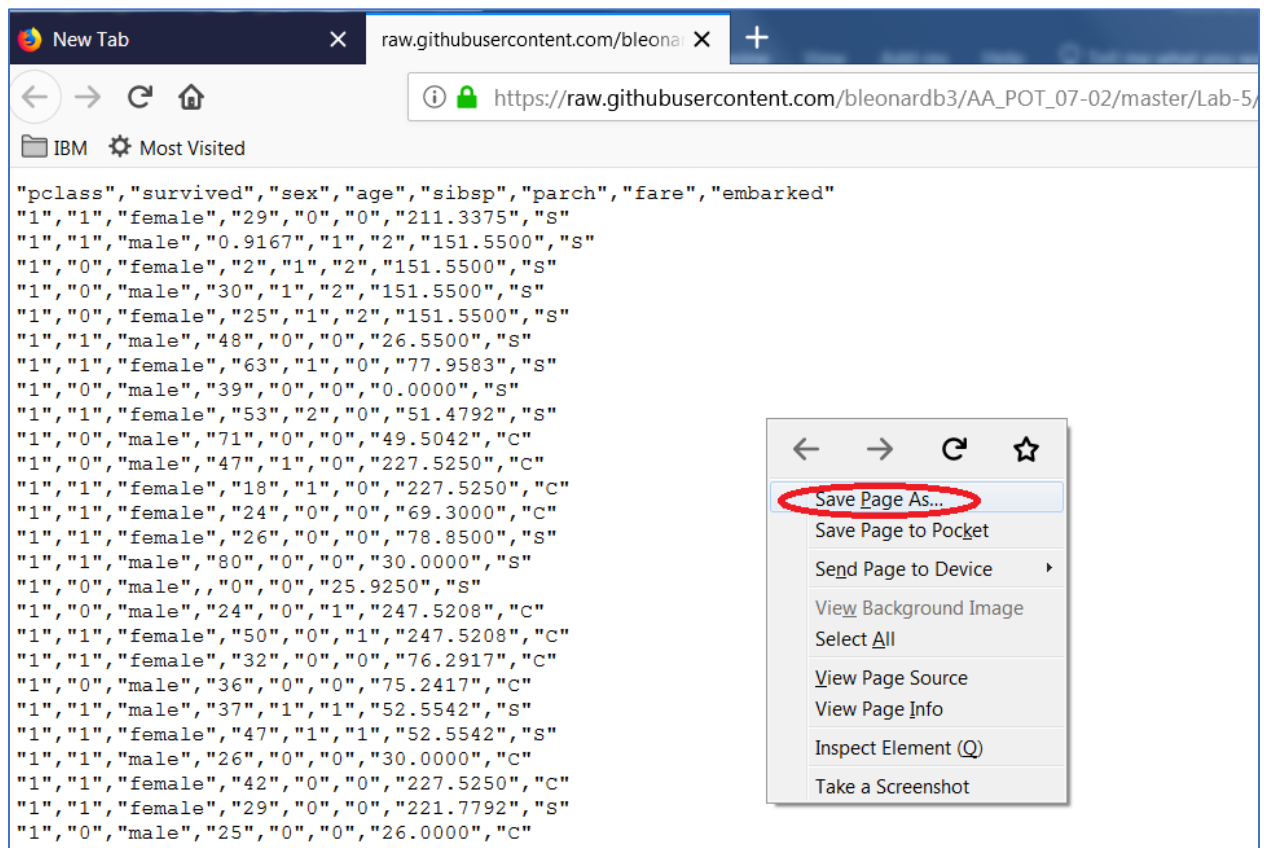
3. Click on **Save**. Note, if the file is named titanic_cleansed.csv.txt, change it to be titanic_cleansed.csv.
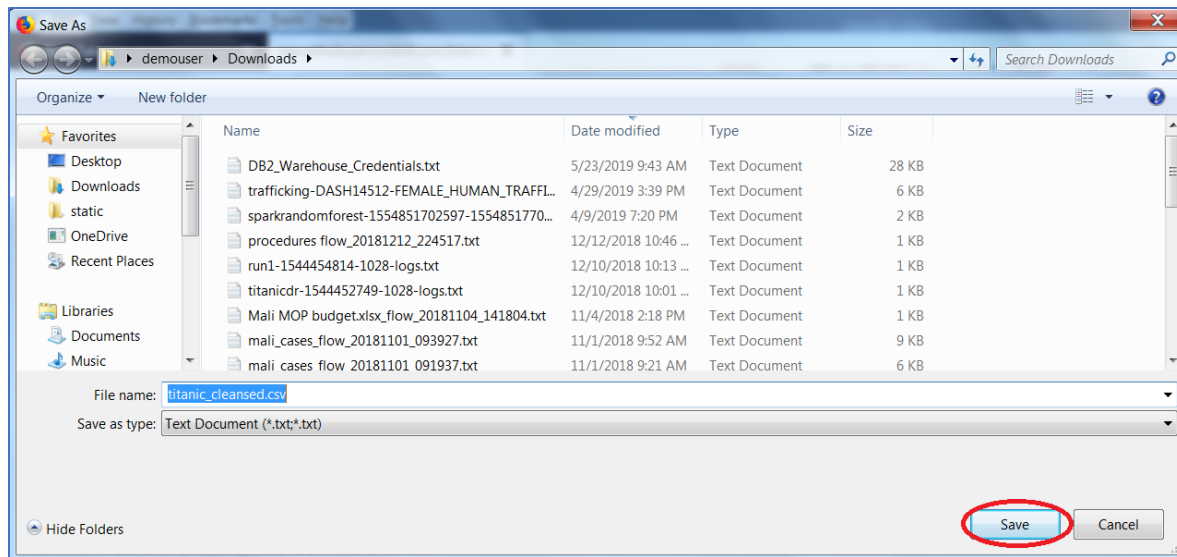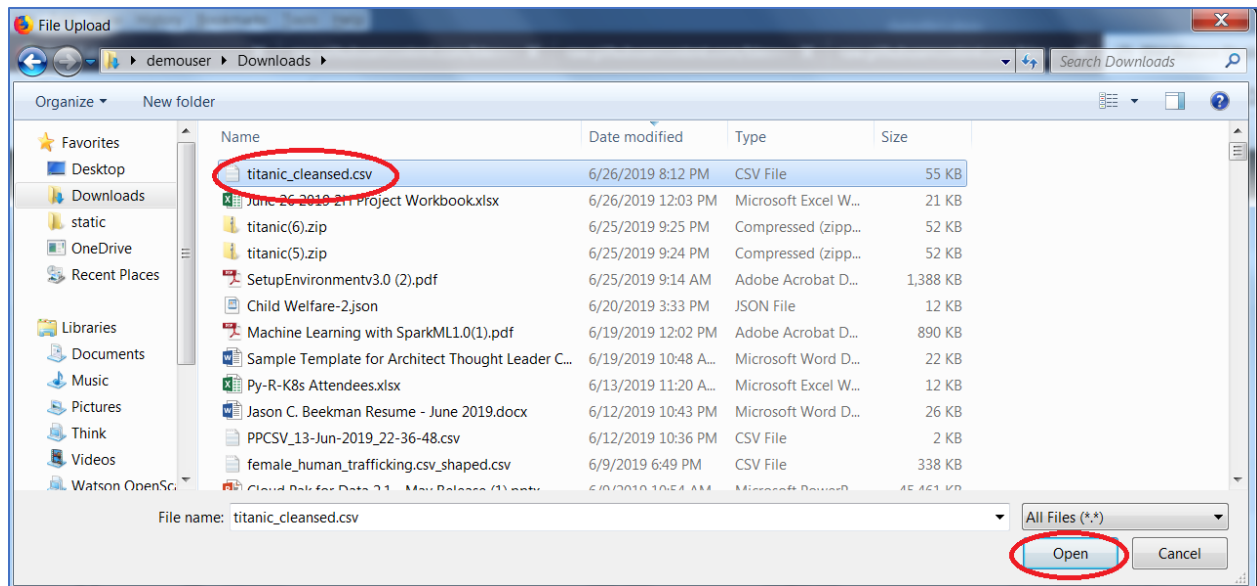
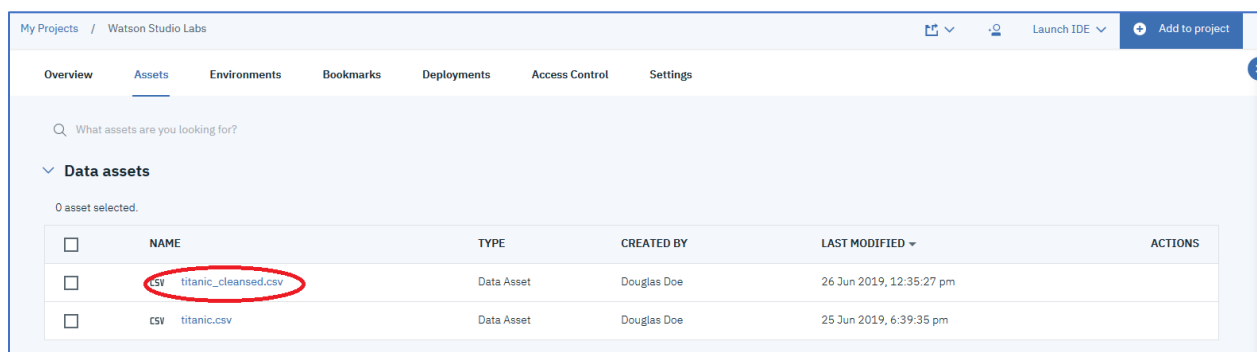4. Go back to your Watson Studio Labs project. Click on the ⬚ icon.

5. Click on the **Load** tab and then click on **browse**. If you don't see the **Load** tab, click on the ⬚ icon again.

6. Go to the folder where the titanic_cleansed.csv file is stored. Select the titanic_cleansed.csv file and then click **Open**.
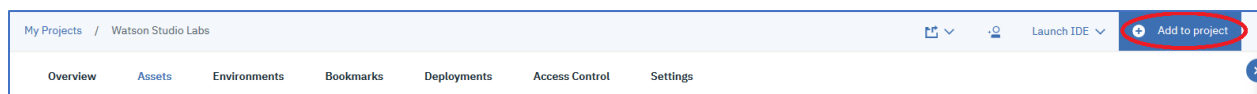
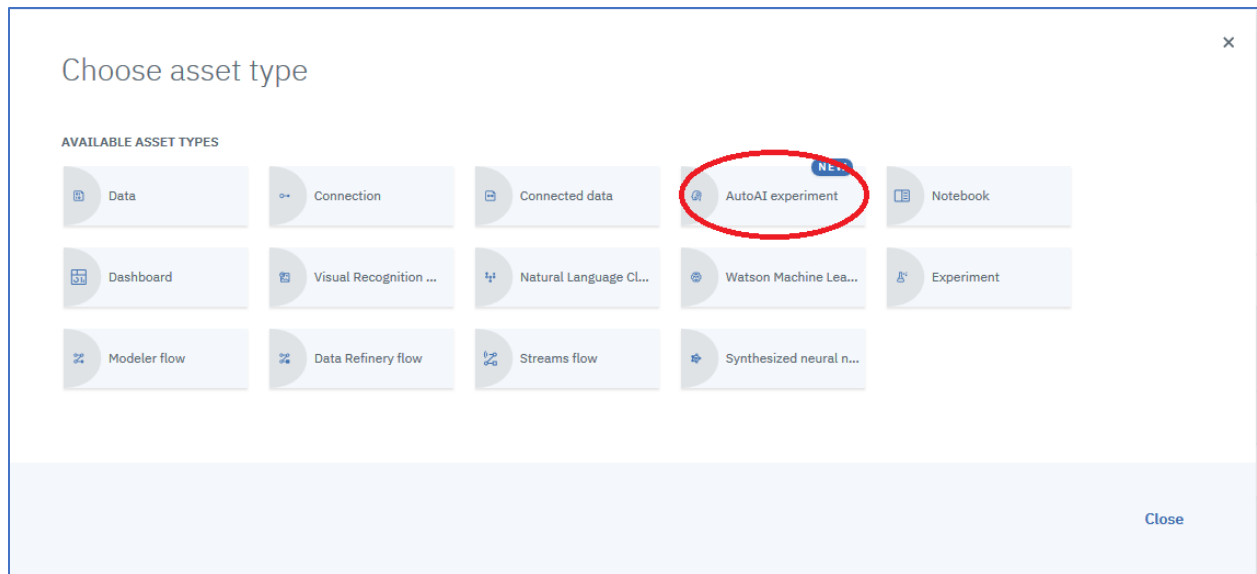7. The titanic_cleansed.csv file is now added as a Data Asset.

Step 2: Add an AutoAI Experiment

1. Click on **Add to project**.

2. Click on **AutoAI experiment**

3. Enter an **Asset name**, leave the defaults for the **Watson Machine Learning** and **Compute configuration** and click on **Create**.

4. Click on **Select from project**.
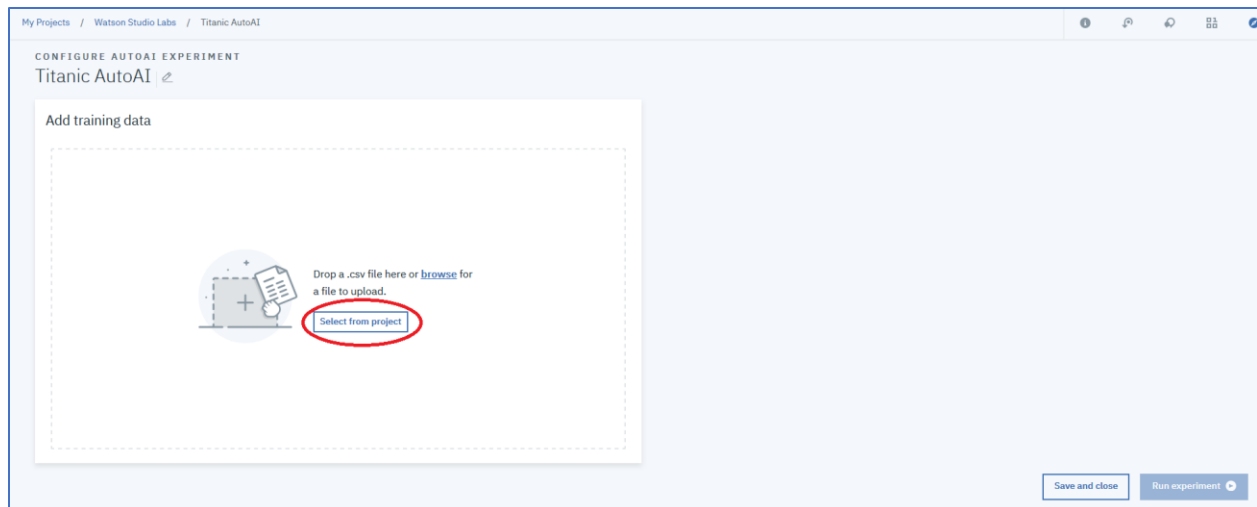
5. Click on **titanic_cleansed.csv** and then click on **Select asset**.
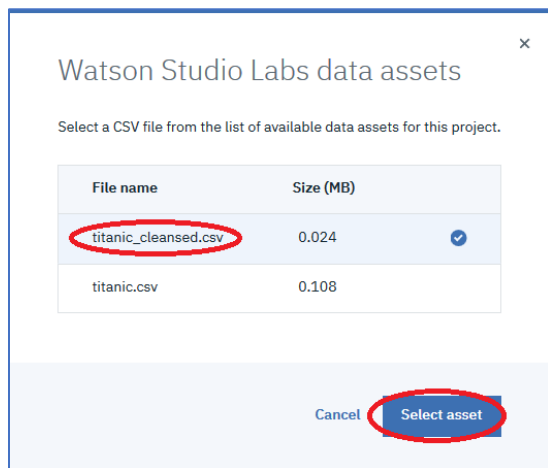
6. Click on **survived** as **the column to predict**. Note, based on this selection, the **Prediction Type** is **Binary Classification**, and the **Optimized Metric** is the **Area under the Receiver Operating Characteristic**(ROC AUC) curve. Further note, the **Positive Class** is correctly defaulted as "1" – survived.

Select column to predict

DATA SOURCE: titanic_cleansed.csv

Column name	Type
pclass	Integer
survived	Integer
sex	String
age	Decimal
sibsp	Integer
parch	Integer

Selected prediction

PREDICTION TYPE | POSITIVE CLASS | OPTIMIZED METRIC
Binary Classification ⓘ | **1** | **ROC AUC** ⓘ

Experiment settings ⚙ | Run experiment ▶

7. You can click on **Experiment settings** to change the defaults. For now, accept the defaults and click on **Run experiment.**
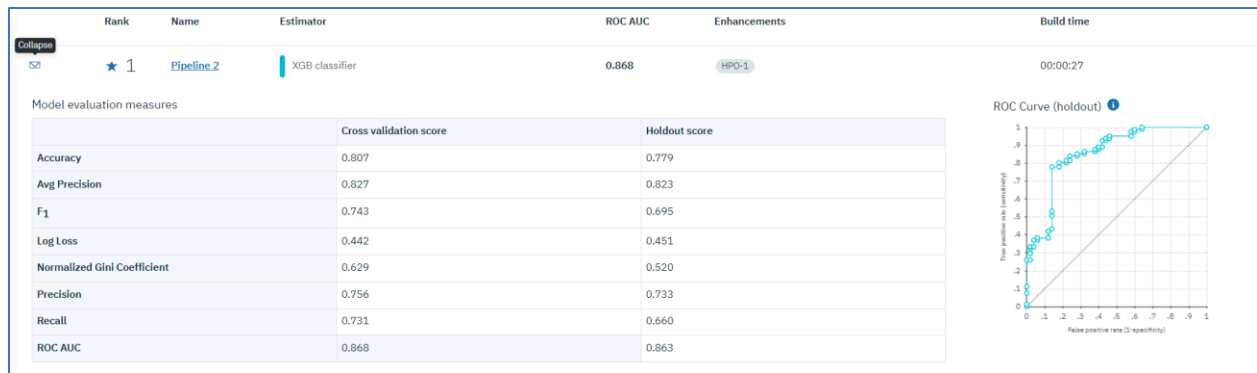
8. Please wait several minutes as the alternative solutions are analyzed. The pipeline summary is then displayed. Click on the right arrow ⊠ next to the top ranked pipeline.

9. Scores are displayed for different metrics for both the training sample and the holdout sample.

Rank	Name	Estimator	ROC AUC	Enhancements	Build time
★ 1	Pipeline 2	XGB classifier	0.868	HPO-1	00:00:27

Model evaluation measures

	Cross validation score	Holdout score
Accuracy	0.807	0.779
Avg Precision	0.827	0.823
F_1	0.743	0.695
Log Loss	0.442	0.451
Normalized Gini Coefficient	0.629	0.520
Precision	0.756	0.733
Recall	0.731	0.660
ROC AUC	0.868	0.863

ROC Curve (holdout)

10. Click on the **Pipeline** link for the top ranked pipeline.

Rank	Name	Estimator	ROC AUC	Enhancements	Build time
★ 1	Pipeline 2	XGB classifier	0.868	HPO-1	00:00:27

Model evaluation measures

	Cross validation score	Holdout score
Accuracy	0.807	0.779
Avg Precision	0.827	0.823
F_1	0.743	0.695
Log Loss	0.442	0.451
Normalized Gini Coefficient	0.629	0.520
Precision	0.756	0.733
Recall	0.731	0.660
ROC AUC	0.868	0.863

ROC Curve (holdout)

11. The model evaluation metrics for the training sample are repeated. On the left are options for additional information. Click on the **Confusion Matrix**.

← Back to Titanic Auto AI

RANK 1 Pipeline 2 ▾

ROC AUC (Optimized) 0.868   ESTIMATOR XGB classifier   ENHANCEMENTS HPO-1   BUILD TIME 00:00:27   Save as model

XGB Classifier

EVALUATION

Model Evaluation

Confusion Matrix

Precision Recall Curve

MODEL VIEWER

Model Information

Feature Importance

Model Evaluation ⓘ
TARGET : SURVIVED

ROC Curve ⓘ

Model Evaluation Measures

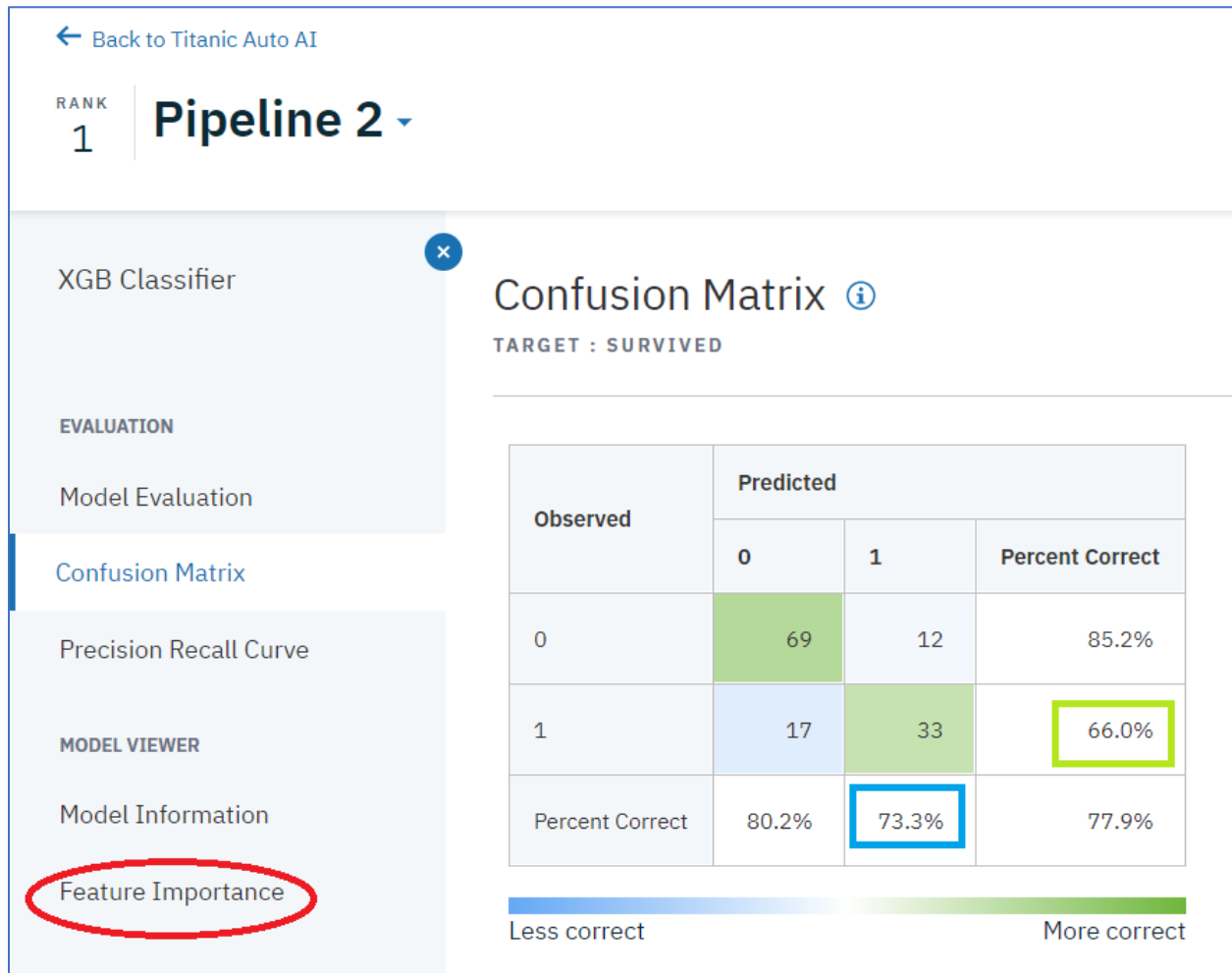|	Holdout Score	Cross Validation Score
Accuracy	0.779	0.807
Area Under ROC Curve	0.863	0.868
Precision	0.733	0.756
Recall	0.660	0.731
F_1 Measure	0.695	0.743
Average Precision	0.823	0.827

12. The Confusion Matrix is displayed for the holdout sample. The different metrics are computed based on the numbers in the Confusion Matrix. For example, Precision is defined by the percentage of predicted positives that are actually positive (i.e. the percentage of predicted survivors that survived). Recall is defined as the percentage of observed positives that the model predicts are positive (i.e. the percentage of survivors

that the model predicted would survive). Note the higher the Precision the lower the Recall.

Precision = True Positive/ (True Positive + False Positive) – shown inside blue rectangle on diagram below.

Recall = True Positive/(True Positive + False Negative) - shown inside green rectangle on diagram below

After viewing the Confusion Matrix, click on the **Feature Importance** option.

← Back to Titanic Auto AI

RANK
1 **Pipeline 2** ▾

XGB Classifier

Confusion Matrix ⓘ
TARGET : SURVIVED

EVALUATION

Model Evaluation

Confusion Matrix

Precision Recall Curve

MODEL VIEWER

Model Information

Feature Importance

Observed	Predicted		
	0	1	Percent Correct
0	69	12	85.2%
1	17	33	66.0%
Percent Correct	80.2%	73.3%	77.9%

Less correct ▬▬▬▬▬▬▬ More correct

13. According to the Feature Importance, the sex variable is considered the most important feature followed by the passenger class.
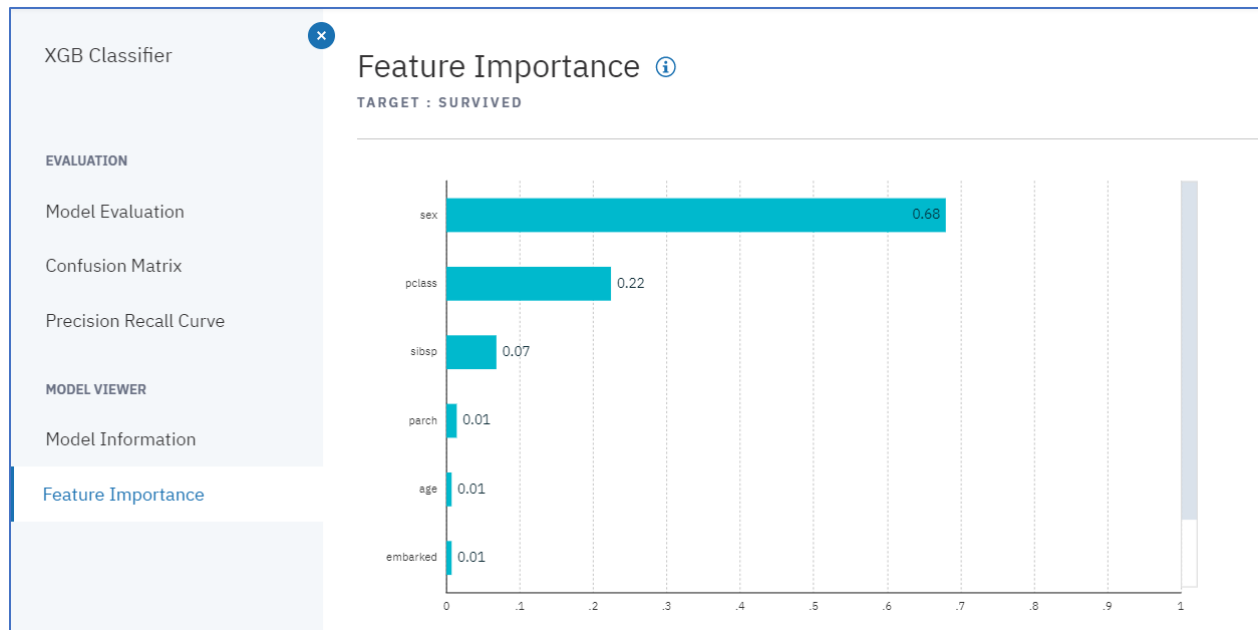
Step 3 – Save and Deploy the Selected Model

1. Click on **Save as Model**

2. Optionally change the default name and click on **Save**.

3. Click on **View in Project**.

4. Click on **Deployments**

5. Click on **Add Deployment**.

6. Enter a **Name** and click **Save**.

7. The model is successfully deployed on the IBM Cloud.

8. Click on **Titanic AutoAI Deployed**.

9. Click on **Test**.

10. Enter values for a passenger. For example,
    pclass – 1
    sex – female
    age – 5
    sibsp – 1
    parch – 2
    fare – 23
    embarked – S

    and click **Predict.**

Titanic AutoAI Deployed

Overview   Implementation   Test

Enter input data

pclass

1

sex

female

age

5

sibsp

Predict

11. The model predicts this passenger would survive, with almost 80% confidence. Click on **Implementation**.

Titanic Auto AI Deployed

Overview   Implementation   Test

Enter input data

1

_____

parch

2

_____

fare

23

_____

embarked

S

_____

Predict

```
{
  "predictions": [
    {
      "fields": [
        "prediction",
        "probability"
      ],
      "values": [
        [
          1,
          [
            0.20552849769592285,
            0.79447150230401771
          ]
        ]
      ]
    }
  ]
}
```

12. 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, or just leave this panel available to cut and paste the scoring endpoint. We will need the scoring endpoint in the next section.

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 web front-end application 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 account, when you click **Deploy**, it is automatically added with the free Lite plan selected.

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

1. Click on the **Deploy to IBM Cloud** 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 IBM Cloud

2. In the **Select Region**, make sure that the region is Dallas. If not, change it to Dallas, then click **Create**.

3. Scroll down and click on the **Create**+ button to create an IBM Cloud API key.

4. Click on the **Create** button.

Create a new API key with full access

Warning: This will create a new API key that allows anyone who has it the ability to do anything you could do. You can improve your security posture by using the IAM UI to create a service ID API key that limits access to only what your pipeline requires, and then pasting that into the template UI instead.

Key will be called: API Key for TitanicNew-20190925163447233

For more information on API keys and access see the IAM documentation.

Cancel   **Create**

5. Please wait until the Region, Organization, and Space are filled in. **Note that these must match the corresponding Region, Organization, and Space where the Titanic model was deployed.** Make sure the **Region** is Dallas(Production), and the Organization should display the userid, and the space should be filled in as well. If this is not the case, try a different **Region**. Click on **Create**.
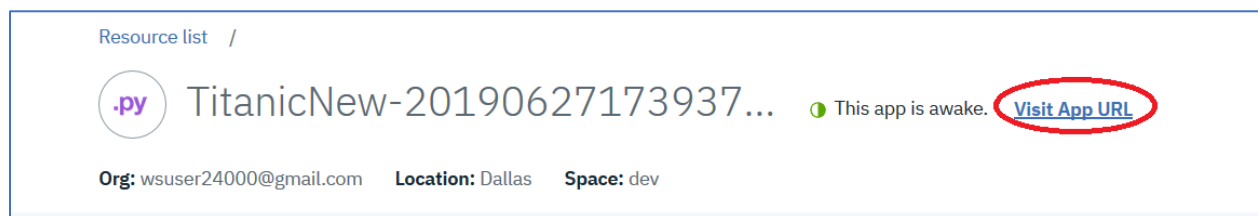

6. Your app is being created! To watch the pipeline deploy your app, click **Delivery Pipeline**.
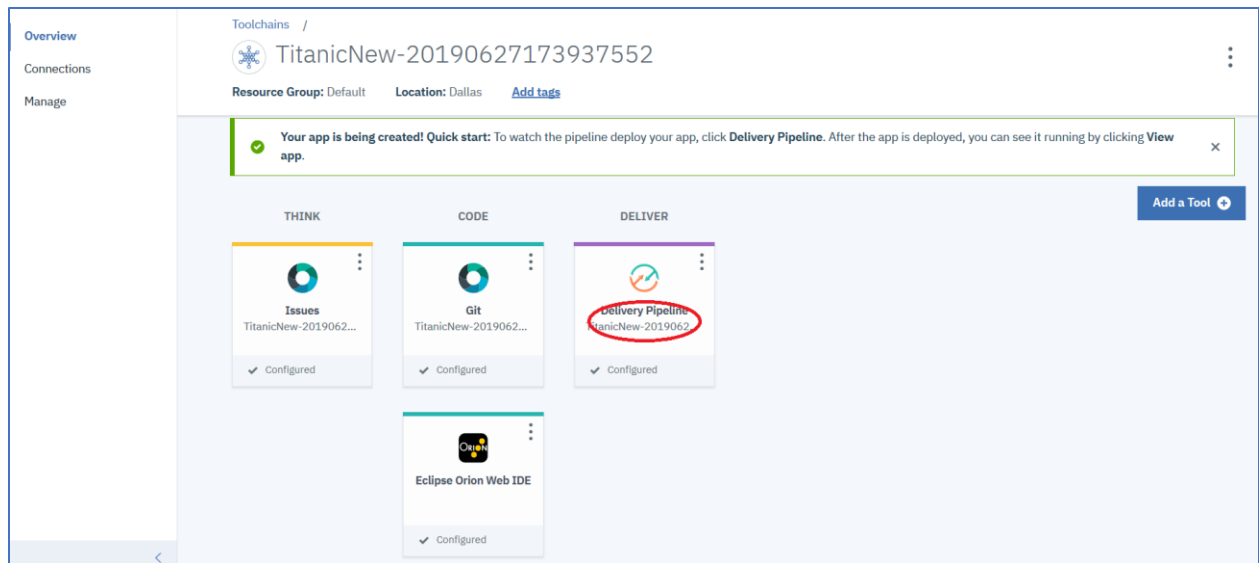
7. After the app is deployed successfully (should say Deploy Passed in the Deploy stage-may take about 2 minutes), view the running app by clicking on **View Console**
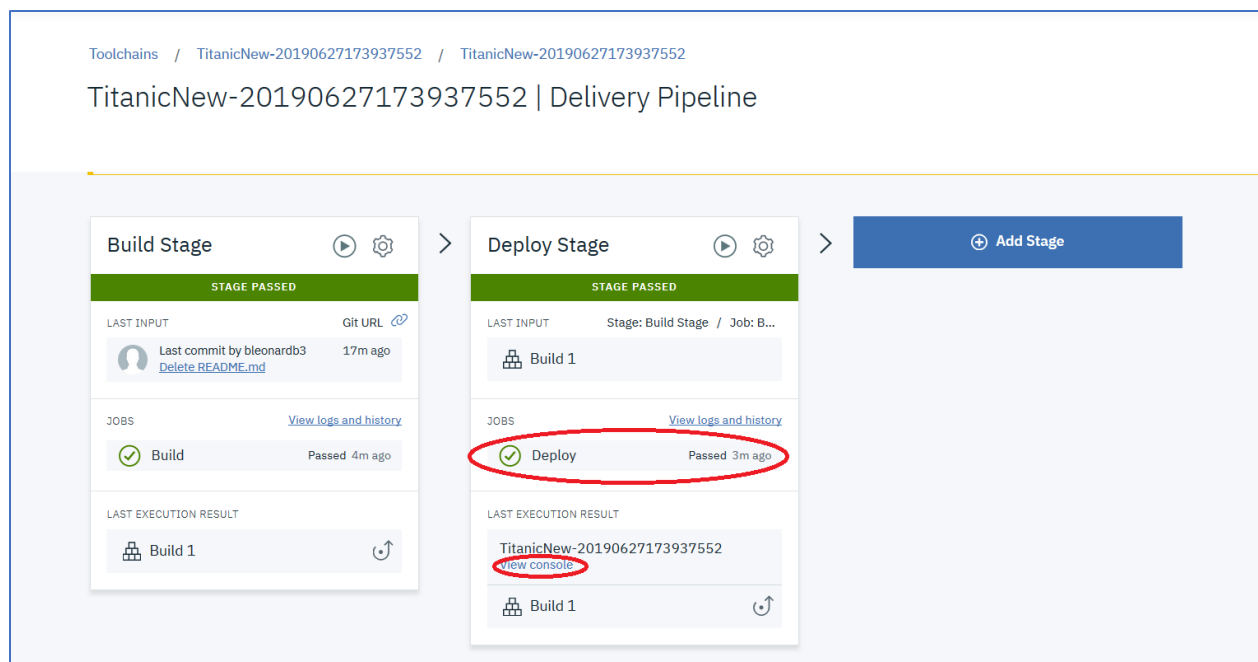
8. Click on **Visit App URL**

9. 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! Close the Titanic Prediction browser tab.

Titanic Prediction

To determine the survival prediction,please enter the following:

Passenger Class:

○ First
○ Second
○ Third

Gender:

○ Male
○ Female

Number of siblings/spouses: [　　　　　　]

Number of parents/children: [　　　　　　]

Fare: [　　　　　　]

Embark Location:

○ South Hampton
○ Cherbourg
○ Queenstown

Age: [　　　　　　]

[Submit]

10. We are now going to connect the application to the Watson Machine Learning service that was created earlier. Scroll down until you see the Connections panel. Click on **Create Connection**.

11. You should see at least 2 services listed, a Cloud Object Storage service and a Watson Machine Learning service. Point the cursor on the **Watson Machine Learning** service for your application, and then click on **Connect**.

12. A **Connect IAM-enabled service** pop up will appear. Click **Connect**.

13. A **Restage app** pop up will appear. Click on **Restage**.

14. Wait for the application status to change to ◑ This app is awake , or something similar.

15. 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
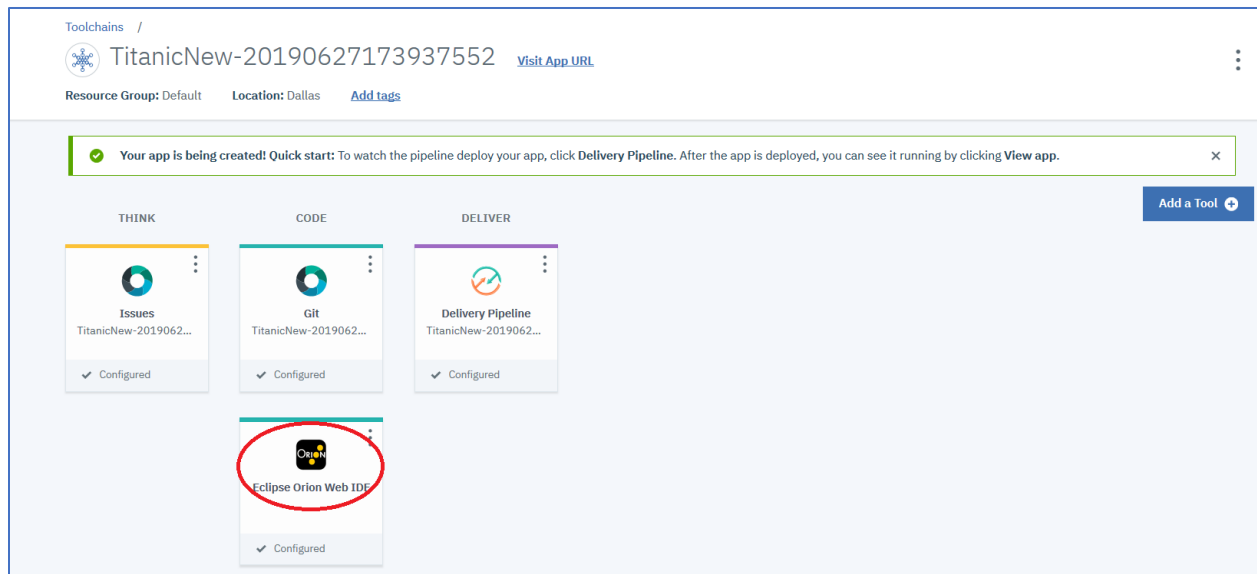
available to select and then embed in the web application. Click on the ▬ icon and click on DevOps in the pulldown to navigate to the Toolchain.
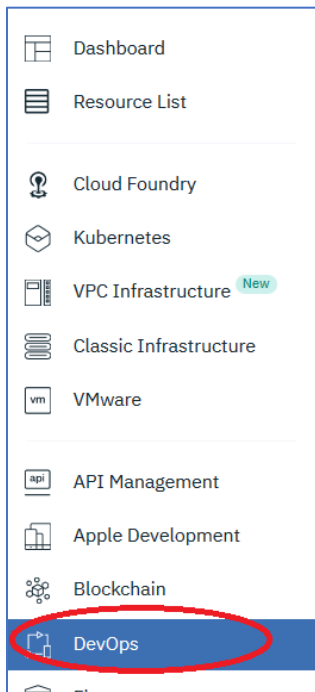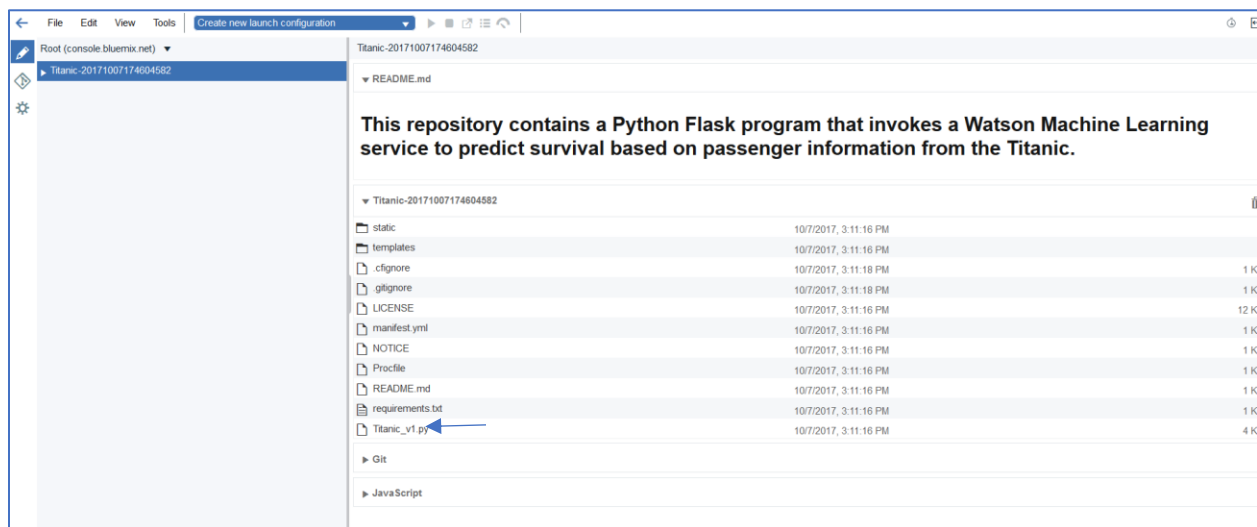

16. We are now going to paste the scoring endpoint into the application code. Click on the Toolchain (Titanic-2019xxxxx below). If no toolchain appears, switch the region to Dallas.
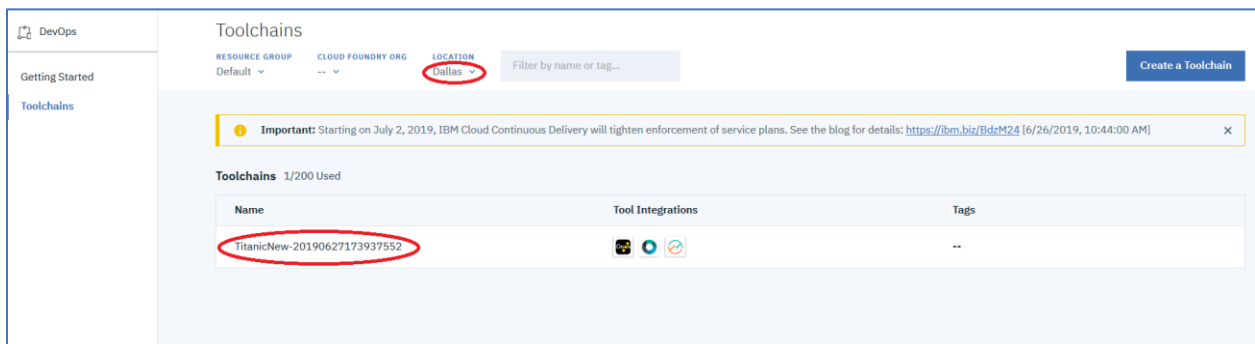
17. Click on the Eclipse Orion Web IDE.

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

19. 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:// you 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). Copy the scoring endpoint from the Watson Studio Implementation panel (in Notepad) and use Ctrl-V to paste the scoring endpoint into the Titanic_v1.py file. Enter Ctrl-S or File > Save to save the file. Then click on the icon on the top left.
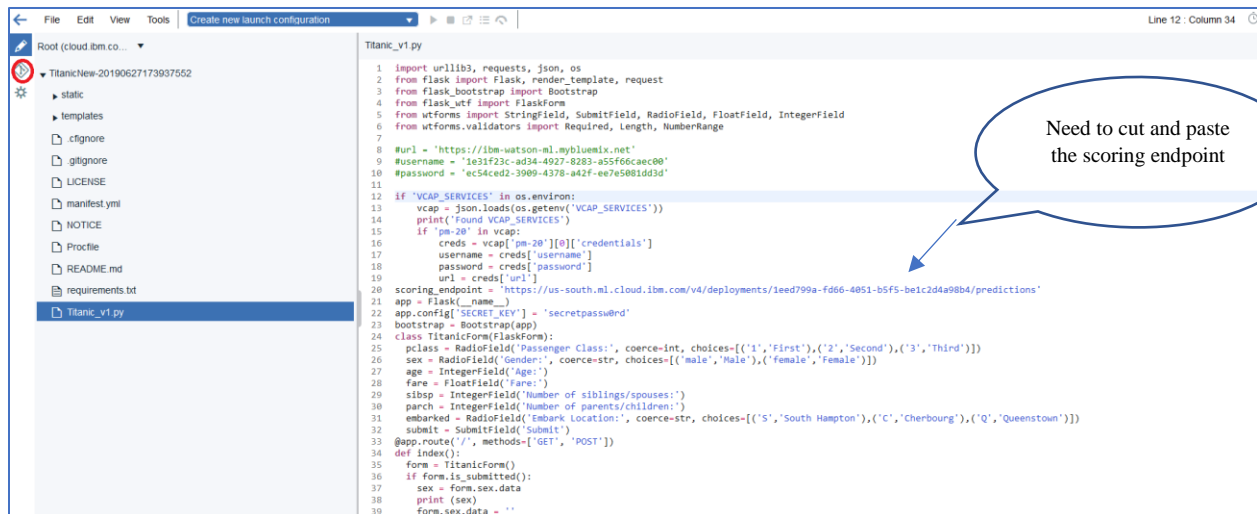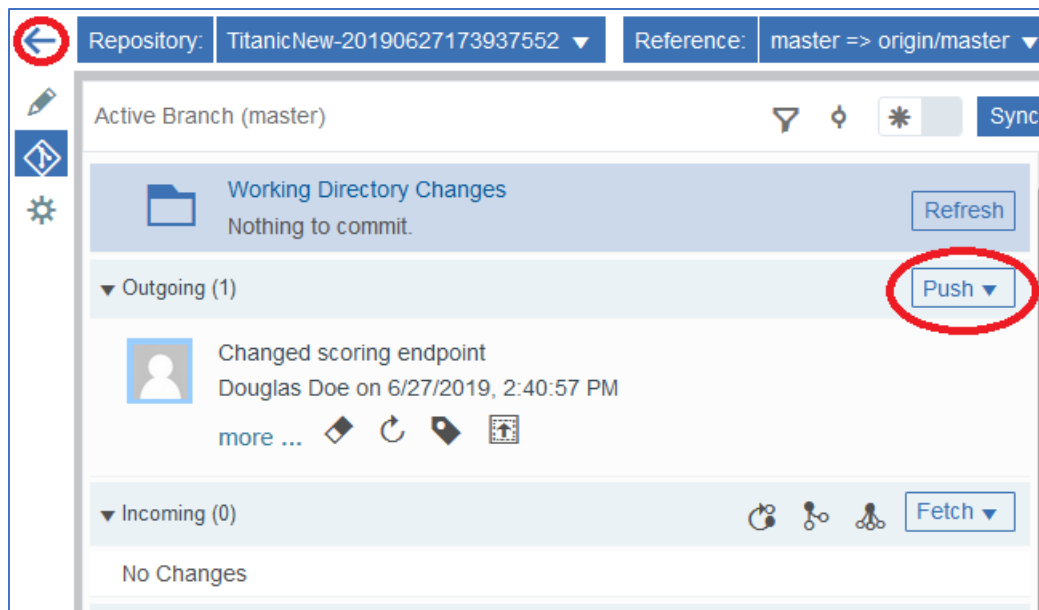
20. The next step is to commit the change to the git repository. Enter "Changed the Scoring Endpoint" in the Enter Commit Message field, and then click on **Commit**.
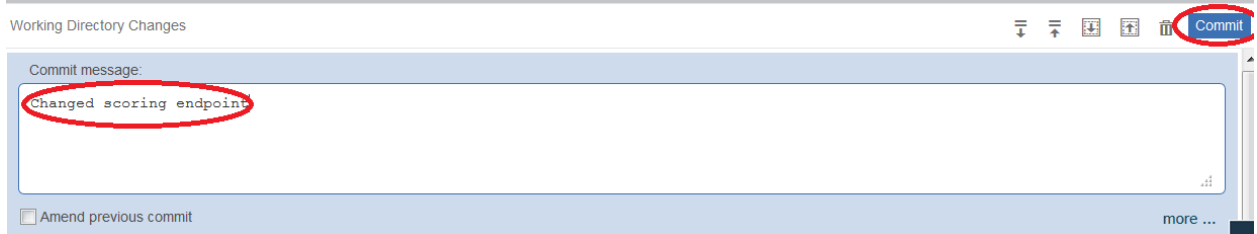
21. Then click on **Push** to push the changes to the central Git repo which will start the build and deploy of the application. Click on the left arrow to return to the Toolchain.

22. Click on the **Delivery Pipeline** to view status of the deployment as before. Once the Deploy Stage status shows **Deploy passed now** (it shouldn't take longer than 2 minutes

so reload the browser in case the UI didn't update), click on **View Console**, and then **Visit App URL** (refer to steps 6,7,8 above) to view the running application.

23. The web form should appear. Enter data in all the fields and click on the **Submit** button.

24. 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.

Congratulation! You have completed the Lab!!!

✓ Added an AutoAI Experiment

- ✓ Saved and Deployed the selected model
- ✓ Deployed a simple web front-end application and connected it to the deployed model.
- ✓ Made a coding change and re-built the application using the DevOps toolchain
- ✓ Ran the application, inputted passenger data, and received the survivability prediction.