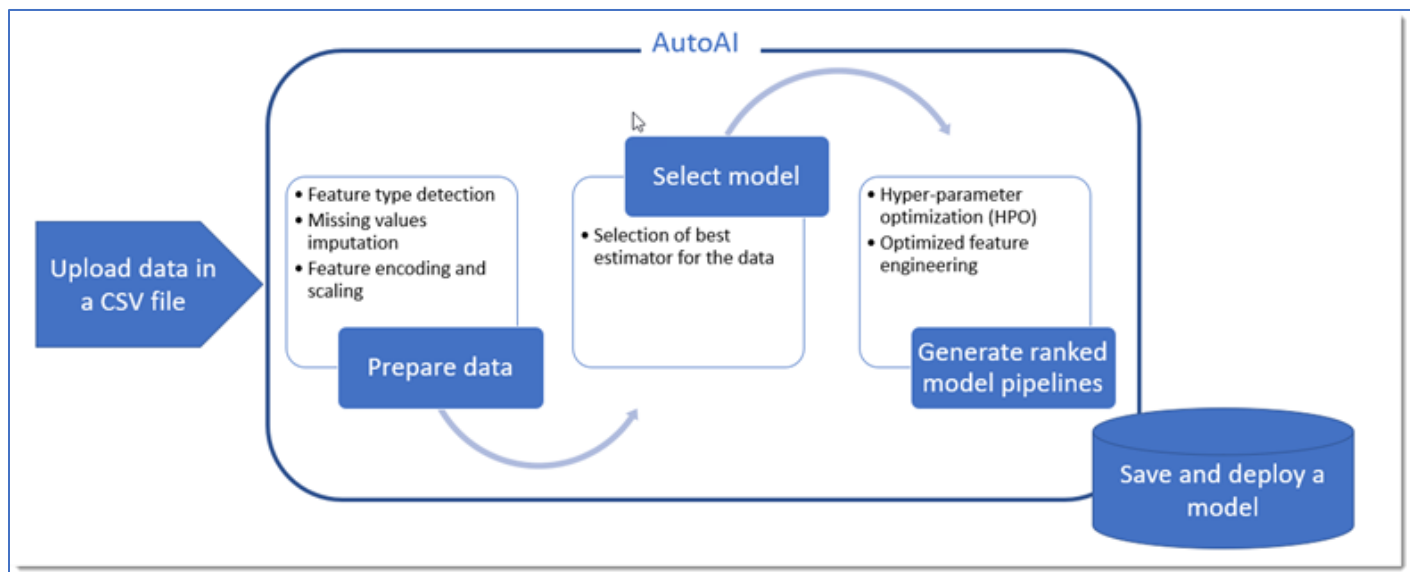


AutoAI Lab + DevOps

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](#)

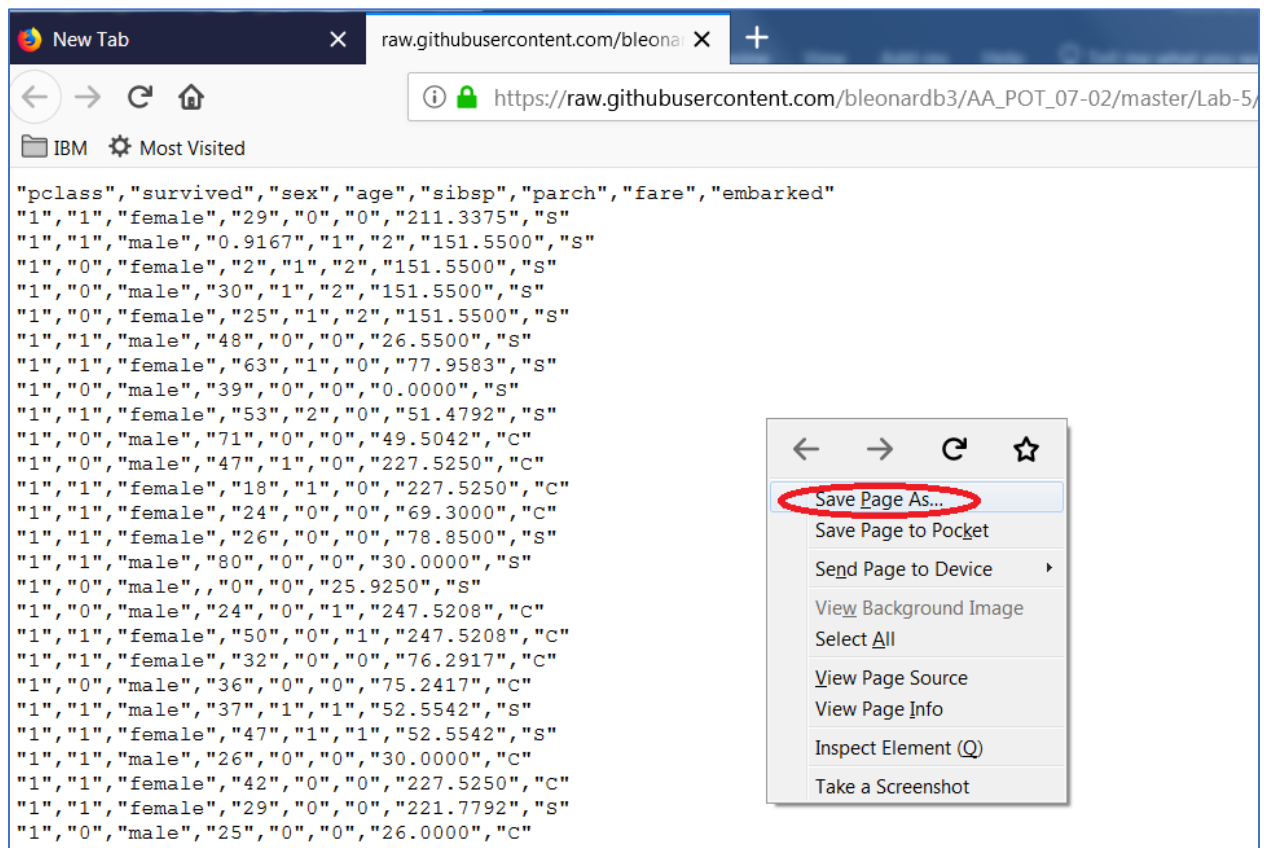
We will perform the following steps in this lab:

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

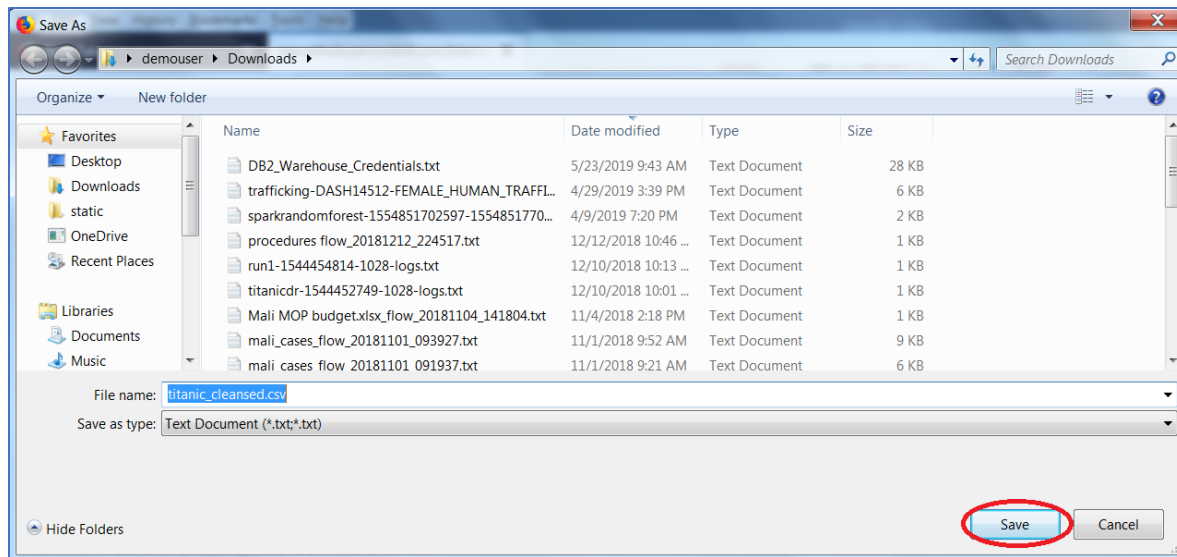
4. Deploy a simple web front-end application and connecting it to the deployed model using an IBM Cloud toolchain.
5. Make a coding change and re-build the application using the DevOps toolchain
6. Run the application, input passenger data, and receive the survivability prediction.


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 [Titanic Data](#).
2. Right-click on the window, and click **Save Page As...**



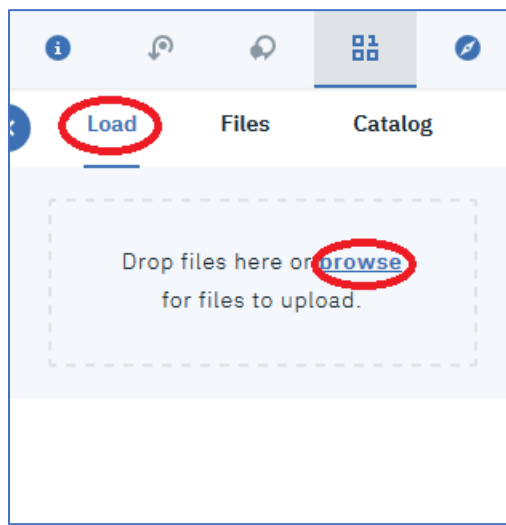
3. Click on **Save**.



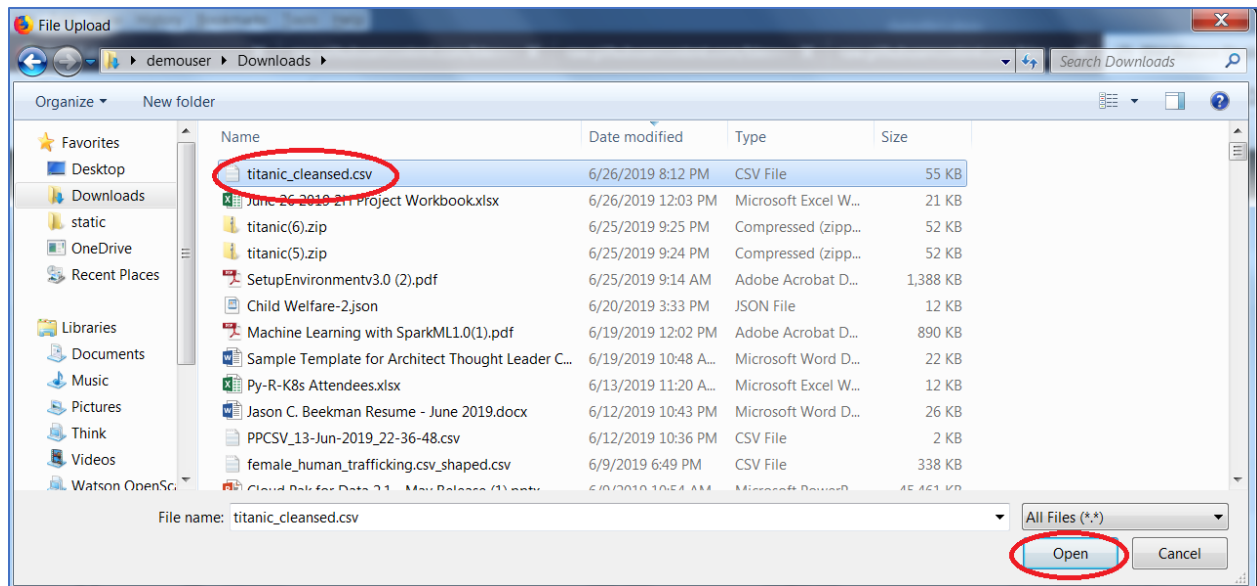
4. Go back to your Watson Studio Labs project. Click on the  icon.



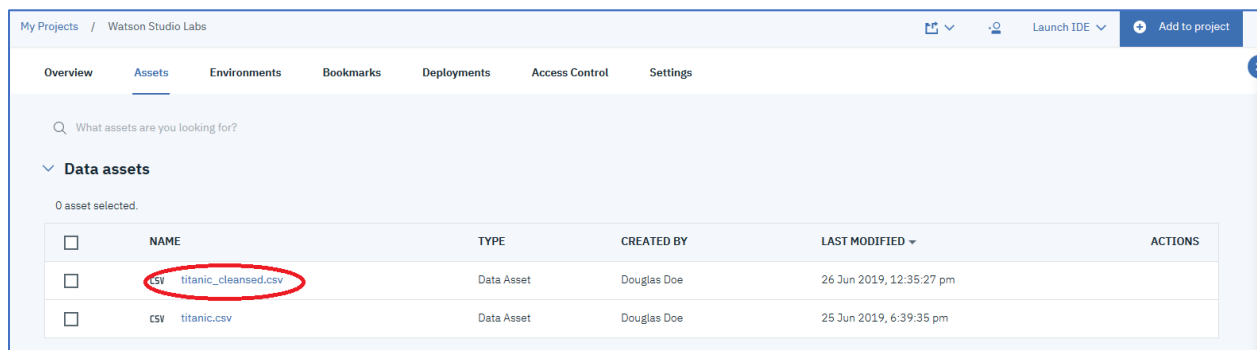
5. Click on the **Load** tab and then click on **browse**. If you don't see the **Load** tab, click on the  icon again.



6. Go to the folder where the titanic_cleansed.csv file is stored. Select the titanic_cleansed.csv file and then click **Open**.

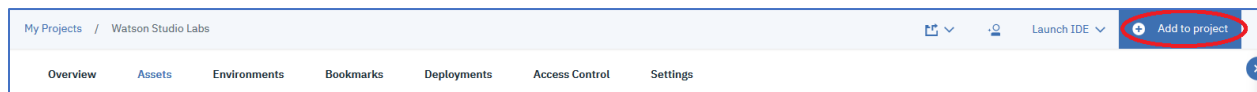


7. The titanic_cleansed.csv file is now added as a Data Asset.

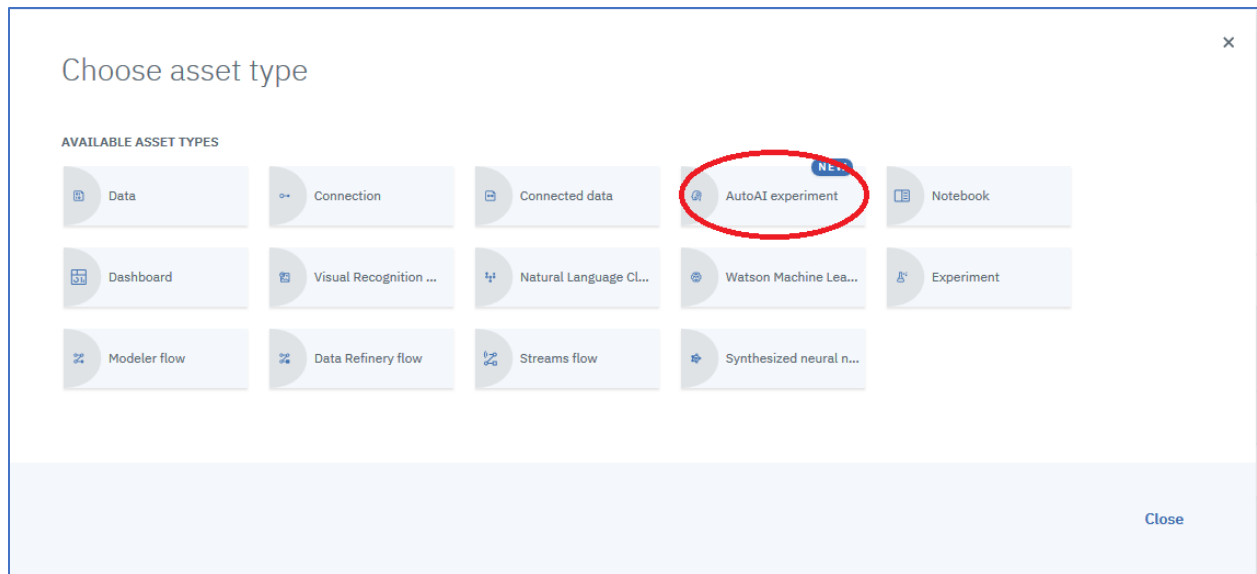


Step 2: Add an AutoAI Experiment

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**.

Create AutoAI experiment

Use this no-code approach to generate a prediction based on data you provide. AutoAI automatically finds the best algorithm for your data and use case. [Learn more](#)

Define AutoAI details

Create AutoAI experiment type

☒ From blank ☐ From sample

Asset name *

Titanic AutoAI

Description

Description of AutoAI experiment

Associated services

Watson Machine Learning Service Instance *

WatsonMachineLearning

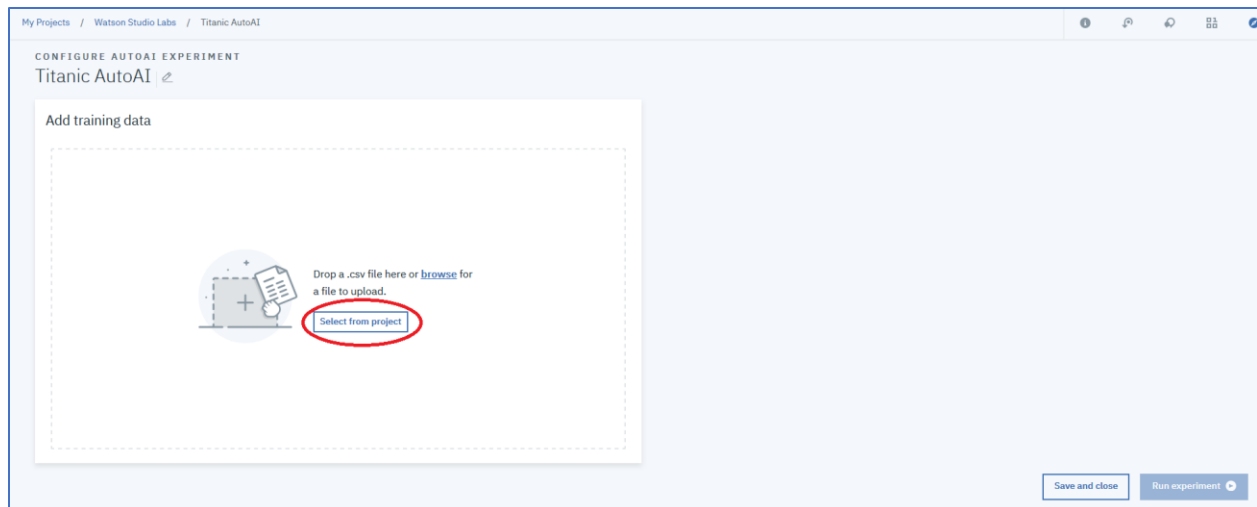
Compute configuration *

8 vCPU and 32 GB RAM

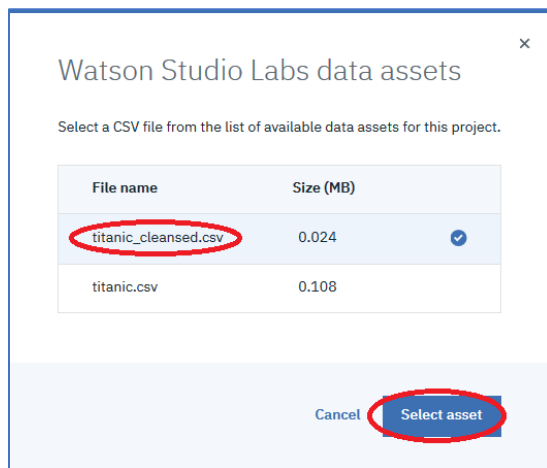
This compute configuration consumes 20 capacity units per hour. [Learn more](#) about capacity unit hours and Watson Machine Learning pricing plans.

Cancel 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, accept the default for **OPTIMIZED METRIC**, and click on **Run experiment**.

Select column to predict


DATA SOURCE: titanic_cleansed.csv

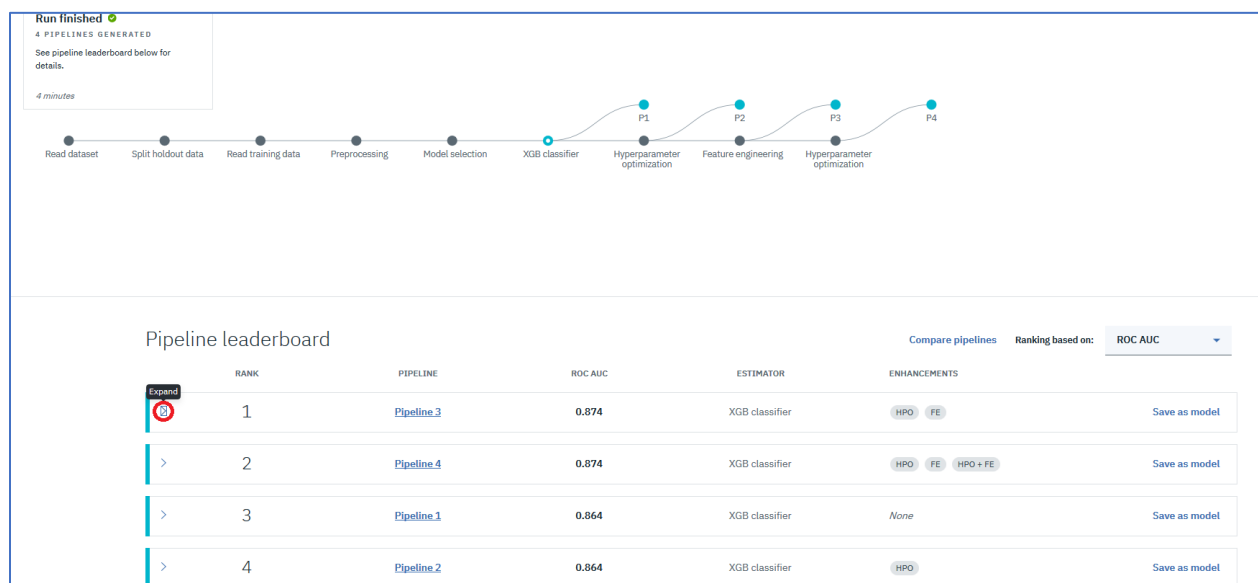
Column name	Type
survived	Integer
sex	String
age	Decimal
sibsp	Integer
parch	Integer
fare	Decimal

Selected prediction Edit prediction

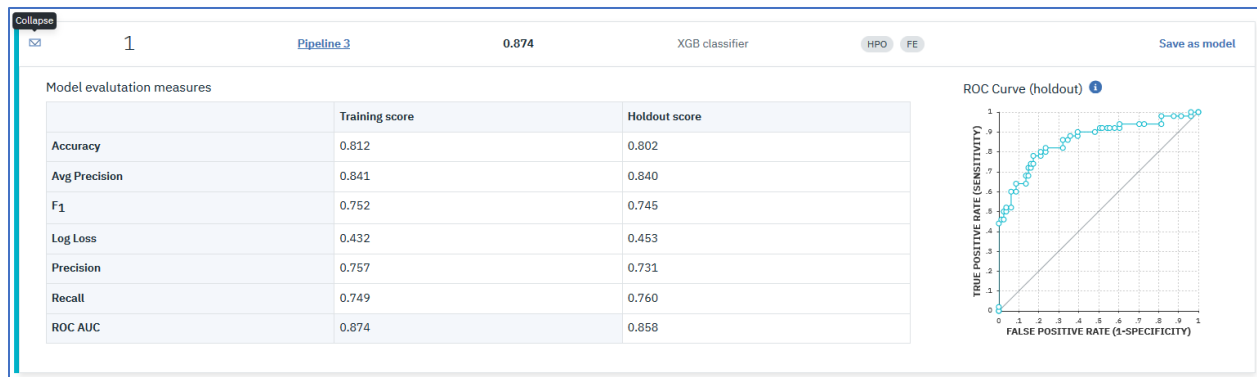
PREDICTION TYPE **Binary Classification** i OPTIMIZED METRIC **ROC AUC** i

Save and close Run experiment

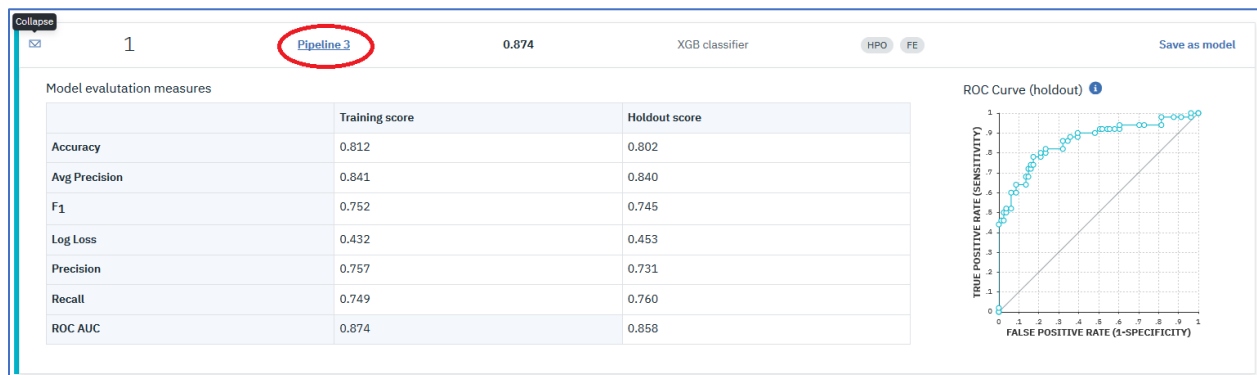
- Please wait several minutes as the alternative solutions are analyzed. The pipeline summary is then displayed. Click on the right arrow  next to the first pipeline (Pipeline 3).



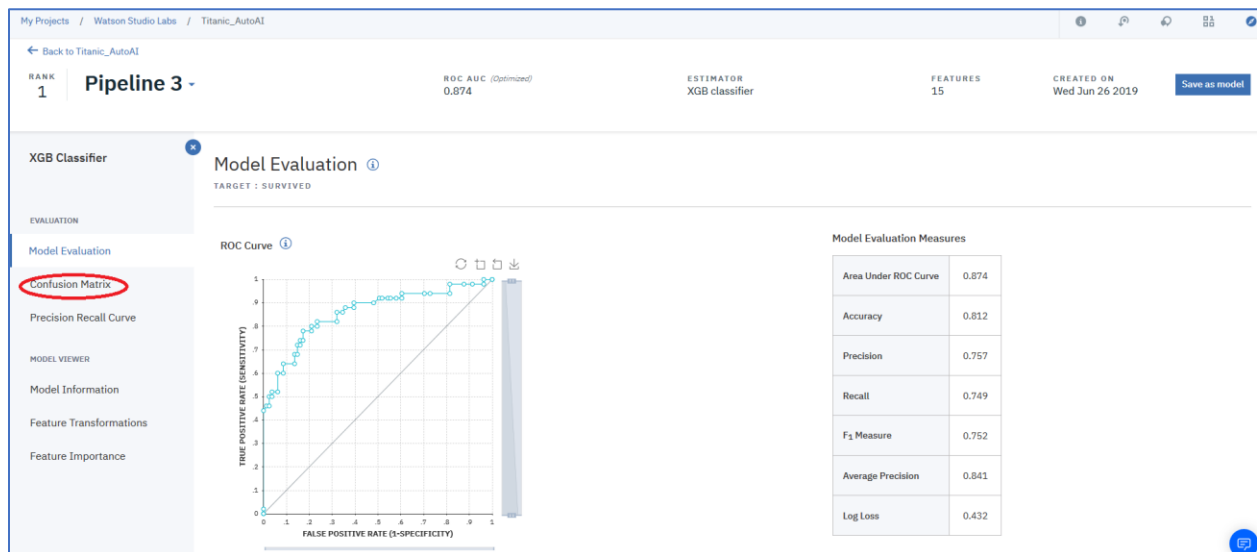
- Scores are displayed for different metrics for both the training sample and the holdout sample.



9. Click on the **Pipeline3** link.



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



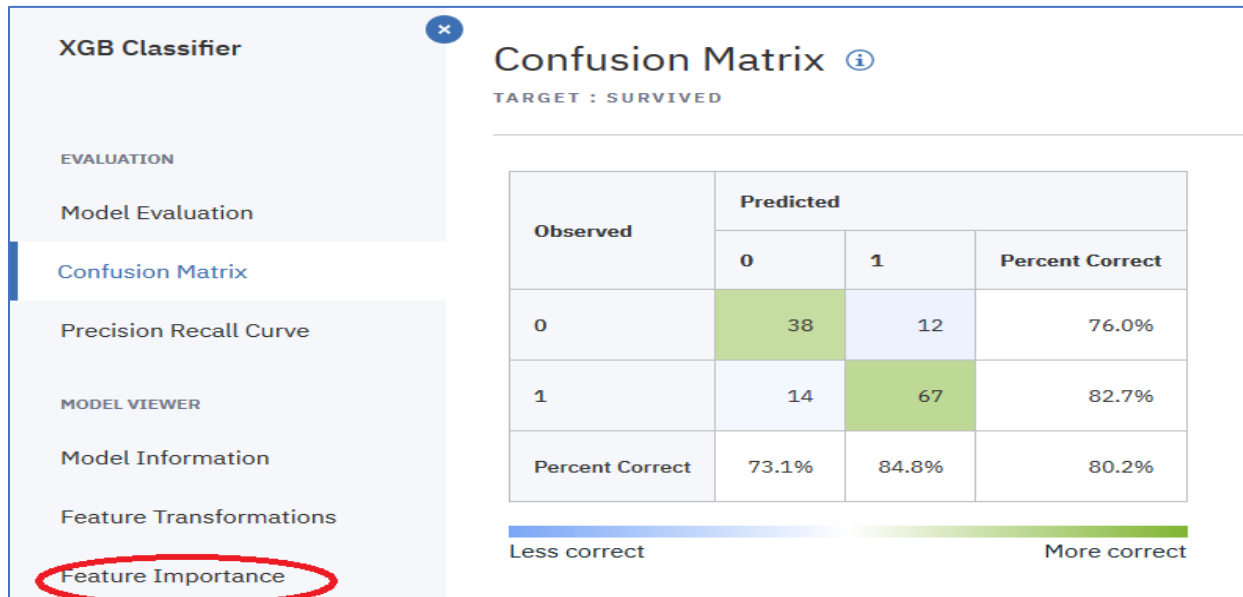
11. The Confusion Matrix is displayed. The different metrics are computed based on the numbers in the Confusion Matrix. For example, Precision is defined by the percentage of predicted positives that are actually positive (i.e. the percentage of predicted survivors that survived). Recall is defined as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

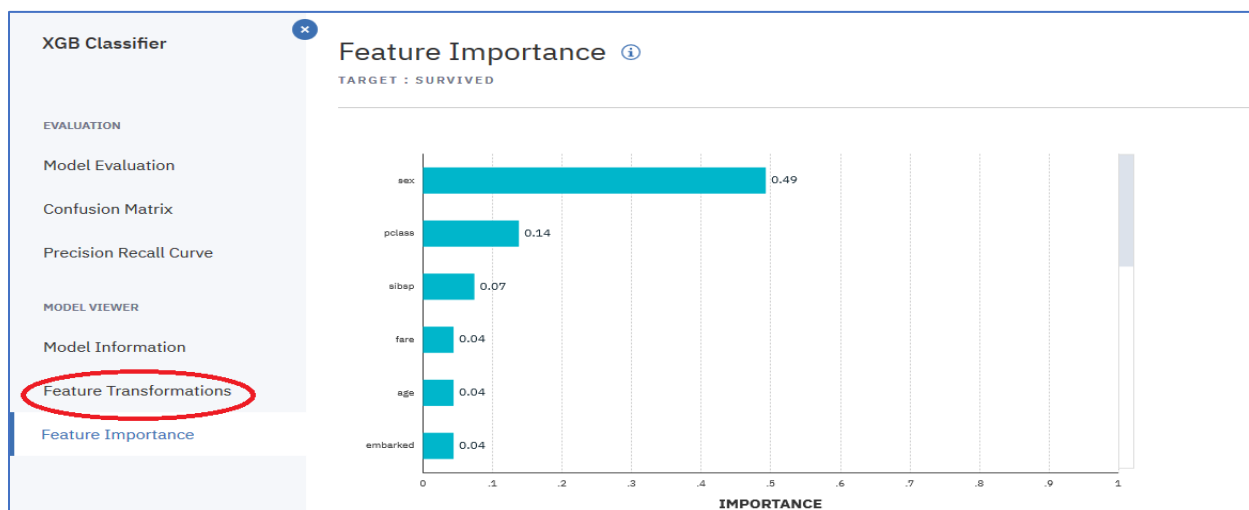
Precision = True Positive/ (True Positive + False Positive)

Recall = True Positive/(True Positive + False Negative)

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



12. According to the Feature Importance, the sex variable is considered the most important feature followed by the passenger class. Click on the **Feature Transformations** option.



13. Feature Transformation applies a transform to features to improve the model performance.

XGB Classifier

EVALUATION

Model Evaluation

Confusion Matrix

Precision Recall Curve

MODEL VIEWER

Model Information

Feature Transformations

Feature Importance

Feature Transformations ⓘ

TARGET : SURVIVED

New Feature	Original Feature	Transformation
NewFeature_0	pclass	tangent(pclass)
NewFeature_1	sex	tangent(sex)
NewFeature_2	sibsp	tangent(sibsp)
NewFeature_3	parch	tangent(parch)
NewFeature_4	fare	tangent(fare)
NewFeature_5	embarked	tangent(embarked)

Step 3 – Save and Deploy the Selected Model

1. Click on **Save as Model**

My Projects / Watson Studio Labs / Titanic_AutoAI

Back to Titanic_AutoAI

RANK 1

Pipeline 3 -

ROC AUC (Optimized) 0.874

ESTIMATOR XGB classifier

FEATURES 15

CREATED ON Wed Jun 26 2019

Save as model

XGB Classifier

Feature Transformations ⓘ

TARGET : SURVIVED

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

×

Save as model

Save this model as a project asset so you can deploy, train, and test it.

Model name

Titanic_AutoAI - P3 XGBClassifierEstimator

Description (optional)

Description of model

Cancel

Save

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

IBM Watson Studio

Upgrade

Douglas Doe's Account

DD

My Projects / Watson Studio Labs / Titanic_AutoAI

← Back to Titanic_AutoAI

RANK		ROC AUC (Optimized)	ESTIMATOR	FEATURES	CREATED
1	Pipeline 3	0.874	XGB classifier	15	Wed 3

XGB Classifier

Feature Transformations

'Titanic_AutoAI - P3 XGBClassifierEstimator' was successfully saved to 'Watson Studio Labs'.

View in project

4. Click on **Deployments**

My Projects / Watson Studio Labs / Titanic_AutoAI - P3 XGBClassifierEs

MODEL

Titanic_AutoAI - P3 XGBClassifierEstimator

Overview

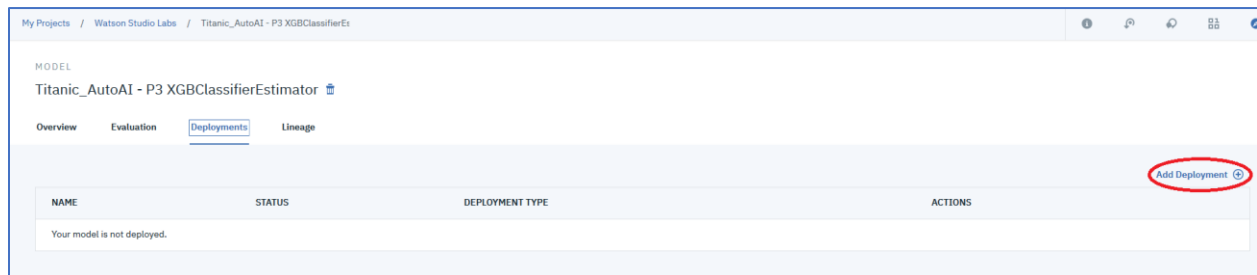
Deployments

Lineage

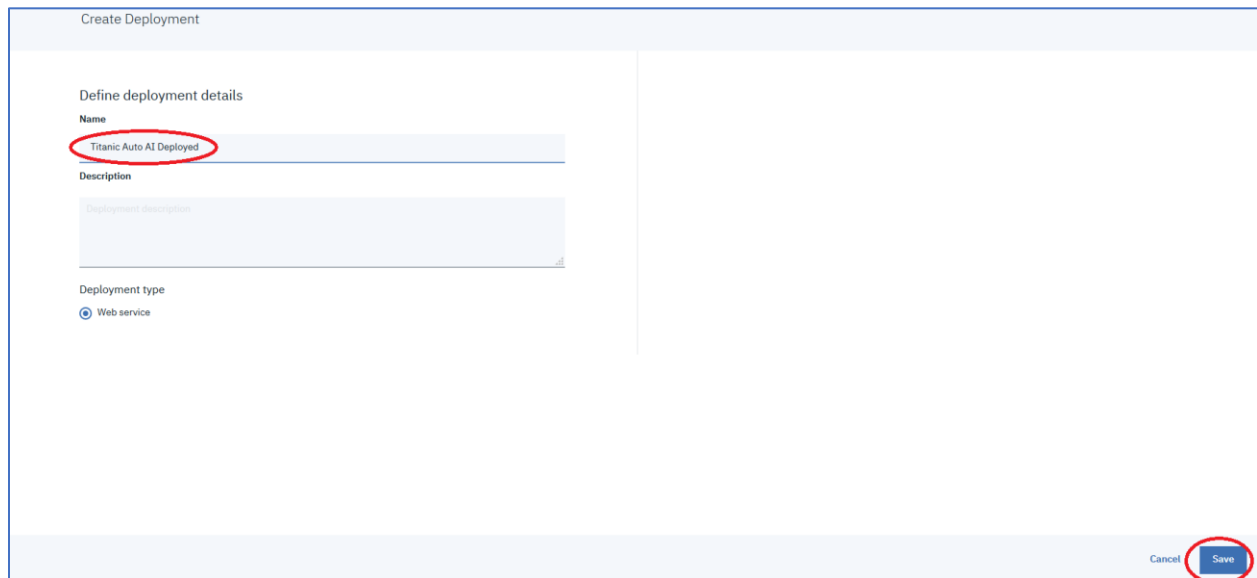
Summary

Machine learning service	WatsonMachineLearning
Model Type	wml-hybrid_0.1
Runtime environment	/v4/runtimes/hybrid_0.1
Training date	27 Jun 2019, 11:14 AM
Latest version	1d0716e7-2f25-4128-a03c-72a965a518d4

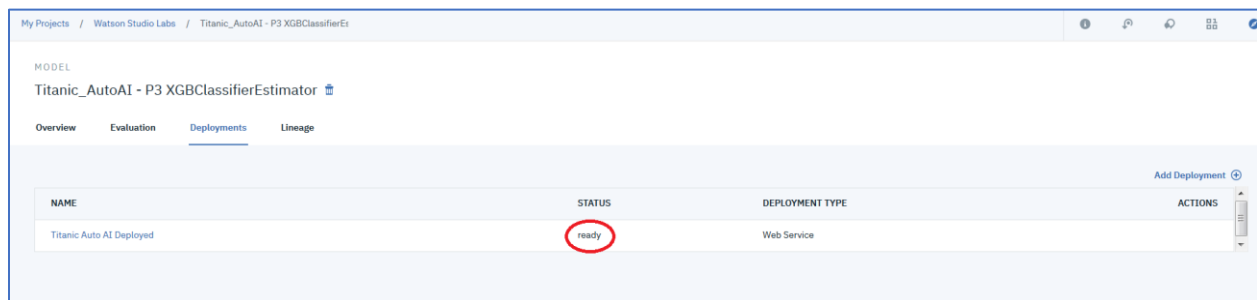
5. Click on **Add Deployment**.



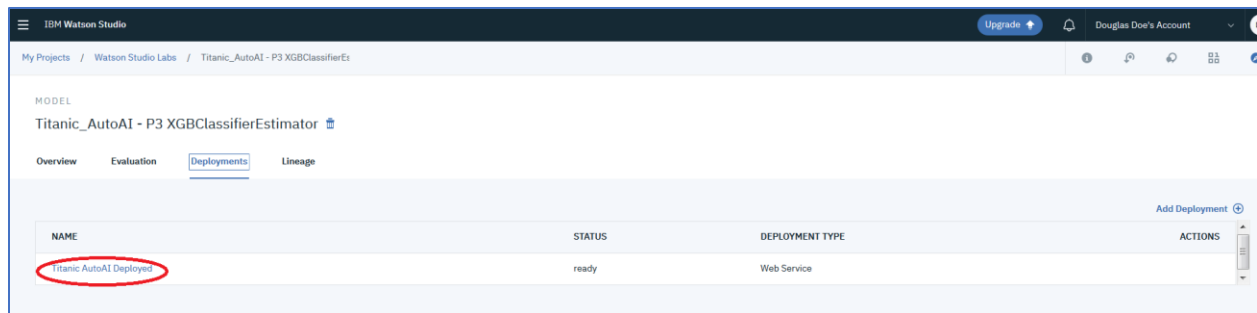
6. Enter a **Name** and click **Save**.



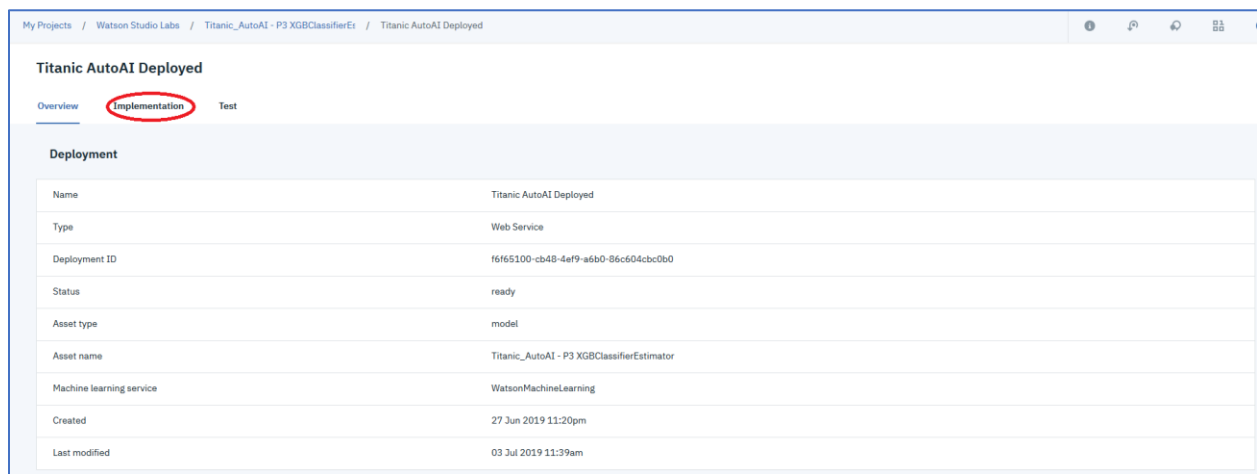
7. The model is successfully deployed on the IBM Cloud.



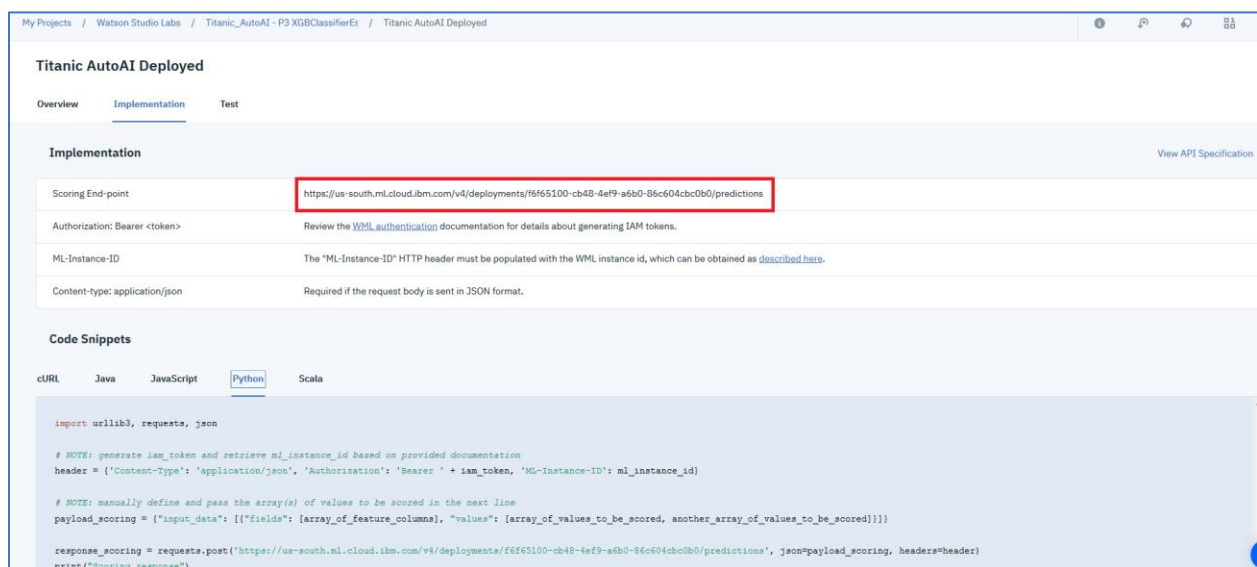
8. Click on **Titanic AutoAI Deployed**.



9. Click on **Implementation**.



10. 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 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 IBM Cloud](#)

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

Toolchains / Create a toolchain /

Deploy to IBM Cloud: TitanicNew app

After you click **Deploy**, your app will be deployed to IBM Cloud.

Your app's code will be automatically loaded into a Git repo. Each time you commit changes to the repo, they are automatically deployed by using a toolchain that is associated with your app. You can add more tools to the toolchain and share it with your team. [Learn more](#).

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

TEMPLATE INFO

GIT URL <https://github.com/open-tool...>

GIT BRANCH

Toolchain Name: TitanicNew-20190627173711359

Select Region: Washington DC

Select a resource group: Default

Tokyo
Frankfurt
London
Dallas
Washington DC

Git Repos and Issue Tracking
Eclipse Orion Web IDE
Delivery Pipeline Required

Deploy

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

Git Repos and Issue Tracking Eclipse Orion Web IDE **Delivery Pipeline** Required

The Delivery Pipeline automates continuous deployment.

App name: TitanicNew-20190627172944935

IBM Cloud API key: IBM Cloud API key **Create +**

The value is required.

Region	Organization	Space
The IBM Cloud CF region	The IBM Cloud CF org	The IBM Cloud CF space
The value is required.	The value is required.	The value is required.

Deploy

- Click on the **Create** button.

Create API key

The following API key will be created to deploy your application from the Delivery Pipeline.

API Key for TitanicNew-20190627172944935

API keys can be managed from **Manage > Security > Platform API keys**.

Cancel **Create**

- 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 **Deploy**.

The screenshot shows the 'Delivery Pipeline' configuration page in IBM Cloud. At the top, there are three tabs: 'Git Repos and Issue Tracking', 'Eclipse Orion Web IDE', and 'Delivery Pipeline'. The 'Delivery Pipeline' tab is selected. Below the tabs, there is a description: 'The Delivery Pipeline automates continuous deployment.' The 'App name' field is filled with 'TitanicNew-20190627172944935'. The 'IBM Cloud API key' field is masked with dots. Below these fields, there are three dropdown menus: 'Region' (set to 'Dallas'), 'Organization' (set to 'wsuser24000@gmail.c...'), and 'Space' (set to 'dev'). At the bottom right, there is a blue 'Deploy' button.

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

The screenshot shows the 'Toolchains' overview page for the app 'TitanicNew-20190627173937552'. The page has a sidebar with 'Overview', 'Connections', and 'Manage'. The main content area shows a status message: 'Your app is being created! Quick start: To watch the pipeline deploy your app, click **Delivery Pipeline**. After the app is deployed, you can see it running by clicking **View app**.' Below this, there are four tool cards: 'Issues', 'Git', 'Delivery Pipeline', and 'Eclipse Orion Web IDE'. Each card has a checkmark and the word 'Configured'. The 'Delivery Pipeline' card is highlighted with a red circle. At the bottom right, there is a blue 'Add a Tool' button.

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**



Toolchains / TitanicNew-20190627173937552 / TitanicNew-20190627173937552

TitanicNew-20190627173937552 | Delivery Pipeline

The screenshot shows the 'Delivery Pipeline' for 'TitanicNew-20190627173937552'. It displays two stages: 'Build Stage' and 'Deploy Stage'. Both stages are marked 'STAGE PASSED'. The 'Build Stage' shows a 'Build' job that passed 4m ago. The 'Deploy Stage' shows a 'Deploy' job that passed 3m ago, which is circled in red. Below the 'Deploy' job, under 'LAST EXECUTION RESULT', there is a link 'view console' which is also circled in red. A blue button 'Add Stage' is visible on the right.

8. Click on **Visit App URL**

Resource list /

 TitanicNew-20190627173937...  This app is awake. [Visit App URL](#)

Org: wsuser24000@gmail.com Location: Dallas Space: dev

The screenshot shows the 'Resource list' for the app 'TitanicNew-20190627173937...'. It features a Python logo, the app name, and a green status icon with the text 'This app is awake.'. A link 'Visit App URL' is circled in red. Below the app name, it shows 'Org: wsuser24000@gmail.com', 'Location: Dallas', and 'Space: dev'.

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:

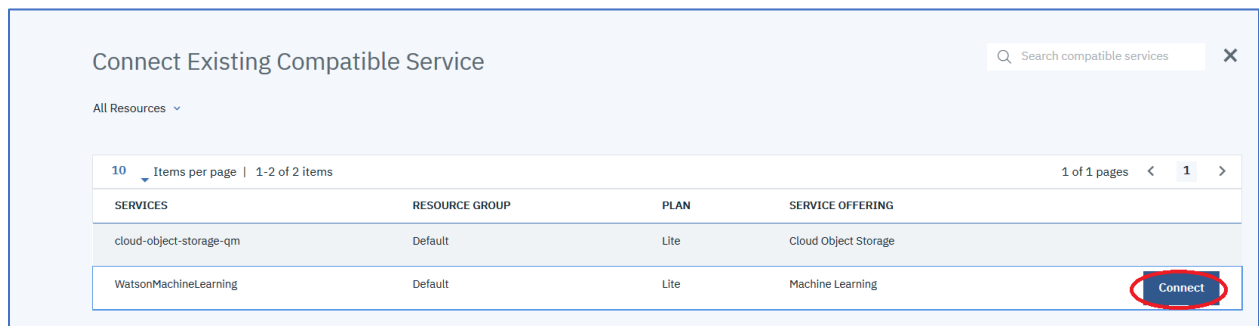
10. We are now going to connect the application to the Watson Machine Learning service that was created earlier. Scroll down until you see the Connections panel. Click on **Create Connection**.

The screenshot shows the IBM Cloud dashboard for an application named "TitanicNew-20190627173937...". The interface includes a left sidebar with navigation links: Getting started, Overview, Runtime, Connections, Logs, Autoscaling (Beta), Monitoring, and API Management. The main content area displays the following information:

- Runtime**
 - BUILDPACK**: Python
 - INSTANCES**: 1 (All instances are running, Health is 100%)
 - MB MEMORY PER INSTANCE**: 128
 - TOTAL MB ALLOCATION**: 128 (128 MB still available)
- Connections**: No services are connected to this app. You can bind a service: [Create connection](#) (highlighted with a red circle).
- Runtime cost**
 - \$0.00**: Current charges for billing period.
 - \$0.00**: Estimated total for billing period (Jun 1, 2019 - Jun 30, 2019).

At the bottom, a note states: "Current and estimated cost excludes [connected services](#)."

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**.



Connect Existing Compatible Service

Search compatible services

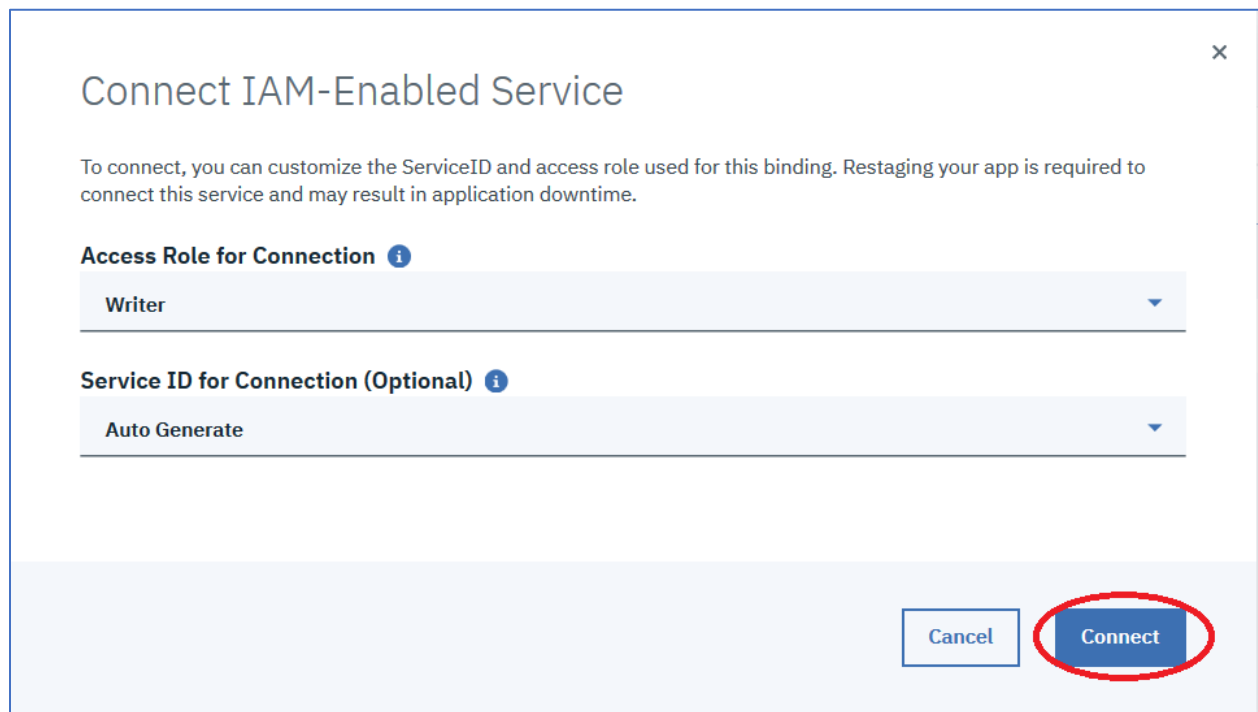
All Resources

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

SERVICES	RESOURCE GROUP	PLAN	SERVICE OFFERING
cloud-object-storage-qm	Default	Lite	Cloud Object Storage
WatsonMachineLearning	Default	Lite	Machine Learning

Connect

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



Connect IAM-Enabled Service

To connect, you can customize the ServiceID and access role used for this binding. Restaging your app is required to connect this service and may result in application downtime.

Access Role for Connection ⓘ

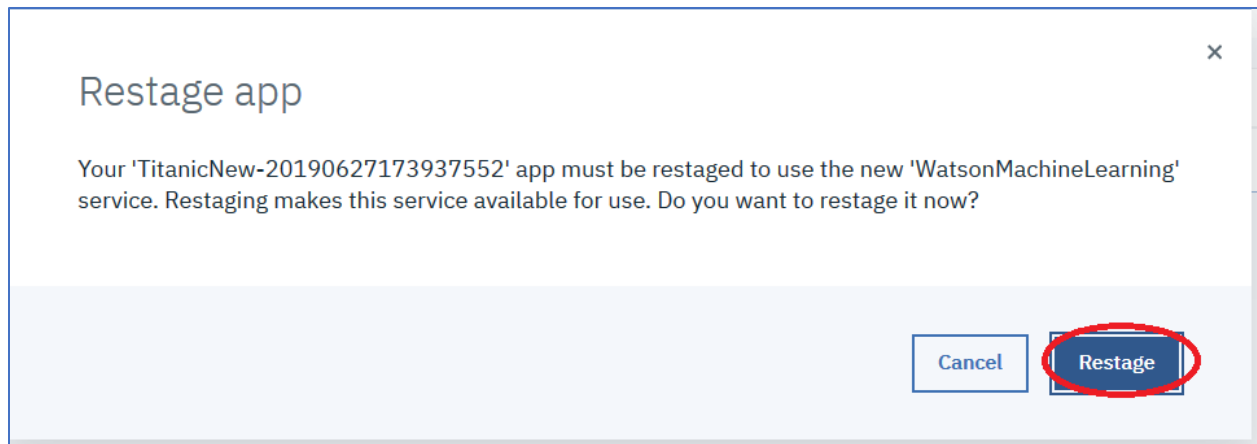
Writer

Service ID for Connection (Optional) ⓘ

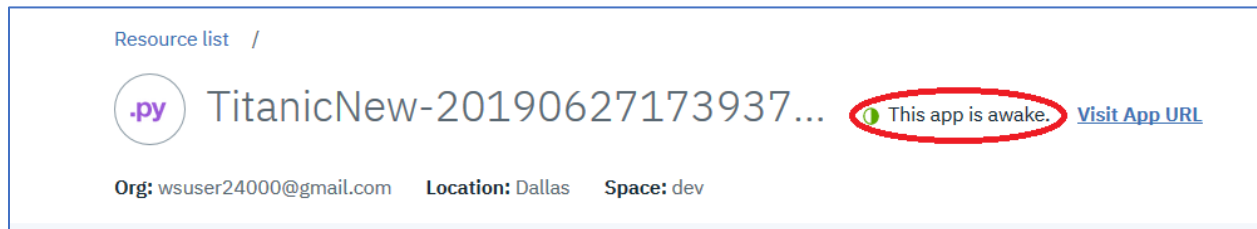
Auto Generate


Cancel Connect

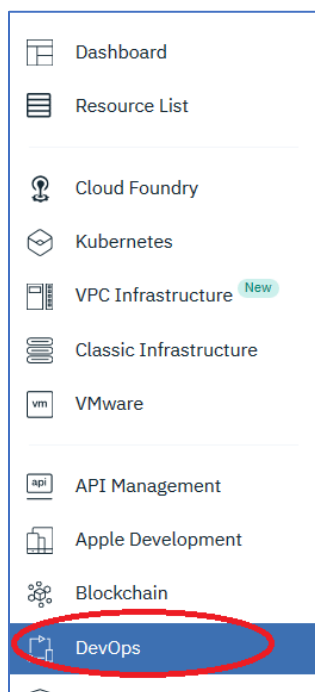
13. A **Restage app** pop up will appear. Click on **Restage**.



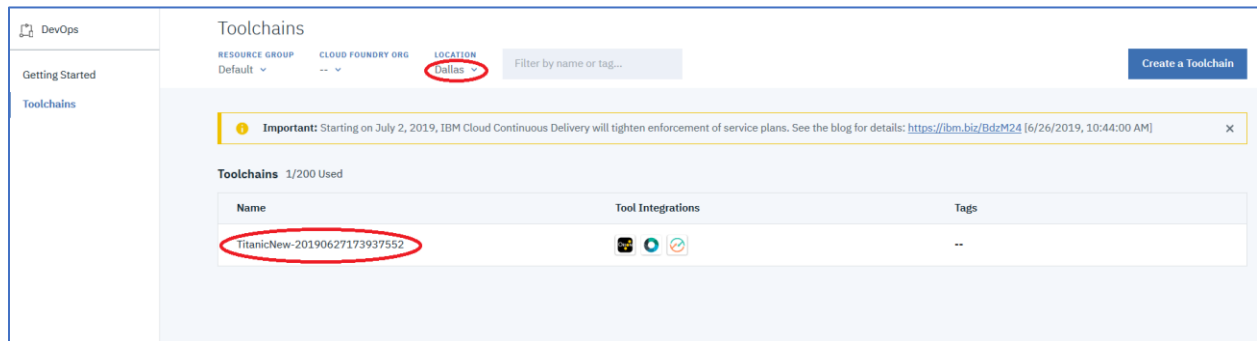
14. Wait for the application status to change to  This app is awake , or something similar.



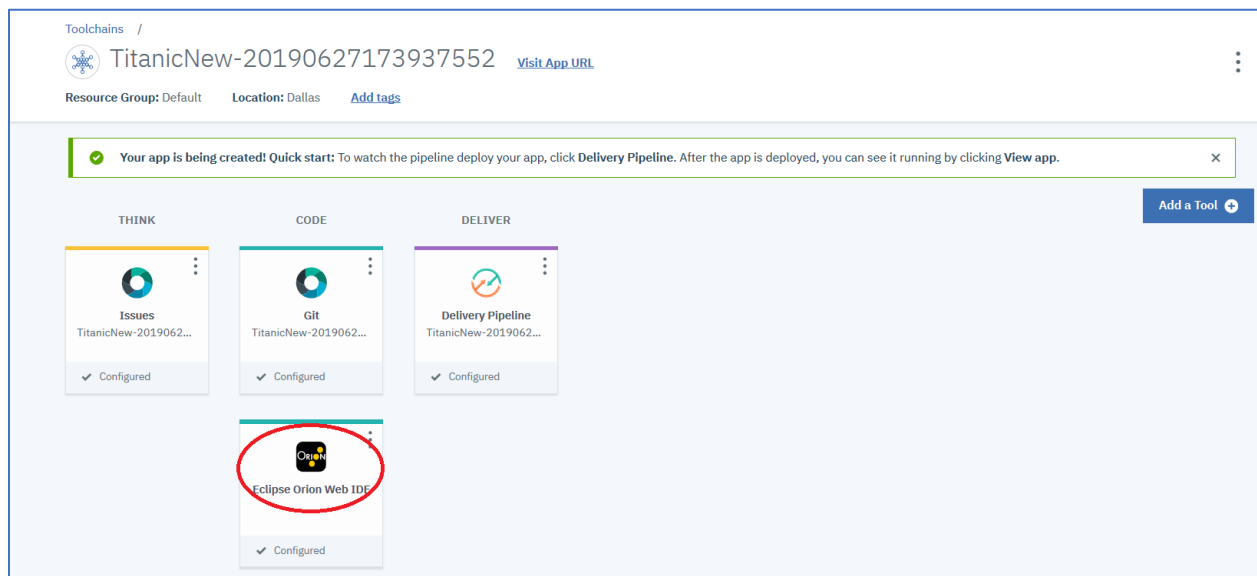
15. We now have tied the web application to the Watson Machine Learning service. Note that the Watson Machine Learning service could have more than one deployed model available to select and then embed in the web application. Click on the  icon and click on DevOps in the pulldown to navigate to the Toolchain.



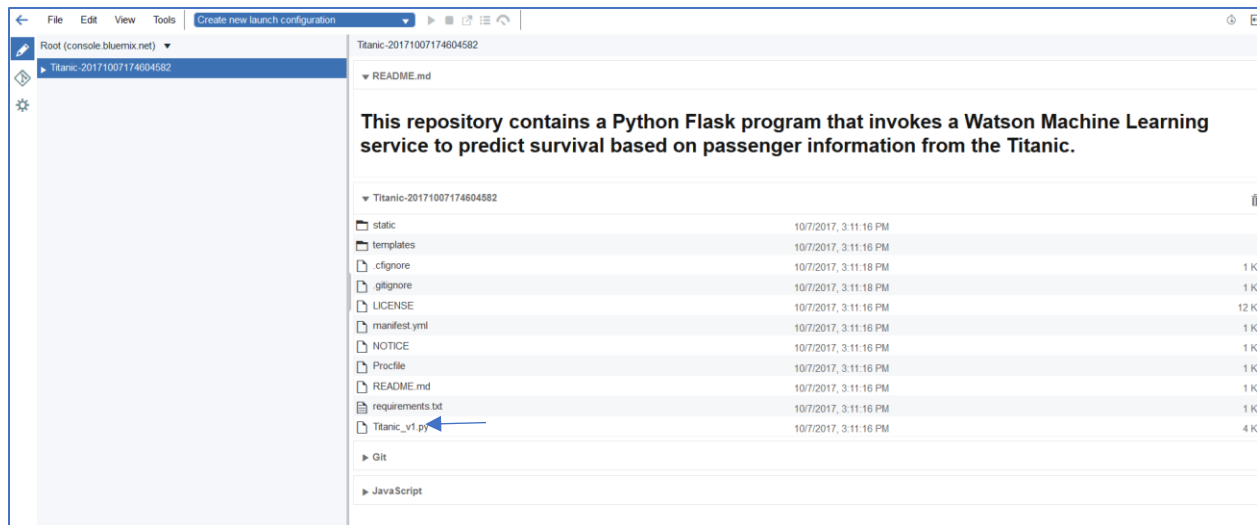
16. We are now going to paste the scoring endpoint into the application code. Click on the Toolchain (Titanic-2019xxxxx below). If no toolchain appears, switch the region to Dallas.




17. Click on the Eclipse Orion Web IDE.

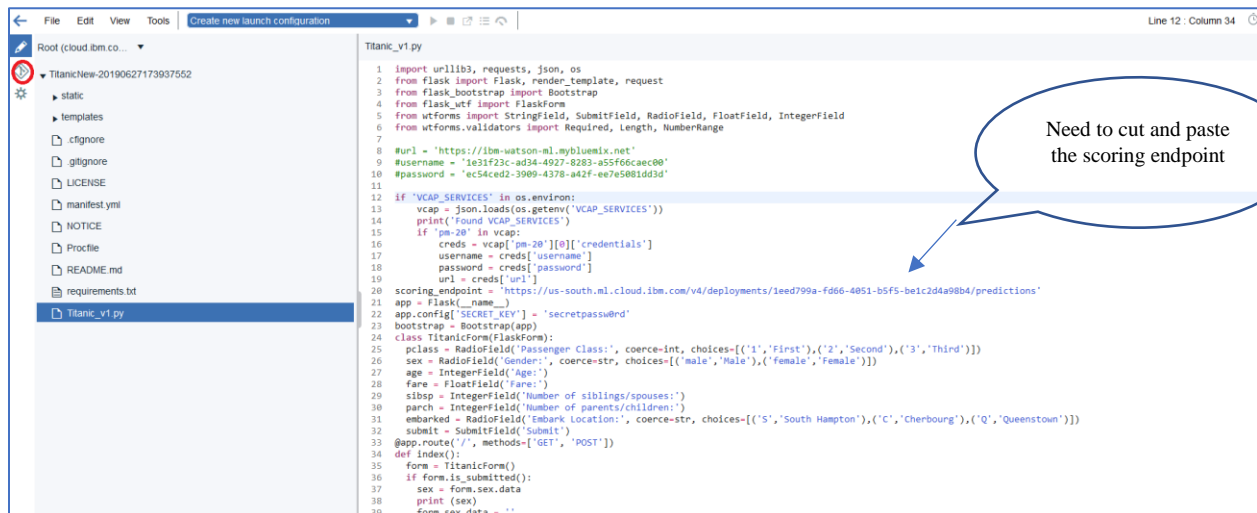


18. Click on the Titanic_v1.py file. This is a python source file.

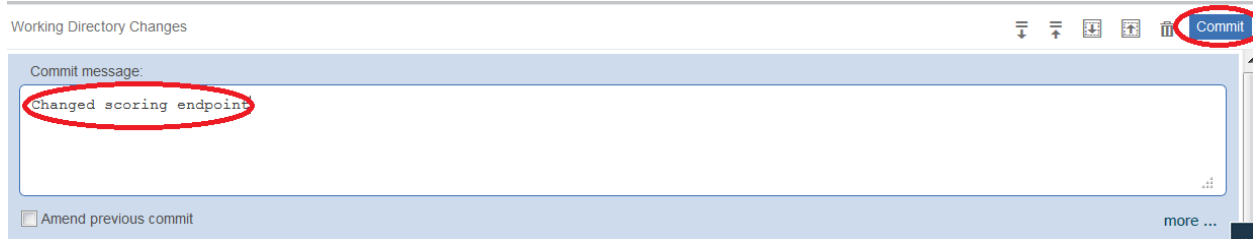


19. Look around line 20 in the Titanic_v1.py file for the “scoring endpoint =”. Select the scoring endpoint value in line 20 (starting with https:// you may want to use Shift-End to get to the end of the line, and then back up one space to not select the endpoint quote – if you do just make sure to put it back in). Copy the scoring endpoint from the Watson Studio Implementation panel and use Ctrl-V to paste the scoring endpoint into the

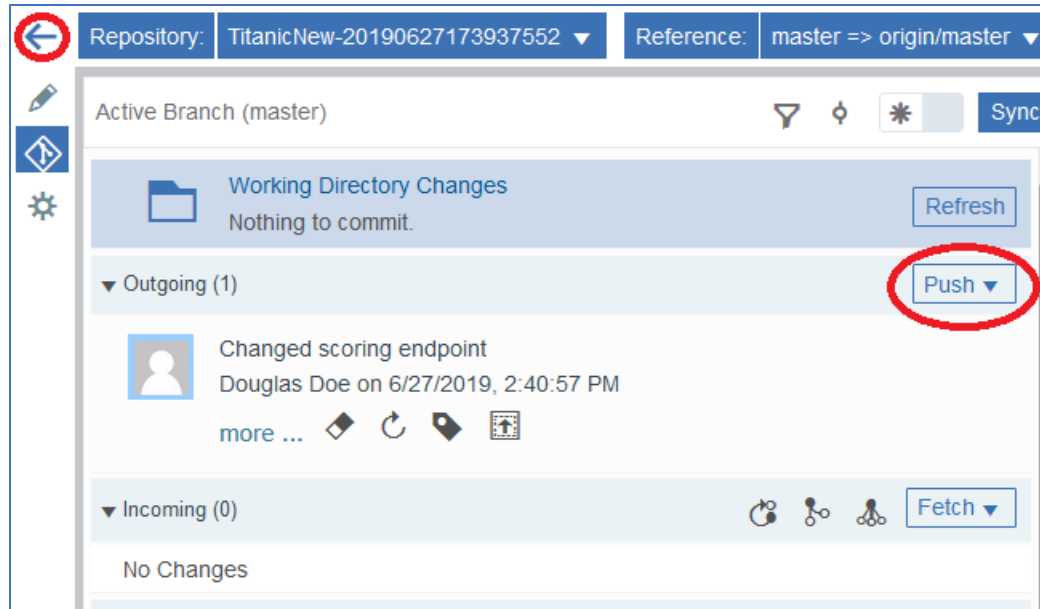
Titanic_v1.py file. Enter Ctrl-S or File > Save to save the file. Then click on the  icon on the top left.



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.

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:

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.

Titanic Prediction

prediction: did not survive
probability: 0.554811477661

[Try Again!](#)

Congratulation! You have completed Lab-5 !!!

✓ 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.