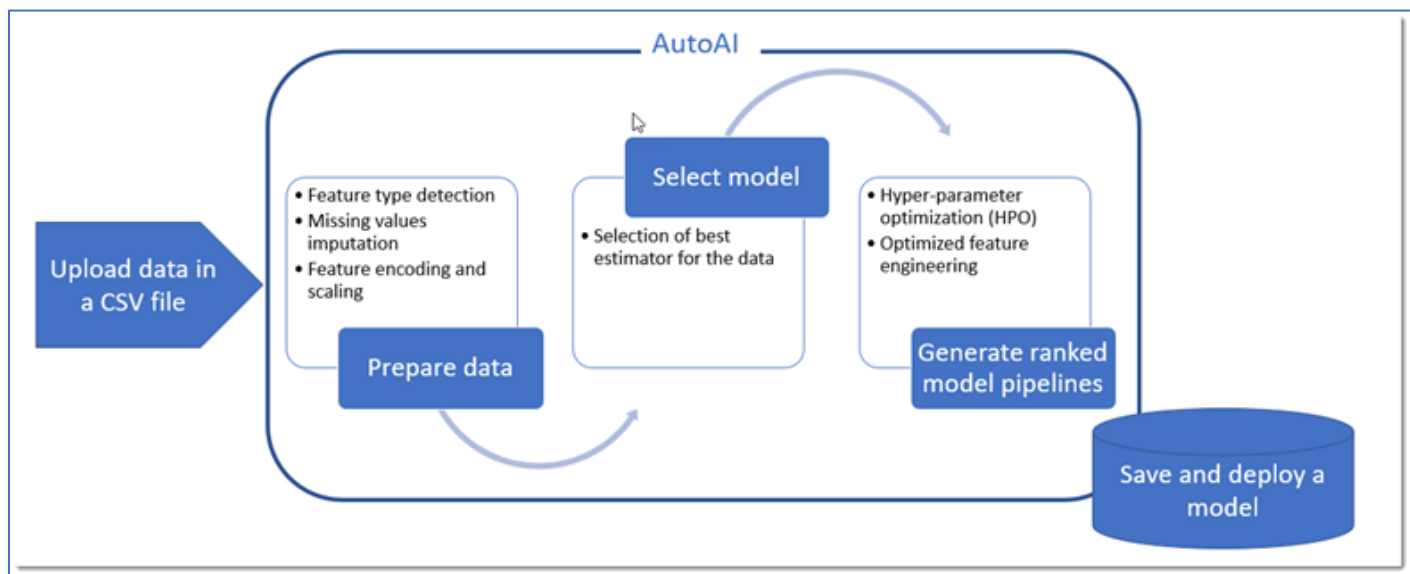


AutoAI Lab

This lab will demonstrate the new and exciting AutoAI capability to build and deploy an optimized model based on the Titanic data set.

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:

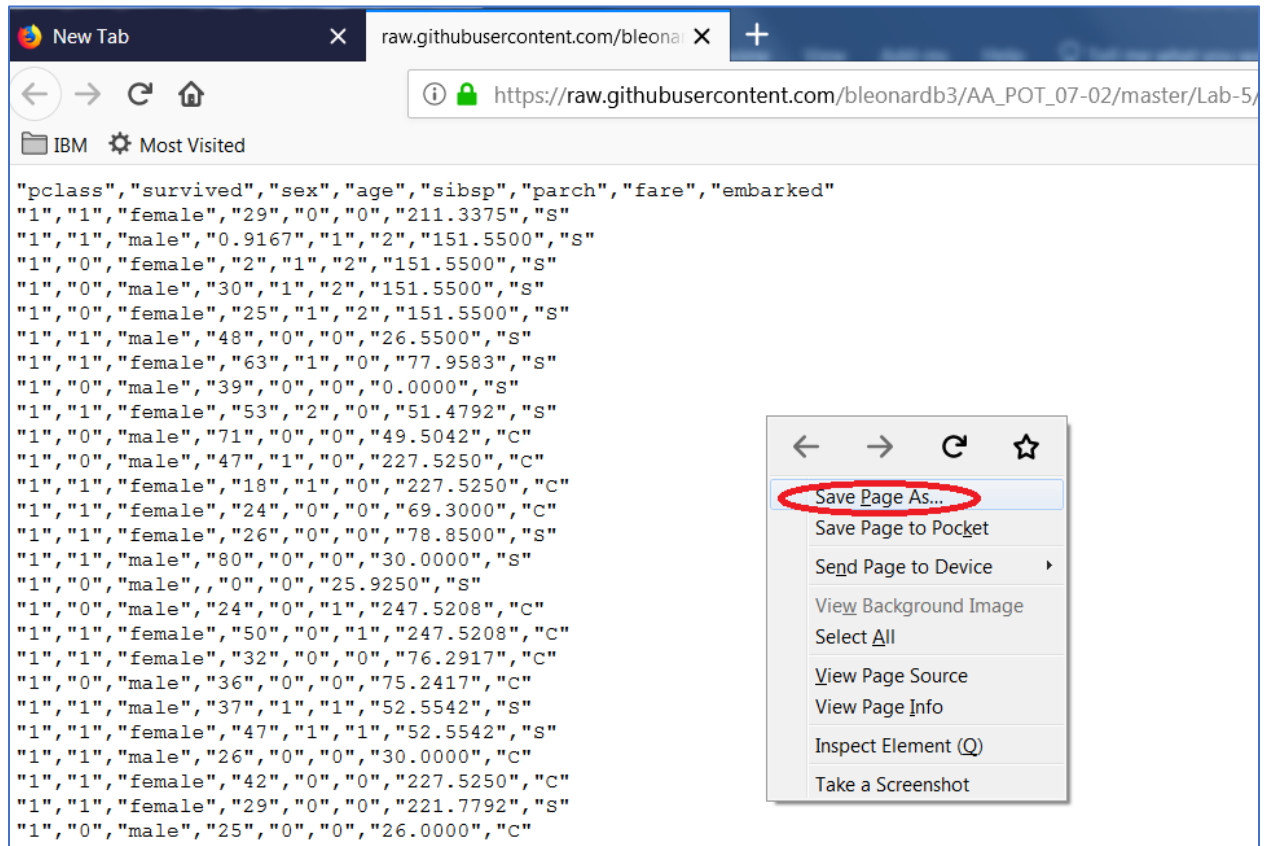
- [Data pre-processing](#)
- [Automated model selection](#)
- [Automated feature engineering](#)
- [Hyperparameter optimization](#)

We will perform the following steps in this lab:

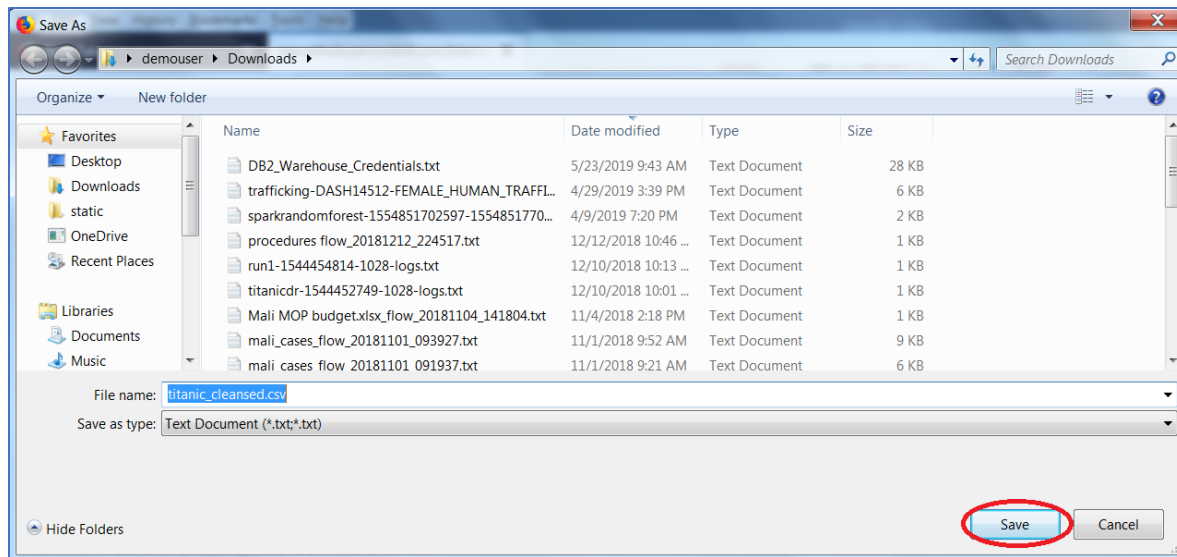
1. Download a Titanic cleansed data set
2. Add an Auto AI Experiment
3. Save and Deploy the selected model.


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...**




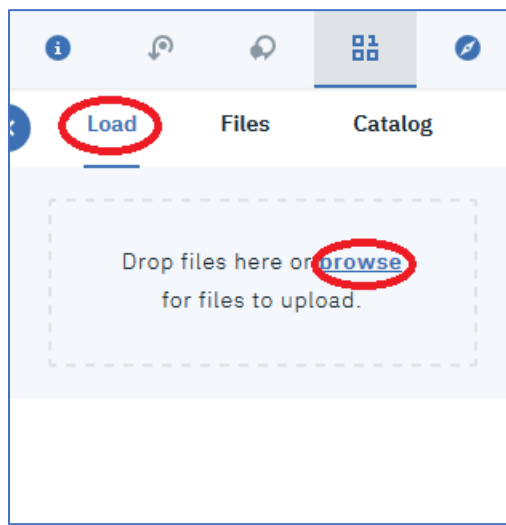
3. Click on **Save**. Note, if the file is named titanic_cleansed.csv.txt, change it to be titanic_cleansed.csv.



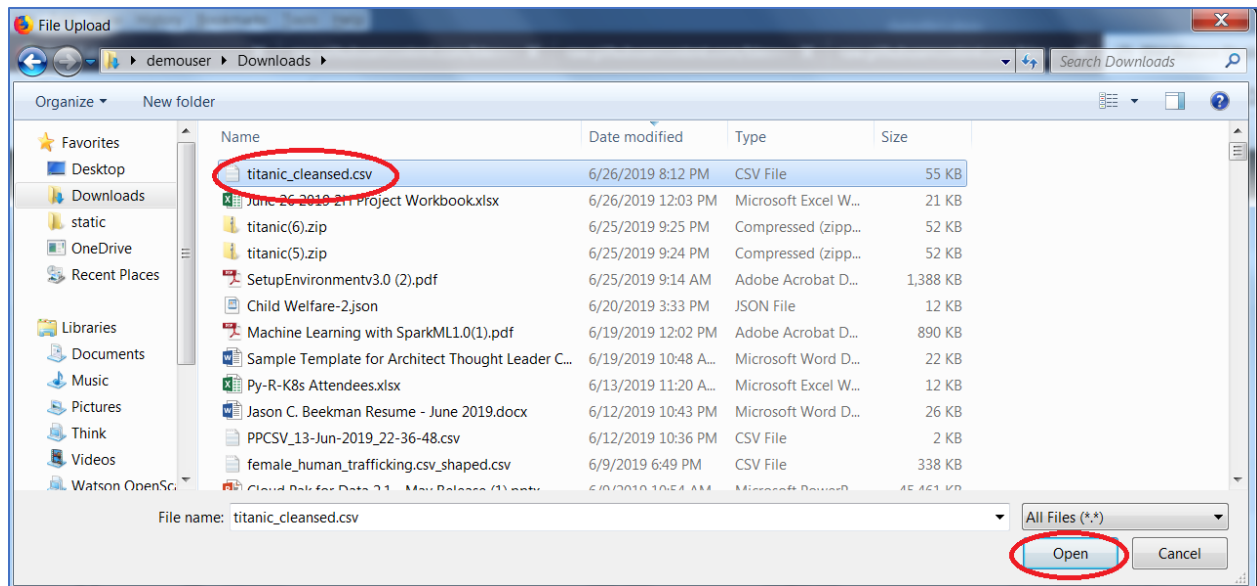
4. Go back to your Watson Studio Labs project. Click on the  icon.



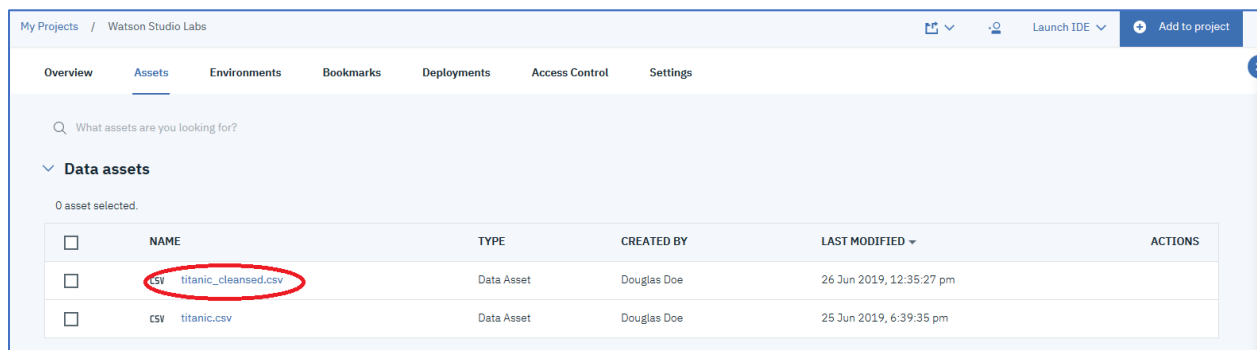
5. Click on the **Load** tab and then click on **browse**. If you don't see the **Load** tab, click on the  icon again.



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

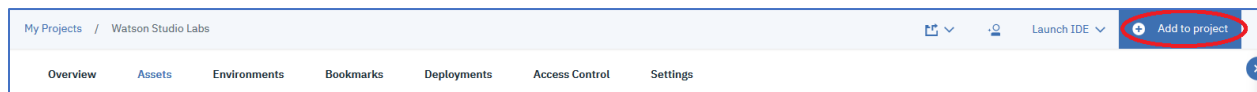


7. The titanic_cleansed.csv file is now added as a Data Asset.

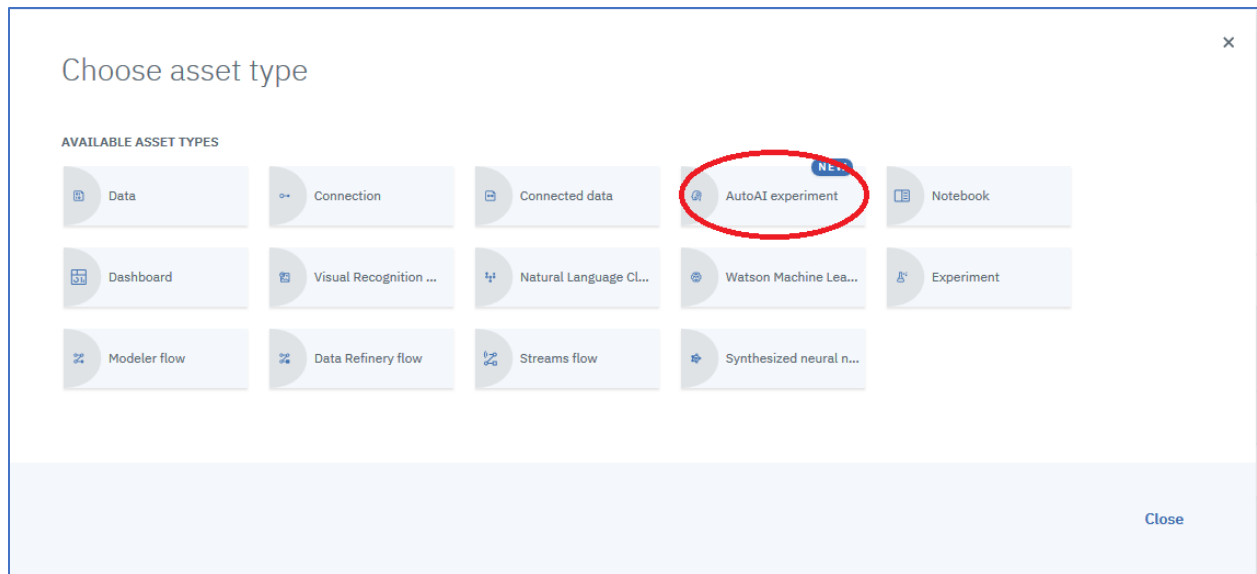


Step 2: Add an AutoAI Experiment

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**.

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

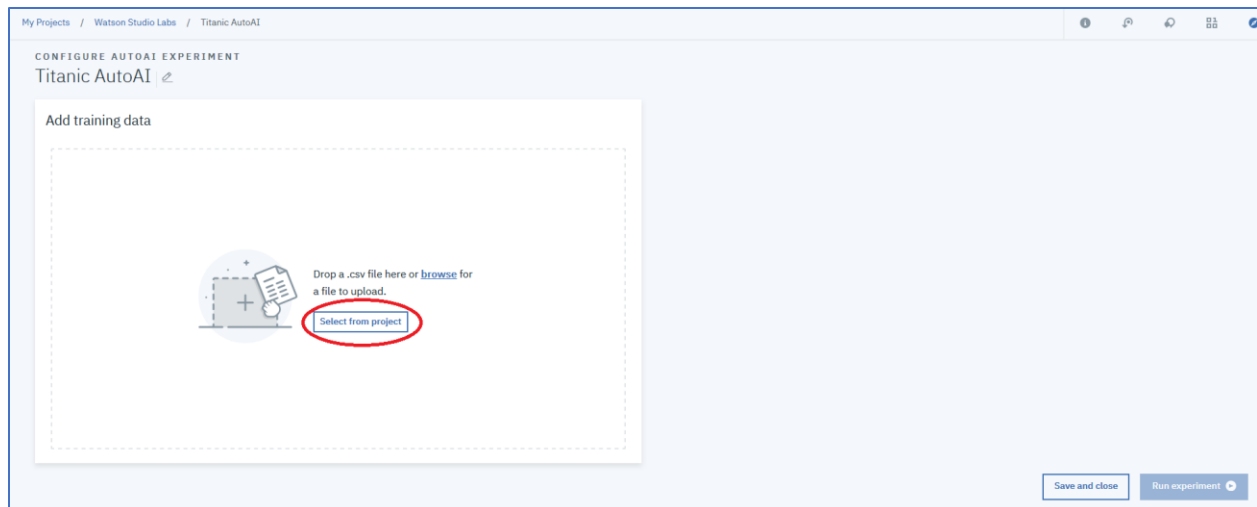
Compute configuration *

8 vCPU and 32 GB RAM

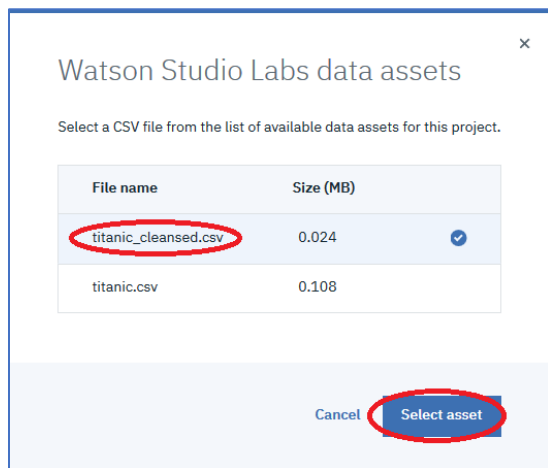
This compute configuration consumes 20 capacity units per hour. [Learn more](#) about capacity unit hours and Watson Machine Learning pricing plans.

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. 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

Binary Classification

POSITIVE CLASS

1

OPTIMIZED METRIC

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**.

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

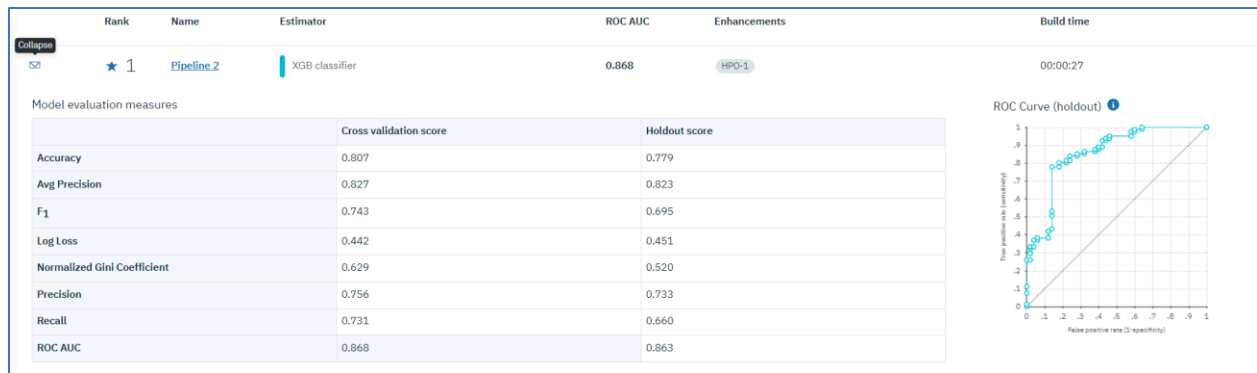
Run experiment

- 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.

Pipeline diagram showing steps: Read dataset, Split holdout data, Read training data, Preprocessing, Model selection, XGB classifier, Hyperparameter optimization (P1), Feature engineering (P2), Hyperparameter optimization (P3), and P4.

Rank	Name	Estimator	ROC AUC	Enhancements	Build time
	★ 1 Pipeline 2	XGB classifier	0.868	HPO-1	00:00:27
>	2 Pipeline 4	XGB classifier	0.866	HPO-1 FE HPO-2	00:02:26
>	3 Pipeline 3	XGB classifier	0.866	HPO-1 FE	00:02:17
>	4 Pipeline 1	XGB classifier	0.863	None	00:00:01

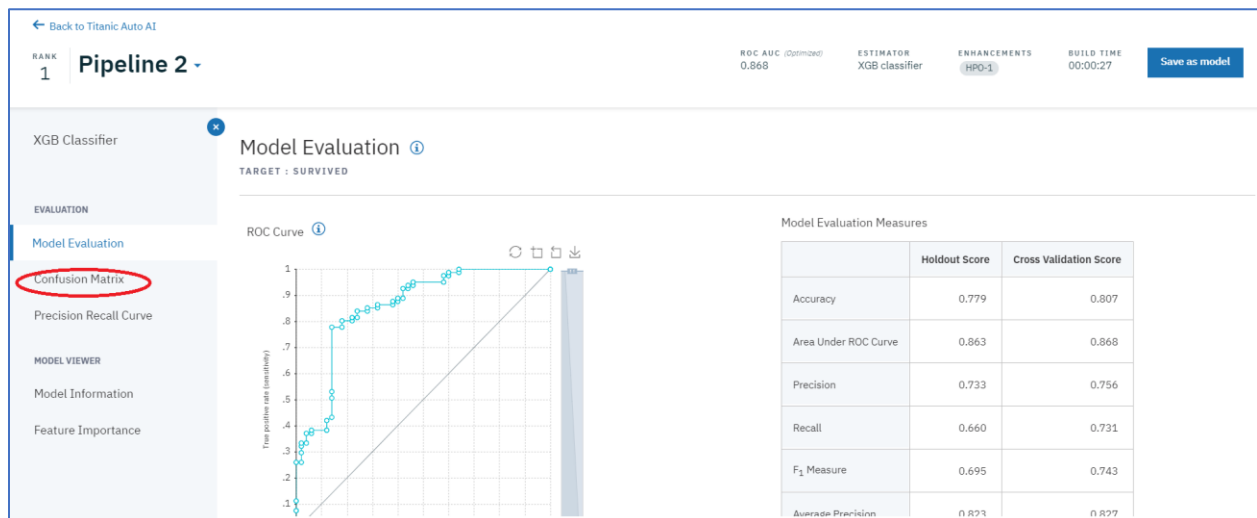
- Scores are displayed for different metrics for both the training sample and the holdout sample.



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



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



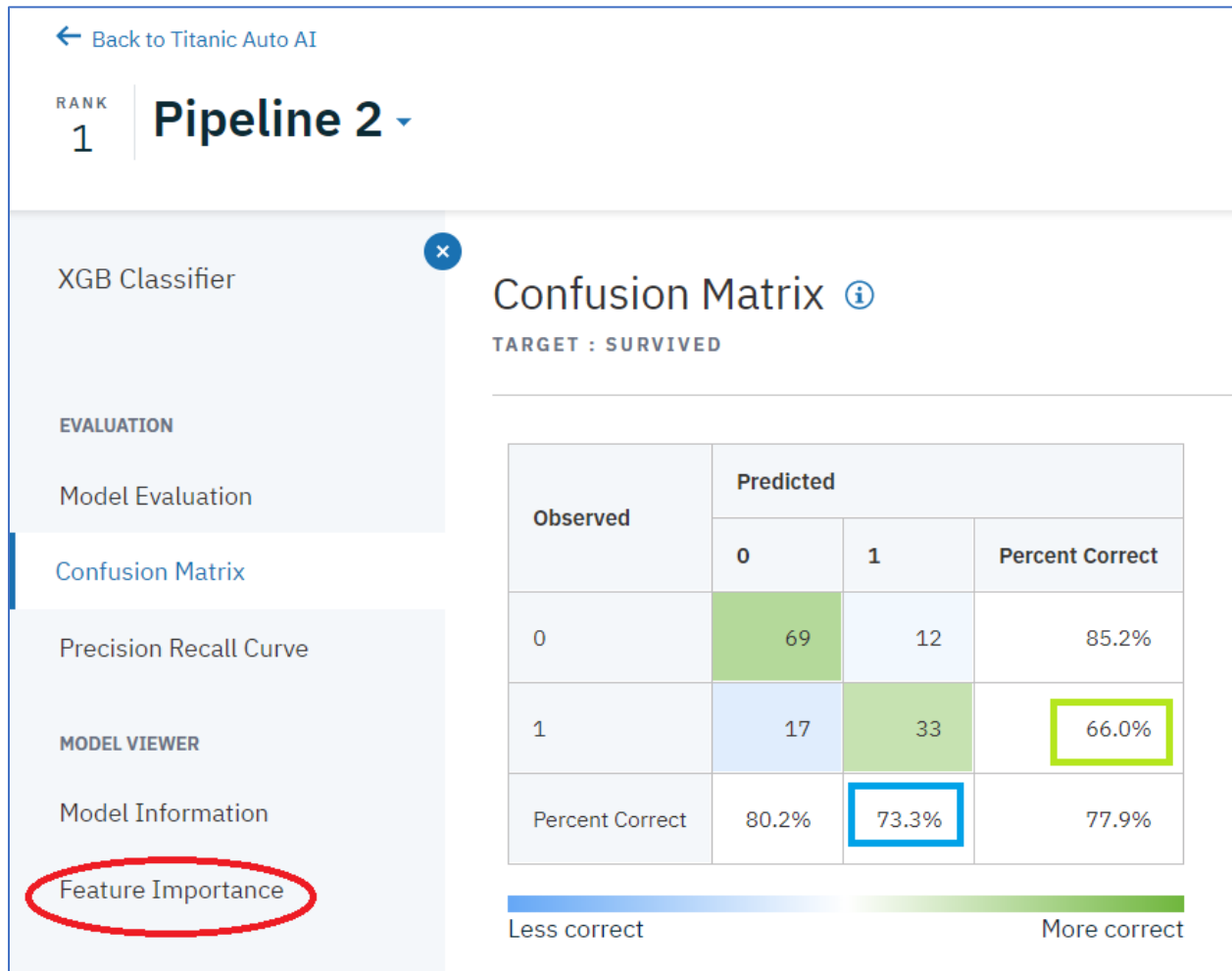
12. The Confusion Matrix is displayed for the holdout sample. The different metrics are computed based on the numbers in the Confusion Matrix. For example, Precision is defined by the percentage of predicted positives that are actually positive (i.e. the percentage of predicted survivors that survived).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

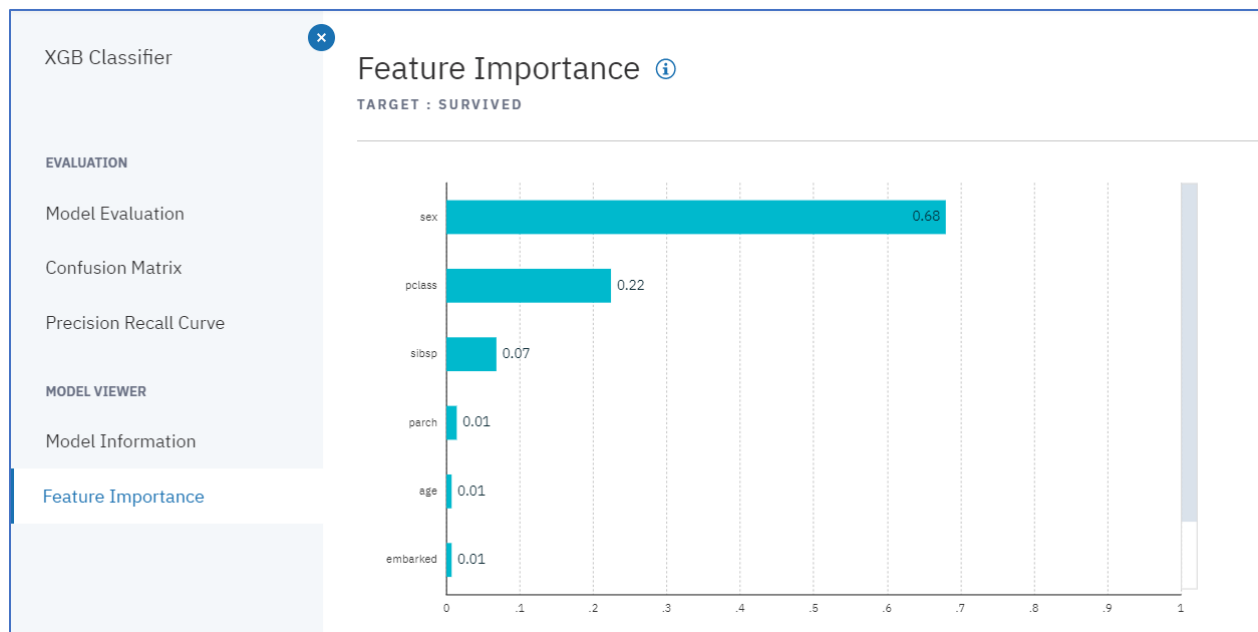
Precision = True Positive/ (True Positive + False Positive) – shown inside blue rectangle on diagram below.

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.



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**.

×

Save as model

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

Model name

Titanic Auto AI - P2 XGBClassifierEstimator

Description (optional)

Description of model

Associated project

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 XGBClassifierEst

MODEL
Titanic_AutoAI - P3 XGBClassifierEstimator

Overview Evaluation **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**.

My Projects / Watson Studio Labs / Titanic_AutoAI - P3 XGBClassifierEst

MODEL
Titanic_AutoAI - P3 XGBClassifierEstimator

Overview Evaluation **Deployments** Lineage

Add Deployment (+)

NAME	STATUS	DEPLOYMENT TYPE	ACTIONS
Your model is not deployed.			

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

Create Deployment

Define deployment details

Name
Titanic Auto AI Deployed

Description
Deployment description

Deployment type
☒ Web service

Cancel **Save**

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

NAME	STATUS	DEPLOYMENT TYPE	ACTIONS
Titanic Auto AI Deployed	ready	Web Service	

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

NAME	STATUS	DEPLOYMENT TYPE	ACTIONS
Titanic AutoAI Deployed	ready	Web Service	

9. Click on **Test**.

Deployment	
Name	Titanic AutoAI Deployed
Type	Web Service
Deployment ID	9e80b93b-dd52-4d00-a097-637888488180
Status	ready
Asset type	model
Asset name	Titanic_AutoAI - P3 XGBClassifierEstimator
Machine learning service	WatsonMachineLearning
Created	20 Jul 2019 05:51pm
Last modified	20 Jul 2019 05:53pm

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

sex

age

sibsp

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,
        0.7944715023040771
      ]
    }
  ]
}
```

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.

Titanic AutoAI Deployed

Overview Implementation Test

Implementation

[View API Specification](#)

Scoring End-point	https://us-south.ml.cloud.ibm.com/v4/deployments/f6f65100-cb48-4ef9-a6b0-86c604cbc0b0/predictions
Authorization: Bearer <token>	Review the WML authentication documentation for details about generating IAM tokens.
ML-Instance-ID	The "ML-Instance-ID" HTTP header must be populated with the WML instance id, which can be obtained as described here .
Content-type: application/json	Required if the request body is sent in JSON format.

Code Snippets

cURL Java JavaScript Python Scala

```
import urllib3, requests, json

# NOTE: generate iam_token and retrieve ml_instance_id based on provided documentation
header = {'Content-Type': 'application/json', 'Authorization': 'Bearer ' + iam_token, 'ML-Instance-ID': ml_instance_id}

# NOTE: manually define and pass the array(s) of values to be scored in the next line
payload_scoring = {"input_data": [{"fields": [array_of_feature_columns], "values": [array_of_values_to_be_scored, another_array_of_values_to_be_scored]}]}

response_scoring = requests.post('https://us-south.ml.cloud.ibm.com/v4/deployments/f6f65100-cb48-4ef9-a6b0-86c604cbc0b0/predictions', json=payload_scoring, headers=header)
print("Scoring response")
```