Breast Cancer Risk Prediction System

Dr.Kanojia Sindhuben Babulal
Assistant Professor
Central University of Jharkhand

1) Introduction

1.1) Overview

Breast cancer is one of the main causes of cancer death worldwide. Early diagnostics significantly increases the chances of correct treatment and survival, but this process is tedious and often leads to a disagreement between pathologists. Computer-aided diagnosis systems showed the potential for improving diagnostic accuracy. But early detection and prevention can significantly reduce the chances of death. It is important to detect breast cancer as early as possible.

We will be building a model in Watson Studio and deploying the model in IBM Watson Machine Learning. To interact with the model we will be using Node-Red and scoring Endpoint.

Solution Requirements:

Develop a model that is capable of detecting the Breast Cancer in early stages. The Machine learning model is trained and deployed on IBM Watson Studio and an endpoint is created. The web application is built using IBM Node-Red.

1.2) Purpose

Risk prediction models can be an important tool for **breast cancer** prevention, by identifying women at high **risk** who would mostly benefit from targeted preventive measures such as mammography screening, or chemoprevention, e.g. with tamoxifen or raloxifene.

2) Literature Survey

2.1 Existing problem

Most existing breast cancer screening programs are based on mammography at similar time intervals -- typically, annually or every two years -- for all women. This "one size fits all" approach is not optimized for cancer detection on an individual level and may hamper the effectiveness of screening programs.

Breast cancer represents one of the diseases that make a high number of deaths every year. It is the most common type of all cancers and the main cause of women's deaths worldwide. Classification and data mining methods are an effective way to classify data. Especially in medical field, where those methods are widely used in diagnosis and analysis to make decisions.

The second major cause of women's death is breast cancer (after lung cancer) [1]. 246,660 of women's new cases of invasive breast cancer are expected to be diagnosed in the US during 2016 and 40,450 of women's death is estimated [2]. Breast cancer represents about 12% of all new cancer cases and 25% of all cancers in women [3]. Information and Communication Technologies (ICT) can play potential roles in cancer care. In fact, Big data has advanced not only the size of data but also creating value from it; Big data, that becomes a synonymous of data mining, business analytics and business intelligence, has made a big change in BI from reporting and decision to prediction results [4] . Data mining approaches, for instance, applied to medical science topics rise rapidly due to their high performance in predicting outcomes, reducing costs of medicine, promoting patients' health, improving healthcare value and quality and in making real time decision to save people's lives. There are many algorithms for classification and prediction of breast cancer outcomes.

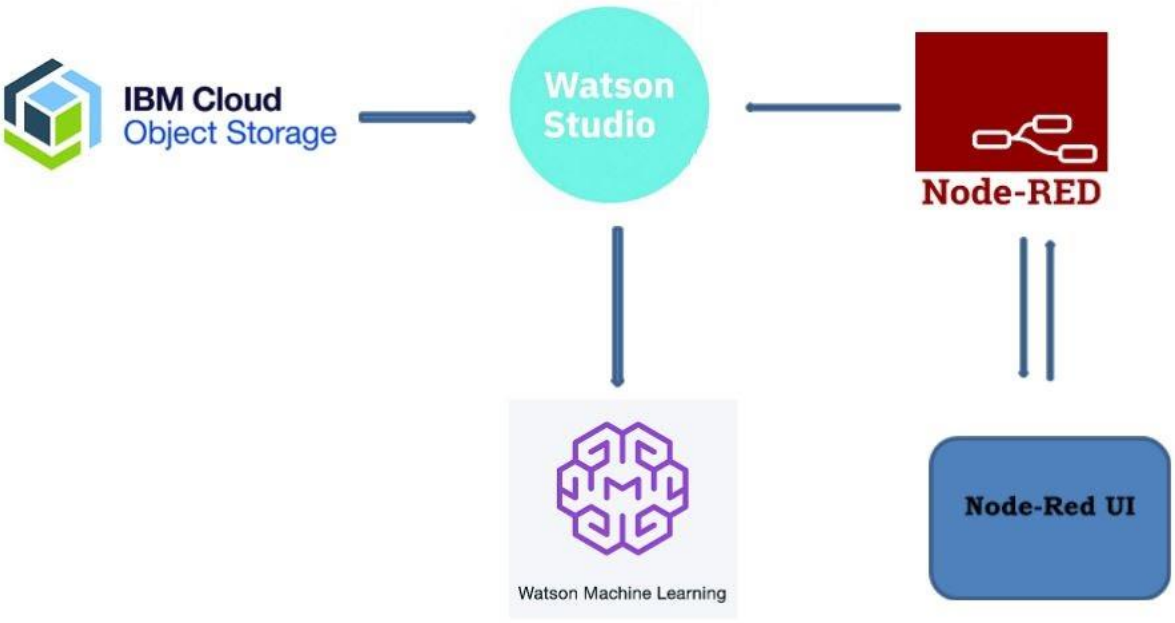
2.2 Proposed Solution

1. Use Watson Studio
2. Create Project
3. Add an Auto AI Experiment
4. Create Machine Learning Instance
5. Associte Machine Learning Instance with our Project
6. Load dataset downloaded from kaggle to Cloud Object Storage

7. Select Prediction Parameter in the dataset
8. Train the Model for various Algorithm and choose the best result and save as model
9. Deploy the model
10. Use Node Red to create the Flow and build the Webbased Application

3) Theoritical Analysis

3.1 Block diagram



4) Experimental Investigation

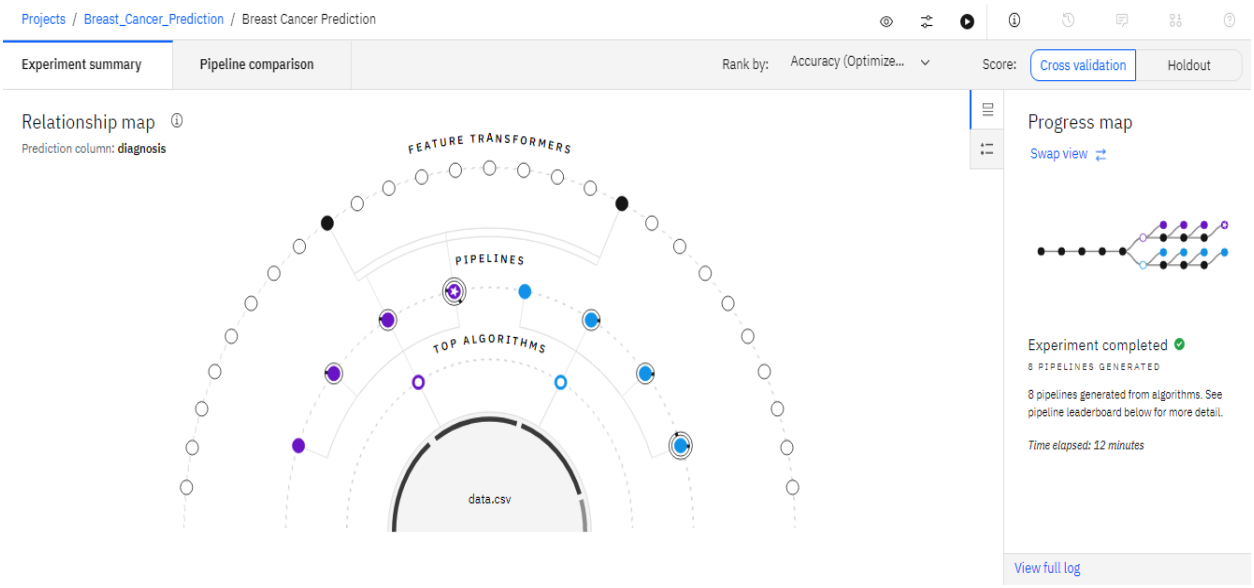
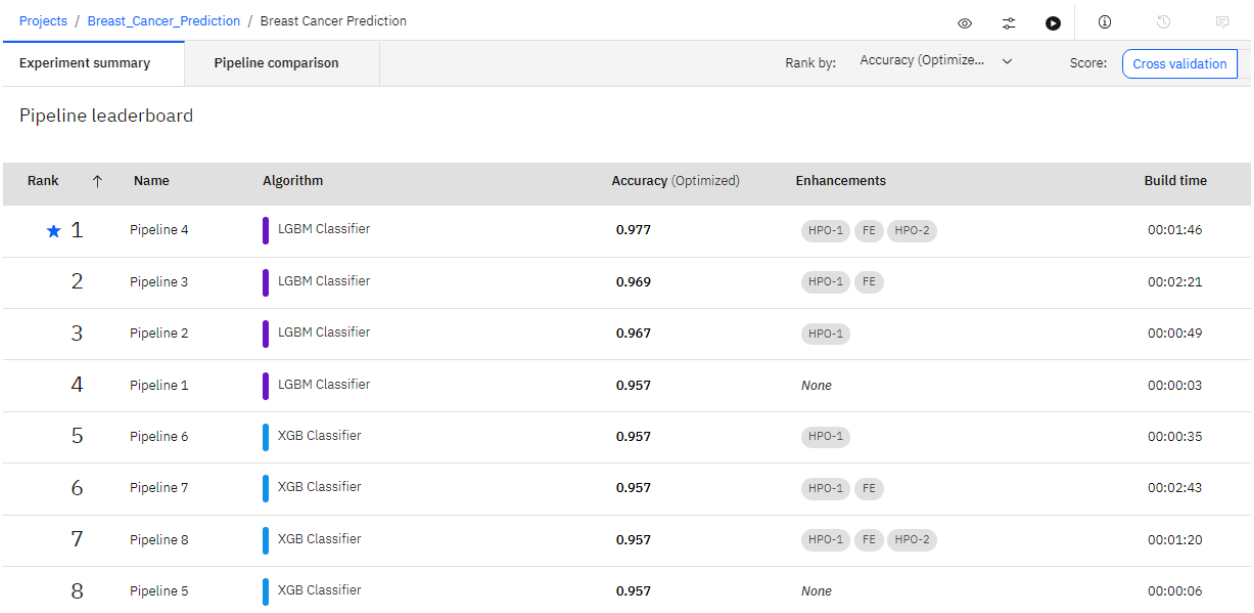


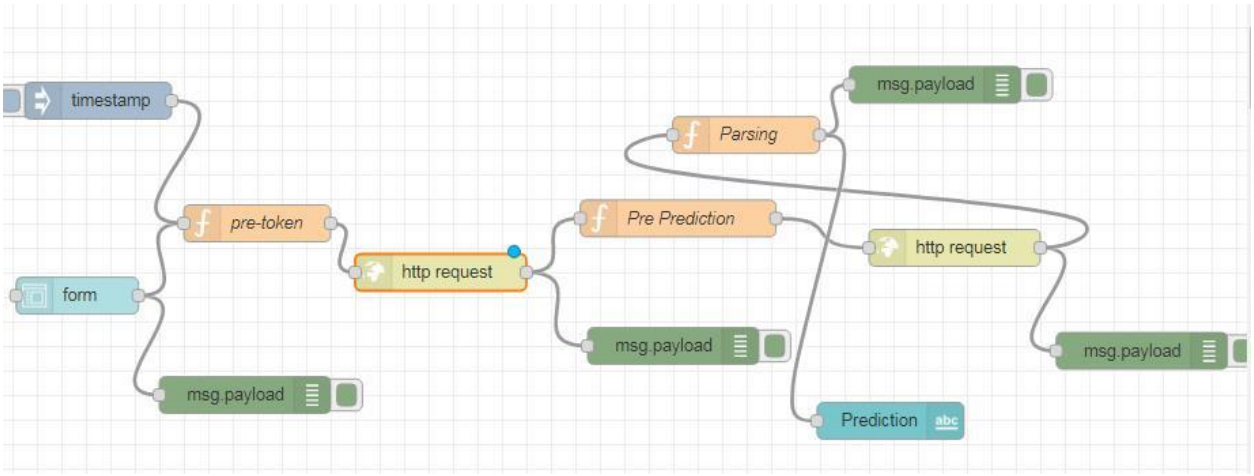
Figure 1 :- Comparison betweeen various algorithm



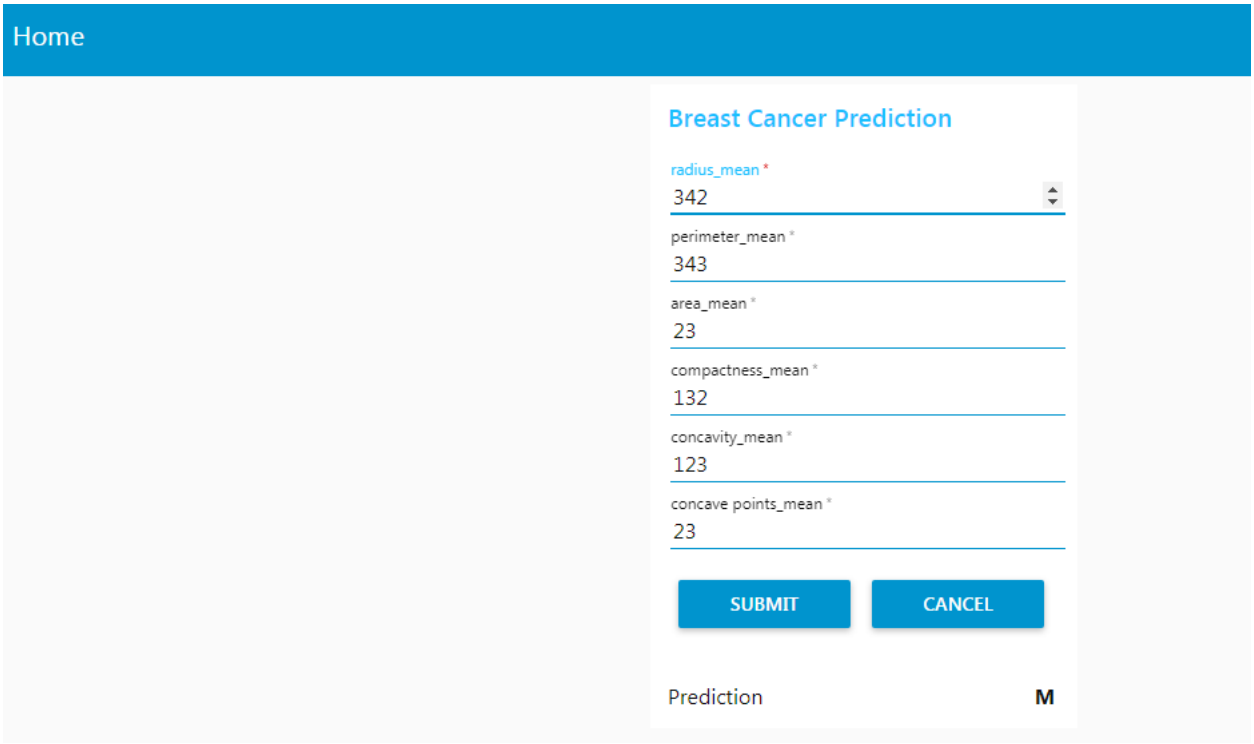
Rank	Name	Algorithm	Accuracy (Optimized)	Enhancements	Build time
★ 1	Pipeline 4	LGBM Classifier	0.977	HPO-1 FE HPO-2	00:01:46
2	Pipeline 3	LGBM Classifier	0.969	HPO-1 FE	00:02:21
3	Pipeline 2	LGBM Classifier	0.967	HPO-1	00:00:49
4	Pipeline 1	LGBM Classifier	0.957	*None*	00:00:03
5	Pipeline 6	XGB Classifier	0.957	HPO-1	00:00:35
6	Pipeline 7	XGB Classifier	0.957	HPO-1 FE	00:02:43
7	Pipeline 8	XGB Classifier	0.957	HPO-1 FE HPO-2	00:01:20
8	Pipeline 5	XGB Classifier	0.957	*None*	00:00:06

Figure 2 :- Leaderboard Representation of various Algorithm

5) Flowchart



6) Results



7) Advantages Disadvantages

The advantage is the result is good as various algorithm are involved for comparision and the best is saved.

The disadvantage is the authenticity of data set used for prediction. Authentic and real data can be used for prediction.

8) Applications

1. Prediction of any of the disease if can be done prior is always beneficial to the person as far as is health is concerned.
2. Better precaution can be taken
3. Can he helpful to doctors and medical practioners.

9) Future Scope

More algorithms can be included prediction process and the algorithm complexity can be reduced.

10) Bibilography

1) U.S. Cancer Statistics Working Group. United States Cancer Statistics: 1999–2008 Incidence and Mortality Web-based Report. Atlanta (GA): Department of Health and Human Services, Centers for Disease Control and Prevention, and National Cancer Institute; 2012.

2) Siegel RL, Miller KD, Jemal A. Cancer Statistics , 2016. 2016;00(00):1-24. doi:10.3322/caac.21332.

3) “Globocan 2012 - Home.” [Online]. Available: http://globocan.iarc.fr/Default.aspx. [Accessed: 28-Dec-2015].

4) Asri H, Mousannif H, Al Moatassime H, Noel T. Big data in healthcare: Challenges and opportunities. 2015 Int Conf Cloud Technol Appl. 2015:1-7. doi:10.1109/CloudTech.2015.7337020.

Appendix

Setup

Before you use the sample code in this notebook, you must perform the following setup tasks:

- ibm_watson_machine_learning installation
- autoai-libs installation/upgrade
- lightgbm or xgboost installation/downgrade if they are needed.

In [ ]:

```
!pip install -U ibm-watson-machine-learning
```

In [ ]:

```
!pip install -U autoai-libs
!pip install -U lightgbm==2.2.3
```

AutoAI experiment metadata

This cell defines COS credentials required to retrieve AutoAI pipeline.

In [ ]:

```
# @hidden_cell
from ibm_watson_machine_learning.helpers import DataConnection, S3Connection, S3Location

training_data_reference = [DataConnection(
    connection=S3Connection(
        api_key='ZXoWn-eAAMw8AOVxUg1AIyW5QXm9c6_5SjI3vCDqXTCi',
        auth_endpoint='https://iam.bluemix.net/oidc/token/',
        endpoint_url='https://s3-api.us-geo.objectstorage.softlayer.net'
    ),
        location=S3Location(
        bucket='breastcancerprediction-donotdelete-pr-pyfb9vshryybwb',
        path='data.csv'
    ))
]
training_result_reference = DataConnection(
    connection=S3Connection(
        api_key='ZXoWn-eAAMw8AOVxUg1AIyW5QXm9c6_5SjI3vCDqXTCi',
        auth_endpoint='https://iam.bluemix.net/oidc/token/',
        endpoint_url='https://s3-api.us-geo.objectstorage.softlayer.net'
    ),
    location=S3Location(
        bucket='breastcancerprediction-donotdelete-pr-pyfb9vshryybwb',
        path='auto_ml/5cddb7a1-95b6-4f20-8581-ec937da13815/wml_data/e80d218b-9f6a-4a0c-ad42-
7f2d6d4699ed/data/automl',
        model_location='auto_ml/5cddb7a1-95b6-4f20-8581-ec937da13815/wml_data/e80d218b-9f6a-4a0c-
ad42-7f2d6d4699ed/data/automl/hpo_c_output/Pipeline1/model.pickle',
        training_status='auto_ml/5cddb7a1-95b6-4f20-8581-ec937da13815/wml_data/e80d218b-9f6a-4a0c-ad42-
7f2d6d4699ed/training-status.json'
    ))
```

Following cell contains input parameters provided to run the AutoAI experiment in Watson Studio

In [ ]:

```
experiment_metadata = dict(
    prediction_type='classification',
    prediction_column='diagnosis',
    test_size=0.1,
    scoring='accuracy',
    project_id='8cd0d60b-fa82-444c-aff5-e153277cd9c1',
    csv_separator=',',
    random_state=33,
```

```
  max_number_of_estimators=2,
  training_data_reference = training_data_reference,
  training_result_reference = training_result_reference,
  deployment_url='https://us-south.ml.cloud.ibm.com')

pipeline_name='Pipeline_4'
```

Pipeline inspection

In this section you will get the trained pipeline model from the AutoAI experiment and inspect it.
You will see pipeline as a pythone code, graphically visualized and at the end, you will perform a local test.

Get historical optimizer instance

The next cell contains code for retrieving fitted optimizer.

In [ ]:

```
from ibm_watson_machine_learning.experiment import AutoAI

optimizer = AutoAI().runs.get_optimizer(metadata=experiment_metadata)
```

Get pipeline model

The following cell loads selected AutoAI pipeline model. If you want to get pure scikit-learn pipeline specify as_type='sklearn' parameter. By default enriched scikit-learn pipeline is returned as_type='lale'.

In [ ]:

```
pipeline_model = optimizer.get_pipeline(pipeline_name=pipeline_name)
```

Preview pipeline model as python code

In the next cell, downloaded pipeline model could be previewed as a python code.
You will be able to see what exact steps are involved in model creation.

In [ ]:

```
pipeline_model.pretty_print(combinators=False, ipython_display=True)
```

Visualize pipeline model

Preview pipeline model stages as graph. Each node's name links to detailed description of the stage.

In [ ]:

```
pipeline_model.visualize()
```

Read training data

Retrieve training dataset from AutoAI experiment as pandas DataFrame.

In [ ]:

```
train_df = optimizer.get_data_connections()[0].read()
test_df = train_df.sample(n=5).drop([experiment_metadata['prediction_column']], axis=1)
```

Test pipeline model locally

You can predict target value using trained AutoAI model by calling predict().

In [ ]:

```
y_pred = pipeline_model.predict(test_df.values)
print(y_pred)
```

Pipeline refinery and testing (optional)

In this section you will learn how to refine and retrain the best pipeline returned by AutoAI. It can be performed by:

- modifying pipeline definition source code
- using lale library for semi-automated data science

Note: In order to run this section change following cells to 'code' cell.

Pipeline definition source code

Following cell lets you experiment with pipeline definition in python, e.g. change steps parameters.

It will inject pipeline definition to the next cell.
pipeline_model.pretty_print(combinators=False, ipython_display='input')

Lale library

Note: This is only an exemplary usage of lale package. You can import more different estimators to refine downloaded pipeline model.

Import estimators

from sklearn.linear_model import LogisticRegression as E1 from sklearn.tree import DecisionTreeClassifier as E2 from sklearn.neighbors import KNeighborsClassifier as E3 from lale.lib.lale import Hyperopt from lale.operators import TrainedPipeline from lale import wrap_imported_operators from lale.helpers import import_from_sklearn_pipeline wrap_imported_operators()

Pipeline decomposition and new definition

In this step the last stage from pipeline is removed.
prefix = pipeline_model.remove_last().freeze_trainable() prefix.visualize()new_pipeline = prefix >> (E1 | E2 | E3) new_pipeline.visualize()

New optimizer hyperopt configuration and training

This section can introduce other results than the original one and it should be used by more advanced users.

New pipeline is re-trained by passing train data to it and calling fit method.

Following cell performs dataset split for refined pipeline model.
from sklearn.model_selection import train_test_split train_X =
train_df.drop([experiment_metadata['prediction_column']], axis=1).values train_y =
train_df[experiment_metadata['prediction_column']].values train_X, test_X, train_y, test_y =
train_test_split(train_X, train_y, test_size=experiment_metadata['test_size'], stratify=train_y,
random_state=experiment_metadata['random_state'])hyperopt = Hyperopt(estimator=new_pipeline, cv=3,
max_evals=20) fitted_hyperopt = hyperopt.fit(train_X, train_y)hyperopt_pipeline =
fitted_hyperopt.get_pipeline() new_pipeline = hyperopt_pipeline.export_to_sklearn_pipeline()prediction =
new_pipeline.predict(test_X)from sklearn.metrics import accuracy_score score =
accuracy_score(y_true=test_y, y_pred=prediction) print('accuracy_score: ', score)

Deploy and Score

In this section you will learn how to deploy and score pipeline model as webservice using WML instance.

Connection to WML

Authenticate the Watson Machine Learning service on IBM Cloud.

Tip: Your Cloud API key can be generated by going to the **Users** section of the Cloud console. From that page, click your name, scroll down to the **API Keys** section, and click **Create an IBM Cloud API key**. Give your key a name and click **Create**, then copy the created key and paste it below.

Note: You can also get service specific apikey by going to the **Service IDs** section of the Cloud Console. From that page, click **Create**, then copy the created key and paste it below.

Action: Enter your api_key in the following cell.

In [ ]:

```
api_key = "PUT_YOUR_API_KEY_HERE"

wml_credentials = {
  "apikey": api_key,
  "url": experiment_metadata["deployment_url"]
}
```

Create deployment

Action: If you want to deploy refined pipeline please change the pipeline_name to new_pipeline. If you prefer you can also change the deployment_name.

Action: To perform deployment please specify target_space_id.

In [ ]:

```
target_space_id = "PUT_YOUR_TARGET_SPACE_ID_HERE"

from ibm_watson_machine_learning.deployment import WebService
service = WebService(source_wml_credentials=wml_credentials,
                     target_wml_credentials=wml_credentials,
                     source_project_id=experiment_metadata['project_id'],
                     target_space_id=target_space_id)
service.create(
model=pipeline_name,
metadata=experiment_metadata,
deployment_name=f'{pipeline_name}_webservice'
)
```

Deployment object could be printed to show basic information:

In [ ]:

```
print(service)
```

To be able to show all available information about deployment use .get_params() method:

In [ ]:

```
service.get_params()
```

Score webservice

You can make scoring request by calling score() on deployed pipeline.

In [ ]:

```
predictions = service.score(payload=test_df)
predictions
```

If you want to work with the webservice in external Python application you can retrieve the service object by:

- initialize service by:
- service = WebService(target_wml_credentials=wml_credentials,
  target_space_id=target_space_id)
- get deployment_id by service.list() method
- get webservice object by service.get('deployment_id') method

After that you can call service.score() method.

Delete deployment

You can delete an existing deployment by calling service.delete()