Predict Heart Failure Using IBM Auto Ai Service

Dr.Kanojia Sindhuben Babulal
Assistant Professor
Department of Computer Science &Technology
Central University of Jharkhand

1) Introduction

1.1) Overview

Heart disease is the main cause of morbidity and mortality worldwide. Cardiovascular diseases (CVDs) are the number 1 cause of death globally, taking an estimated 17.9 million lives each year, which accounts for 31% of all deaths worldwide.

Heart failure is a common event caused by CVDs and this dataset contains 9 features that can be used to predict mortality by heart failure. In this project, we have build a model using Auto AI and build a web application where we can get the prediction of heart failure.

1.2) Purpose

Most heart diseases are highly preventable and simple lifestyle modifications(such as reducing tobacco use, eating healthily, obesity and exercising) coupled with early treatment greately improve their prognoses. It is, however, difficult to identify high risk patients because of the mulfactorial nature of several contributory risk factors such as diabetes, high blood pressure, high cholesterol et cetera. Due to such constraints, scientists have turned towards modern approaches like Data Mining and Machine Learning for predicting the disease.

Machine learning (ML), due to its superiority in pattern detection and classification,

proves to be effective in assisting decision making and risk assesment from the large quantity of data produced by the healthcare industry on heart disease.

2) Literature Survey

2.1 Existing problem

One common application of machine learning is the prediction of an outcome based upon existing data.Today the heart disease is one of the most important causes of death in the world. So its early prediction and diagnosis is important in medical field, which could help in on time treatment, decreasing health costs and decreasing death caused by it.

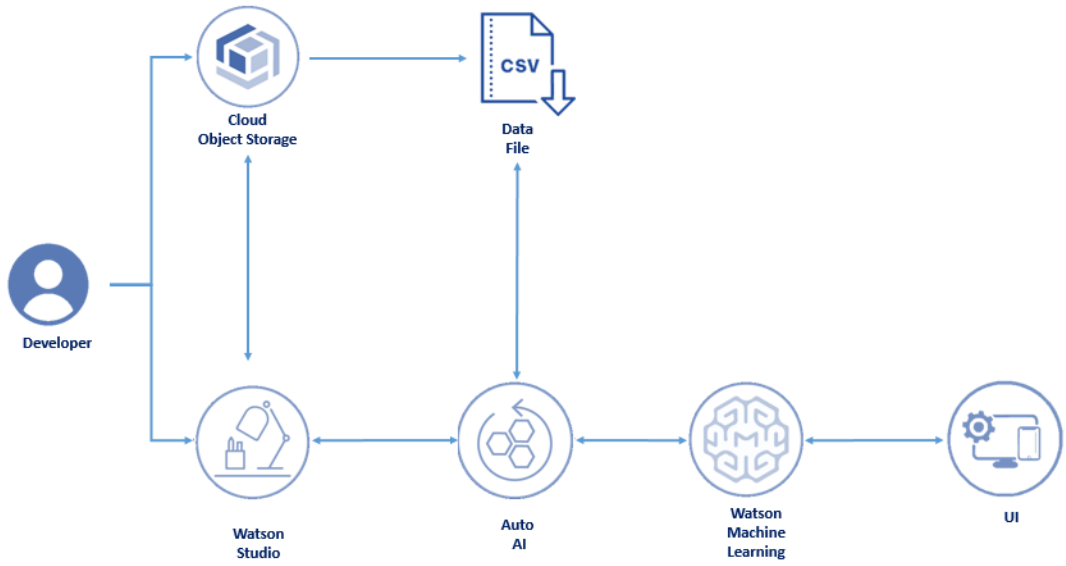
Machine learning or data mining is useful for a diverse set of problems. One of the applications of this technique is in predicting a dependent variable from the values of independent variables. The healthcare field is an application area of data mining since it has vast data resources that are difficult to be handled manually. Heart disease has been identified as one of the largest causes of death even in developed countries [1]. One of the reasons for fatality due to heart disease is due to the fact that the risks are either not identified, or they are identified only at a later stage. However, machine learning techniques can be useful for overcoming this problem and to predict risk at an early stage. Some of the techniques used for such prediction problems are the Support Vector Machines (SVM), Neural Networks, Decision Trees, Regression and Naïve Bayes classifiers. SVM was identified as the best predictor with 92.1% accuracy, followed by neural networks with 91% accuracy, and decision trees showed a lesser accuracy of 89.6% [2]. Sex, age, smoking, hypertension, and diabetes were considered to be the risk factors for heart disease [3].

2.2 Proposed Solution

- Use Watson Studio
- Create Project
- Add an Auto AI Experiment
- Create Machine Learning Instance
- Associte Machine Learning Instance with our Project
- Load dataset downloaded from kaggle to Cloud Object Storage
- Select Prediction Parameter in the dataset
- Train the Model for various Algorithm and choose the best result and save as model
- Deploy the model
- Use Node Red to create the Flow and build the Webbased Application

3) Theoritical Analysis

3.1 Block diagram



4) Experimental Investigation

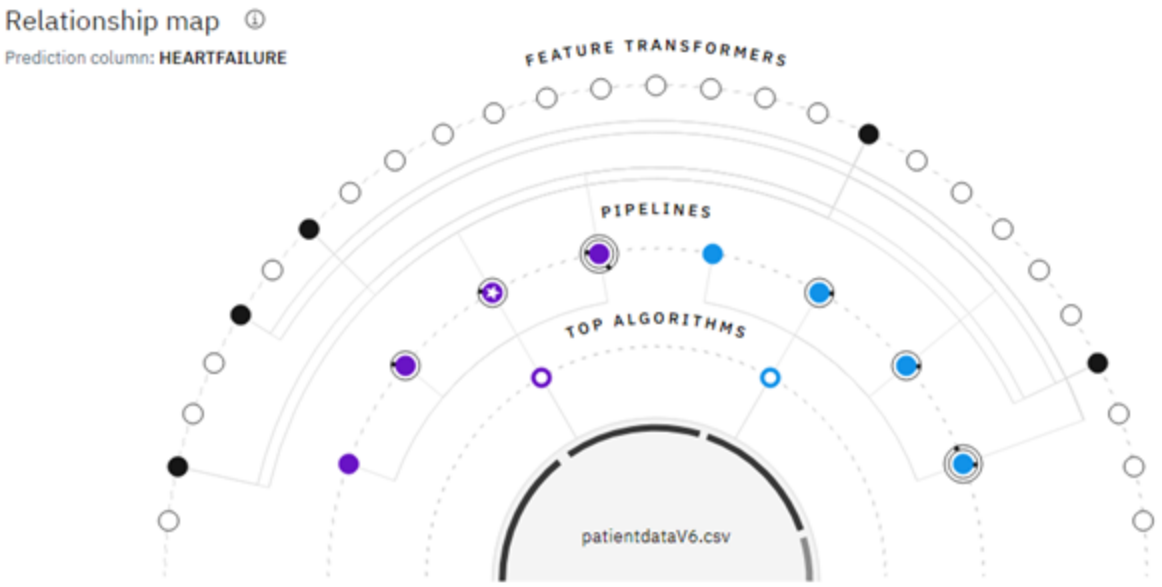


Figure 1:- Comparison between various algorithms

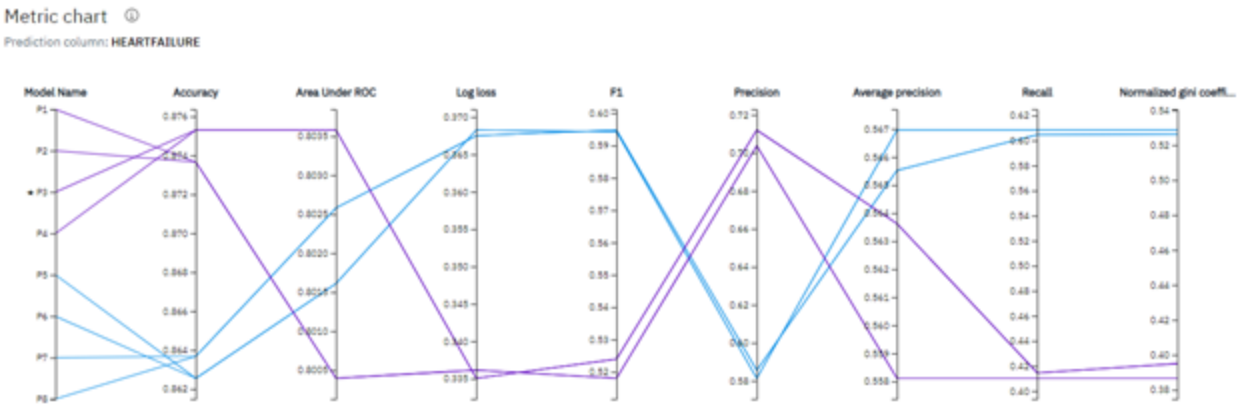


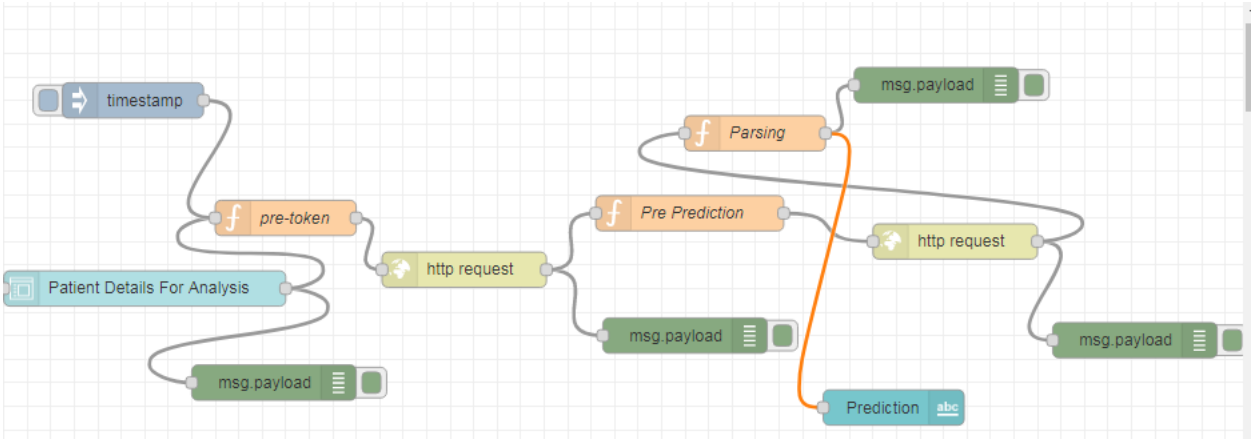
Figure 2:- Metric Chart Representation

Pipeline leaderboard

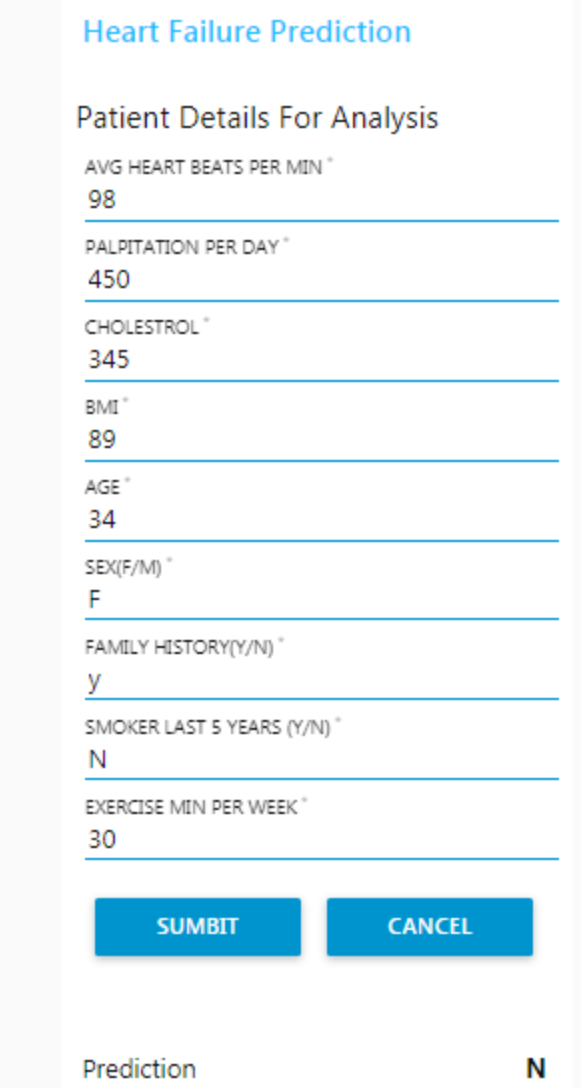
Rank	Name	Algorithm	Accuracy (Optimized)	Average prec...	F_1	Log loss	Normalized ...	Precision	Recall	ROC AUC
★ 1	Pipeline 3	Gradient Boosting Classifier	0.875	0.564	0.524	0.335	0.395	0.712	0.415	0.804
2	Pipeline 4	Gradient Boosting Classifier	0.875	0.564	0.524	0.335	0.395	0.712	0.415	0.804
3	Pipeline 1	Gradient Boosting Classifier	0.874	0.558	0.518	0.336	0.387	0.704	0.410	0.800
4	Pipeline 2	Gradient Boosting Classifier	0.874	0.558	0.518	0.336	0.387	0.704	0.410	0.800
5	Pipeline 7	XGB Classifier	0.864	0.566	0.595	0.368	0.526	0.586	0.605	0.803
6	Pipeline 8	XGB Classifier	0.864	0.566	0.595	0.368	0.526	0.586	0.605	0.803
7	Pipeline 5	XGB Classifier	0.863	0.567	0.594	0.368	0.529	0.581	0.608	0.802
8	Pipeline 6	XGB Classifier	0.863	0.567	0.594	0.368	0.529	0.581	0.608	0.802

Figure 3:- 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

- Prediction of any of the disease if can be done prior is always beneficial to the person as far as is health is concerned.

- Better precaution can be taken
- 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) K. Vanisree, JyothiSingaraju**Decision support system for congenital heart disease diagnosis based on signs and symptoms using neural networks**
Int J Comput Appl, 19 (6) (April 2011)

2). Xiao Liu, Xiaoli Wang, Qiang Su, Mo Zhang, Yanhong Zhu, Qiugen Wang, Qian Wang**A hybrid classification system for heart disease diagnosis based on the RFRS method**
Comput. Math. Methods Med., 2017 (2017), pp. 1-11

3). Yanwei Xing, Jie Wang, Zhihong Zhao Yonghong Gao**Combination data mining methods with new medical data to predicting outcome of Coronary Heart Disease**
Convergence Information Technology (2007), pp. 868-872

4) https://cloud.ibm.com/catalog/services/watson-studio

5) https://www.youtube.com/watch?v=unyZ8SAhuPQ

6)
https://developer.ibm.com/components/node-red/tutorials/how-to-create-a-node-red-starter-application/

Appendix

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='i7PBUQmOg-vXdCjzEG7htcyW2hX_uZcIAlWGK0SXafIW',
    auth_endpoint='https://iam.bluemix.net/oidc/token/',
    endpoint_url='https://s3-api.us-geo.objectstorage.softlayer.net'
  ),
    location=S3Location(
    bucket='heartattackprediction-donotdelete-pr-g62qzwbjkwxwhr',
    path='patientdataV6.csv'
  ))
]
training_result_reference = DataConnection(
  connection=S3Connection(
    api_key='i7PBUQmOg-vXdCjzEG7htcyW2hX_uZcIAlWGK0SXafIW',
    auth_endpoint='https://iam.bluemix.net/oidc/token/',
    endpoint_url='https://s3-api.us-geo.objectstorage.softlayer.net'
  ),
  location=S3Location(
    bucket='heartattackprediction-donotdelete-pr-g62qzwbjkwxwhr',

path='auto_ml/c9d7ee38-3056-495c-9cba-374c2c1af1bb/wml_data/dce29059-6ab4-4069-9df1-8183c1d40756/data/automl',

model_location='auto_ml/c9d7ee38-3056-495c-9cba-374c2c1af1bb/wml_data/dce29059-6ab4-4069-9df1-8183c1d40756/data/automl/cognito_output/Pipeline1/model.pickle',

training_status='auto_ml/c9d7ee38-3056-495c-9cba-374c2c1af1bb/wml_data/dce29059-6ab4-4069-9df1-8183c1d40756/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='HEARTFAILURE',
  test_size=0.1,
```

```
  scoring='accuracy',
  project_id='17c7cabd-968d-425f-8865-70b687073010',
  deployment_url='https://us-south.ml.cloud.ibm.com',
  csv_separator=',',
  random_state=33,
  excel_sheet=0,
  max_number_of_estimators=2,
  training_data_reference = training_data_reference,
  training_result_reference = training_result_reference)

pipeline_name='Pipeline_3'
```

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_model to new_pipeline. If you prefer you can also change the deployment_name. 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(target_wml_credentials=wml_credentials,
                     target_space_id=target_space_id)
service.create(
model=pipeline_model,
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().