HEART FAILURE RISK PREDICTION THROUGH CLINICAL DECISION SUPPORT SYSTEM (HFRP - CDSS) DEVELOPED USING IBM AUTO AI SERVICE

1. INTRODUCTION

1.1. Overview

Diagnosis of Cardio Vascular Diseases (CVDs) is a daunting and challenging task and researchers across the world have developed numerous artificially intelligent systems for enhanced heart disease diagnosis and clinical decision support. According to the World Heart Federation, "More people die from CVDs worldwide than from any other cause and over 17.9 million deaths every year worldwide, according to the World Health Organization. Of these deaths, 80% are due to coronary heart diseases and cerebrovascular diseases and mostly affect low and middle income countries."

1.2. Purpose

The aim of the project, Heart Failure Risk Prediction through Clinical Decision Support System (HFRP - CDSS) developed using IBM AutoAI service, is to build a low cost, high efficiency and robust web application to predict the risk of heart failure using specific indicators or features. This is an important and pertinent project in current times since cardiovascular diseases are at a rise and the mortality rates are high, primarily due to lifestyle changes, which influence the health of the heart.

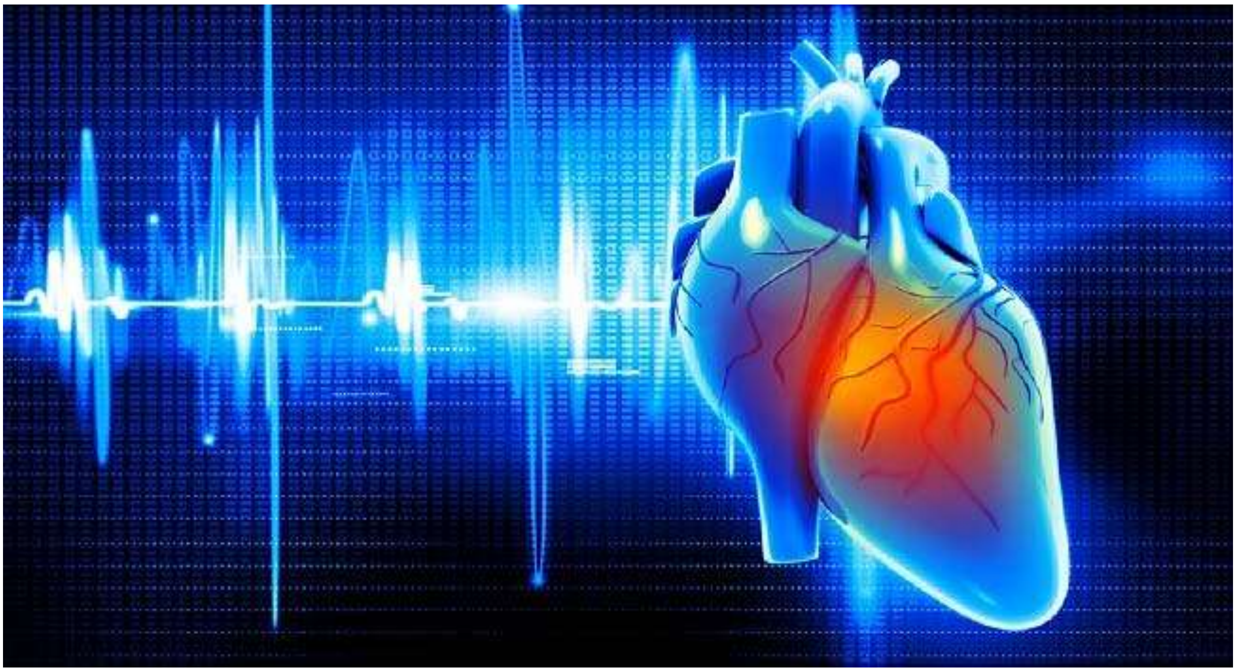


Image Source: WebMD

2. LITERATURE SURVEY

2.1. Existing problem

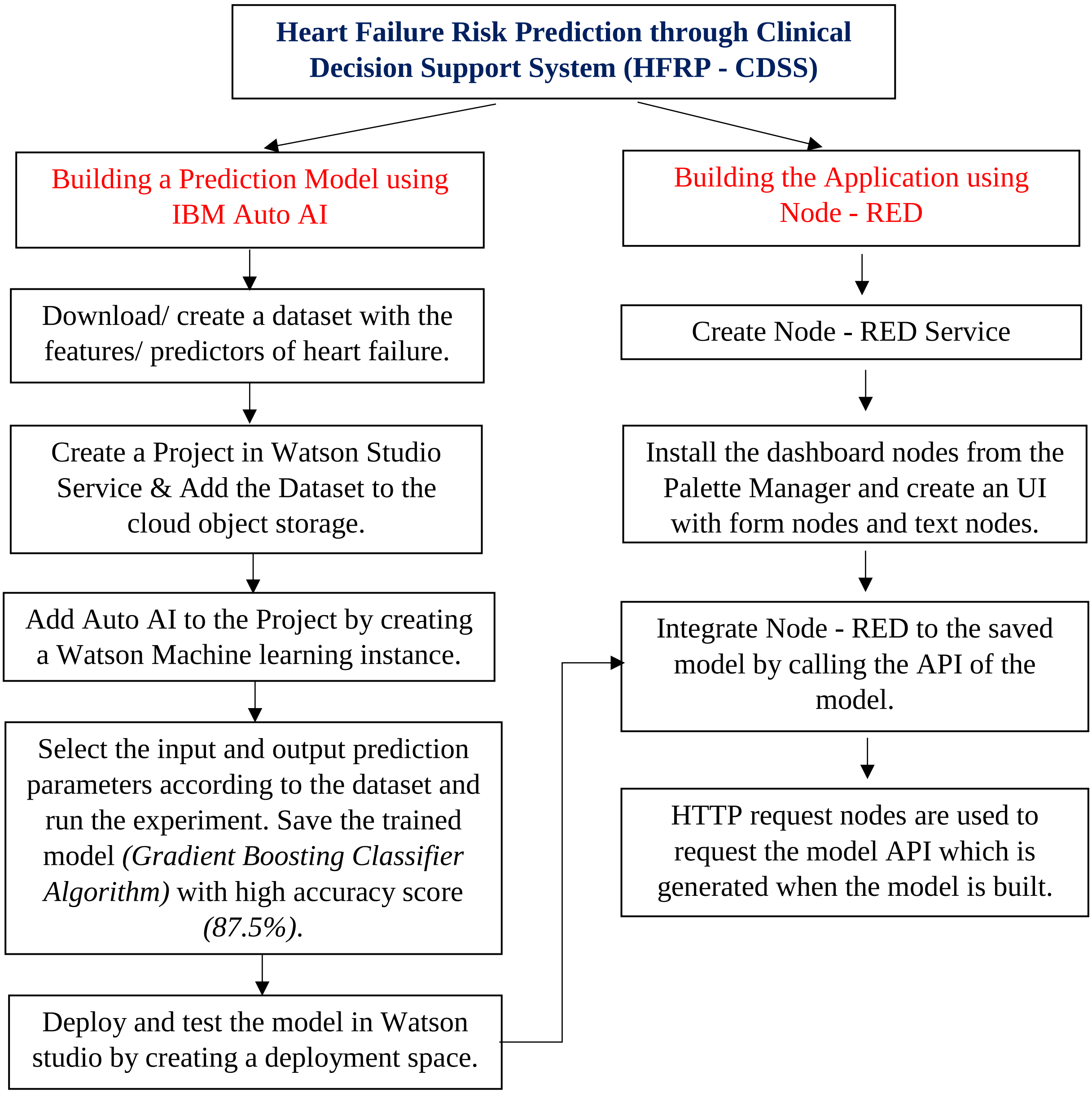
Cardio Vascular Diseases can be diagnosed by: Blood tests, ECG, Treadmill tests, Echocardiography, X- Ray, CT, MRI etc. These tests are either very expensive or invasive thereby creating a scope for a prediction tool which is non-invasive.

2.2. Proposed solution

The objective of this project is to come up with a solution to the challenge of diagnosing Cardio Vascular Diseases non-invasively, by employing Machine Learning tools and creating a web based application to predict heart failure caused due to CVDs.

3. THEORITICAL ANALYSIS

3.1. Block diagram



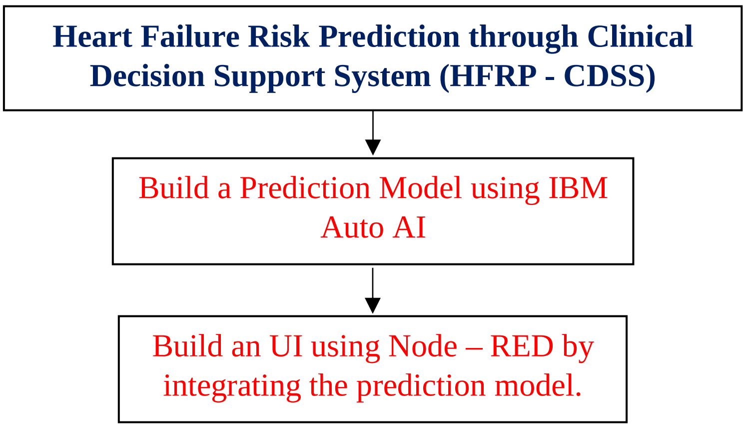
3.2. Hardware / Software designing

The following software tools are used in designing the heart failure prediction system: IBM AutoAI service, IBM Watson Studio, IBM Watson Machine Learning, Node-RED Dataset with nine input features and one output parameter heart failure prediction is used to train and build the prediction model: https://github.com/IBM/predictive-model-on-watson-ml/blob/master/data/patientdataV6.csv

4. EXPERIMENTAL INVESTIGATIONS

The tools in Machine Learning and Watson Studio available in IBM services catalog were explored to create the project in addition to Node-RED to create the UI.

5. FLOWCHART



6. RESULT

The web based application for Heart Failure Risk Prediction through Clinical Decision Support System (HFRP - CDSS) is developed using IBM AutoAI service, to predict the risk of heart failure using these nine input features – average heart beats per minute, no. of palpitations per day, cholesterol value, body mass index (BMI), age, sex, having a family history of CVDs, being a smoker for the last 5yrs, no. of minutes of exercise done per week. (https://smartinternz.com/Student/badge_workspace/5670)

7. ADVANTAGES & DISADVANTAGES

7.1. Advantages

HFRP – CDSS is a non-invasive, robust approach to predict heart failure caused by Cardio Vascular Diseases, as opposed to other invasive tests.

7.2. Disadvantages

The disadvantage of the online prediction tool is its sensitivity and accuracy for clinical use. It completely depends on the dataset used to train the model for prediction.

8. APPLICATIONS

The same machine learning prediction approach can be used to solve other challenging issues like diagnosis, classification and detection of various diseases like cancer, tumours, Alzheimer’s, Parkinson’s, skin diseases, renal failure etc.

9. CONCLUSION

The project built using Auto AI and Node-RED will aid in predicting the heart failure in humans with 87.5% accuracy using the Heart Failure Risk Prediction through Clinical Decision Support System (HFRP - CDSS) which employs the Gradient Boosting Classifier Algorithm.

10. FUTURE SCOPE

Signal and Image Processing tools in conjunction with machine learning algorithms can be applied to innovate non-invasive and robust solutions to several healthcare problems.

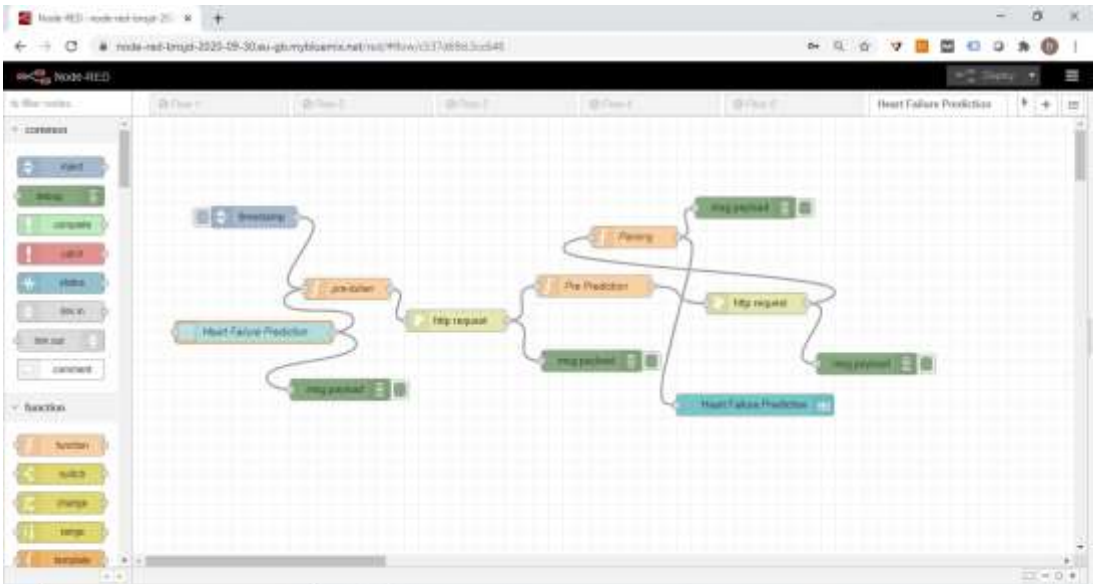
11. BIBILOGRAPHY

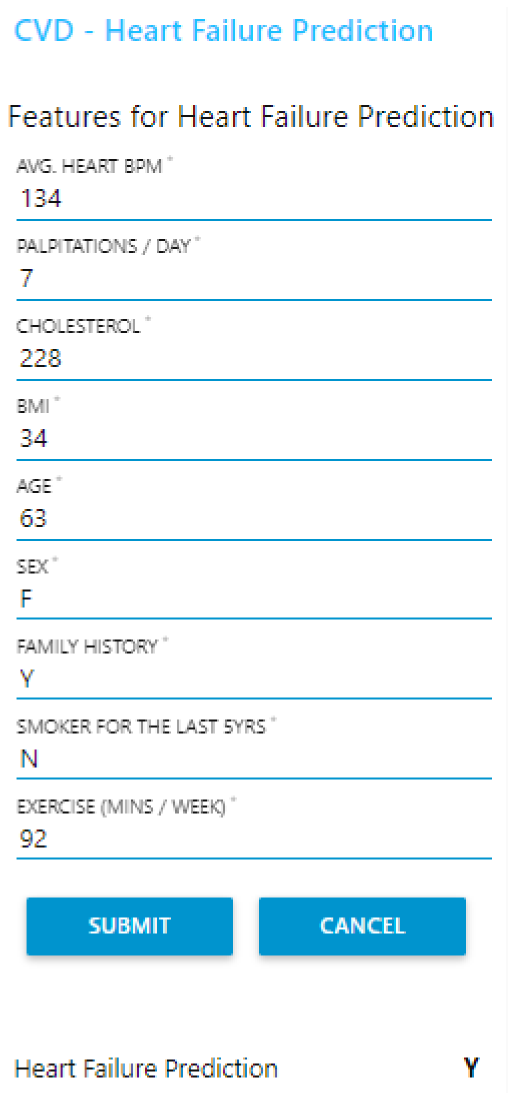
11.1. REFERENCES

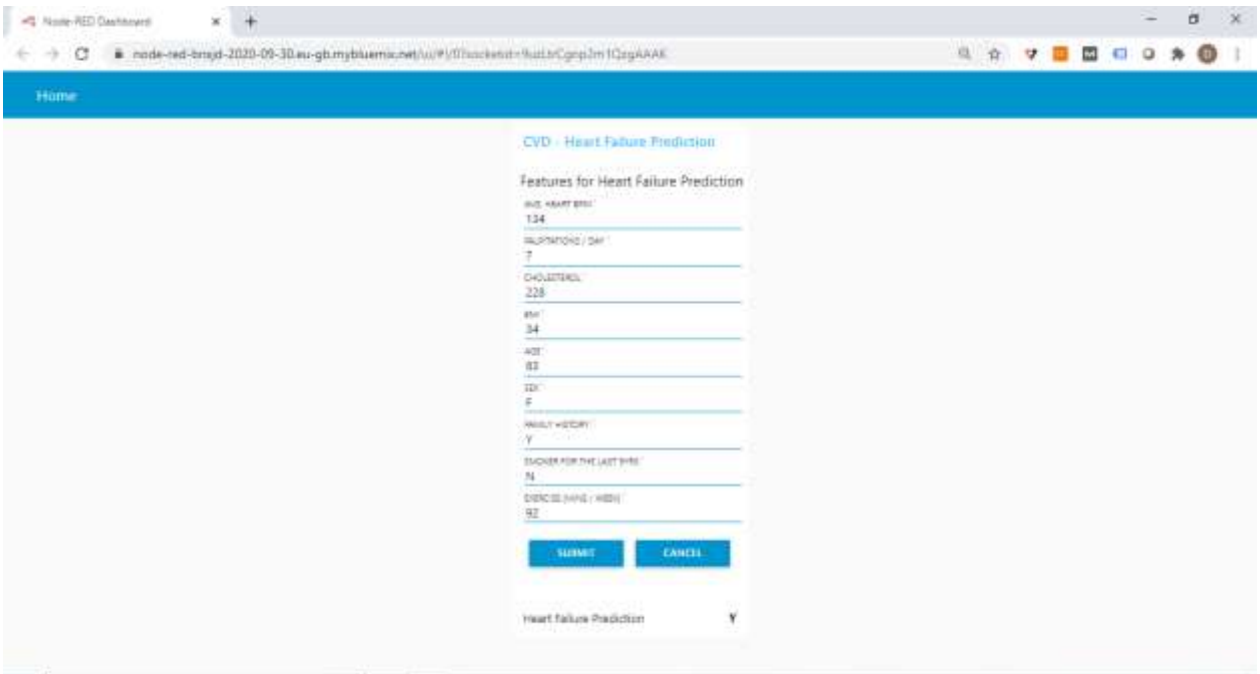
- https://www.kaggle.com/datasets
- https://cloud.ibm.com/
- https://cloud.ibm.com/catalog/services/watson-studio
- https://cloud.ibm.com/developer/appservice/create-app
- https://smartinternz.com/assets/Steps-to-be-followed-to-download-Watson-Studio-in-your-Local-System.pdf
- https://youtu.be/_glHhDbUCD8
- https://www.youtube.com/watch?v=unyZ8SAhuPQ

11.2. APPENDIX

A. Screenshots:




CVD - Heart Failure Prediction
Features for Heart Failure Prediction
AVG. HEART BPM *
134
PALPITATIONS / DAY *
7
CHOLESTEROL *
228
BMI *
34
AGE *
63
SEX *
F
FAMILY HISTORY *
Y
SMOKER FOR THE LAST 5YRS *
N
EXERCISE (MINS / WEEK) *
92
SUBMIT
CANCEL
Heart Failure Prediction
Y


Home
CVD - Heart Failure Prediction
Features for Heart Failure Prediction
134
7
228
34
F
Y
N
92
Heart Failure Prediction
Y

B. Source code:

- AutoAI generated Notebook:


```
{"cells": [{"cell_type": "markdown", "metadata": {},
"source": "### IBM AutoAI-SDK Auto-Generated
Notebook v1.13.1\n\n**Note:** Notebook code
generated using AutoAI will execute successfully. If
code is modified or reordered,   \nthere is no
guarantee it will successfully execute. This
pipeline is optimized for the original dataset.
\nThe pipeline may fail or produce sub-optimium
results if used with different data. For different
data,  \nplease consider returning to AutoAI
Experiments to generate a new pipeline. Please read
our documentation   \nfor more information:   \n<a
href=\"https://dataplatform.cloud.ibm.com/docs/conte
nt/wsj/analyze-data/autoai-notebook.html\">Cloud
Platform</a>  \n\n\nBefore modifying the pipeline or
trying to re-fit the pipeline, consider:   \nThe
notebook converts dataframes to numpy arrays before
fitting the pipeline   \n(a current restriction of
the preprocessor pipeline). The known_values_list is
passed by reference   \nand populated with
categorical values during fit of the preprocessing
pipeline. Delete its members before re-fitting."},
{"cell_type": "markdown", "metadata": {}, "source":
"<a id=\"content\"></a>\n## Notebook content\n\nThis
notebook contains steps and code to demonstrate
AutoAI pipeline. This notebook introduces commands
for getting data,  \npipeline model, model
inspection and testing.\n\nSome familiarity with
Python is helpful. This notebook uses Python 3."},
{"cell_type": "markdown", "metadata": {"pycharm":
{"name": "#%% md\n"}}, "source": "## Notebook
goals\n\n-  inspection of trained pipeline via
graphical vizualization and source code preview\n-
pipeline evaluation\n-  pipeline deployment and
webservice scoring.\n\n## Contents\n\nThis notebook
contains the following parts:\n\n1.\t[Setup](#setup)
\n    a.  [AutoAI experiment
metadata](#variables_definition)
\n2.\t[Pipeline inspection](#inspection)      \n
a.  [Get historical optimizer
instance](#get_hist_and_train)      \n    b.  [Get
pipeline model](#get_pipeline)      \n    c.
[Preview pipeline model as python
code](#preview_model_to_python_code)      \n    d.
[Visualize pipeline model](#visualize_pipeline)
\n    e.  [Read training data](#train_read)
\n    f.  [Test pipeline model locally](#test_model)
\n3.\t[Pipeline refinery and testing
(optional)](#refinery)  \n    a.  [Pipeline
definition source code](#pipeline_definition)
\n    b.  [Lale library](#lale_library)
```

\n4.\t[Deploy and score](#scoring)   \n   a.
[Insert WML credentials](#wml_credentials)   \n
b. [Create deployment](#deployment)   \n   c.
[Score webservice](#online_scoring)   \n   d.
[Delete deployment](#delete_deployment)   \n5.
[Authors](#authors)   "}, {"cell_type":
"markdown", "metadata": {}, "source": "<a
id=\"setup\">\n# Setup\n\nBefore you use the
sample code in this notebook, you must perform the
following setup tasks:\n - `watson-machine-learning-
client` uninstallation of the old client\n -
`ibm_watson_machine_learning` installation\n -
`autoai-libs` installation/upgrade\n - `lightgbm` or
`xgboost` installation/downgrade if they are
needed."}, {"cell_type": "code", "execution_count":
null, "metadata": {"pycharm": {"name": "#%%\n"}},
"outputs": [], "source": "!pip uninstall watson-
machine-learning-client -y"}, {"cell_type": "code",
"execution_count": null, "metadata": {"pycharm":
{"is_executing": false, "name": "#%%\n"}},
"outputs": [], "source": "!pip install -U ibm-
watson-machine-learning"}, {"cell_type": "code",
"execution_count": null, "metadata": {"pycharm":
{"name": "#%%\n"}}, "outputs": [], "source": "!pip
install -U autoai-libs"}, {"cell_type": "markdown",
"metadata": {}, "source": "<a
id=\"variables_definition\">\n### AutoAI
experiment metadata\n\nThis cell defines COS
credentials required to retrieve AutoAI pipeline."},
{"cell_type": "code", "execution_count": null,
"metadata": {"pycharm": {"name": "#%%\n"}},
"outputs": [], "source": "# @hidden_cell\nfrom
ibm_watson_machine_learning.helpers import
DataConnection, S3Connection,
S3Location\n\ntraining_data_reference =
[DataConnection(\n   connection=S3Connection(\n
api_key='-_z4YjA9X-P6jU-g-HhIL-S_VAT75xTjNZGFiDsw-
La1',\n
auth_endpoint='https://iam.bluemix.net/oidc/token/',
\n   endpoint_url='https://s3-api.us-
geo.objectstorage.softlayer.net'\n   ),\n
location=S3Location(\n
bucket='heartfailureprediction-donotdelete-pr-
q99lnevqgs44yp',\n
path='patientdataset.csv'\n
))\n]\ntraining_result_reference = DataConnection(\n
connection=S3Connection(\n   api_key='-
_z4YjA9X-P6jU-g-HhIL-S_VAT75xTjNZGFiDsw-La1',\n
auth_endpoint='https://iam.bluemix.net/oidc/token/',
\n   endpoint_url='https://s3-api.us-
geo.objectstorage.softlayer.net'\n   ),\n
location=S3Location(\n
bucket='heartfailureprediction-donotdelete-pr-
q99lnevqgs44yp',\n   path='auto_ml/13cc89e7-
56d3-464a-874d-cd835ca4570d/wml_data/0fe6a87e-a98c-
4705-96b2-18dd7620e926/data/automl',\n

model_location='auto_ml/13cc89e7-56d3-464a-874d-cd835ca4570d/wml_data/0fe6a87e-a98c-4705-96b2-18dd7620e926/data/automl/cognito_output/Pipeline1/model.pickle',\n
training_status='auto_ml/13cc89e7-56d3-464a-874d-cd835ca4570d/wml_data/0fe6a87e-a98c-4705-96b2-18dd7620e926/training-status.json'\n   ))"}, {"cell_type": "markdown", "metadata": {}, "source": "Following cell contains input parameters provided to run the AutoAI experiment in Watson Studio"}, {"cell_type": "code", "execution_count": null, "metadata": {"pycharm": {"name": "#%%\n"}}, "outputs": [], "source": "experiment_metadata = dict(\n   prediction_type='classification',\n prediction_column='HEARTFAILURE',\n test_size=0.1,\n   scoring='accuracy',\n project_id='f078ffdc-ac05-4688-be79-4c56c62f75f5',\n deployment_url='https://us-south.ml.cloud.ibm.com',\n   csv_separator=',',\n random_state=33,\n   excel_sheet=0,\n max_number_of_estimators=2,\n training_data_reference = training_data_reference,\n training_result_reference = training_result_reference)\n\npipeline_name='Pipeline_3'"}, {"cell_type": "markdown", "metadata": {"pycharm": {"name": "#%% md\n"}}, "source": "\n## Pipeline inspection\nIn this section you will get the trained pipeline model from the AutoAI experiment and inspect it. \nYou will see pipeline as a pythone code, graphically visualized and at the end, you will perform a local test.\n"}, {"cell_type": "markdown", "metadata": {"pycharm": {"name": "#%% md\n"}}, "source": "\n### Get historical optimizer instance\n\nThe next cell contains code for retrieving fitted optimizer."}, {"cell_type": "code", "execution_count": null, "metadata": {"pycharm": {"is_executing": false, "name": "#%%\n"}}, "outputs": [], "source": "from ibm_watson_machine_learning.experiment import AutoAI\n\noptimizer = AutoAI().runs.get_optimizer(metadata=experiment_metadata)"}, {"cell_type": "markdown", "metadata": {"pycharm": {"name": "#%% md\n"}}, "source": "\n### Get pipeline model\n\nThe 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'`."}, {"cell_type": "code", "execution_count": null, "metadata": {"pycharm": {"is_executing": false, "name": "#%%\n"}}, "outputs": [], "source": "pipeline_model = optimizer.get_pipeline(pipeline_name=pipeline_name)"}, {"cell_type": "markdown", "metadata": {"pycharm": {"name": "#%% md\n"}}, "source": "<a

id=\"preview_model_to_python_code\">\n###
Preview pipeline model as python code\nIn the next
cell, downloaded pipeline model could be previewed
as a python code. \nYou will be able to see what
exact steps are involved in model creation."},
{"cell_type": "code", "execution_count": null,
"metadata": {"pycharm": {"name": "#%%\n"}},
"outputs": [], "source":
"pipeline_model.pretty_print(combinators=False,
ipython_display=True)"}, {"cell_type": "markdown",
"metadata": {"pycharm": {"name": "#%% md\n"}},
"source": "\n###
Visualize pipeline model\n\nPreview pipeline model
stages as graph. Each node's name links to detailed
description of the stage.\n"}, {"cell_type": "code",
"execution_count": null, "metadata": {"pycharm":
{"is_executing": false, "name": "#%%\n"}},
"outputs": [], "source":
"pipeline_model.visualize()"}, {"cell_type":
"markdown", "metadata": {"pycharm": {"name": "#%%
md\n"}}, "source": "\n###
Read training data\n\nRetrieve training dataset from
AutoAI experiment as pandas DataFrame."},
{"cell_type": "code", "execution_count": null,
"metadata": {"pycharm": {"name": "#%%\n"}},
"outputs": [], "source": "train_df =
optimizer.get_data_connections()[0].read()\ntest_df
=
train_df.sample(n=5).drop([experiment_metadata['pred
iction_column']], axis=1)"}, {"cell_type":
"markdown", "metadata": {"pycharm": {"name": "#%%
md\n"}}, "source": "\n###
Test pipeline model locally\nYou can predict target
value using trained AutoAI model by calling
`predict()`."}, {"cell_type": "code",
"execution_count": null, "metadata": {"pycharm":
{"name": "#%%\n"}}, "outputs": [], "source": "y_pred
=
pipeline_model.predict(test_df.values)\nprint(y_pred
)"}, {"cell_type": "markdown", "metadata":
{"pycharm": {"name": "#%% md\n"}}, "source": "<a
id=\"refinery\">\n## Pipeline refinery and
testing (optional)\n\nIn this section you will learn
how to refine and retrain the best pipeline returned
by AutoAI.\nIt can be performed by:\n - modifying
pipeline definition source code\n - using
[lale](https://lale.readthedocs.io/en/latest/)
library for semi-automated data science\n\n**Note**:
In order to run this section change following cells
to 'code' cell."}, {"cell_type": "markdown",
"metadata": {}, "source": "<a
id=\"pipeline_definition\">\n### Pipeline
definition source code\nFollowing cell lets you
experiment with pipeline definition in python, e.g.
change steps parameters.\n\nIt will inject pipeline
definition to the next cell."}, {"cell_type": "raw",

"metadata": {"pycharm": {"name": "#%%\n"}}, "source": "pipeline_model.pretty_print(combinators=False, ipython_display='input')"}, {"cell_type": "markdown", "metadata": {"pycharm": {"name": "#%% md\n"}}, "source": "\n### Lale library\n\n**Note**: This is only an exemplary usage of lale package. You can import more different estimators to refine downloaded pipeline model."}, {"cell_type": "markdown", "metadata": {}, "source": "#### Import estimators"}, {"cell_type": "raw", "metadata": {"pycharm": {"is_executing": false, "name": "#%%\n"}}, "source": "from sklearn.linear_model import LogisticRegression as E1\nfrom sklearn.tree import DecisionTreeClassifier as E2\nfrom sklearn.neighbors import KNeighborsClassifier as E3\nfrom lale.lib.lale import Hyperopt\nfrom lale.operators import TrainedPipeline\nfrom lale import wrap_imported_operators\nfrom lale.helpers import import_from_sklearn_pipeline\nwrap_imported_operators()"}, {"cell_type": "markdown", "metadata": {"pycharm": {"name": "#%% md\n"}}, "source": "\n#### Pipeline decomposition and new definition\nIn this step the last stage from pipeline is removed."}, {"cell_type": "raw", "metadata": {"pycharm": {"name": "#%%\n"}}, "source": "prefix = pipeline_model.remove_last().freeze_trainable()\nprefix.visualize()"}, {"cell_type": "raw", "metadata": {"pycharm": {"is_executing": false, "name": "#%%\n"}}, "source": "new_pipeline = prefix >> (E1 | E2 | E3)\nnew_pipeline.visualize()"}, {"cell_type": "markdown", "metadata": {"pycharm": {"name": "#%% md\n"}}, "source": "\n#### New optimizer `hyperopt` configuration and training\n\nThis section can introduce other results than the original one and it should be used\nby more advanced users.\n\nNew pipeline is re-trained by passing train data to it and calling `fit` method.\n\nFollowing cell performs dataset split for refined pipeline model."}, {"cell_type": "raw", "metadata": {"pycharm": {"name": "#%%\n"}}, "source": "from sklearn.model_selection import train_test_split\n\ntrain_X = train_df.drop([experiment_metadata['prediction_column']], axis=1).values\ntrain_y = train_df[experiment_metadata['prediction_column']].values\n\ntrain_X, test_X, train_y, test_y = train_test_split(train_X, train_y, test_size=experiment_metadata['test_size'],\n stratify=train_y, random_state=experiment_metadata['random_state'])"}, {"cell_type": "raw", "metadata": {"pycharm": {"name": "#%%\n"}}, "source": "hyperopt =

Hyperopt(estimator=new_pipeline, cv=3, max_evals=20)\nfitted_hyperopt = hyperopt.fit(train_X, train_y)"}, {"cell_type": "raw", "metadata": {"pycharm": {"name": "#%%\n"}}, "source": "hyperopt_pipeline = fitted_hyperopt.get_pipeline()\nnew_pipeline = hyperopt_pipeline.export_to_sklearn_pipeline()"}, {"cell_type": "raw", "metadata": {"pycharm": {"name": "#%%\n"}}, "source": "prediction = new_pipeline.predict(test_X)"}, {"cell_type": "raw", "metadata": {"pycharm": {"name": "#%%\n"}}, "source": "from sklearn.metrics import accuracy_score\n\nscore = accuracy_score(y_true=test_y, y_pred=prediction)\nprint('accuracy_score: ', score)"}, {"cell_type": "markdown", "metadata": {"pycharm": {"name": "#%% md\n"}}, "source": "\n## Deploy and Score\n\nIn this section you will learn how to deploy and score pipeline model as webservice using WML instance."}, {"cell_type": "markdown", "metadata": {}, "source": "\n### Connection to WML\nAuthenticate the Watson Machine Learning service on IBM Cloud.\n\n**Tip**: Your Cloud API key can be generated by going to the [**Users** section of the Cloud console](https://cloud.ibm.com/iam#/users). 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.\n\n**Note:** You can also get service specific apikey by going to the [**Service IDs** section of the Cloud Console](https://cloud.ibm.com/iam/serviceids). From that page, click **Create**, then copy the created key and paste it below.\n\n**Action**: Enter your `api_key` in the following cell."}, {"cell_type": "code", "execution_count": null, "metadata": {"pycharm": {"name": "#%%\n"}}, "outputs": [], "source": "api_key = \"PUT_YOUR_API_KEY_HERE\"\n\nwml_credentials = {\n \"apikey\": api_key,\n \"url\": experiment_metadata[\"deployment_url\"]\n}"}, {"cell_type": "markdown", "metadata": {}, "source": "\n\n### Create deployment\n **Action**: If you want to deploy refined pipeline please change the `pipeline_model` to\n`new_pipeline`.\nIf you prefer you can also change the `deployment_name`.\nTo perform deployment please specify `target_space_id`\n"}, {"cell_type": "code", "execution_count": null, "metadata": {"pycharm": {"name": "#%%\n"}}, "outputs": [], "source": "target_space_id = \"PUT_YOUR_TARGET_SPACE_ID_HERE\"\n\nfrom ibm_watson_machine_learning.deployment import

WebService\nservice =
WebService(target_wml_credentials=wml_credentials,\n
target_space_id=target_space_id)\nservice.create(\nm
odel=pipeline_model,\nmetadata=experiment_metadata,\
ndeployment_name=f'{pipeline_name}_webservice'\n)"},
{"cell_type": "markdown", "metadata": {}, "source":
"Deployment object could be printed to show basic
information:"}, {"cell_type": "code",
"execution_count": null, "metadata": {"pycharm":
{"name": "#%%\n"}}, "outputs": [], "source":
"print(service)"}, {"cell_type": "markdown",
"metadata": {}, "source": "To be able to show all
available information about deployment use
`.get_params()` method:"}, {"cell_type": "code",
"execution_count": null, "metadata": {"pycharm":
{"name": "#%%\n"}}, "outputs": [], "source":
"service.get_params()"}, {"cell_type": "markdown",
"metadata": {"pycharm": {"name": "#%% md\n"}},
"source": "\n### Score
webservice\nYou can make scoring request by calling
`score()` on deployed pipeline."}, {"cell_type":
"code", "execution_count": null, "metadata":
{"pycharm": {"name": "#%%\n"}}, "outputs": [],
"source": "predictions =
service.score(payload=test_df)\npredictions"},
{"cell_type": "markdown", "metadata": {}, "source":
"If you want to work with the webservice in external
Python application you can retrieve the service
object by:\n - initialize service by:\n```\n service
=
WebService(target_wml_credentials=wml_credentials,\n
target_space_id=target_space_id)\n```\n - get
deployment_id by `service.list()` method\n - get
webservice object by `service.get('deployment_id')`
method\n\nAfter that you can call `service.score()`
method."}, {"cell_type": "markdown", "metadata":
{"pycharm": {"name": "#%% md\n"}}, "source": "<a
id=\"delete_deployment\">\n### Delete
deployment\n\nYou can delete an existing deployment
by calling `service.delete()`."}, {"cell_type":
"markdown", "metadata": {"pycharm": {"name": "#%%
md\n"}}, "source": "\n###
Authors\n\nLicensed Materials - Copyright \u00a9
2020 IBM. This notebook and its source code are
released under the terms of the ILAN License.\nUse,
duplication disclosure restricted by GSA ADP
Schedule Contract with IBM Corp.\n\n**Note:** The
auto-generated notebooks are subject to the
International License Agreement for Non-Warranted
Programs \n(or equivalent) and License Information
document for Watson Studio Auto-generated Notebook
(License Terms), \nsuch agreements located in the
link below. Specifically, the Source Components and
Sample Materials clause \nincluded in the License
Information document for Watson Studio Auto-
generated Notebook applies to the auto-generated

```
notebooks.  \n\nBy downloading, copying, accessing,
or otherwise using the materials, you agree to the
<a href=\"http://www14.software.ibm.com/cgi-
bin/weblap/lap.pl?li_formnum=L-AMCU-
BHU2B7&title=IBM%20Watson%20Studio%20Auto-
generated%20Notebook%20V2.1\">License Terms</a>
\n\n___"}], "metadata": {"kernelspec":
{"display_name": "Python 3", "language": "python",
"name": "python3"}, "language_info":
{"codemirror_mode": {"name": "ipython", "version":
3}, "file_extension": ".py", "mimetype": "text/x-
python", "name": "python", "nbconvert_exporter":
"python", "pygments_lexer": "ipython3", "version":
"3.6.10"}}, "nbformat": 4, "nbformat_minor": 2}
```

- JSON file:


```
[{"id":"1353fecd.371aa1","type":"ui_form","z":"c537d69
d.3cc648","name":"Heart Failure
Prediction","label":"Features for Heart Failure
Prediction","group":"6314b0ed.78bdb","order":1,"width"
:0,"height":0,"options":[{"label":"AVG. HEART
BPM","value":"AVGHEARTBEATSPERMIN","type":"number","re
quired":true,"rows":null},{"label":"PALPITATIONS /
DAY","value":"PALPITATIONSPERDAY","type":"number","req
uired":true,"rows":null},{"label":"CHOLESTEROL","value
":"CHOLESTEROL","type":"number","required":true,"rows"
:null},{"label":"BMI","value":"BMI","type":"number","r
equired":true,"rows":null},{"label":"AGE","value":"AGE
","type":"number","required":true,"rows":null},{"label
":"SEX","value":"SEX","type":"text","required":true,"r
ows":null},{"label":"FAMILY
HISTORY","value":"FAMILYHISTORY","type":"text","requir
ed":true,"rows":null},{"label":"SMOKER FOR THE LAST
5YRS","value":"SMOKERLAST5YRS","type":"text","required
":true,"rows":null},{"label":"EXERCISE (MINS /
WEEK)","value":"EXERCISEMINPERWEEK","type":"number","r
equired":true,"rows":null}],"formValue":{"AVGHEARTBEAT
SPERMIN":"","PALPITATIONSPERDAY":"","CHOLESTEROL":"","
BMI":"","AGE":"","SEX":"","FAMILYHISTORY":"","SMOKERLA
ST5YRS":"","EXERCISEMINPERWEEK":""},"payload":"","subm
it":"submit","cancel":"cancel","topic":"","x":170,"y":
300,"wires":[["f486bf6b.7e5e2","a982fc9d.061ad"]],"inf
o":"# `Dr. Deepa Madathil`\n## `VIT,
Vellore`"},{"id":"f486bf6b.7e5e2","type":"debug","z":"
c537d69d.3cc648","name":"","active":true,"tosidebar":t
rue,"console":false,"tostatus":false,"complete":"paylo
ad","targetType":"msg","statusVal":"","statusType":"au
to","x":290,"y":380,"wires":[]},{"id":"a982fc9d.061ad"
,"type":"function","z":"c537d69d.3cc648","name":"pre-
token","func":"global.set(\"AVGHEARTBEATSPERMIN\",msg.
payload.AVGHEARTBEATSPERMIN)\nglobal.set(\"PALPITATION
SPERDAY\",msg.payload.PALPITATIONSPERDAY)\nglobal.set(
\"CHOLESTEROL\",msg.payload.CHOLESTEROL)\nglobal.set(\
```

"BMI\",msg.payload.BMI)\nglobal.set(\"AGE\",msg.payloa
d.AGE)\nglobal.set(\"SEX\",msg.payload.SEX)\nglobal.se
t(\"FAMILYHISTORY\",msg.payload.FAMILYHISTORY)\nglobal
.set(\"SMOKERLAST5YRS\",msg.payload.SMOKERLAST5YRS)\ng
lobal.set(\"EXERCISEMINPERWEEK\",msg.payload.EXERCISEM
INPERWEEK)\nvar
apikey=\"0AH_ucPLKAk4R0H77A7uzt7cvXdhBALP9DPvmKFKIQN4\
";\nmsg.headers={\"content-type\":\"application/x-www-
form-
urlencoded\"}\nmsg.payload={\"grant_type\":\"urn:ibm:p
arams:oauth:grant-
type:apikey\",\"apikey\":apikey}\nreturn
msg;\n","outputs":1,"noerr":0,"initialize":"","finaliz
e":"","x":300,"y":240,"wires":[["864c19b7.0a9db8"]]},{
"id":"864c19b7.0a9db8","type":"http
request","z":"c537d69d.3cc648","name":"","method":"POS
T","ret":"obj","paytoqs":"ignore","url":"https://iam.c
loud.ibm.com/identity/token","tls":"","persist":false,
"proxy":"","authType":"","x":452.00000381469727,"y":28
4.00000286102295,"wires":[["5af7f0eb.170c1","dd9fca02.
6afa58"]]},{"id":"5af7f0eb.170c1","type":"debug","z":"
c537d69d.3cc648","name":"","active":true,"tosidebar":t
rue,"console":false,"tostatus":false,"complete":"paylo
ad","targetType":"msg","statusVal":"","statusType":"au
to","x":640,"y":340,"wires":[]},{"id":"dd9fca02.6afa58
","type":"function","z":"c537d69d.3cc648","name":"Pre
Prediction","func":"var AVGHEARTBEATSPERMIN =
global.get(\"AVGHEARTBEATSPERMIN\")\nvar
PALPITATIONSPERDAY =
global.get(\"PALPITATIONSPERDAY\")\nvar CHOLESTEROL =
global.get(\"CHOLESTEROL\")\nvar BMI =
global.get(\"BMI\")\nvar AGE =
global.get(\"AGE\")\nvar SEX =
global.get(\"SEX\")\nvar FAMILYHISTORY =
global.get(\"FAMILYHISTORY\")\nvar SMOKERLAST5YRS =
global.get(\"SMOKERLAST5YRS\")\nvar EXERCISEMINPERWEEK
= global.get(\"EXERCISEMINPERWEEK\")\nvar
token=msg.payload.access_token\nmsg.headers={'Content-
Type': 'application/json',\"Authorization\":\"Bearer
\"+token,\"Accept\":\"application/json\"}\nmsg.payload
={\"input_data\":[{\"fields\":[\"AVGHEARTBEATSPERMIN\"
,\"PALPITATIONSPERDAY\",\"CHOLESTEROL\",\"BMI\",\"AGE\
",\"SEX\",\"FAMILYHISTORY\",\"SMOKERLAST5YRS\",\"EXERC
ISEMINPERWEEK\"],\"values\":[[AVGHEARTBEATSPERMIN,PALP
ITATIONSPERDAY,CHOLESTEROL,BMI,AGE,SEX,FAMILYHISTORY,S
MOKERLAST5YRS,EXERCISEMINPERWEEK]]}]}\nreturn
msg;","outputs":1,"noerr":0,"initialize":"","finalize"
:"","x":644.0000076293945,"y":236.0000021457672,"wires
":[["777e95e0.b65a7c"]]},{"id":"9bc41a62.ae5db8","type
":"debug","z":"c537d69d.3cc648","name":"","active":tru
e,"tosidebar":true,"console":false,"tostatus":false,"c
omplete":"payload","targetType":"msg","statusVal":"","
statusType":"auto","x":1022.9999914169312,"y":344.0000
0381469727,"wires":[]},{"id":"b837f8a9.61caa8","type":
"inject","z":"c537d69d.3cc648","name":"","props":[{"p"
:"payload"},{"p":"topic","vt":"str"}],"repeat":"","cro

ntab":"","once":false,"onceDelay":0.1,"topic":"","payload":"","payloadType":"date","x":170,"y":140,"wires":[["a982fc9d.061ad"]]},{"id":"6d78da9d.a8da24","type":"debug","z":"c537d69d.3cc648","name":"","active":true,"tosidebar":true,"console":false,"tostatus":false,"complete":"payload","targetType":"msg","statusVal":"","statusType":"auto","x":853.9999885559082,"y":127.0000057220459,"wires":[]},{"id":"4e805938.4fa318","type":"function","z":"c537d69d.3cc648","name":"Parsing","func":"msg.payload=msg.payload.predictions[0].values[0][0]\nreturn
msg;","outputs":1,"noerr":0,"initialize":"","finalize":"","x":699.6000595092773,"y":168.00000190734863,"wires":[["6d78da9d.a8da24","21747005.f8501"]]},{"id":"21747005.f8501","type":"ui_text","z":"c537d69d.3cc648","group":"6314b0ed.78bdb","order":2,"width":0,"height":0,"name":"","label":"Heart Failure
Prediction","format":"{{msg.payload}}","layout":"row-spread","x":867.6000366210938,"y":401.20001220703125,"wires":[]},{"id":"777e95e0.b65a7c","type":"http
request","z":"c537d69d.3cc648","name":"","method":"POST","ret":"obj","paytoqs":"ignore","url":"https://us-south.ml.cloud.ibm.com/ml/v4/deployments/b5f155f7-ce00-4ff8-96a4-4ff388645fe9/predictions?version=2020-09-01","tls":"","persist":false,"proxy":"","authType":"","x":870,"y":260,"wires":[["9bc41a62.ae5db8","4e805938.4fa318"]]},{"id":"6314b0ed.78bdb","type":"ui_group","z":"","name":"CVD - Heart Failure
Prediction","tab":"34222134.6dc82e","order":2,"disp":true,"width":"6","collapse":false},{"id":"34222134.6dc82e","type":"ui_tab","z":"","name":"Home","icon":"dashboard","disabled":false,"hidden":false}]