Project Report

Project title:

Predicting Life Expectancy using Machine Learning

Project done by,

Harisankar M

TKMCE

1. Introduction:

1.1Overview: Life expectancy is a statistical measure of the average time a human being is expected to live. This project is about building a model while will consider historical data from a time period of 2000 to 2015 for all the countries. The model trained in factors like Regional variations, Economic Circumstances, Sex Differences, Mental Illnesses, Physical Illnesses, Education, Year of their birth and other demographic factors. This project will be helpful in predicting the life expectancy of the countrymen so that the preventive measures can be taken accordingly to save them. The project will also prove helpful in predicting the other crucial factors such as the effect and rate of alcohol intake, effects of GDP and etc.

1.2Purpose and working: The sole purpose of this project is to predict the life expectancy of a person considering the various crucial factors. The project will be helpful in improving the health condition of the country and give insights about some crucial factors such as Alcohol intake, GDP growth, schooling, adult mortality, total and cost expenditure and etc. The project uses a Random Forest which is a classification algorithm consisting of many decisions trees. It uses bagging and feature randomness when building each individual tree to try to create an uncorrelated forest of trees whose prediction by committee is more accurate than that of any individual tree. The dataset used or the training of the model was downloaded from kaggle.com and Python is used to write the code for machine learning model.

The project is developed using various IBM services mentioned below:-

1) IBM Node-Red: IBM Node-Red was used to create the UI interface for the machine learning model which will be helpful for the user to input their own values.
2) IBM Watson: IBM Watson is one of the most popular and useful services provided by the IBM which allows users to create their own ML model along with the help of IBM Watson Machine Learning.
3) IBM Auto AI experiment: It is another unique services provided by IBM which helps us to create the ML model without the use of python and coding.

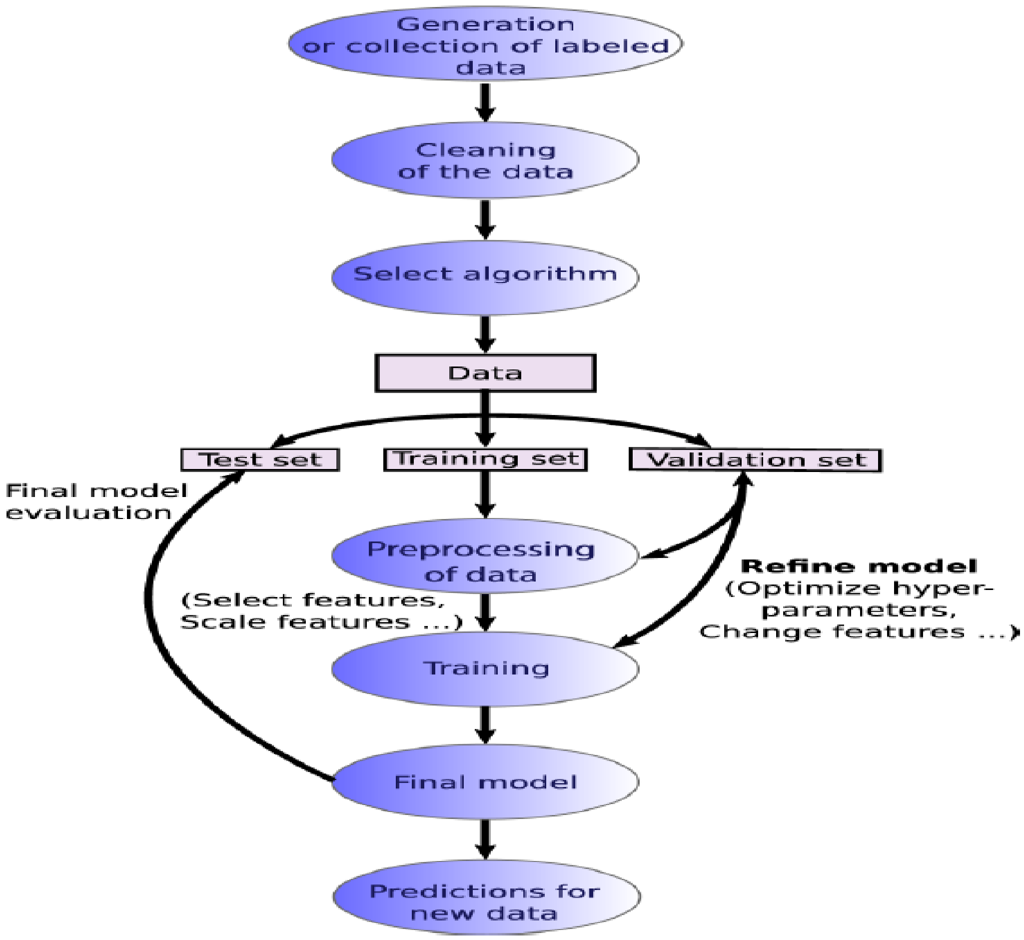
2. Literature Survey:

There are so many organizations that are making research in the prediction of life expectancy. Many research papers dealing with the creation of this model under many algorithms such as Machine Learning, Deep learning and programming languages such as Python and Java script.

2.1. Existing Problem The World Health Organization (WHO) began producing annual life tables for all Member States in 1999. These life tables are a basic input to all WHO estimates of global, regional and country-level patterns and trends in all-cause and cause-specific mortality. After the publication of life tables for years to 2009 in the 2011 edition of World Health Statistics, WHO has shifted to a two year cycle for the updating of life tables for all Member States. Even still the model is not really updated in every fields. WHO applies standard methods to the analysis of Member State data to ensure comparability of estimates across countries. This will inevitably result in differences for some Member States with official estimates for quantities such as life expectancy, where a variety of different projection methods and other methods are used. **2.2**. Proposed Solution So many people were expecting to use a model of life expectancy prediction. In order to that, many institutions and companies are leading their team to build that model. In my project, I have proposed a solution to predict the life expectancy using machine learning. Machine Learning is the process of training the computer to think and decide solutions like human. The reason why I have chosen this architecture was only with the help of Machine Learning, deep understanding of the data and an ability to create a model can be done. Design a Regression model to predict life expectancy ratio of a given country based on some features provided such as year, GDP (gross domestic product), education, alcohol intake of people in the country, expenditure on healthcare system and some specific disease related deaths that happened in the country.

3.Theroritical Analysis

3.1 Block diagram



3.2Project Requirements

The given project requires a dataset containing the details of life expectancy rates of different countries based on many parameters. From the parameters we can identify how much life expectancy rate depends on a given parameter and identify the life expectancy rate of a country based on the values of its parameters.

3.2.1Functional Requirements

- The functional requirements includes :
- Analysis of the given dataset and to clearly identify which of the parameters are needed to consider.
- Train the model according to the given dataset to predict the life expectancy rate.
- Adjust the hyperparameters to get the maximum accuracy for training and validation.
- Predict the life expectancy rate of the given country for the given parameters.

3.2.2 Technical Requirements

1. The technical requirements includes:
2. ibm cloud
3. ibm Watson

3.2.3The software requirements include:

- IBM cloud services
- IBM Watson services
- IBM Watson Studio
- IBM AutoAI experiment
- IBM Node-Red application
- SmartInternz Project Workspace
- Jupyter Notebook
- Github
- Slack
- Zoho document writer

4. Experimental Investigation

Life Expectancy Dataset:

The dataset used is a life expectancy dataset released by the World Health Organization.

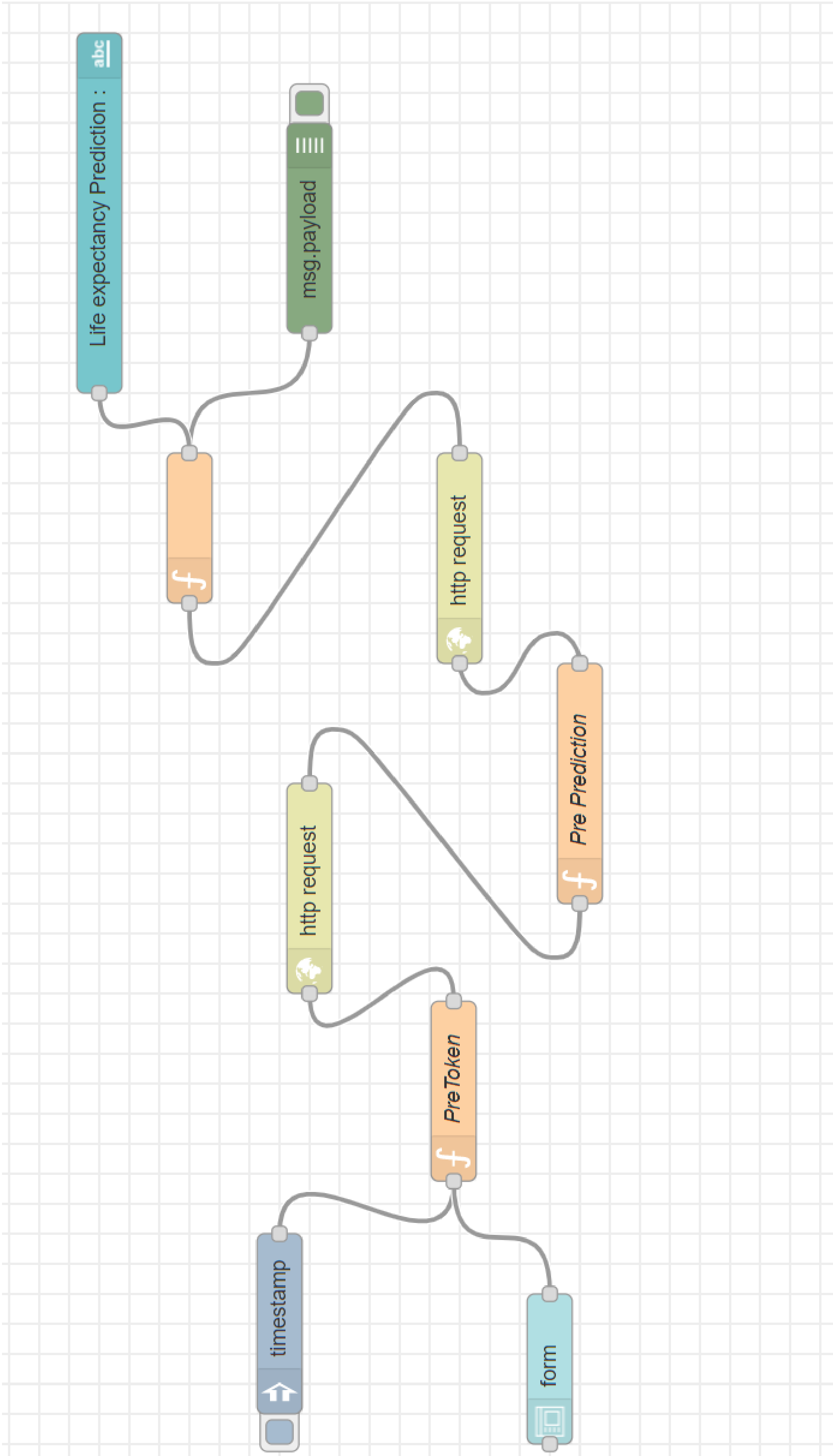
The data set has the following features:

The data is saved as a csv file as LifeExpectancy.csv and it is read and stored in the life data variable. The Year column is dropped as it will not be used in the analysis. The first 5 rows are shown below. The data contains 21 columns and 2938 rows with the header row. The table contains data about:

- Countries
- Status
- Life Expectancy
- Adult Mortality
- Alcohol
- percentage expenditure
- Hepatitis B
- Measles
- BMI
- under-five deaths
- Polio
- Total expenditure
- Diphtheria
- HIV/AIDS
- GDP
- Population
- thinness 1-19 years
- thinness 5-9 years
- Income composition of resources
- Schooling

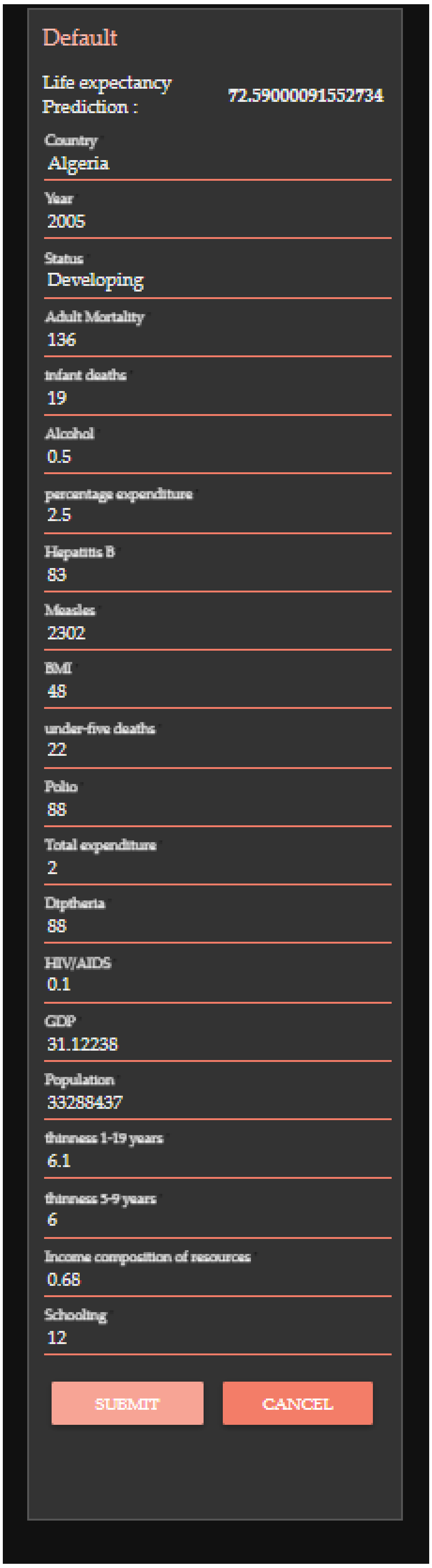
5.Flowchart

To integrate the ML model with the UI, we would be using the Node Red functionality provided by the IBM Watson Studio. To design the UI, we need to import the flow of the UI. Once, we have setup the flow, we need to integrate the ML model with it. To integrate the ML Model with it we need to access the endpoint URL of our ML Model. Components of the flow are: Form: The form contains all the elements of the UI. All the labels are associated with a variable.


Life expectancy Prediction :
msg.payload
http request
Pre Prediction
http request
PreToken
timestamp
form

6.Result

The user-friendly Graphical User interface is shown in Figure below This GUI is connected to the trained machine learning model present in the backend (IBM Watson notebook).



7.ADVANTAGES AND DISADVANTAGES:

7.1 Advantages:

- Easy process of unstructured data
- Fills human limitations
- Handle enormous quantities of data
- Easy for users to interact with the model via UI
- User-friendly
- Easy to build and deploy
- Requires less storage space

7.2. Disadvantages:

- IBM Cloud is only available in English
- Requires Internet Connection
- Error in data can result in wrong prediction
- Accuracy is not 100%.

8. Applications

- Personalized Life Expectancy: Individuals can predict their own life expectancy by inputting values in the corresponding fields. This could help make people more aware of their general health, and its improvement or deterioration over time. This may motivate them to make healthier lifestyle choices.
- Government: It could help the government bodies take appropriate measures to control the population growth and also direct the utilization of the increase in human resources and skillset acquired by people over many years. Across countries, high life expectancy is associated with high income per capita. Increase in life expectancy also leads to an increase in the "manpower" of a country. The knowledge asset of a country increases with the number of individuals in a country.
- Health Sector: Based on the factors used to calculate life expectancy of an individual and the outcome, health care will be able to fund and provide better services to those with greater need.
- Insurance Companies: Insurance sector will be able to provide individualized services to people based on the life expectancy outcomes and factors.

9. Conclusion:

Thus, we have developed a model that will predict the life expectancy of a specific demographic region based on the inputs provided. Various factors have a significant impact on the life span such as Adult Mortality, Population, Under 5 Deaths, Thinness1-5 Years, Alcohol, HIV, Hepatitis B, GDP, Percentage Expenditure and many more. Users can interact with the system via a simple Graphical user interface which is in the form of a form with input spaces which the user needs to fill the inputs into and then press the "submit" button.

10. Future Scope:

As future scope, we can connect the model to the database which can predict the life Expectancy of not only human beings but also of the plants and different animals present on the earth. This will help us analyse the trends in the life span. A model with country wise bifurcation can be made, which will help to segregate the data demographically.

11.Bibliograpgy:

1. Dataset: https://www.kaggle.com/kumarajarshi/life-expectancy-who
2. IBM Tutorials: https://developer.ibm.com/tutorials/
3. GitHub link: https://github.com/SmartPracticeschool/llSPS-INT-1991-Predicting-Life-Expectancy-using-Machine-Learning.git

Appendix

A. Source code

IBM AutoAI-SDK Auto-Generated Notebook v1.12.2

Note: Notebook code generated using AutoAI will execute successfully. If code is modified or reordered,
there is no guarantee it will successfully execute. This pipeline is optimized for the original dataset. The pipeline may fail or produce sub-optimium results if used with different data. For different data, please consider returning to AutoAI Experiments to generate a new pipeline. Please read our documentation
for more information:
Cloud Platform
Before modifying the pipeline or trying to re-fit the pipeline, consider:
The notebook converts dataframes to numpy arrays before fitting the pipeline (a current restriction of the preprocessor pipeline). The known_values_list is passed by reference and populated with categorical values during fit of the preprocessing pipeline. Delete its members before re-fitting.

Notebook content

This notebook contains steps and code to demonstrate AutoAI pipeline. This notebook introduces commands for getting data,
pipeline model, model inspection and testing.

Some familiarity with Python is helpful. This notebook uses Python 3.

Notebook goals

- inspection of trained pipeline via graphical vizualization and source code preview.
- pipeline evaluation.
- pipeline deployment and webservice scoring

Contents

This notebook contains the following parts:

Setup

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

- `watson-machine-learning-client` uninstallation of the old client
- `watson-machine-learning-client-V4` installation
- `autoai-libs` installation/upgrade
- `lightgbm` or `xgboost` installation/downgrade if they are needed

In [1]:

```
!pip uninstall watson-machine-learning-client -y
Uninstalling watson-machine-learning-client-1.0.376:
  Successfully uninstalled watson-machine-learning-client-1.0.376
```

In [2]:

```
!pip install -U watson-machine-learning-client-V4
Collecting watson-machine-learning-client-V4
  Downloading https://files.pythonhosted.org/packages/cf/9a/cd255fb8e3a67a688c36748233eb57ac4a4331fa574ef678c3cd69e14e44/watson_machine_learning_client_V4-1.0.99-py3-none-any.whl (1.2MB)
     |████████████████████████████████| 1.2MB 7.7MB/s eta 0:00:01
Requirement already satisfied, skipping upgrade: tabulate in /opt/conda/envs/Python36/lib/python3.6/site-packages (from watson-machine-learning-client-V4) (0.8.2)
Requirement already satisfied, skipping upgrade: urllib3 in /opt/conda/envs/Python36/lib/python3.6/site-packages (from watson-machine-learning-client-V4) (1.24.1)
Requirement already satisfied, skipping upgrade: certifi in /opt/conda/envs/Python36/lib/python3.6/site-packages (from watson-machine-learning-client-V4) (2020.4.5.1)
Requirement already satisfied, skipping upgrade: lomond in /opt/conda/envs/Python36/lib/python3.6/site-packages (from watson-machine-learning-client-V4) (0.3.3)
```

```
Collecting ibm-cos-sdk==2.6.0 (from watson-machine-learning-client-V4)
  Downloading https://files.pythonhosted.org/packages/6f/91/86b2816c7b77d81
6b03a1ad6cf7db4b1f67556af395d5b93fdae6086c933/ibm-cos-sdk-2.6.0.tar.gz (53k
B)
     |████████████████████████████████| 61kB 22.1MB/s eta 0:00:01
Requirement already satisfied, skipping upgrade: requests in /opt/conda/env
s/Python36/lib/python3.6/site-packages (from watson-machine-learning-client
-V4) (2.21.0)
Requirement already satisfied, skipping upgrade: pandas<=0.25.3 in /opt/con
da/envs/Python36/lib/python3.6/site-packages (from watson-machine-learning-
client-V4) (0.24.1)
Requirement already satisfied, skipping upgrade: six>=1.10.0 in /opt/conda/
envs/Python36/lib/python3.6/site-packages (from lomond->watson-machine-lear
ning-client-V4) (1.12.0)
Collecting ibm-cos-sdk-core==2.6.0 (from ibm-cos-sdk==2.6.0->watson-machine
-learning-client-V4)
  Downloading https://files.pythonhosted.org/packages/ea/c1/c823507c472bf88
dbd045445df6850744111d34fd218c6ea3b9c9bde2cfe/ibm-cos-sdk-core-2.6.0.tar.gz
(763kB)
     |████████████████████████████████| 768kB 31.4MB/s eta 0:00:01
Collecting ibm-cos-sdk-s3transfer==2.6.0 (from ibm-cos-sdk==2.6.0->watson-m
achine-learning-client-V4)
  Downloading https://files.pythonhosted.org/packages/6f/92/682a28b99777a3f
dc65e6d5641ed7e1ca470d0eab3bb2826cc30c6b60e21/ibm-cos-sdk-s3transfer-2.6.0.
tar.gz (221kB)
     |████████████████████████████████| 225kB 32.0MB/s eta 0:00:01
Requirement already satisfied, skipping upgrade: jmespath<1.0.0,>=0.7.1 in
/opt/conda/envs/Python36/lib/python3.6/site-packages (from ibm-cos-sdk==2.6
.0->watson-machine-learning-client-V4) (0.9.3)
Requirement already satisfied, skipping upgrade: chardet<3.1.0,>=3.0.2 in /
opt/conda/envs/Python36/lib/python3.6/site-packages (from requests->watson-
machine-learning-client-V4) (3.0.4)
Requirement already satisfied, skipping upgrade: idna<2.9,>=2.5 in /opt/con
da/envs/Python36/lib/python3.6/site-packages (from requests->watson-machine
-learning-client-V4) (2.8)
Requirement already satisfied, skipping upgrade: numpy>=1.12.0 in /opt/cond
a/envs/Python36/lib/python3.6/site-packages (from pandas<=0.25.3->watson-ma
chine-learning-client-V4) (1.15.4)
Requirement already satisfied, skipping upgrade: python-dateutil>=2.5.0 in
/opt/conda/envs/Python36/lib/python3.6/site-packages (from pandas<=0.25.3->
watson-machine-learning-client-V4) (2.7.5)
Requirement already satisfied, skipping upgrade: pytz>=2011k in /opt/conda/
envs/Python36/lib/python3.6/site-packages (from pandas<=0.25.3->watson-mach
ine-learning-client-V4) (2018.9)
Requirement already satisfied, skipping upgrade: docutils<0.16,>=0.10 in /o
pt/conda/envs/Python36/lib/python3.6/site-packages (from ibm-cos-sdk-core==
2.6.0->ibm-cos-sdk==2.6.0->watson-machine-learning-client-V4) (0.14)
```

```
Building wheels for collected packages: ibm-cos-sdk, ibm-cos-sdk-core, ibm-
cos-sdk-s3transfer
  Building wheel for ibm-cos-sdk (setup.py) ... done
  Stored in directory: /home/dsxuser/.cache/pip/wheels/37/9c/c4/a2c610ccb87
7d37c2cb87a5bfe55845fecffd6bb01bcd5e9d5
  Building wheel for ibm-cos-sdk-core (setup.py) ... done
  Stored in directory: /home/dsxuser/.cache/pip/wheels/75/93/e6/23071b2c037
147a0993d34b64a03e51abca84435fc9cd6a278
  Building wheel for ibm-cos-sdk-s3transfer (setup.py) ... done
  Stored in directory: /home/dsxuser/.cache/pip/wheels/23/d9/d7/43fd95b014e
ed89466154d8373bf4cffbb3d972de7841e213c
Successfully built ibm-cos-sdk ibm-cos-sdk-core ibm-cos-sdk-s3transfer
Installing collected packages: ibm-cos-sdk-core, ibm-cos-sdk-s3transfer, ib
m-cos-sdk, watson-machine-learning-client-V4
  Found existing installation: ibm-cos-sdk-core 2.4.3
    Uninstalling ibm-cos-sdk-core-2.4.3:
      Successfully uninstalled ibm-cos-sdk-core-2.4.3
  Found existing installation: ibm-cos-sdk-s3transfer 2.4.3
    Uninstalling ibm-cos-sdk-s3transfer-2.4.3:
      Successfully uninstalled ibm-cos-sdk-s3transfer-2.4.3
  Found existing installation: ibm-cos-sdk 2.4.3
    Uninstalling ibm-cos-sdk-2.4.3:
      Successfully uninstalled ibm-cos-sdk-2.4.3
Successfully installed ibm-cos-sdk-2.6.0 ibm-cos-sdk-core-2.6.0 ibm-cos-sdk
-s3transfer-2.6.0 watson-machine-learning-client-V4-1.0.99
```

In [3]:

```
!pip install -U autoai-libs
Collecting autoai-libs
  Downloading https://files.pythonhosted.org/packages/39/cb/4144b9ee74fcb05
8cea934478c87f2444cfa7b38a01396507f1a417030cb/autoai_libs-1.10.12-37-cp36-c
p36m-manylinux1_x86_64.whl (4.3MB)
     |████████████████████████████████| 4.3MB 5.2MB/s eta 0:00:01
Collecting pandas<1.0.0,>=0.24.2 (from autoai-libs)
  Downloading https://files.pythonhosted.org/packages/52/3f/f6a428599e0d449
7e1595030965b5ba455fd8ade6e977e3c819973c4b41d/pandas-0.25.3-cp36-cp36m-many
linux1_x86_64.whl (10.4MB)
     |████████████████████████████████| 10.4MB 34.7MB/s eta 0:00:01
Collecting category-encoders==2.1.0 (from autoai-libs)
  Downloading https://files.pythonhosted.org/packages/a0/52/c54191ad3782de6
33ea3d6ee3bb2837bda0cf3bc97644bb6375cf14150a0/category_encoders-2.1.0-py2.p
y3-none-any.whl (100kB)
     |████████████████████████████████| 102kB 26.1MB/s ta 0:00:01
Collecting numpy>=1.16.4 (from autoai-libs)
  Downloading https://files.pythonhosted.org/packages/b3/a9/b1bc4c935ed0637
66bce7d3e8c7b20bd52e515ff1c732b02caacf7918e5a/numpy-1.18.5-cp36-cp36m-manyl
inux1_x86_64.whl (20.1MB)
     |████████████████████████████████| 20.1MB 33.0MB/s eta 0:00:01
```

```
Requirement already satisfied, skipping upgrade: scikit-learn==0.20.3 in /o
pt/conda/envs/Python36/lib/python3.6/site-packages (from autoai-libs) (0.20
.3)
Requirement already satisfied, skipping upgrade: python-dateutil>=2.6.1 in
/opt/conda/envs/Python36/lib/python3.6/site-packages (from pandas<1.0.0,>=0
.24.2->autoai-libs) (2.7.5)
Requirement already satisfied, skipping upgrade: pytz>=2017.2 in /opt/conda
/envs/Python36/lib/python3.6/site-packages (from pandas<1.0.0,>=0.24.2->aut
oai-libs) (2018.9)
Requirement already satisfied, skipping upgrade: patsy>=0.4.1 in /opt/conda
/envs/Python36/lib/python3.6/site-packages (from category-encoders==2.1.0->
autoai-libs) (0.5.1)
Requirement already satisfied, skipping upgrade: scipy>=0.19.0 in /opt/cond
a/envs/Python36/lib/python3.6/site-packages (from category-encoders==2.1.0-
>autoai-libs) (1.2.0)
Requirement already satisfied, skipping upgrade: statsmodels>=0.6.1 in /opt
/conda/envs/Python36/lib/python3.6/site-packages (from category-encoders==2
.1.0->autoai-libs) (0.9.0)
Requirement already satisfied, skipping upgrade: six>=1.5 in /opt/conda/env
s/Python36/lib/python3.6/site-packages (from python-dateutil>=2.6.1->pandas
<1.0.0,>=0.24.2->autoai-libs) (1.12.0)
ERROR: tensorflow 1.13.1 requires tensorboard<1.14.0,>=1.13.0, which is not
installed.
Installing collected packages: numpy, pandas, category-encoders, autoai-lib
s
  Found existing installation: numpy 1.15.4
    Uninstalling numpy-1.15.4:
      Successfully uninstalled numpy-1.15.4
  Found existing installation: pandas 0.24.1
    Uninstalling pandas-0.24.1:
      Successfully uninstalled pandas-0.24.1
  Found existing installation: category-encoders 2.0.0
    Uninstalling category-encoders-2.0.0:
      Successfully uninstalled category-encoders-2.0.0
  Found existing installation: autoai-libs 1.10.5
    Uninstalling autoai-libs-1.10.5:
      Successfully uninstalled autoai-libs-1.10.5
Successfully installed autoai-libs-1.10.12 category-encoders-2.1.0 numpy-1.
18.5 pandas-0.25.3
```

AutoAI experiment metadata

This cell contains input parameters provided to run the AutoAI experiment in Watson Studio and COS credentials required to retrieve AutoAI pipeline.

In [5]:

```
from watson_machine_learning_client.helpers import DataConnection, S3Connec
tion, S3Location
```

```
experiment_metadata = dict(
   prediction_type='regression',
   prediction_column='Life expectancy ',
   test_size=0.1,
   scoring='neg_root_mean_squared_error',
   max_number_of_estimators=2,
   training_data_reference = [DataConnection(
        connection=S3Connection(
            api_key='C62mlxK8Hh2F7ISrs3sgMhuToOJClyLmO3Uxn3LNnAHn',
            auth_endpoint='https://iam.bluemix.net/oidc/token/',
            endpoint_url='https://s3.eu-geo.objectstorage.softlayer.net'
        ),
            location=S3Location(
            bucket='lifeexpectancy-donotdelete-pr-8zkwvbijbvnbq5',
            path='Life Expectancy Data.csv'
        ))
    ],
    training_result_reference = DataConnection(
        connection=S3Connection(
            api_key='C62mlxK8Hh2F7ISrs3sgMhuToOJClyLmO3Uxn3LNnAHn',
            auth_endpoint='https://iam.bluemix.net/oidc/token/',
            endpoint_url='https://s3.eu-geo.objectstorage.softlayer.net'
        ),
        location=S3Location(
            bucket='lifeexpectancy-donotdelete-pr-8zkwvbijbvnbq5',
            path='auto_ml/0b512f5f-2a82-4743-a739-889cf4732ba9/wml_data/361
f02d2-b483-472a-b381-46549869afa8/data/automl',
            model_location='auto_ml/0b512f5f-2a82-4743-a739-889cf4732ba9/wm
l_data/361f02d2-b483-472a-b381-46549869afa8/data/automl/cognito_output/Pipe
line1/model.pickle',
            training_status='auto_ml/0b512f5f-2a82-4743-a739-889cf4732ba9/w
ml_data/361f02d2-b483-472a-b381-46549869afa8/training-status.json'
        )
    ))

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 [6]:

```
from watson_machine_learning_client.experiment import AutoAI

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

```
Warning: To use AutoAI with xgboost estimators, you need to have xgboost 0.
90 installed.
Warning: To use AutoAI with lightgbm estimators, you need to have lightgbm
2.2.3 installed.
```

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 [7]:

```
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 [8]:

```
pipeline_model.pretty_print(combinators=False, ipython_display=True)
from lale.lib.autoai_libs import NumpyColumnSelector
from lale.lib.autoai_libs import CompressStrings
from lale.lib.autoai_libs import NumpyReplaceMissingValues
from lale.lib.autoai_libs import NumpyReplaceUnknownValues
from lale.lib.autoai_libs import boolean2float
from lale.lib.autoai_libs import CatImputer
from lale.lib.autoai_libs import CatEncoder
import numpy as np
from lale.lib.autoai_libs import float32_transform
from lale.operators import make_pipeline
from lale.lib.autoai_libs import FloatStr2Float
from lale.lib.autoai_libs import NumImputer
from lale.lib.autoai_libs import OptStandardScaler
from lale.operators import make_union
from lale.lib.autoai_libs import NumpyPermuteArray
from lale.lib.autoai_libs import TA2
import autoai_libs.utils.fc_methods
from lale.lib.autoai_libs import FS1
from lale.lib.autoai_libs import TA1
from lale.lib.sklearn import ExtraTreesRegressor

numpy_column_selector_0 = NumpyColumnSelector(columns=[0, 1, 2, 7, 11,
13])
compress_strings = CompressStrings(compress_type='hash', dtypes_list=['
char_str', 'int_num', 'char_str', 'float_int_num', 'float_int_num', 'fl
oat_int_num'], missing_values_reference_list=['', '-', '?', float('nan'
)], misslist_list=[[], [], [], [float('nan')], [float('nan')], [float('
nan')]])
numpy_replace_missing_values_0 = NumpyReplaceMissingValues(filling_valu
es=float('nan'), missing_values=[float('nan')])
numpy_replace_unknown_values = NumpyReplaceUnknownValues(filling_values
=float('nan'), filling_values_list=[float('nan'), float('nan'), float('
nan'), 100001, 100001, 100001], known_values_list=[[1470012248565269748
322780651633380610390, 2606993433293434202502634814624962044760, 11611833
71344515775493411338524259586010, 4675113895347093777217589166948697530
3, 329516381543508488967110039675080977027, 707333284213987214973227661
43577833507, 212739582897968815372252009904290618533, 90734833563168770
0647559265365332851010, 20279788233570032589745791750316665578400, 15082050
00985104818134154273839034261840, 1458775348853010492672159125673176434
3
```

1, 146042710492699183527689945925558913871, 329038285037642035448290014465150269573, 66569159131289518452679319382561400624, 134599758546830170946961007508765805273, 143673223417367409597806890422171484583, 43514824087351888171828013500097472829, 21059961804156297207093684778998036902, 306190046251434926279001033014007339831, 257386254094586331070623087230906766309, 200250612019127811152210464978673413843, 144662901196442033596517283743993697168, 88162585471654943964443952621227739261, 70896928478956771791708447884634167142, 2166848313003821530343725614844784907, 53840538962412723146725100117737849769, 53783522580166489497692045789692745562, 137433131769320511325160171555965548550, 9011956582324283455402195937934571568, 100867498053965421926683001219366720175, 90871431497902965602878565155781788327, 226182883875297231091414872587504708035, 298029156183361964131144349277717108303, 61669071338325875054847411897241946095, 231725186024138418821343173350662523706, 317953072729355616170708786288199146493, 88003295792748799025659169863539916864, 303846827209572410860542030980915489108, 117650741643591557067501287780038143451, 114382657829694160542291743577444141255, 68843515830881058478891148667609669513, 311265945460282885639496306569138650308, 66336313050842184927485246200945105096, 6278551754760904136723792210512775816, 264133828541403902365817922285691706663, 237672356660114685772687796104882206514, 88072156716790939851742967815210212361, 191655845835528033820508997679585417644, 135810150341533680846069152682895393814, 102836150724757230410725033869934678966, 301866387735507500246434251973708559904, 310276828594327150016809604736391170592, 217391883616977436792885859915168492233, 268223368701600857288201315803513880319, 158620802268026701334269447796294077650, 301204325272916423115124616267654497404, 113902216111633146684341394651838421184, 148167959092401686537001101821165870815, 4037797258022100113842494415863664385, 336187489452407039394791603109816227725, 301285256860312455785994867529604192476, 312173900060238937270521904129323847731, 288027277195630534052428568134696149071, 307384067113960046097974087241019802729, 142816848122656347891616720396877128120, 92639897809877101370166021321317973767, 197447874931603182353652347006218732046, 189153405801674659090813067540413446960, 60344653786708434424191955208148258744, 275547366651759454322811788273436150404, 245291117150675811910476227597757516612, 324534391971762113271501167168766491614, 332939221964977781373840866644320659361, 243990464967881250218096721123374832934, 166412482558944353431194533546670637099, 93418832550626031277006178804300582857, 100868194317926940154051177041401533500, 120809162883000100157782458277608403667, 9159657324535943692931225915386977603, 120069515845204766982513747293765138127, 21308572128415219572389994084010018174, 35708372757894372072386266404379628553, 111185081072713903471978902125976427675, 147055629285626963240164699118015634288, 164962277741819786224663597464248928300, 197516484511383225722150740031787721089, 112016882230000284908020292505580427074, 6939489273329044312136123259432836815, 125332411340882160702044993227052216001, 172336042347945209501542869303082370283, 213503579105731811632182799702211304277, 239759361309886661677188287055236569981, 310518635096297103991571280952035141839, 133154737170846526745897437786271419044, 154416530328912057507365665258369549472, 288469045950546730190193078847193098970, 8489659610198709691142666728949069345, 241570600237737391575071415960427031848, 149654325505539066448015487336620561262, 83815552068744036481632236478853198433, 130447722156717403042775871542217532557, 309775428472593936764999908740771513965, 194433639230741600743331401076929663747, 287105209433439742579762737349674202805, 10570423956451357204371014450649656715, 188392261694543715233845217008066282321, 313379486574562434335207500262629208144, 249118345345366939574908646319607340890, 104442926034210402647486596299287548193, 102735254899278162311633387159256797168, 221751288586957193672400042500503422193, 44527298243131759828592721352141504605, 140836680648

1778640321443748955475485555, 170055084650409142322787336947434585942, 2
21302440909767410133343183967811973171, 2620180144423681375725170420655
53566541, 79743781199220374265767606077128191664, 254125374157180781180
627978456850690728, 124301341133029451973017544367696533615, 2840869703
95620794029446821886845268390, 37840028582551292843841493013743517702,
246455200204341180287241602094154852031, 143453842366453184196547081438
629502439, 327575913974710642489356558011361228625, 1529447672648246361
64907421713899865093, 176501628211118575758228455305804789656, 15931368
2139237172887381305992317026546, 19743213616794516822058432183263147598
9, 311629718944900440174423894212659294477, 165862387931083018136327956
233088224545, 200992098551941933530619672502718259277, 1321128360465014
4315074545178289956239, 16601616560883432713160786478393331488, 1041661
2163005955661073717074093387062, 20934736196851481506884448148850036678
6, 279050323757951516155403396511033587716, 264459266514432677140218732
179794369611, 11137867482523875936741238848881502531, 22948376358970605
2603319193282544725320, 244856816748827073573575820596953224849, 204069
336107114946409550070958759494527, 637556697585320252140393497852503910
83, 280323114304267706123307379055557896439, 20114174970833660960382585
4380051921870, 92455594258272590489960197032004391800, 1816243274390450
60956820224446253696114, 189246905584152577259182499180590580, 21992470
1173169887305713158245281315340, 19419364610924018940777588033866280088
7, 266654382484335115128215294640512574736, 138080610306521808113457784
13029230717, 192066837354053545233955008532086692251, 31812420312441846
5236696385800557006393, 154551519819035509041533706282186084707, 133278
040923567948891376187569301318446, 329872229251628136616936033528845404
207, 267113292827397478038595255593918245163, 7817789045017950859085489
1703042317668, 159821081429796844717628326887647430719, 241470080325170
968009603514017433045249, 21534237697347747059400795433402612858, 33148
5374723644292946494024341588888803, 13488206953706492194337998636897954
158, 40481344139765028445355540856481018998, 29424074155464333193989421
1760785349296, 132374248062000760689647888900256269745, 285590841995412
959284881794170248825349, 45343673659765490367261512791957359538, 32679
0586085090417519839246359297909943, 11125941256967804096400462481105650
5841, 319179760373270718270012367276983387589, 190656778724682718055384
399764338191549, 274586769213526317171775513233049188730, 6501470247460
2264403928616790135130394, 41300458528712720009228826094222395870, 1559
01190345235079326433256810539957264, 2135275394955021483797222986816466
46713, 337664892250511449878491415602564465320, 14918789820365009415970
5085163227929822, 192946003755599889927910624585418472243, 151926378701
363846636626095694661523383, 234826980371610571253952539720242605948, 2
09109830453134779187508710962259670256], [2000, 2001, 2002, 2003, 2004,
2005, 2006, 2007, 2008, 2009, 2010, 2011, 2012, 2013, 2014, 2015], [316
899882848231090403546313428573395088, 269864576280947885342328752525343
33277], [1.0, 2.0, 4.0, 5.0, 6.0, 7.0, 8.0, 9.0, 11.0, 14.0, 15.0, 17.0
, 18.0, 21.0, 22.0, 23.0, 24.0, 25.0, 26.0, 27.0, 28.0, 29.0, 31.0, 32.
0, 33.0, 35.0, 36.0, 37.0, 38.0, 39.0, 41.0, 42.0, 43.0, 44.0, 45.0, 46
.0, 47.0, 48.0, 49.0, 51.0, 52.0, 53.0, 54.0, 55.0, 56.0, 57.0, 58.0, 5
9.0, 61.0, 62.0, 63.0, 64.0, 65.0, 66.0, 67.0, 68.0, 69.0, 71.0, 72.0,
73.0, 74.0, 75.0, 76.0, 77.0, 78.0, 79.0, 81.0, 82.0, 83.0, 84.0, 85.0,
86.0, 87.0, 88.0, 89.0, 91.0, 92.0, 93.0, 94.0, 95.0, 96.0, 97.0, 98.0,
99.0], [3.0, 4.0, 5.0, 6.0, 7.0, 8.0, 9.0, 23.0, 24.0, 26.0, 31.0, 32.0
, 33.0, 35.0, 36.0, 37.0, 38.0, 39.0, 41.0, 42.0, 43.0, 44.0, 45.0, 46.
0, 47.0, 48.0, 49.0, 51.0, 52.0, 53.0, 54.0, 55.0, 56.0, 57.0, 58.0, 59
.0, 61.0, 62.0, 63.0, 64.0, 65.0, 66.0, 67.0, 68.0, 69.0, 71.0, 72.0, 7
3.0, 74.0, 75.0, 76.0, 77.0, 78.0, 79.0, 81.0, 82.0, 83.0, 84.0, 85.0,
86.0, 87.0, 88.0, 89.0, 91.0, 92.0, 93.0, 94.0, 95.0, 96.0, 97.0, 98.0,
99.0], [2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0, 9.0, 19.0, 21.0, 23.0, 24.0,
25.0, 26.0, 27.0, 28.0, 29.0, 31.0, 32.0, 33.0, 34.0, 35.0, 36.0, 37.0,

```
38.0, 39.0, 41.0, 42.0, 43.0, 44.0, 45.0, 46.0, 47.0, 48.0, 49.0, 51.0,
52.0, 53.0, 54.0, 55.0, 56.0, 57.0, 58.0, 59.0, 61.0, 62.0, 63.0, 64.0,
65.0, 66.0, 67.0, 68.0, 69.0, 71.0, 72.0, 73.0, 74.0, 75.0, 76.0, 77.0,
78.0, 79.0, 81.0, 82.0, 83.0, 84.0, 85.0, 86.0, 87.0, 88.0, 89.0, 91.0,
92.0, 93.0, 94.0, 95.0, 96.0, 97.0, 98.0, 99.0]], missing_values_refere
nce_list=['', '-', '?', float('nan')])
cat_imputer = CatImputer(missing_values=float('nan'), sklearn_version_f
amily='20', strategy='most_frequent')
cat_encoder = CatEncoder(dtype=np.float64, handle_unknown='error', skle
arn_version_family='20')
pipeline_0 = make_pipeline(numpy_column_selector_0, compress_strings, n
umpy_replace_missing_values_0, numpy_replace_unknown_values, boolean2fl
oat(), cat_imputer, cat_encoder, float32_transform())
numpy_column_selector_1 = NumpyColumnSelector(columns=[3, 4, 5, 6, 8, 9
, 10, 12, 14, 15, 16, 17, 18, 19, 20])
float_str2_float = FloatStr2Float(dtypes_list=['float_int_num', 'int_nu
m', 'float_num', 'float_num', 'int_num', 'float_num', 'int_num', 'float
_num', 'float_num', 'float_num', 'float_num', 'float_num', 'float_num',
'float_num', 'float_num'], missing_values_reference_list=[float('nan')]
)
numpy_replace_missing_values_1 = NumpyReplaceMissingValues(filling_valu
es=float('nan'), missing_values=[float('nan')])
num_imputer = NumImputer(missing_values=float('nan'), strategy='median'
)
opt_standard_scaler = OptStandardScaler(num_scaler_copy=None, num_scale
r_with_mean=None, num_scaler_with_std=None, use_scaler_flag=False)
pipeline_1 = make_pipeline(numpy_column_selector_1, float_str2_float, n
umpy_replace_missing_values_1, num_imputer, opt_standard_scaler, float3
2_transform())
union = make_union(pipeline_0, pipeline_1)
numpy_permute_array = NumpyPermuteArray(axis=0, permutation_indices=[0,
1, 2, 7, 11, 13, 3, 4, 5, 6, 8, 9, 10, 12, 14, 15, 16, 17, 18, 19, 20])
ta2 = TA2(fun=np.add, name='sum', datatypes1=['intc', 'intp', 'int_', '
uint8', 'uint16', 'uint32', 'uint64', 'int8', 'int16', 'int32', 'int64'
, 'short', 'long', 'longlong', 'float16', 'float32', 'float64'], feat_c
onstraints1=[autoai_libs.utils.fc_methods.is_not_categorical], datatype
s2=['intc', 'intp', 'int_', 'uint8', 'uint16', 'uint32', 'uint64', 'int
8', 'int16', 'int32', 'int64', 'short', 'long', 'longlong', 'float16',
'float32', 'float64'], feat_constraints2=[autoai_libs.utils.fc_methods.
is_not_categorical], col_names=['Country', 'Year', 'Status', 'Adult Mor
tality', 'infant deaths', 'Alcohol', 'percentage expenditure', 'Hepatit
is B', 'Measles ', ' BMI ', 'under-five deaths ', 'Polio', 'Total expen
diture', 'Diphtheria ', ' HIV/AIDS', 'GDP', 'Population', ' thinness  1
-19 years', ' thinness 5-9 years', 'Income composition of resources', '
Schooling'], col_dtypes=[np.dtype('float32'), np.dtype('float32'), np.d
type('float32'), np.dtype('float32'), np.dtype('float32'), np.dtype('fl
oat32'), np.dtype('float32'), np.dtype('float32'), np.dtype('float32'),
np.dtype('float32'), np.dtype('float32'), np.dtype('float32'), np.dtype
('float32'), np.dtype('float32'), np.dtype('float32'), np.dtype('float3
2'), np.dtype('float32'), np.dtype('float32'), np.dtype('float32'), np.
dtype('float32'), np.dtype('float32')])
fs1_0 = FS1(cols_ids_must_keep=range(0, 21), additional_col_count_to_ke
ep=20, ptype='regression')
ta1 = TA1(fun=np.square, name='square', datatypes=['numeric'], feat_con
straints=[autoai_libs.utils.fc_methods.is_not_categorical], col_names=[
'Country', 'Year', 'Status', 'Adult Mortality', 'infant deaths', 'Alcoh
ol', 'percentage expenditure', 'Hepatitis B', 'Measles ', ' BMI ', 'und
er-five deaths ', 'Polio', 'Total expenditure', 'Diphtheria ', ' HIV/AI
```

```
DS', 'GDP', 'Population', ' thinness  1-19 years', ' thinness 5-9 years
', 'Income composition of resources', 'Schooling', 'sum(Country__Adult
Mortality)', 'sum(Adult Mortality__Country)', 'sum(Adult Mortality__Alc
ohol)', 'sum(Adult Mortality__Hepatitis B)', 'sum(Adult Mortality__Tota
l expenditure)', 'sum(Adult Mortality__ HIV/AIDS)', 'sum(Adult Mortalit
y__ thinness  1-19 years)', 'sum(Adult Mortality__ thinness 5-9 years)'
, 'sum(Adult Mortality__Income composition of resources)', 'sum(Adult M
ortality__Schooling)', 'sum(Alcohol__Adult Mortality)', 'sum(Hepatitis
B__Adult Mortality)', 'sum(Total expenditure__Adult Mortality)', 'sum(
HIV/AIDS__Adult Mortality)', 'sum( thinness  1-19 years__Adult Mortalit
y)', 'sum( thinness 5-9 years__Adult Mortality)', 'sum(Income compositi
on of resources__Adult Mortality)', 'sum(Income composition of resource
s__Schooling)', 'sum(Schooling__Adult Mortality)', 'sum(Schooling__Inco
me composition of resources)'], col_dtypes=[np.dtype('float32'), np.dty
pe('float32'), np.dtype('float32'), np.dtype('float32'), np.dtype('floa
t32'), np.dtype('float32'), np.dtype('float32'), np.dtype('float32'), n
p.dtype('float32'), np.dtype('float32'), np.dtype('float32'), np.dtype(
'float32'), np.dtype('float32'), np.dtype('float32'), np.dtype('float32
'), np.dtype('float32'), np.dtype('float32'), np.dtype('float32'), np.d
type('float32'), np.dtype('float32'), np.dtype('float32'), np.dtype('fl
oat32'), np.dtype('float32'), np.dtype('float32'), np.dtype('float32'),
np.dtype('float32'), np.dtype('float32'), np.dtype('float32'), np.dtype
('float32'), np.dtype('float32'), np.dtype('float32'), np.dtype('float3
2'), np.dtype('float32'), np.dtype('float32'), np.dtype('float32'), np.
dtype('float32'), np.dtype('float32'), np.dtype('float32'), np.dtype('f
loat32'), np.dtype('float32'), np.dtype('float32')])
fs1_1 = FS1(cols_ids_must_keep=range(0, 21), additional_col_count_to_ke
ep=20, ptype='regression')
extra_trees_regressor = ExtraTreesRegressor(bootstrap=True, n_jobs=2, o
ob_score=True, random_state=33)
pipeline = make_pipeline(union, numpy_permute_array, ta2, fs1_0, ta1, f
s1_1, extra_trees_regressor)
```

Visualize pipeline model

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

In [9]:

```
pipeline_model.visualize()
```

Read training and holdout data

Retrieve training dataset from AutoAI experiment as pandas DataFrame.

In [10]:

```
training_df, holdout_df = optimizer.get_data_connections()[0].read(with_hol
dout_split=True)

train_X = training_df.drop([experiment_metadata['prediction_column']], axis
=1).values
train_y = training_df[experiment_metadata['prediction_column']].values

test_X = holdout_df.drop([experiment_metadata['prediction_column']], axis=1
).values
```

```
y_true = holdout_df[experiment_metadata['prediction_column']].values
```

Test pipeline model locally

Note: you can chose the metric to evaluate the model by your own, this example contains only a basic scenario.

In [11]:

```
from sklearn.metrics import r2_score

predictions = pipeline_model.predict(test_X)
score = r2_score(y_true=y_true, y_pred=predictions)
print('r2_score: ', score)
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-11-afb8e4ee449d> in <module>
      2 
      3 predictions = pipeline_model.predict(test_X)
----> 4 score = r2_score(y_true=y_true, y_pred=predictions)
      5 print('r2_score: ', score)

/opt/conda/envs/Python36/lib/python3.6/site-packages/sklearn/metrics/regres
sion.py in r2_score(y_true, y_pred, sample_weight, multioutput)
    532     """
    533     y_type, y_true, y_pred, multioutput = _check_reg_targets(
--> 534         y_true, y_pred, multioutput)
    535     check_consistent_length(y_true, y_pred, sample_weight)
    536 

/opt/conda/envs/Python36/lib/python3.6/site-packages/sklearn/metrics/regres
sion.py in _check_reg_targets(y_true, y_pred, multioutput)
     74     """
     75     check_consistent_length(y_true, y_pred)
---> 76     y_true = check_array(y_true, ensure_2d=False)
     77     y_pred = check_array(y_pred, ensure_2d=False)
     78 

/opt/conda/envs/Python36/lib/python3.6/site-packages/sklearn/utils/validati
on.py in check_array(array, accept_sparse, accept_large_sparse, dtype, orde
r, copy, force_all_finite, ensure_2d, allow_nd, ensure_min_samples, ensure_
min_features, warn_on_dtype, estimator)
    571         if force_all_finite:
    572             _assert_all_finite(array,
--> 573                                allow_nan=force_all_finite == 'allow
-nan')
    574 
    575     shape_repr = _shape_repr(array.shape)

/opt/conda/envs/Python36/lib/python3.6/site-packages/sklearn/utils/validati
on.py in _assert_all_finite(X, allow_nan)
```

```
     54                     not allow_nan and not np.isfinite(X).all()):
     55                 type_err = 'infinity' if allow_nan else 'NaN, infinity'
---> 56                 raise ValueError(msg_err.format(type_err, X.dtype))
     57 
     58 

ValueError: Input contains NaN, infinity or a value too large for dtype('fl
oat64').
```

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 LinearRegression as E1 from sklearn.tree import DecisionTreeRegressor as E2 from sklearn.neighbors import KNeighborsRegressor 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.
hyperopt = Hyperopt(estimator=new_pipeline, cv=3, max_evals=20, scoring='r2') fitted_hyperopt = hyperopt.fit(train_X, train_y)hyperopt_pipeline = fitted_hyperopt.get_pipeline() new_pipeline = hyperopt_pipeline.export_to_sklearn_pipeline()predictions = new_pipeline.predict(train_X)predictions = new_pipeline.predict(test_X) refined_score = r2_score(y_true=y_true, y_pred=predictions) print('r2_score: ', score) print('refined_r2_score: ', refined_score)

Deploy and Score

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

Connect to WML client in order to create deployment

Action: Next you will need credentials for Watson Machine Learning and training run_id:

- go to Cloud catalog resources list
- click on Services and chose Machine Learning service. Once you are there
- click the **Service Credentials** link on the left side of the screen
- click to expand specific credentials name.
- copy and paste your WML credentials into the cell below

Take in mind that WML Service instance should be the same as used to generate this notebook.

In [12]:

```
wml_credentials = {
  "apikey": "Bl2ZIXYrgxgnuAF2FeFsRM8_WzHQU2ZMYbs3oE0fx_At",
  "iam_apikey_description": "Auto-generated for key a272c57b-748f-4970-b0ca-7f41a39f2a54",
  "iam_apikey_name": "wdp-writer",
  "iam_role_crn": "crn:v1:bluemix:public:iam::::serviceRole:Writer",
  "iam_serviceid_crn": "crn:v1:bluemix:public:iam-identity::a/3fe08c4cba0a47f5b384236f106e17fe::serviceid:ServiceId-face6cbf-8b03-480b-bc46-8dadcce81b9e",
  "instance_id": "4873d4fc-3a05-4017-9a92-08f87ba189ec",
  "url": "https://eu-gb.ml.cloud.ibm.com"
}
```

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`.

In [13]:

```
from watson_machine_learning_client.deployment import WebService

service = WebService(wml_credentials)

service.create(
    model=pipeline_model,
    metadata=experiment_metadata,
    deployment_name=f'{pipeline_name}_webservice'
)
```

```
Preparing an AutoAI Deployment...
Published model uid: b1abd955-bc66-4eba-826b-7ba796759ee7
Deploying model b1abd955-bc66-4eba-826b-7ba796759ee7 using V4 client.


########################################################################################

Synchronous deployment creation for uid: 'b1abd955-bc66-4eba-826b-7ba796759ee7' started
```

```
################################################################################
############


initializing.......
ready


--------------------------------------------------------------------------------
---------------------
Successfully finished deployment creation, deployment_uid='48853f05-699a-44
9a-87ab-1c009c845ecd'
--------------------------------------------------------------------------------
---------------------
```

Deployment object could be printed to show basic information:

In [14]:

```
print(service)
```

```
name: Pipeline_3_webservice, id: 48853f05-699a-449a-87ab-1c009c845ecd, scor
ing_url: https://eu-gb.ml.cloud.ibm.com/v4/deployments/48853f05-699a-449a-8
7ab-1c009c845ecd/predictions, asset_id: b1abd955-bc66-4eba-826b-7ba796759ee
7
```

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

In [15]:

```
service.get_params()
```

Out[15]:


```
{'metadata': {'parent': {'href': ''},
  'name': 'Pipeline_3_webservice',
  'guid': '48853f05-699a-449a-87ab-1c009c845ecd',
  'description': '',
  'id': '48853f05-699a-449a-87ab-1c009c845ecd',
  'modified_at': '2020-06-12T13:13:19.071Z',
  'created_at': '2020-06-12T13:13:19.071Z',
  'href': '/v4/deployments/48853f05-699a-449a-87ab-1c009c845ecd'},
 'entity': {'name': 'Pipeline_3_webservice',
  'custom': {},
  'online': {},
  'description': '',
  'status': {'state': 'ready',
   'online_url': {'url': 'https://eu-gb.ml.cloud.ibm.com/v4/deployments/488
53f05-699a-449a-87ab-1c009c845ecd/predictions'}},
  'asset': {'id': 'b1abd955-bc66-4eba-826b-7ba796759ee7',
   'href': '/v4/models/b1abd955-bc66-4eba-826b-7ba796759ee7?rev=300cab38-a9
5f-4789-9834-b16276cd3d22',
```

'rev': '300cab38-a95f-4789-9834-b16276cd3d22'}}}

Score webservice

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

In [16]:

```
predictions = service.score(payload=holdout_df.drop([experiment_metadata['prediction_column']], axis=1).iloc[:10])
predictions
```

Out[16]:

{'predictions': [{'fields': ['prediction'],
   'values': [[71.72000122070312],
    [67.33999862670899],
    [66.11000137329101],
    [75.26999893188477],
    [64.28000106811524],
    [56.96999969482422],
    [71.02000045776367],
    [53.86000022888184],
    [74.52000045776367],
    [72.13999786376954]]}]}

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

- initialize service by `service = WebService(wml_credentials)`
- 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()`.

Authors