

INTERNSHIP REPORT

(Life expectancy prediction using ML)

By

G. Thilakar Rajaa

(B.Tech CSE)

CSE17U026

IIIT Trichy

Table of Contents:

1.	Introduction 1.1 Overview 1.2 Purpose	3
2.	Literature Survey 2.1 Existing Problem 2.2 Proposed Solution	4
3.	Theoretical Analysis 3.1 Block diagram 3.2 Hardware / Software requirements 3.3 Application design	5
4.	Experimental Investigations	7
5.	Flowchart	9
6.	Result	10
7.	Advantages and disadvantages	11
8.	Applications	12
9.	Conclusion	13
10.	Future Scope	14
11.	Bibliography	15
	Appendix A. Source Code	16

Introduction:

1.1. Overview:

Life expectancy is a statistical measure of the average time a human being is expected to live. Life expectancy depends on various factors: Regional variations, Economic Circumstances, Sex Differences, Mental Illnesses, Physical Illnesses, Education, Year of their birth and other demographic factors. This problem statement provides a way to predict average life expectancy of people living in a country when various factors such as year, GDP, education, alcohol intake of people in the country, expenditure on healthcare system and some specific disease related deaths that happened in the country are given.

1.2 Purpose:

The purpose of this project is to use the current state-of-the-art methodologies in machine learning to implement a life expectancy predictor. The project is deployed on cloud to make it accessible at remote locations. It helps in predicting life expectancy of a person in a particular region and also to estimate important parameters contributing for the same by measuring correlation to look into possibilities of improving life expectancy.

Literature Survey:

2.1 Existing Problem:

Models in the literature were characterized by the use of life years rather than a risk of death, questionable approaches to comorbidity, implausible estimates, questionable recommendations, and poor clinical feasibility. We found tools based on applying an invalidated approach to assessing comorbidity to a clearly erroneous life expectancy table, or required a treatment decision be made before life expectancy could be calculated or gave implausible estimates, such as a substantial risk of prostate cancer specific mortality even for a highly comorbid 80 year old with Gleason 6 disease.

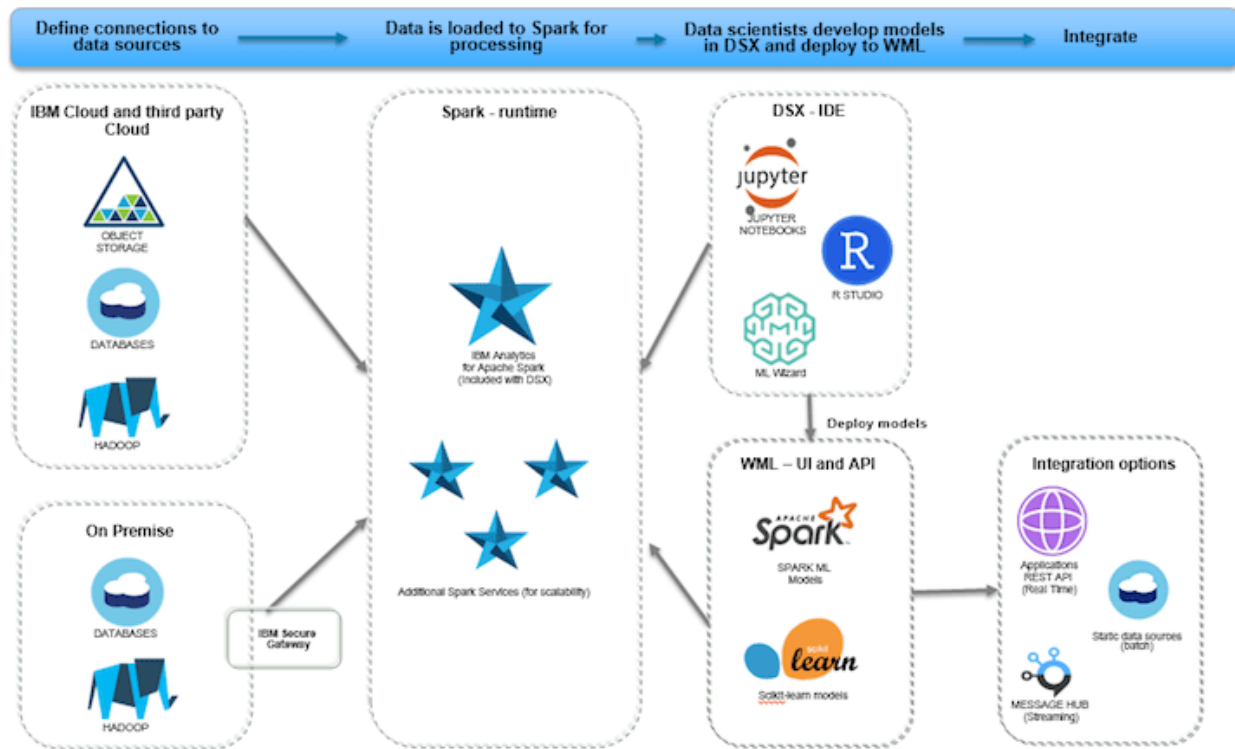
Understanding different parameters and predicting life expectancy is complicated using manual calculation. The development in computational abilities of modern computers can be utilized effectively to address this issue. In organizations such as WHO these data provide useful insights to incorporate relevant measures to increase life expectancy. This is possible only if we can predict them using available data and find most contributing factor for the same.

2.2 Proposed solution:

Develop a ML based solution to accurately predict life expectancy taking into account various factors that can be easily done using modern computers. Auto AI has been used to predict life expectancy using ExtraTrees Algorithm after hyper parameter tuning and feature engineering. This state-of-the-art model is saved and deployed as scoring endpoint in IBM cloud to integrate it with web UI. The cloud functionality is utilized to make the model accessible globally. Data transmission between different http requests is done with help of POST method to return JSON data. Bearer authentication is used to access the deployed model in cloud using tokens. It is proposed that the model is deployed in backend using Rest API. The user can input values of different parameters through UI and the deployed model shows the predicted value in UI.

Theoretical Analysis:

3.1 Block diagram:



3.2 Hardware and software requirements:

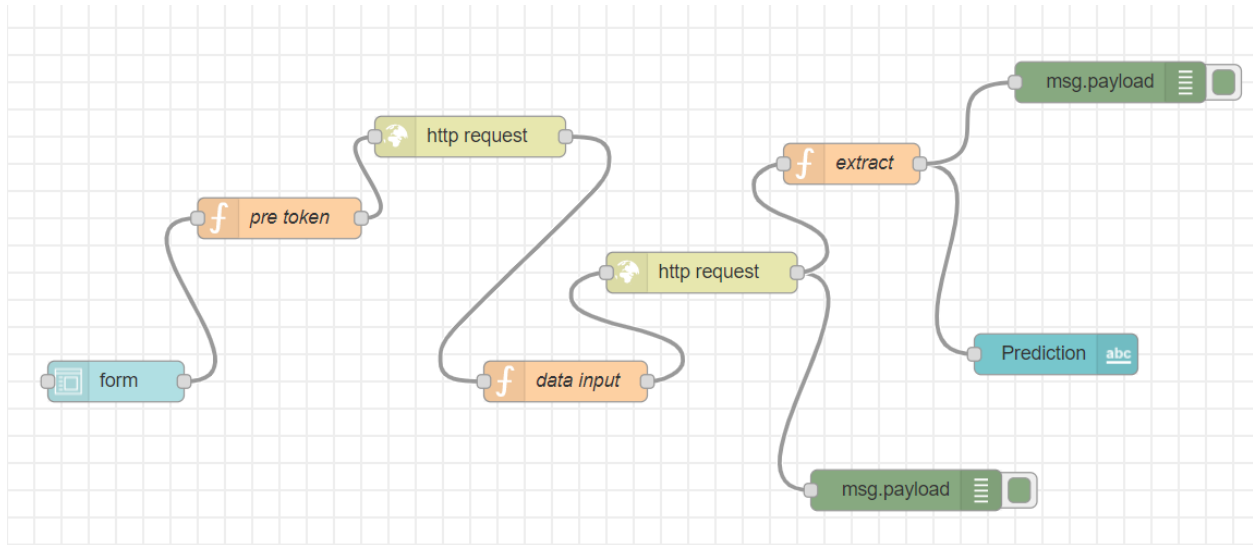
Hardware requirements:

1. A computer with good internet connection

Software requirements:

1. Microsoft edge / Google chrome
2. IBM cloud academic initiative account

3.3 Application Design:



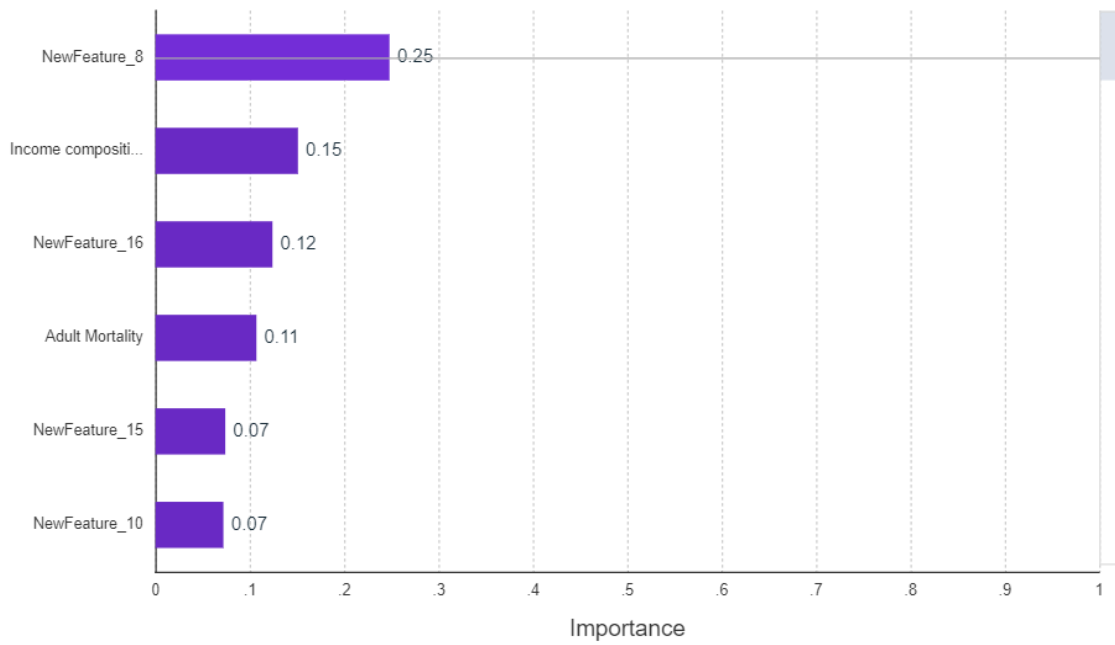
Experimental Investigation:

Among all the algorithms Extra Trees Regressor is most accurate and the pipeline is Hyper parameter optimization (max_depth, n_estimators, n_jobs, learning_rate) and feature engineering (max_features and addl. features).

Following are the evaluation metrics of the above model that determines its accuracy and efficiency

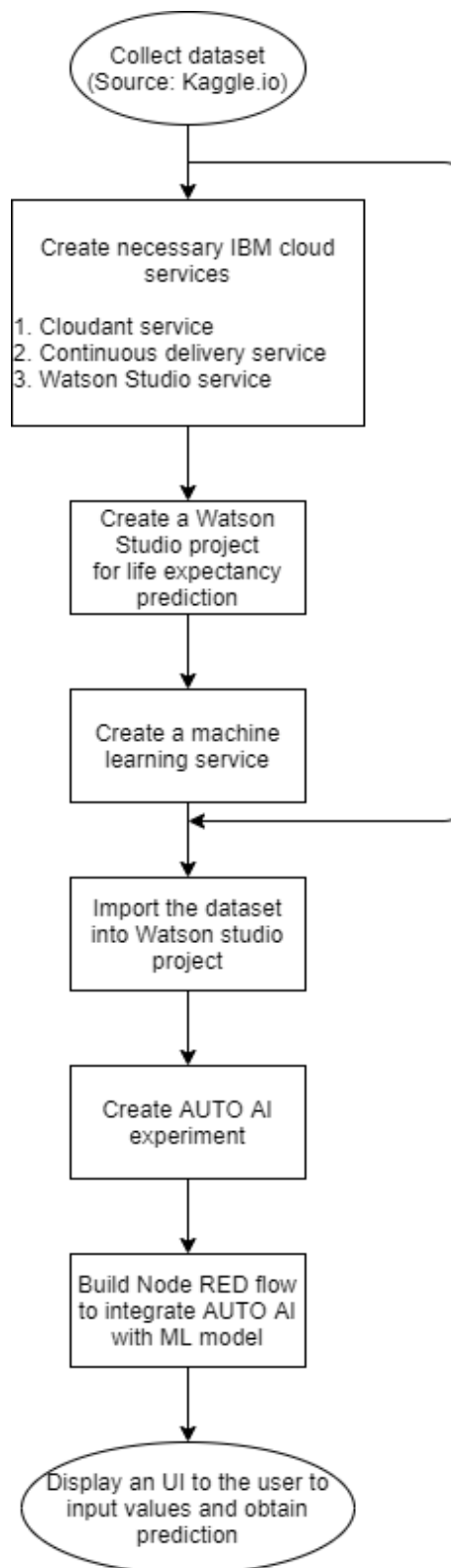
Evaluation Metrics	Holdout Score	Cross Validation Score
Root Mean Squared Error (RMSE)	1.918	2.016
R ²	0.957	0.955
Explained Variance	0.957	0.955
Mean Squared Error (MSE)	3.680	4.097
Mean Squared Log Error (MSLE)	0.001	0.001
Mean Absolute Error (MAE)	1.185	1.288
Median Absolute Error (MedAE)	0.640	0.770
Root Mean Squared Log Error (RMSLE)	0.029	0.031

Most Important features contributing to Life Expectancy



The contributing features are *HIV/AIDS*, *Income level*, and *adult mortality*.


Flow chart:



Result:

The project has been implemented successfully with RMSE value of 1.198 . The ML model has been created using Extreme forest classifier with the help of AUTO AI. The model was also saved to Watson studio project and deployed to the IBM cloud. The scoring endpoint was successfully integrated to UI using Node RED. The node RED flow was deployed and the UI was tested and prediction is displayed successfully to the user based on the input values.

Screenshot of output:



The screenshot shows a web application interface for a 'Life expectancy predictor'. It features a blue header bar with the word 'Home'. Below the header, the title 'Life expectancy predictor' is displayed in blue. The main content area contains a 'Prediction' result of '56.95999984741211' in bold. Below this, there are several input fields with labels and values: 'Country *' with 'India', 'Year *' with '2004', 'Status *' with 'Developing', 'Adult Mortality *' with '293', 'infant deaths *' with '87', 'Alcohol *' with '0.02', 'percentage expenditure *' with '15.29606643', 'Hepatitis B *' with '67', 'Measles *' with '466', and 'BMI *' with '13.8'. At the bottom of the form, there are two blue buttons labeled 'SUBMIT' and 'CANCEL'.

Field	Value
Prediction	56.95999984741211
Country *	India
Year *	2004
Status *	Developing
Adult Mortality *	293
infant deaths *	87
Alcohol *	0.02
percentage expenditure *	15.29606643
Hepatitis B *	67
Measles *	466
BMI *	13.8

Advantages and disadvantages:

Advantages:

1. The model is integrated with UI and hence helps predicting life expectancy by entering values through web interface.
2. The model helps finding the most important factors contributing to life expectancy. This helps in creating awareness to the public and help improving health of people. In this case, creating awareness about HIV/AIDS helps increasing life expectancy.
3. The prediction can be made in a fraction of time.

Disadvantages:

1. The model may require additional manual fine tuning as it is implemented through AUTO AI.
2. The insights are purely based on available data and there may be other contributing factors.

Applications:

This project could be deployed in production environment in health care organizations. The model has good RMSE score and hence very accurate in terms of prediction.

The node RED flow based UI is accessible from anywhere on the globe and hence can be used as a real-time application to predict life expectancy of a particular nation.

The analysis made during model implementation such as calculation of feature importances helps determining factors of poor health conditions in a country and elevating those factors.

Conclusion:

The life expectancy predictor has been implemented successfully. Different ML models has been analyzed for their performance and finally Extreme forest classifier (Extra trees classifier) algorithm with hyper parameter optimization and feature scaling has been used here. The model gave an RMSE score of 1.19. The UI has been designed in a way to fetch the predicted values from ML model deployed through Rest API by passing JSON message passing. The scoring endpoint has been integrated with UI using node RED to display the predicted value through forms. The application has been tested and deployed successfully.

Future Scope:

The development of this project surely prompts many new areas of investigation. The model's UI can be further enhanced by using interactive frameworks for front-end. Additional features could be included by using a more enhanced dataset.

Some of the limitations, such as AUTO AI requiring additional manual fine tuning can be done.

The process of IBM cloud could be upgraded to paid version to make the UI work faster in the cloud for full time production scale deployment.

Ability to view important features and correlation analysis through UI can be included.

Bibliography:

1. <https://www.allbusinesstemplates.com/download/?filecode=2KBA4&lang=en&iuid=9f9faa69-9fab-40ee-8457-ea0e5df8c8de>
2. <https://developer.ibm.com/tutorials/how-to-create-a-node-red-starter-application/>
3. <https://developer.ibm.com/technologies/machine-learning/series/learning-path-machine-learning-for-developers/>
4. <https://bookdown.org/caoying4work/watsonstudio-workshop/jn.html>
5. <https://bookdown.org/caoying4work/watsonstudio-workshop/auto.html>

Appendix:

Source Code:

```
try:
    import autoai_libs
except Exception as e:
    import subprocess
    out = subprocess.check_output('pip install autoai_libs'.split(' '))
    for line in out.splitlines():
        print(line)
    import autoai_libs
import sklearn
try:
    import xgboost
except:
    print('xgboost, if needed, will be installed and imported later')
try:
    import lightgbm
except:
    print('lightgbm, if needed, will be installed and imported later')
from sklearn.cluster import FeatureAgglomeration
import numpy
from numpy import inf, nan, dtype, mean
from autoai_libs.sklearn.custom_scorers import CustomScorers
import sklearn.ensemble
from autoai_libs.cognito.transforms.transform_utils import TExtras, FC
from autoai_libs.transformers.exportable import *
from autoai_libs.utils.exportable_utils import *
from sklearn.pipeline import Pipeline
known_values_list=[]
```

In []:

```
# compose a decorator to assist pipeline instantiation via import of modules and installation of packages
```

```
def decorator_retries(func):
    def install_import_retry(*args, **kwargs):
        retries = 0
        successful = False
        failed_retries = 0
        while retries < 100 and failed_retries < 10 and not successful:
            retries += 1
```



```

failed_retries += 1
try:
    result = func(*args, **kwargs)
    successful = True
except Exception as e:
    estr = str(e)
    if estr.startswith('name ') and estr.endswith(' is not defined'):
        try:
            import importlib
            module_name = estr.split(' ')[1]
            module = importlib.import_module(module_name)
            globals().update({module_name: module})
            print('import successful for ' + module_name)
            failed_retries -= 1
        except Exception as import_failure:
            print('import of ' + module_name + ' failed with: ' + str(import_failure))
            import subprocess
            if module_name == 'lightgbm':
                try:
                    print('attempting pip install of ' + module_name)
                    process = subprocess.Popen('pip install ' + module_name, shell=True)
                    process.wait()
                except Exception as E:
                    print(E)
                    try:
                        import sys
                        print('attempting conda install of ' + module_name)
                        process = subprocess.Popen('conda install --yes --prefix {sys.prefix} -c powerai '
+ module_name, shell = True)
                        process.wait()
                    except Exception as lightgbm_installation_error:
                        print('lightgbm installation failed!' + lightgbm_installation_error)
                else:
                    print('attempting pip install of ' + module_name)
                    process = subprocess.Popen('pip install ' + module_name, shell=True)
                    process.wait()
            try:
                print('re-attempting import of ' + module_name)
                module = importlib.import_module(module_name)
                globals().update({module_name: module})
                print('import successful for ' + module_name)
                failed_retries -= 1
            except Exception as import_or_installation_failure:

```

```

        print('failure installing and/or importing ' + module_name + ' error was: ' + str(
            import_or_installation_failure))
        raise (ModuleNotFoundError('Missing package in environment for ' + module_name
+
                                   '? Try import and/or pip install manually?'))
    elif type(e) is AttributeError:
        if 'module' in estr and ' has no attribute ' in estr:
            pieces = estr.split(" ")
            if len(pieces) == 5:
                try:
                    import importlib
                    print('re-attempting import of ' + pieces[3] + ' from ' + pieces[1])
                    module = importlib.import_module('.' + pieces[3], pieces[1])
                    failed_retries -= 1
                except:
                    print('failed attempt to import ' + pieces[3])
                    raise (e)
            else:
                raise (e)
        else:
            raise (e)
    if successful:
        print('Pipeline successfully instantiated')
    else:
        raise (ModuleNotFoundError(
            'Remaining missing imports/packages in environment? Retry cell and/or try pip install
manually?'))
    return result
    return install_import_retry
from sklearn.model_selection import train_test_split
X, X_holdout, y, y_holdout = train_test_split(X_prep, df_y.values, test_size=holdout_fraction,
random_state=random_state)
pipeline.fit(X,y)
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)
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'  
)  
predictions = service.score(payload=holdout_df.drop([experiment_metadata['prediction_column']],  
axis=1).iloc[:10])  
predictions
```