Project report on ‘Forecasting Life Expectancy Based on Economic Factors’ for the ‘Python in Machine Learning’ course

Monika Szuba

Jakub Wójcikowski

Bartłomiej Radecki

Mikołaj Żelazny

1. Introduction

In today's world, global socio-economic changes present us with numerous challenges. One of them is maintaining the high quality of citizens' lives based on instituted economic factors. A happy and healthy life influences long life expectancy by reducing stress, promoting healthier lifestyle choices, and fostering strong social connections. Positive emotions and well-being are associated with better mental and physical health, contributing to increased resilience and potential longevity.

This project focuses on applying machine learning techniques to analyze data concerning global millionaires and their countries, aiming to develop a predictive model for life expectancy based on significant economic aspects. The data collected from an extensive dataset on millionaires provide a unique perspective to understand the relationship between the economic stratum of society and the predicted length of life.

Based on preliminary research within the data, a significant impact of the Consumer Price Index (CPI) on life expectancy has been observed. The project aims to demonstrate whether the CPI is an economic factor that can predict life expectancy more effectively than any other economic factors.

2. Dataset description and related work

We were working with “Billionaires Statistics Dataset (2023)” dataset available at kaggle.com [1]. It consists of various statistics about the wealthiest people across the world. The Author states that the dataset is a compilation of data that was collected from various sources, mainly financial publications [1]. This dataset consists of 2641 data samples described by 35 features. The features include:

- Basic personal information like first and last name, age, birthdate, country of citizenship, state they currently reside
- Information about their wealth: net worth, category of industry they are involved in, source of wealth (wealth inherited or not), company or organization
- Majority of information concerns the country of billionaire’s residence, e.g. consumer price index and its yearly change, country’s education enrollment, life expectancy, tax rate and revenue, population, latitude and longitude of the country

We began our work with researching work previously done using this dataset. We have reviewed projects uploaded at Kaggle in the code section at the dataset page. On the 27th of October 2023 there were 18 projects completed, that tackled this data [1]. Vast majority of them were data visualization projects aimed to show information about world’s billionaires in an informative way. The most significant one of these is Bilionares Statistic 2023 - Data Insight [2]. This project’s goal was to check for existence of invaluable data samples (missing values), checking if the data distribution is reasonable, examining billionaires’ wealth distribution across industries, countries, regions, age, genders, analyzing proportions between self-made billionaires and those who inherited their wealth and study correlation between country’s human development index and billionaires’ wealth. The author found out that most of the billionaires are engaged in finance and investment and reside in the USA. He also found out that an average age of a billionaire is between 55 and 65 years, mostly male. The proportion of self-made in the scope of all billionaires is around 70%. The project’s results show that there is no correlation between country’s Human Development Index and billionaires’ net worth [2].

Another project worth mentioning is regression_millionairedatas [3]. This was actually the only project concerning other topic than dataset visualization. It was aimed to do a basic data visualization and build a regression system to predict bilionaires' net worth. Author used Random Forest Regressor and XGB Regressor. Predicted values in relation to reference values were presented in the form of scatterplots with no sums of errors pointed out, so we had some difficulties in assessing outcome of author's work. However, basing on the effects posted by the author it can be inferred that XGB Regressor performed better than Random Forest Regressor. Errors produced by Random Forest Regressor were significant, there can be a lot of outliers found [3].

Almost every project using this dataset was prepared using Python and its libraries, such as: Pandas, Matplotlib, Seaborn, Numpy. Some of the projects were also done using the R programming language.

3. Exploratory data analysis

Main task of the exploratory data analysis is showing relationships between features using visualization techniques. For the first task we extracted features describing countries' economic factors and generated correlation heatmap presented on Figure 1.

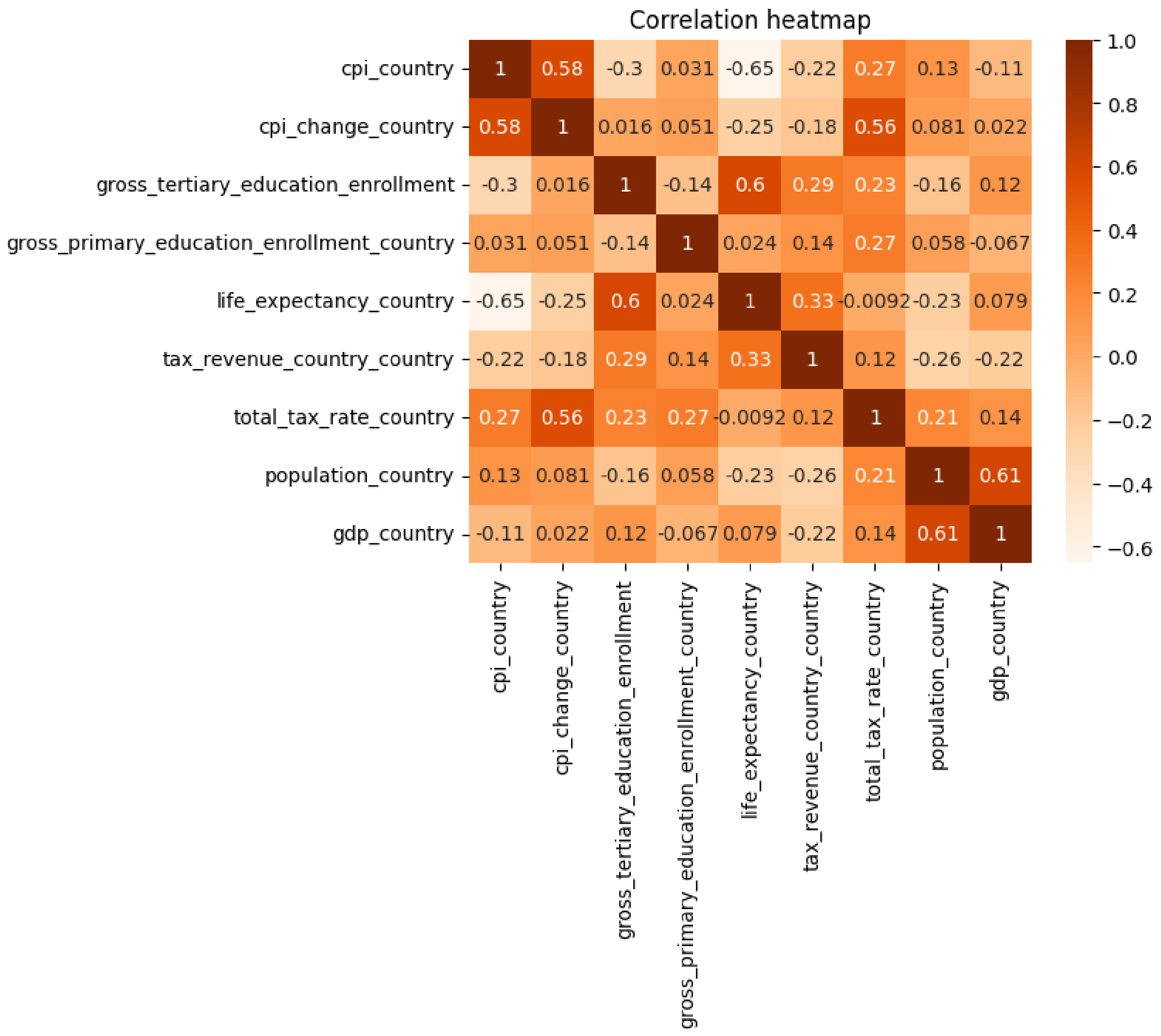


Figure 1. Correlation heatmap of all economic factors available in dataset.

The highest value of correlation appearing in the map is -0,65 for correlation between life expectancy and CPI. This leads to phrasing a hypothesis: **'CPI is the most important economic factor influencing people's life expectancy'**.

To prove it we made two regression models. First is predicting life expectancy based on only CPI, second one is used to predict life expectancy based on values of others economic factors.

Based on results of Figure 1. correlation heatmap we chose four factors (besides CPI) that are the most correlated with life expectancy. For better visualization we prepared second correlation heatmap presented on Figure 2.

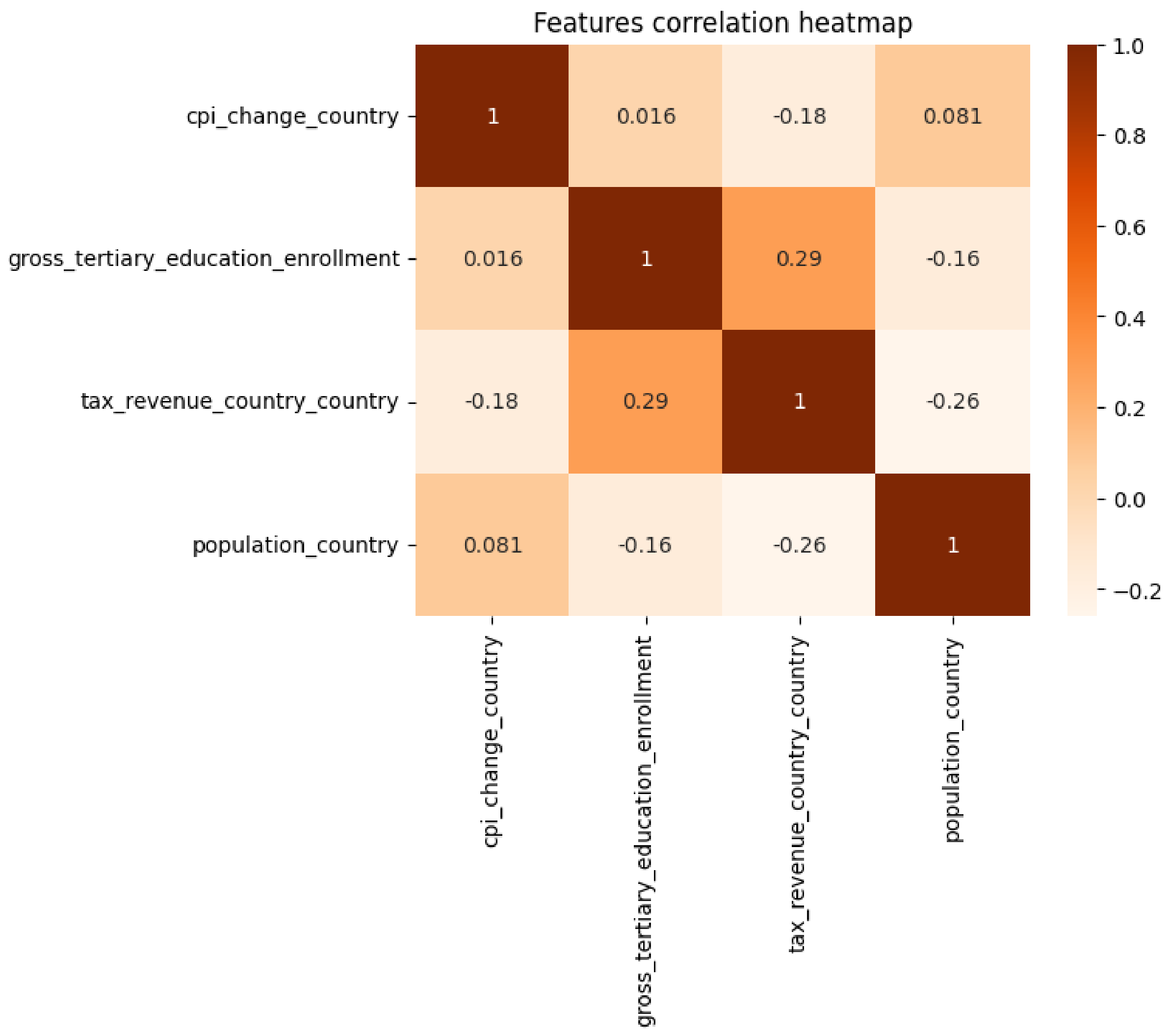
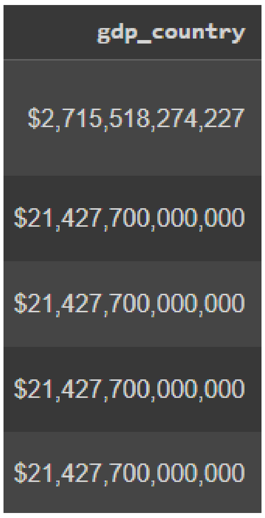


Figure 2. Correlation heatmap for four extracted factors.

4. Implementation and configuration of models

As pointed out in the previous chapter, our goal was to develop two regression models to predict life expectancy. To prepare models, we had to prepare the dataset for processing (e.g. plotting correlation heatmaps presented in chapter 3). Based on our hypothesis and goals we set for ourselves, after loading the dataset into workspace, we overwrote it, so that it contained only country-related features that seemed usable. As pointed out by our tutor, we divided data into training and testing subsets. Another step of data preparation was ensuring that all economic features values are numerical. Looking into the dataset, it became clear that values in the column describing Gross Domestic Product are not in the desired format. Initial string values for 'gdp_country' column looked as below:



gdp_country
$2,715,518,274,227
$21,427,700,000,000
$21,427,700,000,000
$21,427,700,000,000
$21,427,700,000,000

Figure 3. Initial content of "gdp_country" column of the dataset

We converted values in the column into string format using code shown below:

```
#changing string value into numerical to compare in heatmap
new_data['gdp_country'] = new_data['gdp_country'].str.replace(',','')
new_data['gdp_country'] = new_data['gdp_country'].str.replace('$','')
new_data['gdp_country'] = new_data['gdp_country'].astype(int)
new_data['gdp_country'] = new_data['gdp_country']/1000000000 #in
billions
```

Besides converting values in 'gdp_country' into numerical values, we also scaled them by dividing every value in the column, so that the unit of GDP was 1 billion dollars. Our dataset contained lots of duplicate data samples (resulting from lots of billionaires originating from the same countries). We got rid of them using drop_duplicates() pandas method. We also deleted rows with missing values.

That allowed us to plot correlation heatmaps presented with figures 1 and 2. Based on these we could choose features to use in our second model. We wanted to choose features that are highly correlated with life expectancy, but at the same time are not correlated with each other, not to carry the same information in more than one dimension. Chosen features:

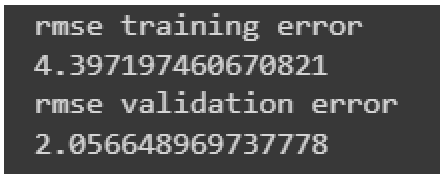
- gross_tertiary_education_enrollment (biggest correlation with life expectancy after cpi_country)
- population_country (not correlated with with gross_tertiary_education_enrollment)
- cpi_change_country (not correlated with the other features)

At the end of data preprocessing path, we applied standardization. This was aimed to reduce the influence of population of a country value being many orders of magnitude larger than values of the other features used. After that we proceeded to finding the best model for our solution.

To summarize, we agreed to build two different models predicting life expectancy in a country. The first model was meant to be based only on CPI ratio of a country. The second one was meant to be based on the three features chosen accordingly to their correlation with life expectancy. From now on we will present workflow for each model separately, comparing their results at the end.

Model 1

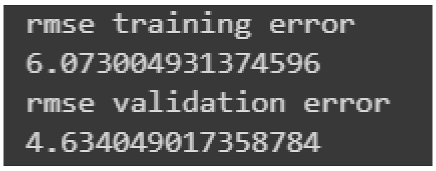
After dropping duplicates and rows with missing values, size of our dataset reduced vastly. The size of the training dataset after preprocessing (which included dividing training dataset into training and subsets) was 48 rows (validation subset consisted of 13 data samples). The reason for that is limited number of countries that “posses” billionaires. Given the small size of our dataset we suspected that we need to use relatively simple models. That is why we started with simple linear regression. Soon it appeared that to train any model using only 1 feature, we need to convert DataFrames into Numpy Arrays and reshape them using method reshape(-1,1). After doing that we trained linear regression model. The results were as follows:



```
rmse training error
4.397197460670821
rmse validation error
2.056648969737778
```

Figure 4. Root mean square errors for the training and validation subsets, Linear Regression, Model 1

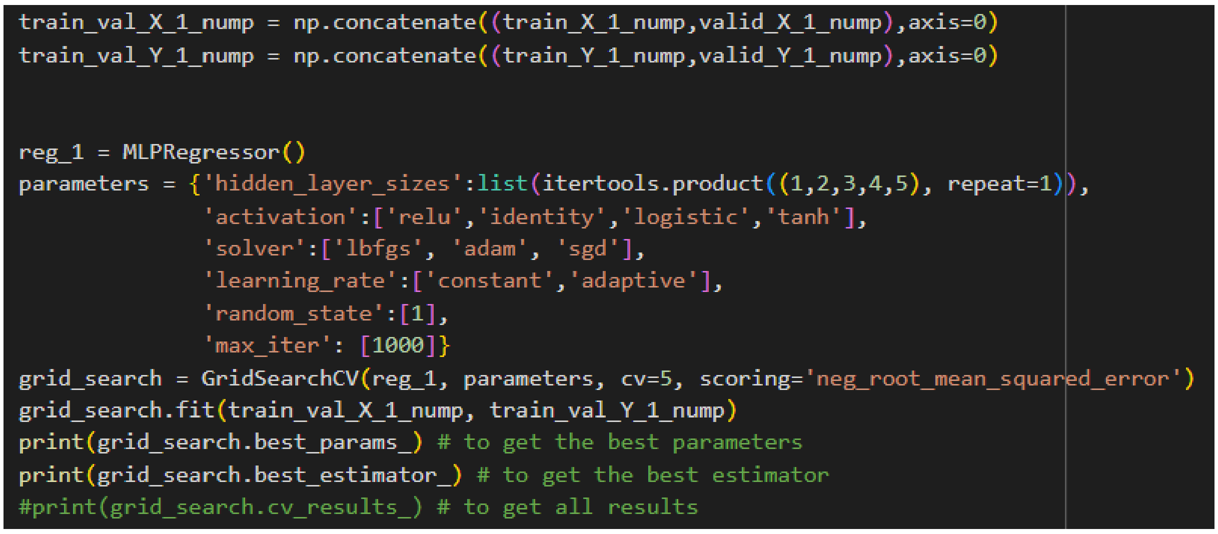
After that we tried two layer MLP with 4 neurons. Errors produced were very unsatisfying. The values were around 78 (years are the unit or we can say so, provided we are using root mean square errors). Thinking that we need to try something simpler than tested MLP, we tried SVM. We tested all of the possible kernel functions. The best results were obtained by using 3rd degree polynomial kernel:



```
rmse training error
6.073004931374596
rmse validation error
4.634049017358784
```

Figure 5. Root mean square errors for training and validation subsets,
SVM - 3rd degree polynomial kernel, Model 1

Knowing that we should be able to at least meet errors obtained with linear regression using MLP, we tried to find the best possible configurable parameters. We tried to find them using grid search algorithm. Knowing we have few data samples, we tested simple one layer perceptron with at most 5 neurons. We tried different activation functions and training solvers:



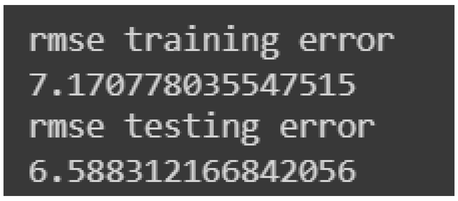
```
train_val_X_1_nump = np.concatenate((train_X_1_nump,valid_X_1_nump),axis=0)
train_val_Y_1_nump = np.concatenate((train_Y_1_nump,valid_Y_1_nump),axis=0)


reg_1 = MLPRegressor()
parameters = {'hidden_layer_sizes':list(itertools.product((1,2,3,4,5), repeat=1)),
              'activation':['relu','identity','logistic','tanh'],
              'solver':['lbfgs', 'adam', 'sgd'],
              'learning_rate':['constant','adaptive'],
              'random_state':[1],
              'max_iter': [1000]}
grid_search = GridSearchCV(reg_1, parameters, cv=5, scoring='neg_root_mean_squared_error')
grid_search.fit(train_val_X_1_nump, train_val_Y_1_nump)
print(grid_search.best_params_) # to get the best parameters
print(grid_search.best_estimator_) # to get the best estimator
#print(grid_search.cv_results_) # to get all results
```

Figure 6. Grid search testing of one hidden layer perceptron

We conducted tests on training and validation subsets merged. We used 5-fold cross validation. Aiming to make the code briefer we used *itertools.product()* (which is a part of python standard library) function to make the algorithm test all combinations of passed numbers of neurons. In this case this was not necessary, but during testing two layer perceptron it appeared to be very convenient.

The output of the grid search algorithm was that we would obtain best results for 4 neurons in one hidden layer (with *'logistic'* activation curve and *'lbfgs'* solver). Results obtained:

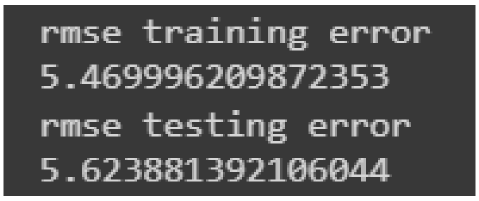


```
rmse training error
7.170778035547515
rmse testing error
6.588312166842056
```

Figure 7. Root mean square errors for training and test subsets, MLP - 1 hidden layer, 4 neurons, Model 1

The results still were not close to those obtained with linear regression. We tried testing multiple configurations for two hidden layer perceptron. Parameters for the best performing configuration:

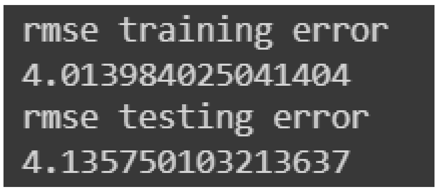
- number of layers: 2
- number of neurons in a hidden layer: 4 in the first layer, 2 in the second
- learning rate: 'constant'
- activation function: 'logistic'
- solver: 'lbfgs'
- maximum iterations: 1000



```
rmse training error
5.469996209872353
rmse testing error
5.623881392106044
```

Figure 8. Root mean square errors for training and test subsets for the best MLP configuration, Model 1

From all models that we tested we see that linear regression gave the best results. The final step was to merge training and validation subsets and test the model with the test subset extracted at the very beginning. The results were as follows:

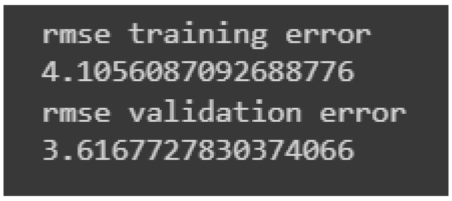


```
rmse training error
4.013984025041404
rmse testing error
4.135750103213637
```

Figure 9. Root mean square errors for training and test subsets for the linear regression, Model 1

Model 2

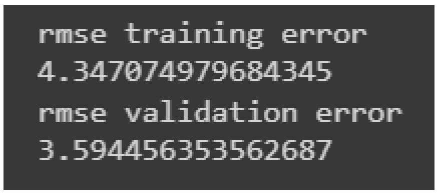
For the second model we also started with testing linear regression. In this case, there was no need to convert DataFrame to Pandas' arrays. Results for linear regression we obtained:



```
rmse training error
4.1056087092688776
rmse validation error
3.6167727830374066
```

Figure 10. Root mean square errors for the training and validation subsets, Linear Regression, Model 2

Similarly as in the case of first model we tried basic perceptron that produced large errors. Then we tried SVM for different kernel functions. The best outcomes produced SVM with sigmoidal kernel function applied:



```
rmse training error
4.347074979684345
rmse validation error
3.594456353562687
```

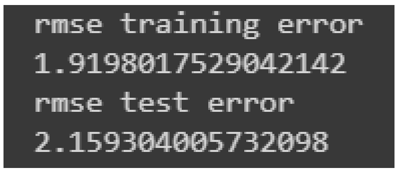
Figure 11. Root mean square errors for training and validation subsets, SVM - sigmoidal, Model 2

For the second model we tried different configurations for one hidden layer and two hidden layers. The best parameters we acquired:

- number of layers: 1

- number of neurons in a hidden layer: 8
- learning rate: ‘constant’
- activation function: ‘logistic’
- solver: ‘lbfgs’
- maximum iterations: 86

Root mean square errors that we were able to obtain using these parameters:



```
rmse training error
1.9198017529042142
rmse test error
2.159304005732098
```

Figure 12. Root mean square errors for training and test subsets for the best MLP configuration, Model 2

These were the best results that we were able to obtain using all tested models.

Results comparison

According to the results presented in this chapter, we see that building a model consisting of more features than just Consumer Price Index resulted in obtaining better results. To measure errors we used root mean squared error, which unit can be interpreted as years in this case. Comparing results for testing subsets we say, that sum of errors for the best configuration of the second model was better than for the first model for almost 2 years, which makes 48% improvement in predicting accuracy. According to mean world’s life expectancy being 73,4 years [4], obtained error around 2,16 years is acceptable. This disproves our hypothesis that Consumer Price Index value is the most important economical factor influencing peoples’ life expectancy.

5. Summary and conclusions

As pointed out in the previous chapter, our work has disproven the hypothesis we initially stated. During our project, we managed to build various regression models and evaluate their relevance to our task. We think that the size of our dataset could greatly influence efficiency of the models we trained. The size was directly associated with the matter we dealt with, however we were able to build a satisfying regressor, but only in the case of using three features (different than stated in the hypothesis).

6. Acknowledgements

For preparation of this report ChatGPT was used. The purpose of this tool was to help with writing in English - a foreign language for Authors. The abilities of ChatGPT made sentences smoother and more accurate.

Source of the project can be found in the GitHub repository. Link is located in chapter 8. Appendices.

In the project we used command *itertools.product()* - function creating iterators for efficient looping. Documentation can be found on the website:

https://docs.python.org/3/library/itertools.html

Contributions of each author presents as below:

- Hypothesis formulation – B. Radecki, J. Wójcikowski
- Relevant work research – M. Żelazny, B. Radecki, J. Wójcikowski
- Exploratory Data analysis – M. Szuba
- Model implementation and configuration – J. Wójcikowski, B. Radecki, M. Żelazny
- Model testing – B. Radecki, J. Wójcikowski
- Report writing – M. Szuba, M. Żelazny
- Project organization – M. Żelazny
- Project results presentation – M. Szuba, M. Żelazny

7. References

[1] https://www.kaggle.com/datasets/nelgiriyewithana/billionaires-statistics-dataset

[2] https://www.kaggle.com/code/achmadryanfauzi/bilionares-statistic-2023-data-insight

[3] https://www.kaggle.com/code/mawro73/regressionon-millionairedatas

[4] https://www.worldometers.info/demographics/life-expectancy/#google_vignette

8. Appendices

The whole code we have written while preparing this project that was briefly explained in chapter 4 is available under this url address:

https://github.com/jwojcikow127/Python_For_Machine_Learning.git