Happiness Score prediction using Multivariate Linear Regression and Multi-Layer Perceptron Models

Menaka Kollu
Arizona State University
mkollu@asu.edu

Nishanth Solomon
Arizona State University
nsolomo2@asu.edu

Sushilkumar Muralikumar
Arizona State University
smural32@asu.edu

***Abstract*-The Assignment focuses on creating a linear regression and Multilayer perceptron(MLP) for Happiness Score dataset.Report presents 1. Preparation of the data 2.Features used in the model 3.RMSE of linear regression model 4. Training a multilayer perceptron 4.Comparing the modeling errors with different features and parameters**

I. Problem Formulation

The goal of the assignment is to predict Happiness score using linear regression model and Multilayer perceptron model. To achieve this below steps are followed:

1. Preparation of data and choosing features
2. Preparation of data and dividing into training and testing dataset
3. Building linear regression model
4. Calculating Root Mean Square Error(RMSE) of linear regression model
5. Training multilayer perceptron
6. Calculating RMSE of MLP model
7. Adjusting the parameters of MLP to achieve comparable performance as linear regression model in terms of RMSE

II. Methods

Preparation of Data:
The provided dataset contained the happiness scores of about 156 countries, and the metrics used to determine these scores. The data for a period of 5 years was provided(2015-2019). Upon comparison of the metrics used for arriving at the ranks for different years, it was found out that only six metrics were common between the datasets. Thus, the given dataset was prepared accordingly. The metrics used are as follows:

- Happiness score
- GDP
- Life Expectancy
- Freedom
- Generosity
- Perception of Corruption

It was also observed that the dataset for the year 2018 had a missing field for one of its metrics and thus, the row corresponding to the data(Row 21 of the 2018 dataset) has been omitted. The resulting datasets were combined using Python, to create a single list of data(.csv file) in the ascending order of the years in which these data were recorded. Final dataset used for the project is present in the CombinedData_all.csv file.
The size of the dataset is 782*8(where the first two columns contain information about the overall rank and the name of the country and the third column represents the happiness scores).

The data was split for training and testing of the models. The split was 80-20, where 80% of the data is used to train the models and the remaining 20% is used for testing. The data was split randomly using Python thus ensuring that no single year contributes more to the testing phase, thus, ensuring proper functioning of the models.

Linear Regression Model:

In this model we are predicting the happiness score using the five selected features (*gdp, life_expectancy, freedom, generosity, corruption and generosity*) .
This is a Multivariate Linear Regression problem.
Hypothesis for this model is given by :

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$$

where, n=5
and x is the input
We are using LinearRegression model provided by sklearn[1]

Multilayer Perceptron model:

The goal is to formulate a Multi-Layer perceptron model which can perform on par compared to the Multivariate Linear Regression model used earlier. The Sklearn library's MLPRegressor[2] model has been used for this purpose.

A Multi-Layer Perceptron model with one hidden layer was chosen for this application. The model has five input neurons, 100 neurons in the hidden layer and one output neuron. This was arrived at based on different test cases run by varying the metrics used in the hidden layer and the execution of the program. The metrics that were varied are as follows:

- Number of neurons in the hidden layer
- Activation function
- solver
- batch_size

The results of the analysis are as shown in results section

III. Results

Linear Regression Results:

The RMSE value for the linear regression model:

```
RMS error of linear regression model = 0.6176113713889252
```

Figure 1: RMSE for linear regression model

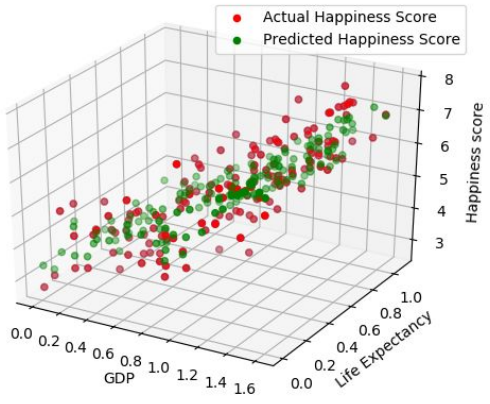

Figure 2: Plot between gdp, life_expentency and Happiness score

Multi-layer Perceptron Results:

The RMSE value for the multi-layer perceptron :

```
RMS error of mlp model = 0.6168465606440675
```

Figure 3: RMSE for Multi-layer perceptron model

Number of neurons in hidden layer	activation	solver	max_iter	batch_size	RMSE
100	relu	lbgs	100	5	0.597219
100	relu	adam	100	5	0.587751
1000	relu	adam	100	5	0.571601
1000	identity	adam	100	5	0.613991
100	relu	adam	100	10	0.594426
100	relu	sgd	100	5	0.616847

Figure 4:Analysis of different Hidden Layer Metrics

Analysis of different Hidden Layer Metrics is available in the MLPResults excel sheet.

From the Figure 4 , we can observe that the multi-layer perceptron with the following parameters works as comparable to the linear regression model:

Parameters:

1. Number of neurons in hidden layer:100
2. Activation function:Rectified Linear Unit(RELU)
3. solver:sgd (stochastic gradient descent)
4. max_iter=100
5. batch_size=5

RMSE obtained with this RMSE is 0.6054212091371728

In addition to this , we can observe that the RMSE can be further decreased using different parameters.

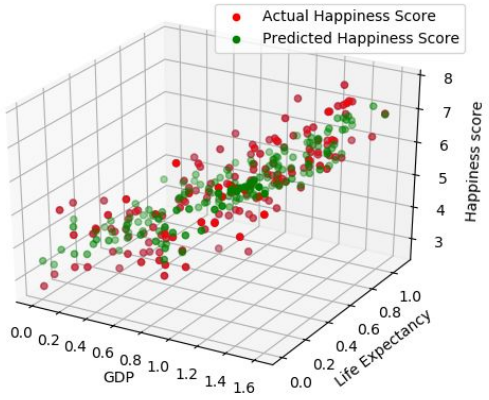

Figure 5: Plot between gdp, life_expentency and Happiness score

IV. Conclusions

This assignment has focussed on developing a Multi-Layer Perceptron model that can perform on par with a Multivariate Linear Regression model. The performance of these two models was performed using the RMSE metric. The RMSE obtained for the Linear Regression model is 0.6176 and the RMSE for the Multi-Layer Perceptron model was found to be 0.6168. It was also observed that the model performed better when a different solver such as Low Memory Broyden-Fletcher-Goldfarb-Shanno(LBFGS) was used. Thus, the proposed MLP model performs satisfactorily.

References

[1]https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html

[2]https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPRegressor.html#sklearn.neural_network.MLPRegressor