Computational Machine Learning Assignment 1

Syed Haider Saleem 28, March 2020

s3796258

Project Overview

The basic purpose of this project is to use a real world dataset to explore and practice the typical machine learning approach. We are going to apply a wide range of regression algorithm(s) to a real world data-set, analyse the output of those regression algorithm(s), research how to extend these regression techniques and then provide an ultimate judgment of the final trained model that we could use in a real-world setting.

Problem Statement

The goal is to apply regression technique(s) and predict the target life expectancy of a newborn based on several attributes (features) related to the region in which he/she was born with the maximum accuracy on an unknown dataset given multiple input features.

Metrics

We will be using **mean absolute error** as a primary means of evaluation, as our data contains outliers in multiple features meaning we can't use mean squared error as it is going to put a major penalty on data points that are far away. Along with this we will also be taking model.score as a means of identifying the best combination of validation and training score. Our primary goal here will be to improve validation score as it could tell us better if our model is underfit, overfit or just about right.

We will pick the model that would provide us with minimum mean absolute error and maximum score on both training and validation sets.

Data Source

The dataset is taken from “Global Health Observatory data repository”. It basically tells us the life expectancy of an individual considering multiple factors that might affect it. The data contains 23 different features (attributes) and a total of 2071 records.

```
In [5]: data.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2071 entries, 0 to 2070
Data columns (total 24 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   ID                              2071 non-null   int64
 1   TARGET_LifeExpectancy           2071 non-null   float64
 2   Country                         2071 non-null   int64
 3   Year                            2071 non-null   int64
 4   Status                          2071 non-null   int64
 5   AdultMortality                  2071 non-null   int64
 6   AdultMortality-Male             2071 non-null   int64
 7   AdultMortality-Female           2071 non-null   int64
 8   InfantDeaths                    2071 non-null   int64
 9   Alcohol                         2071 non-null   float64
 10  PercentageExpenditure           2071 non-null   float64
 11  Measles                         2071 non-null   int64
 12  BMI                             2071 non-null   float64
 13  UnderFiveDeaths                 2071 non-null   int64
 14  Polio                           2071 non-null   int64
 15  TotalExpenditure                2071 non-null   float64
 16  Diphtheria                      2071 non-null   float64
 17  HIV-AIDS                        2071 non-null   float64
 18  GDP                             2071 non-null   float64
 19  Population                      2071 non-null   int64
 20  Thinness1-19years               2071 non-null   float64
 21  Thinness5-9years                2071 non-null   float64
 22  IncomeCompositionOfResources    2071 non-null   float64
 23  Schooling                       2071 non-null   float64
dtypes: float64(12), int64(12)
memory usage: 388.4 KB
```

Data Handling

After exploring our data and realising what we are dealing with, we decided to separate the input features from its output which is life expectancy.

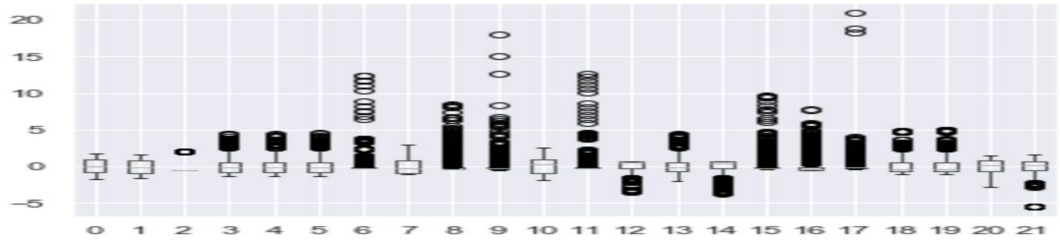
Seperating input and output features

```
In [10]: # features on which we are going to train our data
         dataX = data.iloc[:,2:]

In [11]: # expected outout feature
         dataY = data['TARGET_LifeExpectancy']
```

Outliers

Outliers are those extreme values that deviate from other observations in data. As we can clearly see in the image below that our data does contain many outliers hence we are using mean absolute error as our metric for this purpose. We have used box plot visualisation methods to identify these outliers.



Train and Test Split

Now it's time to split our data into train and test sets as before going on to predicting on totally unknown data it is a good practice to keep a part of training data separate so that we could later use it as a validation set to predict and get an idea of how well our model is going to perform on the unseen data. We have set aside 30% of data from our training set for this purpose since we have only about 2000 records hence we need to keep a considerably larger test set to get an idea of how our model is performing.

Train and Test set split

```
In [16]: X_train, X_test, y_train, y_test = train_test_split(dataX, dataY, test_size=0.30, random_state=42)
```

Standardization

Standardization basically rescales the data to have a mean of 0 and a standard deviation of 1, since different features in our data have different scales that could create a bias towards some features and could severely affect the outcome of our model if we don't bring them to a common scale.

Standardisation

```
In [14]: standard_scaler = preprocessing.StandardScaler()
         X_train = standard_scaler.fit_transform(X_train)
         X_test = standard_scaler.transform(X_test)
```

Models

So we trained our data on different models including Linear regression, Polynomial regression with Regularization and without Regularization. We also tried to see what impact would normalisation have on our models.

- Linear regression
- Polynomial regression
- Regularization
- normalization

Results

Linear Regression without normalising our data

Training score: 0.765354116881042

Test score: 0.7320675964838607

Mean absolute error (validation set): 3.575247374541553

Linear Regression after normalising our data

Training score: 0.7653541168810418

Test score: 0.73206759648337

Mean absolute error (validation set): 3.5752473745549436

Polynomial regression without Regularization/normalisation

Training score: 0.7564448206204832

Test score: 0.49059224350182734

Mean absolute error (validation set): 4.108726792048489

Polynomial regression without Regularization (normalised)

Training score: 0.9047819148884483

Test score: 0.5648712221848406

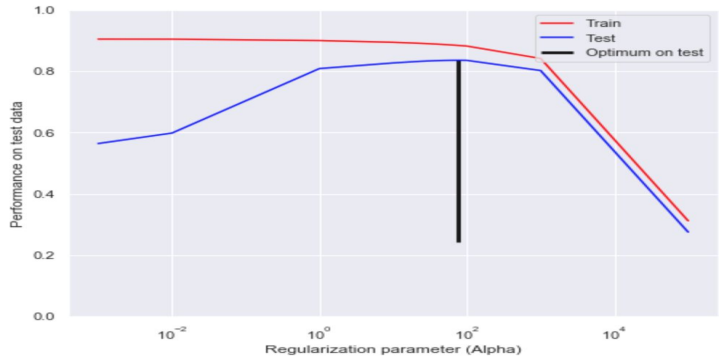
Mean absolute error (validation set): 3.1412708785372914

Regularization

Now regularization is the concept of tuning your hyperparameters to find an equation that best fits the data and reduces overfitting by neglecting noise. For this purpose we need to find an optimal alpha for regularization which is basically the penalty size. I have used a loop to find different training and validation scores and have picked the most optimal alpha for my case using a graph.

Polynomial regression with L2 Regularization and normalised data (Ridge model)

```
Optimal regularization parameter for ridge model : 75
```



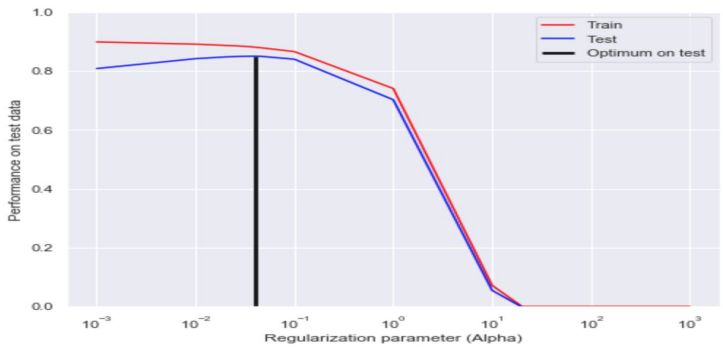
Training score: 0.8841565485365511

Test score: 0.8355169350261873

Mean absolute error (validation set): 2.749649391143768

Polynomial regression using L1 Regularization and normalised data (Lasso model) - WINNER

```
Optimal regularization parameter for Lasso model : 0.04
```



Training score: 0.8806634684245571

Test score: 0.8501837882788352

Mean absolute error (validation set): 2.628834285034522

Analysis

So we can clearly see that **polynomial regression (second degree) using L1 Regularization (Lasso model)** is our winner as it is out performing all other models by giving us the best training and validation score as well as minimum Mean absolute value on validation set which was our metrics defined in the start. One more thing to realise is that standardization does not have a major impact on linear regression in lower degrees but it highly boosted the performance of polynomial regression when we went into higher degree and it makes sense as we go in higher degree the impact of unbalanced scale is definitely going to increase. Regularization also helped in achieving a good score on the validation set as it was clear that as we went into higher degrees the model started to overfit and get more complex by learning the noise as well, so regularization definitely helped us in achieving a more generic model by removing noise and reducing overfitting.

Also in regularization why I believe that L1 regularization gave me a better result was due to the fact that Lasso essentially shrinks the less important features coefficients to zero hence removing some features altogether and since we had so many features in our data this type of regularization performed better compared to L2 regularization.

Improvements

I believe we will make some major improvements in the future if we do some early feature engineering and try to use our features more effectively as it's effect can be seen in L1 regularization but this was out of scope for this assignment. Some other machine learning models like random forest might give us better results and k cross validation could help us as well in giving an all rounded better performance.