MACHINE LEARNING EXPERIMENTATION REPORT

Nithin Sai Jalukuru
School of Engineering and Applied Sciences(Data Sciences and Applications MPS)
University at Buffalo,Buffalo,NY,14260
njalukur@buffalo.edu
October 1, 2022

Experimentation

****This particular data is utilized across all the Ml models for better understanding of model and comparisons**

1.1 Data

Expectancy WHO data is taken from Kaggle and helps in predicting life expectancy with the help of various factors for a period of 15 years (2000-2015).
The World Health Organization's (WHO) Global Health Observatory (GHO) data repository keeps track of all countries' health status as well as many other related parameters. The datasets are made available to the public for the aim of analyzing health data. The dataset on life expectancy and health factors for 193 nations was obtained from the same WHO data repository website, as was the comparable economic data from the United Nations website. Despite the fact that much research have been conducted in the past on factors influencing life expectancy, including demographic variables, income composition, and death rates.

Each row of data contains information about the country, continent, year, status, life expectancy, adult mortality, infant mortality, alcohol, GDP, and other input characteristics. Life Expectancy data consists of 2938 rows × 22 columns.

1.2 Data Preparation

1.2.1 Data Cleaning

In this dataset we have 2 categorical features , "country" and "status"(status describes whether the country is developed or developing) and remaining all are numeric type. First step is to rename the feature names to a proper format (to all lower case and removed spaces between words).

Lets check for null values in the data set, it is found to be having 194 null values in "alcohol" feature, 10 in "adult_mortality", 553 in "hepatitis_B", 34 in "BMI"(Body Mass Index), 19 in "polio", 226 in "total_expenditure", 226 in "diphtheria", 448 in "GDP", 652 in "population", 34 in "thinness_1_19_years", 34 in "thinness_1_5_years", 167 in "income_composition_of_resources" and 163 in "schooling" feature.

As the missing values are under the MAR(missing at random) pattern, In this scenario, the likelihood of missing values is determined by the features of the observable data. So, we will be using KNNimputer, KNNImputer aids in the impute the missing values in observations by locating the nearest neighbors using the Euclidean distance matrix. In this problem, 2 neighbors are chosen for estimating the missing values in the data.

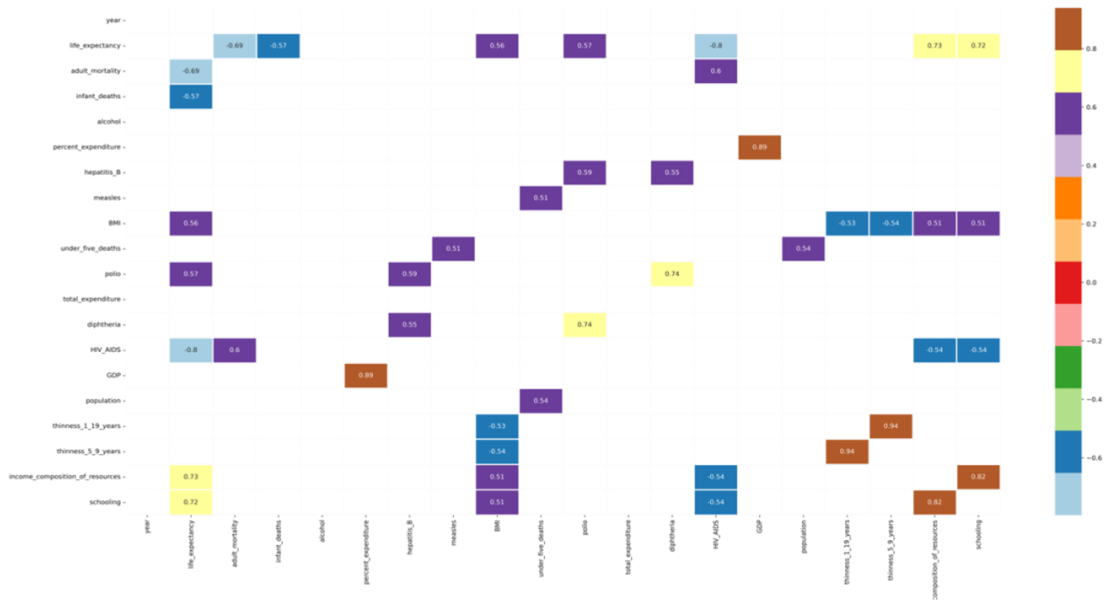
Most important step in data cleaning is to identify and treatment of outliers. I have used IQR method for identifying the outliers. It is found out that in our current data, we have outliers as follows

Feature	**Outlier Count**	**% of outlier's in data**
life_expectancy	17	0.58
adult_mortality	86	2.93
infant_deaths	315	10.72
hepatitis_B	316	10.76
polio	279	9.5
total_expenditure	51	1.74
HIV_AIDS	542	18.45
income_composition_of_resources	130	4.42

Based on upper and lower limit of inter quartile range, outliers have been removed. This step is essential because linear and logistic regression models are not robust with outliers.

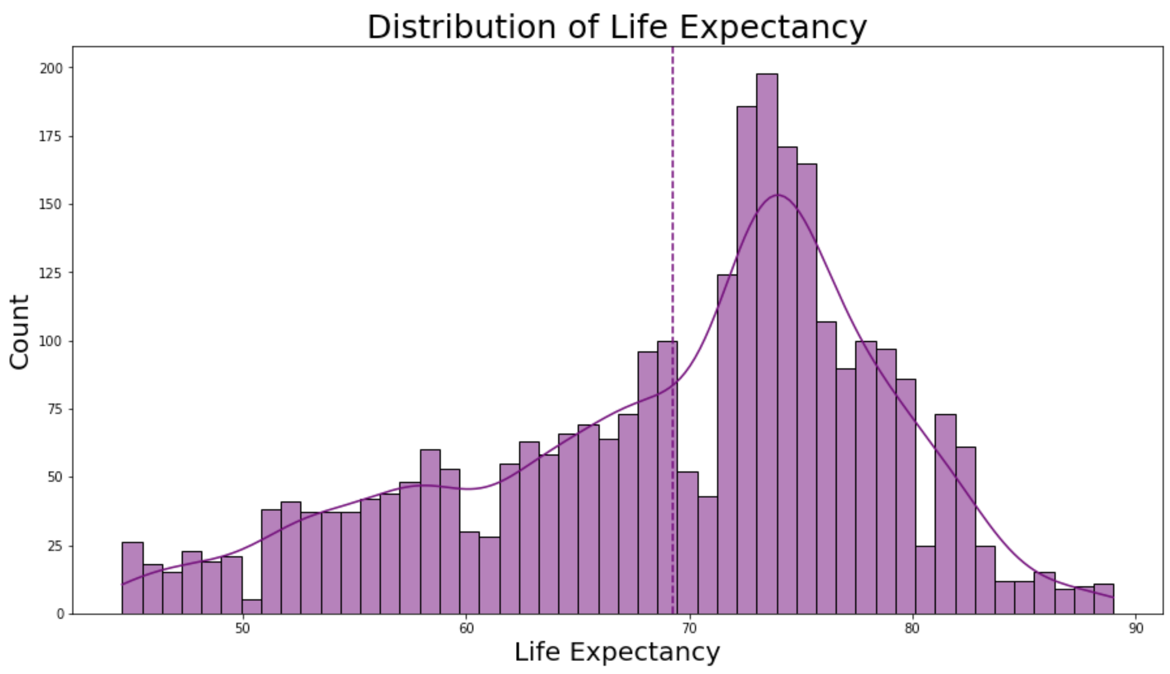
1.2.2 **Data Visualizations**

I. Let's make a correlation plot to examine how the features are related and check for is there any multicollinearity between 2 independent variables.



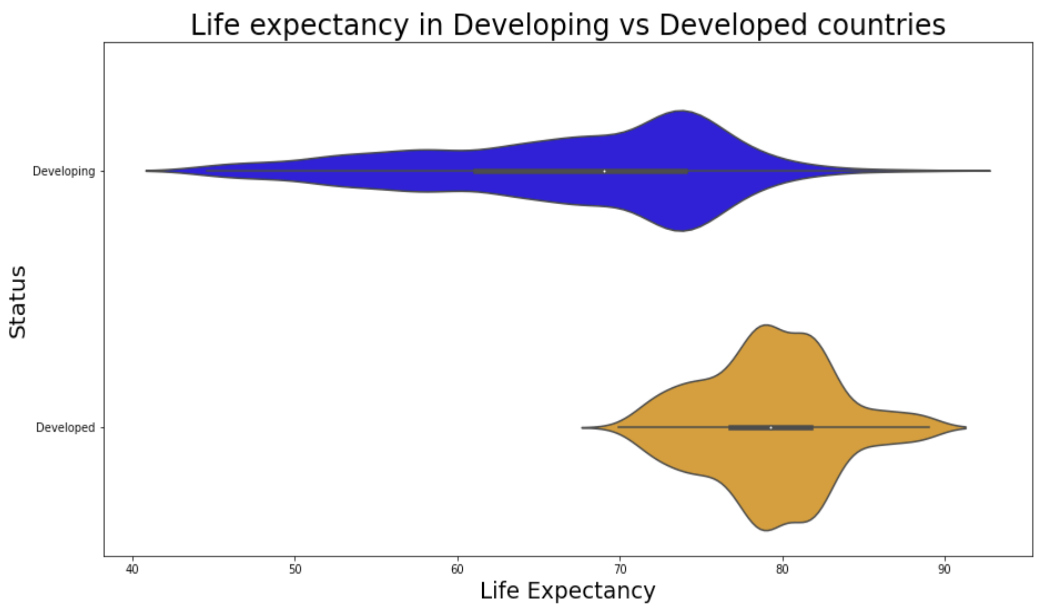
I have filtered out low correlated featured which are less than 0.5, the results are quite interesting. Life_expectancy and schooling (Average schooling years for entire population) are highly correlated, this could be related to the healthy lifestyle that they learn in school. GDP and percentage expenditure are highly positive correlated(0.89). life_expectancy and BMI, life_expectancy and adult mortality late also have good correlation between them.

II. In next visualization, we will plot normal distribution of life expectancy.



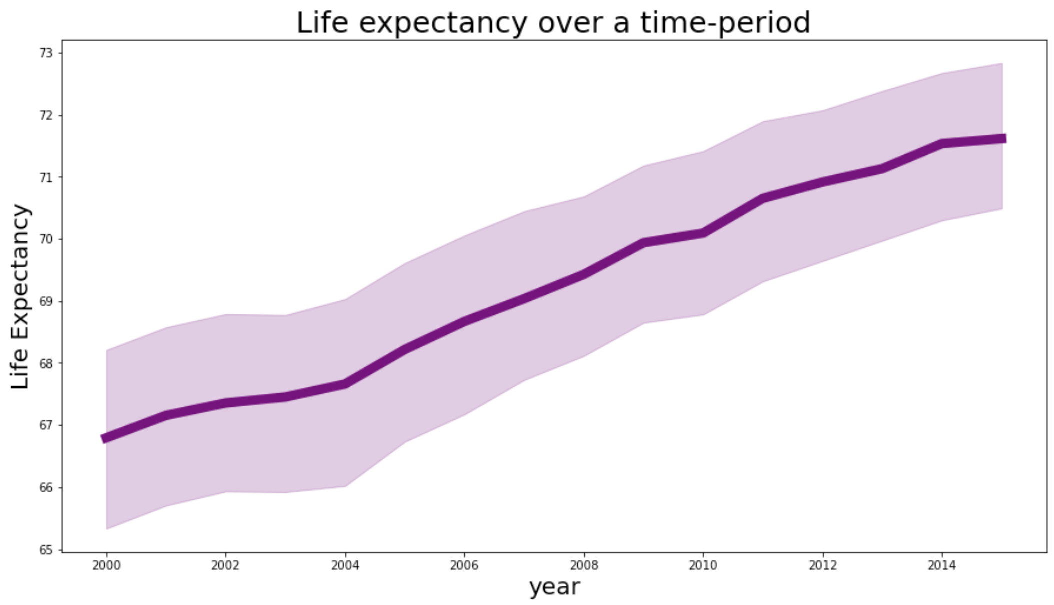
From above plot, we can infer that the mean of the Life expectancy is around 70 years and f from the graph it is evident life expectancy for most of the people is higher than 70 years.

III. Violin plot between life expectancy and status of the countries(developed/developing)



We can observe from the violin plot that life expectancy of the developed countries is higher compared to the developing countries.

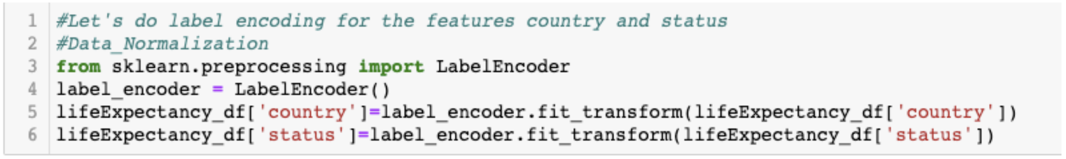
IV. Life Expectancy over the years



We can say that over the period of time from 2000 to 2015, the life expectancy has been increased from 67 to 72.

1.2.3 Feature Transformation

Feature Encoding : We must encode the data once it has been cleaned so that the ML algorithm may ingest it. In our data set we have 2 categorical features one is country and other is status. I have performed Label Encoder on these 2 features and converted them into numeric of 0 to (n-1) classes.



```
#Let's do label encoding for the features country and status
#Data_Normalization
from sklearn.preprocessing import LabelEncoder
label_encoder = LabelEncoder()
lifeExpectancy_df['country']=label_encoder.fit_transform(lifeExpectancy_df['country'])
lifeExpectancy_df['status']=label_encoder.fit_transform(lifeExpectancy_df['status'])
```

one-hot encoding transforms a categorical feature with n categories into n features, one for each category, with value 1 for the data's category and 0 for the remaining all, therefore it cannot be performed on the country feature. There are 197 different countries, so when we perform one hot encoding it adds new 197 features.

1.2.4 Feature Scaling

Feature Standardization: I have performed feature standardization on each feature. Doing this feature values are rescaled to have a mean of 0 and standard deviation of 1

```python
#data_Standardization
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
lifeExpectancy_df = pd.DataFrame(scaler.fit_transform(lifeExpectancy_df), columns=lifeExpectancy_df.columns)
lifeExpectancy_df
```

1 Simple Linear Regression

Regression is a powerful and simple model for predicting numeric responses from a set of independent variables (1 independent variable in case of Simple linear regression). The linear regression algorithm is still widely used across a variety of domains due to its effectiveness, ease of interpretation, and flexibility, even though linear regression may seem to be overlooked in the ever-increasing world of complex neural network architectures in modern machine learning. It is essential to understand linear regression to build a solid foundation in machine learning.

Regression analysis' main goal is to look at two things, one is Is it possible to accurately forecast an outcome (dependent) variable using a set of predictor variables? And the other is Which individual variables are highly important predictors of the outcome variable, and how do they affect the outcome variable?

By determining the line that best "fits" the data, the supervised algorithm known as linear regression learns to model the dependent variable, y, as a function of some independent variable, or "features," x_i.
In general, the simple linear regression equation is

$$y=\beta_0+\beta x_1$$

Where y is dependent variable/Target and x_1 is the independent variable and beta are the weights/coefficients of our model. These are important as they are what our model learns during optimization

To know whether our model performs best or bad, we have metrics to define that. Main two metrics are MSE and Rsquared.

Mean Squared Error :

We'll use MSE (Mean Squared Error) to measure how closely a regression line is to a set of points because it measures how close a projected value is to the actual value.

$$MSE = \frac{1}{n}\Sigma_{i=1}^{n}(y_i - \hat{y}_i)^2$$

In general use, our regression model will be fitted to a set of training data and its performance will be assesses using MSE on the test dataset

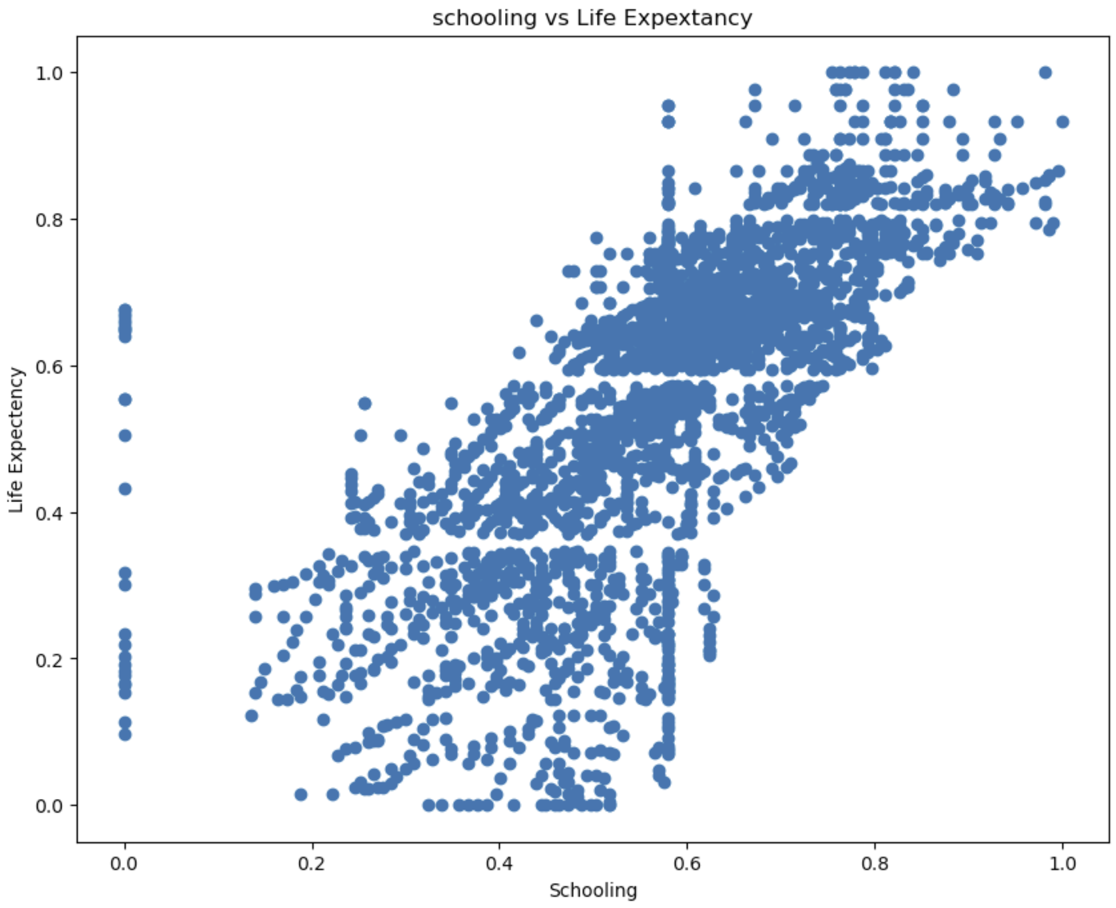
Rsquared :

The so-called goodness of fit measurements, which encapsulate how well a model fits a set of data, can also be used to assess regression models. R-squared is the most often used goodness of fit metric for linear regression, a metric that represents the % of variance in y explained by our features x_i

$$R^2 = 1 - \frac{\Sigma_{i=1}^{n}(y_i - \hat{y}_i)^2}{\Sigma_{i=1}^{n}(y_i - \bar{y}_i)^2}$$

R-squared has a maximum value of 1, which denotes a model that completely accounts for variance. A negative r-squared indicates that our model is performing worse than a flat line across the mean of our data would (it captures less variance).

<u>Problem Statement</u>: I want to predict Life expectancy with schooling (Average schooling years for entire population), this can be done with simple linear regression

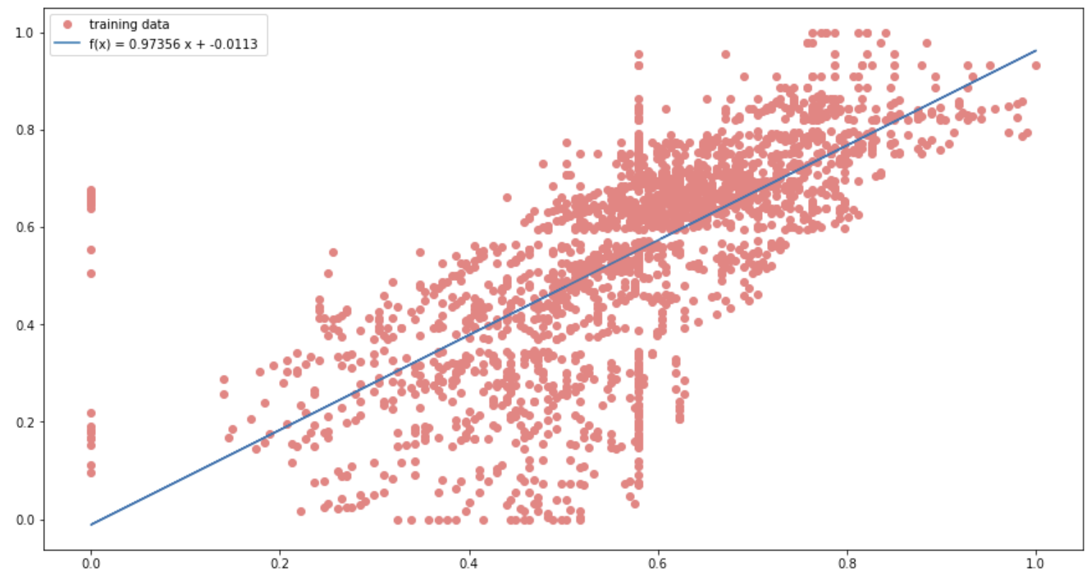


We can see that there is a linear relationship between schooling and life expectancy. It is now clear that the longer the schooling years, the healthier learning (eating habits, not having addictions) that they will learn in school, the longer their life expectancy.

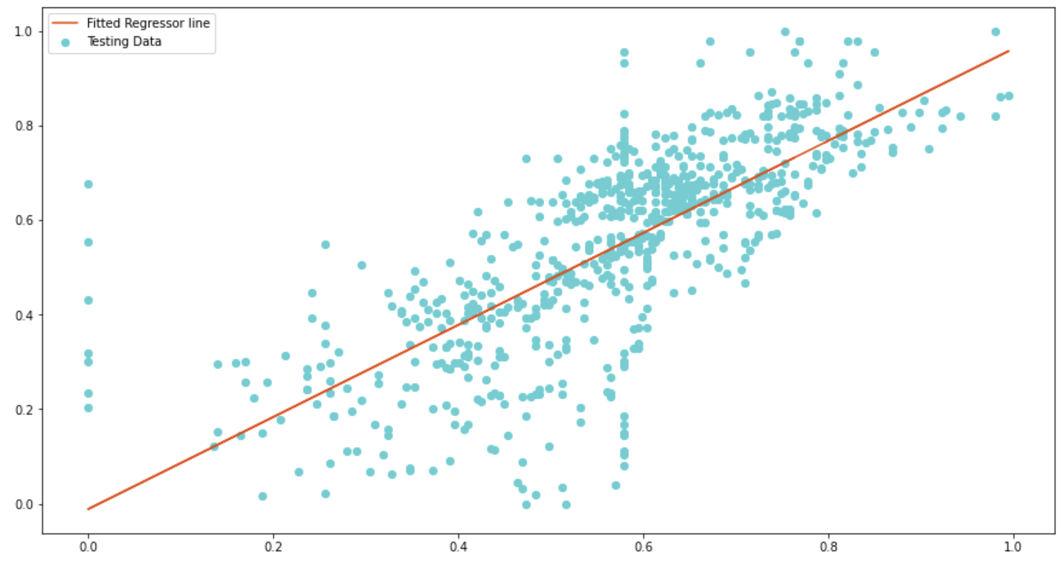
1.1 Model Fitting

I have split the data in 75% inti training (x) and 25% into testing (y). and passes x and y for learning the model. After learning, estimated coefficient and intercept are 0.97356 and -0.011316

Now we know the fitted function - f(x)= -0.011316 +0.97356 *x , we can plot it.



The next step is to evaluate how good our model is on predicting.



This model has produced a mean squared error(MSE) of 0.019199471386668943. which is relatively low (Lower the MSE , better is model). This can be improved by adding additional features which support the life expectancy. This can be done by multi linear regression

2 Multi Linear Regression

Regression is a powerful and simple model for predicting numeric responses from a set of independent variables. A regression model typically includes multiple features. A Multiple regression model is what we're using. There are few assumptions for regression model, Validity, representativeness, Additivity and linearity, Independence of Errors , Homoscedasticity, Normality of errors.

By determining the line that best "fits" the data, the supervised algorithm known as linear regression learns to model the dependent variable, y, as a function of some independent variable, or "features," x_i.
In general, the Multi linear regression equation is

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_p x_p + \epsilon$$

Where y is dependent variable/Target and x_i is the independent variables and beta's are the weights/coefficients of our model. These are important as they are what our model learns during optimization and ϵ is the model's irreducible error, that encompasses all of our data's unmodeled components.

2.1 Model Fitting

Feature Selection

As we are performing regression on predicting the life Expectancy, to best fit the model irrelevant features must be eliminated. To do this , I performed recursive feature elimination(RFE) to get top 10 features that can contribute to predicting life Expectancy.

After performing RFE, top 10 features are year, infant_deaths, alcohol, hepatitis_B, polio, total_expenditure,HIV_AIDS,thinness_5_9_years,income_composition_of_resources,schooling and dropped remaining features.

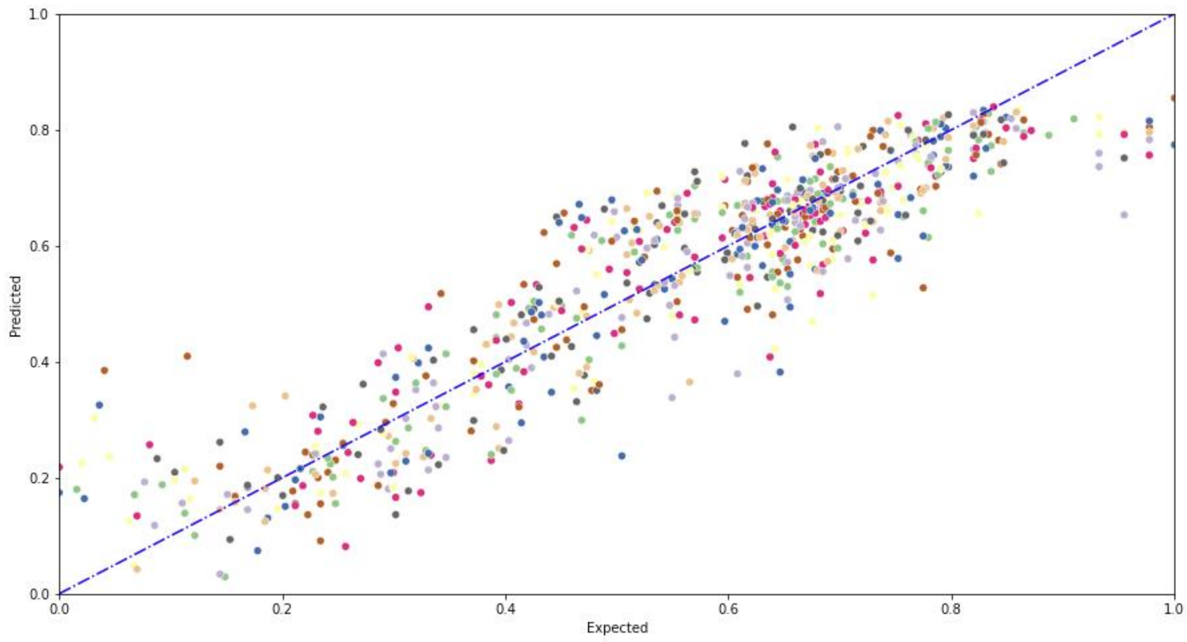
Problem statement : Predict Life expectancy for selected features (as stated above) which describes the life expectancy

I have split the data in 75% inti training (x) and 25% into testing (y). and passes x and y for learning the model. After learning, estimated coefficients and intercept are 0.03, -0.06 , -0.09 , 0.01, -0.04 , 0.02 , 0.09 , 0.01 , -0.26 , -0.07 , 0.19 , 0.09 and 0.45596908

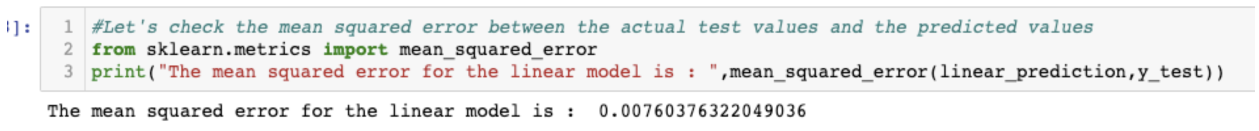
Learning Linear model Equation is

$$Y = 0.03x_1 - 0.06x_2 - 0.09x_3 + \ldots\ldots\ldots\ldots + 0.456$$

The next step is to evaluate how good our model is on predicting.

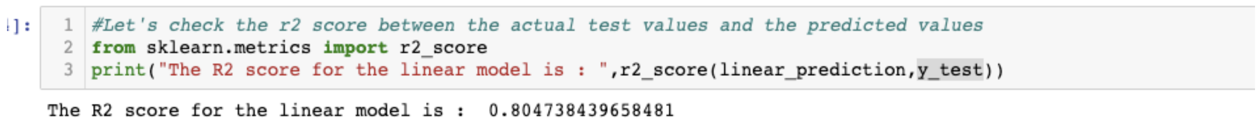


Coming to the evaluating the model, as discussed already we have 2 metrics MSE and Rsquared.



```
#Let's check the mean squared error between the actual test values and the predicted values
from sklearn.metrics import mean_squared_error
print("The mean squared error for the linear model is : ",mean_squared_error(linear_prediction,y_test))
```

```
The mean squared error for the linear model is :  0.00760376322049036
```



```
#Let's check the r2 score between the actual test values and the predicted values
from sklearn.metrics import r2_score
print("The R2 score for the linear model is : ",r2_score(linear_prediction,y_test))
```

```
The R2 score for the linear model is :  0.804738439658481
```

This model has produced a mean squared error(MSE) of 0.00760376 which is relatively low (Lower the MSE , better is model) and Rsquare of 0.8047385, which indicates the model that accounts for the variance(As shown above). From the above results we can conclude that our model has done good job in predicting life expectancy.

References:

1. https://mlu-explain.github.io/linear-regression/
2. https://github.com/Avik-Jain/100-Days-Of-ML-Code/blob/master/Code/Day2_Simple_Linear_Regression.md
3. https://github.com/jns1406/themlsbook/tree/master/jupyter_book

3 Logistic Regression

The method of modeling the likelihood of a discrete result given an input variable is known as logistic regression. One that can only have two values, such as true or false, yes or no, etc., is the binary outcome that is most frequently modeled by logistic regression. In other words, simply Logistic regression is a supervised learning approach that may be used to categorize data into groups or classes by estimating the likelihood that an observation falls into a specific class based on its attributes.
Logistic Regression is often used for Binary classification, assessing whether or not a data point belongs to one of two categories, or whether or not an event will occur.

The typical setup for logistic regression is as follows: there is an outcome y that falls into one of two categories (say 0 or 1), and the following equation is used to estimate the probability that y belongs to a particular category given inputs $X = (x_1, x_2, ..., x_k)$

$$P(y = 1|X) = \text{sigmoid}(z) = \frac{1}{1 + e^{-z}}$$

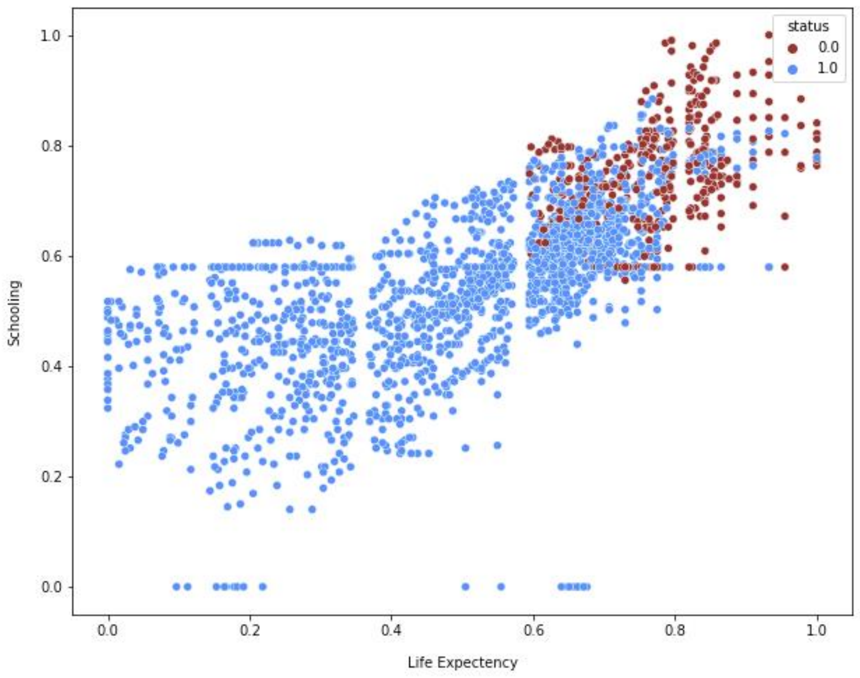
Where $z = \beta_0+\beta_1 x_1+ \beta_2 x_2+\ldots\ldots\ldots\ldots\ldots+ \beta_k x_k$, looks similar like linear regression model, this is called linear predictor, and this linear predictor is transformed by sigmoid function so that the values of features falls between 0 and 1

Evaluating our Logistic Model:
The purpose of fitting our model is to discover the parameters that optimize a function that describes how well the model performs. Simply said, the goal is to create predictions that are as near to 1 as possible when the outcome is 1 and as close to 0 as possible when the event is 0. The function to be optimized in machine learning is known as the loss function or cost function. The loss function in logistic regression is called log-loss function.

$$\text{Log-Loss} = \sum_{i=0}^{n} -(y_i * \log(p_i) + (1 - y_i) * \log(1 - p_i))$$

Where n is number of samples, y_i is true class/label for index i and p_i is the model prediction for the index i. Since Log-Loss is the inverse of the Log-Likelihood, minimizing it is identical to maximizing it. Our goal is to find the coefficients that minimizes the loss function.

Problem statement : My goal is to predict whether a country is developing or developed bases on Life expectancy and schooling (continuing from linear model)

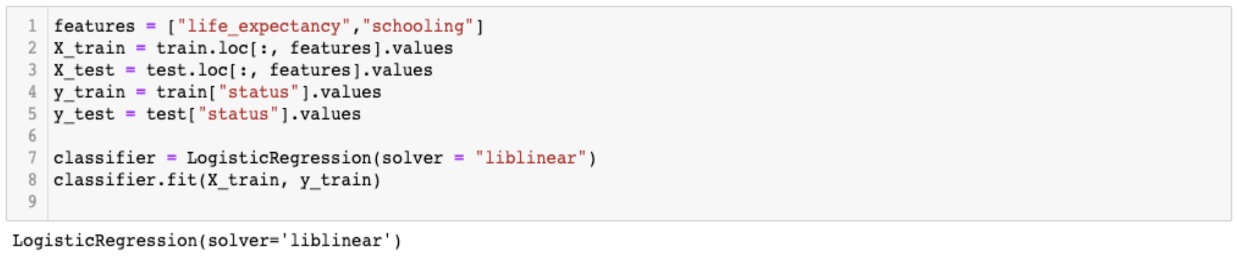


We can see that for developed countries (status 0) , the life expectancy is more and also schooling years than of developing countries (status 1)

3.1 Model Fitting

Feature Selection :
Independent variables : life Expectancy and schooling
Target variable : status

While learning logistic regression on training data, I have used liblinear solver(Algorithm to solve the problem) as shown below snippet. This solver is a good choice for small/medium datasets (like our current dataset)



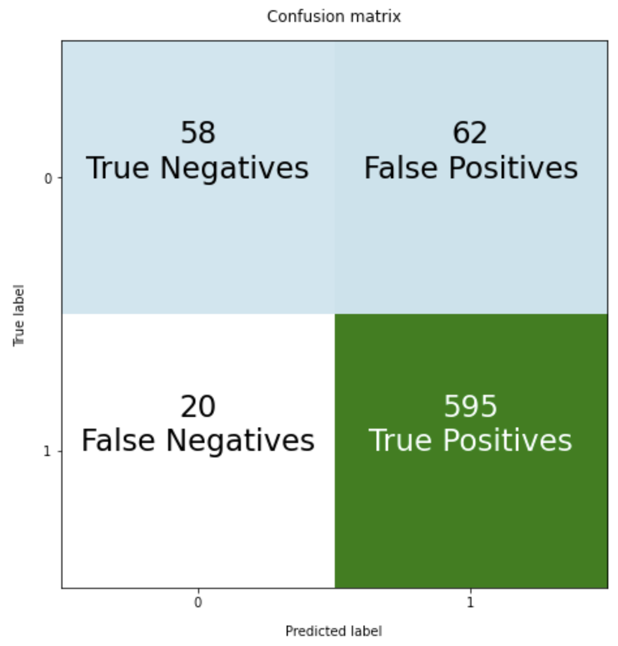
```
features = ["life_expectancy","schooling"]
X_train = train.loc[:, features].values
X_test = test.loc[:, features].values
y_train = train["status"].values
y_test = test["status"].values

classifier = LogisticRegression(solver = "liblinear")
classifier.fit(X_train, y_train)

```

```
LogisticRegression(solver='liblinear')
```

After learning the logistic model on the training data, it’s time for prediction on the testing data. And following are the results obtained.

First metric is confusion Matrix, which is used to define the performance of the classification algorithm.

Let’s see what results I have got,



This shows the model has 58+595 = 653 correct predictions and 62+20 = 82 wrong predictions

And also look at the classification report.

```
                      precision    recall  f1-score   support

 developed countries       0.74      0.48      0.59       120
developing countries       0.91      0.97      0.94       615

            accuracy                           0.89       735
           macro avg       0.82      0.73      0.76       735
        weighted avg       0.88      0.89      0.88       735
```

Precision is no of true positives/(no of true positives + no of false positives), Precision is intuitively the classifier's capacity to not categorize a sample as positive if it is negative.

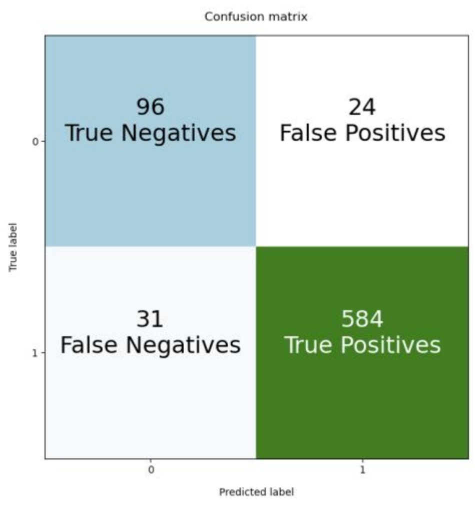
Recall is no of true positives/(no of true positives + no of false negatives), The recall is intuitively the classifier's capacity to find all positive samples.

Interpretation : Our model has high recall 89% , and false negatives are less of concern and also has a accuracy of 89% which is good.

This can be improved by adding more features while learning the model.

We do the same process as we followed in feature selection of linear regression. But instead of life expectancy, we will select the features which are relevant to status.

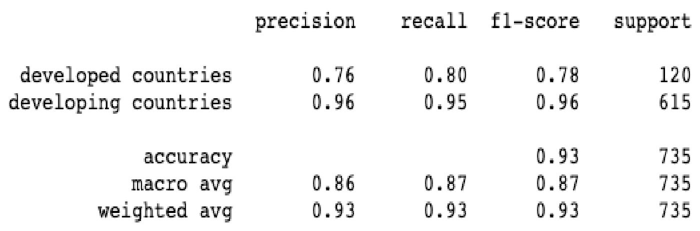
After learning the model and predicting (same as above), the new results are quite interesting. Confusion matrix is



We can see that adding more relevant features can improve the learning accuracy of the model and hence can predict right. Now this model has very low false positives compared to above model with less features.

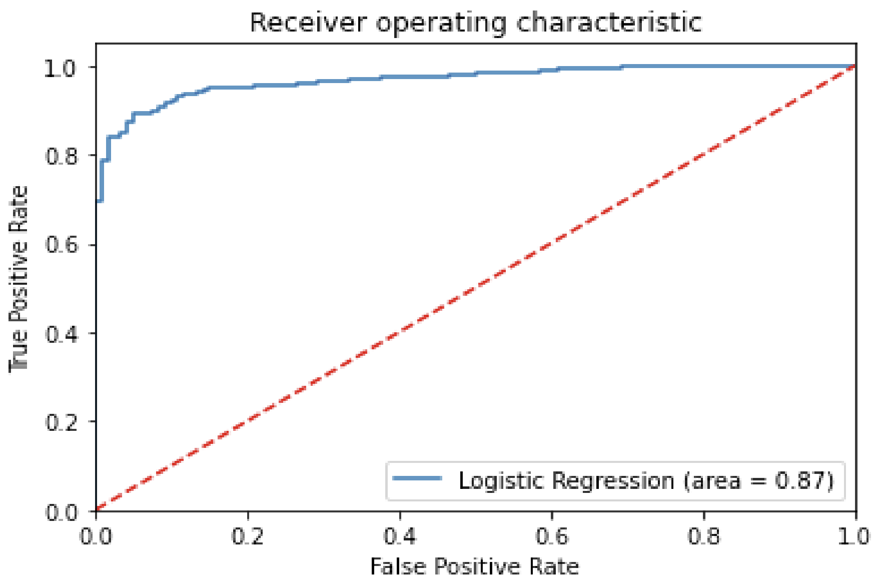
Right predictions : 96+584 = 680
False Predictions : 31+24 =55

Also, we look at new classification report,


```
                      precision    recall  f1-score   support

 developed countries       0.76      0.80      0.78       120
developing countries       0.96      0.95      0.96       615

            accuracy                           0.93       735
           macro avg       0.86      0.87      0.87       735
        weighted avg       0.93      0.93      0.93       735
```

Surely, the recall is quite high, and accuracy has been increased to **93%.** So overall the model has been improved and logistic regression did a excellent job in classifying the status.

Also let's see what RoC Curve tells,



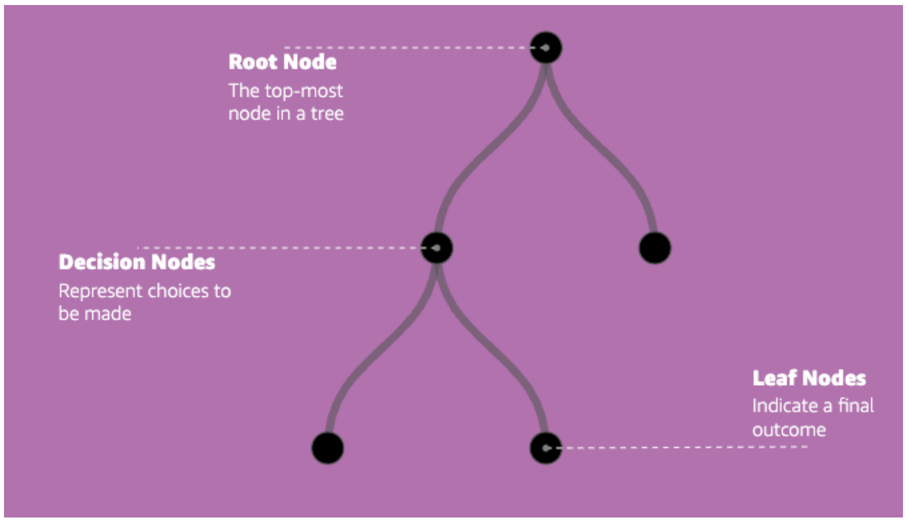
When comparing models, the rule of thumb is that curves that are above a random classifier's ROC Curve (the diagonal red line) are good or decent. The higher they are (i.e., the closer they are to the elusive perfect classifier's curve), the better. Therefore, as we can see that our logistic curve is way higher than the random regressor curve, which is better.

And AOC(Area under the curve), 1 is considered perfect. In our model AoC is 0.87. which is actually sign of a good model.

4 Decision Trees

Decision Trees are popular supervised machine learning techniques. They're popular due to their simplicity of interpretation and wide range of uses. They are applicable to both regression and classification applications.

Training a Decision Tree using data entails determining the order in which decisions should be assembled from the root to the leaves. Test data can then be transmitted down from the top until it reaches a leaf node, which represents a prediction for that data point.



Source: MLU-Explain

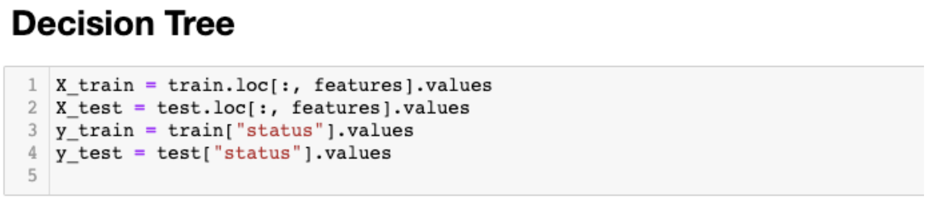
Lets build a decision tree for our problem. If we assume if we were given life expectancy and schooling and to predict to which country status (developed / developing) does it belong to.Based on conditions/rules we give; the leaf nodes are formed. And other rule is to not go too deep, if we go too deep our decision tree starts training from the noise.

From a high level , decision trees create a series of sequential rules for sub dividing the data into well-defined regions for classification. With given wide range of potential options , the algorithm exactly knows where the data to be partitioned. Other important concept is of Entropy, "Entropy measures the amount of information of some variable or event. We'll make use of it to identify regions consisting of a large number of similar (pure) or dissimilar (impure) elements."

Problem statement : My goal is to predict whether a country is developing or developed based on Life expectancy and schooling (continuing from logistic model) and then both logistic and Decision trees in classifying the data.

3.1Model Fitting

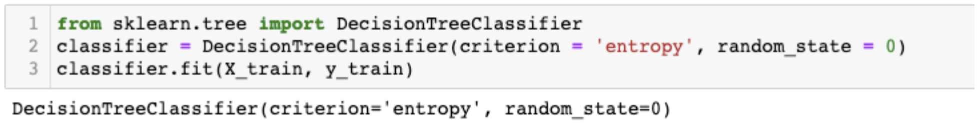
Feature Selection :
Independent variables : life Expectancy and schooling
Target variable : status



Decision Tree

```
X_train = train.loc[:, features].values
X_test = test.loc[:, features].values
y_train = train["status"].values
y_test = test["status"].values

```

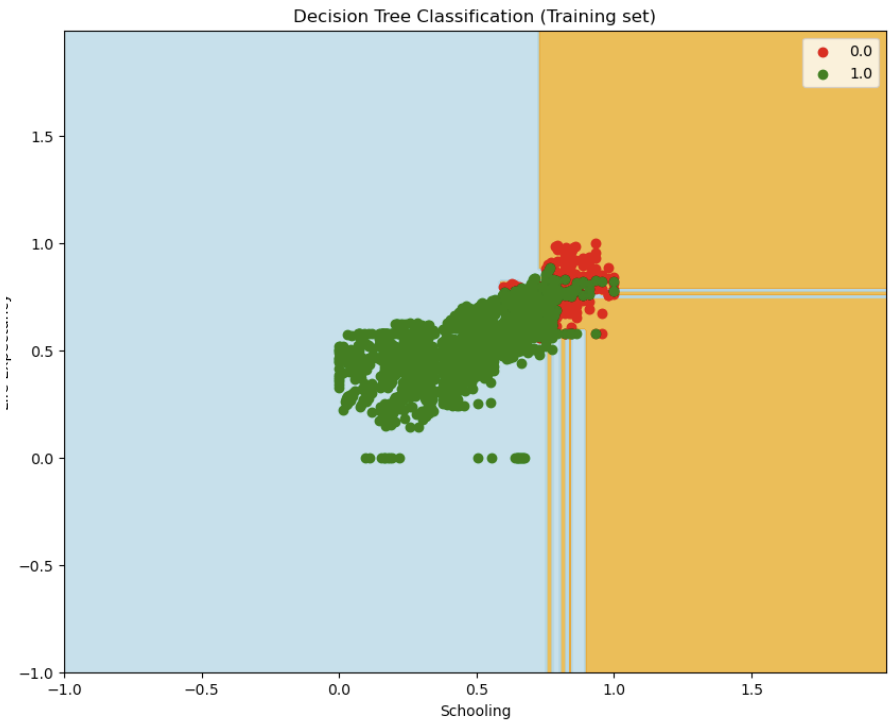
While learning the decision tree classifier model on training set, we are considering Entropy partition criterion.



```
from sklearn.tree import DecisionTreeClassifier
classifier = DecisionTreeClassifier(criterion = 'entropy', random_state = 0)
classifier.fit(X_train, y_train)
```

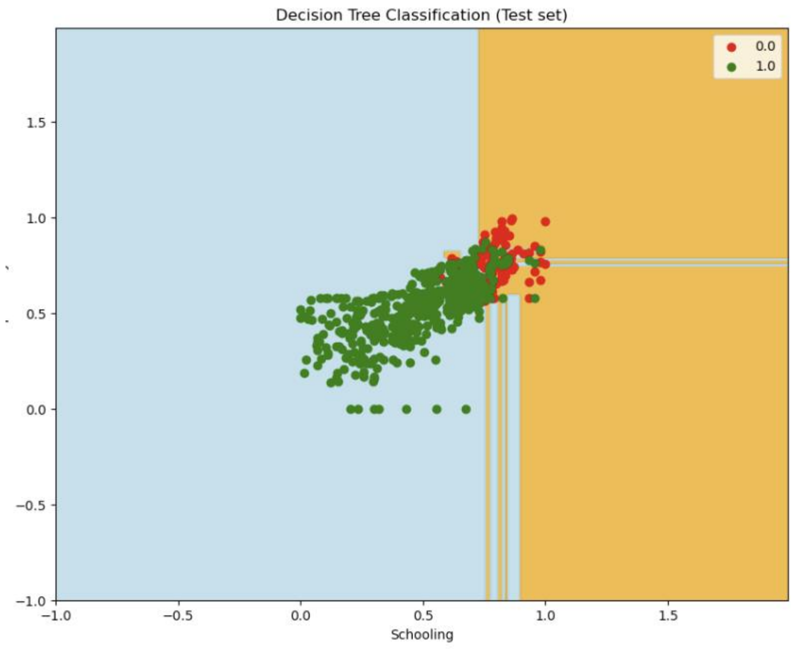
```
DecisionTreeClassifier(criterion='entropy', random_state=0)
```

Let's see how the model has learned/trained on training set



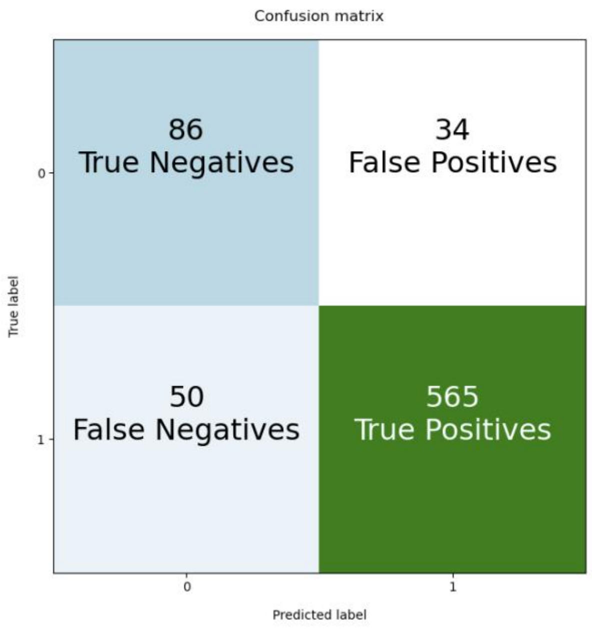
As we can see that decision tree classifier has done better job of classifying the status on training set (developed or developing). More developing data compared to developed status countries.

We will proceed further this model on prediction on testing set and see the results obtained.



In above plot, green dots represent developing status of the country and red dots represents developed status of the country. When learning model performed on the testing set, from plot it looks decision tree classifier has partitioned very well and classified the data. Let's check out the metrics to confirm it.

First, we will look at the confusion matrix (best for estimating the performance of classifier)

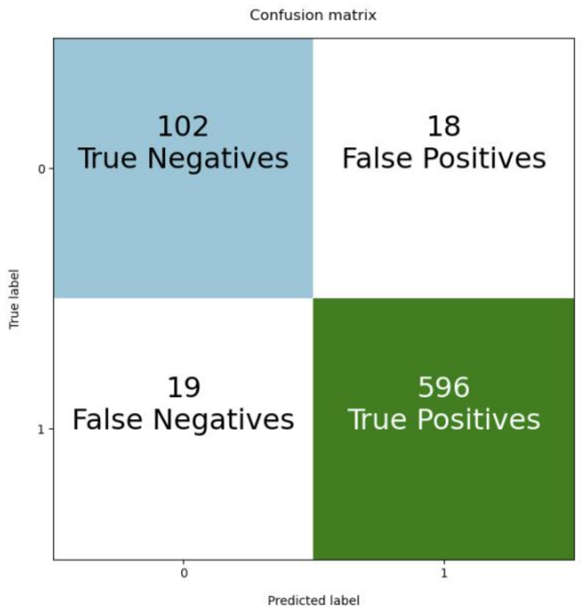


Total of 86+565 = 651 correct predictions
50+23 = 84 Incorrect predictions

Decision tree classifier predicted 651 correct predictions and only 84 wrong predictions. Let's look at the classification report to enhance this.

```
                     precision    recall  f1-score   support

developed countries       0.63      0.72      0.67       120
developing countries      0.94      0.92      0.93       615

           accuracy                           0.89       735
          macro avg       0.79      0.82      0.80       735
       weighted avg       0.89      0.89      0.89       735
```

Recall is 0.89 which is relatively higher for a model, and we have got a accuracy of 89% with only 2 features.
Our next task is to check whether the classifier performance changes with increase in number of features into the model.
Performed same steps same as before for feature selection (As we done in logistic reg) and used those features in decision tress as well.
Repeat the steps of learning the model and prediction of test data. We obtain the following test results. Lets see the confusion matrix.



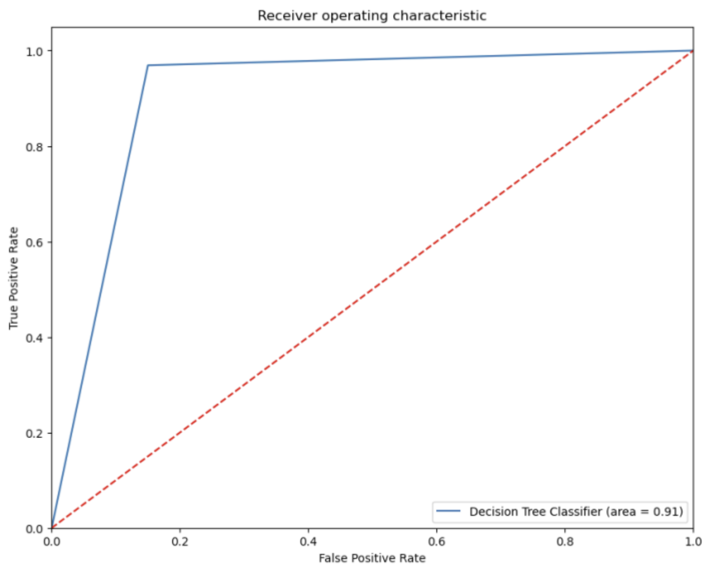
Decision tree classifier has predicted total 102+596 = 698 correct predictions and 37 Wrong predictions. Model has nearly predicted every observation right. Now will see the classification report

```
                     precision    recall  f1-score   support

developed countries       0.84      0.85      0.85       120
developing countries      0.97      0.97      0.97       615

           accuracy                           0.95       735
          macro avg       0.91      0.91      0.91       735
       weighted avg       0.95      0.95      0.95       735
```

We can see that model has improved a lot with more "significant" features. Recall has been increased to 0.91 and the model has the accuracy of 95%, which is excellent for any model.

Lastly, let us examine at the RoC Curve and AuC,

We already know that a effective classifier stays as far away from that line the line as much as possible (toward the top-left corner).



We can see that the decision tree classifier is far away from random regressor and the area under curve (AoC) is 0.91.

Conclusions:

1. The logistic and Decision tree performed almost similar when there are 2 input features , both had a accuracy of 89%. Also, both models performed well on classifying whether the status is developing or developed country when we are given Life expectancy and schooling.
2. When we considered all the significant features, it is evident that Decision tree classifier outperforms the logistic regression on classifying the status. Decision tree model had a accuracy of 95%
3. Confusion matrix and RoC helped to determine the model's performance.

References:

1. https://towardsdatascience.com/building-a-logistic-regression-in-python-step-by-step-becd4d56c9c8
2. https://github.com/Avik-Jain/100-Days-Of-ML-Code/blob/master/Code/Day%2025%20Decision%20Tree.md
3. https://www.lucidchart.com/pages/decision-tree