Explorations and Caveats: Some Comments and Insights on the Exploratory Data Analysis Process

- By Devjyoti Chakrabarti

Introduction

This is Part I in a 2-part series of some explorations of data tools and approaches to analysis. This part of the series had began as an attempt to perform some exploratory analysis on a given dataset. However as I progressed through the analysis, I realized the small nuances and details at various stages of this analysis which might lead us to draw very different conclusions from the same data. This report draws on these nuances to detail some caveats and insights on exploratory data analysis (EDA) performed through Python and R. In Part II of the series, I will build my own dataset through API calls to websites and carry out an EDA while keeping track of these nuances and caveats.

The Colour

There is a plethora of rich datasets available with data being collected by international agencies (World Bank, IMF, UN), governments of different countries and even non-profit data repositories (like Our World in Data). This project has been performed on the Human Development Index data from 1990-2019 obtained from the UNDP data centre. There are 3 main libraries that we will be working with, “pandas” for storing data and data manipulations, “plotly” for all of the data visualisations and “sklearn” which is one of the most vast and powerful libraries for machine learning.

```
#Libraries
import pandas as pd
import numpy as np
import plotly.express as px
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import FunctionTransformer
from sklearn.linear_model import LinearRegression
```

After initializing the dataset through the “read_excel” function of “pandas”, some preprocessing had to be done. While for most of the developed world, the HDI data including information for all constituent parameters was readily available going as far as back as 1990, for a lot of the countries in the Third World this data becomes available only much later in 2000 and even as late as 2010 in some cases (This is an important thing to note which we will come back to later). In the dataset, these countries only had blanks in the HDI columns for these years. All these null values have been dropped from the dataset.

```
data = pd.read_excel("HDI.xlsx") #Initializing the Data in a Dataframe
data = data.dropna() # Dropping all Null Values
```

The goal of EDA is to look for patterns in our data, observe the presence of any outliers and errors and get a deeper understanding of the relationships between variables. The easiest and most insightful way of doing this is through graphical representations of the dataset and trying to observe such patterns. Using the "plotly" library, a choropleth of the HDI over the years is plotted as shown below.

```
fig = px.choropleth(
    data_frame = data,
    locations = data["ISO-3 Code"],
    color = data["Human development index (HDI)"],
    hover_name = data["Country"],
    animation_frame = data["Year"],
    animation_group = data["Country"],
    title = "Human Development Index 1990 - 2021",
    color_continuous_scale = "rdbu",
    width = 1250,
    )

fig.update_layout(
    font_family = "Times New Roman",
    font_size = 12,
    legend_itemclick = "toggleothers",
    paper_bgcolor = "#FAF9F6",
    hovermode = "closest",
    hoverlabel_bgcolor = "#FAF9F6",
    hoverlabel_bordercolor = "black",
    transition_duration = 5,
    transition_easing = "exp-in-out"

)

fig.show()
```

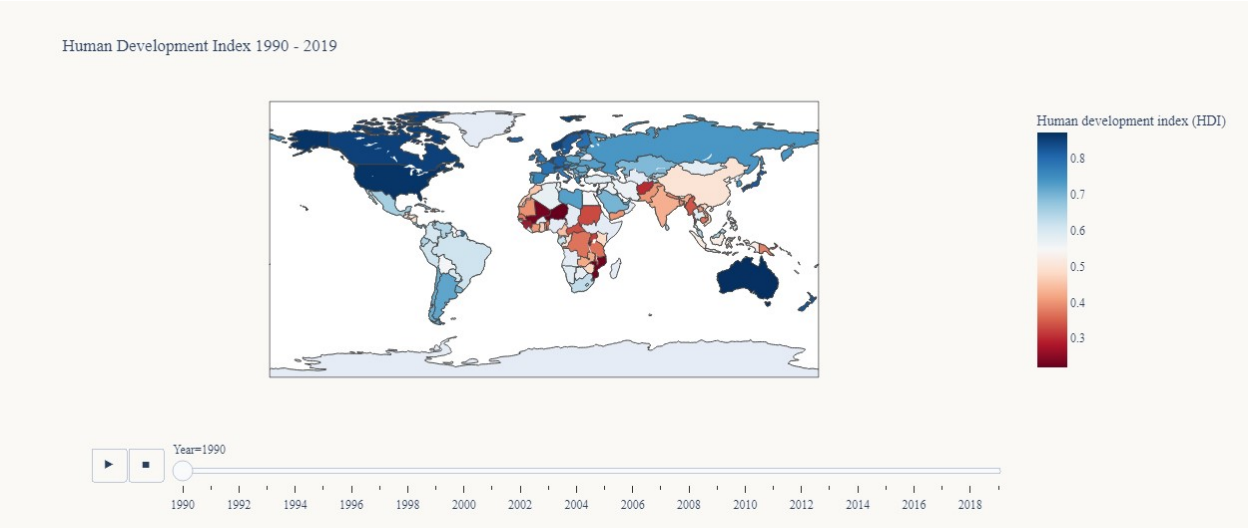

Figure 1. Human Development Index 1990

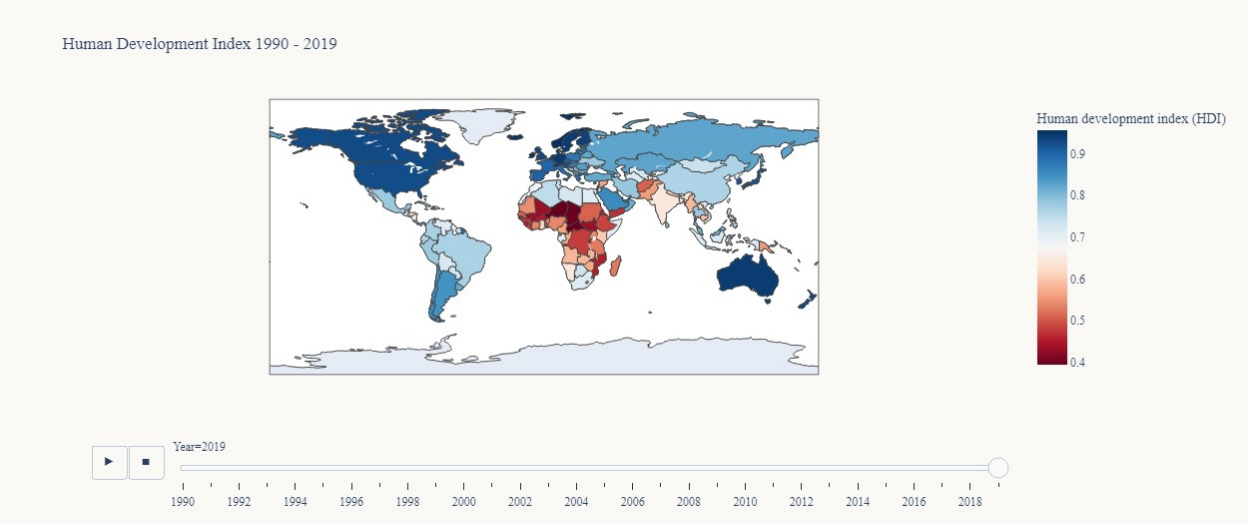

Figure 2. Human Development Index 2019

There are three things which seeing this plot points towards. 1) Much of the developed First World Countries in Europe and North America begin with an already high HDI value in 1990 which they maintain over the next 30 years. 2) Countries in Asia show significant signs of improvement with China, India, Bangladesh, Nepal etc. succeeding in providing a much better quality of life for their citizens. 3) The experience of Africa is mixed. In the two snapshots of the choropleth above, over time the situation in Africa appears to get worse. By adding the following attribute in the "px.choropleth" function we can understand more deeply what is happening here.

```
scope = "africa" #Focusing only on the African continent
```

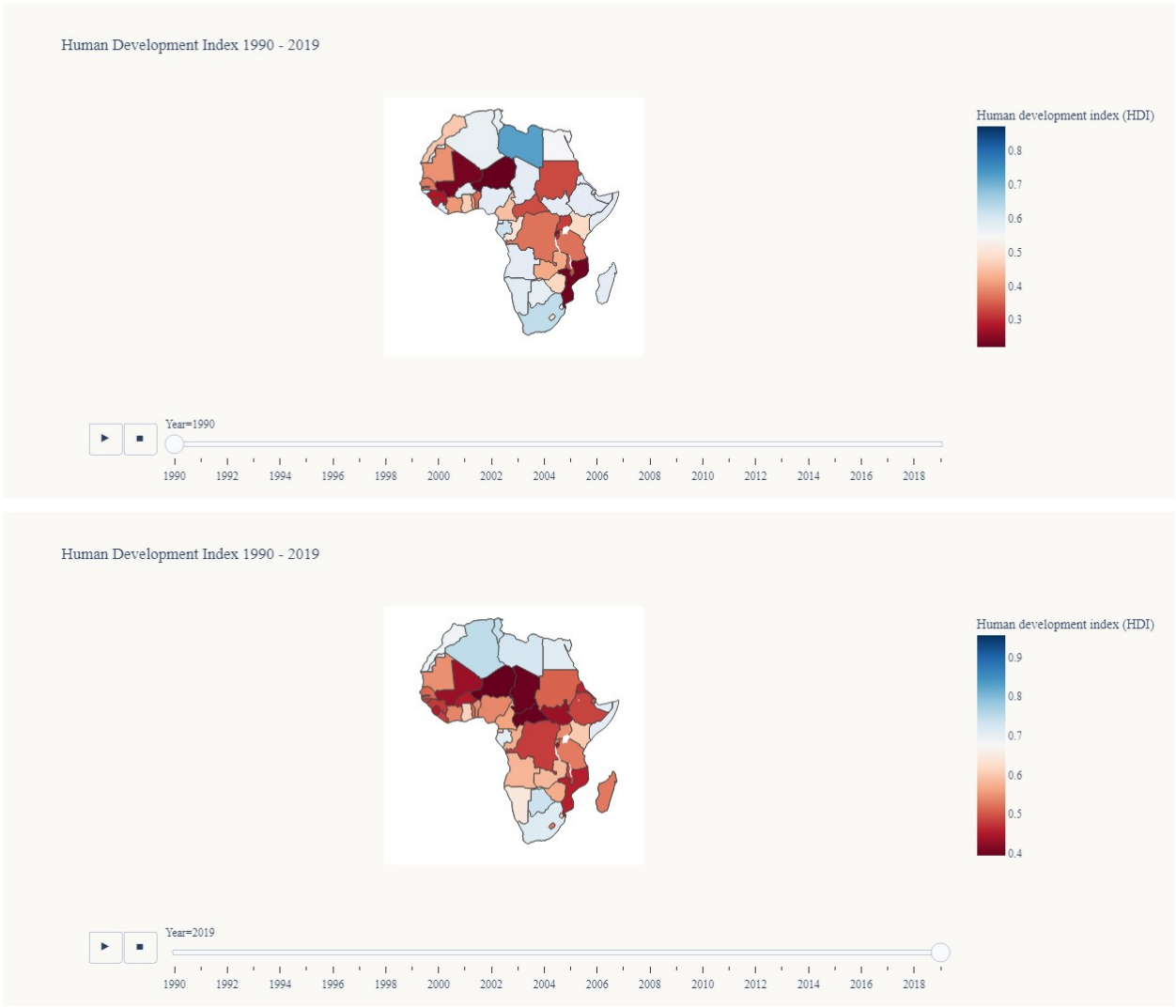

Figure 3. Comparison of HDI in Africa in 1990 and 2019

The most intuitive and seemingly correct conclusion to draw from Figure 3 is that since there is more red in 2019 than in 1990, HDI in Africa has been deteriorating. However, once you start digging deeper into the data and the functions used to plot this data, our conjectures from Figure 3 would start appearing a lot more dubious. There are two reasons why I believe this conclusion is wrong. 1) As noted before, data on HDI for many countries in this region only begins to be collected much later than 1990. 2) The choropleth function of Plotly, automatically adjusts the legend as we go through the animation. A quick look at the legend in the above Figure verifies this.

This brings us to ask a related question. Which countries have improved/deteriorated the most in terms of HDI values over the years? To circumvent the problem in point 1 above, we only consider data from 2010 - 2019. A function is written that queries our dataset to get HDI values for each country in 2010 and 2019, and then calculates the percentage change in HDI values in

this decade. For the problem in point 2, we plot the same choropleth again in a more suitable colour scheme.

```
def get_HDI(country):
    hdi_2010 = float(data[(data["Country"] == country) & (data["Year"] == 2010)]["Human development index (HDI)"])
    hdi_2019 = float(data[(data["Country"] == country) & (data["Year"] == 2019)]["Human development index (HDI)"])
  percent_change = round(((hdi_2019 - hdi_2010) * 100 / hdi_2010), 4)
  return(percent_change)

table = pd.DataFrame(columns = ("Country", "Percentage Change"))
table["Country"] = data["Country"].unique()
percent_change = list()

for index in table.index:
  country = table["Country"][index]
  percent_change.append(get_HDI(country))

table["Percentage Change"] = percent_change
```

First, consider the below output for countries where this percentage change is negative. Out of the 7 countries where HDI values declined over this period of 2010-2019, 4 countries are in the conflict-embroiled region of the Middle East, and only 1 country from Africa is on this list.

```
table[(table["Percentage Change"] < 0)].sort_values(by = ["Percentage Change"], ascending = True)
```

Country	Percentage Change
Syrian Arab Republic	-15.6250
Libya	-9.2732
Yemen	-7.1146
Venezuela (Bolivarian Republic of)	-6.0766
Timor-Leste	-3.5032
Lebanon	-2.8721
Jordan	-1.0855

Table 1. Countries where HDI has declined from 2010-2019

Second, we get the top 15 countries which have seen the largest percentage increase in HDI values in this decade. 12 out of these 15 countries are in Africa which further supports the conjecture that the deteriorating conditions that are reflected by the choropleths in Figure 1-3 might not be representative of the actual state of things.

```
table.nlargest(25, "Percentage Change")
```

Country	Percentage Change
Eswatini	19.8039
Niger	19.0332
Zimbabwe	18.4647
Burkina Faso	17.7083
Djibouti	15.4185
Ethiopia	15.2019
Côte d'Ivoire	14.9573
Guinea	14.6635
Lesotho	14.5652
Bhutan	13.9373
Mozambique	13.7157
Bangladesh	13.4650
Sierra Leone	13.2832
Myanmar	13.2039
Angola	12.3791

Table 2. 15 Countries with the highest percentage increase in HDI 2010-2019

We plot the same choropleth again using Plotly but this time the colour of the plot has been changed using the following attribute. This gives us a slightly better visualization than before.

```
color_continuous_scale = "pubu"
```

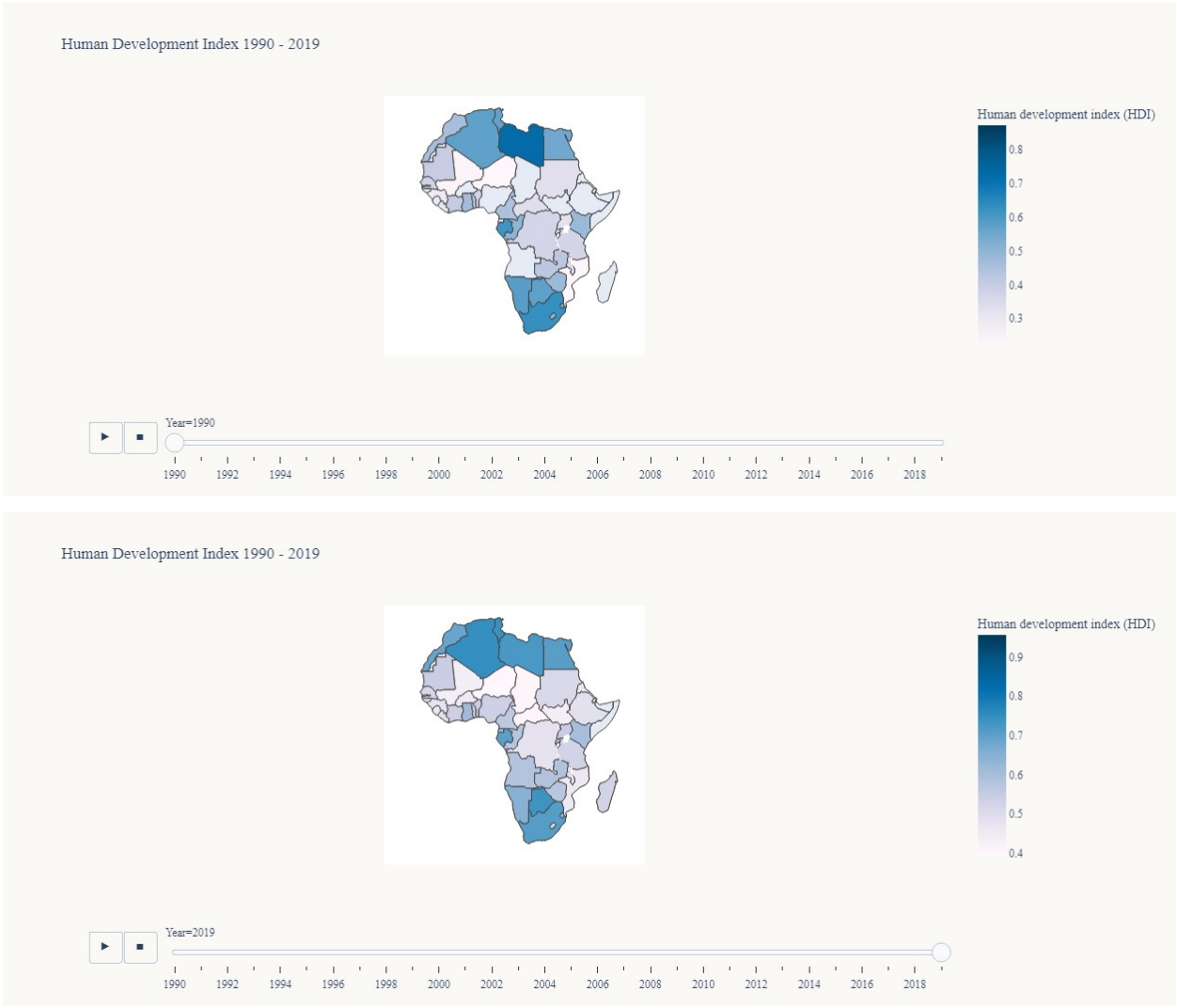

Figure 4. Comparison of HDI in Africa in 1990 and 2019

Regression

We now move on to the next stage of the EDA process. Once some data visualizations have been done, usually we would have some rough ideas about what the relationships exist between the different variables. The most popular method of testing these relationships is through regression analysis. There are three steps to this; get our data, preprocess it and build our model.

```
hdi_2021_data = pd.read_excel("HDI_2021.xlsx") #Obtaining the Data
```

Preprocessing includes handling missing values, feature scaling extracting and reducing features. It is always a good idea to properly scale the data as some ML techniques tend to be sensitive to scaling. This is one of the lucky situations where UNDP has elaborate definitions and examples of how the data is scaled for HDI and we know exactly how to scale our dataset. The UNDP after collecting information on life expectancy, expected years of schooling, mean years of schooling, and gross national income (GNI) per capita, converts this into indices for health, education and income using formulae which can be viewed at https://hdr.undp.org/sites/default/files/2021-22_HDR/hdr2021-22_technical_notes.pdf.

```
#Feature scaling
hdi_2021_data["Health Index"] = (hdi_2021_data["Life expectancy at birth"] -
20) / 65
hdi_2021_data["Education Index"] = ((hdi_2021_data["Expected years of
schooling"] / 18) + (hdi_2021_data["Mean years of schooling"] / 15)) / 2
hdi_2021_data["Income Index"] = (np.log(hdi_2021_data["Gross national income
(GNI) per capita"]) - np.log(100)) / (np.log(75000) - np.log(100))
```

Building the model is made extremely easy by the "sklearn" library. We split our dataset into training and test datasets and fit a standard linear regression model on the training dataset and observe the performance of our model on the test dataset.

```
#Model Building
SEED = 42

y = hdi_2021_data.iloc[:, 5 ]  #Response Variable
X  =  hdi_2021_data[["Health  Index",  "Education  Index",  "Income  Index"]]
#Independent Variables

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2)

reg = LinearRegression().fit(X_train, y_train)
y_pred = reg.predict(X_test)
print("OLS R2:", reg.score(X_test, y_test))
```

We manage to get a near-perfect R^2 value of 0.9984. But there is still room for improving this model. HDI is calculated as the geometric mean of the three indices.

$$HDI = (I_{Health} * I_{Education} * I_{Income})^{1/3}$$

So ideally we want to do 2 things. 1) We want to raise our independent variables to a particular value, i.e. X^i and then train our model on this transformed value of the independent variable. 2) Find what is the optimal value of i for which our independent variables should be raised. Using the power function of Numpy we can perform the above transformation. To find the optimal value, we iterate through some possible values of i in a given range and try to find the value which maximizes R^2. This is similar in principle to the GridSearchCV function of sklearn library (https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html).

```
#Finding Optimum i
model_scores = dict()
for i in np.arange(-5, 5, 0.01):
  X_train_new = np.power(X_train, round(i, 2))
  X_test_new = np.power(X_test, round(i, 2))

  model_new = LinearRegression().fit(X_train_new, y_train)
  model_scores[f'model_{round(i, 2)}'] = model_new.score(X_test_new, y_test)


print(max(zip(model_scores.values(), model_scores.keys())))
```

```
#Linear Regression on the Optimum i
X_train_new = np.power(X_train, 0.83)
X_test_new = np.power(X_test, 0.83)

best_model = LinearRegression().fit(X_train_new, y_train)
improved_y_pred = best_model.predict(X_test_new)
print("Modified Regression R2:", best_model.score(X_test_new, y_test))
print("Model Intercept:", best_model.intercept_)
print("Model Coefficients:", best_model.coef_)
```

Through this we are able to improve the R^2 of the model. The Actual vs Predicted plot from this model has been plotted below.

```
scatter = px.scatter(
    x = y_test,
    y = improved_y_pred,
    labels = {"x" : "Actual Y", "y" : "Predicted Y"},
    trendline = "ols",
    trendline_color_override = "black",
    title = "Actual vs Predicted Plot for Modified Linear Regression"
```

```
)
```

```
scatter.update_layout(
    font_family = "Times New Roman",
    font_size = 12,
    legend_itemclick = "toggleothers",
    paper_bgcolor = "#FAF9F6",
    hovermode = "closest",
    hoverlabel_bgcolor = "#FAF9F6",
    hoverlabel_bordercolor = "black",
    transition_duration = 5,
    transition_easing = "exp-in-out"

)

scatter.update_traces(marker=dict(color='#DA70D6'))

scatter.show()
```

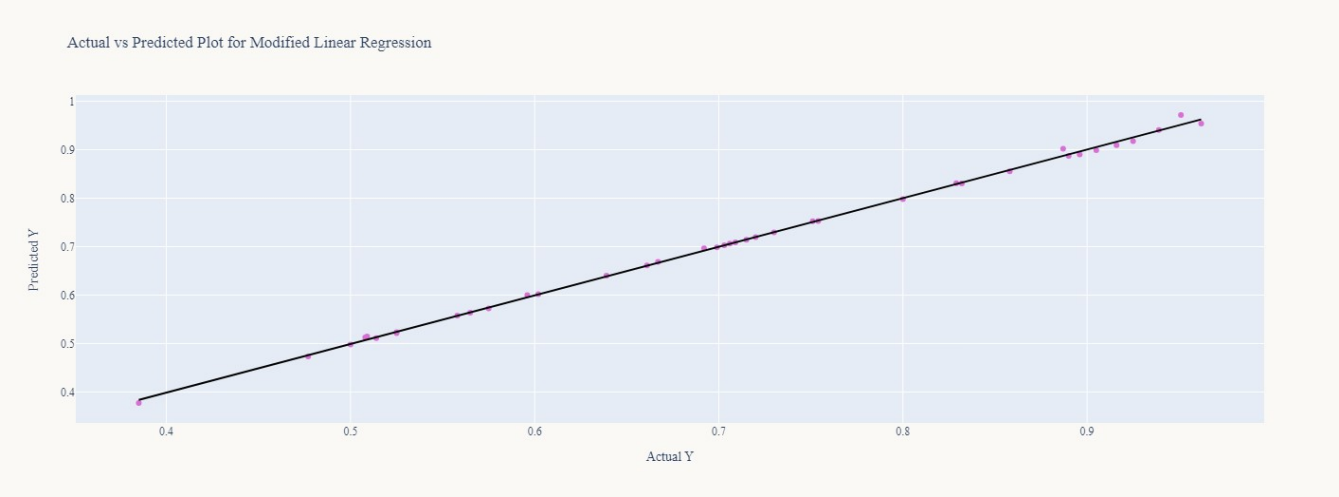

Figure 5. Actual vs Predicted Plot

Concluding Remarks

The EDA process can be childishly exciting and rewarding. Plotting data, observing relationships, and testing those relationships while continuously redoing and rethinking the whole process is a true joy. But it is also a process where seemingly credible plots and charts can be used to misunderstand insights and misconstrue data. Even in the regression model built above, the very high R^2 of the model was only possible because of the availability of properly collected and semi-processed data and well-defined definitions for feature scaling. These are

luxuries which data analysts rarely have in performing EDA. It is with this motivation that Part II of this series tries to begin from scratch and tries to build our own dataset to perform EDA on. In the appendix of this report, the code for performing the same data manipulations in R has been attached.

Appendix

```r
#Libraries
install.packages("dplyr")
install.packages("readxl")

library(dplyr)
library(readxl)

#Initialising the Data in a Dataframe
data <- read_excel('HDI.xlsx')
print(data)

# Dropping all Null Values
data_complete <- na.omit(data)

get_HDI <- function(country){
  hdi_2010 <- filter(data, Country == country & Year == 2010)[4]
  hdi_2019 <- filter(data, Country == country & Year == 2019)[4]
  percent_change <- round(((hdi_2019 - hdi_2010) * 100 / hdi_2010), 4)
  return(percent_change)
}

table <- data.frame(matrix(vector(), 0, 2, dimnames = list(c(), c("Country",
"Percentage_Change"))), stringsAsFactors = FALSE)

for (country in unique(data_complete$Country)){
  row <- data.frame(Country = country, Percentage_Change = get_HDI(country))
  table <- rbind(table, row)
}

colnames(table) <- c("Country", "Percentage_Change")
print(table)
```

```
arrange(filter(table, Percentage_Change < 0), Percentage_Change) #Countries where
HDI declined from 2010 - 2019
table %>% arrange(desc(Percentage_Change)) %>% slice(1:15) #Top 15 Countries with
the highest percentage increase in HDI

#Linear Regression

hdi_2021_data = read_excel('HDI_2021.xlsx')
colnames(hdi_2021_data)     <-     c("Country",     "Life_expectancy_at_birth",
"Expected_years_of_schooling",                          "Mean_years_of_schooling",
"Gross_national_income_per_capita", "Human_Development_Index")
print(hdi_2021_data)

#Feature Scaling
Health_Index <- c(round(((hdi_2021_data$Life_expectancy_at_birth - 20) / 65), 6))
Education_Index <- c(round((((hdi_2021_data$Expected_years_of_schooling / 18) +
(hdi_2021_data$Mean_years_of_schooling / 15)) / 2), 6))
Income_Index  <-  c(round((log(hdi_2021_data$Gross_national_income_per_capita)  -
log(100)) / (log(75000) - log(100)), 6))

hdi_2021_data    <-    cbind(hdi_2021_data,    Health_Index,    Education_Index,
Income_Index)
# print(hdi_2021_data)

# Model Building
set.seed(42)

sample <- sample.split(hdi_2021_data, SplitRatio = 0.8)
train_set  <- subset(hdi_2021_data, sample == TRUE)
test_set   <- subset(hdi_2021_data, sample == FALSE)
model <- lm(formula = Human_Development_Index ~ Health_Index + Education_Index +
Income_Index, data = train_set)
y_pred <- predict(model, newdata = test_set)
print(paste("OLS R2:", summary(model)$r.squared))

#Linear Regression on the Optimum i
best_model <- lm(formula = Human_Development_Index ~ I(Health_Index ^ (0.83) +
Education_Index ^ (0.83) + Income_Index ^ (0.83)), data = train_set)
improved_y_pred <- predict(best_model, newdata = test_set)
```

```r
print(paste("Modified Regression R2:", summary(best_model)$r.squared))

x <- test_set$Human_Development_Index
y <- improved_y_pred

plot(x = test_set$Human_Development_Index,y = improved_y_pred,
   xlab = "Actual Y",
   ylab = "Predicted Y",
   main = "Actual vs Predicted Plot for Modified Linear Regression",
   pch = 19, col = "#DA70D6"
)

abline(lm(y ~ x), col = "black")
```