World Happiness Analysis – Evolution Through the Years

**Table of Contents**

[1. Introduction 3](#__RefHeading___Toc108_1240888190)

[2. Background and Implementation 4](#__RefHeading___Toc77_4146056077)

[2.1 Spark Configuration: 4](#__RefHeading___Toc79_4146056077)

[2.2 Dataset 6](#__RefHeading___Toc81_4146056077)

[2.3 Loading Data 7](#__RefHeading___Toc83_4146056077)

[3. Data Analysis 12](#__RefHeading___Toc85_4146056077)

[3.1 Correlation Matrix 12](#__RefHeading___Toc294_42618093811)

[3.2 Linear regression 13](#__RefHeading___Toc757_1514544151)

[Reference List 20](#__RefHeading___Toc759_1514544151)

[Appendix: 21](#__RefHeading___Toc761_1514544151)

# 1. Introduction

Since the era of mass production, happiness has been a constant focus of analysis. Early researchers discovered that the overall happiness of an individual is positively correlated with his productivity. Because of this correlation, many countries have started developing institutes to learn more about the root origin of happiness and to prevent it, since unhappiness leads to lower productivity and greatly affects the sociological aspect of life. In the last decade, society has suffered from various sources of crisis, therefore it is vital to analyse this subject in more depth. Also, because people tend to make hedonistic choices regarding their lives, the choice of happiness becomes a great part of an individual’s life. Ergo, studying the determinants of happiness and trying to come up with new solutions regarding this topic is of the upmost importance.

Nevertheless, how is happiness measured? Happiness is a state of mind, which means that quantifying an emotional state may be a difficult task to achieve. Luckily, researchers from the Sustainable Development Solutions have come with a methodology to quantify relative happiness around the globe (World Happiness Report, n.d.). However, we still need to analyse and perform various tests to check whether the determinants of happiness are correlated with happiness itself.

For this, report a dataset from Kaggle will be used to make the required analysis. The Dataset consists of panel data, from 2008 to 2022 from various countries around the globe. It will consist of 12 Features and one target variable. A Linear Regression will be performed to check if the determinants of happiness are statistically significant and make inferences about them. All of the Data Pre-processing and analysis will be made using pyspark, on a Linux-based Partition.

Visual graphics will also be presented in this report, to provide the reader with easier and retrievable information. All of the visual aids will be made exclusively with Tableau, on Windows 11.

Therefore, the objective of this report is to provide insightful ideas of how happiness changed over the years, especially the impact it had in pandemics or crisis, and to check whether happiness has been increasing or decreasing in the last decade and a half. This is going to be accomplished with the help of common classification techniques, data analysis and the creation of charts. In the end, this report is expected to explain in detail how happiness has evolved over the years

# **2. Background and Implementation**

## 2.1 Spark Configuration:

Spark is a multi-language engine for executing data engineering, data science, and machine learning on single-node machines or clusters. It is mainly used for large-scale data processing (Apache Spark, n.d.). In this report, we will use pyspark, which is a Python Application Programming Interface (API) for Spark. In other words, we can use Spark software using the high-level programming language python. However, we need to set it up first. All of the following steps were made using Ubuntu 22.0.4 LTS, a linux-based operating system.

First, we need to install the Java Development Kit (JDK) using the following command line:

sudo apt-get install default-jdk

To check if Java was properly installed, we can use the following line:

java -version

Afterwards, we must install the spark package from the following website:

<https://spark.apache.org/downloads.html>

The package used in this report was the following:

[spark-3.2.3-bin-hadoop3.2.tgz](https://www.apache.org/dyn/closer.lua/spark/spark-3.2.3/spark-3.2.3-bin-hadoop3.2.tgz)

With the following link, we create a new directory on Ubuntu using this code:

mkdir -p /opt/spark

Afterwards, we select the directory that we have just created:

cd /opt/spark

Then, we download the package to the directory that we have selected:

sudo wget https://dlcdn.apache.org/spark/spark-3.2.3/spark-3.2.3-bin-hadoop3.2.tgz

After the installation is complete, we must extract the files to a folder:

tar -xvf spark-3.2.3-bin-hadoop3.2.tgz

After the files are extracted, we need to set up the Environment Variables. First we edit the bashrc using this line:

vi ~/.bashrc

At the end of the file, we paste the following text:

SPARK\_HOME=/opt/spark/spark-3.2.3-bin-hadoop3.2

export PATH=$PATH:$SPARK\_HOME/bin:$SPARK\_HOME/sbin

Afterwards, we just save the file and Source the file using:

source ~/.bashrc

The final step is to ensure that Spark is properly installed. We just need to write:

spark-shell

Inside the output,there will be a line similar to this:

Spark context Web UI available at [http://192.168.0.10:4040](http://192.168.0.10:4040/)

We just need to open the http to check if spark is running properly in the system.

Now, Pyspark should run in a python environment without any problems or warnings.

## 2.2 Dataset

The dataset that will be used in this report contains yearly information about “happiness score”,i.e. life ladder, of 165 countries, which is going to be the target variable. The dataset also contains some other determinants. The determinants of happiness contained in this dataset are as follows, with a brief explanation of what they are:

* **Log GDP per capita** – Average Gross Domestic Product per person at constant prices (Inflation-Adjusted). Higher the Better;
* **Social Support** – Binary Response Survey (0-1), to the question “If you were in trouble, do you have relatives or friends you can count on to help you whenever you need them, or not?” (Buttar, 2023). Higher is Better;
* **Healthy Life Expectancy At Birth** – On average, the amount of years that a newborn will live. Higher is Better;
* **Freedom To Make Life Choices** – Binary Response Survey to the question “Are you satisfied or dissatisfied with your freedom to choose what you do with your life?” (Buttar, 2023). Higher is Better.
* **Generosity** – Residual of regressing the national average responses to the donation question “Have you donated money to a charity in the past month?” on log GDP per capita (Buttar, 2023). Higher is Better.
* **Perceptions Of Corruption** – Binary Response Survey to the question “Is corruption widespread throughout the government or not?” and “Is corruption widespread within businesses or not?” (Buttar, 2023). Lower is better.
* **Positive Affect** - Average of previous-day effects measures for laughter, enjoyment, and interest. The general form for the affect questions is “Did you experience the following feelings during a lot of the day yesterday?” (Buttar, 2023). Higher is Better.
* **Negative Affect**- Average of previous-day effects measures for worry, sadness, and anger (Buttar, 2023). Lower is Better.
* **Confidence In National Government** -

There are other variables, such as Country Name, Regional Indicator and Year, however, these are variables only to define space/time parameters that are axiomatic.

## 2.3 Loading Data

In order to load the data, we have used the following code:

**from** **pyspark.sql** **import** SparkSession

**from** **pyspark** **import** SparkContext

**import** **os**

os.environ['SPARK\_LOCAL\_IP'] = "192.168.0.10" # Set local ipv4 Address

sc = SparkContext.getOrCreate()

print(sc.version)

# create a SparkSession

spark = SparkSession.builder.appName("csv\_file").getOrCreate()

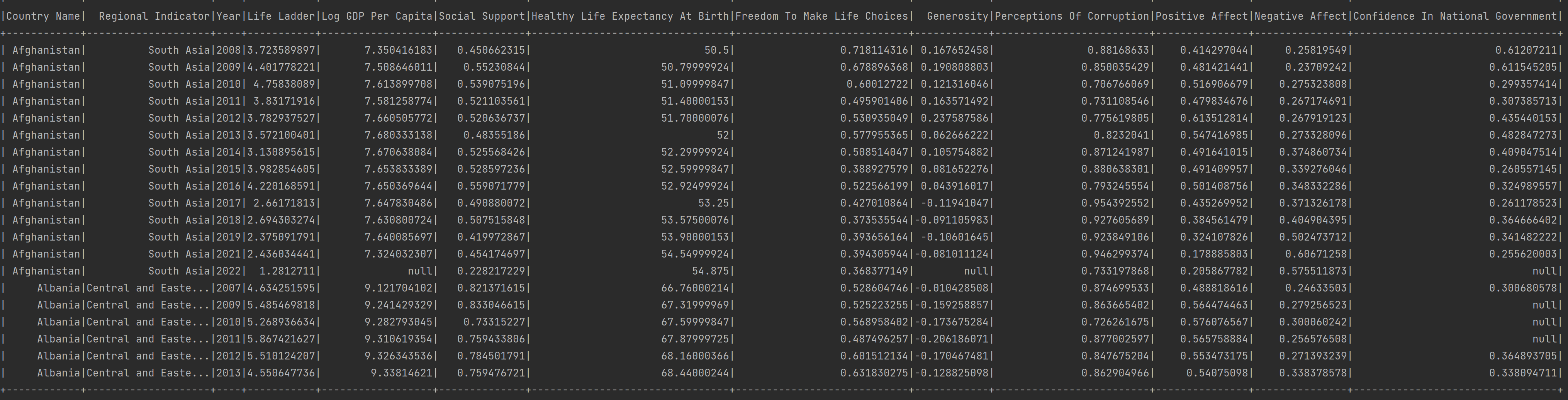
# read the CSV file into a Spark DataFrame

df = spark.read.format("csv").option("header", "true").load("happiness.csv")

We can see the dataset format using:

df.show()

Which will give the following output:



We can see from the screenshot, how the dataset is formatted.

We can see which columns have missing values by using these lines of code:

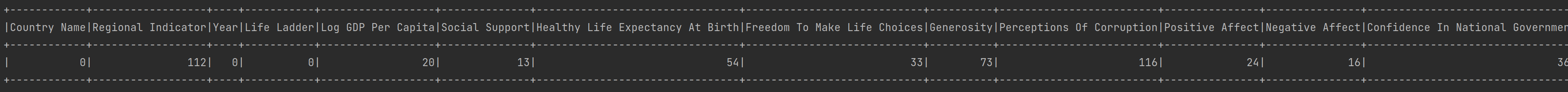
# Count the number of null values in each column

missing\_count = df.select([sum(col(column).isNull().cast("int")).alias(column) **for** column **in** df.columns]) # (NNK, 2021)

# Show how many missing values in each column

missing\_count.show()

Output:

As we can see from the screenshot taken above, there are some missing values in the Regional Indicator Column. After reviewing the data, it seems that there are countries with many missing values that lack the information of their Regional Indicator, therefore removing these countries will not affect the analysis that will be made to this dataset. We can remove the rows that have no values in the “Regional Indicator” column by using:

# Drop Missing Values on Regional Indicator

df = df.dropna(subset=["Regional Indicator"])

Now, we have cleaned the dataset from unecessary information.

However, there are still some missing values from the dataset. However, we can fill the missing values with imputation methods. Because most columns are based on survey responses, we can fill the missing values with the average of each country for each column by using the window function provided by pyspark:

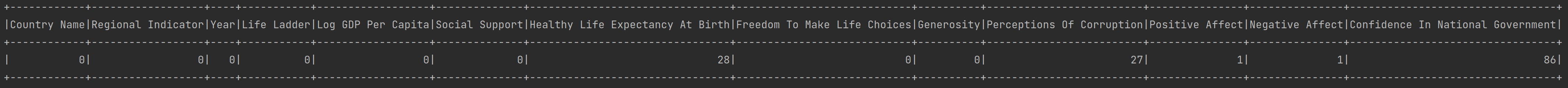
# Define a Window, that is divided by 'Country Name' (Davis, 2022)

w = Window.partitionBy('Country Name')

# Fill missing values with the mean of each column for each country

df = df.select("Country Name", "Regional Indicator", \*[when(col(c).isNull(), avg(col(c)).over(w)).otherwise(col(c)).alias(c) **for** c **in** df.columns[**2**:]])

With these lines of code, we were able to reduce the number of missing values to this:



Therefore, because there is a very low percentage of missing values (compared to the number of observations), we can make inferences without any compromises on quality.

##################################################

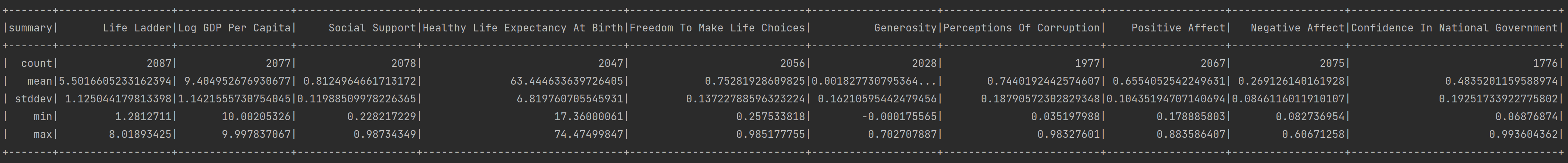
We can compute the basic statistics for each column to see if we need to normalize the values. We can achieve this by using the following code:

# (Apache, n.d.)

sum\_stats = df.drop("Country Name","Regional Indicator","Year") # No use in asssesing these columns

sum\_stats.describe().show()

Output:



We can see from the values above that there is not much dispersion from each column to another. This is because most columns are binary-based, which means that values range between 0 and 1, as shown in the min/max row. Regarding their deviation, there are no much difference, except in the Healthy Life Expectancy at Birth. Nevertheless, it is not a difference that will affect results, therefore no normalisation needs to be made.

Ergo, because most columns have similar classifications and low-dispersed values, normalisation is not needed, therefore the process of data-preprocessing is finished.

## 

# 3. Data Analysis

In this section, we a Linear Regression will be made to check whether the determinants of well-being are statistically significant or not. We will conduct several tests to verify the Gauss-Markov assumptions, so that the regression can be inferred correctly.

## 3.1 Correlation Matrix

Before making the linear regression, we will check the correlation between the variables that will be used in the regression. We can achieve this by using the Correlation function from pyspark.

#(How to Get the Correlation Matrix of a Pyspark Data Frame?, 2018)

# Assemble the columns into a vector column

vectorAssembler = VectorAssembler(inputCols=columns, outputCol="features", handleInvalid="skip")

vector\_df = vectorAssembler.transform(df).select("features")

# Calculate the correlation matrix

correlation\_matrix = Correlation.corr(vector\_df, "features").head()

# Convert to Array

corr\_matrix = correlation\_matrix[**0**].toArray()

corr\_df = pd.DataFrame(corr\_matrix, columns=columns, index=columns) # Converted to Pandas Dataframe

# Print the correlation matrix

pd.set\_option("display.max\_columns", **None**)

print(corr\_df)

The results given by the code above are displayed on the Appendix.

We can infer fro the values that most of the independent variables do not suffer for collinearity. However, if we focus on Log GDP per capita, we can see that it has a very high correlation with Social Support and Healthy Life Expectancy at Birth. In the Regression, it may cause biased results. However, we must tread carefully and make necessary adjustments to overcome the multicollinearity problem.

## 3.2 Linear regression

To check the strength/significance of the determinants of Life-Ladder (i.e.,Happiness Score), we will conduct a Pooled Ordinary Least Squares. This method can be done using built in functions from pyspark. In this linear regression, it was performed cross-validation to fetch the best regParam (Parameter usually used to avoid over-fitting/under-fitting), according to the best Mean Squared Error value (MSE). In other words, lower the MSE, better the model. We have also split the dataset into testing, training and validation data to further tune the hyper-parameters and to check whether there is over-fitting of the model. The following code was used:

""" Linear Regression Section. The following code was based on (Hejazi, 2023), and derived using some other parameters in Pyspark Documnetation"""

# Define the list of determinants and the dependent variable

determinants = ["Log GDP Per Capita", "Freedom To Make Life Choices", "Generosity", "Perceptions Of Corruption",

"Negative Affect", "Positive Affect", "Social Support"]

target = "Life Ladder"

# Combine the determinants into a single features column

assembler = VectorAssembler(inputCols=determinants, outputCol="features", handleInvalid="skip")

# Transform the df

data = assembler.transform(df)

# Split data into training, validation and testing

(trainingData, validationData, testData) = data.randomSplit([**0.6**, **0.2**, **0.2**],

seed=**42**) # (pyspark.sql.DataFrame.randomSplit — PySpark 3.1.3 Documentation, n.d.)

# Define initial values and set of possibilities for the loop to iterate

reg\_params = np.arange(**1**, **50**, **1**)

best\_mse = float("inf")

best\_reg\_param = **None**

**for** reg\_param **in** reg\_params: # Loop through the regParam values

# Train a linear regression model with the current regParam value

lr = LinearRegression(featuresCol="features", labelCol=target, solver="auto", fitIntercept=**True**,

regParam=reg\_param) # (LinearRegression, n.d.)

model = lr.fit(trainingData)

# Make predictions on the validation set

predictions = model.transform(validationData) # (ML Tuning - Spark 3.4.0 Documentation, n.d.)

# Evaluate the Mean Squared Error (MSE) on the validation set

evaluator = RegressionEvaluator(labelCol=target, predictionCol="prediction", metricName="mse")

mse = evaluator.evaluate(predictions)

**if** mse < best\_mse:

best\_mse = mse # If current Mean Squared Error is lower, it becomes the Best MSE

best\_reg\_param = reg\_param # If the current MSE is lower, then current RegParam is the best

# Combine the training and validation sets

train\_val\_data = trainingData.union(validationData) # Increase number of samples to improve performance

# Fit the final linear regression model using the best regParam on the combined Training/Validation datasets

lr = LinearRegression(featuresCol="features", labelCol=target, solver="auto", fitIntercept=**True**,

regParam=best\_reg\_param)

model = lr.fit(train\_val\_data)

# Fetch Summary Values

summary = model.summary

# Fetch P-values

p\_values = summary.pValues

# Make predictions on the testing data. This is done to evaluate the model's performance on unseen data

# and ensure that the MSE of the combined training/validation set is similar to the testing set.

predictions = model.transform(testData)

# Evaluate the model's mean squared error (MSE) on the testing set

evaluator = RegressionEvaluator(labelCol=target, predictionCol="prediction", metricName="mse")

test\_mse = evaluator.evaluate(predictions)

# Print the results

print("Best regParam:", best\_reg\_param)

print("Coefficients:", list(zip(determinants, model.coefficients)))

print("P-Values:", list(zip(determinants, p\_values)))

print("R Squared:", summary.r2)

print("Training and Validation Mean Squared Error:", best\_mse)

print("Testing Mean Squared Error:", test\_mse)

The results of the model were the following:

Best regParam: **1**

Coefficients: [('Log GDP Per Capita', **0.2727669228154929**), ('Freedom To Make Life Choices', **0.8311402705463417**), ('Generosity', **0.2795709411428252**), ('Perceptions Of Corruption', -**0.6053910778416889**), ('Negative Affect', -**0.4640227491046634**), ('Positive Affect', **1.464969243905528**), ('Social Support', **1.9718422107156515**)]

P-Values: [('Log GDP Per Capita', **0.0**), ('Freedom To Make Life Choices', **0.0**), ('Generosity', **8.636806030182242e-05**), ('Perceptions Of Corruption', **0.0**), ('Negative Affect', **0.0006502242051695806**), ('Positive Affect', **0.0**), ('Social Support', **0.0**)]

R Squared: **0.7004978770138346**

Training **and** Validation Mean Squared Error: **0.3733870399493276**

Testing Mean Squared Error: **0.3725522038161966**

We can see from the values above that all coefficient signs make sense. For instance, it is known that if a Country has higher GDP per capita, happiness is likely to be higher. In a different case, if a country is perceived to have a corrupted government or high corruption on the private sector, it is more likely to see their happiness decreased, since corruption does only allocate the money to a very small percentage of people, leaving individuals/families harmed.

We can also determine that all p-values are statistically significant. We can see that some p-values are extremely close to 0 (See Appendix), which is a good sign, however it may also be an indication of overfitting.

The R-Squared tells us the variance that is explained in the independent variables. In other words, it means that approximately 70% of the variance is explained by the predictors, which is a very high number. However, it may be also a case of overfitting. However, due to the division of the dataset into three, we were able to check the Mean Squared Error of the Training/Validation and the Testing/Set. The inferences we can make from this is that if the MSE’s values are close to one another, the chances of a overfitted model are lower. Ergo, as we can see, the value are similar to the 2nd decimal point, which is very promising.

To summarize, we have conducted a linear regression of Well-being against the determinants, and found out that all determinants are statistically significant, and they show a very strong relationship with well-being. Therefore, we can start conducting the visual analysis for better acuity.

# **Reference List**

Hejazi, Y. (2023, January 11). Beginner’s Guide to Linear Regression with PySpark - Towards Data Science. Towards Data Science. Retrieved April 15, 2023, from <https://towardsdatascience.com/beginners-guide-to-linear-regression-with-pyspark-bfc39b45a9e9>

ML Tuning - Spark 3.4.0 Documentation. (n.d.). Apache Spark. Retrieved April 15, 2023, from <https://spark.apache.org/docs/latest/ml-tuning.html>

pyspark.sql.DataFrame.randomSplit — PySpark 3.1.3 documentation. (n.d.). Apache Spark. Retrieved April 15, 2023, from <https://spark.apache.org/docs/3.1.3/api/python/reference/api/pyspark.sql.DataFrame.randomSplit.html>

World Happiness Report. (n.d.). About. The World Happiness Report. Retrieved April 10, 2023, from <https://worldhappiness.report/about/>

Apache Spark. (n.d.). *Unified Engine for large-scale data analytics*. Retrieved April 10, 2023, from <https://spark.apache.org/>

Buttar, U. (2023, March 25). World Happiness Report, 2005-Present. Kaggle. Retrieved April 12, 2023, from <https://www.kaggle.com/datasets/usamabuttar/world-happiness-report-2005-present>

NNK. (2021, April 9). PySpark – Find Count of null, None, NaN Values. SparkBy{Examples}. Retrieved April 12, 2023, from <https://sparkbyexamples.com/pyspark/pyspark-find-count-of-null-none-nan-values/>

Apache. (n.d.). pyspark.sql.DataFrame.summary. Apache Spark. Retrieved April 13, 2023, from <https://spark.apache.org/docs/3.1.2/api/python/reference/api/pyspark.sql.DataFrame.summary.html>

Davis, J. (2022, July 7). Group by does not maintain order in Pyspark; use a window function instead. Medium. Retrieved April 13, 2023, from <https://medium.com/@davisjustin42/group-by-does-not-maintain-order-in-pyspark-use-a-window-function-instead-9178e70497a9>

How to get the correlation matrix of a pyspark data frame? (2018, September 7). Stack Overflow. Retrieved April 14, 2023, from <https://stackoverflow.com/questions/52214404/how-to-get-the-correlation-matrix-of-a-pyspark-data-frame>

LinearRegression. (n.d.). Apache Spark. Retrieved April 14, 2023, from <https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.ml.regression.LinearRegression.html>

# **Appendix:**

Although some of the p-values displayed are 0, it does not mean that their values is zero. It is statistically impossible to have a p-value of 0. This happens because of floating precision that python uses. What is displayed here is a value that is so close to zero, that the interpreter classifies it as zero. For instance:

Number: 0.0000000000000000000000000123

Python Interpreter: 0

Correlation Matrix.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Correlation Matrix** | **Life Ladder** | **Log GDP Per Capita** | **Social Support** | **Healthy Life Expectancy At Birth** | **Freedom To Make Life Choices** | **Generosity** | **Perceptions Of Corruption** | **Positive Affect** | **Negative Affect** | **Confidence In National Government** |
| **Life Ladder** | 1 | 0.791368 | 0.727331 | 0.727285 | 0.528783 | 0.187623 | -0.462765 | 0.518622 | -0.332615 | -0.086882 |
| **Log GDP Per Capita** |  | 1 | 0.692295 | 0.832092 | 0.359391 | -0.002487 | -0.359260 | 0.244602 | -0.272800 | -0.199951 |
| **Social Support** |  |  | 1 | 0.598294 | 0.418579 | 0.076351 | -0.241249 | 0.451285 | -0.449963 | -0.151612 |
| **Healthy Life Expectancy At Birth** |  |  |  | 1 | 0.381162 | 0.021117 | -0.320460 | 0.235444 | -0.145561 | -0.169230 |
| **Freedom To Make Life Choices** |  |  |  |  | 1 | 0.320661 | -0.501100 | 0.571685 | -0.251399 | 0.421924 |
| **Generosity** |  |  |  |  |  | 1 | -0.284544 | 0.321622 | -0.078126 | 0.274818 |
| **Perceptions Of Corruption** |  |  |  |  |  |  | 1 | -0.304968 | 0.286422 | -0.464043 |
| **Positive Affect** |  |  |  |  |  |  |  | 1 | -0.314201 | 0.120768 |
| **Negative Affect** |  |  |  |  |  |  |  |  | 1 | -0.129993 |
| **Confidence In National Government** |  |  |  |  |  |  |  |  |  | 1 |