Happiness Study

DATA1002

EILEEN WANG 450240984
YIRAN JING 460244129

Contents

Introduction

Pursuing happiness has always been a fundamental part of human nature, with some arguing that it is the reason why we strive for success, maintain relationships and do more than just exist. Philosophically and semantically, one can easily define happiness and describe what constitutes this feeling. However, given that happiness is an intangible feeling that is subjective depending on the individual, happiness is often difficult to compare and measure, especially between people among different cultures. Nevertheless, past studies and literature have attempted to quantify happiness using a combination of subjective measures such as responses from surveys and objective numerical measures like life expectancy and financial measures. Such studies have given rise to the term 'happiness economics', an academic field which grew extensively during the late 20th century. 'Happiness economics' or the 'economics of happiness' is the theoretical and quantitative study of the relationship between happiness, well-being, or life satisfaction with factors derived from sociological and economic indicators.

In our research, we analyse a dataset containing happiness scores for each country with corresponding subjective and objective measures in an attempt to answer the following questions:

1. On average, is there an increasing or decreasing trend in happiness scores overtime for specific countries?
2. Which countries and which specific regions tend to have higher happiness scores?
3. What determinants lead to higher happiness scores for a country?
4. Do nations that report higher/lower levels of happiness demonstrate consistent patterns in factors that influence happiness?
5. What are some other interesting relationships/insights found in our dataset (that is not necessarily related to happiness scores)?

Part 1: Dataset and Results

1.1 Dataset Description:

The dataset used in this study contains 126 observations detailing the happiness scores for 44 unique countries for the three years 2015, 2016, 2017. Additionally, the data comprises of 20 features with the majority of these attributes relating to sociological, economic and environmental factors. The dataset was created by amalgamating attributes from numerous datasets where these attributes were selected based on the fact that they could potentially explain a nation's happiness score. The following points list the attribute names and their corresponding descriptions:

Attribute	Description
Location	The English name of the country of interest.
TIME	The year the observation was collected – either 2015, 2016 or 2017.
Happiness Score	Score related to the happiness of the country. This variable is obtained by asking respondents the Cantril Ladder question where individuals are asked to rate their own current lives from a scale of 0-10 with 10 indicating the best possible life that the respondent can imagine for themselves. The scores were then averaged over all respondents to attain an aggregate score for each nation.
Economy (GDP per Capita)	Average score related to GDP per capita. Individuals rank their quality of life based on how much they earn.
Family	Score related to the quality of family life.
Freedom	Score related to how much an individual can conduct themselves based on their own free will.
Trust (Government corruption)	Score related to government trust.
Generosity	Score related to how much the country is involved in peacekeeping and global aid.
Value_disposable_income	The annual gross adjusted disposable income of the country expressed as USD per capita.
Value_adult_education	Percentage of adult population that has completed tertiary education.
Value_air_pollution	Percentage of population exposed to more than 10 micrograms/m^3 of fine particulate matter (PM2.5) which is an air pollutant that poses the greatest risk to health globally and increases risk of respiratory diseases.
Value_alcohol_consumption	Alcohol consumption measured in litres per capita (only considers individuals aged 15 and over).
Value_overweight_population	Proportion of 15+ year olds who are overweight (BMI is between 25 or 30) or obese (BMI over 30).
Value_avg_annual_hours	Total number of hours worked by all employees over the year divided by the average number of people in employment; incorporates both part-time and full-time workers.

Happiness Rank	Rank of the country based on happiness score. Lower number signifies a happier country.
HDI_value	Measure of a country's average achievement in the key dimensions of human development: health, education and income.
Income_inequality	Gini Coefficients describing the degree of inequality in a nation's wealth distribution.
unemployment	Annual unemployment rate.
HDI Rank	Rank of the country based on HDI value. Lower number signifies higher HDI value.
Region	The region the country belongs to.

***Table 1:** description corresponding to each attribute.*

We note that the attributes Happiness Score, Economy, Family, Health, Freedom, Trust and Generosity all originated from the same dataset called the World Happiness Report. This dataset was released by the United Nations on March 20th 2017 and originally comprised of happiness scores and related economic metrics for over 150 countries. Further, the attributes adult education, air pollution, disposable income, alcohol consumption, average annual hours, and overweight population were collected from several datasets found in the Organisation for Economic Co-operation and Development's online database. Finally, the attributes unemployment, income inequality and HDI were obtained from the United Nation's online database.

1.2 Results:

1. Do happiness scores increase overtime?

Figure 1 below illustrates the happiness scores for each region across the three years 2015, 2016, 2017. Examining the bar plot, it is evident that there are no drastic changes in the happiness scores for each region overtime. However, for the regions North America, Latin America and Carribean, Southeastern Asia, and Southern Asia there appears to be a slight decrease in scores in each consecutive year. In contrast, the plot shows that Central and Eastern Europe experienced a steady increase in happiness in the past three years. Nevertheless, since the change in scores for each region over time is insignificant and small, we focus our analysis on only the year 2015 in the latter parts of this section.

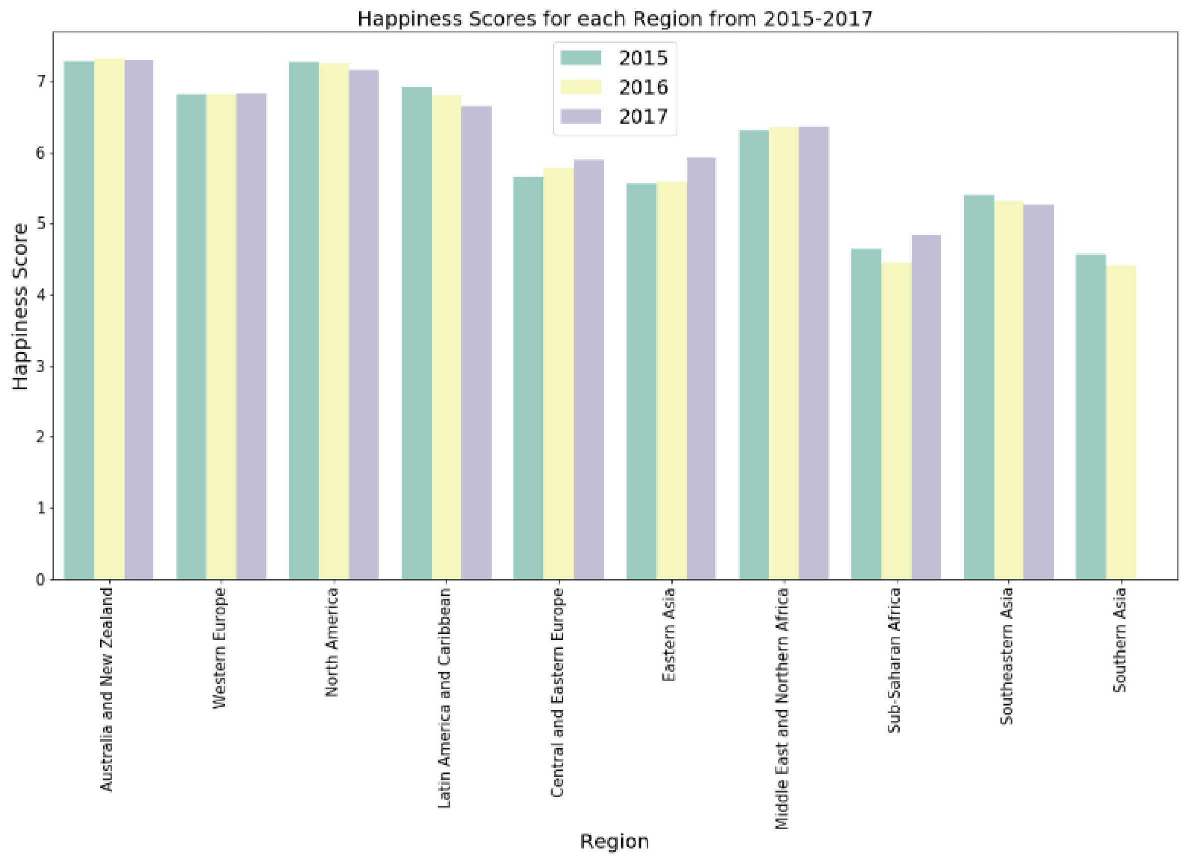


***Figure 1:** bar chart for happiness scores across each region over 2015-2017.*

2. Which countries and which specific regions tend to have higher happiness scores?

With regards to the first question, we found that countries in Australia and New Zealand region, some parts of Western Europe and North America have the highest happiness Scores. Conversely, Sub-Saharan Africa and Southern Asia were recorded to have the lowest levels of happiness in our dataset in 2015. More specifically, we found that Switzerland produced the highest happiness score of 7.59 while India had the lowest score of 4.565.

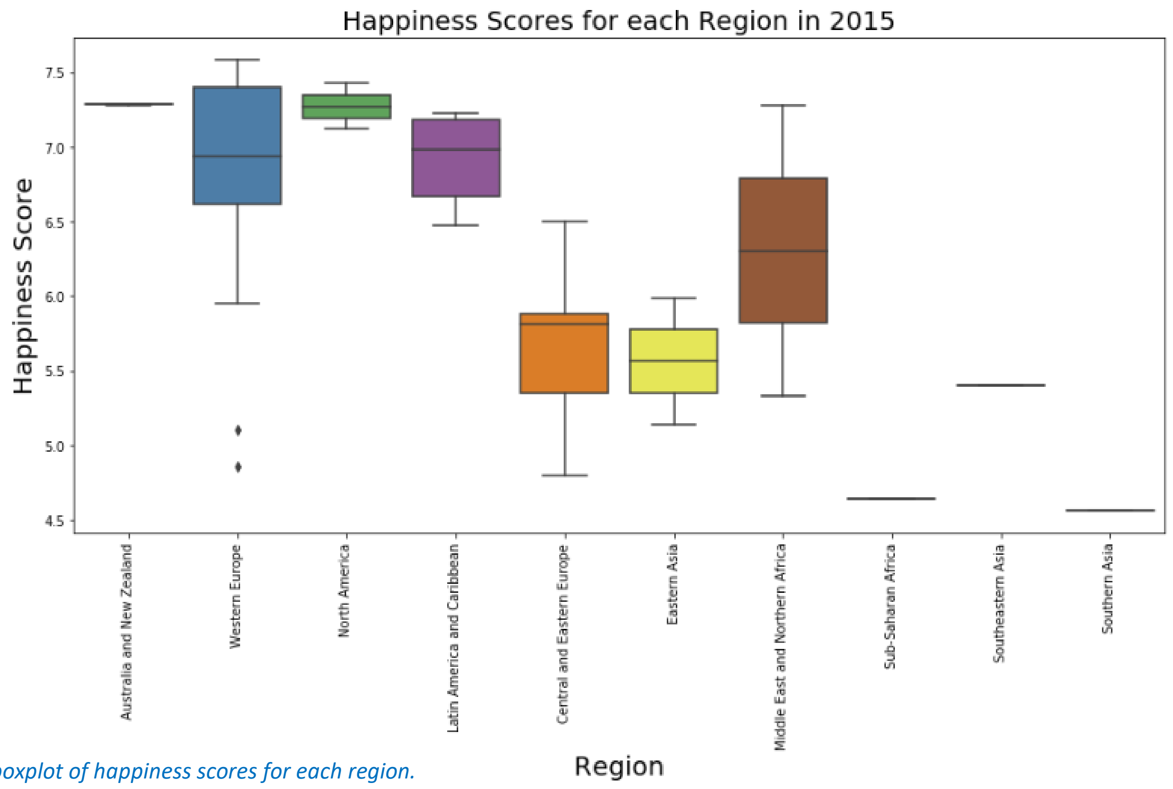


Figure 2: *boxplot of happiness scores for each region.*

Closer inspection of the boxplot reveals that Western Europe comprises of countries which produced the highest happiness levels as well as some significant outliers consisting of countries with very low happiness. Thus, we examined the top 5 and bottom 5 Western Europe countries by happiness score and the results are displayed in *Table 2 below:*

Top 5 Happiest Western Europe Countries	Bottom 5 Happiest Western Europe Countries
1. Switzerland	1. Greece
2. Iceland	2. Portugal
3. Denmark	3. Italy
4. Norway	4. Spain
5. Finland	5. France

Table 2: *top 5 and bottom 5 ranked happiest Western Europe Countries.*

Interestingly, we find that Greece, Portugal, Italy, Spain and France are all geographically close situated in South Western Europe. In contrast, Finland, Norway, Denmark, Iceland and Switzerland are all located in the Northern parts of Western Europe.

3. What determinants lead to higher happiness scores for a country?

Observing *Figure 3*, there appears to be a positive relationship between the factors GDP per capita, family, freedom, generosity, Trust, and HDI with a country's score. All of these factors intuitively make sense. For example, countries with high GDP per capita tend to be wealthier, developed nations where citizens can live more comfortably, thus resulting in higher happiness scores while the converse is true for countries with lower GDP per capita. For the attributes in the bottom row (average annual hours, unemployment, and air pollution), the relationship is not as clear. Closer inspection reveals that there is perhaps a negative relationship between average annual hours and happiness scores and similarly, a negative relationship between unemployment rate and scores - however, the relationship is not as apparent and requires further study.

We also note that colours tend to be loosely clustered together, indicating that countries from the same region share similar values. For instance, in the HDI plot, most of the orange points are grouped near the top right hand corner, depicting the fact that Western Europe countries generally have higher HDI values and higher happiness scores. Similarly, Western Europe countries seem to have lower unemployment rates and lower average annual working hours.

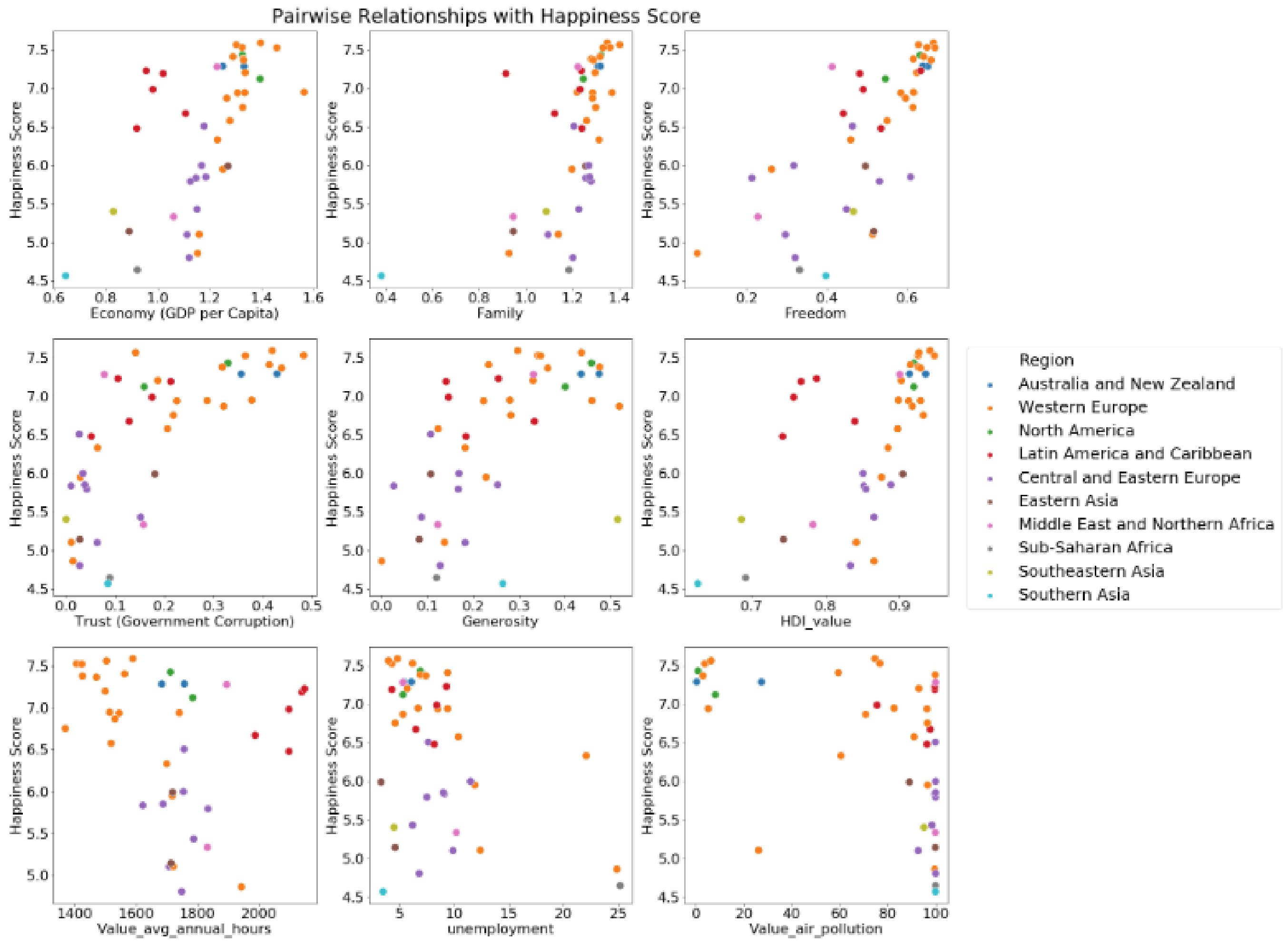


Figure 3: *scatter plots between Happiness Score and other variables.*

4. Do nations that report higher/lower levels of happiness demonstrate consistent patterns in factors that influence happiness?

For this question, we took a closer look at the statistics corresponding to the top 5 and bottom 5 happiest countries in 2015. The heatmaps below can be interpreted as follows: a score closer to 1 indicated by a bluer colour means that the country has a high value in that attribute. Conversely, scores closer to 0 represented by yellow tones denote lower values for the corresponding attribute.

Thus, examining the heatmaps (*Figure 4* and *Figure 5*), we see that the top 5 countries share higher values in attributes such as Freedom, Trust and disposable income as indicated by the blue tones in columns 1-6 and 9 with the exception of Iceland which recorded a Trust score of 0.28. Conversely, *Figure* 5 shows more green and yellow tones in these areas, depicting a low score for these attributes. In terms of unemployment, the top 5 countries appear to have much lower rates. Air pollution in unhappier countries also appears to be high, with Greece, Hungary, South Africa and India reporting extremely large scores. Finally, we observe that there is not much discrepancy in alcohol consumption scores between the two groups, indicating that this variable may not be a good predictor for happiness. Thus, overall, we conclude that very happy and very unhappy countries do demonstrate consistent patterns for some variables.

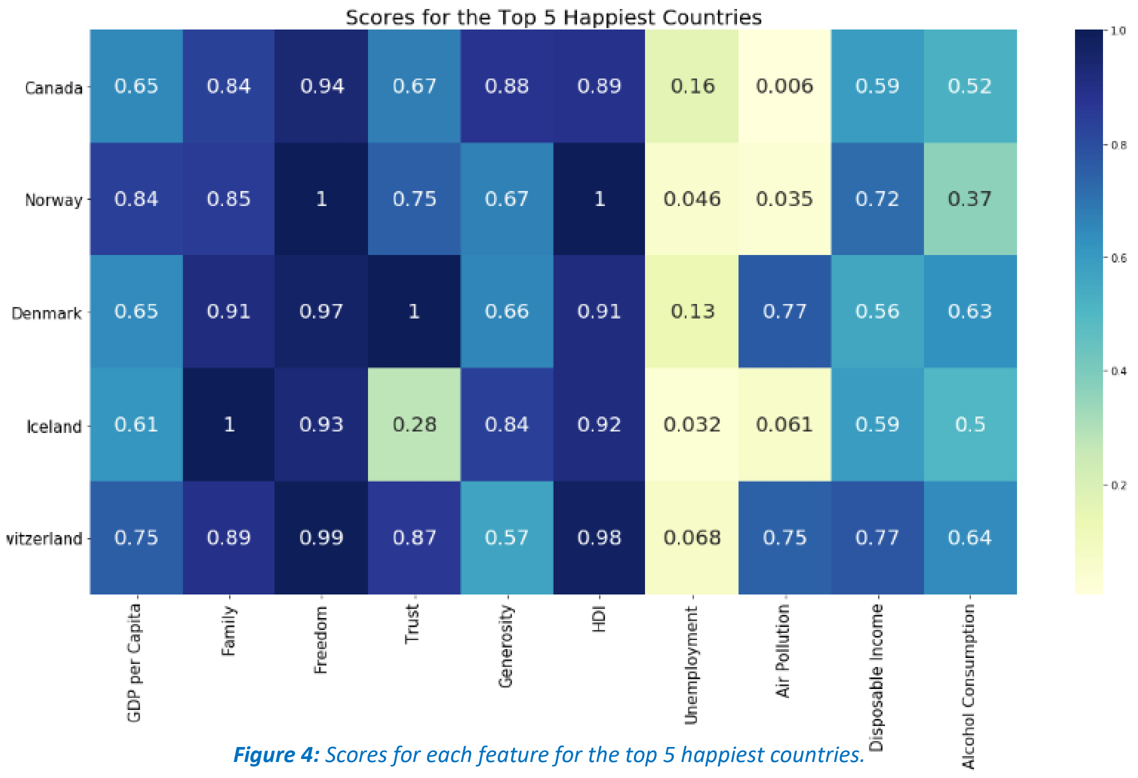


Figure 4: *Scores for each feature for the top 5 happiest countries.*

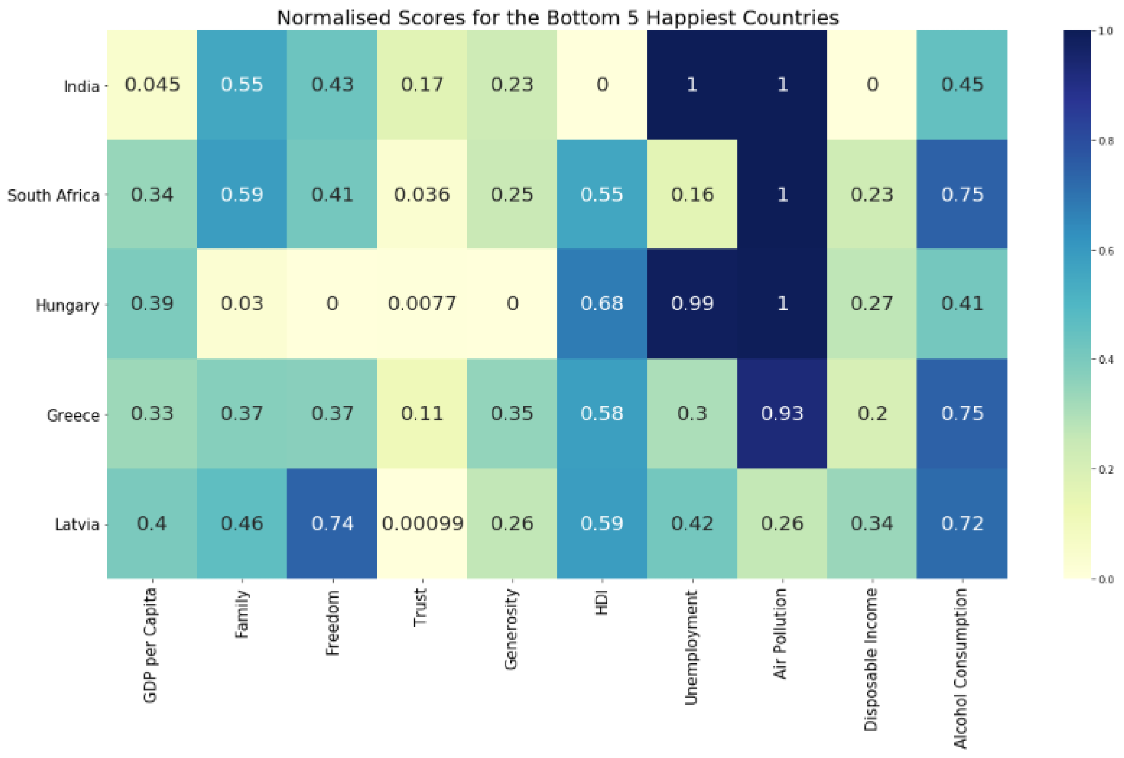


Figure 5: *Scores for each feature for the bottom 5 happiest countries.*

5. What are some other interesting relationships/insight found in our dataset (that are not necessarily related to happiness scores)?

The attributes Economy (GDP Per Capita), Family, Health (Life Expectancy), Freedom, Trust (Government Corruption), and Generosity are social scores which come from the original Happiness dataset. Hence, another area worth exploring is how much of the total score for these 6 factors for each region is explained by each individual factor.

With reference to *Figure* 6, each colour in the plot represents the proportion of the total region score explained by that attribute. For example, GDP per Capita (orange bar) constitutes approximately 5% of Southern Asia's total score, indicating that the score for GDP per Capita in this region is very low relative to the other attributes. Interestingly, we observe that most countries rate quality of trust in the government poorly relative to the other factors (especially for Southern Asia) with perhaps the exception of Australia and New Zealand and Western Europe. In fact, we find that Australia and New Zealand and Western Europe have evenly distributed proportions in the 6 attributes. As they were the top 2 happiest regions, we conclude that happier regions tend to have similar scores for the 6 features. Finally, we observe that Sub-Saharan Africa has no yellow bar for Health (Life Expectancy) rating as the single country found in this region rated 0 for this feature which is unsurprising considering this region experiences extreme poverty.

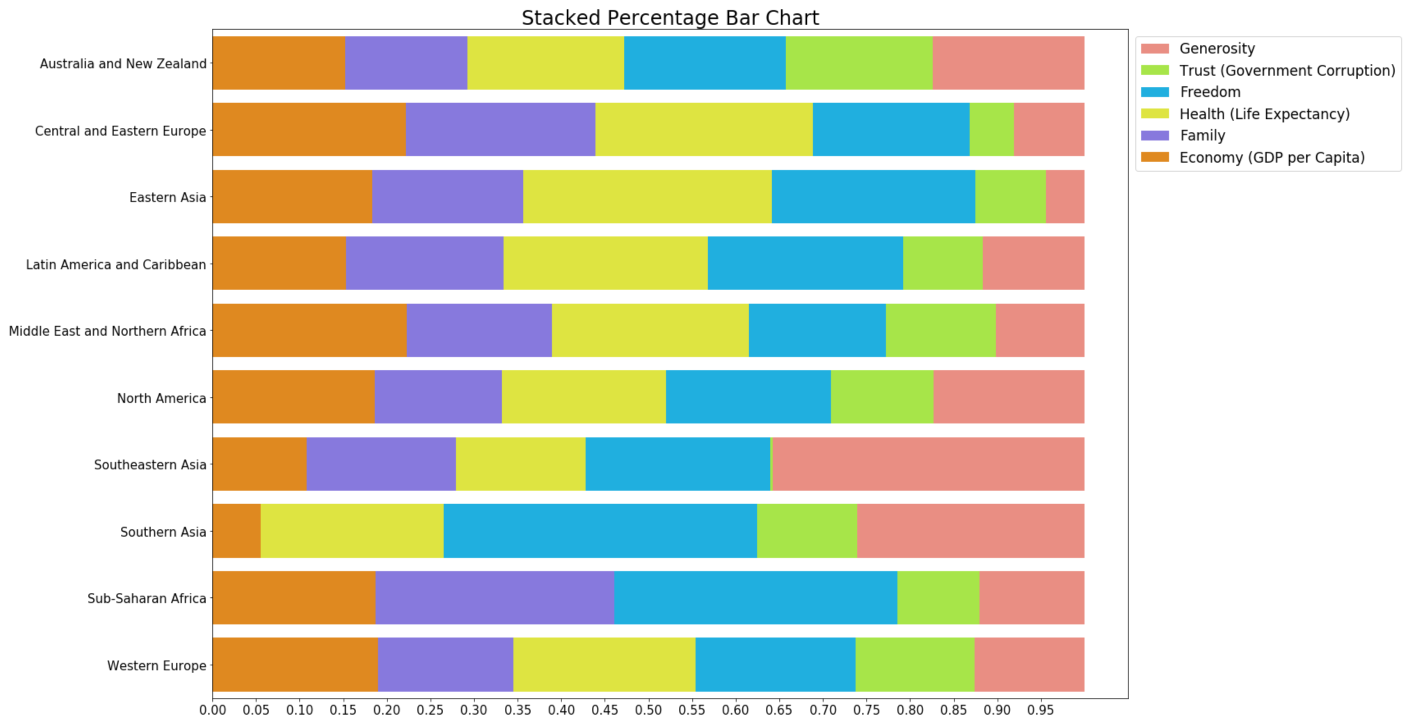


Figure 6: *Proportion of total region score explained by an attribute for each region.*

We note that each colour bar represents a proportion *relative* to that region's total score. Thus, a longer bar in one region compared to another does not mean that that the absolute value of that attribute was higher by that region.

Additionally, scatter plots (*Figure* 7 and *Figure* 8) were constructed to **examine 4-way relationships** between other variables. For both *Figure* 7 and *Figure* 8, a larger marker size represents a higher happiness score for that observation. Examining the left scatter plot, we find that there is a positive trend between a nation's disposable income with the proportion of adults who completed tertiary education. Moreover, countries with higher HDI values (indicated by markers with bluer tones) tend to be more educated and thus have greater levels of disposable income. With regards to the marker sizes, we find that happiness scores are generally higher when education levels and disposable income are high as smaller points are clustered towards the bottom left hand corner of the plot.

Observing the right figure, it is evident that GDP per capita decreases as average annual working hours decreases. This could be attributed to the fact that more working hours does not necessarily mean greater productivity.

Countries such as those in Latin America and Caribbean may have lower education (as shown in *Figure 9*) and social security welfare and thus, people are more confined to low-end jobs which usually require longer hours but do not contribute significantly to GDP. Nevertheless, *Figure 7* and *8* reveal that the Latin America and Caribbean region has quite high happiness scores, indicating that long working hours and low GDP per capita does not explain the higher happiness levels for this region.

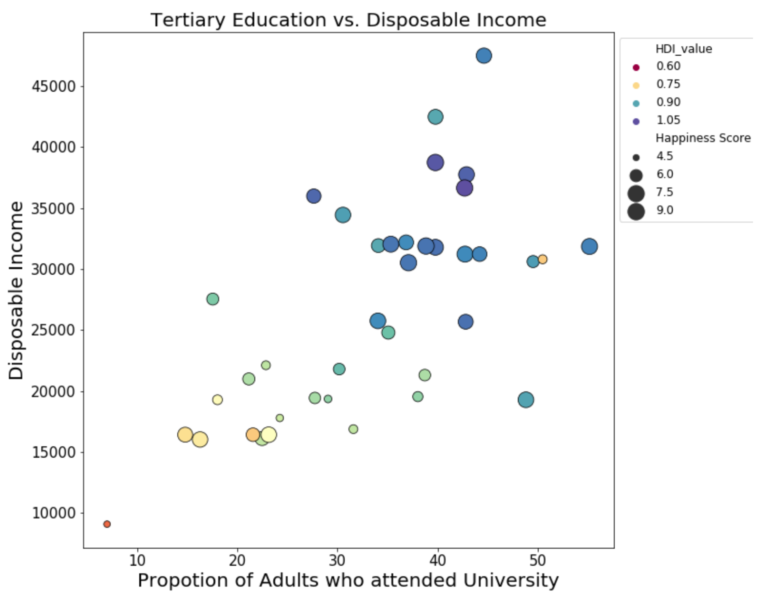


Figure 7: *scatter plot for education vs. disposable income.*

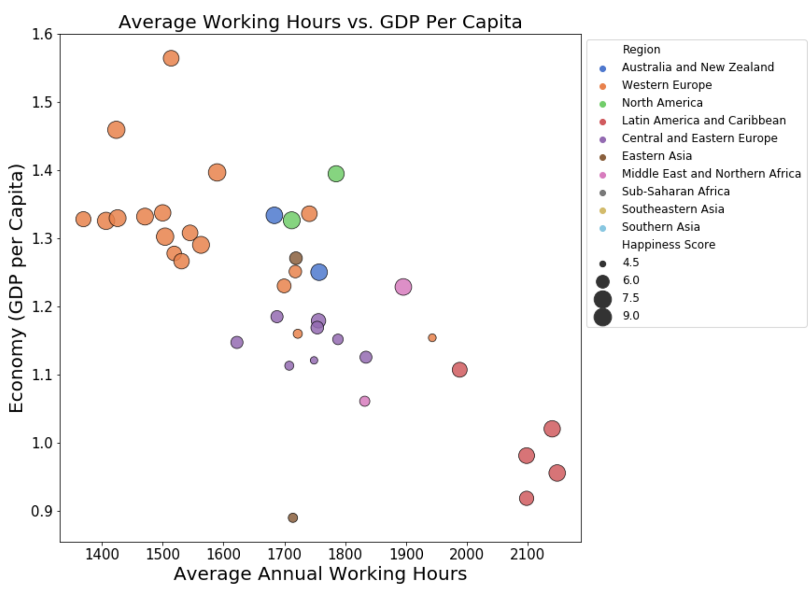


Figure 8: *scatter plot for average annual hours vs. GDP per capita.*

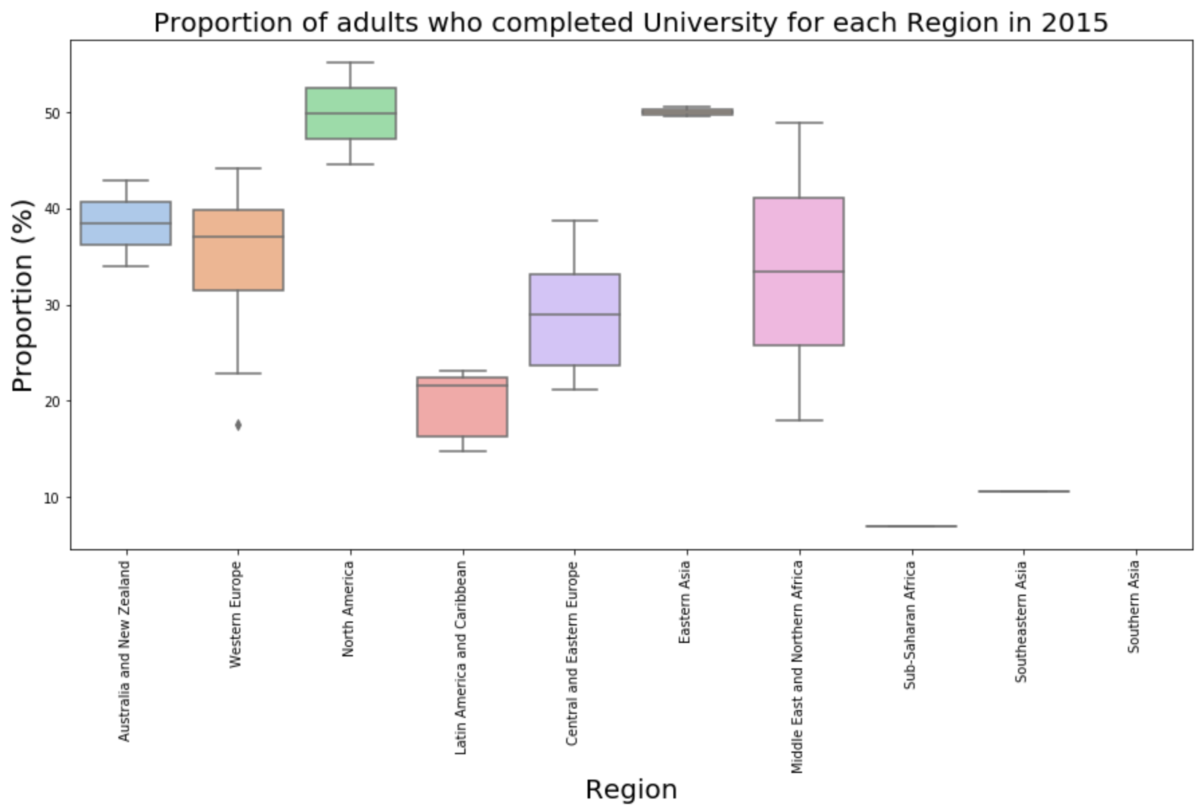


Figure 9: *proportion of adults who completed tertiary education each region*

.

1.3 More questions

2. ***Question 1: What causes the outliers shown in the scatter plot?*** From *Figure 3*, there are some apparent outliers for unemployment and air pollution when plotted against happiness score. Hence, it is a good idea to check whether these values are correctly recorded and if they are, we should explore what might be the underlying cause for these extreme values. For example, one country appears to have relatively high unemployment rate but also high happiness score which intuitively does not make sense.

3. ***Question 2: Does low air pollution make people happier?*** Based *on Figure 3*, although there is no clear relationship between air pollution and happiness scores, we can see the countries with quite low air pollution generally have extremely high happiness score as indicated by the heatmaps (*Figure 4* and *5*). Thus, perhaps there is a relationship between the two features. More data is necessary to explore this issue.

4. ***Question 3: Is there a relationship between working hours and happiness score?*** From *Figure 8*, we can see that the countries with relatively long workers and relatively short working hours have higher happiness score since larger markers are situated near the top left and bottom right of the plot. However, *average annual working hours* actually might have no relationship with happiness score. Instead, the relationship may be attributed to some other cause such as *technology*' or '*productivity level*' since some tech jobs are known to require fewer working hours yet yield more value than other jobs such as factory workers or farmers.

Part 2: Methodology

This section of the report details the methodology and code used to attain our main findings in the previous section. The main tools used were the 'pandas', 'seaborn' and 'sklearn' library in Python. The 'pandas' package is an open-source library, providing flexible data structures and powerful methods designed to easily manipulate and transform data. On the other hand, 'seaborn' is a data visualisation library based on 'matplotlib', offering a high-level interface for producing detailed statistical graphs. Finally, the 'sklearn' module provides numerous implementations of machine learning algorithms as well as efficient tools for data mining and data analysis.

1. Do happiness scores increase overtime?

This question was examined using a bar plot (*Figure* 1). The steps for constructing this plot first involves creating a subplot layout with the desired figure size as done in the first line of the code below. The *barplot* function from seaborn was then called where the argument 'hue' controls how the bars are grouped. In this case, we wanted the bars corresponding to each of the 3 years from the 'TIME' attribute to be grouped together for each region. Thus, we set *hue = "TIME"*. Moreover, 'Happiness Score' should be on the *y*-axis and 'Region' should be on the *x*-axis as indicated by our first two input arguments in *sns.barplot*.

```
fig, ax = plt.subplots(figsize=(20,10)) #create plot layout and define figure size.
plt.rcParams["axes.labelsize"] = 20 #set the font size of axes labels.
#draw the plot!
g = sns.barplot(x = "Region",y = "Happiness Score", hue = "TIME", data = df, palette="Set3")
ax.set_title("Happiness Scores for each Region from 2015-2017", fontsize = 20)
ax.xaxis.set_tick_params(labelsize=10) #set tick label font size to 10.
ax.yaxis.set_tick_params(labelsize=10)
plt.xticks(rotation=90) #rotate the x label axis to be vertical.
plt.show()
```

As we are dealing with categorical data on the horizontal axis in this case, a barplot was thought to be the most appropriate. We note that we initially tried representing the information as a boxplot but the result appeared messy, given that we had too many categories to consider. Furthermore, a boxplot was unnecessary as we just wanted to determine whether there was an increasing/decreasing trend in happiness scores over the 3 years for each region and not what the distribution looked like for each year. Also, with a barplot, the trend is easy to identify as the viewer is just required to compare whether the bars for each region are decreasing or increasing. However, the limitation of our bar plot is that it may be difficult to determine exact numerical differences between bars.

Finally, *Table* 2 was produced by filtering and sorting the dataframe as follows where the *head* function is used to obtain the first 5 observations. Similarly, *tail* would be used to obtain the last 5 rows.

```
df[(df["TIME"] == 2015) & (df["Region"] == "Western Europe")].sort_values(by = "Happiness Score").head(5)
```

2. Which countries and which specific regions tend to have higher happiness scores?

The boxplot created to answer this question (*Figure* 2) was achieved using the *boxplot* function from seaborn. Additionally, we ensured that the data inputted in the third argument is filtered to only contain observations from 2015.

```
fig, ax = plt.subplots(figsize=(15,7)) #create plot layout and define figure size.
sns.boxplot(x="Region", y = "Happiness Score", data = df[df["TIME"] == 2015], palette = "Set1")
ax.set_title("Happiness Scores for each Region in 2015", fontsize = 20) #set title of plot.
plt.xticks(rotation=90) #rotate the x labels to be vertical.
```

A boxplot was chosen to represent the happiness scores for each region as it provides a five-number summary for each region. Additionally, it is useful for identifying how the distributions differ between each category. For instance, the viewer just has to compare the size of the boxes to determine whether one region has a larger spread compared to another. Statistics such as median, maximum or minimum are also easy to compare across the categories and outliers can be identified at a glance. The limitation of a boxplot however is that we cannot identify where each observation (country) lies in the distribution.

3. What determinants lead to higher happiness scores for a country?

For this question, scatter plots (*Figure* 3) were used to inspect the pairwise relationships between specific variables with Happiness Score where the colours of each point represent the region for which that observation originates from. Firstly, we stored the features we wanted to examine in an array called 'columns'. The dataframe was then filtered to contain only the interested features and the observations from 2015. Next, a 3 × 3 figure layout which stores 9

```
#interested columns
columns = ['Happiness Score', 'Region', 'Economy (GDP per Capita)', 'Family','Freedom',
           'Trust (Government Corruption)', 'Generosity', "HDI_value",
            'Value_avg_annual_hours', 'unemployment','Value_air_pollution']
```

```
df_pair = df[df['TIME'] == 2015][columns] #filter dataframe to contain only those columns
```

```
fig, ax = plt.subplots(3,3, figsize = (25,25))

index = 2
for i in range(0,3): #for each row
    for j in range(0,3): #for each column

        if i == 2 and j == 2:
            #if we're constructing the last scatterplot, insert a legend.
            sns.scatterplot(columns[index],"Happiness Score",hue='Region', data=df_pair, ax = ax[i,j], s = 150)
            ax[i,j].legend(bbox_to_anchor=(2.1,2.2), fontsize = 20)
        else:
            sns.scatterplot(columns[index],"Happiness Score",hue='Region', data=df_pair, ax = ax[i,j], legend = False, s = 150)

        #format the label sizes
        ax[i,j].xaxis.set_tick_params(labelsize=15)
        ax[i,j].yaxis.set_tick_params(labelsize=15)
        ax[i,j].xaxis.get_label().set_fontsize(15)
        ax[i,j].yaxis.get_label().set_fontsize(15)
        index += 1

fig.suptitle('Pairwise Relationships with Happiness Score', fontsize = 25, x = 0.5, y = 0.9) #add title.
```

subplots was constructed and we then iterated through each axes with each iteration calling *sns.scatterplot*. The 'index' variable keeps track of the features in the 'columns' array, allowing us to plot a different scatterplot for each feature in each iteration. Moreover, the 'hue' parameter in *sns.scatterplot* is used to colour code the markers according to region category.

A scatterplot was chosen as such plots are extremely useful for analysing pairwise relationships and linear trends. Identifying whether there is a trend between a certain feature and the response variable (happiness score) is important as later in Part 3, we will be required to choose some appropriate features for our machine learning models. Moreover, we decided to colour code the markers by region so that we could examine whether regions are clustered together, which may indicate that countries from the same region share similar values. However, with a scatter plot, it is not possible to identify the exact quantitative measure of the relationship between the variables. Instead, analysing such plots rely purely on visual inspection which may lead to inaccurate conclusions.

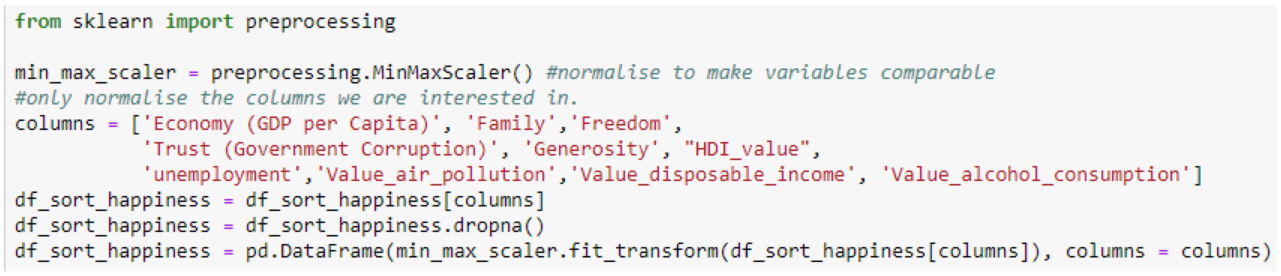
4. Do nations that report higher/lower levels of happiness demonstrate consistent patterns in factors that influence happiness?

To obtain the heatmaps (*Figure* 4 and *Figure* 5), the dataframe was first sorted in ascending order according to happiness scores into a dataframe called 'df_sort_happiness'. At first, the data values observed from the dataset was used to create the heatmap. However, we realised that this was not a fair comparison as the data for some of the features were measured using different units. Thus, the attributes were first normalised using the '*MinMaxScaler*' function from sklearn's '*preprocessing*' module to scale the values between 0 and 1.

Normalisation for an observation 'X' where 'X' belongs to a particular attribute is done using the following formula:

$$\boldsymbol{X_{normalised}} = \frac{\boldsymbol{X - X_{minimum}}}{\boldsymbol{X_{maximum} - X_{minimum}}}$$

In the above equation, $X_{normalised}$ is the normalised value, $X_{minimuim}$ is the observed minimum value of the attribute and similarly, $X_{maximum}$ is the observed maximum value of the attribute in the dataset. Normalisation was implemented so that we could eliminate the units of measurement for each attribute which in turn, enables for easier comparison across different attributes. The code for this process is depicted below:



```
from sklearn import preprocessing

min_max_scaler = preprocessing.MinMaxScaler() #normalise to make variables comparable
#only normalise the columns we are interested in.
columns = ['Economy (GDP per Capita)', 'Family','Freedom',
           'Trust (Government Corruption)', 'Generosity', "HDI_value",
           'unemployment','Value_air_pollution','Value_disposable_income', 'Value_alcohol_consumption']
df_sort_happiness = df_sort_happiness[columns]
df_sort_happiness = df_sort_happiness.dropna()
df_sort_happiness = pd.DataFrame(min_max_scaler.fit_transform(df_sort_happiness[columns]), columns = columns)
```

To create a heatmap, we simply take the first 5 observations in 'df_sort_happiness' (or the last 5 if we wish to examine the top 5 happiest countries) and call *sns.heatmap* with a dataframe containing these 5 rows as shown in the code:

```
df_top5 = df_sort_happiness.tail(5) #least 5 happiest countries.

Index = top5_countries
Cols = list(df_top5.columns)

fig, ax = plt.subplots(figsize=(20,10))

#create new dataframe storing the 5 countries.
df_top5_heat = pd.DataFrame(df_top5.values,index = Index, columns = Cols)
#construct a heatmap.
sns.heatmap(df_top5_heat, annot = True, cmap="YlGnBu", annot_kws={"size": 20})
ax.set_title("Normalised Scores for the Top 5 Happiest Countries", fontsize = 15)
plt.yticks(rotation=0)
```

Heatmaps were useful for our purpose because they utilise colour to communicate complex statistical data and thus, are capable of displaying a more generalised view of numeric values. Moreover, colours are usually easier to distinguish values at a glance rather than just displaying raw numbers. This is especially beneficial in our case as we wanted to examine many attributes across different countries.

5. What are some other interesting relationships/insight found in our dataset (that are not necessarily related to happiness scores)?

The stacked barplot in *Figure* 6 was created as follows:

1. Firstly, the 6 attributes/scores (Economy (GDP Per Capita), Family, Health (Life Expectancy), Freedom, Trust (Government Corruption), and Generosity) were normalised so that they could be directly comparable.
2. Next, we calculate the total score of the region by summing up the 6 scores for each country belonging to that region.
3. The region's total score for each attribute was then calculated. This creates 6 tables (each table corresponding to the 6 attributes), whereby each table consists of the 9 regions and their corresponding summed score for that attribute.
4. Finally, the region's total score for each attribute (obtained in step 3) was divided by its corresponding total region score (obtained in step 1). This results in the proportion of the total region score explained by that attribute.
5. The data structure used to store these proportions was a dictionary called 'score_proportions'. The key in this dictionary is the attribute name (either Economy (GDP Per Capita), Family, Health (Life Expectancy), Freedom, Trust (Government Corruption), or Generosity) and the values are lists of length 9 where each value (representing a proportion) in the list corresponds to one of the 9 regions.

```
f,ax = plt.subplots(figsize = (20,15))

region_list =list(df_norm["Region"].unique()) #contains of list of region names.
columns = ['Economy (GDP per Capita)', 'Family', 'Health (Life Expectancy)','Freedom',
           'Trust (Government Corruption)', 'Generosity']
columns_reverse = columns[::-1]
colours = ['#FA8072', '#ADFF2F', 'deepskyblue', '#F7FF26', 'mediumslateblue', 'darkorange']

for i in range(0, 6):
    col = columns_reverse[i]
    sns.barplot(x=score_proportions[col], y = region_list, label = col, color = colours[i])

plt.xticks(np.arange(0, 1, 0.05)) #increase frequency of x ticks.
ax.xaxis.set_tick_params(labelsize=15)
ax.yaxis.set_tick_params(labelsize=15)
ax.legend(bbox_to_anchor=(1, 1), fontsize = 17) #add legend outside of plot.
ax.set_title("Stacked Percentage Bar Chart", fontsize = 24)
plt.show()
```

A stacked bar chart was selected as such charts are powerful for showing how a larger category can be partitioned into smaller subcategories. They are also capable of revealing what relationship the smaller subcategories have on the total amount. Furthermore, viewers can generally identify how the proportions differ for each region at a glance by comparing the lengths of bars which are of the same colour. We note that stacked bar plots are difficult for comparison when the number of bars in each stack increases. However, this is not particularly an issue in our case since there are only 6 attributes. For this task, we initially tried a pie chart to show the proportions for each region, but this resulted in 9 different charts. Additionally, with a pie chart, each slice of the same colour corresponding to the same attribute would be on separate plots and not stacked together like the stacked bar chart, making comparison across regions difficult. We thus conclude that a stacked bar chart is much better in condensing information and displaying proportions compared to a pie chart.

The scatter plots in *Figure* 7 and *Figure* 8 were built a similar way to *Figure* 9 using *sns.scatterplot*. However, we inputted additional features such as setting the parameter '*size'* equal to "Happiness Score". This effectively allows the scatterplot markers to vary depending on the size of the happiness score for that observation. Code snippet used to produce *Figure* 7 is shown below.

```
#added size to dots to represent happiness
fig, ax = plt.subplots(figsize = (10,10))
sns.scatterplot("Value_avg_annual_hours","Economy (GDP per Capita)",hue='Region', alpha=.85,edgecolor='black',
                size = "Happiness Score", sizes=(40, 350), data=df[df['TIME'] == 2015], palette="muted", ax = ax)
ax.xaxis.set_tick_params(labelsize=15)
ax.yaxis.set_tick_params(labelsize=15)
ax.set_xlabel("Average Annual Working Hours")
ax.set_title("Average Working Hours vs. GDP Per Capita", fontsize = 20)
ax.legend(bbox_to_anchor=(1, 1), fontsize = 12)
```

By changing the size of the points according to happiness score, we can express even more relationships between different variables, allowing us to further identify underlying patterns. For example, *Figure* 7 is capable of capturing a 4-way relationship between average annual working hours, GDP per capita, region and happiness score.

Finally, the boxplot in *Figure* 9 was created using the same approach as *Figure* 2*:*

```
fig, ax = plt.subplots(figsize=(15,7))
sns.boxplot(x="Region", y = "Value_adult_education", data = df[df["TIME"] == 2015], palette = "pastel")
ax.set_title("Proportion of adults who completed University for each Region in 2015", fontsize = 20)
ax.set_ylabel("Proportion (%)")
plt.xticks(rotation=90)
```

Part 3: Predictive Models

3.1 Data Preparation

3.1.1 Splitting the Data into a Training and Testing Set

To train our models, we used the data corresponding to the 2015 and 2016 observations as the training set and used 2017 dataset for testing. This resulted in approximately **67.5%** data in the train set and **33.5%** for testing.

```
train = df[df['TIME']!=2017]
test = df[df['TIME']==2017]
```

We note that we did not use the 'random' split method, since we wanted to divide the dataset according to the year of the observation. Furthermore, as we aim to build a model capable of predicting the future, it makes sense to use the 2017 observations for testing. However, we note that the limitations of this approach is that the data in 2017 may show slightly different patterns than the data in 2015-2016 used for training the model. The training data for the predictors is stored in ‘X_train’ and the corresponding continuous target variable is in ‘y_train’. Similarly, the predictors and response variables for the test set are stored in ‘y_train’ and ‘y_test’ respectively.

```
X_train = train[features_group]
X_test = test[features_group]  # 9 features

y_train = train['Happiness Score']
y_test = test['Happiness Score']

y_test = y_test.as_matrix()
```

3.1.2 Standardising Predictors

```
def standardization(features_group, X_train, X_test):
    for feature in features_group:
         ### get mean and std from train dataset
        (mu_train, sigma_train) = norm.fit(X_train[feature])
        ### normalization this feature
        X_train[feature] = (X_train[feature] - mu_train) / sigma_train
        X_test[feature] = (X_test[feature] - mu_train) / sigma_train
    return X_train, X_test

X_train, X_test = standardization(possible_features, X_train, X_test)
```

Before inputting the variables into our models, we first standardised each feature in both the training and testing set by subtracting the feature’s mean from each observation and dividing it by the feature’s standard deviation. This formula transforms the features such that the mean of all observations is scaled to 0 and its standard deviation is scaled to 1. Standardisation was done for two primary reasons. Firstly, it allows for easier and more meaningful interpretation of the intercept in the regression model as all predictors are now scaled to have a mean of 0. Additionally, standardisation is extremely important for the KNN model, as the algorithm for KNN involves calculating the Euclidean Distance between observations to determine the nearest neighbour. Thus, it is necessary that all features are converted to the same scale such that distances are computed correctly.

Furthermore, for convenience, features were renamed accordingly:

Attribute’s old name	Attribute’s new name
Economy (GDP per Capita)	Econ
Value_disposable_income	DisIncome
Value_avg_annual_hours	AvgHours
Family	Family
Health (Life Expectancy)	Health
Trust (Government Corruption)	Trust
Generosity	Generosity
HDI_value	HDI
Freedom	Freedom

Table 3: *Renamed features used in models.*

3.1.3 Variable selection

Based on our domain knowledge, we selected 14 features that could possibly explain a country’s happiness score. There 14 features considered were:

```
## Draw plots to check if linear relationship exists between
possible_features = [
         'Value_disposable_income','Value_adult_education',
          'HDI_value','Value_avg_annual_hours',
         'Value_alcohol_consumption', 'Value_air_pollution',
         'income_inequality',
        'unemployment',
        'Economy (GDP per Capita)', 'Family', 'Health (Life Expectancy)',
        'Freedom', 'Trust (Government Corruption)', 'Generosity']
```

To select the 'best' subset of predictors, we use the **sequential forward selection** algorithm for variable selection based on the 14 initial variables above. This approach starts from an empty model with no features and iteratively adds features to the model until no new additional feature improves the performance of the model, whereby the performance of the model is determined by highest negative MSE (i.e. lowest mean squared error). There are 4 predominant reasons for why feature selection is an important procedure:

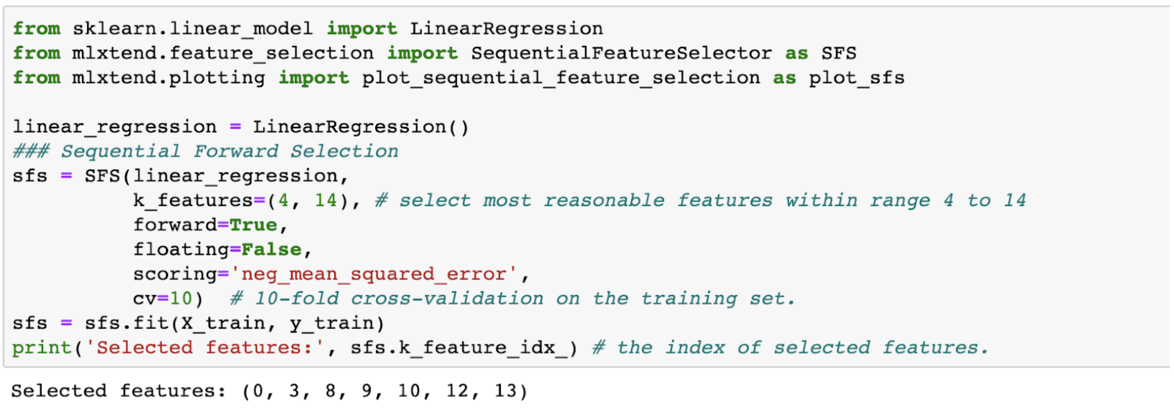
1. Firstly, we want to explain the data in the simplest way, thus having too many variables in the model is generally not a good idea.
2. Secondly, redundant predictors can deteriorate model performance as unnecessary predictors tend to model the noise in the data, resulting in overfitting such that the error in the test set is large but the error on the train set is low.
3. Further, correlated variables should not be included in the model as this creates multicollinearity. Multicollinearity is a result of intercorrelations between predictors which results in coefficients of the model to be estimated imprecisely.
4. Finally, reducing the number of variables improves computational efficiency when running the model for prediction.

The Sequential Forward Selection process was done using *10-fold cross-validation* on the training set. That is, the training data is split into 10 partitions and in each of the 10 iterations, 1 partition is used to test what the optimal subset of features are.

3.2 Model Construction

Model 1: Linear regression

Using the *LinearRegression* module from the *Sklearn.linear_model* and the *SequentialFeatureSelector* from the *mlxtend.feature_selection* module, we performed feature selection for the linear regression model using the following code:



```
from sklearn.linear_model import LinearRegression
from mlxtend.feature_selection import SequentialFeatureSelector as SFS
from mlxtend.plotting import plot_sequential_feature_selection as plot_sfs

linear_regression = LinearRegression()
### Sequential Forward Selection
sfs = SFS(linear_regression,
          k_features=(4, 14), # select most reasonable features within range 4 to 14
          forward=True,
          floating=False,
          scoring='neg_mean_squared_error',
          cv=10)  # 10-fold cross-validation on the training set.
sfs = sfs.fit(X_train, y_train)
print('Selected features:', sfs.k_feature_idx_) # the index of selected features.
```

```
Selected features: (0, 3, 8, 9, 10, 12, 13)
```

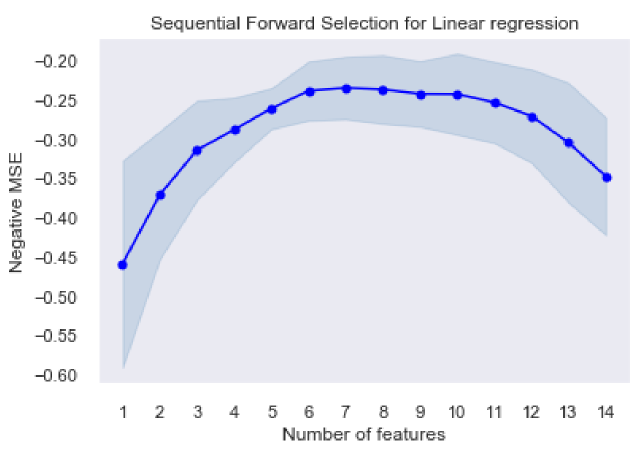


Figure 10: *Negative MSE using different number of features in the linear regression model.*

Figure 10 above shows the negative MSE values corresponding to the number of features selected. Evidently, the model performs poorly when the number of features is too low or too high. Closer inspection reveals that a feature subset of size 7 is the optimal choice.

Furthermore, the 7 selected variables are displayed in the code below:

```
selected_features =['Value_disposable_income','Value_avg_annual_hours', 'Economy (GDP per Capita)', 'Family',
                    'Health (Life Expectancy)','Trust (Government Corruption)', 'Generosity']
## Create new train dataset based on new features
X_train_linear = X_train[selected_features]
X_test_linear = X_test[selected_features]
```

Using only the newly selected features, we then created a new training and test set and then fitted the linear regression model with the optimal feature. The code for this process is depicted below:

```
from sklearn.linear_model import LinearRegression
linear_regression = LinearRegression()
linear_regression.fit(X_train_linear, y_train)

print('Intercept: \n', linear_regression.intercept_)
print('Coefficients: \n', linear_regression.coef_)
r_sq = linear_regression.score(X_train_linear, y_train)
print('coefficient of determination: \n', r_sq)
```

```
Intercept:
 6.429870588235295
Coefficients:
 [0.1022099  0.34518179 0.22518494 0.1951286  0.14364598 0.36897174
 0.24753753]
coefficient of determination:
 0.7789240720443047
```

This produced the following multiple linear regression model:

$$\begin{aligned}\boldsymbol{Happiness\ score} &= \boldsymbol{6.43 + 0.10 * DisIncome + 0.345 * AvgHours + 0.225 * Econ} \\ &\quad \boldsymbol{+ 0.195 * Family + 0.144]times\ Health + 0.369 \times Trust + 0.248 \times Generosity}\end{aligned}$$

The name of our linear regression model was called *linear_regression* in our code. To predict new examples from our test set, we simply call *linear_regression.predict().*

Model 2: Regression based on k-nearest neighbours (KNN)

The kNN model employs an instance-based learning algorithm and is considered to be more flexible than the Linear Regression model, as it is capable of capturing underlying nonlinear patterns inherent in the data. In essence, kNN's algorithm involves predicting the happiness score of the current example by averaging the happiness scores of its 'k' closest neighbours in the dataset. A 'close' neighbour is defined as a point in the dataset that exhibits similar feature values in comparison to the feature values for the current example we are predicting for, whereby similarity is measured using Euclidean Distance. A lower Euclidean Distance implies greater similarity while a higher distance means that that datapoint is quite different.

Turning the hyper-parameter 'k':

The kNN model has one tuning parameter, 'k' which represents the number of neighbours to average over for prediction. The optimal 'k' was chosen using *10-fold cross validation* by utilising the *cross_val_score* method from the *sklearn.model_selection* module and the *KNeighborsRegressor* from the *sklearn.neighbors module.* More specifically, we train KNN using 1-20 neighbours, calculate the average cross-validation score (negative MSE) for each possible value of 'k'. The optimal 'k' is then defined as the one which produces the highest negative MSE.

```python
from sklearn import neighbors
from sklearn.model_selection import cross_val_score
rmse_val = [0] #to store rmse values for different k
for K in range(20):
    K = K+1
    model = neighbors.KNeighborsRegressor(n_neighbors = K)
    scores = cross_val_score(model, X_train, y_train, cv=10, scoring='neg_mean_squared_error')
    rmse_val.append(scores.mean())  # Save the mean of 10 folders.

#plotting the rmse values against k values
curve = pd.DataFrame(rmse_val) #elbow curve
#curve.plot()
plt.plot(curve)
plt.ylabel('Negative MSE')
plt.xlabel('Number of neighbors')
plt.title("The Negative MSE of different neighbors")
```

To visualise the performance of each ‘k’ value, the negative MSE was plotted against different values of ‘k’.

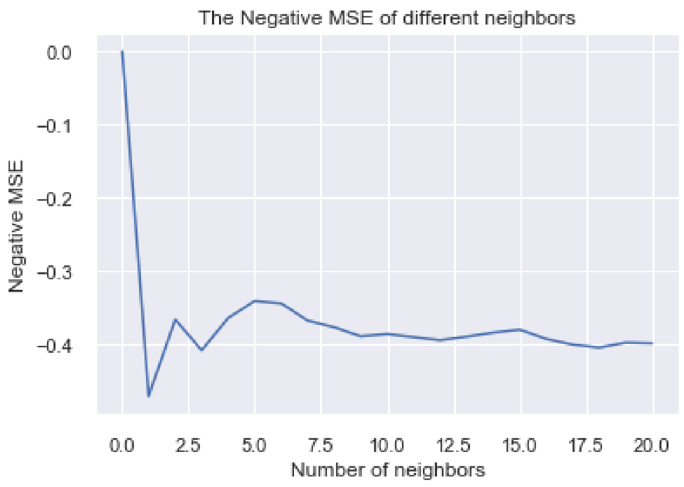


Figure 11: *Negative MSE in terms of different number of neighbours*

Based on the above plot, it is evident that the optimal choice of k is 5.

Next, we perform variable selection with the ‘k’ parameter set to 5 using a similar process to the linear regression model. This resulted in 4 selected variables. A new train and test set was then created using the selected features and we then fitted our kNN model with k =5. We also fitted a KNN model with k=10 using the same selected features to compare its performance with the optimal ‘k’ found from cross validation.

```python
selected_features = ['HDI_value', 'Value_avg_annual_hours', 'Trust (Government Corruption)', 'Generosity']
## Create new train dataset based on new features
X_train_KNN = X_train[selected_features]
X_test_KNN = X_test[selected_features]
```

```python
KNN = neighbors.KNeighborsRegressor(n_neighbors = 5)  # select k = 5
KNN.fit(X_train_KNN, y_train)
```

The name for this model was simply called KNN in our code.

Model 2: Regression Trees

Regression trees are another flexible model, capable of modelling non-linear patterns in the data and additionally, can deal with both categorical and continuous data. Tree-based models involve performing recursive binary splits according to certain cut-off values in the features to optimise a specific loss function such as the MSE. Through splitting, different subsets of the data are produced and the final subsets are located in the leaf nodes of the tree. Prediction for a continuous target variable for an example is done by starting at the root node and travelling down the tree according to the feature cut-off rules until a leaf node is reached. The new prediction is then just the average outcome of the training data in this leaf node.

Turning the hyper-parameter 'max_depth':

The tuning parameter for a regression trees is the maximum depth of the tree. We note that a tree that is too deep results in very small subsets in the leaf nodes, resulting in overfitting. Thus, it is good practice to constrain the maximum depth of the tree. The optimal depth is found by using 10-fold cross validation on the trainomg data with the *cross_val_score* method from the *sklearn.model_selection* module and the *DecisionTreeRegressor* from the *sklearn.tree* module. More specifically, with 10-fold cross validation, we trained the regression tree using different possible values for the maximum depth (considering only values from 1 to 20) and the optimal maximum depth was chosen based on the average highest negative MSE over the 10 folds. Code for this procedure is included below:

```
#### find optimal depth:
rmse_val = [0] #to store rmse values for different k
for depth in range(1, 20):
    decision_tree = DecisionTreeRegressor(random_state = 0,max_depth= depth)
    scores = cross_val_score(decision_tree, X_train, y_train, cv=10, scoring='neg_mean_squared_error')
    rmse_val.append(scores.mean())
```

To visualise the results, we plot the negative RMSE at different tree depths:

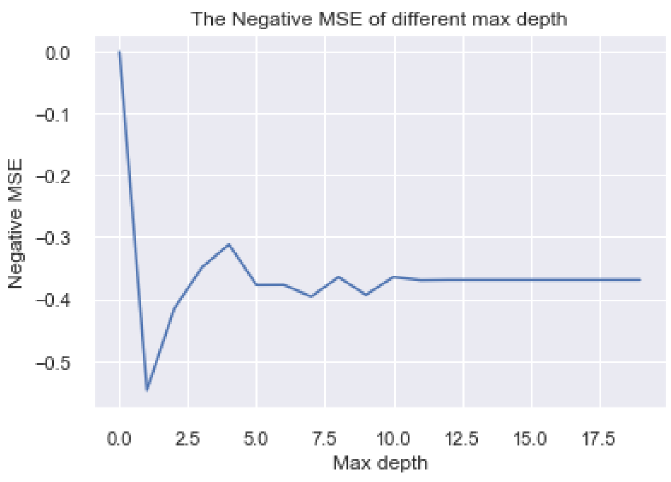


Figure 12: *Negative MSE in terms of different max_depth*

Examining the plot, the optimal choice of the maximum depth was found to be 4.

Next, we perform variable selection with the maximum tree depth set to 4 using a similar process to the previous models. The result was the 6 selected variables below. We then created a new train and test set based on the selected variables and fitted the optimal regression model.

```
selected_features = ['Value_disposable_income', 'HDI_value', 'Value_avg_annual_hours',
                     'Health (Life Expectancy)','Freedom']
## Create new train dataset based on new features
X_train_tree = X_train[selected_features]
X_test_tree = X_test[selected_features]

decision_tree = DecisionTreeRegressor(random_state = 0,max_depth=4)
decision_tree.fit(X_train_tree, y_train)
```

The name of our regression decision tree model was named 'decision_tree' in our code. For prediction, we simply call this model (e.g. using decision_tree.predict()).

3.3 Model Evaluation

Prediction Results Analysis:

To evaluate the performance of models, we made predictions on the test set and measured the accuracy of the predictions. Once the model is constructed in Python, prediction is a simple process and can be done by calling the model name with the *'predict'* function. The single argument in the predict function is the test set data without the target variable which we named 'X_test' as described in Section 3.1.1. The following code snippets outline this procedure:

Prediction using multiple linear regression:

```
print("\n---------------Evaluation-----------------------------------")
y_pred = linear_regression.predict(X_test_linear) # predict
regression_results(y_test, y_pred)
```

Prediction using KNN with k=5:

```
print("\n---------------Evaluation-----------------------------------")
y_pred = KNN.predict(X_test_KNN) # predict
regression_results(y_test, y_pred)
```

Prediction using regreesion tree with max depth = 4:

```
print("\n---------------Evaluation-----------------------------------")
y_pred = decision_tree.predict(X_test_tree) # predict
regression_results(y_test, y_pred)
```

A self-defined helper function 'regression_results' was also created to compute accuracy metrics such as the R^2, MAE (mean absolute error) and $RMSE$ (root mean square error). All of these metrics are based on measuring the error between the predicted data value with the ground truth. A model performs well if R^2 is high and when MAE and RMSE are low. As shown in the code below, computations of these metrics were achieved using the *sklearn.metrics* modules which is a module that assists in calculating performance metrics, score functions and distance computations, typically used for evaluating model performance.

```
from sklearn.metrics import mean_squared_error
import sklearn.metrics as metrics
"""
Method to evaluate model performance based on true data and predicted data in testset.
"""
def regression_results(y_true, y_pred):
    # Regression metrics
    explained_variance=metrics.explained_variance_score(y_true, y_pred)
    mean_absolute_error=metrics.mean_absolute_error(y_true, y_pred)
    mse=metrics.mean_squared_error(y_true, y_pred)
    mean_squared_log_error=metrics.mean_squared_log_error(y_true, y_pred)
    median_absolute_error=metrics.median_absolute_error(y_true, y_pred)
    r2=metrics.r2_score(y_true, y_pred)

    print('explained_variance: ', round(explained_variance,4))
    print('mean_squared_log_error: ', round(mean_squared_log_error,4))
    print('r2: ', round(r2,4))
    print('MAE: ', round(mean_absolute_error,4))
    print('MSE: ', round(mse,4))
    print('RMSE: ', round(np.sqrt(mse),4))
```

The following table summarises the accuracy metrics for each model with an additional model called kNN (k=10). This is a k-nearest neighbour model with a parameter of k=10 where the value 'k' was chosen by us randomly. We note that this model uses the same selected features as kNN (k=5).

Model	Train R2	Test R2	Test MAE	Test RMSE
Linear regression	0.779	0.562	0.384	0.504
KNN (K = 5)	**0.887**	**0.890**	**0.207**	**0.253**
KNN (K = 10)	0.856	0.842	0.236	0.303
Regression tree (depth = 4)	0.952	0.739	0.260	0.389

***Table 4:** The model performance comparison.*

Observing *Table 4*, we conclude that:

1. KNN using 5 neighbours results in the best performance with highest test R^2, lowest RMSE and MAE among the four models. Moreover, kNN(k=5) is also the most robust model since the discrepancy between the training R^2 and testing R^2 is quite small.
2. In our analysis, we see the benefits of tuning the model's hyper-parameters which in turn, reduces the issue of over and under fitting. Taking the kNN regressor for instance, using the optimal k = 5 decided by 10-fold cross validation gives a significantly higher accuracy and more robust results than using our randomly chosen value of 10.
3. Finally, we note that both KNN and regression tree generate better predictions than the multiple linear regression model. We believe that this is attributed to the fact that both the KNN and regression tree model can model non-linearity whereas the linear regression model assumes that predictors are linearly related to the response variable (happiness score).

Residual Plot Analysis:

Residual plots are used to analyse the variance of the error terms to justify the model's quality, whereby error terms are defined by the difference between the predicted value and the true value. To construct a residual plot (*Figure* 13), we used *ResidualsPlot from* the *yellowbrick.regressor* module which requires inputting the model's name into the *ResidualsPlot* object which we named 'visualizer.' The output is shown below:

```
#### Create Residual Plots
from yellowbrick.regressor import ResidualsPlot
visualizer = ResidualsPlot(KNN)
visualizer.fit(X_train, y_train)  # Fit the training data to the visualizer
visualizer.score(X_test, y_test)  # Evaluate the model on the test data
visualizer.show()                 # Finalize and render the figure
```

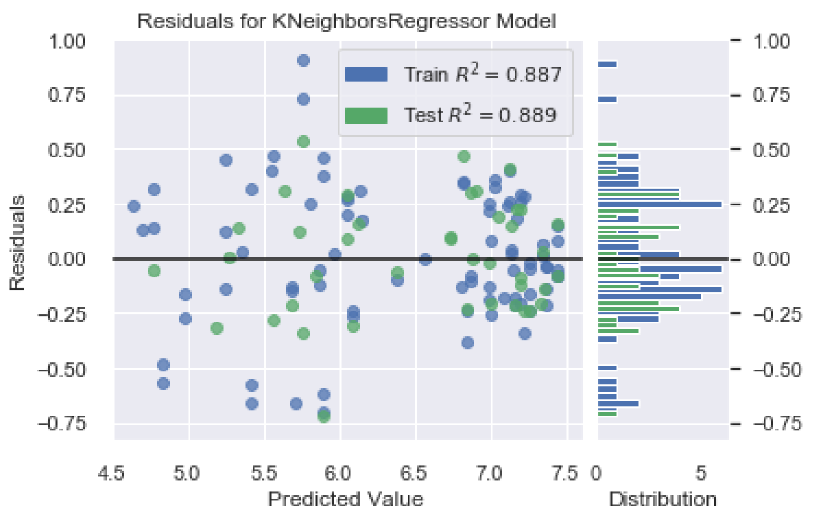


***Figure 13:** Residual plot for KNN (k=5)*

Observing the above residual plot, we can see that the points are randomly dispersed around the horizontal axis and there are numerous residuals far away from 0, indicating that the kNN model does fail to capture some patterns in the data.

3.4 Model Comparison

We compare our three models (multiple linear regression, kNN with k=5 and regression tree with max depth = 4) with regards to the following 4 dimensions:

1. **Accuracy**: As mentioned previously, kNN achieved the highest prediction accuracy, performing significantly better than the other 2 models.

2. **Model Interpretability**:

   - **Linear regression model:** this model is simple to interpret as a positive coefficient simply means that the variable has a positive relationship with the response. However, we note that our estimated coefficients may be biased. For example, the feature 'annual working hours' is shown to have more of a negative relationship with happiness score in *Figure 3.* However, in our linear regression model equation, its coefficient corresponding to 'AvgHours' is positive 0.345. We believe that this is caused by what is called the *omitted-variable bias*, whereby important relevant variables are not included in the model such that the model attempts to attribute the effects of the missing variables to the estimated effects of the included variables. The result is that the coefficients of the included variables become biased.

   - **KNN model:** As a non-parametric model, KNN's algorithm is simple to explain. However, the results do not give insight into what kind of relationship each selected feature has with the target.

   - **Regression Tree model:** The regression tree is perhaps the most difficult model to be understood since the grown tree consists of multiple nodes on each level with each node based off a binary decision. Thus, as you travel down the tree, each node is conditional on every node above it.

3. **Model robustness:** KNN was found to be the most robust model since the difference between the train R^2 and test R^2 is quite small. By contrast, the other two models appear be overfitting due to the large discrepancy between the train and test R^2.

4. **Efficiency and Complexity**: All methods have quite low computation cost, considering that the number of variables is small. However, it is necessary to write additional code for models that involve hyper-parameter tuning. For this reason, the linear regression model is the least time-consuming to build. By contrast, it is necessary to find the optimal number of neighbours in the KNN model and the maximum depth in the regression tree model before fitting the model for prediction.

Model	Accuracy	Interpretability	Robust	Efficiency	Complexity
Linear regression	Low	Normal	Low	High	Low
KNN (K = 5)	High	Normal	Strong	Medium	High
Regression tree (depth = 4)	Medium	Hard	Low	Medium	High

Table 5: *Strengths and weakness of each model.*

Overall, considering these 4 dimensions, we believe that KNN is the best model. Despite the fact that this model may be difficult to interpret and entails slight computational inefficiencies, it produces the most accurate predictions and appears to be robust to overfitting.

3.5 Further Recommendation

Since forward selection is a *greedy algorithm* that adds the best feature (or deletes the worst feature) at each round, the features we selected are 'locally' optimal rather than 'globally' optimal and thus, is not necessarily the best subset. For future work, we can try backward selection to attain a different subset that could potentially produce better results. Furthermore, we believe that the following questions may be worth exploring in the future.