Contributing Factors to a Country's Happiness

"Happiness" can be quite a subjective term. Some people may be happy if they earn enough to buy them lunch and dinner for their family on a daily basis, while others in the exact same scenario may be grossly unhappy because they believe they deserve more but are being impeded by the current socio-economic situations in their homeland. Even two neighbors who share just a wall between them can have exceedingly mismatched views on their nation's current affairs! We attempted to answer the following questions about the data collected by the World Happiness Report.

1. Does a country's income level affect the average level of positive or negative feelings?
2. Which features have the greatest impact on a country's happiness and how well can we use these features to predict a country's happiness?
3. Can we make predictions about future happiness levels based on the happiness levels over time?
4. Do countries with lower levels of happiness have more variability in their happiness levels?

CLASSIFICATION: LOGISTIC REGRESSION

One of the major topics we wanted to explore was whether or not income level affects one's positive or negative feelings. These feelings are captured in the positive and negative affect variables within the dataset. Positive affect was measured as the national average of the binary responses to questions that aim to capture whether the respondent smiled, laughed, or enjoyed during the previous day. Negative affect was measured as the national average of the binary responses to questions that aim to capture whether the respondent felt worried, sad, or angry during the previous day. We found this to be an interesting dynamic to explore when visualizing the data, since the graph of negative affect versus positive affect, with high income countries' data points in green and low income countries' data points in red, does not seem to show that the income level majorly affects the way people feel. This graph is shown below.

We decided to use logistic regression to determine how effective the positive and negative affect variables are at predicting a country's income level (High or Low). The binary outcome that the model predicts is income level (0 for low income and 1 for high income). Originally, we fit a straightforward logistic regression model on the full dataset with positive and negative affect as the independent variables. The results showed that positive and negative affect have a significant p-value, at the $\alpha = 0.05$ level, indicating that the model classifies the data well based on positive and negative affect. The positive value of the positive affect's coefficient shows it is positively related with higher income group, and the negative value of the negative affect variable's coefficient indicates it is negatively related with higher income group and positively related with lower income group.

Logit Regression Results

Dep. Variable:	IncomeGroup	**No. Observations:**	1888
Model:	Logit	**Df Residuals:**	1885
Method:	MLE	**Df Model:**	2
Date:	Wed, 19 May 2021	**Pseudo R-squ.:**	0.03757
Time:	20:01:12	**Log-Likelihood:**	-1219.2
converged:	True	**LL-Null:**	-1266.8
Covariance Type:	nonrobust	**LLR p-value:**	2.142e-21

	coef	std err	z	P>|z|	[0.025	0.975]
const	-1.9943	0.434	-4.590	0.000	-2.846	-1.143
Positive affect	3.9164	0.490	7.989	0.000	2.956	4.877
Negative affect	-1.2553	0.614	-2.044	0.041	-2.459	-0.051

To further explore the effect that these variables have, we split the data into a training and test set, using 1/5 of the data for the test set, and chose the best variables by minimizing the Aikake Information Criterion (AIC) of the model. The variables selected to obtain the minimum AIC were both the positive affect and negative affect variable, which supports our previous results. When using this model to make predictions on the test set, it achieved an accuracy of 0.6269841. This is pretty good considering there are only two predictors and the test accuracy is clearly better than random.

Logit Regression Results

Dep. Variable:	IncomeGroup	**No. Observations:**	1888
Model:	Logit	**Df Residuals:**	1885
Method:	MLE	**Df Model:**	2
Date:	Fri, 21 May 2021	**Pseudo R-squ.:**	0.03757
Time:	02:08:24	**Log-Likelihood:**	-1219.2
converged:	True	**LL-Null:**	-1266.8
Covariance Type:	nonrobust	**LLR p-value:**	2.142e-21

	coef	std err	z	P>|z|	[0.025	0.975]
const	-1.9943	0.434	-4.590	0.000	-2.846	-1.143
Positive affect	3.9164	0.490	7.989	0.000	2.956	4.877
Negative affect	-1.2553	0.614	-2.044	0.041	-2.459	-0.051

In order to extend the analysis and see how the other variables affect the ability to predict income groups using logistic regression, we wanted to do a regression with the whole dataset. We excluded the variable of Log GDP per capita as that is too closely related to income groups. After splitting the data into train and test sets (80/20 split) and selecting the model that minimized AIC cost on the training data, we found that the best variables to use are year, life ladder, social support, healthy life expectancy at birth, negative affect, access to electricity, agricultural land, mortality rate under 5, forest area, military expenditure, and rural population. When we used the model to make predictions on the test set we achieved an accuracy of 0.9081272. This is a high test accuracy rate, which makes sense since we added more predictors.

Logit Regression Results

Dep. Variable:	IncomeGroup	**No. Observations:**	1130
Model:	Logit	**Df Residuals:**	1119
Method:	MLE	**Df Model:**	10
Date:	Fri, 21 May 2021	**Pseudo R-squ.:**	0.6957
Time:	02:13:13	**Log-Likelihood:**	-230.90
converged:	True	**LL-Null:**	-758.85
Covariance Type:	nonrobust	**LLR p-value:**	1.686e-220

	coef	std err	z	P>|z|	[0.025	0.975]
const	206.7859	73.757	2.804	0.005	62.226	351.346
year	-0.0956	0.037	-2.614	0.009	-0.167	-0.024
Life Ladder	0.6252	0.215	2.914	0.004	0.205	1.046
Healthy life expectancy at birth	-0.1895	0.058	-3.278	0.001	-0.303	-0.076
Negative affect	-5.5734	1.737	-3.208	0.001	-8.978	-2.169
Access to electricity (% of population)	0.0415	0.015	2.714	0.007	0.012	0.071
Agricultural land (% of land area)	-0.0446	0.009	-4.861	0.000	-0.063	-0.027
Mortality rate, under-5 (per 1,000 live births)	-0.0848	0.017	-4.910	0.000	-0.119	-0.051
Forest area (% of land area)	0.0219	0.009	2.537	0.011	0.005	0.039
Military expenditure (% of GDP)	0.2507	0.122	2.055	0.040	0.012	0.490
Rural population (% of total population)	-0.0998	0.011	-9.112	0.000	-0.121	-0.078

These findings suggest that, contrary to what we originally expected, a country's positive affect and negative affect in life is indicative of their income level when observed independently of other variables. In this case a greater positive affect corresponds with a higher income group, and a greater negative affect corresponds to a lower income group. However, when considered with all the variables from the dataset, positive affect is no longer selected when minimizing AIC, while negative affect is selected as a good predictor to use. This suggests that overall, a country's negative affect rate appears to be a more important factor in predicting the country's income level in the context of the rest of the data provided.

LINEAR REGRESSION

Many factors contribute to a country's level of happiness. Given the data provided, we want to determine which factors can be used to best predict a country's level of happiness at a given point in time, perhaps also suggesting which factors a country could focus on improving to make their country happier. To accomplish this, we can use linear regression to fit the data to a model that best represents each factor's relation to a country's happiness level.

Feature Selection

The first step is to select the features that are the best predictors of happiness. Since the data has many (more than 20) features, making a linear regression model using all features will likely result in overfitting. To alleviate this risk, we adopt a "bottom-up" strategy, starting with a single best feature and building up to the 10 best features for predicting the happiness score.

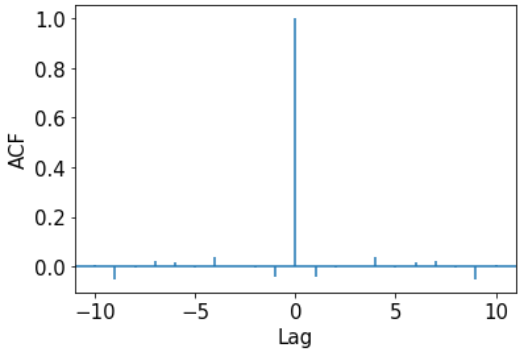
To start this process, we split the data into a training and testing set, with one-third of the records in the testing set. Then we construct a linear regression model using the training data with one feature at a time. In order to select the best features, we select the feature that results in the highest R-squared value, accounting for the largest amount of variance, from its respective model. Next, we then pair this variable with all other variables one at a time and find the model with the best R-squared value. This would mean that those two features paired together make the best model. Similarly, we iterate through, until we have our set of top-10 variables that best predict happiness.

In order to use linear regression, we must test that our data meets the assumptions made by linear regression models.

Testing assumptions

Assumption 1: Check mutual independence of residuals:

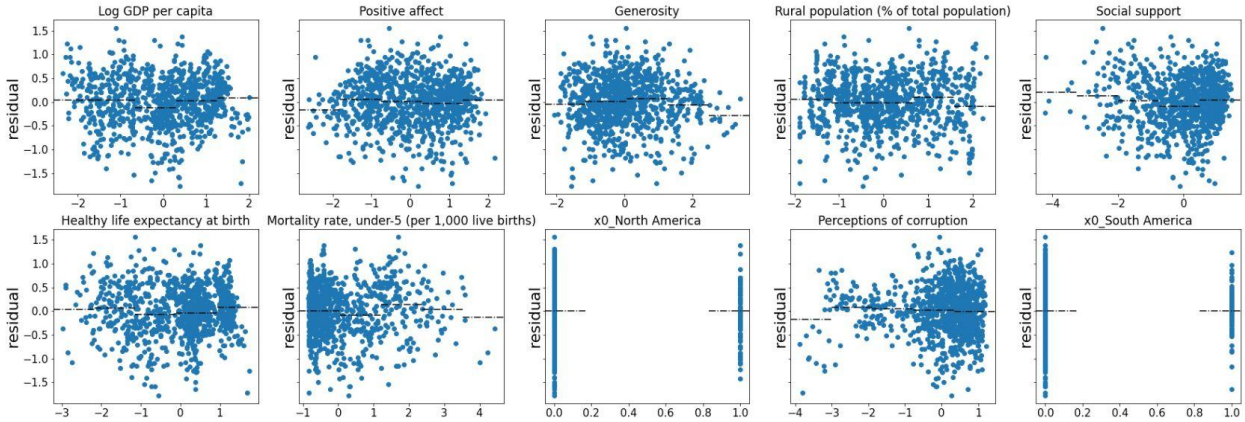
To test that the residuals are mutually independent, we can plot the autocorrelation function. In the plot, we observed that none of the autocorrelations are outside the test bounds, indicating that the residuals are independent of each other.



Assumption 2: Check whether the residuals are independent of the covariates:

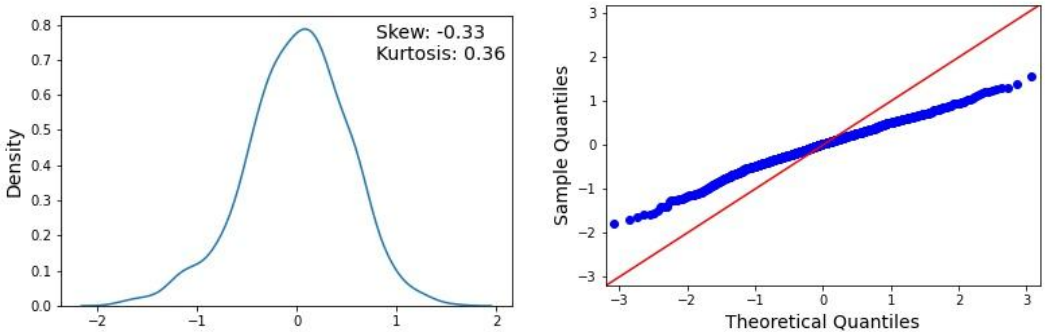
To test whether the residuals are independent of the covariates, we plot the residuals versus the covariates and analyze the average of the residuals using the "window trick".

From the below plots of residuals vs each covariate, we do not see any discernible pattern. On the plots, the black lines are the mean value of the residuals in each "window", and we have set the number of windows as 5. The mean value in each plot seems to be centered around the zero mark and the mean also does not change much as the covariates change.



Assumption 3: Check whether the residuals are normally distributed:

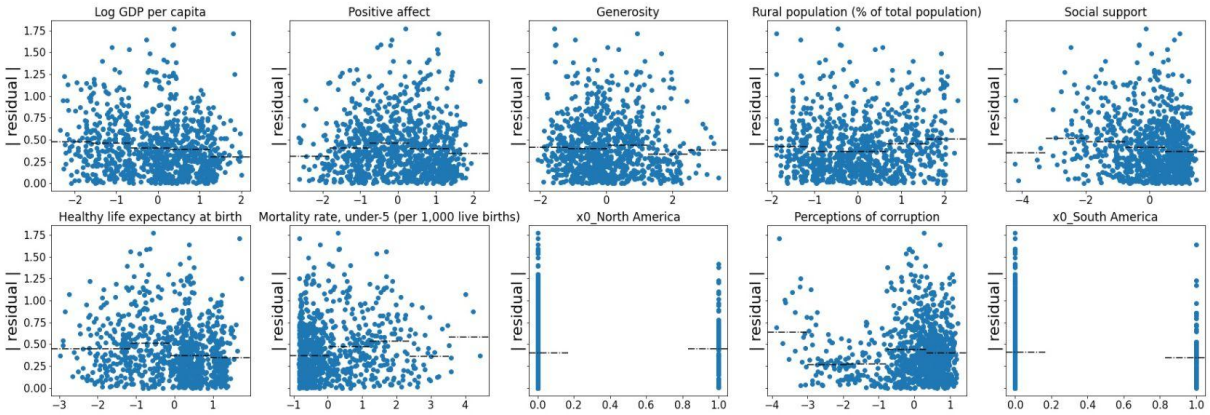
To test whether the residuals are normally distributed, we can use a kernel density estimate (KDE) plot and a QQ-plot.



The QQ-plot seems like a convex-concave type, having light tails. A normal distribution has values of skewness and kurtosis as zero, and as we see from the KDE plot, the skewness and kurtosis of the distribution are well within the acceptable range to be classified as fairly normal, if not exactly so.

Assumption 4: Check whether the residuals have a constant variance:

To test that the residuals have constant variance, we plot the absolute value of the residuals against the fitted values of the covariants. We use the "window trick" to determine if the average of the absolute value of residuals remains relatively constant.



From the above charts, we see that the absolute values of the residuals are independent of the covariates, indicating a fairly homoscedastic nature.

Since all four linear regression assumptions have been met, we can use linear regression with the selected features to make predictions about a country's level of happiness.

Model

After determining that the data meets the assumptions and finding which features to use, we train the model on the training data and obtain the following results:

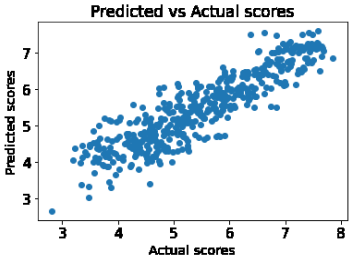
OLS Regression Results

Dep. Variable:	Life Ladder	**R-squared:**	0.792
Model:	OLS	**Adj. R-squared:**	0.790
Method:	Least Squares	**F-statistic:**	356.0
Date:	Thu, 20 May 2021	**Prob (F-statistic):**	1.30e-310
Time:	23:50:13	**Log-Likelihood:**	-713.45
No. Observations:	946	**AIC:**	1449.
Df Residuals:	935	**BIC:**	1502.
Df Model:	10		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	5.3769	0.020	266.531	0.000	5.337	5.417
Log GDP per capita	0.5030	0.049	10.217	0.000	0.406	0.600
Positive affect	0.1364	0.025	5.429	0.000	0.087	0.186
Generosity	0.1356	0.021	6.588	0.000	0.095	0.176
Rural population (% of total population)	-0.0743	0.034	-2.179	0.030	-0.141	-0.007
Social support	0.2316	0.026	9.036	0.000	0.181	0.282
Healthy life expectancy at birth	0.4382	0.052	8.431	0.000	0.336	0.540
Mortality rate, under-5 (per 1,000 live births)	0.3694	0.047	7.805	0.000	0.276	0.462
x0_North America	0.3841	0.067	5.756	0.000	0.253	0.515
Perceptions of corruption	-0.1027	0.020	-5.093	0.000	-0.142	-0.063
x0_South America	0.3164	0.080	3.969	0.000	0.160	0.473

In the model, the R-squared value is 0.792, meaning that 79.2% of the variability in life ladder (happiness level) is accounted for by the features in the model. Additionally, all of the features have p-values of less than 0.05, indicating these features are useful additions to the model in predicting happiness level.

After making the predictions on the test data, we calculate the root mean square error (RMSE) to determine how accurately the model predicts happiness on this data. The RMSE for our model is 0.5, indicating that our predictions on the testing data are on average within 0.5 of the actual value.

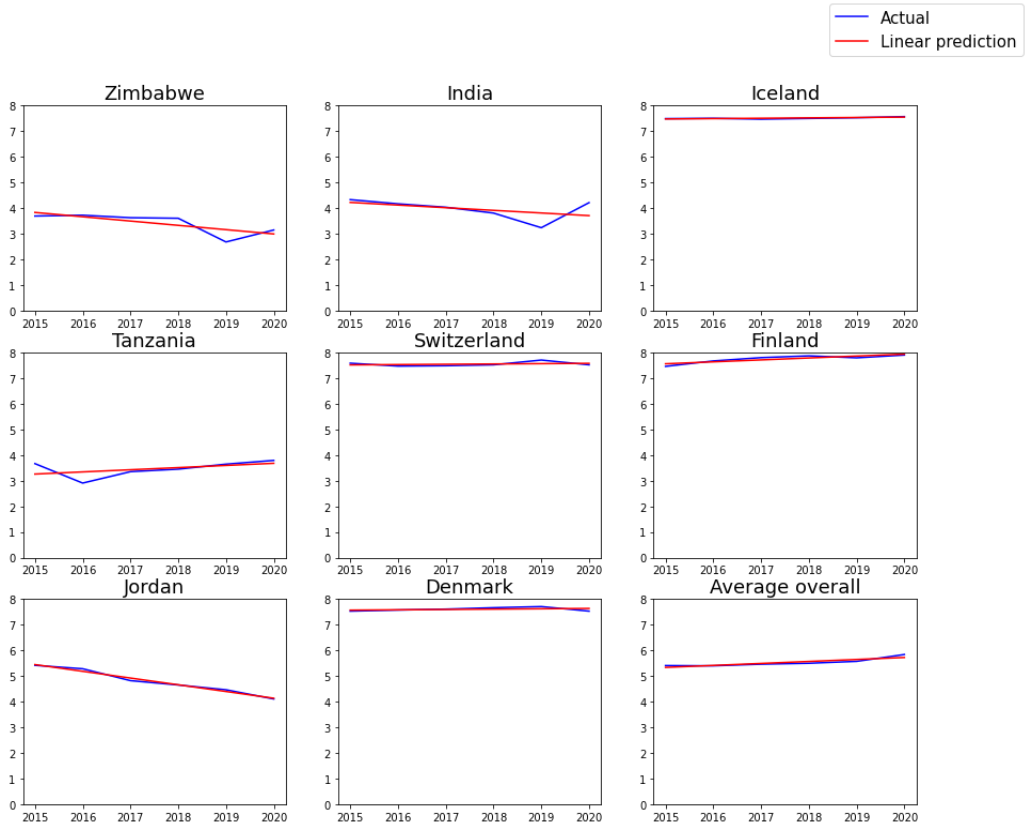


TIME-SERIES FORECASTING

Each year, the level of happiness of individual countries as well as the average happiness level of the world changes. Some countries, such as Denmark, have a remarkably high mean happiness level, while others, such as Zimbabwe, have relatively low average happiness levels.

We decided to analyze the average happiness of each of the top 4 happiest countries, bottom 4 happiest countries, and the world to make predictions and forecast the happiness levels of these countries and the world for the next two years. We first constructed linear regression models for each country as well as the world. Below we plotted the actual measurements and the linear regression model's predictions of happiness levels for the years 2015 through 2020.

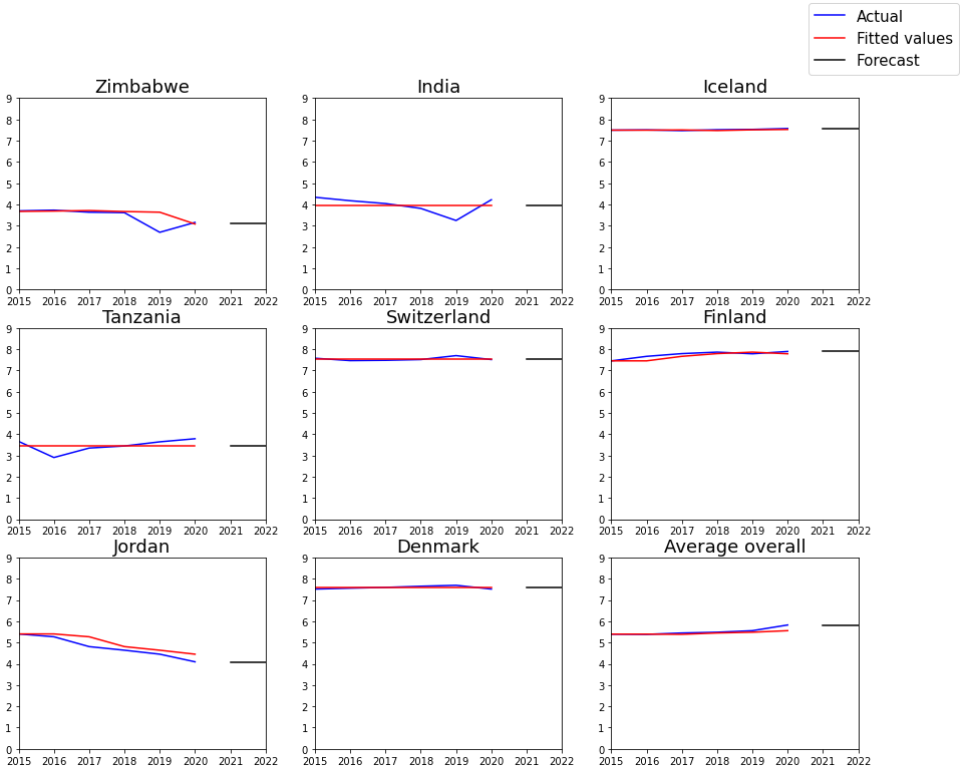
Country	R^2	Adj R^2	F Statistics	Prob	AIC
Zimbabwe	0.570	0.463	5.312	0.0825	4.311
Tanzania	0.245	0.056	1.299	0.318	4.457
Jordan	0.978	0.973	181.5	0.000176	-11.50
India	0.233	0.041	1.215	0.332	7.318
Switzerland	0.068	-0.165	0.2914	0.618	-9.806
Denmark	0.115	-0.106	0.5205	0.511	-12.25
Iceland	0.627	0.503	5.041	0.110	-20.61
Finland	0.754	0.692	12.26	0.0249	-10.29
World Average	0.777	0.721	13.90	0.0203	-10.74



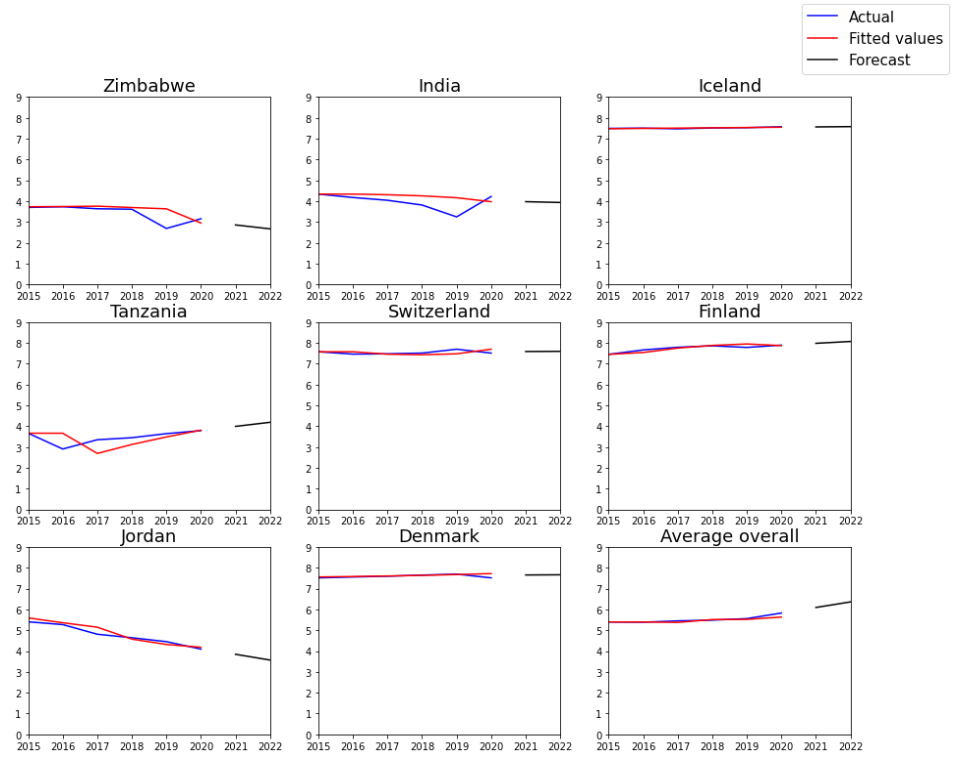
The adjusted R-squared values found for each of the countries as well as the world average suggest that the linear regression model does a poor job of fitting the data for most of these countries. To make better predictions and extrapolate predictions into the future, we can use simple exponential smoothing or Holt's method. Since the plots do not show any particular seasonality in the data, we decided not to use Holt-Winters forecasting.

Country	Simple Exponential Smoothing			Holt's Method		
	2021	2022	SSE	2021	2022	SSE
Zimbabwe	3.13	3.13	0.91	2.86	2.67	0.95
Tanzania	3.46	3.46	0.50	3.99	4.18	1.14
Jordan	4.09	4.09	0.42	3.84	3.57	0.19
India	3.98	3.98	0.80	3.98	3.94	1.20
Switzerland	7.54	7.54	0.04	7.58	7.59	0.10
Denmark	7.59	7.59	0.03	7.65	7.66	0.04
Iceland	7.57	7.57	0.01	7.57	7.58	0.00
Finland	7.89	7.89	0.08	7.98	8.07	0.05
World Average	5.83	5.83	0.08	6.09	6.36	0.05

Simple Exponential Smoothing

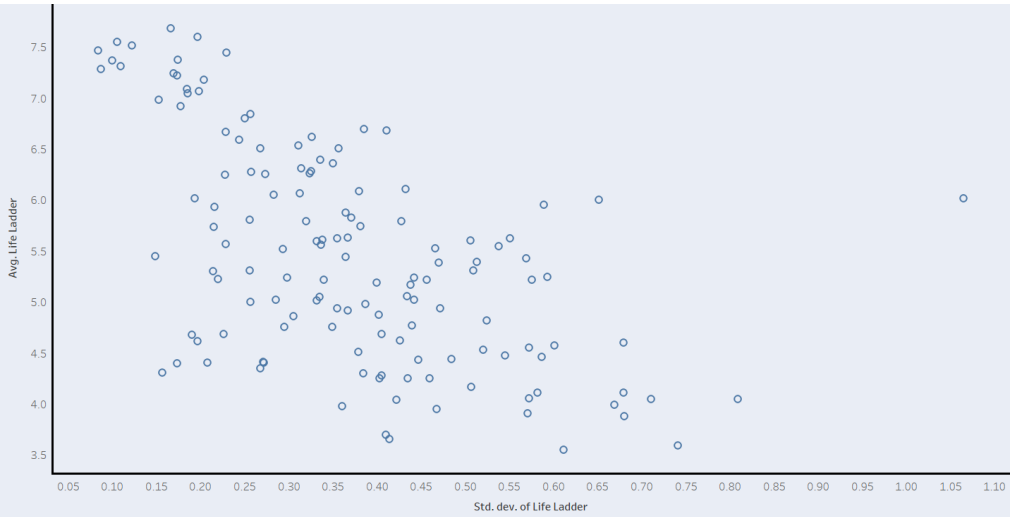


Holt's Method



Simple exponential smoothing forecasts the data without taking into account the trend of the data over time. Therefore, the resulting forecasted happiness level remains constant over 2021 and 2022. The chart above outlines both the forecasted happiness level as well as the sum of squared errors (SSE) for each country. Similarly to simple exponential smoothing, Holt's method also forecasts the data. However, it also takes into account the trend of the happiness levels over time when making predictions. For example, countries with happiness levels that appear to trend downward will have a continued downward trend in the forecasted happiness. The above chart also outlines the forecasted happiness levels and SSEs for each country using Holt's method.

One particularly interesting trend in the SSEs that we noticed is that the countries with lower happiness scores had higher SSEs for both forecasting models. This seems to suggest some instability in the happiness scores of these countries over time which can be observed in the differences between actual and fitted values. This also supports the plot from Milestone 1 indicating that countries with higher average happiness scores had a lower standard deviation of happiness scores shown below. We think this could be due to the fact that this variation could reflect instability within the country.



Additionally, with the exception of Tanzania, countries with lower happiness scores seemed to have a decreasing trend in happiness whereas countries with higher scores have an increasing trend in happiness as indicated by the increasing and decreasing forecasts made by Holt's method.

When choosing which forecasting method to use, we compared the SSEs of the countries between the two models. It is not clear which model is better overall for forecasting because Simple Exponential Smoothing has a lower SSE for some countries while Holt's Method has a lower SSE for others. When looking at the world average happiness scores, Holt's Method has a slightly lower SSE. When Holt's method performs better, this suggests that a country's happiness level over time follows a clearer trend either upward or downward.

CONCLUSION

After conducting these different analytical methods, we were able to answer our proposed questions and extend our analysis. We quantified a relationship between positive and negative affect and income level, identified the variables that affect overall happiness the most while creating a linear model, and forecasted happiness levels for a few countries for the next two years while exploring the relationship between the variability of happiness and the average happiness.

Appendix

Logistic Regression:

```
#math
import numpy as np
import random
#dataframes
import pandas as pd
#plotting
import matplotlib.pyplot as plt
#regression tools
import statsmodels.api as sm
from patsy import dmatrices
from sklearn.model_selection import train_test_split
from sklearn.feature_selection import SelectFromModel
from sklearn.linear_model import LassoCV
#visualization tools
import seaborn as sns
df=pd.read_csv('final_dataset.csv')
#drop rows with null values
imp=df[['Positive affect', 'Negative affect', 'IncomeGroup']]
imp=imp.dropna()
#replace lower middle income and lower income with 0; and upper middle and
upper with 1
imp['IncomeGroup']= imp['IncomeGroup'].replace(['Lower middle income','Low
income'], 0)
imp['IncomeGroup']=imp['IncomeGroup'].replace(['Upper middle income','High
income'], 1)
#fit logistic regression on full data set for preliminary observations
X = sm.add_constant(imp.drop(['IncomeGroup'],axis=1))
y = imp['IncomeGroup']
model = sm.Logit(y,X).fit()
model.summary()
#split train and test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=0)
#cost function
def minAIC(X,y):
    variables = X.columns
```

```
    model = sm.Logit(y,X[variables]).fit()
    while True:
        maxp = np.max(model.pvalues)
        newvariables = variables[model.pvalues < maxp]
        newmodel = sm.Logit(y,X[newvariables]).fit()
        if newmodel.aic < model.aic:
            model = newmodel
            variables = newvariables
        else:
            break
    return model,variables
# select on training set, fit on test set
model,variables = minAIC(X_train, y_train)
model = sm.Logit(y_train,X_train[variables]).fit()
model.summary()
res=(model.predict(X_test[variables])>=0.5).astype(int)
error=((abs(res-y_test)).sum())/len(y_test)
#test accuracy
accuracy=1-error
#get full data set for regression
#drop rows with null values
imp2=df.drop(['Region','Continent','Country name','Log GDP per
capita'],axis=1)
imp2=imp2.dropna()
#replace lower middle income and lower income with 0; and upper middle and
upper with 1
imp2['IncomeGroup']= imp2['IncomeGroup'].replace(['Lower middle
income','Low income'], 0)
imp2['IncomeGroup']=imp2['IncomeGroup'].replace(['Upper middle
income','High income'], 1)
#fit logistic regression
X = sm.add_constant(imp2.drop(['IncomeGroup'],axis=1))
y = imp2['IncomeGroup']
#split train and test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=0)
# select on training set, fit on test set
model,variables = minAIC(X_train, y_train)
model = sm.Logit(y_train,X_train[variables]).fit()
model.summary()
```

```
res=(model.predict(X_test[variables])>=0.5).astype(int)
error=((abs(res-y_test)).sum())/len(y_test)
#test accuracy
accuracy= 1-error
```

Linear Regression:

```
import pandas as pd
import numpy as np
import statsmodels.api as sm
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import OneHotEncoder, StandardScaler
from sklearn.compose import ColumnTransformer
from sklearn.metrics import mean_squared_error
plt.rcParams.update({'font.size': 14, 'xtick.labelsize': 15,
'ytick.labelsize': 15})

df = pd.read_excel('final data/final_dataset.xlsx')
df = df.sort_values(by='year')
df.head()

df.dropna(inplace=True)
X = df.iloc[:, 3:] # all features
y = df.iloc[:, 2] # the life ladder (happiness score)

# list of all numerical columns
a = X.dtypes
num_cols = a[a!= 'O'].index.tolist()

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.33, random_state=0)

ct = ColumnTransformer([
    ('num', StandardScaler(), num_cols),
    ('cat', OneHotEncoder(sparse=False, drop='first'), ['Continent',
'IncomeGroup'])
], remainder='drop')

ft = ct.fit_transform(X_train)
```

```
ls = ct.transformers_[0][2] +
ct.transformers_[1][1].get_feature_names().tolist()

X_train = pd.DataFrame(ft, columns=ls)
X_train.head()


# resetting the index, since the splitting have made the indexing
# non-continuous, and it trips up statsmodels

X_train.reset_index(drop=True, inplace=True)
y_train.reset_index(drop=True, inplace=True)

# finding the top 10 columns:

cols_list = list(X_train.columns)
high_rsq_cols = []
for i in range(10):
    max_rsq = 0 # initializing with zero as the max rsquared
    max_rsq_col = None
    for col in cols_list:
        model = sm.OLS(y_train, sm.add_constant(X_train[high_rsq_cols
+ [col]])).fit()
        if model.rsquared > max_rsq:
            max_rsq = model.rsquared
            max_rsq_col = col

    print(max_rsq_col)
    cols_list.remove(max_rsq_col)
    high_rsq_cols.append(max_rsq_col)

X_train_hrc = X_train[high_rsq_cols] # hrc: high rsquared columns

model = sm.OLS(y_train, sm.add_constant(X_train_hrc)).fit()
model.summary()
```

TESTING MODEL ASSUMPTIONS:

```
(ASSUMPTION 1)
plt.acorr(model.resid)
plt.xlabel("Lag", fontsize=15)
plt.ylabel("ACF", fontsize=15)
plt.show()
# plt.savefig('assumption1.jpeg')
```

```
(ASSUMPTION 2)
X = X_train_hrc.copy()
model = sm.OLS(y_train, sm.add_constant(X)).fit()
resid = model.resid
fig, axes = plt.subplots(2,5, figsize=(30,10), sharey=True)
r, c = 0, 0
windows = 5 # user-specified number of windows.

for col in X.columns: # X is the dataframe of all the covariates

    if col == 'Healthy life expectancy at birth':
        r, c = 1, 0
    axes[r][c].scatter(X[col], resid)
    axes[r][c].set_title(col)
    axes[r][c].set_ylabel('residual', fontsize=22)
    mn = X[col].min()
    mx = X[col].max()

    bd = np.linspace(mn,mx,windows+1) # the boundaries of each window

    for i in range(windows):
        ixs = []
        low = bd[i]
        high = bd[i+1]
        for ix, vl in enumerate(X[col]):
            if (vl >= low) and (vl <= high): #edited this to make it
closed ended comparison
                ixs.append(ix)
        avg = np.average(resid[ixs])
        divs = 1/(windows)
        axes[r][c].axhline(y=avg, xmin=(i*divs), xmax=(i+1)*divs,
color='k', ls='-.')
    c += 1
plt.show()
# plt.savefig('assumption2.jpeg')

(ASSUMPTION 3)
PART A:
sns.kdeplot(model.resid)
plt.text(y=0.7,x=0.8,s=f'Skew:
{np.round(model.resid.skew(),2)}\nKurtosis:
{np.round(model.resid.kurt(),2)}')
plt.show()
# plt.savefig('assumption3_1.jpeg')
```

PART B:

```
sm.qqplot(model.resid, line='45')
plt.show()
# plt.savefig('assumption3_2.jpeg', pad_inches=0)

(ASSUMPTION 4)
X = X_train_hrc.copy()
model = sm.OLS(y_train, sm.add_constant(X)).fit()
resid = np.abs(model.resid)
fig, axes = plt.subplots(2,5, figsize=(30,10), sharey=True)
r, c = 0, 0
windows = 5 # user-specified number of windows.

for col in X.columns: # X is the dataframe of all the covariates

    if col == 'Healthy life expectancy at birth':
        r, c = 1, 0
#     print(r,c)
    axes[r][c].scatter(X[col], resid)
    axes[r][c].set_title(col)
    axes[r][c].set_ylabel('| residual |', fontsize=22)
    mn = X[col].min()
    mx = X[col].max()

    bd = np.linspace(mn,mx,windows+1) # the boundaries of each window

    for i in range(windows):
        ixs = []
        low = bd[i]
        high = bd[i+1]
        for ix, vl in enumerate(X[col]):
            if (vl >= low) and (vl <= high): #edited this to make it
closed ended comparison
                ixs.append(ix)
        avg = np.average(resid[ixs])
        divs = 1/(windows)
        axes[r][c].axhline(y=avg, xmin=(i*divs), xmax=(i+1)*divs,
color='k', ls='-.')
    c += 1
plt.show()
# plt.savefig('assumption4.jpeg')
```

MODEL TESTING (PREDICTIONS):

```
X_test = ct.transform(X_test)
X_test = pd.DataFrame(X_test, columns=ls)
# keeping only the 10 features previously selected, in the test set
X_test_hrc = X_test[high_rsq_cols]

X_test_hrc = sm.add_constant(X_test_hrc)

preds = model.predict(X_test_hrc) # making the predictions

y_test = y_test.reset_index(drop=True)

rmse = np.sqrt(mean_squared_error(y_test, preds))
print(rmse)

plt.scatter(y_test, preds)
plt.xlabel('Actual scores')
plt.ylabel('Predicted scores')
plt.title("Predicted vs Actual scores")
plt.show()
# plt.savefig("pva.jpeg")
```

Forecasting:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import statsmodels.api as sm
from datetime import datetime
from statsmodels.tsa.holtwinters import Holt, ExponentialSmoothing,
SimpleExpSmoothing

df = pd.read_csv('final_dataset.csv')

df = df[df['year'] >= 2015]
df2 = df[df['year'] == 2020]
df2 = df2.sort_values('Life Ladder')
df3 = df2.head(4)
df4 = df2.tail(4)
df2 = pd.concat([df3, df4], axis = 0)
```

```
countries = df2['Country name']

fig, axs = plt.subplots(3, 3, figsize=(15, 12))
plt.rcParams.update({'font.size': 15, 'xtick.labelsize': 10,
'ytick.labelsize': 10})
indX = 0
indY = 0
rsqr_adj = []
for countr in countries:
  dfCountry = df[df['Country name'] == countr]
  X = dfCountry['year']
  y = dfCountry['Life Ladder']
  axs[indX, indY].plot(X, y, color='blue', label='Actual')
  axs[indX, indY].set_title(countr)
  axs[indX, indY].set_ylim(0, 8)
  X = sm.add_constant(X)
  model = sm.OLS(y, X).fit()
  Yhat = model.predict(X)
  print(countr)
  print(model.summary())
  rsqr_adj.append(model.rsquared)
  axs[indX, indY].plot(X['year'], Yhat, color='red', label='Linear
prediction')
  indX += 1
  if indX > 2:
    indX = 0
    indY += 1
X = df.groupby('year')['Life Ladder'].mean()
X.reset_index()
y = X
X = X.index
axs[2, 2].plot(X, y, color='blue', label='Actual')
axs[2, 2].set_title('Average overall')
axs[2, 2].set_ylim(0, 8)
X = sm.add_constant(X)
model = sm.OLS(y, X).fit()
print('World average')
print(model.summary())
rsqr_adj.append(model.rsquared_adj)
Yhat = model.predict(X)
```

```
axs[2, 2].plot(X[:, 1], Yhat, color='red', label='Linear prediction')
handles, labels = axs[2, 2].get_legend_handles_labels()
fig.legend(handles, labels, loc='upper right')
print(rsqr_adj)

fig, axs = plt.subplots(3, 3, figsize=(15, 12))
plt.rcParams.update({'font.size': 15, 'xtick.labelsize': 10,
'ytick.labelsize': 10})
indX = 0
indY = 0
sse = []
for countr in countries:
  dfCountry = df[df['Country name'] == countr]
  X = dfCountry.set_index('year')['Life Ladder']
  meanX = X.mean()
  X = X.reindex(range(2015, 2021), fill_value=meanX)
  X = X.reindex(range(2015, 2021), fill_value=meanX)
  X.index = pd.to_datetime(X.index, format='%Y')
  fit = SimpleExpSmoothing(X).fit()
  Yhat = fit.fittedvalues
  fore = fit.forecast(2)
  sse.append(fit.sse)
  fore.index = fore.index.year
  print(countr)
  print(fore)
  print(fit.sse)
  axs[indX, indY].plot(X.index.year, X, color='blue', label='Actual')
  axs[indX, indY].set_title(countr)
  axs[indX, indY].set_ylim(0, 9)
  axs[indX, indY].set_xlim(2015, 2022)
  axs[indX, indY].plot(Yhat.index.year, Yhat, color='red', label='Fitted
values')
  axs[indX, indY].plot(fore, color='black', label='Forecast')
  indX += 1
  if indX > 2:
    indX = 0
    indY += 1
X = df.groupby('year')['Life Ladder'].mean()
meanX = X.mean()
X = X.reindex(range(2015, 2021), fill_value=meanX)
```

```
X.index = pd.to_datetime(X.index, format='%Y')
axs[2, 2].plot(X.index.year, X, color='blue', label='Actual')
axs[2, 2].set_title('Average overall')
axs[2, 2].set_ylim(0, 9)
axs[2, 2].set_xlim(2015, 2022)
fit = SimpleExpSmoothing(X).fit()
sse.append(fit.sse)
Yhat = fit.fittedvalues
fore = fit.forecast(2)
fore.index = fore.index.year
print('World average')
print(fore)
print(fit.sse)
axs[2, 2].plot(Yhat.index.year, Yhat, color='red', label='Fitted values')
axs[indX, indY].plot(fore, color='black', label='Forecast')
handles, labels = axs[2, 2].get_legend_handles_labels()
fig.legend(handles, labels, loc='upper right')
print(sse)

fig, axs = plt.subplots(3, 3, figsize=(15, 12))
plt.rcParams.update({'font.size': 15, 'xtick.labelsize': 10,
'ytick.labelsize': 10})
indX = 0
indY = 0
sse = []
for countr in countries:
  dfCountry = df[df['Country name'] == countr]
  X = dfCountry.set_index('year')['Life Ladder']
  meanX = X.mean()
  X = X.reindex(range(2015, 2021), fill_value=meanX)
  X.index = pd.to_datetime(X.index, format='%Y')
  fit = Holt(X).fit()
  sse.append(fit.sse)
  Yhat = fit.fittedvalues
  fore = fit.forecast(2)
  fore.index = fore.index.year
  print(countr)
  print(fore)
  print(fit.sse)
  axs[indX, indY].plot(X.index.year, X, color='blue', label='Actual')
```

```
  axs[indX, indY].set_title(countr)
  axs[indX, indY].set_ylim(0, 9)
  axs[indX, indY].set_xlim(2015, 2022)
  axs[indX, indY].plot(Yhat.index.year, Yhat, color='red', label='Fitted
values')
  axs[indX, indY].plot(fore, color='black', label='Forecast')
  indX += 1
  if indX > 2:
    indX = 0
    indY += 1
X = df.groupby('year')['Life Ladder'].mean()
meanX = X.mean()
X = X.reindex(range(2015, 2021), fill_value=meanX)
X.index = pd.to_datetime(X.index, format='%Y')
axs[2, 2].plot(X.index.year, X, color='blue', label='Actual')
axs[2, 2].set_title('Average overall')
axs[2, 2].set_ylim(0, 9)
axs[2, 2].set_xlim(2015, 2022)
fit = Holt(X).fit()
sse.append(fit.sse)
Yhat = fit.fittedvalues
fore = fit.forecast(2)
fore.index = fore.index.year
print('World average')
print(fore)
print(fit.sse)
axs[2, 2].plot(Yhat.index.year, Yhat, color='red', label='Fitted values')
axs[indX, indY].plot(fore, color='black', label='Forecast')
handles, labels = axs[2, 2].get_legend_handles_labels()
fig.legend(handles, labels, loc='upper right')
print(sse)
```

References

Folaraz. (2017, October) . World Countries and Continents Details, Version 3. Retrieved April 27, 2021 from https://www.kaggle.com/folaraz/world-countries-and-continents-details

Helliwell, John F., Richard Layard, Jeffrey Sachs, and Jan-Emmanuel De Neve. (2021) . World Happiness Report 2021. New York: Sustainable Development Solutions Network. Retrieved April 27, 2021 from https://worldhappiness.report/ed/2021/#appendices-and-data

Raschka, Sebastian. "Sequential Feature Selector." *Mlxtend*, MkDocs, rasbt.github.io/mlxtend/user_guide/feature_selection/SequentialFeatureSelector/

World Bank, World Development Indicators. (2017) . *Indicators*. Retrieved April 27, 2021 from https://data.worldbank.org/indicator