HAPPINESS ATLAS: SEEKING NIRVANA

Presented By:

Nirvana Ganpat, Bryant Beckford, Todd Meier, Stephan Guillaume

Outline

Can We Predict World Happiness?

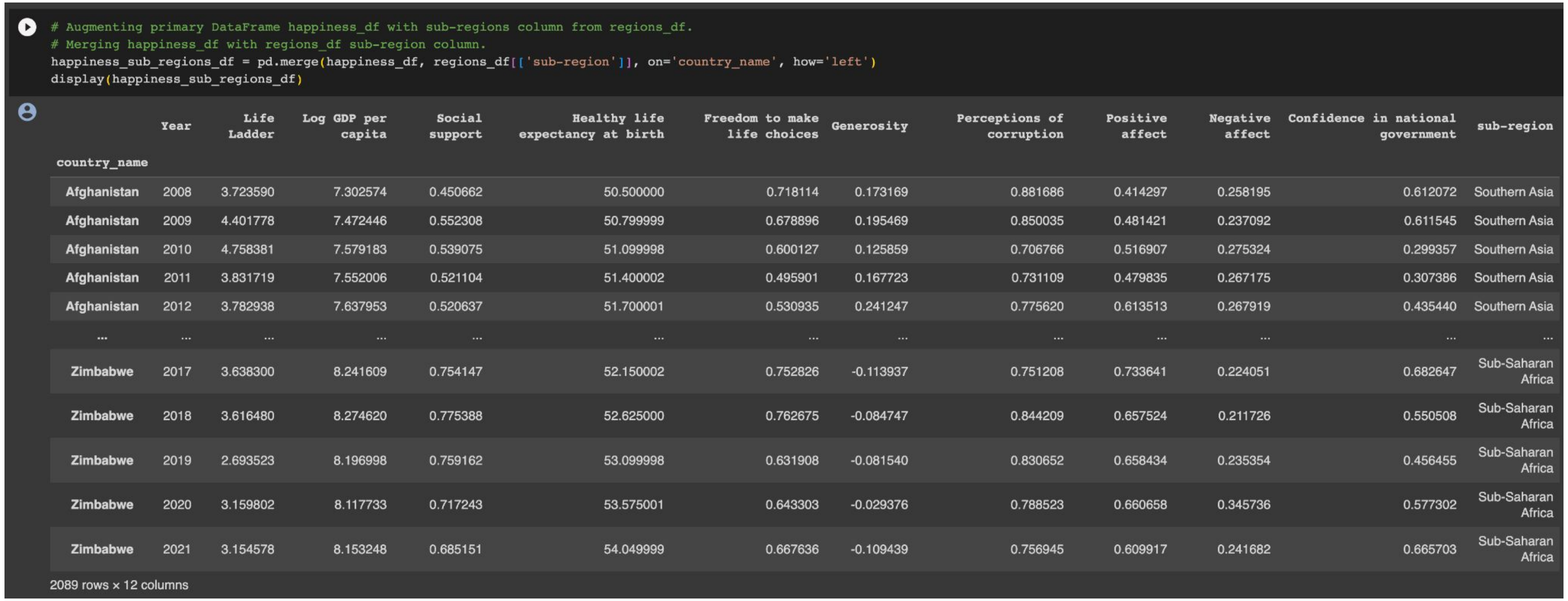
- World Happiness Report dataset spanning the years 2005-2021, an annual publication that measures happiness levels across countries.
- This dataset offers a wealth of information on factors like economic growth, social support, freedom, generosity, and trust in institutions, all of which impact well-being.
- Our project focuses on comparing two machine learning models using this dataset to gain insights into what drives happiness trends globally.

Defining Happiness

- We defined happiness as the Cantril Life Ladder, or self-anchoring scale, developed and published by Princeton psychology researcher Hadley Cantril in 1965 ("Pattern of Human Concerns").
- Gallup has been conducting a global Cantril Life Ladder survey since 2005, asking respondents to value their lives today on an integer scale, with the worst possible life as a 0 and the best possible life as a 10.
- According to Gallup, this survey allows us to compare happiness levels and inequality in different parts of the world.

Exploring World Happiness Data

This dataset captured factors influencing happiness levels in different countries. Some of the key variables included: *GDP per capita, Social Support, Health Expectancy, Freedom, Generosity, and Corruption Perception*. These were all valuable resources for predicting happiness.



```
# Augmenting primary DataFrame happiness_df with sub-regions column from regions_df.
# Merging happiness_df with regions_df sub-region column.
happiness_sub_regions_df = pd.merge(happiness_df, regions_df[['sub-region']], on='country_name', how='left')
display(happiness_sub_regions_df)
```

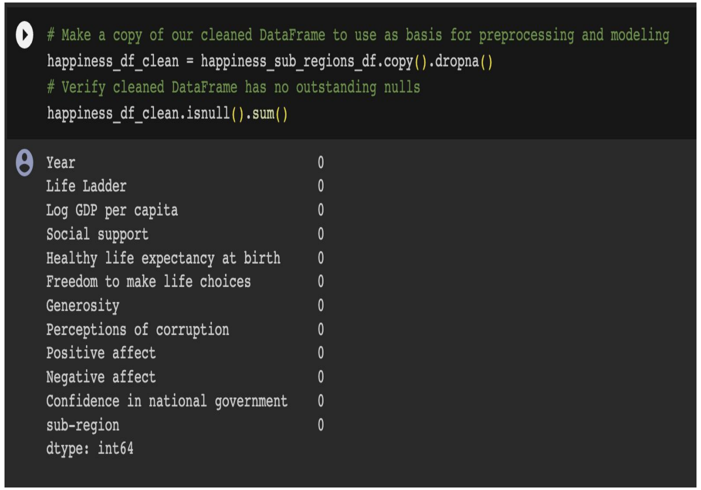
country_name	Year	Life Ladder	Log GDP per capita	Social support	Healthy life expectancy at birth	Freedom to make life choices	Generosity	Perceptions of corruption	Positive affect	Negative affect	Confidence in national government	sub-region
Afghanistan	2008	3.723590	7.302574	0.450662	50.500000	0.718114	0.173169	0.881686	0.414297	0.258195	0.612072	Southern Asia
Afghanistan	2009	4.401778	7.472446	0.552308	50.799999	0.678896	0.195469	0.850035	0.481421	0.237092	0.611545	Southern Asia
Afghanistan	2010	4.758381	7.579183	0.539075	51.099998	0.600127	0.125859	0.706766	0.516907	0.275324	0.299357	Southern Asia
Afghanistan	2011	3.831719	7.552006	0.521104	51.400002	0.495901	0.167723	0.731109	0.479835	0.267175	0.307386	Southern Asia
Afghanistan	2012	3.782938	7.637953	0.520637	51.700001	0.530935	0.241247	0.775620	0.613513	0.267919	0.435440	Southern Asia
...	...	...	...	...	...	...	...	...	...	...	...	...
Zimbabwe	2017	3.638300	8.241609	0.754147	52.150002	0.752826	-0.113937	0.751208	0.733641	0.224051	0.682647	Sub-Saharan Africa
Zimbabwe	2018	3.616480	8.274620	0.775388	52.625000	0.762675	-0.084747	0.844209	0.657524	0.211726	0.550508	Sub-Saharan Africa
Zimbabwe	2019	2.693523	8.196998	0.759162	53.099998	0.631908	-0.081540	0.830652	0.658434	0.235354	0.456455	Sub-Saharan Africa
Zimbabwe	2020	3.159802	8.117733	0.717243	53.575001	0.643303	-0.029376	0.788523	0.660658	0.345736	0.577302	Sub-Saharan Africa
Zimbabwe	2021	3.154578	8.153248	0.685151	54.049999	0.667636	-0.109439	0.756945	0.609917	0.241682	0.665703	Sub-Saharan Africa

2089 rows × 12 columns

Data Preprocessing Techniques

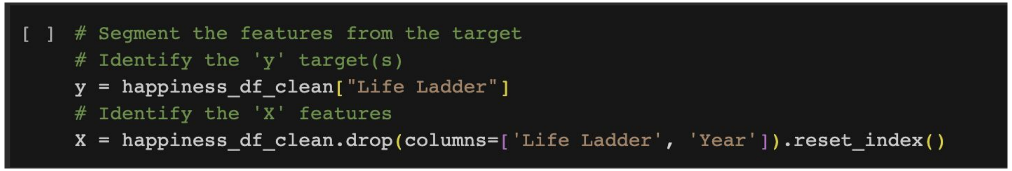
In order to guarantee optimal performance for our machine learning model(s), there were crucial steps taken to ensure the data was well prepared and the model was effectively trained and evaluated.

- Handling missing values, encoding categorical variables, and scaling numerical features.
- Outlier removal and addressing skewed distributions to improve model performance.
- Separating the features (X) from the target (y)

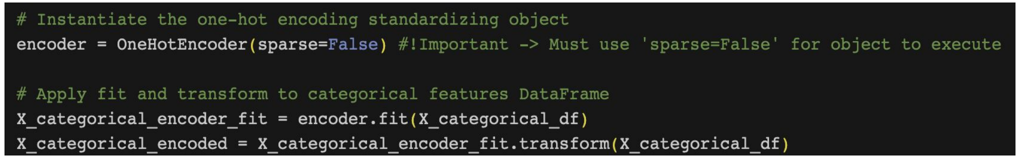


```
# Make a copy of our cleaned DataFrame to use as basis for preprocessing and modeling
happiness_df_clean = happiness_sub_regions_df.copy().dropna()
# Verify cleaned DataFrame has no outstanding nulls
happiness_df_clean.isnull().sum()
```

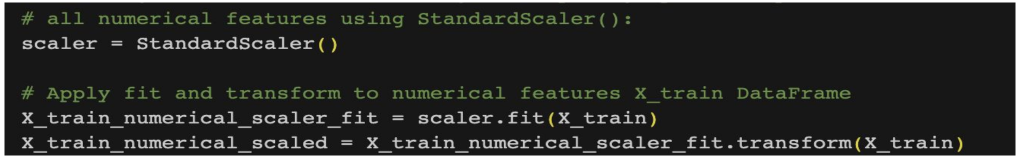
```
Year                                0
Life Ladder                         0
Log GDP per capita                  0
Social support                      0
Healthy life expectancy at birth    0
Freedom to make life choices        0
Generosity                          0
Perceptions of corruption           0
Positive affect                     0
Negative affect                     0
Confidence in national government   0
sub-region                          0
dtype: int64
```



```
[ ] # Segment the features from the target
    # Identify the 'y' target(s)
    y = happiness_df_clean["Life Ladder"]
    # Identify the 'X' features
    X = happiness_df_clean.drop(columns=['Life Ladder', 'Year']).reset_index()
```

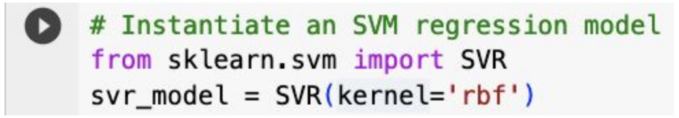


```
# Instantiate the one-hot encoding standardizing object
encoder = OneHotEncoder(sparse=False) #!Important -> Must use 'sparse=False' for object to execute

# Apply fit and transform to categorical features DataFrame
X_categorical_encoder_fit = encoder.fit(X_categorical_df)
X_categorical_encoded = X_categorical_encoder_fit.transform(X_categorical_df)
```

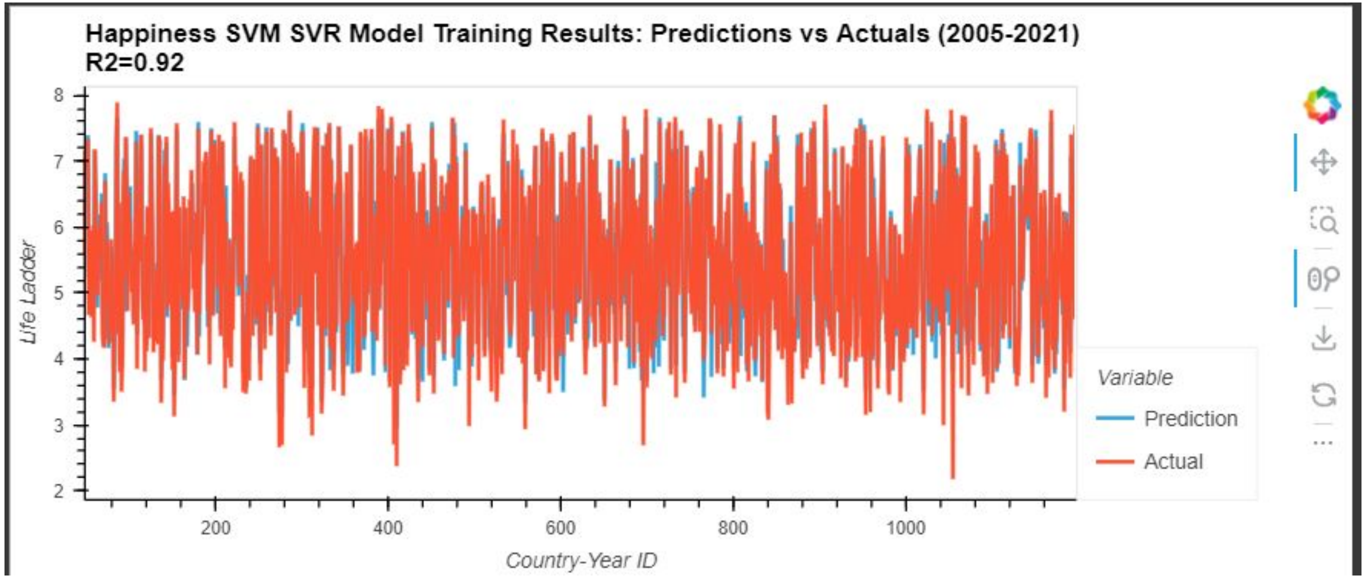


```
# all numerical features using StandardScaler():
scaler = StandardScaler()

# Apply fit and transform to numerical features X_train DataFrame
X_train_numerical_scaler_fit = scaler.fit(X_train)
X_train_numerical_scaled = X_train_numerical_scaler_fit.transform(X_train)
```

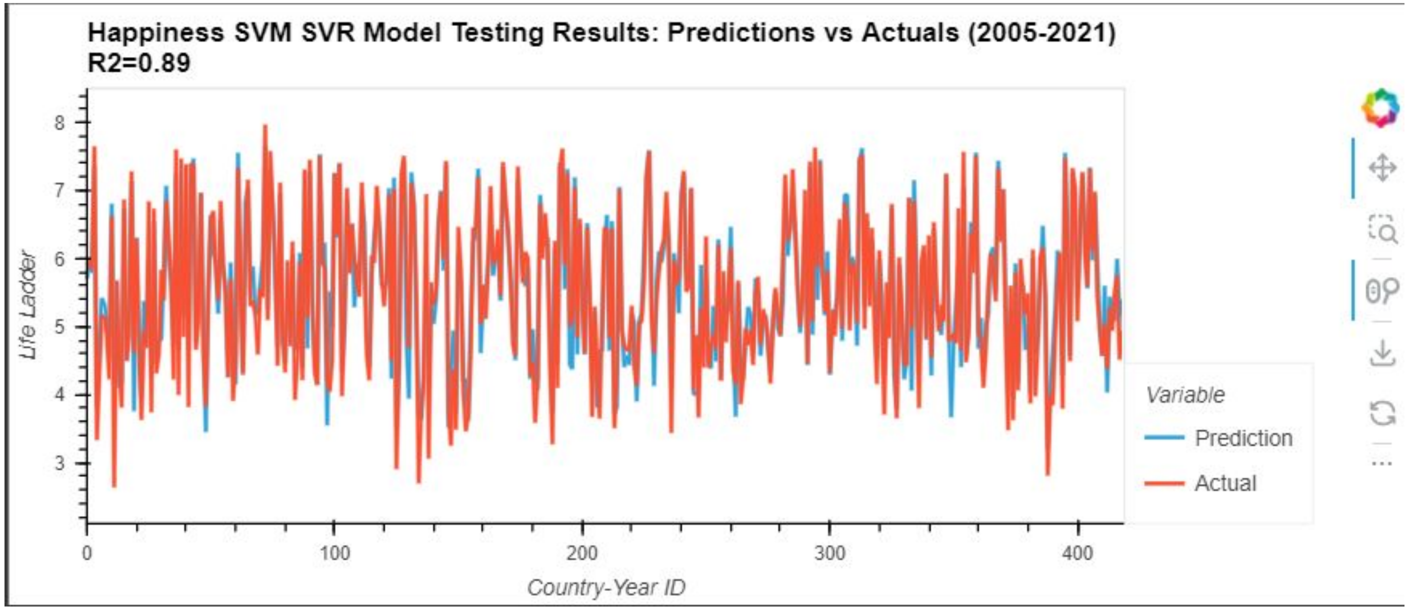
Support Vector Model Regression Model (Train)



```
# Instantiate an SVM regression model
from sklearn.svm import SVR
svr_model = SVR(kernel='rbf')
```



Support Vector Model Regression Model (Test)



Neural Network Model

The Neural Network Model:

- One input layer with 164 inputs
- One hidden layer with 328 nodes. ReLu activation function.
- One output layer with 1 class. ReLu activation function.

```
# Define the model's parameters
node_multiplier = 2
number_inputs = 164
number_hidden_nodes = node_multiplier * X_standardized_recombined_df.shape[1]
#number_hidden_nodes = node_multiplier * X.shape[1]
#print(number_hidden_nodes)
number_classes = 1
# Observations: Keeping epochs constant at 50, the loss function struggled to \n
# converge to the SVR MSE when a 2-3 node multiplier was used on the pre-standardized \n
# features count of 11, and therefore the standardized features dataset count of 164 \n
# was relied on instead to inform the model.  Using a node_multiplier of 1-3 against \n
# the 164 features demonstrated reasonable convergence of the loss function, however, \n
# once the multiplier was dropped to 0.5, convergence was no longer satisfactory.
```

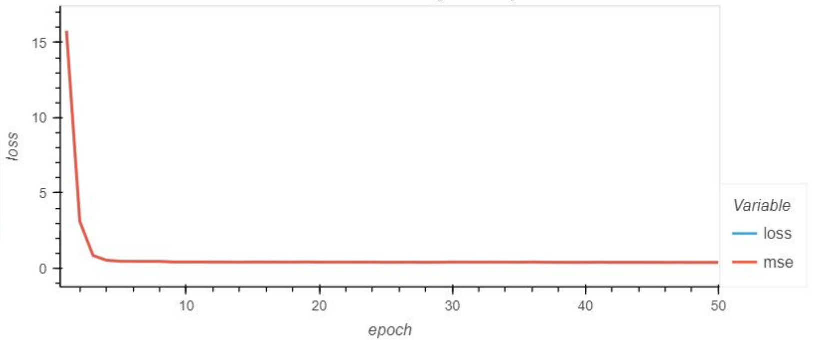
```
# Add the input and hidden layer(s) to the model
# Occam's Razor/KISS. We will first attempt to model the problem with a single hidden layer
# Using ReLu as activation function instead of Linear to capture potential non-linear relations
neural_net.add(Dense(units = number_hidden_nodes, activation = "relu", input_dim = number_inputs))
```

```
# Add the output layer(s) to the model
neural_net.add(Dense(units=number_classes, activation="relu"))
```

```
# Review the setup of our neural network model, layers, and parameters
neural_net.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_2 (Dense)             (None, 328)               54120

 dense_3 (Dense)             (None, 1)                 329

=================================================================
Total params: 54449 (212.69 KB)
Trainable params: 54449 (212.69 KB)
Non-trainable params: 0 (0.00 Byte)
```

Neural Net Model Loss Function Training History



Comparing Both Models

SVM Model

- Preparing the data for both the Support Vector Machine (SVM) Regression Neural Net models as difficult.
- This model gave us a robust testing outcome using training data.
- The model was a more intuitive traditional regression model than the neural net.

Neural Network Model

- The model converged quickly on a training solution and confirmed the mean squared error (MSE) of the SVM.
- Evaluation of the test data also confirmed a robust neural net model, and an MSE comparable to the SVM's.
- The neural net model output is less intuitive and varies with each iterative session

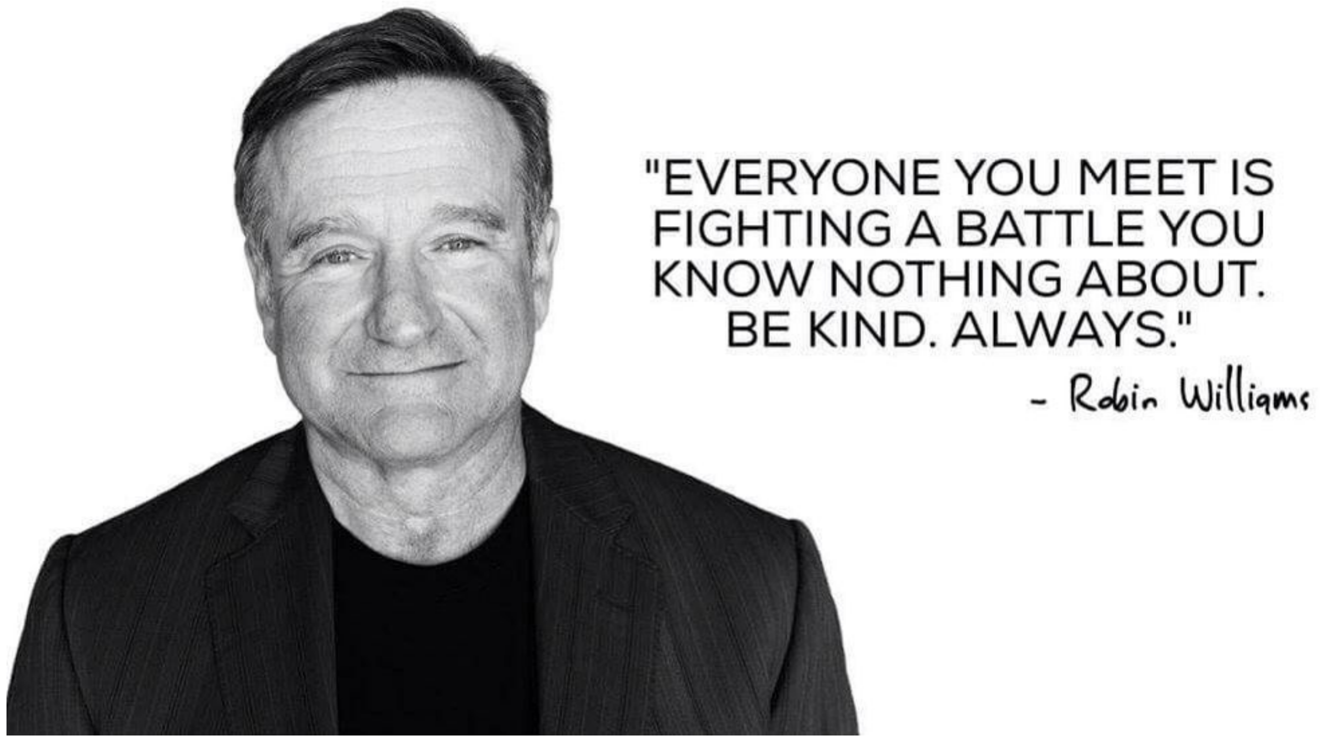
Parting Shots, or Thoughts

Fictional Character - Mr. Micawber's Observations on Happiness From 'David Copperfield':

" Annual income twenty pounds, annual expenditure nineteen nineteen and six, result happiness. Annual income twenty pounds, annual expenditure twenty pounds ought and six, result misery. "

Charles Dickens (David Copperfield)

Final Thoughts



"EVERYONE YOU MEET IS FIGHTING A BATTLE YOU KNOW NOTHING ABOUT. BE KIND. ALWAYS."

- Robin Williams

Thank You For Attending!

Q&A

https://github.com/boz-tcm/happiness_atlas.git

Post-Mortem

Considerations if we had more time:

- Incorporate GPS coordinates by center–of-country and center-of-region mass to view happiness globally in Geoviews.
- Incorporate and explore other explanatory features, such as Form of Government and/or Economy, Developing vs. Developed Economies.
- Numerical scaling alternatives: StandardScaler() vs. MinMaxScaler() vs MaxAbsScaler(). For example, it wasn't appropriate for us to use StandardScaler on our 'Year' column since it was uniformly, not normally distributed.
- Expanding our target from a single Life Ladder target to a multi-target, such as Positive Affect, Negative Affect, and Life Ladder, in a neural net model.
- Where we incorporate a time-series, introduce ARIMA or ARMA to model unexplained regression residuals.