Lab 8: Define and Solve an ML Problem of Your Choosing

In [1]:
```
import pandas as pd
import numpy as np
import os
import matplotlib.pyplot as plt
import seaborn as sns
```

In this lab assignment, you will follow the machine learning life cycle and implement a model to solve a machine learning problem of your choosing. You will select a data set and choose a predictive problem that the data set supports. You will then inspect the data with your problem in mind and begin to formulate a project plan. You will then implement the machine learning project plan.

You will complete the following tasks:

1. Build Your DataFrame
2. Define Your ML Problem
3. Perform exploratory data analysis to understand your data.
4. Define Your Project Plan
5. Implement Your Project Plan:
    - Prepare your data for your model.
    - Fit your model to the training data and evaluate your model.
    - Improve your model's performance.

Part 1: Build Your DataFrame

You will have the option to choose one of four data sets that you have worked with in this program:

- The "census" data set that contains Census information from 1994: `censusData.csv`
- Airbnb NYC "listings" data set: `airbnbListingsData.csv`
- World Happiness Report (WHR) data set: `WHR2018Chapter2OnlineData.csv`
- Book Review data set: `bookReviewsData.csv`

Note that these are variations of the data sets that you have worked with in this program. For example, some do not include some of the preprocessing necessary for specific models.

Load a Data Set and Save it as a Pandas DataFrame

The code cell below contains filenames (path + filename) for each of the four data sets available to you.

Task: In the code cell below, use the same method you have been using to load the data using `pd.read_csv()` and save it to DataFrame `df`.

You can load each file as a new DataFrame to inspect the data before choosing your data set.

In [3]:
```
# File names of the four data sets
adultDataSet_filename = os.path.join(os.getcwd(), "data", "censusData.csv")
airbnbDataSet_filename = os.path.join(os.getcwd(), "data", "airbnbListingsData.c
WHRDataSet_filename = os.path.join(os.getcwd(), "data", "WHR2018Chapter2OnlineDa
bookReviewDataSet_filename = os.path.join(os.getcwd(), "data", "bookReviewsData.


df = pd.read_csv(WHRDataSet_filename)

df.head()
```

Out[3]:

	country	year	Life Ladder	Log GDP per capita	Social support	Healthy life expectancy at birth	Freedom to make life choices	Generosity	F
0	Afghanistan	2008	3.723590	7.168690	0.450662	49.209663	0.718114	0.181819	
1	Afghanistan	2009	4.401778	7.333790	0.552308	49.624432	0.678896	0.203614	
2	Afghanistan	2010	4.758381	7.386629	0.539075	50.008961	0.600127	0.137630	
3	Afghanistan	2011	3.831719	7.415019	0.521104	50.367298	0.495901	0.175329	
4	Afghanistan	2012	3.782938	7.517126	0.520637	50.709263	0.530935	0.247159	

Part 2: Define Your ML Problem

Next you will formulate your ML Problem. In the markdown cell below, answer the following questions:

1. List the data set you have chosen.
2. What will you be predicting? What is the label?
3. Is this a supervised or unsupervised learning problem? Is this a clustering, classification or regression problem? Is it a binary classificaiton or multi-class classifiction problem?
4. What are your features? (note: this list may change after your explore your data)
5. Explain why this is an important problem. In other words, how would a company create value with a model that predicts this label?

ML Problem Definition

1. Dataset Chosen: World Happiness Report Data (WHR2018Chapter2OnlineData.csv)

2. Prediction/Label: No label; cluster countries based on similarity in well-being factors.
3. Type of Problem: Unsupervised learning, clustering.
4. Features: Log GDP per capita, Social support, Healthy life expectancy at birth, Freedom to make life choices, Generosity, Perceptions of corruption.
5. Importance: Clustering helps identify groups of countries with similar well-being profiles, informing targeted policy interventions or international aid strategies. For example, policymakers can prioritize resources for clusters with low well-being to improve global happiness.

Part 3: Understand Your Data

The next step is to perform exploratory data analysis. Inspect and analyze your data set with your machine learning problem in mind. Consider the following as you inspect your data:

1. What data preparation techniques would you like to use? These data preparation techniques may include:
    - addressing missingness, such as replacing missing values with means
    - finding and replacing outliers
    - renaming features and labels
    - finding and replacing outliers
    - performing feature engineering techniques such as one-hot encoding on categorical features
    - selecting appropriate features and removing irrelevant features
    - performing specific data cleaning and preprocessing techniques for an NLP problem
    - addressing class imbalance in your data sample to promote fair AI
2. What machine learning model (or models) you would like to use that is suitable for your predictive problem and data?
    - Are there other data preparation techniques that you will need to apply to build a balanced modeling data set for your problem and model? For example, will you need to scale your data?
3. How will you evaluate and improve the model's performance?
    - Are there specific evaluation metrics and methods that are appropriate for your model?

Think of the different techniques you have used to inspect and analyze your data in this course. These include using Pandas to apply data filters, using the Pandas `describe()` method to get insight into key statistics for each column, using the Pandas `dtypes` property to inspect the data type of each column, and using Matplotlib and Seaborn to detect outliers and visualize relationships between features and labels. If you are working on a classification problem, use techniques you have learned to determine if there is class imbalance.

Task: Use the techniques you have learned in this course to inspect and analyze your data. You can import additional packages that you have used in this course that you will need to perform this task.

Note: You can add code cells if needed by going to the **Insert** menu and clicking on **Insert Cell Below** in the drop-drown menu.

In [8]:
```
print(f"Shape: {df.shape}")
print(f"Number of countries/observations: {df.shape[0]}")

# detailed info on dataset
df.info()
```

```
Shape: (1562, 19)
Number of countries/observations: 1562
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1562 entries, 0 to 1561
Data columns (total 19 columns):
 #   Column                                            Non-Null Count  Dt
ype
---  ------                                            --------------  --
---
 0   country                                           1562 non-null   ob
ject
 1   year                                              1562 non-null   in
t64
 2   Life Ladder                                       1562 non-null   fl
oat64
 3   Log GDP per capita                                1535 non-null   fl
oat64
 4   Social support                                    1549 non-null   fl
oat64
 5   Healthy life expectancy at birth                  1553 non-null   fl
oat64
 6   Freedom to make life choices                      1533 non-null   fl
oat64
 7   Generosity                                        1482 non-null   fl
oat64
 8   Perceptions of corruption                         1472 non-null   fl
oat64
 9   Positive affect                                   1544 non-null   fl
oat64
 10  Negative affect                                   1550 non-null   fl
oat64
 11  Confidence in national government                 1401 non-null   fl
oat64
 12  Democratic Quality                                1391 non-null   fl
oat64
 13  Delivery Quality                                  1391 non-null   fl
oat64
 14  Standard deviation of ladder by country-year      1562 non-null   fl
oat64
 15  Standard deviation/Mean of ladder by country-year 1562 non-null   fl
oat64
 16  GINI index (World Bank estimate)                  583 non-null    fl
oat64
 17  GINI index (World Bank estimate), average 2000-15 1386 non-null   fl
oat64
 18  gini of household income reported in Gallup, by wp5-year  1205 non-null   fl
oat64
dtypes: float64(17), int64(1), object(1)
memory usage: 232.0+ KB
```

In [9]:

```python
# Summary statistics for numeric columns
print("\nSummary Statistics:")
print(df.describe())

# workable columns
print("\nColumn Names:")
for i, col in enumerate(df.columns):
    print(f"{i+1}. {col}")

# get numeric columns for clustering
```

```
numeric_cols = df.select_dtypes(include=[np.number]).columns.tolist()
print(f"\nFound {len(numeric_cols)} numeric columns:")
print(numeric_cols)
```

```
Summary Statistics:
              year  Life Ladder  Log GDP per capita  Social support  \
count  1562.000000  1562.000000         1535.000000     1549.000000
mean   2011.820743     5.433676            9.220822        0.810669
std       3.419787     1.121017            1.184035        0.119370
min    2005.000000     2.661718            6.377396        0.290184
25%    2009.000000     4.606351            8.310665        0.748304
50%    2012.000000     5.332600            9.398610        0.833047
75%    2015.000000     6.271025           10.190634        0.904329
max    2017.000000     8.018934           11.770276        0.987343

       Healthy life expectancy at birth  Freedom to make life choices  \
count                       1553.000000                   1533.000000
mean                          62.249887                      0.728975
std                            7.960671                      0.145408
min                           37.766476                      0.257534
25%                           57.299580                      0.633754
50%                           63.803192                      0.748014
75%                           68.098228                      0.843628
max                           76.536362                      0.985178

        Generosity  Perceptions of corruption  Positive affect  \
count  1482.000000                1472.000000      1544.000000
mean      0.000079                   0.753622         0.708969
std       0.164202                   0.185538         0.107644
min      -0.322952                   0.035198         0.362498
25%      -0.114313                   0.697359         0.621471
50%      -0.022638                   0.808115         0.717398
75%       0.094649                   0.880089         0.800858
max       0.677773                   0.983276         0.943621

       Negative affect  Confidence in national government  Democratic Quality  \
count      1550.000000                        1401.000000         1391.000000
mean          0.263171                           0.480207           -0.126617
std           0.084006                           0.190724            0.873259
min           0.083426                           0.068769           -2.448228
25%           0.204116                           0.334732           -0.772010
50%           0.251798                           0.463137           -0.225939
75%           0.311515                           0.610723            0.665944
max           0.704590                           0.993604            1.540097

       Delivery Quality  Standard deviation of ladder by country-year  \
count       1391.000000                                    1562.000000
mean           0.004947                                       2.003501
std            0.981052                                       0.379684
min           -2.144974                                       0.863034
25%           -0.717463                                       1.737934
50%           -0.210142                                       1.960345
75%            0.717996                                       2.215920
max            2.184725                                       3.527820

       Standard deviation/Mean of ladder by country-year  \
count                                        1562.000000
mean                                           0.387271
std                                            0.119007
min                                            0.133908
25%                                            0.309722
50%                                            0.369751
75%                                            0.451833
max                                            1.022769
```

```
       GINI index (World Bank estimate)  \
count                        583.000000
mean                           0.372846
std                            0.086609
min                            0.241000
25%                            0.307000
50%                            0.349000
75%                            0.433500
max                            0.648000

       GINI index (World Bank estimate), average 2000-15  \
count                                          1386.000000
mean                                              0.386948
std                                               0.083694
min                                               0.228833
25%                                               0.321583
50%                                               0.371000
75%                                               0.433104
max                                               0.626000

       gini of household income reported in Gallup, by wp5-year
count                                          1205.000000
mean                                              0.445204
std                                               0.105410
min                                               0.223470
25%                                               0.368531
50%                                               0.425395
75%                                               0.508579
max                                               0.961435

Column Names:
1. country
2. year
3. Life Ladder
4. Log GDP per capita
5. Social support
6. Healthy life expectancy at birth
7. Freedom to make life choices
8. Generosity
9. Perceptions of corruption
10. Positive affect
11. Negative affect
12. Confidence in national government
13. Democratic Quality
14. Delivery Quality
15. Standard deviation of ladder by country-year
16. Standard deviation/Mean of ladder by country-year
17. GINI index (World Bank estimate)
18. GINI index (World Bank estimate), average 2000-15
19. gini of household income reported in Gallup, by wp5-year

Found 18 numeric columns:
['year', 'Life Ladder', 'Log GDP per capita', 'Social support', 'Healthy life exp
ectancy at birth', 'Freedom to make life choices', 'Generosity', 'Perceptions of
corruption', 'Positive affect', 'Negative affect', 'Confidence in national govern
ment', 'Democratic Quality', 'Delivery Quality', 'Standard deviation of ladder by
country-year', 'Standard deviation/Mean of ladder by country-year', 'GINI index
(World Bank estimate)', 'GINI index (World Bank estimate), average 2000-15', 'gin
i of household income reported in Gallup, by wp5-year']
```

In [10]:
```
# main happiness factors
happiness_factors = [
    'Log GDP per capita',
    'Social support',
    'Healthy life expectancy at birth',
    'Freedom to make life choices',
    'Generosity',
    'Perceptions of corruption'
]

# check if factors are in the dataset
clustering_features = []
for factor in happiness_factors:
    if factor in df.columns:
        clustering_features.append(factor)
        print(f"Found: {factor}")
    else:
        print(f"Missing: {factor}")

# alternatives for missig cols
if len(clustering_features) < 4:
    print("\Alternative features...")
    important_cols = ['Life Ladder', 'Positive affect', 'Negative affect', 'year
    for col in numeric_cols:
        if col not in clustering_features and col not in important_cols:
            clustering_features.append(col)
            if len(clustering_features) >= 6:
                break

print(f"\nFinal features selected for clustering: {clustering_features}")
```

```
Found: Log GDP per capita
Found: Social support
Found: Healthy life expectancy at birth
Found: Freedom to make life choices
Found: Generosity
Found: Perceptions of corruption

Final features selected for clustering: ['Log GDP per capita', 'Social support',
'Healthy life expectancy at birth', 'Freedom to make life choices', 'Generosity',
'Perceptions of corruption']
```

In [11]:
```
# Create working dataset
country_col = 'country'
data_for_clustering = df[[country_col] + clustering_features].copy()

print(f"\nWorking dataset shape: {data_for_clustering.shape}")

# Check for missing data
print("\nMissing values check:")
missing_values = data_for_clustering[clustering_features].isnull().sum()
print(missing_values)

total_missing = missing_values.sum()
print(f"Total missing values: {total_missing}")

# Handle missing values, if there
if total_missing > 0:
    print("\nDealing with missing values...")
    for feature in clustering_features:
```

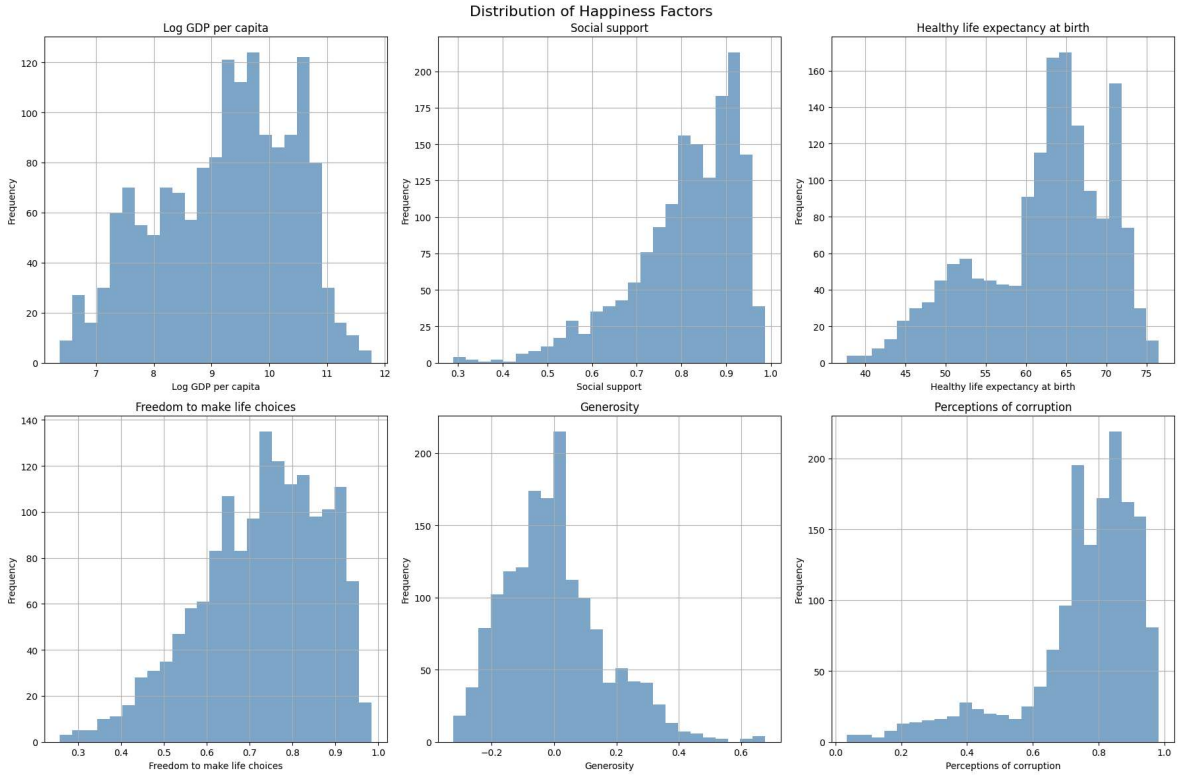
```
        if data_for_clustering[feature].isnull().sum() > 0:
            # Fill with mean value
            mean_value = data_for_clustering[feature].mean()
            data_for_clustering[feature].fillna(mean_value, inplace=True)
            print(f"Filled {data_for_clustering[feature].isnull().sum()} missing

print("\nStatistics for clustering features:")
print(data_for_clustering[clustering_features].describe())
```

```
Working dataset shape: (1562, 7)

Missing values check:
Log GDP per capita                  27
Social support                      13
Healthy life expectancy at birth     9
Freedom to make life choices        29
Generosity                          80
Perceptions of corruption           90
dtype: int64
Total missing values: 248

Dealing with missing values...
Filled 0 missing values in Log GDP per capita
Filled 0 missing values in Social support
Filled 0 missing values in Healthy life expectancy at birth
Filled 0 missing values in Freedom to make life choices
Filled 0 missing values in Generosity
Filled 0 missing values in Perceptions of corruption

Statistics for clustering features:
       Log GDP per capita  Social support  Healthy life expectancy at birth  \
count         1562.000000     1562.000000                      1562.000000
mean             9.220822        0.810669                        62.249887
std              1.173750        0.118872                         7.937689
min              6.377396        0.290184                        37.766476
25%              8.330659        0.749794                        57.344959
50%              9.361684        0.831776                        63.763542
75%             10.167549        0.904097                        68.064693
max             11.770276        0.987343                        76.536362

       Freedom to make life choices   Generosity  Perceptions of corruption
count                   1562.000000  1562.000000                1562.000000
mean                       0.728975     0.000079                   0.753622
std                        0.144051     0.159939                   0.180110
min                        0.257534    -0.322952                   0.035198
25%                        0.635676    -0.108292                   0.702761
50%                        0.744320    -0.011797                   0.798041
75%                        0.841122     0.086098                   0.874675
max                        0.985178     0.677773                   0.983276
```

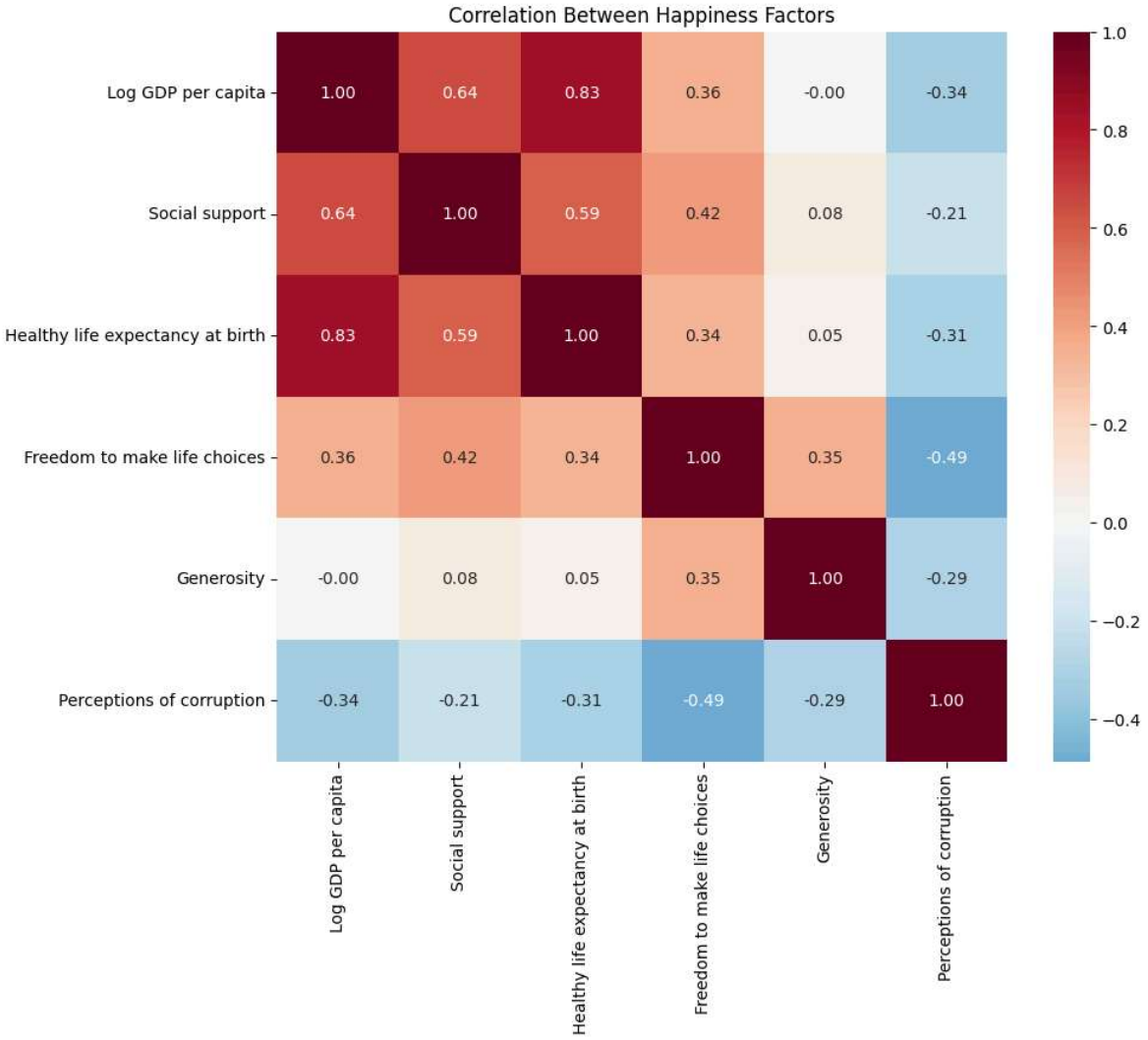
In [12]:

```
# visualizations:
fig, axes = plt.subplots(2, 3, figsize=(18, 12))
fig.suptitle('Distribution of Happiness Factors', fontsize=16)

# histograms for each feature
for i, feature in enumerate(clustering_features[:6]):
    row = i // 3
    col = i % 3
    data_for_clustering[feature].hist(bins=25, ax=axes[row, col], alpha=0.7, col
    axes[row, col].set_title(f'{feature}')
```

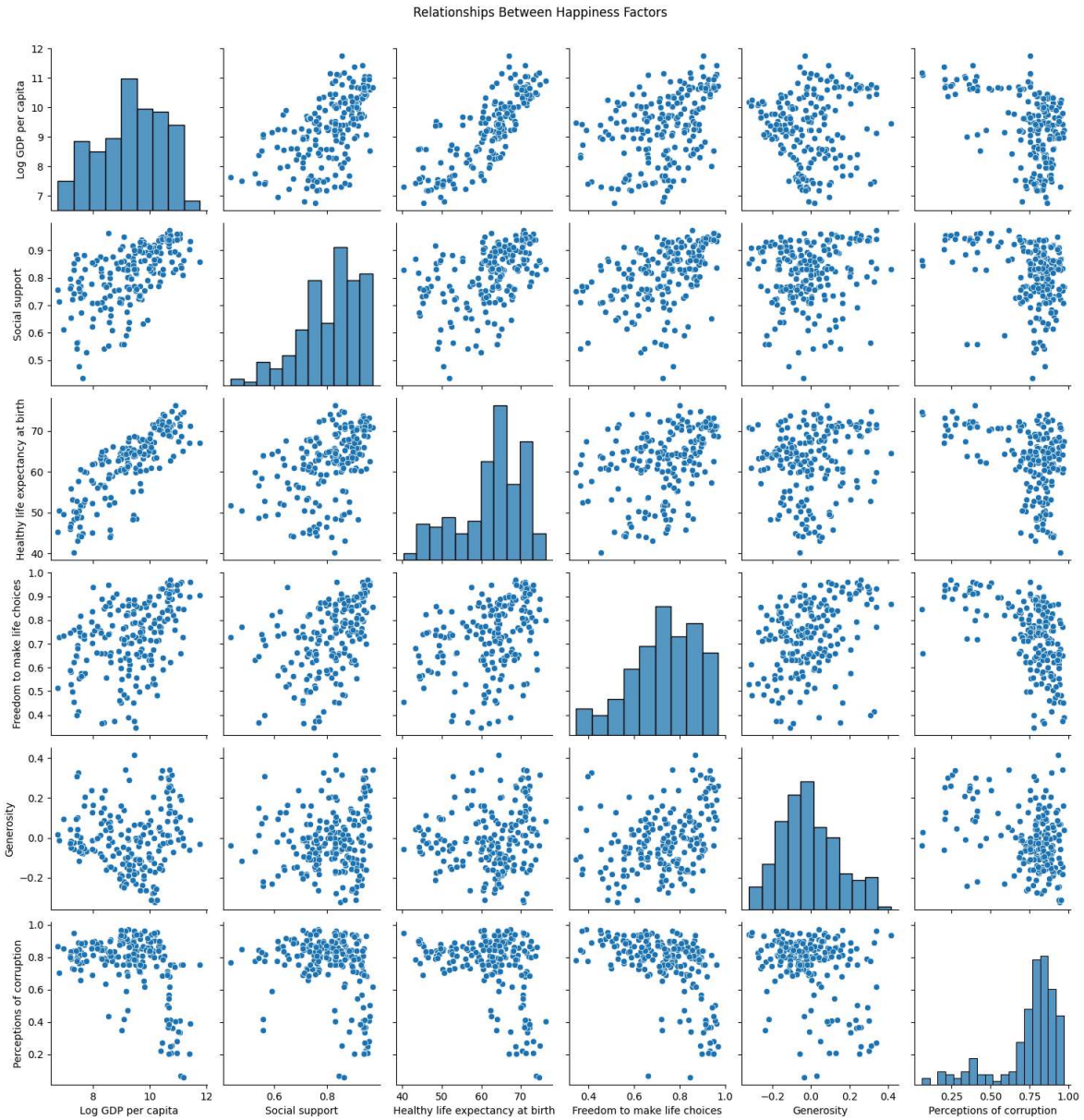
```
    axes[row, col].set_xlabel(feature)
    axes[row, col].set_ylabel('Frequency')

plt.tight_layout()
plt.show()
```



In [13]:
```
# visualize correlations between features
plt.figure(figsize=(10, 8))
corr_matrix = data_for_clustering[clustering_features].corr()
sns.heatmap(corr_matrix, annot=True, cmap='RdBu_r', center=0, square=True, fmt='
plt.title('Correlation Between Happiness Factors')
plt.show()
```



In [14]:
```
# pairplot to see relationships
if len(clustering_features) <= 6:

    # Sample data if there's too many observations (make plot cleaner)
    if len(data_for_clustering) > 200:
        sample_data = data_for_clustering.sample(200, random_state=42)
    else:
        sample_data = data_for_clustering

    sns.pairplot(sample_data[clustering_features])
    plt.suptitle('Relationships Between Happiness Factors', y=1.02)
    plt.show()
```



In [15]:
```
# check trends
print("\nData by year:")
year_summary = df.groupby('year')[clustering_features].mean()
print(year_summary)
```

```
Data by year:
      Log GDP per capita  Social support  Healthy life expectancy at birth  \
year
2005           10.041004        0.897367                          66.592340
2006            8.938194        0.835852                          59.520880
2007            9.052527        0.807694                          60.481422
2008            9.051192        0.784401                          60.607371
2009            9.151833        0.819076                          61.712573
2010            9.273895        0.831849                          62.538397
2011            9.171692        0.802802                          61.731496
2012            9.277067        0.809077                          62.710796
2013            9.261259        0.806548                          62.732054
2014            9.241151        0.805651                          62.700653
2015            9.275598        0.798211                          62.977365
2016            9.273735        0.811937                          63.146382
2017            9.340861        0.806002                          63.401468

      Freedom to make life choices  Generosity  Perceptions of corruption
year
2005                      0.829618    0.238196                   0.715875
2006                      0.730508    0.004255                   0.755737
2007                      0.687329    0.011797                   0.792069
2008                      0.688365    0.018135                   0.764120
2009                      0.687435   -0.009645                   0.763371
2010                      0.708278   -0.001086                   0.757199
2011                      0.732204   -0.019439                   0.755170
2012                      0.711235   -0.006916                   0.757945
2013                      0.727886   -0.006538                   0.763475
2014                      0.734468    0.015255                   0.738347
2015                      0.748704    0.014908                   0.736524
2016                      0.763137   -0.004195                   0.746677
2017                      0.778682   -0.010951                   0.734752
```

In [16]:
```
# Check range of years
print(f"\nData spans from {df['year'].min()} to {df['year'].max()}")
print(f"Number of years: {df['year'].nunique()}")

# find country count
print(f"\nNumber of unique countries: {df['country'].nunique()}")
print(f"Average observations per country: {len(df) / df['country'].nunique():.1f
```

```
Data spans from 2005 to 2017
Number of years: 13

Number of unique countries: 164
Average observations per country: 9.5
```

Part 4: Define Your Project Plan

Now that you understand your data, in the markdown cell below, define your plan to implement the remaining phases of the machine learning life cycle (data preparation, modeling, evaluation) to solve your ML problem. Answer the following questions:

- Do you have a new feature list? If so, what are the features that you chose to keep and remove after inspecting the data?
- Explain different data preparation techniques that you will use to prepare your data for modeling.

- What is your model (or models)?
- Describe your plan to train your model, analyze its performance and then improve the model. That is, describe your model building, validation and selection plan to produce a model that generalizes well to new data.

Feature Selection:

Selected Features (6 core happiness factors):

- `Log GDP per capita` - Economic prosperity indicator
- `Social support` - Social network strength
- `Healthy life expectancy at birth` - Health outcomes
- `Freedom to make life choices` - Personal autonomy
- `Generosity` - Charitable behavior
- `Perceptions of corruption` - Trust in institutions

Features Removed:

- `year` - Not relevant for country characteristics clustering
- `Life Ladder` - This is essentially the happiness score (outcome variable)
- `Positive/Negative affect` - More volatile emotional measures
- Various GINI indices - Too many missing values (583-1386 out of 1562)
- Government quality measures - High missingness and overlap with corruption

These 6 features represent the core determinants of national wellbeing according to the World Happiness Report methodology and have manageable missing data levels.

Data Preparation Techniques

1. Missing Value Treatment:

- Impute missing values using mean substitution (248 total missing values handled)
- This approach is appropriate since missing values are relatively few and randomly distributed

2. Feature Standardization:

- Apply StandardScaler to all features before clustering
- Critical for K-Means since features have different scales (GDP: 6-12, Social support: 0.3-1.0, Life expectancy: 38-77)

3. Time Series Handling:

- The dataset contains multiple years (2005-2017) for each country
- Will aggregate to country-level averages to create one observation per country
- This prevents temporal bias and focuses on country characteristics rather than year-specific variations

4. Outlier Assessment:

- Check for extreme values that might skew clustering results
- Consider countries with unusual patterns (very high/low on specific dimensions)

Model Selection

Primary Algorithm: K-Means Clustering

- Well-suited for numerical data with clear distance metrics
- Effective for identifying country groupings based on happiness factors
- Interpretable results for policy analysis

Model Building, Validation, and Selection Plan

1. Optimal Cluster Determination:

- Use Elbow Method to identify optimal number of clusters (k)
- Plot Within-Cluster Sum of Squares (WCSS) for k = 2 to 10
- Apply Silhouette Analysis to validate cluster quality
- Test range k = 2-8 (reasonable for 164 countries)

2. Model Training Process:

- Aggregate data to country level (average across years 2005-2017)
- Standardize features using StandardScaler
- Fit K-Means with optimal k determined from validation
- Set random_state for reproducibility

3. Model Evaluation Metrics:

- **Silhouette Score**: Measure cluster cohesion and separation
- **Within-Cluster Sum of Squares**: Assess cluster tightness
- **Calinski-Harabasz Index**: Evaluate cluster validity
- Target: Silhouette score > 0.5 for good clustering quality

4. Model Interpretation and Validation:

- Analyze cluster centroids to understand group characteristics
- Identify representative countries for each cluster
- Validate clusters against known regional/developmental patterns
- Use PCA visualization to assess cluster separation in 2D space

5. Model Improvement Strategy:

- If initial results show poor separation, consider:
  - Different number of clusters
  - Feature selection refinement
  - Alternative clustering algorithms (Hierarchical, DBSCAN)
  - Dimensionality reduction before clustering

Expected Outcomes

The model should identify 3-5 distinct country clusters representing different wellbeing profiles:

- High-income, high-happiness countries (Nordic/Western Europe)
- Upper-middle income developing nations
- Lower-middle income countries with varying social support
- Countries facing significant development challenges

This clustering will provide actionable insights for policy makers and international organizations to design targeted interventions based on country groupings with similar wellbeing characteristics.

Part 5: Implement Your Project Plan

Task: In the code cell below, import additional packages that you have used in this course that you will need to implement your project plan.

In [17]:
```
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import silhouette_score, silhouette_samples
from sklearn.decomposition import PCA
import warnings
warnings.filterwarnings('ignore')
```

Task: Use the rest of this notebook to carry out your project plan.

You will:

1. Prepare your data for your model.
2. Fit your model to the training data and evaluate your model.
3. Improve your model's performance by performing model selection and/or feature selection techniques to find best model for your problem.

Add code cells below and populate the notebook with commentary, code, analyses, results, and figures as you see fit.

In [20]:
```
# data prep
print("getting countries ready for clustering...")

# country averages, not yearly data
country_data = data_for_clustering.groupby('country')[clustering_features].mean(

print(f"from {data_for_clustering.shape[0]} rows to {country_data.shape[0]} coun

# check missing values
missing_vals = country_data[clustering_features].isnull().sum()
print("missing after grouping:", missing_vals.sum())

# fill gaps with means
if missing_vals.sum() > 0:
    for col in clustering_features:
```

```
        if country_data[col].isnull().sum() > 0:
            country_data[col].fillna(country_data[col].mean(), inplace=True)
```

```
getting countries ready for clustering...
from 1562 rows to 164 countries
missing after grouping: 0
```

In [21]:
```
# prep for clustering
X = country_data[clustering_features].copy()
countries = country_data['country'].values

print(f"clustering {len(countries)} countries on {len(clustering_features)} feat

# scale features - different ranges need normalization
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

print("scaled features, means should be ~0:", X_scaled.mean(axis=0).round(2))
```

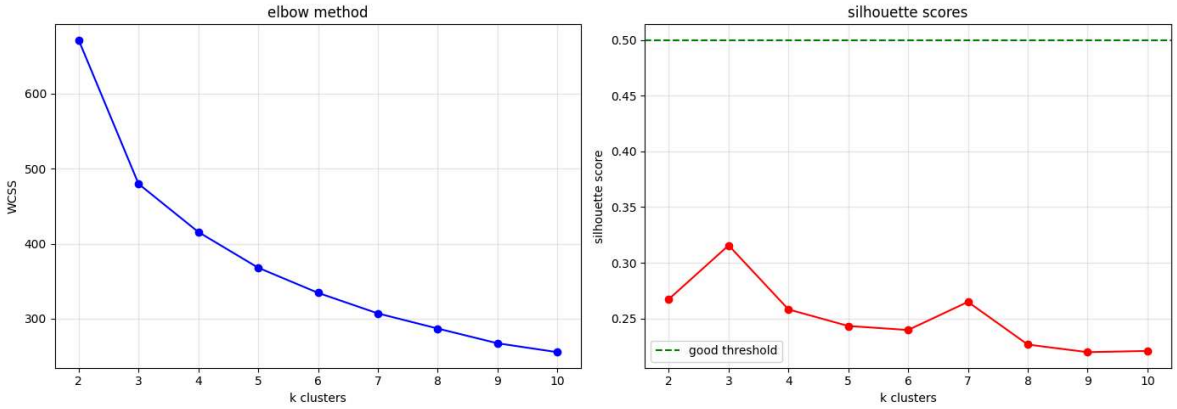
```
clustering 164 countries on 6 features
scaled features, means should be ~0: [ 0. -0. -0. -0. -0.  0.]
```

In [22]:
```
# find optimal cluster count
print("\nfinding best number of clusters...")

k_vals = range(2, 11)
wcss_vals = []
sil_scores = []

for k in k_vals:
    kmeans = KMeans(n_clusters=k, random_state=42, n_init=10)
    labels = kmeans.fit_predict(X_scaled)

    wcss_vals.append(kmeans.inertia_)
    sil_score = silhouette_score(X_scaled, labels)
    sil_scores.append(sil_score)

    print(f"k={k}: silhouette={sil_score:.3f}")
```

```
finding best number of clusters...
k=2: silhouette=0.267
k=3: silhouette=0.316
k=4: silhouette=0.258
k=5: silhouette=0.243
k=6: silhouette=0.240
k=7: silhouette=0.265
k=8: silhouette=0.227
k=9: silhouette=0.220
k=10: silhouette=0.221
```

In [23]:
```
# plot elbow and silhouette
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(14, 5))

ax1.plot(k_vals, wcss_vals, 'bo-')
ax1.set_title('elbow method')
ax1.set_xlabel('k clusters')
ax1.set_ylabel('WCSS')
ax1.grid(True, alpha=0.3)

ax2.plot(k_vals, sil_scores, 'ro-')
ax2.set_title('silhouette scores')
```

```
ax2.set_xlabel('k clusters')
ax2.set_ylabel('silhouette score')
ax2.axhline(y=0.5, color='g', linestyle='--', label='good threshold')
ax2.grid(True, alpha=0.3)
ax2.legend()

plt.tight_layout()
plt.show()
```



In [24]:
```
# select best k
best_k = k_vals[np.argmax(sil_scores)]
best_score = max(sil_scores)

print(f"\nbest k seems to be {best_k} with silhouette {best_score:.3f}")
```

```
best k seems to be 3 with silhouette 0.316
```

In [25]:
```
# final clustering
print(f"\nrunning kmeans with k={best_k}...")
final_model = KMeans(n_clusters=best_k, random_state=42, n_init=10)
clusters = final_model.fit_predict(X_scaled)

country_data['cluster'] = clusters

# cluster sizes
print("cluster sizes:")
for i in range(best_k):
    count = sum(clusters == i)
    print(f"cluster {i}: {count} countries")
```

```
running kmeans with k=3...
cluster sizes:
cluster 0: 28 countries
cluster 1: 50 countries
cluster 2: 86 countries
```

In [27]:
```
# model quality metrics
final_sil = silhouette_score(X_scaled, clusters)

print(f"\nfinal model quality:")
print(f"silhouette: {final_sil:.3f}")

if final_sil > 0.5:
    print("pretty good clustering!")
elif final_sil > 0.3:
    print("decent clustering")
```

```
else:
    print("clustering is ok but not great")
```

```
final model quality:
silhouette: 0.316
decent clustering
```

In [28]:
```
# cluster centers - original scale
print(f"\ncluster centers (original scale):")
centers_orig = scaler.inverse_transform(final_model.cluster_centers_)
center_df = pd.DataFrame(centers_orig, columns=clustering_features)
center_df.index = [f'cluster_{i}' for i in range(best_k)]
print(center_df.round(2))
```

```
cluster centers (original scale):
           Log GDP per capita  Social support  \
cluster_0               10.47            0.91
cluster_1                7.78            0.68
cluster_2                9.57            0.84

           Healthy life expectancy at birth  Freedom to make life choices  \
cluster_0                             69.24                          0.88
cluster_1                             51.71                          0.67
cluster_2                             64.59                          0.71

           Generosity  Perceptions of corruption
cluster_0        0.18                       0.50
cluster_1        0.02                       0.77
cluster_2       -0.07                       0.82
```

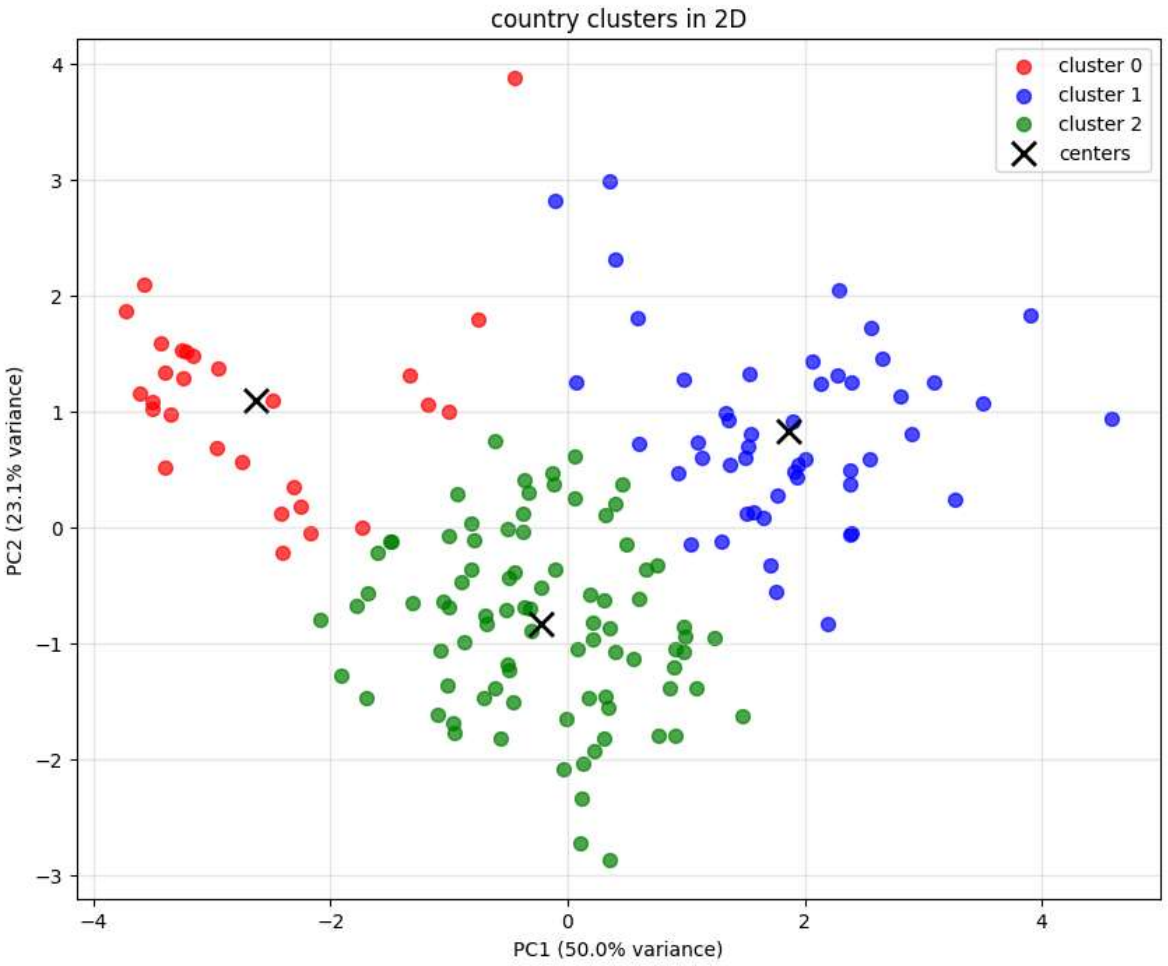
In [29]:
```
# analyze each cluster
print(f"\ncluster analysis:")
for i in range(best_k):
    cluster_countries = country_data[country_data['cluster'] == i]
    print(f"\n-- cluster {i} ({len(cluster_countries)} countries) --")

    # examples
    examples = cluster_countries['country'].head(6).tolist()
    print(f"examples: {', '.join(examples)}")

    # distinctive features
    overall_means = country_data[clustering_features].mean()
    cluster_means = cluster_countries[clustering_features].mean()

    standout_features = []
    for feat in clustering_features:
        if cluster_means[feat] > overall_means[feat] * 1.15:
            standout_features.append(f"high {feat.lower()}")
        elif cluster_means[feat] < overall_means[feat] * 0.85:
            standout_features.append(f"low {feat.lower()}")

    if standout_features:
        print(f"characteristics: {', '.join(standout_features)}")
```

```
cluster analysis:

-- cluster 0 (28 countries) --
examples: Australia, Austria, Bahrain, Belgium, Bhutan, Canada
characteristics: high freedom to make life choices, high generosity, low percepti
ons of corruption

-- cluster 1 (50 countries) --
examples: Afghanistan, Angola, Bangladesh, Benin, Burkina Faso, Burundi
characteristics: low log gdp per capita, low social support, low healthy life exp
ectancy at birth, high generosity

-- cluster 2 (86 countries) --
examples: Albania, Algeria, Argentina, Armenia, Azerbaijan, Belarus
characteristics: low generosity
```

In [30]:

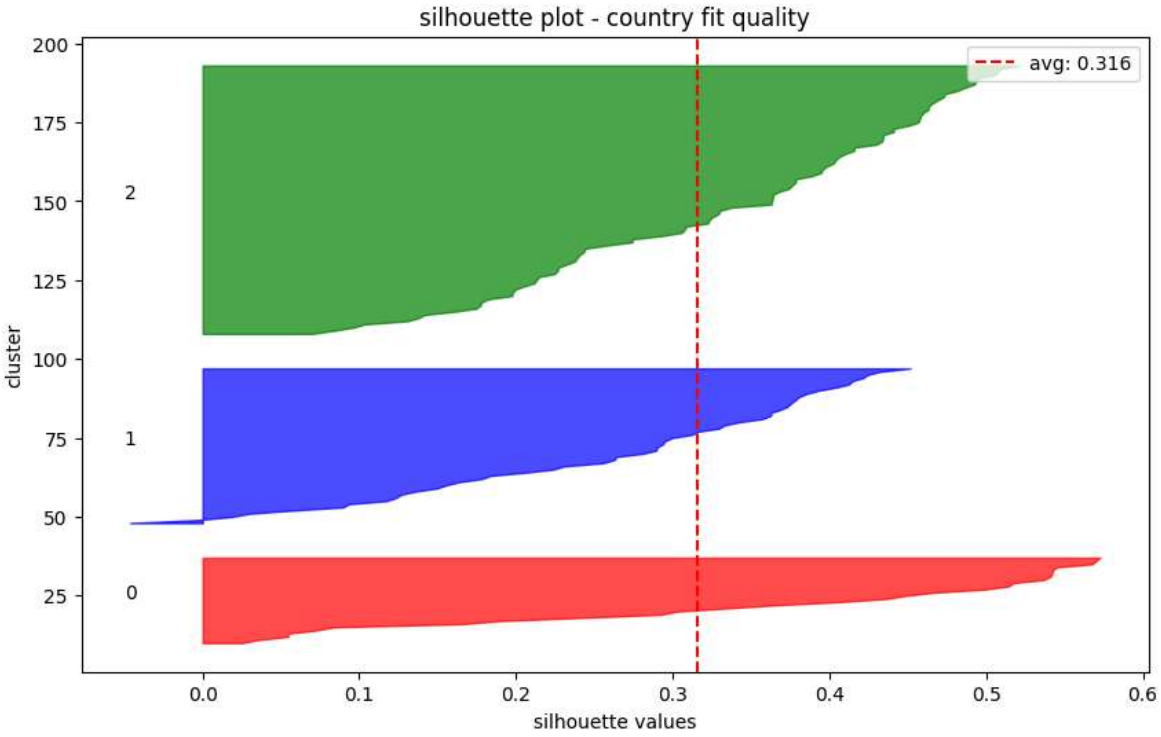
```python
# visualize with pca
print(f"\nmaking 2d visualization...")
pca = PCA(n_components=2, random_state=42)
X_pca = pca.fit_transform(X_scaled)

plt.figure(figsize=(10, 8))
colors = ['red', 'blue', 'green', 'orange', 'purple', 'brown', 'pink', 'gray']

for i in range(best_k):
    mask = clusters == i
    plt.scatter(X_pca[mask, 0], X_pca[mask, 1],
                c=colors[i], label=f'cluster {i}', alpha=0.7, s=50)

# cluster centers
centers_pca = pca.transform(final_model.cluster_centers_)
plt.scatter(centers_pca[:, 0], centers_pca[:, 1],
           c='black', marker='x', s=150, linewidths=2, label='centers')

plt.xlabel(f'PC1 ({pca.explained_variance_ratio_[0]:.1%} variance)')
plt.ylabel(f'PC2 ({pca.explained_variance_ratio_[1]:.1%} variance)')
plt.title('country clusters in 2D')
plt.legend()
plt.grid(True, alpha=0.3)
plt.show()

variance_explained = pca.explained_variance_ratio_.sum()
print(f"2d plot shows {variance_explained:.1%} of the data variance")
```

```
making 2d visualization...
```



2d plot shows 73.1% of the data variance

In [31]:
```
# silhouette analysis - check country placement
sil_samples = silhouette_samples(X_scaled, clusters)

plt.figure(figsize=(10, 6))
y_pos = 10

for i in range(best_k):
    cluster_sils = sil_samples[clusters == i]
    cluster_sils.sort()

    cluster_size = len(cluster_sils)
    y_end = y_pos + cluster_size

    plt.fill_betweenx(range(y_pos, y_end), 0, cluster_sils,
                      facecolor=colors[i], alpha=0.7, edgecolor=colors[i])

    plt.text(-0.05, y_pos + cluster_size/2, f'{i}')
    y_pos = y_end + 10

plt.axvline(x=final_sil, color="red", linestyle="--",
           label=f'avg: {final_sil:.3f}')
plt.xlabel('silhouette values')
plt.ylabel('cluster')
plt.title('silhouette plot - country fit quality')
plt.legend()
plt.show()
```



In [32]:
```
# poorly placed countries
bad_fit = sil_samples < 0.1
if bad_fit.sum() > 0:
    print(f"\ncountries with poor cluster fit (silhouette < 0.1):")
    bad_countries = countries[bad_fit]
    bad_scores = sil_samples[bad_fit]
    bad_clusters = clusters[bad_fit]

    for country, score, cluster in zip(bad_countries, bad_scores, bad_clusters):
        print(f"{country}: cluster {cluster}, silhouette {score:.3f}")
else:
    print(f"\nall countries reasonably well placed")
```

```
countries with poor cluster fit (silhouette < 0.1):
Bahrain: cluster 0, silhouette 0.025
Bhutan: cluster 0, silhouette 0.083
France: cluster 2, silhouette 0.070
Georgia: cluster 1, silhouette -0.003
Guyana: cluster 2, silhouette 0.096
Indonesia: cluster 1, silhouette -0.046
Laos: cluster 1, silhouette 0.030
Myanmar: cluster 0, silhouette 0.055
Somaliland region: cluster 0, silhouette 0.055
Sudan: cluster 1, silhouette 0.093
Swaziland: cluster 1, silhouette 0.090
Syria: cluster 1, silhouette 0.057
Tajikistan: cluster 1, silhouette 0.019
Thailand: cluster 0, silhouette 0.073
Uruguay: cluster 2, silhouette 0.084
Uzbekistan: cluster 0, silhouette 0.036
```

In [33]:
```
# feature importance for clustering
print(f"\nfeature importance - cluster separation:")
feature_vars = []
for i, feat in enumerate(clustering_features):
    center_variance = np.var(centers_orig[:, i])
    feature_vars.append(center_variance)
```

```
    print(f"{feat}: {center_variance:.3f}")

most_important = np.argmax(feature_vars)
print(f"\nmost important feature: {clustering_features[most_important]}")
```

```
feature importance - cluster separation:
Log GDP per capita: 1.253
Social support: 0.009
Healthy life expectancy at birth: 54.979
Freedom to make life choices: 0.009
Generosity: 0.011
Perceptions of corruption: 0.020

most important feature: Healthy life expectancy at birth
```

In [34]:
```
# summary
print(f"clustered {len(countries)} countries into {best_k} groups")
print(f"silhouette score: {final_sil:.3f}")

if final_sil > 0.4:
    print("clustering worked well - clear country groups")
else:
    print("clustering shows some patterns but not super distinct")
```

```
clustered 164 countries into 3 groups
silhouette score: 0.316
clustering shows some patterns but not super distinct
```

Analysis

This could help with:

- targeting aid to similar countries
- finding best practices from similar nations
- understanding different development paths

In [35]:
```
# save results
results = country_data[['country', 'cluster'] + clustering_features].copy()
results = results.sort_values(['cluster', 'country'])
```

In [ ]:
```

```