ROBERT GORDON UNIVERSITY ABERDEEN

INFORMATICS INSTITUTE OF TECHNOLOGY

Academic Year	2024/25
Semester	Two
Module Number	CMM704
Module Title	Data Mining
Module Co-Ordinator	Mr. Nipuna Senanayake
IIT Student Number	20231720
RGU ID	2330844
Student Name	Nithin Kolamunna

Table of Contents

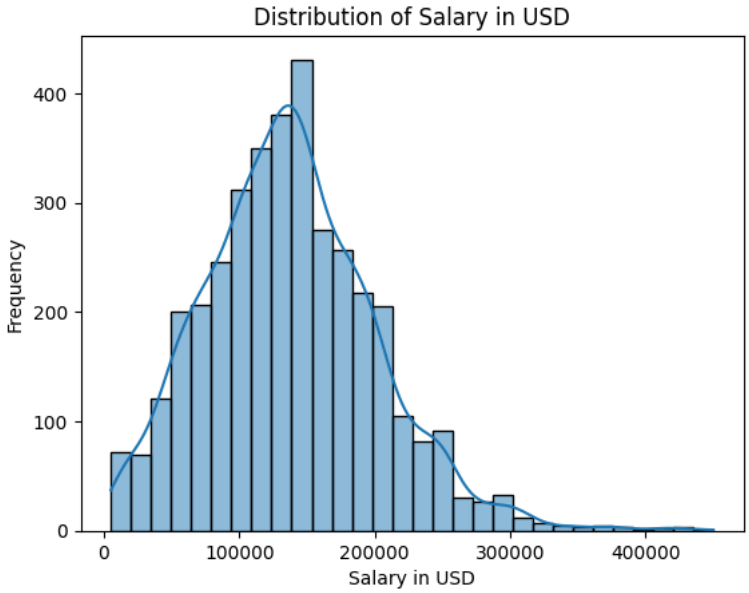
Question 01: Regression

1.1. Perform EDA & Identify Required Features

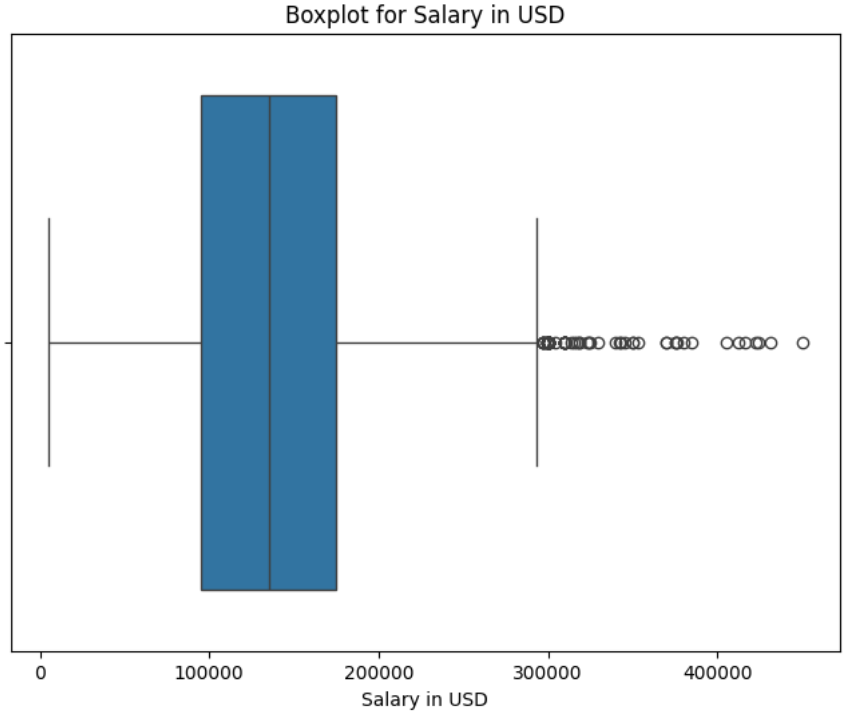
What was done:

- Previewed dataset structure (head(), info())
- Identified data types and values for each column.
- Plotted:
    - Missing value barplot

    - Histogram of *salary in usd*



- Boxplot for detecting outliers



Key Observation:

- Few columns had empty strings, and these are converted to NaNs
- *salary_in_usd* showed a right-skewed distribution.
- Before removing features, it's better to run a correlation heatmap analysis so we can identify and drop the least correlated variables

Justification:

- Understanding the basic structure and distributions is essential before modeling.

1.2. Missing value imputation

What was done:

- Filled in missing **categorical** features using the **mode**.
- Filled in missing **numerical** values (if any) using the **median**.

Justification:

- Mode and median imputation are effective methods that prevent distortion caused by skewed data or outliers.

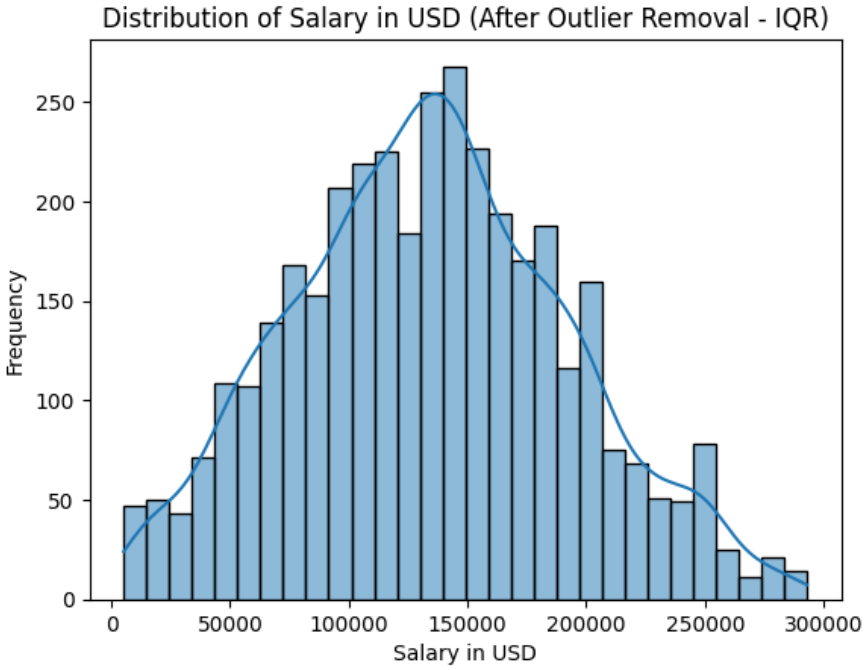
1.3. Outlier detection and removal

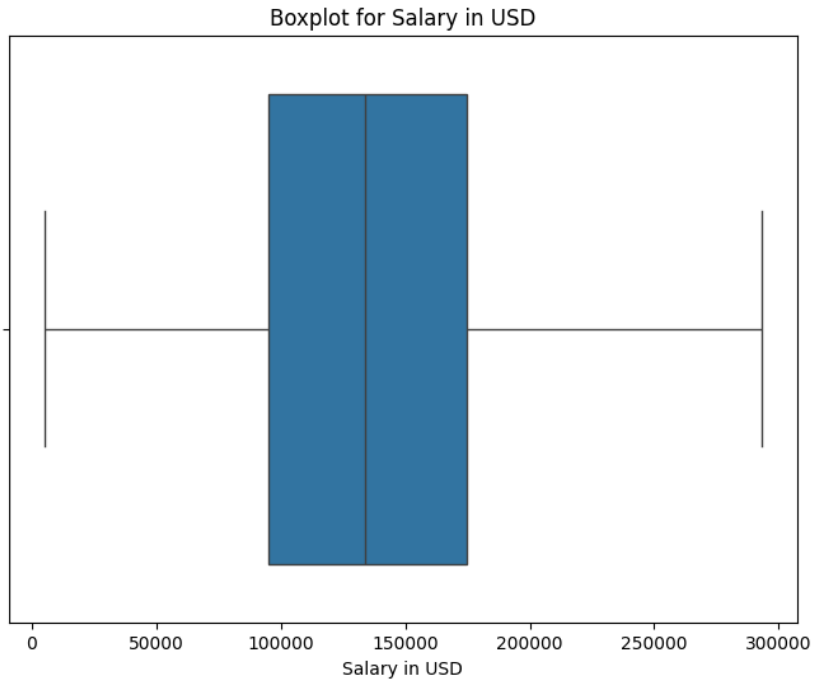
What was done:

- Applied the Interquartile Range (IQR) method to detect and remove outliers from *salary_in_usd*.

Result:

- Minimized the effects of high-value salary outliers.
- Improved model performance resulting from reduced distortions.





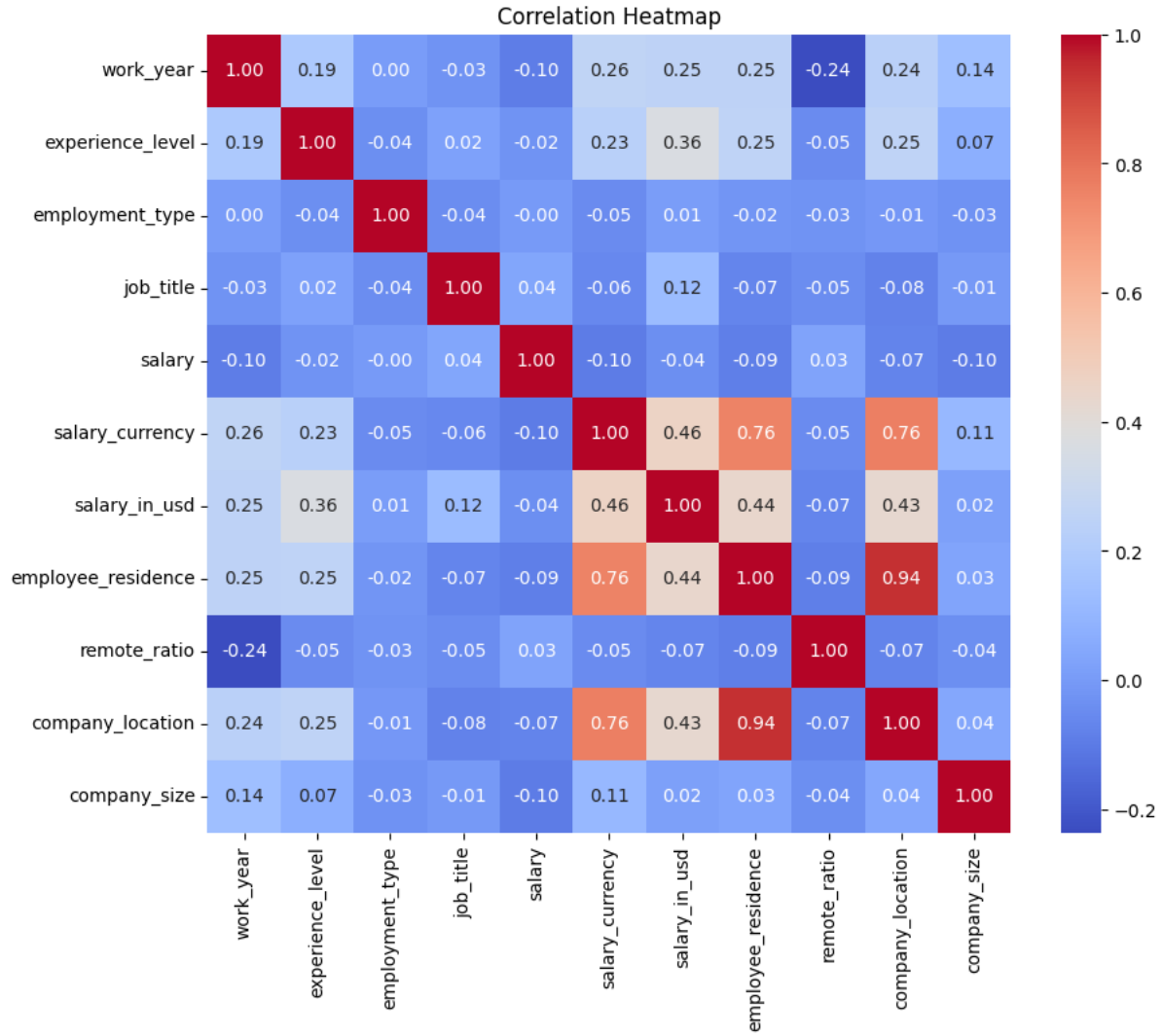
1.4. Feature selection/extraction

What was done:

- I performed a correlation analysis to identify and filter the relevant features.
- I used label encoding on the categorical features to ensure compatibility with the model.
- Dropped: *salary, remote_ratio, employment_type, company_size*

Justification:

- Reduced dimensionality.
- Ensured that only significant features were used in the models.




```
Correlation with Target (salary_in_usd):
salary_in_usd          1.000000
salary_currency        0.459339
employee_residence     0.436646
company_location       0.426759
experience_level       0.364036
work_year              0.250661
job_title              0.123619
company_size           0.016895
employment_type        0.006937
salary                -0.039625
remote_ratio          -0.069823
```

1.5. Normalization

What was done:

- Used StandardScaler to normalize the feature space.

Justification:

- It is important to ensure that all features are given equal weight, particularly when working with linear models.
- Enhances the convergence of the model and makes it easier to interpret.

1.6. Building two models

Models Built:

- Linear Regression
- Random Forest Regressor
- (Also tested XGBoost Regressor)

Why Chosen:

- Linear regression provides a fundamental approach for making continuous predictions.
- Linear regression understands linear relationships between features and *salary_in_usd*.
- Random Forest effectively manages non-linear relationships and interactions between features.
- Capturing complex patterns and handling missing data and outliers is better than linear regression.

1.7. Evaluation

Metrics Used:

- MAE, MSE, RMSE, and R^2 Score
- Results Before Tuning:

Model	MAE	MSE	RMSE	R^2 Score
Linear Reg.	39,272.19	2.45M	49,547.64	0.23
Random Forest	33,876.17	1.84M	42,940.75	0.46
XGBoost	34,254.74	1.89M	43,565.32	0.45

Conclusion:

- **Random Forest** gave the best performance.
- **Linear regression** did not perform well because it assumes a linear relationship between variables.
- **XGBoost** was close, showing potential with proper tuning.

Hyperparameter Tuning for Random Forest

What was done:

- Implemented GridSearchCV to perform hyperparameter tuning on the Random Forest Regression.

Model	MAE	MSE	RMSE	R^2 Score
Random Forest (Tuned)	33,828.78	1.83M	42,812.62	**0.47**
Random Forest	33,876.17	1.84M	42,940.75	0.46

Observations:

- After tuning, **Random Forest** improved its R^2 score from **0.46 to 0.47.**
- The Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) decreased, indicating that the model now predicts salaries with greater accuracy.
- This confirms that hyperparameter optimization effectively fine-tunes model complexity and enhances generalization ability.

Justification:

- Random Forest has several internal parameters that influence tree behavior. Without proper tuning, the default values may not be suitable for a specific dataset.

Question 02: Clustering

2.1. Perform EDA & Identify Required Features

Activities:

- Loaded 2015.csv and 2019.csv datasets.
- I examined the data types, shapes, and null values, and provided statistical summaries.
- Removed unwanted columns
  2019 : ['Overall rank',' Country or region']
  2015 : ['Happiness Rank','Region','Country','Standard Error','Dystopia Residual']
  *Country also removed due to unique column

Findings:

- Datasets had minor structural differences in column names and counts.
- No significant missing values

```
Missing Values in 2015 Dataset:
Economy (GDP per Capita)         0
Family                           0
Freedom                          0
Generosity                       0
Happiness Score                  0
Health (Life Expectancy)         0
Trust (Government Corruption)    0
dtype: int64
```

```
Missing Values in 2019 Dataset:
Freedom to make life choices    0
GDP per capita                  0
Generosity                      0
Healthy life expectancy         0
Perceptions of corruption       0
Score                           0
Social support                  0
dtype: int64
```

2.2. Outlier detection and removal

Techniques Used:

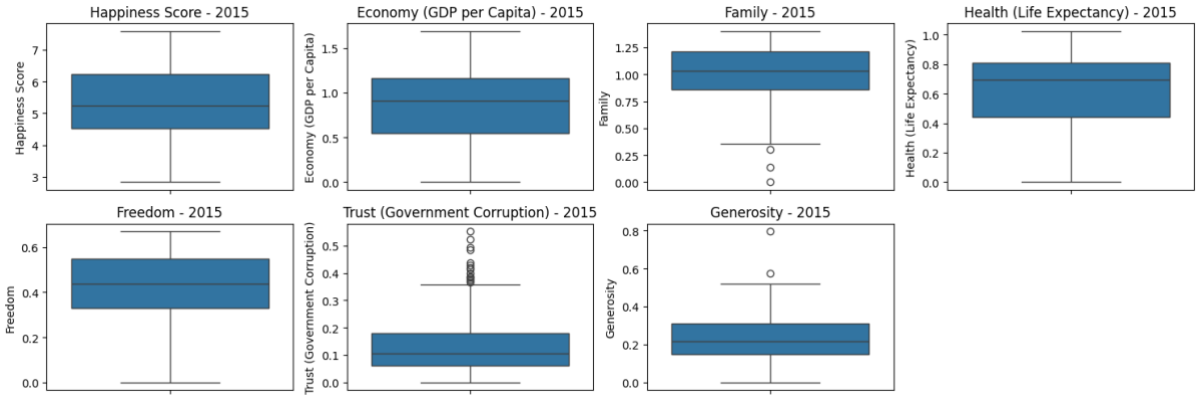
- Visualized outliers using boxplots.
- Implemented both IQR and Z-score methods.

Resulting Dataset Sizes:

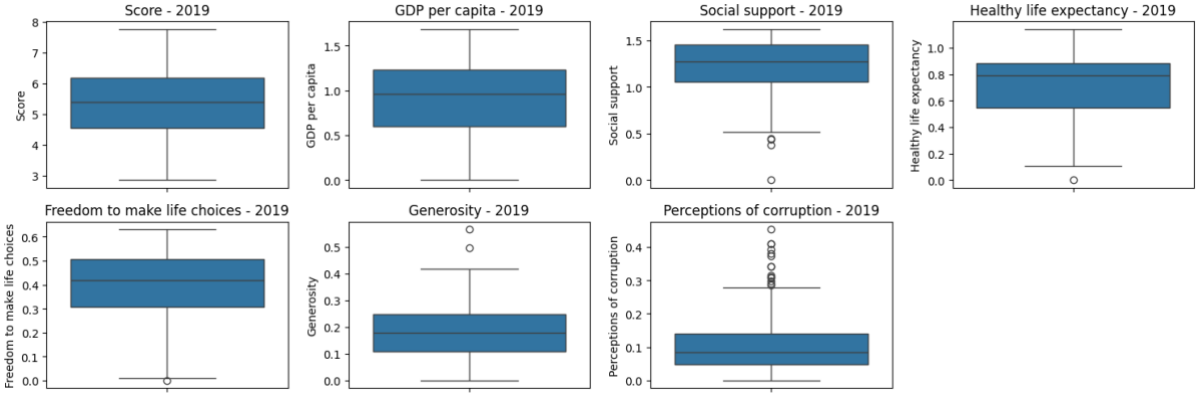
Year	Source Data	After IQR	After Z-Score
2015	158	136	153
2019	156	133	149

```
2015 Data: (158, 7) → (136, 7) after removing outliers (iqr)
2019 Data: (156, 7) → (133, 7) after removing outliers (iqr)
2015 Data: (158, 7) → (153, 7) after removing outliers (z-score)
2019 Data: (156, 7) → (149, 7) after removing outliers (z-score)
```

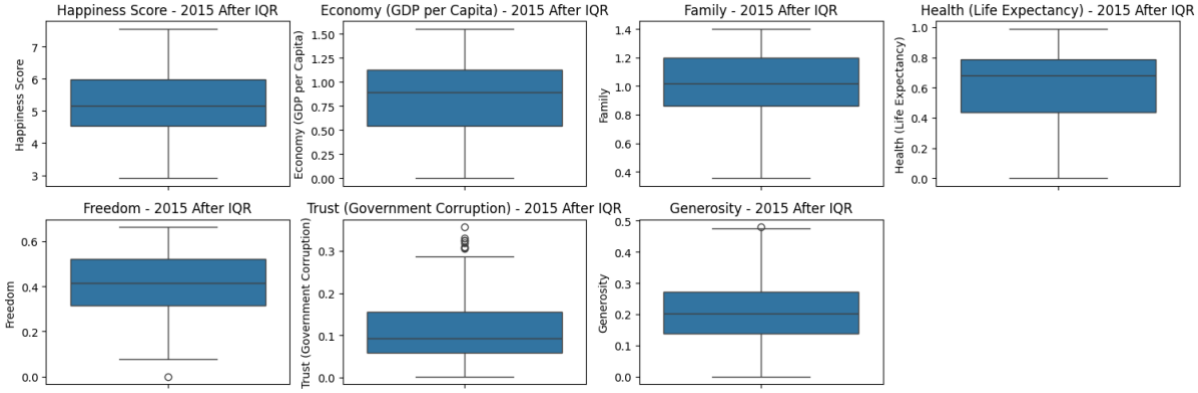
Visualizing Outliers in 2015 Data



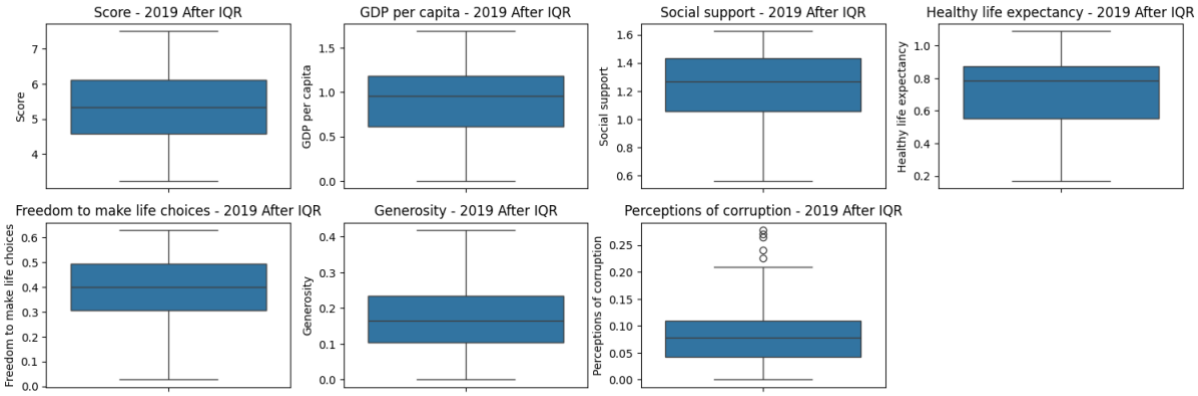
Visualizing Outliers in 2019 Data



Data After Outlier Removal (IQR) – 2015

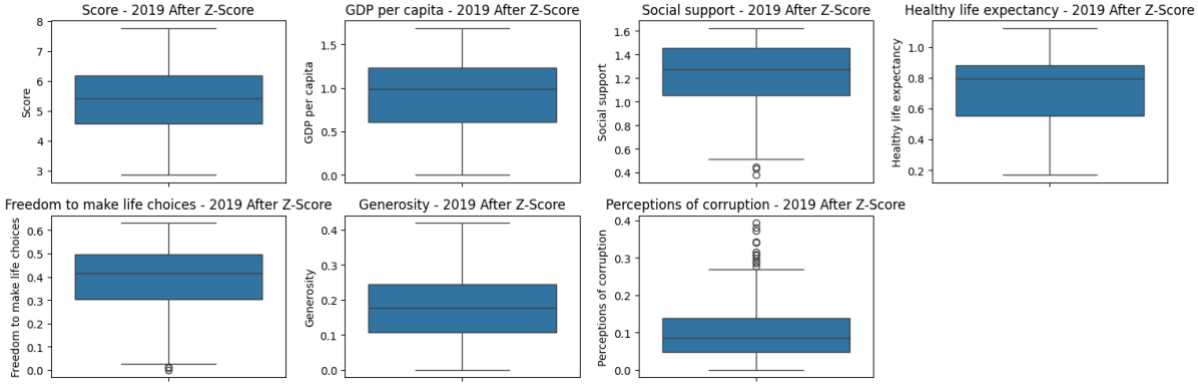


Data After Outlier Removal (IQR) – 2019



Data After Outlier Removal (Z-Score) - 2015

Data After Outlier Removal (Z-Score) - 2019



Decision:

- The Z-score method was selected due to better balance (fewer data points removed).
- Resulted in clean datasets:
  - 2015: 153 rows
  - 2019: 149 rows

2.3. Feature scaling and transformation

Steps Taken:

- Select the columns for clustering.
- Applied StandardScaler to normalize data.

Justification:

- K-means++ relies on distance metrics; scaling ensures fair treatment of all features.

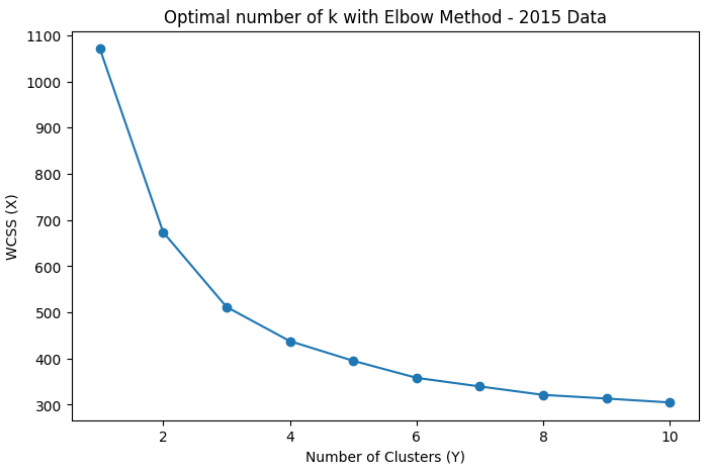
2.4. Clustering model development

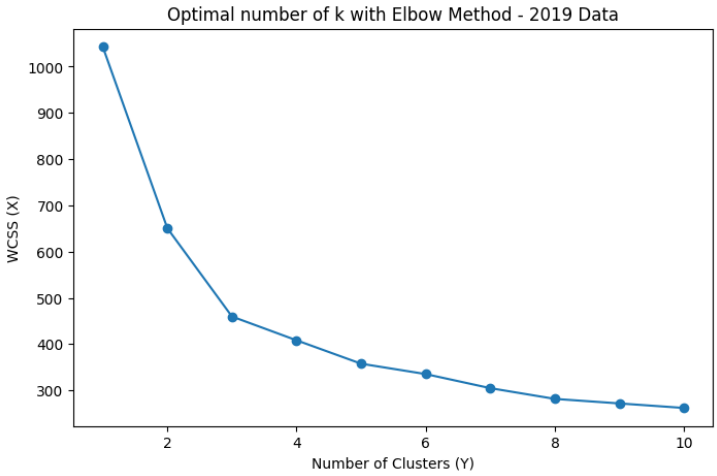
Algorithm Used:

- Applied K-Means++ clustering
- Finding the optimal number of k with the Elbow Method

Findings:

- From both datasets (2019 and 2015), the "elbow" is clearly observed around
- After k = 2 for 2015 and k = 3 for 2019, there is a significant drop, suggesting that adding more clusters does not significantly improve the clustering.





Decision:

- I have decided to select k = 2 for 2015 and k = 3 for 2019 in the future.

Why K-Means++?

- Offers improved centroid initialization to prevent getting stuck in unfavorable local minima.

```
 Applying K-Means++ Clustering to 2015 Data...

 Cluster Centers for K=2:
 [[-0.70038203 -0.61183276 -0.58807073 -0.56319348 -0.55673919 -0.35597253
  -0.24381719]
 [ 0.89899783  0.78533758  0.75483705  0.72290506  0.71462045  0.45691997
   0.31295937]]
Silhouette Score for K=2: 0.3021465397743113

 Applying K-Means++ Clustering to 2019 Data...

 Cluster Centers for K=3:
 [[-1.00901782 -1.17056662 -1.13903568 -1.18597995 -0.45897221  0.37040735
  -0.09254984]
 [ 1.43480381  1.20161977  0.94733547  1.09110994  1.01051778  1.01476245
   1.70655308]
 [ 0.16651982  0.33561226  0.39485418  0.37890281 -0.03436837 -0.53542877
  -0.46932604]]
Silhouette Score for K=3: 0.3386713402124577
```

2.5. Cluster evaluation

Evaluation Metric:

- I have used the Silhouette Score to evaluate how compact and well-separated the clusters are.

Results:

```
Applying K-Means++ Clustering to 2015 Data
K = 2 : Silhouette Score = 0.3021
K = 3 : Silhouette Score = 0.2860
K = 4 : Silhouette Score = 0.2451
K = 5 : Silhouette Score = 0.2329
K = 6 : Silhouette Score = 0.2245
K = 7 : Silhouette Score = 0.2148

Applying K-Means++ Clustering to 2019 Data
K = 2 : Silhouette Score = 0.3206
K = 3 : Silhouette Score = 0.3387
K = 4 : Silhouette Score = 0.2397
K = 5 : Silhouette Score = 0.2363
K = 6 : Silhouette Score = 0.2231
K = 7 : Silhouette Score = 0.2284
```

Year	Optimal K	Silhouette Score
2015	2	0.3021
2016	3	0.3387

Conclusion:

- The data structure evolved: in 2015, *two* clusters were favored, while in 2019, *three* clusters were required to best represent its distribution.

2.6. Model drift analysis

What was done:

- Feature Alignment (Manual Mapping)
    - The 2019 dataset's column names differed from those in 2015 (e.g., "GDP per capita" compared to "Economy (GDP per Capita)").
      To ensure fair clustering comparisons, the columns in the 2019 dataset were manually renamed using a mapping dictionary.
    - This manual mapping aligned seven features.

```
# Rename 2019 dataset columns to match 2015 names
column_mapping_2019 = {
    'GDP per capita': 'Economy (GDP per Capita)',
    'Social support': 'Family',
    'Healthy life expectancy': 'Health (Life Expectancy)',
    'Freedom to make life choices': 'Freedom',
    'Perceptions of corruption': 'Trust (Government Corruption)',
    'Generosity': 'Generosity',
    'Score': 'Happiness Score'
}
```

- Both aligned datasets were standardized using StandardScaler to ensure that they are comparable in the clustering process.

- Re-run the k-means++ for both datasets after the feature rename

- When re-running K-Means++ for centroid shift and cluster membership changes, it's better to use the same value of k for both years. Since we're making a comparison, this ensures a fair and consistent analysis.

```
Common features: ['Generosity', 'Health (Life Expectancy)', 'Happiness Score', 'Freedom', 'Economy (GDP per Capita)', 'Trust (Government Corruption)', 'Family']
Shapes after aligning features: (153, 7) (149, 7)

Applying K-Means++ Clustering to 2015 Data...
Silhouette Score for K=2: 0.3021

Applying K-Means++ Clustering to 2019 (2 clusters) Data...
Silhouette Score for K=2: 0.3206

Applying K-Means++ Clustering to 2019 (3 clusters) Data...
Silhouette Score for K=3: 0.3387
```

Centroid Shift:

- I calculated the distances between the corresponding centroids for the years 2015 (with K=2) and 2019 (with K=2), wherever applicable.

  Centroid Shifts between 2015 and 2019:
  Cluster 0: Shift Distance = 0.6685
  Cluster 1: Shift Distance = 0.6231

Cluster Membership Change:

- I re-clustered both datasets using aligned K=2 to ensure a fair comparison of the cluster assignments.
- Comparison was conducted using only the minimum overlapping entries due to a row mismatch (153 in 2015 and 149 in 2019).

- **30 out of 149 countries have changed their cluster membership.

```
Centroid Shifts between 2015 and 2019:
Cluster 0: Shift Distance = 0.6685
Cluster 1: Shift Distance = 0.6231

Cluster changes between 2015 and 2019:
⭐ ⭐ 30 out of 149 countries have changed their cluster
```

2.7. Visualization

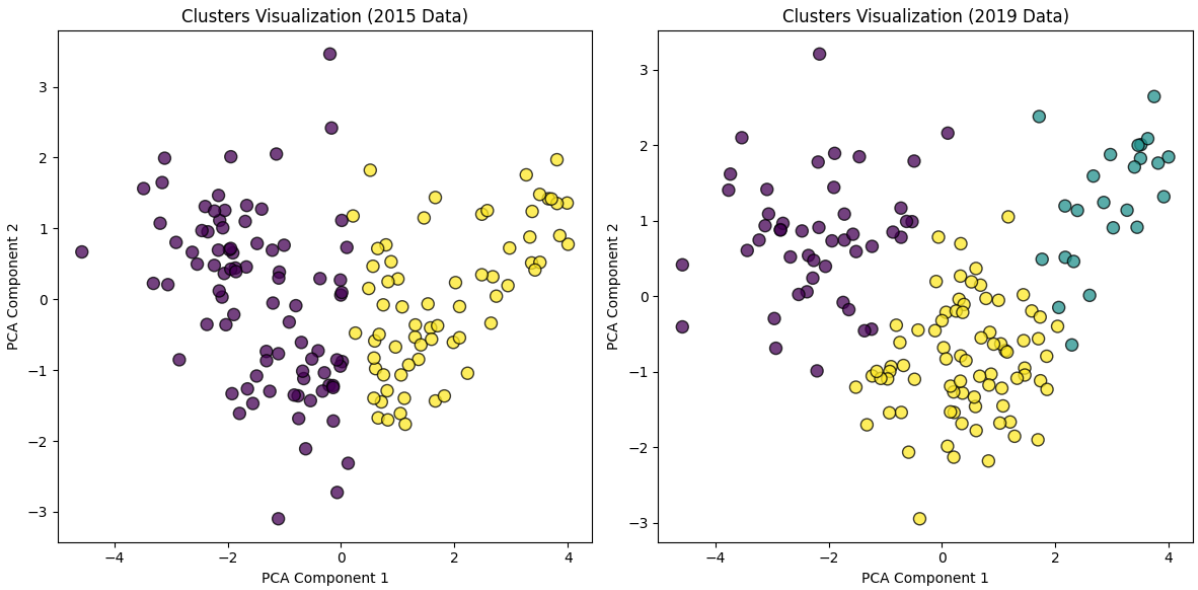
What was done:

- Performed Principal Component Analysis (PCA) to reduce feature dimensions from seven to two for visualization purposes.
- I created a scatter plot that overlays the PCA data from 2015 (k=2) and 2019 (k=3).
- Cluster centroids were highlighted, and arrows were drawn to indicate the drift of centroids between years in a scatter plot that overlays the PCA data from 2015.

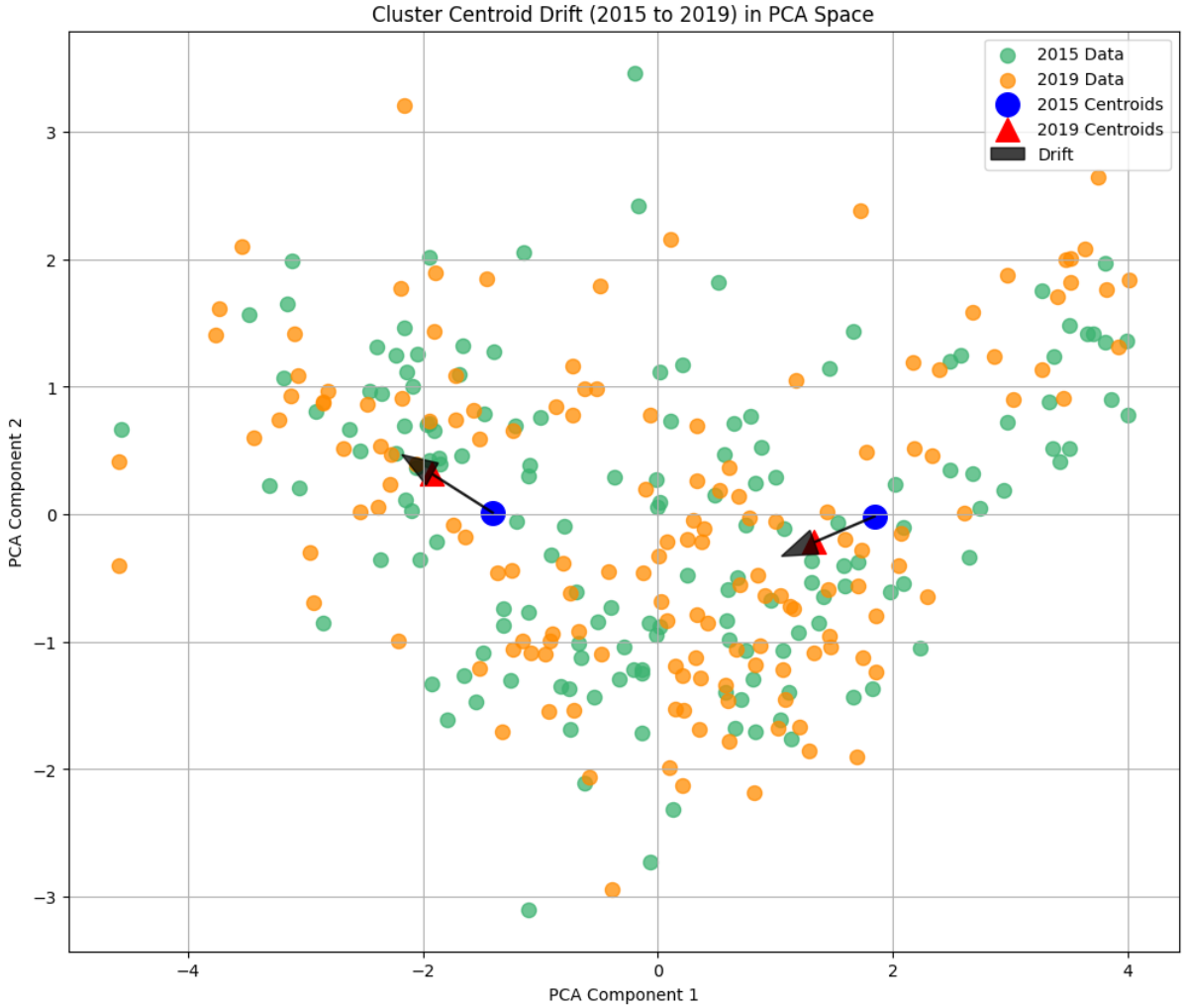
Visual Insights:

- In 2015, the clusters appeared more compact and clearly separated.
  In 2019, the clusters were more dispersed, indicating greater variation across countries with the new cluster.
- Arrows indicate significant shifts in centroids, confirming major transitions in economic and social dynamics over time.

Principal Component Analysis (PCA):



Cluster Change Over Time:



2.8. Conclusion and insight

- In 2015, the data was best grouped into 2 clusters.
- In 2019, the optimal number of clusters increased to 3, indicating a more segmented and complex structure.
- This change indicates the rise of a new group of countries with unique attributes related to happiness.

- The PCA visualizations clearly indicated that the positions of the cluster centroids shifted significantly between 2015 and 2019.
- The distances between centroids were uneven, indicating that some clusters experienced a bigger transformation than others.

- In 2015, the clusters were closely grouped together, indicating that the Countries within each cluster shared more similarities.
- In 2019, the clusters were more spread out, indicating greater diversity and variability within country groupings, and introducing a new cluster.

- ** 30 out of 149 countries shifted cluster membership from 2015 to 2019

Socio-Economic Trends

- Economic Development:

Countries that are experiencing growth in their GDP or have stronger economies may have moved into higher-scoring clusters.

- Healthcare Development and Life Expectancy:

Improved public health systems contributed to the rise of clusters with higher levels of happiness.

- Freedom and Corruption:

Countries that have improved their good governance, transparency, and civil democracy may have moved out of lower-scoring clusters.

Literature review on state-of-the-art Techniques

Question 01

Research Paper:
Performance Comparison of Simple Regression, Random Forest and XGBoost Algorithms for Forecasting Electricity Demand

https://www.researchgate.net/

While searching for ways to improve my regression model, I have seen this research paper that compares regression techniques. I had already implemented simple linear regression and random forest regression. After reviewing the paper, I became interested in adding XGBoost to my existing models for comparison.

In this paper, XGBoost demonstrated high performance compared to linear regression and random forest. As well as effectively capture the temporal patterns

Although I included XGBoost in my analysis, I noticed no notable performance improvement. This lack of improvement may be due to the small size of my dataset, which might not be sufficient to utilize the capabilities of XGBoost fully. But its figures are more aligned with the Random Forest.

XGBoost is an advanced algorithm that uses gradient boosting techniques. It constructs trees sequentially, with each new tree concentrating on correcting the errors made by the previous ones. XGBoost is renowned for its high performance, efficient processing, and capability to handle complex, non-linear relationships effectively.

These are the figures I have gathered from my model:

Model	MAE	MSE	RMSE	R^2 Score
Linear Reg.	39,272.19	2.45M	49,547.64	0.23
Random Forest	33,876.17	1.84M	42,940.75	0.46
XGBoost	34,254.74	1.89M	43,565.32	0.45

Question 02

Research Paper:
k-means++: The Advantages of Careful Seeding

http://ilpubs.stanford.edu:8090/778/1/2006-13.pdf

This is the same paper we used for our presentation. I recognized its potential, which is why I'm interested in using it for this question as well.
After reviewing the paper, I implemented its methodology for the clustering question in my coursework. Specifically, I utilized K-Means++ instead of standard K-Means throughout the clustering process.

This research paper introduces K-Means++, an enhancement of the traditional K-Means clustering algorithm that improves the selection of initial cluster centroids.

I didn't notice a significant impact on our current dataset. Based on several articles I read, this could be due to the relatively small size of our dataset, which may prevent methods like K-Means++ from showing noticeable differences.
Nevertheless, I chose to proceed with K-Means++ to maintain consistency and align with the approach discussed in the paper.

```
# run K-Means Clustering with Optimal K
def apply_kmeans(data, k):
    kmeans = KMeans(n_clusters    = k,
                    init          = 'random',
                    random_state  = 123
                    )
    lbl = kmeans.fit_predict(data)
    print(f"Cluster Centers for K={k}:\n", kmeans.cluster_centers_)
    print(f"Silhouette Score for K={k}: {silhouette_score(data, lbl):.2f}")
    return lbl, kmeans


# K based on the elbow method results
optimal_k_2015 = 2
optimal_k_2019 = 2

print("\n Applying K-Means Clustering to 2015 Data")
lbl_2015, kmeans_2015 = apply_kmeans(data_2015_scaled, optimal_k_2015)

print("\n Applying K-Means Clustering to 2019 Data")
lbl_2019, kmeans_2019 = apply_kmeans(data_2019_scaled, optimal_k_2019)
```

```
 Applying K-Means Clustering to 2015 Data
Cluster Centers for K=2:
 [[ 0.81071003 -0.79046535  0.2058028  -0.65045209 -0.63577354 -0.62160294
  -0.56976402 -0.34212985 -0.19799391 -0.31303372]
 [-0.89063919  0.86839855 -0.22609321  0.71458117  0.69845543  0.68288773
   0.62593794  0.37586096  0.21751443  0.3438962 ]]
Silhouette Score for K=2: 0.28

 Applying K-Means Clustering to 2019 Data
Cluster Centers for K=2:
 [[ 0.94608432 -0.9102789  -0.85220497 -0.86203229 -0.86724849 -0.52659386
   0.0691068  -0.22765179]
 [-0.73208906  0.70438248  0.65944432  0.66704879  0.67108514  0.40748334
  -0.0534755   0.17615912]]
Silhouette Score for K=2: 0.34
```

```
[27] # run K-Means++ Clustering with Optimal K

    def apply_kmeans(data, k):
        kmeans = KMeans(n_clusters  = k,
                        init        = 'k-means++',
                        random_state= 123)
        lbl = kmeans.fit_predict(data)
        print(f"Cluster Centers for K={k}:\n", kmeans.cluster_centers_)
        print(f"Silhouette Score for K={k}: {silhouette_score(data, lbl):.2f}")
        return lbl, kmeans

    # K based on the elbow method results
    optimal_k_2015 = 2
    optimal_k_2019 = 2


    print("\n Applying K-Means Clustering to 2015 Data")
    lbl_2015, kmeans_2015 = apply_kmeans(data_2015_scaled, optimal_k_2015)

    print("\n Applying K-Means Clustering to 2019 Data")
    lbl_2019, kmeans_2019 = apply_kmeans(data_2019_scaled, optimal_k_2019)
```

```
 Applying K-Means Clustering to 2015 Data
Cluster Centers for K=2:
 [[ 0.74569057 -0.73505333  0.16542453 -0.60953701 -0.58128513 -0.56117653
  -0.49490604 -0.36840972 -0.13413078 -0.31086037]
 [-0.96366166  0.94991508 -0.2137794   0.78770936  0.75119925  0.72521274
   0.63957088  0.47609872  0.17333823  0.40172724]]
Silhouette Score for K=2: 0.28

 Applying K-Means Clustering to 2019 Data
Cluster Centers for K=2:
 [[ 0.94608432 -0.9102789  -0.85220497 -0.86203229 -0.86724849 -0.52659386
   0.0691068  -0.22765179]
 [-0.73208906  0.70438248  0.65944432  0.66704879  0.67108514  0.40748334
  -0.0534755   0.17615912]]
Silhouette Score for K=2: 0.34
```

References

1. https://medium.com/data-science/a-practical-guide-on-k-means-clustering-ca3bef3c853d

2. https://medium.com/@robertb909/k-means-clustering-a64f859a1074

3. https://medium.com/data-scientists-diary/principal-component-analysis-in-python-f735fefccdec

4. https://www.tpointtech.com/evaluation-of-clustering-in-data-mining#:~:text=Davies%2DBouldin%20Index%3A%20The%20average,and%20minimum%20inter%2Dcluster%20distances.

5. https://www.youtube.com/watch?v=q_shb4Lyqg0&ab_channel=JCharisTech

6. https://stackoverflow.com/questions/68040073/euclidean-distance-between-the-two-points-using-vectorized-approach