TABLE OF CONTENTS

1. Introduction (#introduction)

2. Importing Necessary Libraries

3. Loading the Dataset

4. Exploratory Data Analysis (EDA)

- 4.1 [Understanding the Data Structure]

- 4.2 [Statistical Summary]

- 4.3 [Checking for Missing Values]

- 4.4 [Checking for Duplicates]

- 4.5 [Data Visualization]

5. Data Cleaning

- 5.1 [Handling Missing Values]

6. Feature Engineering

- 6.1 [Creating New Features]

- 6.2 [Encoding Categorical Variables]

7. Feature Selection

- 7.1 [Correlation Analysis]

8. Preparing Data for Modeling

9. Conclusion

10.Next Steps

**Project title:**

"Predicting Global Happiness Scores:

A Comprehensive Analysis and Machine Learning Approach"

***1. Introduction***:

The World Happiness Report is a landmark survey of the state of global happiness. It ranks countries by how happy their citizens perceive themselves to be, based on various factors such as economic production, social support, life expectancy, freedom, absence of corruption, and generosity.

**Objective:**

- To clean and preprocess the dataset effectively to build a robust machine learning model that accurately predicts the happiness score of countries based on various socio-economic factors.

- To explore and understand the underlying factors contributing to happiness and how they interact with each other.

**2. Importing Necessary Libraries:**

First, we'll import all the necessary Python libraries required for data manipulation, visualization, and modelling.

**Python:**

# Data manipulation and analysis

* import pandas as pd
* import numpy as np

**Data visualization:**

* import matplotlib.pyplot as plt
* import seaborn as sns

**Machine learning:**

* from sklearn.preprocessing import LabelEncoder, OneHotEncoder, StandardScaler, MinMaxScaler
* from sklearn.ensemble import RandomForestRegressor
* from sklearn.model\_selection import train\_test\_split, cross\_val\_score
* from sklearn.metrics import mean\_squared\_error, r2\_score

**3. Loading the Dataset:**

We shall load the dataset into a pandas DataFrame for analysis.

**Python:**

# Define the file path

file\_path = '/World\_Happiness\_Report.csv'

# Load the dataset

df = pd.read\_csv(file\_path)

# Display the first five rows

df.head()

**4. Exploratory Data Analysis (EDA):**

EDA is a crucial step to understand the dataset's structure, identify patterns, and detect anomalies.

4.1 **Understanding the Data Structure:**

**Python:**

# Check the shape of the dataset

print(f"The dataset contains {df.shape[0]} rows

and {df.shape[1]} columns.")

# Get an overview of the dataset

df.info()

**Explanation:**

- ‘df.shape’ gives the number of rows and columns.

- ‘df.info()’ provides details about data types and non-null counts for each column.

**4.2 Statistical Summary:**

**python**

# Get statistical summary of numerical columns

df.describe()

**Explanation:**

- ‘df.describe()’ provides count, mean, standard deviation, min, max, and quartile values for numerical features.

**4.3 Checking for Missing Values:**

**python**

# Check for missing values

missing\_values = df.isnull().sum().sort\_values(ascending=False)

missing\_values = missing\_values[missing\_values > 0]

if missing\_values.empty:

print("No missing values found in the dataset.")

else:

print("Missing values found in the following columns:")

print(missing\_values)

**Explanation:**

- ‘df.isnull().sum()’ calculates the total missing values per column.

- The code checks if there are any columns with missing values and lists them.

**4.4 Checking for Duplicates:**

**python**

# Check for duplicate rows

duplicate\_rows = df[df.duplicated()]

print(f"Number of duplicate rows: {duplicate\_rows.shape[0]}")

**4.5 Data Visualization:**

Visualizations help in understanding the distribution and relationships between variables.

**4.5.1 Distribution of Happiness Score:**

**python**

# Histogram of Happiness Score

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

sns.histplot(df['Happiness Score'], kde=True, bins=30)

plt.title('Distribution of Happiness Score')

plt.xlabel('Happiness Score')

plt.ylabel('Frequency')

plt.show()

**Explanation:**

- The histogram shows how happiness scores are distributed across countries.

**4.5.2 Correlation Heatmap:**

python

# Compute correlation matrix

corr\_matrix = df.corr()

plt.figure(figsize=(12,8))

sns.heatmap(corr\_matrix, annot=True)

plt.title('Correlation Heatmap')

plt.show()

**4.5.3 Pairplot of Key Features:**

**python**

# Select key features for pairplot

key\_features = ['Happiness Score', 'GDP per Capita',

'Social Support', 'Healthy Life Expectancy', 'Freedom',

'Generosity', 'Perceptions of Corruption']

# Plot pairplot

sns.pairplot(df[key\_features], diag\_kind='kde')

plt.show()

**Explanation:**

- Pairplot shows pairwise relationships between key features.

- Diagonal plots show distributions of individual features.

**5. Data Cleaning:**

Data cleaning involves handling missing values, duplicates, and outliers to improve data quality.

**5.1 Handling Missing Values:**

**python**

# Visualize missing values using heatmap

plt.figure(figsize=(12,6))

sns.heatmap(df.isnull(), cbar=False, cmap='viridis')

plt.title('Missing Values Heatmap')

plt.show()

**5.2 Handling Outliers:**

Outliers can skew the data and adversely affect model performance.

**Detecting Outliers using Boxplots:**

**Python:**

# Plot boxplots for numerical features

plt.figure(figsize=(15,10))

for i, col in enumerate(numerical\_cols):

plt.subplot(3, 3, i+1)

sns.boxplot(y=df[col])

plt.title(f'Boxplot of {col}')

plt.tight\_layout()

plt.show()

**6. Feature Engineering:**

Feature engineering enhances model performance by creating new relevant features or transforming existing ones.

**6.1Encoding Categorical Variables:**

**python**

# List of categorical columns

categorical\_cols =

df\_clean.select\_dtypes(include=['object']).columns

print("Categorical columns:", categorical\_cols)

**7. Feature Selection:**

Selecting the most relevant features enhances model performance and reduces complexity.

**7.1 Correlation Analysis:**

**python**

# Calculate correlation matrix

corr\_matrix = df\_clean.corr()

# Select features highly correlated with 'Happiness Score'

corr\_target = abs(corr\_matrix['Happiness Score'])

relevant\_features = corr\_target[corr\_target]

**7.2: Feature Selection:**

Use correlation analysis to select the most important features

that contribute to the happiness score.

**python**

# Correlation matrix

correlation\_matrix = df.corr()

correlation\_matrix ['Happiness

Score'].sort\_values(ascending=False))

selected\_features = correlation\_matrix['Happiness

Score'].sort\_values(ascending=False).index[1:6]

print("Selected features for modeling:", selected\_features)

# Prepare the dataset for modeling

X = df[selected\_features]

y = df['Happiness Score']

**7.3: Splitting Data for Training and Testing:**

Split the data into training and testing sets.

**python**

from sklearn.model\_selection import train\_test\_split

# Split the dataset into training and testing sets

X\_train, X\_test, y\_train, y\_test =

train\_test\_split(X,y,test\_size=0.2, random\_state=42)

print("Training set shape:", X\_train.shape)

print("Testing set shape:", X\_test.shape)

**7.4: Model Building:**

Start building your machine learning model. For example, let's use a RandomForestRegressor to predict the happiness score.

**python**

from sklearn.ensemble import RandomForestRegressor

from sklearn.metrics import mean\_squared\_error, r2\_score

# Initialize the model

model = RandomForestRegressor(random\_state=42)

# Train the model

model.fit(X\_train, y\_train)

# Make predictions on the test set

y\_pred = model.predict(X\_test)

**7.5: Evaluate the model:**

mse = mean\_squared\_error(y\_test, y\_pred)

r2 = r2\_score(y\_test, y\_pred)

print("Mean Squared Error:", mse)

print("R-squared:", r2)

**CONCLUSION:**

In this analysis, we explored the application of machine learning for regression using the Random Forest model. After training the model and evaluating its performance on a held-out test set, we observed **promising results with a Mean Squared Error (MSE) of 0.17 and an R-squared of 0.79**.

These metrics indicate that the model effectively captures a significant portion of the variance in the target variable and provides accurate predictions on unseen data.

Future work could involve exploring alternative regression models such as Linear Regression, Support Vector Regression, Gradient Boosting Machines (e.g., XGBoost, LightGBM), or Neural Networks.

A comparative analysis of these models, along with hyperparameter tuning, could lead to the identification of an even more effective model for this particular regression task.

Furthermore, additional evaluation metrics like RMSE and MAE, and visualization techniques such as residual plots, could provide a more comprehensive understanding of the model's strengths and weaknesses.

REFERENCES:

1. [www.geeksforgeeks.org/support-vector-machine-vs-extreme-gradient-boosting/](https://www.geeksforgeeks.org/support-vector-machine-vs-extreme-gradient-boosting/" \t "_blank)
2. [www.analyticsvidhya.com/blog/2020/03/support-vector-regression-tutorial-for-machine-learning/](https://www.analyticsvidhya.com/blog/2020/03/support-vector-regression-tutorial-for-machine-learning/" \t "_blank)
3. [medium.com/@taniyaghosh29/machine-learning-algorithms-what-are-the-differences-9b71df4f248f](mailto:medium.com/@taniyaghosh29/machine-learning-algorithms-what-are-the-differences-9b71df4f248f)
4. [www.geeksforgeeks.org/machine-learning-algorithms/](http://www.geeksforgeeks.org/machine-learning-algorithms/)
5. [eigenvector.com/wp-content/uploads/2020/03/Wise\_APACT\_Nonlinear\_Comparison.pdf](https://eigenvector.com/wp-content/uploads/2020/03/Wise_APACT_Nonlinear_Comparison.pdf" \t "_blank)