THANK YOU

AIML - Cohort1

Life Expectancy

Sunil & Sreeram

Agenda

- To write/create an automation to bring R square value for various methods and understand different parameters and their effects.

Data Summary (Life Expectancy Analysis)

- 193 countries participation
- 22 Factors (Columns)
- 15 years of data spread
- Factors involved are:
    - Immunization
    - Mortality
    - Economical
    - Social

Column Name(s)	Column Data Type
Country, Status, Year	Discrete
Other	Continuous

Note: We are following regression model as output “Life Expectancy” is a continuous type output.

Phases

1. Data Column Fixes
2. Missing Data Identification and Columns reduction
3. Data Correlation and Column reductions
4. Outliers and Data Normalization
5. Data Distribution Analysis
6. Write Script/Automation for full analysis with various methods

1a. Data Column Fixes

Summary:

- Remove whitespaces from both ends
- Replace underscore with whitespace in between [optional]
- Replace all upper cases to lower cases for uniformity [optional]

Result: Get uniform and clean columns names

1b. Data Column Fixes

Before

```
Data columns (total 22 columns):
 #   Column                           Non-Null Count  Dtype
---  ------                           --------------  -----
 0   Country                          2938 non-null   object
 1   Year                             2938 non-null   int64
 2   Status                           2938 non-null   object
 3   Life expectancy                  2928 non-null   float64
 4   Adult Mortality                  2928 non-null   float64
 5   infant deaths                    2938 non-null   int64
 6   Alcohol                          2744 non-null   float64
 7   percentage expenditure           2938 non-null   float64
 8   Hepatitis B                      2385 non-null   float64
 9   Measles                          2938 non-null   int64
 10   BMI                             2904 non-null   float64
 11  under-five deaths                2938 non-null   int64
 12  Polio                            2919 non-null   float64
 13  Total expenditure                2712 non-null   float64
 14  Diphtheria                       2919 non-null   float64
 15   HIV/AIDS                        2938 non-null   float64
 16  GDP                              2490 non-null   float64
 17  Population                       2286 non-null   float64
 18   thinness  1-19 years            2904 non-null   float64
 19   thinness 5-9 years              2904 non-null   float64
 20  Income composition of resources  2771 non-null   float64
 21  Schooling                        2775 non-null   float64
dtypes: float64(16), int64(4), object(2)
```

After

```
Data columns (total 22 columns):
 #   Column                           Non-Null Count  Dtype
---  ------                           --------------  -----
 0   country                          2938 non-null   object
 1   year                             2938 non-null   int64
 2   status                           2938 non-null   object
 3   life_expectancy                  2928 non-null   float64
 4   adult_mortality                  2928 non-null   float64
 5   infant_deaths                    2938 non-null   int64
 6   alcohol                          2744 non-null   float64
 7   percentage_expenditure           2938 non-null   float64
 8   hepatitis_b                      2385 non-null   float64
 9   measles                          2938 non-null   int64
 10  bmi                              2904 non-null   float64
 11  under-five_deaths                2938 non-null   int64
 12  polio                            2919 non-null   float64
 13  total_expenditure                2712 non-null   float64
 14  diphtheria                       2919 non-null   float64
 15  hiv/aids                         2938 non-null   float64
 16  gdp                              2490 non-null   float64
 17  population                       2286 non-null   float64
 18  thinness__1-19_years             2904 non-null   float64
 19  thinness_5-9_years               2904 non-null   float64
 20  income_composition_of_resources  2771 non-null   float64
 21  schooling                        2775 non-null   float64
dtypes: float64(16), int64(4), object(2)
```

2. Missing Data Identification

Summary:

- **29%** data missing from **“Population”** column
- **23%** data missing from **“Hepatitis B”** column
- **18%** data is missing from **“GDP”** column

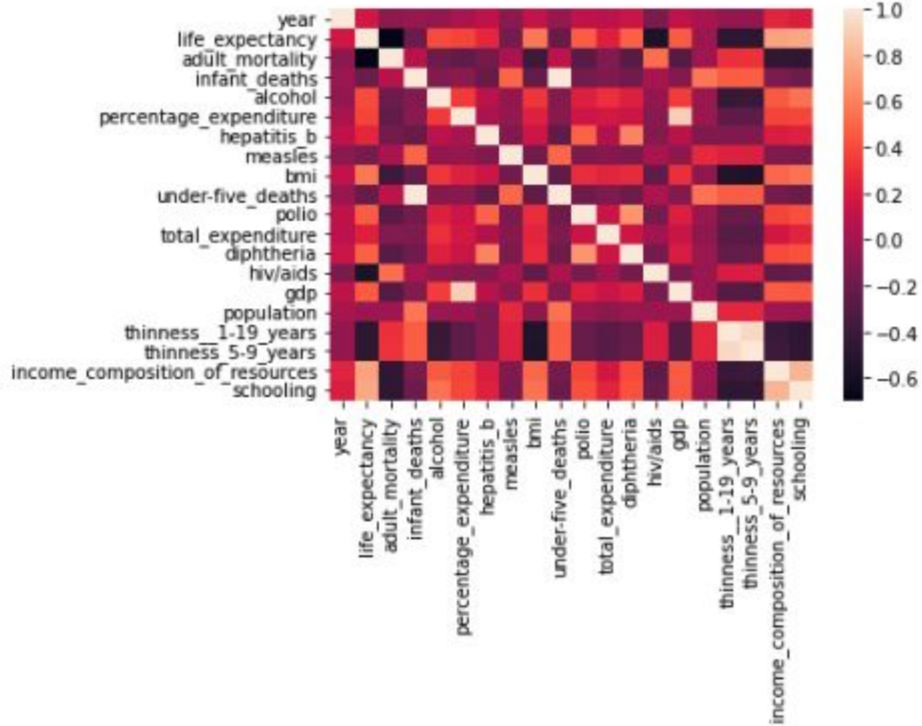
Result: Removed columns having high volume of missing data.

3a. Correlation of Data columns

Summary:

- Infant deaths and under-five death have **0.99** correlation
  - Infant deaths are superset of under-five
  - Keeping “**infant_deaths**” for the spread
- Thinness 5-9 years and Thinness 1-19 years have **0.93** correlation
  - 1-19 is superset of 5-9
  - Keeping “**thinness_1-19_years**” data for spread
- GDP vs percentage expenditure have **0.89** correlation
  - Keeping “**percentage_expenditure**” as GDP has big missing data chunk

3b. Correlation by Visualization [Heatmap] (cont.)

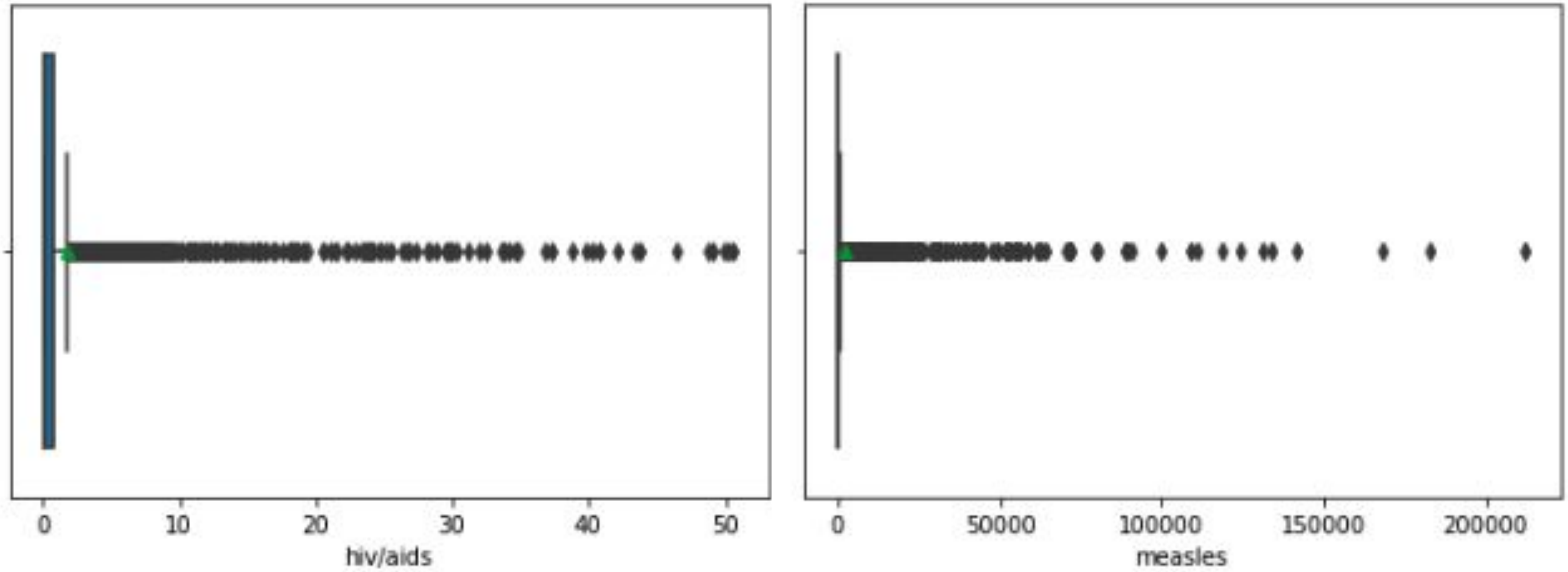


4a. Outliers Identification and Imputation

Summary:

Data Point	Outlier Records (Count)	Outlier Records (%)	Method of handling
hiv/aids	542	18.44%	Dropped the column
measles	542	18.44%	Dropped the column
under-five_deaths	394	13.41%	Drop the rows
percentage_expenditure	389	13.34%	Drop the rows
infant_deaths	315	10.72%	Drop the rows

4b. Outliers by Visualization [BoxPlot]

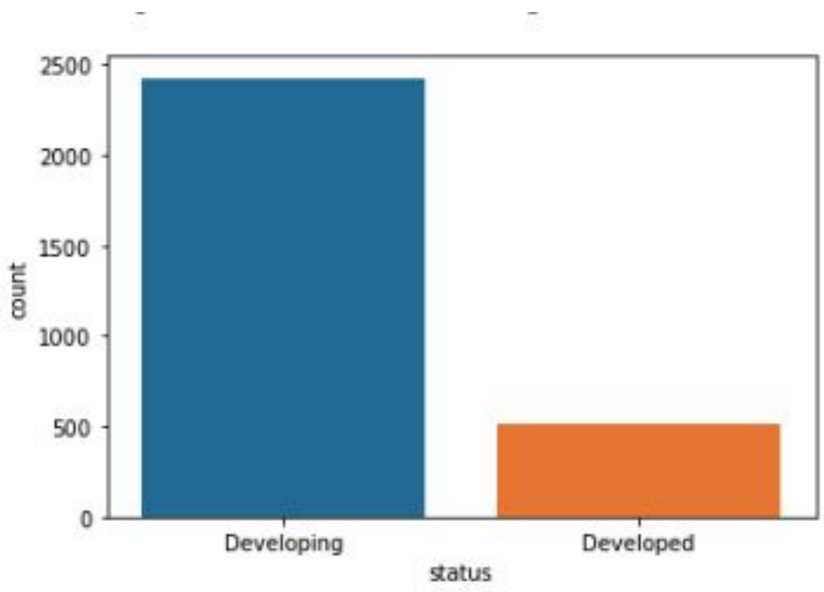


5a. Data Distribution

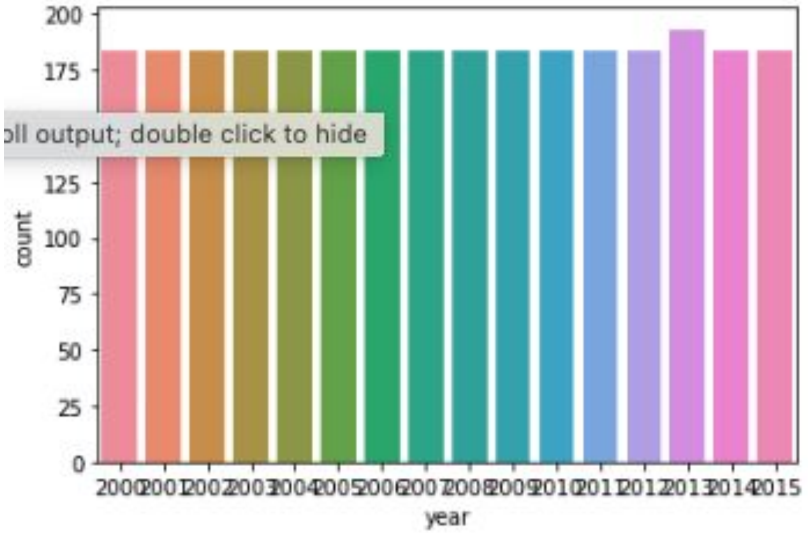
Summary:

- Identify data uniformity
- Low Density [handled in previous step]
- Too much diversity

Data Distribution

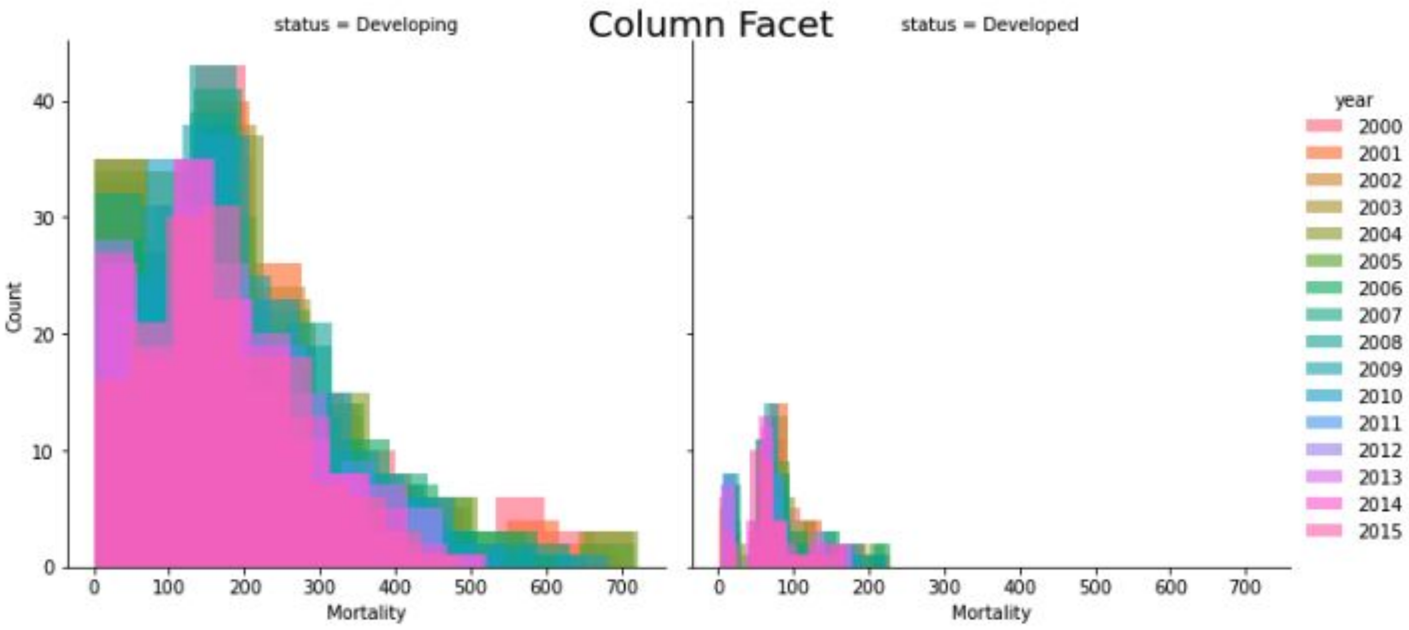


Uneven Status Distribution
[Bar Plot]



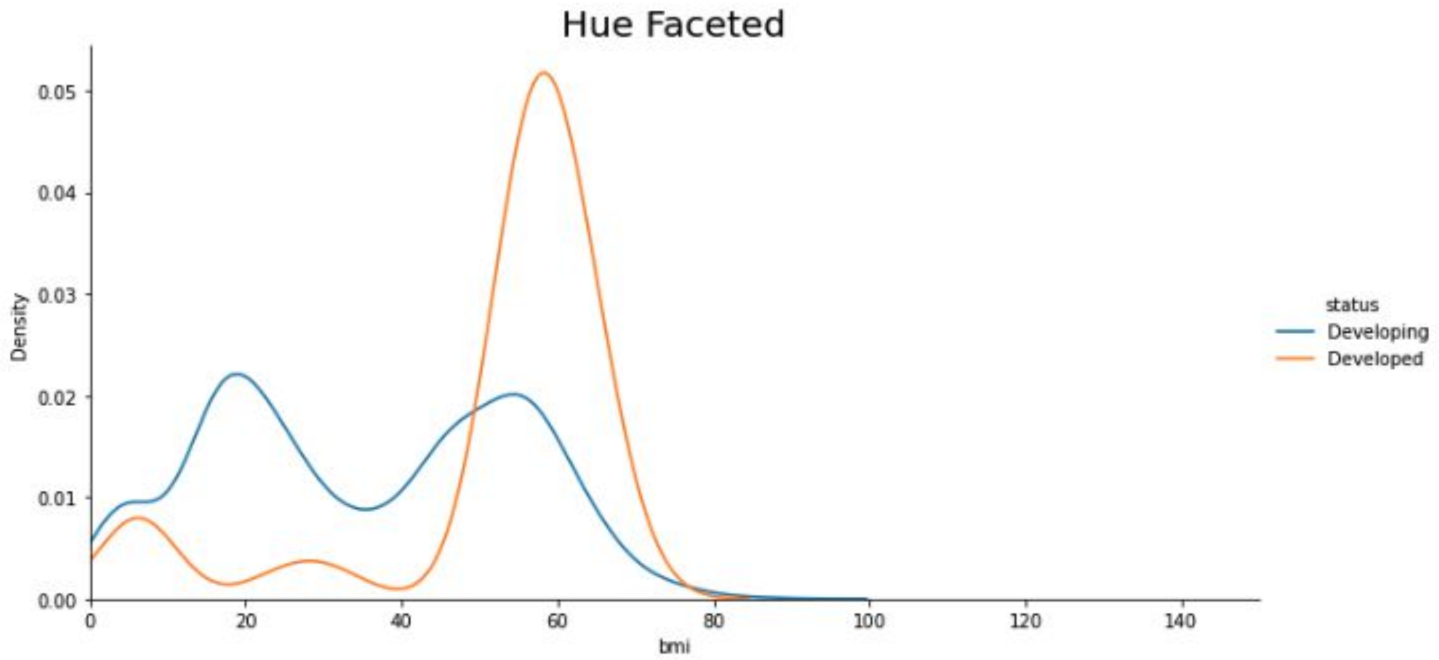
Uniform Data Samples YoY
[Count Plot]

Data Spread/Distribution



Mortality rate was high and reducing over the years
[FacetGrid Plot]

Data Spread/Distribution [FacetGrid Plot]



6. Script & Automation

Script Config Parameters:

1. Input Data Path – *gr5regression.csv*
2. Output Component – *Life Expectancy*
3. Discrete Components – *[“Country”, “Year”, “Status”]*
4. Random State Values – *Prime Numbers between 1-500*
5. KNN Imputer num of neighbors – *4*
6. Highly Correlation – *> 0.9 or < -0.9*
7. Test Size – *10% of actual data*
8. Acceptance % of empty data – *10% of actual data*
9. Unique values count for One hot encoding – *< 8*
10. Outliers column Removal – *> 5%*

6a. Methods Used in Script

1. Mode Or Median
2. KNN Imputer
3. Normalisation
4. Standardisation
5. PCA
6. Outliers Removal
7. One Hot Encoding

6b. Important Methods

```python
def get_null_columns(self):
    """
    Returns columns dict which has null values
    """
    null_count_series = self.data.isnull().sum()
    null_count_dict = null_count_series.to_dict()  # gets {"<column>" :  <null_count> int }
    return {column: null_count_dict[column] for column in null_count_dict.keys() if null_count_dict[column] != 0}

def clean_data_with_mode_or_median(self):
    """
    Replaces empty data with either mode(for discrete data)  and median(for continuous data)
    :return:
    """
    log.logger.info("Replacing empty data with either mode / median")
    null_columns_dict = self.get_null_columns()
    self.cleaned_data = self.data.copy()
    column_data_types = self.cleaned_data.dtypes
    for column in null_columns_dict.keys():
        replaceable_value = self.cleaned_data[column].median() if column_data_types[column] != "O" else \
            self.cleaned_data[column].mode()[0]
        self.cleaned_data[column] = self.cleaned_data[column].fillna(replaceable_value)
```

6b. Important Methods

```python
def get_highly_correlated_columns(self):
    """
    Calculates correlation between columns and returns which are higher
    :return: highly correlated columns
    """
    correlation = self.cleaned_data.corr()
    highly_correlated_columns = {}
    for column1 in correlation.columns:
        for column2 in correlation.columns:
            if column1 == column2:
                continue
            if correlation[column1][column2] >= 0.90 or correlation[column1][column2] <= -0.90:
                if column2 in highly_correlated_columns.keys() and highly_correlated_columns[column2] == column1:
                    continue
                highly_correlated_columns[column1] = column2
                log.logger.debug(f"{column1} and {column2} are highly correlated:  {correlation[column1][column2]}")
    return highly_correlated_columns.keys()
```

6b. Important Methods

```python
def clean_data_with_knnimputer(self):
    """

    Replaces empty data using KNN imputer method
    :return:
    """
    log.logger.info("Replacing empty data using KNN Imputer method")
    null_columns_dict = self.get_null_columns()
    knn_imp = KNNImputer(n_neighbors=4, weights="uniform")
    self.cleaned_data = pd.DataFrame()
    knn_cleaned_data = knn_imp.fit_transform(self.data[null_columns_dict.keys()])
    knn_cleaned_data = pd.DataFrame(knn_cleaned_data, columns=null_columns_dict.keys())
    for pos, column in enumerate(self.data.columns):
        column_data = self.data[column] if column not in knn_cleaned_data.columns else knn_cleaned_data[column]
        self.cleaned_data.insert(pos, column, column_data, True)
```

6b. Important Methods

```python
def calculate_outliers(component):
    quartile_1, quartile_3 = np.percentile(component, [25, 75])
    iqr = quartile_3 - quartile_1
    lower_bound = quartile_1 - (iqr * 1.5)
    upper_bound = quartile_3 + (iqr * 1.5)
    outliers_tuple = np.where((component > upper_bound) | (component < lower_bound))
    return set(outliers_tuple[0])
```

```python
def remove_outliers(self):
    """
    Calculates Outliers and removes respective rows,
     and also columns whose outliers count is > 5% of data
    """
    log.logger.info("Processing Outliers")
    outliers_dict = {}
    num_of_rows = self.data.shape[0]
    column_data_types = self.cleaned_data.dtypes
    for column in self.cleaned_data.columns:
        outliers_dict[column] = calculate_outliers(self.cleaned_data[column]) if column_data_types[
                                                                                  column] != "O" else set()
    # Sample output of outliers_dict {'Country': set(), 'Life expectancy': {2306, 2307, 2308}}
    rows_to_be_removed = set()
    for column in outliers_dict.keys():
        if len(outliers_dict[column])/num_of_rows > 0.05:
            log.logger.debug(f"Dropping column {column} as it has {len(outliers_dict[column])} outliers ie > 5%")
            self.cleaned_data = self.cleaned_data.drop(column, axis=1)
        elif len(outliers_dict[column]) > 0 and len(outliers_dict[column])/num_of_rows <= 0.05:
            log.logger.debug(f"Dropping {len(outliers_dict[column])} outliers from '{column}' ")
            rows_to_be_removed = rows_to_be_removed.union(outliers_dict[column])
    log.logger.info(f"Total outliers removed are {len(rows_to_be_removed)}")
    self.cleaned_data = self.cleaned_data.drop(rows_to_be_removed)
```

```
(.env) sreeram.ganta@sreeram-ltm79v4 AIML_Project % python3 l2_project.py
2022-06-08 20:09:43,959 - REGRESSION-LEARNING-DEBUG - Logs will be stored in /var/tmp/regression_learning_20220608200943.log
2022-06-08 20:09:43,959 - REGRESSION-LEARNING-INFO  - ===== Starting Regression Learning =====
2022-06-08 20:09:43,959 - REGRESSION-LEARNING-INFO  - Reading data from ./gr5regression.csv
2022-06-08 20:09:43,968 - REGRESSION-LEARNING-INFO  - Continuous Columns are : ['Life expectancy', 'Adult Mortality', 'infant deaths', 'Alcohol', 'percentage expenditure', 'Hep
atitis B', 'Measles', 'BMI', 'under-five deaths', 'Polio', 'Total expenditure', 'Diphtheria', 'HIV/AIDS', 'GDP', 'Population', 'thinness  1-19 years', 'thinness 5-9 years', 'In
come composition of resources', 'Schooling']
2022-06-08 20:09:43,968 - REGRESSION-LEARNING-INFO  - ========= Method: Mode_or_Median =========
2022-06-08 20:09:43,968 - REGRESSION-LEARNING-INFO  - Replacing empty data with either mode / median
2022-06-08 20:09:43,976 - REGRESSION-LEARNING-INFO  - Calculating highly correlated columns
2022-06-08 20:09:43,979 - REGRESSION-LEARNING-DEBUG - infant deaths and under-five deaths are highly correlated:  0.996628882039801
2022-06-08 20:09:43,980 - REGRESSION-LEARNING-DEBUG - percentage expenditure and GDP are highly correlated:  0.9018191027160023
2022-06-08 20:09:43,983 - REGRESSION-LEARNING-DEBUG - thinness  1-19 years and thinness 5-9 years are highly correlated:  0.9391873974004579
2022-06-08 20:09:43,983 - REGRESSION-LEARNING-INFO  - Dropping highly correlated columns dict_keys(['infant deaths', 'percentage expenditure', 'thinness  1-19 years'])
2022-06-08 20:09:43,984 - REGRESSION-LEARNING-INFO  - Formulating R2 using Mode or Median
2022-06-08 20:09:46,117 - REGRESSION-LEARNING-INFO  - ========= Method: KNNImputer =========
2022-06-08 20:09:46,117 - REGRESSION-LEARNING-INFO  - Replacing empty data using KNN Imputer method
2022-06-08 20:09:46,359 - REGRESSION-LEARNING-INFO  - Calculating highly correlated columns
2022-06-08 20:09:46,363 - REGRESSION-LEARNING-DEBUG - infant deaths and under-five deaths are highly correlated:  0.996628882039801
2022-06-08 20:09:46,366 - REGRESSION-LEARNING-DEBUG - thinness  1-19 years and thinness 5-9 years are highly correlated:  0.9392700801649311
2022-06-08 20:09:46,366 - REGRESSION-LEARNING-INFO  - Dropping highly correlated columns dict_keys(['infant deaths', 'thinness  1-19 years'])
2022-06-08 20:09:46,368 - REGRESSION-LEARNING-INFO  - Formulating R2 using KNNImputer
2022-06-08 20:09:48,670 - REGRESSION-LEARNING-INFO  - ========= Method: Normalisation =========
2022-06-08 20:09:48,670 - REGRESSION-LEARNING-INFO  - Normalising all continuous columns
2022-06-08 20:09:48,671 - REGRESSION-LEARNING-INFO  - Replacing empty data using KNN Imputer method
2022-06-08 20:09:48,901 - REGRESSION-LEARNING-INFO  - Normalised all continuous columns
2022-06-08 20:09:48,902 - REGRESSION-LEARNING-INFO  - Calculating highly correlated columns
2022-06-08 20:09:48,905 - REGRESSION-LEARNING-DEBUG - infant deaths and under-five deaths are highly correlated:  0.9966288820398042
2022-06-08 20:09:48,908 - REGRESSION-LEARNING-DEBUG - thinness  1-19 years and thinness 5-9 years are highly correlated:  0.9392700801649325
2022-06-08 20:09:48,909 - REGRESSION-LEARNING-INFO  - Dropping highly correlated columns dict_keys(['infant deaths', 'thinness  1-19 years'])
2022-06-08 20:09:48,910 - REGRESSION-LEARNING-INFO  - Formulating R2 using Normalisation
2022-06-08 20:09:51,125 - REGRESSION-LEARNING-INFO  - ========= Method: Standardisation =========
2022-06-08 20:09:51,125 - REGRESSION-LEARNING-INFO  - Standardising all continuous columns
2022-06-08 20:09:51,125 - REGRESSION-LEARNING-INFO  - Replacing empty data using KNN Imputer method
2022-06-08 20:09:51,375 - REGRESSION-LEARNING-INFO  - Standardised all continuous columns
2022-06-08 20:09:51,375 - REGRESSION-LEARNING-INFO  - Calculating highly correlated columns
2022-06-08 20:09:51,379 - REGRESSION-LEARNING-DEBUG - infant deaths and under-five deaths are highly correlated:  0.9966288820398009
2022-06-08 20:09:51,383 - REGRESSION-LEARNING-DEBUG - thinness  1-19 years and thinness 5-9 years are highly correlated:  0.9392700801649303
2022-06-08 20:09:51,384 - REGRESSION-LEARNING-INFO  - Dropping highly correlated columns dict_keys(['infant deaths', 'thinness  1-19 years'])
2022-06-08 20:09:51,385 - REGRESSION-LEARNING-INFO  - Formulating R2 using Standardisation
2022-06-08 20:09:53,648 - REGRESSION-LEARNING-INFO  - ========= Method: PCA1 =========
2022-06-08 20:09:53,648 - REGRESSION-LEARNING-INFO  - Reducing dimensionality using PCA
2022-06-08 20:09:53,648 - REGRESSION-LEARNING-INFO  - Removing columns with missing data more than 10%
```

```
2022-06-08 10:30:45,889 - REGRESSION-LEARNING-INFO  - ========= Method: PCA1 =========
2022-06-08 10:30:45,889 - REGRESSION-LEARNING-INFO  - Reducing dimensionality using PCA
2022-06-08 10:30:45,889 - REGRESSION-LEARNING-INFO  - Removing columns with missing data more than 10%
2022-06-08 10:30:45,892 - REGRESSION-LEARNING-INFO  - Replacing empty data using KNN Imputer method
2022-06-08 10:30:45,994 - REGRESSION-LEARNING-INFO  - Processing Outliers
2022-06-08 10:30:46,004 - REGRESSION-LEARNING-DEBUG - Dropping 12 outliers from 'Life expectancy'
2022-06-08 10:30:46,004 - REGRESSION-LEARNING-DEBUG - Dropping 82 outliers from 'Adult Mortality'
2022-06-08 10:30:46,004 - REGRESSION-LEARNING-DEBUG - Dropping column infant deaths as it has 315 outliers ie > 5%
2022-06-08 10:30:46,006 - REGRESSION-LEARNING-DEBUG - Dropping 1 outliers from 'Alcohol'
2022-06-08 10:30:46,006 - REGRESSION-LEARNING-DEBUG - Dropping column percentage expenditure as it has 389 outliers ie > 5%
2022-06-08 10:30:46,007 - REGRESSION-LEARNING-DEBUG - Dropping column Measles as it has 542 outliers ie > 5%
2022-06-08 10:30:46,008 - REGRESSION-LEARNING-DEBUG - Dropping column under-five deaths as it has 394 outliers ie > 5%
2022-06-08 10:30:46,009 - REGRESSION-LEARNING-DEBUG - Dropping column Polio as it has 280 outliers ie > 5%
2022-06-08 10:30:46,010 - REGRESSION-LEARNING-DEBUG - Dropping 37 outliers from 'Total expenditure'
2022-06-08 10:30:46,010 - REGRESSION-LEARNING-DEBUG - Dropping column Diphtheria as it has 300 outliers ie > 5%
2022-06-08 10:30:46,011 - REGRESSION-LEARNING-DEBUG - Dropping column HIV/AIDS as it has 542 outliers ie > 5%
2022-06-08 10:30:46,012 - REGRESSION-LEARNING-DEBUG - Dropping 89 outliers from 'thinness  1-19 years'
2022-06-08 10:30:46,012 - REGRESSION-LEARNING-DEBUG - Dropping 99 outliers from 'thinness 5-9 years'
2022-06-08 10:30:46,012 - REGRESSION-LEARNING-DEBUG - Dropping 132 outliers from 'Income composition of resources'
2022-06-08 10:30:46,012 - REGRESSION-LEARNING-DEBUG - Dropping 40 outliers from 'Schooling'
2022-06-08 10:30:46,029 - REGRESSION-LEARNING-INFO  - Total outliers removed are 352
2022-06-08 10:30:46,030 - REGRESSION-LEARNING-INFO  - One hot encoding discrete columns which has < 8 unique values
2022-06-08 10:30:46,030 - REGRESSION-LEARNING-DEBUG - Number of unique values of Country are 187
2022-06-08 10:30:46,031 - REGRESSION-LEARNING-DEBUG - Number of unique values of Year are 16
2022-06-08 10:30:46,031 - REGRESSION-LEARNING-DEBUG - Number of unique values of Status are 2
2022-06-08 10:30:46,031 - REGRESSION-LEARNING-INFO  - Encoding 'Status' column
2022-06-08 10:30:46,039 - REGRESSION-LEARNING-INFO  - Variance ratio : [0.96773336]
2022-06-08 10:30:46,040 - REGRESSION-LEARNING-INFO  - Formulating R2 using PCA1
2022-06-08 10:30:46,406 - REGRESSION-LEARNING-INFO  - ========= Method: PCA2 =========
2022-06-08 10:30:46,406 - REGRESSION-LEARNING-INFO  - Reducing dimensionality using PCA
```

6c. R2 Report

```
2022-06-08 10:30:48,139 - REGRESSION-LEARNING-INFO  - Formulating R2 using PCA5
2022-06-08 10:30:48,557 - REGRESSION-LEARNING-INFO  - ==================  R2 Analysis Report  ==================
     Random State  Mode_or_Median  KNNImputer  Normalisation  Standardisation  PCA1  PCA2  PCA3  PCA4  PCA5
0               5            0.82        0.84           0.84             0.84  0.45  0.58  0.63  0.63  0.63
1               7            0.80        0.83           0.83             0.83  0.50  0.59  0.70  0.70  0.70
2               9            0.82        0.84           0.84             0.84  0.45  0.59  0.67  0.67  0.67
3              11            0.81        0.82           0.82             0.82  0.51  0.58  0.68  0.68  0.68
4              13            0.80        0.82           0.82             0.82  0.40  0.47  0.54  0.55  0.55
..            ...             ...         ...            ...              ...   ...   ...   ...   ...   ...
243           491            0.83        0.84           0.84             0.84  0.44  0.58  0.66  0.66  0.66
244           493            0.81        0.82           0.82             0.82  0.30  0.48  0.55  0.56  0.56
245           495            0.83        0.85           0.85             0.85  0.38  0.52  0.59  0.60  0.60
246           497            0.80        0.81           0.81             0.81  0.37  0.49  0.56  0.56  0.57
247           499            0.79        0.80           0.80             0.80  0.55  0.62  0.68  0.68  0.69

[248 rows x 10 columns]
2022-06-08 10:30:48,563 - REGRESSION-LEARNING-INFO  - ==================  Highest R2 Values ==================
       Method Used  Random State  Max R2
0   Mode_or_Median           269    0.86
1       KNNImputer            31    0.86
2    Normalisation            31    0.86
3  Standardisation            31    0.86
4             PCA1           351    0.61
5             PCA2           255    0.68
6             PCA3            87    0.76
7             PCA4            87    0.76
8             PCA5            87    0.76
--Return--
```

DEMO

Our Learnings

1. KNNImputer / Normalisation / Standardisation alone gets effective R2 score. These can be used when dealing with less columns
2. Reducing dimensions using PCA might affect R2 score
3. Got negative R2 score in PCA1 method, this occurs when data has more outliers columns.
4. Negative R2 also comes when we delete more outliers rows from data.
5. Figured out list of config parameters which affect the R2 score for each method

Future Scopes

1. Optimise script to support all kinds of data
2. Have a config file as an input to the script
3. Create different tabular data for different config parameters

Code Reference: https://github.com/sreeram514/AIML_Project

Q&A