Mapping Malignancies:

A Prediction Model for Estimating Cancer Rate Categories Across Countries

John Pauline Pineda

November 27, 2023

OUTLINE

- Executive Summary
- Introduction
- Methodology
- Results Summary
  - Data Gathering and Description
  - Data Quality Assessment
  - Data Preprocessing
  - Data Exploration
  - Model Development and Validation
- Detailed Findings
- Discussion
  - Overall Findings and Implications
- Conclusion
- Appendix

EXECUTIVE SUMMARY

- ## Study Objective
  - Conduct data analysis and modeling to investigate the potential drivers for high cancer rates between countries given various world performance indicators (social protection and labor, education, economy and growth, environment, climate change, agricultural and rural development, social development, health, science and technology, urban development) and indices (human development, environmental performance).
- ## Methodology and Tools
  - Data Quality Assessment using Python Pandas and NumPy APIs
  - Data Preprocessing using Python Pandas and Scikit-Learn APIs
  - Data Exploration using Python Matplotlib, Seaborn and SciPy APIs
  - Model Development and Validation using Python Scikit-Learn and Imbalanced-Learn APIs
- ## Overall Findings
  - Data quality issues were identified and handled with the appropriate data preprocessing methods.
  - Relevant numeric indicators which highly correlated with cancer rates were determined.
  - Relevant categorical indices that effectively differentiated varying levels of cancer rates were determined.
  - Cancer rate categories were modeled with actions applied to address class imbalance.

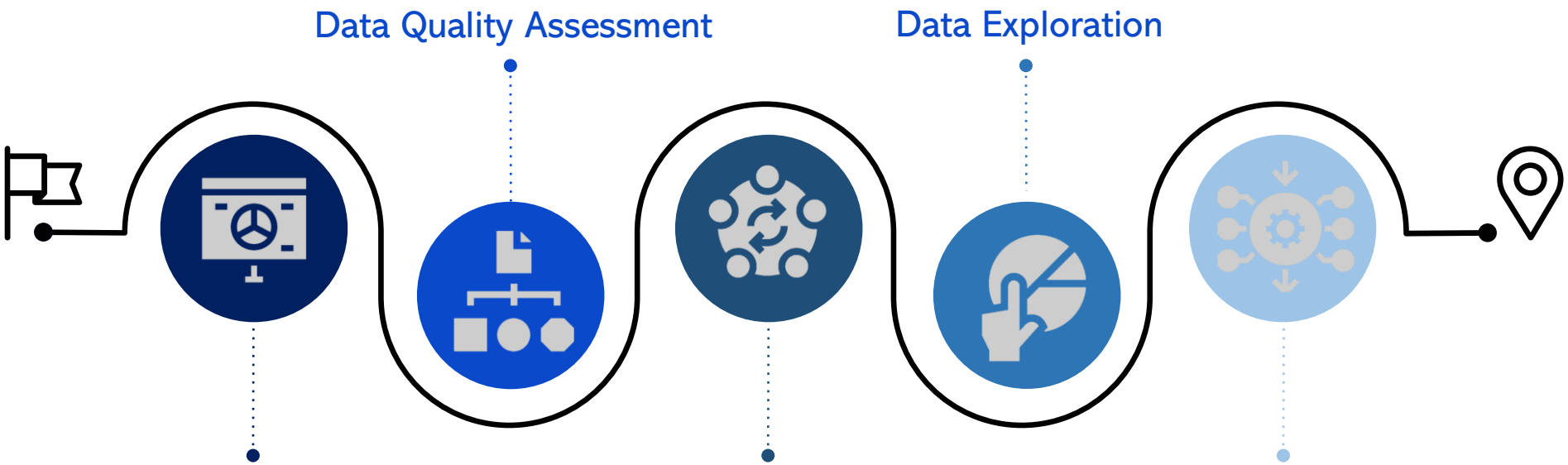
INTRODUCTION

- Cancer – a complex group of diseases characterized by the uncontrolled growth and spread of abnormal cells, has emerged as a leading cause of morbidity and mortality worldwide.
- While the prevalence of cancer varies significantly between countries, anticipating elevated cancer categories given the factors that contribute to these variations is crucial for effective prevention, early detection, and treatment strategies.
- This capstone project generally aims to develop a classification model that could provide robust and reliable estimates of cancer category probabilities from an optimal set of observations and predictors, while delivering accurate predictions when applied to new unseen data.
    - In particular, multiple classification models will be formulated with remedial actions applied to address class imbalance. The final classification model will be selected among candidates based on robust performance estimates. The final model performance and generalization ability will be evaluated through external validation in an independent set.

Section 1

Methodology

METHODOLOGY



- Source assessment
- Data download
- Variable description
- Data dimension
- Column types
- Numeric statistics
- Categorical statistics

- Row duplicates
- Column fill rate
- Row fill rate
- Low variance
- High data skew

- Data cleaning
- Data imputation
- Outlier treatment
- Collinearity
- Transformation
- Centering and scaling
- Data Encoding

- Visual exploration
- Hypothesis testing

- Model development
- Hyperparameter tuning
- Class weighting
- Upsampling (SMOTE)
- Downsampling (CNN)
- Model validation
- Model stacking
- Model selection

Section 2

Results Summary

RESULTS – DATA GATHERING

World Performance Indicators

Social Protection and Labor [Source: World Bank]
Education [Source: World Bank]
Economy and Growth [Source: World Bank]
Environment [Source: World Bank]
Climate Change [Source: World Bank]
Agricultural and Rural Development [Source: World Bank]
Social Development [Source: World Bank]
Health [Source: World Bank]
Science and Technology [Source: World Bank]
Urban Development [Source: World Bank]

World Performance Indices

Human Development [Source: Human Development Reports]
Environmental Performance [Source: Yale Center]

- The study hypothesizes that **world performance indicators** (social protection and labor, education, economy and growth, environment, climate change, agricultural and rural development, social development, health, science and technology, urban development) and **indices** (human development, environmental performance) directly influence **cancer rates** across countries.

World Cancer Rates
[Source: World Population Review]

- The objective of the study is to explore and prepare the dataset to determine which among the **world performance indicators** and **indices** are the significant key drivers of **cancer rates** across countries.

RESULTS – DATA DESCRIPTION

Country 177 Unique Values	COUNTRY		
GDP Per Person Employed (USD) Min: 1.7K | Mean: 45.3K | Max: 234.6K	GDPPER	Human Development Index Category 4 Categories	HDICAT
Urban Population (% of Total) Min: 13.3 | Mean: 59.8 | Max: 100.0	URBPOP	Environmental Performance Index (Score) Min: 18.9 | Mean: 42.9 | Max: 77.9	EPISCO
Patent Applications by Residents (Count) Min: 1.0 | Mean: 20.6K | Max: 1344.8K	PATRES	GDP per Capita (USD) Min: 0.2K | Mean: 13.9K | Max: 117.3K	GDPCAP
R&D Expenditure (% of GDP) Min: 0.1 | Mean: 1.2 | Max: 5.3	RNDGDP	Tertiary School Enrollment (% Gross) Min: 2.4 | Mean: 50.0 | Max: 143.3	ENRTER
Annual Population Growth (%) Min: -2.1 | Mean: +1.1 | Max: +3.7	POPGRO	Population Density (Persons per Km^2) Min: 2.1 | Mean: 200.8 | Max: 7918.9	POPDEN
Life Expectancy at Birth (Years) Min: 52.7 | Mean: 71.7 | Max: 84.5	LIFEXP	PM2.5 Air Pollution Exposure (% of Total) Min: 0.3 | Mean: 91.9 | Max: 100.0	PM2EXP
Tuberculosis Incidence per 100K (Count) Min: 0.7 | Mean: 105.0 | Max: 592.0	TUBINC	CO2 Emissions (Metric Tons per Capita) Min: 0.0 | Mean: 3.7 | Max: 31.7	CO2EMI
Death by Communicable Disease (% of Total) Min: 1.3 | Mean: 21.3 | Max: 65.2	DTHCMD	Forest Area (% of Land Area) Min: 0.0 | Mean: 32.2 | Max: 97.4	FORARE
Agricultural Land (% of Land Area) Min: 0.5 | Mean: 38.8 | Max: 80.8	AGRLND	Methane Emissions (Kiloton) Min: 0.0K | Mean: 47.8K | Max: 1186.2K	METEMI
Greenhouse Gas Emissions (Kiloton) Min: 0.1K | Mean: 259.5K | Max: 12,942.8K	GHGEMI	Renewable Electricity Output (% of Total) Min: 0.0 | Mean: 39.7 | Max: 100.0	RELOUT

- Original data consisted of:
  - 177 observation rows
  - 1 categorical response column
  - 19 numeric predictor columns
  - 1 categorical predictor column
  - 1 object metadata column
- Numeric data are of different scales.

Target

CANRAT — Cancer Rate Category
2 Unique Values | Low 25% | High 75%

21 Predictors + Metadata

177 Observations

- Categorical Target
- Numeric Predictor
- Categorical Predictor
- Object Metadata

RESULTS – DATA QUALITY ASSESSMENT

Country	COUNTRY		
GDP Per Person Employed (USD) MIS | MUL | OUT	GDPPER	Human Development Index Category MIS	HDICAT
Urban Population (% of Total) MIS	URBPOP	Environmental Performance Index (Score) MIS | OUT	EPISCO
Patent Applications by Residents (Count) MIS | SKW	PATRES	GDP per Capita (USD) MIS | MUL | OUT	GDPCAP
R&D Expenditure (% of GDP) MIS	RNDGDP	Tertiary School Enrollment (% Gross) MIS	ENRTER
Annual Population Growth (%) MIS	POPGRO	Population Density (Persons per Km^2) MIS | SKW | OUT	POPDEN
Life Expectancy at Birth (Years) MIS	LIFEXP	PM2.5 Air Pollution Exposure (% of Total) MIS | NZV | SKW | OUT	PM2EXP
Tuberculosis Incidence per 100K (Count) MIS | OUT	TUBINC	CO2 Emissions (Metric Tons per Capita) MIS | OUT	CO2EMI
Death by Communicable Disease (% of Total) MIS	DTHCMD	Forest Area (% of Land Area) MIS	FORARE
Agricultural Land (% of Land Area) MIS	AGRLND	Methane Emissions (Kiloton) MIS | SKW | MUL | OUT	METEMI
Greenhouse Gas Emissions (Kiloton) MIS | SKW | MUL | OUT	GHGEMI	Renewable Electricity Output (% of Total) MIS	RELOUT

- Data quality issues identified included:
  - Missing data
  - Skewed distributions
  - Near zero variance
  - High multicollinearity
  - High outlier ratio
- Data preprocessing is needed to address data quality issues.

Target

CANRAT — Cancer Rate Category

21 Predictors + Metadata

177 Observations

- Categorical Target
- Numeric Predictor
- Categorical Predictor
- Object Metadata

MIS High Missing Data
SKW Skewed Distribution
NZV Near-Zero Variance
MUL High Multicollinearity
OUT High Outlier Ratio

RESULTS – DATA PREPROCESSING

Predictors + Metadata	DATA PREPROCESSING STEPS	Observations
21		177
17	High Column-Wise Missing Data[A]: Column Fill Rate<90% | High Row-Wise Missing Data[B]: Row Missing Rate>20%	163
17	Data Imputation: Iterative Imputation for Low Column-Wise Missing Data	163
15	Multicollinearity Detection: Pairwise Correlation Coefficient>0.90[C]	163
15	Skewness Detection | Outlier Detection: Skewness>|3|, Value>75th Percentile+1.5 IQR, Value<25th Percentile-1.5 IQR	163
15	Shape Transformation: Yeo-Johnson Transformation for Skewness and Outlier Reduction	163
14	Post-Transformation Outlier | Skewness Detection: Outlier Count>25[D], Skewness>|3|[D]	163
17	Centering | Scaling: Standardization for Comparable Range | Data Encoding: One-Hot Encoding for Categorical Column	163

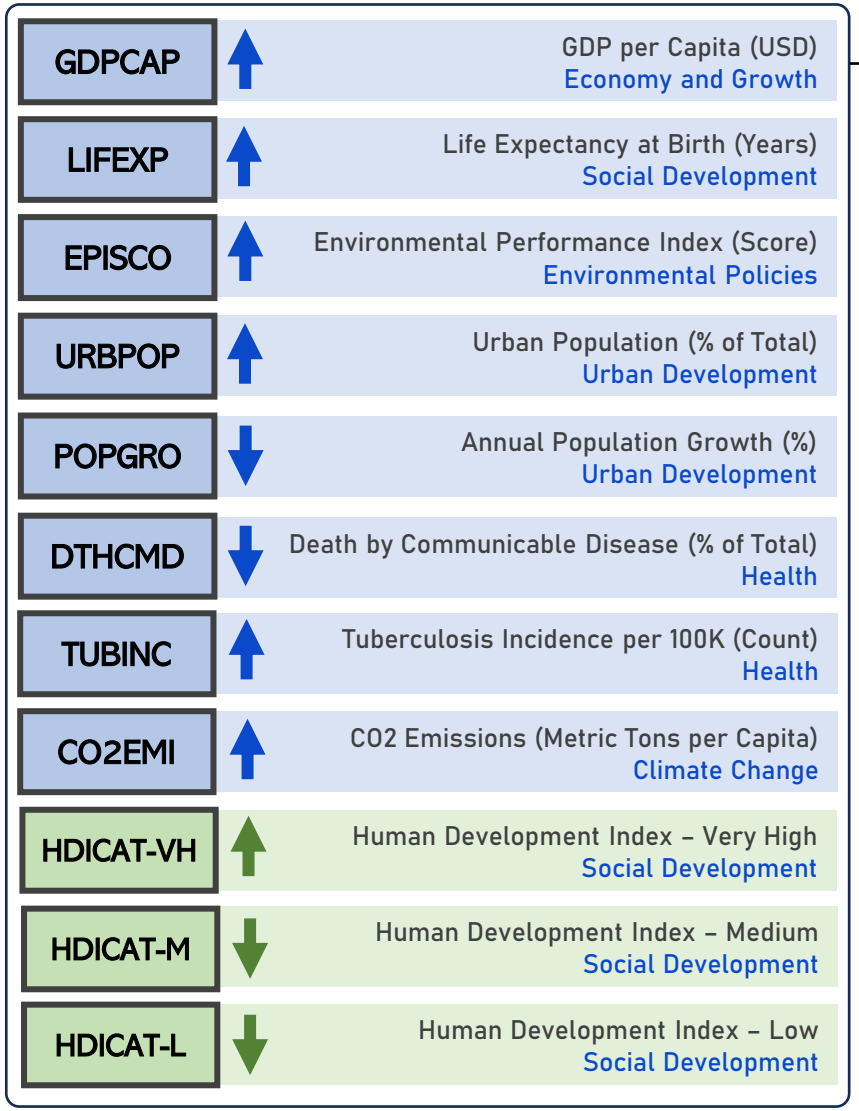
Removed 7 Predictors

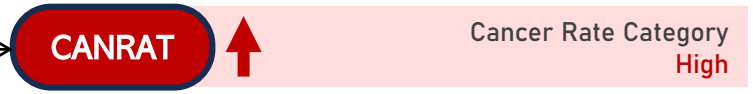
- [A]RNDGDP
- [A]PATRES
- [A]ENRTER
- [A]RELOUT
- [C]GDPPER
- [C]METEMI
- [D]PM2EXP

Removed 14 Observations

[B]COUNTRY: Guadeloupe, Martinique, French Guiana, New Caledonia, French Polynesia, Guam, Puerto Rico, North Korea, Somalia, South Sudan, Venezuela, Libya, Eritrea, Yemen

RESULTS – DATA EXPLORATION

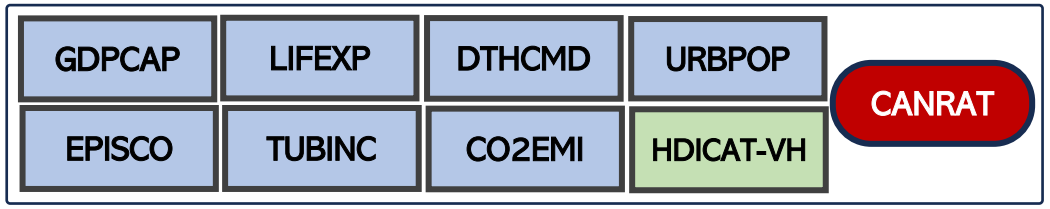




Statistically significant associations with predictors were determined.

- Cancer rates categorized as high were generally observed among progressive countries as characterized by the following:
  - Higher economic growth leading to advanced industrialization with increased exposure to pollutants but potentially better environmental policies
  - Better socioeconomic policies to support cancer screening and diagnosis
  - More robust healthcare infrastructure to support the completeness of cancer registries and reporting mechanisms
  - Shifting demographics including a potentially aging population with increased disease incidence trends
- Given the findings, a high level of progressiveness may not inherently imply higher cancer rates; rather, the relationship might have been influenced by a combination of demographic, environmental, and healthcare-related factors associated with developed countries.
- The relationship between a country's level of progressiveness and cancer rates may not be straightforward, and other potential drivers must be considered in the future when exploring this issue including data on lifestyle factors (smoking rates, obesity levels) or genetic diversity (infectious agents).

RESULTS – MODEL DEVELOPMENT



Predictors + Metadata: 8 → 8

Observations: 163 → A 114 / B 49

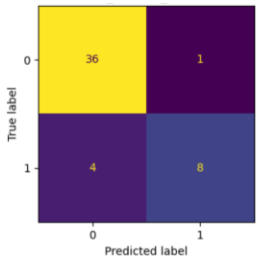
A 70% for Model Training

B 30% for Model External Validation (EV)

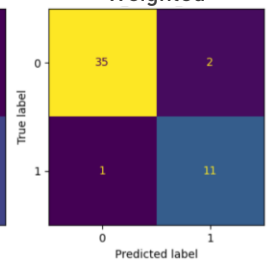
C 5-Fold Cross-Validation for Model Hyperparameter Tuning
5 Cycles
4 Folds=Train
1 Fold=Test

Logistic Regression

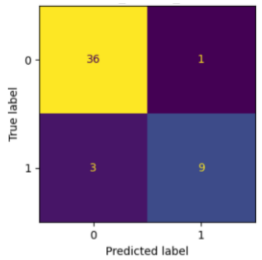
Hyperparameter Tuned C



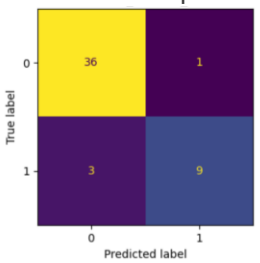
Class Weighted C D



SMOTE Upsampled C

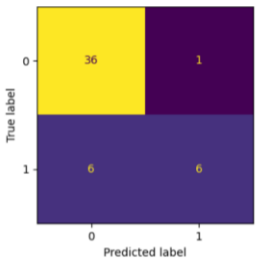


CNN Downsampled C

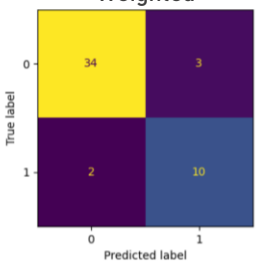


Decision Tree

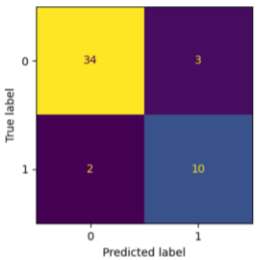
Hyperparameter Tuned C



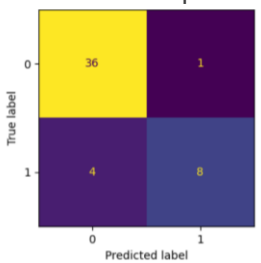
Class Weighted C



SMOTE Upsampled C

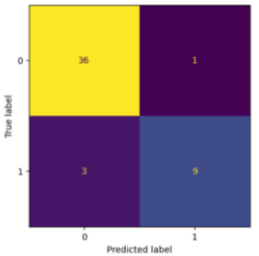


CNN Downsampled C



Stacked

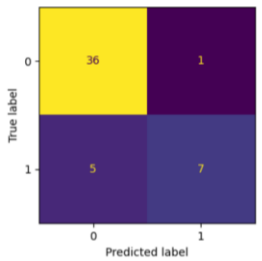
Ensemble Learned



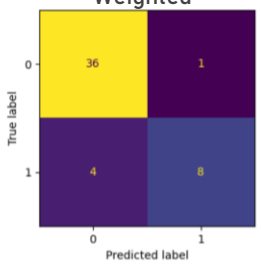
Meta-Learner
Logistic Regression
+
Base-Learners
SMOTE Upsampled - Logistic Regression
Decision Tree
Random Forest
Support Vector Machine

Random Forest

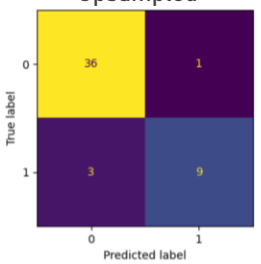
Hyperparameter Tuned C



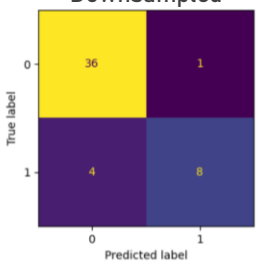
Class Weighted C



SMOTE Upsampled C

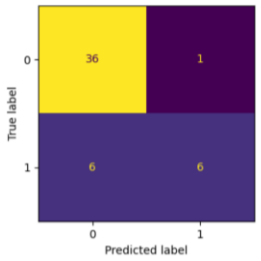


CNN Downsampled C

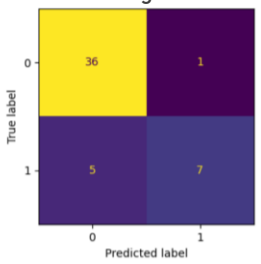


Support Vector Machine

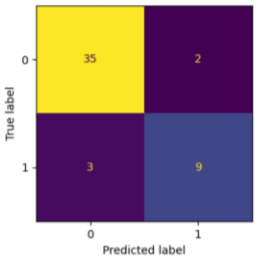
Hyperparameter Tuned C



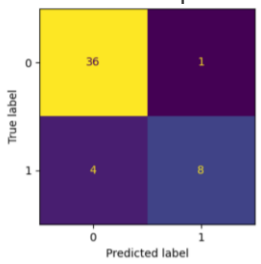
Class Weighted C



SMOTE Upsampled C



CNN Downsampled C



D FINAL MODEL: Demonstrated the best EV F1 score of 88.00% – with no excessive overfitting comparing the Apparent A and EV B performance.

Section 3

Detailed Findings

RESULTS – DATA QUALITY ASSESSMENT

- **Findings** (GitHub URL: Python Notebook | Markdown Presentation)

1. No duplicated rows observed.
2. Missing data noted for 20 variables with Null.Count>0 and Fill.Rate<1.0.
   - RNDGDP: Null.Count = 103, Fill.Rate = 0.418
   - PATRES: Null.Count = 69, Fill.Rate = 0.610
   - ENRTER: Null.Count = 61, Fill.Rate = 0.655
   - RELOUT: Null.Count = 24, Fill.Rate = 0.864
   - GDPPER: Null.Count = 12, Fill.Rate = 0.932
   - EPISCO: Null.Count = 12, Fill.Rate = 0.932
   - HDICAT: Null.Count = 10, Fill.Rate = 0.943
   - PM2EXP: Null.Count = 10, Fill.Rate = 0.943
   - DTHCMD: Null.Count = 7, Fill.Rate = 0.960
   - METEMI: Null.Count = 7, Fill.Rate = 0.960
   - CO2EMI: Null.Count = 7, Fill.Rate = 0.960
   - GDPCAP: Null.Count = 7, Fill.Rate = 0.960
   - GHGEMI: Null.Count = 7, Fill.Rate = 0.960
   - FORARE: Null.Count = 4, Fill.Rate = 0.977
   - TUBINC: Null.Count = 3, Fill.Rate = 0.983
   - AGRLND: Null.Count = 3, Fill.Rate = 0.983
   - POPGRO: Null.Count = 3, Fill.Rate = 0.983
   - POPDEN: Null.Count = 3, Fill.Rate = 0.983
   - URBPOP: Null.Count = 3, Fill.Rate = 0.983
   - LIFEXP: Null.Count = 3, Fill.Rate = 0.983
3. 120 observations noted with at least 1 missing data. From this number, 14 observations reported high Missing.Rate>0.2.
   - COUNTRY=Guadeloupe: Missing.Rate= 0.909
   - COUNTRY=Martinique: Missing.Rate= 0.909
   - COUNTRY=French Guiana: Missing.Rate= 0.909
   - COUNTRY=New Caledonia: Missing.Rate= 0.500
   - COUNTRY=French Polynesia: Missing.Rate= 0.500
   - COUNTRY=Guam: Missing.Rate= 0.500
   - COUNTRY=Puerto Rico: Missing.Rate= 0.409
   - COUNTRY=North Korea: Missing.Rate= 0.227
   - COUNTRY=Somalia: Missing.Rate= 0.227
   - COUNTRY=South Sudan: Missing.Rate= 0.227
   - COUNTRY=Venezuela: Missing.Rate= 0.227
   - COUNTRY=Libya: Missing.Rate= 0.227
   - COUNTRY=Eritrea: Missing.Rate= 0.227
   - COUNTRY=Yemen: Missing.Rate= 0.227
4. Low variance observed for 1 variable with First.Second.Mode.Ratio>5.
   - PM2EXP: First.Second.Mode.Ratio = 53.000
5. No low variance observed for any variable with Unique.Count.Ratio>10.
6. High skewness observed for 5 variables with Skewness>3 or Skewness<(-3).
   - POPDEN: Skewness = +10.267
   - GHGEMI: Skewness = +9.496
   - PATRES: Skewness = +9.284
   - METEMI: Skewness = +5.801
   - PM2EXP: Skewness = -3.141

RESULTS – DATA CLEANING

- **Findings** (GitHub URL: Python Notebook | Markdown Presentation)

1. Subsets of rows and columns with high rates of missing data were removed from the dataset:
   - 4 variables with Fill.Rate<0.9 were excluded for subsequent analysis.
     - RNDGDP: Null.Count = 103, Fill.Rate = 0.418
     - PATRES: Null.Count = 69, Fill.Rate = 0.610
     - ENRTER: Null.Count = 61, Fill.Rate = 0.655
     - RELOUT: Null.Count = 24, Fill.Rate = 0.864
   - 14 rows with Missing.Rate>0.2 were exluded for subsequent analysis.
     - COUNTRY=Guadeloupe: Missing.Rate= 0.909
     - COUNTRY=Martinique: Missing.Rate= 0.909
     - COUNTRY=French Guiana: Missing.Rate= 0.909
     - COUNTRY=New Caledonia: Missing.Rate= 0.500
     - COUNTRY=French Polynesia: Missing.Rate= 0.500
     - COUNTRY=Guam: Missing.Rate= 0.500
     - COUNTRY=Puerto Rico: Missing.Rate= 0.409
     - COUNTRY=North Korea: Missing.Rate= 0.227
     - COUNTRY=Somalia: Missing.Rate= 0.227
     - COUNTRY=South Sudan: Missing.Rate= 0.227
     - COUNTRY=Venezuela: Missing.Rate= 0.227
     - COUNTRY=Libya: Missing.Rate= 0.227
     - COUNTRY=Eritrea: Missing.Rate= 0.227
     - COUNTRY=Yemen: Missing.Rate= 0.227
2. No variables were removed due to zero or near-zero variance.
3. The cleaned dataset is comprised of:
   - **163 rows** (observations)
   - **18 columns** (variables)
     - **1/18 metadata** (object)
       - COUNTRY
     - **1/18 target** (categorical)
       - CANRAT
     - **15/18 predictor** (numeric)
       - GDPPER
       - URBPOP
       - POPGRO
       - LIFEXP
       - TUBINC
       - DTHCMD
       - AGRLND
       - GHGEMI
       - METEMI
       - FORARE
       - CO2EMI
       - PM2EXP
       - POPDEN
       - GDPCAP
       - EPISCO
     - **1/18 predictor** (categorical)
       - HDICAT

RESULTS – MISSING DATA IMPUTATION

- Findings (GitHub URL: Python Notebook | Markdown Presentation)

1. Missing data for numeric variables were imputed using the iterative imputer algorithm with a linear regression estimator.
    - GDPPER: Null.Count = 1
    - FORARE: Null.Count = 1
    - PM2EXP: Null.Count = 5
2. Missing data for categorical variables were imputed using the most frequent value.
    - HDICAP: Null.Count = 1

RESULTS – OUTLIER TREATMENT

- **Findings** (GitHub URL: Python Notebook | Markdown Presentation)

1. High number of outliers observed for 5 numeric variables with Outlier.Ratio>0.10 and marginal to high Skewness.
   - PM2EXP: Outlier.Count = 37, Outlier.Ratio = 0.226, Skewness=-3.061
   - GHGEMI: Outlier.Count = 27, Outlier.Ratio = 0.165, Skewness=+9.299
   - GDPCAP: Outlier.Count = 22, Outlier.Ratio = 0.134, Skewness=+2.311
   - POPDEN: Outlier.Count = 20, Outlier.Ratio = 0.122, Skewness=+9.972
   - METEMI: Outlier.Count = 20, Outlier.Ratio = 0.122, Skewness=+5.688
2. Minimal number of outliers observed for 5 numeric variables with Outlier.Ratio<0.10 and normal Skewness.
   - TUBINC: Outlier.Count = 12, Outlier.Ratio = 0.073, Skewness=+1.747
   - CO2EMI: Outlier.Count = 11, Outlier.Ratio = 0.067, Skewness=+2.693
   - GDPPER: Outlier.Count = 3, Outlier.Ratio = 0.018, Skewness=+1.554
   - EPISCO: Outlier.Count = 3, Outlier.Ratio = 0.018, Skewness=+0.635
   - CANRAT: Outlier.Count = 2, Outlier.Ratio = 0.012, Skewness=+0.910

RESULTS – COLLINEARITY

- Findings (GitHub URL: Python Notebook | Markdown Presentation)

1. Majority of the numeric variables reported moderate to high correlation which were statistically significant.
2. Among pairwise combinations of numeric variables, high Pearson.Correlation.Coefficient values were noted for:
    - GDPPER and GDPCAP: Pearson.Correlation.Coefficient = +0.921
    - GHGEMI and METEMI: Pearson.Correlation.Coefficient = +0.905
3. Among the highly correlated pairs, variables with the lowest correlation against the target variable were removed.
    - GDPPER: Pearson.Correlation.Coefficient = +0.690
    - METEMI: Pearson.Correlation.Coefficient = +0.062
4. The cleaned dataset is comprised of:
    - **163 rows** (observations)
    - **16 columns** (variables)
        - **1/16 metadata** (object)
            - COUNTRY
        - **1/16 target** (categorical)
            - CANRAT
        - **13/16 predictor** (numeric)
            - URBPOP
            - POPGRO
            - LIFEXP
            - TUBINC
            - DTHCMD
            - AGRLND
            - GHGEMI
            - FORARE
            - CO2EMI
            - PM2EXP
            - POPDEN
            - GDPCAP
            - EPISCO
        - **1/16 predictor** (categorical)
            - HDICAT

RESULTS – SHAPE TRANSFORMATION

- ## Findings (GitHub URL: Python Notebook | Markdown Presentation)

1. A Yeo-Johnson transformation was applied to all numeric variables to improve distributional shape.
2. Most variables achieved symmetrical distributions with minimal outliers after transformation.
3. One variable which remained skewed even after applying shape transformation was removed.
    - PM2EXP

4. The transformed dataset is comprised of:
    - **163 rows** (observations)
    - **15 columns** (variables)
        - **1/15 metadata** (object)
            - COUNTRY
        - **1/15 target** (categorical)
            - CANRAT
        - **12/15 predictor** (numeric)
            - URBPOP
            - POPGRO
            - LIFEXP
            - TUBINC
            - DTHCMD
            - AGRLND
            - GHGEMI
            - FORARE
            - CO2EMI
            - POPDEN
            - GDPCAP
            - EPISCO
        - **1/15 predictor** (categorical)
            - HDICAT

RESULTS – CENTERING AND SCALING

- ## Findings (GitHub URL: Python Notebook | Markdown Presentation)

1. All numeric variables were transformed using the standardization method to achieve a comparable scale between values.
2. The scaled dataset is comprised of:
    - **163 rows** (observations)
    - **15 columns** (variables)
        - **1/15 metadata** (object)
            - COUNTRY
        - **1/15 target** (categorical)
            - CANRAT
        - **12/15 predictor** (numeric)
            - URBPOP
            - POPGRO
            - LIFEXP
            - TUBINC
            - DTHCMD
            - AGRLND
            - GHGEMI
            - FORARE
            - CO2EMI
            - POPDEN
            - GDPCAP
            - EPISCO
        - **1/15 predictor** (categorical)
            - HDICAT

RESULTS – DATA ENCODING

- **Findings** (GitHub URL: Python Notebook | Markdown Presentation)

1. One-hot encoding was applied to the HDICAP_VH variable resulting to 4 additional columns in the dataset:
   - HDICAP_L
   - HDICAP_M
   - HDICAP_H
   - HDICAP_VH

RESULTS – EXPLORATORY DATA ANALYSIS

- **Findings** (GitHub URL: Python Notebook | Markdown Presentation)

1. Bivariate analysis identified individual predictors with generally positive association to the target variable based on visual inspection.
2. Higher values or higher proportions for the following predictors are associated with the CANRAT HIGH category:
   - URBPOP
   - LIFEXP
   - CO2EMI
   - GDPCAP
   - EPISCO
   - HDICAP_VH=1
3. Decreasing values or smaller proportions for the following predictors are associated with the CANRAT LOW category:
   - POPGRO
   - TUBINC
   - DTHCMD
   - HDICAP_L=0
   - HDICAP_M=0
   - HDICAP_H=0
4. Values for the following predictors are not associated with the CANRAT HIGH or LOW categories:
   - AGRLND
   - GHGEMI
   - FORARE
   - POPDEN

RESULTS – HYPOTHESIS TESTING

- ## Findings (GitHub URL: Python Notebook | Markdown Presentation)

1. The relationship between the numeric predictors to the CANRAT target variable was statistically evaluated using the following hypotheses:
   - **Null**: Difference in the means between groups LOW and HIGH is equal to zero
   - **Alternative**: Difference in the means between groups LOW and HIGH is not equal to zero
2. There is sufficient evidence to conclude of a statistically significant difference between the means of the numeric measurements obtained from LOW and HIGH groups of the CANRAT target variable in 9 of the 12 numeric predictors given their high t-test statistic values with reported low p-values less than the significance level of 0.05.
   - GDPCAP: T.Test.Statistic=-11.937, T.Test.PValue=0.000
   - EPISCO: T.Test.Statistic=-11.789, T.Test.PValue=0.000
   - LIFEXP: T.Test.Statistic=-10.979, T.Test.PValue=0.000
   - TUBINC: T.Test.Statistic=+9.609, T.Test.PValue=0.000
   - DTHCMD: T.Test.Statistic=+8.376, T.Test.PValue=0.000
   - CO2EMI: T.Test.Statistic=-7.031, T.Test.PValue=0.000
   - URBPOP: T.Test.Statistic=-6.541, T.Test.PValue=0.000
   - POPGRO: T.Test.Statistic=+4.905, T.Test.PValue=0.000
   - GHGEMI: T.Test.Statistic=-2.243, T.Test.PValue=0.026
3. The relationship between the categorical predictors to the CANRAT target variable was statistically evaluated using the following hypotheses:
   - **Null**: The categorical predictor is independent of the categorical target variable
   - **Alternative**: The categorical predictor is dependent of the categorical target variable
4. There is sufficient evidence to conclude of a statistically significant relationship difference between the categories of the categorical predictors and the LOW and HIGH groups of the CANRAT target variable in all 4 categorical predictors given their high chisquare statistic values with reported low p-values less than the significance level of 0.05.
   - HDICAT_VH: ChiSquare.Test.Statistic=76.764, ChiSquare.Test.PValue=0.000
   - HDICAT_H: ChiSquare.Test.Statistic=13.860, ChiSquare.Test.PValue=0.000
   - HDICAT_M: ChiSquare.Test.Statistic=10.286, ChiSquare.Test.PValue=0.001
   - HDICAT_L: ChiSquare.Test.Statistic=9.081, ChiSquare.Test.PValue=0.002

RESULTS – PRE-MODELLING DATA ANALYSIS

- **Findings** (GitHub URL: Python Notebook | Markdown Presentation)

1. Among the 9 numeric variables determined to have a statistically significant difference between the means of the numeric measurements obtained from LOW and HIGH groups of the CANRAT target variable, only 7 were retained with absolute T-Test statistics greater than 5.
   - GDPCAP: T.Test.Statistic=-11.937, T.Test.PValue=0.000
   - EPISCO: T.Test.Statistic=-11.789, T.Test.PValue=0.000
   - LIFEXP: T.Test.Statistic=-10.979, T.Test.PValue=0.000
   - TUBINC: T.Test.Statistic=+9.609, T.Test.PValue=0.000
   - DTHCMD: T.Test.Statistic=+8.376, T.Test.PValue=0.000
   - CO2EMI: T.Test.Statistic=-7.031, T.Test.PValue=0.000
   - URBPOP: T.Test.Statistic=-6.541, T.Test.PValue=0.000
2. Among the 4 categorical predictors determined to have a statistically significant relationship difference between the categories of the categorical predictors and the LOW and HIGH groups of the CANRAT target variable, only 1 was retained with absolute Chi-Square statistics greater than 15.
   - HDICAT_VH: ChiSquare.Test.Statistic=76.764, ChiSquare.Test.PValue=0.000
3. The original data which reflect a 3:1 class imbalance between the LOW and HIGH CANRAT categories was used for model training and testing.

RESULTS – TUNED LOGISTIC REGRESSION

- ## Findings (GitHub URL: Python Notebook | Markdown Presentation)

1. The logistic regression model from the **sklearn.linear_model** Python library API was implemented.
2. The model contains 5 hyperparameters:
   - C = inverse of regularization strength held constant at a value of 1
   - penalty = penalty norm made to vary between L1 and L2
   - solver = algorithm used in the optimization problem made to vary between Saga and Liblinear
   - class_weight = weights associated with classes held constant at a value of None
   - max_iter = maximum number of iterations taken for the solvers to converge held constant at a value of 500
3. The original data which reflect a 3:1 class imbalance between the LOW and HIGH CANRAT categories was used for model training and testing.
4. Hyperparameter tuning was conducted using the 5-fold cross-validation method with optimal model performance using the F1 score determined for:
   - C = 1
   - penalty = L1 norm
   - solver = Liblinear
   - class_weight = None
   - max_iter = 500
5. The apparent model performance of the optimal model is summarized as follows:
   - **Accuracy** = 0.9473
   - **Precision** = 0.8709
   - **Recall** = 0.9310
   - **F1 Score** = 0.9000
   - **AUROC** = 0.9419
6. The independent test model performance of the final model is summarized as follows:
   - **Accuracy** = 0.8979
   - **Precision** = 0.8889
   - **Recall** = 0.6667
   - **F1 Score** = 0.7619
   - **AUROC** = 0.8198
7. High difference in the apparent and independent test model performance observed, indicative of the presence of excessive model overfitting.

RESULTS – TUNED DECISION TREE

- ## Findings (GitHub URL: Python Notebook | Markdown Presentation)

1. The decision tree model from the **sklearn.tree** Python library API was implemented.
2. The model contains 5 hyperparameters:
   - criterion = function to measure the quality of a split made to vary between Gini, Entropy and Log-Loss
   - max_depth = maximum depth of the tree made to vary between 3, 5 and 7
   - min_samples_leaf = minimum number of samples required to split an internal node made to vary between 3, 5 and 10
   - class_weight = weights associated with classes held constant at a value of None
3. The original data which reflect a 3:1 class imbalance between the LOW and HIGH CANRAT categories was used for model training and testing.
4. Hyperparameter tuning was conducted using the 5-fold cross-validation method with optimal model performance using the F1 score determined for:
   - criterion = Entropy
   - max_depth = 5
   - min_samples_leaf = 3
   - class_weight = None
5. The apparent model performance of the optimal model is summarized as follows:
   - **Accuracy** = 0.9736
   - **Precision** = 1.0000
   - **Recall** = 0.8965
   - **F1 Score** = 0.9454
   - **AUROC** = 0.9482
6. The independent test model performance of the final model is summarized as follows:
   - **Accuracy** = 0.8571
   - **Precision** = 0.8571
   - **Recall** = 0.5000
   - **F1 Score** = 0.6315
   - **AUROC** = 0.7364
7. High difference in the apparent and independent test model performance observed, indicative of the presence of excessive model overfitting.

RESULTS – TUNED RANDOM FOREST

- ## Findings (GitHub URL: Python Notebook | Markdown Presentation)

1. The random forest model from the **sklearn.ensemble** Python library API was implemented.
2. The model contains 5 hyperparameters:
   - criterion = function to measure the quality of a split made to vary between Gini, Entropy and Log-Loss
   - max_depth = maximum depth of the tree made to vary between 3, 5 and 7
   - min_samples_leaf = minimum number of samples required to split an internal node made to vary between 3, 5 and 10
   - n_estimators = number of trees in the forest made to vary between 100, 150 and 200
   - max_features = number of features to consider when looking for the best split made to vary between Sqrt and Log2 of n_estimators
   - class_weight = weights associated with classes held constant at a value of None
3. The original data which reflect a 3:1 class imbalance between the LOW and HIGH CANRAT categories was used for model training and testing.
4. Hyperparameter tuning was conducted using the 5-fold cross-validation method with optimal model performance using the F1 score determined for:
   - criterion = Gini
   - max_depth = 3
   - min_samples_leaf = 3
   - n_estimators = 100
   - max_features = Sqrt n_estimators
   - class_weight = None
5. The apparent model performance of the optimal model is summarized as follows:
   - **Accuracy** = 0.9561
   - **Precision** = 0.9285
   - **Recall** = 0.8965
   - **F1 Score** = 0.9122
   - **AUROC** = 0.9365
6. The independent test model performance of the final model is summarized as follows:
   - **Accuracy** = 0.8775
   - **Precision** = 0.8750
   - **Recall** = 0.5833
   - **F1 Score** = 0.7000
   - **AUROC** = 0.7781
7. High difference in the apparent and independent test model performance observed, indicative of the presence of excessive model overfitting.

RESULTS – TUNED SVM

- ## Findings (GitHub URL: Python Notebook | Markdown Presentation)

1. The support vector machine model from the **sklearn.svm** Python library API was implemented.
2. The model contains 5 hyperparameters:
   - C = inverse of regularization strength held constant at a value of 1
   - kernel = kernel type to be used in the algorithm made to vary between Linear, Poly, RBF and Sigmoid
   - class_weight = weights associated with classes held constant at a value of None
3. The original data which reflect a 3:1 class imbalance between the LOW and HIGH CANRAT categories was used for model training and testing.
4. Hyperparameter tuning was conducted using the 5-fold cross-validation method with optimal model performance using the F1 score determined for:
   - C = 1
   - kernel = Poly
   - class_weight = None
5. The apparent model performance of the optimal model is summarized as follows:
   - **Accuracy** = 0.9473
   - **Precision** = 0.9600
   - **Recall** = 0.8275
   - **F1 Score** = 0.8888
   - **AUROC** = 0.9079
6. The independent test model performance of the final model is summarized as follows:
   - **Accuracy** = 0.8571
   - **Precision** = 0.8571
   - **Recall** = 0.5000
   - **F1 Score** = 0.6315
   - **AUROC** = 0.7364
7. High difference in the apparent and independent test model performance observed, indicative of the presence of excessive model overfitting.

RESULTS – WEIGHTED LOGISTIC REGRESSION

- ## Findings (GitHub URL: Python Notebook | Markdown Presentation)

1. The logistic regression model from the **sklearn.linear_model** Python library API was implemented.
2. The model contains 5 hyperparameters:
   - C = inverse of regularization strength held constant at a value of 1
   - penalty = penalty norm made to vary between L1 and L2
   - solver = algorithm used in the optimization problem made to vary between Saga and Liblinear
   - class_weight = weights associated with classes held constant at a value of 25-75 between classes 0 and 1
   - max_iter = maximum number of iterations taken for the solvers to converge held constant at a value of 500
3. The original data reflecting a 3:1 class imbalance between the LOW and HIGH CANRAT categories was used for model training and testing.
4. Hyperparameter tuning was conducted using the 5-fold cross-validation method with optimal model performance using the F1 score determined for:
   - C = 1
   - penalty = L1 norm
   - solver = Liblinear
   - class_weight = 25-75 between classes 0 and 1
   - max_iter = 500
5. The apparent model performance of the optimal model is summarized as follows:
   - **Accuracy** = 0.8947
   - **Precision** = 0.7073
   - **Recall** = 1.0000
   - **F1 Score** = 0.8285
   - **AUROC** = 0.9294
6. The independent test model performance of the final model is summarized as follows:
   - **Accuracy** = 0.9387
   - **Precision** = 0.8461
   - **Recall** = 0.9167
   - **F1 Score** = 0.8800
   - **AUROC** = 0.9313
7. Considerable difference in the apparent and independent test model performance observed, indicative of the presence of moderate model overfitting.

RESULTS – WEIGHTED DECISION TREE

- **Findings** (GitHub URL: Python Notebook | Markdown Presentation)

1. The decision tree model from the **sklearn.tree** Python library API was implemented.
2. The model contains 5 hyperparameters:
   - criterion = function to measure the quality of a split made to vary between Gini, Entropy and Log-Loss
   - max_depth = maximum depth of the tree made to vary between 3, 5 and 7
   - min_samples_leaf = minimum number of samples required to split an internal node made to vary between 3, 5 and 10
   - class_weight = weights associated with classes held constant at a value of 25-75 between classes 0 and 1
3. The original data which reflect a 3:1 class imbalance between the LOW and HIGH CANRAT categories was used for model training and testing.
4. Hyperparameter tuning was conducted using the 5-fold cross-validation method with optimal model performance using the F1 score determined for:
   - criterion = Gini
   - max_depth = 3
   - min_samples_leaf = 3
   - class_weight = 25-75 between classes 0 and 1
5. The apparent model performance of the optimal model is summarized as follows:
   - **Accuracy** = 0.9736
   - **Precision** = 1.0000
   - **Recall** = 0.8965
   - **F1 Score** = 0.9454
   - **AUROC** = 0.9482
6. The independent test model performance of the final model is summarized as follows:
   - **Accuracy** = 0.8571
   - **Precision** = 0.8571
   - **Recall** = 0.5000
   - **F1 Score** = 0.6315
   - **AUROC** = 0.7364
7. High difference in the apparent and independent test model performance observed, indicative of the presence of excessive model overfitting.

RESULTS – WEIGHTED RANDOM FOREST

- ## Findings (GitHub URL: Python Notebook | Markdown Presentation)

1. The random forest model from the **sklearn.ensemble** Python library API was implemented.
2. The model contains 5 hyperparameters:
   - criterion = function to measure the quality of a split made to vary between Gini, Entropy and Log-Loss
   - max_depth = maximum depth of the tree made to vary between 3, 5 and 7
   - min_samples_leaf = minimum number of samples required to split an internal node made to vary between 3, 5 and 10
   - n_estimators = number of trees in the forest made to vary between 100, 150 and 200
   - max_features = number of features to consider when looking for the best split made to vary between Sqrt and Log2 of n_estimators
   - class_weight = weights associated with classes held constant at a value of 25-75 between classes 0 and 1
3. The original data which reflect a 3:1 class imbalance between the LOW and HIGH CANRAT categories was used for model training and testing.
4. Hyperparameter tuning was conducted using the 5-fold cross-validation method with optimal model performance using the F1 score determined for:
   - criterion = Gini
   - max_depth = 5
   - min_samples_leaf = 3
   - n_estimators = 100
   - max_features = Sqrt n_estimators
   - class_weight = 25-75 between classes 0 and 1
5. The apparent model performance of the optimal model is summarized as follows:
   - **Accuracy** = 0.9736
   - **Precision** = 0.9062
   - **Recall** = 1.0000
   - **F1 Score** = 0.9508
   - **AUROC** = 0.9823
6. The independent test model performance of the final model is summarized as follows:
   - **Accuracy** = 0.8979
   - **Precision** = 0.8888
   - **Recall** = 0.6666
   - **F1 Score** = 0.7619
   - **AUROC** = 0.8198
7. High difference in the apparent and independent test model performance observed, indicative of the presence of excessive model overfitting.

RESULTS – WEIGHTED SVM

- **Findings** (GitHub URL: Python Notebook | Markdown Presentation)

1. The support vector machine model from the **sklearn.svm** Python library API was implemented.
2. The model contains 5 hyperparameters:
   - C = inverse of regularization strength held constant at a value of 1
   - kernel = kernel type to be used in the algorithm made to vary between Linear, Poly, RBF and Sigmoid
   - class_weight = weights associated with classes held constant at a value of 25-75 between classes 0 and 1
3. The original data which reflect a 3:1 class imbalance between the LOW and HIGH CANRAT categories was used for model training and testing.
4. Hyperparameter tuning was conducted using the 5-fold cross-validation method with optimal model performance using the F1 score determined for:
   - C = 1
   - kernel = Poly
   - class_weight = 25-75 between classes 0 and 1
5. The apparent model performance of the optimal model is summarized as follows:
   - **Accuracy** = 0.9649
   - **Precision** = 0.9629
   - **Recall** = 0.8965
   - **F1 Score** = 0.9285
   - **AUROC** = 0.9423
6. The independent test model performance of the final model is summarized as follows:
   - **Accuracy** = 0.8775
   - **Precision** = 0.8750
   - **Recall** = 0.5833
   - **F1 Score** = 0.7000
   - **AUROC** = 0.7781
7. High difference in the apparent and independent test model performance observed, indicative of the presence of excessive model overfitting.

RESULTS – SMOTE + LOGISTIC REGRESSION

- ## Findings (GitHub URL: Python Notebook | Markdown Presentation)

1. The logistic regression model from the **sklearn.linear_model** Python library API was implemented.
2. The model contains 5 hyperparameters:
   - C = inverse of regularization strength held constant at a value of 1
   - penalty = penalty norm made to vary between L1 and L2
   - solver = algorithm used in the optimization problem made to vary between Saga and Liblinear
   - class_weight = weights associated with classes held constant at a value of None
   - max_iter = maximum number of iterations taken for the solvers to converge held constant at a value of 500
3. The extended model training data by upsampling the minority HIGH CANRAT category was used.
4. Hyperparameter tuning was conducted using the 5-fold cross-validation method with optimal model performance using the F1 score determined for:
   - C = 1
   - penalty = L1 norm
   - solver = Saga
   - class_weight = None
   - max_iter = 500
5. The apparent model performance of the optimal model is summarized as follows:
   - **Accuracy** = 0.9649
   - **Precision** = 0.9032
   - **Recall** = 0.9655
   - **F1 Score** = 0.9333
   - **AUROC** = 0.9651
6. The independent test model performance of the final model is summarized as follows:
   - **Accuracy** = 0.9183
   - **Precision** = 0.9000
   - **Recall** = 0.7500
   - **F1 Score** = 0.8181
   - **AUROC** = 0.8614
7. High difference in the apparent and independent test model performance observed, indicative of the presence of excessive model overfitting.

RESULTS – SMOTE + DECISION TREE

- **Findings** (GitHub URL: Python Notebook | Markdown Presentation)

1. The decision tree model from the **sklearn.tree** Python library API was implemented.
2. The model contains 5 hyperparameters:
   - criterion = function to measure the quality of a split made to vary between Gini, Entropy and Log-Loss
   - max_depth = maximum depth of the tree made to vary between 3, 5 and 7
   - min_samples_leaf = minimum number of samples required to split an internal node made to vary between 3, 5 and 10
   - class_weight = weights associated with classes held constant at a value of None
3. The extended model training data by upsampling the minority HIGH CANRAT category was used.
4. Hyperparameter tuning was conducted using the 5-fold cross-validation method with optimal model performance using the F1 score determined for:
   - criterion = Entropy
   - max_depth = 3
   - min_samples_leaf = 5
   - class_weight = None
5. The apparent model performance of the optimal model is summarized as follows:
   - **Accuracy** = 0.9210
   - **Precision** = 0.7631
   - **Recall** = 1.0000
   - **F1 Score** = 0.8656
   - **AUROC** = 0.9470
6. The independent test model performance of the final model is summarized as follows:
   - **Accuracy** = 0.8979
   - **Precision** = 0.7692
   - **Recall** = 0.8333
   - **F1 Score** = 0.8000
   - **AUROC** = 0.8761
7. Considerable difference in the apparent and independent test model performance observed, indicative of the presence of moderate model overfitting.

RESULTS – SMOTE + RANDOM FOREST

- ## Findings (GitHub URL: Python Notebook | Markdown Presentation)

1. The random forest model from the **sklearn.ensemble** Python library API was implemented.
2. The model contains 5 hyperparameters:
   - criterion = function to measure the quality of a split made to vary between Gini, Entropy and Log-Loss
   - max_depth = maximum depth of the tree made to vary between 3, 5 and 7
   - min_samples_leaf = minimum number of samples required to split an internal node made to vary between 3, 5 and 10
   - n_estimators = number of trees in the forest made to vary between 100, 150 and 200
   - max_features = number of features to consider when looking for the best split made to vary between Sqrt and Log2 of n_estimators
   - class_weight = weights associated with classes held constant at a value of None
3. The extended model training data by upsampling the minority HIGH CANRAT category was used.
4. Hyperparameter tuning was conducted using the 5-fold cross-validation method with optimal model performance using the F1 score determined for:
   - criterion = Entropy
   - max_depth = 7
   - min_samples_leaf = 3
   - n_estimators = 100
   - max_features = Sqrt n_estimators
   - class_weight = None
5. The apparent model performance of the optimal model is summarized as follows:
   - **Accuracy** = 0.9912
   - **Precision** = 0.9666
   - **Recall** = 1.0000
   - **F1 Score** = 0.9830
   - **AUROC** = 0.9941
6. The independent test model performance of the final model is summarized as follows:
   - **Accuracy** = 0.9183
   - **Precision** = 0.9000
   - **Recall** = 0.7500
   - **F1 Score** = 0.8181
   - **AUROC** = 0.8614
7. High difference in the apparent and independent test model performance observed, indicative of the presence of excessive model overfitting.

RESULTS – SMOTE + SVM

- ## Findings (GitHub URL: Python Notebook | Markdown Presentation)

1. The support vector machine model from the **sklearn.svm** Python library API was implemented.
2. The model contains 5 hyperparameters:
   - C = inverse of regularization strength held constant at a value of 1
   - kernel = kernel type to be used in the algorithm made to vary between Linear, Poly, RBF and Sigmoid
   - class_weight = weights associated with classes held constant at a value of None
3. The extended model training data by upsampling the minority HIGH CANRAT category was used.
4. Hyperparameter tuning was conducted using the 5-fold cross-validation method with optimal model performance using the F1 score determined for:
   - C = 1
   - kernel = Linear
   - class_weight = None
5. The apparent model performance of the optimal model is summarized as follows:
   - **Accuracy** = 0.9736
   - **Precision** = 0.9062
   - **Recall** = 1.0000
   - **F1 Score** = 0.9508
   - **AUROC** = 0.9823
6. The independent test model performance of the final model is summarized as follows:
   - **Accuracy** = 0.8979
   - **Precision** = 0.8181
   - **Recall** = 0.7500
   - **F1 Score** = 0.7826
   - **AUROC** = 0.8479
7. High difference in the apparent and independent test model performance observed, indicative of the presence of excessive model overfitting.

RESULTS – CNN + LOGISTIC REGRESSION

- ## Findings (GitHub URL: Python Notebook | Markdown Presentation)

1. The logistic regression model from the **sklearn.linear_model** Python library API was implemented.
2. The model contains 5 hyperparameters:
   - C = inverse of regularization strength held constant at a value of 1
   - penalty = penalty norm made to vary between L1 and L2
   - solver = algorithm used in the optimization problem made to vary between Saga and Liblinear
   - class_weight = weights associated with classes held constant at a value of None
   - max_iter = maximum number of iterations taken for the solvers to converge held constant at a value of 500
3. The reduced model training data by downsampling the majority LOW CANRAT category was used.
4. Hyperparameter tuning was conducted using the 5-fold cross-validation method with optimal model performance using the F1 score determined for:
   - C = 1
   - penalty = L1 norm
   - solver = Liblinear
   - class_weight = None
   - max_iter = 500
5. The apparent model performance of the optimal model is summarized as follows:
   - **Accuracy** = 0.9473
   - **Precision** = 0.8484
   - **Recall** = 0.9655
   - **F1 Score** = 0.9032
   - **AUROC** = 0.9533
6. The independent test model performance of the final model is summarized as follows:
   - **Accuracy** = 0.9183
   - **Precision** = 0.9000
   - **Recall** = 0.7500
   - **F1 Score** = 0.8181
   - **AUROC** = 0.8614
7. High difference in the apparent and independent test model performance observed, indicative of the presence of excessive model overfitting.

RESULTS – CNN + DECISION TREE

- ## Findings (GitHub URL: Python Notebook | Markdown Presentation)

1. The decision tree model from the **sklearn.tree** Python library API was implemented.
2. The model contains 5 hyperparameters:
   - criterion = function to measure the quality of a split made to vary between Gini, Entropy and Log-Loss
   - max_depth = maximum depth of the tree made to vary between 3, 5 and 7
   - min_samples_leaf = minimum number of samples required to split an internal node made to vary between 3, 5 and 10
   - class_weight = weights associated with classes held constant at a value of None
3. The reduced model training data by downsampling the majority LOW CANRAT category was used.
4. Hyperparameter tuning was conducted using the 5-fold cross-validation method with optimal model performance using the F1 score determined for:
   - criterion = Gini
   - max_depth = 3
   - min_samples_leaf = 5
   - class_weight = None
5. The apparent model performance of the optimal model is summarized as follows:
   - **Accuracy** = 0.9385
   - **Precision** = 0.9230
   - **Recall** = 0.8275
   - **F1 Score** = 0.8727
   - **AUROC** = 0.9020
6. The independent test model performance of the final model is summarized as follows:
   - **Accuracy** = 0.8979
   - **Precision** = 0.8888
   - **Recall** = 0.6666
   - **F1 Score** = 0.7619
   - **AUROC** = 0.8198
7. High difference in the apparent and independent test model performance observed, indicative of the presence of excessive model overfitting.

RESULTS – CNN + RANDOM FOREST

- ## Findings (GitHub URL: Python Notebook | Markdown Presentation)

1. The random forest model from the **sklearn.ensemble** Python library API was implemented.
2. The model contains 5 hyperparameters:
   - criterion = function to measure the quality of a split made to vary between Gini, Entropy and Log-Loss
   - max_depth = maximum depth of the tree made to vary between 3, 5 and 7
   - min_samples_leaf = minimum number of samples required to split an internal node made to vary between 3, 5 and 10
   - n_estimators = number of trees in the forest made to vary between 100, 150 and 200
   - max_features = number of features to consider when looking for the best split made to vary between Sqrt and Log2 of n_estimators
   - class_weight = weights associated with classes held constant at a value of None
3. The reduced model training data by downsampling the majority LOW CANRAT category was used.
4. Hyperparameter tuning was conducted using the 5-fold cross-validation method with optimal model performance using the F1 score determined for:
   - criterion = Gini
   - max_depth = 3
   - min_samples_leaf = 3
   - n_estimators = 100
   - max_features = Sqrt n_estimators
   - class_weight = None
5. The apparent model performance of the optimal model is summarized as follows:
   - **Accuracy** = 0.9649
   - **Precision** = 0.9032
   - **Recall** = 0.9655
   - **F1 Score** = 0.9333
   - **AUROC** = 0.9651
6. The independent test model performance of the final model is summarized as follows:
   - **Accuracy** = 0.8979
   - **Precision** = 0.8888
   - **Recall** = 0.6666
   - **F1 Score** = 0.7619
   - **AUROC** = 0.8198
7. High difference in the apparent and independent test model performance observed, indicative of the presence of excessive model overfitting.

RESULTS – CNN + SVM

- ## Findings (GitHub URL: Python Notebook | Markdown Presentation)

1. The support vector machine model from the **sklearn.svm** Python library API was implemented.
2. The model contains 5 hyperparameters:
   - C = inverse of regularization strength held constant at a value of 1
   - kernel = kernel type to be used in the algorithm made to vary between Linear, Poly, RBF and Sigmoid
   - class_weight = weights associated with classes held constant at a value of None
3. The reduced model training data by downsampling the majority LOW CANRAT category was used.
4. Hyperparameter tuning was conducted using the 5-fold cross-validation method with optimal model performance using the F1 score determined for:
   - C = 1
   - kernel = Linear
   - class_weight = None
5. The apparent model performance of the optimal model is summarized as follows:
   - **Accuracy** = 0.9561
   - **Precision** = 0.9285
   - **Recall** = 0.8965
   - **F1 Score** = 0.9122
   - **AUROC** = 0.9365
6. The independent test model performance of the final model is summarized as follows:
   - **Accuracy** = 0.8979
   - **Precision** = 0.8888
   - **Recall** = 0.6666
   - **F1 Score** = 0.7619
   - **AUROC** = 0.8198
7. High difference in the apparent and independent test model performance observed, indicative of the presence of excessive model overfitting.

RESULTS – STACKED ENSEMBLE LEARNING

- ## Findings (GitHub URL: Python Notebook | Markdown Presentation)

1. The logistic regression model from the **sklearn.linear_model** Python library API was implemented as a meta-learner for the stacking algorithm.
2. The model used default hyperparameters with no tuning applied:
   - C = inverse of regularization strength held constant at a value of 1
   - penalty = penalty norm held constant at a value of L2
   - solver = algorithm used in the optimization problem held constant at a value of Lbfgs
   - class_weight = weights associated with classes held constant at a value of 25-75 between classes 0 and 1
   - max_iter = maximum number of iterations taken for the solvers to converge held constant at a value of 500
3. The original data reflecting a 3:1 class imbalance between the LOW and HIGH CANRAT categories was used for model training and testing.
4. The apparent model performance of the optimal model is summarized as follows:
   - **Accuracy** = 0.9736
   - **Precision** = 0.9062
   - **Recall** = 1.0000
   - **F1 Score** = 0.9508
   - **AUROC** = 0.9823
5. The independent test model performance of the final model is summarized as follows:
   - **Accuracy** = 0.9183
   - **Precision** = 0.9000
   - **Recall** = 0.7500
   - **F1 Score** = 0.8181
   - **AUROC** = 0.8614
6. High difference in the apparent and independent test model performance observed, indicative of the presence of excessive model overfitting.

RESULTS – MODEL VALIDATION AND SELECTION

- **Findings** (GitHub URL: Python Notebook | Markdown Presentation)

1. Among the formulated versions of the logistic regression model, the model which applied upsampling of the minority class using SMOTE was used as a base learner for the model stacking algorithm.
   - **Accuracy** = 0.9183
   - **Precision** = 0.9000
   - **Recall** = 0.7500
   - **F1 Score** = 0.8181
   - **AUROC** = 0.8614
2. Among the formulated versions of the decision tree model, the model which applied upsampling of the minority class using SMOTE was used as a base learner for the model stacking algorithm.
   - **Accuracy** = 0.8979
   - **Precision** = 0.7692
   - **Recall** = 0.8333
   - **F1 Score** = 0.8000
   - **AUROC** = 0.8761
3. Among the formulated versions of the random forest model, the model which applied upsampling of the minority class using SMOTE was used as a base learner for the model stacking algorithm.
   - **Accuracy** = 0.9387
   - **Precision** = 0.8461
   - **Recall** = 0.9167
   - **F1 Score** = 0.8800
   - **AUROC** = 0.9313
4. Among the formulated versions of the support vector machine model, the model which applied upsampling of the minority class using SMOTE was used as a base learner for the model stacking algorithm.
   - **Accuracy** = 0.8979
   - **Precision** = 0.8181
   - **Recall** = 0.7500
   - **F1 Score** = 0.7826
   - **AUROC** = 0.8479

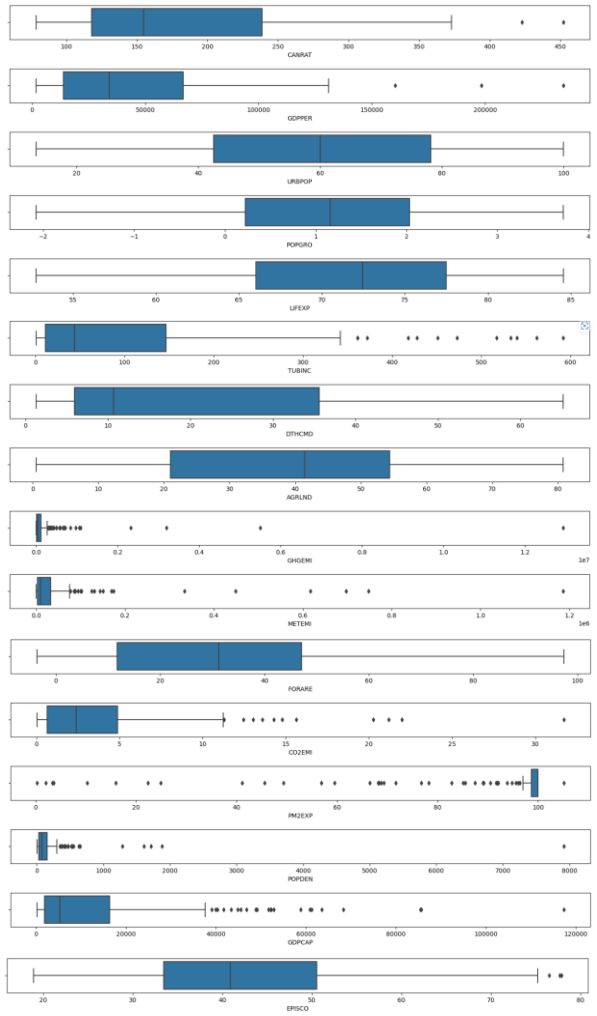
RESULTS – MODEL VALIDATION AND SELECTION

- **Findings** (GitHub URL: Python Notebook | Markdown Presentation)

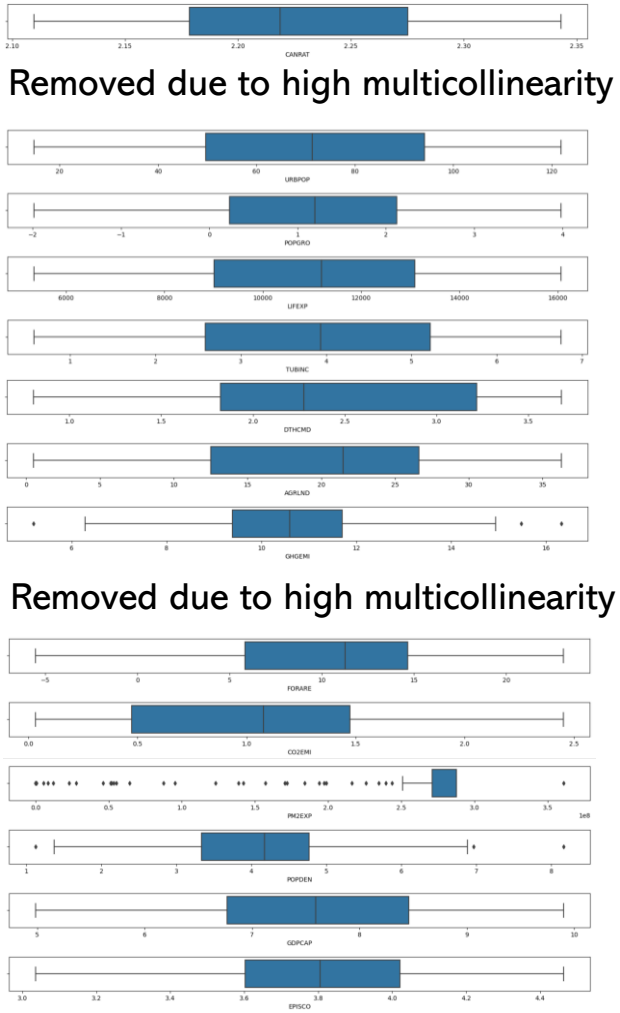
5. The stacked logistic regression model comprised of the individual base learners demonstrated sufficient class discrmination:
   - **Accuracy** = 0.9183
   - **Precision** = 0.9000
   - **Recall** = 0.7500
   - **F1 Score** = 0.8181
   - **AUROC** = 0.8614
6. Comparing all results from the formulated base and stacked models formulated, the logistic regression model which applied class weights still demonstrated the best independent test model performance and was selected as the final model for classification.
   - **Accuracy** = 0.9387
   - **Precision** = 0.8461
   - **Recall** = 0.9167
   - **F1 Score** = 0.8800
   - **AUROC** = 0.9313

PLOTS – OUTLIER ANALYSIS

- Original Data



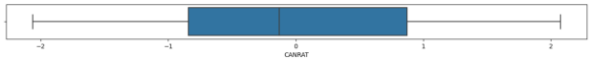
- Transformed Data



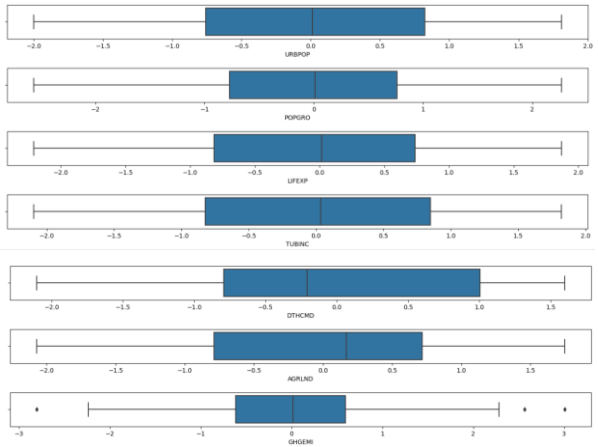
Removed due to high multicollinearity

Removed due to high multicollinearity

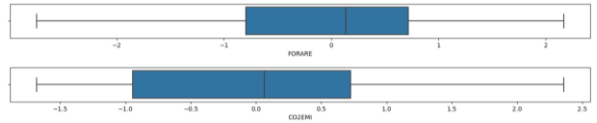
- Centered and Scaled Data



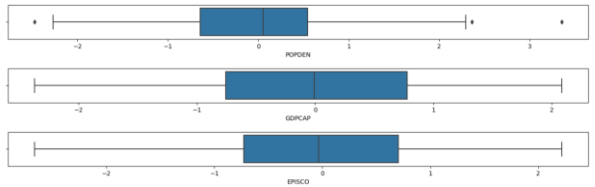
Removed due to high multicollinearity



Removed due to high multicollinearity

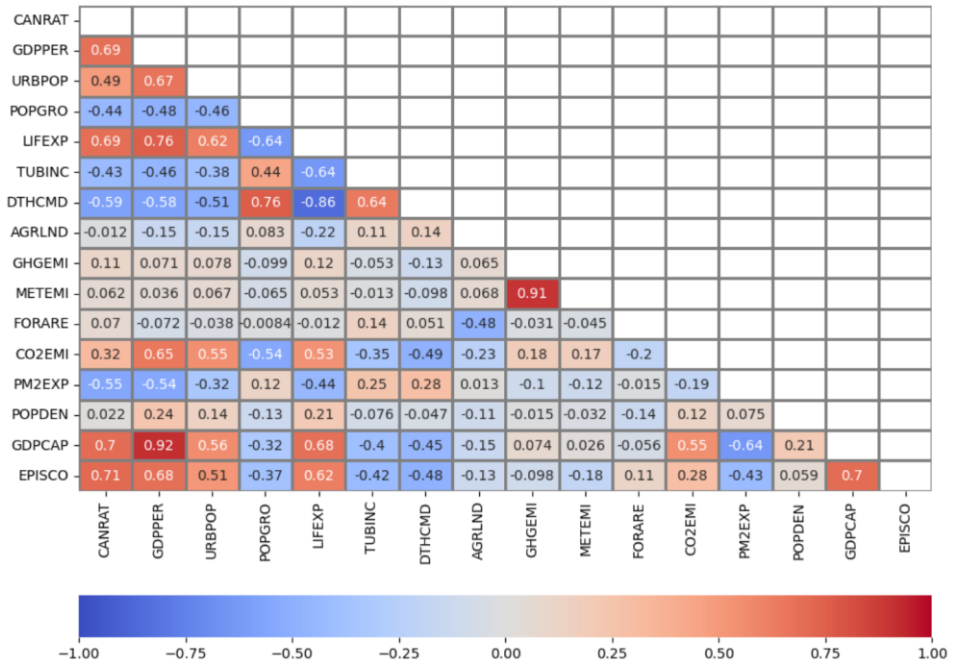


Removed due to persistent skew after transformation

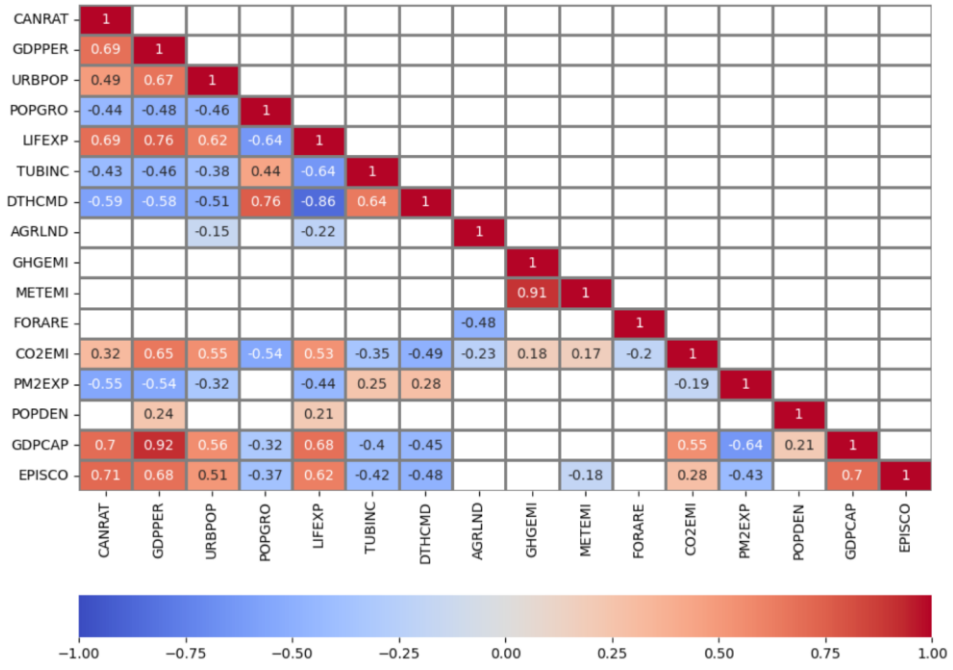


PLOTS – CORRELATION ANALYSIS

- All Correlations

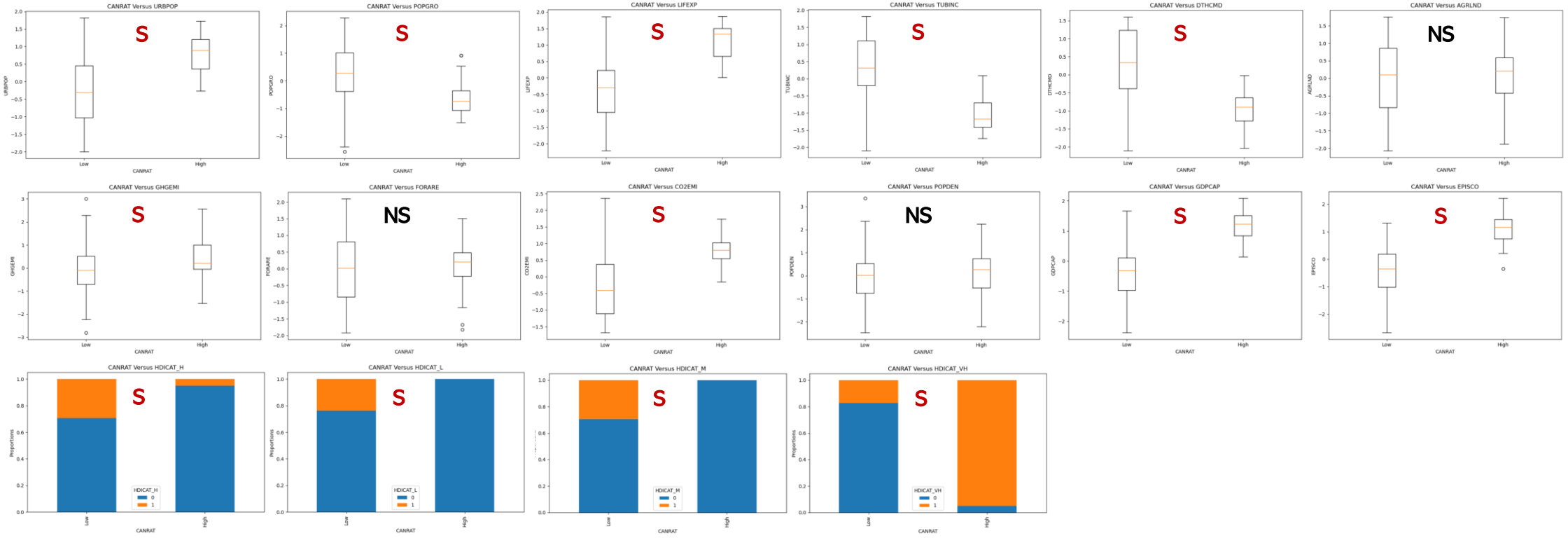


- Statistically Significant Correlations



PLOTS – EXPLORATORY DATA ANALYSIS

- Relationship Boxplots

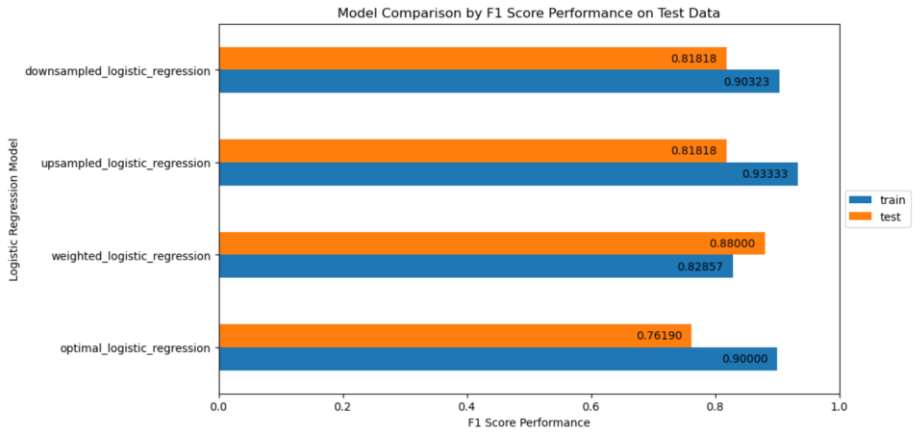


S: Association between predictor and target was statistically significant

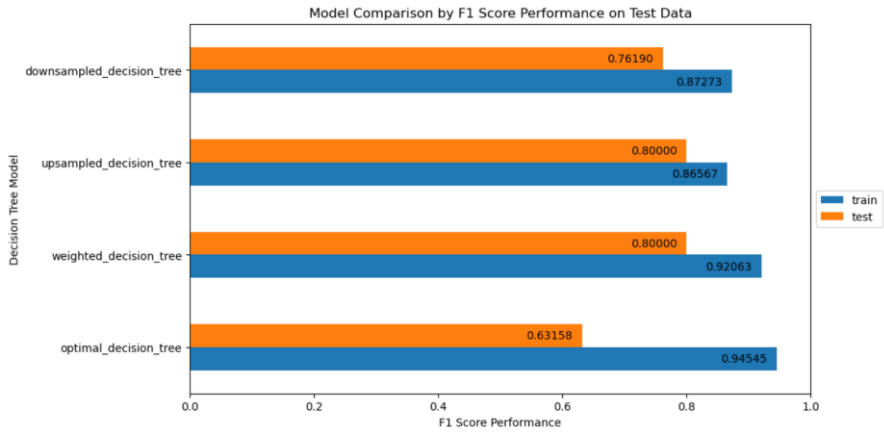
NS: Association between predictor and target was not statistically significant

PLOTS – MODEL FORMULATION ON TRAIN DATA

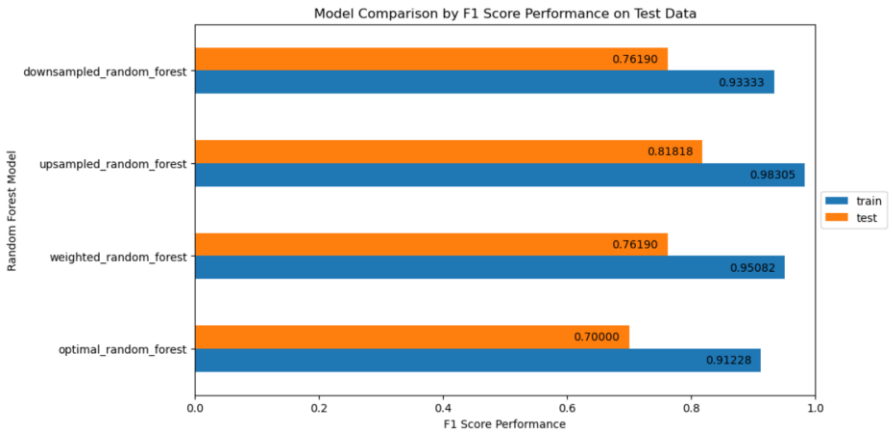
- Logistic Regression



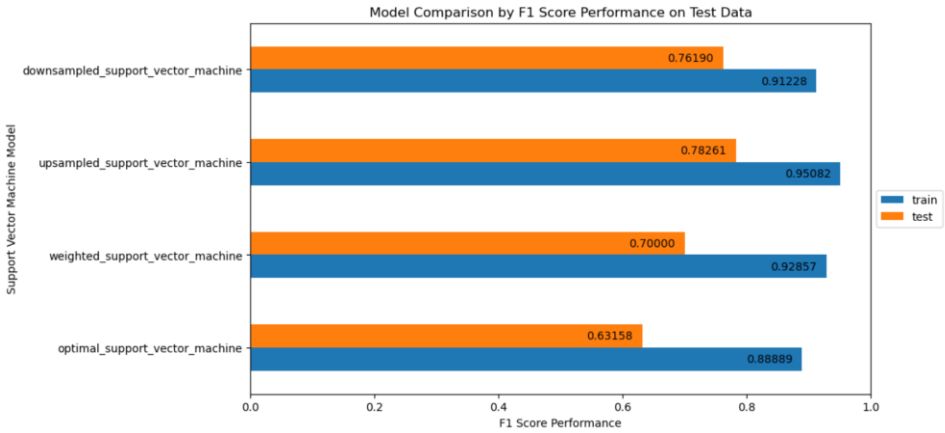
- Decision Tree



- Random Forest

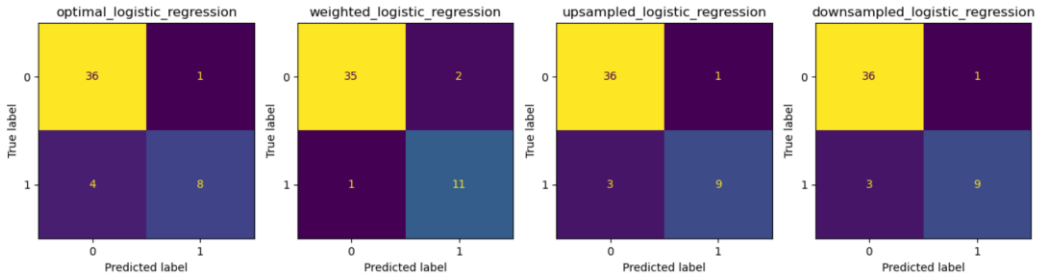


- Support Vector Machine

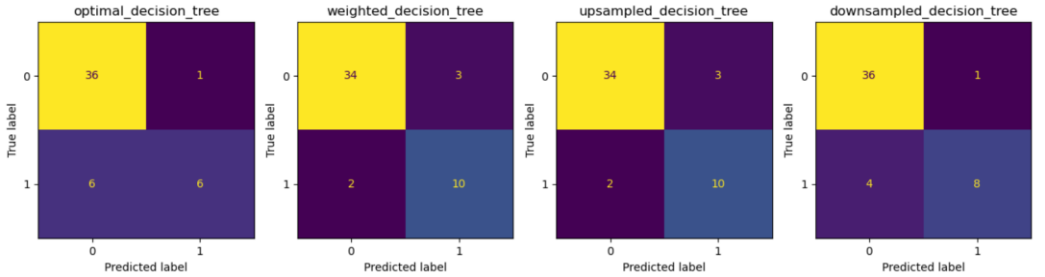


PLOTS – MODEL VALIDATION ON TEST DATA

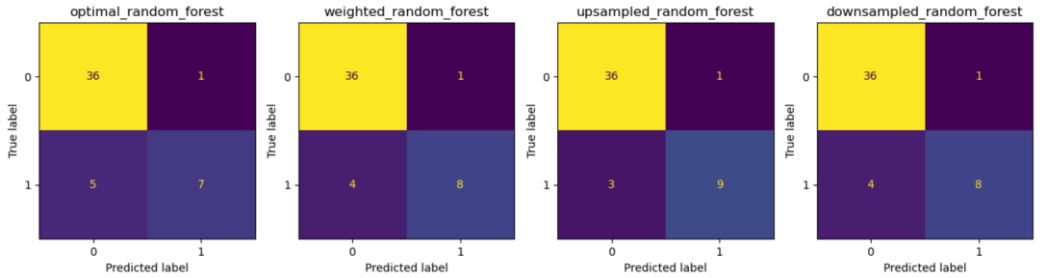
- Logistic Regression



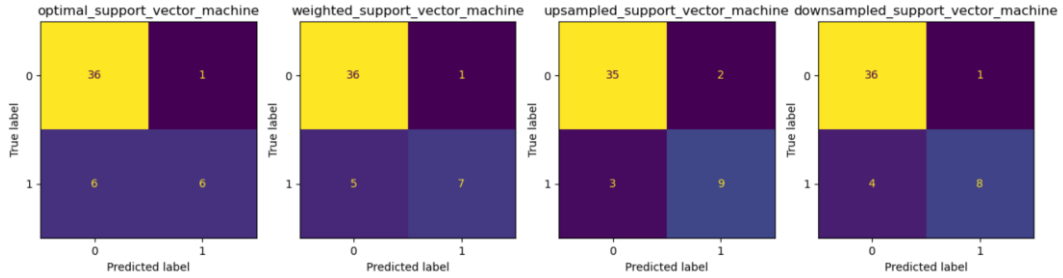
- Decision Tree



- Random Forest

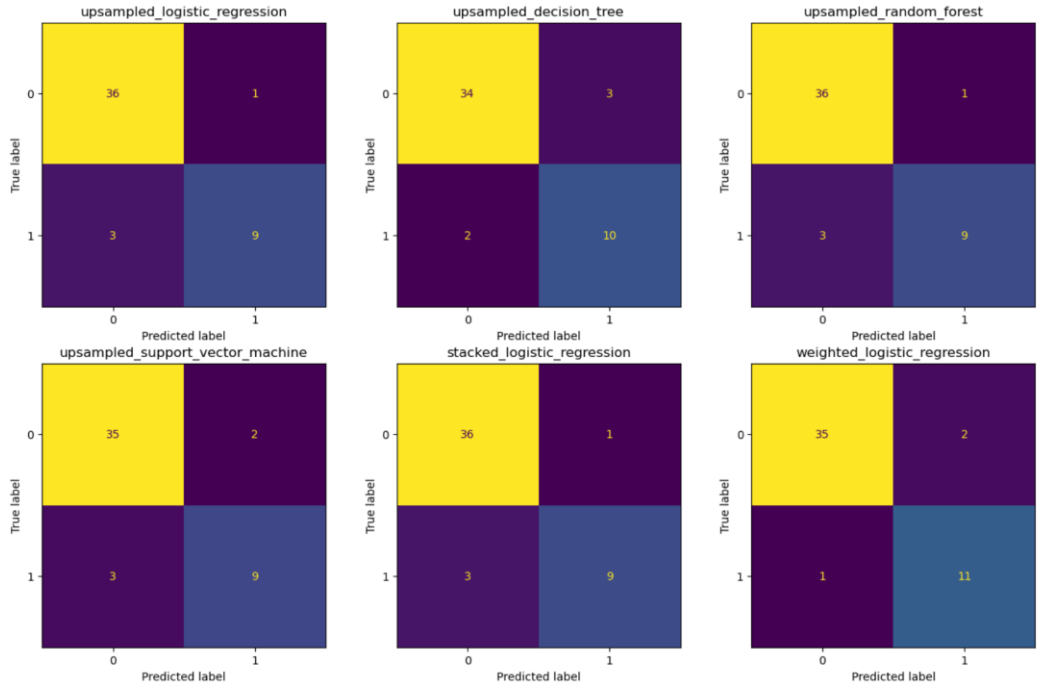


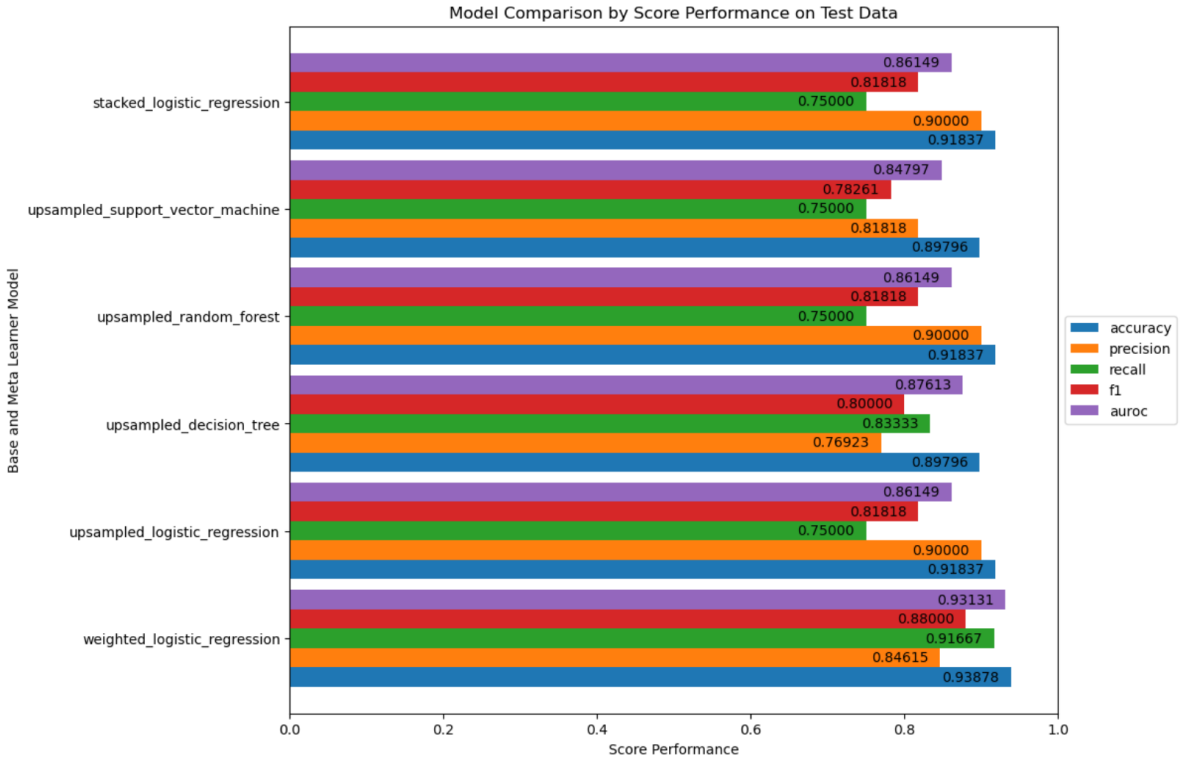
- Support Vector Machine



PLOTS – MODEL SELECTION

- Stacked Ensemble





Section 4

Summary

OVERALL FINDINGS AND IMPLICATIONS

- ## Key Findings
  - Using the F1 score, the best-performing **logistic regression model** applied **class weights.**
  - The best-performing **decision tree**, **random forest**, and **support vector machine** models were those applied with **upsampling using SMOTE**, as compared to the other evaluated remedial measures to address class imbalance including **hyperparameter tuning**, **class weights**, and **downsampling using CNN**.
  - Using the individual models that applied **upsampling using SMOTE** as base-learners for a stacking ensemble that used logistic regression as a meta-learner demonstrated good discrimination but was still inferior to the **logistic regression model** which applied **class weights**.
- ## Overall Implications
  - The final classification model using the **logistic regression model** which applied **class weights** could provide robust and reliable estimates of cancer categories based on the estimated metrics as follows.
    - Accuracy = 93.87%
    - Precision = 84.61%
    - Recall = 91.67%
    - F1 Score = 88.00%
    - AUROC = 0.9313

CONCLUSION

- Overall Summary
  - Data collection for the analysis involved world performance indicators and indices hypothesized to be directly influencing cancer rates across countries.
  - The quality of gathered data was assessed and potential issues were identified.
  - Appropriate pre-processing methods including remedial procedures to address duplicate, missing, outlying, and non-normalized data were applied to prepare the data for subsequent analysis. Additional data scaling and transformation were implemented.
  - EDA using visualization presented the various distributions and comparisons between the indicators and indices as evaluated against cancer rate categories – eventually identifying the key drivers of higher cancer rates. Statistical hypothesis testing identified the individual drivers that were significantly associated with cancer rate categories.
  - A final classification model was selected among candidates that could provide robust and reliable probability estimates of cancer rate categories from an optimal set of observations and predictors.
  - Overall analysis findings were discussed and their practical implications were highlighted.

Section 5

Appendix

APPENDIX

- Source Data
  - Cancer Rates: World Population Review
  - Social Protection and Labor Indicator: World Bank
  - Education Indicator: World Bank
  - Economy and Growth Indicator: World Bank
  - Environment Indicator: World Bank
  - Climate Change Indicator: World Bank
  - Agricultural and Rural Development Indicator: World Bank
  - Social Development Indicator: World Bank
  - Health Indicator: World Bank
  - Science and Technology Indicator: World Bank
  - Urban Development Indicator: World Bank
  - Human Development Indices: Human Development Reports
  - Environmental Performance Indices: Yale Center for Environmental Law and Policy

APPENDIX

- Python Notebooks | Codes
  - GitHub URL: Data Background
  - GitHub URL: Data Description
  - GitHub URL: Data Quality Assessment
  - GitHub URL: Data Preprocessing
    - GitHub URL: Data Cleaning
    - GitHub URL: Missing Data Imputation
    - GitHub URL: Outlier Treatment
    - GitHub URL: Collinearity
    - GitHub URL: Shape Transformation
    - GitHub URL: Centering and Scaling
    - GitHub URL: Data Encoding
    - GitHub URL: Preprocessed Data Description
  - GitHub URL: Data Exploration
    - GitHub URL: Exploratory Data Analysis
    - GitHub URL: Hypothesis Testing

APPENDIX

- Python Notebooks | Codes
  - GitHub URL: Model Development with Hyperparameter Tuning
    - GitHub URL: Premodelling Data Description
    - GitHub URL: Logistic Regression
    - GitHub URL: Decision Tree
    - GitHub URL: Random Forest
    - GitHub URL: Support Vector Machine
  - GitHub URL: Model Development with Class Weights
    - GitHub URL: Premodelling Data Description
    - GitHub URL: Logistic Regression
    - GitHub URL: Decision Tree
    - GitHub URL: Random Forest
    - GitHub URL: Support Vector Machine

APPENDIX

- Python Notebooks | Codes
  - GitHub URL: Model Development with SMOTE Upsampling
    - GitHub URL: Premodelling Data Description
    - GitHub URL: Logistic Regression
    - GitHub URL: Decision Tree
    - GitHub URL: Random Forest
    - GitHub URL: Support Vector Machine
  - GitHub URL: Model Development with CNN Downsampling
    - GitHub URL: Premodelling Data Description
    - GitHub URL: Logistic Regression
    - GitHub URL: Decision Tree
    - GitHub URL: Random Forest
    - GitHub URL: Support Vector Machine

APPENDIX

- Python Notebooks | Codes
  - GitHub URL: Model Development with Stacking Ensemble Learning
    - GitHub URL: Premodelling Data Description
    - GitHub URL: Logistic Regression
  - GitHub URL: Consolidated Findings

APPENDIX

References

- [Book] Data Preparation for Machine Learning: Data Cleaning, Feature Selection, and Data Transforms in Python by Jason Brownlee
- [Book] Feature Engineering and Selection: A Practical Approach for Predictive Models by Max Kuhn and Kjell Johnson
- [Book] Feature Engineering for Machine Learning by Alice Zheng and Amanda Casari
- [Book] Applied Predictive Modeling by Max Kuhn and Kjell Johnson
- [Book] Data Mining: Practical Machine Learning Tools and Techniques by Ian Witten, Eibe Frank, Mark Hall and Christopher Pal
- [Book] Data Cleaning by Ihab Ilyas and Xu Chu
- [Book] Data Wrangling with Python by Jacqueline Kazil and Katharine Jarmul
- [Book] Regression Modeling Strategies by Frank Harrell
- [Book] Ensemble Methods for Machine Learning by Gautam Kunapuli
- [Python Library API] NumPy by NumPy Team
- [Python Library API] pandas by Pandas Team
- [Python Library API] seaborn by Seaborn Team
- [Python Library API] matplotlib.pyplot by MatPlotLib Team
- [Python Library API] itertools by Python Team
- [Python Library API] operator by Python Team
- [Python Library API] sklearn.experimental by Scikit-Learn Team
- [Python Library API] sklearn.impute by Scikit-Learn Team
- [Python Library API] sklearn.linear_model by Scikit-Learn Team
- [Python Library API] sklearn.preprocessing by Scikit-Learn Team
- [Python Library API] sklearn.metrics by Scikit-Learn Team
- [Python Library API] sklearn.model_selection by Scikit-Learn Team
- [Python Library API] sklearn.pipeline by Scikit-Learn Team
- [Python Library API] scipy by SciPy Team

APPENDIX

References

- **[Python Library API]** sklearn.tree by Scikit-Learn Team
- **[Python Library API]** sklearn.ensemble by Scikit-Learn Team
- **[Python Library API]** sklearn.svm by Scikit-Learn Team
- **[Python Library API]** sklearn.metrics by Scikit-Learn Team
- **[Python Library API]** sklearn.model_selection by Scikit-Learn Team
- **[Python Library API]** imblearn.over_sampling by Imbalanced-Learn Team
- **[Python Library API]** imblearn.under_sampling by Imbalanced-Learn Team
- **[Article]** Step-by-Step Exploratory Data Analysis (EDA) using Python by Malamahadevan Mahadevan (Analytics Vidhya)
- **[Article]** Exploratory Data Analysis in Python — A Step-by-Step Process by Andrea D'Agostino (Towards Data Science)
- **[Article]** Exploratory Data Analysis with Python by Douglas Rocha (Medium)
- **[Article]** 4 Ways to Automate Exploratory Data Analysis (EDA) in Python by Abdishakur Hassan (BuiltIn)
- **[Article]** 10 Things To Do When Conducting Your Exploratory Data Analysis (EDA) by Alifia Harmadi (Medium)
- **[Article]** How to Handle Missing Data with Python by Jason Brownlee (Machine Learning Mastery)
- **[Article]** Statistical Imputation for Missing Values in Machine Learning by Jason Brownlee (Machine Learning Mastery)
- **[Article]** Imputing Missing Data with Simple and Advanced Techniques by Idil Ismiguzel (Towards Data Science)
- **[Article]** Missing Data Imputation Approaches | How to handle missing values in Python by Selva Prabhakaran (Machine Learning +)
- **[Article]** Master The Skills Of Missing Data Imputation Techniques In Python(2022) And Be Successful by Mrinal Walia (Analytics Vidhya)
- **[Article]** How to Preprocess Data in Python by Afroz Chakure (BuiltIn)
- **[Article]** Easy Guide To Data Preprocessing In Python by Ahmad Anis (KDNuggets)

APPENDIX

References

- [Article] Data Preprocessing in Python by Tarun Gupta (Towards Data Science)
- [Article] Data Preprocessing using Python by Suneet Jain (Medium)
- [Article] Data Preprocessing in Python by Abonia Sojasingarayar (Medium)
- [Article] Data Preprocessing in Python by Afroz Chakure (Medium)
- [Article] Detecting and Treating Outliers | Treating the Odd One Out! by Harika Bonthu (Analytics Vidhya)
- [Article] Outlier Treatment with Python by Sangita Yemulwar (Analytics Vidhya)
- [Article] A Guide to Outlier Detection in Python by Sadrach Pierre (BuiltIn)
- [Article] How To Find Outliers in Data Using Python (and How To Handle Them) by Eric Kleppen (Career Foundry)
- [Article] Statistics in Python — Collinearity and Multicollinearity by Wei-Meng Lee (Towards Data Science)
- [Article] Understanding Multicollinearity and How to Detect it in Python by Terence Shin (Towards Data Science)
- [Article] A Python Library to Remove Collinearity by Gianluca Malato (Your Data Teacher)
- [Article] 8 Best Data Transformation in Pandas by Tirendaz AI (Medium)
- [Article] Data Transformation Techniques with Python: Elevate Your Data Game! by Siddharth Verma (Medium)
- [Article] Data Scaling with Python by Benjamin Obi Tayo (KDNuggets)
- [Article] How to Use StandardScaler and MinMaxScaler Transforms in Python by Jason Brownlee (Machine Learning Mastery)
- [Article] Feature Engineering: Scaling, Normalization, and Standardization by Aniruddha Bhandari (Analytics Vidhya)
- [Article] How to Normalize Data Using scikit-learn in Python by Jayant Verma (Digital Ocean)
- [Article] What are Categorical Data Encoding Methods | Binary Encoding by Shipra Saxena (Analytics Vidhya)
- [Article] Guide to Encoding Categorical Values in Python by Chris Moffitt (Practical Business Python)

APPENDIX

- ## References
  - **[Article]** Categorical Data Encoding Techniques in Python: A Complete Guide by Soumen Atta (Medium)
  - **[Article]** Categorical Feature Encoding Techniques by Tara Boyle (Medium)
  - **[Article]** Ordinal and One-Hot Encodings for Categorical Data by Jason Brownlee (Machine Learning Mastery)
  - **[Article]** Hypothesis Testing with Python: Step by Step Hands-On Tutorial with Practical Examples by Ece Polat (Towards Data Science)
  - **[Article]** 17 Statistical Hypothesis Tests in Python (Cheat Sheet) by Jason Brownlee (Machine Learning Mastery)
  - **[Article]** A Step-by-Step Guide to Hypothesis Testing in Python using Scipy by Gabriel Rennó (Medium)

APPENDIX

References

- **[Publication]** Data Quality for Machine Learning Tasks by Nitin Gupta, Shashank Mujumdar, Hima Patel, Satoshi Masuda, Naveen Panwar, Sambaran Bandyopadhyay, Sameep Mehta, Shanmukha Guttula, Shazia Afzal, Ruhi Sharma Mittal and Vitobha Munigala (KDD '21: Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining)
- **[Publication]** Overview and Importance of Data Quality for Machine Learning Tasks by Abhinav Jain, Hima Patel, Lokesh Nagalapatti, Nitin Gupta, Sameep Mehta, Shanmukha Guttula, Shashank Mujumdar, Shazia Afzal, Ruhi Sharma Mittal and Vitobha Munigala (KDD '20: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining)
- **[Publication]** Multiple Imputation of Discrete and Continuous Data by Fully Conditional Specification by Stef van Buuren (Statistical Methods in Medical Research)
- **[Publication]** Mathematical Contributions to the Theory of Evolution: Regression, Heredity and Panmixia by Karl Pearson (Royal Society)
- **[Publication]** A New Family of Power Transformations to Improve Normality or Symmetry by In-Kwon Yeo and Richard Johnson (Biometrika)
- **[Publication]** The Origins of Logistic Regression by JS Cramer (Econometrics eJournal)
- **[Publication]** Classification and Regression Trees by Leo Breiman, Jerome Friedman, Richard Olshen and Charles Stone (Computer Science)
- **[Publication]** Random Forest by Leo Breiman (Machine Learning)
- **[Publication]** A Training Algorithm for Optimal Margin Classifiers by Bernhard Boser, Isabelle Guyon and Vladimir Vapnik (Proceedings of the Fifth Annual Workshop on Computational Learning Theory)
- **[Publication]** SMOTE: Synthetic Minority Over-Sampling Technique by Nitesh Chawla, Kevin Bowyer, Lawrence Hall and Philip Kegelmeyer (Journal of Artificial Intelligence Research)
- **[Publication]** The Condensed Nearest Neighbor Rule by Peter Hart (IEEE Transactions on Information Theory)

Thank You!