AI AND MACHINE LEARNING PROJECT

Predictive modelling of Global Region based on Micronutrient Deficiency using Supervised Machine Learning Algorithms

Panu Khumchoo-Joonpijarn
23135551

Table of Contents

1 Report Introduction

1.1 Introduction

As stated in journal (Beal et al., 2017), we characterise the global trends in dietary quality by estimating micronutrient density of the food supply. Undernourishment remains very high in about 11% of the global population, such hidden hunger is associated with pregnancy complications and child growth failure, increase susceptibility to disease, and impair cognitive development. On the other hand, excessive consumption is associated with increased incidence of obesity and risk of diabetes and heart disease.

The dataset being used in this study is from the journal detailing the "Global trends in dietary micronutrient supplies and estimated prevalence of inadequate intakes" between 1961 and 2011. The dataset uses a variety of figures to visualise the prevalence of inadequate intake of 14 micronutrients globally. The journal suggests "at the global level, micronutrients with the lowest levels of adequate estimated intake are calcium, iron, vitamin A, and zinc, but there are strong differences between countries and regions". The aim of this report is to explore this statement, and whether it is viable to apply a variety of Supervised Machine Learning Algorithms, to correctly predict and classify a record's global region based on the micronutrient features found in this dataset.

1.2 Dataset Identification

The "S4_Dataset" from the journal, can also be found on fig share. The dataset contains 111817 rows of records, ordered by country and year for each micronutrient.

Attribute	Description
Zone	Global Region the record is from in string form.
Country	The country the record is sourced from in string form.
ISO3	The 3-letter code identifier for a country in string form.
Year	The year in which the data record is collated from between 1961-2011 in int form.
Population	The total population of the country record is sourced from in float form.
Fortification	The binary classifier which identifies if a country supplements micronutrients to a nation's food supplies (via food and water) in int form.
PCDEA	National per capita daily energy availability in int form measured in Kilocalories (Kcal).
MDI	Micronutrient Density Index for each country-year in float form.
Tagname	Abbreviated identifier for Micronutrient in string form.
Micronutrient	Identifies the specific Micronutrient data the record is referring to. One of the 14 Micronutrients in string form.
Units	The measurement unit specified for all 14 micronutrients in string form.
Estimated Intake	Explained in journal as an estimate of micronutrient intake "per capita derived from FBS data by calculating coefficient of variation of intake based on within-subject

	variation from published dietary intake studies" in float form.
Requirements	Refers to "Estimated average requirements (EARs) obtained by the Institute of Medicine for all nutrients except zinc, iron, and calcium" in float form.
Prevalence of Inadequate Intake	The result from the "EAR cut-point method to estimate based on approximated micronutrient intakes. It "relies on three assumptions: the distribution of intakes varies more than the distribution of requirements; the distribution of requirements is symmetrical; and intakes and requirements are not correlated" in float form.

Table 1

1.3 Supervised Learning Task Identification

As mentioned in the introduction, from the S4_Dataset, the aim is to develop and apply a machine learning model which will predict the global region (Zone) attribute, basing its prediction on the other features and attributes of the dataset shown in Fig. 1.

This model will be using classification algorithms to correctly identify each record's corresponding Zone. In these models (other than Deep Learning), the Zone feature is label encoded, meaning each Zone is represented by a corresponding numerical value (0-7). As such, traditional binary classification methodology is limited in this application and requires visualisations to be adjusted.

The applied models which will be explored are Logistic Regression, Support Vector Machine (SVC), Random Forests, K-Nearest Neighbour, Multinomial Naïve Bayes, Decision Tree Classifier, and Deep Learning. Their performance will be compared and tested under different pre-processing and sampling methods.

2 Exploratory Data Analysis Process and Results

2.1 Question(s) Identification

Question No.	Question	Assumption
1	What is the spread of Zones in the dataset? Is it balanced data?	The dataset in the Zones column is spread among 8 categories, with a count of 29120 values at highest and 3640 at lowest. This means the Zone category is extremely unbalanced.
2	Does MDI effect Micronutrient intake?	Beal states the MDI decline from "1979 to 1993 in sub-Saharan Africa was due to the increased availability of lower micro-nutrient density grains and vegetable oils", whilst a decreased proportionality of micronutrient rich products.

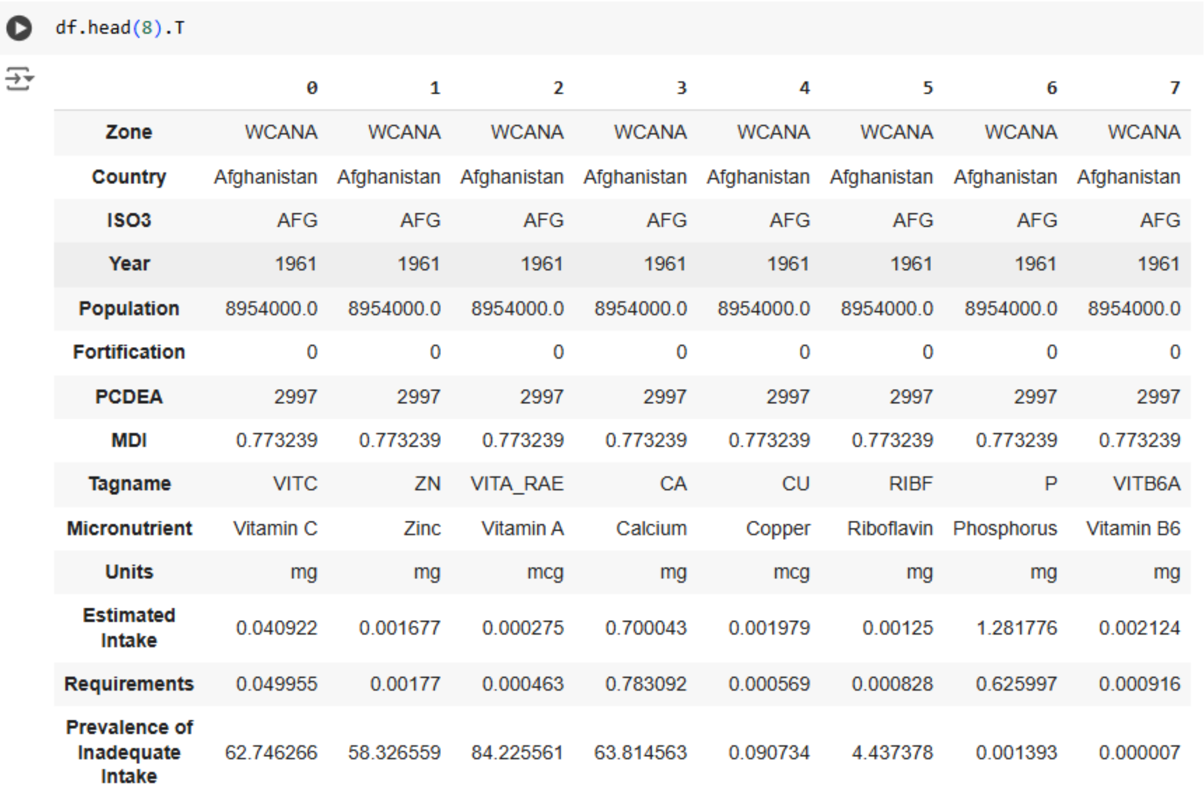
3	Does the dataset have any null or missing values to be handled?	There are no null values or missing data that needs to be imputed.
4	Does computational speed of each model provide any relevance?	As the aim is correct classification of data, accuracy of prediction algorithm should be prioritised, but computational speed will be considered in this report to provide comprehensive performance review for each algorithm on this set of data.
5	What is the importance of the global nutrient database?	As mentioned in the discussion of the (Schmidhuber et al., 2018) article, "the global nutrient database provides the opportunity to answer important questions about macronutrients and micronutrients across nations. For example, the database can be used to identify countries with insufficient supplies of specific nutrients that are therefore at risk of nutrient deficiencies. Additionally, the database can be used to determine the dependency on specific foods for each nutrient in a country and over time. This information is necessary to make both agriculture and trade more nutrition-sensitive and to inform food-based interventions to prevent micronutrient deficiencies in high-risk countries."
6	What is the supply trend of micronutrients globally?	Schmidhuber also mentions the "supply of micronutrients has generally increased between 1980 and 2013 across levels of development".
7	Does Fortification influence a population's micronutrient consumption?	It is within a government's best interest to ensure their population is well-nourished, thus national policy for fortification of food products is common. As mentioned in (Mannar and Sankar, 2004), studies by World Bank shows that "countries whose populations suffer from micronutrient deficiencies encounter economic losses as high as 5% of gross domestic product (GDP)" and later goes on to suggest solutions as implementing education programmes, targeted distribution programmes, or "fortifying commonly eaten foods with the missing micronutrients" ensuring the "stability and bioavailability of the nutrient".
8	Where globally is the highest PCDEA, MDI, Estimated Intake?	Schmidhuber mentions the energy availability widely varied across levels of development ranging from 2170 kcal (2090–2250) per person per day in low-SDI countries to 3270 (3220–3310) kcal per person per day in high-SDI (Socio-demographic Index) countries. Among the most populous countries, the highest level of energy availability was in the USA (3500 kcal [3450–3560] per person per day) and the lowest was in Ethiopia (1880 kcal [1810–1940] per person per day). Beal finds the 5 countries with highest MDI in 2011 were "Antigua and Barbuda, Kazakhstan, Armenia, Albania, and Serbia".

		Additionally, Beal finds estimated intake increased globally through fortification of certain micronutrients.

Table 2

2.2 Exploratory Data Analysis Process and Results

Using the head function, the first few data records in the data-frame can be displayed as shown in Fig. 1, allowing for the visualisation of the data and give a better understanding of the data held in the dataset and how they interact.

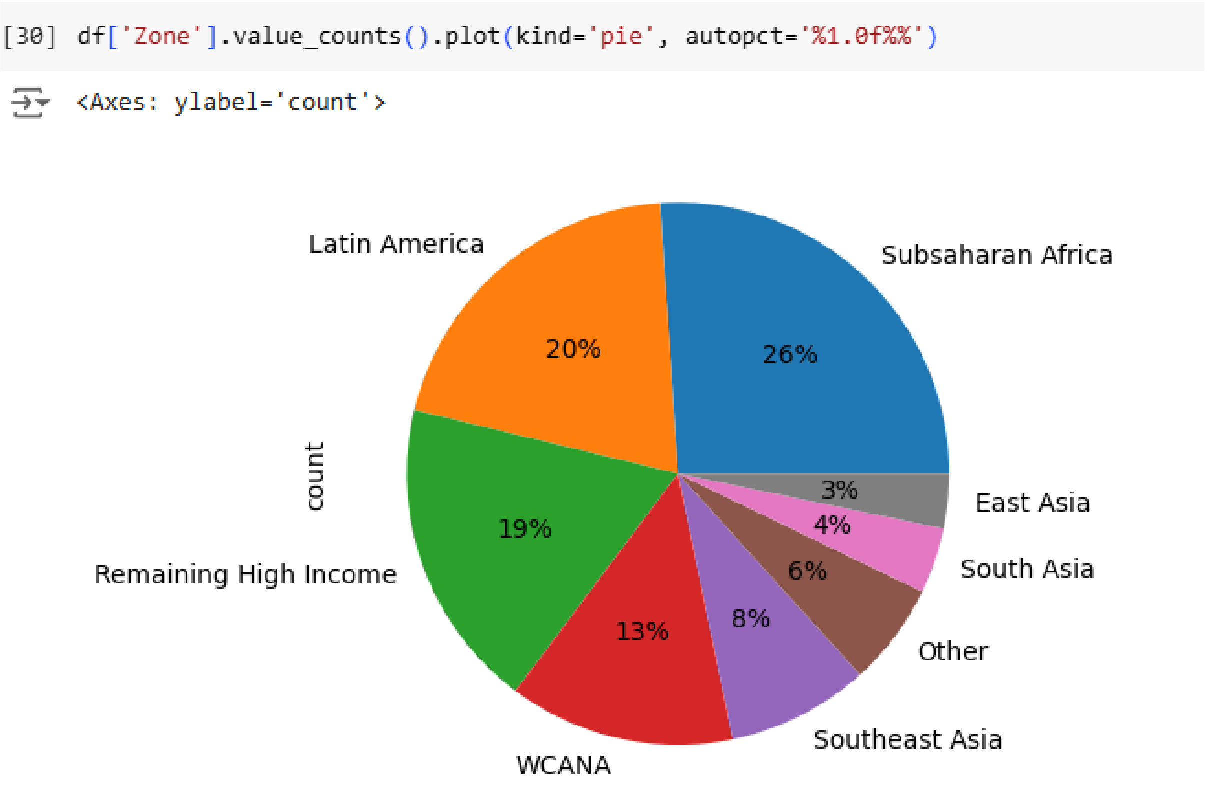


```
df.head(8).T
```

	0	1	2	3	4	5	6	7
Zone	WCANA	WCANA	WCANA	WCANA	WCANA	WCANA	WCANA	WCANA
Country	Afghanistan	Afghanistan	Afghanistan	Afghanistan	Afghanistan	Afghanistan	Afghanistan	Afghanistan
ISO3	AFG	AFG	AFG	AFG	AFG	AFG	AFG	AFG
Year	1961	1961	1961	1961	1961	1961	1961	1961
Population	8954000.0	8954000.0	8954000.0	8954000.0	8954000.0	8954000.0	8954000.0	8954000.0
Fortification	0	0	0	0	0	0	0	0
PCDEA	2997	2997	2997	2997	2997	2997	2997	2997
MDI	0.773239	0.773239	0.773239	0.773239	0.773239	0.773239	0.773239	0.773239
Tagname	VITC	ZN	VITA_RAE	CA	CU	RIBF	P	VITB6A
Micronutrient	Vitamin C	Zinc	Vitamin A	Calcium	Copper	Riboflavin	Phosphorus	Vitamin B6
Units	mg	mg	mcg	mg	mcg	mg	mg	mg
Estimated Intake	0.040922	0.001677	0.000275	0.700043	0.001979	0.00125	1.281776	0.002124
Requirements	0.049955	0.00177	0.000463	0.783092	0.000569	0.000828	0.625997	0.000916
Prevalence of Inadequate Intake	62.746266	58.326559	84.225561	63.814563	0.090734	4.437378	0.001393	0.000007

Fig. 1

Using the Pandas plot function, Fig. 2 shows the percentage of Zone category data. The pie chart shows large imbalance in data records to corresponding categories. This visualisation helps better illustrate the scale of imbalance.



```
[30] df['Zone'].value_counts().plot(kind='pie', autopct='%1.0f%%')
```

```
<Axes: ylabel='count'>
```

Fig. 2

Histograms are another useful visualisation to show the distribution of data for each category, showing the skewness of a specific feature. Commonly the aim is to have data in a normal distribution often referred to as a 'bell curve' shape. As the data is from the real world, it is accepted that skewness and imbalance is present in the visualisations. However, to such a large extent with large outliers can cause bias and skew predictions towards these outliers. As such, it is important to recognise what data must be kept, maintaining the characteristics of a feature, and what information is likely to induce bias.

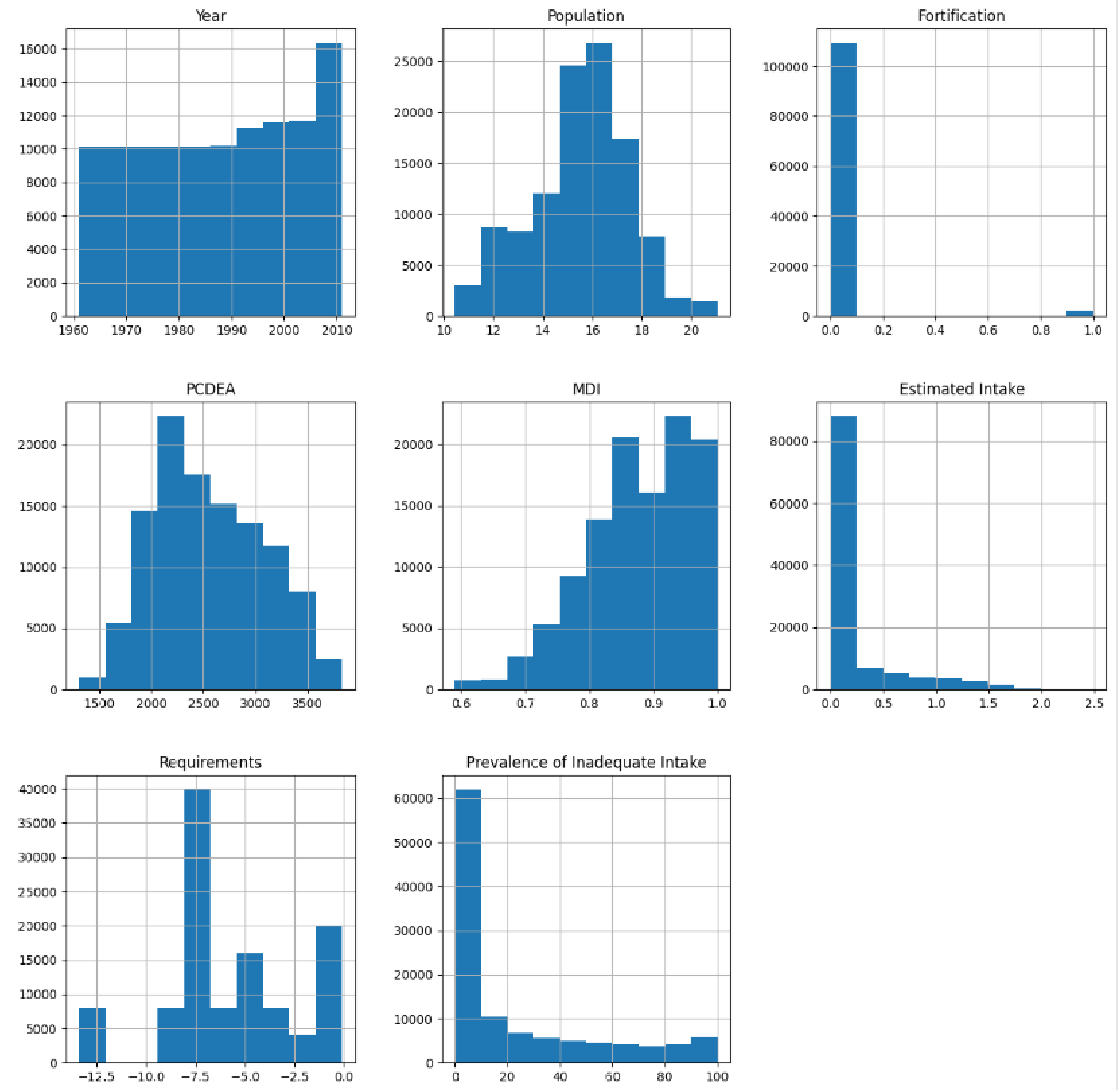


Fig. 3 (Using logarithmic transformation on population and requirements)

An example of a case where the graph must be interpreted is fortification in Fig. 3 where data is 0 or 1. Due to this, the 1 value are not outliers, but a fundamental characteristic to contextualise the following data of prevalence of inadequate intake.

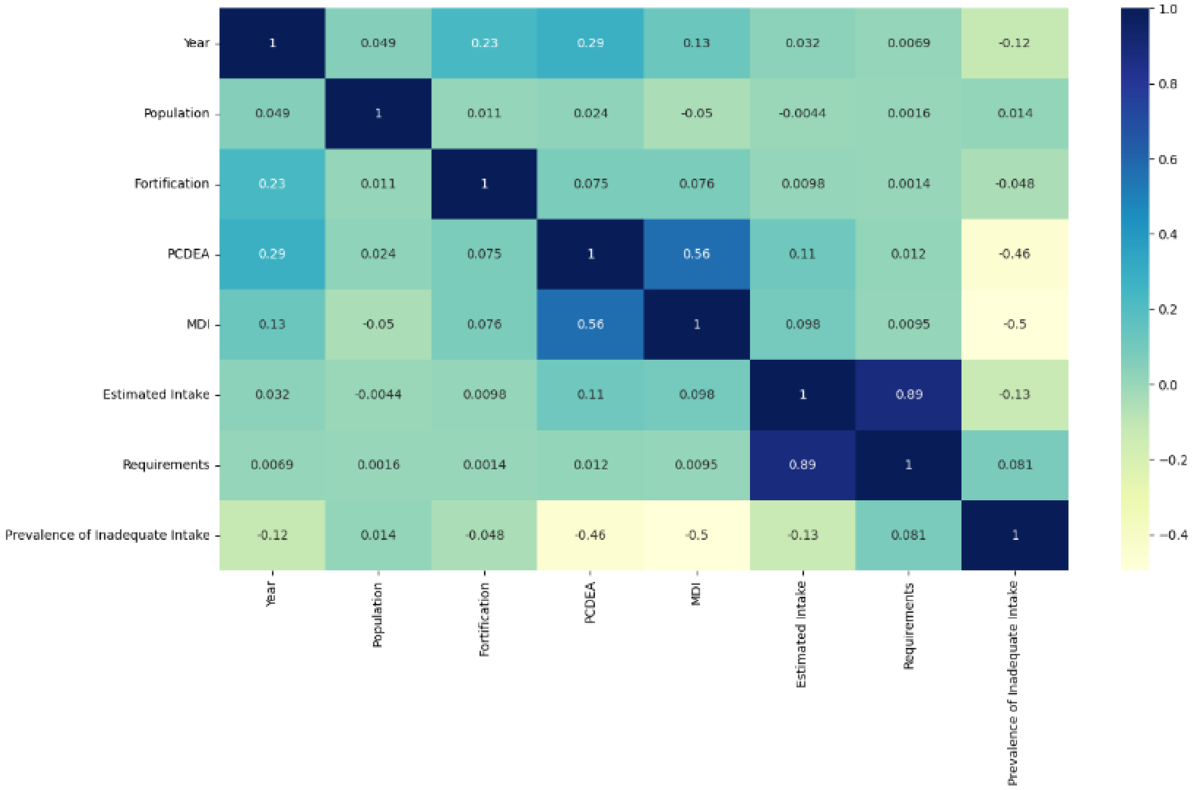


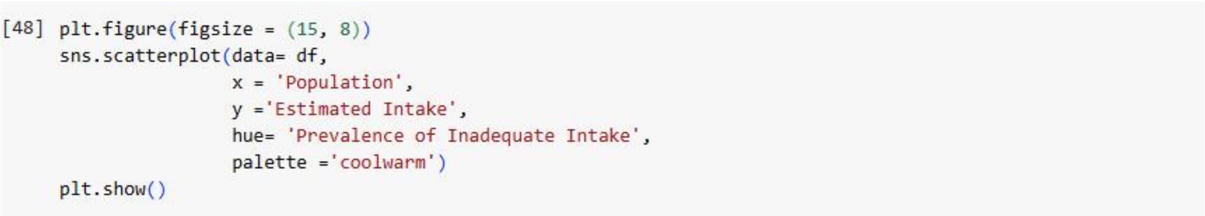
Fig. 4 (Heatmap)

The heatmap in Fig. 4 is produced using the '.corr' function on the data-frame shows the diagonal which correctly shows each feature correlates with itself.

The Heatmap on the raw data additionally shows strong positive correlation between Requirements and Estimated intake - a score of 0.89. This states that as a population's requirements for micronutrients increase, the estimated intake of a population also increase. This is interpreted as a government being able to meet the growing micronutrient on average.

The Heat map also shows a positive correlation between PCDEA and MDI - a score of 0.56. This states as National per capita daily energy availability increases, the micronutrient density index increases each year. This is interpreted as a government being better able to facilitate its growing population's micronutrient needs on average, thus increasing MDI.

The Heatmap shows a negative correlation between Prevalence of inadequate intake and the features: PCDEA and MDI - a score of -0.46 and -0.5 respectively. This is interpreted as the increase in Prevalence of inadequate intake correlates to decreased decreasing national capita daily energy capacity and decreasing Micronutrient density index.



```
[48] plt.figure(figsize = (15, 8))
     sns.scatterplot(data= df,
                     x = 'Population',
                     y ='Estimated Intake',
                     hue= 'Prevalence of Inadequate Intake',
                     palette ='coolwarm')
     plt.show()
```

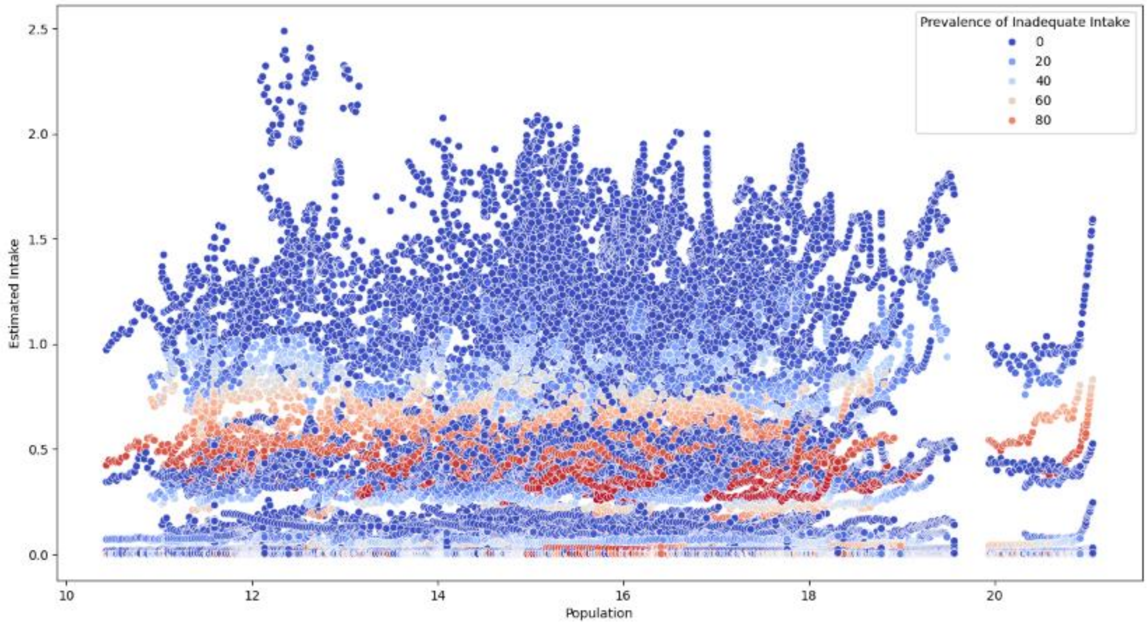


Fig. 5 (Scatterplot of population, estimated intake, prevalence of inadequate intake)

Visualisations such as the scatter-graph in Fig. 5 give a broader view of the features of a dataset and how they interact.

The scatterplot shows that the prevalence of inadequate intake disregards population factors and is more heavily concentrated where estimated intake is at 0.5. The highest estimated intake records have the lowest Prevalence of Inadequate Intake value which aligns with the view mentioned in the 2.1 Questions section, that high intake can lead to higher levels of obesity and other health issues. A further point for other reports to explore would be to see if these values are associated with global regions with high GDP or low GDP per capita and other forms of socio-economic analysis.

2.3 Exploratory Data Analysis Conclusions

1. Target column data (Zone) is imbalanced with regions having counts ranging from 29,120 to 3,640 values.

2. There is some correlation between some features, but most have no correlation to one another (Fig. 4).

3. The distribution of data is skewed but relevant to the context of the problem (Fig. 3).

4. There is a mix of categorical and numerical features in the dataset which need to be encoded.

5. The target feature has 8 different outputs.

3 Experimental Design

3.1 Identification of Chosen Supervised Learning Algorithm(s)

This report will focus on K Nearest Neighbour, Random Forests, Decision Tree Classifier, and Deep Learning. The other three algorithms: SVM, Logistic Regression and Multinomial Bayes will be referenced but used as comparative algorithms.

Algorithm	Description
Deep Learning	Deep learning utilises layered nodes with input and activation nodes which output based on their function type. Deep Learning utilises a loss function which determines the weight change in each epoch until the loss function is successfully minimised or epoch number is reached.
Decision Tree Classifier	(www.javatpoint.com, 2021) states Decision Tree is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules, and each leaf node represents the outcome. In essence: 1. Split data 2. Recursively iterate through tree 3. Decision making with Entropy Entropy in a decision tree measures the impurity of a node – i.e. the uncertainty.
K-Nearest Neighbour	(IBM, 2021) explains that K-nearest neighbours algorithm uses proximity to make classifications or predictions about the grouping of individual data points. It works off the assumption similar points can be found near one another. It is like Random Forests, utilising majority voting concept (requires > 50%).
Random Forests Classifier	Random Forests Classifier Algorithm merges the outputs of multiple decision trees to produce a single outcome. Random Forests uses ensemble modelling, where a classification goes through majority voting. In essence: 1. subset of data points and features are selected for constructing each decision tree 2. Individual decision trees are constructed for each sample 3. Each decision tree will generate output

	4. Final output is considered based on majority voting

3.2 Identification of Appropriate Evaluation Techniques

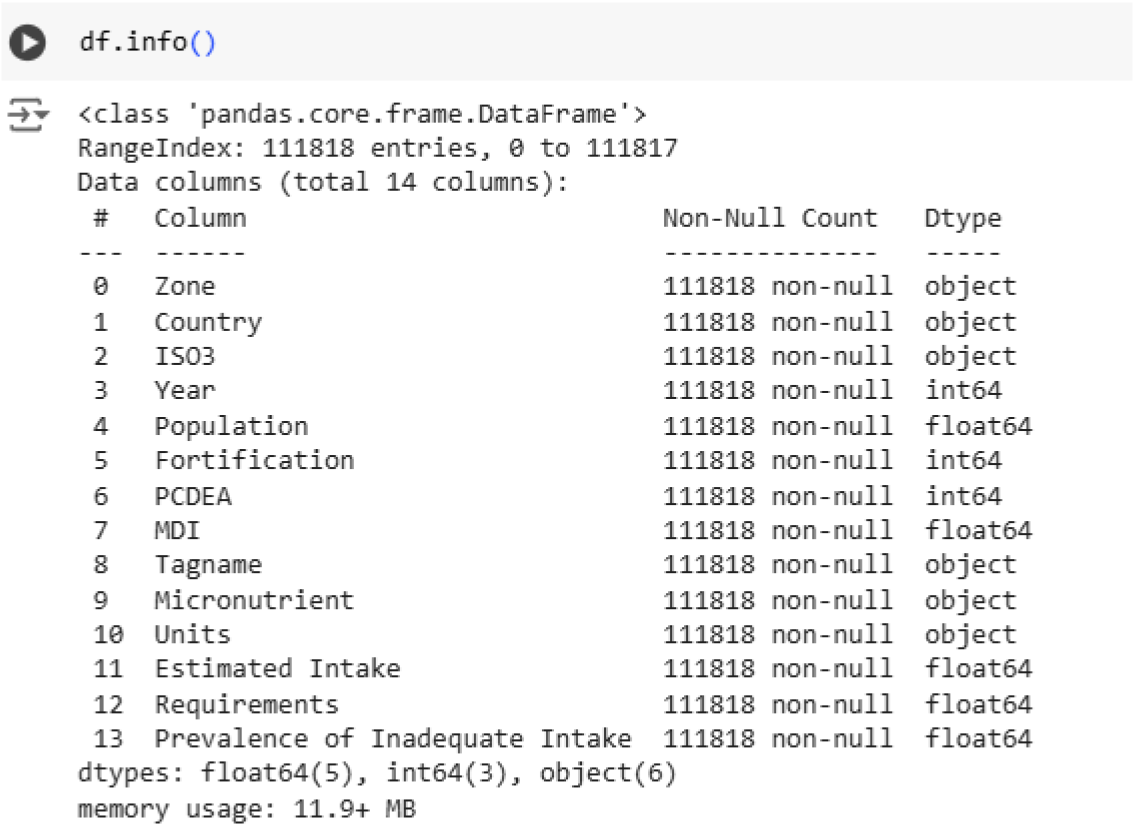
Using accuracy, precision, recall and f1-score helps to explain effectiveness for datasets with imbalance .

Accuracy, precision and recall will give the surface level insight into the predictions the model makes and will be easy to compare to the other models.

However, most comparisons will be made utilising the F1-score, which gives a more contextualised view of the confusion matrix, considering false positive and negative predictions. In essence, the F1-score is the harmonic mean of precision and recall.

3.3 Data Cleaning and Pre-Processing Transformations

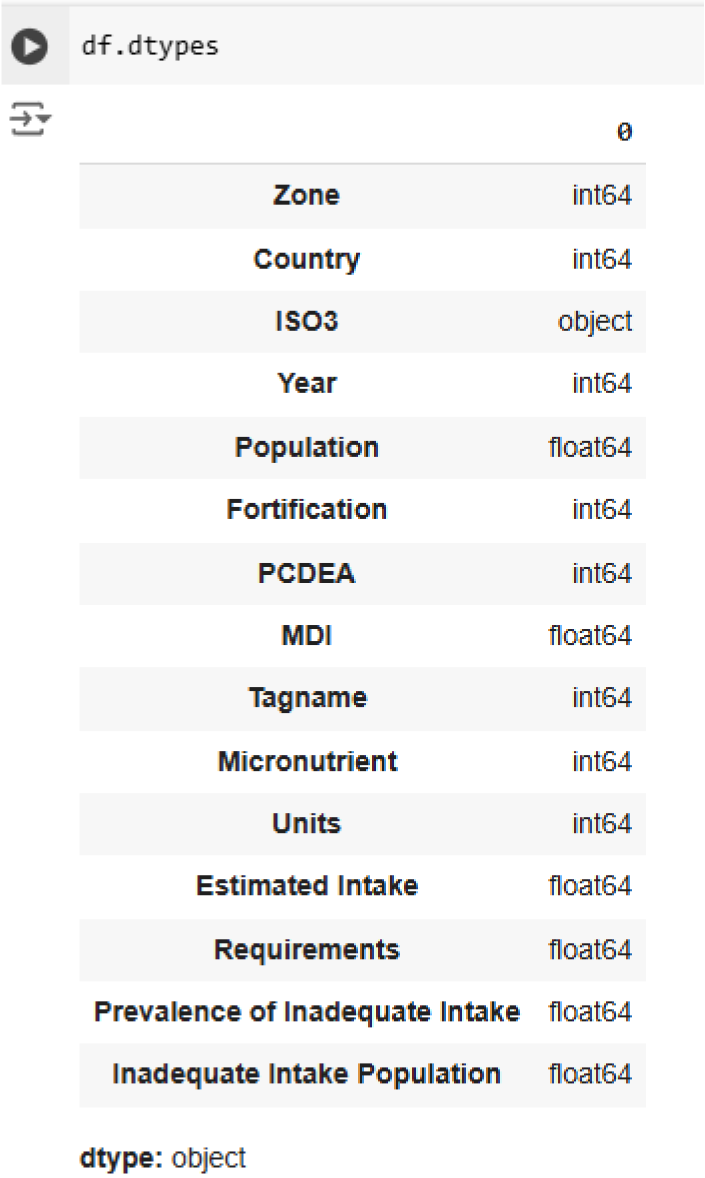
As shown in Fig. 6, the dataset has 111,818 entries with each feature holding 111818 non-null values, meaning no data needs to be imputed. It should be noted that the data types differ for each feature.



```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 111818 entries, 0 to 111817
Data columns (total 14 columns):
 #   Column                         Non-Null Count   Dtype  
---  ------                         --------------   -----  
 0   Zone                           111818 non-null  object 
 1   Country                        111818 non-null  object 
 2   ISO3                           111818 non-null  object 
 3   Year                           111818 non-null  int64  
 4   Population                     111818 non-null  float64
 5   Fortification                  111818 non-null  int64  
 6   PCDEA                          111818 non-null  int64  
 7   MDI                            111818 non-null  float64
 8   Tagname                        111818 non-null  object 
 9   Micronutrient                  111818 non-null  object 
 10  Units                          111818 non-null  object 
 11  Estimated Intake               111818 non-null  float64
 12  Requirements                   111818 non-null  float64
 13  Prevalence of Inadequate Intake  111818 non-null  float64
dtypes: float64(5), int64(3), object(6)
memory usage: 11.9+ MB
```

Fig. 6 (Before Pre-Processing)

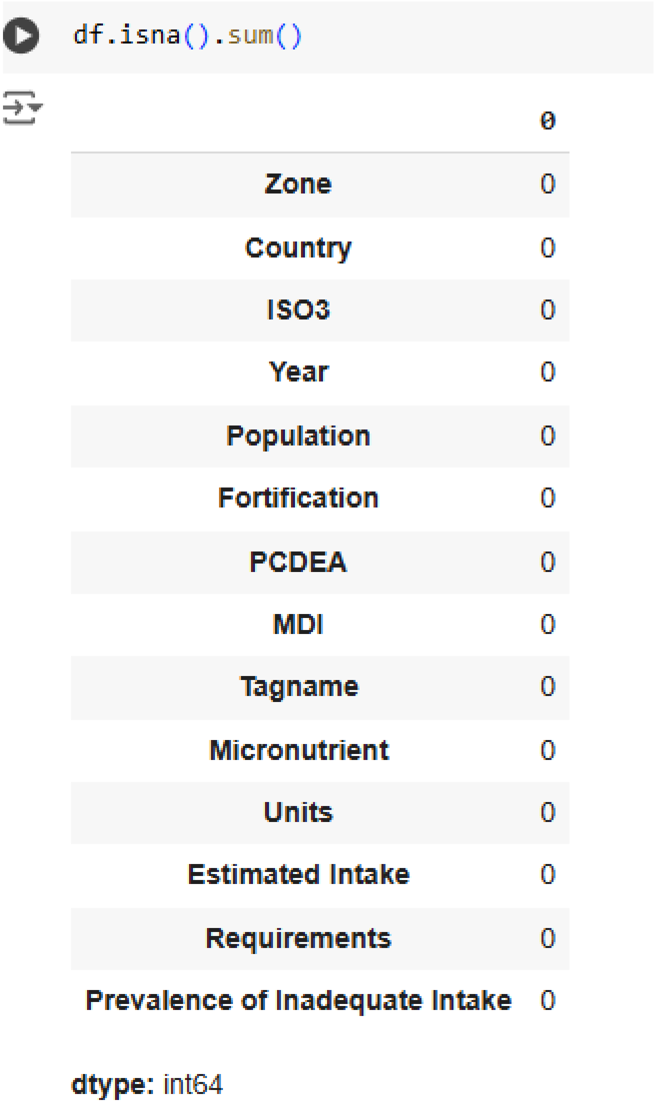


```
df.dtypes
```

```
                                             0
                        Zone             int64
                     Country             int64
                        ISO3            object
                        Year             int64
                  Population           float64
               Fortification             int64
                       PCDEA             int64
                         MDI           float64
                     Tagname             int64
               Micronutrient             int64
                       Units             int64
            Estimated Intake           float64
                Requirements           float64
Prevalence of Inadequate Intake        float64
  Inadequate Intake Population         float64

dtype: object
```

Fig. 7 (After Pre-Processing)

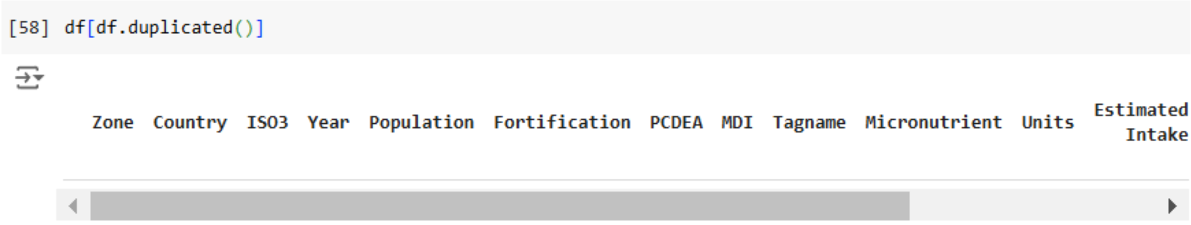
In Fig. 7 a mix of both integer and float values are present. It is important to keep nuance of each feature which is a float, as rounding the values would give an inaccurate representation of data, and with the specified quantity or micronutrients being in smaller units, it is important that the precision remains unchanged.



```
df.isna().sum()
```

```
                                     0
                        Zone         0
                     Country         0
                        ISO3         0
                        Year         0
                  Population         0
               Fortification         0
                       PCDEA         0
                         MDI         0
                     Tagname         0
               Micronutrient         0
                       Units         0
            Estimated Intake         0
                Requirements         0
Prevalence of Inadequate Intake      0

dtype: int64
```

Fig. 8

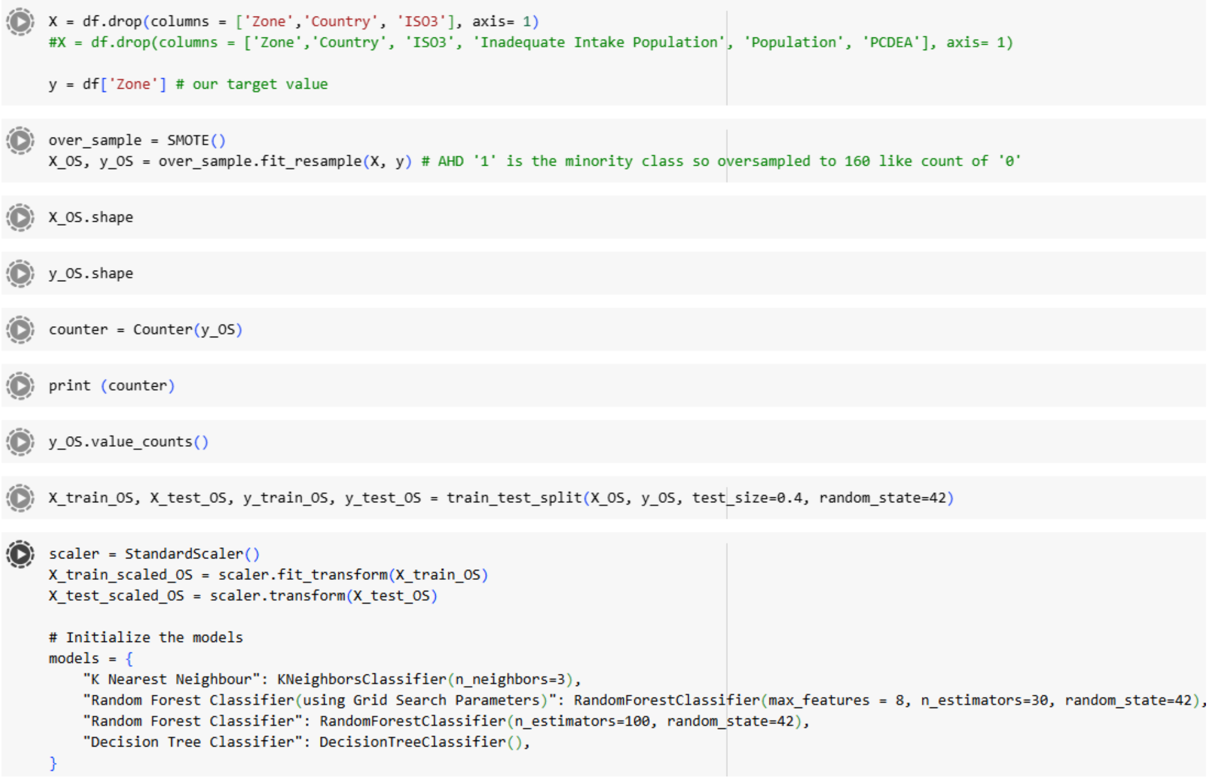


```
[58] df[df.duplicated()]
```

```
                                                                                             Estimated
Zone  Country  ISO3  Year  Population  Fortification  PCDEA  MDI  Tagname  Micronutrient  Units  Intake
```

Fig. 9

In Fig. 8, the dataset can be determined to be already clean with no null values and there are no duplicate values as shown in Fig. 9.



```
X = df.drop(columns = ['Zone','Country', 'ISO3'], axis= 1)
#X = df.drop(columns = ['Zone','Country', 'ISO3', 'Inadequate Intake Population', 'Population', 'PCDEA'], axis= 1)

y = df['Zone'] # our target value
```

```
over_sample = SMOTE()
X_OS, y_OS = over_sample.fit_resample(X, y) # AHD '1' is the minority class so oversampled to 160 like count of '0'
```

```
X_OS.shape
```

```
y_OS.shape
```

```
counter = Counter(y_OS)
```

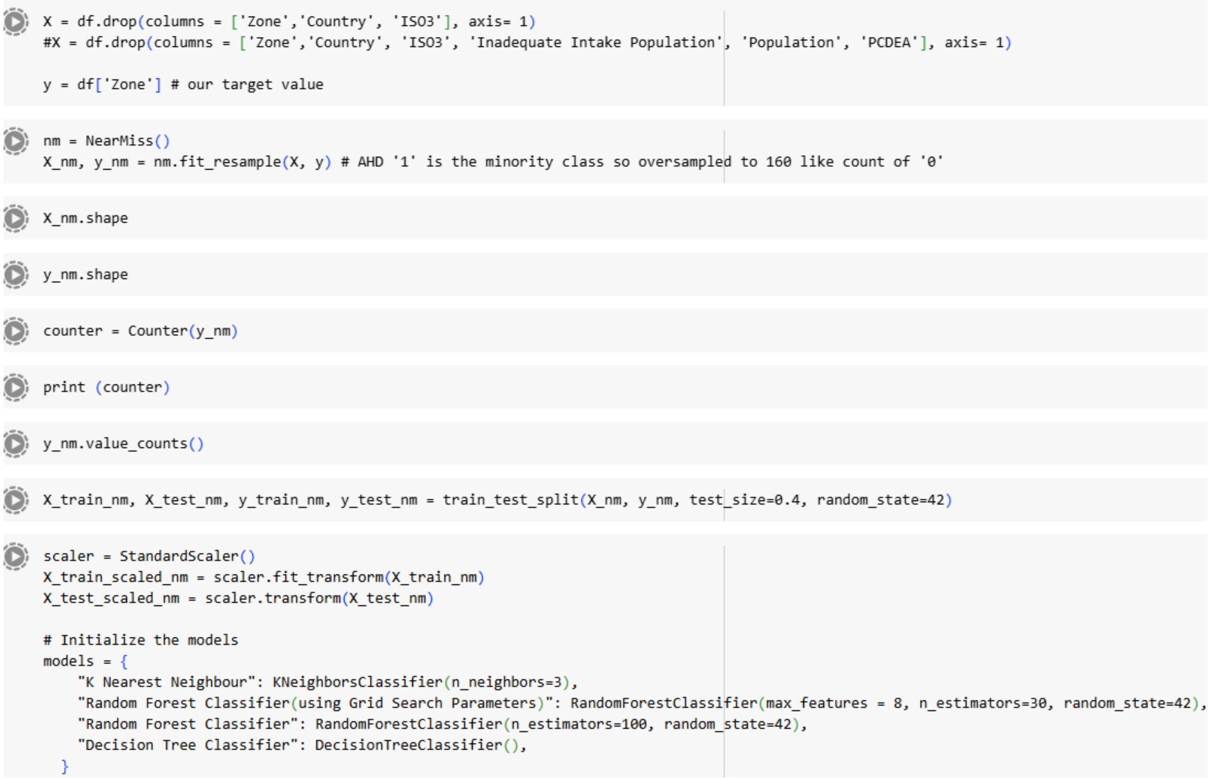
```
print (counter)
```

```
y_OS.value_counts()
```

```
X_train_OS, X_test_OS, y_train_OS, y_test_OS = train_test_split(X_OS, y_OS, test_size=0.4, random_state=42)
```

```
scaler = StandardScaler()
X_train_scaled_OS = scaler.fit_transform(X_train_OS)
X_test_scaled_OS = scaler.transform(X_test_OS)

# Initialize the models
models = {
    "K Nearest Neighbour": KNeighborsClassifier(n_neighbors=3),
    "Random Forest Classifier(using Grid Search Parameters)": RandomForestClassifier(max_features = 8, n_estimators=30, random_state=42),
    "Random Forest Classifier": RandomForestClassifier(n_estimators=100, random_state=42),
    "Decision Tree Classifier": DecisionTreeClassifier(),
}
```

Fig. 10



```
X = df.drop(columns = ['Zone','Country', 'ISO3'], axis= 1)
#X = df.drop(columns = ['Zone','Country', 'ISO3', 'Inadequate Intake Population', 'Population', 'PCDEA'], axis= 1)

y = df['Zone'] # our target value
```

```
nm = NearMiss()
X_nm, y_nm = nm.fit_resample(X, y) # AHD '1' is the minority class so oversampled to 160 like count of '0'
```

```
X_nm.shape
```

```
y_nm.shape
```

```
counter = Counter(y_nm)
```

```
print (counter)
```

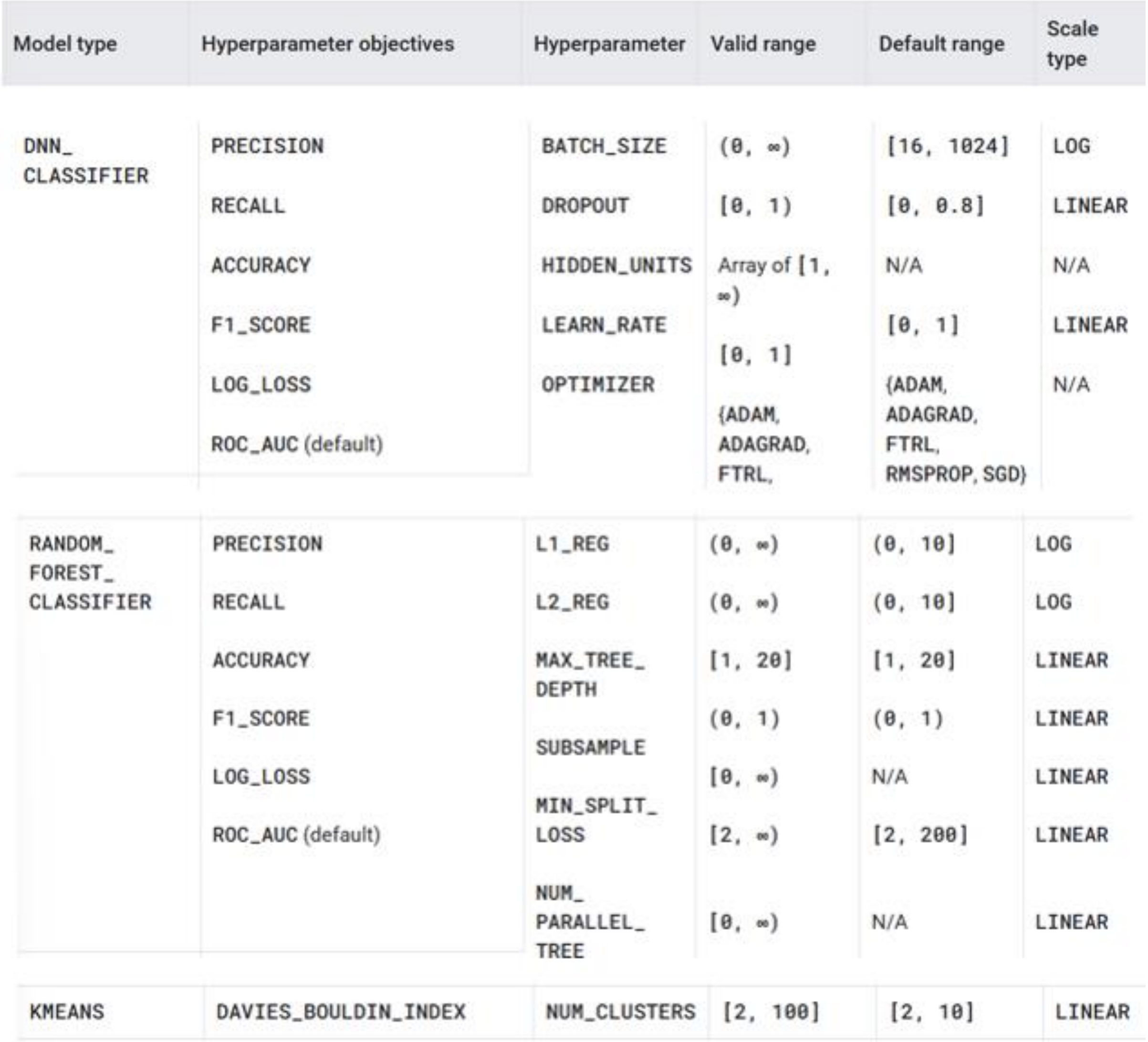
```
y_nm.value_counts()
```

```
X_train_nm, X_test_nm, y_train_nm, y_test_nm = train_test_split(X_nm, y_nm, test_size=0.4, random_state=42)
```

```
scaler = StandardScaler()
X_train_scaled_nm = scaler.fit_transform(X_train_nm)
X_test_scaled_nm = scaler.transform(X_test_nm)

# Initialize the models
models = {
    "K Nearest Neighbour": KNeighborsClassifier(n_neighbors=3),
    "Random Forest Classifier(using Grid Search Parameters)": RandomForestClassifier(max_features = 8, n_estimators=30, random_state=42),
    "Random Forest Classifier": RandomForestClassifier(n_estimators=100, random_state=42),
    "Decision Tree Classifier": DecisionTreeClassifier(),
  }
```

Fig. 11

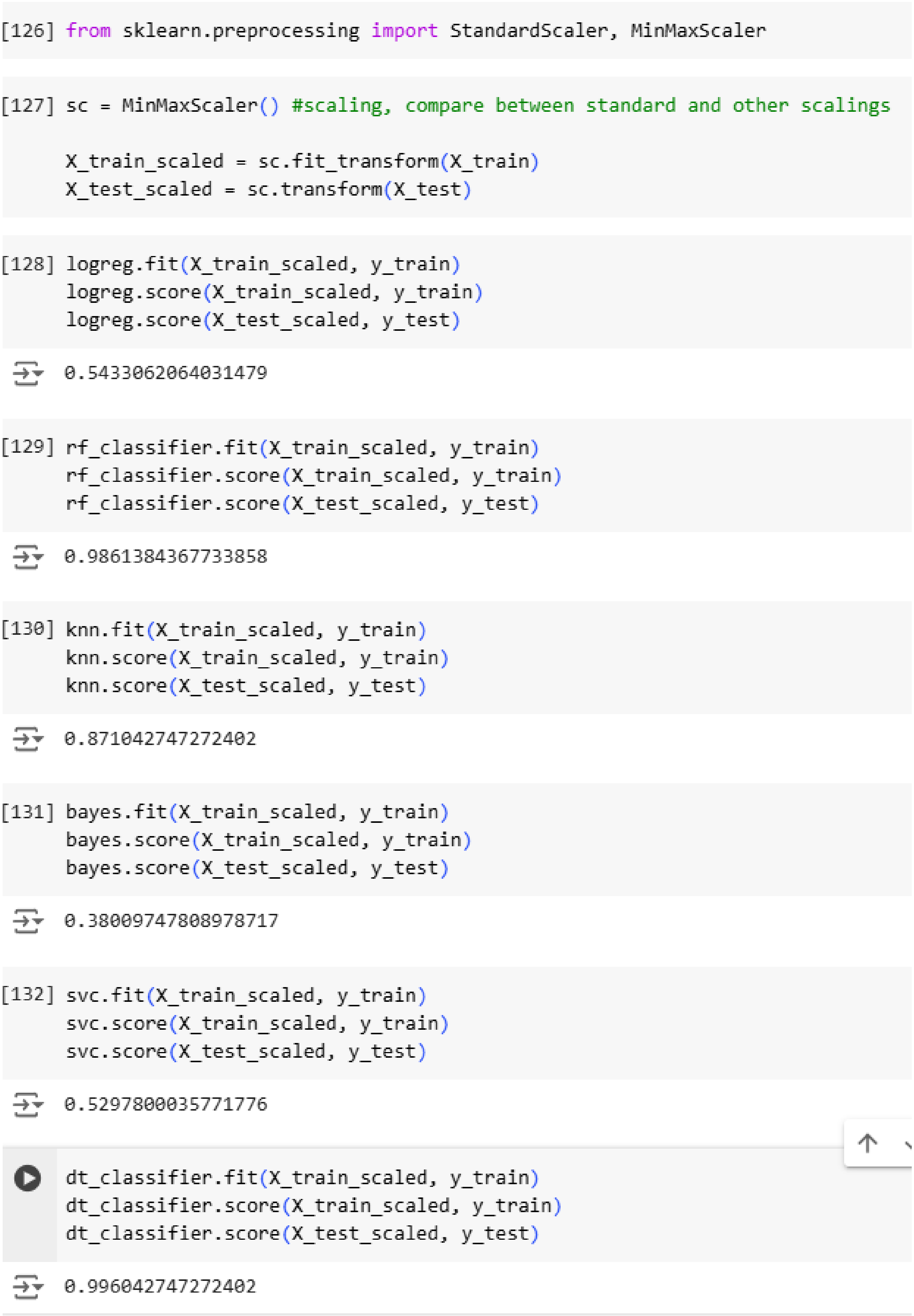
Synthetic Minority Oversampling Technique (SMOTE) is applied in Fig. 10 and Near-Miss Under-sampling Technique Fig. 11 is applied as a comparison to the execution of the unbalanced dataset data. As seen in both figures, X (the features to train model) and y (the target feature). X and y variables are then applied to fit their respective resampling technique before they are split into train and test variables, which are then scaled by a scaler algorithm.



Model type	Hyperparameter objectives	Hyperparameter	Valid range	Default range	Scale type
DNN_ CLASSIFIER	PRECISION	BATCH_SIZE	(0, ∞)	[16, 1024]	LOG
	RECALL	DROPOUT	[0, 1)	[0, 0.8]	LINEAR
	ACCURACY	HIDDEN_UNITS	Array of [1, ∞)	N/A	N/A
	F1_SCORE	LEARN_RATE	[0, 1]	[0, 1]	LINEAR
	LOG_LOSS	OPTIMIZER	{ADAM, ADAGRAD, FTRL,	{ADAM, ADAGRAD, FTRL, RMSPROP, SGD}	N/A
	ROC_AUC (default)				
RANDOM_ FOREST_ CLASSIFIER	PRECISION	L1_REG	(0, ∞)	(0, 10]	LOG
	RECALL	L2_REG	(0, ∞)	(0, 10]	LOG
	ACCURACY	MAX_TREE_ DEPTH	[1, 20]	[1, 20]	LINEAR
	F1_SCORE	SUBSAMPLE	(0, 1)	(0, 1)	LINEAR
	LOG_LOSS	MIN_SPLIT_ LOSS	[0, ∞)	N/A	LINEAR
	ROC_AUC (default)		[2, ∞)	[2, 200]	LINEAR
		NUM_ PARALLEL_ TREE	[0, ∞)	N/A	LINEAR
KMEANS	DAVIES_BOULDIN_INDEX	NUM_CLUSTERS	[2, 100]	[2, 10]	LINEAR

Fig. 12 (Hyperparameter tuning table from (Google Cloud, 2024))

The algorithms in focus for this report and their adjustable hyperparameters are shown in Fig. 12 which the article (Google Cloud, 2024) states “in machine learning, hyperparameter tuning identifies a set of optimal hyperparameters for a learning algorithm. A hyperparameter is a model argument whose value is set before the learning process begins”.



```
[126] from sklearn.preprocessing import StandardScaler, MinMaxScaler
```

```
[127] sc = MinMaxScaler() #scaling, compare between standard and other scalings

      X_train_scaled = sc.fit_transform(X_train)
      X_test_scaled = sc.transform(X_test)
```

```
[128] logreg.fit(X_train_scaled, y_train)
      logreg.score(X_train_scaled, y_train)
      logreg.score(X_test_scaled, y_test)
```

```
0.5433062064031479
```

```
[129] rf_classifier.fit(X_train_scaled, y_train)
      rf_classifier.score(X_train_scaled, y_train)
      rf_classifier.score(X_test_scaled, y_test)
```

```
0.9861384367733858
```

```
[130] knn.fit(X_train_scaled, y_train)
      knn.score(X_train_scaled, y_train)
      knn.score(X_test_scaled, y_test)
```

```
0.871042747272402
```

```
[131] bayes.fit(X_train_scaled, y_train)
      bayes.score(X_train_scaled, y_train)
      bayes.score(X_test_scaled, y_test)
```

```
0.38009747808978717
```

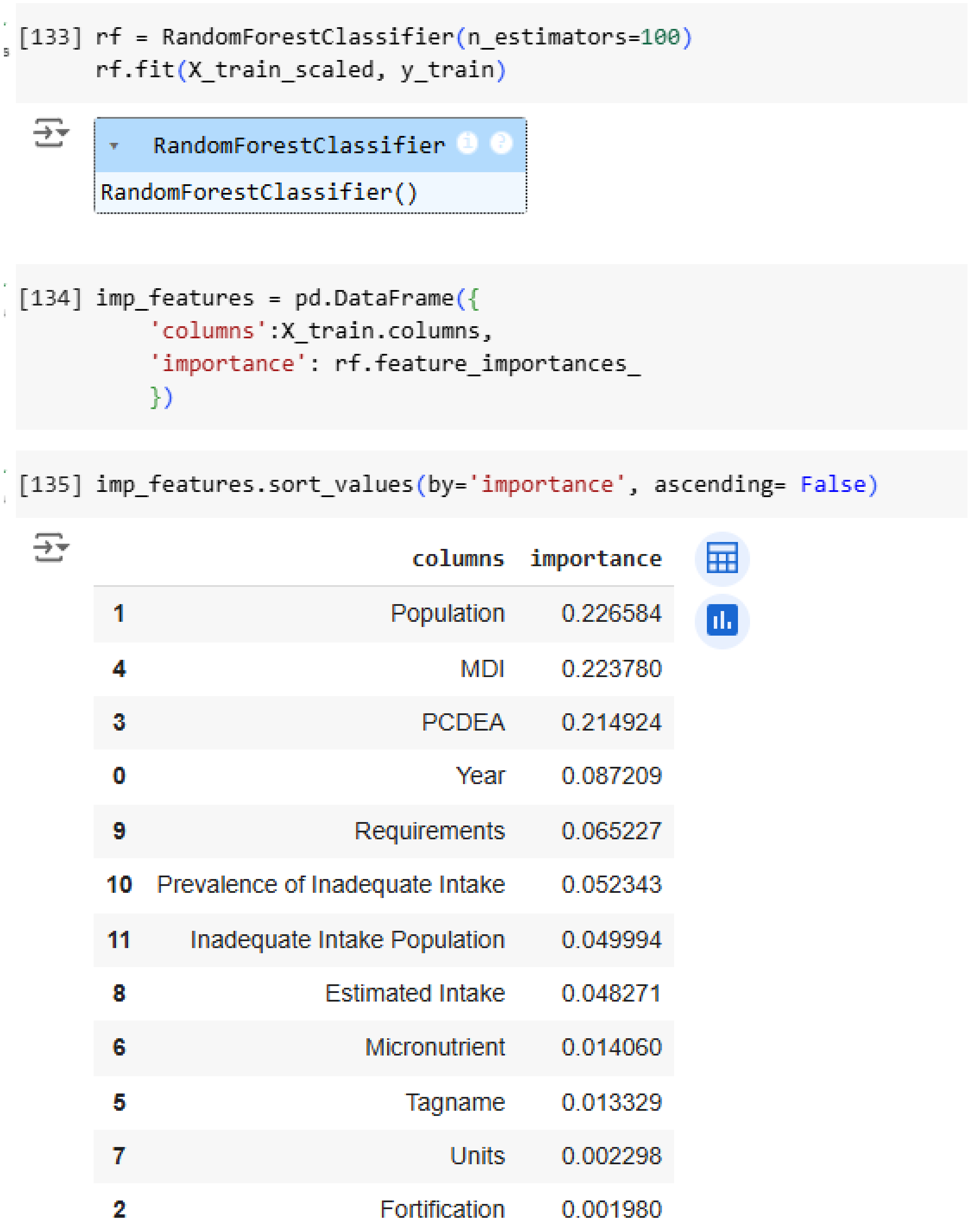
```
[132] svc.fit(X_train_scaled, y_train)
      svc.score(X_train_scaled, y_train)
      svc.score(X_test_scaled, y_test)
```

```
0.5297800035771776
```

```
dt_classifier.fit(X_train_scaled, y_train)
dt_classifier.score(X_train_scaled, y_train)
dt_classifier.score(X_test_scaled, y_test)
```

```
0.996042747272402
```

Fig. 13



```
[133] rf = RandomForestClassifier(n_estimators=100)
      rf.fit(X_train_scaled, y_train)
```

```
RandomForestClassifier
RandomForestClassifier()
```

```
[134] imp_features = pd.DataFrame({
          'columns':X_train.columns,
          'importance': rf.feature_importances_
          })
```

```
[135] imp_features.sort_values(by='importance', ascending= False)
```

```
                         columns  importance
 1                    Population    0.226584
 4                           MDI    0.223780
 3                         PCDEA    0.214924
 0                          Year    0.087209
 9                  Requirements    0.065227
10  Prevalence of Inadequate Intake 0.052343
11     Inadequate Intake Population 0.049994
 8               Estimated Intake   0.048271
 6                  Micronutrient   0.014060
 5                        Tagname   0.013329
 7                          Units   0.002298
 2                  Fortification   0.001980
```

Fig. 14

As shown in Fig .13, values for each feature are scaled. (Punyakeerthi BL, 2024) states feature scaling is a fundamental preprocessing step in machine learning aimed at ensuring that numerical features have a similar scale.

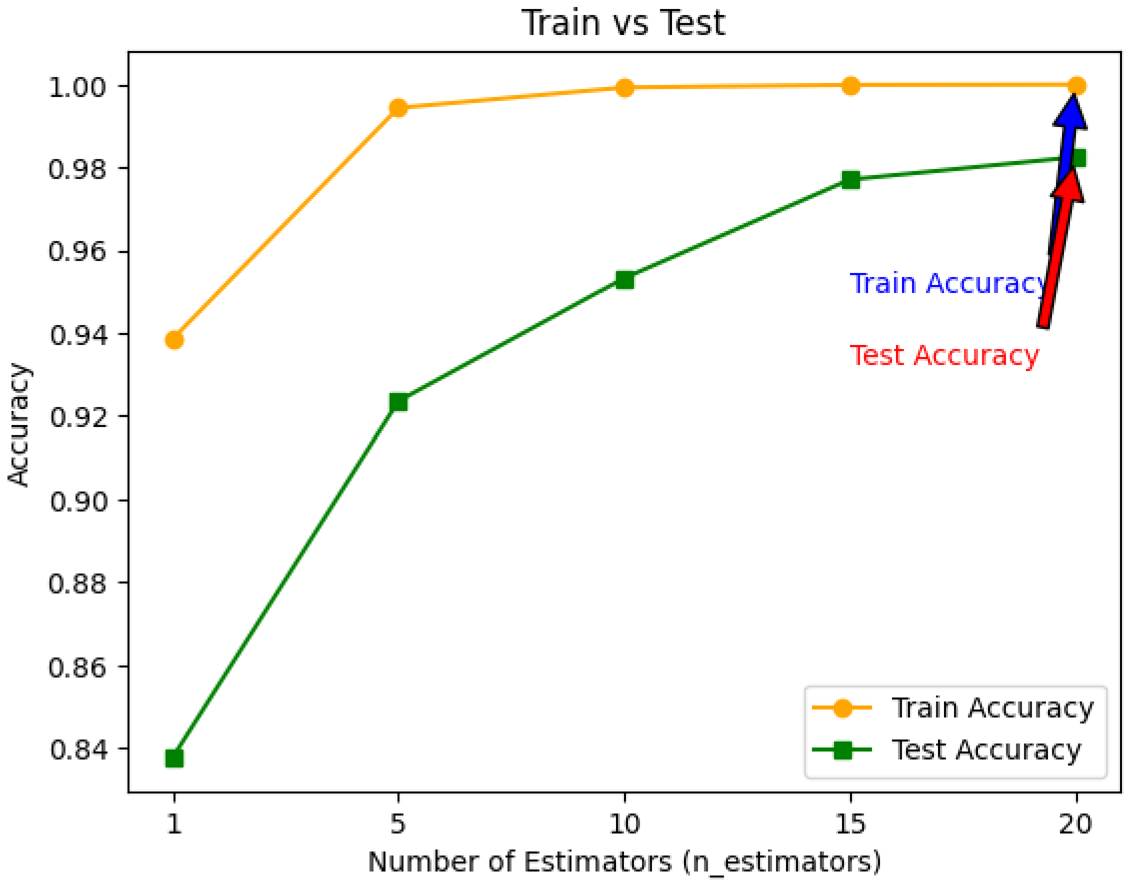
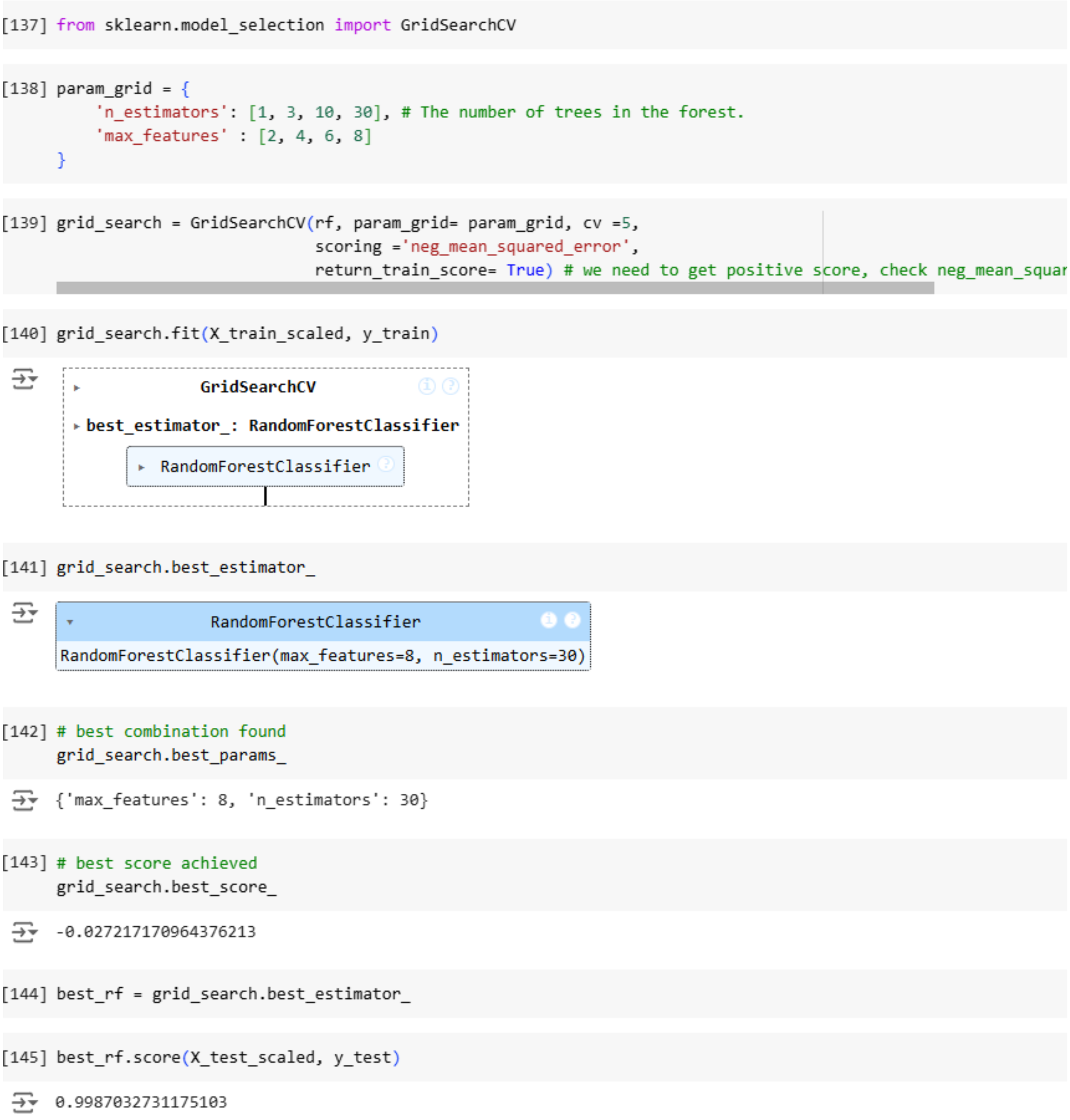


Fig. 15 (Hyperparameter testing for Random Forest Classifier)



```
[137] from sklearn.model_selection import GridSearchCV
```

```
[138] param_grid = {
          'n_estimators': [1, 3, 10, 30], # The number of trees in the forest.
          'max_features' : [2, 4, 6, 8]
      }
```

```
[139] grid_search = GridSearchCV(rf, param_grid= param_grid, cv =5,
                                 scoring ='neg_mean_squared_error',
                                 return_train_score= True) # we need to get positive score, check neg_mean_squar
```

```
[140] grid_search.fit(X_train_scaled, y_train)
```

```
GridSearchCV
best_estimator_: RandomForestClassifier
RandomForestClassifier
```

```
[141] grid_search.best_estimator_
```

```
RandomForestClassifier
RandomForestClassifier(max_features=8, n_estimators=30)
```

```
[142] # best combination found
      grid_search.best_params_
```

```
{'max_features': 8, 'n_estimators': 30}
```

```
[143] # best score achieved
      grid_search.best_score_
```

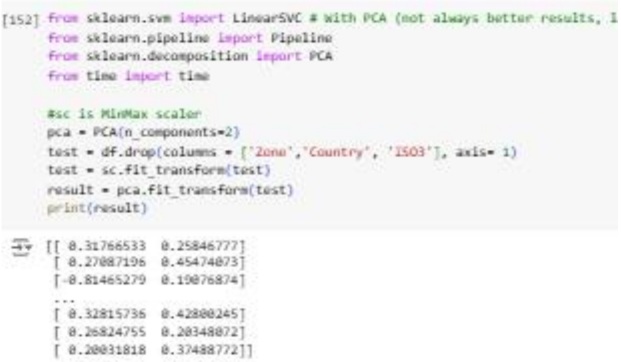
```
-0.027217170964376213
```

```
[144] best_rf = grid_search.best_estimator_
```

```
[145] best_rf.score(X_test_scaled, y_test)
```

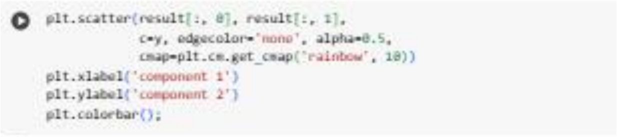
```
0.9987032731175103
```

Fig. 16 (Grid Search for Hyperparameter tuning for Random Forest Classifier)

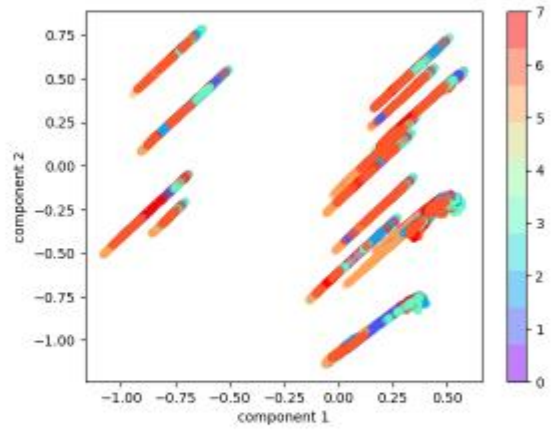


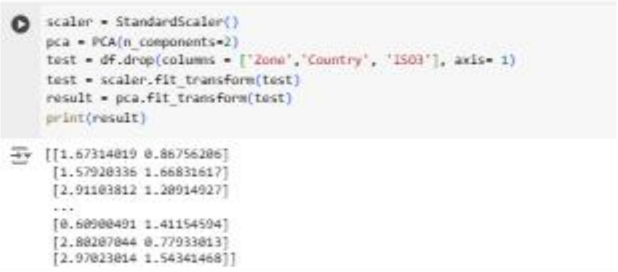
```
[152] from sklearn.svm import LinearSVC # With PCA (not always better results, I
      from sklearn.pipeline import Pipeline
      from sklearn.decomposition import PCA
      from time import time

      #sc is MinMax scaler
      pca = PCA(n_components=2)
      test = df.drop(columns = ['Zone','Country', 'ISO3'], axis= 1)
      test = sc.fit_transform(test)
      result = pca.fit_transform(test)
      print(result)
```

```
[[ 0.31766533  0.25846777]
 [ 0.27087196  0.45474073]
 [-0.81465279  0.19076874]
 ...
 [ 0.32815736  0.42800245]
 [ 0.26824755  0.20348072]
 [ 0.20031818  0.37488772]]
```



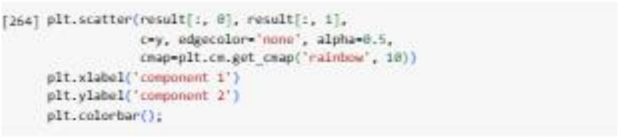
```
plt.scatter(result[:, 0], result[:, 1],
            c=y, edgecolor='none', alpha=0.5,
            cmap=plt.cm.get_cmap('rainbow', 10))
plt.xlabel('component 1')
plt.ylabel('component 2')
plt.colorbar();
```





```
scaler = StandardScaler()
pca = PCA(n_components=2)
test = df.drop(columns = ['Zone','Country', 'ISO3'], axis= 1)
test = scaler.fit_transform(test)
result = pca.fit_transform(test)
print(result)
```

```
[[1.67314019 0.86756206]
 [1.57920336 1.66831617]
 [2.91103812 1.20914927]
 ...
 [0.60900491 1.41154594]
 [2.88207044 0.77933013]
 [2.97023014 1.54341468]]
```



```
[264] plt.scatter(result[:, 0], result[:, 1],
                  c=y, edgecolor='none', alpha=0.5,
                  cmap=plt.cm.get_cmap('rainbow', 10))
      plt.xlabel('component 1')
      plt.ylabel('component 2')
      plt.colorbar();
```

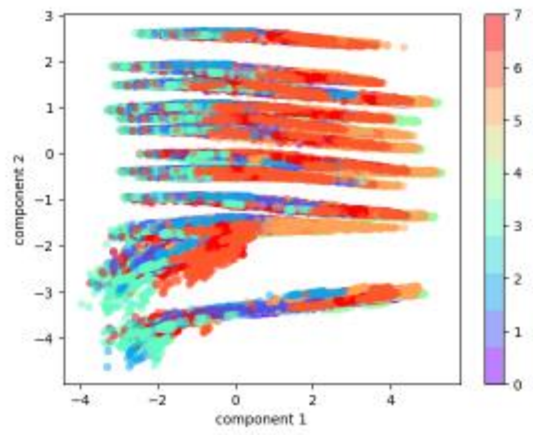


Fig. 17 (Comparing Standard and Min Max scaler output)

Principal components Fig. 17 in compute the percentage of variance (information) accounted for by each component.

The full data is a 14-dimensional point cloud, and these points are the projection of each data point along the directions with the largest variance (highest information axes). Essentially, we have found the optimal stretch and rotation in 14-dimensional space that allows us to see the layout of the data in two dimensions, and we have done this in an unsupervised manner—that is, without reference to the labels.

i.e. There are clear splits and groupings of data represented by both scalers with similar performance as shown in Fig. 13 when compared to results in Fig. 31 and Fig. 32.

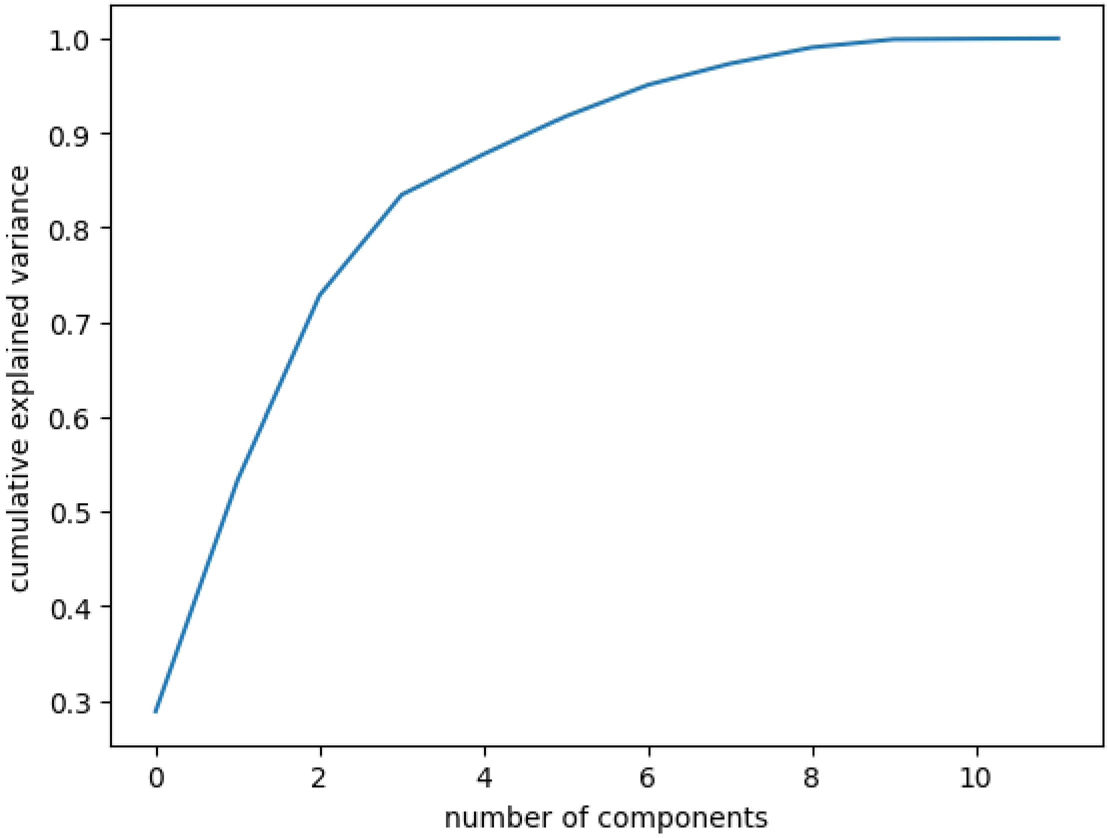


Fig. 18 (PCA result Random Forest)

PCA analysis in Fig. 18 shows 80% information from dataset can be maintained with 5 components. This would reduce the computational requirement without losing performance of Random Forest model.

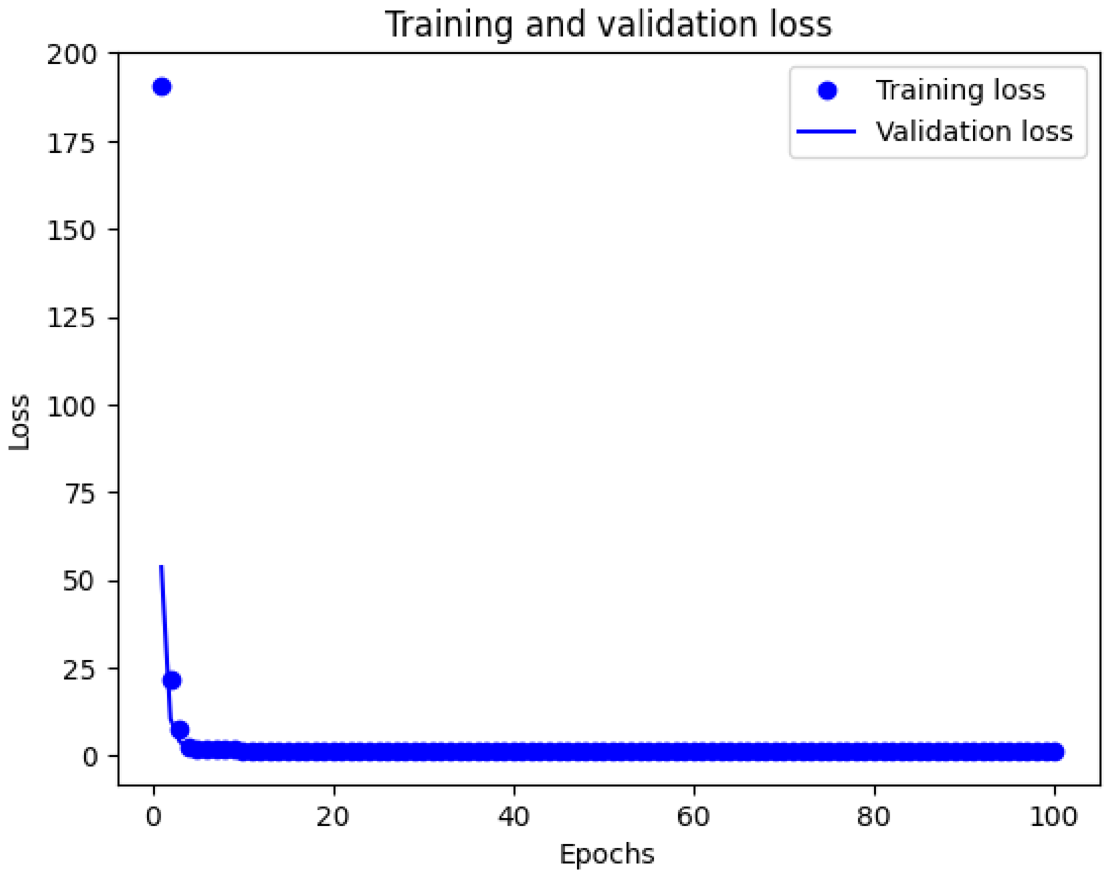


Fig. 19 (Loss score each epoch in Deep Learning)

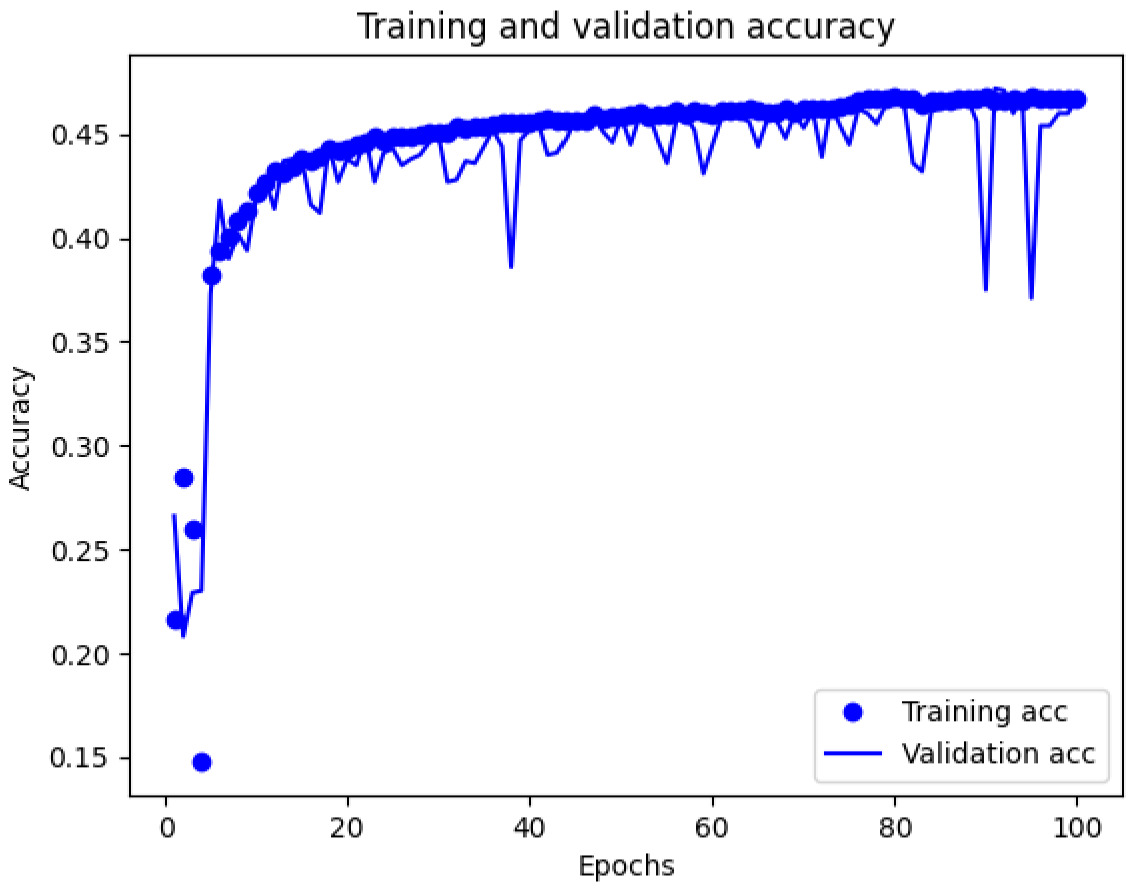
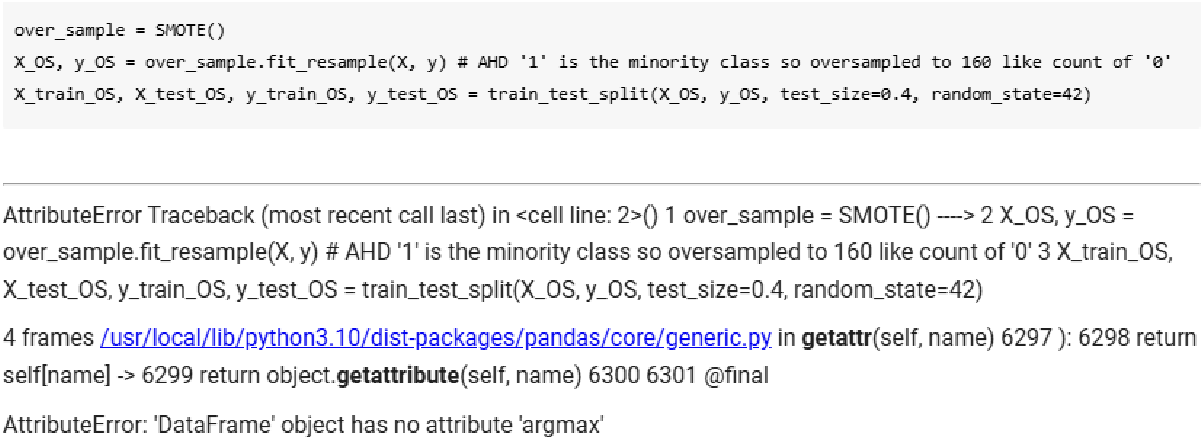


Fig. 20 (Accuracy of training and validation over epochs)

In Fig. 19 it is seen that as number of epochs increase from around 10, validation and training loss decreases significantly before beginning to plateau with little to no improvement as epochs increase. Similarly in Fig. 20 after 10 epochs, there is a linear positive correlation in training accuracy. Validation accuracy increases to extreme peaks and troughs throughout the increasing epochs but generally trends in line with Training accuracy in a positive correlation.

3.4 Limitations and Options



```
over_sample = SMOTE()
X_OS, y_OS = over_sample.fit_resample(X, y) # AHD '1' is the minority class so oversampled to 160 like count of '0'
X_train_OS, X_test_OS, y_train_OS, y_test_OS = train_test_split(X_OS, y_OS, test_size=0.4, random_state=42)
```

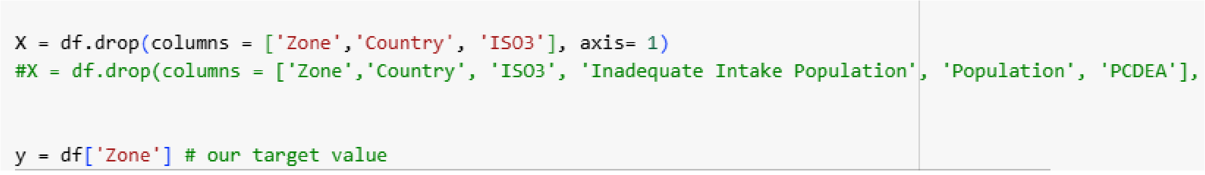
```
AttributeError Traceback (most recent call last) in <cell line: 2>() 1 over_sample = SMOTE() ----> 2 X_OS, y_OS = over_sample.fit_resample(X, y) # AHD '1' is the minority class so oversampled to 160 like count of '0' 3 X_train_OS, X_test_OS, y_train_OS, y_test_OS = train_test_split(X_OS, y_OS, test_size=0.4, random_state=42)

4 frames /usr/local/lib/python3.10/dist-packages/pandas/core/generic.py in getattr(self, name) 6297 ): 6298 return self[name] -> 6299 return object.getattribute(self, name) 6300 6301 @final

AttributeError: 'DataFrame' object has no attribute 'argmax'
```

Fig. 21

- Limitations is the near miss and smote do not like y being encoded with multiple categories, causes a classifier issue (as seen in (fig. 21).
- Deep learning confusion matrix does not like multiple categories, will compare zone by zone.
- Deep Learning multiple labels could be selected by prediction, not limited to one category prediction.
- Overfitting, model too specific and ungeneralised due to some redundant features or unintentional data leakage.

4 Predictive Modelling / Model Development

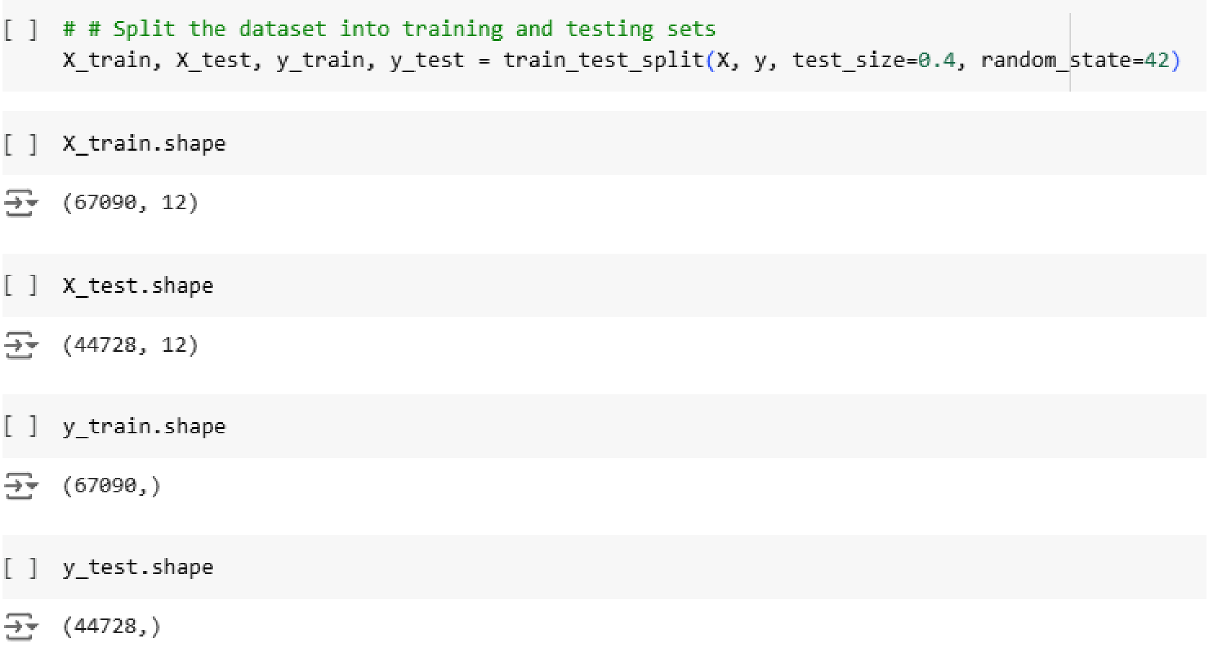
4.1 Splitting the Dataset



```
X = df.drop(columns = ['Zone','Country', 'ISO3'], axis= 1)
#X = df.drop(columns = ['Zone','Country', 'ISO3', 'Inadequate Intake Population', 'Population', 'PCDEA'],


y = df['Zone'] # our target value
```

Fig. 22

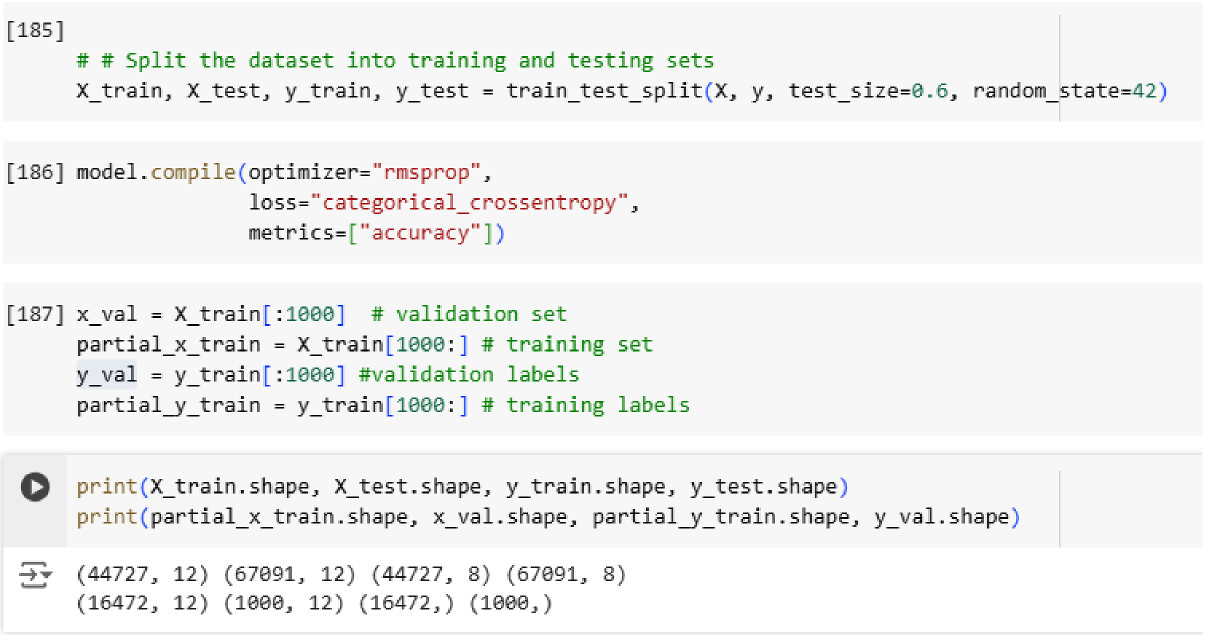
Except for Deep learning, the split of X and Y split keeps the target data ‘zone’ unseen to the model predicting the data. This will split will be utilised for each model. (Tim Mucci, 2024) states Data Leakage occurs when a model uses information during training that would not be available at time of prediction and is a key concept to be avoided. He states there are two types of leakage: Target and Train-Test contamination. The commented X in figure 1, shows a previous iteration which reduced the overfitting/data leakage issue by hiding more data from the model. However, this made the classification more akin to a black box, with features and patterns which were not visible to the human in the minimal data given, would give a prediction. The aim of developing the model is to have it generalised when test with unseen data.



```
[ ]  # # Split the dataset into training and testing sets
     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state=42)

[ ]  X_train.shape

     (67090, 12)

[ ]  X_test.shape

     (44728, 12)

[ ]  y_train.shape

     (67090,)

[ ]  y_test.shape

     (44728,)
```

Fig. 23

Except for Deep Learning, the train test split is 60/40, which provides an opportunity to utilise validation. However, in this case it is used to prevent the overfitting of each model on the training data.



```
[185]
      # # Split the dataset into training and testing sets
      X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.6, random_state=42)
```

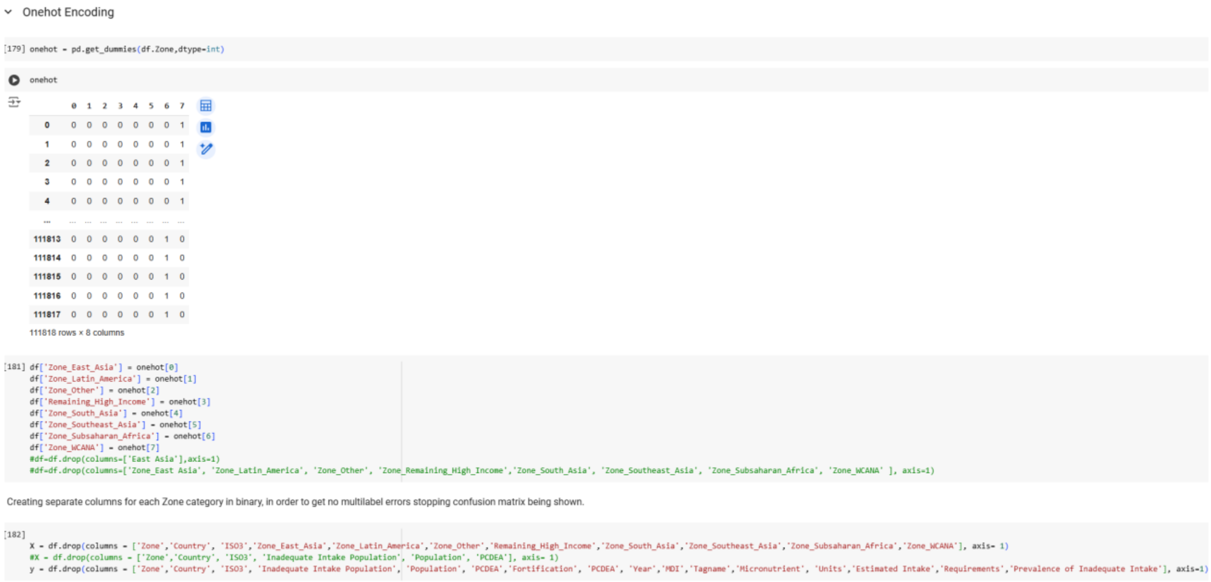
```
[186] model.compile(optimizer="rmsprop",
                    loss="categorical_crossentropy",
                    metrics=["accuracy"])
```

```
[187] x_val = X_train[:1000]  # validation set
      partial_x_train = X_train[1000:] # training set
      y_val = y_train[:1000] #validation labels
      partial_y_train = y_train[1000:] # training labels
```

```
print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)
print(partial_x_train.shape, x_val.shape, partial_y_train.shape, y_val.shape)
```

```
(44727, 12) (67091, 12) (44727, 8) (67091, 8)
(16472, 12) (1000, 12) (16472,) (1000,)
```

Fig. 24



Onehot Encoding

```
[179] onehot = pd.get_dummies(df.Zone,dtype=int)
```

```
onehot
```

	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	1
1	0	0	0	0	0	0	0	1
2	0	0	0	0	0	0	0	1
3	0	0	0	0	0	0	0	1
4	0	0	0	0	0	0	0	1
...	...	...	...	...	...	...	...	...
111813	0	0	0	0	0	0	1	0
111814	0	0	0	0	0	0	1	0
111815	0	0	0	0	0	0	1	0
111816	0	0	0	0	0	0	1	0
111817	0	0	0	0	0	0	1	0

111818 rows × 8 columns

```
[181] df['Zone_East_Asia'] = onehot[0]
      df['Zone_Latin_America'] = onehot[1]
      df['Zone_Other'] = onehot[2]
      df['Remaining_High_Income'] = onehot[3]
      df['Zone_South_Asia'] = onehot[4]
      df['Zone_Southeast_Asia'] = onehot[5]
      df['Zone_Subsaharan_Africa'] = onehot[6]
      df['Zone_WCANA'] = onehot[7]
      #df=df.drop(columns=['East Asia'],axis=1)
      #df=df.drop(columns=['Zone_East Asia', 'Zone_Latin_America', 'Zone_Other', 'Zone_Remaining_High_Income','Zone_South_Asia', 'Zone_Southeast_Asia', 'Zone_Subsaharan_Africa', 'Zone_WCANA' ], axis=1)
```

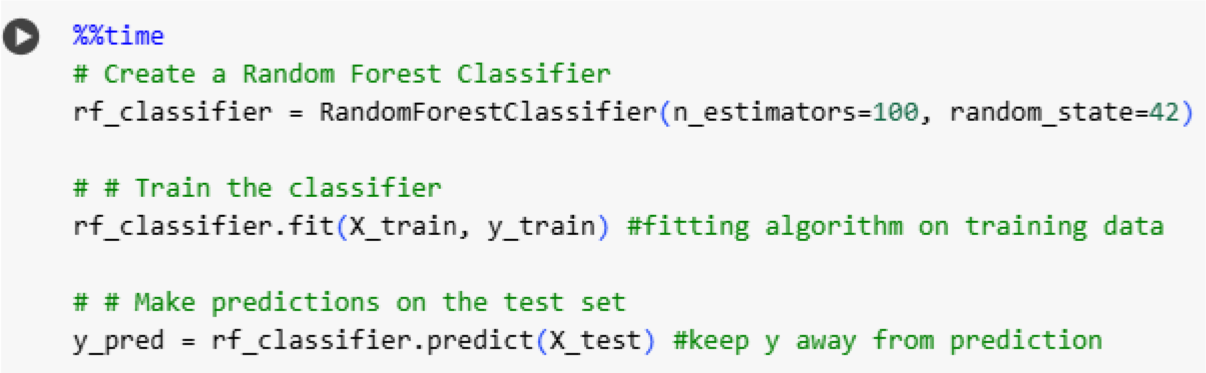
Creating separate columns for each Zone category in binary, in order to get no multilabel errors stopping confusion matrix being shown.

```
[182]
      X = df.drop(columns = ['Zone','Country', 'ISO3','Zone_East_Asia','Zone_Latin_America','Zone_Other','Remaining_High_Income','Zone_South_Asia','Zone_Southeast_Asia','Zone_Subsaharan_Africa','Zone_WCANA'], axis= 1)
      #X = df.drop(columns = ['Zone','Country', 'ISO3', 'Inadequate Intake Population', 'Population', 'PCDEA'], axis= 1)
      y = df.drop(columns = ['Zone','Country', 'ISO3', 'Inadequate Intake Population', 'Population', 'PCDEA','Fortification', 'PCDEA', 'Year','MDI','Tagname','Micronutrient', 'Units','Estimated Intake','Requirements','Prevalence of Inadequate Intake'], axis=1)
```

Fig. 25 (Encoding for Deep Learning Example of y values)

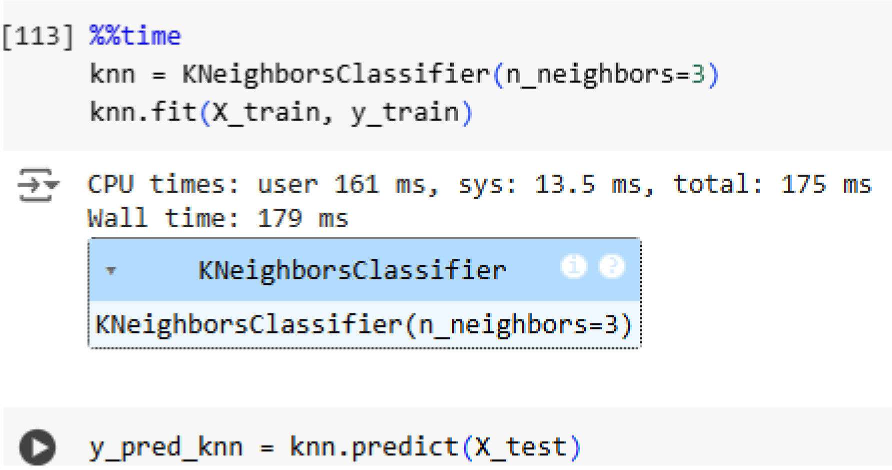
For deep learning, the use of one-hot encoding in pre-processing as shown in Fig.25 separated each categorical input of Zone, utilising the ‘pd.get_dummies’ to encode into a data-frame.

4.2 The Predictive Modelling Process



```
%%time
# Create a Random Forest Classifier
rf_classifier = RandomForestClassifier(n_estimators=100, random_state=42)

# # Train the classifier
rf_classifier.fit(X_train, y_train) #fitting algorithm on training data

# # Make predictions on the test set
y_pred = rf_classifier.predict(X_test) #keep y away from prediction
```

Fig. 26



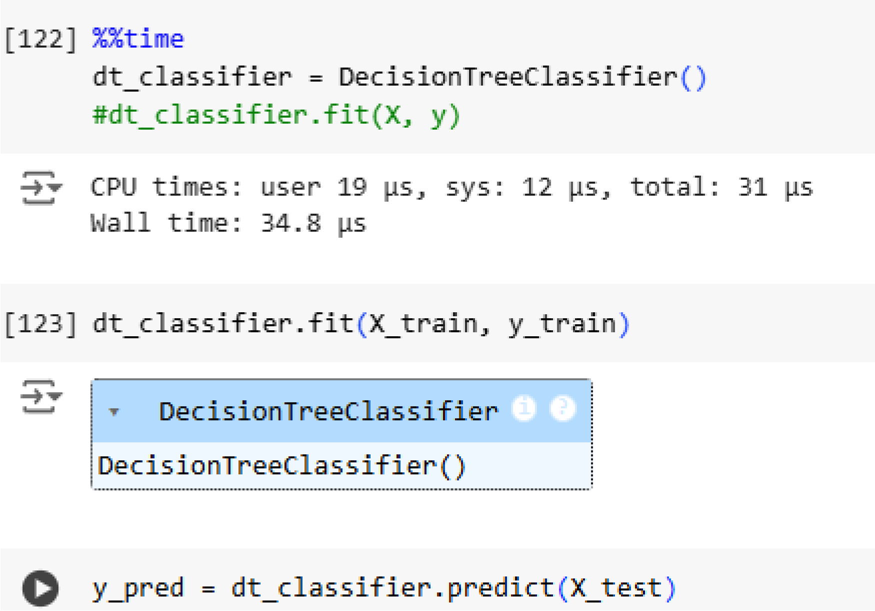
```
[113] %%time
      knn = KNeighborsClassifier(n_neighbors=3)
      knn.fit(X_train, y_train)
```

```
CPU times: user 161 ms, sys: 13.5 ms, total: 175 ms
Wall time: 179 ms
```

```
KNeighborsClassifier
KNeighborsClassifier(n_neighbors=3)
```

```
y_pred_knn = knn.predict(X_test)
```

Fig. 27



```
[122] %%time
      dt_classifier = DecisionTreeClassifier()
      #dt_classifier.fit(X, y)
```

```
CPU times: user 19 µs, sys: 12 µs, total: 31 µs
Wall time: 34.8 µs
```

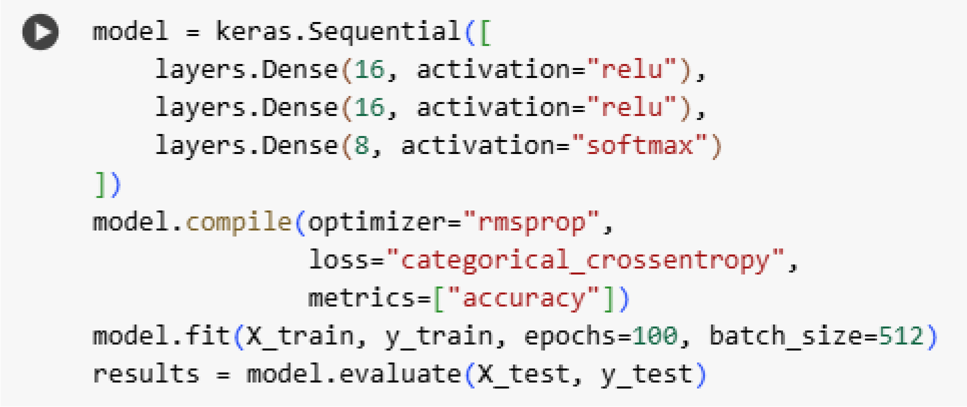
```
[123] dt_classifier.fit(X_train, y_train)
```

```
DecisionTreeClassifier
DecisionTreeClassifier()
```

```
y_pred = dt_classifier.predict(X_test)
```

Fig. 28

To fit values before applying them to algorithms such as in Fig. 26, Fig. 27, Fig. 28 with a scaler, SMOTE, Near-Miss adjusted variables, change the predict parameter to the adjusted variable.



```
model = keras.Sequential([
    layers.Dense(16, activation="relu"),
    layers.Dense(16, activation="relu"),
    layers.Dense(8, activation="softmax")
])
model.compile(optimizer="rmsprop",
              loss="categorical_crossentropy",
              metrics=["accuracy"])
model.fit(X_train, y_train, epochs=100, batch_size=512)
results = model.evaluate(X_test, y_test)
```

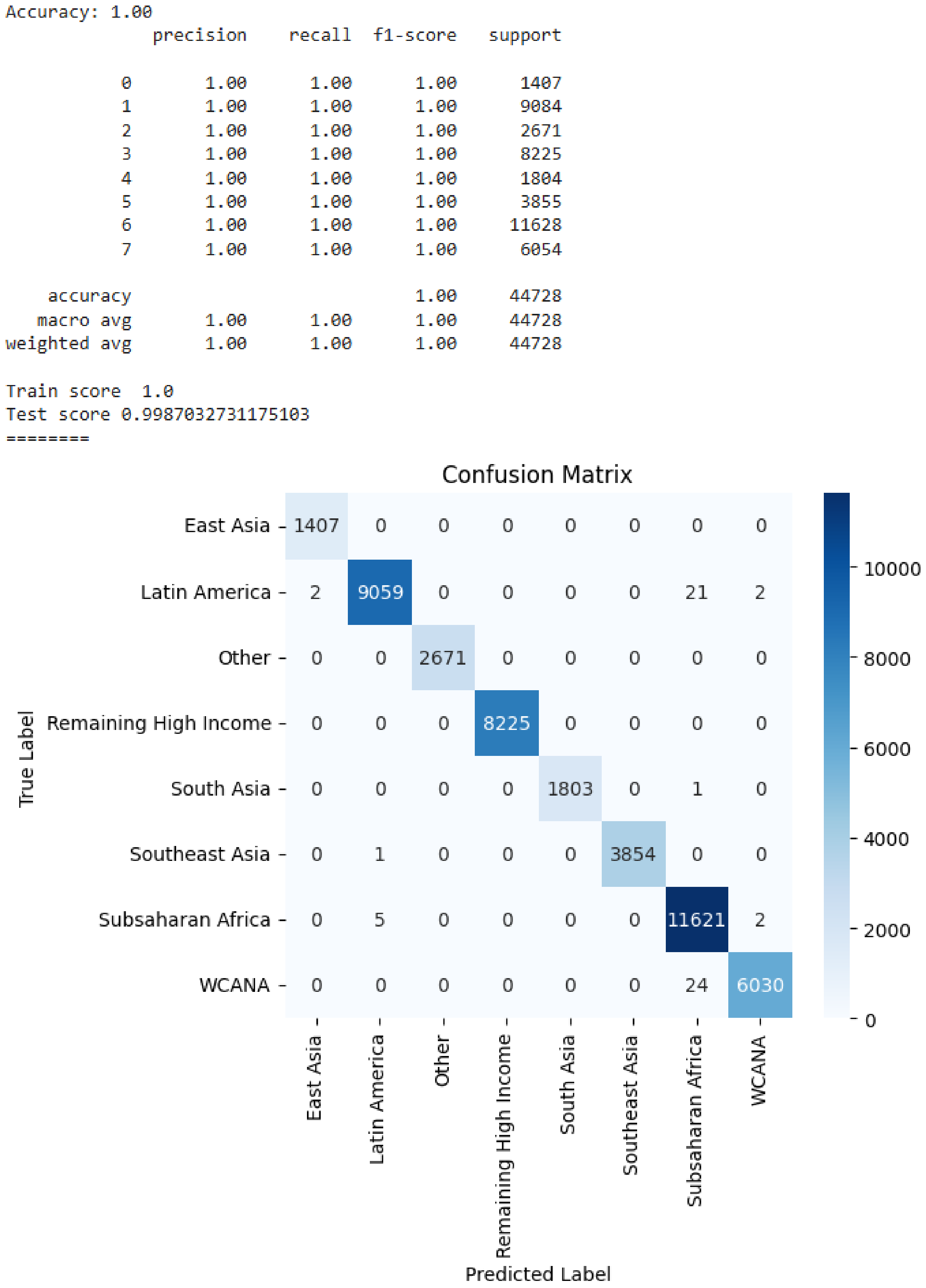
Fig. 29

As stated in (Furnieles, 2022), SoftMax is an activation function like sigmoid. Soft max applies to multiclass problems which is suited to the problem as 'zone' is separated over 8 potential classifications. Additionally soft max is already vectorised "meaning that takes a vector with the same number of entries as classes we have and outputs another vector where each component represents the probability of belonging to that class".

4.3 Evaluation Results on “Seen” Data

4.3.1 Grid Search

As shown in Fig.16 the use of Grid Search helps set the hyperparameters utilised to optimise prediction performance, and computational processing speed e.g. max_features = 8 and n_estimators = 30 for Grid search on Random Forests.



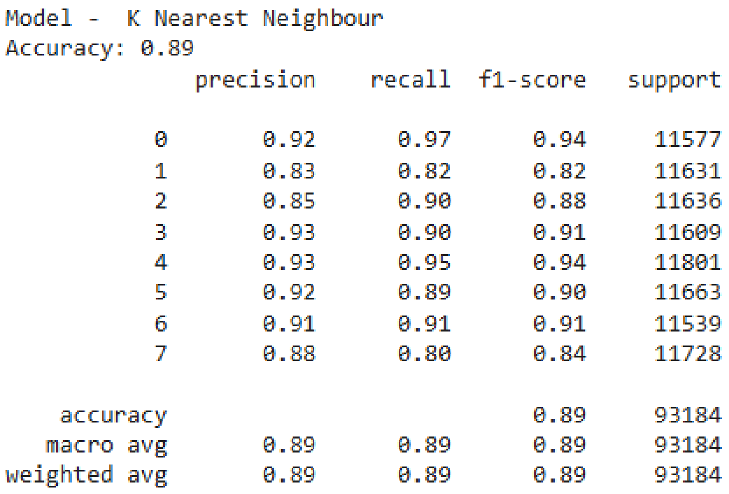
```
Accuracy: 1.00
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      1407
           1       1.00      1.00      1.00      9084
           2       1.00      1.00      1.00      2671
           3       1.00      1.00      1.00      8225
           4       1.00      1.00      1.00      1804
           5       1.00      1.00      1.00      3855
           6       1.00      1.00      1.00     11628
           7       1.00      1.00      1.00      6054

    accuracy                           1.00     44728
   macro avg       1.00      1.00      1.00     44728
weighted avg       1.00      1.00      1.00     44728

Train score  1.0
Test score 0.9987032731175103
========
```

Fig. 30 (Grid Search Random Forest Result)

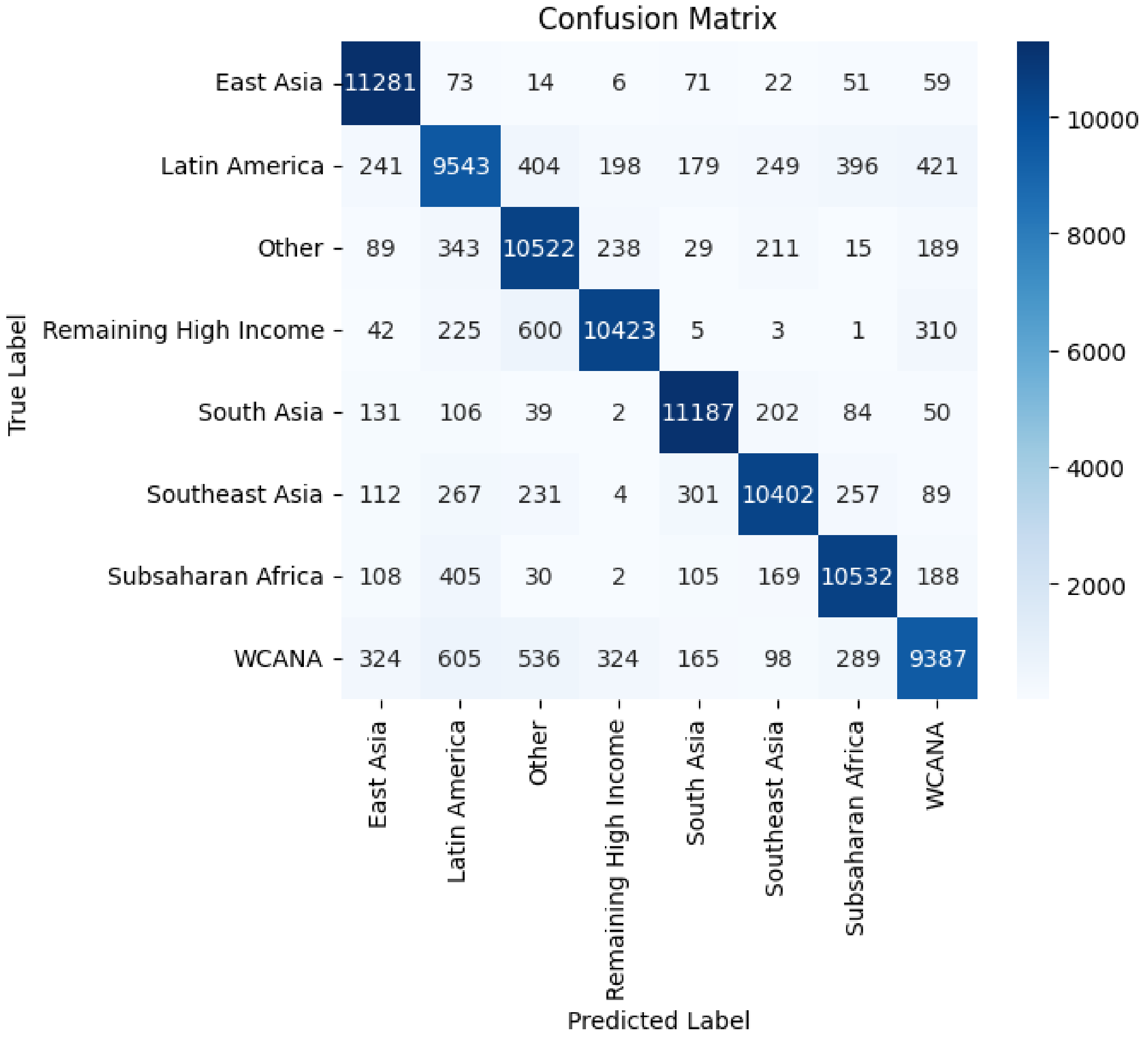
Random forests using the grid search optimised parameters leads to an extremely high prediction accuracy. This is extenuated by the now scaled yet unbalanced data. Additionally, the data leakage of the most important features is another cause of the extremely high accuracy. Evidently performing better than the previous Random Forests application.

4.3.2 Predictions Using Over Sampling (SMOTE)

Formulating data for features with a lower count to balance dataset output values.

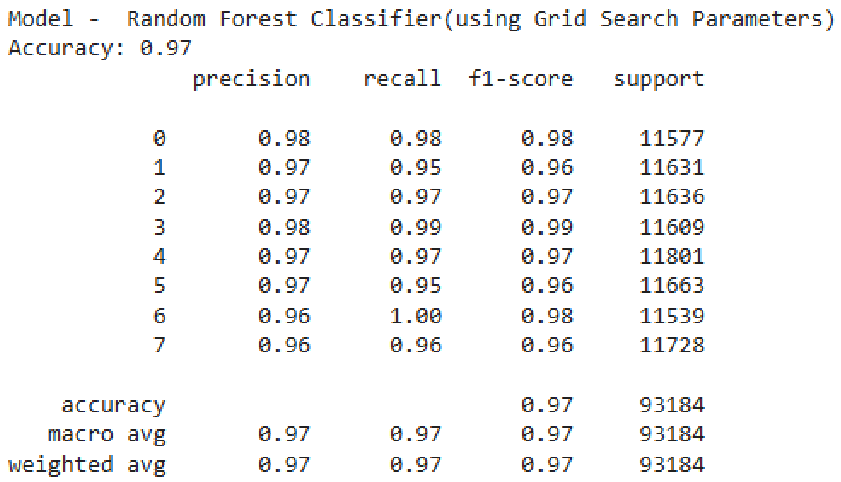


```
Model -  K Nearest Neighbour
Accuracy: 0.89
              precision    recall  f1-score   support

           0       0.92      0.97      0.94     11577
           1       0.83      0.82      0.82     11631
           2       0.85      0.90      0.88     11636
           3       0.93      0.90      0.91     11609
           4       0.93      0.95      0.94     11801
           5       0.92      0.89      0.90     11663
           6       0.91      0.91      0.91     11539
           7       0.88      0.80      0.84     11728

    accuracy                           0.89     93184
   macro avg       0.89      0.89      0.89     93184
weighted avg       0.89      0.89      0.89     93184
```

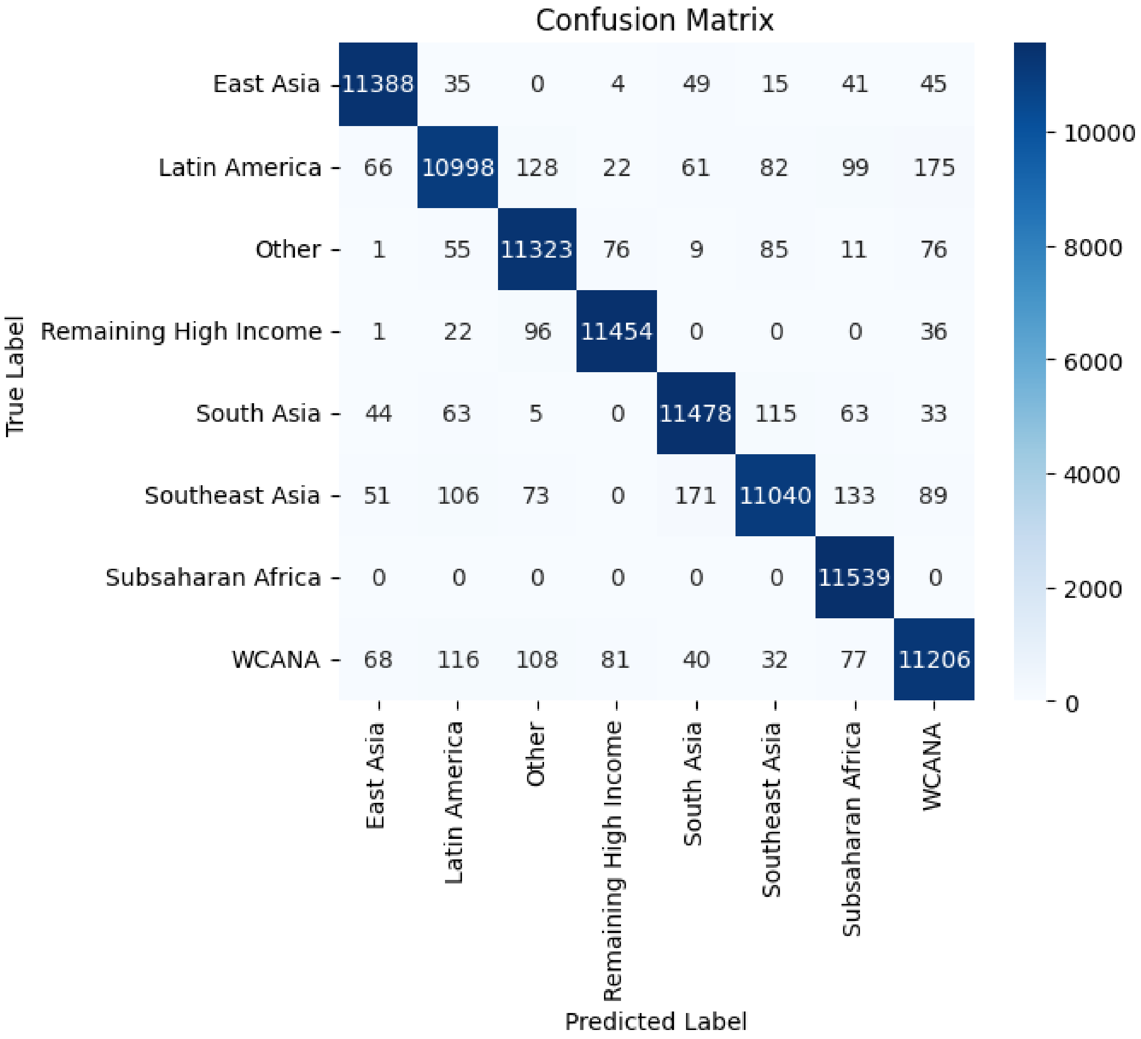


	East Asia	Latin America	Other	Remaining High Income	South Asia	Southeast Asia	Subsaharan Africa	WCANA
East Asia	11281	73	14	6	71	22	51	59
Latin America	241	9543	404	198	179	249	396	421
Other	89	343	10522	238	29	211	15	189
Remaining High Income	42	225	600	10423	5	3	1	310
South Asia	131	106	39	2	11187	202	84	50
Southeast Asia	112	267	231	4	301	10402	257	89
Subsaharan Africa	108	405	30	2	105	169	10532	188
WCANA	324	605	536	324	165	98	289	9387

Fig. 31



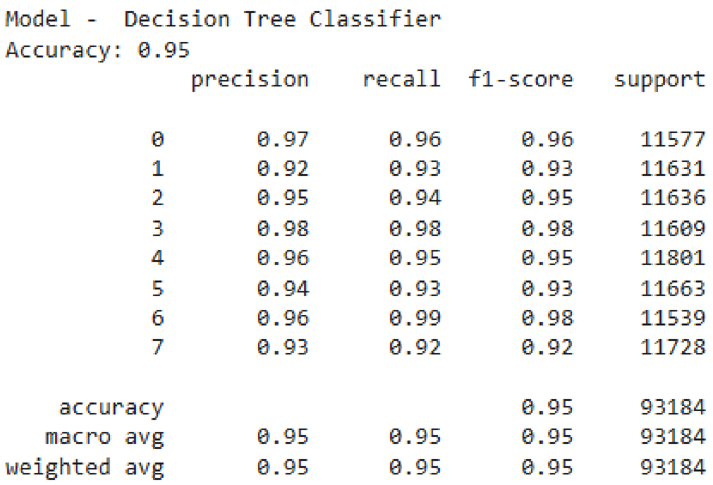
```
Model -  Random Forest Classifier(using Grid Search Parameters)
Accuracy: 0.97
              precision    recall  f1-score   support

           0       0.98      0.98      0.98     11577
           1       0.97      0.95      0.96     11631
           2       0.97      0.97      0.97     11636
           3       0.98      0.99      0.99     11609
           4       0.97      0.97      0.97     11801
           5       0.97      0.95      0.96     11663
           6       0.96      1.00      0.98     11539
           7       0.96      0.96      0.96     11728

    accuracy                           0.97     93184
   macro avg       0.97      0.97      0.97     93184
weighted avg       0.97      0.97      0.97     93184
```



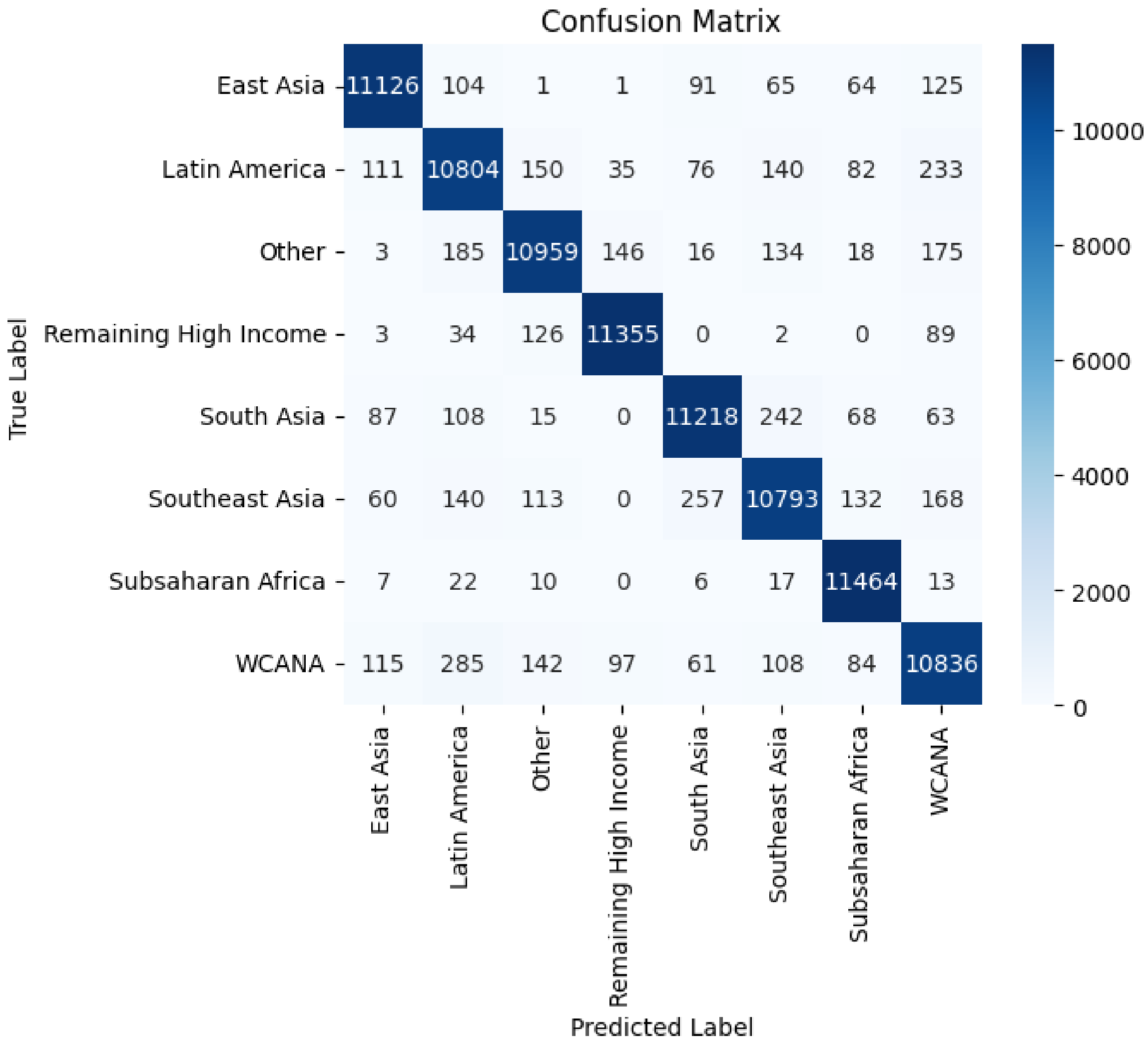
Confusion Matrix

True Label \ Predicted Label	East Asia	Latin America	Other	Remaining High Income	South Asia	Southeast Asia	Subsaharan Africa	WCANA
East Asia	11388	35	0	4	49	15	41	45
Latin America	66	10998	128	22	61	82	99	175
Other	1	55	11323	76	9	85	11	76
Remaining High Income	1	22	96	11454	0	0	0	36
South Asia	44	63	5	0	11478	115	63	33
Southeast Asia	51	106	73	0	171	11040	133	89
Subsaharan Africa	0	0	0	0	0	0	11539	0
WCANA	68	116	108	81	40	32	77	11206

Fig. 32



```
Model -  Decision Tree Classifier
Accuracy: 0.95
              precision    recall  f1-score   support

           0       0.97      0.96      0.96     11577
           1       0.92      0.93      0.93     11631
           2       0.95      0.94      0.95     11636
           3       0.98      0.98      0.98     11609
           4       0.96      0.95      0.95     11801
           5       0.94      0.93      0.93     11663
           6       0.96      0.99      0.98     11539
           7       0.93      0.92      0.92     11728

    accuracy                           0.95     93184
   macro avg       0.95      0.95      0.95     93184
weighted avg       0.95      0.95      0.95     93184
```



Confusion Matrix

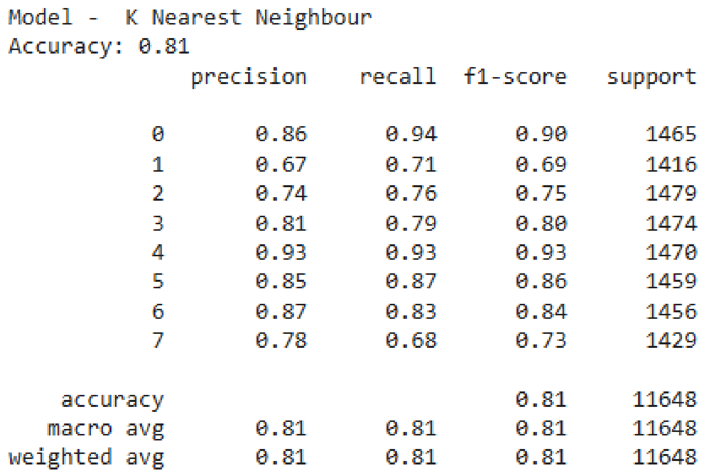
True Label	East Asia	Latin America	Other	Remaining High Income	South Asia	Southeast Asia	Subsaharan Africa	WCANA
East Asia	11126	104	1	1	91	65	64	125
Latin America	111	10804	150	35	76	140	82	233
Other	3	185	10959	146	16	134	18	175
Remaining High Income	3	34	126	11355	0	2	0	89
South Asia	87	108	15	0	11218	242	68	63
Southeast Asia	60	140	113	0	257	10793	132	168
Subsaharan Africa	7	22	10	0	6	17	11464	13
WCANA	115	285	142	97	61	108	84	10836

Fig. 33

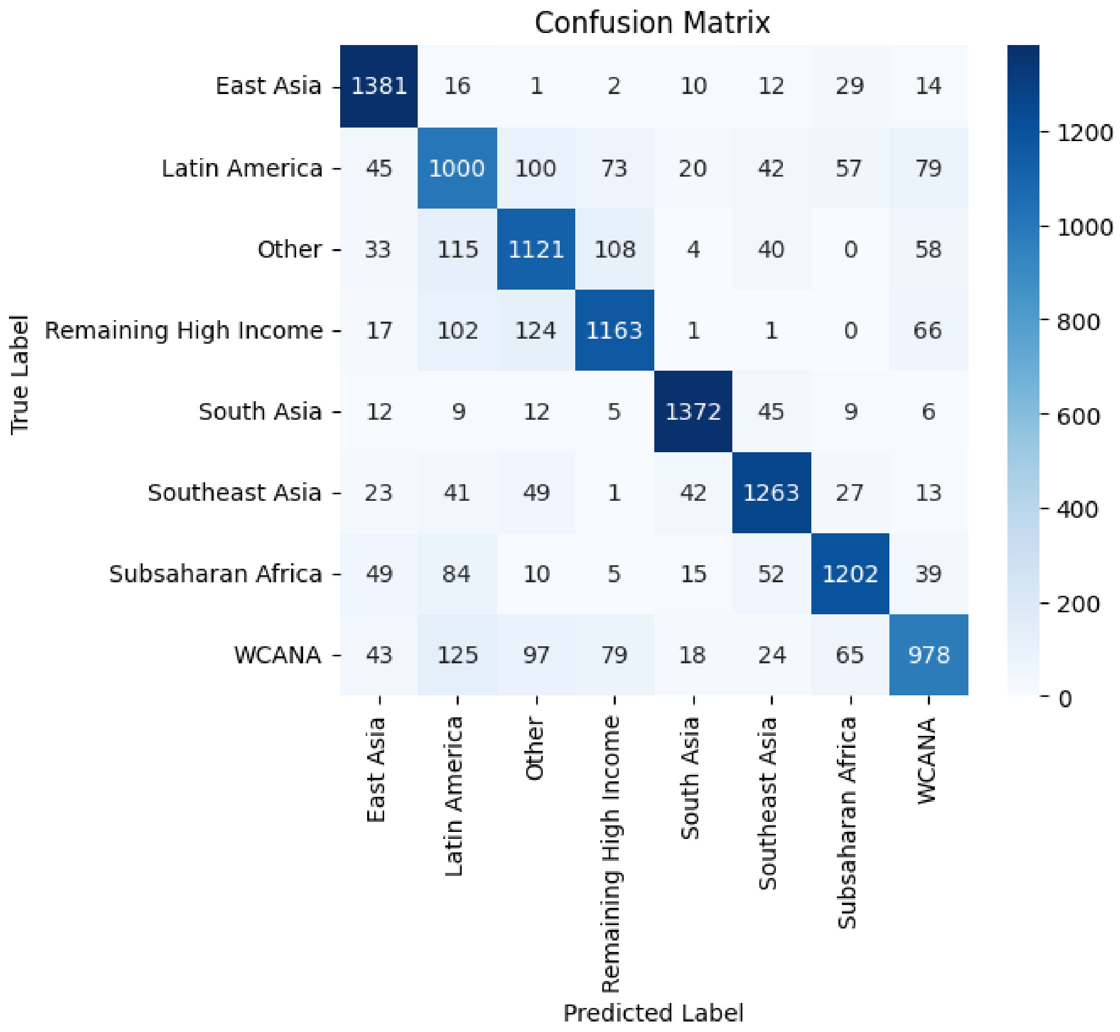
Despite synthetic data, K nearest neighbour (KNN) is still able to predict with high f1-score and accuracy and should be tested on test data without synthesised data. The Random Forest Classifier with and without grid-search perform similarly well for differing computational load. Decision Tree classifier performs expectedly worse than Random Forests yet still outperforms KNN.

4.3.3 Predictions Using Under Sampling (Near Miss)

It works by selecting samples closest to the minority class and deletes records so all categories have the same number of samples.

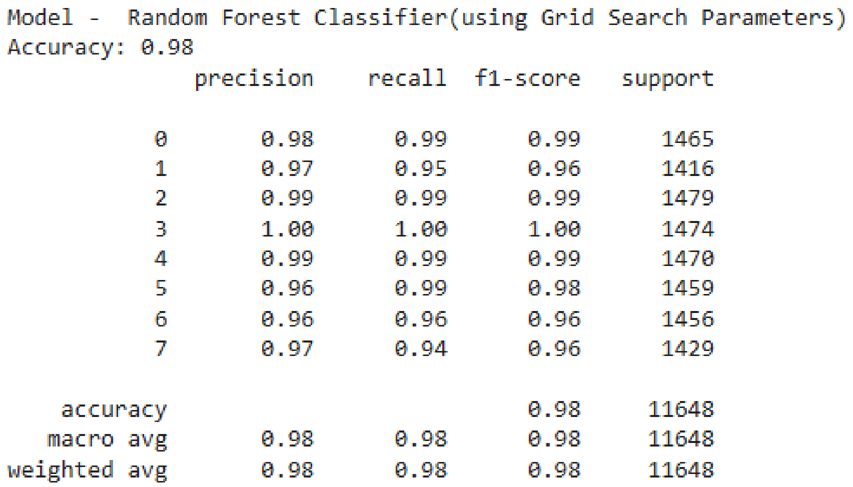


```
Model -  K Nearest Neighbour
Accuracy: 0.81
              precision    recall  f1-score   support

           0       0.86      0.94      0.90      1465
           1       0.67      0.71      0.69      1416
           2       0.74      0.76      0.75      1479
           3       0.81      0.79      0.80      1474
           4       0.93      0.93      0.93      1470
           5       0.85      0.87      0.86      1459
           6       0.87      0.83      0.84      1456
           7       0.78      0.68      0.73      1429

    accuracy                           0.81     11648
   macro avg       0.81      0.81      0.81     11648
weighted avg       0.81      0.81      0.81     11648
```

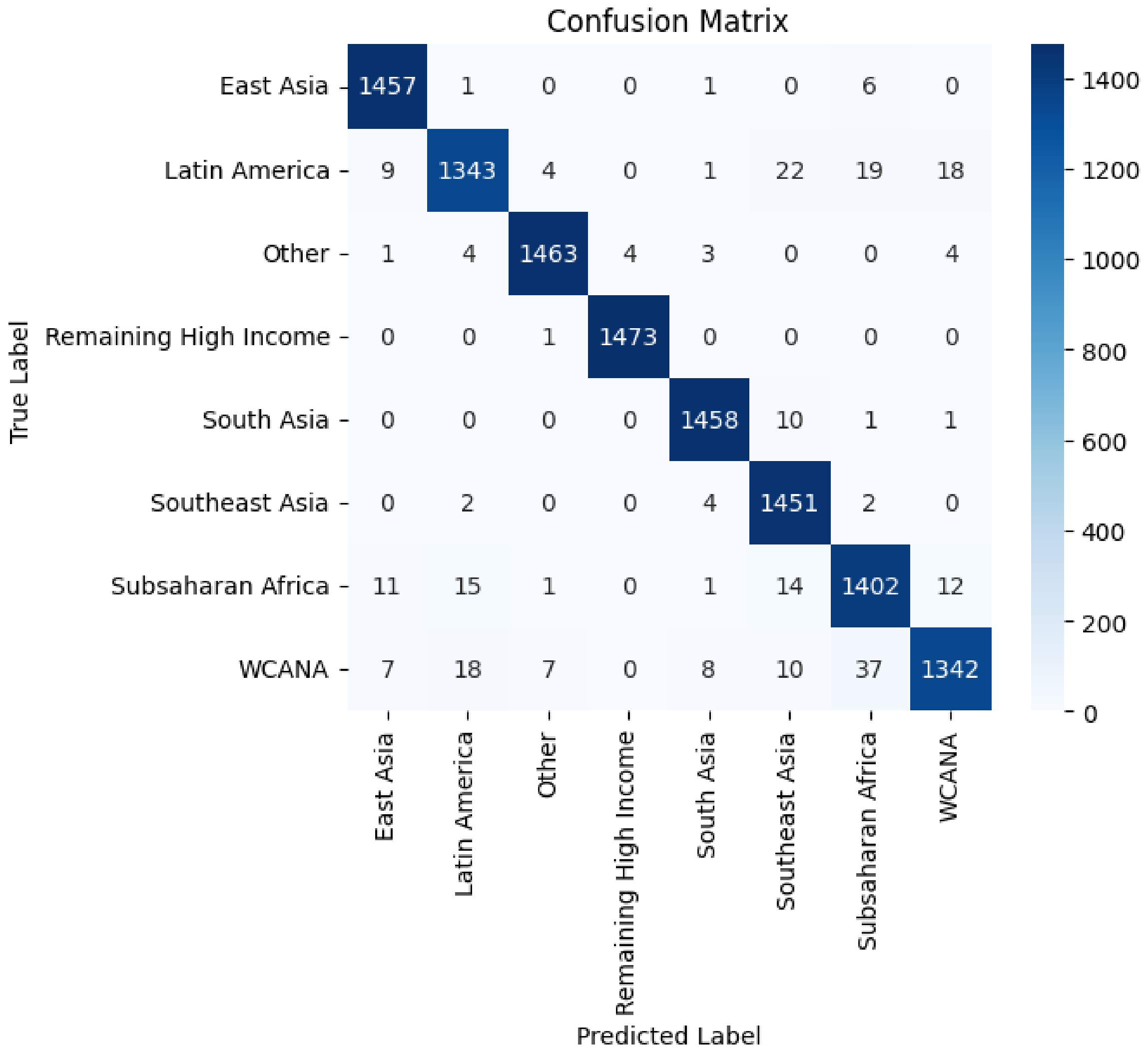


	East Asia	Latin America	Other	Remaining High Income	South Asia	Southeast Asia	Subsaharan Africa	WCANA
East Asia	1381	16	1	2	10	12	29	14
Latin America	45	1000	100	73	20	42	57	79
Other	33	115	1121	108	4	40	0	58
Remaining High Income	17	102	124	1163	1	1	0	66
South Asia	12	9	12	5	1372	45	9	6
Southeast Asia	23	41	49	1	42	1263	27	13
Subsaharan Africa	49	84	10	5	15	52	1202	39
WCANA	43	125	97	79	18	24	65	978

Fig. 34



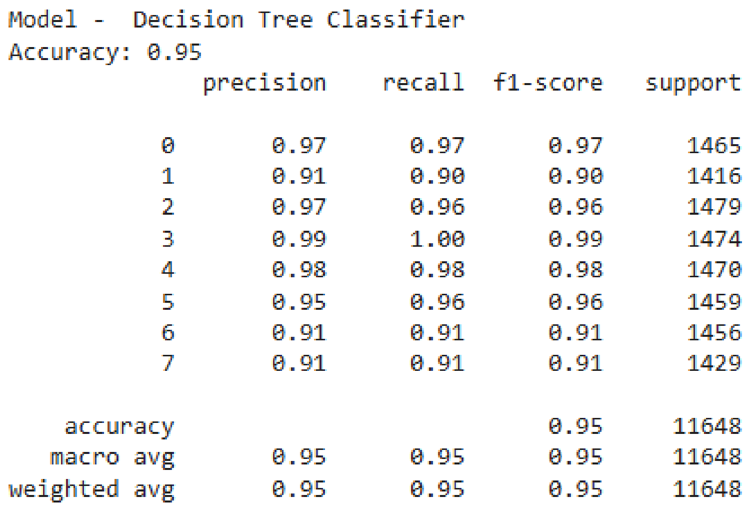
```
Model -  Random Forest Classifier(using Grid Search Parameters)
Accuracy: 0.98
              precision    recall  f1-score   support

           0       0.98      0.99      0.99      1465
           1       0.97      0.95      0.96      1416
           2       0.99      0.99      0.99      1479
           3       1.00      1.00      1.00      1474
           4       0.99      0.99      0.99      1470
           5       0.96      0.99      0.98      1459
           6       0.96      0.96      0.96      1456
           7       0.97      0.94      0.96      1429

    accuracy                           0.98     11648
   macro avg       0.98      0.98      0.98     11648
weighted avg       0.98      0.98      0.98     11648
```



Confusion Matrix

True Label \ Predicted Label	East Asia	Latin America	Other	Remaining High Income	South Asia	Southeast Asia	Subsaharan Africa	WCANA
East Asia	1457	1	0	0	1	0	6	0
Latin America	9	1343	4	0	1	22	19	18
Other	1	4	1463	4	3	0	0	4
Remaining High Income	0	0	1	1473	0	0	0	0
South Asia	0	0	0	0	1458	10	1	1
Southeast Asia	0	2	0	0	4	1451	2	0
Subsaharan Africa	11	15	1	0	1	14	1402	12
WCANA	7	18	7	0	8	10	37	1342

Fig. 35



```
Model -  Decision Tree Classifier
Accuracy: 0.95
              precision    recall  f1-score   support

           0       0.97      0.97      0.97      1465
           1       0.91      0.90      0.90      1416
           2       0.97      0.96      0.96      1479
           3       0.99      1.00      0.99      1474
           4       0.98      0.98      0.98      1470
           5       0.95      0.96      0.96      1459
           6       0.91      0.91      0.91      1456
           7       0.91      0.91      0.91      1429

    accuracy                           0.95     11648
   macro avg       0.95      0.95      0.95     11648
weighted avg       0.95      0.95      0.95     11648
```

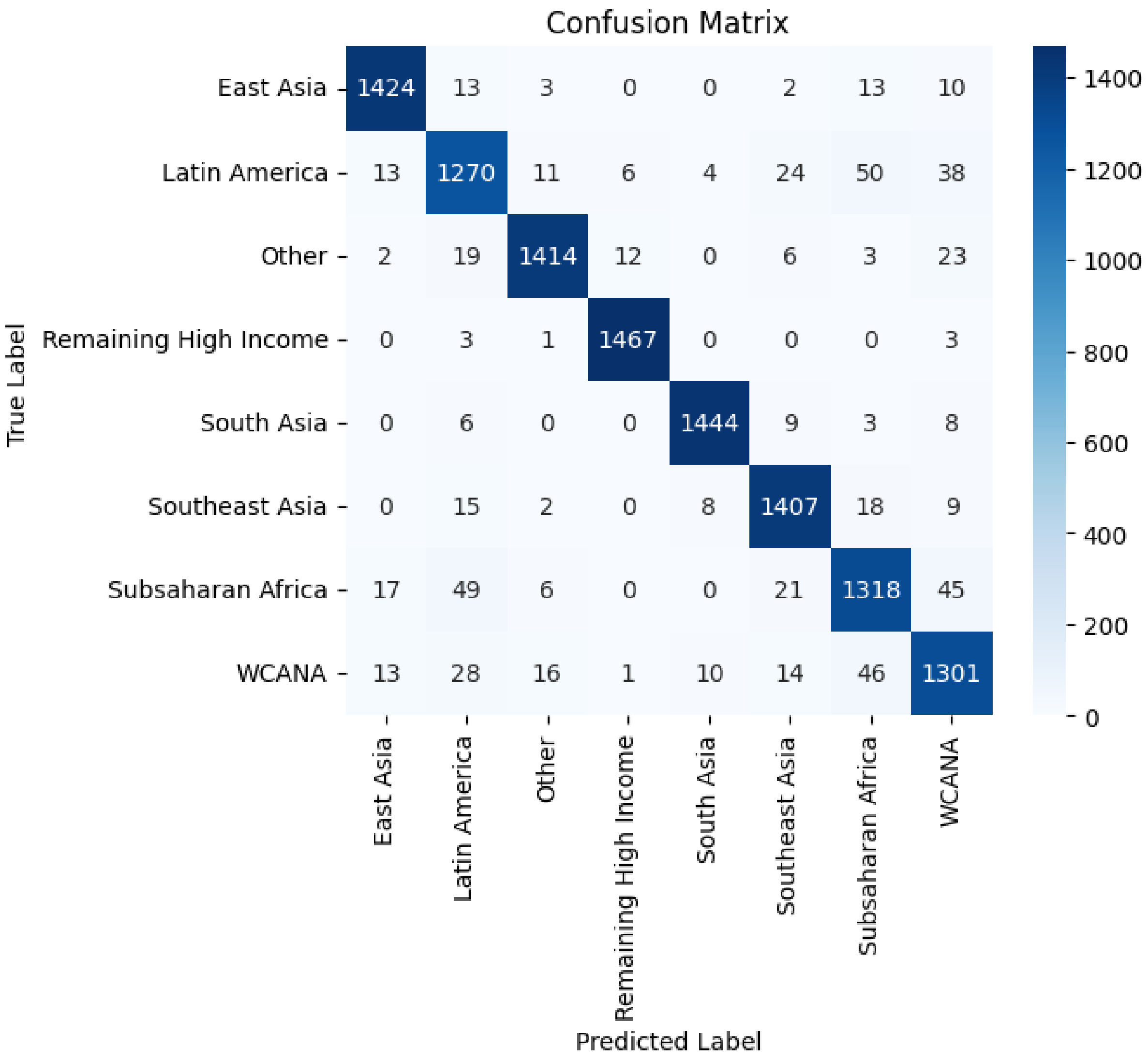
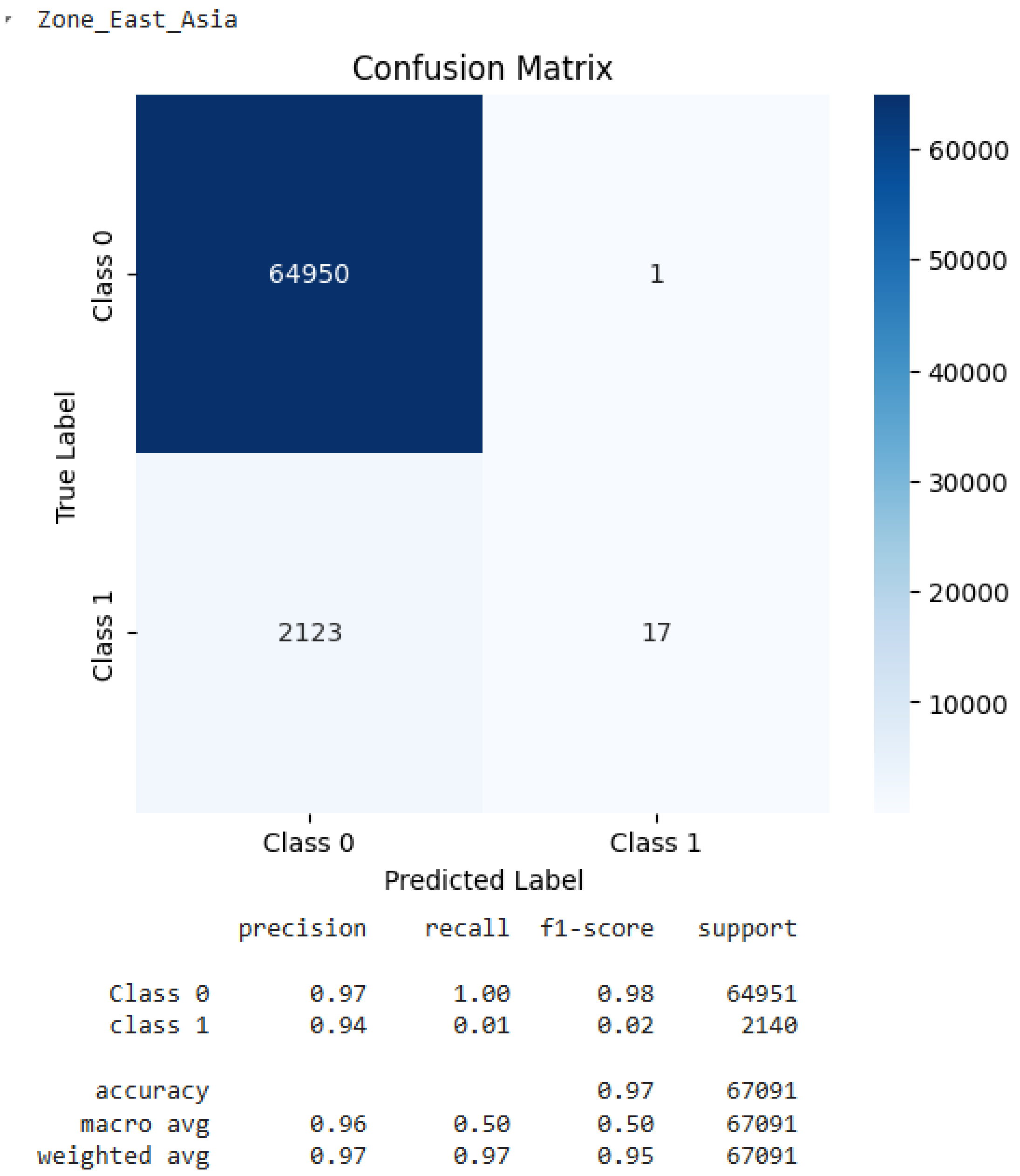


Fig. 36

KNN performs less accurately than when oversampled, Random Forest with Grid Search params performs the best of all algorithms with minimal spread on inaccuracy.

4.3.4 Deep Learning

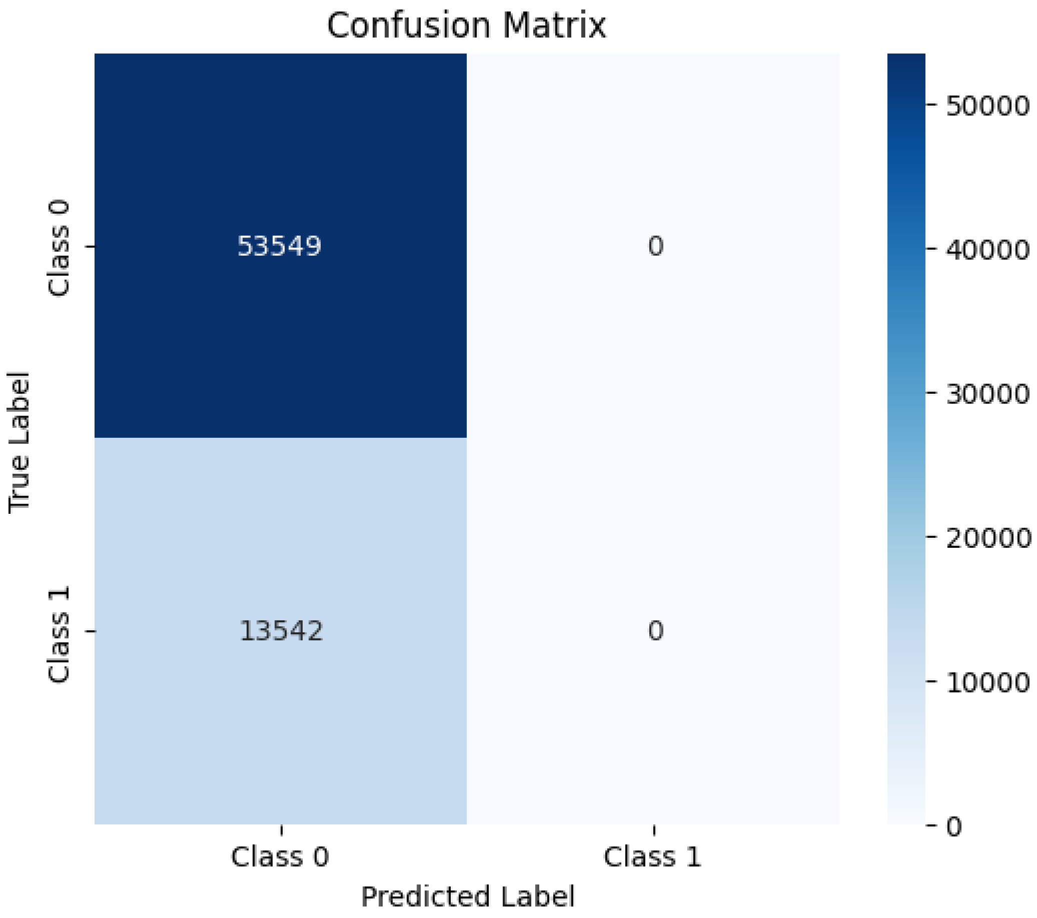


```
▸ Zone_East_Asia
```

```
              precision    recall  f1-score   support

     Class 0       0.97      1.00      0.98     64951
     class 1       0.94      0.01      0.02      2140

    accuracy                           0.97     67091
   macro avg       0.96      0.50      0.50     67091
weighted avg       0.97      0.97      0.95     67091
```

Fig. 37

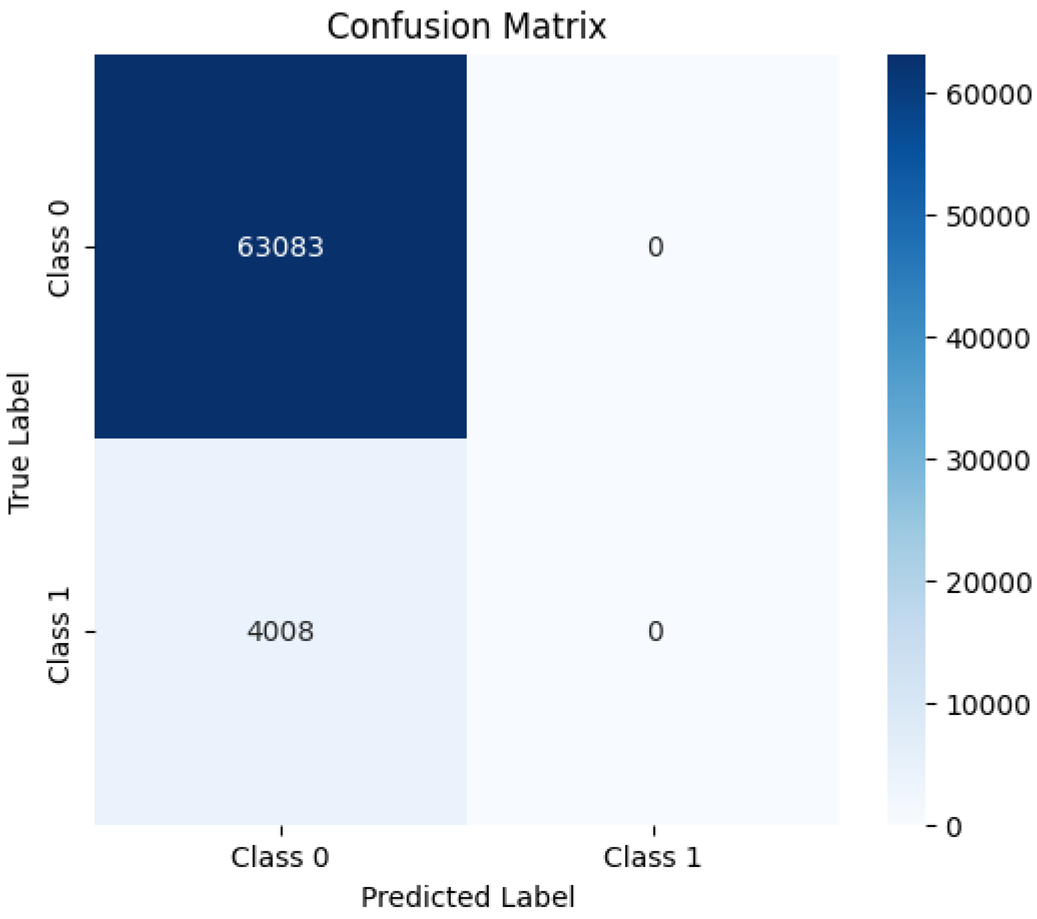
```
Zone_Latin_America
```



```
              precision    recall  f1-score   support

     Class 0       0.80      1.00      0.89     53549
     class 1       0.00      0.00      0.00     13542

    accuracy                           0.80     67091
   macro avg       0.40      0.50      0.44     67091
weighted avg       0.64      0.80      0.71     67091
```

Fig. 38

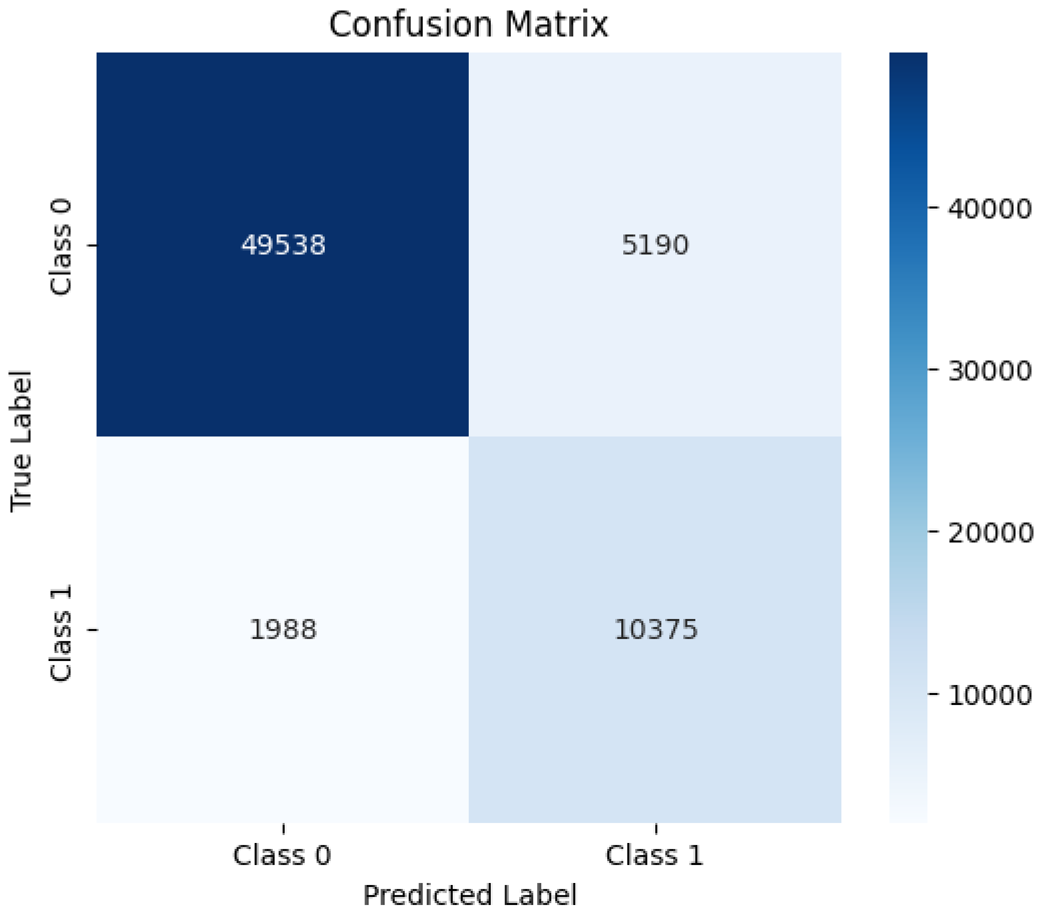
```
Zone_Other
```



```
              precision    recall  f1-score   support

     Class 0       0.94      1.00      0.97     63083
     class 1       0.00      0.00      0.00      4008

    accuracy                           0.94     67091
   macro avg       0.47      0.50      0.48     67091
weighted avg       0.88      0.94      0.91     67091
```

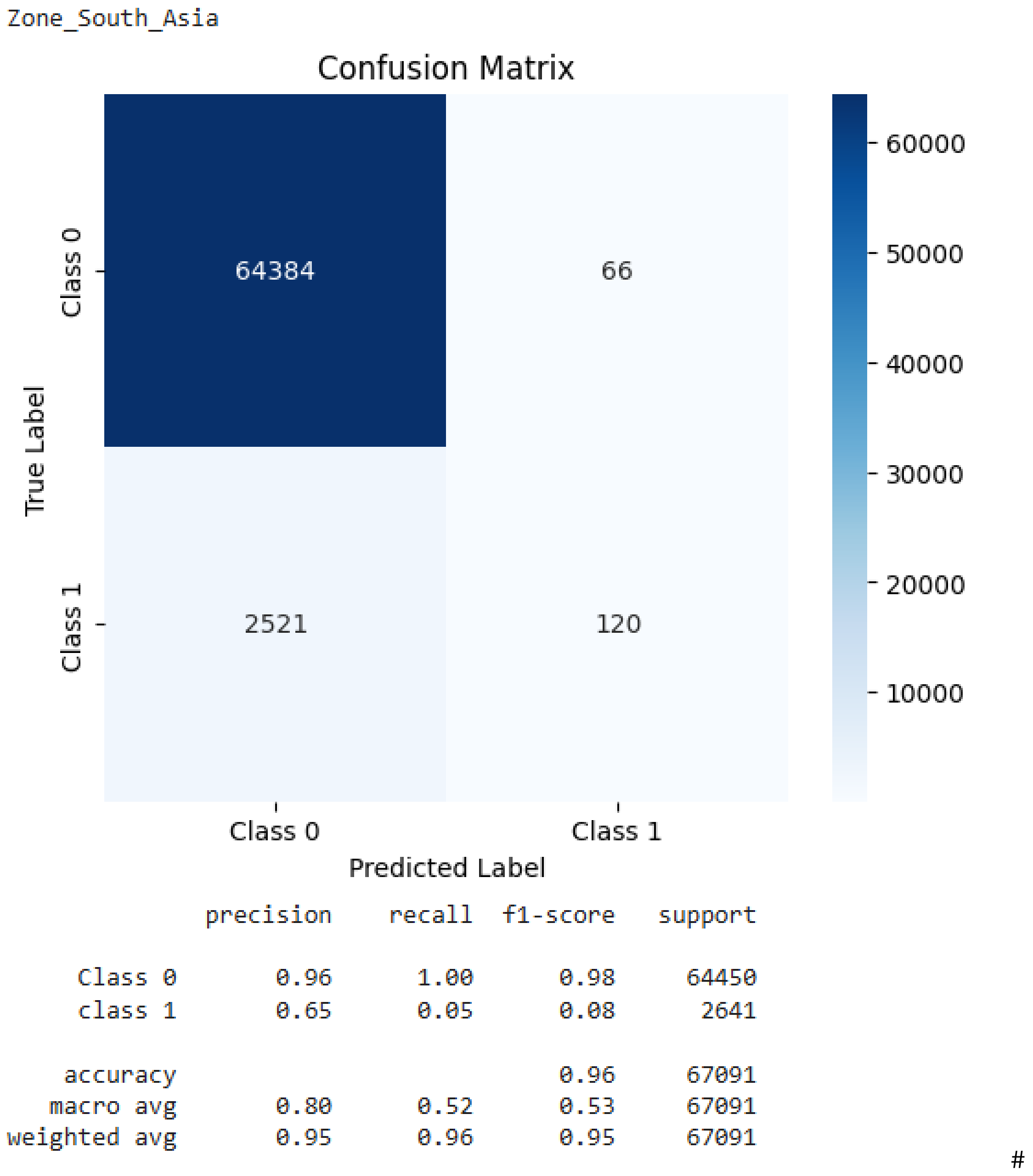
Fig. 39

```
Remaining_High_Income
```



```
              precision    recall  f1-score   support

     Class 0       0.96      0.91      0.93     54728
     class 1       0.67      0.84      0.74     12363

    accuracy                           0.89     67091
   macro avg       0.81      0.87      0.84     67091
weighted avg       0.91      0.89      0.90     67091
```

Fig. 40

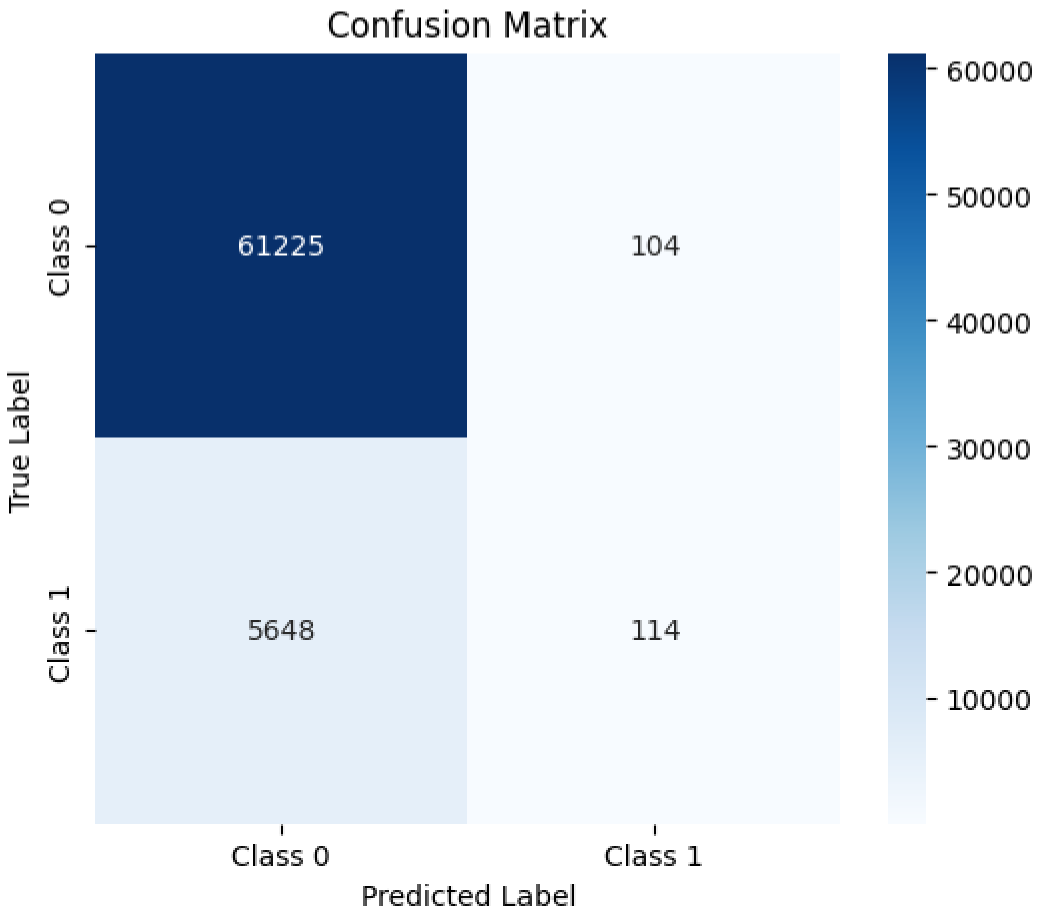


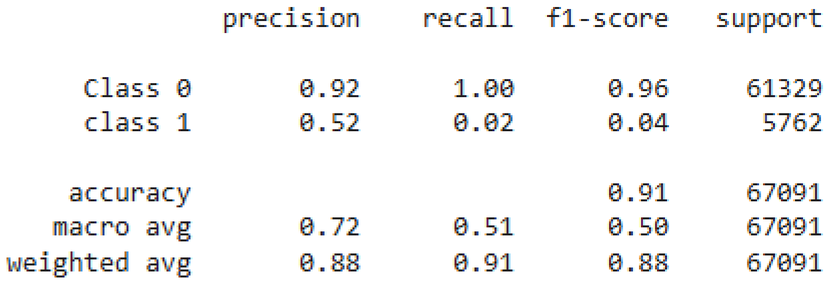
```
Zone_South_Asia
```

```
              precision    recall  f1-score   support

     Class 0       0.96      1.00      0.98     64450
     class 1       0.65      0.05      0.08      2641

    accuracy                           0.96     67091
   macro avg       0.80      0.52      0.53     67091
weighted avg       0.95      0.96      0.95     67091
                                                                  #
```

Fig. 41

```
Zone_Southeast_Asia
```

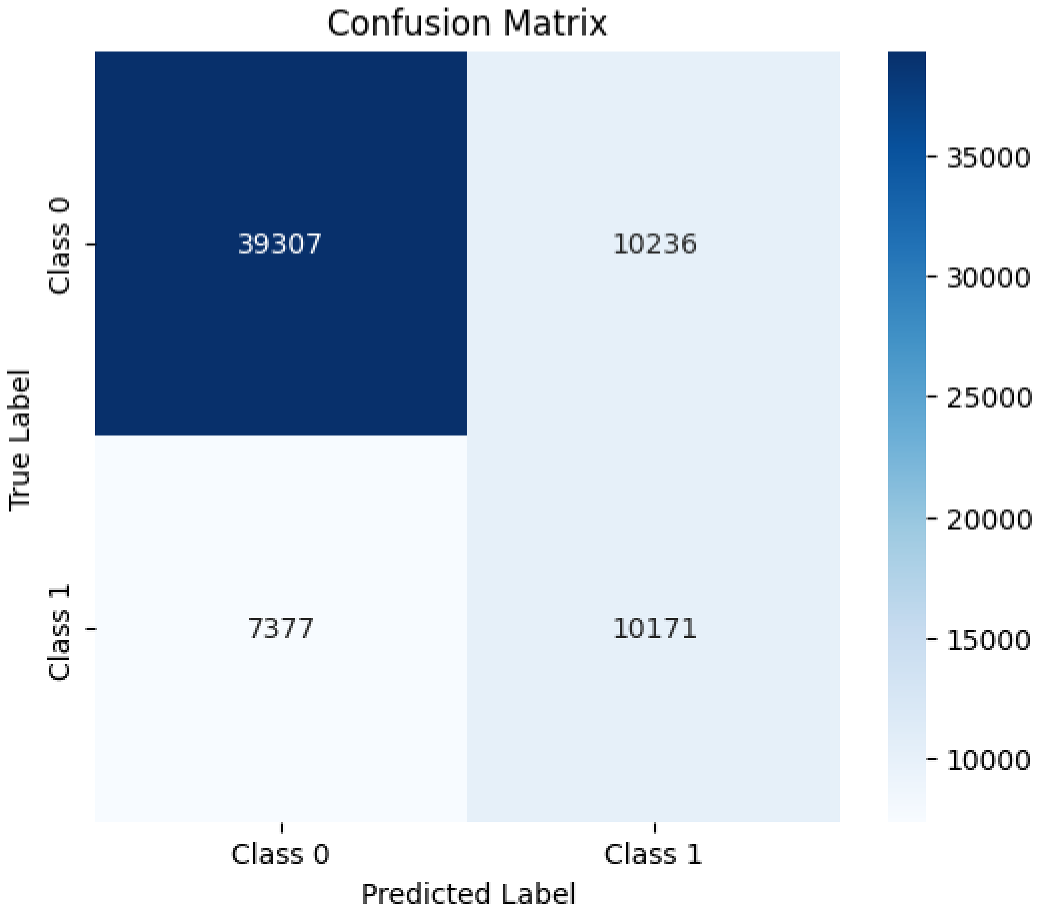




```
              precision    recall  f1-score   support

     Class 0       0.92      1.00      0.96     61329
     class 1       0.52      0.02      0.04      5762

    accuracy                           0.91     67091
   macro avg       0.72      0.51      0.50     67091
weighted avg       0.88      0.91      0.88     67091
```

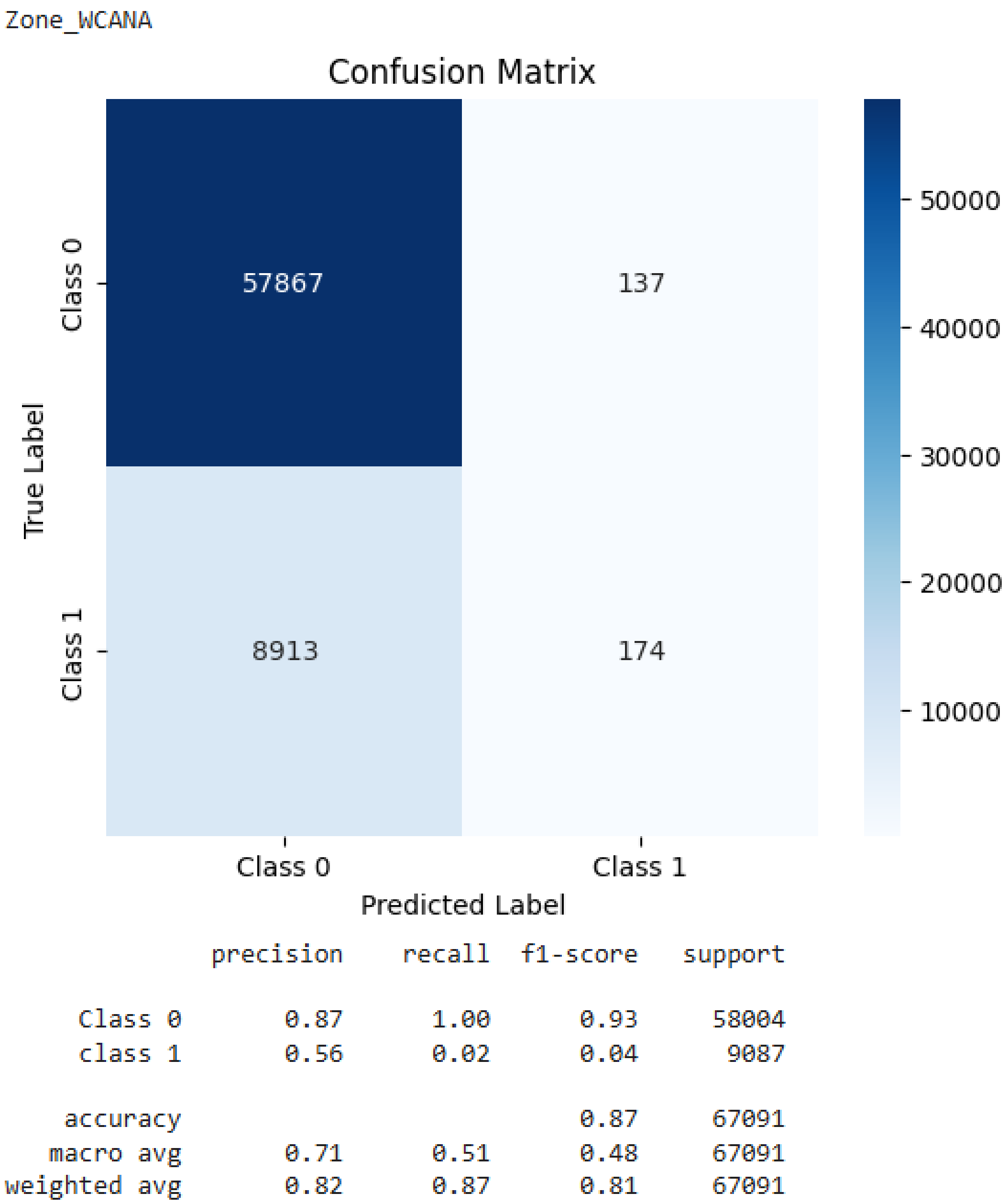
Fig.42

```
Zone_Subsaharan_Africa
```



```
              precision    recall  f1-score   support

     Class 0       0.84      0.79      0.82     49543
     class 1       0.50      0.58      0.54     17548

    accuracy                           0.74     67091
   macro avg       0.67      0.69      0.68     67091
weighted avg       0.75      0.74      0.74     67091
```

Fig. 43



```
Zone_WCANA
```

```
              precision    recall  f1-score   support

     Class 0       0.87      1.00      0.93     58004
     class 1       0.56      0.02      0.04      9087

    accuracy                           0.87     67091
   macro avg       0.71      0.51      0.48     67091
weighted avg       0.82      0.87      0.81     67091
```

Fig. 44

Initially y was converted from “nd.array” format back to a data-frame type (refer to Fig. 25) and compared the prediction for each category individually and independent of each other. Overall, the result for each category is predicted to a satisfactory f1-score and accuracy. Many False Negative predictions can be seen in Latin America and WCANA predictions. Many false positives in Southeast Asia and Remaining High income.

The Class1 true and predicted id data correctly classifying the item not being in the column "0" for this category (already classified in another category).

Train test split for Over sampling and near miss struggles to deal with one hot encoded train test split.

5 Evaluation and Further Modelling Improvements

Discussion Topic Questions	Answer
Which models performed the best? Were there big differences?	Models involving decision tree, majority voting or layers performed better for this dataset than other algorithms that do not use any one of the listed features did. As seen by some models predicting with <50% accuracy.
What were uniquely prevalent issues with the dataset?	Due to being a classification model with multiple categories for outputs, there were issues with classification reports for deep learning. It should also be noted huge imbalance in output data meant predictions is susceptible to skew and bias, meaning the model is affected by data leakage.
Do you think it's more important to be sensitive to negativity or positivity? Do we want more positive things incorrectly marked as negative, or more negative things marked as positive?	Whether an error is positively or negatively predicted is irrelevant for this dataset, as it is not a binary classification, the error could be one of 7 other categories. In the case of deep learning, the focus is on the accuracy, precision, recall and f1-score of the prediction for each correctly predicted item.
They all had very different training times. Which ones offer the best combination of performance and not making you wait around for an hour?	Random Forest Classifier had the best combination of prediction performance and training time. Performing well in all scenarios and under different sampling methods.
Is 75% accuracy good?	In this case, accuracy is a viable metric of consideration as a multiple category output. The aim is for each item to be precisely and uniquely identifiable. So, 75% across all categories is "good". However, in the last visualisation of deep learning, when comparing one category at a time, accuracy is a less prevalent metric of consideration and does not properly highlight the nuance of the confusion matrix that f1-score does.

- Further modelling improvements could be applying hyperparameter tuning to all machine algorithms in focus for the report.
- Finding a method to apply SMOTE and Near Miss without errors to Deep Learning to see how the algorithm dals with synthetic data.
- Applying the one-hot encoding to all algorithms and comparing the results to the label transformed Zone feature which means it is easier to identify if the algorithm is predicting multiple regions for one record.

- Utilising PCA more to help fix the overfitting and overly accurate predictions on training data and test data.
- Making the models more generalised by introducing data from 2012-2024.

6 Conclusion

6.1 Summary of Results

In summary, with such high accuracy values in the initial data (baseline testing seen in colab) demonstrates the model is likely not generalised and suffers from data leakage, likely sourced from one of the higher importance features in the importance table in Fig. 14. However, the performance in cases where the values are scaled, under or over sampled, perform statistically worse, but are better generalised to the task and are less likely to be overfitting as severely. Deep Learning with the one-hot encoded Zone categories gives great one to one data analysis, but an overall confusion matrix cannot be produced and there is a possibility the algorithm is predicting the record fitting into multiple categories unlike the other algorithms predicting 0-7 in Zone. Due to data leakage in the baseline, it is difficult to comment on the reliability and effectiveness of each model and their f1-scores. With further testing and generalisation, the application of classification prediction of global region based on micronutrient values is possible.

6.2 Reflection on Individual Learning

This module has sparked an excitement in my development as a computer scientist and will likely be a prominent feature driving my progression and passion for the field. It has been an eye-opening course brilliantly delivered, going through the trial by fire, which is the development of the code, encountering errors in all sorts of parts of the project has been fantastic, empowering me to no end. I feel I understand the machine learning rabbit-hole better, but I understand that this is merely the surface. From watching YouTube videos about reinforcement learning on a visual model, to having learnt its fundamentals in lectures has been surreal. I am glad the choice was made to apply this project on real-world and a dataset I sourced myself. I will most definitely look forward to undertaking more machine learning projects in future, big data, autonomation, etc.

7 References

[1] Beal, T., Massiot, E., Arsenault, J.E., Smith, M.R. and Hijmans, R.J. (2017). Global trends in dietary micronutrient supplies and estimated prevalence of inadequate intakes. *PLoS ONE*, [online] 12(4), pp.e0175554–e0175554. doi:https://doi.org/10.1371/journal.pone.0175554.

[2] Schmidhuber, J., Sur, P., Fay, K., Huntley, B., Salama, J., Lee, A., Cornaby, L., Horino, M., Murray, C. and Ashkan Afshin (2018). The Global Nutrient Database: availability of macronutrients and micronutrients in 195 countries from 1980 to 2013. *The Lancet Planetary Health*, [online] 2(8), pp.e353–e368. doi:https://doi.org/10.1016/s2542-5196(18)30170-0.

[3] Mannar, M.G.V. and Sankar, R. (2004). Micronutrient fortification of foods — rationale, application and impact. *The Indian Journal of Pediatrics*, [online] 71(11), pp.997–1002. doi:https://doi.org/10.1007/bf02828115.

[4] Tim Mucci (2024). *Data leakage machine learning*. [online] Ibm.com. Available at: https://www.ibm.com/think/topics/data-leakage-machine-learning [Accessed 11 Dec. 2024].

[5] Google Cloud. (2024). *Hyperparameter tuning overview*. [online] Available at: https://cloud.google.com/bigquery/docs/hp-tuning-overview#:~:text=In%20machine%20learning%2C%20hyperparameter%20tuning,a%20linear%20model%20are%20learned. [Accessed 11 Dec. 2024].

[6] Furnieles, G. (2022). *Sigmoid and SoftMax Functions in 5 minutes - Towards Data Science*. [online] Medium. Available at: https://towardsdatascience.com/sigmoid-and-softmax-functions-in-5-minutes-f516c80ea1f9 [Accessed 11 Dec. 2024].

[7] www.javatpoint.com. (2021). *Decision Tree Algorithm in Machine Learning - Javatpoint*. [online] Available at: https://www.javatpoint.com/machine-learning-decision-tree-classification-algorithm [Accessed 11 Dec. 2024].

[8] IBM (2021). *KNN*. [online] Ibm.com. Available at: https://www.ibm.com/topics/knn#:~:text=The%20k%2Dnearest%20neighbors%20(KNN)%20algorithm%20is%20a%20non,used%20in%20machine%20learning%20today. [Accessed 11 Dec. 2024].

[9] Google Cloud. (2024). *Hyperparameter tuning overview*. [online] Available at: https://cloud.google.com/bigquery/docs/hp-tuning-overview#:~:text=In%20machine%20learning%2C%20hyperparameter%20tuning,a%20linear%20model%20are%20learned. [Accessed 12 Dec. 2024].

[10] Punyakeerthi BL (2024). *Understanding Feature Scaling in Machine Learning - Punyakeerthi BL - Medium*. [online] Medium. Available at: https://medium.com/@punya8147_26846/understanding-feature-scaling-in-machine-learning-fe2ea8933b66 [Accessed 12 Dec. 2024].