National College of Ireland

Configuration Manual

MSc Research Project
MSc Data Analytics

Niharika Subbarayappa
Student ID: X23102420

School of Computing
National College of Ireland

Supervisor: Professor. John Kelly

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name: Niharika Subbarayappa

Student ID: X23102420

Programme: MSc Data Analytics **Year:** 2023 – 2024

Module: MSc Research Project

Lecturer: Professor. John Kelly

Submission Due Date: 12th August 2024

Project Title: Configuration Manual: Predicting Maternal Mortality Risk: Sub Urban and Rural India

Word Count: 713 **Page Count: 8**

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Niharika Subbarayappa

Date: 12th August 2024

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Niharika Subbarayappa
Student ID: X23102420

1 Environment

The development, implementation and execution of the Maternal Health Risk Prediction models were conducted in the python-based environment. The environment setup includes the following components: an operating system with Microsoft Windows 1, Internal RAM of 8 GB, Hard Disk with 512 bytes, processor of 11th Gen Intel®, core™ i3-1115G4 @3.00Ghz, 2995 MHz, 2 core(s), 4 Logical Processors and GPU of Intel ® UHD Graphics.

2 Libraries and Tools

2.1 Tools

This section gives us the information about Tools used for development process and having them installed is a prerequisite to continue with this work.
Tools include: Anaconda Navigator with version 2.4.0, Jupyter notebook installed version 6.4.12 and Python with version of 3 with 64 bits.

2.2 Python Libraries

This section provides the information on the libraries imported and used for the model development and implementation purpose.

Libraries Used	Purpose
pandas	Provides data functions to manipulate and analyse data
numpy	Support for arrays with collection of mathematical functions
sklearn. model selection	Validations, hyperparameter tuning and other evaluations
sklearn. preprocessing	Functions for data preprocessing, encoding categorical values
sklearn. ensemble	Implements ensemble methods and combining predictions
xgboost	Building efficient ML model
tensorflow. keras.models	Building and managing neural network models
tensorflow.keras.layers	Provides variety of building blocks to build neural network
tensorflow.keras.utils	Utility functions to encode, plotting and managing data sets.
sklearn.metrics	Evaluation metrics such accuracy, precision, recall and more
matplotlib.pyplot	Providing Visualizations in Python
seaborn	Creating attractive and informative statistical graphics
itertools	Provides functions for efficient looping
keras_tuner	Hyperparameter tuning for neural networks
tensorflow.keras.optimizers	Optimization algorithms such Adam, SGD and more

3 Model Building and Execution

This section provides detailed information on implementation steps.

3.1 Import Libraries

Start by importing all the necessary libraries:

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier
from tensorflow.keras.models import Sequential
from keras.layers import Dense, Input
from tensorflow.keras.utils import to_categorical
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_sc
from sklearn.model_selection import learning_curve
import matplotlib.pyplot as plt
import seaborn as sns
import itertools
from keras_tuner import RandomSearch
from tensorflow.keras.optimizers import Adam
```

3.2 Data Preprocessing

Data preprocessing involves several steps which are, loading the dataset, visualizing the data, encoding categorical values, splitting the data into train and test and standardizing the features.

```
# Handle categorical data
le = LabelEncoder()
data['RiskLevel'] = le.fit_transform(data['RiskLevel'])

# Split features and target
X = data.drop('RiskLevel', axis=1)
y = data['RiskLevel']

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Standardize the data
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

```
# Load data
file_path = 'C:/Users/NIHARIKA/R&C/Maternal Health Risk Data Set.csv'
data = pd.read_csv(file_path)

# Plot histograms for each feature
plt.figure(figsize=(14, 10))
for i, column in enumerate(data.columns):
    plt.subplot(len(data.columns) // 3 + 1, 3, i + 1)
    plt.hist(data[column], bins=20, edgecolor='k', alpha=0.7)
    plt.title(f'Histogram of {column}')
    plt.xlabel(column)
    plt.ylabel('Frequency')
plt.tight_layout()
plt.show()

# Plot boxplots for numeric features
numeric_columns = data.select_dtypes(include=[np.number]).columns
plt.figure(figsize=(14, 10))
for i, column in enumerate(numeric_columns):
    plt.subplot(len(numeric_columns) // 3 + 1, 3, i + 1)
    sns.boxplot(x=data[column])
    plt.title(f'Boxplot of {column}')
    plt.xlabel(column)
plt.tight_layout()
plt.show()

# Plot correlation heatmap
plt.figure(figsize=(12, 10))
correlation_matrix = data.corr(numeric_only=True)
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f', linewidth
plt.title('Correlation Heatmap')
plt.show()
```

3.3 Model Building and Evaluation

Three individual models were built and tuned separately and evaluated using metrics.

Random forest:

```
# Hyperparameter tuning for RandomForest
param_grid_rf = {
    'n_estimators': [100, 200, 300],
    'max_depth': [None, 10, 20, 30],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}

grid_search_rf = GridSearchCV(estimator=RandomForestClassifier(random_state=42)
grid_search_rf.fit(X_train, y_train)
best_rf = grid_search_rf.best_estimator_

# Predict with the best Random Forest model
rf_preds = best_rf.predict(X_test)
rf_accuracy = accuracy_score(y_test, rf_preds)
print(f'Tuned RF Accuracy: {rf_accuracy}')
```

XG Boost:

```
# Hyperparameter tuning for XGBoost
param_grid_xgb = {
    'n_estimators': [100, 200, 300],
    'max_depth': [3, 6, 9],
    'learning_rate': [0.01, 0.1, 0.2],
    'subsample': [0.8, 0.9, 1.0],
    'colsample_bytree': [0.8, 0.9, 1.0]
}

grid_search_xgb = GridSearchCV(estimator=XGBClassifier(use_label_encoder=False,
grid_search_xgb.fit(X_train, y_train)
best_xgb = grid_search_xgb.best_estimator_

# Predict with the best XGBoost model
xgb_preds = best_xgb.predict(X_test)
xgb_accuracy = accuracy_score(y_test, xgb_preds)
print(f'Tuned XGB Accuracy: {xgb_accuracy}')
```

Feed Forward Neural Network:

```
def build_model(hp):
    model = Sequential()

    # Specify the input shape using an Input layer
    model.add(Input(shape=(X_train.shape[1],)))

    # First dense layer
    model.add(Dense(units=hp.Int('units', min_value=32, max_value=256, step=32),

    # Additional dense layers based on the number of layers hyperparameter
    for i in range(hp.Int('num_layers', 1, 3)):
        model.add(Dense(units=hp.Int('units_' + str(i), min_value=32, max_value=

    # Output layer with softmax activation
    model.add(Dense(3, activation='softmax'))

    # Compiling the model
    model.compile(optimizer=Adam(hp.Choice('learning_rate', values=[1e-3, 1e-4])
                  loss='categorical_crossentropy',
                  metrics=['accuracy'])
    return model
```

3.4 Hybrid Model Development

The hybrid model consists of combined predictions of above all the three models and takes the output of these models as input and builds a hybrid neural network model.

```
# Combine predictions of individual models for hybrid model input
meta_features_train = np.column_stack([
    best_rf.predict_proba(X_train),
    best_xgb.predict_proba(X_train),
    fnn_model.predict(X_train)
])

meta_features_test = np.column_stack([
    best_rf.predict_proba(X_test),
    best_xgb.predict_proba(X_test),
    fnn_model.predict(X_test)
])
```

```
#Building hybrid model

def create_hybrid_model(input_dim):
    model = Sequential()
    model.add(Input(shape=(input_dim,)))
    model.add(Dense(64, activation='relu'))
    model.add(Dense(32, activation='relu'))
    model.add(Dense(3, activation='softmax'))
    model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['a
# Train the hybrid model
# Ensure the target is in categorical form for Keras
y_train_cat_meta = to_categorical(y_train)
hybrid_model = create_hybrid_model(meta_features_train.shape[1])
hybrid_model.fit(meta_features_train, y_train_cat_meta, epochs=50, batch_size=10, verbose=1)

# Evaluate the hybrid model
meta_preds = hybrid_model.predict(meta_features_test)
hybrid_preds_classes = np.argmax(meta_preds, axis=1)
hybrid_accuracy = accuracy_score(y_test, hybrid_preds_classes)
print(f'Hybrid Model Accuracy: {hybrid_accuracy}')
```

3.6 Graphs Plotting and visualizations

All the graphs, confusion matrices will be plotted properly, the execution of the complete code will go through smooth by importing all the libraries mentioned in the above section 3.2.