Configuration Manual

MSc Research Project
Programme Name

Shakoor Ahmad Bhat
Student ID: x19236280

School of Computing
National College of Ireland

Supervisor: Dr. Rashmi Gupta
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:

Date: 14th August 2021

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.
1 Introduction

The motive of this report is to provide details of the process followed during the coding phase of the research project. Hardware and software configurations are defined to reproduce the research in future. This contains the programming and employment stages for glossy code execution and the steps taken for executing the code.

2 System Configuration

2.1 Hardware Configuration

The hardware description and specification is shown in Figure 1 on which the code is executed:

Figure 1: Hardware configuration of the system

2.2 Software Configuration

This section provides the details of the software and its specifications.

2.2.1 Anaconda - Jupyter Notebook:

Anaconda is an open source[^1]. Anaconda is a Python and R distribution (prebuilt and preconfigured collection of packages) that is commonly used for data science. Anaconda Navigator is a GUI tool that is included in the Anaconda distribution and makes it easy to configure, install, and launch tools such as Jupyter Notebook. It can downloaded from the official website of Anaconda[^2]. The download options for Windows, MacOS and Linux

[^1]: https://www.anaconda.com/
[^2]: https://www.anaconda.com/products/individual
is shown in Figure 2

After installing the anaconda, the home page of Anaconda Navigator will display different Integrated Development Environment (IDE) Figure 3. Jupiter Notebook IDE is launched for code execution and development of various models using Python version 3.

2.2.2 Other Softwares

For report documentation we used Overleaf, Figure 4 shows the overleaf home page for report documentation.
The data visualization is done by using Scikit-learn package in Python as shown in Figure 5. The line chart shows the comparison of all models based on ROC Curve plot.

![ROC Comparison of All Models](https://scikit-learn.org/0.24/visualizations.html)
3 Data Preparation

The dataset was taken from Kaggle repository as shown in Figure 6. The dataset has one folder and a csv file is provided with category (0 as Normal women and 1 as PCOS women).

![Figure 6: Polycystic ovary syndrome (PCOS) dataset for research project](https://www.kaggle.com/prasoonkottarathil/polycystic-ovary-syndrome-pcos)

After this dataset was loaded into the python by using this code as shown in Figure 7.

![Figure 7: Loading the PCOS dataset](https://www.kaggle.com/prasoonkottarathil/polycystic-ovary-syndrome-pcos)

Then we imported all the libraries as shown in Figure 8.

![Figure 8: Importing Libraries](https://www.kaggle.com/prasoonkottarathil/polycystic-ovary-syndrome-pcos)

Further, after checking the missing values in our dataset by using isna().sum() function.
Univariate Feature selection method\^[7] were implemented to select the top 10 features by using SelectKbest and Chi2 packages which will help us to detect the PCOS, code shown in Figure 9.

![Univariate feature selection method](image)

**Figure 9:** Implementation of Univariate Feature Selection Method

### 4 Data Transformation

After the data pre-processing data, new dataframe is created based on top 10 features as shown in Figure 10.

![New Dataframe based on top 10 features](image)

**Figure 10:** New Dataframe based on top 10 features

After making a new dataframe, the data is split into train and test having test size as 0.25 and random state as 27 as shown in Figure 11.

![Splitting](image)

**Figure 11:** Splitting of dataset into train and test set

Furthermore, SMOTE was used to solve the imbalance problem by randomly increasing

minority class examples by replicating them. It was used on training and testing set separately as shown in Figure 12.

![SMOTE](image)

**SMOTE**

```python
1 # Upsampling the Training set
2 sm = SMOTE(random_state=23, sampling_strategy='minority')
3 x_train_sm, y_train_sm = sm.fit_resample(x_train, y_train)
4 print(len(x_train_sm), len(y_train_sm))
```

```python
1 # Upsampling the Testing set
2 sm_test = SMOTE(random_state=23, sampling_strategy='minority')
3 x_test_sm, y_test_sm = sm_test.fit_resample(x_test, y_test)
4 print(len(x_test_sm), len(y_test_sm))
```

Figure 12: SMOTE on training and testing set

5 Implementation of Baseline Models

After data pre-processing and data transformation, data can be used for implementation using the baseline models such as Gradient Boosting, Random Forest, Logistic Regression, HRFLR, SVM, Decision Tree, MLP.

5.1 Gradient Boosting

5.1.1 Model Building

After importing the Gradient Boosting classifier, as it helps to minimize the loss, or the difference between the actual class value of the training example and the predicted class value. The hyper parameter settings were \( n\text{\_estimators} = 20, learning\_rate = 0.5, max\_features = 2, max\_depth = 2, random\_state = 0 \). The code for model development of Gradient Boosting is shown in Figure 13. Moreover, Data were prepared for start and end time by using fit() function and prediction were made on training and testing time.

![Gradient Boosting Classifier](image)

```python
1 from sklearn.ensemble import GradientBoostingClassifier
2 gb = GradientBoostingClassifier(n_estimators=20, learning_rate = 0.5, max_features=2, max_depth = 2, random_state = 0)
3 gb.fit(x_train, y_train)
```

Figure 13: Model building of Gradient Boosting
5.1.2 Model Evaluation

The evaluation matrices were accuracy, precision, recall, f1-score, ROC curve plot and AUC score. Further, confusion matrix and classification report is generated using sklearn.metrics\(^6\). The code and calculation for these matrices is shown in Figure 14.

![Figure 14: Model evaluation of Gradient Boosting](image)

After implementing the above code we got the output for all the matrices as shown in Figure 15.

![Figure 15: Confusion matrix and Classification report of Gradient Boosting](image)

Now we will check how the model is expected to perform in general when used to make predictions on data not used during the training of the model by using the K-fold Cross validation accuracy selecting k=10,20,30,40. The code is shown in Figure 16.

\(^6\)https://scikit-learn.org/0.15/modules/model_evaluation.html
After implementing the above code we got the output of K fold cross validation accuracy for Gradient Boosting as shown in Figure 17.

![Figure 17: Output of K fold cross validation accuracy](image)

5.2 Random Forest

5.2.1 Model Building

After importing the Random Forest classifier, as it builds multiple decision trees and merges them together to get a more accurate and stable prediction. The hyper parameter settings were \((n\_estimators = 10, criterion = entropy, random\_state = 0)\). The code for model development of RF Classifier is shown in Figure 18.
5.2.2 Model Evaluation

The RF model is evaluated in the same way as explained in Gradient Boosting. The code is same for RF as well using the RF.classifier.fit() and RF.classifier.predict() function for evaluating the matrix as shown in Figure 19. Figure 20 shows the output of confusion matrix, classification report, and cross validation accuracy when $k=10,20,30,40$. 

```python
from sklearn.ensemble import RandomForestClassifier

RF_classifier = RandomForestClassifier(n_estimators = 100, criterion = 'entropy', random_state = 0)
results = []

# Training the model
start = time()
RF_classifier.fit(x_train_sm, y_train_sm)
end = time()
results['training_time'] = end - start

# Testing the model
start = time()
RF_Prediction = RF_classifier.predict(x_test_sm)
end = time()
results['testing_time'] = end - start
```

Figure 18: Model building for Random Forest

Figure 19: Model evaluation for Random Forest
5.3 Logistic regression

5.3.1 Model Building

After importing the logistic regression classifier, as it is used to predict the categorical dependent variable using a given set of independent variables. The hyper parameter setting are (max_iter=10000, random_state = 0). The code for development of Logistic Regression is shown in Figure 21.

![Figure 21: Model Building for Logistic Regression](image)

5.3.2 Model Evaluation

The model is evaluated by using the two variables such as LR_classifier.fit() and LR_classifier.predict(). The code for accuracy, precision, recall, f1-score, cross validation accuracy is shown in Figure 22. The output confusion matrix, classification report and cross validation accuracy is shown in Figure 23.

![Figure 22: Model Evaluation for Logistic Regression](image)

![Figure 23: Output of Model Evaluation](image)
5.4 Hybrid Random Forest and Logistic Regression (HRFLR)

5.4.1 Model Building

First the sub model were created using the estimators = [], than logistic model were defined using hyper parameter setting as (random state = 0,C=1, max_iter=10000). After this three Random Forest models were defined such as model121, 122 and 123. At last HRFLR model were ensemble using the voting classifier package taking voting = soft. The code is shown in Figure 24.

Figure 22: Model evaluation for Logistic Regression

Figure 23: Model evaluation output for Logistic Regression
5.4.2 Model Evaluation

The model is evaluated by using the two variables such as HRFLR_ensemble.fit() and HRFLR_ensemble.predict(). The code for accuracy, precision, recall, f1-score, cross validation accuracy is shown in Figure 25. The output confusion matrix, classification report and cross validation accuracy with cross validation time is shown in Figure 26.
5.5 Feature Importance

By using the permutation_importance() function with (HRFLR_ensemble, x_train_sm, y_train_sm, n_repeats=10, random_state=0) as hyper parameter which will improve the efficiency and effectiveness of a predictive model on the problem. The code with output is shown in Figure 27.
5.6 Support Vector Machine

5.6.1 Model Building

SVM is developed by using the Support Vector Classifier SVC() function having random_state = 0, probability=True as hyper parameter. SVM algorithm creates a line or a hyperplane which separates the data into classes. The code is shown in Figure 28.

```
# Support Vector Machines

results = {}
4 #Training the model
start = time()
5 SVMClassifier.fit(x_train_sm, y_train_sm)
6 end = time()
7 results['training_time'] = end - start
8
10 #Testing the model
11 start = time()
12 SVMPrediction = SVMClassifier.predict(x_test_sm)
13 end = time()
14 results['testing_time'] = end - start
```

Figure 28: Model building for SVM

5.6.2 Model Evaluation

The model is evaluated by using SVMClassifier.fit() and SVMClassifier.predict(). The code for accuracy, precision, recall, f1-score, cross validation accuracy is shown in Figure 29. The output confusion matrix, classification report and cross validation accuracy with cross validation time is shown in Figure 30.
5.7 Decision Tree

5.7.1 Model Building

The model is created by using the DecisionTreeClassifier() function having criterion=entropy and random_state = 0. It is an framework to quantify the values of outcomes and the probabilities of achieving them because DT handles non-linear data sets effectively. The code is shown in Figure 31.
5.7.2 Model Evaluation

The model is evaluated by using `DT_classifier.fit()` and `DT_classifier.predict()`. The code for accuracy, precision, recall, f1-score, cross validation accuracy is shown in Figure 32. The output confusion matrix, classification report and cross validation accuracy with cross validation time is shown in Figure 33.
5.8 Multi layer Perceptron

5.8.1 Model Building

MLP classifier is used for building Multi layer Perceptron with three 8,8,8 hidden layer, RELU as activation, 500=iterations and random_state=0. It is suitable for classification prediction problems where inputs are assigned a class or label. Code is shown in Figure 34.

![Figure 34: Model building for MLP](image)

5.8.2 Model Evaluation

The model is evaluated by using MLP.classifier.fit() and MLP.classifier.predict(). The code for accuracy, precision, recall, f1-score, cross validation accuracy is shown in Figure 35. The output confusion matrix, classification report and cross validation accuracy with cross validation time is shown in Figure 36.

![Figure 33: Model evaluation output for Decision Tree](image)
5.9 Comparison of all baseline models based on ROC curve plot

We used a majority class no skill prediction code where 0 for 1 range is used on test data. Lables were generated as true positive rate and false negative rate and ROC comparison were made of baseline approach models. This graphical way tells us the connection between sensitivity and specificity for every possible cut-off of a data test. Code is shown in Figure 37. Output is shown in Figure 38.
6 Implementation of Newly Proposed Models

Hybrid ensemble of Extreme Boosting with Random Forest (XGBRF) and CatBoost is the novelty of this research project. These two models are never used for PCOS detection. Both models deal with handling the classification problem if the data is categorical in nature. The code is referred from [Bhatele and Bhadauria, 2020] [Li et al., 2020].

6.1 Multi Hybrid ensemble of Extreme Boosting with Random Forest Perceptron

6.1.1 Model Building

First, XGBRFCClassifier is imported having hyper parameters as (max_depth=3, random_state=8) then xgb clf.fit() is used on pre-trained data. They are assigned to all independent variables and are fed into decision trees to predict the results. Figure 39 shows us the code for XGBRF.
6.1.2 Model Evaluation

acc_xgb_clf_train() and acc_xgb_clf_test() is used to get the accuracy of the model. xgb_clf.fit() and xgb_clf.predict() is used for precision, recall, f1-score, cross validation accuracy is shown in Figure 40. The output confusion matrix, classification report and cross validation accuracy with cross validation time is shown in Figure 41.
6.2 CatBoost

6.2.1 Model Building

After importing CatBoostClassifier with np.set_printoptions(precision=4) having hyper parameters as (iterations=199, learning_rate=0.1), cat_clf.fit() is used on pre-trained data. CatBoost is very effective algorithm of handling categorical features. It is fast and easy to use. Code is shown in Figure 42.

```python
# CatBoost
import os
import pandas as pd
import numpy as np
np.set_printoptions(precision=4)
import catboost
print(catboost.__version__)
from catboost import CatBoostClassifier

# CatBoost Classifier
cat_clf = CatBoostClassifier(iterations=199,
learning_rate=0.1)
cat_clf.fit(x_train,y_train)
acc_cat_clf_train = round(cat_clf.score(x_train, y_train)*100,2)
acc_cat_clf_test = round(cat_clf.score(x_test,y_test)*100,2)
#result.append(acc_cat_clf_train)
print("Training Accuracy: % {}\".format(acc_cat_clf_train))
#result.append(acc_cat_clf_test)
print("Testing Accuracy: % {}\".format(acc_cat_clf_test))

# Training the model
start = time()
cat_clf.fit(x_train_sm, y_train_sm)
end = time()
results[['training_time']] = end - start

# Testing the model
start = time()
cat_Prediction = cat_clf.predict(x_test_sm)
end = time()
results[['testing_time']] = end - start
```

Figure 41: Model Evaluation output of XGBRF

Figure 42: Model Building of CatBoost
6.2.2 Model Evaluation

acc_cat_clf_train() and acc_cat_clf_test() is used to get the accuracy of the model. cat_clf.fit() and cat_clf.predict() is used for precision, recall, f1-score, cross validation accuracy is shown in Figure 43. The output confusion matrix, classification report and cross validation accuracy with cross validation time is shown in Figure 44.

```python
# Definition

@cat_clf = CatBoostClassifier() # For classification

# Train
@cat_clf.train(x_train, y_train)

# Predict
@prediction = cat_clf.predict(x_test)

# Evaluation
print('The precision score for CatBoostClassifier is: %.2f' % precision_score(y_test, prediction))
print('The recall score for CatBoostClassifier is: %.2f' % recall_score(y_test, prediction))
print('The F1 Score for CatBoostClassifier is: %.2f' % f1_score(y_test, prediction))

classification_report(x_test, y_test, cat_clf)

cross_val_score(x_train, y_train, cat_clf)

classification_report(x_train, y_train, cat_clf)

cross_val_score(x_test, y_test, cat_clf)
```

Figure 43: Model Evaluation of CatBoost

Figure 44: Model Evaluation output of CatBoost

The scripts and functions mentioned above are all provided in the ICT solution along with this project.
References
