Configuration Manual

MSc Research Project
Data Analytics

Aruna Saravanapandian
Student ID: x22182349

School of Computing
National College of Ireland

Supervisor: Prof. Mohammed Hasanuzzaman

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name: Aruna Saravanapandian

Student ID: x22182349

Programme: Data Analytics **Year:** 2024

Module: MSc Research project

Lecturer: Mohammed Hasanuzzaman

Submission Due Date: 12/12/2024

Project Title: Prediction of Resource utilization in cloud computing using machine learning

Word Count: 698 **Page Count:** 10

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: Aruna Saravanapandian

Date: 03/12/2024

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

Aruna Saravanapandian
X22182349

1 Introduction

This document allows us to setup the environment required for the research project along with the hardware and the software requirements. The instructions for the research work is included in the document such as data preprocessing, data transformation, model building, evaluation.

2 Hardware and Software Requirements

2.1 Hardware configuration

The research has been performed in a personal machine with the following configuration with 8GB of RAM, 64-bit operating system, and Intel core i7 processor.

2.2 Software configuration

The programming language used is Python 3.11.5, along with Jupyter notebook as the Integrated development environment.

3 Implementation

3.1 Data collection

The dataset that is used for the research project is obtained from the open-source dataset Kaggle as shown in Figure 1.

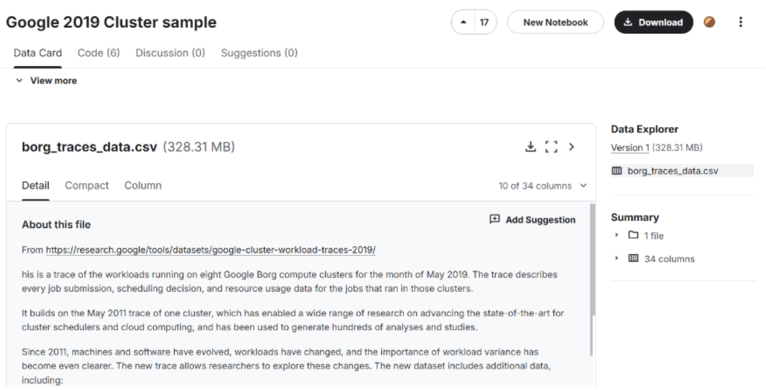

Figure 1 : Dataset from kaggle

3.2 Importing libraries

The necessary python libraries that are required for the project are installed. The libraries allow the data to allow the exploratory data analysis, preprocessing, model building and evaluation as seen in Figure 2.

```
[3]: import numpy as np
import pandas as pd
import re
import time
from sklearn.preprocessing import OneHotEncoder
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from sklearn.linear_model import LinearRegression, Ridge, Lasso
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor
from sklearn.tree import DecisionTreeRegressor
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import Sequential, layers
from tensorflow.keras.layers import Dense
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
```

Figure 2: Importing libraries

3.3 Reading the input file

The input data file is borg_traces_data.csv is the file obtained from Kaggle and is read using the pd.readcsv() function as shown in Figure 3.

```
# Read the input data

resource = pd.read_csv('borg_traces_data.csv',index_col=0)
resource.head()
```

Figure 3 : Reading the input file

3.4 Data Visualization

The Figure 4 shows the histogram of the data and their distribution across the values.

Figure 4 : Shows the histogram of the data

3.5 Analysing the missing data

The missing data is analysed and the column with higher than 75 percent of the data that is missing is removed in Figure 5.

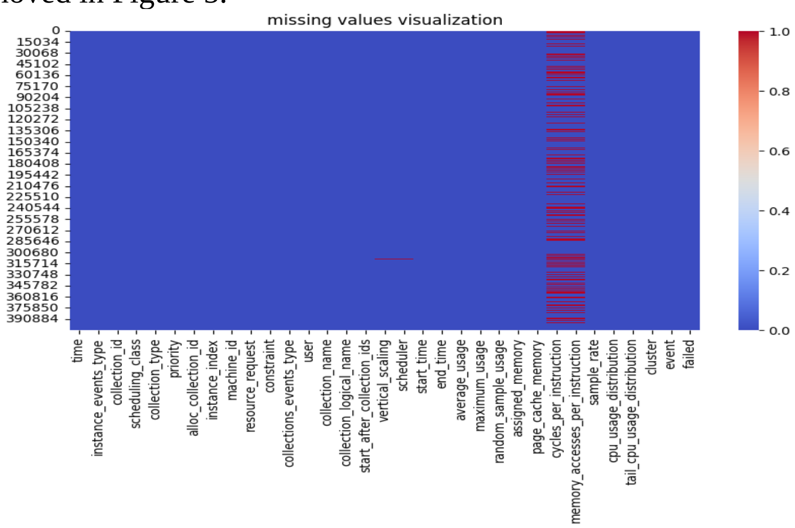

Figure 5: visualization of the missing data

3.6 Data Preprocessing

The outliers are determined in Figure 6 and removed.

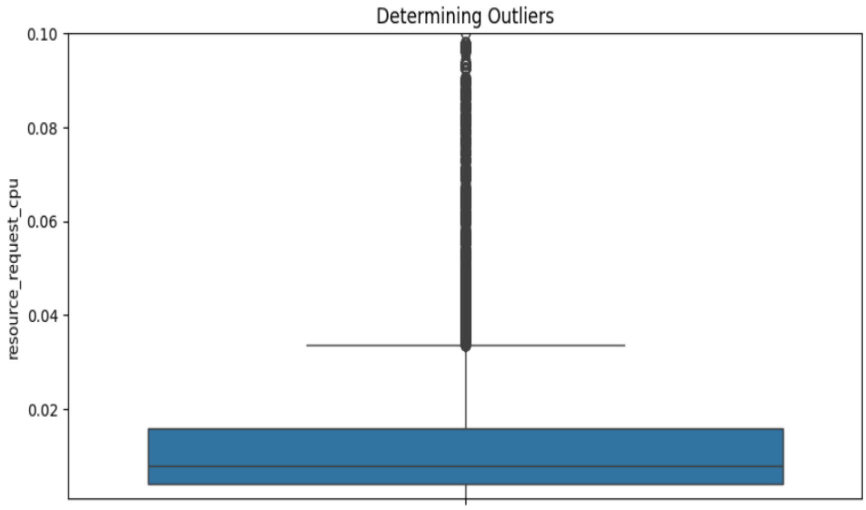

Figure 6: outlier detection

3.7 Data Normalisation

The variables are normalised using minmaxscaler() as shown in Figure 7.

Normalizing the variables

```
[39]: from sklearn.preprocessing import MinMaxScaler
```

```
[40]: scaler = MinMaxScaler()
      X_train = scaler.fit_transform(X_train)
      X_test = scaler.transform(X_test)
```

Figure 7: MinMaxScaler for normalisation

3.8 Model building

The machine learning models and statistical models like linear regression, ridge regression, lasso regression, Random Forest regression, Decision tree regression, gradient boosting regression and Artificial neural network as shown in Figure 8,9,10,11,12,13,14 respectively.

Linear Regression

```
[41]: from sklearn.linear_model import LinearRegression

[42]: from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score

[43]: model = LinearRegression()

[44]: start = time.time()

[45]: #Training model with Training data
      model.fit(X_train, y_train)

[45]: LinearRegression
      LinearRegression()

[46]: linear_execution_time = round(time.time()-start,2)

[47]: y_pred = model.predict(X_test)
```

Figure 8: Linear regression

Ridge Regression

```
[50]: from sklearn.linear_model import Ridge

[51]: model = Ridge(alpha=1.0)

[52]: start = time.time()

[53]: #Training model with Training data
      model.fit(X_train, y_train)

[53]: Ridge
      Ridge()

[54]: ridge_execution_time = round(time.time()-start,2)

[55]: y_pred = model.predict(X_test)
```

Figure 9: Ridge regression

Lasso Regression

```
[58]: from sklearn.linear_model import Lasso

[59]: model = Lasso(alpha=0.0001)

[60]: start = time.time()

[61]: #Training model with Training data
      model.fit(X_train, y_train)

[61]: Lasso
      Lasso(alpha=0.0001)

[62]: lasso_execution_time = round(time.time()-start,2)

[63]: y_pred = model.predict(X_test)
```

Figure 10: Lasso regression

Random Forest Regression

```
[66]: from sklearn.ensemble import RandomForestRegressor

[67]: model = RandomForestRegressor(n_estimators=10, random_state=42)

[68]: start = time.time()

[69]: #Training model with Training data
      model.fit(X_train, y_train)

[69]: RandomForestRegressor
      RandomForestRegressor(n_estimators=10, random_state=42)

[70]: random_execution_time = round(time.time()-start,2)

[71]: y_pred = model.predict(X_test)
```

Figure 11: Random forest regression

Gradient Boosting Regression

```
[74]: from sklearn.ensemble import GradientBoostingRegressor

[75]: model = GradientBoostingRegressor(n_estimators=10, learning_rate=0.1, max_depth=3, random_state=42)

[76]: start = time.time()

[77]: #Training model with Training data
      model.fit(X_train, y_train)

[77]: GradientBoostingRegressor
      GradientBoostingRegressor(n_estimators=10, random_state=42)

[78]: gradient_execution_time = round(time.time()-start,2)

[79]: y_pred = model.predict(X_test)
```

Figure 12: Gradient boosting regression

Decision Tree Regression

```
[82]: from sklearn.tree import DecisionTreeRegressor

[83]: model = DecisionTreeRegressor()

[84]: start = time.time()

[85]: #Training model with Training data
      model.fit(X_train, y_train)

[85]: DecisionTreeRegressor
      DecisionTreeRegressor()

[86]: decision_execution_time = round(time.time()-start,2)

[87]: y_pred = model.predict(X_test)
```

Figure 13: Decision tree regression

Aritificial Neural Network

```
[91]: import tensorflow as tf
      from tensorflow import keras
      from tensorflow.keras import Sequential, layers
      from tensorflow.keras.layers import Dense
```

```
[92]: model = keras.Sequential([
          keras.layers.Dense(32, input_shape=(45,), activation='relu'),
          keras.layers.Dense(16, activation = 'relu'),
          keras.layers.Dense(1)
      ])
```

```
C:\Users\Ramad\AppData\Local\Programs\Python\Python311\Lib\site-packages\keras\src\layers\core\dense.py:87: UserWarning: D
input_dim` argument to a layer. When using Sequential models, prefer using an `Input(shape)` object as the first layer in
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

```
[93]: model.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 32)	1,472
dense_1 (Dense)	(None, 16)	528
dense_2 (Dense)	(None, 1)	17

Total params: 2,017 (7.88 KB)
Trainable params: 2,017 (7.88 KB)
Non-trainable params: 0 (0.00 B)

```
[94]: model.compile(optimizer='adam', loss='mse', metrics=['mae'])
```

```
[95]:
      start = time.time()

      ann = model.fit(X_train, y_train, epochs=10, batch_size=32, validation_split=0.2, verbose=1)
```

Figure 14: Artificial Neural Network

3.9 Evaluation

The outcome of the models with all the features in Figure 15 and without the feature removed as shown in Figure 16.

```
[169]: metrics = pd.DataFrame({"Model": models, "R2": r2_score, "MSE": mse, "MAE": mae, "RMSE" : rsme, "Execution Time": execution_time})

       print(metrics)
```

```
                       Model        R2       MSE       MAE      RMSE  \
0          Linear Regression  0.547141  0.000016  0.002850  0.003992
1           Ridge Regression  0.547167  0.000016  0.002852  0.003992
2           Lasso Regression  0.425282  0.000020  0.003438  0.004497
3          Random Regression  0.921258  0.000003  0.000553  0.001665
4        Decision Regression  0.900014  0.000004  0.000580  0.001876
5        Gradient Regression  0.493325  0.000018  0.003274  0.004222
6  Artificial Neural Network  0.832437  0.000006  0.001376  0.002428

   Execution Time
0            1.97
1            1.18
2            1.21
3           15.35
4            2.04
5            4.69
6          120.52
```

Figure 15: Evaluation of model with all the features

```
[182]: metrics = pd.DataFrame({"Model": models, "R2": r2_score, "MSE": mse, "MAE": mae, "RMSE" : rsme, "Execution Time": execution_time})

print(metrics)
                     Model        R2       MSE       MAE      RMSE  \
0        Linear Regression  0.530796  0.000017  0.002960  0.004063
1         Ridge Regression  0.530481  0.000017  0.002961  0.004065
2         Lasso Regression  0.072922  0.000033  0.004557  0.005711
3        Random Regression  0.920811  0.000003  0.000555  0.001669
4      Decision Regression  0.900484  0.000004  0.000580  0.001871
5      Gradient Regression  0.492490  0.000018  0.003292  0.004226
6  Artificial Neural Network  0.819923  0.000006  0.001454  0.002517

   Execution Time
0            1.10
1            0.17
2            0.30
3           16.21
4            2.00
5            4.59
6          120.34
```

Figure 16: Evaluation of model with features removed

References