National College of Ireland

Configuration Manual

MSc Research Project
Data Analytics

Digvijay Rai
Student ID: x18134645

School of Computing
National College of Ireland

Supervisor: Dr. Muhammad Iqbal

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Digvijay Rai
Student ID:	x18134645
Programme:	Data Analytics
Year:	2019
Module:	MSc Research Project
Supervisor:	Dr. Muhammad Iqbal
Submission Due Date:	13/12/2019
Project Title:	Configuration Manual
Word Count:	1945
Page Count:	13

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:	11th December 2019

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

Digvijay Rai
x18134645
MSc Research Project in Data Analytics

13th December 2019

1 Introduction

This configuration manual provides information on the system setup, the necessary hardware, and software requirements and programming codes used to implement this research project:

"Predicting Energy Consumption in Commercial Buildings using Property Features and Machine Learning Algorithms."

2 System Configuration

To start with any machine learning project, it is necessary to have a system with high configuration, which will help to carry out the research project smoothly. It is also necessary to have all the prerequisites installed on the system before starting with the research project.

2.1 Hardware

Processor: Intel(R) Core (TM) i7-8550U CPU@ 4GHz GPU: Intel UHD Graphics 620, Radeon (TM) 530; RAM:16GB Storage: 1 TB HDD, SSD: 128 GB; Operating system: Windows 10, 64-bit.

2.2 Software

1. Microsoft Excel 2016: Used for saving data, data analysis, and to plot explorative graphs.

2. Jupyter Notebook: Data manipulation, cleansing and pre-processing, feature engineering methods, and execution of machine learning models.

3. Google Colab: The entire project is moved to cloud-based software, which provides free service and GPU.

3 Project Development

The development of this project contains numerous steps: data analysis (statistics, charts), data pre-processing (data preparation by removing columns having more than 20% of missing values, binning data in four categories and handling class imbalance issue), feature engineering (using dimensionality reduction method i.e., Principal Component Analysis (PCA) and feature selection method Analysis of Variance (ANOVA) and implementation of classification machine learning algorithms. A numerous line of codes have been written to execute several steps for the analysis: data re-sampling, saving models, extracting parameters of the confusion matrix, and K10 cross-fold validation technique. The code developed to execute this research project is shown below, with explanations provided at essential steps.

3.1 Data Preparation and Preprocessing

The 2012 Commercial Building Energy Consumption Survey (CBECS) dataset was downloaded in .csv format from U.S. Energy Information Administration (EIA) website, which was released in June 2015 and revised in August 2016[1] .

The first step is to import the necessary libraries required to start with data loading and data preprocessing steps.

```
#Importing necessary packages required for the prokect
import matplotlib.pyplot as plt # matplotlib --> to draw a 2D figure and pyplot to do changes in that figure.
import pandas as pd  # providing high performance and it is used data manipulation and analysis
import seaborn as sns  # to draw attractive and informational graphs (To visualize the data)
from sklearn.model_selection import cross_val_score
# sklearn is where all libraries are stored and model_selection to split data into test and train
```

The next step is to import the selected dataset, and the imported dataset is stored in the data frame known as “data”. The CBECS dataset contains 6,720 rows and 1119 variables, which was gathered from 5.6 million commercial buildings.

```
# Importing the dataset
data = pd.read_csv('datas.csv')
data.shape
```

```
(6720, 1119)
```

Removing missing values from the dataset was one of the most significant decisions to make, (Robinson et al.; 2017) removed the columns which were having more than 25% of missing values. This research was tested by removing columns that were having more than 20% and 30% of missing values, and there wasn’t a sign of the difference in terms of all the four-evaluation metrics. To avoid the data loss, this research was tested by removing the columns which were having more than 20% of missing values.

```
data.shape
```

```
(6720, 812)
```

[1]https://www.eia.gov/consumption/commercial/data/2012/index.php?view=microdata

Binning the dependent variable “Major Fuel in British Thermal unit” (MFBTU) of the dataset into four different categories by selecting a different range of energy consumption value is in Btu’s and storing the new generated categorical data in a new column called “EC” (Energy Consumption).

```
# Binning the data in four categories


bins = [0, 605000, 3100000, 15200000, 1418866360]
labels =['verylow','low','mid','high']

# Create new column name Electricity consumption
data['EC'] = pd.cut(data['MFBTU'], bins,labels=labels)
```

```
data.shape
```

```
(6720, 813)
```

Then the number of observations in each category must be checked in order to handle the class imbalance issue.

```
data[data['EC'] == 'verylow']
```

684 rows × 813 columns

```
data[data['EC'] == 'low']
```

678 rows × 813 columns

```
data[data['EC'] == 'mid']
```

696 rows × 813 columns

```
data[data['EC'] == 'high']
```

700 rows × 813 columns

Category “low” is having least rows counts amongst all the four categories, so random down-sampling have been performed on rest of three majority categories which bring down majority categories to minority category. This method helps to avoid unnecessary creation and addition of noise data to the primary dataset. In order to handle class imbalance issues first, the data was shuffled and then stored in four different data frames.

```
# Shuffle the Dataset.
#frac : float, optional, #Fraction of axis items to return. Cannot be used with n.
shuffle = data.sample(frac=1,random_state=10) # random_state --> Seed for the random number generator
```

```
# Put all the energy consumption levels in a separate dataset.
#Randomly select 678 observations from the both the majority class (low )
verylow = shuffle.loc[shuffle['EC'] == "low"]
low = shuffle.loc[shuffle['EC'] == "verylow"].sample(n=678,random_state=10)
mid = shuffle.loc[shuffle['EC'] == 'mid'].sample(n=678,random_state=10)
high = shuffle.loc[shuffle['EC'] == 'high'].sample(n=678,random_state=10)
```

```
verylow.shape
```

```
(678, 813)
```

```
low.shape
```

```
(678, 813)
```

```
mid.shape
```

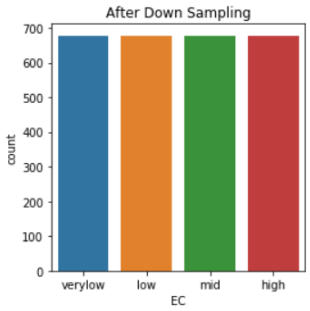
```
(678, 813)
```

```
high.shape
```

```
(678, 813)
```

All four different data frames were concatenated and stored in a single data frame known as "data." A graph was plotted with a figure size of 4,4 to check if all the categories have an equal number of observations.

```
# Concatenate both dataframes again
data = pd.concat([verylow, low, mid, high])
# Plot the graph to check if each class has equal number of observations.
plt.figure(figsize=(4, 4))
sns.countplot('EC', data=data)
plt.title('After Down Sampling')
plt.show()
```



Categorical values are the hidden text for machine learning algorithms, and it is necessary to encode the data correctly in advance. Before executing another machine learning algorithm, it is necessary to encode the data again and store the values in X and y and then store the data in X_train and y_train by dividing the dataset into 80% of training and 20% of testing data.

```
#Lable encoding for categorical data
from sklearn.preprocessing import LabelEncoder
#Lable encoding for catagorical data
columns = data.nunique()[data.nunique() == 4].keys().tolist()
labelEncoder = LabelEncoder()
for i in columns :
    data[i] = labelEncoder.fit_transform(data[i])

X = data.iloc[:, : -1].values
y = data.iloc[:, -1].values
```

```
# Splitting the encoded data into the Training set and Test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20, random_state = 10)
```

```
X_train.shape
```

```
(2169, 812)
```

```
X_test.shape
```

```
(543, 812)
```

3.2 Feature Engineering

Two different feature engineering methods have been used in this project, namely, principal component analysis (PCA) and analysis of variance (ANOVA).

A "for loop" is written and executed for both principal component analysis and analysis of variance method from 810 features and reduced the number of elements by 10 for each time the "for loop" runs, and it is re-run till the machine learning algorithm achieves the best accuracy.

3.2.1 Principal Component Analysis

A dimensionality reduction method is known as "Principal Component Analysis," is used to downscale the variables from a vast dataset. The variable count is reduced to 10 to test accuracy, precision, recall, and f1 score of classification machine learning algorithms, and the dataset was divided into 80% of train data and 20% of test data.

"For loop" for Principal Component Analysis Method

```
# Dimensionality Reduction using Principal Component Analysis
from sklearn.decomposition import PCA
features = range(10, 810, 10)
for i in list(reversed(features)):

    pca = PCA(n_components = i)
    X1 = pca.fit_transform(X)
    # Splitting the dataset into the Training set and Test set

    X_pca_train, X_pca_test, y_pca_train, y_pca_test = train_test_split(X1, y, test_size = 0.20, random_state = 10)
```

3.2.2 Analysis of Variance

Analysis of Variance (ANOVA) helps in choosing the best features from the dataset. A statistical method used to study the means for various groups which are significantly different from each other.

"For loop" for Analysis of Variance Method

```
# Feature Selection using Analysis of Variance (ANOVA)
features = range(50, 810, 10)
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import f_classif
for i in list(reversed(features)):
    print("###  Current Features {}".format(i))
    fvalue_selector = SelectKBest(f_classif, k= i)
    X_kbest = fvalue_selector.fit_transform(X, y)

# Splitting the dataset into the Training set and Test set
X_anova_train, X_anova_test, y_anova_train, y_anova_test = train_test_split(X_kbest, y, test_size = 0.20, random_state = 10)
```

4 Codes for machine learning models

4.1 Implementation using Principal Component Analysis

A dimensionality reduction method is known as principal component analysis (PCA) that is used by the researcher (Platon et al.; 2015) to predict the hourly energy consumption of the institutional building.

4.1.1 Gaussian Naive Bayes

Before executing the "for loop," it is necessary to execute the label encoding step as it will encode the dataset so that it can be fed to the classification machine learning algorithms. The written "for loop" is executed from 810 features to one feature, and it is found that Gaussian Naïve Bayes made the best accuracy by using two features.

```
from sklearn.decomposition import PCA
pca = PCA(n_components = 2)
X1 = pca.fit_transform(X)
# Splitting the dataset into the Training set and Test set
X_pca_train, X_pca_test, y_pca_train, y_pca_test = train_test_split(X1, y, test_size = 0.20, random_state = 10)
```

Machine learning code for Gaussian Naive Bayes with all four evaluation metrics namely accuracy, f1score, precision, and recall[2]. The accuracy of the model is evaluated using a k10 fold cross-validation accuracy and confusion matrix.

```
# Gaussian Naive Bayes
from sklearn.naive_bayes import GaussianNB
gaussian = GaussianNB()
gaussian.fit(X_pca_train, y_pca_train)
# Predicting the Test set results
y_pred = gaussian.predict(X_pca_test)
#10-fold cross validation score
accuracy = cross_val_score(estimator = gaussian, X = X_pca_train, y = y_pca_train, cv =10)
across_cv = accuracy.mean()
#Accuracy
from sklearn.metrics import accuracy_score
accuracy = accuracy_score(y_pca_test, y_pred)
print(i)
print(accuracy)
print(across_cv)

#Confusion matrix  and Evaluation Metrics
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
EC
0.9337016574585635
0.9330684932045594
[[118   0  10   0]
 [  0 110   3   8]
 [  0  10 131   0]
 [  0   5   0 148]]
             precision    recall  f1-score   support

          0       1.00      0.92      0.96       128
          1       0.88      0.91      0.89       121
          2       0.91      0.93      0.92       141
          3       0.95      0.97      0.96       153

avg / total       0.94      0.93      0.93       543
```

4.1.2 Random Forest Classifier

Before executing the "for loop," it is necessary to execute the label encoding step as it will encode the dataset so that it can be fed to the classification machine learning algorithms.

"for loop" has been run and it is noticed that random forest classifier is providing best accuracy by considering seven features.

```
from sklearn.decomposition import PCA
#features = range(1, 16, 1)
#for i in list(reversed(features)):
# Seven features have been considered for Random Forest Classifier
pca = PCA(n_components = 7)
X1 = pca.fit_transform(X)
# Splitting the dataset into the Training set and Test set

X_pca_train, X_pca_test, y_pca_train, y_pca_test = train_test_split(X1, y, test_size = 0.20, random_state = 10)
```

[2]https://www.datacamp.com/community/tutorials/naive-bayes-scikit-learn

Machine learning code for Random Forest Classifier with all four evaluation metrics namely accuracy, f1score, precision, and recall[3]. The accuracy of the model is evaluated using a k10 fold cross-validation accuracy and confusion matrix.

```
# Random Forest
from sklearn.ensemble import RandomForestClassifier
randomf = RandomForestClassifier()
randomf.fit(X_pca_train, y_pca_train)
# Predicting the Test set results
y_pred = randomf.predict(X_pca_test)
#10-fold cross validation score
accuracy = cross_val_score(estimator = randomf, X = X_pca_train, y = y_pca_train, cv =10)
randomf_cross = accuracy.mean()
#Accuracy
accuracy = accuracy_score(y_pca_test, y_pred)
print("Random Forest")
#Confusion matrix and Evaluation Metrics
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
Random Forest
EC
-- 0.9613259668508287
-- 0.9709070780145888
[[125   0   3   0]
 [  0 117   2   2]
 [  3   6 132   0]
 [  0   5   0 148]]
             precision    recall  f1-score   support

          0       0.98      0.98      0.98       128
          1       0.91      0.97      0.94       121
          2       0.96      0.94      0.95       141
          3       0.99      0.97      0.98       153

avg / total       0.96      0.96      0.96       543
```

4.1.3 Logistic Regression

Before executing the "for loop," it is necessary to execute the label encoding step as it will encode the dataset so that it can be fed to the classification machine learning algorithms.

After the execution of "for loop," it is noticed that the logistic regression machine learning model is providing the best accuracy by considering 410 features. And then, the dataset is divided into 80% for training and 20% for testing.

```
from sklearn.decomposition import PCA
#features = range(10, 810, 10)
#for i in list(reversed(features)):
# 410 features have been considered for Logistic Regression
pca = PCA(n_components = 410)
X1 = pca.fit_transform(X)
# Splitting the dataset into the Training set and Test set

X_pca_train, X_pca_test, y_pca_train, y_pca_test = train_test_split(X1, y, test_size = 0.20, random_state = 10)
```

Machine learning code for Logistic Regression with all four evaluation metrics namely accuracy, f1score, precision, and recall[4]. The accuracy of the model is evaluated using a k10 fold cross-validation accuracy and confusion matrix. As per the result, the logistic regression model is achieving an accuracy of 69.98%, f1 score, precision, recall accuracy of 70%, and a K10-fold cross validation accuracy of 66.98%.

[3]https://towardsdatascience.com/an-implementation-and-explanation-of-the-random-forest-in-python-77bf308a9b76

[4]https://towardsdatascience.com/logistic-regression-python-7c451928efee

```
#Logistic Regression
from sklearn.linear_model import LogisticRegression
lregression = LogisticRegression()
lregression.fit(X_pca_train, y_pca_train)
y_pred = lregression.predict(X_pca_test)
#10-fold cross validation score
accuracy = cross_val_score(estimator = lregression, X = X_pca_train, y = y_pca_train, cv =10)
lregression_cross = accuracy.mean()
#Accuracy
from sklearn.metrics import accuracy_score
accuracy = accuracy_score(y_pca_test, y_pred)
print("-- {}".format(accuracy))
print("-- {}".format(lregression_cross))
#Confusion matrix and Evaluation metrics
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
-- 0.6998158379373849
-- 0.6698703345561161
[[105   2  21   0]
 [  0  24  47  50]
 [  1   6 127   7]
 [  0  12  17 124]]
             precision    recall  f1-score   support

          0       0.99      0.82      0.90       128
          1       0.55      0.20      0.29       121
          2       0.60      0.90      0.72       141
          3       0.69      0.81      0.74       153

avg / total       0.70      0.70      0.67       543
```

4.1.4 K-Nearest Neighbor

Before executing the "for loop," it is necessary to execute the label encoding step as it will encode the dataset so that it can be fed to the classification machine learning algorithms.

"for loop" has been run and it is noticed that random forest classifier is providing best accuracy by considering 12 features.

```
from sklearn.decomposition import PCA
#features = range(1, 16, 1)
#for i in list(reversed(features)):
    # 12 features considered to get best accuracy
pca = PCA(n_components = 12)
X1 = pca.fit_transform(X)
# Splitting the dataset into the Training set and Test set

X_pca_train, X_pca_test, y_pca_train, y_pca_test = train_test_split(X1, y, test_size = 0.20, random_state = 10)
```

Machine learning code for K-Nearest Neighbor with all four evaluation metrics namely accuracy, f1score, precision, and recall[5]. The accuracy of the model is evaluated using a k10-fold crossvalidation accuracy and confusion matrix. As per the result, the K-Nearest Neighbor model is achieving an accuracy of 97.05%, f1 score, precision, recall accuracy of 97%, and a ten-fold cross-validation accuracy of 97.41%

[5]https://stackabuse.com/k-nearest-neighbors-algorithm-in-python-and-scikit-learn/

```
# KNN

from sklearn.neighbors import KNeighborsClassifier
KNC = KNeighborsClassifier(n_neighbors=5)
KNC.fit(X_pca_train, y_pca_train)
y_pred = KNC.predict(X_pca_test)
#10-fold cross validation score
accuracy = cross_val_score(estimator = KNC, X = X_pca_train, y = y_pca_train, cv =10)
KNC_cross = accuracy.mean()
print("10-Fold Cross  Validation Score of KNC", KNC_cross)

from sklearn.metrics import accuracy_score
accuracy = accuracy_score(y_pca_test, y_pred)
print("-- {}".format(accuracy))
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
10-Fold Cross  Validation Score of KNC 0.9741264837834622
-- 0.9705340699815838
[[123   0   5   0]
 [  0 116   0   5]
 [  1   4 136   0]
 [  0   1   0 152]]
             precision    recall  f1-score   support

          0       0.99      0.96      0.98       128
          1       0.96      0.96      0.96       121
          2       0.96      0.96      0.96       141
          3       0.97      0.99      0.98       153

avg / total       0.97      0.97      0.97       543
```

4.2 Implementation using Analysis of Variance

A statistical feature selection method that helps to complete the job of choosing the best features. It Analysis of variance famously known as ANOVA performs F-tet check to find if any significant diversities are there between the groups. The outcome of ANOVA's F-ratio will be near to 1 if there are no significant diversities between the groups than that means all the variance are equal.

It is recommended to rerun all the steps from loading the dataset till doing label encoding on the dataset and dividing it into train and test.

4.2.1 Gaussian Naive Bayes

ANOVA code for selecting the best number of features for Gaussian Naïve Bayes.

The "for loop" was executed, and it is found that by considering 420 features Gaussian Naïve Bayes algorithm is achieving the best accuracy.

```
# Feature Selection using Analysis of Variance (ANOVA)
#features = range(10, 810, 10)
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import f_classif
#for i in list(reversed(features)):
#print("###  Current Features {}".format(i))
fvalue_selector = SelectKBest(f_classif, k= 420)
X_kbest = fvalue_selector.fit_transform(X, y)

# Splitting the dataset into the Training set and Test set

X_anova_train, X_anova_test, y_anova_train, y_anova_test = train_test_split(X_kbest, y, test_size = 0.20, random_state = 10)
```

Machine learning code for Gaussian Naïve Bayes with accuracy, k10 fold cross-validation accuracy, confusion matrix, and evaluation metrics[6].

[6]https://www.datacamp.com/community/tutorials/naive-bayes-scikit-learn

```
# Gaussian Naive Bayes

from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
gaussian = GaussianNB()
gaussian.fit(X_anova_train, y_anova_train)
# Predicting the Test set results
y_pred = gaussian.predict(X_anova_test)
#10-fold cross validation score
accuracy = cross_val_score(estimator = gaussian, X = X_anova_train, y = y_anova_train, cv =10)
across_cv = accuracy.mean()
#Accuracy
accuracy = accuracy_score(y_anova_test, y_pred)
print("Guassian NB")
print("-- {}".format(accuracy))
print("-- {}".format(across_cv))
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
Guassian NB
-- 0.8895027624309392
-- 0.8823201639204035
[[109   0  19   0]
 [  0 102   4  15]
 [  3  11 127   0]
 [  0   8   0 145]]
             precision    recall  f1-score   support

          0       0.97      0.85      0.91       128
          1       0.84      0.84      0.84       121
          2       0.85      0.90      0.87       141
          3       0.91      0.95      0.93       153

avg / total       0.89      0.89      0.89       543
```

4.2.2 Random Forest Classifier

Before executing the "for loop," it is necessary to execute the label encoding step as it will encode the dataset so that it can be fed to the classification machine learning algorithms.

ANOVA code for selecting the best number of features for Random Forest.

Post-implementation of "for loop," it is found that the Random forest model is achieving the best accuracy by considering 350 features. That is why the value of k is equal to 350.

```
# Feature Selection using Analysis of Variance (ANOVA)
#features = range( 10, 810, 10)
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import f_classif
#for i in list(reversed(features)):
    #print("###  Current Features {}".format(i))
fvalue_selector = SelectKBest(f_classif, k=350)
X_kbest = fvalue_selector.fit_transform(X, y)

# Splitting the dataset into the Training set and Test set
X_anova_train, X_anova_test, y_anova_train, y_anova_test = train_test_split(X_kbest, y, test_size = 0.20, random_state = 10)
```

Machine learning code for Random Forest Classifier with all four evaluation metrics namely accuracy, f1score, precision, and recall[7]. The accuracy of the model is evaluated using a k10 fold cross-validation accuracy and confusion matrix.

[7]https://towardsdatascience.com/an-implementation-and-explanation-of-the-random-forest-in-python-77bf308a9b76

```
# Random Forest
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
randomf = RandomForestClassifier()
randomf.fit(X_anova_train, y_anova_train)
# Predicting the Test set results
y_pred = randomf.predict(X_anova_test)
#10-fold cross validation score
accuracy = cross_val_score(estimator = randomf, X = X_anova_train, y = y_anova_train, cv =10)
randomf_cross = accuracy.mean()
#Accuracy
accuracy = accuracy_score(y_anova_test, y_pred)
print("Random Forest")
print("-- {}".format(accuracy))
print("-- {}".format(randomf_cross))
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
Random Forest
-- 0.9521178637200737
-- 0.9358827387056463
[[123   0   5   0]
 [  0 121   0   0]
 [  4  11 126   0]
 [  0   6   0 147]]
             precision    recall  f1-score   support

          0       0.97      0.96      0.96       128
          1       0.88      1.00      0.93       121
          2       0.96      0.89      0.93       141
          3       1.00      0.96      0.98       153

avg / total       0.96      0.95      0.95       543
```

4.2.3 Logistic Regression

Before executing the "for loop," it is necessary to execute the label encoding step as it will encode the dataset so that it can be fed to the classification machine learning algorithms.

ANOVA code for selecting the best number of features for Logistic Regression.

The "for loop" was executed, and it is found that by considering 270 features Logistic Regression algorithm is achieving the best accuracy.

```
# Feature Selection using Analysis of Variance (ANOVA)
#features = range(50, 810, 10)
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import f_classif
#for i in list(reversed(features)):
    #print("###  Current Features {}".format(i))
fvalue_selector = SelectKBest(f_classif, k=270)
X_kbest = fvalue_selector.fit_transform(X, y)
# Splitting the dataset into the Training set and Test set
X_anova_train, X_anova_test, y_anova_train, y_anova_test = train_test_split(X_kbest, y, test_size = 0.20, random_state = 10)
```

Machine learning code for Logistic Regression with all four evaluation metrics, namely accuracy, f1score, precision, and recall[8]. The accuracy of the model is evaluated using a k10 fold cross-validation accuracy and confusion matrix. As per the result, the Logistic Regression model is achieving an accuracy of 79.74%, f1 score, precision, recall accuracy of 79%,80%, and 80% respectively, and a ten-fold cross-validation accuracy of 79.75%

[8]https://towardsdatascience.com/logistic-regression-python-7c451928efee

```
  #Logistic Regression
from sklearn.linear_model import LogisticRegression
lregression = LogisticRegression()
lregression.fit(X_anova_train, y_anova_train)
y_pred = lregression.predict(X_anova_test)
#10-fold cross validation score
accuracy = cross_val_score(estimator = lregression, X = X_anova_train, y = y_anova_train, cv =10)
lregression_cross = accuracy.mean()
print("10-Fold Cross  Validation Score of LogisticRegression", lregression_cross)
accuracy = accuracy_score(y_anova_test, y_pred)
print("Logistics Regression")
print("-- {}".format(accuracy))
print("-- {}".format(lregression_cross))
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
10-Fold Cross  Validation Score of LogisticRegression 0.7975927631973524
Logistics Regression
-- 0.7974217311233885
-- 0.7975927631973524
[[120   0   8   0]
 [  0  85  20  16]
 [ 35  11  95   0]
 [  0  16   4 133]]
             precision    recall  f1-score   support

          0       0.77      0.94      0.85       128
          1       0.76      0.70      0.73       121
          2       0.75      0.67      0.71       141
          3       0.89      0.87      0.88       153

avg / total       0.80      0.80      0.79       543
```

4.2.4 K-Nearest Neighbor

Before executing the "for loop," it is necessary to execute the label encoding step as it will encode the dataset so that it can be fed to the classification machine learning algorithms.

ANOVA code for selecting the best number of features for K-Nearest Neighbor.

Post-implementation of "for loop," it is found that the K-Nearest Neighbor model is achieving the best accuracy by considering 430 features.

```
#features = range(400, 460, 10)
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import f_classif
#for i in list(reversed(features)):
    #print("###  Current Features {}".format(i))
fvalue_selector = SelectKBest(f_classif, k=430)
X_kbest = fvalue_selector.fit_transform(X, y)
# Splitting the dataset into the Training set and Test set
X_anova_train, X_anova_test, y_anova_train, y_anova_test = train_test_split(X_kbest, y, test_size = 0.20, random_state = 10)
```

Machine learning code for K-Nearest Neighbor with all four evaluation metrics, namely accuracy, f1score, precision, and recall[9]. The accuracy of the model is evaluated using a k10 fold cross-validation accuracy and confusion matrix. As per the result, the K-Nearest Neighbor model is achieving an accuracy of 97.05%, f1 score, precision, recall accuracy of 97%, and a ten-fold cross-validation accuracy of 97.41%

[9]https://stackabuse.com/k-nearest-neighbors-algorithm-in-python-and-scikit-learn/

```
# KNN
from sklearn.neighbors import KNeighborsClassifier
KNC = KNeighborsClassifier(n_neighbors=5)
KNC.fit(X_anova_train, y_anova_train)
y_pred = KNC.predict(X_anova_test)
#10-fold cross validation score
accuracy = cross_val_score(estimator = KNC, X = X_anova_train, y = y_anova_train, cv =10)
KNC_cross = accuracy.mean()
print("10-Fold Cross  Validation Score of KNC", KNC_cross)
from sklearn.metrics import accuracy_score
accuracy = accuracy_score(y_anova_test, y_pred)
print(accuracy)
print("KNN")
print("-- {}".format(accuracy))
print("-- {}".format(KNC_cross))
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
10-Fold Cross  Validation Score of KNC 0.9741392851489412
0.9705340699815838
KNN
-- 0.9705340699815838
-- 0.9741392851489412
[[123   0   5   0]
 [  0 116   0   5]
 [  1   4 136   0]
 [  0   1   0 152]]
             precision    recall  f1-score   support

          0       0.99      0.96      0.98       128
          1       0.96      0.96      0.96       121
          2       0.96      0.96      0.96       141
          3       0.97      0.99      0.98       153

avg / total       0.97      0.97      0.97       543
```

References

Platon, R., Dehkordi, V. R. and Martel, J. (2015). Hourly prediction of a building's electricity consumption using case-based reasoning, artificial neural networks and principal component analysis, *Energy and Buildings* **92**: 10–18.
URL: *http://dx.doi.org/10.1016/j.enbuild.2015.01.047*

Robinson, C., Dilkina, B., Hubbs, J., Zhang, W., Guhathakurta, S., Brown, M. A. and Pendyala, R. M. (2017). Machine learning approaches for estimating commercial building energy consumption, *Applied Energy* **208**(September): 889–904.