Int247-Machine Learning Foundations
Online Assignment 2
Project-Computer Hardware

Name- Mahankali.Prudhviraj
Reg.No.-11809816
Roll No.- A20

Submitted to-Dr.Aditya Khamparia

Abstract

Computer hardware includes the physical parts of a computer, such as the case, central processing unit (CPU), monitor, mouse, keyboard, computer data storage, graphics card, sound card, speakers and motherboard.

By contrast, software is the set of instructions that can be stored and run by hardware. Hardware is so-termed because it is "hard" or rigid with respect to changes, whereas software is "soft" because it is easy to change.

Hardware is typically directed by the software to execute any command or instruction. A combination of hardware and software forms a usable computing system, although other systems exist with only hardware.

INTRODUCTION

In this There was a time when I was a student when I was obsessed with more speed and more cores so I could run my algorithms faster and for longer. I have changed my perspective. Big hardware still matters, but only after you have considered a bunch of other factors. if you are just starting out, you're hardware doesn't matter. Focus on learning with small datasets that fit in memory, such as those from the UCI Machine Learning Repository. Learn good experimental design and make sure you ask the right questions and challenge your intuitions by testing diverse algorithms and interpreting your results through the lens of statistical hypothesis testing.Once hardware does start to matter and you really need lots of cores and a whole lot of RAM, rent it just-in-time for your carefully designed project or experiment.

In this computer hardware project first of all Iam doing the preparing data code,that was the feature Identification.after that data reiterating for optimization means training of data. and Iam using 3 models for this code. And also implement data file in this code.data file name is

hardware.csv and I run that code. It shows the output of all csv file data(like vendor,chmin,, chmax,binary search etc.) and also iam monted the drive file also,when implemented the drive file it shows one chrome link and in that link we have code, then I copied that code and enter output of drive file then the drive file is running in the code.in this code. When implement the csv file and iam read that file so, iam using df2 classifier. And it shows Boolean table form of that csv file. and in that file iam separate the creating files of that csv file.and plot the graph of that csv file with using attributes. finally iam using 3 models and doing confusion matrix for all three models and prediction also.

Implementation of models

iam using 3 models in this code so, the three models are:

model1: nonlinear SVM- Non-linear transformation is to make a dataset higher-dimensional space (Mapping a higher dimension). And it is also the fundamental of a non-linear system. The below graph reveals a non-linear dataset and how it can not be used Linear kernel rather than the Gaussian kernel.

Classifying a non-linearly separable dataset using a SVM is As mentioned above SVM is a linear classifier which learns an (n – 1)-dimensional classifier for classification of data into two classes. However, it can be used for classifying a non-linear dataset. This can be done by projecting the dataset into a higher dimension in which it is linearly separable

Model2: linear kernel- Let's create a Linear Kernel SVM using the sklearn library of Python and the Iris Dataset that can be found in the dataset library of Python.
Linear Kernel is used when the data is Linearly separable, that is, it can be separated using a single Line. It is one of the most common kernels to be used. It is mostly used when there are a Large number of Features in a particular Data Set. One of the examples where there are a lot of features, is Text Classification, as each alphabet is a new feature. So we mostly use Linear Kernel in Text Classification.

Model3: neural network- Neural networks are one approach to machine learning, which is one application of AI. Machine learning algorithms are able to improve without being explicitly programmed. In other words, they are able to find patterns in the data and apply those patterns to new challenges in the future.
These three are models of my project. ,in this model1&2 are under svm models.neural network is also model but seperable of svm model.and iam doing confusion matrix for all these models.
And iam doing the hyperparameter tuning of all these models

Hyperparametertuning: hyperparameter optimization or tuning is the problem of choosing a set of optimal hyperparameters for a learning algorithm. A hyperparameter is a parameter whose value is used to control the learning process. By contrast, the values of other parameters (typically node weights) are learned. The same kind of machine learning model can require different constraints, weights or learning rates to generalize different data patterns.

Iam doing confusion matrics and prediction for all data in this code.

Confusion matrix: A confusion matrix is a technique for summarizing the performance of an classification accuracy alone can be misleading if you have an unequal number of observations in each class or if you have more than two classes in your dataset.

Prediction: Prediction refers to the output of an algorithm after it has been trained on a historical dataset and applied to new data when forecasting the likelihood of a particular outcome, such as whether or not a customer will churn in 30 days.

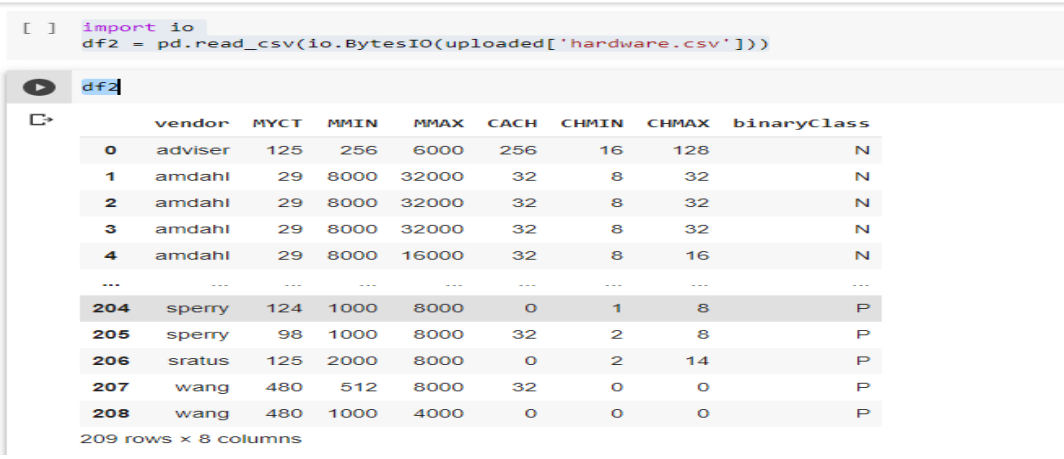
And finally iam doing the prediction code for all the data file,sctter plotting,and confusion matrices also. Just like a hypothesis, a prediction is a type of guess. However, a prediction is an estimation made from observations. For example, you observe that every time the wind blows, flower petals fall from the tree. Therefore, you could predict that if the wind blows, petals will fall from the tree.

Result and discussion

Step-1: This is the program which i have implemented in the program importing all the important libraries like numpy, pandas, seaborn.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

Step-2: After that i am read.csv file and i will get output these are data values which consists of 209 rows and 8 columns.



```
[ ]  import io
     df2 = pd.read_csv(io.BytesIO(uploaded['hardware.csv']))
```

```
df2
```

	vendor	MYCT	MMIN	MMAX	CACH	CHMIN	CHMAX	binaryClass
0	adviser	125	256	6000	256	16	128	N
1	amdahl	29	8000	32000	32	8	32	N
2	amdahl	29	8000	32000	32	8	32	N
3	amdahl	29	8000	32000	32	8	32	N
4	amdahl	29	8000	16000	32	8	16	N
...	...	...	...	...	...	...	...	...
204	sperry	124	1000	8000	0	1	8	P
205	sperry	98	1000	8000	32	2	8	P
206	sratus	125	2000	8000	0	2	14	P
207	wang	480	512	8000	32	0	0	P
208	wang	480	1000	4000	0	0	0	P

209 rows × 8 columns

Step-3: After that this data has interpting the values as the null values which as the null value are not.

```
[ ]  #creating files

     sha=df2.shape

[ ]  df2.isnull().sum()
```

```
vendor          0
MYCT            0
MMIN            0
MMAX            0
CACH            0
CHMIN           0
CHMAX           0
binaryClass     0
dtype: int64
```

Step-4: After that i am describing the dataset like count ,mean of the rows, std, min, 25percent, 50 percent, 75percent max using the describe() (describe function).

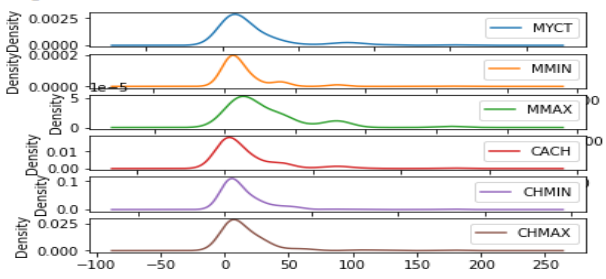
```
[ ]  df2.describe()
```

	MYCT	MMIN	MMAX	CACH	CHMIN	CHMAX
count	209.000000	209.000000	209.000000	209.000000	209.000000	209.000000
mean	203.822967	2867.980861	11796.153110	25.205742	4.698565	18.267943
std	260.262926	3878.742758	11726.564377	40.628722	6.816274	25.997318
min	17.000000	64.000000	64.000000	0.000000	0.000000	0.000000
25%	50.000000	768.000000	4000.000000	0.000000	1.000000	5.000000
50%	110.000000	2000.000000	8000.000000	8.000000	2.000000	8.000000
75%	225.000000	4000.000000	16000.000000	32.000000	6.000000	24.000000
max	1500.000000	32000.000000	64000.000000	256.000000	52.000000	176.000000

Step-5: After that here we are plotting the graph like density graph values ranging and how the values are varying from one row to another row which as the highest density values as 5 something.

```
[ ]  si=df2.groupby('binaryClass').size()

     plt.figure(figsize=(15,10))
     data1=pd.DataFrame(df2[['vendor','MYCT','MMIN','MMAX','CACH','CHMIN','CHMAX','binaryClass',]])
     data1.plot(kind='density',subplots=True,sharex=False)
     plt.show()
```

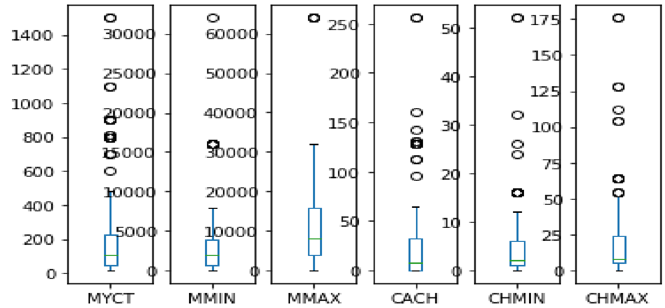
```
<Figure size 1080x720 with 0 Axes>
```



Step-6: After that i have done the box plotting which represent by the if there is out layers are not it shows that 0 to 1400 represents the mean value like middle value 6 shoes the mean value and there is one out Layer in the V5 two out layer are more than we are going to remove in this graph there is output layer.

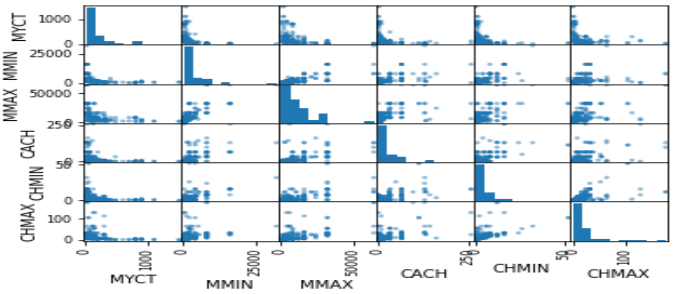
```
#BOX PLOTTING: SHOWS THE MEAN VALUE AND OUTLIERS OUTSIDE THE BOX
data1.plot(kind='box',subplots=True,sharex=False,sharey=False)
```

```
MYCT        AxesSubplot(0.125,0.125;0.110714x0.755)
MMIN     AxesSubplot(0.257857,0.125;0.110714x0.755)
MMAX     AxesSubplot(0.390714,0.125;0.110714x0.755)
CACH     AxesSubplot(0.523571,0.125;0.110714x0.755)
CHMIN    AxesSubplot(0.656429,0.125;0.110714x0.755)
CHMAX    AxesSubplot(0.789286,0.125;0.110714x0.755)
dtype: object
```



Step-7: After that we have to plot the scatter plotting which shows that how the values are one values to another values are scattering.

```
pd.plotting.scatter_matrix(data1)
plt.show()
```



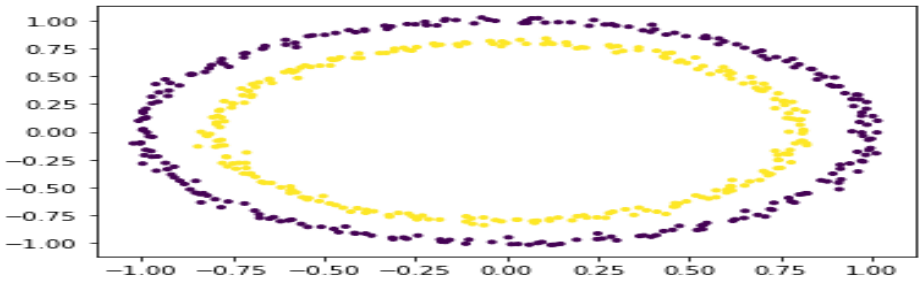
Step8:after that iam doing data taking for identification of data using csv file.

```
#data taking
data=pd.read_csv('hardware.csv')
print(data.head())
y=data['binaryClass'].values
print(y)
x=data.drop(['binaryClass'],axis=1).values
y=np.where(y=='stable',1,-1)
print(y)
```

```
     vendor  MYCT  MMIN   MMAX  CACH  CHMIN  CHMAX binaryClass
0  adviser   125   256   6000   256     16    128           N
1   amdahl    29  8000  32000    32      8     32           N
2   amdahl    29  8000  32000    32      8     32           N
3   amdahl    29  8000  32000    32      8     32           N
4   amdahl    29  8000  16000    32      8     16           N
['N' 'N' 'N' 'N' 'N' 'N' 'N' 'N' 'N' 'N' 'P' 'P' 'P' 'N' 'P' 'P' 'P' 'P'
 'P' 'N' 'P' 'P' 'P' 'P' 'P' 'P' 'P' 'P' 'P' 'P' 'N' 'N' 'P' 'P' 'N' 'N'
 'P' 'P' 'P' 'P' 'P' 'P' 'P' 'P' 'P' 'P' 'P' 'P' 'P' 'P' 'P' 'P' 'P' 'P'
 'P' 'P' 'P' 'P' 'P' 'P' 'P' 'P' 'P' 'P' 'N' 'N' 'P' 'P' 'P' 'P' 'P' 'P'
 'P' 'P' 'P' 'P' 'P' 'P' 'P' 'P' 'P' 'P' 'P' 'P' 'P' 'P' 'P' 'P' 'N' 'P'
 'N' 'N' 'P' 'P' 'N' 'N' 'N' 'N' 'N' 'P' 'P' 'P' 'P' 'P' 'P' 'P' 'P' 'P'
 'P' 'P' 'P' 'P' 'P' 'P' 'P' 'P' 'P' 'P' 'P' 'P' 'N' 'N' 'P' 'P' 'P' 'P'
 'P' 'P' 'P' 'P' 'N' 'P' 'P' 'P' 'P' 'P' 'P' 'P' 'P' 'P' 'P' 'P' 'P' 'P'
 'P' 'N' 'N' 'N' 'N' 'P' 'N' 'N' 'N' 'N' 'N' 'N' 'N' 'P' 'P' 'P' 'P' 'P'
 'P' 'P' 'P' 'P' 'P' 'N' 'N' 'N' 'P' 'P' 'P' 'P' 'P' 'P' 'P' 'P' 'P' 'P'
 'P' 'P' 'P' 'P' 'P' 'P' 'P' 'P' 'P' 'N' 'N' 'N' 'N' 'P' 'N' 'N' 'N' 'N'
 'N' 'N' 'P' 'P' 'P' 'P' 'P' 'P' 'P' 'P' 'P']
[-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
```

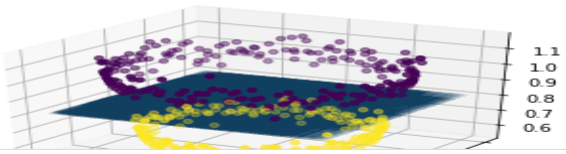
Step9:after that iam implementing the models so,this is non linear svm ,in this iam using matplotlib,and generating the data and vistualizing the data.

```
[ ]  # importing models of non linear svm
     import numpy as np
     import matplotlib.pyplot as plt
     from sklearn.datasets import make_circles
     from mpl_toolkits.mplot3d import Axes3D

     # generating data
     X, Y = make_circles(n_samples = 500, noise = 0.02)

     # visualizing data
     plt.scatter(X[:, 0], X[:, 1], c = Y, marker = '.')
     plt.show()
```



Step10:and this step is the linear kernel model,in this model iam giving values hyperplane,then this code get out of projection.iam giving projection values also,and this is also svm model.

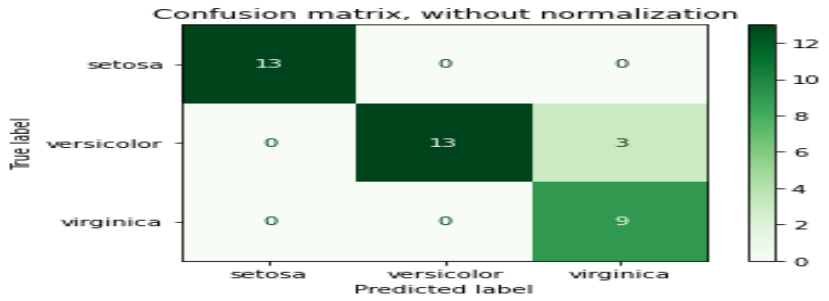
```
#importing models of linear kernal
# create support vector classifier using a linear kernel
from sklearn import svm

svc = svm.SVC(kernel = 'linear')
svc.fit(X, Y)
w = svc.coef_
b = svc.intercept_

# plotting the separating hyperplane
x1 = X[:, 0].reshape((-1, 1))
x2 = X[:, 1].reshape((-1, 1))
x1, x2 = np.meshgrid(x1, x2)
x3 = -(w[0][0]*x1 + w[0][1]*x2 + b) / w[0][2]

fig = plt.figure()
axes2 = fig.add_subplot(111, projection = '3d')
axes2.scatter(X1, X2, X1**2 + X2**2, c = Y, depthshade = True)
axes1 = fig.gca(projection = '3d')
axes1.plot_surface(x1, x2, x3, alpha = 0.01)
plt.show()
```

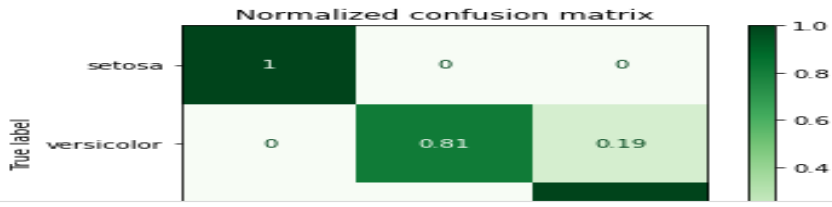


Step11:this step is for confusion matrix for svm models of minear kernel and non linear svm,in this iam using matplotlib and iris model for giving output. The output is in matrix model.finally,the confusion matrix for svm is ready. And that matrix gives normalized and without normalization values of matrix.

```
#confusion matrix for SVM
import numpy as np
import matplotlib.pyplot as plt

from sklearn import svm, datasets
from sklearn.model_selection import train_test_split
from sklearn.metrics import plot_confusion_matrix

# import some data to play with
iris = datasets.load_iris()
X = iris.data
y = iris.target
class_names = iris.target_names

# Split the data into a training set and a test set
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0)

# Run classifier, using a model that is too regularized (C too low) to see
# the impact on the results
classifier = svm.SVC(kernel='linear', C=0.02).fit(X_train, y_train)

np.set_printoptions(precision=4)

# Plot non-normalized confusion matrix
titles_options = [("Confusion matrix, without normalization", None),
                  ("Normalized confusion matrix", 'true')]
for title, normalize in titles_options:
    disp = plot_confusion_matrix(classifier, X_test, y_test,
                                 display_labels=class_names,
                                 cmap=plt.cm.Greens,
                                 normalize=normalize)
```

```
Confusion matrix, without normalization
[[13  0  0]
 [ 0 13  3]
 [ 0  0  9]]
Normalized confusion matrix
[[1.     0.     0.    ]
 [0.     0.8125 0.1875]
 [0.     0.     1.    ]]
```





Step12: this is model of neural network and also for confusion matrix of neural network.

```
#importing models of neural network
import numpy as np

# array of any amount of numbers. n = m
X = np.array([[1, 2, 3],
      [3, 4, 1],
      [2, 5, 3]])

# multiplication
y = np.array([[.5, .3, .2]])

# transpose of y
y = y.T

# sigma value
sigm = 2

# find the delta
delt = np.random.random((3, 3)) - 1

for j in range(100):
  # find confusion matrix 1. 100 layers.
  m1 = (y - (1/(1 + np.exp(-(np.dot((1/(1 + np.exp(
        -(np.dot(X, sigm))))), delt))))))*((1/(
          1 + np.exp(-(np.dot((1/(1 + np.exp(
        -(np.dot(X, sigm))))), delt)))))*(1-(1/(
          1 + np.exp(-(np.dot((1/(1 + np.exp(
        -(np.dot(X, sigm))))), delt)))))))

  # find confusion matrix 2
  m2 = m1.dot(delt.T) * ((1/(1 + np.exp(-(np.dot(X, sigm)))))
```

```
  # find confusion matrix 2
  m2 = m1.dot(delt.T) * ((1/(1 + np.exp(-(np.dot(X, sigm)))))
          * (1-(1/(1 + np.exp(-(np.dot(X, sigm)))))))
  # find delta
  delt = delt + (1/(1 + np.exp(-(np.dot(X, sigm))))).T.dot(m1)

  # find sigma
  sigm = sigm + (X.T.dot(m2))

# print output from the matrix
print(1/(1 + np.exp(-(np.dot(X, sigm)))))
```

```
[[0.99999329 0.99999375 0.99999385]
 [0.99999988 0.99999989 0.99999989]
 [1.         1.         1.        ]]
```

Step13:this step is for prediction of all data and for confusion matrix also.

```
# Importing the dataset.
from sklearn.datasets import fetch_openml
mnist = fetch_openml('mnist_784', version=1)

# Creating independent and dependent variables.
X, y = mnist['data'], mnist['target']

# Splitting the data into training set and test set.
X_train, X_test, y_train, y_test = X[:60000], X[60000:], y[:60000], y[60000:]

"""
The training set is already shuffled for us, which is good as this guarantees that all
cross-validation folds will be similar.
"""

# Training a binary classifier.
y_train_5 = (y_train == 5) # True for all 5s, False for all other digits.
y_test_5 = (y_test == 5)

"""
Building a dumb classifier that just classifies every single image in the “not-5” class.
"""

from sklearn.model_selection import cross_val_score
from sklearn.base import BaseEstimator
class Never5Classifier(BaseEstimator):
  def fit(self, X, y=None):
    pass
  def predict(self, X):
    return np.zeros((len(X), 1), dtype=bool)
```

```
array([1., 1., 1.])
```

```
#prediction
# Importing the dependancies
from sklearn import metrics
# Predicted values
y_pred = ["a", "b", "c", "a", "b"]
# Actual values
y_act = ["a", "b", "c", "c", "a"]
# Printing the confusion matrix
# The columns will show the instances predicted for each label,
# and the rows will show the actual number of instances for each label.
print(metrics.confusion_matrix(y_act, y_pred, labels=["a", "b", "c"]))
# Printing the precision and recall, among other metrics
print(metrics.classification_report(y_act, y_pred, labels=["a",
"b","c"]))
```

```
[[1 1 0]
 [0 1 0]
 [1 0 1]]
              precision    recall  f1-score   support

           a       0.50      0.50      0.50         2
           b       0.50      1.00      0.67         1
           c       1.00      0.50      0.67         2

    accuracy                           0.60         5
   macro avg       0.67      0.67      0.61         5
weighted avg       0.70      0.60      0.60         5
```

Conclusion

The Scope of this project is to apply machine learning concepts taught in class on our data set. We made use of open dataset. We are able to apply several classification machine learning methods on the computer hardware data set to achieve the goal of the project.
The goal of the project is to model the data, so as to accurately predict the stability and binery search for computer hardware. Also, infer the relationship between predictors and the response. We achieved these goals by using several classification methods and in this project we have

download data sets in the UCI and we have used the data sets in the form of .csv file in this. We have applied the concepts of the machine learning and we have learned that the computer hardware is simple hardware contains 8 Boolean attributes and 10 classes, the set of decimal digits. Recall that computer hardware contain 7 hardware numbers. Hence the reason for 8 attributes. The class attribute is an integer ranging between 0 to 9 inclusive , representing the possible digits show on the code. The problem would be easy if not for the introduction of noise. In this In this case, each attribute value has the 10% probability of having its value inverted.

References

UCI machine learning repositary:
https://archive.ics.uci.edu/ml/datasets/Computer+Hardware

https://www.geeksforgeeks.org/hardware

https://www.openml.org/d/40496

Project Colab link

Colab file name:Mahankali prudhviraj 11809816 A20

Link:https://colab.research.google.com/drive/1XF31vW7I4aDuj1CAHb7S16JkWHsUo8Ug?usp=sharing