**MACHINE LEARNING ASSIGNMENT**

Classification for Mushrooms

SE4060

Bachelor of Science (Honours) in Information Technology

Department of Software Engineering

Sri Lanka Institute of Information Technology

 May 2019

List of Figures

Figure 2.2.1 specify features 6

Figure 3.1.1 Neural Network Model 7

Figure 4.1.1 Covert into a numerical array 9

Figure 4.1.2 convert train and test data sets into 9

NumPy arrays

Figure 4.2.1 Data splitting 10

Feature 4.3.1 Model creation 11

Feature 4.3.2 Model Summary 11

Feature 4.3.4 Model fitting 12

Figure 4.3.5 RMSprop optimizer 12

Figure 4.3.6 Adagrad optimizer 13

Figure 4.3.7 Gradient Descent Optimizer 13

Table of Content

1. Introduction 4

2. Used dataset 4

2.1 Description of dataset 4

2.2 Features and Labels 5

3. Methodology 7

3.1 Feed Forward Neural Network 7

4. Implementation 8

4.1 Data Processing 8

4.2 Data Splitting 10

4.3 Model Creating 10

5. Discussion 13

6. Appendix 14

# 1. INTRODUCTION

Although this dataset was originally contributed to the UCI Machine Learning repository nearly 30 years ago, mushroom hunting is enjoying new peaks in popularity. The dataset helps to learn which features spell certain death and which are most palatable in this dataset of mushroom characteristics. The purpose of this documentation is to come up with the most accurate model to classify the mushroom verities. Hence, this documentation contains a classification problem to classify the mushrooms whether they are diet able or poisons. here I will be using Feed Forward Neural Network to classify them into given two categories. At the end of the document the accuracy level and the error has been plated in terms of different optimizations.

# 2. USED DATASET

2.1 Description of dataset

This dataset includes descriptions of hypothetical samples corresponding to 23 species of gilled mushrooms in the Agaricus and Lepiota Family Mushroom drawn from The Audubon Society Field Guide to North American Mushrooms (1981). Each species is identified as definitely edible, definitely poisonous, or of unknown edibility and not recommended. This latter class was combined with the poisonous one. The following link is for the hosted environment of the dataset. The data set consists with 8124 samples.

<https://www.kaggle.com/uciml/mushroom-classification>

2.2 Features & Labels

There are 22 features has been taken into consideration.

* cap-shape: bell=b,conical=c,convex=x,flat=f, knobbed=k,sunken=s
* cap-surface: fibrous=f,grooves=g,scaly=y,smooth=s
* cap-color: brown=n,buff=b,cinnamon=c,gray=g,green=r,pink=p,purple=u,red=e,white=w,yellow=y
* bruises: bruises=t,no=f
* odor: almond=a,anise=l,creosote=c,fishy=y,foul=f,musty=m,none=n,pungent=p,spicy=s
* gill-attachment: attached=a,descending=d,free=f,notched=n
* gill-spacing: close=c,crowded=w,distant=d
* gill-size: broad=b,narrow=n
* gill-color: black=k,brown=n,buff=b,chocolate=h,gray=g, green=r,orange=o,pink=p,purple=u,red=e,white=w,yellow=y
* stalk-shape: enlarging=e,tapering=t
* stalk-root: bulbous=b,club=c,cup=u,equal=e,rhizomorphs=z,rooted=r,missing=?
* stalk-surface-above-ring: fibrous=f,scaly=y,silky=k,smooth=s
* stalk-surface-below-ring: fibrous=f,scaly=y,silky=k,smooth=s
* stalk-color-above-ring: brown=n,buff=b,cinnamon=c,gray=g,orange=o,pink=p,red=e,white=w,yellow=y
* stalk-color-below-ring: brown=n,buff=b,cinnamon=c,gray=g,orange=o,pink=p,red=e,white=w,yellow=y
* veil-type: partial=p,universal=u
* veil-color: brown=n,orange=o,white=w,yellow=y
* ring-number: none=n,one=o,two=t
* ring-type: cobwebby=c,evanescent=e,flaring=f,large=l,none=n,pendant=p,sheathing=s,zone=z
* spore-print-color: black=k,brown=n,buff=b,chocolate=h,green=r,orange=o,purple=u,white=w,yellow=y
* population: abundant=a,clustered=c,numerous=n,scattered=s,several=v,solitary=y
* habitat: grasses=g,leaves=l,meadows=m,paths=p,urban=u,waste=w,woods=d

In addition to these 22 features, the labels/classes are

* edible=e
* poisonous=p



Figure 2.2.1 specify features

3. METHODOLOGY

3.1. Feed Forward Neural Network

A feedforward neural network is an artificial neural network wherein connections between the nodes do not form a cycle. As such, it is different from recurrent neural networks. The feedforward neural network was the first and the simplest type of artificial neural network devised. Following is the sketch to my suggested model.



Figure 3.1.1 Neural Network Model

22– neurons for the input layer

11- neurons in the first hidden layer

11- neurons in the second hidden layer

2– neurons for the output layer

Here I have used the sequential neural network model to create my model with three hidden dense layers. I have used two different activation functions (“relu” and “softwamx”) for the hidden layers and “adam” is the optimization technique that I have used while categorical cross entropy is the error loss calculation function. Since there are only 2 labels are available, my model has three output neurons which gives the probabilities as the output. The neuron which gives the highest probability would be taken as the positive class.

4. Implementation

4.1 Data Processing

The original data set has characteristic values but we need to provide numerical values to the feed forward neural network. Therefore, the data set has been looped and came up with an appropriate matrix that can be considered as the input data set. The target values also have been converted into appropriate matrix.

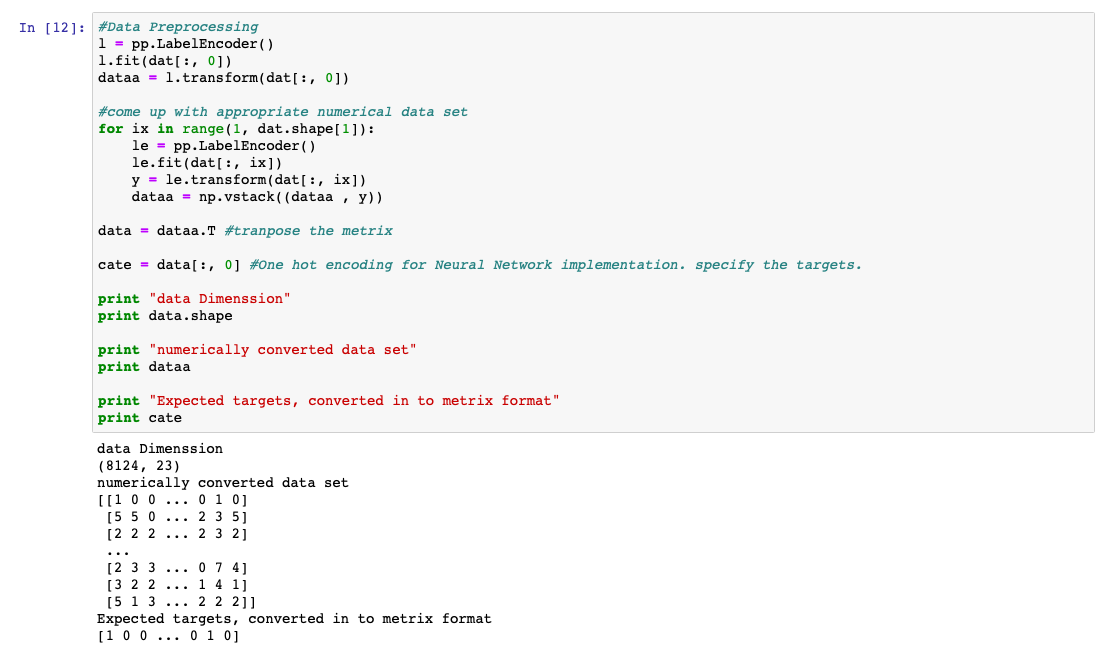


Figure 4.1.1 Covert into a numerical array

After converting them into appropriate array dimensions they have been converted into NumPy arrays.

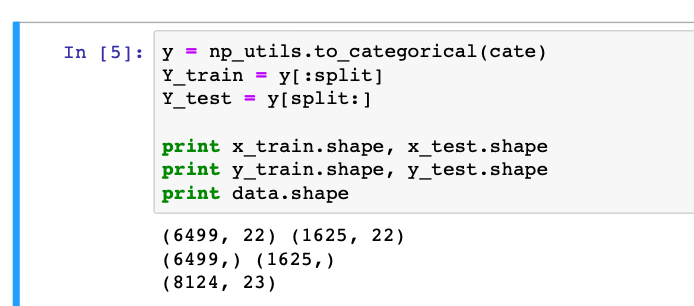


Figure 4.1.2 convert train and test data sets into NumPy arrays

4.2 Data Splitting

The whole data set has been split into two sections which are training data set and testing data set. 80% of the whole data set which would be 6499, was taken to train the neural network. The rest which is 1625 was taken for testing the accuracy of the implemented model.

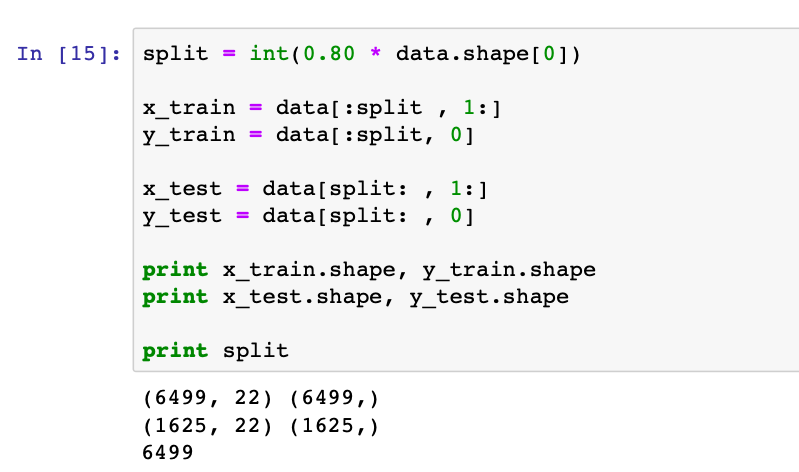
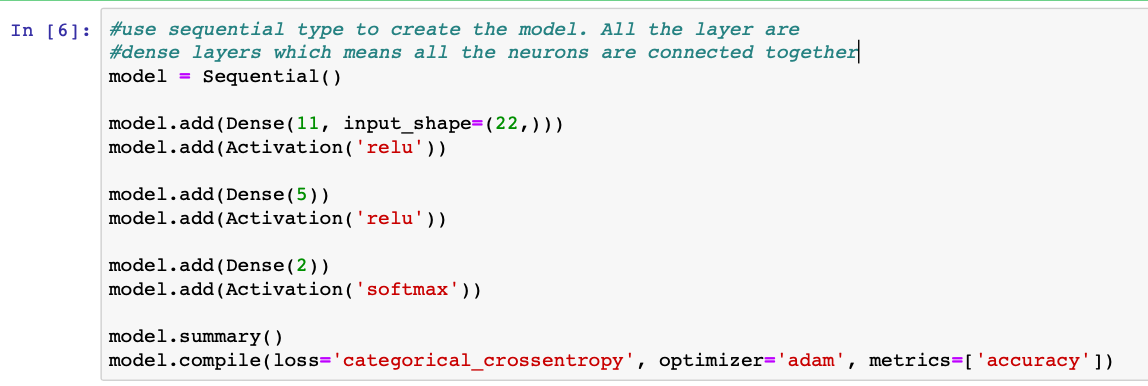


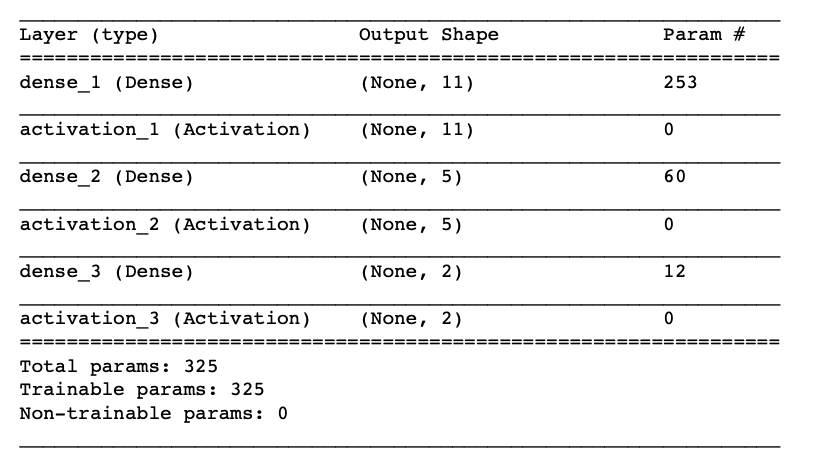
Figure 4.2.1 Data splitting

4.3 Model Creating

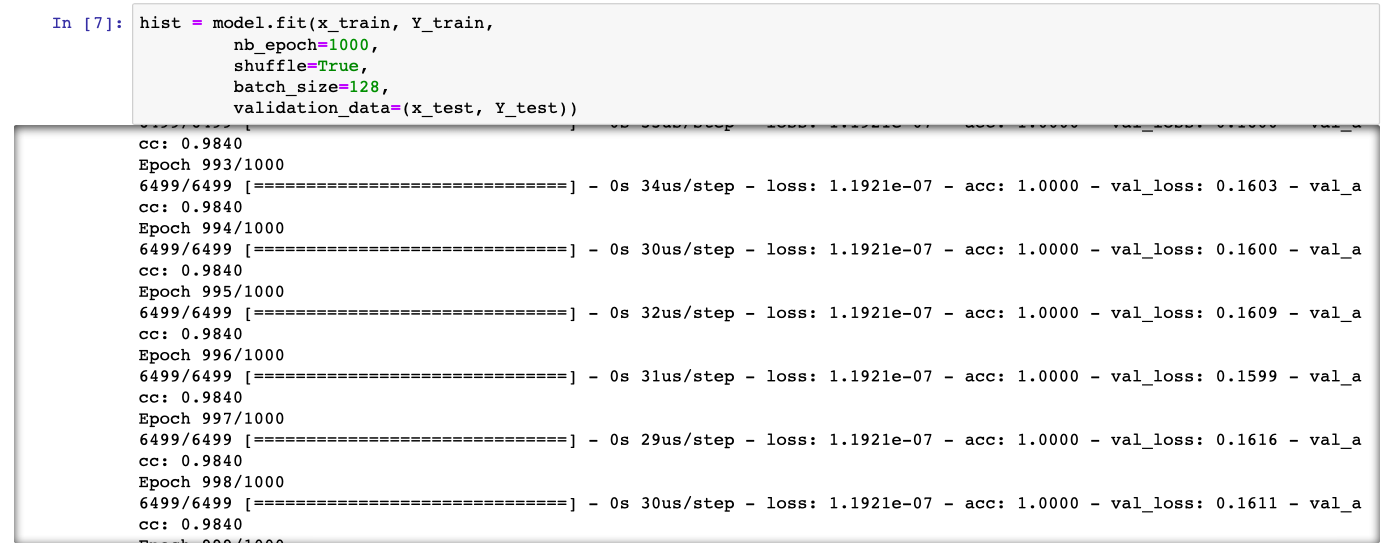
Next, I have come up with the designed neural network model using sequential model provided by Keras models.



Feature 4.3.1 Model creation



Feature 4.3.2 Model Summary



Feature 4.3.4 Model fitting

Here I have used 1000 of epochs where one epoch would be to train the implemented model for all the instances/samples. By doing that I have tried to increase the accuracy level of my model. While training the model I have used. Following are the plot changes with respect to different optimizers in terms of accuracy and error loss.

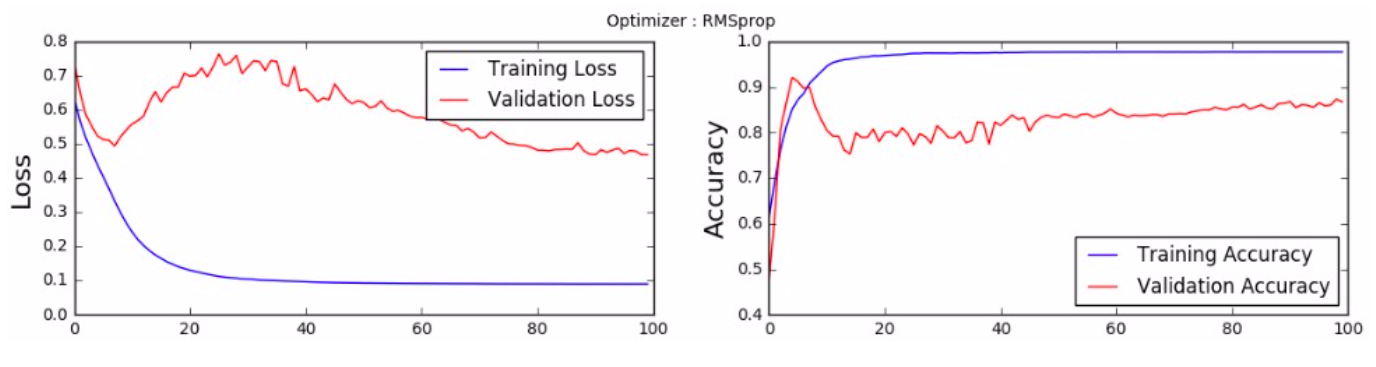


Figure 4.3.5 RMSprop optimizer

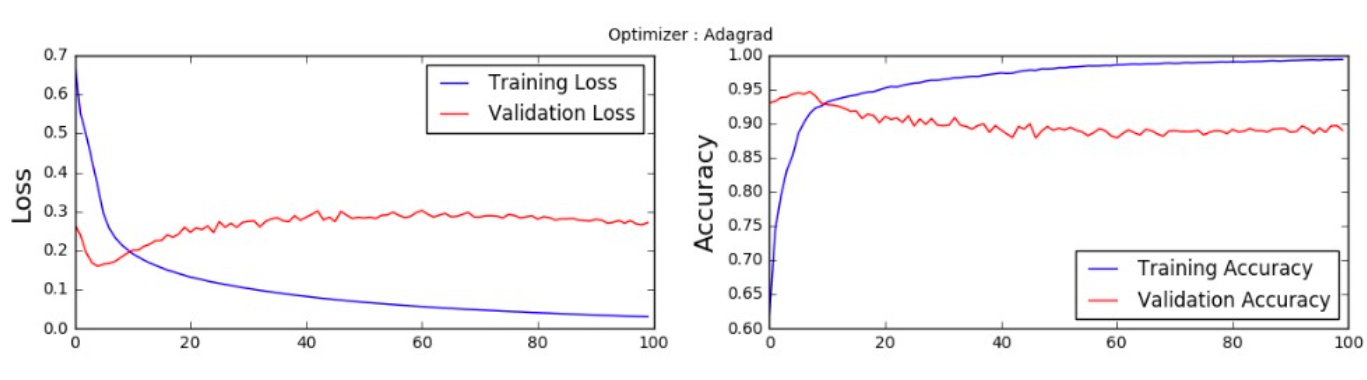


Figure 4.3.6 Adagrad optimizer

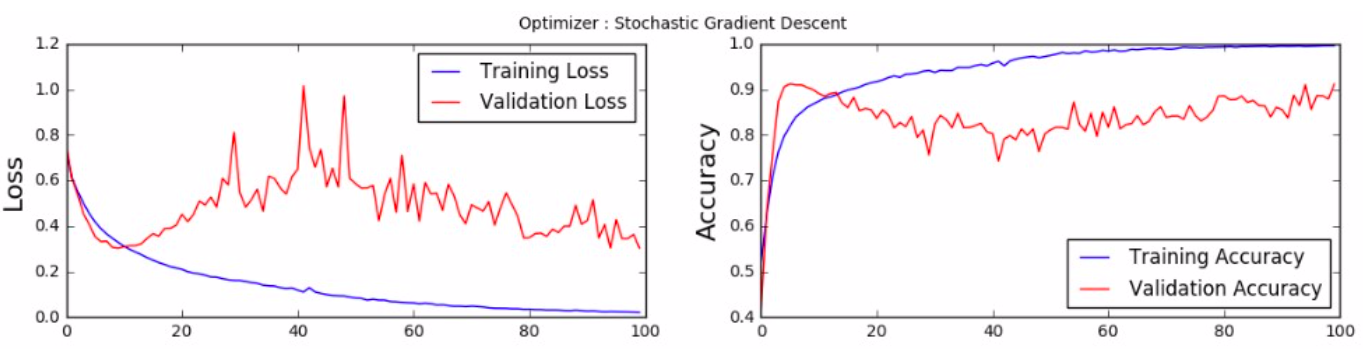


Figure 4.3.7 Gradient Descent Optimizer

5. Discussion

This feed forward neural network has proven approximately 97% accuracy level when classifying given dataset. The accuracy level could be increased by changing the model or by adding more neurons or by changing the number of hidden layers. The data set was gathered by examining 22 features of different types of mushrooms but I would suggest to come up with a simple convolutional neural network to classify the mushrooms via image processing (images of mushrooms) as feature work.

6. Appendix

import numpy as np

from sklearn.ensemble import RandomForestClassifier as RFC

from sklearn.tree import DecisionTreeClassifier as DTC

from matplotlib import pyplot as plt

%matplotlib inline

import datetime

import pandas as pd

from sklearn import preprocessing as pp

from sklearn.linear\_model import LogisticRegression as LR

from sklearn.neighbors import KNeighborsClassifier as KNN

import keras

from keras.models import Sequential

from keras.layers import Dense, Activation

from keras.utils import np\_utils

ds = pd.read\_csv('mushrooms.csv')

dat = ds.values

print dat.shape

headers = list(ds.columns.values) #store features of mushrooms

print(headers)

#Data Preprocessing

l = pp.LabelEncoder()

l.fit(dat[:, 0])

dataa = l.transform(dat[:, 0])

#come up with appropriate numerical data set

for ix in range(1, dat.shape[1]):

le = pp.LabelEncoder()

le.fit(dat[:, ix])

y = le.transform(dat[:, ix])

dataa = np.vstack((dataa , y))

data = dataa.T #tranpose the metrix

cate = data[:, 0] #One hot encoding for Neural Network implementation. specify the targets.

print "data Dimenssion"

print data.shape

print "numerically converted data set"

print dataa

print "Expected targets, converted in to metrix format"

print cate

split = int(0.80 \* data.shape[0])

x\_train = data[:split , 1:]

y\_train = data[:split, 0]

x\_test = data[split: , 1:]

y\_test = data[split: , 0]

print x\_train.shape, y\_train.shape

print x\_test.shape, y\_test.shape

print split

y = np\_utils.to\_categorical(cate)

Y\_train = y[:split]

Y\_test = y[split:]

print x\_train.shape, x\_test.shape

print y\_train.shape, y\_test.shape

print data.shape

#use sequential type to create the model. All the layer are

#dense layers which means all the neurons are connected together

model = Sequential()

model.add(Dense(11, input\_shape=(22,)))

model.add(Activation('relu'))

model.add(Dense(5))

model.add(Activation('relu'))

model.add(Dense(2))

model.add(Activation('softmax'))

model.summary()

model.compile(loss='categorical\_crossentropy', optimizer='adam', metrics=['accuracy'])

hist = model.fit(x\_train, Y\_train,

nb\_epoch=1000,

shuffle=True,

batch\_size=128,

validation\_data=(x\_test, Y\_test))

plt.figure(figsize=(14,3))

plt.subplot(1, 2, 1)

plt.suptitle('Optimizer : Adam', fontsize=10)

plt.ylabel('Loss', fontsize=16)

plt.plot(hist.history['loss'], 'b', label='Training Loss')

plt.plot(hist.history['val\_loss'], 'r', label='Validation Loss')

plt.legend(loc='upper right')

plt.subplot(1, 2, 2)

plt.ylabel('Accuracy', fontsize=16)

plt.plot(hist.history['acc'], 'b', label='Training Accuracy')

plt.plot(hist.history['val\_acc'], 'r', label='Validation Accuracy')

plt.legend(loc='lower right')

plt.show()