Index

Practical No	Title	Sign
1	Performing matrix multiplication and finding eigen vectors and eigen values using TensorFlow	
2	Solving XOR problem using deep feed forward network.	
3	Implementing deep neural network for performing binary classification task.	
4	a) Using deep feed forward network with two hidden layers for performing multiclass classification and predicting the class. b) Using a deep feed forward network with two hidden layers for performing classification and predicting the probability of class. c) Using a deep feed forward network with two hidden layers for performing linear regression and predicting values.	
5	a) Evaluating feed forward deep network for regression using KFold cross validation. b) Evaluating feed forward deep network for multiclass Classification using KFold cross-validation.	
6	Implementing regularization to avoid overfitting in binary classification.	
7	Demonstrate recurrent neural network that learns to perform sequence analysis for stock price.	
8	Performing encoding and decoding of images using deep autoencoder.	
9	Implementation of convolutional neural network to predict numbers from number images	
10	Denoising of images using autoencoder.	
11	RBM/DBM/DBN Recommendation System	

Practical 1

Write a program to perform matrix multiplication and finding eigenvectors and eigenvalues using TensorFlow.

Code

```
# -*- coding: utf-8 -*-
"""
Created on Thu Jan 20 23:16:43 2022
@author: Ankit Patel
Sapid: 53004200018
"""
#Importing Tensorflow
import tensorflow as tf
# Creating Matrix A
e_matrix_A = tf.random.uniform([2, 2], minval=3, maxval=10, dtype=tf.float32, name="matrixA")
print("Matrix A: \n{}\n\n".format(e_matrix_A))
# Calculating the eigen values and vectors using tf.linalg.eigh function of tensorflow
eigen_values_A, eigen_vectors_A = tf.linalg.eigh(e_matrix_A)
print("Eigen Vectors: \n{} \n\nEigen Values: \n{}\n".format(eigen_vectors_A, eigen_values_A))
# Multiplying our eigen vector by random number
sv = tf.multiply(5, eigen_vectors_A)
print(sv)
```

Output

```
# -*- coding: utf-8 -*-
"""
Created on Thu Jan 20 23:16:43 2022
@author: Ankit Patel
Sapid: 53004200018
"""
#Importing Tensorflow
import tensorflow as tf
# Creating Matrix A
e_matrix_A = tf.random.uniform([2, 2], minval=3, maxval=10, dtype=tf.float32, name="matrixA")
print("Matrix A: \n{}\n\n".format(e_matrix_A))
# Calculating the eigen values and vectors using tf.linalg.eigh function of tensorflow
eigen_values_A, eigen_vectors_A = tf.linalg.eigh(e_matrix_A)
print("Eigen Vectors: \n{} \n\nEigen Values: \n{}\n".format(eigen_vectors_A, eigen_values_A))
# Multiplying our eigen vector by random number
sv = tf.multiply(5, eigen_vectors_A)
print(sv)
```

```
Matrix A:
[[8.945024  3.7673373]
 [7.9602427 9.297421 ]]


Eigen Vectors:
[[-0.71488786 -0.6992391 ]
 [ 0.6992391  -0.71488786]]

Eigen Values:
[ 1.1590301 17.083414 ]

tf.Tensor(
[[-3.5744393 -3.4961953]
 [ 3.4961953 -3.5744393]], shape=(2, 2), dtype=float32)
```

Practical 2

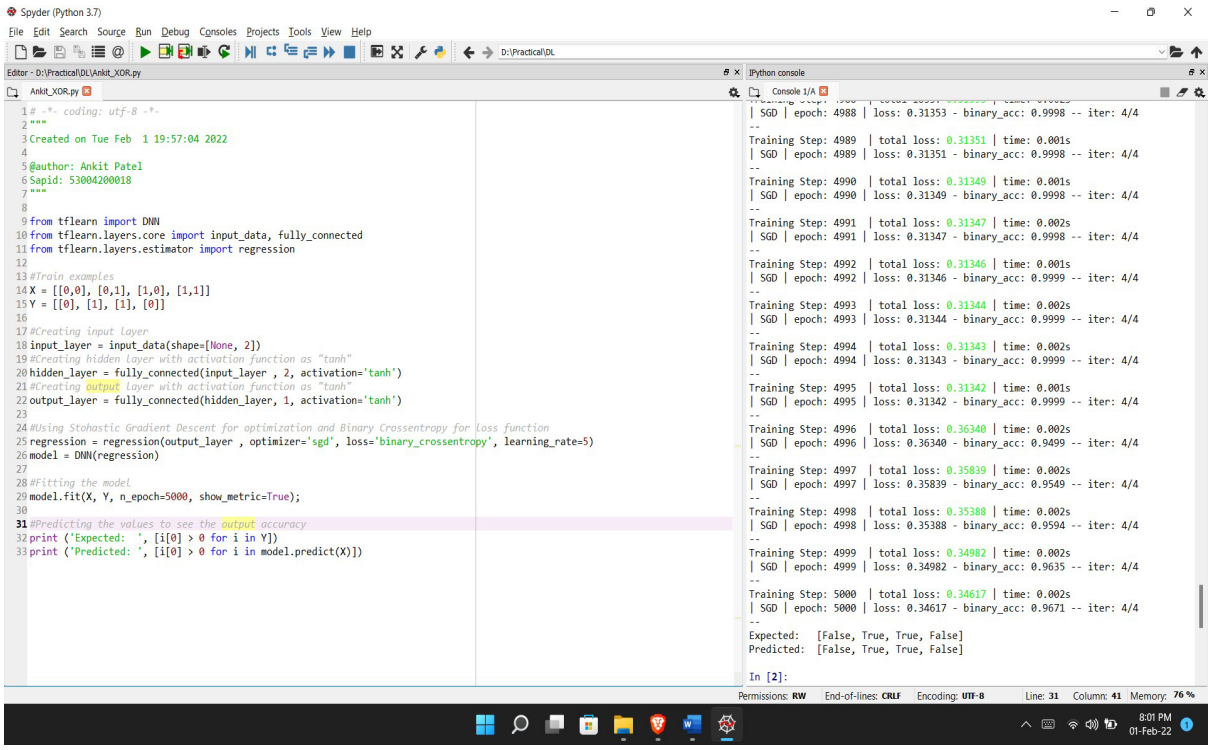
Solving XOR problem using deep feed forward network.

Code

```
# -*- coding: utf-8 -*-
"""
Created on Sun Jan 30 20:03:53 2022
@author: Ankit Patel
Sapid: 53004200018
"""
#Importing the neccassary libraries
from tflearn import DNN
from tflearn.layers.core import input_data, fully_connected
from tflearn.layers.estimator import regression
#Training dataset
X = [[0,0], [0,1], [1,0], [1,1]]
Y = [[0], [1], [1], [0]]
#Creating input layer of size 2
input_layer = input_data(shape=[None, 2])
#Creating hidden layer of size 2 with activation function as "tanh"
hidden_layer = fully_connected(input_layer , 2, activation='tanh')
#Creating output layer of size 1 with activation function as "tanh"
output_layer = fully_connected(hidden_layer, 1, activation='tanh')
#Using Stohastic Gradient Descent for optimization and Binary Crossentropy for loss function with a learning rate of 5
regression = regression(output_layer , optimizer='sgd', loss='binary_crossentropy', learning_rate=5)
model = DNN(regression)
#Fitting the model with 5000 iterations
model.fit(X, Y, n_epoch=5000, show_metric=True);
#Predicting the values to see the results accuracy
print ('Expected:  ', [i[0] > 0 for i in Y])
```

print ('Predicted: ', [i[0] > 0 for i in model.predict(X)])

Output



Practical 3

Implementing deep neural network for performing binary classification task

Code

```
# -*- coding: utf-8 -*-
"""
Created on Sun Feb  6 23:32:03 2022
@author: Ankit Patel
Sapid: 53004200018
"""
#importing all the necessary libraries
import pandas as pd
import tensorflow as tf
from tensorflow import keras
from sklearn.model_selection import train_test_split
#Reading the Dataset
df = pd.read_csv('molecular_activity.csv')
#split into input (X) and output (y) variables
X = df.iloc[:, 0:4].values
y = df['Activity']
#Traing the dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)
#Creating a Neural Network
model = keras.Sequential([
    keras.layers.Flatten(input_shape=(4,)), #Input Layer
    keras.layers.Dense(16, activation=tf.nn.relu), #Hidden Layer 1 with relu as activation function
        keras.layers.Dense(16, activation=tf.nn.relu), #Hidden Layer 2 with relu as activation function
    keras.layers.Dense(1, activation=tf.nn.sigmoid), #Output Layer with sigmoid as activation function
])
```

```
#Compiling Neaural Network with Loss function as Cross-entropy
model.compile(optimizer='adam',
              loss='binary_crossentropy',
              metrics=['accuracy'])
#Fitting the model
model.fit(X_train, y_train, epochs=50, batch_size=1)
#Calculating Model Accuracy and Loss
test_loss, test_acc = model.evaluate(X_test, y_test)
print('Test accuracy:', test_acc)
```

Output

File Edit Format Run Options Window Help

```
#Ankit Patel
# 53004300018

#importing all the necessary libraries
import pandas as pd
import tensorflow as tf
from tensorflow import keras
from sklearn.model_selection import train_test_split

#Reading the Dataset
df = pd.read_csv('heart.csv')
#split into input (X) and output (y) variables
X = df.iloc[:, 0:4].values
y = df['trestbps']
#Traing the dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)
#Creating a Neural Network
model = keras.Sequential([
keras.layers.Flatten(input_shape=(4,)), #Input Layer
keras.layers.Dense(16, activation=tf.nn.relu), #Hidden Layer 1
keras.layers.Dense(16, activation=tf.nn.relu), #Hidden Layer 2
keras.layers.Dense(1, activation=tf.nn.sigmoid), #Output Layer
])

#Compiling Neaural Network with Loss function as Cross-entropy
model.compile(optimizer='adam',
loss='binary_crossentropy',
metrics=['accuracy'])
#Fitting the model
model.fit(X_train, y_train, epochs=50, batch_size=1)
#Calculating Model Accuracy and Loss
test_loss, test_acc = model.evaluate(X_test, y_test)
print('Test accuracy:', test_acc)
```

Console 1/A

```
accuracy: 0.8201
Epoch 39/50
378/378 [==============================] - 0s 872us/step - loss: 0.4296 -
accuracy: 0.8228
Epoch 40/50
378/378 [==============================] - 0s 849us/step - loss: 0.4342 -
accuracy: 0.8148
Epoch 41/50
378/378 [==============================] - 0s 879us/step - loss: 0.4508 -
accuracy: 0.8254
Epoch 42/50
378/378 [==============================] - 0s 868us/step - loss: 0.4579 -
accuracy: 0.8386
Epoch 43/50
378/378 [==============================] - 0s 850us/step - loss: 0.4522 -
accuracy: 0.8307
Epoch 44/50
378/378 [==============================] - 0s 849us/step - loss: 0.4383 -
accuracy: 0.8228
Epoch 45/50
378/378 [==============================] - 0s 874us/step - loss: 0.4338 -
accuracy: 0.8333
Epoch 46/50
378/378 [==============================] - 0s 849us/step - loss: 0.4412 -
accuracy: 0.8201
Epoch 47/50
378/378 [==============================] - 0s 902us/step - loss: 0.4186 -
accuracy: 0.8228
Epoch 48/50
378/378 [==============================] - 0s 899us/step - loss: 0.4083 -
accuracy: 0.8307
Epoch 49/50
378/378 [==============================] - 0s 849us/step - loss: 0.4479 -
accuracy: 0.8148
Epoch 50/50
378/378 [==============================] - 0s 875us/step - loss: 0.4422 -
accuracy: 0.8280
6/6 [==============================] - 0s 2ms/step - loss: 0.4352 - accuracy:
0.8210
Test accuracy: 0.8209876418113708
```

Practical 4A

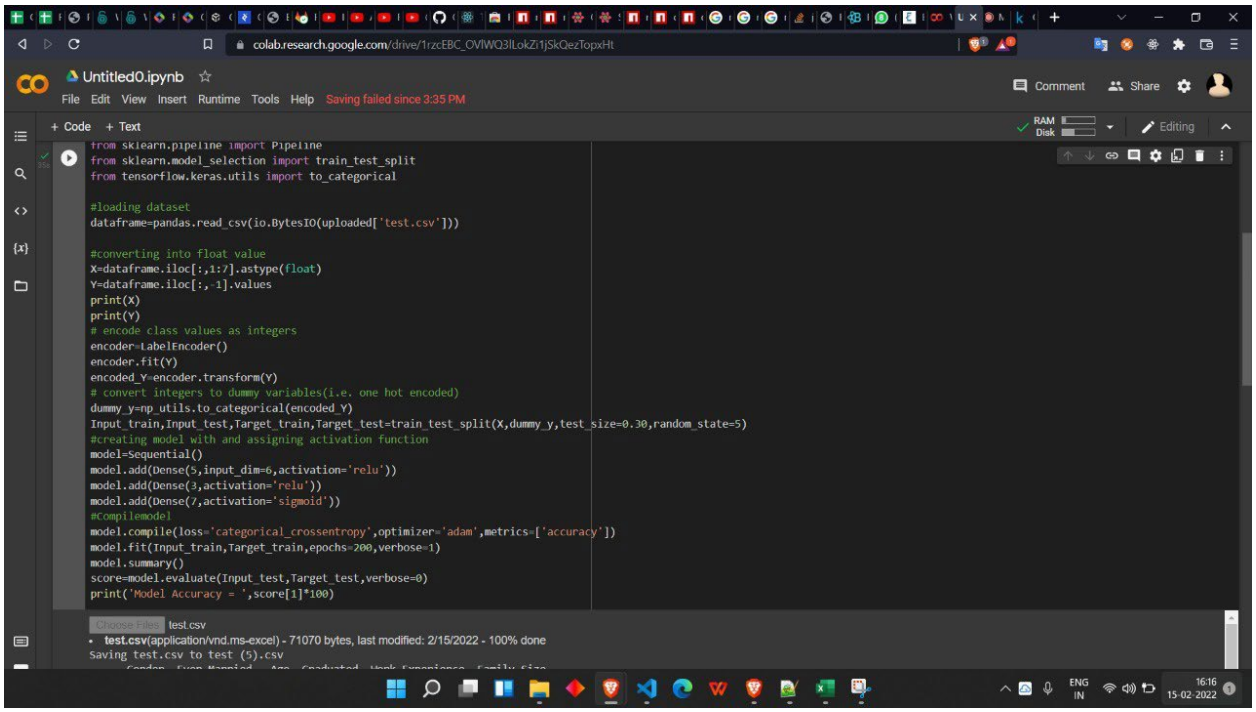
Using deep feed forward network with two hidden layers for performing multiclass classification and predicting the class.

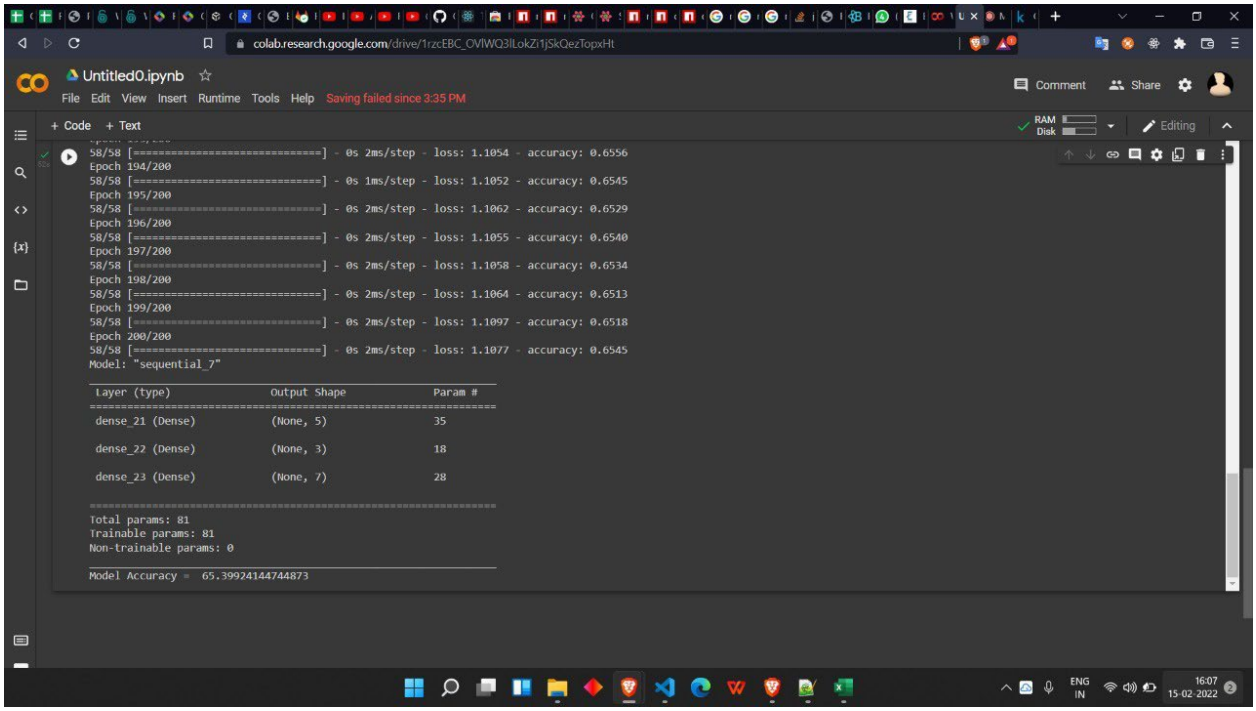
Code

```
"""
DL prac
4a.ipynbAnkit
Patel
53004200018
"""
#code for uploading data from
google.colab import files
uploaded=files.upload()
import io
#importing io for uploading .csv
fil4import pandas
from keras.models import Sequential
from keras.layers import Dense
from keras.wrappers.scikit_learn import KerasClassifier
from keras.utils import np_utils
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
from sklearn.preprocessing import LabelEncoder
from sklearn.pipeline import Pipeline
from sklearn.model_selection import train_test_split
from tensorflow.keras.utils import to_categorical

#loading dataset
dataframe=pandas.read_csv(io.BytesIO(uploaded['test.csv']))

#converting into float value
X=dataframe.iloc[:,1:7].astype(float
)Y=dataframe.iloc[:,-1].values
print(X)
print(Y)
# encode class values as integers
encoder=LabelEncoder()
encoder.fit(Y)
encoded_Y=encoder.transform(Y)
# convert integers to dummy variables(i.e. one hot encoded) dummy_y=np_utils.to_categorical(encoded_Y)
Input_train,Input_test,Target_train,Target_test=train_test_split(X,dummy_y,test_size=0.30,random_sta te=5)
```

```
#creating model with and assigning activation function
model=Sequential()
model.add(Dense(5,input_dim=6,activation='relu'))
model.add(Dense(3,activation='relu')
model.add(Dense(7,activation='sigmoid'))
#Compilemodel
model.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['accuracy'])
model.fit(Input_train,Target_train,epochs=200,verbose=1)
model.summary()
score=model.evaluate(Input_test,Target_test,verbose=0 )
print('Model Accuracy = ',score[1]*100)
```

Output


colab.research.google.com/drive/1rzcEBC_OVlWQ3lLokZi1jSkQezTopxHt
Untitled0.ipynb
File Edit View Insert Runtime Tools Help Saving failed since 3:35 PM
Comment Share
+ Code + Text
RAM Disk
Editing
from sklearn.pipeline import Pipeline
from sklearn.model_selection import train_test_split
from tensorflow.keras.utils import to_categorical
#loading dataset
dataframe=pandas.read_csv(io.BytesIO(uploaded['test.csv']))
#converting into float value
X=dataframe.iloc[:,1:7].astype(float)
Y=dataframe.iloc[:,-1].values
print(X)
print(Y)
# encode class values as integers
encoder=LabelEncoder()
encoder.fit(Y)
encoded_Y=encoder.transform(Y)
# convert integers to dummy variables(i.e. one hot encoded)
dummy_y=np_utils.to_categorical(encoded_Y)
Input_train,Input_test,Target_train,Target_test=train_test_split(X,dummy_y,test_size=0.30,random_state=5)
#creating model with and assigning activation function
model=Sequential()
model.add(Dense(5,input_dim=6,activation='relu'))
model.add(Dense(3,activation='relu'))
model.add(Dense(7,activation='sigmoid'))
#Compilemodel
model.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['accuracy'])
model.fit(Input_train,Target_train,epochs=200,verbose=1)
model.summary()
score=model.evaluate(Input_test,Target_test,verbose=0)
print('Model Accuracy = ',score[1]*100)
Choose Files test.csv
test.csv(application/vnd.ms-excel) - 71070 bytes, last modified: 2/15/2022 - 100% done
Saving test.csv to test (5).csv
ENG IN
16:16
15-02-2022


colab.research.google.com/drive/1rzcEBC_OVlWQ3lLokZi1jSkQezTopxHt
Untitled0.ipynb
File Edit View Insert Runtime Tools Help Saving failed since 3:35 PM
Comment Share
+ Code + Text
RAM Disk
Editing
58/58 [==========] - 0s 2ms/step - loss: 1.1054 - accuracy: 0.6556
Epoch 194/200
58/58 [==========] - 0s 1ms/step - loss: 1.1052 - accuracy: 0.6545
Epoch 195/200
58/58 [==========] - 0s 2ms/step - loss: 1.1062 - accuracy: 0.6529
Epoch 196/200
58/58 [==========] - 0s 2ms/step - loss: 1.1055 - accuracy: 0.6540
Epoch 197/200
58/58 [==========] - 0s 2ms/step - loss: 1.1058 - accuracy: 0.6534
Epoch 198/200
58/58 [==========] - 0s 2ms/step - loss: 1.1064 - accuracy: 0.6513
Epoch 199/200
58/58 [==========] - 0s 2ms/step - loss: 1.1097 - accuracy: 0.6518
Epoch 200/200
58/58 [==========] - 0s 2ms/step - loss: 1.1077 - accuracy: 0.6545
Model: "sequential_7"
Layer (type) Output Shape Param #
dense_21 (Dense) (None, 5) 35
dense_22 (Dense) (None, 3) 18
dense_23 (Dense) (None, 7) 28
Total params: 81
Trainable params: 81
Non-trainable params: 0
Model Accuracy = 65.39924144744873
ENG IN
16:07
15-02-2022

Practical 4B

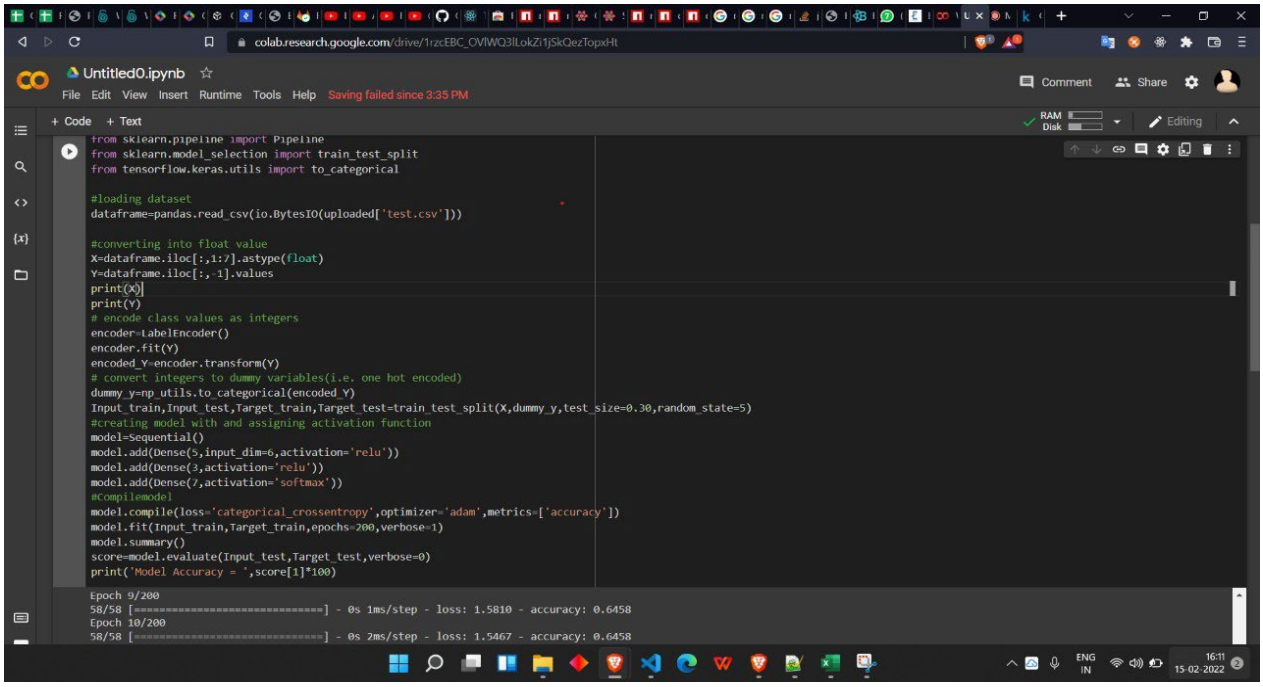
Using a deep feed forward network with two hidden layers for performing classification and predicting the probability of class.

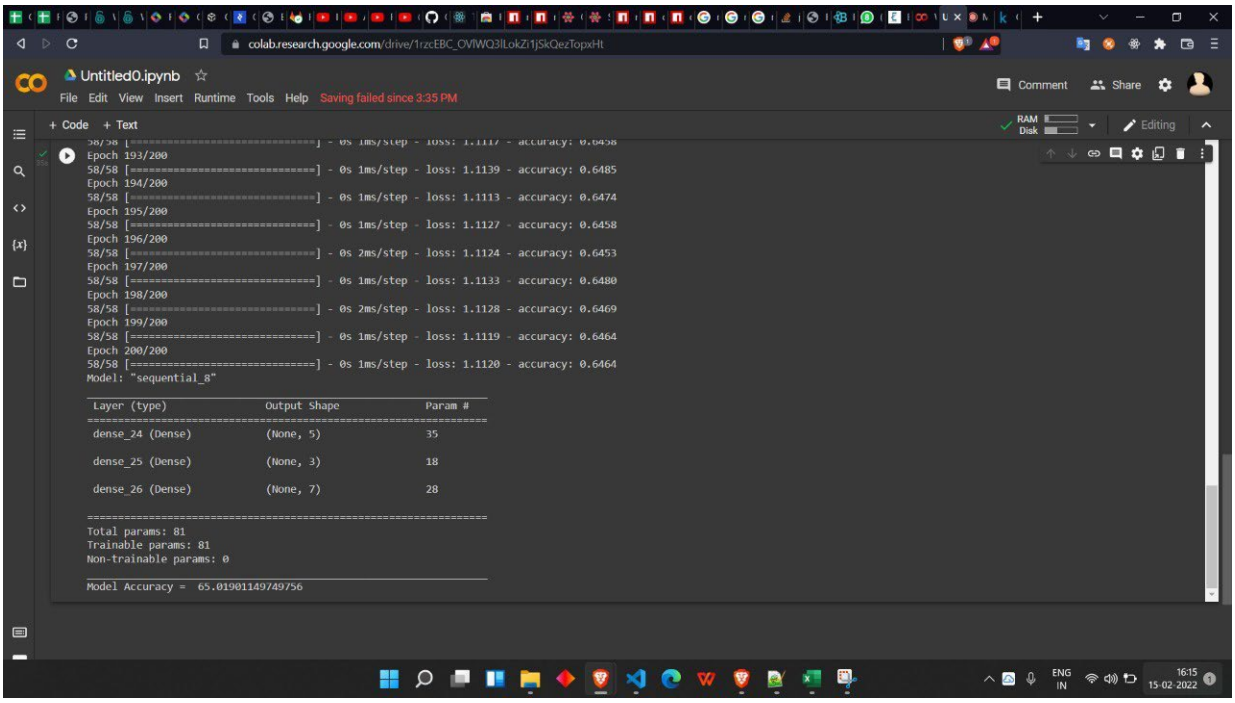
Code

```
"""
Ankit Patel
53004200018
"""

#code for uploading data from
google.colab import files
uploaded=files.upload()
import io
#importing io for uploading .csv
fil4import pandas
from keras.models import Sequential
from keras.layers import Dense
from keras.wrappers.scikit_learn import KerasClassifier
from keras.utils import np_utils
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
from sklearn.preprocessing import LabelEncoder
from sklearn.pipeline import Pipeline
from sklearn.model_selection import train_test_split
from tensorflow.keras.utils import to_categorical

#loading dataset
dataframe=pandas.read_csv(io.BytesIO(uploaded['test.csv']))

#converting into float value
X=dataframe.iloc[:,1:7].astype(float)
Y=dataframe.iloc[:,-1].values
print(X)
print(Y)
# encode class values as integers
encoder=LabelEncoder()
encoder.fit(Y)
encoded_Y=encoder.transform(Y)
# convert integers to dummy variables(i.e. one hot encoded) dummy_y=np_utils.to_categorical(encoded_Y)
Input_train,Input_test,Target_train,Target_test=train_test_split(X,dummy_y,test_size=0.30,random_sta te=5)
#creating model with and assigning activation function
model=Sequential()
```

```
model.add(Dense(5,input_dim=6,activation='relu'))
model.add(Dense(3,activation='relu'))
model.add(Dense(7,activation='softmax'))
#Compilemodel
model.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['accuracy'])
model.fit(Input_train,Target_train,epochs=200,verbose=1)
model.summary()
score=model.evaluate(Input_test,Target_test,verbose=0) print('Model
Accuracy = ',score[1]*100)
```

Output


colab.research.google.com/drive/1rzcEBC_OVlWQ3lLokZi1jSkQezTopxHt
Untitled0.ipynb
File Edit View Insert Runtime Tools Help Saving failed since 3:35 PM
Comment Share
+ Code + Text
RAM Disk
Editing
from sklearn.pipeline import Pipeline
from sklearn.model_selection import train_test_split
from tensorflow.keras.utils import to_categorical
#loading dataset
dataframe=pandas.read_csv(io.BytesIO(uploaded['test.csv']))
#converting into float value
X=dataframe.iloc[:,1:7].astype(float)
Y=dataframe.iloc[:,-1].values
print(X)
print(Y)
# encode class values as integers
encoder=LabelEncoder()
encoder.fit(Y)
encoded_Y=encoder.transform(Y)
# convert integers to dummy variables(i.e. one hot encoded)
dummy_y=np_utils.to_categorical(encoded_Y)
Input_train,Input_test,Target_train,Target_test=train_test_split(X,dummy_y,test_size=0.30,random_state=5)
#creating model with and assigning activation function
model=Sequential()
model.add(Dense(5,input_dim=6,activation='relu'))
model.add(Dense(3,activation='relu'))
model.add(Dense(7,activation='softmax'))
#compilemodel
model.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['accuracy'])
model.fit(Input_train,Target_train,epochs=200,verbose=1)
model.summary()
score=model.evaluate(Input_test,Target_test,verbose=0)
print('Model Accuracy = ',score[1]*100)
Epoch 9/200
58/58 [==========] - 0s 1ms/step - loss: 1.5810 - accuracy: 0.6458
Epoch 10/200
58/58 [==========] - 0s 2ms/step - loss: 1.5467 - accuracy: 0.6458
ENG IN
16:11 15-02-2022


colab.research.google.com/drive/1rzcEBC_OVlWQ3lLokZi1jSkQezTopxHt
Untitled0.ipynb
File Edit View Insert Runtime Tools Help Saving failed since 3:35 PM
Comment Share
+ Code + Text
RAM Disk
Editing
Epoch 193/200
58/58 [==========] - 0s 1ms/step - loss: 1.1139 - accuracy: 0.6485
Epoch 194/200
58/58 [==========] - 0s 1ms/step - loss: 1.1113 - accuracy: 0.6474
Epoch 195/200
58/58 [==========] - 0s 1ms/step - loss: 1.1127 - accuracy: 0.6458
Epoch 196/200
58/58 [==========] - 0s 2ms/step - loss: 1.1124 - accuracy: 0.6453
Epoch 197/200
58/58 [==========] - 0s 1ms/step - loss: 1.1133 - accuracy: 0.6480
Epoch 198/200
58/58 [==========] - 0s 2ms/step - loss: 1.1128 - accuracy: 0.6469
Epoch 199/200
58/58 [==========] - 0s 1ms/step - loss: 1.1119 - accuracy: 0.6464
Epoch 200/200
58/58 [==========] - 0s 1ms/step - loss: 1.1120 - accuracy: 0.6464
Model: "sequential_8"
Layer (type) Output Shape Param #
dense_24 (Dense) (None, 5) 35
dense_25 (Dense) (None, 3) 18
dense_26 (Dense) (None, 7) 28
Total params: 81
Trainable params: 81
Non-trainable params: 0
Model Accuracy = 65.01901149749756
ENG IN
16:15 15-02-2022

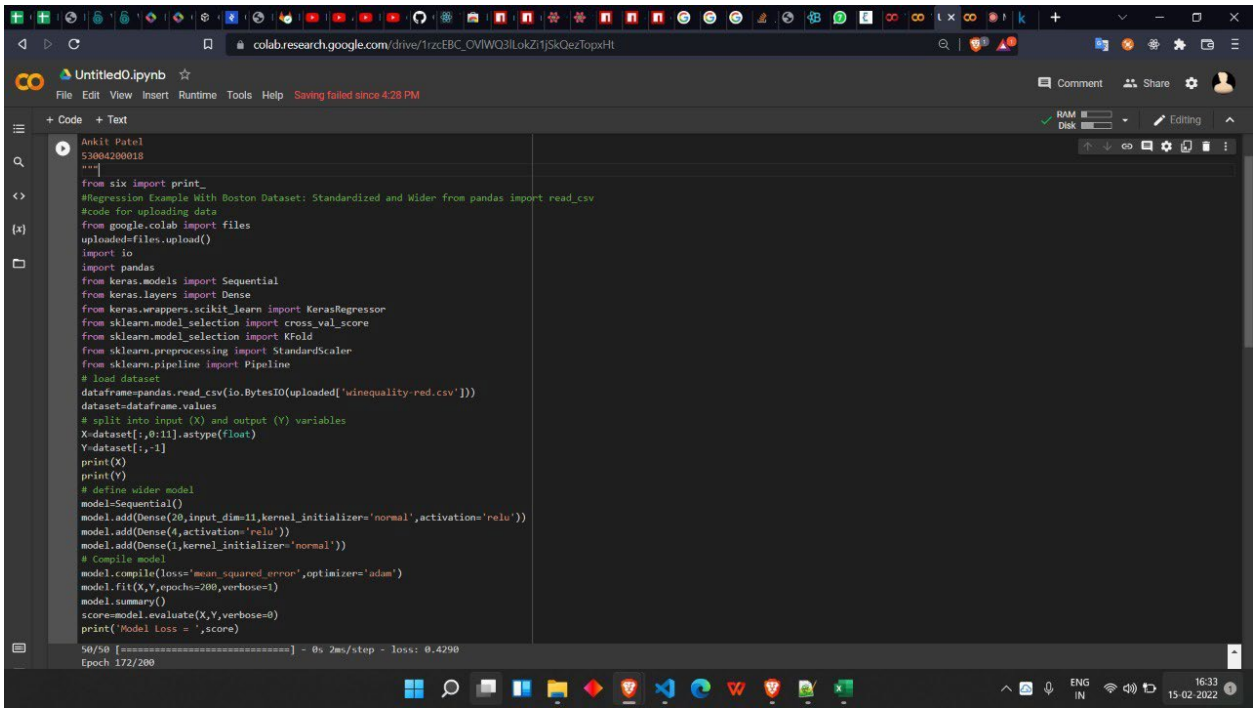
Practical 4C

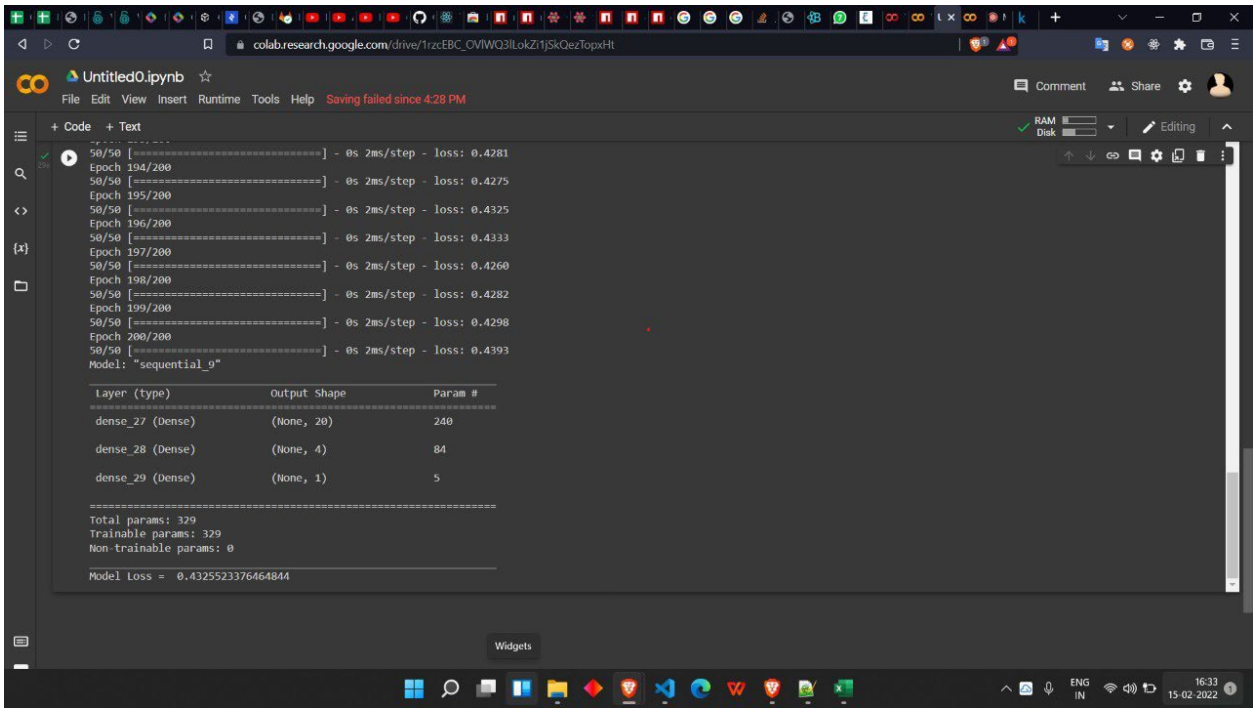
Using a deep feed forward network with two hidden layers for performing linear regression and predicting values.

Code

```
"""
DL prac4c.ipynb
Ankit Patel
53004200018 """

from six import print_
#Regression Example With Boston Dataset: Standardized and Wider from pandas import read_csv
#code for uploading data
from google.colab import files
uploaded=files.upload()

import io
import pandas
from keras.models import Sequential
from keras.layers import Dense
from keras.wrappers.scikit_learn import KerasRegressor
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline

# load dataset
dataframe=pandas.read_csv(io.BytesIO(uploaded['winequality-red.csv']))
dataset=dataframe.values
# split into input (X) and output (Y) variables
X=dataset[:,0:11].astype(float)
Y=dataset[:,1]
print(X)
print(Y)
# define wider model
model=Sequential()
model.add(Dense(20,input_dim=11,kernel_initializer='normal',activation='relu'))
model.add(Dense(4,activation='relu')) model.add(Dense(1,kernel_initializer='normal'))
# Compile model
model.compile(loss='mean_squared_error',optimizer='adam')
model.fit(X,Y,epochs=200,verbose=1)
model.summary()
score=model.evaluate(X,Y,verbose=0)
print('Model Loss = ',score)
```

Output


colab.research.google.com/drive/1rzcEBC_OVlWQ3lLokZi1jSkQezTopxHt
Untitled0.ipynb
File Edit View Insert Runtime Tools Help Saving failed since 4:28 PM
Comment Share
+ Code + Text
RAM Disk
Editing
Ankit Patel
53004200018
"""
from six import print_
#Regression Example With Boston Dataset: Standardized and Wider from pandas import read_csv
#code for uploading data
from google.colab import files
uploaded=files.upload()
import io
import pandas
from keras.models import Sequential
from keras.layers import Dense
from keras.wrappers.scikit_learn import KerasRegressor
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline
# load dataset
dataframe=pandas.read_csv(io.BytesIO(uploaded['winequality-red.csv']))
dataset=dataframe.values
# split into input (X) and output (Y) variables
X=dataset[:,0:11].astype(float)
Y=dataset[:,-1]
print(X)
print(Y)
# define wider model
model=Sequential()
model.add(Dense(20,input_dim=11,kernel_initializer='normal',activation='relu'))
model.add(Dense(4,activation='relu'))
model.add(Dense(1,kernel_initializer='normal'))
# Compile model
model.compile(loss='mean_squared_error',optimizer='adam')
model.fit(X,Y,epochs=200,verbose=1)
model.summary()
score=model.evaluate(X,Y,verbose=0)
print('Model Loss = ',score)
50/50 [==============================] - 0s 2ms/step - loss: 0.4290
Epoch 172/200
ENG IN
16:33 15-02-2022


colab.research.google.com/drive/1rzcEBC_OVlWQ3lLokZi1jSkQezTopxHt
Untitled0.ipynb
File Edit View Insert Runtime Tools Help Saving failed since 4:28 PM
Comment Share
+ Code + Text
RAM Disk
Editing
50/50 [==============================] - 0s 2ms/step - loss: 0.4281
Epoch 194/200
50/50 [==============================] - 0s 2ms/step - loss: 0.4275
Epoch 195/200
50/50 [==============================] - 0s 2ms/step - loss: 0.4325
Epoch 196/200
50/50 [==============================] - 0s 2ms/step - loss: 0.4333
Epoch 197/200
50/50 [==============================] - 0s 2ms/step - loss: 0.4260
Epoch 198/200
50/50 [==============================] - 0s 2ms/step - loss: 0.4282
Epoch 199/200
50/50 [==============================] - 0s 2ms/step - loss: 0.4298
Epoch 200/200
50/50 [==============================] - 0s 2ms/step - loss: 0.4393
Model: "sequential_9"
Layer (type) Output Shape Param #
dense_27 (Dense) (None, 20) 240
dense_28 (Dense) (None, 4) 84
dense_29 (Dense) (None, 1) 5
Total params: 329
Trainable params: 329
Non-trainable params: 0
Model Loss = 0.4325523376464844
Widgets
ENG IN
16:33 15-02-2022

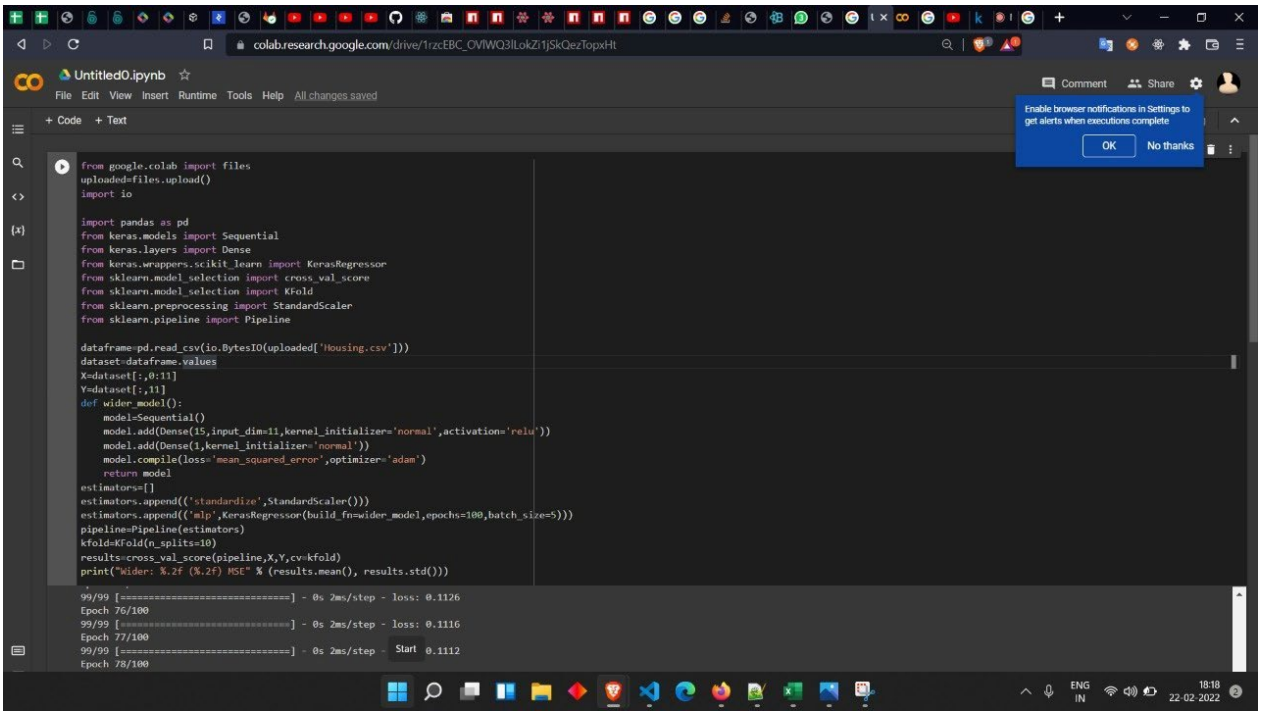
Practical 5A

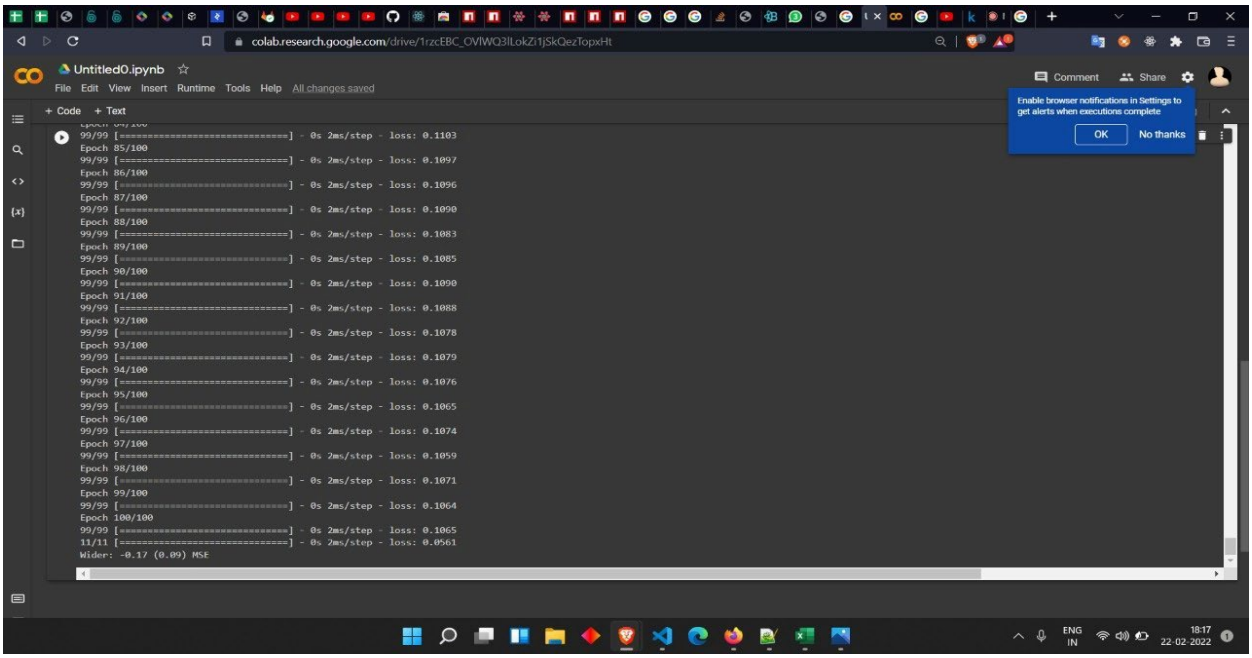
Evaluating feed forward deep network for regression using KFold cross validation.

Code

```
from google.colab import files
uploaded=files.upload() import io

import pandas as pd
from keras.models import Sequentialfrom
keras.layers import Dense
from keras.wrappers.scikit_learn import KerasRegressorfrom
sklearn.model_selection import cross_val_score from
sklearn.model_selection import KFold
from sklearn.preprocessing import StandardScalerfrom
sklearn.pipeline import Pipeline

dataframe=pd.read_csv(io.BytesIO(uploaded['Housing.csv'])) dataset=dataframe.values
X=dataset[:,0:11]
Y=dataset[:,11]

def wider_model():
   model=Sequential()
   model.add(Dense(15,input_dim=11,kernel_initializer='normal',activation='relu'))
   model.add(Dense(1,kernel_initializer='normal'))
   model.compile(loss='mean_squared_error',optimizer='adam')
   return model

estimators=[] estimators.append(('standardize',StandardScaler()))
estimators.append(('mlp',KerasRegressor(build_fn=wider_model,epochs=100,batch_size=5)))
pipeline=Pipeline(estimators)
kfold=KFold(n_splits=10)
results=cross_val_score(pipeline,X,Y,cv=kfold)
print("Wider: %.2f (%.2f) MSE" % (results.mean(), results.std()))
```

Output


colab.research.google.com/drive/1rzcEBC_OVlWQ3lLokZi1jSkQezTopxHt
Untitled0.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
Comment Share
Enable browser notifications in Settings to get alerts when executions complete
OK No thanks
from google.colab import files
uploaded=files.upload()
import io
import pandas as pd
from keras.models import Sequential
from keras.layers import Dense
from keras.wrappers.scikit_learn import KerasRegressor
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline
dataframe=pd.read_csv(io.BytesIO(uploaded['Housing.csv']))
dataset=dataframe.values
X=dataset[:,0:11]
Y=dataset[:,11]
def wider_model():
model=Sequential()
model.add(Dense(15,input_dim=11,kernel_initializer='normal',activation='relu'))
model.add(Dense(1,kernel_initializer='normal'))
model.compile(loss='mean_squared_error',optimizer='adam')
return model
estimators=[]
estimators.append(('standardize',StandardScaler()))
estimators.append(('mlp',KerasRegressor(build_fn=wider_model,epochs=100,batch_size=5)))
pipeline=Pipeline(estimators)
kfold=KFold(n_splits=10)
results=cross_val_score(pipeline,X,Y,cv=kfold)
print("Wider: %.2f (%.2f) MSE" % (results.mean(), results.std()))
99/99 [==========] - 0s 2ms/step - loss: 0.1126
Epoch 76/100
99/99 [==========] - 0s 2ms/step - loss: 0.1116
Epoch 77/100
99/99 [==========] - 0s 2ms/step - loss: 0.1112
Epoch 78/100
Start
ENG IN
18:18
22-02-2022


colab.research.google.com/drive/1rzcEBC_OVlWQ3lLokZi1jSkQezTopxHt
Untitled0.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
Comment Share
Enable browser notifications in Settings to get alerts when executions complete
OK No thanks
99/99 [==========] - 0s 2ms/step - loss: 0.1103
Epoch 85/100
99/99 [==========] - 0s 2ms/step - loss: 0.1097
Epoch 86/100
99/99 [==========] - 0s 2ms/step - loss: 0.1096
Epoch 87/100
99/99 [==========] - 0s 2ms/step - loss: 0.1090
Epoch 88/100
99/99 [==========] - 0s 2ms/step - loss: 0.1083
Epoch 89/100
99/99 [==========] - 0s 2ms/step - loss: 0.1085
Epoch 90/100
99/99 [==========] - 0s 2ms/step - loss: 0.1090
Epoch 91/100
99/99 [==========] - 0s 2ms/step - loss: 0.1088
Epoch 92/100
99/99 [==========] - 0s 2ms/step - loss: 0.1078
Epoch 93/100
99/99 [==========] - 0s 2ms/step - loss: 0.1079
Epoch 94/100
99/99 [==========] - 0s 2ms/step - loss: 0.1076
Epoch 95/100
99/99 [==========] - 0s 2ms/step - loss: 0.1065
Epoch 96/100
99/99 [==========] - 0s 2ms/step - loss: 0.1074
Epoch 97/100
99/99 [==========] - 0s 2ms/step - loss: 0.1059
Epoch 98/100
99/99 [==========] - 0s 2ms/step - loss: 0.1071
Epoch 99/100
99/99 [==========] - 0s 2ms/step - loss: 0.1064
Epoch 100/100
99/99 [==========] - 0s 2ms/step - loss: 0.1065
11/11 [==========] - 0s 2ms/step - loss: 0.0561
Wider: -0.17 (0.09) MSE
ENG IN
18:17
22-02-2022

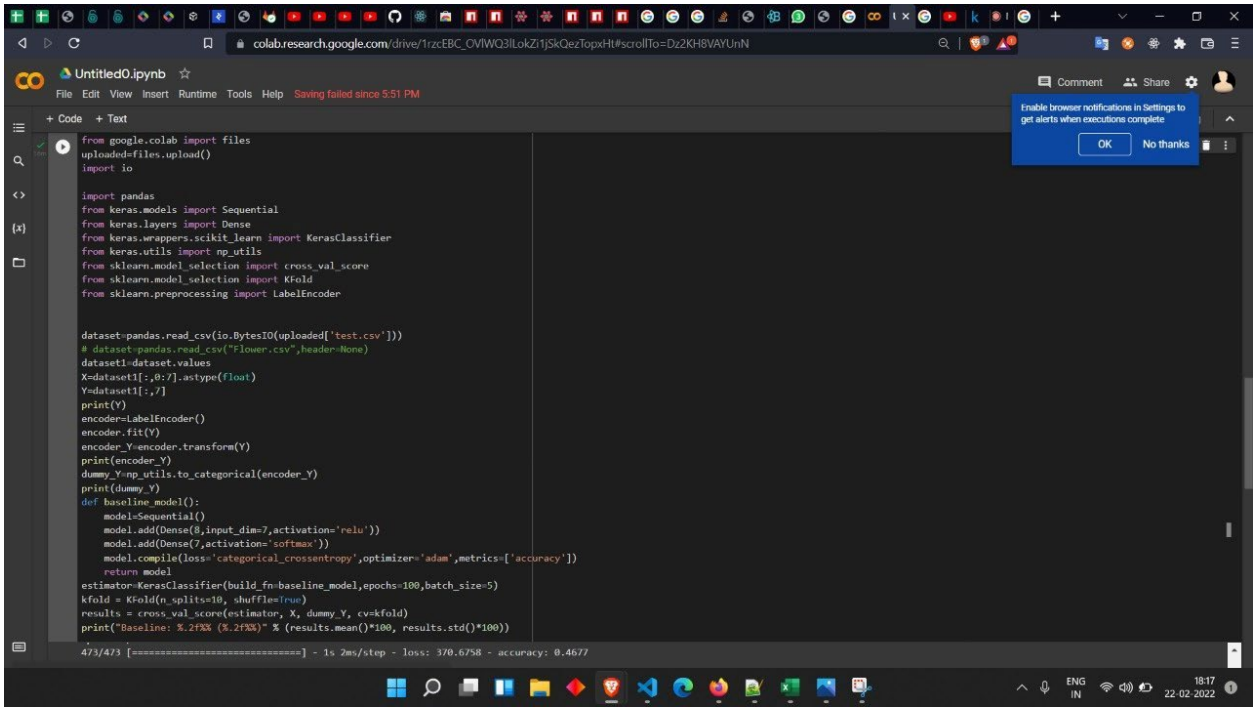
Practical 5B

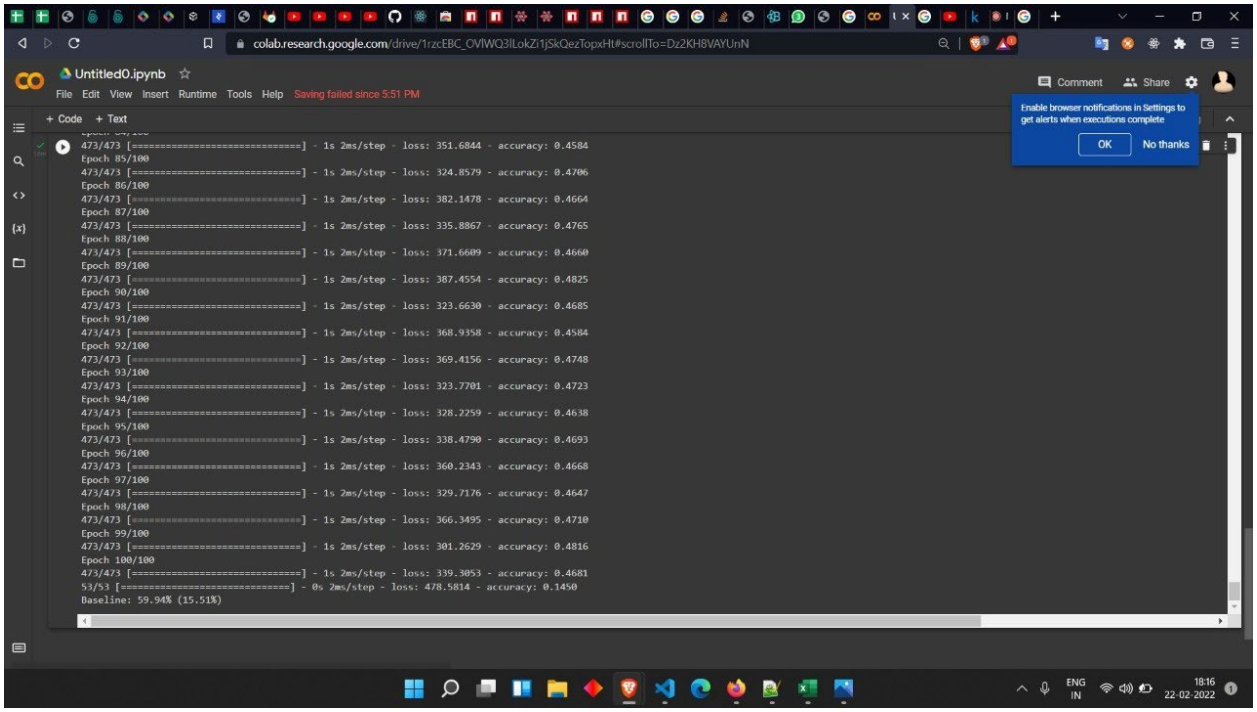
Evaluating feed forward deep network for multiclass Classification using KFold cross-validation.

Code

```
# -*- coding: utf-8 -*-
"""
Created on Sun Feb 20 19:01:55 2022
@author: Ankit Patel
Sapid: 53004200018
"""
from google.colab import files
uploaded=files.upload() import io
import pandas
from keras.models import Sequentialfrom
keras.layers import Dense
from keras.wrappers.scikit_learn import KerasClassifierfrom
keras.utils import np_utils
from sklearn.model_selection import cross_val_scorefrom
sklearn.model_selection import KFold
from sklearn.preprocessing import LabelEncoder


dataset=pandas.read_csv(io.BytesIO(uploaded['test.csv'])) #
dataset=pandas.read_csv("Flower.csv",header=None)
dataset1=dataset.values
X=dataset1[:,0:7].astype(float)Y=dataset1[:,7]
print(Y) encoder=LabelEncoder()
encoder.fit(Y)
encoder_Y=encoder.transform(Y) print(encoder_Y)
dummy_Y=np_utils.to_categorical(encoder_Y)
print(dummy_Y)
def baseline_model():
    model=Sequential()
    model.add(Dense(8,input_dim=7,activation='relu')) model.add(Dense(7,activation='softmax'))
    model.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['accuracy']) return model
estimator=KerasClassifier(build_fn=baseline_model,epochs=100,batch_size=5) kfold =
KFold(n_splits=10, shuffle=True)
results = cross_val_score(estimator, X, dummy_Y, cv=kfold)
print("Baseline: %.2f%% (%.2f%%)" % (results.mean()*100, results.std()*100))
```

Output


colab.research.google.com/drive/1rzcEBC_OVlWQ3lLokZi1jSkQezTopxHt#scrollTo=Dz2KH8VAYUnN
Untitled0.ipynb
File Edit View Insert Runtime Tools Help Saving failed since 5:51 PM
Comment Share
+ Code + Text
Enable browser notifications in Settings to get alerts when executions complete
OK No thanks
from google.colab import files
uploaded=files.upload()
import io
import pandas
from keras.models import Sequential
from keras.layers import Dense
from keras.wrappers.scikit_learn import KerasClassifier
from keras.utils import np_utils
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
from sklearn.preprocessing import LabelEncoder
dataset=pandas.read_csv(io.BytesIO(uploaded['test.csv']))
# dataset=pandas.read_csv("Flower.csv",header=None)
dataset1=dataset.values
X=dataset1[:,0:7].astype(float)
Y=dataset1[:,7]
print(Y)
encoder=LabelEncoder()
encoder.fit(Y)
encoder_Y=encoder.transform(Y)
print(encoder_Y)
dummy_Y=np_utils.to_categorical(encoder_Y)
print(dummy_Y)
def baseline_model():
model=Sequential()
model.add(Dense(8,input_dim=7,activation='relu'))
model.add(Dense(7,activation='softmax'))
model.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['accuracy'])
return model
estimator=KerasClassifier(build_fn=baseline_model,epochs=100,batch_size=5)
kfold = KFold(n_splits=10, shuffle=True)
results = cross_val_score(estimator, X, dummy_Y, cv=kfold)
print("Baseline: %.2f%% (%.2f%%)" % (results.mean()*100, results.std()*100))
473/473 [==========] - 1s 2ms/step - loss: 370.6758 - accuracy: 0.4677
ENG IN 18:17 22-02-2022


colab.research.google.com/drive/1rzcEBC_OVlWQ3lLokZi1jSkQezTopxHt#scrollTo=Dz2KH8VAYUnN
Untitled0.ipynb
File Edit View Insert Runtime Tools Help Saving failed since 5:51 PM
Comment Share
+ Code + Text
Enable browser notifications in Settings to get alerts when executions complete
OK No thanks
473/473 [==========] - 1s 2ms/step - loss: 351.6844 - accuracy: 0.4584
Epoch 85/100
473/473 [==========] - 1s 2ms/step - loss: 324.8579 - accuracy: 0.4706
Epoch 86/100
473/473 [==========] - 1s 2ms/step - loss: 382.1478 - accuracy: 0.4664
Epoch 87/100
473/473 [==========] - 1s 2ms/step - loss: 335.8867 - accuracy: 0.4765
Epoch 88/100
473/473 [==========] - 1s 2ms/step - loss: 371.6609 - accuracy: 0.4660
Epoch 89/100
473/473 [==========] - 1s 2ms/step - loss: 387.4554 - accuracy: 0.4825
Epoch 90/100
473/473 [==========] - 1s 2ms/step - loss: 323.6630 - accuracy: 0.4685
Epoch 91/100
473/473 [==========] - 1s 2ms/step - loss: 368.9358 - accuracy: 0.4584
Epoch 92/100
473/473 [==========] - 1s 2ms/step - loss: 369.4156 - accuracy: 0.4748
Epoch 93/100
473/473 [==========] - 1s 2ms/step - loss: 323.7701 - accuracy: 0.4723
Epoch 94/100
473/473 [==========] - 1s 2ms/step - loss: 328.2259 - accuracy: 0.4638
Epoch 95/100
473/473 [==========] - 1s 2ms/step - loss: 338.4790 - accuracy: 0.4693
Epoch 96/100
473/473 [==========] - 1s 2ms/step - loss: 360.2343 - accuracy: 0.4668
Epoch 97/100
473/473 [==========] - 1s 2ms/step - loss: 329.7176 - accuracy: 0.4647
Epoch 98/100
473/473 [==========] - 1s 2ms/step - loss: 366.3495 - accuracy: 0.4710
Epoch 99/100
473/473 [==========] - 1s 2ms/step - loss: 301.2629 - accuracy: 0.4816
Epoch 100/100
473/473 [==========] - 1s 2ms/step - loss: 339.3053 - accuracy: 0.4681
53/53 [==========] - 0s 2ms/step - loss: 478.5814 - accuracy: 0.1450
Baseline: 59.94% (15.51%)
ENG IN 18:16 22-02-2022

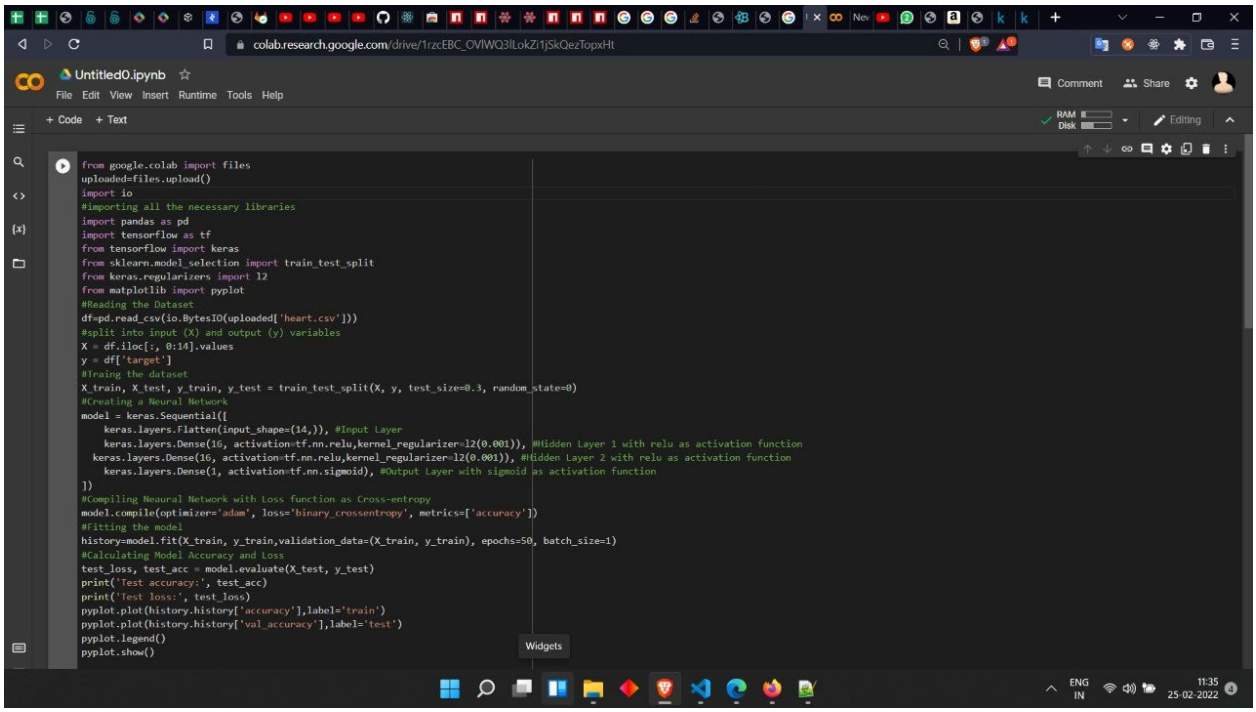
Practical 6

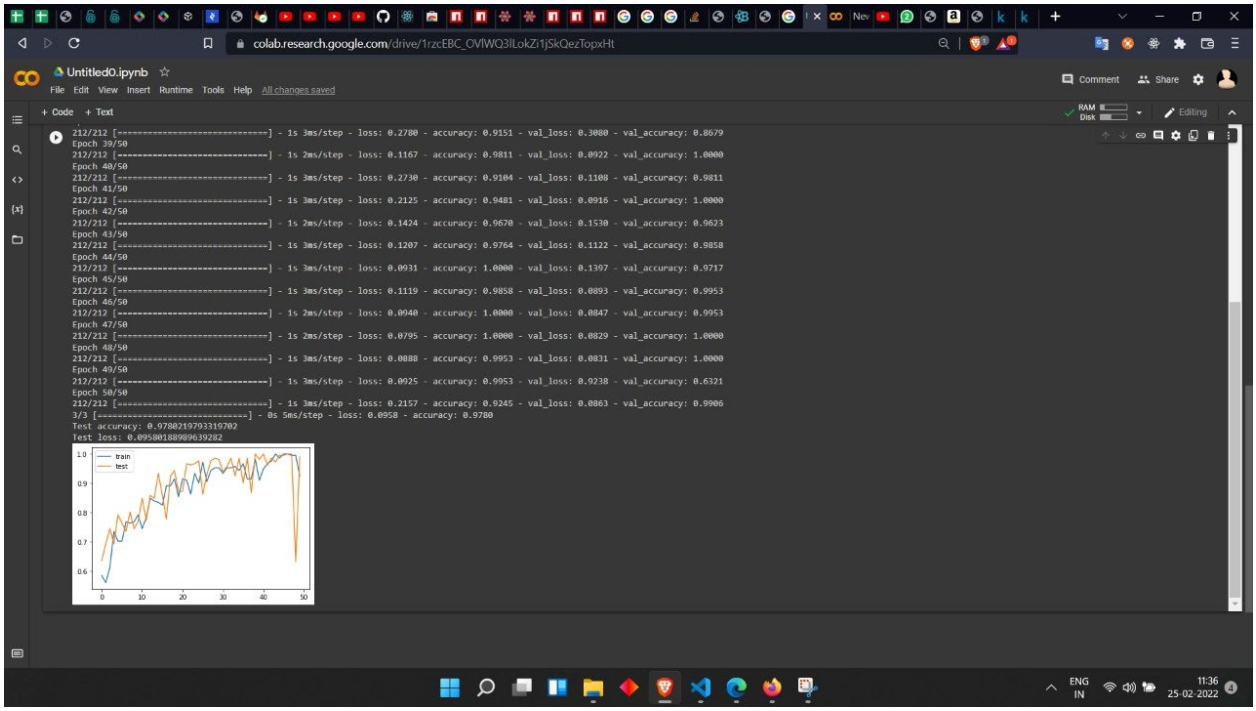
Implementing regularization to avoid overfitting in binary classification.

Code

```
from google.colab import files
uploaded=files.upload()
import io

#importing all the necessary libraries
import pandas as pd
import tensorflow as tf
from tensorflow import keras
from sklearn.model_selection import train_test_split
from keras.regularizers import l2
from matplotlib import pyplot

#Reading the Dataset
df=pd.read_csv(io.BytesIO(uploaded['heart.csv']))
#split into input (X) and output (y) variables
X = df.iloc[:, 0:14].values
y = df['target']

#Traing the dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)
#Creating a Neural Network
model = keras.Sequential([
   keras.layers.Flatten(input_shape=(14,)), #Input Layer
   keras.layers.Dense(16, activation=tf.nn.relu,kernel_regularizer=l2(0.001)), #Hidden Layer 1 with relu as
activation function
        keras.layers.Dense(16, activation=tf.nn.relu,kernel_regularizer=l2(0.001)), #Hidden Layer 2 with relu as
activation function
   keras.layers.Dense(1, activation=tf.nn.sigmoid), #Output Layer with sigmoid as activation function
])
#Compiling Neaural Network with Loss function as Cross-entropy
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
#Fitting the model
history=model.fit(X_train, y_train,validation_data=(X_train, y_train), epochs=50, batch_size=1)
#Calculating Model Accuracy and Loss
test_loss, test_acc = model.evaluate(X_test, y_test)
print('Test accuracy:', test_acc)
print('Test loss:', test_loss)
pyplot.plot(history.history['accuracy'],label='train')
pyplot.plot(history.history['val_accuracy'],label='test')
pyplot.legend()
pyplot.show()
```

Output


colab.research.google.com/drive/1rzcEBC_OVlWQ3lLokZi1jSkQezTopxHt
Untitled0.ipynb
File Edit View Insert Runtime Tools Help
+ Code + Text
Comment Share
RAM Disk
Editing
from google.colab import files
uploaded=files.upload()
import io
#importing all the necessary libraries
import pandas as pd
import tensorflow as tf
from tensorflow import keras
from sklearn.model_selection import train_test_split
from keras.regularizers import l2
from matplotlib import pyplot
#Reading the Dataset
df=pd.read_csv(io.BytesIO(uploaded['heart.csv']))
#split into input (X) and output (y) variables
X = df.iloc[:, 0:14].values
y = df['target']
#Traing the dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)
#Creating a Neural Network
model = keras.Sequential([
keras.layers.Flatten(input_shape=(14,)), #Input Layer
keras.layers.Dense(16, activation=tf.nn.relu,kernel_regularizer=l2(0.001)), #Hidden Layer 1 with relu as activation function
keras.layers.Dense(16, activation=tf.nn.relu,kernel_regularizer=l2(0.001)), #Hidden Layer 2 with relu as activation function
keras.layers.Dense(1, activation=tf.nn.sigmoid), #Output Layer with sigmoid as activation function
])
#Compiling Neaural Network with Loss function as Cross-entropy
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
#Fitting the model
history=model.fit(X_train, y_train,validation_data=(X_train, y_train), epochs=50, batch_size=1)
#Calculating Model Accuracy and Loss
test_loss, test_acc = model.evaluate(X_test, y_test)
print('Test accuracy:', test_acc)
print('Test loss:', test_loss)
pyplot.plot(history.history['accuracy'],label='train')
pyplot.plot(history.history['val_accuracy'],label='test')
pyplot.legend()
pyplot.show()
Widgets
ENG IN
11:35
25-02-2022


colab.research.google.com/drive/1rzcEBC_OVlWQ3lLokZi1jSkQezTopxHt
Untitled0.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
Comment Share
RAM Disk
Editing
212/212 [==========] - 1s 3ms/step - loss: 0.2780 - accuracy: 0.9151 - val_loss: 0.3080 - val_accuracy: 0.8679
Epoch 39/50
212/212 [==========] - 1s 2ms/step - loss: 0.1167 - accuracy: 0.9811 - val_loss: 0.0922 - val_accuracy: 1.0000
Epoch 40/50
212/212 [==========] - 1s 3ms/step - loss: 0.2730 - accuracy: 0.9104 - val_loss: 0.1108 - val_accuracy: 0.9811
Epoch 41/50
212/212 [==========] - 1s 3ms/step - loss: 0.2125 - accuracy: 0.9481 - val_loss: 0.0916 - val_accuracy: 1.0000
Epoch 42/50
212/212 [==========] - 1s 2ms/step - loss: 0.1424 - accuracy: 0.9670 - val_loss: 0.1530 - val_accuracy: 0.9623
Epoch 43/50
212/212 [==========] - 1s 3ms/step - loss: 0.1207 - accuracy: 0.9764 - val_loss: 0.1122 - val_accuracy: 0.9858
Epoch 44/50
212/212 [==========] - 1s 3ms/step - loss: 0.0931 - accuracy: 1.0000 - val_loss: 0.1397 - val_accuracy: 0.9717
Epoch 45/50
212/212 [==========] - 1s 3ms/step - loss: 0.1119 - accuracy: 0.9858 - val_loss: 0.0893 - val_accuracy: 0.9953
Epoch 46/50
212/212 [==========] - 1s 2ms/step - loss: 0.0940 - accuracy: 1.0000 - val_loss: 0.0847 - val_accuracy: 0.9953
Epoch 47/50
212/212 [==========] - 1s 2ms/step - loss: 0.0795 - accuracy: 1.0000 - val_loss: 0.0829 - val_accuracy: 1.0000
Epoch 48/50
212/212 [==========] - 1s 3ms/step - loss: 0.0888 - accuracy: 0.9953 - val_loss: 0.0831 - val_accuracy: 1.0000
Epoch 49/50
212/212 [==========] - 1s 3ms/step - loss: 0.0925 - accuracy: 0.9953 - val_loss: 0.9238 - val_accuracy: 0.6321
Epoch 50/50
212/212 [==========] - 1s 3ms/step - loss: 0.2157 - accuracy: 0.9245 - val_loss: 0.0863 - val_accuracy: 0.9906
3/3 [==========] - 0s 5ms/step - loss: 0.0958 - accuracy: 0.9780
Test accuracy: 0.9780219793319702
Test loss: 0.09580188989639282
train
test
ENG IN
11:36
25-02-2022

Practical 7

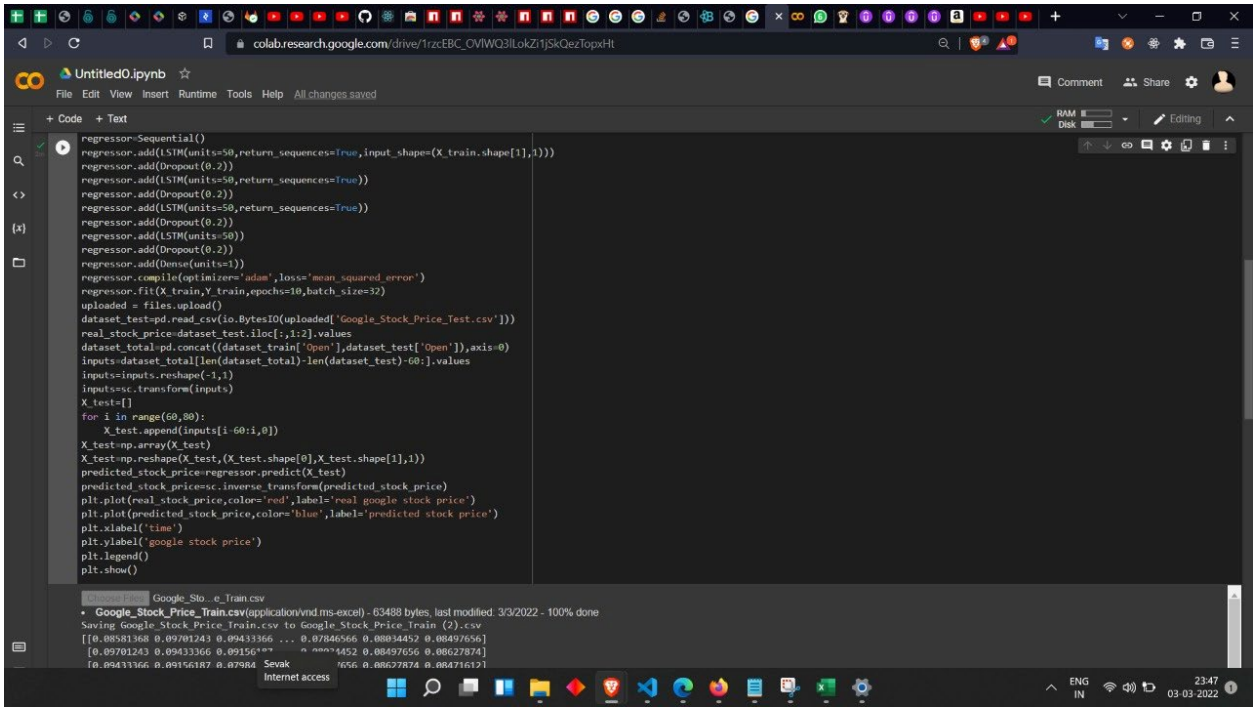
Demonstrate recurrent neural network that learns to perform sequence analysis for stock price.

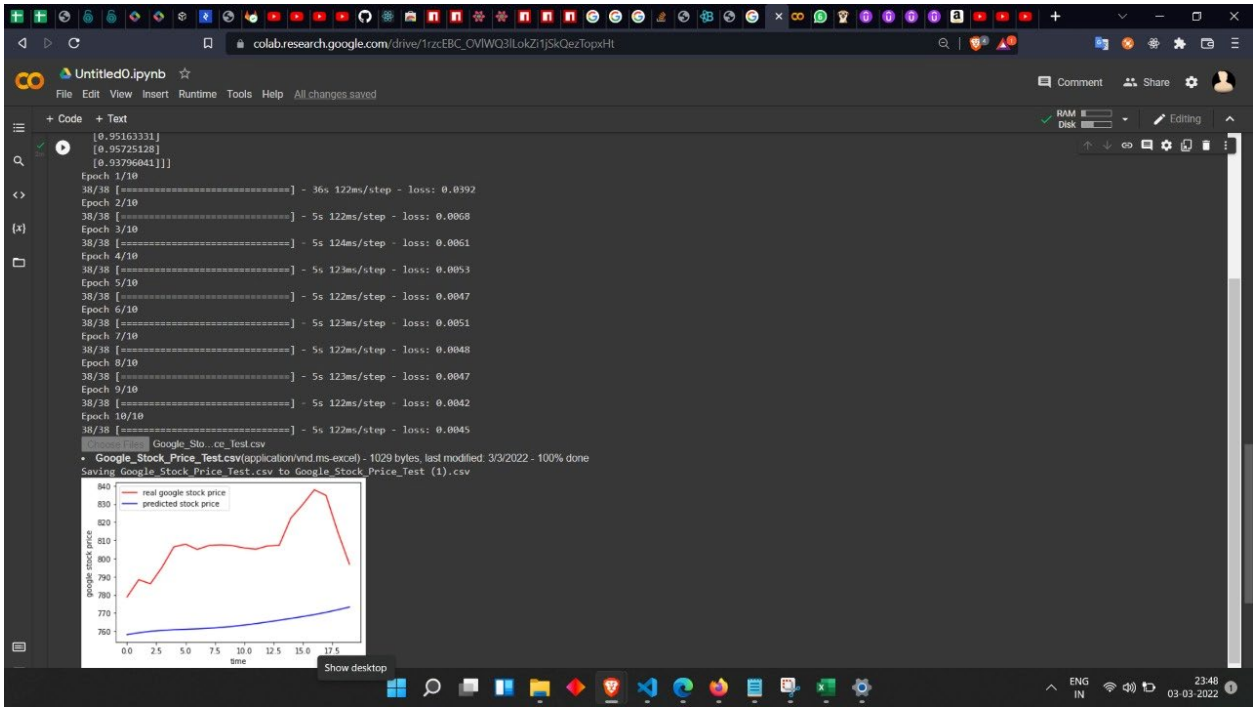
Code

```
# -*- coding: utf-8 -*-
"""
Created on Sat Feb 26 23:31:10 2022
@author: Ankit Patel
Sapid: 53004200018
"""
from google.colab import files
uploaded = files.upload()
import io
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from keras.layers import Dropout
from sklearn.preprocessing import MinMaxScaler
dataset_train=pd.read_csv(io.BytesIO(uploaded['Google_Stock_Price_Train.csv']))
#print(dataset_train)
training_set=dataset_train.iloc[:,1:2].values
#print(training_set)
sc=MinMaxScaler(feature_range=(0,1))
training_set_scaled=sc.fit_transform(training_set)
#print(training_set_scaled)
X_train=[]
Y_train=[]
for i in range(60,1258):
    X_train.append(training_set_scaled[i-60:i,0])
    Y_train.append(training_set_scaled[i,0])
X_train,Y_train=np.array(X_train),np.array(Y_train)
print(X_train)
print('***************')
print(Y_train)
X_train=np.reshape(X_train,(X_train.shape[0],X_train.shape[1],1))
print('***************')
print(X_train)
regressor=Sequential()
regressor.add(LSTM(units=50,return_sequences=True,input_shape=(X_train.shape[1],1)))
```

```
regressor.add(Dropout(0.2))
regressor.add(LSTM(units=50,return_sequences=True))
regressor.add(Dropout(0.2))
regressor.add(LSTM(units=50,return_sequences=True))
regressor.add(Dropout(0.2))
regressor.add(LSTM(units=50))
regressor.add(Dropout(0.2))
regressor.add(Dense(units=1))
regressor.compile(optimizer='adam',loss='mean_squared_error')
regressor.fit(X_train,Y_train,epochs=10,batch_size=32)
uploaded = files.upload()
dataset_test=pd.read_csv(io.BytesIO(uploaded['Google_Stock_Price_Test.csv']))
real_stock_price=dataset_test.iloc[:,1:2].values
dataset_total=pd.concat((dataset_train['Open'],dataset_test['Open']),axis=0)
inputs=dataset_total[len(dataset_total)-len(dataset_test)-60:].values
inputs=inputs.reshape(-1,1)
inputs=sc.transform(inputs)
X_test=[]
for i in range(60,80):
   X_test.append(inputs[i-60:i,0])
X_test=np.array(X_test)
X_test=np.reshape(X_test,(X_test.shape[0],X_test.shape[1],1))
predicted_stock_price=regressor.predict(X_test)
predicted_stock_price=sc.inverse_transform(predicted_stock_price)
plt.plot(real_stock_price,color='red',label='real google stock price')
plt.plot(predicted_stock_price,color='blue',label='predicted stock price')
plt.xlabel('time')
plt.ylabel('google stock price')
plt.legend()
plt.show()
```

Output


colab.research.google.com/drive/1rzcEBC_OVIWQ3ILokZi1jSkQezTopxHt
Untitled0.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
regressor=Sequential()
regressor.add(LSTM(units=50,return_sequences=True,input_shape=(X_train.shape[1],1)))
regressor.add(Dropout(0.2))
regressor.add(LSTM(units=50,return_sequences=True))
regressor.add(Dropout(0.2))
regressor.add(LSTM(units=50,return_sequences=True))
regressor.add(Dropout(0.2))
regressor.add(LSTM(units=50))
regressor.add(Dropout(0.2))
regressor.add(Dense(units=1))
regressor.compile(optimizer='adam',loss='mean_squared_error')
regressor.fit(X_train,Y_train,epochs=10,batch_size=32)
uploaded = files.upload()
dataset_test=pd.read_csv(io.BytesIO(uploaded['Google_Stock_Price_Test.csv']))
real_stock_price=dataset_test.iloc[:,1:2].values
dataset_total=pd.concat((dataset_train['Open'],dataset_test['Open']),axis=0)
inputs=dataset_total[len(dataset_total)-len(dataset_test)-60:].values
inputs=inputs.reshape(-1,1)
inputs=sc.transform(inputs)
X_test=[]
for i in range(60,80):
X_test.append(inputs[i-60:i,0])
X_test=np.array(X_test)
X_test=np.reshape(X_test,(X_test.shape[0],X_test.shape[1],1))
predicted_stock_price=regressor.predict(X_test)
predicted_stock_price=sc.inverse_transform(predicted_stock_price)
plt.plot(real_stock_price,color='red',label='real google stock price')
plt.plot(predicted_stock_price,color='blue',label='predicted stock price')
plt.xlabel('time')
plt.ylabel('google stock price')
plt.legend()
plt.show()
Choose Files Google_Sto...e_Train.csv
Google_Stock_Price_Train.csv(application/vnd.ms-excel) - 63488 bytes, last modified: 3/3/2022 - 100% done
Saving Google_Stock_Price_Train.csv to Google_Stock_Price_Train (2).csv
[[0.08581368 0.09701243 0.09433366 ... 0.07846566 0.08034452 0.08497656]
[0.09701243 0.09433366 0.09156187 ... 0.08034452 0.08497656 0.08627874]
[0.09433366 0.09156187 0.07984 ... 0.08497656 0.08627874 0.08471612]
Sevak
Internet access
ENG IN
23:47
03-03-2022


colab.research.google.com/drive/1rzcEBC_OVIWQ3ILokZi1jSkQezTopxHt
Untitled0.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
[0.95163331]
[0.95725128]
[0.93796041]]]
Epoch 1/10
38/38 [==============================] - 36s 122ms/step - loss: 0.0392
Epoch 2/10
38/38 [==============================] - 5s 122ms/step - loss: 0.0068
Epoch 3/10
38/38 [==============================] - 5s 124ms/step - loss: 0.0061
Epoch 4/10
38/38 [==============================] - 5s 123ms/step - loss: 0.0053
Epoch 5/10
38/38 [==============================] - 5s 122ms/step - loss: 0.0047
Epoch 6/10
38/38 [==============================] - 5s 123ms/step - loss: 0.0051
Epoch 7/10
38/38 [==============================] - 5s 122ms/step - loss: 0.0048
Epoch 8/10
38/38 [==============================] - 5s 123ms/step - loss: 0.0047
Epoch 9/10
38/38 [==============================] - 5s 122ms/step - loss: 0.0042
Epoch 10/10
38/38 [==============================] - 5s 122ms/step - loss: 0.0045
Choose Files Google_Sto...ce_Test.csv
Google_Stock_Price_Test.csv(application/vnd.ms-excel) - 1029 bytes, last modified: 3/3/2022 - 100% done
Saving Google_Stock_Price_Test.csv to Google_Stock_Price_Test (1).csv
real google stock price
predicted stock price
google stock price
time
Show desktop
ENG IN
23:48
03-03-2022

Practical 8

Performing encoding and decoding of images using deep autoencoder.

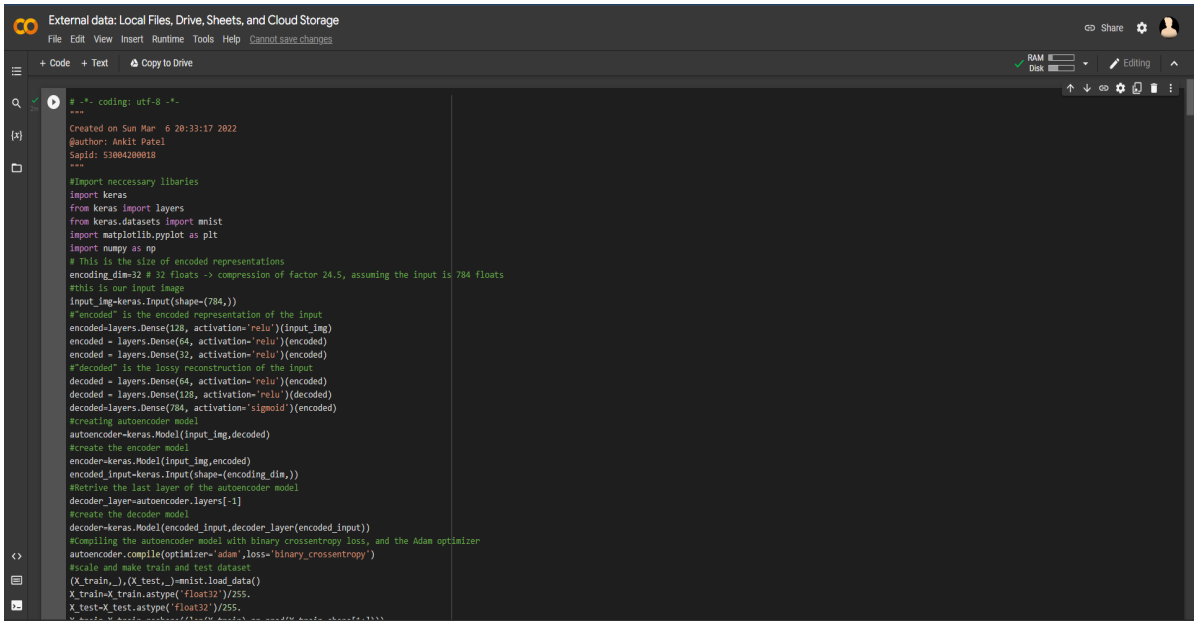
Code

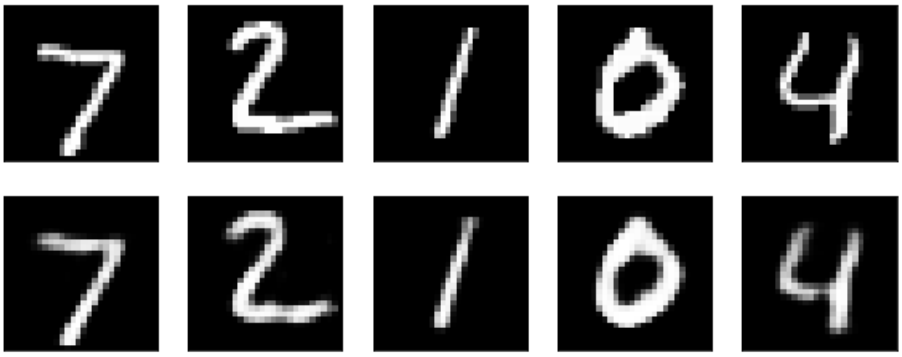
```
# -*- coding: utf-8 -*-
"""
Created on Sun Mar  6 20:33:17 2022
@author: Ankit Patel
Sapid: 53004200018
"""
#Import neccessary libaries
import keras
from keras import layers
from keras.datasets import mnist
import matplotlib.pyplot as plt
import numpy as np
# This is the size of encoded representations
encoding_dim=32 # 32 floats -> compression of factor 24.5, assuming the input is 784 floats
#this is our input image
input_img=keras.Input(shape=(784,))
#"encoded" is the encoded representation of the input
encoded=layers.Dense(128, activation='relu')(input_img)
encoded = layers.Dense(64, activation='relu')(encoded)
encoded = layers.Dense(32, activation='relu')(encoded)
#"decoded" is the lossy reconstruction of the input
decoded = layers.Dense(64, activation='relu')(encoded)
decoded = layers.Dense(128, activation='relu')(decoded)
decoded=layers.Dense(784, activation='sigmoid')(encoded)
#creating autoencoder model
autoencoder=keras.Model(input_img,decoded)
```

```
#create the encoder model
encoder=keras.Model(input_img,encoded)
encoded_input=keras.Input(shape=(encoding_dim,))
#Retrive the last layer of the autoencoder model
decoder_layer=autoencoder.layers[-1]
#create the decoder model
decoder=keras.Model(encoded_input,decoder_layer(encoded_input))
#Compiling the autoencoder model with binary crossentropy loss, and the Adam optimizer
autoencoder.compile(optimizer='adam',loss='binary_crossentropy')
#scale and make train and test dataset
(X_train,_),(X_test,_)=mnist.load_data()
X_train=X_train.astype('float32')/255.
X_test=X_test.astype('float32')/255.
X_train=X_train.reshape((len(X_train),np.prod(X_train.shape[1:])))
X_test=X_test.reshape((len(X_test),np.prod(X_test.shape[1:])))
print(X_train.shape)
print(X_test.shape)
#train autoencoder with training dataset with 50 epochs and using minibatch approach
autoencoder.fit(X_train,X_train,epochs=50,batch_size=256,shuffle=True,validation_data=(X_test,X_test))
# Encode and decode some digits
encoded_imgs=encoder.predict(X_test)
decoded_imgs=decoder.predict(encoded_imgs)
# How many digits we will display
n = 5
plt.figure(figsize=(10, 4))
for i in range(n):
    # Display original
    ax = plt.subplot(2, n, i + 1)
    plt.imshow(X_test[i].reshape(28, 28))
    plt.gray()
    ax.get_xaxis().set_visible(False)
    ax.get_yaxis().set_visible(False)
```

```
    # Display reconstruction
    ax = plt.subplot(2, n, i + 1 + n)
    plt.imshow(decoded_imgs[i].reshape(28, 28))
    plt.gray()
    ax.get_xaxis().set_visible(False)
    ax.get_yaxis().set_visible(False)
plt.show()
```

Output


External data: Local Files, Drive, Sheets, and Cloud Storage
File Edit View Insert Runtime Tools Help Cannot save changes
Share
+ Code + Text Copy to Drive
RAM Disk
Editing
-*- coding: utf-8 -*-
"""
Created on Sun Mar 6 20:33:17 2022
@author: Ankit Patel
Sapid: 53004200018
"""
#Import neccessary libaries
import keras
from keras import layers
from keras.datasets import mnist
import matplotlib.pyplot as plt
import numpy as np
# This is the size of encoded representations
encoding_dim=32 # 32 floats -> compression of factor 24.5, assuming the input is 784 floats
#this is our input image
input_img=keras.Input(shape=(784,))
#"encoded" is the encoded representation of the input
encoded=layers.Dense(128, activation='relu')(input_img)
encoded = layers.Dense(64, activation='relu')(encoded)
encoded = layers.Dense(32, activation='relu')(encoded)
#"decoded" is the lossy reconstruction of the input
decoded = layers.Dense(64, activation='relu')(encoded)
decoded = layers.Dense(128, activation='relu')(decoded)
decoded=layers.Dense(784, activation='sigmoid')(encoded)
#creating autoencoder model
autoencoder=keras.Model(input_img,decoded)
#create the encoder model
encoder=keras.Model(input_img,encoded)
encoded_input=keras.Input(shape=(encoding_dim,))
#Retrive the last layer of the autoencoder model
decoder_layer=autoencoder.layers[-1]
#create the decoder model
decoder=keras.Model(encoded_input,decoder_layer(encoded_input))
#Compiling the autoencoder model with binary crossentropy loss, and the Adam optimizer
autoencoder.compile(optimizer='adam',loss='binary_crossentropy')
#scale and make train and test dataset
(X_train,_),(X_test,_)=mnist.load_data()
X_train=X_train.astype('float32')/255.
X_test=X_test.astype('float32')/255.



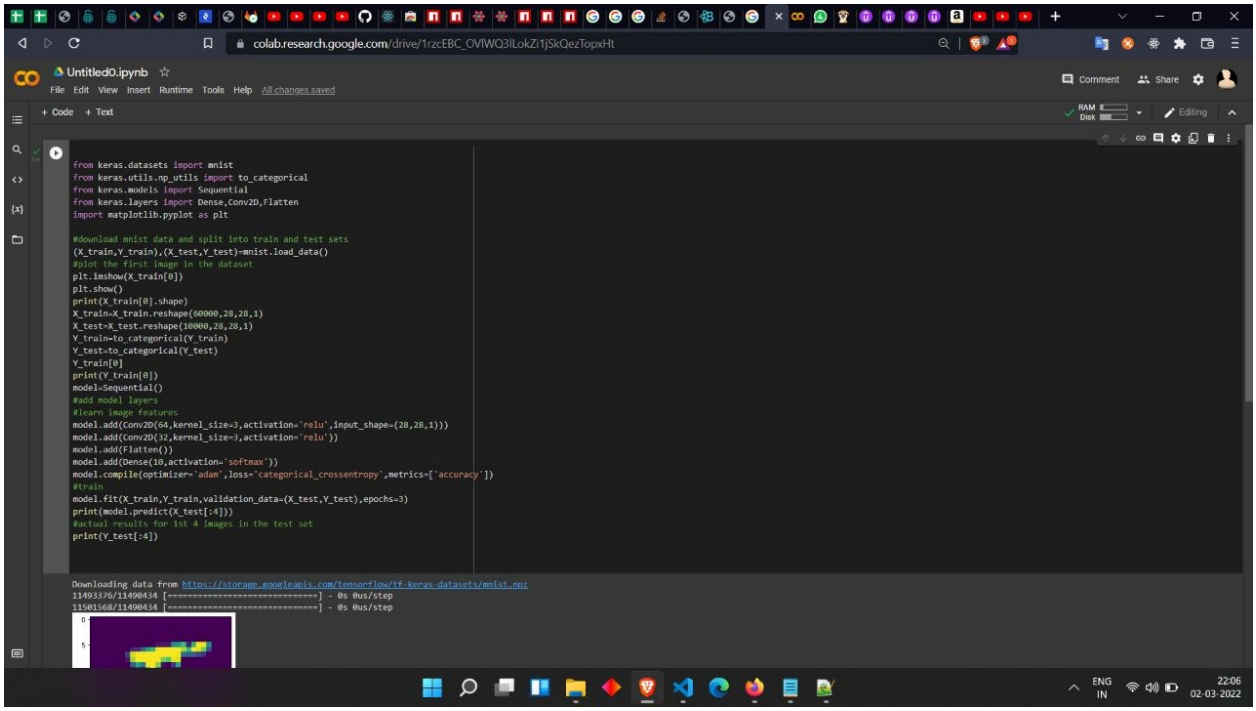
Practical 9

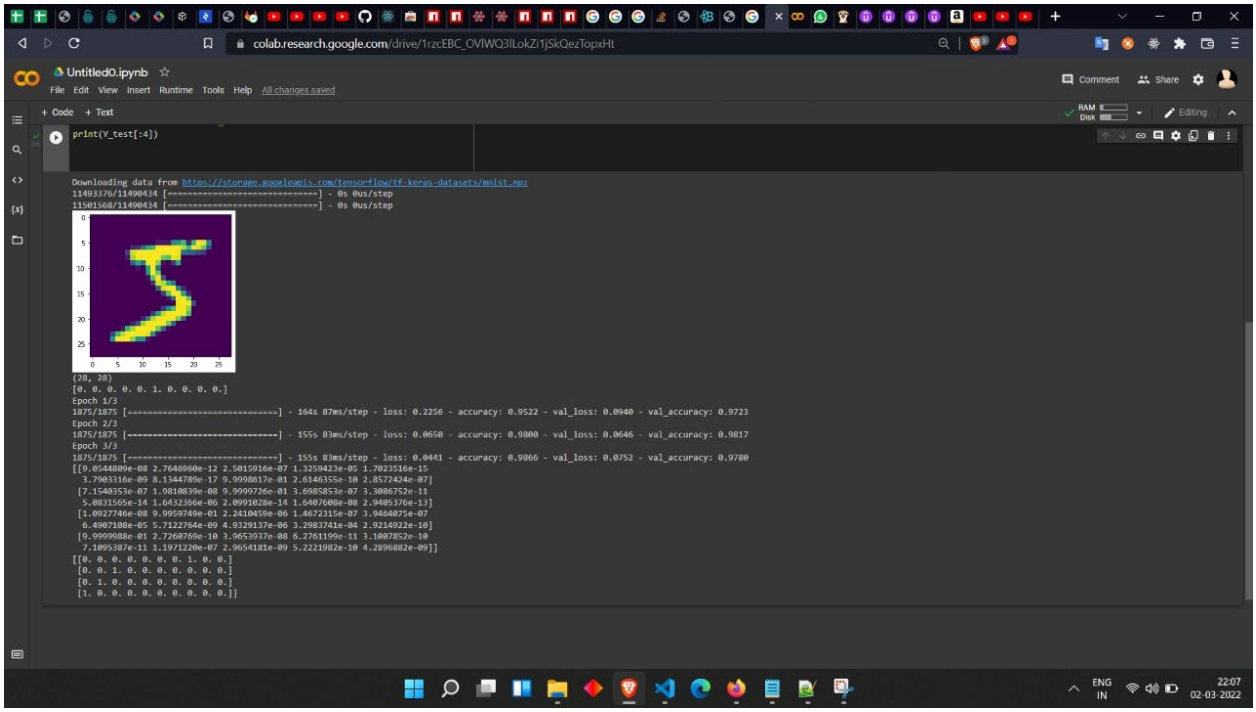
Implementing convolutional neural network for Modified National Institute of Standards and Technology (MNIST) dataset.

Code

```
# -*- coding: utf-8 -*-
"""
Created on Sun Feb 20 01:47:01 2022
@author: Ankit Patel
Sapid: 53004200018
"""

from keras.datasets import mnist
from keras.utils.np_utils import to_categorical
from keras.models import Sequential
from keras.layers import Dense,Conv2D,Flatten
import matplotlib.pyplot as plt

#download mnist data and split into train and test sets
(X_train,Y_train),(X_test,Y_test)=mnist.load_data()
#plot the first image in the dataset
plt.imshow(X_train[0])
plt.show()
print(X_train[0].shape)
X_train=X_train.reshape(60000,28,28,1)
X_test=X_test.reshape(10000,28,28,1)
Y_train=to_categorical(Y_train)
Y_test=to_categorical(Y_test)
Y_train[0]
print(Y_train[0])
model=Sequential()
#add model layers
#learn image features
model.add(Conv2D(64,kernel_size=3,activation='relu',input_shape=(28,28,1)))
model.add(Conv2D(32,kernel_size=3,activation='relu'))
model.add(Flatten())
model.add(Dense(10,activation='softmax'))
model.compile(optimizer='adam',loss='categorical_crossentropy',metrics=['accuracy'])
#train
model.fit(X_train,Y_train,validation_data=(X_test,Y_test),epochs=3)
print(model.predict(X_test[:4]))
#actual results for 1st 4 images in the test set
print(Y_test[:4])
```

Output


colab.research.google.com/drive/1rzcEBC_OVlWQ3lLokZi1jSkQezTopxHt
Untitled0.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
from keras.datasets import mnist
from keras.utils.np_utils import to_categorical
from keras.models import Sequential
from keras.layers import Dense,Conv2D,Flatten
import matplotlib.pyplot as plt
#download mnist data and split into train and test sets
(X_train,Y_train),(X_test,Y_test)=mnist.load_data()
#plot the first image in the dataset
plt.imshow(X_train[0])
plt.show()
print(X_train[0].shape)
X_train=X_train.reshape(60000,28,28,1)
X_test=X_test.reshape(10000,28,28,1)
Y_train=to_categorical(Y_train)
Y_test=to_categorical(Y_test)
Y_train[0]
print(Y_train[0])
model=Sequential()
#add model layers
#learn image features
model.add(Conv2D(64,kernel_size=3,activation='relu',input_shape=(28,28,1)))
model.add(Conv2D(32,kernel_size=3,activation='relu'))
model.add(Flatten())
model.add(Dense(10,activation='softmax'))
model.compile(optimizer="adam",loss="categorical_crossentropy",metrics=['accuracy'])
#train
model.fit(X_train,Y_train,validation_data=(X_test,Y_test),epochs=3)
print(model.predict(X_test[:4]))
#actual results for 1st 4 images in the test set
print(Y_test[:4])
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11493376/11490434 [==============================] - 0s 0us/step
11501568/11490434 [==============================] - 0s 0us/step
Comment Share
RAM Disk
Editing
ENG IN
22:06
02-03-2022


colab.research.google.com/drive/1rzcEBC_OVlWQ3lLokZi1jSkQezTopxHt
Untitled0.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
print(Y_test[:4])
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11493376/11490434 [==============================] - 0s 0us/step
11501568/11490434 [==============================] - 0s 0us/step
(28, 28)
[0. 0. 0. 0. 0. 1. 0. 0. 0. 0.]
Epoch 1/3
1875/1875 [==============================] - 164s 87ms/step - loss: 0.2256 - accuracy: 0.9522 - val_loss: 0.0940 - val_accuracy: 0.9723
Epoch 2/3
1875/1875 [==============================] - 155s 83ms/step - loss: 0.0650 - accuracy: 0.9800 - val_loss: 0.0646 - val_accuracy: 0.9817
Epoch 3/3
1875/1875 [==============================] - 155s 83ms/step - loss: 0.0441 - accuracy: 0.9866 - val_loss: 0.0752 - val_accuracy: 0.9780
[[9.8544809e-08 2.7646960e-12 2.5015916e-07 1.3259423e-05 1.7023516e-15
3.7903316e-09 8.1344789e-17 9.9998617e-01 2.6146355e-10 2.8572424e-07]
[7.1540353e-07 1.9810839e-08 9.9999726e-01 3.6985853e-07 3.3086752e-11
5.0831565e-14 1.6432366e-06 2.0991028e-14 1.6407608e-08 2.9405376e-13]
[1.0927746e-08 9.9959749e-01 2.2410459e-06 1.4672315e-07 3.9464075e-07
6.4987188e-05 5.7122764e-09 4.8329137e-06 3.2983741e-04 2.9214922e-10]
[9.9999988e-01 2.7260769e-10 3.9653937e-08 6.2761199e-11 3.1087852e-10
7.1095387e-11 1.1971220e-07 2.9654181e-09 5.2221982e-10 4.2896882e-09]]
[[0. 0. 0. 0. 0. 0. 0. 1. 0. 0.]
[0. 0. 1. 0. 0. 0. 0. 0. 0. 0.]
[0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
[1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]]
Comment Share
RAM Disk
Editing
ENG IN
22:07
02-03-2022

Practical 10

Denoising of images using autoencoder.

Code

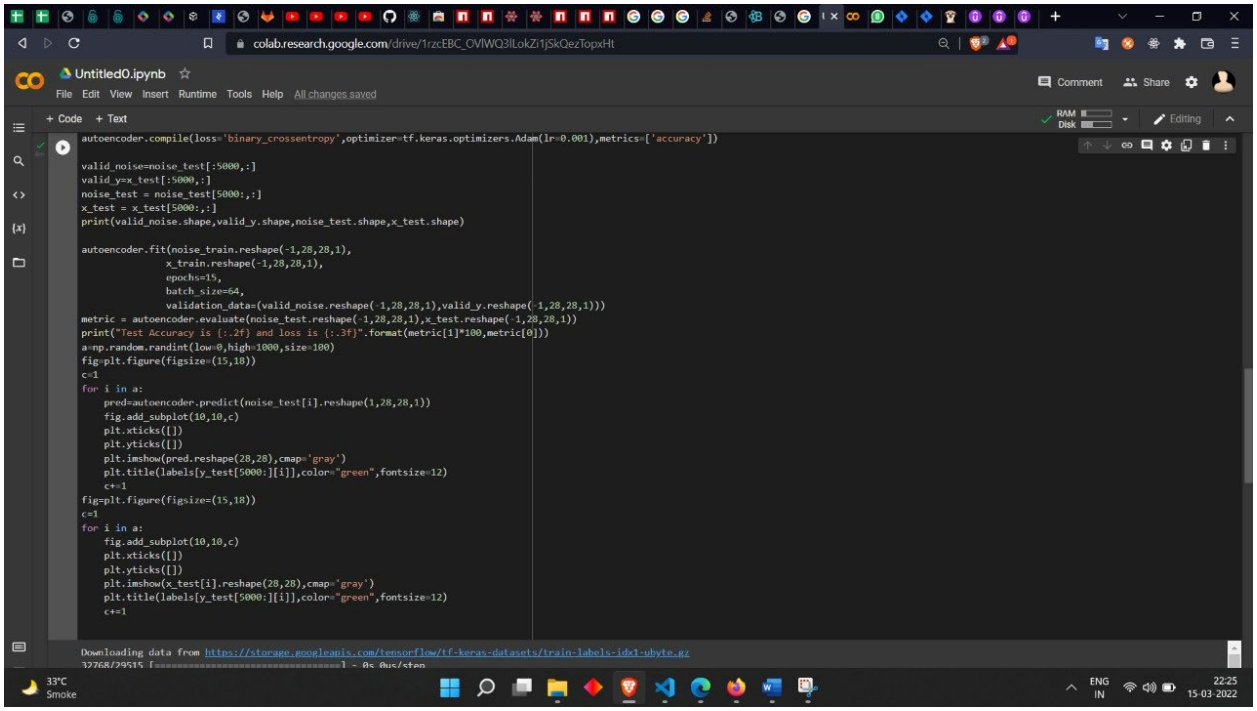
```
# -*- coding: utf-8 -*-
"""
Created on Mon Mar  7 00:42:04 2022
@author: Ankit Patel
Sapid: 53004200018
"""
import tensorflow as tf
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import random

(x_train,y_train),(x_test,y_test)= tf.keras.datasets.fashion_mnist.load_data()
print(x_train.shape)
print(y_train.shape)
print(x_test.shape)
print(y_test.shape)

labels={0:"T-Shirt",1:"Trousers",2:"Pullover",3:"Dress",4:"Coat",5:"Sandal",6:"T-
Shirt",7:"Sneakers",8:"Bag",9:"Ankle Boot"}

a=np.random.randint(low=0,high=1000,size=100)
fig=plt.figure(figsize=(15,18))
c=1
for i in a:
    fig.add_subplot(10,10,c)
    plt.xticks([])
```

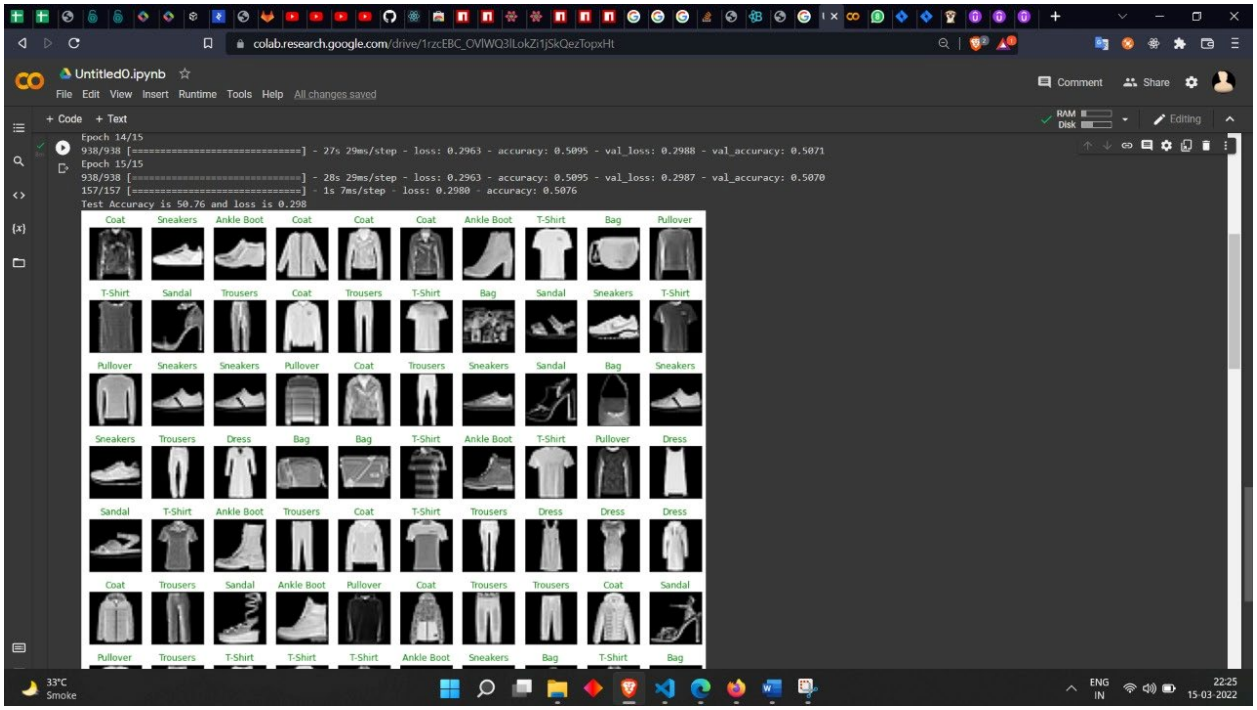
```
    plt.yticks([])
    plt.imshow(x_train[i],cmap='gray')
    plt.title(labels[y_train[i]],color="green",fontsize=12)
    c+=1

x_train=x_train/255.
x_test=x_test/255.

noise_factor=0.3
noise_train=[]

for img in x_train:
    noisy_img = img + noise_factor* np.random.randn(*img.shape)
    noisy_img = np.clip(noisy_img , 0, 1)
    noise_train.append(noisy_img)

a=np.random.randint(low=0,high=1000,size=50)
fig=plt.figure(figsize=(15,8))
c=1
for i in a:
    fig.add_subplot(5,10,c)
    plt.xticks([])
    plt.yticks([])
    plt.imshow(noise_train[i],cmap='gray')
    plt.title(labels[y_train[i]],color="green",fontsize=12)
    c+=1

noise_factor=0.3
noise_test=[]

for img in x_test:
    noisy_img = img + noise_factor* np.random.randn(*img.shape)
```

```
    noisy_img = np.clip(noisy_img , 0, 1)
    noise_test.append(noisy_img)

a=np.random.randint(low=0,high=1000,size=25)
fig=plt.figure(figsize=(7,8))
c=1

for i in a:
    fig.add_subplot(5,5,c)
    plt.xticks([])
    plt.yticks([])
    plt.imshow(noise_test[i],cmap='gray')
    plt.title(labels[y_test[i]],color="green",fontsize=12)
    c+=1

noise_test=np.array(noise_test)
noise_train=np.array(noise_train)

autoencoder= tf.keras.Sequential([
    tf.keras.layers.Conv2D(32,(3,3),strides=2,padding='same',input_shape=(28,28,1)),
    tf.keras.layers.Conv2D(16,(3,3),strides=2,padding='same'),
    tf.keras.layers.Conv2D(16,(3,3),strides=1,padding='same'),
    tf.keras.layers.Conv2DTranspose(32,(3,3),strides=2,padding='same'),
    tf.keras.layers.Conv2DTranspose(1,(3,3),strides=2,padding='same',activation='sigmoid'),
])
autoencoder.summary()

autoencoder.compile(loss='binary_crossentropy',optimizer=tf.keras.optimizers.Adam(lr=0.001),metrics=['accurac
y'])

valid_noise=noise_test[:5000,:]
valid_y=x_test[:5000,:]
noise_test = noise_test[5000:,:]
```

```
x_test = x_test[5000:,:]
print(valid_noise.shape,valid_y.shape,noise_test.shape,x_test.shape)


autoencoder.fit(noise_train.reshape(-1,28,28,1),
                x_train.reshape(-1,28,28,1),
                epochs=15,
                batch_size=64,
                validation_data=(valid_noise.reshape(-1,28,28,1),valid_y.reshape(-1,28,28,1)))
metric = autoencoder.evaluate(noise_test.reshape(-1,28,28,1),x_test.reshape(-1,28,28,1))
print("Test Accuracy is {:.2f} and loss is {:.3f}".format(metric[1]*100,metric[0]))
a=np.random.randint(low=0,high=1000,size=100)
fig=plt.figure(figsize=(15,18))
c=1
for i in a:
    pred=autoencoder.predict(noise_test[i].reshape(1,28,28,1))
    fig.add_subplot(10,10,c)
    plt.xticks([])
    plt.yticks([])
    plt.imshow(pred.reshape(28,28),cmap='gray')
    plt.title(labels[y_test[5000:][i]],color="green",fontsize=12)
    c+=1
fig=plt.figure(figsize=(15,18))
c=1
for i in a:
    fig.add_subplot(10,10,c)
    plt.xticks([])
    plt.yticks([])
    plt.imshow(x_test[i].reshape(28,28),cmap='gray')
    plt.title(labels[y_test[5000:][i]],color="green",fontsize=12)
    c+=1
```

Output


colab.research.google.com/drive/1rzcEBC_OVlWQ3lLokZi1jSkQezTopxHt
UntitledO.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
+ Code + Text
RAM Disk
Editing
autoencoder.compile(loss='binary_crossentropy',optimizer=tf.keras.optimizers.Adam(lr=0.001),metrics=['accuracy'])
valid_noise=noise_test[:5000,:]
valid_y=x_test[:5000,:]
noise_test = noise_test[5000:,:]
x_test = x_test[5000:,:]
print(valid_noise.shape,valid_y.shape,noise_test.shape,x_test.shape)
autoencoder.fit(noise_train.reshape(-1,28,28,1),
x_train.reshape(-1,28,28,1),
epochs=15,
batch_size=64,
validation_data=(valid_noise.reshape(-1,28,28,1),valid_y.reshape(-1,28,28,1)))
metric = autoencoder.evaluate(noise_test.reshape(-1,28,28,1),x_test.reshape(-1,28,28,1))
print("Test Accuracy is {:.2f} and loss is {:.3f}".format(metric[1]*100,metric[0]))
a=np.random.randint(low=0,high=1000,size=100)
fig=plt.figure(figsize=(15,18))
c=1
for i in a:
pred=autoencoder.predict(noise_test[i].reshape(1,28,28,1))
fig.add_subplot(10,10,c)
plt.xticks([])
plt.yticks([])
plt.imshow(pred.reshape(28,28),cmap='gray')
plt.title(labels[y_test[5000:][i]],color="green",fontsize=12)
c+=1
fig=plt.figure(figsize=(15,18))
c=1
for i in a:
fig.add_subplot(10,10,c)
plt.xticks([])
plt.yticks([])
plt.imshow(x_test[i].reshape(28,28),cmap='gray')
plt.title(labels[y_test[5000:][i]],color="green",fontsize=12)
c+=1
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-labels-idx1-ubyte.gz
33°C Smoke
ENG IN
22:25 15-03-2022


colab.research.google.com/drive/1rzcEBC_OVlWQ3lLokZi1jSkQezTopxHt
UntitledO.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
+ Code + Text
RAM Disk
Editing
Epoch 14/15
938/938 [==========] - 27s 29ms/step - loss: 0.2963 - accuracy: 0.5095 - val_loss: 0.2988 - val_accuracy: 0.5071
Epoch 15/15
938/938 [==========] - 28s 29ms/step - loss: 0.2963 - accuracy: 0.5095 - val_loss: 0.2987 - val_accuracy: 0.5070
157/157 [==========] - 1s 7ms/step - loss: 0.2980 - accuracy: 0.5076
Test Accuracy is 50.76 and loss is 0.298
33°C Smoke
ENG IN
22:25 15-03-2022

Practical 11

RBM/DBM/DBN Recommendation System

Code

```
# -*- coding: utf-8 -*-
"""
Created on Tue Mar 29 13:20:17 2022
@author: Ankit Patel
Sapid: 53004200018
"""
#importing all the neccassary libraries
import tensorflow.compat.v1 as tf
tf.disable_v2_behavior()
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# Load the movies dataset and also pass header=None since files don't contain any headers
movies_df = pd.read_csv('D:/Practical/DL/Datasets/movies.dat', sep='::', header=None, engine='python')
print(movies_df.head())

# Load the ratings dataset
ratings_df = pd.read_csv('D:/Practical/DL/Datasets/ratings.dat', sep='::', header=None, engine='python')
print(ratings_df.head())

# Lets rename our columns in these data frames so we can convey their data better
movies_df.columns = ['MovieID', 'Title', 'Genres']
ratings_df.columns = ['UserID', 'MovieID', 'Rating', 'Timestamp']

# Verify the changes done to the dataframes
print(movies_df.head())
print(ratings_df.head())
```

```
# Data Correction and Formatting
print('The Number of Movies in Dataset', len(movies_df))

"""
- Our Movie ID's vary from 1 to 3952 while we have 3883 movies.
- Due to this, we won't be able to index movies through their ID since we would get memory indexing errors.
- To amend we can create a column that shows the spot in our list that particular movie is in:
"""

movies_df['List Index'] = movies_df.index
print(movies_df.head())

# Merge movies_df with ratings_df by MovieID
merged_df = movies_df.merge(ratings_df, on='MovieID')

# Drop unnecessary columns
merged_df = merged_df.drop('Timestamp', axis=1).drop('Title', axis=1).drop('Genres', axis=1)

# Display the result
print(merged_df.head())

# Lets Group up the Users by their user ID's
user_Group = merged_df.groupby('UserID')
print(user_Group.head())

"""
Formatting the data into input for the RBM.
Store the normalized users ratings into a list of lists called trX.
"""

# Amount of users used for training
```

```
amountOfUsedUsers = 1000

# Creating the training list
trX = []

# For each user in the group
for userID, curUser in user_Group:

    # Create a temp that stores every movie's rating
    temp = [0]*len(movies_df)

    # For each movie in curUser's movie list
    for num, movie in curUser.iterrows():

        # Divide the rating by 5 and store it
        temp[movie['List Index']] = movie['Rating']/5.0

    # Add the list of ratings into the training list
    trX.append(temp)

    # Check to see if we finished adding in the amount of users for training
    if amountOfUsedUsers == 0:
        break
    amountOfUsedUsers -= 1
print(trX)

# Setting the models Parameters
hiddenUnits = 50
visibleUnits = len(movies_df)
vb = tf.placeholder(tf.float32, [visibleUnits])  # Number of unique movies
hb = tf.placeholder(tf.float32, [hiddenUnits])  # Number of features were going to learn
W = tf.placeholder(tf.float32, [visibleUnits, hiddenUnits])  # Weight Matrix
```

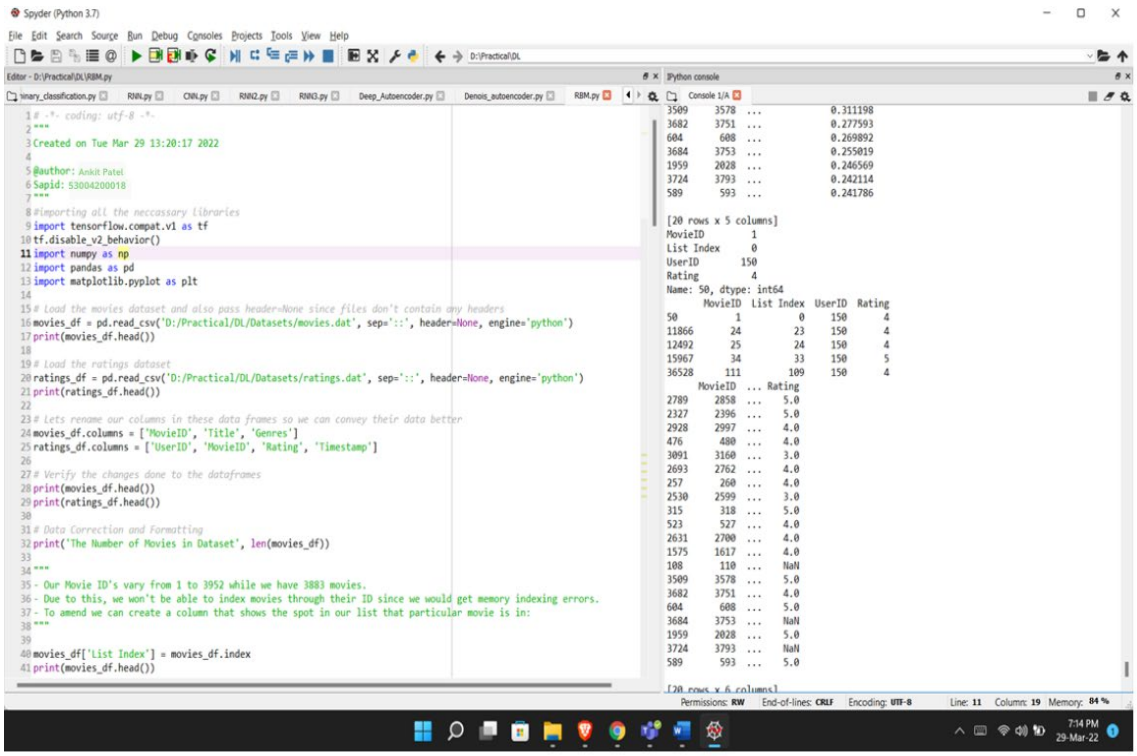
```
# Phase 1: Input Processing
v0 = tf.placeholder("float", [None, visibleUnits])
_h0 = tf.nn.sigmoid(tf.matmul(v0, W) + hb)  # Visible layer activation
h0 = tf.nn.relu(tf.sign(_h0 - tf.random_uniform(tf.shape(_h0))))  # Gibb's Sampling

# Phase 2: Reconstruction
_v1 = tf.nn.sigmoid(tf.matmul(h0, tf.transpose(W)) + vb)  # Hidden layer activation
v1 = tf.nn.relu(tf.sign(_v1 - tf.random_uniform(tf.shape(_v1))))
h1 = tf.nn.sigmoid(tf.matmul(v1, W) + hb)

""" Set RBM Training Parameters """

# Learning rate
alpha = 1.0

# Create the gradients
w_pos_grad = tf.matmul(tf.transpose(v0), h0)
w_neg_grad = tf.matmul(tf.transpose(v1), h1)

# Calculate the Contrastive Divergence to maximize
CD = (w_pos_grad - w_neg_grad) / tf.to_float(tf.shape(v0)[0])

# Create methods to update the weights and biases
update_w = W + alpha * CD
update_vb = vb + alpha * tf.reduce_mean(v0 - v1, 0)
update_hb = hb + alpha * tf.reduce_mean(h0 - h1, 0)

# Set the error function, here we use Mean Absolute Error Function
err = v0 - v1
err_sum = tf.reduce_mean(err*err)
```

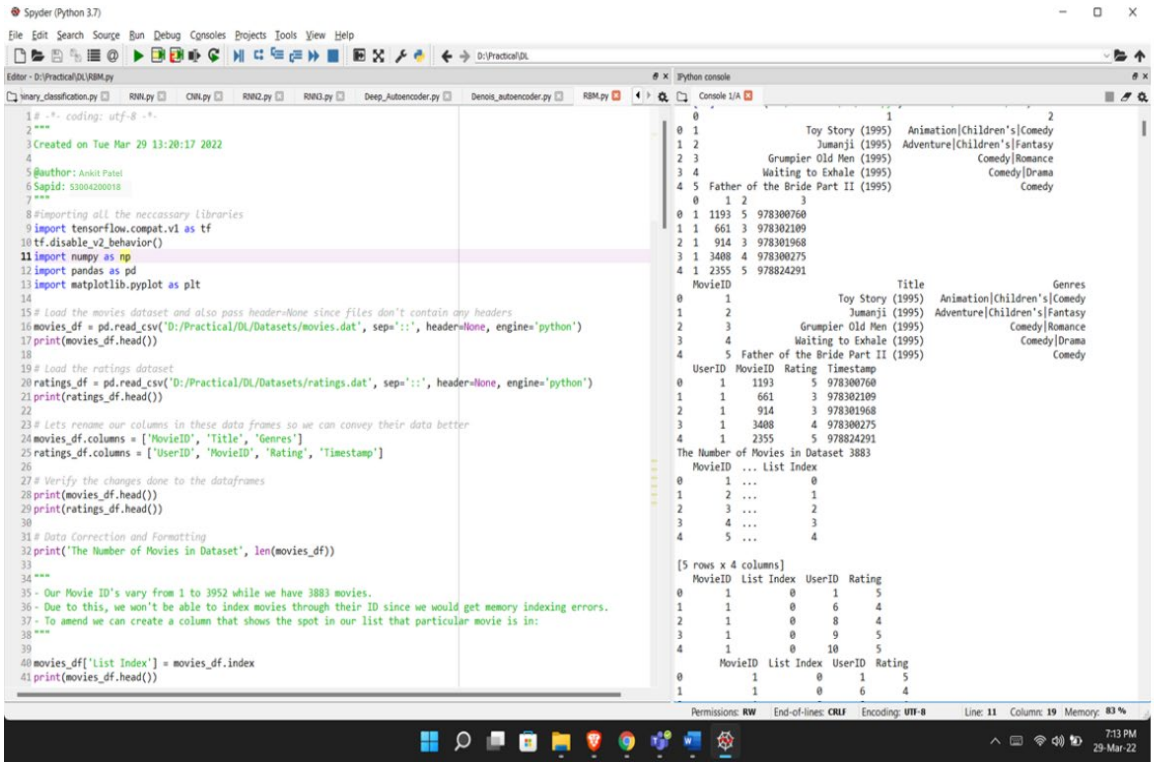
```
""" Initialize our Variables with Zeroes using Numpy Library """

# Current weight
cur_w = np.zeros([visibleUnits, hiddenUnits], np.float32)

# Current visible unit biases
cur_vb = np.zeros([visibleUnits], np.float32)

# Current hidden unit biases
cur_hb = np.zeros([hiddenUnits], np.float32)

# Previous weight
prv_w = np.zeros([visibleUnits, hiddenUnits], np.float32)

# Previous visible unit biases
prv_vb = np.zeros([visibleUnits], np.float32)

# Previous hidden unit biases
prv_hb = np.zeros([hiddenUnits], np.float32)
sess = tf.Session()
sess.run(tf.global_variables_initializer())

# Train RBM with 15 Epochs, with Each Epoch using 10 batches with size 100, After training print out the error by epoch
epochs = 15
batchsize = 100
errors = []
for i in range(epochs):
    for start, end in zip(range(0, len(trX), batchsize), range(batchsize, len(trX), batchsize)):
        batch = trX[start:end]
        cur_w = sess.run(update_w, feed_dict={v0: batch, W: prv_w, vb: prv_vb, hb: prv_hb})
        cur_vb = sess.run(update_vb, feed_dict={v0: batch, W: prv_w, vb: prv_vb, hb: prv_hb})
        cur_hb = sess.run(update_hb, feed_dict={v0: batch, W: prv_w, vb: prv_vb, hb: prv_hb})
```

```
        prv_w = cur_w
        prv_vb = cur_vb
        prv_hb = cur_hb
    errors.append(sess.run(err_sum, feed_dict={v0: trX, W: cur_w, vb: cur_vb, hb: cur_hb}))
    print(errors[-1])
plt.plot(errors)
plt.ylabel('Error')
plt.xlabel('Epoch')
plt.show()

"""
Recommendation System :-
- We can now predict movies that an arbitrarily selected user might like.
- This can be accomplished by feeding in the user's watched movie preferences into the RBM and then reconstructing the
  input.
- The values that the RBM gives us will attempt to estimate the user's preferences for movies that he hasn't watched
  based on the preferences of the users that the RBM was trained on.
"""

# Select the input User
inputUser = [trX[50]]

# Feeding in the User and Reconstructing the input
hh0 = tf.nn.sigmoid(tf.matmul(v0, W) + hb)
vv1 = tf.nn.sigmoid(tf.matmul(hh0, tf.transpose(W)) + vb)
feed = sess.run(hh0, feed_dict={v0: inputUser, W: prv_w, hb: prv_hb})
rec = sess.run(vv1, feed_dict={hh0: feed, W: prv_w, vb: prv_vb})

# List the 20 most recommended movies for our mock user by sorting it by their scores given by our model.
scored_movies_df_50 = movies_df
scored_movies_df_50["Recommendation Score"] = rec[0]
```

```
print(scored_movies_df_50.sort_values(["Recommendation Score"], ascending=False).head(20))

""" Recommend User what movies he has not watched yet """

# Find the mock user's UserID from the data
print(merged_df.iloc[50])  # Result you get is UserID 150

# Find all movies the mock user has watched before
movies_df_50 = merged_df[merged_df['UserID'] == 150]
print(movies_df_50.head())

""" Merge all movies that our mock users has watched with predicted scores based on his historical data: """

# Merging movies_df with ratings_df by MovieID
merged_df_50 = scored_movies_df_50.merge(movies_df_50, on='MovieID', how='outer')

# Dropping unnecessary columns
merged_df_50 = merged_df_50.drop('List Index_y', axis=1).drop('UserID', axis=1)

# Sort and take a look at first 20 rows
print(merged_df_50.sort_values(['Recommendation Score'], ascending=False).head(20))
```

Output


Spyder (Python 3.7)
File Edit Search Source Run Debug Consoles Projects Tools View Help
Editor - D:\Practical\DL\RBM.py
IPython console
Console 1/A
-*- coding: utf-8 -*-
"""
Created on Tue Mar 29 13:20:17 2022
@author: Ankit Patel
Sapid: 53004200018
"""
#Importing all the neccassary libraries
import tensorflow.compat.v1 as tf
tf.disable_v2_behavior()
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
# Load the movies dataset and also pass header=None since files don't contain any headers
movies_df = pd.read_csv('D:/Practical/DL/Datasets/movies.dat', sep='::', header=None, engine='python')
print(movies_df.head())
# Load the ratings dataset
ratings_df = pd.read_csv('D:/Practical/DL/Datasets/ratings.dat', sep='::', header=None, engine='python')
print(ratings_df.head())
# Lets rename our columns in these data frames so we can convey their data better
movies_df.columns = ['MovieID', 'Title', 'Genres']
ratings_df.columns = ['UserID', 'MovieID', 'Rating', 'Timestamp']
# Verify the changes done to the dataframes
print(movies_df.head())
print(ratings_df.head())
# Data Correction and Formatting
print('The Number of Movies in Dataset', len(movies_df))
"""
- Our Movie ID's vary from 1 to 3952 while we have 3883 movies.
- Due to this, we won't be able to index movies through their ID since we would get memory indexing errors.
- To amend we can create a column that shows the spot in our list that particular movie is in:
"""
movies_df['List Index'] = movies_df.index
print(movies_df.head())
Permissions: RW End-of-lines: CRLF Encoding: UTF-8 Line: 11 Column: 19 Memory: 84 %
7:14 PM
29-Mar-22


Spyder (Python 3.7)
File Edit Search Source Run Debug Consoles Projects Tools View Help
Editor - D:\Practical\DL\RBM.py
IPython console
Console 1/A
The Number of Movies in Dataset 3883
Permissions: RW End-of-lines: CRLF Encoding: UTF-8 Line: 11 Column: 19 Memory: 83 %
7:13 PM
29-Mar-22