****

Faculty of Computers and Artificial Intelligence

Computer Science Department

2021/2022

**CS 395 Selected Topics in CS-1**

**Research Project**

Report Submitted for Fulfillment of the Requirements and ILO’s for Selected Topics in CS-1 course for Fall 2021

Team No. 24

|  |  |  |  |
| --- | --- | --- | --- |
|  | ID | Name | Grade |
|  | 201900759 | محمود اسلام محمد يوسف |  |
|  | 201900353 | سمير احمد محمد |  |
|  | 201900477 | عزت علي عبدالرحمن علي |  |
|  | 201900374 | شوقي سيد شوقي |  |
|  | 201900499 | عمر احمد شعبان محمد |  |
|  | 201900047 | احمد طارق إسماعيل محمد |  |
|  | 201900939 | هشام احمد سيد معوض |  |

Delivered to:

**Dr. Wessam El-Behaidy**

**Eng. Islam Gamal**

**Eng. Muhammed Kamal**

I. NUMERICAL DATASET

Documentation of ann(numircal set)

Dataset in use : house price data

About the data set: this data set include a group

of houeses and ther describration in term rooms bathroom

and prise is it more ir less the medium and our roll to

classificate these houses.

\*to do this using python we setup some softwares like

Anaconda ,jupyter,python,and we downloded some librarys also like pandas,numpy,keras. And we code it in jupyter.

\*now i will show evry part of the code and describe it.

@@@@@@@@@@@@@@@@@@@@@@@@@@@@

1-First thing we import the librarys

import numpy as np

import pandas as pd

import tensorflow as tf

tf.\_\_version\_\_

2- importing the dataset

df = pd.read\_csv('housepricedata.csv')

3-we can show the data set using this promp

bd

4-Now that we’ve seen what our data looks like, we want to convert it into arrays for our machine to process:

dataset = df.values

5-We now split our dataset into input features (X) and the feature we wish to predict (Y). To do that split, we simply assign the first 10 columns of our array to a variable called X and the last column of our array to a variable called Y. The code to do the first assignment is this:

X = dataset[:,0:10]

We then assign the last column of our array to Y:

Y = dataset[:,10]

6-We first have to import the code that we want to use:

from sklearn import preprocessing

7-we use a function called the min-max scaler, which scales the dataset so that all the input features lie between 0 and 1 inclusive:

min\_max\_scaler = preprocessing.MinMaxScaler()

X\_scale = min\_max\_scaler.fit\_transform(X)

8-we can use this prompt to show the scale

X\_scale

9-We will use the code from scikit-learn called ‘train\_test\_split’, which as the name suggests, split our dataset into a training set and a test set. We first import the code we need:

from sklearn.model\_selection import train\_test\_split

10-Then, split your dataset like this:

X\_train, X\_val\_and\_test, Y\_train, Y\_val\_and\_test = train\_test\_split(X\_scale, Y, test\_size=0

11-  we can use the same function to do the split again on val\_and\_test:

X\_val, X\_test, Y\_val, Y\_test = train\_test\_split(X\_val\_and\_test, Y\_val\_and\_test, test\_size=0.5)

12-If you want to see how the shapes of the arrays are for each of them (i.e. what dimensions they are), simply run

print(X\_train.shape, X\_val.shape, X\_test.shape, Y\_train.shape, Y\_val.shape, Y\_test.shape)

13 - let’s import the necessary code from Keras:

from keras.models import Sequential

from keras.layers import Dense

14-Then, we specify that in our Keras sequential model like this:

model = Sequential([

Dense(32, activation='relu', input\_shape=(10,)),

Dense(32, activation='relu'),

Dense(1, activation='sigmoid'),

])

15- Configuring the model with these settings requires us to call the function model.compile, like this:

model.compile(optimizer='sgd',

loss='binary\_crossentropy',

metrics=['accuracy'])

16- Lastly, we want to track accuracy on top of the loss function. Now once we’ve run that cell, we are ready to train!

Training on the data is pretty straightforward and requires us to write one line of code:

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

batch\_size=32, epochs=100,

validation\_data=(X\_val, Y\_val))

SVM model

1-First thing we import the librarys

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

from sklearn.preprocessing import StandardScaler

from sklearn.svm import SVC

from sklearn.metrics import confusion\_matrix, accuracy\_score

from sklearn.metrics import confusion\_matrix, accuracy\_score

from matplotlib.colors import ListedColormap

2- importing the dataset

dataset = pd.read\_csv('housepricedata.csv')

3-Splitting the dataset into the Training set and Test set

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.25, random\_state = 4)

X\_train

4-Feature Scaling

sc = StandardScaler()

X\_train = sc.fit\_transform(X\_train)

X\_test = sc.transform(X\_test)

5-Training the SVM model on the Training set

classifier = SVC(kernel = 'linear', random\_state = 0)

classifier.fit(X\_train, y\_train)

6- Predicting the Test set results

y\_pred = classifier.predict(X\_test)

print(np.concatenate((y\_pred.reshape(len(y\_pred),1), y\_test.reshape(len(y\_test),1)),1))

7-Making the Confusion Matrix

cm = confusion\_matrix(y\_test, y\_pred)

print(cm)

accuracy\_score(y\_test, y\_pred)

II. IMAGE DATASET

1. Project Introduction

* 1. **Dataset Name**

Mnist

* 1. **Number of classes and their labels**

10 classes

* 1. **Dataset Images Numbers and size**

42000

* 1. **Training, Validation and Testing**

Testing is 25%

Validation 75%

**Documentation for ANN(for Image data set)**

**1 – importing libraries:**

import os

import tensorflow as tf

import numpy as np

import matplotlib.pyplot as plt

import cv2 # To do some image operations

import pandas as pd

**2- Gathering the data set**

#Gathering data set

mnist = tf.keras.datasets.mnist

**3- Split the data into training and testing sets:**

# Split the data into training sets => Two tuples 1-> training data set 2-> testing data set

(x\_train , y\_train) ,(x\_test , y\_test) = mnist.load\_data()

#convert data type from fload64 to float 32

#Normalize its value between 0 to 1 instead 0 to 255 for to be faster

x\_train , x\_test = x\_train / 255.0 , x\_test / 255.0 #255 pixel

print("shape of x\_train" , x\_train.shape)

print("shape of x\_test" , x\_test.shape)

#60000 the numnber of input , each image has 28 \* 28 pixel

**4-Bulding ANN Model :**

model = tf.keras.models.Sequential([

# Flatting is coonverting the data into a 1-dimensional

# array for inputting it it the next layer

tf.keras.layers.Flatten(input\_shape = (28 , 28)),

tf.keras.layers.Dense(128 , activation = 'relu'), #AS all layers is densely connected so we must use dense layer type

tf.keras.layers.Dropout(0.2), # By dropping unit out , we temporarily removing it from the nerwork with all its incoming and outgoing connections

tf.keras.layers.Dense(10 , activation = 'softmax') # number of category is 10 , must use function 'softmax' for image classification

])

model.summary()

**5-Complie and Train The model:**

# Compile & Train the model

model.compile(optimizer='adam',

loss='sparse\_categorical\_crossentropy',

metrics=['accuracy']) #Calculate how often predections equal labels

**6-Validation the data :**

r = model.fit(x\_train, y\_train, validation\_data=(x\_test, y\_test), epochs=10) # We use a validation data as a testing data also

**7- Model Evaluation:**

#Model evaluation

print(model.evaluate(x\_test, y\_test))

**8-Confusion Metrix with GUI:**

#Confusion Matrix

from sklearn.metrics import confusion\_matrix

import itertools

def plot\_confusion\_matrix(cm, classes,

normalize = False,

title="Confusion Matrix",

cmap = plt.cm.Greens):

plt.imshow(cm, interpolation='nearest', cmap=cmap)

plt.title(title)

plt.colorbar()

tick\_marks = np.arange(len(classes))

plt.xticks(tick\_marks, classes, rotation=45)

plt.yticks(tick\_marks, classes)

fmt = '.2f' if normalize else 'd'

thresh = cm.max() / 2.

for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):

plt.text(j, i, format(cm[i, j], fmt),

horizontalalignment = 'center',

color = "white" if cm[i, j] > thresh else "black")

plt.tight\_layout()

plt.ylabel('True label')

plt.xlabel('Predicted label')

plt.show()

p\_test = model.predict(x\_test).argmax(axis=1)

cm = confusion\_matrix(y\_test, p\_test)

plot\_confusion\_matrix(cm, list(range(10)))

SVM model (images)

1-First thing we import the librarys

import os

import numpy as np

import cv2

import matplotlib.pyplot as plt

import pickle

import random

from sklearn.model\_selection import train\_test\_split

from sklearn.svm import SVC

from sklearn.metrics import confusion\_matrix

from sklearn.preprocessing import label\_binarize

from sklearn.multiclass import OneVsRestClassifier

from scipy import interp

from sklearn.metrics import roc\_curve, auc

2- importing the dataset

dir = 'C:\\Users\\Mahmoud\\Documents\\archive\\trainingSet\\trainingSet'

3-Splitting the dataset into the Training set and Test set

from sklearn.model\_selection import train\_test\_split

xtrain , xtest , ytrain , ytest = train\_test\_split(features , labels, test\_size= 0.25)

4-Feature Scaling

features = []

labels = []

for feature , label in data:

features.append(feature)

labels.append(label)

5-Training the SVM model on the Training set

model = SVC(C=1 , kernel='poly' , gamma = 'auto' )

model.fit(xtrain ,ytrain)

6- Predicting the Test set results

prediction = model.predict(xtest)

accuracy = model.score(xtest, ytest)

7-Making the Confusion Matrix

plt.plot(confusion\_matrix(ytest,prediction))

plt.show()