

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. 51

|  |  |  |  |
| --- | --- | --- | --- |
|  | ID | Name | Grade |
|  | 201900457 | عبدالله جمال ابراهيم بسطاوي |  |
|  | 201900027 | احمد خالد جابر احمد |  |
|  | 20180441 | كيرلس ميشيل سيد عطاالله |  |
|  | 201900121 | اسامه عبدالرسول ابراهيم صابر |  |
|  | 201900081 | احمد مجدي عبدلله السيد |  |
|  | 201900402 | عبدالرحمن احمد علي حسن غزالي |  |
|  |  |  |  |

Delivered to:

**Dr. Wessam El-Behaidy**

**Eng. Islam Gamal**

**Eng. Muhammed Kamal**

I. NUMERICAL DATASET

1. Project Introduction

* **Dataset Name**

Social\_Network\_Ads

https://www.kaggle.com/akram24/social-network-ads

* **Number of classes and their labels**

(Specify number of classes and their labels.)

two classes (0 , 1 )

purchase the product or not

* **Dataset Samples Numbers**

(The total number of samples in dataset)

This dataset is consisted of (400, 5)

* **Training, Validation and Testing**

(The number of samples used in training, validation and testing.)

**80 % of original set is training**

**20 % of original set is testing**

**20 % of training set is validation**

* Implementation Details
* **Extracted Features**

(How many features were extracted, their names, the dimension of resulted features)

3 features (Gender,Age, EstimatedSalary)

400 for each feature with no null values

* **Cross-validation**

(Is cross-validation is used in any of implemented models? If yes, specify the number of fold and ratio of training/validation)

**We used traditional validation**

* **Artificial Neural Network (ANN)**
* **Hyper-parameters**

(Specify all the hyper-parameters (initial learning rate, optimizer, regularization, batch size, no. of epochs…) with their specified value in implementation)

We use 3 nural networks layers:

input layer = 3 units

function prameters:

(units=3, activation=’relu’, use\_bias=True, kernel\_initializer='glorot\_uniform', bias\_initializer='zeros', kernel\_regularizer=None, bias\_regularizer=None, activity\_regularizer=None, kernel\_constraint=None, bias\_constraint=None, \*\*kwargs)

second layer = 128 units

function prameters:

(units=128, activation=’relu’, use\_bias=True, kernel\_initializer='glorot\_uniform', bias\_initializer='zeros', kernel\_regularizer=None, bias\_regularizer=None, activity\_regularizer=None, kernel\_constraint=None, bias\_constraint=None, \*\*kwargs)

output layer = 1 unit

function prameters:

(units=1, activation=’sigmoid’, use\_bias=True, kernel\_initializer='glorot\_uniform', bias\_initializer='zeros', kernel\_regularizer=None, bias\_regularizer=None, activity\_regularizer=None, kernel\_constraint=None, bias\_constraint=None, \*\*kwargs)

in model compile we use:

learning rate = 1e-3

optimizer = optimizer=tf.keras.optimizers.Adam

function prameters:

model.compile(optimizer=tf.keras.optimizers.Adam(learning\_rate=1e-3),  
loss=tf.keras.losses.BinaryCrossentropy(),  
metrics=[tf.keras.metrics.BinaryAccuracy(),  
tf.keras.metrics.FalseNegatives()])

in model fitting we use:

batch size = 64

epochs = 100

function prameters:

model.fit(x= X\_train, y= y\_train, batch\_size=64, epochs=100, verbose='auto', callbacks=None, validation\_split=0.20, validation\_data=None, shuffle=True, class\_weight=None, sample\_weight=None, initial\_epoch=0, steps\_per\_epoch=None, validation\_steps=None, validation\_batch\_size=None, validation\_freq=1, max\_queue\_size=10, workers=1, use\_multiprocessing=False)

* **Support Vector Machine** **(SVM)**
* **Hyper-parameters**

(Specify all the hyper-parameters (optimizer, regularization, …) with their specified value in implementation)

In the model we use:

Kernel = ‘linear’

Random state = 0

The other parameters set as default like the following:

Svc parameters function:

SVC(\*, C=1.0, kernel='linear', degree=3, gamma='scale', coef0=0.0, shrinking=True, probability=False, tol=0.001, cache\_size=200, class\_weight=None, verbose=False, max\_iter=-1, decision\_function\_shape='ovr', break\_ties=False, random\_state=0)

By using svc\_object.fit() function we fitting the svc model with the following parameters:

classifier.fit(x\_train, y\_train, sample\_weight=None)

* Models Results

**For each model you should show all these results for your model on testing data** (loss curve, accuracy, confusion matrix, ROC curve)

* **ANN Results**

Chart, line chart

Description automatically generated

Chart

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Chart, line chart

Description automatically generated

* **SVM Results**

Chart

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Chart, line chart

Description automatically generated

II. IMAGE DATASET

1. Project Introduction

* **Dataset Name**
* (What is the dataset used?)
* Name = Intel Image Classification
* Link = https://www.kaggle.com/puneet6060/intel-image-classification
* **Number of classes and their labels**

there are two classes (buildings-forest)

there are two labels ( 0 , 1)

* **Dataset Images Numbers and size**
* (The total number of images in dataset and the size of each.)
* 5373 image
* 150 \* 150 pixels
* **Training, Validation and Testing**

training = 3760 , validation = 1075 ,testing =538

2. Implementation Details

* **Extracted Features**

(How many features were extracted, their names, the dimension of resulted features)

the number of features = 16384 (128\*128)

their names ( 0 : 16383 )

the dimension of resulted features = 2 dimension

* **Cross-validation**

(Is cross-validation is used in any of implemented models? If yes, specify the number of fold and ratio of training/validation)

Not used

* **Artificial Neural Network (ANN)**
* **Hyper-parameters**

(Specify all the hyper-parameters (initial learning rate, optimizer, regularization, batch size, no. of epochs…) with their specified value in implementation)

We use 3 nural networks layers:

input layer = 64 units

function prameters:

(units=64, activation=’relu’, use\_bias=True, kernel\_initializer='glorot\_uniform', bias\_initializer='zeros', kernel\_regularizer=None, bias\_regularizer=None, activity\_regularizer=None, kernel\_constraint=None, bias\_constraint=None, \*\*kwargs)

second layer = 64 units

function prameters:

(units=64, activation=’relu’, use\_bias=True, kernel\_initializer='glorot\_uniform', bias\_initializer='zeros', kernel\_regularizer=None, bias\_regularizer=None, activity\_regularizer=None, kernel\_constraint=None, bias\_constraint=None, \*\*kwargs)

output layer = 1 unit

function prameters:

(units=1, activation=’sigmoid’, use\_bias=True, kernel\_initializer='glorot\_uniform', bias\_initializer='zeros', kernel\_regularizer=None, bias\_regularizer=None, activity\_regularizer=None, kernel\_constraint=None, bias\_constraint=None, \*\*kwargs)

in model compile we use:

learning rate = 1e-3

optimizer = optimizer=tf.keras.optimizers.Adam

function prameters:

model.compile(optimizer=tf.keras.optimizers.Adam(learning\_rate=1e-3),  
loss=tf.keras.losses.BinaryCrossentropy(),  
metrics=[tf.keras.metrics.BinaryAccuracy(),  
tf.keras.metrics.FalseNegatives()])

in model fitting we use:

batch size = 56

epochs = 100

function prameters:

model.fit(x=None, y=None, batch\_size=56, epochs=100, verbose='auto', callbacks=None, validation\_split=0.0, validation\_data=None, shuffle=True, class\_weight=None, sample\_weight=None, initial\_epoch=0, steps\_per\_epoch=None, validation\_steps=None, validation\_batch\_size=None, validation\_freq=1, max\_queue\_size=10, workers=1, use\_multiprocessing=False)

* **Support Vector Machine** **(SVM)**
* **Hyper-parameters**

(Specify all the hyper-parameters (optimizer, regularization, …) with their specified value in implementation)

In the model we use:

Kernel = ‘linear’

Random state = 0

The other parameters set as default like the following:

Svc parameters function:

SVC(\*, C=1.0, kernel='linear', degree=3, gamma='scale', coef0=0.0, shrinking=True, probability=False, tol=0.001, cache\_size=200, class\_weight=None, verbose=False, max\_iter=-1, decision\_function\_shape='ovr', break\_ties=False, random\_state=0)

By using svc\_object.fit() function we fitting the svc model with the following parameters:

classifier.fit(x\_train, y\_train, sample\_weight=None)

3. Models Results

**For each model you should show all these results for your model on testing data** (loss curve, accuracy, \*, ROC curve)

* **ANN Results**
* Loss:
* Chart, histogram

  Description automatically generated
* Accuracy:
* **Chart, scatter chart

  Description automatically generated**
* Confusion matrix:
* Graphical user interface, text, application

  Description automatically generated
* ROC curve:
* Chart, line chart

  Description automatically generated
* **SVM Results**
* Chart, histogram

  Description automatically generated
* **Text

  Description automatically generated**

**Chart, line chart

Description automatically generated**