Helwan University

كلية الحاسبات والذكاء الإصطناعي
Faculty of Computers & Artificial Intelligence

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. ****

| | ID | Name | Grade |
|---|---|---|---|
| **1.** | 201900548 | فادى ملاك عطيه ميخائيل | |
| **2.** | 201900881 | مينا فوزى فايز مينا | |
| **3.** | 201900179 | امير حنا ثابت فهيم | |
| **4.** | 201900603 | مارينا رومانى نصر شنودة | |
| **5.** | 201900206 | ايمان محمد محمود عبد الحميد | |
| **6.** | 201900199 | اياد ايمن محمد مصيلحى | |
| **7.** | 201900020 | احمد امير احمد شفيق | |
| **8.** | 201600253 | حازم حافظ محمد حافظ | |

Delivered to:

**Dr. Wessam El-Behaidy**

**Eng. Islam Gamal**

**Eng. Muhammed Kamal**

# I. NUMERICAL DATASET

# 1. Project Introduction

## a. Dataset Name

Tabular Playground Series

## b. Number of classes and their labels

(Specify number of classes and their labels.)
There are 2 Classes
Labels:
1 - Spruce/Fir
2 - Lodgepole Pine

## c. Dataset Samples Numbers

(The total number of samples in dataset)
500K Samples

## d. Training, Validation and Testing

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

Training = 75%  (375K)
Testing  = 25%  (125K)

# 2. Implementation Details

## a. Extracted Features

We have dropped two features because they have the same value =0
So we don't need them.
We have added four features (mean - std – min - max) to improve the accuracy.
Accuracy before adding those four feature

## b. Cross-validation

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

No Cross Validation Used in the Numeric dataset because the data is huge and cross validation takes long time to train

## c. 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 used Adam as an optimizer and we reached to optimal Learning Rate = 0.001 we first initially the learning rate to 0.1 and It was overfitting and number of epochs is 5, after 5 epochs we reached more than 97% Accuracy so it was perfect and in first hidden layer we add 2048 neuron and in second hidden layer we add 1024 neuron and 3rd hidden layer was 512 neuron and in the output layer we add 2 neuron for our 2 classes and the activation function of the hidden layers was relu and activation function of the output layer was sigmoid function and it gives accuracy 95% after changing the activation function of the hidden layers to swish and activation function of the output layer to softmax function and it gives accuracy more than 97%

## d. Support Vector Machine (SVM)

### ℘ Hyper-parameters

The svm we changed the kernel to linear and Regularization and gamma after playing of Parameters it gives accuracy 94% after keeping the svm model to default Parameters it gives accuracy 96%

# 3. Models Results

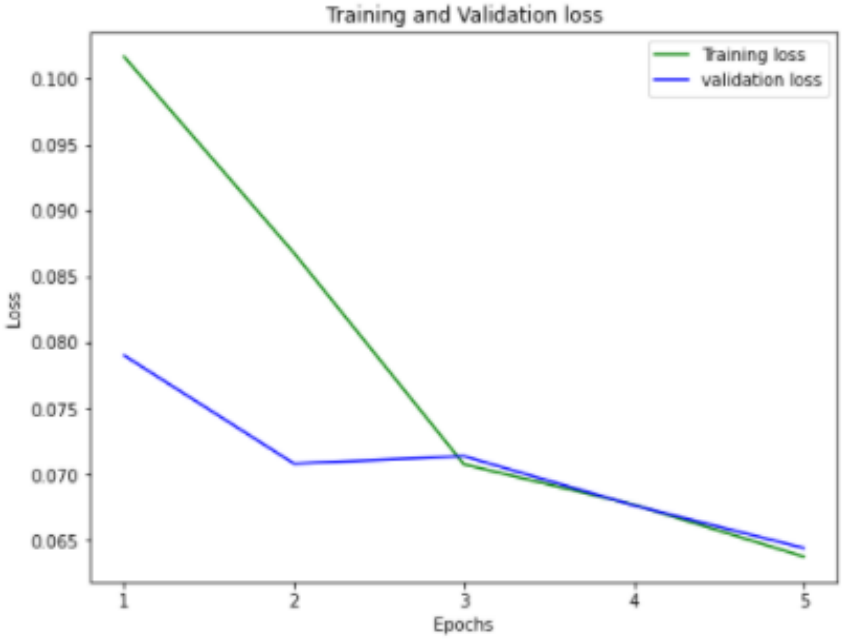
**For each model you should show all these results for your model on testing data**

ANN Accuracy was 97.2% and loss 0.0644 after 5 epochs, ANN confusion matrix in class 1 it predicted 45614 right and make 1178 wrong and for class 2 it predicted 75941 right and make 2267 wrong and SVM accuracy was 96.1% and confusion matrix in class 1 it predicted 45506 right and make 2390 wrong and for class 2 it predicted 74729 right and make 2375 wrong

# a. ANN Results

Confusion Matrix

```
plt.figure(figsize=(8,6))
loss_train = history.history['loss']
loss_val = history.history['val_loss']
epochs = range(1,6)
plt.plot(epochs, loss_train, 'g', label='Training loss')
plt.plot(epochs, loss_val, 'b', label='validation loss')
plt.title('Training and Validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.xticks(np.arange(1,6))
plt.legend()
plt.show()
```




**Accuarcy of the NN**

FM

```
In [40]: nn_model.evaluate(X_test_nn, y_test_nn)

         3907/3907 [==============================] - 45s 12ms/step - loss: 0.0644 - acc: 0.9724

Out[40]: [0.06441473215818405, 0.9724400043487549]
```

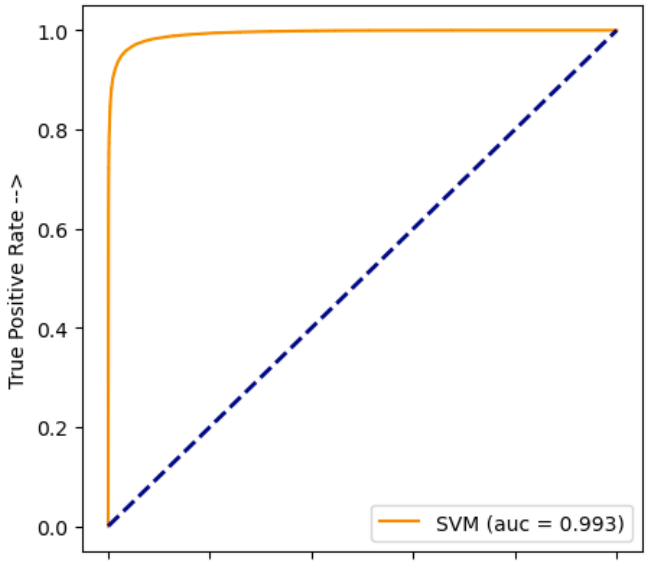
# b.SVM Results

***ROC Curve***

In [42]:
```
y_pred_svm = svm_model.decision_function(X_test)
```

In [47]:
```
y_test_repl = y_test.replace(to_replace=[1,2], value=[0,1])
```

In [138]:
```
svm_fpr, svm_tpr, _ = roc_curve(y_test_repl, y_pred_svm)
auc_svm = auc(svm_fpr, svm_tpr)

plt.figure(figsize=(5, 5), dpi=100)
plt.plot(svm_fpr, svm_tpr, color="darkorange", label='SVM (auc = %0.3f)' % auc_svm)
plt.plot([0, 1], [0, 1], color="navy", lw=2, linestyle="--")

plt.xlabel('False Positive Rate -->')
plt.ylabel('True Positive Rate -->')

plt.legend()

plt.show()
```




***Accuarcy of the model***

In [31]:
```
svm_model = SVC()
train(svm_model, "SVM")
```

```
SVM model score on Training data: 96.91733333333333%
SVM model score on Testing data: 96.188%
```

**c.**

# II. IMAGE DATASET

# 1. Project Introduction

## a. Dataset Name

(What is the dataset used?)
Bayern

## b. Number of classes and their labels

(Specify number of classes and their labels.)
5 Classes
Labels:

1. Kingsley Coman
2. Joshua Kimmich
3. Robert Lewandowski
4. Manuel Neuer
5. Leory Sane

## c. Dataset Images Numbers and size

(The total number of images in dataset and the size of each.)

The data contains 230 images
Each image is 64 * 64 Pixels
Data contains 5 Classes

## d. Training, Validation and Testing

(The number of images used in training, validation and testing.)
**Number of Training data: 172 (75%)**
**Number of Testing data: 58 (25%)**

# 2. Implementation Details

## a. Extracted Features

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

## b. Cross-validation

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

Yes we applied cross validation we used 10 folds and accuracy was 84.34%

**Cross Validation**

```
In [27]: svm_model = SVC(kernel='linear', gamma='auto')
         scores = cross_val_score(svm_model,X,y, cv=10)
         np.average(scores)

Out[27]: 0.8434782608695652
```

## c. 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)

batch_size = 10.
epochs = 8.
optimizer = Adam

learning_rate = 0.0001.
loss = 'categorical_crossentropy'.

## d. Support Vector Machine (SVM)

### ℘ Hyper-parameters

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

cv=10 , gamma =’auto’ ,Kernel= ‘linear’

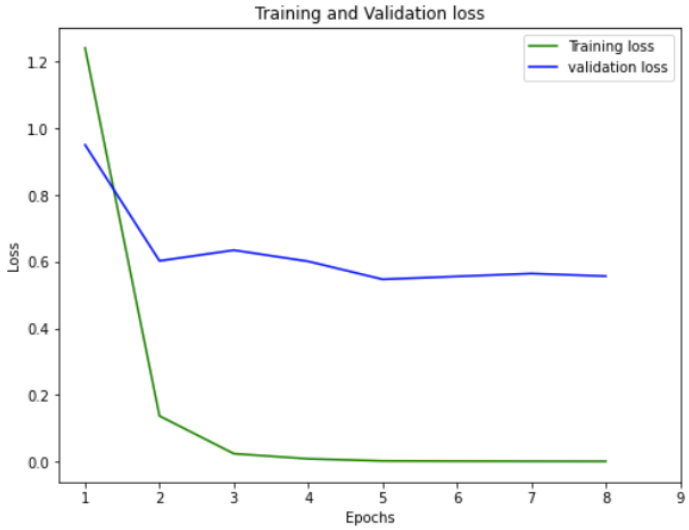
# 3. Models Results

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

# a. ANN Results

**Loss Curve**

In [64]:
```
plt.figure(figsize=(8,6))
loss_train = history.history['loss']
loss_val = history.history['val_loss']
epochs = range(1,9)
plt.plot(epochs, loss_train, 'g', label='Training loss')
plt.plot(epochs, loss_val, 'b', label='validation loss')
plt.title('Training and Validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.xticks(np.arange(1,10))
plt.legend()
plt.show()
```



### Accuracy of ANN

In [62]:
```
nn_model.evaluate(X_test_nn, y_test_nn)
```

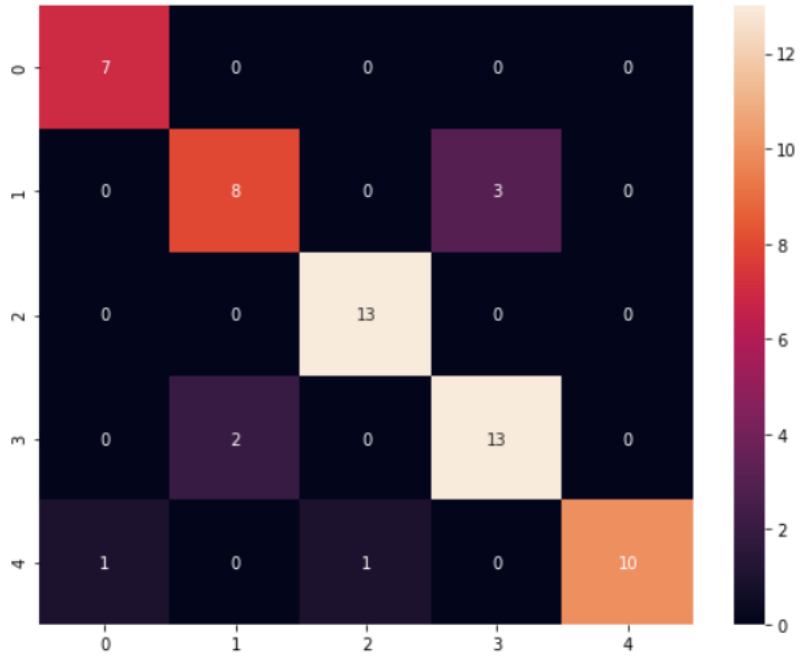
```
2/2 [==============================] - 0s 55ms/step - loss: 0.5565 - acc: 0.8793
```

Out[62]:
```
[0.5564684271812439, 0.8793103694915771]
```

### Confusion Matrix

In [73]:
```
plt.figure(figsize=(9,7))
sns.heatmap(cm, annot=True)
```

Out[73]:
```
<AxesSubplot:>
```



# b.SVM Results

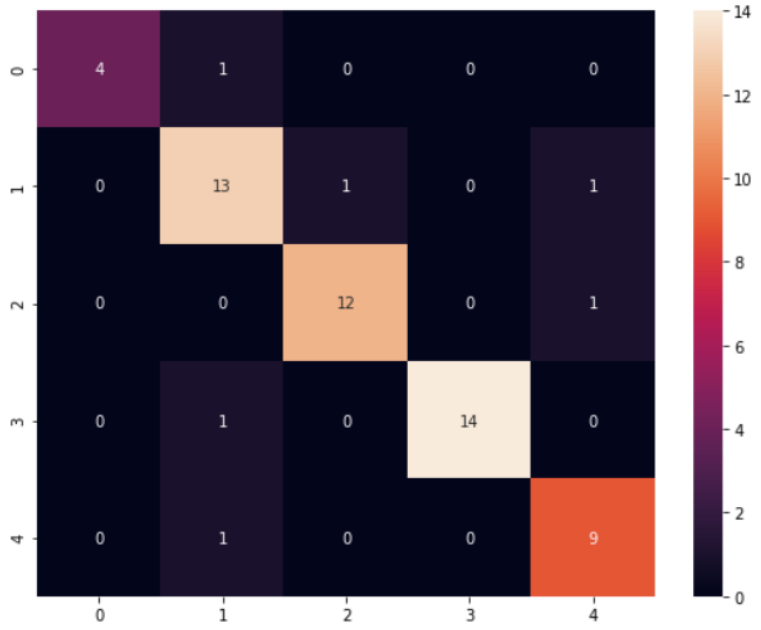
**Score of the model**

```
In [29]: svm_model.score(X_test, y_test)

Out[29]: 0.896551724137931
```

**Confusion Matrix**

```
In [30]: cm = confusion_matrix(y_test, svm_model.predict(X_test))
         cm

Out[30]: array([[ 4,  1,  0,  0,  0],
                [ 0, 13,  1,  0,  1],
                [ 0,  0, 12,  0,  1],
                [ 0,  1,  0, 14,  0],
                [ 0,  1,  0,  0,  9]], dtype=int64)
```

```
In [31]: plt.figure(figsize=(9,7))
         sns.heatmap(cm, annot=True);
```



**ROC Curve**

In [46]:
```
plt.figure(figsize=(10,7))
lw = 2
plt.plot(
    fpr[2],
    tpr[2],
    color="darkorange",
    lw=lw,
    label="ROC curve (area = %0.2f)" % roc_auc[2],
)
plt.plot([0, 1], [0, 1], color="navy", lw=lw, linestyle="--")
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("Receiver operating characteristic")
plt.legend(loc="lower right")
plt.show()
```