AIN SHAMS UNIVERSITY
FACULTY OF ENGINEERING
ICHEP – Senior 2 Level – CESS
CSE485 - Deep Learning
Fall 2023

Final Project
CSE 485: Deep Learning
Leaf Classification

Submitted to:
Dr Mahmoud Khalil
Eng Mahmoud Soheil

Mohamed Obia 19p5170
Mohamed Fathy 19p4707
Ziad Assem 19p6363
Karim Shalaby 19p6044

Contents

1.0 Problem

There are estimated to be nearly half a million species of plant in the world. Classification of species has been historically problematic and often results in duplicate identifications.
The objective of this project is to use extracted features, including shape, margin & texture, to accurately identify 99 species of plants. Leaves, due to their volume, prevalence, and unique characteristics, are an effective means of differentiating plant species. They also provide a fun introduction to applying techniques that involve image-based features.

2.0 Part 1

The data

```
train = pd.read_csv('/content/gdrive/MyDrive/Colab Notebooks/DL Project/train.csv')
train.head()
```

	id	species	margin1	margin2	margin3	margin4	margin5	margin6	margin7	margin8	...	texture55	texture56	texture57	texture58	texture59	tex
0	1	Acer_Opalus	0.007812	0.023438	0.023438	0.003906	0.011719	0.009766	0.027344	0.0	...	0.007812	0.000000	0.002930	0.002930	0.035156	
1	2	Pterocarya_Stenoptera	0.005859	0.000000	0.031250	0.015625	0.025391	0.001953	0.019531	0.0	...	0.000977	0.000000	0.000000	0.000977	0.023438	
2	3	Quercus_Hartwissiana	0.005859	0.009766	0.019531	0.007812	0.003906	0.005859	0.068359	0.0	...	0.154300	0.000000	0.005859	0.000977	0.007812	
3	5	Tilia_Tomentosa	0.000000	0.003906	0.023438	0.005859	0.021484	0.019531	0.023438	0.0	...	0.000000	0.000977	0.000000	0.000000	0.020508	
4	6	Quercus_Variabilis	0.005859	0.003906	0.048828	0.009766	0.013672	0.015625	0.005859	0.0	...	0.096680	0.000000	0.021484	0.000000	0.000000	

5 rows × 194 columns

```
test= pd.read_csv('/content/gdrive/MyDrive/Colab Notebooks/DL Project/test.csv')
test.head()
```

	id	margin1	margin2	margin3	margin4	margin5	margin6	margin7	margin8	margin9	...	texture55	texture56	texture57	texture58	texture59	texture60	te
0	4	0.019531	0.009766	0.078125	0.011719	0.003906	0.015625	0.005859	0.0	0.005859	...	0.006836	0.000000	0.015625	0.000977	0.015625	0.0	
1	7	0.007812	0.005859	0.064453	0.009766	0.003906	0.013672	0.007812	0.0	0.033203	...	0.000000	0.000000	0.006836	0.001953	0.013672	0.0	
2	9	0.000000	0.000000	0.001953	0.021484	0.041016	0.000000	0.023438	0.0	0.011719	...	0.128910	0.000000	0.000977	0.000000	0.000000	0.0	
3	12	0.000000	0.000000	0.009766	0.011719	0.017578	0.000000	0.003906	0.0	0.003906	...	0.012695	0.015625	0.002930	0.036133	0.013672	0.0	
4	13	0.001953	0.000000	0.015625	0.009766	0.039062	0.000000	0.009766	0.0	0.005859	...	0.000000	0.042969	0.016602	0.010742	0.041016	0.0	

5 rows × 193 columns

The training data consists of id column which is a column that has each leaf image id. It has a species column which us the label of each leaf. And finally, the feature columns which consists of 3 features which are margin, shape, and texture of each leaf. Each feature consists of 64 different values that represent this feature.

Understanding the data shapes

```
print(train.shape,test.shape)
```

```
(990, 194) (594, 193)
```

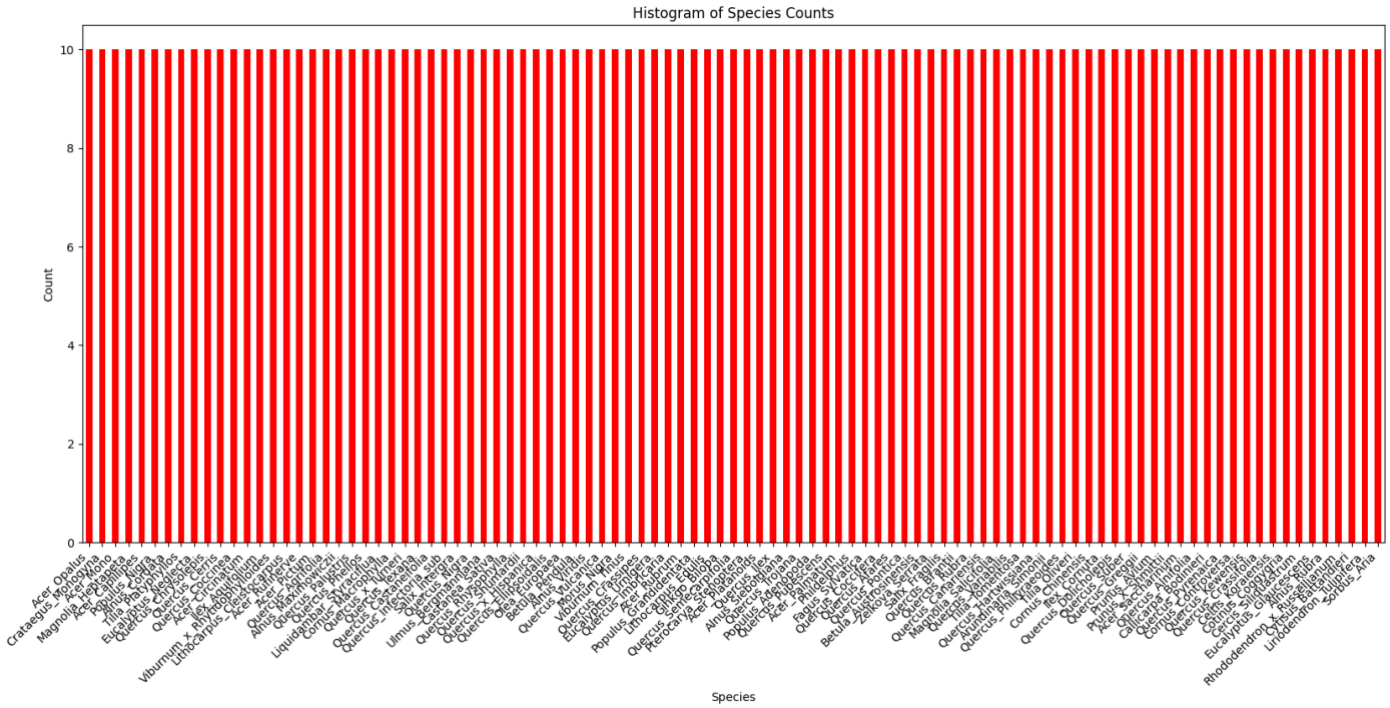
```
missing_values = train.isnull().sum()
missing_values
```

```
id          0
species     0
margin1     0
margin2     0
margin3     0
           ..
texture60   0
texture61   0
texture62   0
texture63   0
texture64   0
Length: 194, dtype: int64
```

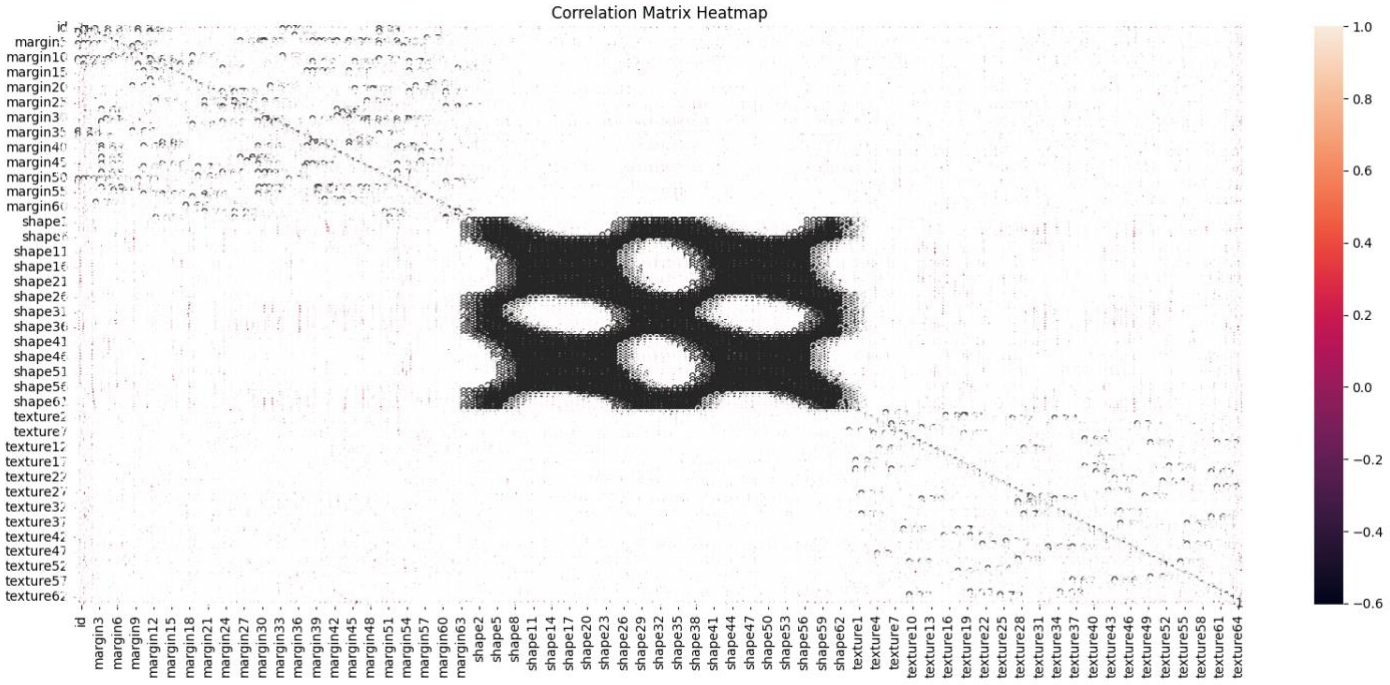
```
duplicate_counts = train.duplicated().sum()
duplicate_counts
```

```
0
```

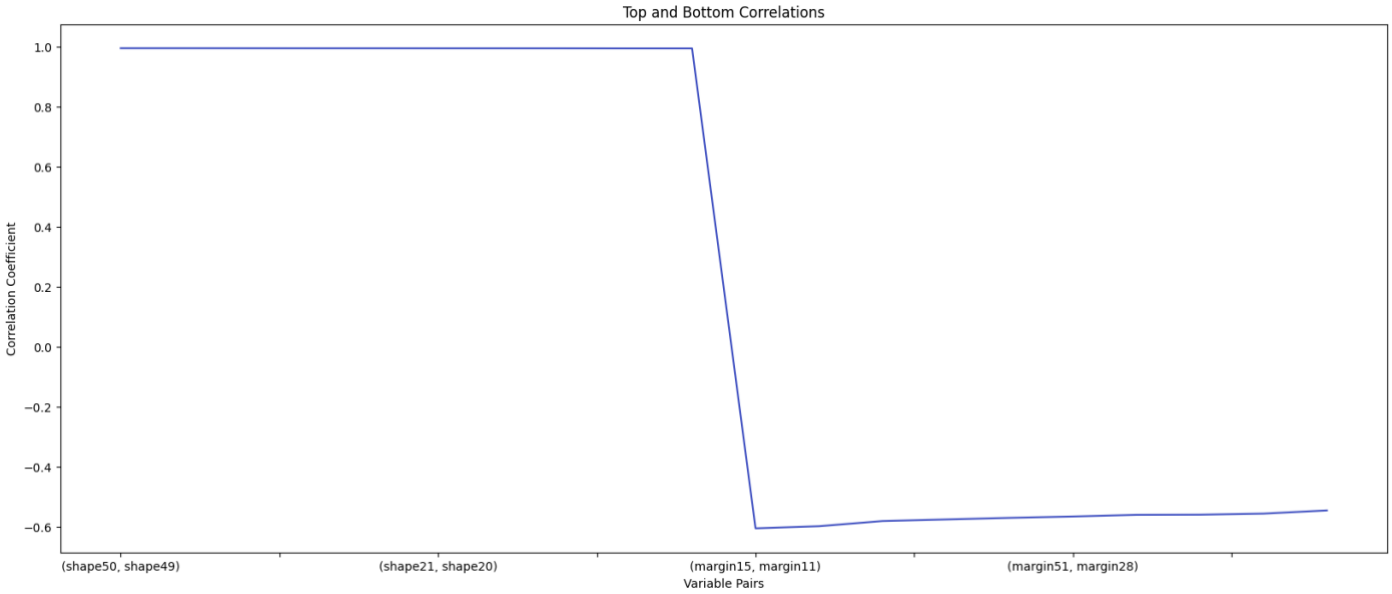
Plotting histogram of the species distribution

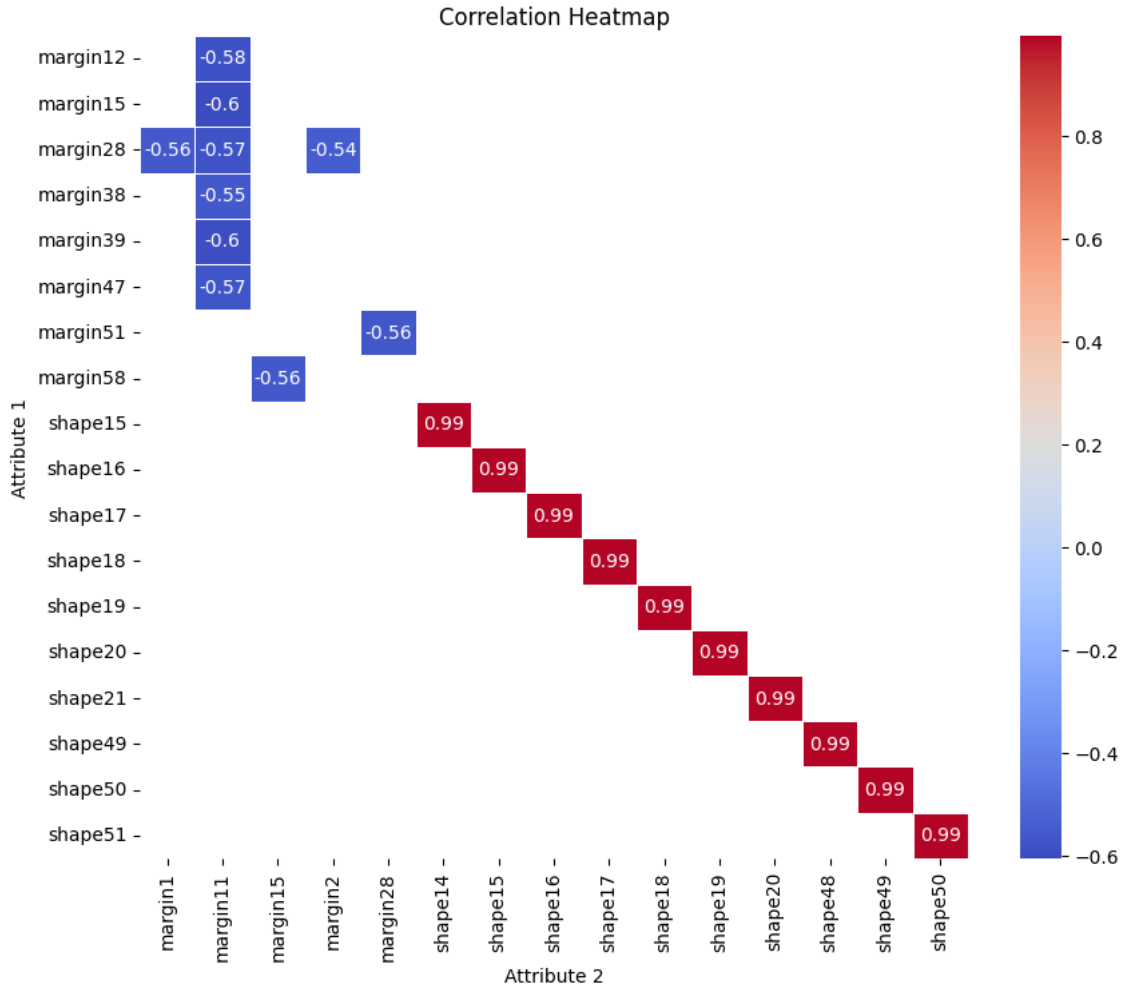


Correlation matrix



The correlation matrix doesn't give a good indication due to the huge amounts of data that we have, so we plotted the correlation heatmap between the highest and lowest features.





3.0 Part 2

Encoding the data: removing the unnecessary columns and encoding the labels to numerical values

```
def encode(train, test):
    label_encoder = LabelEncoder().fit(train.species) # fit labels(species) into numerical values
    labels = label_encoder.transform(train.species) # transfroming them into numerical values
    classes = list(label_encoder.classes_) # contains the unique classes in species column

    train = train.drop(['species', 'id'], axis=1) # dropping the labels from and image ids from the training data
    test_ids=test.id
    test = test.drop('id', axis=1) # dropping image ids from test data

    return train, labels, test, classes,test_ids


train, labels, test, classes,test_ids = encode(train, test)
```

Splitting the data into training and validation with 80:20 ratio.

```
# scalling the data with the z score
scaler = StandardScaler().fit(train.values)
scaled_train = scaler.transform(train.values)
# splitting the data but ennsuring that the class distrubtion of the labels are the constant
sss = StratifiedShuffleSplit(test_size=0.2, random_state=23)
# assigning the indices of the train and validation data
for train_index, valid_index in sss.split(scaled_train, labels):
    X_train, X_valid = scaled_train[train_index], scaled_train[valid_index]
    y_train, y_valid = labels[train_index], labels[valid_index]
```

Changing the 2d vector to 3d vector so that the model focuses on each feature at a time.

```
nb_features = 64 # number of features per features type (shape, texture, margin)
nb_class = len(classes) # getting the number of unique classes
# reshape train data from 2d to 3d to fit the model
X_train_r = np.zeros((len(X_train), nb_features, 3)) # 3d numpy array with dimensions of parameters
X_train_r[:, :, 0] = X_train[:, :nb_features] # assigning shape to the first channel
X_train_r[:, :, 1] = X_train[:, nb_features:128] # assigning texture to the second channel
X_train_r[:, :, 2] = X_train[:, 128:] # assigning margin to the last channel

# reshape validation data
X_valid_r = np.zeros((len(X_valid), nb_features, 3))
X_valid_r[:, :, 0] = X_valid[:, :nb_features]
X_valid_r[:, :, 1] = X_valid[:, nb_features:128]
X_valid_r[:, :, 2] = X_valid[:, 128:]
```

Creating a model with 512 kernels. Each kernel is 1d. the model focuses on each feature of the three previously mentioned features, and we tune the hyper parameters of the model in terms of optimizers and learning rates.

```
# Define a list of optimizers and learning rates to try
optimizers = [SGD, Adam, RMSprop]
learning_rates = [0.001, 0.01, 0.1]

for optimizer_class in optimizers:
    for learning_rate in learning_rates:
        # Create the optimizer instance
        optimizer = optimizer_class(learning_rate=learning_rate)

        # Build and compile the model
        model = Sequential()
        model.add(Convolution1D(512, 1, input_shape=(nb_features, 3)))
        model.add(Activation('relu'))
        model.add(Flatten())
        model.add(Dropout(0.4))
        model.add(Dense(2048, activation='relu'))
        model.add(Dense(1024, activation='relu'))
        model.add(Dense(nb_class, activation='softmax'))
        model.compile(loss='categorical_crossentropy', optimizer=optimizer, metrics=['accuracy'])

        # Train the model
        model.fit(X_train_r, y_train, epochs=nb_epoch, validation_data=(X_valid_r, y_valid), batch_size=64)

        # Evaluate the model
        evaluation = model.evaluate(X_valid_r, y_valid, verbose=0)
        accuracy = evaluation[1]

        print(f'Optimizer: {optimizer_class.__name__}, Learning Rate: {learning_rate}, Validation Accuracy: {accuracy * 100:.2f}%')
```

This is the output using sgd with learning rate 0.001. 6% is a very bad validation accuracy.

```
Epoch 1/15
13/13 [==============================] - 22s 2s/step - loss: 4.6067 - accuracy: 0.0063 - val_loss: 4.6014 - val_accuracy: 0.0152
Epoch 2/15
13/13 [==============================] - 20s 1s/step - loss: 4.6031 - accuracy: 0.0126 - val_loss: 4.5960 - val_accuracy: 0.0152
Epoch 3/15
13/13 [==============================] - 15s 1s/step - loss: 4.5941 - accuracy: 0.0177 - val_loss: 4.5905 - val_accuracy: 0.0152
Epoch 4/15
13/13 [==============================] - 15s 1s/step - loss: 4.5894 - accuracy: 0.0189 - val_loss: 4.5852 - val_accuracy: 0.0152
Epoch 5/15
13/13 [==============================] - 18s 1s/step - loss: 4.5845 - accuracy: 0.0215 - val_loss: 4.5799 - val_accuracy: 0.0152
Epoch 6/15
13/13 [==============================] - 16s 1s/step - loss: 4.5772 - accuracy: 0.0177 - val_loss: 4.5747 - val_accuracy: 0.0253
Epoch 7/15
13/13 [==============================] - 20s 2s/step - loss: 4.5743 - accuracy: 0.0139 - val_loss: 4.5695 - val_accuracy: 0.0253
Epoch 8/15
13/13 [==============================] - 16s 1s/step - loss: 4.5719 - accuracy: 0.0227 - val_loss: 4.5643 - val_accuracy: 0.0253
Epoch 9/15
13/13 [==============================] - 18s 1s/step - loss: 4.5630 - accuracy: 0.0290 - val_loss: 4.5590 - val_accuracy: 0.0404
Epoch 10/15
13/13 [==============================] - 16s 1s/step - loss: 4.5566 - accuracy: 0.0429 - val_loss: 4.5536 - val_accuracy: 0.0455
Epoch 11/15
13/13 [==============================] - 17s 1s/step - loss: 4.5556 - accuracy: 0.0316 - val_loss: 4.5483 - val_accuracy: 0.0556
Epoch 12/15
13/13 [==============================] - 16s 1s/step - loss: 4.5492 - accuracy: 0.0417 - val_loss: 4.5430 - val_accuracy: 0.0556
Epoch 13/15
13/13 [==============================] - 16s 1s/step - loss: 4.5491 - accuracy: 0.0391 - val_loss: 4.5377 - val_accuracy: 0.0556
Epoch 14/15
13/13 [==============================] - 17s 1s/step - loss: 4.5416 - accuracy: 0.0455 - val_loss: 4.5325 - val_accuracy: 0.0606
Epoch 15/15
13/13 [==============================] - 16s 1s/step - loss: 4.5357 - accuracy: 0.0518 - val_loss: 4.5272 - val_accuracy: 0.0657
Optimizer: SGD, Learning Rate: 0.001, Validation Accuracy: 6.57%
```

Increasing the learning rate increased the model's validation accuracy by more than 10-fold.

```
Epoch 1/15
13/13 [==============================] - 17s 1s/step - loss: 4.5839 - accuracy: 0.0152 - val_loss: 4.5484 - val_accuracy: 0.0404
Epoch 2/15
13/13 [==============================] - 18s 1s/step - loss: 4.5349 - accuracy: 0.0467 - val_loss: 4.4943 - val_accuracy: 0.1162
Epoch 3/15
13/13 [==============================] - 14s 1s/step - loss: 4.4834 - accuracy: 0.0922 - val_loss: 4.4371 - val_accuracy: 0.1869
Epoch 4/15
13/13 [==============================] - 14s 1s/step - loss: 4.4341 - accuracy: 0.1313 - val_loss: 4.3754 - val_accuracy: 0.2576
Epoch 5/15
13/13 [==============================] - 15s 1s/step - loss: 4.3668 - accuracy: 0.1944 - val_loss: 4.3066 - val_accuracy: 0.3333
Epoch 6/15
13/13 [==============================] - 14s 1s/step - loss: 4.3022 - accuracy: 0.2487 - val_loss: 4.2298 - val_accuracy: 0.3636
Epoch 7/15
13/13 [==============================] - 15s 1s/step - loss: 4.2165 - accuracy: 0.3068 - val_loss: 4.1423 - val_accuracy: 0.3939
Epoch 8/15
13/13 [==============================] - 16s 1s/step - loss: 4.1350 - accuracy: 0.3232 - val_loss: 4.0445 - val_accuracy: 0.4091
Epoch 9/15
13/13 [==============================] - 20s 2s/step - loss: 4.0301 - accuracy: 0.3750 - val_loss: 3.9350 - val_accuracy: 0.4545
Epoch 10/15
13/13 [==============================] - 20s 1s/step - loss: 3.9128 - accuracy: 0.4419 - val_loss: 3.8126 - val_accuracy: 0.4697
Epoch 11/15
13/13 [==============================] - 16s 1s/step - loss: 3.7871 - accuracy: 0.4760 - val_loss: 3.6762 - val_accuracy: 0.4899
Epoch 12/15
13/13 [==============================] - 15s 1s/step - loss: 3.6514 - accuracy: 0.5341 - val_loss: 3.5237 - val_accuracy: 0.5253
Epoch 13/15
13/13 [==============================] - 19s 1s/step - loss: 3.4891 - accuracy: 0.5657 - val_loss: 3.3558 - val_accuracy: 0.5808
Epoch 14/15
13/13 [==============================] - 14s 1s/step - loss: 3.3131 - accuracy: 0.6313 - val_loss: 3.1694 - val_accuracy: 0.6566
Epoch 15/15
13/13 [==============================] - 14s 1s/step - loss: 3.1135 - accuracy: 0.6730 - val_loss: 2.9651 - val_accuracy: 0.6768
Optimizer: SGD, Learning Rate: 0.01, Validation Accuracy: 67.68%
```

Reaching a maximum validation accuracy for the sgd optimizer

```
Epoch 1/15
13/13 [==============================] - 15s 1s/step - loss: 4.4364 - accuracy: 0.1073 - val_loss: 4.0075 - val_accuracy: 0.3434
Epoch 2/15
13/13 [==============================] - 15s 1s/step - loss: 3.5614 - accuracy: 0.4053 - val_loss: 2.8377 - val_accuracy: 0.3838
Epoch 3/15
13/13 [==============================] - 14s 1s/step - loss: 1.9377 - accuracy: 0.6566 - val_loss: 2.6697 - val_accuracy: 0.3283
Epoch 4/15
13/13 [==============================] - 15s 1s/step - loss: 0.9412 - accuracy: 0.8434 - val_loss: 0.8355 - val_accuracy: 0.7879
Epoch 5/15
13/13 [==============================] - 14s 1s/step - loss: 0.3807 - accuracy: 0.9369 - val_loss: 0.3314 - val_accuracy: 0.9293
Epoch 6/15
13/13 [==============================] - 13s 985ms/step - loss: 0.1397 - accuracy: 0.9886 - val_loss: 0.1655 - val_accuracy: 0.9899
Epoch 7/15
13/13 [==============================] - 13s 994ms/step - loss: 0.0758 - accuracy: 0.9949 - val_loss: 0.1268 - val_accuracy: 0.9798
Epoch 8/15
13/13 [==============================] - 16s 1s/step - loss: 0.0599 - accuracy: 0.9962 - val_loss: 0.1124 - val_accuracy: 0.9899
Epoch 9/15
13/13 [==============================] - 16s 1s/step - loss: 0.0745 - accuracy: 0.9861 - val_loss: 0.1015 - val_accuracy: 0.9899
Epoch 10/15
13/13 [==============================] - 14s 1s/step - loss: 0.0333 - accuracy: 0.9975 - val_loss: 0.0925 - val_accuracy: 0.9899
Epoch 11/15
13/13 [==============================] - 15s 1s/step - loss: 0.0241 - accuracy: 0.9987 - val_loss: 0.0873 - val_accuracy: 0.9848
Epoch 12/15
13/13 [==============================] - 14s 1s/step - loss: 0.0194 - accuracy: 1.0000 - val_loss: 0.1026 - val_accuracy: 0.9848
Epoch 13/15
13/13 [==============================] - 13s 998ms/step - loss: 0.0306 - accuracy: 0.9962 - val_loss: 0.0742 - val_accuracy: 0.9899
Epoch 14/15
13/13 [==============================] - 13s 997ms/step - loss: 0.0147 - accuracy: 1.0000 - val_loss: 0.0697 - val_accuracy: 0.9848
Epoch 15/15
13/13 [==============================] - 15s 1s/step - loss: 0.0134 - accuracy: 1.0000 - val_loss: 0.0638 - val_accuracy: 0.9848
Optimizer: SGD, Learning Rate: 0.1, Validation Accuracy: 98.48%
```

Adam optimizer with 0.001 learning rate was our best optimizer.

```
Epoch 1/15
13/13 [==============================] - 27s 2s/step - loss: 3.2545 - accuracy: 0.3245 - val_loss: 0.8404 - val_accuracy: 0.8283
Epoch 2/15
13/13 [==============================] - 33s 3s/step - loss: 0.4512 - accuracy: 0.8826 - val_loss: 0.5151 - val_accuracy: 0.8636
Epoch 3/15
13/13 [==============================] - 27s 2s/step - loss: 0.2082 - accuracy: 0.9470 - val_loss: 0.3602 - val_accuracy: 0.9242
Epoch 4/15
13/13 [==============================] - 27s 2s/step - loss: 0.0989 - accuracy: 0.9735 - val_loss: 0.2368 - val_accuracy: 0.9444
Epoch 5/15
13/13 [==============================] - 24s 2s/step - loss: 0.0529 - accuracy: 0.9861 - val_loss: 0.1062 - val_accuracy: 0.9747
Epoch 6/15
13/13 [==============================] - 27s 2s/step - loss: 0.0457 - accuracy: 0.9886 - val_loss: 0.1262 - val_accuracy: 0.9646
Epoch 7/15
13/13 [==============================] - 24s 2s/step - loss: 0.0211 - accuracy: 0.9949 - val_loss: 0.0824 - val_accuracy: 0.9747
Epoch 8/15
13/13 [==============================] - 29s 2s/step - loss: 0.0072 - accuracy: 0.9962 - val_loss: 0.1131 - val_accuracy: 0.9798
Epoch 9/15
13/13 [==============================] - 39s 3s/step - loss: 0.0056 - accuracy: 0.9987 - val_loss: 0.1050 - val_accuracy: 0.9646
Epoch 10/15
13/13 [==============================] - 23s 2s/step - loss: 0.0125 - accuracy: 0.9975 - val_loss: 0.2591 - val_accuracy: 0.9747
Epoch 11/15
13/13 [==============================] - 25s 2s/step - loss: 0.0521 - accuracy: 0.9924 - val_loss: 0.0738 - val_accuracy: 0.9798
Epoch 12/15
13/13 [==============================] - 25s 2s/step - loss: 0.0042 - accuracy: 0.9987 - val_loss: 0.0737 - val_accuracy: 0.9899
Epoch 13/15
13/13 [==============================] - 24s 2s/step - loss: 0.0014 - accuracy: 1.0000 - val_loss: 0.0775 - val_accuracy: 0.9848
Epoch 14/15
13/13 [==============================] - 26s 2s/step - loss: 6.9249e-04 - accuracy: 1.0000 - val_loss: 0.0683 - val_accuracy: 0.9899
Epoch 15/15
13/13 [==============================] - 29s 2s/step - loss: 5.0451e-04 - accuracy: 1.0000 - val_loss: 0.0643 - val_accuracy: 0.9899
Optimizer: Adam, Learning Rate: 0.001, Validation Accuracy: 98.99%
```

Here the choice of the learning rate 0.1 with Adam optimizer made the model not learn.

```
Epoch 1/15
13/13 [==============================] - 30s 2s/step - loss: 13019.2080 - accuracy: 0.0038 - val_loss: 4.6095 - val_accuracy: 0.0101
Epoch 2/15
13/13 [==============================] - 29s 2s/step - loss: 4.6797 - accuracy: 0.0076 - val_loss: 4.6092 - val_accuracy: 0.0101
Epoch 3/15
13/13 [==============================] - 26s 2s/step - loss: 7.5585 - accuracy: 0.0101 - val_loss: 4.6056 - val_accuracy: 0.0101
Epoch 4/15
13/13 [==============================] - 26s 2s/step - loss: 4.6219 - accuracy: 0.0088 - val_loss: 4.6038 - val_accuracy: 0.0101
Epoch 5/15
13/13 [==============================] - 26s 2s/step - loss: 7.5300 - accuracy: 0.0038 - val_loss: 4.6016 - val_accuracy: 0.0101
Epoch 6/15
13/13 [==============================] - 27s 2s/step - loss: 4.6217 - accuracy: 0.0063 - val_loss: 4.6012 - val_accuracy: 0.0101
Epoch 7/15
13/13 [==============================] - 26s 2s/step - loss: 4.6179 - accuracy: 0.0088 - val_loss: 4.6005 - val_accuracy: 0.0101
Epoch 8/15
13/13 [==============================] - 26s 2s/step - loss: 4.6216 - accuracy: 0.0038 - val_loss: 4.6018 - val_accuracy: 0.0101
Epoch 9/15
13/13 [==============================] - 30s 2s/step - loss: 4.6176 - accuracy: 0.0076 - val_loss: 4.6001 - val_accuracy: 0.0101
Epoch 10/15
13/13 [==============================] - 27s 2s/step - loss: 4.6188 - accuracy: 0.0101 - val_loss: 4.5997 - val_accuracy: 0.0101
Epoch 11/15
13/13 [==============================] - 26s 2s/step - loss: 4.6168 - accuracy: 0.0063 - val_loss: 4.5993 - val_accuracy: 0.0101
Epoch 12/15
13/13 [==============================] - 26s 2s/step - loss: 4.6174 - accuracy: 0.0038 - val_loss: 4.6003 - val_accuracy: 0.0101
Epoch 13/15
13/13 [==============================] - 28s 2s/step - loss: 4.6166 - accuracy: 0.0076 - val_loss: 4.5995 - val_accuracy: 0.0101
Epoch 14/15
13/13 [==============================] - 25s 2s/step - loss: 4.6153 - accuracy: 0.0101 - val_loss: 4.5989 - val_accuracy: 0.0101
Epoch 15/15
13/13 [==============================] - 29s 2s/step - loss: 4.6189 - accuracy: 0.0063 - val_loss: 4.5993 - val_accuracy: 0.0101
Optimizer: Adam, Learning Rate: 0.1, Validation Accuracy: 1.01%
```

These are the vlaues for RMSprop with differnet learning rates

```
Epoch 1/15
13/13 [==============================] - 26s 2s/step - loss: 3.7152 - accuracy: 0.1881 - val_loss: 2.9181 - val_accuracy: 0.3535
Epoch 2/15
13/13 [==============================] - 26s 2s/step - loss: 1.0037 - accuracy: 0.7487 - val_loss: 0.7159 - val_accuracy: 0.8182
Epoch 3/15
13/13 [==============================] - 24s 2s/step - loss: 0.2360 - accuracy: 0.9470 - val_loss: 0.3561 - val_accuracy: 0.9242
Epoch 4/15
13/13 [==============================] - 25s 2s/step - loss: 0.1358 - accuracy: 0.9634 - val_loss: 0.6360 - val_accuracy: 0.8788
Epoch 5/15
13/13 [==============================] - 33s 3s/step - loss: 0.0835 - accuracy: 0.9760 - val_loss: 0.0820 - val_accuracy: 0.9798
Epoch 6/15
13/13 [==============================] - 25s 2s/step - loss: 0.0230 - accuracy: 0.9924 - val_loss: 0.0586 - val_accuracy: 0.9798
Epoch 7/15
13/13 [==============================] - 25s 2s/step - loss: 0.0266 - accuracy: 0.9949 - val_loss: 0.0945 - val_accuracy: 0.9848
Epoch 8/15
13/13 [==============================] - 26s 2s/step - loss: 0.0076 - accuracy: 0.9987 - val_loss: 0.0729 - val_accuracy: 0.9848
Epoch 9/15
13/13 [==============================] - 24s 2s/step - loss: 0.0122 - accuracy: 0.9975 - val_loss: 0.0597 - val_accuracy: 0.9798
Epoch 10/15
13/13 [==============================] - 25s 2s/step - loss: 0.0269 - accuracy: 0.9949 - val_loss: 0.5088 - val_accuracy: 0.9091
Epoch 11/15
13/13 [==============================] - 25s 2s/step - loss: 0.0782 - accuracy: 0.9886 - val_loss: 0.0622 - val_accuracy: 0.9899
Epoch 12/15
13/13 [==============================] - 24s 2s/step - loss: 0.0022 - accuracy: 1.0000 - val_loss: 0.0518 - val_accuracy: 0.9848
Epoch 13/15
13/13 [==============================] - 27s 2s/step - loss: 8.6455e-04 - accuracy: 1.0000 - val_loss: 0.0534 - val_accuracy: 0.9848
Epoch 14/15
13/13 [==============================] - 23s 2s/step - loss: 5.4233e-04 - accuracy: 1.0000 - val_loss: 0.0514 - val_accuracy: 0.9848
Epoch 15/15
13/13 [==============================] - 26s 2s/step - loss: 4.1337e-04 - accuracy: 1.0000 - val_loss: 0.0518 - val_accuracy: 0.9848
Optimizer: RMSprop, Learning Rate: 0.001, Validation Accuracy: 98.48%
```

```
Epoch 1/15
13/13 [==============================] - 26s 2s/step - loss: 49.0588 - accuracy: 0.0480 - val_loss: 3.9235 - val_accuracy: 0.1717
Epoch 2/15
13/13 [==============================] - 24s 2s/step - loss: 2.6718 - accuracy: 0.3409 - val_loss: 1.4293 - val_accuracy: 0.5960
Epoch 3/15
13/13 [==============================] - 27s 2s/step - loss: 0.9888 - accuracy: 0.7159 - val_loss: 1.1360 - val_accuracy: 0.7020
Epoch 4/15
13/13 [==============================] - 21s 2s/step - loss: 0.5984 - accuracy: 0.8447 - val_loss: 0.4944 - val_accuracy: 0.8687
Epoch 5/15
13/13 [==============================] - 23s 2s/step - loss: 0.5056 - accuracy: 0.8775 - val_loss: 1.8190 - val_accuracy: 0.6970
Epoch 6/15
13/13 [==============================] - 22s 2s/step - loss: 0.3986 - accuracy: 0.9141 - val_loss: 0.6834 - val_accuracy: 0.8838
Epoch 7/15
13/13 [==============================] - 25s 2s/step - loss: 0.2044 - accuracy: 0.9609 - val_loss: 0.3285 - val_accuracy: 0.9242
Epoch 8/15
13/13 [==============================] - 23s 2s/step - loss: 0.5369 - accuracy: 0.9040 - val_loss: 2.2566 - val_accuracy: 0.7020
Epoch 9/15
13/13 [==============================] - 24s 2s/step - loss: 0.4253 - accuracy: 0.9268 - val_loss: 0.7497 - val_accuracy: 0.9242
Epoch 10/15
13/13 [==============================] - 22s 2s/step - loss: 0.1006 - accuracy: 0.9785 - val_loss: 0.1303 - val_accuracy: 0.9697
Epoch 11/15
13/13 [==============================] - 27s 2s/step - loss: 0.0048 - accuracy: 0.9975 - val_loss: 0.2064 - val_accuracy: 0.9596
Epoch 12/15
13/13 [==============================] - 23s 2s/step - loss: 8.2922e-04 - accuracy: 1.0000 - val_loss: 0.1694 - val_accuracy: 0.9596
Epoch 13/15
13/13 [==============================] - 24s 2s/step - loss: 0.5887 - accuracy: 0.9432 - val_loss: 2.4831 - val_accuracy: 0.6717
Epoch 14/15
13/13 [==============================] - 23s 2s/step - loss: 0.5831 - accuracy: 0.9217 - val_loss: 0.2447 - val_accuracy: 0.9343
Epoch 15/15
13/13 [==============================] - 25s 2s/step - loss: 0.0735 - accuracy: 0.9848 - val_loss: 0.1562 - val_accuracy: 0.9596
Optimizer: RMSprop, Learning Rate: 0.01, Validation Accuracy: 95.96%
```

```
Epoch 1/15
13/13 [==============================] - 26s 2s/step - loss: 56017.2070 - accuracy: 0.0114 - val_loss: 12.2069 - val_accuracy: 0.0657
Epoch 2/15
13/13 [==============================] - 23s 2s/step - loss: 7.1413 - accuracy: 0.0934 - val_loss: 5.1623 - val_accuracy: 0.1515
Epoch 3/15
13/13 [==============================] - 24s 2s/step - loss: 5.5381 - accuracy: 0.1944 - val_loss: 3.9549 - val_accuracy: 0.2172
Epoch 4/15
13/13 [==============================] - 26s 2s/step - loss: 3.7594 - accuracy: 0.2563 - val_loss: 2.9000 - val_accuracy: 0.3687
Epoch 5/15
13/13 [==============================] - 23s 2s/step - loss: 4.3099 - accuracy: 0.2980 - val_loss: 4.8522 - val_accuracy: 0.2677
Epoch 6/15
13/13 [==============================] - 21s 2s/step - loss: 8.1198 - accuracy: 0.1515 - val_loss: 5.4197 - val_accuracy: 0.2222
Epoch 7/15
13/13 [==============================] - 23s 2s/step - loss: 5.3500 - accuracy: 0.1250 - val_loss: 5.6733 - val_accuracy: 0.1515
Epoch 8/15
13/13 [==============================] - 21s 2s/step - loss: 10.9419 - accuracy: 0.0808 - val_loss: 6.6217 - val_accuracy: 0.0859
Epoch 9/15
13/13 [==============================] - 22s 2s/step - loss: 6.6156 - accuracy: 0.0657 - val_loss: 5.0785 - val_accuracy: 0.0152
Epoch 10/15
13/13 [==============================] - 22s 2s/step - loss: 12.2534 - accuracy: 0.0126 - val_loss: 4.6764 - val_accuracy: 0.0101
Epoch 11/15
13/13 [==============================] - 22s 2s/step - loss: 4.6915 - accuracy: 0.0063 - val_loss: 4.6311 - val_accuracy: 0.0101
Epoch 12/15
13/13 [==============================] - 26s 2s/step - loss: 4.6873 - accuracy: 0.0063 - val_loss: 4.6137 - val_accuracy: 0.0101
Epoch 13/15
13/13 [==============================] - 21s 2s/step - loss: 10.0536 - accuracy: 0.0088 - val_loss: 4.7847 - val_accuracy: 0.0101
Epoch 14/15
13/13 [==============================] - 23s 2s/step - loss: 7.3702 - accuracy: 0.0051 - val_loss: 4.6127 - val_accuracy: 0.0101
Epoch 15/15
13/13 [==============================] - 21s 2s/step - loss: 4.6475 - accuracy: 0.0063 - val_loss: 4.6050 - val_accuracy: 0.0101
Optimizer: RMSprop, Learning Rate: 0.1, Validation Accuracy: 1.01%
```

We can see that increasing the learning rates with the RMSprop optimizer reduces the socre of the model untill it stops the learning process.

So we choose our best model which was Adam with learning rate 0.001 and tried to change the batch size and see the results.

```python
# tuning adam with different batch sizes
optimizer = Adam(learning_rate=0.001)

best_model = Sequential()
best_model.add(Convolution1D(512, 1, input_shape=(nb_features, 3)))
best_model.add(Activation('relu'))
best_model.add(Flatten())
best_model.add(Dropout(0.4))
best_model.add(Dense(2048, activation='relu'))
best_model.add(Dense(1024, activation='relu'))
best_model.add(Dense(nb_class, activation='softmax'))

best_model.compile(loss='categorical_crossentropy', optimizer=optimizer, metrics=['accuracy'])

# Train the model with the current (parameter) epochs: int
best_model.fit(X_train_r, y_train, epochs=nb_epoch, validation_data=(X_valid_r, y_valid), batch_size=32)

# Evaluate the model
evaluation = best_model.evaluate(X_valid_r, y_valid, verbose=0)
accuracy = evaluation[1]
print(f'Batch Size: {32}, Validation Accuracy: {accuracy * 100:.2f}%')
```

```
Epoch 1/15
25/25 [==============================] - 42s 2s/step - loss: 3.2477 - accuracy: 0.3662 - val_loss: 1.1003 - val_accuracy: 0.7273
Epoch 2/15
25/25 [==============================] - 41s 2s/step - loss: 0.5188 - accuracy: 0.8788 - val_loss: 0.5943 - val_accuracy: 0.8535
Epoch 3/15
25/25 [==============================] - 40s 2s/step - loss: 0.2476 - accuracy: 0.9381 - val_loss: 0.2010 - val_accuracy: 0.9545
Epoch 4/15
25/25 [==============================] - 42s 2s/step - loss: 0.0583 - accuracy: 0.9874 - val_loss: 0.1065 - val_accuracy: 0.9646
Epoch 5/15
25/25 [==============================] - 39s 2s/step - loss: 0.0393 - accuracy: 0.9937 - val_loss: 0.2575 - val_accuracy: 0.9394
Epoch 6/15
25/25 [==============================] - 43s 2s/step - loss: 0.0425 - accuracy: 0.9924 - val_loss: 0.1145 - val_accuracy: 0.9646
Epoch 7/15
25/25 [==============================] - 43s 2s/step - loss: 0.0129 - accuracy: 0.9962 - val_loss: 0.0550 - val_accuracy: 0.9848
Epoch 8/15
25/25 [==============================] - 42s 2s/step - loss: 0.0095 - accuracy: 0.9949 - val_loss: 0.0856 - val_accuracy: 0.9798
Epoch 9/15
25/25 [==============================] - 40s 2s/step - loss: 0.0028 - accuracy: 1.0000 - val_loss: 0.0558 - val_accuracy: 0.9848
Epoch 10/15
25/25 [==============================] - 42s 2s/step - loss: 6.6693e-04 - accuracy: 1.0000 - val_loss: 0.0739 - val_accuracy: 0.9848
Epoch 11/15
25/25 [==============================] - 39s 2s/step - loss: 4.7949e-04 - accuracy: 1.0000 - val_loss: 0.0471 - val_accuracy: 0.9899
Epoch 12/15
25/25 [==============================] - 41s 2s/step - loss: 3.1404e-04 - accuracy: 1.0000 - val_loss: 0.0502 - val_accuracy: 0.9899
Epoch 13/15
25/25 [==============================] - 41s 2s/step - loss: 2.5178e-04 - accuracy: 1.0000 - val_loss: 0.0497 - val_accuracy: 0.9899
Epoch 14/15
25/25 [==============================] - 41s 2s/step - loss: 2.4190e-04 - accuracy: 1.0000 - val_loss: 0.0508 - val_accuracy: 0.9899
Epoch 15/15
25/25 [==============================] - 43s 2s/step - loss: 1.7988e-04 - accuracy: 1.0000 - val_loss: 0.0570 - val_accuracy: 0.9899
Batch Size: 32, Validation Accuracy: 98.99%
```

We achieved the same results as the 64 batch size

We then used the test set to predict the output of species

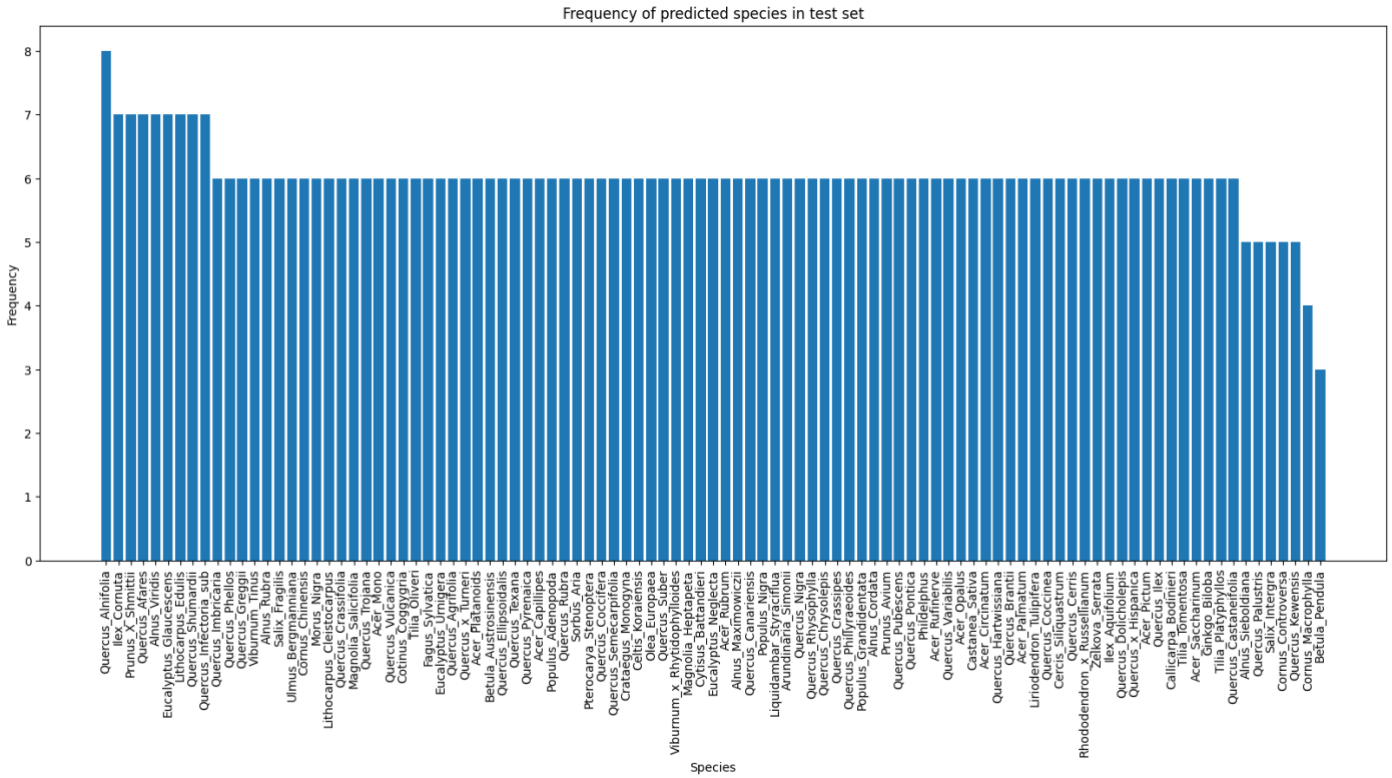
```
[22] scaler = StandardScaler().fit(test.values)
     scaled_test = scaler.transform(test.values)

[23] test_dataset = np.zeros((len(scaled_test), nb_features, 3))
     test_dataset[:, :, 0] = scaled_test[:, :nb_features]
     test_dataset[:, :, 1] = scaled_test[:, nb_features:128]
     test_dataset[:, :, 2] = scaled_test[:, 128:]

     preds_test = best_model.predict(test_dataset)
     preds_test

     19/19 [==============================] - 3s 128ms/step
     array([[[6.29572270e-08, 9.27303176e-07, 2.94246814e-11, ...,
              7.80022802e-09, 9.17741716e-07, 8.68282086e-08],
             [3.46198092e-08, 7.44975495e-08, 1.57151902e-07, ...,
              2.04599814e-06, 1.60488656e-09, 4.77554522e-07],
             [6.94620983e-09, 9.99670863e-01, 4.58709841e-11, ...,
              1.05751824e-10, 3.52248342e-10, 4.68155895e-06],
             ...,
             [6.99989258e-08, 1.26810917e-07, 7.99961042e-10, ...,
              1.17515930e-09, 6.45039577e-09, 1.16469828e-07],
             [1.23316149e-10, 1.06381903e-09, 2.36864958e-06, ...,
              1.04182718e-09, 1.13895707e-11, 2.85817325e-09],
             [1.28978106e-10, 6.13536733e-09, 1.56060711e-08, ...,
              9.68338369e-08, 1.07057119e-08, 5.69028735e-08]], dtype=float32)
```

We then plotted a histogram of the species predicted and their counts.



And this is the classification report of the accuracy of the model on the validation data.

accuracy			0.99	198
macro avg	0.99	0.99	0.99	198
weighted avg	0.99	0.99	0.99	198