Predicting the Rings (Age) of abalone

Final Project for MATH 156

Wesley Bian

In [ ]:
```
import pandas as pd
import matplotlib.pyplot as plt
```

Exploratory Data Analysis

In [ ]:
```
df = pd.read_csv("abalone.csv")
df.head()
```

Out[ ]:

	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings
0	M	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.150	15
1	M	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.070	7
2	F	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.210	9
3	M	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.155	10
4	I	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.055	7

In [ ]:
```
print("Rows: " + str(df.shape[0]))
print("Colums: " + str(df.shape[1]))
```

```
Rows: 4177
Colums: 9
```

In [ ]:
```
print("unique ring values:", len(df.value_counts("Rings").index))
print("Highest number of rings:", max(df["Rings"]))
```

```
unique ring values: 28
Highest number of rings: 29
```

28 different abalone Ring values are represented in the dataset. The largest one is 29.

In [ ]:
```
round(df.describe(), 2)
```

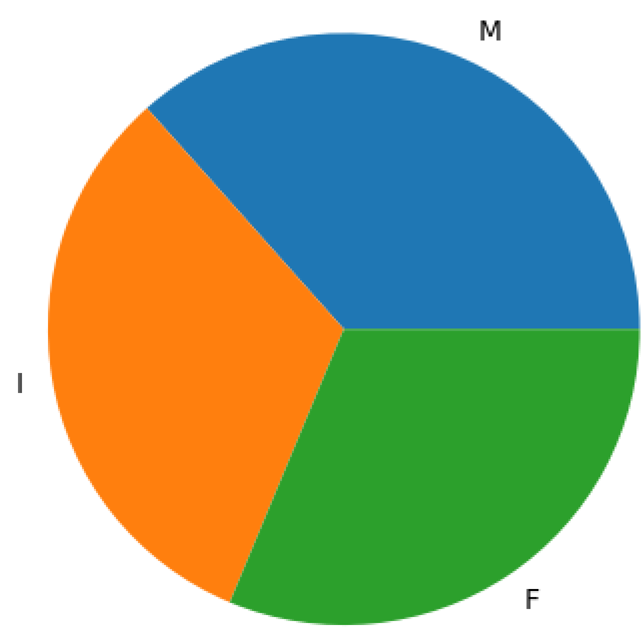
Out[ ]:

	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings
count	4177.00	4177.00	4177.00	4177.00	4177.00	4177.00	4177.00	4177.00
mean	0.52	0.41	0.14	0.83	0.36	0.18	0.24	9.93
std	0.12	0.10	0.04	0.49	0.22	0.11	0.14	3.22
min	0.08	0.06	0.00	0.00	0.00	0.00	0.00	1.00
25%	0.45	0.35	0.12	0.44	0.19	0.09	0.13	8.00
50%	0.55	0.42	0.14	0.80	0.34	0.17	0.23	9.00
75%	0.62	0.48	0.16	1.15	0.50	0.25	0.33	11.00
max	0.82	0.65	1.13	2.83	1.49	0.76	1.00	29.00

In [ ]:

```
gender_nums = df["Sex"].value_counts()
labels = gender_nums.index
values = gender_nums.values
plt.pie(values, labels = labels)
plt.show()

round(df.groupby("Sex").mean(), 2)
```



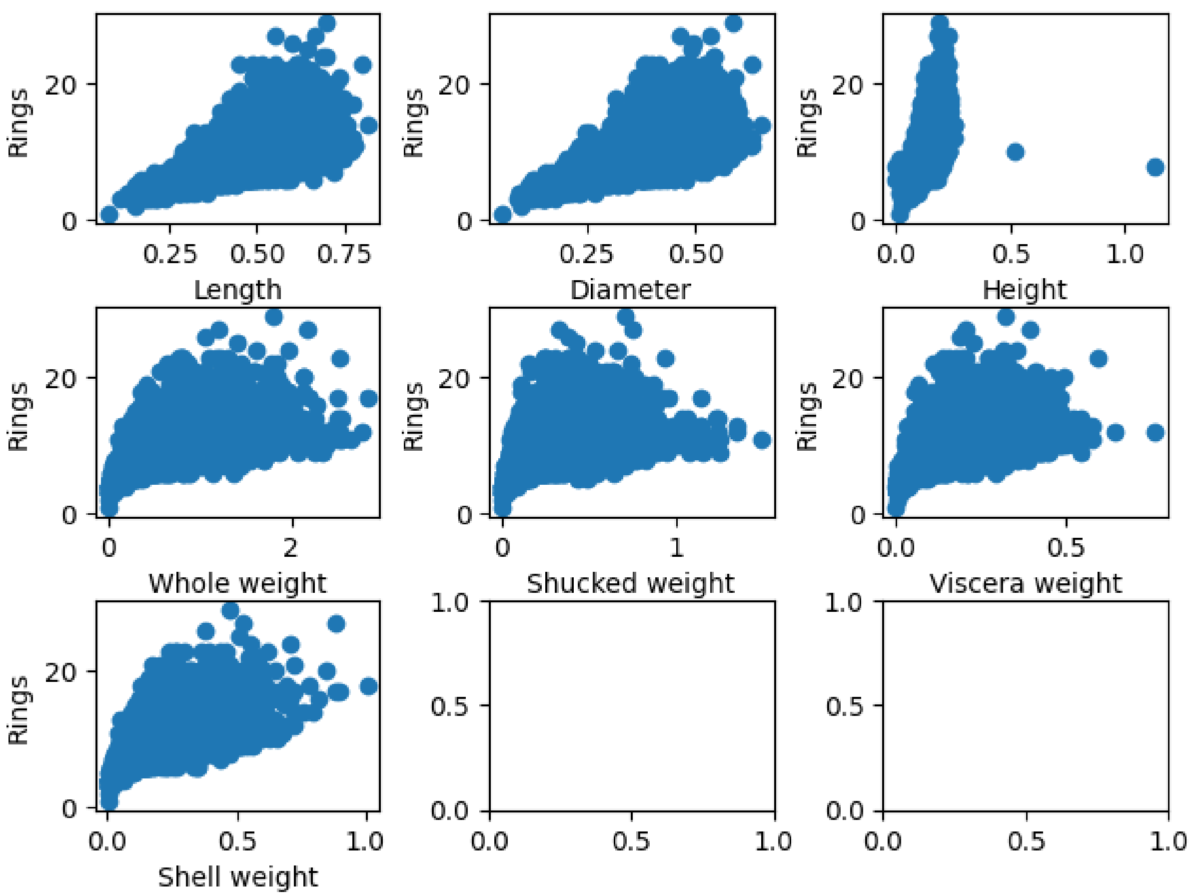
Out[ ]:

	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings
Sex								
F	0.58	0.45	0.16	1.05	0.45	0.23	0.30	11.13
I	0.43	0.33	0.11	0.43	0.19	0.09	0.13	7.89
M	0.56	0.44	0.15	0.99	0.43	0.22	0.28	10.71

Data is evenly distributed between M = Male, F = Female, I = Infant. Generally, it seems that all measurements are on average lowest for Infant, in the middle for Male, and highest for Female.

In [ ]:

```
fig, ax = plt.subplots(3, 3)
fig.tight_layout()
explanatory_vars = df.drop(["Sex", "Rings"], axis = 1)
for i, col in enumerate(explanatory_vars.columns):
    ax[i // 3, i % 3].scatter(explanatory_vars[col], df["Rings"])
    ax[i // 3, i % 3].set_xlabel(col)
    ax[i // 3, i % 3].set_ylabel("Rings")
    #plt.scatter(df[col], df["Age"])
    #plt.show()
plt.show()
```



In [ ]:
```
#df.boxplot("Rings", "Sex")
```

There is generally a positive correlation between the variables and the Ring number. The correlation seems to be the strongest for Height vs Rings.

Data Preprocessing

Sex must be one hot encoded.

In [ ]:
```
cat_list = pd.get_dummies(df["Sex"], prefix = "Sex", dtype = float)
df = df.join(cat_list)
df = df.drop("Sex", axis = 1)
df.head()
```

Out[ ]:

	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings	Sex_F	Sex_I	Sex_
0	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.150	15	0.0	0.0	
1	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.070	7	0.0	0.0	
2	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.210	9	1.0	0.0	
3	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.155	10	0.0	0.0	
4	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.055	7	0.0	1.0	

We are predicting the Rings value, and in order to use the multi-class classification functionality of a neural network as learned in class, we must treat the Rings value as a categorical variable. However, our model may face accuracy issues because if predictions are 6 and 1000 when the true number of Rings is 7, both inaccurate predictions are penalized equally because 6 and 1000 are different categories than 7. However, we want to take into account that 6 is close to 7, and this prediction would have value. Therefore, we will turn the output into classes of groups of 4, so label 0 represents 0-3 Rings, 1 represents 4-7 Rings, 2 represents 8 to 11 rings, and so on. This way, our accuracy is improved while still generating useful predictions.

In [ ]:
```
#transform y as described above.
df["group"] = df["Rings"] // 4
df.head()
```

Out[ ]:

	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings	Sex_F	Sex_I	Sex_
0	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.150	15	0.0	0.0	
1	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.070	7	0.0	0.0	
2	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.210	9	1.0	0.0	
3	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.155	10	0.0	0.0	
4	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.055	7	0.0	1.0	

A min max scaler is applied to each column to standardize measurements.

In [ ]:

```
df = df.drop("Rings", axis = 1)
x = df.iloc[:, df.columns != "group"].values
y = df.iloc[:, df.columns == "group"].values

from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
x = sc.fit_transform(x)
```

In [ ]:

```
from tensorflow.keras.utils import to_categorical
import numpy as np
groups_present = np.unique(y)
num_groups = len(groups_present)
y = to_categorical(y, num_classes = num_groups)
```

Training the model

In [ ]:

```
from keras.models import Sequential
from keras.layers import Dense

from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2)

model = Sequential()
model.add(Dense(15, input_dim=10, activation="relu"))
model.add(Dense(12, activation="relu"))
model.add(Dense(num_groups, activation="softmax"))
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy
history = model.fit(x_train, y_train, validation_data = (x_test, y_test), epochs=50
```

```
Epoch 1/50
53/53 [==============================] - 7s 34ms/step - loss: 1.7624 - accuracy: 0.5
391 - val_loss: 1.5190 - val_accuracy: 0.5825
Epoch 2/50
53/53 [==============================] - 0s 9ms/step - loss: 1.3836 - accuracy: 0.56
60 - val_loss: 1.2092 - val_accuracy: 0.5825
Epoch 3/50
53/53 [==============================] - 1s 13ms/step - loss: 1.1524 - accuracy: 0.6
438 - val_loss: 1.0425 - val_accuracy: 0.6663
Epoch 4/50
53/53 [==============================] - 1s 14ms/step - loss: 1.0346 - accuracy: 0.6
630 - val_loss: 0.9547 - val_accuracy: 0.6758
Epoch 5/50
53/53 [==============================] - 1s 14ms/step - loss: 0.9734 - accuracy: 0.6
648 - val_loss: 0.9096 - val_accuracy: 0.6770
Epoch 6/50
53/53 [==============================] - 1s 9ms/step - loss: 0.9392 - accuracy: 0.66
57 - val_loss: 0.8785 - val_accuracy: 0.6770
Epoch 7/50
53/53 [==============================] - 1s 12ms/step - loss: 0.9151 - accuracy: 0.6
699 - val_loss: 0.8582 - val_accuracy: 0.6818
Epoch 8/50
53/53 [==============================] - 1s 18ms/step - loss: 0.8952 - accuracy: 0.6
702 - val_loss: 0.8431 - val_accuracy: 0.6830
Epoch 9/50
53/53 [==============================] - 0s 8ms/step - loss: 0.8800 - accuracy: 0.67
35 - val_loss: 0.8306 - val_accuracy: 0.6794
Epoch 10/50
53/53 [==============================] - 1s 22ms/step - loss: 0.8680 - accuracy: 0.6
752 - val_loss: 0.8220 - val_accuracy: 0.6794
Epoch 11/50
53/53 [==============================] - 0s 9ms/step - loss: 0.8582 - accuracy: 0.67
97 - val_loss: 0.8136 - val_accuracy: 0.6782
Epoch 12/50
53/53 [==============================] - 0s 7ms/step - loss: 0.8492 - accuracy: 0.68
06 - val_loss: 0.8062 - val_accuracy: 0.6794
Epoch 13/50
53/53 [==============================] - 0s 9ms/step - loss: 0.8400 - accuracy: 0.68
27 - val_loss: 0.8008 - val_accuracy: 0.6866
Epoch 14/50
53/53 [==============================] - 0s 6ms/step - loss: 0.8312 - accuracy: 0.68
45 - val_loss: 0.7945 - val_accuracy: 0.6794
Epoch 15/50
53/53 [==============================] - 0s 8ms/step - loss: 0.8242 - accuracy: 0.68
60 - val_loss: 0.7905 - val_accuracy: 0.6818
Epoch 16/50
53/53 [==============================] - 0s 7ms/step - loss: 0.8180 - accuracy: 0.68
66 - val_loss: 0.7827 - val_accuracy: 0.6878
Epoch 17/50
53/53 [==============================] - 0s 9ms/step - loss: 0.8123 - accuracy: 0.68
63 - val_loss: 0.7778 - val_accuracy: 0.6854
Epoch 18/50
53/53 [==============================] - 0s 8ms/step - loss: 0.8067 - accuracy: 0.68
39 - val_loss: 0.7733 - val_accuracy: 0.6866
Epoch 19/50
53/53 [==============================] - 0s 8ms/step - loss: 0.8025 - accuracy: 0.68
```

```
48 - val_loss: 0.7734 - val_accuracy: 0.6938
Epoch 20/50
53/53 [==============================] - 0s 9ms/step - loss: 0.7970 - accuracy: 0.68
39 - val_loss: 0.7725 - val_accuracy: 0.6926
Epoch 21/50
53/53 [==============================] - 0s 8ms/step - loss: 0.7944 - accuracy: 0.68
93 - val_loss: 0.7663 - val_accuracy: 0.6962
Epoch 22/50
53/53 [==============================] - 1s 11ms/step - loss: 0.7903 - accuracy: 0.6
896 - val_loss: 0.7672 - val_accuracy: 0.6914
Epoch 23/50
53/53 [==============================] - 1s 13ms/step - loss: 0.7854 - accuracy: 0.6
926 - val_loss: 0.7636 - val_accuracy: 0.6914
Epoch 24/50
53/53 [==============================] - 1s 12ms/step - loss: 0.7845 - accuracy: 0.6
911 - val_loss: 0.7605 - val_accuracy: 0.6974
Epoch 25/50
53/53 [==============================] - 1s 20ms/step - loss: 0.7794 - accuracy: 0.6
929 - val_loss: 0.7582 - val_accuracy: 0.6950
Epoch 26/50
53/53 [==============================] - 1s 12ms/step - loss: 0.7780 - accuracy: 0.6
899 - val_loss: 0.7586 - val_accuracy: 0.7045
Epoch 27/50
53/53 [==============================] - 1s 12ms/step - loss: 0.7742 - accuracy: 0.6
932 - val_loss: 0.7577 - val_accuracy: 0.7033
Epoch 28/50
53/53 [==============================] - 1s 14ms/step - loss: 0.7730 - accuracy: 0.6
956 - val_loss: 0.7595 - val_accuracy: 0.7057
Epoch 29/50
53/53 [==============================] - 0s 7ms/step - loss: 0.7710 - accuracy: 0.69
44 - val_loss: 0.7535 - val_accuracy: 0.6890
Epoch 30/50
53/53 [==============================] - 1s 13ms/step - loss: 0.7671 - accuracy: 0.6
995 - val_loss: 0.7521 - val_accuracy: 0.6962
Epoch 31/50
53/53 [==============================] - 0s 8ms/step - loss: 0.7674 - accuracy: 0.68
72 - val_loss: 0.7516 - val_accuracy: 0.6914
Epoch 32/50
53/53 [==============================] - 0s 9ms/step - loss: 0.7643 - accuracy: 0.69
62 - val_loss: 0.7515 - val_accuracy: 0.6938
Epoch 33/50
53/53 [==============================] - 1s 14ms/step - loss: 0.7637 - accuracy: 0.6
956 - val_loss: 0.7532 - val_accuracy: 0.6998
Epoch 34/50
53/53 [==============================] - 0s 9ms/step - loss: 0.7618 - accuracy: 0.69
32 - val_loss: 0.7537 - val_accuracy: 0.6986
Epoch 35/50
53/53 [==============================] - 1s 28ms/step - loss: 0.7584 - accuracy: 0.6
956 - val_loss: 0.7502 - val_accuracy: 0.6986
Epoch 36/50
53/53 [==============================] - 1s 19ms/step - loss: 0.7592 - accuracy: 0.6
974 - val_loss: 0.7478 - val_accuracy: 0.6950
Epoch 37/50
53/53 [==============================] - 1s 13ms/step - loss: 0.7568 - accuracy: 0.6
968 - val_loss: 0.7490 - val_accuracy: 0.6998
Epoch 38/50
```

```
53/53 [==============================] - 1s 22ms/step - loss: 0.7556 - accuracy: 0.6
977 - val_loss: 0.7487 - val_accuracy: 0.6986
Epoch 39/50
53/53 [==============================] - 1s 12ms/step - loss: 0.7537 - accuracy: 0.7
001 - val_loss: 0.7510 - val_accuracy: 0.6926
Epoch 40/50
53/53 [==============================] - 1s 11ms/step - loss: 0.7542 - accuracy: 0.6
965 - val_loss: 0.7498 - val_accuracy: 0.6986
Epoch 41/50
53/53 [==============================] - 1s 16ms/step - loss: 0.7502 - accuracy: 0.6
977 - val_loss: 0.7510 - val_accuracy: 0.6998
Epoch 42/50
53/53 [==============================] - 1s 12ms/step - loss: 0.7504 - accuracy: 0.6
992 - val_loss: 0.7481 - val_accuracy: 0.7010
Epoch 43/50
53/53 [==============================] - 1s 10ms/step - loss: 0.7497 - accuracy: 0.6
959 - val_loss: 0.7517 - val_accuracy: 0.6938
Epoch 44/50
53/53 [==============================] - 2s 37ms/step - loss: 0.7485 - accuracy: 0.6
983 - val_loss: 0.7465 - val_accuracy: 0.6950
Epoch 45/50
53/53 [==============================] - 3s 66ms/step - loss: 0.7480 - accuracy: 0.6
950 - val_loss: 0.7462 - val_accuracy: 0.6962
Epoch 46/50
53/53 [==============================] - 2s 32ms/step - loss: 0.7461 - accuracy: 0.6
971 - val_loss: 0.7469 - val_accuracy: 0.6962
Epoch 47/50
53/53 [==============================] - 1s 21ms/step - loss: 0.7451 - accuracy: 0.6
977 - val_loss: 0.7494 - val_accuracy: 0.6962
Epoch 48/50
53/53 [==============================] - 1s 15ms/step - loss: 0.7452 - accuracy: 0.6
998 - val_loss: 0.7467 - val_accuracy: 0.6974
Epoch 49/50
53/53 [==============================] - 1s 19ms/step - loss: 0.7445 - accuracy: 0.7
019 - val_loss: 0.7503 - val_accuracy: 0.7045
Epoch 50/50
53/53 [==============================] - 1s 21ms/step - loss: 0.7442 - accuracy: 0.7
001 - val_loss: 0.7460 - val_accuracy: 0.6950
```

In this first model, the accuracy peaks around 70% after 50 epochs. A different combination of layers might work better

In [ ]:

```
model2 = Sequential()
model2.add(Dense(15, input_dim=10, activation="relu"))
model2.add(Dense(13, activation="relu"))
model2.add(Dense(11, activation="relu"))
model2.add(Dense(9, activation="relu"))
model2.add(Dense(num_groups, activation="softmax"))
model2.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accurac
history2 = model2.fit(x_train, y_train, validation_data = (x_test, y_test), epochs=
```

```
Epoch 1/100
14/14 [==============================] - 9s 129ms/step - loss: 2.0082 - accuracy: 0.
3349 - val_loss: 1.9406 - val_accuracy: 0.4737
Epoch 2/100
14/14 [==============================] - 0s 17ms/step - loss: 1.8843 - accuracy: 0.5
459 - val_loss: 1.8066 - val_accuracy: 0.5825
Epoch 3/100
14/14 [==============================] - 0s 17ms/step - loss: 1.7397 - accuracy: 0.5
660 - val_loss: 1.6572 - val_accuracy: 0.5825
Epoch 4/100
14/14 [==============================] - 0s 19ms/step - loss: 1.5872 - accuracy: 0.5
660 - val_loss: 1.5121 - val_accuracy: 0.5825
Epoch 5/100
14/14 [==============================] - 1s 52ms/step - loss: 1.4461 - accuracy: 0.5
660 - val_loss: 1.3621 - val_accuracy: 0.5825
Epoch 6/100
14/14 [==============================] - 1s 67ms/step - loss: 1.2982 - accuracy: 0.5
705 - val_loss: 1.2031 - val_accuracy: 0.6663
Epoch 7/100
14/14 [==============================] - 0s 27ms/step - loss: 1.1472 - accuracy: 0.6
609 - val_loss: 1.0645 - val_accuracy: 0.6555
Epoch 8/100
14/14 [==============================] - 0s 20ms/step - loss: 1.0490 - accuracy: 0.6
450 - val_loss: 0.9952 - val_accuracy: 0.6567
Epoch 9/100
14/14 [==============================] - 1s 38ms/step - loss: 0.9978 - accuracy: 0.6
606 - val_loss: 0.9520 - val_accuracy: 0.6770
Epoch 10/100
14/14 [==============================] - 1s 60ms/step - loss: 0.9633 - accuracy: 0.6
645 - val_loss: 0.9176 - val_accuracy: 0.6794
Epoch 11/100
14/14 [==============================] - 1s 53ms/step - loss: 0.9373 - accuracy: 0.6
696 - val_loss: 0.8916 - val_accuracy: 0.6806
Epoch 12/100
14/14 [==============================] - 0s 31ms/step - loss: 0.9170 - accuracy: 0.6
699 - val_loss: 0.8744 - val_accuracy: 0.6746
Epoch 13/100
14/14 [==============================] - 1s 61ms/step - loss: 0.9020 - accuracy: 0.6
684 - val_loss: 0.8614 - val_accuracy: 0.6818
Epoch 14/100
14/14 [==============================] - 1s 75ms/step - loss: 0.8891 - accuracy: 0.6
705 - val_loss: 0.8527 - val_accuracy: 0.6818
Epoch 15/100
14/14 [==============================] - 0s 24ms/step - loss: 0.8810 - accuracy: 0.6
696 - val_loss: 0.8450 - val_accuracy: 0.6806
Epoch 16/100
14/14 [==============================] - 0s 15ms/step - loss: 0.8717 - accuracy: 0.6
761 - val_loss: 0.8375 - val_accuracy: 0.6818
Epoch 17/100
14/14 [==============================] - 1s 49ms/step - loss: 0.8635 - accuracy: 0.6
764 - val_loss: 0.8314 - val_accuracy: 0.6818
Epoch 18/100
14/14 [==============================] - 0s 37ms/step - loss: 0.8577 - accuracy: 0.6
851 - val_loss: 0.8240 - val_accuracy: 0.6890
Epoch 19/100
14/14 [==============================] - 0s 18ms/step - loss: 0.8495 - accuracy: 0.6
```

```
815 - val_loss: 0.8190 - val_accuracy: 0.6842
Epoch 20/100
14/14 [==============================] - 0s 19ms/step - loss: 0.8437 - accuracy: 0.6
827 - val_loss: 0.8144 - val_accuracy: 0.6866
Epoch 21/100
14/14 [==============================] - 0s 15ms/step - loss: 0.8442 - accuracy: 0.6
752 - val_loss: 0.8118 - val_accuracy: 0.6854
Epoch 22/100
14/14 [==============================] - 1s 38ms/step - loss: 0.8361 - accuracy: 0.6
824 - val_loss: 0.8064 - val_accuracy: 0.6914
Epoch 23/100
14/14 [==============================] - 1s 78ms/step - loss: 0.8279 - accuracy: 0.6
893 - val_loss: 0.8036 - val_accuracy: 0.6926
Epoch 24/100
14/14 [==============================] - 1s 57ms/step - loss: 0.8235 - accuracy: 0.6
887 - val_loss: 0.8003 - val_accuracy: 0.6902
Epoch 25/100
14/14 [==============================] - 0s 31ms/step - loss: 0.8219 - accuracy: 0.6
845 - val_loss: 0.7961 - val_accuracy: 0.6926
Epoch 26/100
14/14 [==============================] - 0s 34ms/step - loss: 0.8172 - accuracy: 0.6
872 - val_loss: 0.7931 - val_accuracy: 0.6902
Epoch 27/100
14/14 [==============================] - 1s 65ms/step - loss: 0.8145 - accuracy: 0.6
875 - val_loss: 0.7921 - val_accuracy: 0.6902
Epoch 28/100
14/14 [==============================] - 0s 19ms/step - loss: 0.8111 - accuracy: 0.6
842 - val_loss: 0.7901 - val_accuracy: 0.6938
Epoch 29/100
14/14 [==============================] - 0s 21ms/step - loss: 0.8076 - accuracy: 0.6
902 - val_loss: 0.7884 - val_accuracy: 0.6962
Epoch 30/100
14/14 [==============================] - 1s 51ms/step - loss: 0.8056 - accuracy: 0.6
869 - val_loss: 0.7866 - val_accuracy: 0.6902
Epoch 31/100
14/14 [==============================] - 1s 42ms/step - loss: 0.8019 - accuracy: 0.6
878 - val_loss: 0.7832 - val_accuracy: 0.6926
Epoch 32/100
14/14 [==============================] - 1s 53ms/step - loss: 0.7996 - accuracy: 0.6
884 - val_loss: 0.7813 - val_accuracy: 0.6986
Epoch 33/100
14/14 [==============================] - 0s 21ms/step - loss: 0.7962 - accuracy: 0.6
896 - val_loss: 0.7808 - val_accuracy: 0.6986
Epoch 34/100
14/14 [==============================] - 1s 42ms/step - loss: 0.7944 - accuracy: 0.6
902 - val_loss: 0.7786 - val_accuracy: 0.6962
Epoch 35/100
14/14 [==============================] - 1s 101ms/step - loss: 0.7939 - accuracy: 0.
6896 - val_loss: 0.7795 - val_accuracy: 0.6914
Epoch 36/100
14/14 [==============================] - 0s 36ms/step - loss: 0.7933 - accuracy: 0.6
884 - val_loss: 0.7774 - val_accuracy: 0.6950
Epoch 37/100
14/14 [==============================] - 1s 44ms/step - loss: 0.7897 - accuracy: 0.6
911 - val_loss: 0.7724 - val_accuracy: 0.7010
Epoch 38/100
```

```
14/14 [==============================] - 1s 78ms/step - loss: 0.7887 - accuracy: 0.6
932 - val_loss: 0.7738 - val_accuracy: 0.7033
Epoch 39/100
14/14 [==============================] - 0s 36ms/step - loss: 0.7870 - accuracy: 0.6
914 - val_loss: 0.7741 - val_accuracy: 0.7010
Epoch 40/100
14/14 [==============================] - 0s 21ms/step - loss: 0.7848 - accuracy: 0.6
947 - val_loss: 0.7749 - val_accuracy: 0.6998
Epoch 41/100
14/14 [==============================] - 0s 22ms/step - loss: 0.7816 - accuracy: 0.6
920 - val_loss: 0.7719 - val_accuracy: 0.6986
Epoch 42/100
14/14 [==============================] - 0s 34ms/step - loss: 0.7821 - accuracy: 0.6
893 - val_loss: 0.7700 - val_accuracy: 0.6974
Epoch 43/100
14/14 [==============================] - 0s 15ms/step - loss: 0.7794 - accuracy: 0.6
926 - val_loss: 0.7696 - val_accuracy: 0.7010
Epoch 44/100
14/14 [==============================] - 0s 15ms/step - loss: 0.7774 - accuracy: 0.6
935 - val_loss: 0.7709 - val_accuracy: 0.6974
Epoch 45/100
14/14 [==============================] - 0s 27ms/step - loss: 0.7772 - accuracy: 0.6
914 - val_loss: 0.7698 - val_accuracy: 0.7010
Epoch 46/100
14/14 [==============================] - 1s 50ms/step - loss: 0.7765 - accuracy: 0.6
932 - val_loss: 0.7667 - val_accuracy: 0.7057
Epoch 47/100
14/14 [==============================] - 0s 30ms/step - loss: 0.7737 - accuracy: 0.6
941 - val_loss: 0.7691 - val_accuracy: 0.7057
Epoch 48/100
14/14 [==============================] - 0s 28ms/step - loss: 0.7724 - accuracy: 0.6
926 - val_loss: 0.7651 - val_accuracy: 0.7045
Epoch 49/100
14/14 [==============================] - 0s 17ms/step - loss: 0.7715 - accuracy: 0.6
929 - val_loss: 0.7638 - val_accuracy: 0.6998
Epoch 50/100
14/14 [==============================] - 0s 14ms/step - loss: 0.7722 - accuracy: 0.6
956 - val_loss: 0.7641 - val_accuracy: 0.7045
Epoch 51/100
14/14 [==============================] - 1s 46ms/step - loss: 0.7688 - accuracy: 0.6
929 - val_loss: 0.7637 - val_accuracy: 0.7022
Epoch 52/100
14/14 [==============================] - 0s 26ms/step - loss: 0.7692 - accuracy: 0.6
944 - val_loss: 0.7627 - val_accuracy: 0.7033
Epoch 53/100
14/14 [==============================] - 0s 27ms/step - loss: 0.7681 - accuracy: 0.6
935 - val_loss: 0.7607 - val_accuracy: 0.7069
Epoch 54/100
14/14 [==============================] - 0s 24ms/step - loss: 0.7673 - accuracy: 0.6
947 - val_loss: 0.7619 - val_accuracy: 0.7057
Epoch 55/100
14/14 [==============================] - 0s 35ms/step - loss: 0.7656 - accuracy: 0.6
923 - val_loss: 0.7616 - val_accuracy: 0.7045
Epoch 56/100
14/14 [==============================] - 1s 49ms/step - loss: 0.7630 - accuracy: 0.6
956 - val_loss: 0.7623 - val_accuracy: 0.7033
```

```
Epoch 57/100
14/14 [==============================] - 0s 13ms/step - loss: 0.7647 - accuracy: 0.6
923 - val_loss: 0.7604 - val_accuracy: 0.7057
Epoch 58/100
14/14 [==============================] - 0s 29ms/step - loss: 0.7622 - accuracy: 0.7
001 - val_loss: 0.7621 - val_accuracy: 0.7033
Epoch 59/100
14/14 [==============================] - 0s 36ms/step - loss: 0.7622 - accuracy: 0.6
995 - val_loss: 0.7589 - val_accuracy: 0.7117
Epoch 60/100
14/14 [==============================] - 0s 31ms/step - loss: 0.7610 - accuracy: 0.6
947 - val_loss: 0.7585 - val_accuracy: 0.7081
Epoch 61/100
14/14 [==============================] - 0s 22ms/step - loss: 0.7606 - accuracy: 0.6
956 - val_loss: 0.7573 - val_accuracy: 0.7045
Epoch 62/100
14/14 [==============================] - 0s 29ms/step - loss: 0.7591 - accuracy: 0.6
935 - val_loss: 0.7577 - val_accuracy: 0.7081
Epoch 63/100
14/14 [==============================] - 0s 35ms/step - loss: 0.7582 - accuracy: 0.6
989 - val_loss: 0.7579 - val_accuracy: 0.7069
Epoch 64/100
14/14 [==============================] - 0s 29ms/step - loss: 0.7577 - accuracy: 0.6
968 - val_loss: 0.7562 - val_accuracy: 0.7022
Epoch 65/100
14/14 [==============================] - 0s 17ms/step - loss: 0.7588 - accuracy: 0.6
974 - val_loss: 0.7588 - val_accuracy: 0.7093
Epoch 66/100
14/14 [==============================] - 0s 22ms/step - loss: 0.7573 - accuracy: 0.6
962 - val_loss: 0.7554 - val_accuracy: 0.7033
Epoch 67/100
14/14 [==============================] - 1s 58ms/step - loss: 0.7594 - accuracy: 0.6
944 - val_loss: 0.7578 - val_accuracy: 0.7069
Epoch 68/100
14/14 [==============================] - 0s 31ms/step - loss: 0.7555 - accuracy: 0.6
953 - val_loss: 0.7565 - val_accuracy: 0.7010
Epoch 69/100
14/14 [==============================] - 0s 14ms/step - loss: 0.7546 - accuracy: 0.6
941 - val_loss: 0.7550 - val_accuracy: 0.7093
Epoch 70/100
14/14 [==============================] - 0s 22ms/step - loss: 0.7533 - accuracy: 0.6
995 - val_loss: 0.7546 - val_accuracy: 0.6974
Epoch 71/100
14/14 [==============================] - 1s 38ms/step - loss: 0.7531 - accuracy: 0.6
971 - val_loss: 0.7577 - val_accuracy: 0.7033
Epoch 72/100
14/14 [==============================] - 0s 30ms/step - loss: 0.7522 - accuracy: 0.6
977 - val_loss: 0.7552 - val_accuracy: 0.7045
Epoch 73/100
14/14 [==============================] - 0s 33ms/step - loss: 0.7527 - accuracy: 0.6
962 - val_loss: 0.7538 - val_accuracy: 0.7093
Epoch 74/100
14/14 [==============================] - 1s 55ms/step - loss: 0.7524 - accuracy: 0.6
956 - val_loss: 0.7530 - val_accuracy: 0.7081
Epoch 75/100
14/14 [==============================] - 2s 132ms/step - loss: 0.7503 - accuracy: 0.
```

```
6974 - val_loss: 0.7539 - val_accuracy: 0.7069
Epoch 76/100
14/14 [==============================] - 0s 22ms/step - loss: 0.7501 - accuracy: 0.6
983 - val_loss: 0.7515 - val_accuracy: 0.7093
Epoch 77/100
14/14 [==============================] - 0s 26ms/step - loss: 0.7494 - accuracy: 0.6
977 - val_loss: 0.7529 - val_accuracy: 0.7033
Epoch 78/100
14/14 [==============================] - 0s 15ms/step - loss: 0.7489 - accuracy: 0.6
989 - val_loss: 0.7552 - val_accuracy: 0.7093
Epoch 79/100
14/14 [==============================] - 1s 37ms/step - loss: 0.7499 - accuracy: 0.6
980 - val_loss: 0.7558 - val_accuracy: 0.7045
Epoch 80/100
14/14 [==============================] - 1s 61ms/step - loss: 0.7493 - accuracy: 0.6
992 - val_loss: 0.7523 - val_accuracy: 0.7105
Epoch 81/100
14/14 [==============================] - 0s 22ms/step - loss: 0.7482 - accuracy: 0.7
028 - val_loss: 0.7517 - val_accuracy: 0.7153
Epoch 82/100
14/14 [==============================] - 0s 23ms/step - loss: 0.7493 - accuracy: 0.6
986 - val_loss: 0.7525 - val_accuracy: 0.7022
Epoch 83/100
14/14 [==============================] - 0s 31ms/step - loss: 0.7473 - accuracy: 0.6
947 - val_loss: 0.7508 - val_accuracy: 0.7081
Epoch 84/100
14/14 [==============================] - 2s 174ms/step - loss: 0.7463 - accuracy: 0.
6980 - val_loss: 0.7525 - val_accuracy: 0.7081
Epoch 85/100
14/14 [==============================] - 2s 149ms/step - loss: 0.7471 - accuracy: 0.
6983 - val_loss: 0.7512 - val_accuracy: 0.7093
Epoch 86/100
14/14 [==============================] - 1s 73ms/step - loss: 0.7444 - accuracy: 0.7
010 - val_loss: 0.7512 - val_accuracy: 0.7093
Epoch 87/100
14/14 [==============================] - 0s 37ms/step - loss: 0.7463 - accuracy: 0.6
977 - val_loss: 0.7514 - val_accuracy: 0.7093
Epoch 88/100
14/14 [==============================] - 0s 37ms/step - loss: 0.7445 - accuracy: 0.7
013 - val_loss: 0.7504 - val_accuracy: 0.7069
Epoch 89/100
14/14 [==============================] - 1s 41ms/step - loss: 0.7450 - accuracy: 0.6
986 - val_loss: 0.7507 - val_accuracy: 0.7081
Epoch 90/100
14/14 [==============================] - 1s 69ms/step - loss: 0.7432 - accuracy: 0.7
016 - val_loss: 0.7502 - val_accuracy: 0.7081
Epoch 91/100
14/14 [==============================] - 0s 20ms/step - loss: 0.7446 - accuracy: 0.6
980 - val_loss: 0.7535 - val_accuracy: 0.7045
Epoch 92/100
14/14 [==============================] - 0s 25ms/step - loss: 0.7450 - accuracy: 0.7
028 - val_loss: 0.7523 - val_accuracy: 0.7057
Epoch 93/100
14/14 [==============================] - 0s 15ms/step - loss: 0.7433 - accuracy: 0.6
983 - val_loss: 0.7504 - val_accuracy: 0.7105
Epoch 94/100
```

```
14/14 [==============================] - 0s 11ms/step - loss: 0.7427 - accuracy: 0.7
034 - val_loss: 0.7513 - val_accuracy: 0.7057
Epoch 95/100
14/14 [==============================] - 0s 27ms/step - loss: 0.7425 - accuracy: 0.7
007 - val_loss: 0.7505 - val_accuracy: 0.7057
Epoch 96/100
14/14 [==============================] - 0s 13ms/step - loss: 0.7418 - accuracy: 0.7
028 - val_loss: 0.7496 - val_accuracy: 0.7093
Epoch 97/100
14/14 [==============================] - 0s 20ms/step - loss: 0.7423 - accuracy: 0.6
998 - val_loss: 0.7493 - val_accuracy: 0.7093
Epoch 98/100
14/14 [==============================] - 0s 12ms/step - loss: 0.7424 - accuracy: 0.7
004 - val_loss: 0.7497 - val_accuracy: 0.7129
Epoch 99/100
14/14 [==============================] - 0s 12ms/step - loss: 0.7410 - accuracy: 0.7
028 - val_loss: 0.7501 - val_accuracy: 0.7117
Epoch 100/100
14/14 [==============================] - 0s 14ms/step - loss: 0.7534 - accuracy: 0.6
962 - val_loss: 0.7510 - val_accuracy: 0.7093
```

In [ ]:

```
model3 = Sequential()
model3.add(Dense(15, input_dim=10, activation="relu"))
model3.add(Dense(256, activation="relu"))
model3.add(Dense(128, activation="relu"))
model3.add(Dense(64, activation="relu"))
model3.add(Dense(num_groups, activation="softmax"))
model3.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accurac
history3 = model3.fit(x_train, y_train, validation_data = (x_test, y_test), epochs=
```

```
Epoch 1/50
14/14 [==============================] - 2s 27ms/step - loss: 1.7800 - accuracy: 0.5
397 - val_loss: 1.3006 - val_accuracy: 0.5825
Epoch 2/50
14/14 [==============================] - 0s 10ms/step - loss: 1.1451 - accuracy: 0.6
154 - val_loss: 0.9551 - val_accuracy: 0.6699
Epoch 3/50
14/14 [==============================] - 0s 11ms/step - loss: 0.9348 - accuracy: 0.6
603 - val_loss: 0.8525 - val_accuracy: 0.6782
Epoch 4/50
14/14 [==============================] - 0s 12ms/step - loss: 0.8817 - accuracy: 0.6
779 - val_loss: 0.8226 - val_accuracy: 0.6926
Epoch 5/50
14/14 [==============================] - 0s 12ms/step - loss: 0.8431 - accuracy: 0.6
872 - val_loss: 0.8009 - val_accuracy: 0.6986
Epoch 6/50
14/14 [==============================] - 0s 10ms/step - loss: 0.8156 - accuracy: 0.6
926 - val_loss: 0.7865 - val_accuracy: 0.6914
Epoch 7/50
14/14 [==============================] - 0s 12ms/step - loss: 0.7994 - accuracy: 0.6
890 - val_loss: 0.7855 - val_accuracy: 0.7010
Epoch 8/50
14/14 [==============================] - 0s 10ms/step - loss: 0.7986 - accuracy: 0.6
884 - val_loss: 0.7760 - val_accuracy: 0.6962
Epoch 9/50
14/14 [==============================] - 0s 10ms/step - loss: 0.7740 - accuracy: 0.6
935 - val_loss: 0.7747 - val_accuracy: 0.7081
Epoch 10/50
14/14 [==============================] - 0s 12ms/step - loss: 0.7673 - accuracy: 0.6
968 - val_loss: 0.7739 - val_accuracy: 0.7045
Epoch 11/50
14/14 [==============================] - 0s 11ms/step - loss: 0.7662 - accuracy: 0.6
965 - val_loss: 0.7684 - val_accuracy: 0.7117
Epoch 12/50
14/14 [==============================] - 0s 12ms/step - loss: 0.7566 - accuracy: 0.6
986 - val_loss: 0.7604 - val_accuracy: 0.7153
Epoch 13/50
14/14 [==============================] - 0s 11ms/step - loss: 0.7575 - accuracy: 0.6
965 - val_loss: 0.7588 - val_accuracy: 0.7010
Epoch 14/50
14/14 [==============================] - 0s 10ms/step - loss: 0.7522 - accuracy: 0.6
923 - val_loss: 0.7548 - val_accuracy: 0.7129
Epoch 15/50
14/14 [==============================] - 0s 11ms/step - loss: 0.7585 - accuracy: 0.6
962 - val_loss: 0.7605 - val_accuracy: 0.7081
Epoch 16/50
14/14 [==============================] - 0s 15ms/step - loss: 0.7487 - accuracy: 0.6
962 - val_loss: 0.7612 - val_accuracy: 0.7022
Epoch 17/50
14/14 [==============================] - 0s 13ms/step - loss: 0.7519 - accuracy: 0.6
974 - val_loss: 0.7506 - val_accuracy: 0.7010
Epoch 18/50
14/14 [==============================] - 0s 9ms/step - loss: 0.7371 - accuracy: 0.70
31 - val_loss: 0.7563 - val_accuracy: 0.7141
Epoch 19/50
14/14 [==============================] - 0s 14ms/step - loss: 0.7405 - accuracy: 0.6
```

```
980 - val_loss: 0.7567 - val_accuracy: 0.6974
Epoch 20/50
14/14 [==============================] - 0s 12ms/step - loss: 0.7370 - accuracy: 0.7
019 - val_loss: 0.7629 - val_accuracy: 0.6998
Epoch 21/50
14/14 [==============================] - 0s 9ms/step - loss: 0.7371 - accuracy: 0.70
25 - val_loss: 0.7523 - val_accuracy: 0.7022
Epoch 22/50
14/14 [==============================] - 0s 9ms/step - loss: 0.7329 - accuracy: 0.69
65 - val_loss: 0.7576 - val_accuracy: 0.7045
Epoch 23/50
14/14 [==============================] - 0s 15ms/step - loss: 0.7365 - accuracy: 0.6
977 - val_loss: 0.7544 - val_accuracy: 0.7057
Epoch 24/50
14/14 [==============================] - 0s 9ms/step - loss: 0.7289 - accuracy: 0.70
07 - val_loss: 0.7507 - val_accuracy: 0.7057
Epoch 25/50
14/14 [==============================] - 0s 15ms/step - loss: 0.7249 - accuracy: 0.7
046 - val_loss: 0.7557 - val_accuracy: 0.7093
Epoch 26/50
14/14 [==============================] - 0s 9ms/step - loss: 0.7275 - accuracy: 0.70
31 - val_loss: 0.7507 - val_accuracy: 0.7105
Epoch 27/50
14/14 [==============================] - 0s 15ms/step - loss: 0.7331 - accuracy: 0.6
986 - val_loss: 0.7543 - val_accuracy: 0.7093
Epoch 28/50
14/14 [==============================] - 0s 9ms/step - loss: 0.7346 - accuracy: 0.69
50 - val_loss: 0.7504 - val_accuracy: 0.7022
Epoch 29/50
14/14 [==============================] - 0s 14ms/step - loss: 0.7253 - accuracy: 0.7
103 - val_loss: 0.7680 - val_accuracy: 0.6938
Epoch 30/50
14/14 [==============================] - 0s 13ms/step - loss: 0.7358 - accuracy: 0.6
953 - val_loss: 0.7634 - val_accuracy: 0.7069
Epoch 31/50
14/14 [==============================] - 0s 14ms/step - loss: 0.7222 - accuracy: 0.7
037 - val_loss: 0.7491 - val_accuracy: 0.7069
Epoch 32/50
14/14 [==============================] - 0s 14ms/step - loss: 0.7184 - accuracy: 0.7
082 - val_loss: 0.7500 - val_accuracy: 0.6998
Epoch 33/50
14/14 [==============================] - 0s 17ms/step - loss: 0.7301 - accuracy: 0.6
962 - val_loss: 0.7531 - val_accuracy: 0.6974
Epoch 34/50
14/14 [==============================] - 0s 13ms/step - loss: 0.7189 - accuracy: 0.7
046 - val_loss: 0.7558 - val_accuracy: 0.7057
Epoch 35/50
14/14 [==============================] - 0s 13ms/step - loss: 0.7208 - accuracy: 0.7
076 - val_loss: 0.7535 - val_accuracy: 0.6986
Epoch 36/50
14/14 [==============================] - 0s 10ms/step - loss: 0.7156 - accuracy: 0.7
094 - val_loss: 0.7477 - val_accuracy: 0.7117
Epoch 37/50
14/14 [==============================] - 0s 9ms/step - loss: 0.7160 - accuracy: 0.71
36 - val_loss: 0.7797 - val_accuracy: 0.6926
Epoch 38/50
```

```
14/14 [==============================] - 0s 15ms/step - loss: 0.7200 - accuracy: 0.7
049 - val_loss: 0.7549 - val_accuracy: 0.6998
Epoch 39/50
14/14 [==============================] - 0s 17ms/step - loss: 0.7129 - accuracy: 0.7
061 - val_loss: 0.7496 - val_accuracy: 0.7022
Epoch 40/50
14/14 [==============================] - 0s 18ms/step - loss: 0.7175 - accuracy: 0.7
076 - val_loss: 0.7528 - val_accuracy: 0.7069
Epoch 41/50
14/14 [==============================] - 0s 17ms/step - loss: 0.7215 - accuracy: 0.7
025 - val_loss: 0.7591 - val_accuracy: 0.7022
Epoch 42/50
14/14 [==============================] - 0s 17ms/step - loss: 0.7123 - accuracy: 0.7
049 - val_loss: 0.7517 - val_accuracy: 0.7105
Epoch 43/50
14/14 [==============================] - 0s 16ms/step - loss: 0.7110 - accuracy: 0.7
088 - val_loss: 0.7588 - val_accuracy: 0.7033
Epoch 44/50
14/14 [==============================] - 0s 17ms/step - loss: 0.7140 - accuracy: 0.7
097 - val_loss: 0.7572 - val_accuracy: 0.6998
Epoch 45/50
14/14 [==============================] - 0s 17ms/step - loss: 0.7325 - accuracy: 0.7
004 - val_loss: 0.7870 - val_accuracy: 0.6974
Epoch 46/50
14/14 [==============================] - 0s 16ms/step - loss: 0.7202 - accuracy: 0.7
121 - val_loss: 0.7558 - val_accuracy: 0.6986
Epoch 47/50
14/14 [==============================] - 0s 15ms/step - loss: 0.7181 - accuracy: 0.7
058 - val_loss: 0.7550 - val_accuracy: 0.6950
Epoch 48/50
14/14 [==============================] - 0s 14ms/step - loss: 0.7216 - accuracy: 0.7
061 - val_loss: 0.7679 - val_accuracy: 0.6962
Epoch 49/50
14/14 [==============================] - 0s 15ms/step - loss: 0.7259 - accuracy: 0.7
031 - val_loss: 0.7564 - val_accuracy: 0.6986
Epoch 50/50
14/14 [==============================] - 0s 12ms/step - loss: 0.7086 - accuracy: 0.7
097 - val_loss: 0.7504 - val_accuracy: 0.7057
```

Analyzing the results

In [ ]:

```
num_groups
```

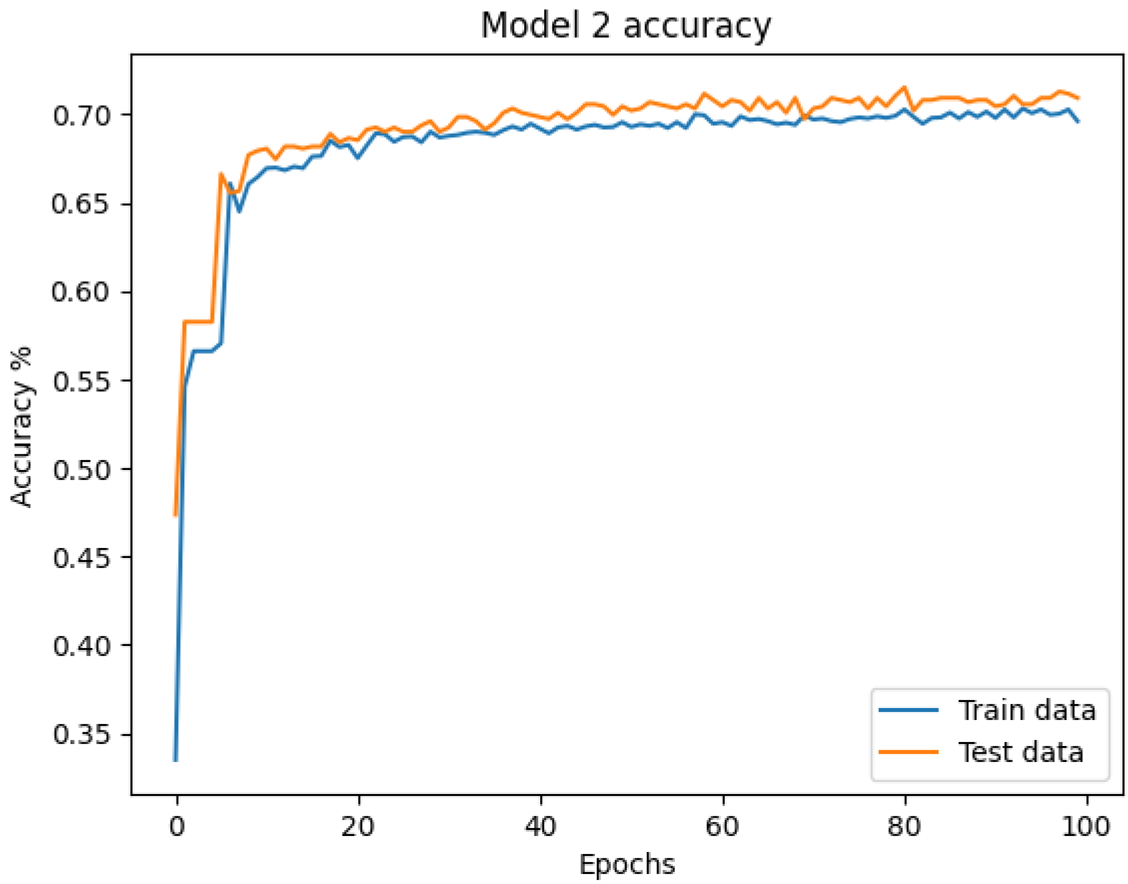
Out[ ]: 8

The accuracy has not improved. 70% accuracy is still good, because since there are 8 possible groups, picking at random would result in a 12.5% accuracy, so 70% accuracy means the model is giving meaningful predictions.

In [ ]:

```
plt.plot(history2.history["accuracy"])
plt.plot(history2.history["val_accuracy"])
plt.title("Model 2 accuracy")
plt.ylabel('Accuracy %')
```

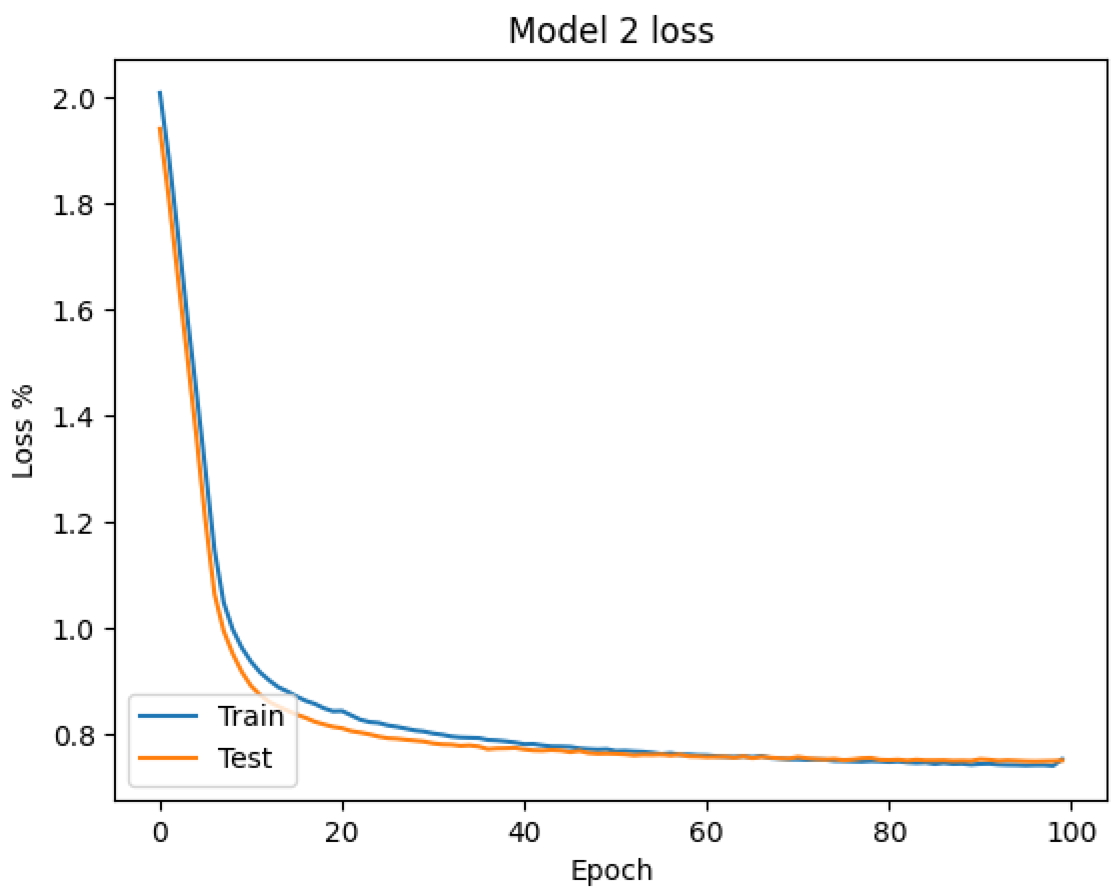
```python
plt.xlabel('Epochs')
plt.legend(['Train data', 'Test data'], loc='lower right')
plt.show()
```



The accuracy of this model goes seems to plateau at around 70%, which is significantly better than random guessing. The train and test accuracy moved together, so that means it did not overfit.

In [ ]:

```python
plt.plot(history2.history['loss'])
plt.plot(history2.history['val_loss'])
plt.title('Model 2 loss')
plt.ylabel('Loss %')
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='lower left')
plt.show()
```



The loss of the model is decreasing, which means that the error of the model is decreaasing, which is good. The train and test loss also are similar which measn the model did not overfit.

The below cell demonstrates the model predicting a high percentage of accurate Ring value groups.

In [ ]:

```
predictions_raw = model2.predict(x)
predictions = [np.argmax(probs_arr) for probs_arr in predictions_raw]
df["predicted group"] = predictions
df.head()
```

```
131/131 [==============================] - 1s 6ms/step
```

Out[ ]:

	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Sex_F	Sex_I	Sex_M	grc
0	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.150	0.0	0.0	1.0	
1	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.070	0.0	0.0	1.0	
2	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.210	1.0	0.0	0.0	
3	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.155	0.0	0.0	1.0	
4	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.055	0.0	1.0	0.0	