In [1]:
```python
datafile = r'C:\Users\HP\Downloads\dataas3\Grocery_Items_35.csv'
```

In [2]:
```python
import pandas as pd
from mlxtend.preprocessing import TransactionEncoder

data= pd.read_csv(datafile)

# Drop any columns with NaN values
g_df = [row.dropna().tolist() for index, row in data.iterrows()]

# Convert the DataFrame into a transaction format using TransactionEncoder
transen = TransactionEncoder()
transen_array = transen.fit(g_df).transform(g_df)

df = pd.DataFrame(transen_array, columns=transen.columns_)
df.head()
```

```
d:\Users\rakesh\anacondainstall\lib\site-packages\scipy\__init__.py:146: UserWarning: A
NumPy version >=1.16.5 and <1.23.0 is required for this version of SciPy (detected versi
on 1.26.4
  warnings.warn(f"A NumPy version >={np_minversion} and <{np_maxversion}"
```

Out[2]:

	Instant food products	UHT-milk	abrasive cleaner	artif. sweetener	baby cosmetics	bags	baking powder	bathroom cleaner	beef	berries	...	turkey	vineg
0	False	False	False	False	False	False	False	False	False	False	...	False	Fa
1	False	False	False	False	False	False	False	False	False	False	...	False	Fa
2	False	False	False	False	False	False	False	False	False	False	...	False	Fa
3	False	False	False	False	False	False	False	False	False	False	...	False	Fa
4	False	True	False	False	False	False	False	False	False	False	...	False	Fa

5 rows × 165 columns

In [3]:
```python
from mlxtend.frequent_patterns import apriori,association_rules

items = apriori(df, min_support=0.01, use_colnames=True)
association_rules(items, metric="confidence", min_threshold=0.1)
```

Out[3]:

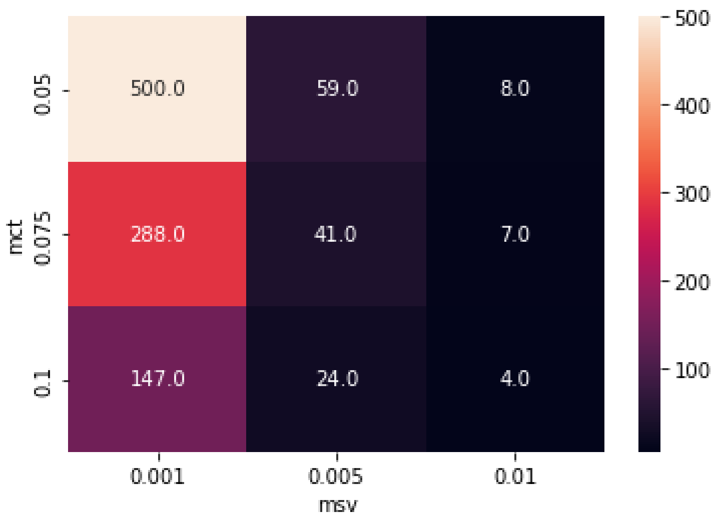
	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction	z
0	(other vegetables)	(whole milk)	0.120625	0.155875	0.01275	0.105699	0.678104	-0.006052	0.943894	
1	(rolls/buns)	(whole milk)	0.110875	0.155875	0.01300	0.117249	0.752200	-0.004283	0.956244	
2	(soda)	(whole milk)	0.100250	0.155875	0.01300	0.129676	0.831922	-0.002626	0.969897	
3	(yogurt)	(whole milk)	0.084875	0.155875	0.01025	0.120766	0.774761	-0.002980	0.960068	

In [4]:
```python
import seaborn as sns

msv =[ 0.001, 0.005, 0.01]
mct =[ 0.05, 0.075, 0.1]

heatmap_data = pd.DataFrame({
    'msv': [i for i in msv for _ in mct],
    'mct': mct * len(msv),
    'count': [len(association_rules(apriori(df, min_support=i, use_colnames=True), metri
              for i in msv for j in mct]
```

```
})
print(heatmap_data)
heatmap_data = heatmap_data.pivot("mct", "msv", "count")
sns.heatmap(heatmap_data, annot=True, fmt=".1f")
```

```
     msv    mct  count
0  0.001  0.050    500
1  0.001  0.075    288
2  0.001  0.100    147
3  0.005  0.050     59
4  0.005  0.075     41
5  0.005  0.100     24
6  0.010  0.050      8
7  0.010  0.075      7
8  0.010  0.100      4
```

Out[4]: <AxesSubplot:xlabel='msv', ylabel='mct'>



In [6]:
```
seta = df.iloc[:len(df)//2]
setb = df.iloc[len(df)//2:]

items = apriori(seta, min_support=0.005, use_colnames=True)
rules1 = association_rules(items, metric="confidence", min_threshold=0.075)

items = apriori(setb, min_support=0.005, use_colnames=True)
rules2 = association_rules(items, metric="confidence", min_threshold=0.075)

common_rules = pd.merge(rules1, rules2,on=['antecedents', 'consequents'])
```

In [7]:
```
rules1
```

Out[7]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
0	(bottled beer)	(other vegetables)	0.04450	0.11850	0.00525	0.117978	0.995591	-0.000023	0.999408
1	(bottled beer)	(whole milk)	0.04450	0.15575	0.00625	0.140449	0.901762	-0.000681	0.982199
2	(bottled water)	(other vegetables)	0.06375	0.11850	0.00525	0.082353	0.694962	-0.002304	0.960609
3	(bottled water)	(soda)	0.06375	0.10300	0.00550	0.086275	0.837617	-0.001066	0.981695
4	(bottled water)	(whole milk)	0.06375	0.15575	0.00625	0.098039	0.629465	-0.003679	0.936016
5	(brown bread)	(whole milk)	0.03600	0.15575	0.00575	0.159722	1.025504	0.000143	1.004727

6	(butter)	(whole milk)	0.03450	0.15575	0.00525	0.152174	0.977040	-0.000123	0.995782
7	(canned beer)	(whole milk)	0.04475	0.15575	0.00550	0.122905	0.789117	-0.001470	0.962553
8	(citrus fruit)	(other vegetables)	0.05375	0.11850	0.00625	0.116279	0.981258	-0.000119	0.997487
9	(citrus fruit)	(whole milk)	0.05375	0.15575	0.00725	0.134884	0.866027	-0.001122	0.975880
10	(citrus fruit)	(yogurt)	0.05375	0.07800	0.00550	0.102326	1.311866	0.001307	1.027098
11	(domestic eggs)	(whole milk)	0.03300	0.15575	0.00600	0.181818	1.167372	0.000860	1.031861
12	(frankfurter)	(other vegetables)	0.03900	0.11850	0.00550	0.141026	1.190090	0.000878	1.026224
13	(margarine)	(whole milk)	0.03750	0.15575	0.00525	0.140000	0.898876	-0.000591	0.981686
14	(pip fruit)	(other vegetables)	0.04925	0.11850	0.00675	0.137056	1.156589	0.000914	1.021503
15	(rolls/buns)	(other vegetables)	0.11025	0.11850	0.00975	0.088435	0.746290	-0.003315	0.967019
16	(other vegetables)	(rolls/buns)	0.11850	0.11025	0.00975	0.082278	0.746290	-0.003315	0.969521
17	(sausage)	(other vegetables)	0.05900	0.11850	0.00675	0.114407	0.965458	-0.000241	0.995378
18	(other vegetables)	(soda)	0.11850	0.10300	0.00975	0.082278	0.798820	-0.002455	0.977421
19	(soda)	(other vegetables)	0.10300	0.11850	0.00975	0.094660	0.798820	-0.002455	0.973668
20	(tropical fruit)	(other vegetables)	0.06725	0.11850	0.00600	0.089219	0.752906	-0.001969	0.967851
21	(other vegetables)	(whole milk)	0.11850	0.15575	0.01500	0.126582	0.812727	-0.003456	0.966605
22	(whole milk)	(other vegetables)	0.15575	0.11850	0.01500	0.096308	0.812727	-0.003456	0.975443
23	(yogurt)	(other vegetables)	0.07800	0.11850	0.00825	0.105769	0.892567	-0.000993	0.985763
24	(pastry)	(whole milk)	0.05200	0.15575	0.00700	0.134615	0.864304	-0.001099	0.975578
25	(pip fruit)	(soda)	0.04925	0.10300	0.00525	0.106599	1.034942	0.000177	1.004028
26	(pip fruit)	(whole milk)	0.04925	0.15575	0.00725	0.147208	0.945156	-0.000421	0.989984
27	(pork)	(whole milk)	0.03575	0.15575	0.00550	0.153846	0.987776	-0.000068	0.997750
28	(sausage)	(rolls/buns)	0.05900	0.11025	0.00675	0.114407	1.037703	0.000245	1.004694
29	(soda)	(rolls/buns)	0.10300	0.11025	0.00775	0.075243	0.682474	-0.003606	0.962144
30	(tropical fruit)	(rolls/buns)	0.06725	0.11025	0.00550	0.081784	0.741808	-0.001914	0.968999
31	(rolls/buns)	(whole milk)	0.11025	0.15575	0.01325	0.120181	0.771630	-0.003921	0.959573
32	(whole milk)	(rolls/buns)	0.15575	0.11025	0.01325	0.085072	0.771630	-0.003921	0.972481
33	(root vegetables)	(soda)	0.06975	0.10300	0.00600	0.086022	0.835160	-0.001184	0.981424
34	(root vegetables)	(whole milk)	0.06975	0.15575	0.00675	0.096774	0.621343	-0.004114	0.934705
35	(sausage)	(soda)	0.05900	0.10300	0.00650	0.110169	1.069607	0.000423	1.008057
36	(sausage)	(whole milk)	0.05900	0.15575	0.00875	0.148305	0.952200	-0.000439	0.991259
37	(shopping	(whole milk)	0.04450	0.15575	0.00625	0.140449	0.901762	-0.000681	0.982199

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
	bags)								
38	(whole milk)	(soda)	0.15575	0.10300	0.01250	0.080257	0.779192	-0.003542	0.975272
39	(soda)	(whole milk)	0.10300	0.15575	0.01250	0.121359	0.779192	-0.003542	0.960859
40	(tropical fruit)	(whole milk)	0.06725	0.15575	0.00750	0.111524	0.716046	-0.002974	0.950223
41	(yogurt)	(whole milk)	0.07800	0.15575	0.00825	0.105769	0.679096	-0.003898	0.944108

In [8]:
```
rules2
```

Out[8]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
0	(bottled beer)	(whole milk)	0.04125	0.15600	0.00725	0.175758	1.126651	0.000815	1.023971
1	(bottled water)	(other vegetables)	0.05675	0.12275	0.00725	0.127753	1.040760	0.000284	1.005736
2	(bottled water)	(whole milk)	0.05675	0.15600	0.00650	0.114537	0.734214	-0.002353	0.953174
3	(canned beer)	(whole milk)	0.04650	0.15600	0.00725	0.155914	0.999449	-0.000004	0.999898
4	(citrus fruit)	(rolls/buns)	0.05175	0.11150	0.00550	0.106280	0.953186	-0.000270	0.994159
5	(citrus fruit)	(whole milk)	0.05175	0.15600	0.00825	0.159420	1.021925	0.000177	1.004069
6	(citrus fruit)	(yogurt)	0.05175	0.09175	0.00500	0.096618	1.053061	0.000252	1.005389
7	(frankfurter)	(other vegetables)	0.03725	0.12275	0.00500	0.134228	1.093509	0.000428	1.013258
8	(frankfurter)	(whole milk)	0.03725	0.15600	0.00500	0.134228	0.860437	-0.000811	0.974853
9	(frozen vegetables)	(whole milk)	0.02675	0.15600	0.00525	0.196262	1.258088	0.001077	1.050093
10	(fruit/vegetable juice)	(whole milk)	0.03825	0.15600	0.00525	0.137255	0.879839	-0.000717	0.978273
11	(newspapers)	(whole milk)	0.03650	0.15600	0.00500	0.136986	0.878117	-0.000694	0.977968
12	(pip fruit)	(other vegetables)	0.04950	0.12275	0.00550	0.111111	0.905182	-0.000576	0.986906
13	(rolls/buns)	(other vegetables)	0.11150	0.12275	0.00975	0.087444	0.712374	-0.003937	0.961311
14	(other vegetables)	(rolls/buns)	0.12275	0.11150	0.00975	0.079430	0.712374	-0.003937	0.965163
15	(root vegetables)	(other vegetables)	0.07175	0.12275	0.00575	0.080139	0.652867	-0.003057	0.953677
16	(sausage)	(other vegetables)	0.06375	0.12275	0.00700	0.109804	0.894533	-0.000825	0.985457
17	(shopping bags)	(other vegetables)	0.04950	0.12275	0.00500	0.101010	0.822893	-0.001076	0.975817
18	(soda)	(other vegetables)	0.09750	0.12275	0.00825	0.084615	0.689331	-0.003718	0.958340
19	(tropical fruit)	(other vegetables)	0.06725	0.12275	0.00600	0.089219	0.726838	-0.002255	0.963185
20	(other vegetables)	(whole milk)	0.12275	0.15600	0.01050	0.085540	0.548332	-0.008649	0.922949
21	(other vegetables)	(yogurt)	0.12275	0.09175	0.01050	0.085540	0.932313	-0.000762	0.993209
22	(yogurt)	(other vegetables)	0.09175	0.12275	0.01050	0.114441	0.932313	-0.000762	0.990618
23	(pastry)	(whole milk)	0.04975	0.15600	0.00600	0.120603	0.773096	-0.001761	0.959749

24	(pip fruit)	(whole milk)	0.04950	0.15600	0.00750	0.151515	0.971251	-0.000222	0.994714
25	(pork)	(whole milk)	0.03725	0.15600	0.00525	0.140940	0.903459	-0.000561	0.982469
26	(root vegetables)	(rolls/buns)	0.07175	0.11150	0.00600	0.083624	0.749988	-0.002000	0.969580
27	(sausage)	(rolls/buns)	0.06375	0.11150	0.00500	0.078431	0.703420	-0.002108	0.964117
28	(rolls/buns)	(soda)	0.11150	0.09750	0.01000	0.089686	0.919857	-0.000871	0.991416
29	(soda)	(rolls/buns)	0.09750	0.11150	0.01000	0.102564	0.919857	-0.000871	0.990043
30	(rolls/buns)	(whole milk)	0.11150	0.15600	0.01275	0.114350	0.733011	-0.004644	0.952972
31	(whole milk)	(rolls/buns)	0.15600	0.11150	0.01275	0.081731	0.733011	-0.004644	0.967581
32	(rolls/buns)	(yogurt)	0.11150	0.09175	0.00900	0.080717	0.879755	-0.001230	0.987999
33	(yogurt)	(rolls/buns)	0.09175	0.11150	0.00900	0.098093	0.879755	-0.001230	0.985134
34	(root vegetables)	(whole milk)	0.07175	0.15600	0.00775	0.108014	0.692397	-0.003443	0.946203
35	(sausage)	(soda)	0.06375	0.09750	0.00600	0.094118	0.965309	-0.000216	0.996266
36	(sausage)	(whole milk)	0.06375	0.15600	0.00875	0.137255	0.879839	-0.001195	0.978273
37	(shopping bags)	(whole milk)	0.04950	0.15600	0.00700	0.141414	0.906501	-0.000722	0.983012
38	(tropical fruit)	(soda)	0.06725	0.09750	0.00525	0.078067	0.800686	-0.001307	0.978921
39	(whole milk)	(soda)	0.15600	0.09750	0.01350	0.086538	0.887574	-0.001710	0.988000
40	(soda)	(whole milk)	0.09750	0.15600	0.01350	0.138462	0.887574	-0.001710	0.979643
41	(yogurt)	(soda)	0.09175	0.09750	0.00700	0.076294	0.782505	-0.001946	0.977043
42	(tropical fruit)	(whole milk)	0.06725	0.15600	0.00850	0.126394	0.810218	-0.001991	0.966111
43	(yogurt)	(whole milk)	0.09175	0.15600	0.01225	0.133515	0.855865	-0.002063	0.974050
44	(whole milk)	(yogurt)	0.15600	0.09175	0.01225	0.078526	0.855865	-0.002063	0.985649

In [9]:
```
common_rules
```

Out[9]:

	antecedents	consequents	antecedent support_x	consequent support_x	support_x	confidence_x	lift_x	leverage_x	convic
0	(bottled beer)	(whole milk)	0.04450	0.15575	0.00625	0.140449	0.901762	-0.000681	0.9
1	(bottled water)	(other vegetables)	0.06375	0.11850	0.00525	0.082353	0.694962	-0.002304	0.9
2	(bottled water)	(whole milk)	0.06375	0.15575	0.00625	0.098039	0.629465	-0.003679	0.9
3	(canned beer)	(whole milk)	0.04475	0.15575	0.00550	0.122905	0.789117	-0.001470	0.9
4	(citrus fruit)	(whole milk)	0.05375	0.15575	0.00725	0.134884	0.866027	-0.001122	0.9
5	(citrus fruit)	(yogurt)	0.05375	0.07800	0.00550	0.102326	1.311866	0.001307	1.0
6	(frankfurter)	(other vegetables)	0.03900	0.11850	0.00550	0.141026	1.190090	0.000878	1.0
7	(pip fruit)	(other vegetables)	0.04925	0.11850	0.00675	0.137056	1.156589	0.000914	1.0
8	(rolls/buns)	(other vegetables)	0.11025	0.11850	0.00975	0.088435	0.746290	-0.003315	0.9
9	(other vegetables)	(rolls/buns)	0.11850	0.11025	0.00975	0.082278	0.746290	-0.003315	0.9

10	(sausage)	(other vegetables)	0.05900	0.11850	0.00675	0.114407	0.965458	-0.000241	0.9
11	(soda)	(other vegetables)	0.10300	0.11850	0.00975	0.094660	0.798820	-0.002455	0.9
12	(tropical fruit)	(other vegetables)	0.06725	0.11850	0.00600	0.089219	0.752906	-0.001969	0.9
13	(other vegetables)	(whole milk)	0.11850	0.15575	0.01500	0.126582	0.812727	-0.003456	0.9
14	(yogurt)	(other vegetables)	0.07800	0.11850	0.00825	0.105769	0.892567	-0.000993	0.9
15	(pastry)	(whole milk)	0.05200	0.15575	0.00700	0.134615	0.864304	-0.001099	0.9
16	(pip fruit)	(whole milk)	0.04925	0.15575	0.00725	0.147208	0.945156	-0.000421	0.9
17	(pork)	(whole milk)	0.03575	0.15575	0.00550	0.153846	0.987776	-0.000068	0.9
18	(sausage)	(rolls/buns)	0.05900	0.11025	0.00675	0.114407	1.037703	0.000245	1.0
19	(soda)	(rolls/buns)	0.10300	0.11025	0.00775	0.075243	0.682474	-0.003606	0.9
20	(rolls/buns)	(whole milk)	0.11025	0.15575	0.01325	0.120181	0.771630	-0.003921	0.9
21	(whole milk)	(rolls/buns)	0.15575	0.11025	0.01325	0.085072	0.771630	-0.003921	0.9
22	(root vegetables)	(whole milk)	0.06975	0.15575	0.00675	0.096774	0.621343	-0.004114	0.9
23	(sausage)	(soda)	0.05900	0.10300	0.00650	0.110169	1.069607	0.000423	1.0
24	(sausage)	(whole milk)	0.05900	0.15575	0.00875	0.148305	0.952200	-0.000439	0.9
25	(shopping bags)	(whole milk)	0.04450	0.15575	0.00625	0.140449	0.901762	-0.000681	0.9
26	(whole milk)	(soda)	0.15575	0.10300	0.01250	0.080257	0.779192	-0.003542	0.9
27	(soda)	(whole milk)	0.10300	0.15575	0.01250	0.121359	0.779192	-0.003542	0.9
28	(tropical fruit)	(whole milk)	0.06725	0.15575	0.00750	0.111524	0.716046	-0.002974	0.9
29	(yogurt)	(whole milk)	0.07800	0.15575	0.00825	0.105769	0.679096	-0.003898	0.9

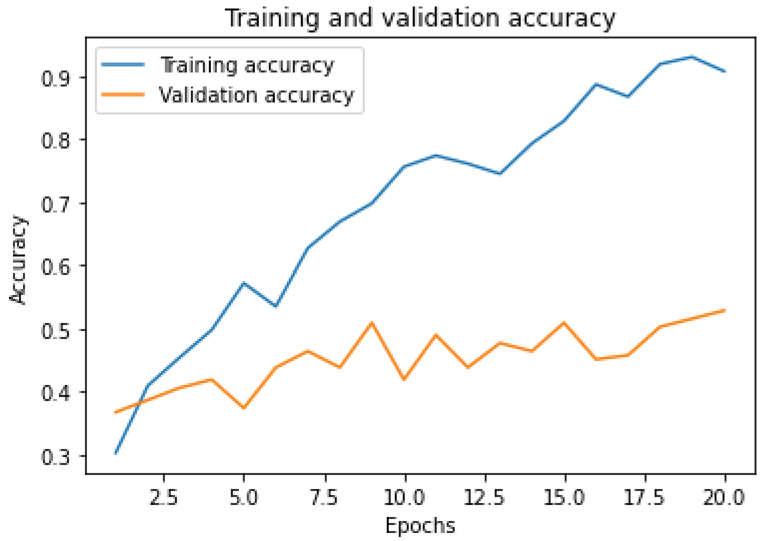
In [24]:
```
directory= r'C:\Users\HP\Downloads\dataas3\Cropped'
```

In [25]:
```
import os
import numpy as np
from tensorflow.keras.preprocessing.image import load_img, img_to_array
from sklearn.model_selection import train_test_split
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense
from tensorflow.keras.optimizers import Adam
import matplotlib.pyplot as plt
from tensorflow.keras.utils import to_categorical

# Define the image dimensions
img_height = 128
img_width = 128


dir1 = r'C:\Users\HP\Downloads\dataas3\Cropped\n02097209-standard_schnauzer'
dir2 = r'C:\Users\HP\Downloads\dataas3\Cropped\n02092002-Scottish_deerhound'
dir3 = r'C:\Users\HP\Downloads\dataas3\Cropped\n02092339-Weimaraner'
dir4 = r'C:\Users\HP\Downloads\dataas3\Cropped\n02108422-bull_mastiff'

def plot_learning_curves(history):
    train_acc = history.history['accuracy']
```

```
    val_acc = history.history['val_accuracy']
    epochs = range(1, len(train_acc) + 1)
    plt.plot(epochs, train_acc , label='Training accuracy')
    plt.plot(epochs, val_acc, label='Validation accuracy')
    plt.title('Training and validation accuracy')
    plt.xlabel('Epochs')
    plt.ylabel('Accuracy')
    plt.legend()
    plt.show()

def load_images_and_labels(directory):
    images = []
    labels = []
    for filename in os.listdir(directory):
        if filename.endswith(".jpg") or filename.endswith(".png"):
            img = load_img(os.path.join(directory, filename), target_size=(img_height, i
            img_array = img_to_array(img)
            images.append(img_array)
            if directory == dir1:
                labels.append(0)
            elif directory == dir2:
                labels.append(1)
            elif directory == dir3:
                labels.append(2)
            elif directory == dir4:
                labels.append(3)
    return images, labels

class1_img, class1_l = load_images_and_labels(dir1)
class2_img, class2_l = load_images_and_labels(dir2)
class3_img, class3_l = load_images_and_labels(dir3)
class4_img, class4_l = load_images_and_labels(dir4)

images = np.concatenate([class1_img, class2_img, class3_img, class4_img], axis=0)
labels = np.concatenate([class1_l, class2_l, class3_l, class4_l], axis=0)

labels = to_categorical(labels)

X_train, X_val, y_train, y_val = train_test_split(images, labels, test_size=0.2, random_

X_train = X_train / 255.0
X_val = X_val / 255.0

model = Sequential([
    Conv2D(8, (3, 3), activation='relu', input_shape=(img_height, img_width, 3)),
    MaxPooling2D((2, 2)),
    Flatten(),
    Dense(16, activation='relu'),
    Dense(4, activation='softmax')
])

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

history = model.fit(X_train, y_train, epochs=20, validation_data=(X_val, y_val))
plot_learning_curves(history)
```

```
Epoch 1/20
20/20 ━━━━━━━━━━━━━━━━━━━━ 2s 60ms/step - accuracy: 0.2812 - loss: 2.9034 - val_accuracy:
 0.3677 - val_loss: 1.3337
Epoch 2/20
20/20 ━━━━━━━━━━━━━━━━━━━━ 1s 43ms/step - accuracy: 0.4166 - loss: 1.3066 - val_accuracy:
 0.3871 - val_loss: 1.2865
Epoch 3/20
20/20 ━━━━━━━━━━━━━━━━━━━━ 1s 42ms/step - accuracy: 0.4636 - loss: 1.2179 - val_accuracy:
```

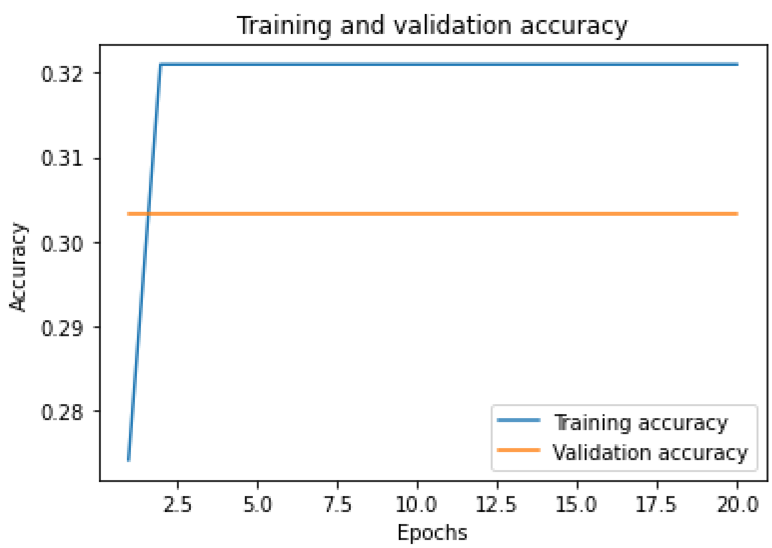
```
 0.4065 - val_loss: 1.2706
Epoch 4/20
20/20 ━━━━━━━━━━━━━━━━━━━━ 1s 37ms/step - accuracy: 0.5215 - loss: 1.1247 - val_accuracy:
 0.4194 - val_loss: 1.2414
Epoch 5/20
20/20 ━━━━━━━━━━━━━━━━━━━━ 1s 38ms/step - accuracy: 0.6130 - loss: 1.0657 - val_accuracy:
 0.3742 - val_loss: 1.3386
Epoch 6/20
20/20 ━━━━━━━━━━━━━━━━━━━━ 1s 32ms/step - accuracy: 0.4933 - loss: 1.1110 - val_accuracy:
 0.4387 - val_loss: 1.2373
Epoch 7/20
20/20 ━━━━━━━━━━━━━━━━━━━━ 1s 32ms/step - accuracy: 0.6298 - loss: 0.9760 - val_accuracy:
 0.4645 - val_loss: 1.2271
Epoch 8/20
20/20 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - accuracy: 0.6683 - loss: 0.9077 - val_accuracy:
 0.4387 - val_loss: 1.3056
Epoch 9/20
20/20 ━━━━━━━━━━━━━━━━━━━━ 1s 32ms/step - accuracy: 0.6994 - loss: 0.8483 - val_accuracy:
 0.5097 - val_loss: 1.1632
Epoch 10/20
20/20 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - accuracy: 0.7791 - loss: 0.7644 - val_accuracy:
 0.4194 - val_loss: 1.2319
Epoch 11/20
20/20 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - accuracy: 0.7610 - loss: 0.7047 - val_accuracy:
 0.4903 - val_loss: 1.2168
Epoch 12/20
20/20 ━━━━━━━━━━━━━━━━━━━━ 1s 32ms/step - accuracy: 0.7731 - loss: 0.6978 - val_accuracy:
 0.4387 - val_loss: 1.4899
Epoch 13/20
20/20 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - accuracy: 0.7444 - loss: 0.7211 - val_accuracy:
 0.4774 - val_loss: 1.3940
Epoch 14/20
20/20 ━━━━━━━━━━━━━━━━━━━━ 1s 34ms/step - accuracy: 0.7722 - loss: 0.6676 - val_accuracy:
 0.4645 - val_loss: 1.3516
Epoch 15/20
20/20 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - accuracy: 0.7699 - loss: 0.6209 - val_accuracy:
 0.5097 - val_loss: 1.2312
Epoch 16/20
20/20 ━━━━━━━━━━━━━━━━━━━━ 1s 34ms/step - accuracy: 0.9056 - loss: 0.4678 - val_accuracy:
 0.4516 - val_loss: 1.2530
Epoch 17/20
20/20 ━━━━━━━━━━━━━━━━━━━━ 1s 32ms/step - accuracy: 0.8564 - loss: 0.4732 - val_accuracy:
 0.4581 - val_loss: 1.2729
Epoch 18/20
20/20 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - accuracy: 0.9024 - loss: 0.4080 - val_accuracy:
 0.5032 - val_loss: 1.1818
Epoch 19/20
20/20 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - accuracy: 0.9361 - loss: 0.3432 - val_accuracy:
 0.5161 - val_loss: 1.2123
Epoch 20/20
20/20 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - accuracy: 0.9254 - loss: 0.3137 - val_accuracy:
 0.5290 - val_loss: 1.2298
```



In [28]:
```
model = Sequential([
    Conv2D(8, (3, 3), activation='relu', input_shape=(128, 128, 3)),
    MaxPooling2D((2, 2)),
    Flatten(),
    Dense(8, activation='relu'),
    Dense(4, activation='softmax')
])

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

history = model.fit(X_train, y_train, epochs=20, validation_data=(X_val, y_val))
plot_learning_curves(history)
```

```
Epoch 1/20
20/20 ━━━━━━━━━━━━━━━━━━━━ 2s 41ms/step - accuracy: 0.2431 - loss: 1.4050 - val_accuracy:
 0.3032 - val_loss: 1.3856
Epoch 2/20
20/20 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - accuracy: 0.3432 - loss: 1.3846 - val_accuracy:
 0.3032 - val_loss: 1.3849
Epoch 3/20
20/20 ━━━━━━━━━━━━━━━━━━━━ 1s 30ms/step - accuracy: 0.3211 - loss: 1.3837 - val_accuracy:
 0.3032 - val_loss: 1.3842
Epoch 4/20
20/20 ━━━━━━━━━━━━━━━━━━━━ 1s 29ms/step - accuracy: 0.3159 - loss: 1.3826 - val_accuracy:
 0.3032 - val_loss: 1.3836
Epoch 5/20
20/20 ━━━━━━━━━━━━━━━━━━━━ 1s 30ms/step - accuracy: 0.3135 - loss: 1.3813 - val_accuracy:
 0.3032 - val_loss: 1.3831
Epoch 6/20
20/20 ━━━━━━━━━━━━━━━━━━━━ 1s 30ms/step - accuracy: 0.3006 - loss: 1.3813 - val_accuracy:
 0.3032 - val_loss: 1.3827
Epoch 7/20
20/20 ━━━━━━━━━━━━━━━━━━━━ 1s 30ms/step - accuracy: 0.3032 - loss: 1.3818 - val_accuracy:
 0.3032 - val_loss: 1.3823
Epoch 8/20
20/20 ━━━━━━━━━━━━━━━━━━━━ 1s 31ms/step - accuracy: 0.3035 - loss: 1.3799 - val_accuracy:
 0.3032 - val_loss: 1.3821
Epoch 9/20
20/20 ━━━━━━━━━━━━━━━━━━━━ 1s 34ms/step - accuracy: 0.3014 - loss: 1.3803 - val_accuracy:
 0.3032 - val_loss: 1.3818
Epoch 10/20
20/20 ━━━━━━━━━━━━━━━━━━━━ 1s 32ms/step - accuracy: 0.3214 - loss: 1.3758 - val_accuracy:
 0.3032 - val_loss: 1.3815
Epoch 11/20
20/20 ━━━━━━━━━━━━━━━━━━━━ 1s 31ms/step - accuracy: 0.3360 - loss: 1.3737 - val_accuracy:
```

```
 0.3032 - val_loss: 1.3812
Epoch 12/20
20/20 ━━━━━━━━━━━━━━━━━━━━ 1s 28ms/step - accuracy: 0.2896 - loss: 1.3809 - val_accuracy: 
 0.3032 - val_loss: 1.3811
Epoch 13/20
20/20 ━━━━━━━━━━━━━━━━━━━━ 1s 29ms/step - accuracy: 0.3167 - loss: 1.3760 - val_accuracy: 
 0.3032 - val_loss: 1.3810
Epoch 14/20
20/20 ━━━━━━━━━━━━━━━━━━━━ 1s 29ms/step - accuracy: 0.3094 - loss: 1.3761 - val_accuracy: 
 0.3032 - val_loss: 1.3810
Epoch 15/20
20/20 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - accuracy: 0.3230 - loss: 1.3751 - val_accuracy: 
 0.3032 - val_loss: 1.3809
Epoch 16/20
20/20 ━━━━━━━━━━━━━━━━━━━━ 1s 32ms/step - accuracy: 0.3195 - loss: 1.3760 - val_accuracy: 
 0.3032 - val_loss: 1.3809
Epoch 17/20
20/20 ━━━━━━━━━━━━━━━━━━━━ 1s 29ms/step - accuracy: 0.3281 - loss: 1.3699 - val_accuracy: 
 0.3032 - val_loss: 1.3810
Epoch 18/20
20/20 ━━━━━━━━━━━━━━━━━━━━ 1s 31ms/step - accuracy: 0.3129 - loss: 1.3742 - val_accuracy: 
 0.3032 - val_loss: 1.3810
Epoch 19/20
20/20 ━━━━━━━━━━━━━━━━━━━━ 1s 29ms/step - accuracy: 0.3558 - loss: 1.3666 - val_accuracy: 
 0.3032 - val_loss: 1.3809
Epoch 20/20
20/20 ━━━━━━━━━━━━━━━━━━━━ 1s 31ms/step - accuracy: 0.3194 - loss: 1.3747 - val_accuracy: 
 0.3032 - val_loss: 1.3809
```

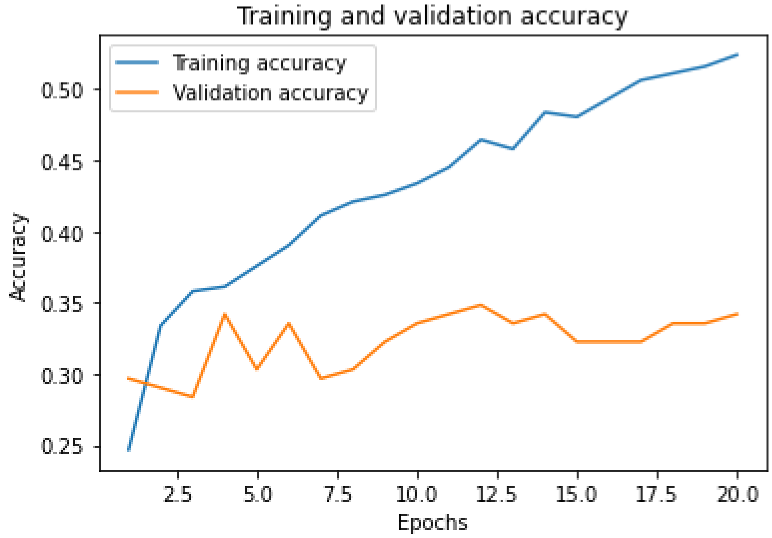


In [21]:

```
model = Sequential([
    Conv2D(8, (3, 3), activation='relu', input_shape=(128, 128, 3)),
    MaxPooling2D((2, 2)),
    Flatten(),
    Dense(32, activation='relu'),
    Dense(4, activation='softmax')
])

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

history = model.fit(X_train, y_train, epochs=20, validation_data=(X_val, y_val))
plot_learning_curves(history)
```

```
Epoch 1/20
20/20 ━━━━━━━━━━━━━━━━━━━━ 2s 56ms/step - accuracy: 0.2386 - loss: 2.2905 - val_accuracy: 
 0.2968 - val_loss: 1.3709
Epoch 2/20
```

```
20/20 ━━━━━━━━━━━━━━━━━━━━ 1s 44ms/step - accuracy: 0.3326 - loss: 1.3420 - val_accuracy:
 0.2903 - val_loss: 1.3591
Epoch 3/20
20/20 ━━━━━━━━━━━━━━━━━━━━ 1s 45ms/step - accuracy: 0.3702 - loss: 1.3318 - val_accuracy:
 0.2839 - val_loss: 1.3753
Epoch 4/20
20/20 ━━━━━━━━━━━━━━━━━━━━ 1s 46ms/step - accuracy: 0.3583 - loss: 1.3056 - val_accuracy:
 0.3419 - val_loss: 1.3351
Epoch 5/20
20/20 ━━━━━━━━━━━━━━━━━━━━ 1s 46ms/step - accuracy: 0.3938 - loss: 1.2394 - val_accuracy:
 0.3032 - val_loss: 1.3646
Epoch 6/20
20/20 ━━━━━━━━━━━━━━━━━━━━ 1s 46ms/step - accuracy: 0.3702 - loss: 1.2378 - val_accuracy:
 0.3355 - val_loss: 1.3088
Epoch 7/20
20/20 ━━━━━━━━━━━━━━━━━━━━ 1s 46ms/step - accuracy: 0.3881 - loss: 1.1937 - val_accuracy:
 0.2968 - val_loss: 1.3490
Epoch 8/20
20/20 ━━━━━━━━━━━━━━━━━━━━ 1s 49ms/step - accuracy: 0.3910 - loss: 1.1784 - val_accuracy:
 0.3032 - val_loss: 1.3447
Epoch 9/20
20/20 ━━━━━━━━━━━━━━━━━━━━ 1s 45ms/step - accuracy: 0.3936 - loss: 1.1333 - val_accuracy:
 0.3226 - val_loss: 1.3190
Epoch 10/20
20/20 ━━━━━━━━━━━━━━━━━━━━ 1s 49ms/step - accuracy: 0.4113 - loss: 1.1038 - val_accuracy:
 0.3355 - val_loss: 1.2889
Epoch 11/20
20/20 ━━━━━━━━━━━━━━━━━━━━ 1s 41ms/step - accuracy: 0.4642 - loss: 1.0629 - val_accuracy:
 0.3419 - val_loss: 1.2854
Epoch 12/20
20/20 ━━━━━━━━━━━━━━━━━━━━ 1s 44ms/step - accuracy: 0.4652 - loss: 1.0963 - val_accuracy:
 0.3484 - val_loss: 1.2860
Epoch 13/20
20/20 ━━━━━━━━━━━━━━━━━━━━ 1s 40ms/step - accuracy: 0.4539 - loss: 1.0446 - val_accuracy:
 0.3355 - val_loss: 1.2943
Epoch 14/20
20/20 ━━━━━━━━━━━━━━━━━━━━ 1s 40ms/step - accuracy: 0.4843 - loss: 1.0269 - val_accuracy:
 0.3419 - val_loss: 1.3001
Epoch 15/20
20/20 ━━━━━━━━━━━━━━━━━━━━ 1s 40ms/step - accuracy: 0.5107 - loss: 0.9794 - val_accuracy:
 0.3226 - val_loss: 1.3012
Epoch 16/20
20/20 ━━━━━━━━━━━━━━━━━━━━ 1s 40ms/step - accuracy: 0.4992 - loss: 1.0131 - val_accuracy:
 0.3226 - val_loss: 1.3016
Epoch 17/20
20/20 ━━━━━━━━━━━━━━━━━━━━ 1s 43ms/step - accuracy: 0.4848 - loss: 0.9886 - val_accuracy:
 0.3226 - val_loss: 1.3017
Epoch 18/20
20/20 ━━━━━━━━━━━━━━━━━━━━ 1s 41ms/step - accuracy: 0.5097 - loss: 0.9768 - val_accuracy:
 0.3355 - val_loss: 1.3050
Epoch 19/20
20/20 ━━━━━━━━━━━━━━━━━━━━ 1s 41ms/step - accuracy: 0.5265 - loss: 0.9386 - val_accuracy:
 0.3355 - val_loss: 1.3116
Epoch 20/20
20/20 ━━━━━━━━━━━━━━━━━━━━ 1s 39ms/step - accuracy: 0.5100 - loss: 0.9628 - val_accuracy:
 0.3419 - val_loss: 1.3592
```


Training and validation accuracy
Training accuracy
Validation accuracy
Accuracy
0.50
0.45
0.40
0.35
0.30
0.25
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
Epochs