In [10]:
```python
import os
import numpy as np
import pandas as pd
from mlxtend.preprocessing import TransactionEncoder
from mlxtend.frequent_patterns import apriori,association_rules
import seaborn as sns
from glob import glob
from sklearn.model_selection import train_test_split
from tensorflow.keras.preprocessing.image import load_img, img_to_array
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.layers import Dense,Conv2D,Flatten,MaxPooling2D
from tensorflow.keras.models import Sequential
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
import warnings
warnings.filterwarnings("ignore")

crop_folder=r'Cropped'
grocery_path=r'Grocery_Items_4.csv'
```

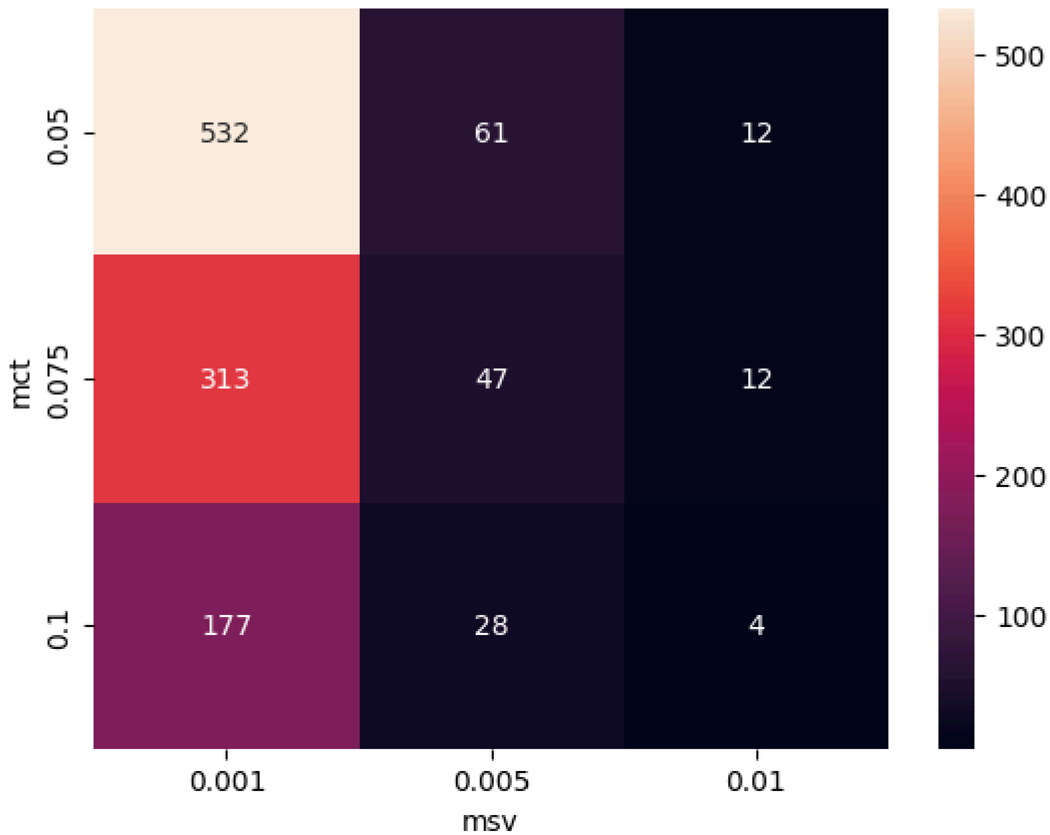
In [4]:
```python
groceries= [line.dropna().tolist() for idx, line in pd.read_csv(grocery_path).iterr

te = TransactionEncoder()
te_ary = te.fit(groceries).transform(groceries)
groceries_df = pd.DataFrame(te_ary, columns=te.columns_)
fi = apriori(groceries_df, min_support=0.01, use_colnames=True)
association_rules(fi, metric="confidence", min_threshold=0.1)
```

Out[4]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage
0	(other vegetables)	(whole milk)	0.117875	0.157	0.014875	0.126193	0.803777	-0.003631
1	(rolls/buns)	(whole milk)	0.115875	0.157	0.015000	0.129450	0.824521	-0.003192
2	(soda)	(whole milk)	0.101625	0.157	0.012625	0.124231	0.791282	-0.003330
3	(yogurt)	(whole milk)	0.084750	0.157	0.012000	0.141593	0.901866	-0.001306

In [5]:
```python
msv =[0.001,0.005,0.01]
mct =[0.05,0.075,0.1]

heatmap_df = pd.DataFrame(columns=['msv', 'mct', 'count'])
for i in msv:
    for j in mct:
        heatmap_df = heatmap_df.append({'msv': i, 'mct': j, 'count': len(associatio
sns.heatmap(heatmap_df.pivot("mct", "msv", "count"),annot=True,fmt=".0f")
```

Out[5]: `<AxesSubplot:xlabel='msv', ylabel='mct'>`



In [6]:
```
subset1 = groceries_df.iloc[:len(groceries_df)//2]
subset2 = groceries_df.iloc[len(groceries_df)//2:]
association_rules(apriori(subset1, min_support=0.005, use_colnames=True), metric="c
```

Out[6]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage
0	(bottled beer)	(whole milk)	0.04450	0.16300	0.00525	0.117978	0.723789	-0.002003
1	(bottled water)	(other vegetables)	0.05250	0.11625	0.00650	0.123810	1.065028	0.000397
2	(bottled water)	(whole milk)	0.05250	0.16300	0.00625	0.119048	0.730353	-0.002307
3	(butter)	(whole milk)	0.03500	0.16300	0.00625	0.178571	1.095530	0.000545
4	(canned beer)	(rolls/buns)	0.05150	0.11925	0.00650	0.126214	1.058395	0.000359
5	(canned beer)	(soda)	0.05150	0.10650	0.00550	0.106796	1.002780	0.000015
6	(canned beer)	(whole milk)	0.05150	0.16300	0.00675	0.131068	0.804098	-0.001644
7	(citrus fruit)	(rolls/buns)	0.05200	0.11925	0.00550	0.105769	0.886954	-0.000701
8	(citrus fruit)	(whole milk)	0.05200	0.16300	0.00675	0.129808	0.796366	-0.001726
9	(frankfurter)	(rolls/buns)	0.04000	0.11925	0.00500	0.125000	1.048218	0.000230
10	(frankfurter)	(whole milk)	0.04000	0.16300	0.00550	0.137500	0.843558	-0.001020
11	(frozen vegetables)	(other vegetables)	0.02950	0.11625	0.00500	0.169492	1.457992	0.001571
12	(frozen vegetables)	(whole milk)	0.02950	0.16300	0.00575	0.194915	1.195799	0.000941
13	(newspapers)	(whole milk)	0.03950	0.16300	0.00550	0.139241	0.854236	-0.000939
14	(rolls/buns)	(other vegetables)	0.11925	0.11625	0.01125	0.094340	0.811524	-0.002613
15	(other vegetables)	(rolls/buns)	0.11625	0.11925	0.01125	0.096774	0.811524	-0.002613
16	(shopping bags)	(other vegetables)	0.05350	0.11625	0.00525	0.098131	0.844136	-0.000969
17	(soda)	(other vegetables)	0.10650	0.11625	0.01050	0.098592	0.848099	-0.001881
18	(other vegetables)	(soda)	0.11625	0.10650	0.01050	0.090323	0.848099	-0.001881
19	(other vegetables)	(whole milk)	0.11625	0.16300	0.01575	0.135484	0.831189	-0.003199
20	(whole milk)	(other vegetables)	0.16300	0.11625	0.01575	0.096626	0.831189	-0.003199
21	(yogurt)	(other vegetables)	0.08450	0.11625	0.00925	0.109467	0.941656	-0.000573
22	(other	(yogurt)	0.11625	0.08450	0.00925	0.079570	0.941656	-0.000573

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage
	vegetables)							
23	(pastry)	(whole milk)	0.05150	0.16300	0.00675	0.131068	0.804098	-0.001644
24	(pip fruit)	(rolls/buns)	0.05175	0.11925	0.00650	0.125604	1.053282	0.000329
25	(pip fruit)	(soda)	0.05175	0.10650	0.00600	0.115942	1.088658	0.000489
26	(pip fruit)	(whole milk)	0.05175	0.16300	0.00700	0.135266	0.829851	-0.001435
27	(root vegetables)	(rolls/buns)	0.07150	0.11925	0.00600	0.083916	0.703699	-0.002526
28	(sausage)	(rolls/buns)	0.06225	0.11925	0.00650	0.104418	0.875620	-0.000923
29	(shopping bags)	(rolls/buns)	0.05350	0.11925	0.00600	0.112150	0.940457	-0.000380
30	(rolls/buns)	(soda)	0.11925	0.10650	0.01150	0.096436	0.905503	-0.001200
31	(soda)	(rolls/buns)	0.10650	0.11925	0.01150	0.107981	0.905503	-0.001200
32	(tropical fruit)	(rolls/buns)	0.06900	0.11925	0.00650	0.094203	0.789961	-0.001728
33	(rolls/buns)	(whole milk)	0.11925	0.16300	0.01525	0.127883	0.784556	-0.004188
34	(whole milk)	(rolls/buns)	0.16300	0.11925	0.01525	0.093558	0.784556	-0.004188
35	(yogurt)	(rolls/buns)	0.08450	0.11925	0.00875	0.103550	0.868346	-0.001327
36	(root vegetables)	(whole milk)	0.07150	0.16300	0.00775	0.108392	0.664979	-0.003904
37	(sausage)	(soda)	0.06225	0.10650	0.00650	0.104418	0.980448	-0.000130
38	(sausage)	(whole milk)	0.06225	0.16300	0.01000	0.160643	0.985537	-0.000147
39	(sausage)	(yogurt)	0.06225	0.08450	0.00600	0.096386	1.140657	0.000740
40	(shopping bags)	(soda)	0.05350	0.10650	0.00500	0.093458	0.877539	-0.000698
41	(shopping bags)	(whole milk)	0.05350	0.16300	0.00850	0.158879	0.974715	-0.000221
42	(soda)	(whole milk)	0.10650	0.16300	0.01575	0.147887	0.907284	-0.001609
43	(whole milk)	(soda)	0.16300	0.10650	0.01575	0.096626	0.907284	-0.001609
44	(tropical fruit)	(whole milk)	0.06900	0.16300	0.00700	0.101449	0.622388	-0.004247
45	(tropical fruit)	(yogurt)	0.06900	0.08450	0.00525	0.076087	0.900437	-0.000581
46	(yogurt)	(whole milk)	0.08450	0.16300	0.01225	0.144970	0.889389	-0.001524
47	(whole milk)	(yogurt)	0.16300	0.08450	0.01225	0.075153	0.889389	-0.001524

In [7]:

```
association_rules(apriori(subset2, min_support=0.005, use_colnames=True), metric="c
```

Out[7]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverag
0	(bottled beer)	(other vegetables)	0.03975	0.11950	0.00500	0.125786	1.052604	0.00025
1	(bottled beer)	(whole milk)	0.03975	0.15100	0.00650	0.163522	1.082927	0.00049
2	(bottled water)	(other vegetables)	0.05775	0.11950	0.00575	0.099567	0.833197	-0.00115
3	(bottled water)	(rolls/buns)	0.05775	0.11250	0.00550	0.095238	0.846561	-0.00099
4	(bottled water)	(whole milk)	0.05775	0.15100	0.00725	0.125541	0.831398	-0.00147
5	(brown bread)	(soda)	0.04325	0.09675	0.00500	0.115607	1.194904	0.00081
6	(brown bread)	(whole milk)	0.04325	0.15100	0.00550	0.127168	0.842170	-0.00103
7	(canned beer)	(whole milk)	0.04975	0.15100	0.00550	0.110553	0.732138	-0.00201
8	(citrus fruit)	(other vegetables)	0.05350	0.11950	0.00600	0.112150	0.938490	-0.00039
9	(citrus fruit)	(whole milk)	0.05350	0.15100	0.00550	0.102804	0.680819	-0.00257
10	(domestic eggs)	(whole milk)	0.04000	0.15100	0.00575	0.143750	0.951987	-0.00029
11	(frankfurter)	(other vegetables)	0.03325	0.11950	0.00525	0.157895	1.321295	0.00127
12	(frankfurter)	(whole milk)	0.03325	0.15100	0.00525	0.157895	1.045661	0.00022
13	(fruit/vegetable juice)	(rolls/buns)	0.03975	0.11250	0.00500	0.125786	1.118099	0.00052
14	(fruit/vegetable juice)	(whole milk)	0.03975	0.15100	0.00650	0.163522	1.082927	0.00049
15	(pip fruit)	(other vegetables)	0.04550	0.11950	0.00500	0.109890	0.919583	-0.00043
16	(rolls/buns)	(other vegetables)	0.11250	0.11950	0.00950	0.084444	0.706648	-0.00394
17	(other vegetables)	(rolls/buns)	0.11950	0.11250	0.00950	0.079498	0.706648	-0.00394
18	(root vegetables)	(other vegetables)	0.07200	0.11950	0.00550	0.076389	0.639238	-0.00310
19	(sausage)	(other vegetables)	0.06100	0.11950	0.00600	0.098361	0.823102	-0.00128
20	(shopping bags)	(other vegetables)	0.04700	0.11950	0.00675	0.143617	1.201816	0.00113
21	(soda)	(other vegetables)	0.09675	0.11950	0.00875	0.090439	0.756814	-0.00281
22	(tropical fruit)	(other vegetables)	0.06650	0.11950	0.00675	0.101504	0.849404	-0.00119

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverag
23	(whipped/sour cream)	(other vegetables)	0.04500	0.11950	0.00600	0.133333	1.115760	0.000623
24	(other vegetables)	(whole milk)	0.11950	0.15100	0.01400	0.117155	0.775860	-0.004044
25	(whole milk)	(other vegetables)	0.15100	0.11950	0.01400	0.092715	0.775860	-0.004044
26	(yogurt)	(other vegetables)	0.08500	0.11950	0.00875	0.102941	0.861432	-0.001407
27	(pastry)	(soda)	0.05550	0.09675	0.00550	0.099099	1.024280	0.000130
28	(pastry)	(whole milk)	0.05550	0.15100	0.00625	0.112613	0.745779	-0.002130
29	(pip fruit)	(whole milk)	0.04550	0.15100	0.00675	0.148352	0.982461	-0.000120
30	(sausage)	(rolls/buns)	0.06100	0.11250	0.00600	0.098361	0.874317	-0.000862
31	(shopping bags)	(rolls/buns)	0.04700	0.11250	0.00550	0.117021	1.040189	0.000212
32	(rolls/buns)	(soda)	0.11250	0.09675	0.00850	0.075556	0.780936	-0.002384
33	(soda)	(rolls/buns)	0.09675	0.11250	0.00850	0.087855	0.780936	-0.002384
34	(rolls/buns)	(whole milk)	0.11250	0.15100	0.01475	0.131111	0.868286	-0.002237
35	(whole milk)	(rolls/buns)	0.15100	0.11250	0.01475	0.097682	0.868286	-0.002237
36	(yogurt)	(rolls/buns)	0.08500	0.11250	0.00750	0.088235	0.784314	-0.002063
37	(root vegetables)	(soda)	0.07200	0.09675	0.00600	0.083333	0.861326	-0.000960
38	(root vegetables)	(whole milk)	0.07200	0.15100	0.00750	0.104167	0.689845	-0.003372
39	(sausage)	(soda)	0.06100	0.09675	0.00650	0.106557	1.101368	0.000598
40	(sausage)	(whole milk)	0.06100	0.15100	0.00950	0.155738	1.031376	0.000289
41	(sausage)	(yogurt)	0.06100	0.08500	0.00525	0.086066	1.012536	0.000065
42	(shopping bags)	(soda)	0.04700	0.09675	0.00575	0.122340	1.264501	0.001203
43	(shopping bags)	(whole milk)	0.04700	0.15100	0.00500	0.106383	0.704523	-0.002097
44	(tropical fruit)	(soda)	0.06650	0.09675	0.00550	0.082707	0.854850	-0.000934
45	(soda)	(whole milk)	0.09675	0.15100	0.00950	0.098191	0.650273	-0.005109
46	(yogurt)	(soda)	0.08500	0.09675	0.00675	0.079412	0.820793	-0.001474
47	(tropical fruit)	(whole milk)	0.06650	0.15100	0.00650	0.097744	0.647314	-0.003542
48	(tropical fruit)	(yogurt)	0.06650	0.08500	0.00550	0.082707	0.973021	-0.000153
49	(yogurt)	(whole milk)	0.08500	0.15100	0.01175	0.138235	0.915466	-0.001085

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverag
50	(whole milk)	(yogurt)	0.15100	0.08500	0.01175	0.077815	0.915466	0.00108

In [8]:
```
pd.merge(association_rules(apriori(subset1, min_support=0.005, use_colnames=True),
```

Out[8]:

	antecedents	consequents	antecedent support_x	consequent support_x	support_x	confidence_x	lift_x	leverag
0	(bottled beer)	(whole milk)	0.04450	0.16300	0.00525	0.117978	0.723789	-0.002
1	(bottled water)	(other vegetables)	0.05250	0.11625	0.00650	0.123810	1.065028	0.000
2	(bottled water)	(whole milk)	0.05250	0.16300	0.00625	0.119048	0.730353	-0.002
3	(canned beer)	(whole milk)	0.05150	0.16300	0.00675	0.131068	0.804098	-0.001
4	(citrus fruit)	(whole milk)	0.05200	0.16300	0.00675	0.129808	0.796366	-0.001
5	(frankfurter)	(whole milk)	0.04000	0.16300	0.00550	0.137500	0.843558	-0.001
6	(rolls/buns)	(other vegetables)	0.11925	0.11625	0.01125	0.094340	0.811524	-0.002
7	(other vegetables)	(rolls/buns)	0.11625	0.11925	0.01125	0.096774	0.811524	-0.002
8	(shopping bags)	(other vegetables)	0.05350	0.11625	0.00525	0.098131	0.844136	-0.000
9	(soda)	(other vegetables)	0.10650	0.11625	0.01050	0.098592	0.848099	-0.001
10	(other vegetables)	(whole milk)	0.11625	0.16300	0.01575	0.135484	0.831189	-0.003
11	(whole milk)	(other vegetables)	0.16300	0.11625	0.01575	0.096626	0.831189	-0.003
12	(yogurt)	(other vegetables)	0.08450	0.11625	0.00925	0.109467	0.941656	-0.000
13	(pastry)	(whole milk)	0.05150	0.16300	0.00675	0.131068	0.804098	-0.001
14	(pip fruit)	(whole milk)	0.05175	0.16300	0.00700	0.135266	0.829851	-0.001
15	(sausage)	(rolls/buns)	0.06225	0.11925	0.00650	0.104418	0.875620	-0.000
16	(shopping bags)	(rolls/buns)	0.05350	0.11925	0.00600	0.112150	0.940457	-0.000
17	(rolls/buns)	(soda)	0.11925	0.10650	0.01150	0.096436	0.905503	-0.001
18	(soda)	(rolls/buns)	0.10650	0.11925	0.01150	0.107981	0.905503	-0.001
19	(rolls/buns)	(whole milk)	0.11925	0.16300	0.01525	0.127883	0.784556	-0.004
20	(whole milk)	(rolls/buns)	0.16300	0.11925	0.01525	0.093558	0.784556	-0.004
21	(yogurt)	(rolls/buns)	0.08450	0.11925	0.00875	0.103550	0.868346	-0.001
22	(root vegetables)	(whole milk)	0.07150	0.16300	0.00775	0.108392	0.664979	-0.003
23	(sausage)	(soda)	0.06225	0.10650	0.00650	0.104418	0.980448	-0.000
24	(sausage)	(whole milk)	0.06225	0.16300	0.01000	0.160643	0.985537	-0.000

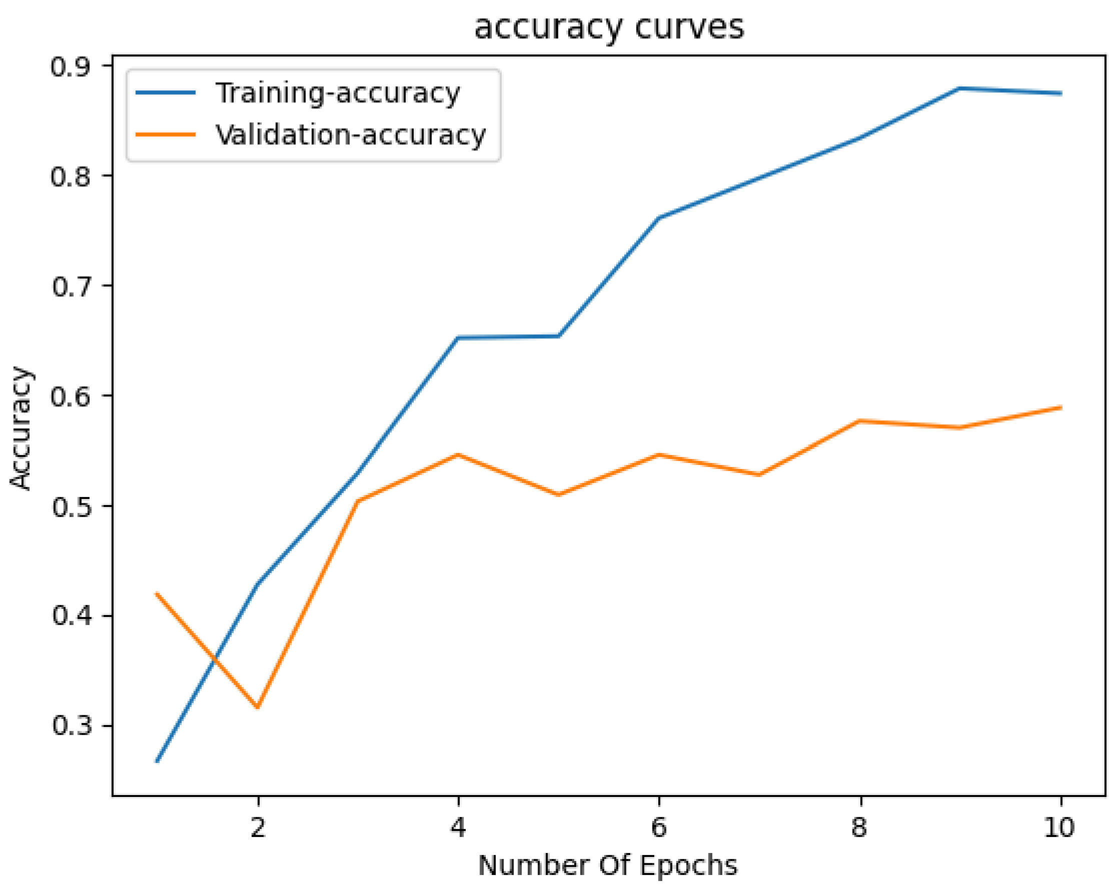
	antecedents	consequents	antecedent support_x	consequent support_x	support_x	confidence_x	lift_x	leverag
25	(sausage)	(yogurt)	0.06225	0.08450	0.00600	0.096386	1.140657	0.000
26	(shopping bags)	(soda)	0.05350	0.10650	0.00500	0.093458	0.877539	-0.000
27	(shopping bags)	(whole milk)	0.05350	0.16300	0.00850	0.158879	0.974715	-0.000
28	(soda)	(whole milk)	0.10650	0.16300	0.01575	0.147887	0.907284	-0.001
29	(tropical fruit)	(whole milk)	0.06900	0.16300	0.00700	0.101449	0.622388	-0.004
30	(tropical fruit)	(yogurt)	0.06900	0.08450	0.00525	0.076087	0.900437	-0.000
31	(yogurt)	(whole milk)	0.08450	0.16300	0.01225	0.144970	0.889389	-0.001
32	(whole milk)	(yogurt)	0.16300	0.08450	0.01225	0.075153	0.889389	-0.001

In [11]:

```python
dir1 = 'Cropped/n02088094-Afghan_hound/'
dir2 = 'Cropped/n02093428-American_Staffordshire_terrier/'
dir3 = 'Cropped/n02110627-affenpinscher/'
dir4 = 'Cropped/n02116738-African_hunting_dog/'


def plt_curve(history):
    tacc = history.history['accuracy']
    vacc = history.history['val_accuracy']
    epochs = range(1, len(tacc) + 1)
    plt.plot(epochs, tacc , label='Training-accuracy')
    plt.plot(epochs, vacc, label='Validation-accuracy')
    plt.title('accuracy curves')
    plt.xlabel('Number Of Epochs')
    plt.ylabel('Accuracy')
    plt.legend()
    plt.show()

def imagebreed(directory):
    imgs = []
    breed = []
    for name in os.listdir(directory):
        if name.endswith(".jpg") or name.endswith(".png"):
            img = load_img(os.path.join(directory, name), target_size=(128,128))
            img_array = img_to_array(img)
            imgs.append(img_array)
            if directory == dir1:
                breed.append(0)
            elif directory == dir2:
                breed.append(1)
            elif directory == dir3:
                breed.append(2)
            elif directory == dir4:
                breed.append(3)
    return imgs, breed
```

```
class1_images, class1_labels = imagebreed(dir1)
class2_images, class2_labels = imagebreed(dir2)
class3_images, class3_labels = imagebreed(dir3)
class4_images, class4_labels = imagebreed(dir4)

imgs = np.concatenate([class1_images, class2_images, class3_images, class4_images],
breed = np.concatenate([class1_labels, class2_labels, class3_labels, class4_labels]

breed = to_categorical(breed)

X_train, X_val, y_train, y_val = train_test_split(imgs, breed, test_size=0.2, rando

X_train = X_train / 255.0
X_val = X_val / 255.0

#requested model

model = Sequential([
    Conv2D(8, (3, 3), activation='relu', input_shape=(128, 128, 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=10, validation_data=(X_val, y_val))
plt_curve(history)
```

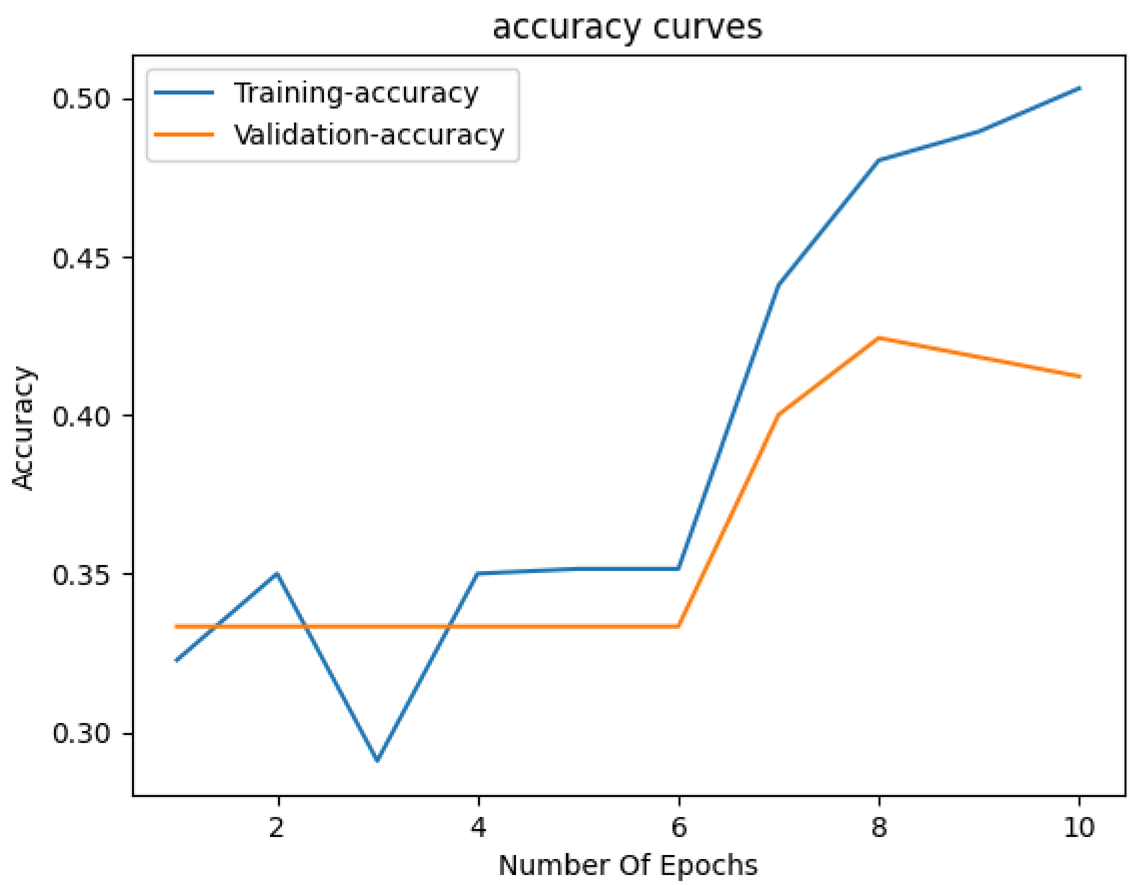
```
Train on 660 samples, validate on 165 samples
Epoch 1/10
660/660 [==============================] - 4s 6ms/sample - loss: 2.5374 - accurac
y: 0.2667 - val_loss: 1.3574 - val_accuracy: 0.4182
Epoch 2/10
660/660 [==============================] - 3s 4ms/sample - loss: 1.3214 - accurac
y: 0.4273 - val_loss: 1.3916 - val_accuracy: 0.3152
Epoch 3/10
660/660 [==============================] - 3s 4ms/sample - loss: 1.0789 - accurac
y: 0.5288 - val_loss: 1.2455 - val_accuracy: 0.5030
Epoch 4/10
660/660 [==============================] - 3s 4ms/sample - loss: 0.9158 - accurac
y: 0.6515 - val_loss: 1.1871 - val_accuracy: 0.5455
Epoch 5/10
660/660 [==============================] - 3s 4ms/sample - loss: 0.8499 - accurac
y: 0.6530 - val_loss: 1.2755 - val_accuracy: 0.5091
Epoch 6/10
660/660 [==============================] - 3s 4ms/sample - loss: 0.7034 - accurac
y: 0.7606 - val_loss: 1.1900 - val_accuracy: 0.5455
Epoch 7/10
660/660 [==============================] - 3s 4ms/sample - loss: 0.6197 - accurac
y: 0.7970 - val_loss: 1.0565 - val_accuracy: 0.5273
Epoch 8/10
660/660 [==============================] - 3s 4ms/sample - loss: 0.5243 - accurac
y: 0.8333 - val_loss: 1.0271 - val_accuracy: 0.5758
Epoch 9/10
660/660 [==============================] - 3s 4ms/sample - loss: 0.4669 - accurac
y: 0.8788 - val_loss: 1.0726 - val_accuracy: 0.5697
Epoch 10/10
660/660 [==============================] - 3s 4ms/sample - loss: 0.4217 - accurac
y: 0.8742 - val_loss: 1.0117 - val_accuracy: 0.5879
```



Banner ID : 916472053

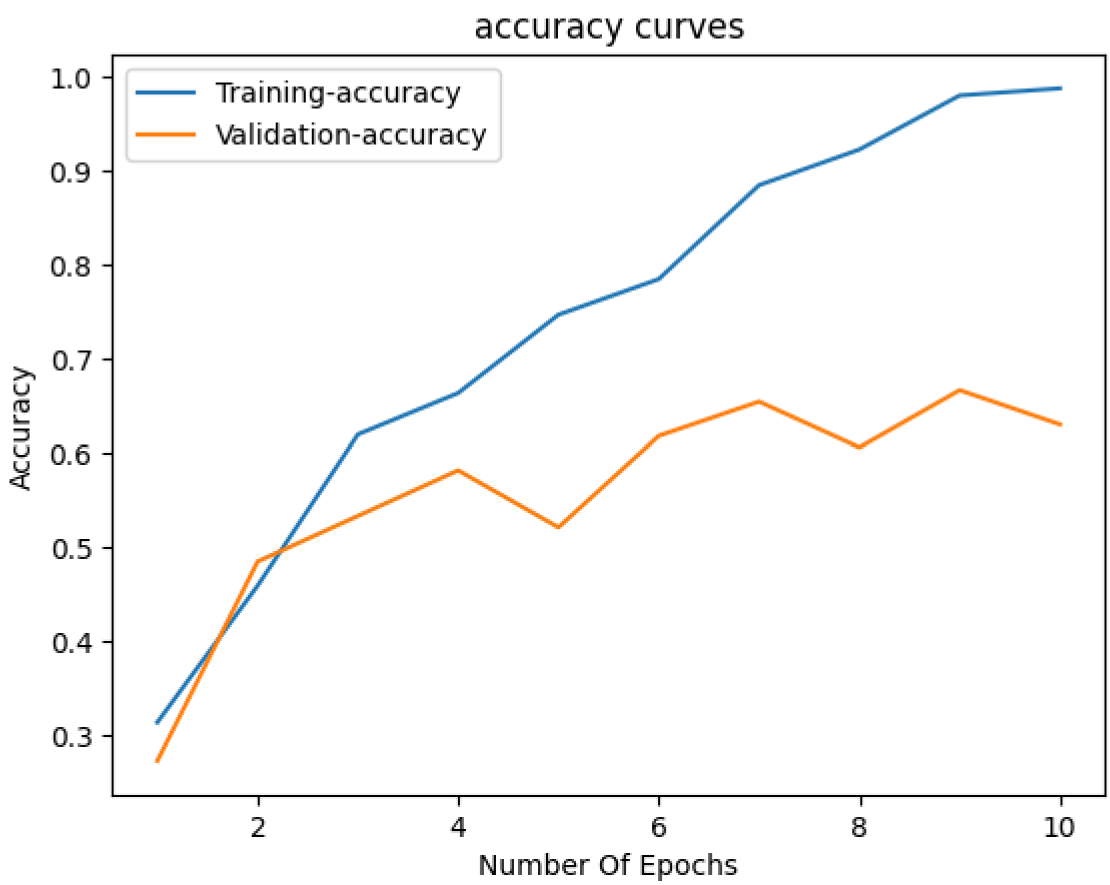
In [12]:
```
model = Sequential([
    Conv2D(8, (5, 5), activation='relu', input_shape=(128, 128, 3)),
    MaxPooling2D((2, 2)),
    Flatten(),
    Dense(16, activation='relu'),
    Dense(4, activation='softmax')
])

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


history = model.fit(X_train, y_train, epochs=10, validation_data=(X_val, y_val))
plt_curve(history)
```

```
Train on 660 samples, validate on 165 samples
Epoch 1/10
660/660 [==============================] - 5s 7ms/sample - loss: 1.3966 - accurac
y: 0.3227 - val_loss: 1.3846 - val_accuracy: 0.3333
Epoch 2/10
660/660 [==============================] - 4s 5ms/sample - loss: 1.3795 - accurac
y: 0.3500 - val_loss: 1.3711 - val_accuracy: 0.3333
Epoch 3/10
660/660 [==============================] - 4s 5ms/sample - loss: 1.3288 - accurac
y: 0.2909 - val_loss: 1.3093 - val_accuracy: 0.3333
Epoch 4/10
660/660 [==============================] - 4s 5ms/sample - loss: 1.2709 - accurac
y: 0.3500 - val_loss: 1.2784 - val_accuracy: 0.3333
Epoch 5/10
660/660 [==============================] - 4s 6ms/sample - loss: 1.2594 - accurac
y: 0.3515 - val_loss: 1.2712 - val_accuracy: 0.3333
Epoch 6/10
660/660 [==============================] - 4s 6ms/sample - loss: 1.2219 - accurac
y: 0.3515 - val_loss: 1.2524 - val_accuracy: 0.3333
Epoch 7/10
660/660 [==============================] - 4s 6ms/sample - loss: 1.1958 - accurac
y: 0.4409 - val_loss: 1.2423 - val_accuracy: 0.4000
Epoch 8/10
660/660 [==============================] - 4s 6ms/sample - loss: 1.1617 - accurac
y: 0.4803 - val_loss: 1.2134 - val_accuracy: 0.4242
Epoch 9/10
660/660 [==============================] - 4s 6ms/sample - loss: 1.1301 - accurac
y: 0.4894 - val_loss: 1.2038 - val_accuracy: 0.4182
Epoch 10/10
660/660 [==============================] - 4s 6ms/sample - loss: 1.1133 - accurac
y: 0.5030 - val_loss: 1.1952 - val_accuracy: 0.4121
```



In [13]:
```
model = Sequential([
    Conv2D(8, (7, 7), activation='relu', input_shape=(128, 128, 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=10, validation_data=(X_val, y_val))
plt_curve(history)
```

```
Train on 660 samples, validate on 165 samples
Epoch 1/10
660/660 [==============================] - 6s 10ms/sample - loss: 1.4071 - accurac
y: 0.3136 - val_loss: 1.3405 - val_accuracy: 0.2727
Epoch 2/10
660/660 [==============================] - 6s 9ms/sample - loss: 1.2297 - accurac
y: 0.4591 - val_loss: 1.1904 - val_accuracy: 0.4848
Epoch 3/10
660/660 [==============================] - 6s 9ms/sample - loss: 1.0289 - accurac
y: 0.6197 - val_loss: 1.2009 - val_accuracy: 0.5333
Epoch 4/10
660/660 [==============================] - 5s 8ms/sample - loss: 0.8410 - accurac
y: 0.6636 - val_loss: 1.0787 - val_accuracy: 0.5818
Epoch 5/10
660/660 [==============================] - 6s 9ms/sample - loss: 0.6607 - accurac
y: 0.7470 - val_loss: 1.2482 - val_accuracy: 0.5212
Epoch 6/10
660/660 [==============================] - 6s 8ms/sample - loss: 0.5413 - accurac
y: 0.7848 - val_loss: 1.0105 - val_accuracy: 0.6182
Epoch 7/10
660/660 [==============================] - 6s 9ms/sample - loss: 0.3439 - accurac
y: 0.8848 - val_loss: 1.0088 - val_accuracy: 0.6545
Epoch 8/10
660/660 [==============================] - 6s 9ms/sample - loss: 0.2637 - accurac
y: 0.9227 - val_loss: 1.0332 - val_accuracy: 0.6061
Epoch 9/10
660/660 [==============================] - 5s 8ms/sample - loss: 0.1594 - accurac
y: 0.9803 - val_loss: 0.9835 - val_accuracy: 0.6667
Epoch 10/10
660/660 [==============================] - 6s 8ms/sample - loss: 0.1016 - accurac
y: 0.9879 - val_loss: 1.0506 - val_accuracy: 0.6303
```



model 3 has high accuracy in trianing and low for validation hence we can say it is overfitting. model 1,2 are just right but model 2 accuracy is not great.

```
In [ ]:
```