1 Imports

[1]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

[ ]:
```
import warnings
warnings.filterwarnings("ignore", category=DeprecationWarning)
```

```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283:
DeprecationWarning: `should_run_async` will not call `transform_cell`
automatically in the future. Please pass the result to `transformed_cell`
argument and any exception that happen during thetransform in
`preprocessing_exc_tuple` in IPython 7.17 and above.
  and should_run_async(code)
```

[ ]:
```
import pandas as pd
from mlxtend.preprocessing import TransactionEncoder
from mlxtend.frequent_patterns import apriori, fpmax, fpgrowth
from mlxtend.frequent_patterns import association_rules
import seaborn as sns
import matplotlib.pyplot as plt
import csv
import random
```

2 Read Association data

[ ]:
```
data = []

with open("/content/drive/MyDrive/data_mining/Grocery_Items_44.csv", "r") as␣
 ↪file_:
    csv_reader = csv.reader(file_)

    next(csv_reader)

    for row in csv_reader:
```

```
        row = list(filter(lambda x: x != '', row))
        data.append(row)
```

3 1 (c).

[ ]:
```
def fit_association_rules(dataset,support,confidence):
  te = TransactionEncoder()
  te_ary = te.fit(dataset).transform(dataset)
  df = pd.DataFrame(te_ary, columns=te.columns_)
  frequent_itemsets = fpgrowth(df, min_support=support, use_colnames=True)
  rules = association_rules(frequent_itemsets, metric="confidence",␣
↪min_threshold=confidence)
  return rules
```

[ ]:
```
rules = fit_association_rules(data,0.01,0.1)
```

[ ]:
```
rules
```

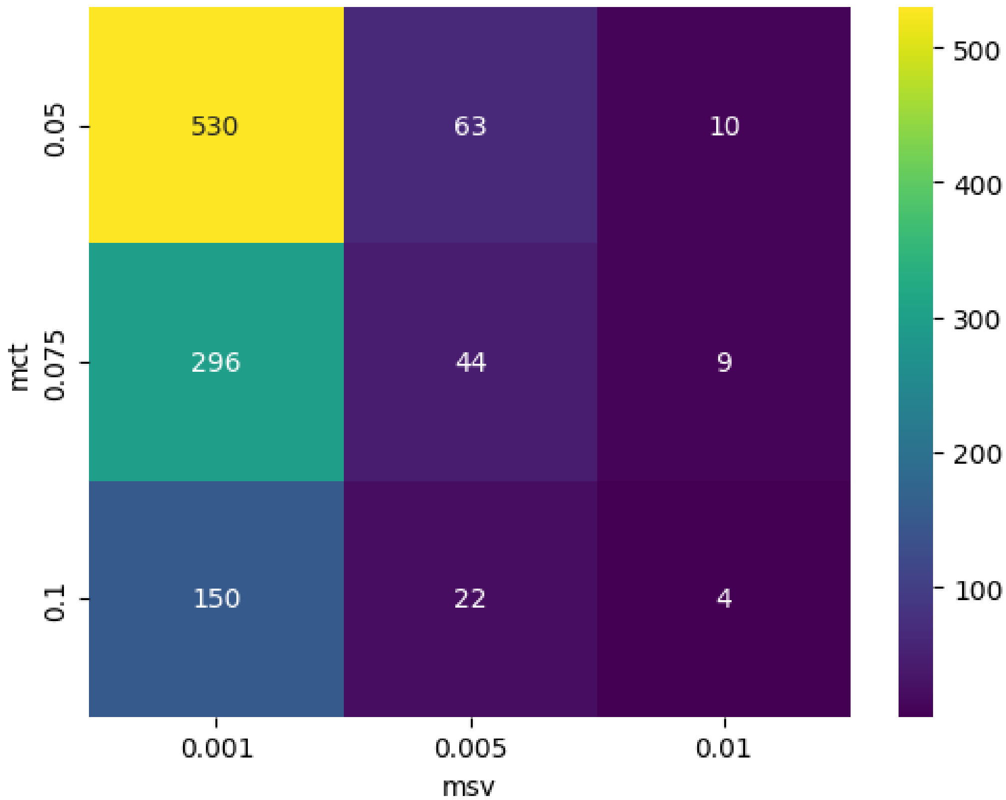
[ ]:
```
           antecedents   consequents  antecedent support  consequent support  \
0               (soda)  (whole milk)            0.094500            0.159375
1   (other vegetables)  (whole milk)            0.123375            0.159375
2         (rolls/buns)  (whole milk)            0.108125            0.159375
3             (yogurt)  (whole milk)            0.090375            0.159375

    support  confidence      lift  leverage  conviction  zhangs_metric
0  0.010000    0.105820  0.663969 -0.005061    0.940107      -0.358526
1  0.015500    0.125633  0.788287 -0.004163    0.961410      -0.234521
2  0.012875    0.119075  0.747138 -0.004357    0.954253      -0.275084
3  0.012125    0.134163  0.841808 -0.002279    0.970882      -0.171218
```

4 1(d)

[ ]:
```
min_supports_values = [0.001, 0.005, 0.01]
min_confidence_values = [0.05, 0.075, 0.1]
heat_map = []
for min_confidence in min_confidence_values:
  temp = []
  for min_support in min_supports_values:
    rules = fit_association_rules(data,min_support,min_confidence)
    temp.append(len(rules))
  heat_map.append(temp)
```

[ ]:
```
sns.heatmap(heat_map, annot=True,fmt='d',cmap='viridis')
plt.xticks(ticks=[0.5, 1.5, 2.5], labels=min_supports_values)
plt.yticks(ticks=[0.5, 1.5, 2.5], labels=min_confidence_values)
```

```
plt.xlabel("msv")
plt.ylabel("mct")
plt.show()
```



5 1 (e)

[ ]:
```
random.shuffle(data)
```

[ ]:
```
split_point = len(data) // 2

dataset_1 = data[:split_point]
dataset_2 = data[split_point:]
```

[ ]:
```
rules_dataset_1 = fit_association_rules(dataset_1,0.005,0.075)
```

[ ]:
```
rules_dataset_1
```

[ ]:
```
       antecedents        consequents  antecedent support  \
0  (tropical fruit)            (yogurt)             0.06925
1          (yogurt)  (tropical fruit)             0.08975
```

2	(tropical fruit)	(whole milk)	0.06925
3	(tropical fruit)	(rolls/buns)	0.06925
4	(tropical fruit)	(other vegetables)	0.06925
5	(tropical fruit)	(soda)	0.06925
6	(citrus fruit)	(whole milk)	0.05550
7	(citrus fruit)	(yogurt)	0.05550
8	(root vegetables)	(whole milk)	0.06875
9	(root vegetables)	(soda)	0.06875
10	(root vegetables)	(rolls/buns)	0.06875
11	(root vegetables)	(other vegetables)	0.06875
12	(yogurt)	(soda)	0.08975
13	(soda)	(yogurt)	0.09625
14	(whole milk)	(yogurt)	0.16125
15	(yogurt)	(whole milk)	0.08975
16	(yogurt)	(rolls/buns)	0.08975
17	(yogurt)	(other vegetables)	0.08975
18	(canned beer)	(yogurt)	0.04875
19	(canned beer)	(whole milk)	0.04875
20	(canned beer)	(other vegetables)	0.04875
21	(frankfurter)	(other vegetables)	0.03800
22	(beef)	(whole milk)	0.03550
23	(butter)	(whole milk)	0.03500
24	(whole milk)	(other vegetables)	0.16125
25	(other vegetables)	(whole milk)	0.12000
26	(sausage)	(yogurt)	0.06325
27	(sausage)	(soda)	0.06325
28	(sausage)	(other vegetables)	0.06325
29	(sausage)	(rolls/buns)	0.06325
30	(sausage)	(whole milk)	0.06325
31	(rolls/buns)	(whole milk)	0.11450
32	(whole milk)	(rolls/buns)	0.16125
33	(rolls/buns)	(other vegetables)	0.11450
34	(other vegetables)	(rolls/buns)	0.12000
35	(bottled water)	(rolls/buns)	0.06175
36	(bottled water)	(other vegetables)	0.06175
37	(bottled water)	(whole milk)	0.06175
38	(soda)	(rolls/buns)	0.09625
39	(soda)	(other vegetables)	0.09625
40	(other vegetables)	(soda)	0.12000
41	(soda)	(whole milk)	0.09625
42	(shopping bags)	(soda)	0.04675
43	(shopping bags)	(whole milk)	0.04675
44	(shopping bags)	(rolls/buns)	0.04675
45	(shopping bags)	(other vegetables)	0.04675
46	(pastry)	(whole milk)	0.04725
47	(pip fruit)	(whole milk)	0.05125
48	(whipped/sour cream)	(whole milk)	0.04150

```
49          (bottled beer)        (whole milk)            0.04425
50  (fruit/vegetable juice)       (rolls/buns)            0.03325

    consequent support  support  confidence      lift  leverage  conviction  \
0              0.08975  0.00700    0.101083  1.126273  0.000785    1.012607
1              0.06925  0.00700    0.077994  1.126273  0.000785    1.009484
2              0.16125  0.00950    0.137184  0.850754 -0.001667    0.972108
3              0.11450  0.00600    0.086643  0.756704 -0.001929    0.969500
4              0.12000  0.00625    0.090253  0.752106 -0.002060    0.967302
5              0.09625  0.00550    0.079422  0.825168 -0.001165    0.981721
6              0.16125  0.00800    0.144144  0.893917 -0.000949    0.980013
7              0.08975  0.00500    0.090090  1.003789  0.000019    1.000374
8              0.16125  0.00675    0.098182  0.608879 -0.004336    0.930066
9              0.09625  0.00600    0.087273  0.906730 -0.000617    0.990164
10             0.11450  0.00775    0.112727  0.984518 -0.000122    0.998002
11             0.12000  0.00550    0.080000  0.666667 -0.002750    0.956522
12             0.09625  0.00750    0.083565  0.868213 -0.001138    0.986159
13             0.08975  0.00750    0.077922  0.868213 -0.001138    0.987173
14             0.08975  0.01425    0.088372  0.984647 -0.000222    0.998489
15             0.16125  0.01425    0.158774  0.984647 -0.000222    0.997057
16             0.11450  0.00675    0.075209  0.656846 -0.003526    0.957514
17             0.12000  0.00725    0.080780  0.673166 -0.003520    0.957333
18             0.08975  0.00525    0.107692  1.199914  0.000875    1.020108
19             0.16125  0.00625    0.128205  0.795071 -0.001611    0.962096
20             0.12000  0.00525    0.107692  0.897436 -0.000600    0.986207
21             0.12000  0.00600    0.157895  1.315789  0.001440    1.045000
22             0.16125  0.00500    0.140845  0.873458 -0.000724    0.976250
23             0.16125  0.00550    0.157143  0.974529 -0.000144    0.995127
24             0.12000  0.01500    0.093023  0.775194 -0.004350    0.970256
25             0.16125  0.01500    0.125000  0.775194 -0.004350    0.958571
26             0.08975  0.00600    0.094862  1.056954  0.000323    1.005647
27             0.09625  0.00700    0.110672  1.149838  0.000912    1.016217
28             0.12000  0.00850    0.134387  1.119895  0.000910    1.016621
29             0.11450  0.00550    0.086957  0.759446 -0.001742    0.969833
30             0.16125  0.00900    0.142292  0.882434 -0.001199    0.977897
31             0.16125  0.01300    0.113537  0.704106 -0.005463    0.946176
32             0.11450  0.01300    0.080620  0.704106 -0.005463    0.963149
33             0.12000  0.01000    0.087336  0.727802 -0.003740    0.964211
34             0.11450  0.01000    0.083333  0.727802 -0.003740    0.966000
35             0.11450  0.00600    0.097166  0.848611 -0.001070    0.980800
36             0.12000  0.00600    0.097166  0.809717 -0.001410    0.974709
37             0.16125  0.00550    0.089069  0.552365 -0.004457    0.920761
38             0.11450  0.00800    0.083117  0.725912 -0.003021    0.965772
39             0.12000  0.01050    0.109091  0.909091 -0.001050    0.987755
40             0.09625  0.01050    0.087500  0.909091 -0.001050    0.990411
41             0.16125  0.01075    0.111688  0.692641 -0.004770    0.944207
42             0.09625  0.00525    0.112299  1.166748  0.000750    1.018080
```

43	0.16125	0.00875	0.187166	1.160718	0.001212	1.031883
44	0.11450	0.00550	0.117647	1.027485	0.000147	1.003567
45	0.12000	0.00500	0.106952	0.891266	-0.000610	0.985389
46	0.16125	0.00600	0.126984	0.787498	-0.001619	0.960750
47	0.16125	0.00750	0.146341	0.907544	-0.000764	0.982536
48	0.16125	0.00525	0.126506	0.784533	-0.001442	0.960224
49	0.16125	0.00775	0.175141	1.086147	0.000615	1.016841
50	0.11450	0.00500	0.150376	1.313327	0.001193	1.042226

	zhangs_metric
0	0.120458
1	0.123171
2	-0.158589
3	-0.256750
4	-0.261515
5	-0.185428
6	-0.111621
7	0.003997
8	-0.408208
9	-0.099471
10	-0.016606
11	-0.349345
12	-0.142924
13	-0.143805
14	-0.018250
15	-0.016841
16	-0.364650
17	-0.347850
18	0.175145
19	-0.213193
20	-0.107257
21	0.249480
22	-0.130592
23	-0.026370
24	-0.256921
25	-0.247863
26	0.057524
27	0.139111
28	0.114288
29	-0.252692
30	-0.124516
31	-0.321841
32	-0.333792
33	-0.296943
34	-0.298246
35	-0.159760
36	-0.200298

```
37      -0.463443
38      -0.294677
39      -0.099626
40      -0.102041
41      -0.329314
42       0.149926
43       0.145255
44       0.028062
45      -0.113462
46      -0.220714
47      -0.096966
48      -0.222718
49       0.082987
50       0.246780
```

[ ]:
```
rules_dataset_2 = fit_association_rules(dataset_2,0.005,0.075)
```

[ ]:
```
rules_dataset_2
```

[ ]:
```
             antecedents          consequents  antecedent support  \
0          (rolls/buns)         (whole milk)             0.10175
1          (whole milk)         (rolls/buns)             0.15750
2          (rolls/buns)  (other vegetables)             0.10175
3    (other vegetables)         (rolls/buns)             0.12675
4           (pip fruit)         (rolls/buns)             0.04750
5           (pip fruit)  (other vegetables)             0.04750
6           (pip fruit)         (whole milk)             0.04750
7          (whole milk)  (other vegetables)             0.15750
8    (other vegetables)         (whole milk)             0.12675
9              (yogurt)  (other vegetables)             0.09100
10   (other vegetables)             (yogurt)             0.12675
11         (rolls/buns)             (yogurt)             0.10175
12             (yogurt)         (rolls/buns)             0.09100
13             (yogurt)         (whole milk)             0.09100
14      (shopping bags)  (other vegetables)             0.04850
15      (shopping bags)   (root vegetables)             0.04850
16      (shopping bags)         (whole milk)             0.04850
17         (newspapers)         (whole milk)             0.03975
18        (canned beer)         (whole milk)             0.04775
19    (root vegetables)  (other vegetables)             0.06925
20    (root vegetables)         (whole milk)             0.06925
21     (tropical fruit)             (yogurt)             0.06550
22     (tropical fruit)         (whole milk)             0.06550
23     (tropical fruit)               (soda)             0.06550
24     (tropical fruit)         (rolls/buns)             0.06550
25      (domestic eggs)         (whole milk)             0.03825
26        (frankfurter)         (whole milk)             0.03800
```

27	(frankfurter)	(other vegetables)	0.03800
28	(sausage)	(whole milk)	0.06275
29	(sausage)	(soda)	0.06275
30	(sausage)	(yogurt)	0.06275
31	(sausage)	(other vegetables)	0.06275
32	(sausage)	(rolls/buns)	0.06275
33	(soda)	(whole milk)	0.09275
34	(rolls/buns)	(soda)	0.10175
35	(soda)	(rolls/buns)	0.09275
36	(soda)	(other vegetables)	0.09275
37	(pastry)	(soda)	0.05450
38	(pastry)	(whole milk)	0.05450
39	(bottled water)	(whole milk)	0.06350
40	(bottled water)	(soda)	0.06350
41	(bottled water)	(yogurt)	0.06350
42	(bottled water)	(other vegetables)	0.06350
43	(bottled beer)	(other vegetables)	0.04750
44	(bottled beer)	(whole milk)	0.04750
45	(citrus fruit)	(whole milk)	0.05200

	consequent support	support	confidence	lift	leverage	conviction \
0	0.15750	0.01275	0.125307	0.795601	-0.003276	0.963195
1	0.10175	0.01275	0.080952	0.795601	-0.003276	0.977370
2	0.12675	0.01100	0.108108	0.852924	-0.001897	0.979098
3	0.10175	0.01100	0.086785	0.852924	-0.001897	0.983613
4	0.10175	0.00575	0.121053	1.189706	0.000917	1.021961
5	0.12675	0.00600	0.126316	0.996574	-0.000021	0.999503
6	0.15750	0.00525	0.110526	0.701754	-0.002231	0.947189
7	0.12675	0.01600	0.101587	0.801478	-0.003963	0.971992
8	0.15750	0.01600	0.126233	0.801478	-0.003963	0.964216
9	0.12675	0.01025	0.112637	0.888658	-0.001284	0.984096
10	0.09100	0.01025	0.080868	0.888658	-0.001284	0.988976
11	0.09100	0.00900	0.088452	0.972001	-0.000259	0.997205
12	0.10175	0.00900	0.098901	0.972001	-0.000259	0.996838
13	0.15750	0.01000	0.109890	0.697715	-0.004332	0.946512
14	0.12675	0.00550	0.113402	0.894691	-0.000647	0.984945
15	0.06925	0.00500	0.103093	1.488704	0.001641	1.037733
16	0.15750	0.00550	0.113402	0.720013	-0.002139	0.950262
17	0.15750	0.00650	0.163522	1.038235	0.000239	1.007199
18	0.15750	0.00650	0.136126	0.864290	-0.001021	0.975258
19	0.12675	0.00575	0.083032	0.655089	-0.003027	0.952324
20	0.15750	0.00675	0.097473	0.618876	-0.004157	0.933490
21	0.09100	0.00500	0.076336	0.838856	-0.000960	0.984124
22	0.15750	0.00700	0.106870	0.678541	-0.003316	0.943312
23	0.09275	0.00525	0.080153	0.864180	-0.000825	0.986305
24	0.10175	0.00500	0.076336	0.750230	-0.001665	0.972486
25	0.15750	0.00500	0.130719	0.829962	-0.001024	0.969192

26	0.15750	0.00675	0.177632	1.127820	0.000765	1.024480
27	0.12675	0.00500	0.131579	1.038098	0.000184	1.005561
28	0.15750	0.01075	0.171315	1.087713	0.000867	1.016671
29	0.09275	0.00500	0.079681	0.859097	-0.000820	0.985800
30	0.09100	0.00625	0.099602	1.094523	0.000540	1.009553
31	0.12675	0.00575	0.091633	0.722946	-0.002204	0.961341
32	0.10175	0.00550	0.087649	0.861419	-0.000885	0.984545
33	0.15750	0.00925	0.099730	0.633209	-0.005358	0.935831
34	0.09275	0.00875	0.085995	0.927171	-0.000687	0.992610
35	0.10175	0.00875	0.094340	0.927171	-0.000687	0.991818
36	0.12675	0.00775	0.083558	0.659234	-0.004006	0.952870
37	0.09275	0.00525	0.096330	1.038601	0.000195	1.003962
38	0.15750	0.00625	0.114679	0.728120	-0.002334	0.951632
39	0.15750	0.00825	0.129921	0.824897	-0.001751	0.968303
40	0.09275	0.00500	0.078740	0.848950	-0.000890	0.984793
41	0.09100	0.00525	0.082677	0.908540	-0.000528	0.990927
42	0.12675	0.00600	0.094488	0.745469	-0.002049	0.964372
43	0.12675	0.00525	0.110526	0.872002	-0.000771	0.981760
44	0.15750	0.00600	0.126316	0.802005	-0.001481	0.964307
45	0.15750	0.00700	0.134615	0.854701	-0.001190	0.973556

	zhangs_metric
0	-0.222403
1	-0.233681
2	-0.161053
3	-0.164903
4	0.167408
5	-0.003596
6	-0.308530
7	-0.227203
8	-0.220970
9	-0.121139
10	-0.125476
11	-0.031072
12	-0.030716
13	-0.322779
14	-0.110086
15	0.345008
16	-0.290118
17	0.038351
18	-0.141552
19	-0.361302
20	-0.398190
21	-0.170513
22	-0.336410
23	-0.143969
24	-0.262678

```
25      -0.175613
26       0.117810
27       0.038150
28       0.086038
29      -0.148931
30       0.092142
31      -0.290219
32      -0.146500
33      -0.389677
34      -0.080416
35      -0.079681
36      -0.362958
37       0.039309
38      -0.283115
39      -0.184782
40      -0.159656
41      -0.097059
42      -0.267179
43      -0.133528
44      -0.205836
45      -0.152057
```

[ ]:
```
pd.merge(rules_dataset_1, rules_dataset_2, on=['antecedents', 'consequents'])
```

[ ]:
```
           antecedents          consequents  antecedent support_x  \
0     (tropical fruit)             (yogurt)               0.06925
1     (tropical fruit)         (whole milk)               0.06925
2     (tropical fruit)         (rolls/buns)               0.06925
3     (tropical fruit)               (soda)               0.06925
4       (citrus fruit)         (whole milk)               0.05550
5    (root vegetables)         (whole milk)               0.06875
6    (root vegetables)   (other vegetables)               0.06875
7             (yogurt)         (whole milk)               0.08975
8             (yogurt)         (rolls/buns)               0.08975
9             (yogurt)   (other vegetables)               0.08975
10       (canned beer)         (whole milk)               0.04875
11       (frankfurter)   (other vegetables)               0.03800
12        (whole milk)   (other vegetables)               0.16125
13  (other vegetables)         (whole milk)               0.12000
14           (sausage)             (yogurt)               0.06325
15           (sausage)               (soda)               0.06325
16           (sausage)   (other vegetables)               0.06325
17           (sausage)         (rolls/buns)               0.06325
18           (sausage)         (whole milk)               0.06325
19        (rolls/buns)         (whole milk)               0.11450
20        (whole milk)         (rolls/buns)               0.16125
21        (rolls/buns)   (other vegetables)               0.11450
```

22	(other vegetables)	(rolls/buns)	0.12000
23	(bottled water)	(other vegetables)	0.06175
24	(bottled water)	(whole milk)	0.06175
25	(soda)	(rolls/buns)	0.09625
26	(soda)	(other vegetables)	0.09625
27	(soda)	(whole milk)	0.09625
28	(shopping bags)	(whole milk)	0.04675
29	(shopping bags)	(other vegetables)	0.04675
30	(pastry)	(whole milk)	0.04725
31	(pip fruit)	(whole milk)	0.05125
32	(bottled beer)	(whole milk)	0.04425

	consequent support_x	support_x	confidence_x	lift_x	leverage_x	\
0	0.08975	0.00700	0.101083	1.126273	0.000785	
1	0.16125	0.00950	0.137184	0.850754	-0.001667	
2	0.11450	0.00600	0.086643	0.756704	-0.001929	
3	0.09625	0.00550	0.079422	0.825168	-0.001165	
4	0.16125	0.00800	0.144144	0.893917	-0.000949	
5	0.16125	0.00675	0.098182	0.608879	-0.004336	
6	0.12000	0.00550	0.080000	0.666667	-0.002750	
7	0.16125	0.01425	0.158774	0.984647	-0.000222	
8	0.11450	0.00675	0.075209	0.656846	-0.003526	
9	0.12000	0.00725	0.080780	0.673166	-0.003520	
10	0.16125	0.00625	0.128205	0.795071	-0.001611	
11	0.12000	0.00600	0.157895	1.315789	0.001440	
12	0.12000	0.01500	0.093023	0.775194	-0.004350	
13	0.16125	0.01500	0.125000	0.775194	-0.004350	
14	0.08975	0.00600	0.094862	1.056954	0.000323	
15	0.09625	0.00700	0.110672	1.149838	0.000912	
16	0.12000	0.00850	0.134387	1.119895	0.000910	
17	0.11450	0.00550	0.086957	0.759446	-0.001742	
18	0.16125	0.00900	0.142292	0.882434	-0.001199	
19	0.16125	0.01300	0.113537	0.704106	-0.005463	
20	0.11450	0.01300	0.080620	0.704106	-0.005463	
21	0.12000	0.01000	0.087336	0.727802	-0.003740	
22	0.11450	0.01000	0.083333	0.727802	-0.003740	
23	0.12000	0.00600	0.097166	0.809717	-0.001410	
24	0.16125	0.00550	0.089069	0.552365	-0.004457	
25	0.11450	0.00800	0.083117	0.725912	-0.003021	
26	0.12000	0.01050	0.109091	0.909091	-0.001050	
27	0.16125	0.01075	0.111688	0.692641	-0.004770	
28	0.16125	0.00875	0.187166	1.160718	0.001212	
29	0.12000	0.00500	0.106952	0.891266	-0.000610	
30	0.16125	0.00600	0.126984	0.787498	-0.001619	
31	0.16125	0.00750	0.146341	0.907544	-0.000764	
32	0.16125	0.00775	0.175141	1.086147	0.000615	

	conviction_x	zhangs_metric_x	antecedent support_y	consequent support_y	\
0	1.012607	0.120458	0.06550	0.09100	
1	0.972108	-0.158589	0.06550	0.15750	
2	0.969500	-0.256750	0.06550	0.10175	
3	0.981721	-0.185428	0.06550	0.09275	
4	0.980013	-0.111621	0.05200	0.15750	
5	0.930066	-0.408208	0.06925	0.15750	
6	0.956522	-0.349345	0.06925	0.12675	
7	0.997057	-0.016841	0.09100	0.15750	
8	0.957514	-0.364650	0.09100	0.10175	
9	0.957333	-0.347850	0.09100	0.12675	
10	0.962096	-0.213193	0.04775	0.15750	
11	1.045000	0.249480	0.03800	0.12675	
12	0.970256	-0.256921	0.15750	0.12675	
13	0.958571	-0.247863	0.12675	0.15750	
14	1.005647	0.057524	0.06275	0.09100	
15	1.016217	0.139111	0.06275	0.09275	
16	1.016621	0.114288	0.06275	0.12675	
17	0.969833	-0.252692	0.06275	0.10175	
18	0.977897	-0.124516	0.06275	0.15750	
19	0.946176	-0.321841	0.10175	0.15750	
20	0.963149	-0.333792	0.15750	0.10175	
21	0.964211	-0.296943	0.10175	0.12675	
22	0.966000	-0.298246	0.12675	0.10175	
23	0.974709	-0.200298	0.06350	0.12675	
24	0.920761	-0.463443	0.06350	0.15750	
25	0.965772	-0.294677	0.09275	0.10175	
26	0.987755	-0.099626	0.09275	0.12675	
27	0.944207	-0.329314	0.09275	0.15750	
28	1.031883	0.145255	0.04850	0.15750	
29	0.985389	-0.113462	0.04850	0.12675	
30	0.960750	-0.220714	0.05450	0.15750	
31	0.982536	-0.096966	0.04750	0.15750	
32	1.016841	0.082987	0.04750	0.15750	

	support_y	confidence_y	lift_y	leverage_y	conviction_y	\
0	0.00500	0.076336	0.838856	-0.000960	0.984124	
1	0.00700	0.106870	0.678541	-0.003316	0.943312	
2	0.00500	0.076336	0.750230	-0.001665	0.972486	
3	0.00525	0.080153	0.864180	-0.000825	0.986305	
4	0.00700	0.134615	0.854701	-0.001190	0.973556	
5	0.00675	0.097473	0.618876	-0.004157	0.933490	
6	0.00575	0.083032	0.655089	-0.003027	0.952324	
7	0.01000	0.109890	0.697715	-0.004332	0.946512	
8	0.00900	0.098901	0.972001	-0.000259	0.996838	
9	0.01025	0.112637	0.888658	-0.001284	0.984096	
10	0.00650	0.136126	0.864290	-0.001021	0.975258	

11	0.00500	0.131579	1.038098	0.000184	1.005561
12	0.01600	0.101587	0.801478	-0.003963	0.971992
13	0.01600	0.126233	0.801478	-0.003963	0.964216
14	0.00625	0.099602	1.094523	0.000540	1.009553
15	0.00500	0.079681	0.859097	-0.000820	0.985800
16	0.00575	0.091633	0.722946	-0.002204	0.961341
17	0.00550	0.087649	0.861419	-0.000885	0.984545
18	0.01075	0.171315	1.087713	0.000867	1.016671
19	0.01275	0.125307	0.795601	-0.003276	0.963195
20	0.01275	0.080952	0.795601	-0.003276	0.977370
21	0.01100	0.108108	0.852924	-0.001897	0.979098
22	0.01100	0.086785	0.852924	-0.001897	0.983613
23	0.00600	0.094488	0.745469	-0.002049	0.964372
24	0.00825	0.129921	0.824897	-0.001751	0.968303
25	0.00875	0.094340	0.927171	-0.000687	0.991818
26	0.00775	0.083558	0.659234	-0.004006	0.952870
27	0.00925	0.099730	0.633209	-0.005358	0.935831
28	0.00550	0.113402	0.720013	-0.002139	0.950262
29	0.00550	0.113402	0.894691	-0.000647	0.984945
30	0.00625	0.114679	0.728120	-0.002334	0.951632
31	0.00525	0.110526	0.701754	-0.002231	0.947189
32	0.00600	0.126316	0.802005	-0.001481	0.964307

	zhangs_metric_y
0	-0.170513
1	-0.336410
2	-0.262678
3	-0.143969
4	-0.152057
5	-0.398190
6	-0.361302
7	-0.322779
8	-0.030716
9	-0.121139
10	-0.141552
11	0.038150
12	-0.227203
13	-0.220970
14	0.092142
15	-0.148931
16	-0.290219
17	-0.146500
18	0.086038
19	-0.222403
20	-0.233681
21	-0.161053
22	-0.164903

```
23        -0.267179
24        -0.184782
25        -0.079681
26        -0.362958
27        -0.389677
28        -0.290118
29        -0.110086
30        -0.283115
31        -0.308530
32        -0.205836
```

6 2 Create and compile model

[ ]:
```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense
```

[ ]:
```
model = Sequential([
    Conv2D(8, (3, 3), activation='relu', input_shape=(256, 256, 3)),
    MaxPooling2D((2, 2)),
    Flatten(),
    Dense(16, activation='relu'),
    Dense(4, activation='softmax')
])
```

[ ]:
```
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=[tf.keras.metrics.CategoricalAccuracy(name='accuracy')])
```

[ ]:
```
model.summary()
```

```
Model: "sequential"

_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 254, 254, 8)       224

 max_pooling2d (MaxPooling2  (None, 127, 127, 8)       0
 D)

 flatten (Flatten)           (None, 129032)            0

 dense (Dense)               (None, 16)                2064528

 dense_1 (Dense)             (None, 4)                 68
```

```
=================================================================
Total params: 2064820 (7.88 MB)
Trainable params: 2064820 (7.88 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

7 Read Image Data

[ ]:
```
path = "/content/drive/MyDrive/data_mining/processed"
```

[ ]:
```
batch_size = 8
dataset = tf.keras.preprocessing.image_dataset_from_directory(
    directory=path,
    labels='inferred',
    label_mode='categorical',
    batch_size=batch_size,
    validation_split=0.2,
    subset='training',
    seed=100
)

# Define the validation dataset
validation_dataset = tf.keras.preprocessing.image_dataset_from_directory(
    directory=path,
    labels='inferred',
    label_mode='categorical',
    batch_size=batch_size,
    validation_split=0.2,
    subset='validation',
    seed=100
)
```

```
Found 713 files belonging to 4 classes.
Using 571 files for training.
Found 713 files belonging to 4 classes.
Using 142 files for validation.
```

8 Train Model

[ ]:
```
history = model.fit(dataset,validation_data=validation_dataset,epochs=20)
```

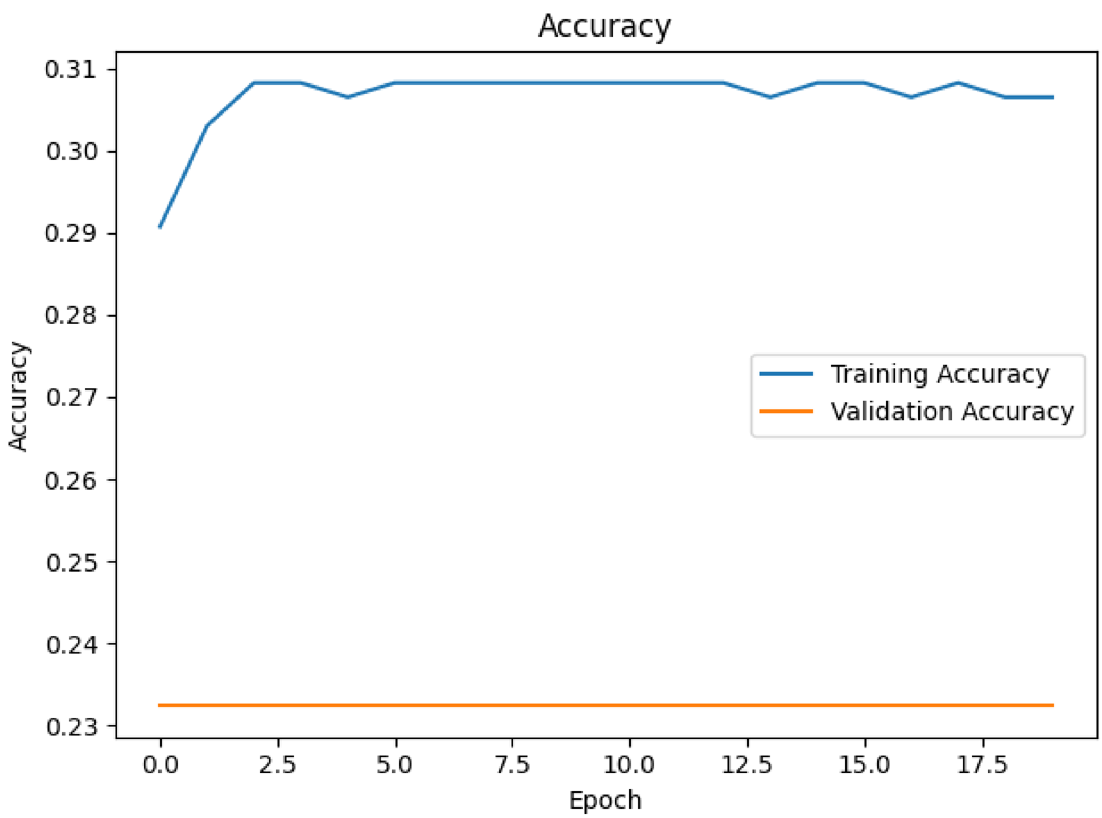
```
Epoch 1/20
72/72 [==============================] - 19s 245ms/step - loss: 169.0852 -
accuracy: 0.2907 - val_loss: 1.4193 - val_accuracy: 0.2324
Epoch 2/20
72/72 [==============================] - 20s 282ms/step - loss: 1.3835 -
accuracy: 0.3030 - val_loss: 1.3867 - val_accuracy: 0.2324
```

```
Epoch 3/20
72/72 [==============================] - 16s 220ms/step - loss: 1.3812 -
accuracy: 0.3082 - val_loss: 1.3858 - val_accuracy: 0.2324
Epoch 4/20
72/72 [==============================] - 17s 237ms/step - loss: 1.3788 -
accuracy: 0.3082 - val_loss: 1.3878 - val_accuracy: 0.2324
Epoch 5/20
72/72 [==============================] - 17s 233ms/step - loss: 1.3772 -
accuracy: 0.3065 - val_loss: 1.3942 - val_accuracy: 0.2324
Epoch 6/20
72/72 [==============================] - 16s 221ms/step - loss: 1.3762 -
accuracy: 0.3082 - val_loss: 1.3919 - val_accuracy: 0.2324
Epoch 7/20
72/72 [==============================] - 17s 231ms/step - loss: 1.3751 -
accuracy: 0.3082 - val_loss: 1.3935 - val_accuracy: 0.2324
Epoch 8/20
72/72 [==============================] - 17s 233ms/step - loss: 1.3742 -
accuracy: 0.3082 - val_loss: 1.3948 - val_accuracy: 0.2324
Epoch 9/20
72/72 [==============================] - 17s 232ms/step - loss: 1.3735 -
accuracy: 0.3082 - val_loss: 1.3961 - val_accuracy: 0.2324
Epoch 10/20
72/72 [==============================] - 17s 231ms/step - loss: 1.3728 -
accuracy: 0.3082 - val_loss: 1.3987 - val_accuracy: 0.2324
Epoch 11/20
72/72 [==============================] - 17s 230ms/step - loss: 1.3725 -
accuracy: 0.3082 - val_loss: 1.3977 - val_accuracy: 0.2324
Epoch 12/20
72/72 [==============================] - 17s 232ms/step - loss: 1.3721 -
accuracy: 0.3082 - val_loss: 1.4017 - val_accuracy: 0.2324
Epoch 13/20
72/72 [==============================] - 17s 234ms/step - loss: 1.3720 -
accuracy: 0.3082 - val_loss: 1.3989 - val_accuracy: 0.2324
Epoch 14/20
72/72 [==============================] - 17s 231ms/step - loss: 1.3713 -
accuracy: 0.3065 - val_loss: 1.4114 - val_accuracy: 0.2324
Epoch 15/20
72/72 [==============================] - 17s 232ms/step - loss: 1.3718 -
accuracy: 0.3082 - val_loss: 1.4003 - val_accuracy: 0.2324
Epoch 16/20
72/72 [==============================] - 17s 231ms/step - loss: 1.3712 -
accuracy: 0.3082 - val_loss: 1.4026 - val_accuracy: 0.2324
Epoch 17/20
72/72 [==============================] - 16s 221ms/step - loss: 1.3711 -
accuracy: 0.3065 - val_loss: 1.4061 - val_accuracy: 0.2324
Epoch 18/20
72/72 [==============================] - 17s 234ms/step - loss: 1.3711 -
accuracy: 0.3082 - val_loss: 1.4052 - val_accuracy: 0.2324
```

```
Epoch 19/20
72/72 [==============================] - 17s 230ms/step - loss: 1.3710 -
accuracy: 0.3065 - val_loss: 1.4060 - val_accuracy: 0.2324
Epoch 20/20
72/72 [==============================] - 17s 231ms/step - loss: 1.3709 -
accuracy: 0.3065 - val_loss: 1.4062 - val_accuracy: 0.2324
```

9 2 (a)

[ ]:
```
training_accuracy = history.history['accuracy']
validation_accuracy = history.history['val_accuracy']


plt.plot(training_accuracy, label='Training Accuracy')
plt.plot(validation_accuracy, label='Validation Accuracy')
plt.title('Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()

plt.tight_layout()
plt.show()
```



10 2 (b). Experiment with filters changes to 4 and 16

[ ]:
```
new_model1 = Sequential([
    Conv2D(4, (3, 3), activation='relu', input_shape=(256, 256, 3)),
    MaxPooling2D((2, 2)),
    Flatten(),
    Dense(16, activation='relu'),
    Dense(4, activation='softmax')
])

new_model2 = Sequential([
    Conv2D(16, (3, 3), activation='relu', input_shape=(256, 256, 3)),
    MaxPooling2D((2, 2)),
    Flatten(),
    Dense(16, activation='relu'),
    Dense(4, activation='softmax')
])
```

11 Compile new models

[ ]:
```
new_model1.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=[tf.keras.metrics.CategoricalAccuracy(name='accuracy')])
new_model2.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=[tf.keras.metrics.CategoricalAccuracy(name='accuracy')])
```

[ ]:
```
new_model1.summary()
```

```
Model: "sequential_1"

_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_1 (Conv2D)           (None, 254, 254, 4)       112

 max_pooling2d_1 (MaxPoolin  (None, 127, 127, 4)       0
 g2D)

 flatten_1 (Flatten)         (None, 64516)             0

 dense_2 (Dense)             (None, 16)                1032272

 dense_3 (Dense)             (None, 4)                 68
```

```
=================================================================
Total params: 1032452 (3.94 MB)
Trainable params: 1032452 (3.94 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

[ ]:
```
new_model2.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_2 (Conv2D)           (None, 254, 254, 16)      448

 max_pooling2d_2 (MaxPoolin  (None, 127, 127, 16)      0
 g2D)

 flatten_2 (Flatten)         (None, 258064)            0

 dense_4 (Dense)             (None, 16)                4129040

 dense_5 (Dense)             (None, 4)                 68

=================================================================
Total params: 4129556 (15.75 MB)
Trainable params: 4129556 (15.75 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

12 Train new model 1

[ ]:
```
new_history1 = new_model1.
 ↪fit(dataset,validation_data=validation_dataset,epochs=20)
```

```
Epoch 1/20
72/72 [==============================] - 16s 208ms/step - loss: 2.0842 -
accuracy: 0.2627 - val_loss: 1.5421 - val_accuracy: 0.2606
Epoch 2/20
72/72 [==============================] - 15s 207ms/step - loss: 1.1186 -
accuracy: 0.4904 - val_loss: 1.5606 - val_accuracy: 0.2746
Epoch 3/20
72/72 [==============================] - 16s 216ms/step - loss: 0.9246 -
accuracy: 0.6130 - val_loss: 1.8147 - val_accuracy: 0.3169
Epoch 4/20
72/72 [==============================] - 15s 204ms/step - loss: 0.8059 -
accuracy: 0.6848 - val_loss: 2.1588 - val_accuracy: 0.3028
```

```
Epoch 5/20
72/72 [==============================] - 16s 218ms/step - loss: 0.7892 -
accuracy: 0.7356 - val_loss: 1.9642 - val_accuracy: 0.3028
Epoch 6/20
72/72 [==============================] - 16s 218ms/step - loss: 0.5530 -
accuracy: 0.7968 - val_loss: 2.2395 - val_accuracy: 0.2676
Epoch 7/20
72/72 [==============================] - 15s 207ms/step - loss: 0.5050 -
accuracy: 0.8319 - val_loss: 2.2508 - val_accuracy: 0.2746
Epoch 8/20
72/72 [==============================] - 15s 206ms/step - loss: 0.3591 -
accuracy: 0.8704 - val_loss: 2.8110 - val_accuracy: 0.2958
Epoch 9/20
72/72 [==============================] - 15s 205ms/step - loss: 0.3166 -
accuracy: 0.8862 - val_loss: 3.0061 - val_accuracy: 0.2746
Epoch 10/20
72/72 [==============================] - 16s 219ms/step - loss: 0.2605 -
accuracy: 0.9019 - val_loss: 3.3791 - val_accuracy: 0.2746
Epoch 11/20
72/72 [==============================] - 16s 219ms/step - loss: 0.2352 -
accuracy: 0.9089 - val_loss: 3.8389 - val_accuracy: 0.2606
Epoch 12/20
72/72 [==============================] - 16s 218ms/step - loss: 0.2152 -
accuracy: 0.9124 - val_loss: 4.0403 - val_accuracy: 0.3028
Epoch 13/20
72/72 [==============================] - 16s 215ms/step - loss: 0.2067 -
accuracy: 0.9212 - val_loss: 3.5537 - val_accuracy: 0.2394
Epoch 14/20
72/72 [==============================] - 16s 218ms/step - loss: 0.2498 -
accuracy: 0.9142 - val_loss: 3.9277 - val_accuracy: 0.2606
Epoch 15/20
72/72 [==============================] - 16s 218ms/step - loss: 0.2288 -
accuracy: 0.9194 - val_loss: 5.9634 - val_accuracy: 0.2535
Epoch 16/20
72/72 [==============================] - 15s 205ms/step - loss: 0.2437 -
accuracy: 0.9107 - val_loss: 3.1110 - val_accuracy: 0.2746
Epoch 17/20
72/72 [==============================] - 15s 205ms/step - loss: 0.1805 -
accuracy: 0.9159 - val_loss: 4.0207 - val_accuracy: 0.2606
Epoch 18/20
72/72 [==============================] - 16s 220ms/step - loss: 0.2009 -
accuracy: 0.9177 - val_loss: 4.1832 - val_accuracy: 0.2887
Epoch 19/20
72/72 [==============================] - 16s 219ms/step - loss: 0.1364 -
accuracy: 0.9299 - val_loss: 5.5756 - val_accuracy: 0.2465
Epoch 20/20
72/72 [==============================] - 15s 210ms/step - loss: 0.2224 -
accuracy: 0.9037 - val_loss: 3.2049 - val_accuracy: 0.2746
```

13 Train New Model 2

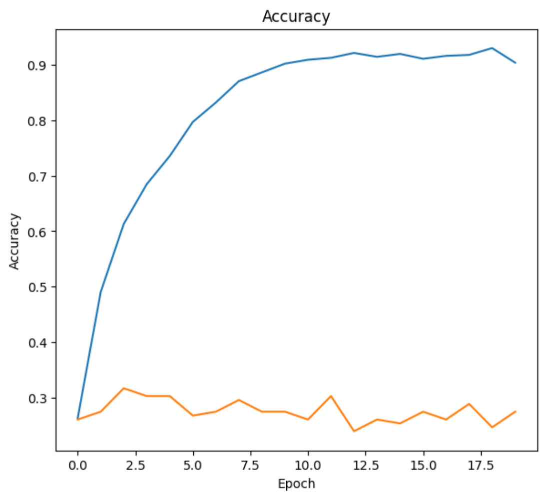
[ ]:
```
new_history2 = new_model2.
  ↪fit(dataset,validation_data=validation_dataset,epochs=20)
```

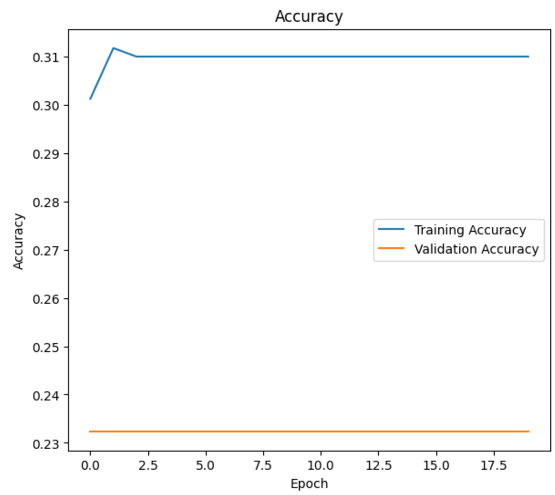
```
Epoch 1/20
72/72 [==============================] - 23s 291ms/step - loss: 240.3787 -
accuracy: 0.3012 - val_loss: 1.4422 - val_accuracy: 0.2324
Epoch 2/20
72/72 [==============================] - 27s 371ms/step - loss: 1.3867 -
accuracy: 0.3117 - val_loss: 1.4179 - val_accuracy: 0.2324
Epoch 3/20
72/72 [==============================] - 19s 265ms/step - loss: 1.3778 -
accuracy: 0.3100 - val_loss: 1.4229 - val_accuracy: 0.2324
Epoch 4/20
72/72 [==============================] - 20s 279ms/step - loss: 1.3760 -
accuracy: 0.3100 - val_loss: 1.4244 - val_accuracy: 0.2324
Epoch 5/20
72/72 [==============================] - 21s 288ms/step - loss: 1.3746 -
accuracy: 0.3100 - val_loss: 1.4252 - val_accuracy: 0.2324
Epoch 6/20
72/72 [==============================] - 28s 392ms/step - loss: 1.3734 -
accuracy: 0.3100 - val_loss: 1.4262 - val_accuracy: 0.2324
Epoch 7/20
72/72 [==============================] - 29s 399ms/step - loss: 1.3724 -
accuracy: 0.3100 - val_loss: 1.4272 - val_accuracy: 0.2324
Epoch 8/20
72/72 [==============================] - 29s 406ms/step - loss: 1.3715 -
accuracy: 0.3100 - val_loss: 1.4282 - val_accuracy: 0.2324
Epoch 9/20
72/72 [==============================] - 23s 314ms/step - loss: 1.3709 -
accuracy: 0.3100 - val_loss: 1.4290 - val_accuracy: 0.2324
Epoch 10/20
72/72 [==============================] - 20s 274ms/step - loss: 1.3703 -
accuracy: 0.3100 - val_loss: 1.4301 - val_accuracy: 0.2324
Epoch 11/20
72/72 [==============================] - 20s 280ms/step - loss: 1.3699 -
accuracy: 0.3100 - val_loss: 1.4312 - val_accuracy: 0.2324
Epoch 12/20
72/72 [==============================] - 20s 271ms/step - loss: 1.3695 -
accuracy: 0.3100 - val_loss: 1.4320 - val_accuracy: 0.2324
Epoch 13/20
72/72 [==============================] - 20s 272ms/step - loss: 1.3692 -
accuracy: 0.3100 - val_loss: 1.4327 - val_accuracy: 0.2324
Epoch 14/20
72/72 [==============================] - 19s 264ms/step - loss: 1.3689 -
accuracy: 0.3100 - val_loss: 1.4335 - val_accuracy: 0.2324
Epoch 15/20
```

```
72/72 [==============================] - 19s 256ms/step - loss: 1.3687 -
accuracy: 0.3100 - val_loss: 1.4342 - val_accuracy: 0.2324
Epoch 16/20
72/72 [==============================] - 23s 315ms/step - loss: 1.3686 -
accuracy: 0.3100 - val_loss: 1.4349 - val_accuracy: 0.2324
Epoch 17/20
72/72 [==============================] - 22s 305ms/step - loss: 1.3685 -
accuracy: 0.3100 - val_loss: 1.4355 - val_accuracy: 0.2324
Epoch 18/20
72/72 [==============================] - 20s 273ms/step - loss: 1.3684 -
accuracy: 0.3100 - val_loss: 1.4362 - val_accuracy: 0.2324
Epoch 19/20
72/72 [==============================] - 23s 311ms/step - loss: 1.3683 -
accuracy: 0.3100 - val_loss: 1.4366 - val_accuracy: 0.2324
Epoch 20/20
72/72 [==============================] - 20s 270ms/step - loss: 1.3683 -
accuracy: 0.3100 - val_loss: 1.4375 - val_accuracy: 0.2324
```

14 2 (c)

[ ]:
```
training_accuracy1 = new_history1.history['accuracy']
validation_accuracy1 = new_history1.history['val_accuracy']

training_accuracy2 = new_history2.history['accuracy']
validation_accuracy2 = new_history2.history['val_accuracy']

fig,axes = plt.subplots(1,2,figsize=(15,6))
axes[0].plot(training_accuracy1, label='Training Accuracy')
axes[0].plot(validation_accuracy1, label='Validation Accuracy')
axes[0].set_title('Accuracy')
axes[0].set_xlabel('Epoch')
axes[0].set_ylabel('Accuracy')

axes[1].plot(training_accuracy2, label='Training Accuracy')
axes[1].plot(validation_accuracy2, label='Validation Accuracy')
axes[1].set_title('Accuracy')
axes[1].set_xlabel('Epoch')
axes[1].set_ylabel('Accuracy')
plt.legend()

plt.show()
```





15 2 (d)

1. The initial model appears to suffer from underfitting, as persistently low accuracies across both training and validation sets.
2. In the experiment model 1 with four filters is overfitting. While the training accuracy increased to 94%, the validation accuracy remained below 25%.
3. In the experiment model 1 with sixteen filters, is underfitting. Both training and validation accuracies are strikingly low.

```
[ ]:
```