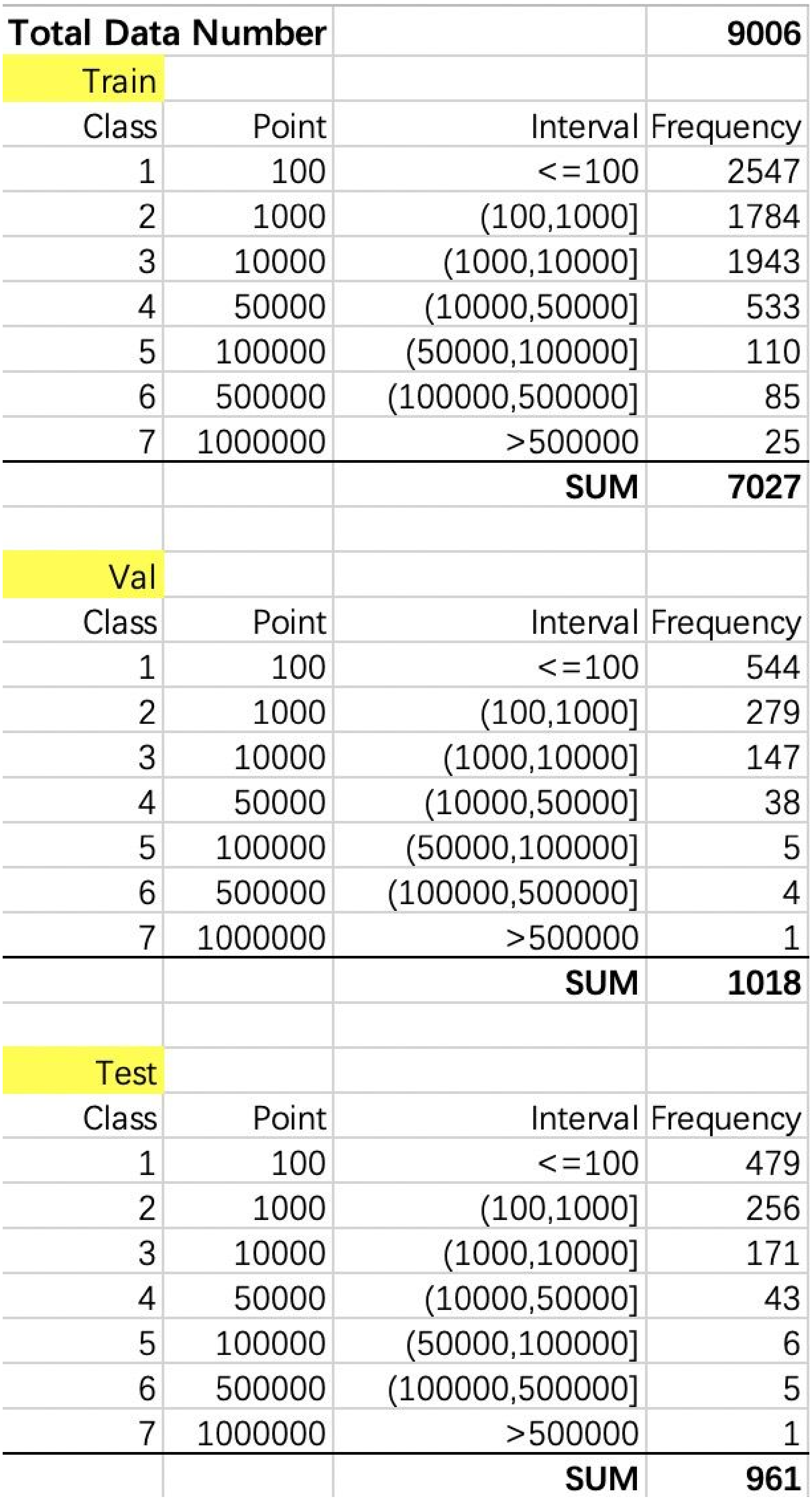
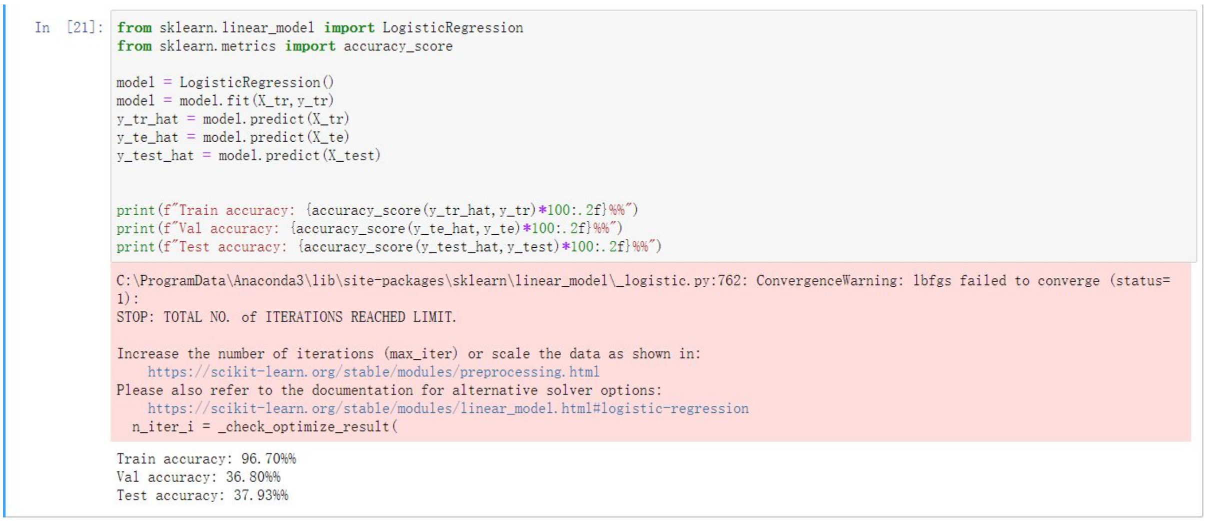
Three Datasets & Corresponding Outcome Using VGG and ResNet：

1. IMAGES_NEW: (ver.1, imbalanced)



Total Data Number			**9006**
Train			
Class	Point	Interval	Frequency
1	100	<=100	2547
2	1000	(100,1000]	1784
3	10000	(1000,10000]	1943
4	50000	(10000,50000]	533
5	100000	(50000,100000]	110
6	500000	(100000,500000]	85
7	1000000	>500000	25
		SUM	**7027**
Val			
Class	Point	Interval	Frequency
1	100	<=100	544
2	1000	(100,1000]	279
3	10000	(1000,10000]	147
4	50000	(10000,50000]	38
5	100000	(50000,100000]	5
6	500000	(100000,500000]	4
7	1000000	>500000	1
		SUM	**1018**
Test			
Class	Point	Interval	Frequency
1	100	<=100	479
2	1000	(100,1000]	256
3	10000	(1000,10000]	171
4	50000	(10000,50000]	43
5	100000	(50000,100000]	6
6	500000	(100000,500000]	5
7	1000000	>500000	1
		SUM	**961**

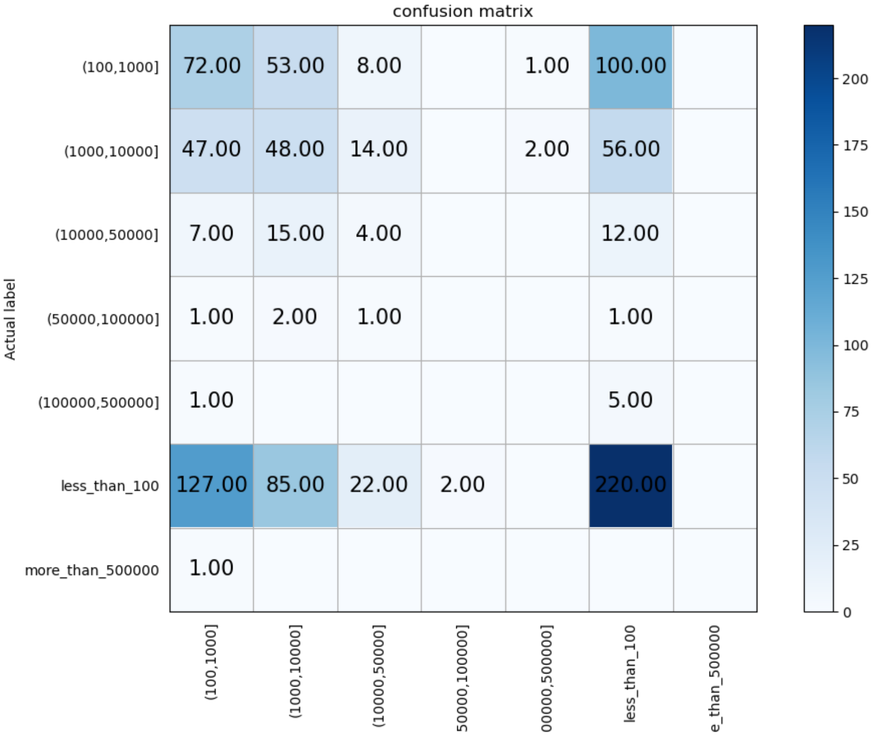
VGG



In [21]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

model = LogisticRegression()
model = model.fit(X_tr, y_tr)
y_tr_hat = model.predict(X_tr)
y_te_hat = model.predict(X_te)
y_test_hat = model.predict(X_test)


print(f"Train accuracy: {accuracy_score(y_tr_hat, y_tr)*100:.2f}%%")
print(f"Val accuracy: {accuracy_score(y_te_hat, y_te)*100:.2f}%%")
print(f"Test accuracy: {accuracy_score(y_test_hat, y_test)*100:.2f}%%")
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:762: ConvergenceWarning: lbfgs failed to converge (status=
1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

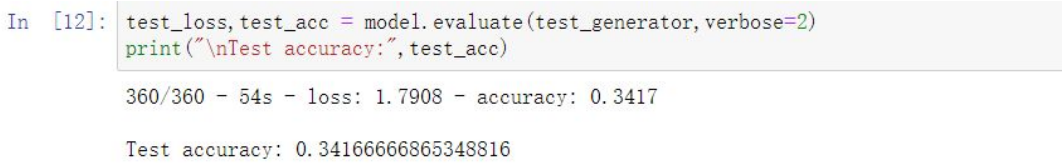
```
Train accuracy: 96.70%%
Val accuracy: 36.80%%
Test accuracy: 37.93%%
```



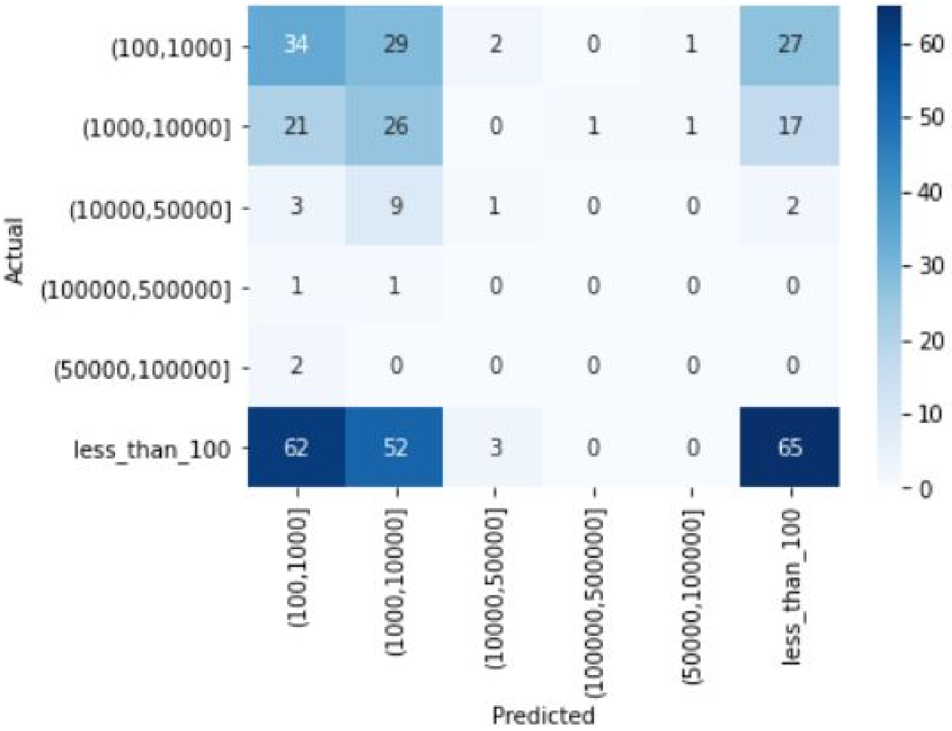
confusion matrix

	(100,1000]	(1000,10000]	(10000,50000]	50000,100000]	00000,500000]	less_than_100	e_than_500000
(100,1000]	72.00	53.00	8.00		1.00	100.00	
(1000,10000]	47.00	48.00	14.00		2.00	56.00	
(10000,50000]	7.00	15.00	4.00			12.00	
(50000,100000]	1.00	2.00	1.00			1.00	
(100000,500000]	1.00					5.00	
less_than_100	127.00	85.00	22.00	2.00		220.00	
more_than_500000	1.00						

RESNET

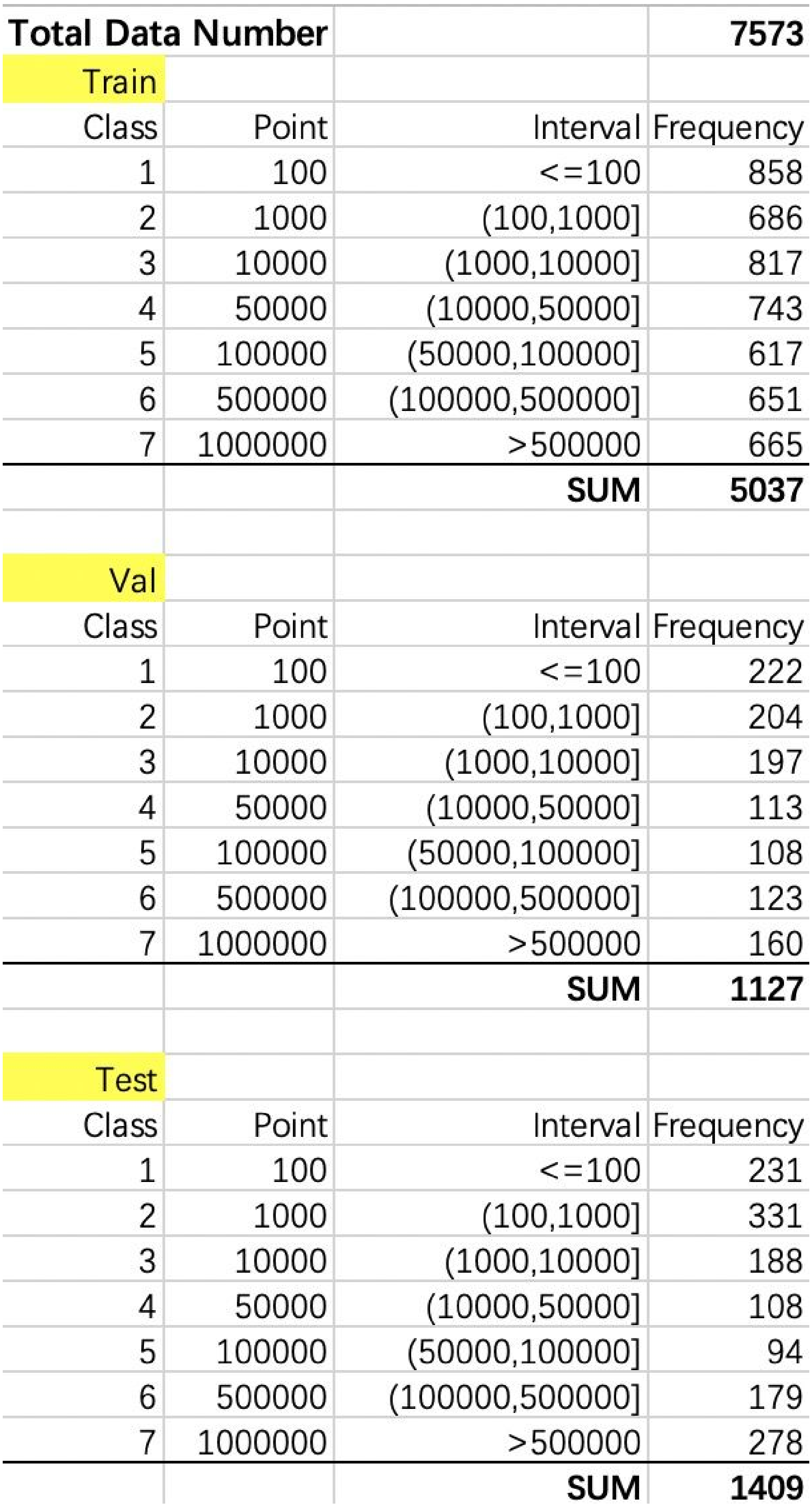


```
In  [12]: test_loss, test_acc = model.evaluate(test_generator, verbose=2)
          print("\nTest accuracy:", test_acc)

          360/360 - 54s - loss: 1.7908 - accuracy: 0.3417

          Test accuracy: 0.34166666865348816
```

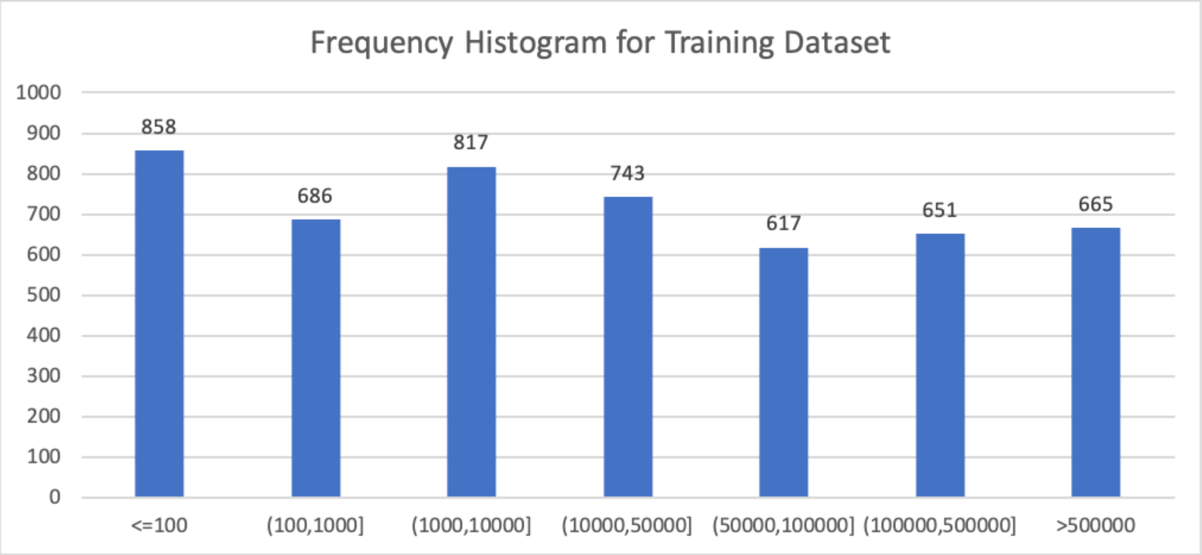


```
test accuracy: 35.0
```

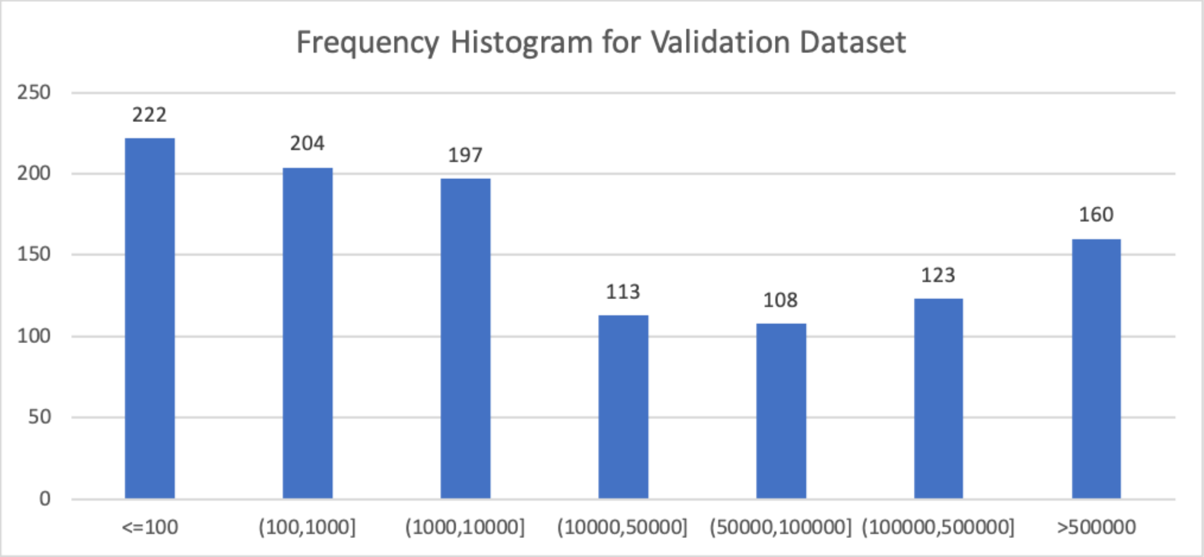
2. coverimage: (the balanced one)



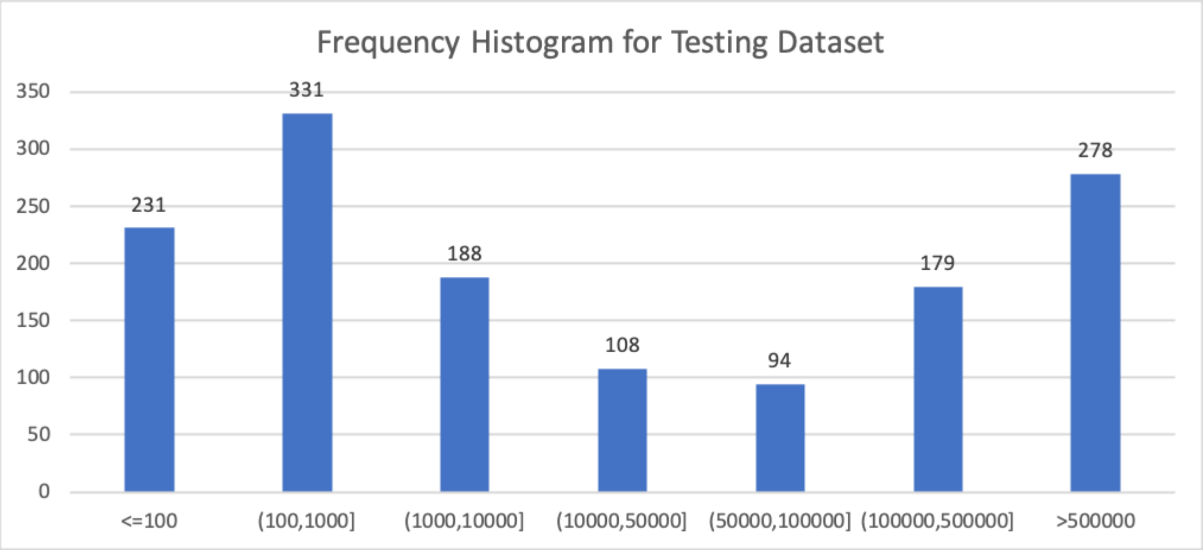
Total Data Number			**7573**
Train			
Class	Point	Interval	Frequency
1	100	<=100	858
2	1000	(100,1000]	686
3	10000	(1000,10000]	817
4	50000	(10000,50000]	743
5	100000	(50000,100000]	617
6	500000	(100000,500000]	651
7	1000000	>500000	665
		SUM	**5037**
Val			
Class	Point	Interval	Frequency
1	100	<=100	222
2	1000	(100,1000]	204
3	10000	(1000,10000]	197
4	50000	(10000,50000]	113
5	100000	(50000,100000]	108
6	500000	(100000,500000]	123
7	1000000	>500000	160
		SUM	**1127**
Test			
Class	Point	Interval	Frequency
1	100	<=100	231
2	1000	(100,1000]	331
3	10000	(1000,10000]	188
4	50000	(10000,50000]	108
5	100000	(50000,100000]	94
6	500000	(100000,500000]	179
7	1000000	>500000	278
		SUM	**1409**



Frequency Histogram for Training Dataset



Frequency Histogram for Validation Dataset

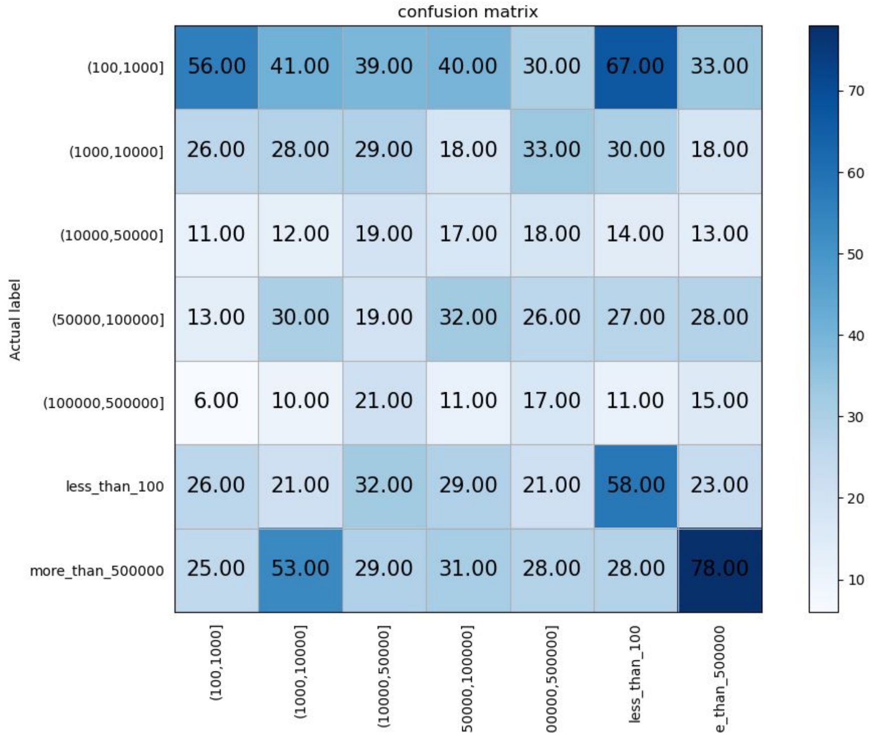


Frequency Histogram for Testing Dataset

VGG

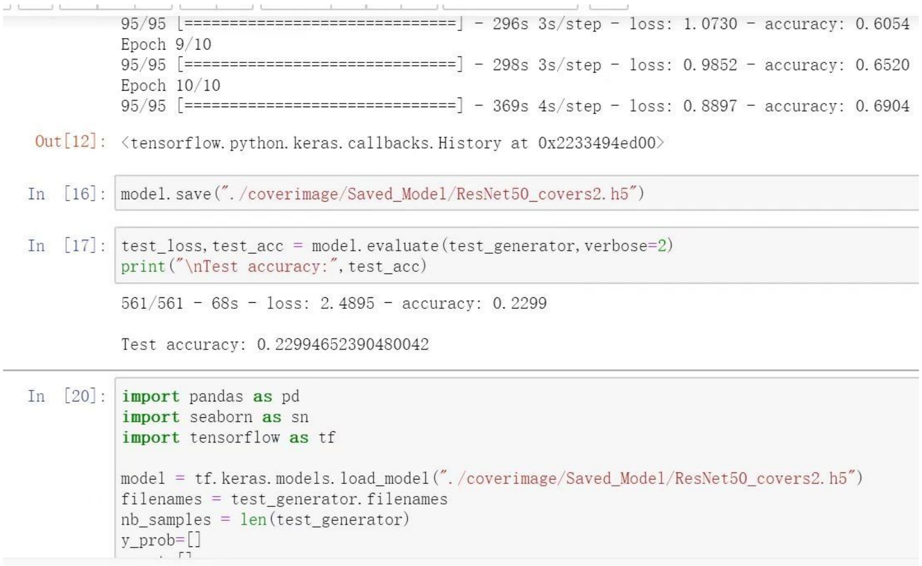
```
In [22]: from sklearn.linear_model import LogisticRegression
         from sklearn.metrics import accuracy_score

         model = LogisticRegression()
         model = model.fit(X_tr,y_tr)
         y_tr_hat = model.predict(X_tr)
         y_te_hat = model.predict(X_te)
         y_test_hat = model.predict(X_test)


         print(f"Train accuracy: {accuracy_score(y_tr_hat,y_tr)*100:.2f}%%")
         print(f"Val accuracy: {accuracy_score(y_te_hat,y_te)*100:.2f}%%")
         print(f"Test accuracy: {accuracy_score(y_test_hat,y_test)*100:.2f}%%")
```

```
Train accuracy: 99.84%%
Val accuracy: 20.39%%
Test accuracy: 21.49%%
```



	(100,1000]	(1000,10000]	(10000,50000]	50000,100000]	00000,500000]	less_than_100	e_than_500000
(100,1000]	56.00	41.00	39.00	40.00	30.00	67.00	33.00
(1000,10000]	26.00	28.00	29.00	18.00	33.00	30.00	18.00
(10000,50000]	11.00	12.00	19.00	17.00	18.00	14.00	13.00
(50000,100000]	13.00	30.00	19.00	32.00	26.00	27.00	28.00
(100000,500000]	6.00	10.00	21.00	11.00	17.00	11.00	15.00
less_than_100	26.00	21.00	32.00	29.00	21.00	58.00	23.00
more_than_500000	25.00	53.00	29.00	31.00	28.00	28.00	78.00

RESNET



```
95/95 [==============================] - 296s 3s/step - loss: 1.0730 - accuracy: 0.6054
Epoch 9/10
95/95 [==============================] - 298s 3s/step - loss: 0.9852 - accuracy: 0.6520
Epoch 10/10
95/95 [==============================] - 369s 4s/step - loss: 0.8897 - accuracy: 0.6904
```

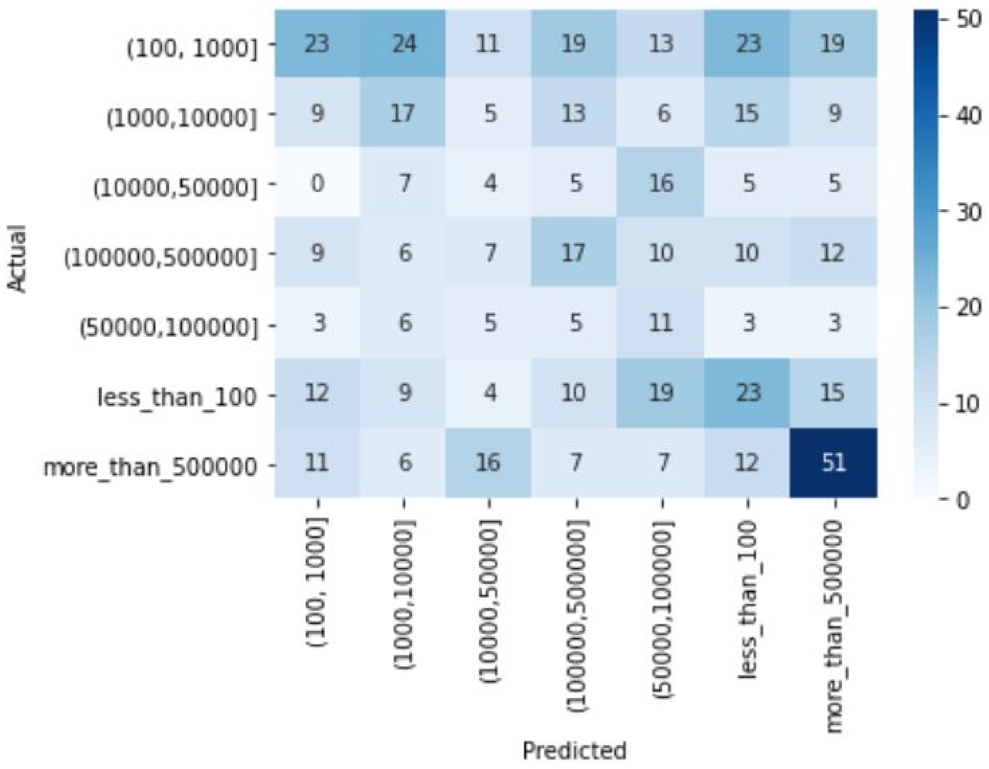
```
Out[12]: <tensorflow.python.keras.callbacks.History at 0x2233494ed00>
```

```
In [16]: model.save("./coverimage/Saved_Model/ResNet50_covers2.h5")
```

```
In [17]: test_loss,test_acc = model.evaluate(test_generator,verbose=2)
         print("\nTest accuracy:",test_acc)
```

```
561/561 - 68s - loss: 2.4895 - accuracy: 0.2299

Test accuracy: 0.22994652390480042
```

```
In [20]: import pandas as pd
         import seaborn as sn
         import tensorflow as tf

         model = tf.keras.models.load_model("./coverimage/Saved_Model/ResNet50_covers2.h5")
         filenames = test_generator.filenames
         nb_samples = len(test_generator)
         y_prob=[]
```

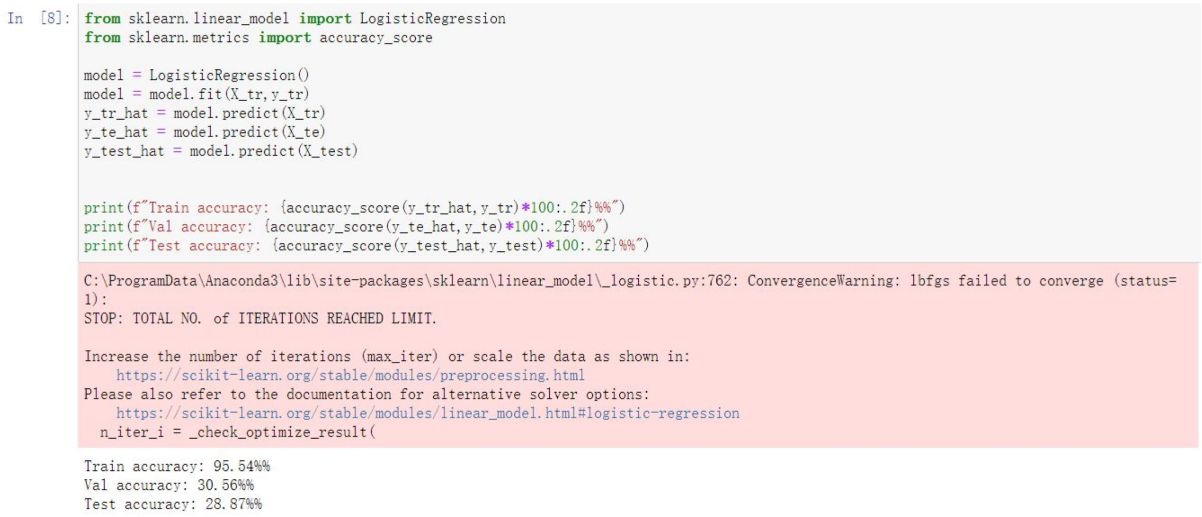


Actual \ Predicted	(100, 1000]	(1000,10000]	(10000,50000]	(100000,500000]	(50000,100000]	less_than_100	more_than_500000
(100, 1000]	23	24	11	19	13	23	19
(1000,10000]	9	17	5	13	6	15	9
(10000,50000]	0	7	4	5	16	5	5
(100000,500000]	9	6	7	17	10	10	12
(50000,100000]	3	6	5	5	11	3	3
less_than_100	12	9	4	10	19	23	15
more_than_500000	11	6	16	7	7	12	51

```
test accuracy: 26.21184919210054
```

3. Image_all:(all data)

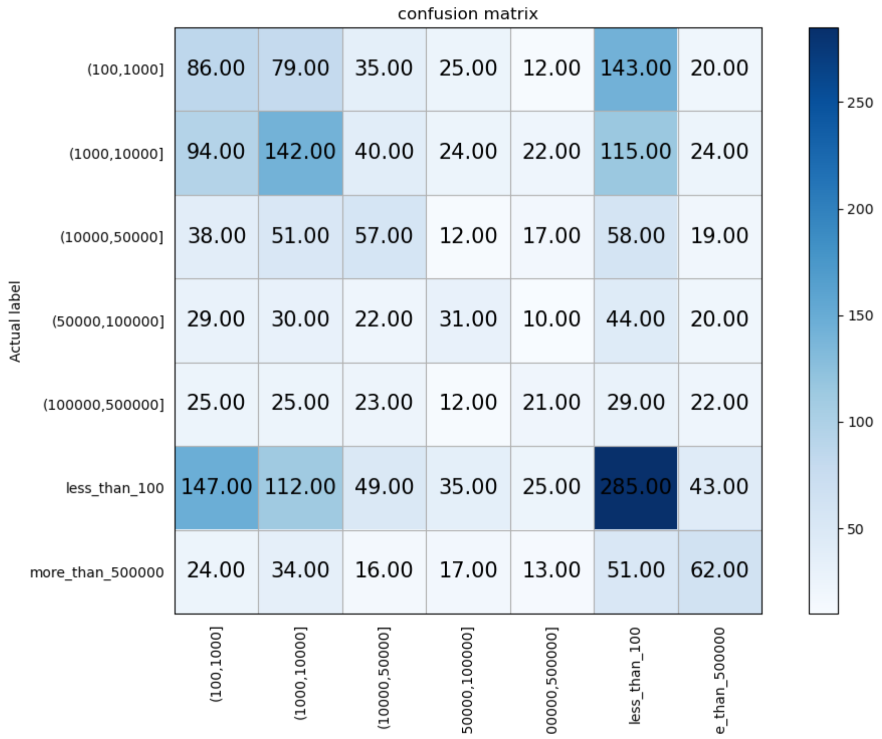
VGG



```
In  [8]: from sklearn.linear_model import LogisticRegression
         from sklearn.metrics import accuracy_score

         model = LogisticRegression()
         model = model.fit(X_tr, y_tr)
         y_tr_hat = model.predict(X_tr)
         y_te_hat = model.predict(X_te)
         y_test_hat = model.predict(X_test)


         print(f"Train accuracy: {accuracy_score(y_tr_hat, y_tr)*100:.2f}%%")
         print(f"Val accuracy: {accuracy_score(y_te_hat, y_te)*100:.2f}%%")
         print(f"Test accuracy: {accuracy_score(y_test_hat, y_test)*100:.2f}%%")
```

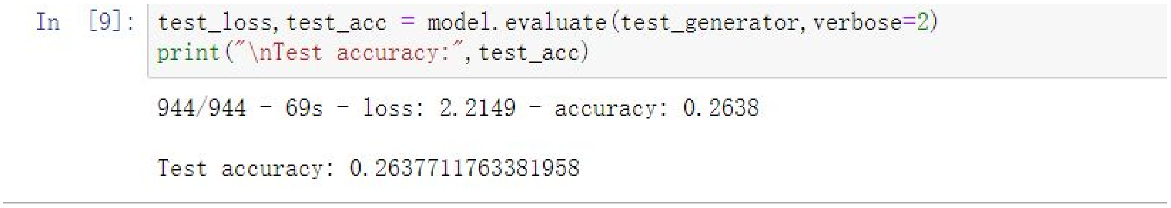
```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:762: ConvergenceWarning: lbfgs failed to converge (status=
1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

```
Train accuracy: 95.54%%
Val accuracy: 30.56%%
Test accuracy: 28.87%%
```

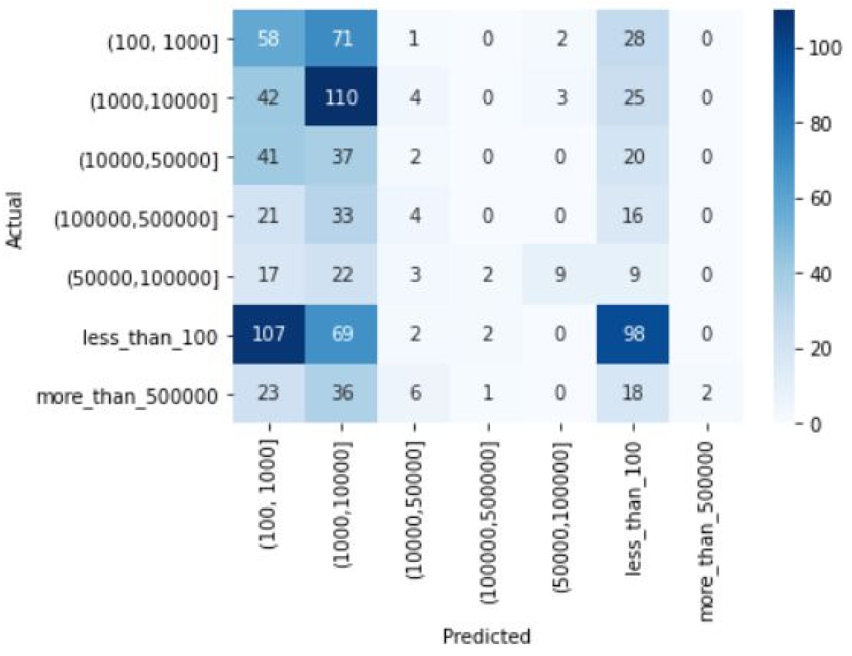


	(100,1000]	(1000,10000]	(10000,50000]	50000,100000]	00000,500000]	less_than_100	e_than_500000
(100,1000]	86.00	79.00	35.00	25.00	12.00	143.00	20.00
(1000,10000]	94.00	142.00	40.00	24.00	22.00	115.00	24.00
(10000,50000]	38.00	51.00	57.00	12.00	17.00	58.00	19.00
(50000,100000]	29.00	30.00	22.00	31.00	10.00	44.00	20.00
(100000,500000]	25.00	25.00	23.00	12.00	21.00	29.00	22.00
less_than_100	147.00	112.00	49.00	35.00	25.00	285.00	43.00
more_than_500000	24.00	34.00	16.00	17.00	13.00	51.00	62.00

RESNET



```
In [9]: test_loss, test_acc = model.evaluate(test_generator, verbose=2)
        print("\nTest accuracy:", test_acc)

        944/944 - 69s - loss: 2.2149 - accuracy: 0.2638

        Test accuracy: 0.2637711763381958
```



Actual \ Predicted	(100, 1000]	(1000,10000]	(10000,50000]	(100000,500000]	(50000,100000]	less_than_100	more_than_500000
(100, 1000]	58	71	1	0	2	28	0
(1000,10000]	42	110	4	0	3	25	0
(10000,50000]	41	37	2	0	0	20	0
(100000,500000]	21	33	4	0	0	16	0
(50000,100000]	17	22	3	2	9	9	0
less_than_100	107	69	2	2	0	98	0
more_than_500000	23	36	6	1	0	18	2

```
test accuracy: 29.55508474576271
```

Test Accuracy

	Dataset 1 (Imbalanced)	Dataset 2 (Balanced)	Dataset 3 (All data)
VGG16	37.93%	21.49%	28.87%
ResNet50	34.17%	26.21%	26.38%
Baseline Accuracy	2547/7027= 36.24%	858/5037= 17.03%	3473/11813= 29.39%