

In [38]:

```
import seaborn as sns
import pandas as pd
import matplotlib.pyplot as plt
from imblearn.over_sampling import RandomOverSampler
import numpy as np
from sklearn.model_selection import train_test_split
import os, cv2
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, Flatten, Dense, MaxPool2D, Activation
```

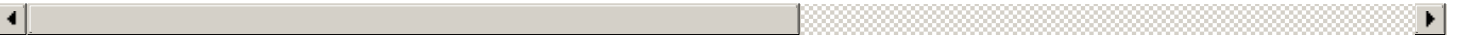
In [39]:

```
data = pd.read_csv('/kaggle/input/skin-cancer-mnist-ham10000/hmnist_28_28_RGB.csv')
data.head()
```

Out[39]:

	pixel0000	pixel0001	pixel0002	pixel0003	pixel0004	pixel0005	pixel0006	pixel0007	pixel0008	pixel0009	...	pixel2343	pixel2344
0	192	153	193	195	155	192	197	154	185	202	...	173	174
1	25	14	30	68	48	75	123	93	126	158	...	60	61
2	192	138	153	200	145	163	201	142	160	206	...	167	168
3	38	19	30	95	59	72	143	103	119	171	...	44	45
4	158	113	139	194	144	174	215	162	191	225	...	209	210

5 rows × 2353 columns



In [40]:

```
data['label'].unique()
```

Out[40]:

```
array([2, 4, 3, 6, 5, 1, 0])
```

In [41]:

```
y = data['label']
x = data.drop(columns = ['label'])
```

In [42]:

```
data.isnull().sum().sum() #no null values present
```

Out[42]:

```
0
```

In [43]:

```
meta_data = pd.read_csv('/kaggle/input/skin-cancer-mnist-ham10000/HAM10000_metadata.csv')
meta_data.head()
```

Out[43]:

	lesion_id	image_id	dx	dx_type	age	sex	localization
0	HAM_0000118	ISIC_0027419	bkl	histo	80.0	male	scalp
1	HAM_0000118	ISIC_0025030	bkl	histo	80.0	male	scalp
2	HAM_0002730	ISIC_0026769	bkl	histo	80.0	male	scalp
3	HAM_0002730	ISIC_0025664	bkl	histo	80.0	male	scalp

	lesion_id	image_id	dx	dx_type	age	sex	localization
3	HAM_0002730	ISIC_0025030	bkl	histo	80.0	male	scalp
4	HAM_0001466	ISIC_0031633	bkl	histo	75.0	male	ear

In [44]:

```
meta_data['dx'].unique()
```

Out[44]:

```
array(['bkl', 'nv', 'df', 'mel', 'vasc', 'bcc', 'akiec'], dtype=object)
```

In [45]:

```
y = data['label']
x = data.drop(columns = ['label'])
```

In [46]:

```
data.isnull().sum().sum() #no null values present
```

Out[46]:

```
0
```

In [47]:

```
meta_data = pd.read_csv('/kaggle/input/skin-cancer-mnist-ham10000/HAM10000_metadata.csv')
meta_data.head()
```

Out[47]:

	lesion_id	image_id	dx	dx_type	age	sex	localization
0	HAM_0000118	ISIC_0027419	bkl	histo	80.0	male	scalp
1	HAM_0000118	ISIC_0025030	bkl	histo	80.0	male	scalp
2	HAM_0002730	ISIC_0026769	bkl	histo	80.0	male	scalp
3	HAM_0002730	ISIC_0025661	bkl	histo	80.0	male	scalp
4	HAM_0001466	ISIC_0031633	bkl	histo	75.0	male	ear

In [48]:

```
meta_data['dx'].unique()
```

Out[48]:

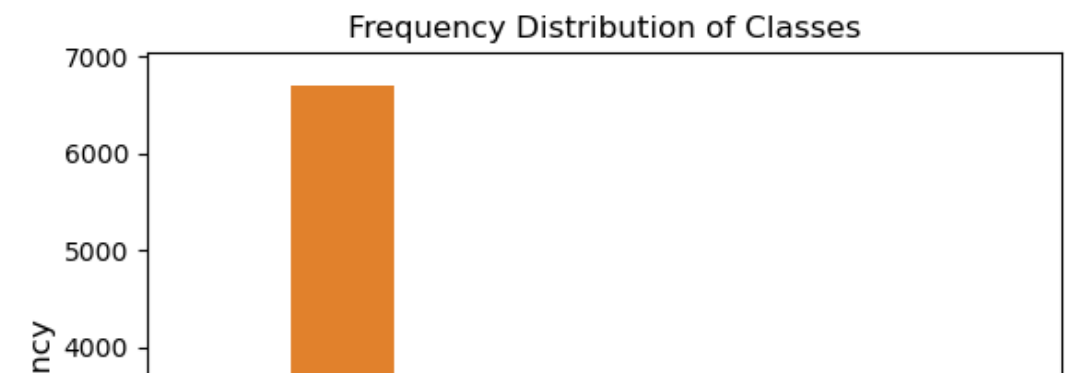
```
array(['bkl', 'nv', 'df', 'mel', 'vasc', 'bcc', 'akiec'], dtype=object)
```

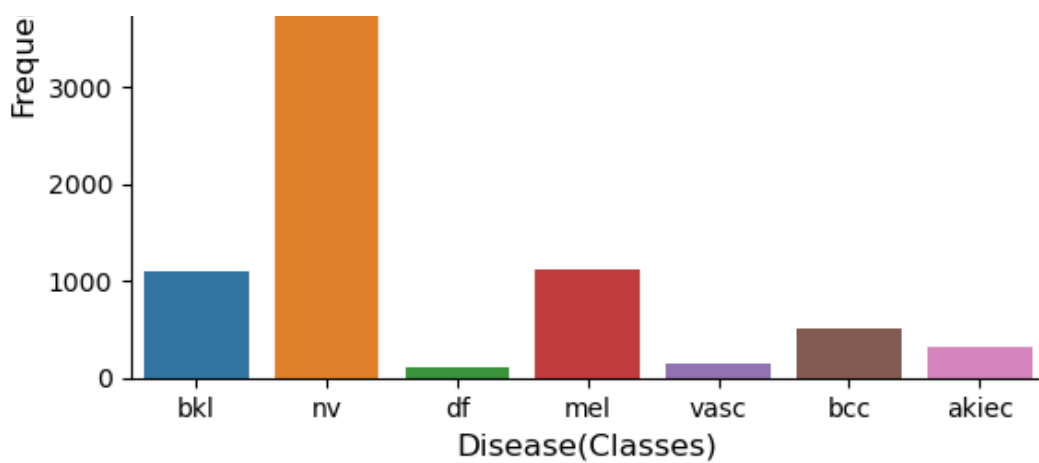
In [49]:

```
sns.countplot(x = 'dx', data = meta_data)
plt.xlabel('Disease (Classes)', size=12)
plt.ylabel('Frequency', size=12)
plt.title('Frequency Distribution of Classes')
```

Out[49]:

```
Text(0.5, 1.0, 'Frequency Distribution of Classes')
```





In [50]:

```
classes = {2:'bkl', 4:'nv', 3:'df', 6:'mel', 5:'vasc', 1:'bcc', 0:'akiec'}

classes_labels=[]
for key in classes.keys():
    classes_labels.append(key)
print(classes_labels)

[2, 4, 3, 6, 5, 1, 0]
```

In [51]:

```
print(x.shape,y.shape)
# To overcome class imbalance
oversample = RandomOverSampler()
x,y = oversample.fit_resample(x,y)
print(x.shape,y.shape)

(10015, 2352) (10015,)
(46935, 2352) (46935,)
```

In [52]:

```
# reshaping the data so that it can be taken by convolution neural network(without distur
bing the no. of samples)
x = np.array(x).reshape(-1,28,28,3)
print('Shape of X :',x.shape)
print('Shape of y :',y.shape)

Shape of X : (46935, 28, 28, 3)
Shape of y : (46935,)
```

In [53]:

```
# Splitting Data
X_train, X_test, Y_train, Y_test = train_test_split(x,y, test_size=0.2, random_state=1)
print(X_train.shape,Y_train.shape)
print(X_test.shape , Y_test.shape)

(37548, 28, 28, 3) (37548,)
(9387, 28, 28, 3) (9387,)
```

In [54]:

```
model_CNN = Sequential()
model_CNN.add(Conv2D(16, kernel_size = (3,3), input_shape = (28, 28, 3), activation = 'r
elu', padding = 'same'))
model_CNN.add(MaxPool2D(pool_size = (2,2)))

model_CNN.add(Conv2D(32, kernel_size = (3,3), activation = 'relu', padding = 'same'))
model_CNN.add(MaxPool2D(pool_size = (2,2), padding = 'same'))

model_CNN.add(Conv2D(64, kernel_size = (3,3), activation = 'relu', padding = 'same'))
model_CNN.add(MaxPool2D(pool_size = (2,2), padding = 'same'))
```


Epoch 1/50
470/470 [=====] - 5s 7ms/step - loss: 1.5914 - accuracy: 0.4616
- val_loss: 0.9658 - val_accuracy: 0.6385 - lr: 0.0010

Epoch 2/50
470/470 [=====] - 3s 5ms/step - loss: 0.8243 - accuracy: 0.6878
- val_loss: 0.6166 - val_accuracy: 0.7682 - lr: 0.0010

Epoch 3/50
470/470 [=====] - 3s 5ms/step - loss: 0.5668 - accuracy: 0.7884
- val_loss: 0.4597 - val_accuracy: 0.8280 - lr: 0.0010

Epoch 4/50
470/470 [=====] - 3s 6ms/step - loss: 0.4142 - accuracy: 0.8456
- val_loss: 0.3404 - val_accuracy: 0.8774 - lr: 0.0010

Epoch 5/50
470/470 [=====] - 3s 6ms/step - loss: 0.3249 - accuracy: 0.8798
- val_loss: 0.3303 - val_accuracy: 0.8820 - lr: 0.0010

Epoch 6/50
470/470 [=====] - 3s 5ms/step - loss: 0.2619 - accuracy: 0.9050
- val_loss: 0.2417 - val_accuracy: 0.9180 - lr: 0.0010

Epoch 7/50
470/470 [=====] - 3s 5ms/step - loss: 0.2310 - accuracy: 0.9152
- val_loss: 0.2317 - val_accuracy: 0.9160 - lr: 0.0010

Epoch 8/50
470/470 [=====] - 3s 6ms/step - loss: 0.2074 - accuracy: 0.9270
- val_loss: 0.3903 - val_accuracy: 0.8561 - lr: 0.0010

Epoch 9/50
470/470 [=====] - 3s 5ms/step - loss: 0.1727 - accuracy: 0.9364
- val_loss: 0.1951 - val_accuracy: 0.9316 - lr: 0.0010

Epoch 10/50
470/470 [=====] - 3s 5ms/step - loss: 0.1683 - accuracy: 0.9406
- val_loss: 0.2472 - val_accuracy: 0.9186 - lr: 0.0010

Epoch 11/50
470/470 [=====] - 3s 5ms/step - loss: 0.1462 - accuracy: 0.9481
- val_loss: 0.1863 - val_accuracy: 0.9413 - lr: 0.0010

Epoch 12/50
470/470 [=====] - 3s 6ms/step - loss: 0.1456 - accuracy: 0.9478
- val_loss: 0.1614 - val_accuracy: 0.9490 - lr: 0.0010

Epoch 13/50
470/470 [=====] - 3s 6ms/step - loss: 0.1176 - accuracy: 0.9580
- val_loss: 0.1580 - val_accuracy: 0.9499 - lr: 0.0010

Epoch 14/50
470/470 [=====] - 3s 5ms/step - loss: 0.0916 - accuracy: 0.9674
- val_loss: 0.3800 - val_accuracy: 0.8885 - lr: 0.0010

Epoch 15/50
470/470 [=====] - 3s 5ms/step - loss: 0.1481 - accuracy: 0.9474
- val_loss: 0.2174 - val_accuracy: 0.9329 - lr: 0.0010

Epoch 16/50
470/470 [=====] - 3s 6ms/step - loss: 0.0966 - accuracy: 0.9657
- val_loss: 0.1117 - val_accuracy: 0.9670 - lr: 0.0010

Epoch 17/50
470/470 [=====] - 3s 6ms/step - loss: 0.0995 - accuracy: 0.9654
- val_loss: 0.1996 - val_accuracy: 0.9459 - lr: 0.0010

Epoch 18/50
470/470 [=====] - 3s 5ms/step - loss: 0.0994 - accuracy: 0.9662
- val_loss: 0.1425 - val_accuracy: 0.9594 - lr: 0.0010

Epoch 19/50
463/470 [=====>.] - ETA: 0s - loss: 0.0772 - accuracy: 0.9736
Epoch 19: ReduceLROnPlateau reducing learning rate to 0.00010000000474974513.
470/470 [=====] - 3s 6ms/step - loss: 0.0785 - accuracy: 0.9732
- val_loss: 0.1910 - val_accuracy: 0.9454 - lr: 0.0010

Epoch 20/50
470/470 [=====] - 3s 7ms/step - loss: 0.0211 - accuracy: 0.9939
- val_loss: 0.0845 - val_accuracy: 0.9786 - lr: 1.0000e-04

Epoch 21/50
470/470 [=====] - 3s 5ms/step - loss: 0.0082 - accuracy: 0.9982
- val_loss: 0.0812 - val_accuracy: 0.9803 - lr: 1.0000e-04

Epoch 22/50
470/470 [=====] - 3s 5ms/step - loss: 0.0054 - accuracy: 0.9989
- val_loss: 0.0800 - val_accuracy: 0.9811 - lr: 1.0000e-04

Epoch 23/50
470/470 [=====] - 3s 5ms/step - loss: 0.0038 - accuracy: 0.9994
- val_loss: 0.0810 - val_accuracy: 0.9815 - lr: 1.0000e-04

```

Epoch 24/50
470/470 [=====] - 3s 6ms/step - loss: 0.0027 - accuracy: 0.9998
- val_loss: 0.0850 - val_accuracy: 0.9811 - lr: 1.0000e-04
Epoch 25/50
463/470 [=====>.] - ETA: 0s - loss: 0.0020 - accuracy: 0.9999
Epoch 25: ReduceLROnPlateau reducing learning rate to 1.0000000474974514e-05.
470/470 [=====] - 3s 6ms/step - loss: 0.0020 - accuracy: 0.9999
- val_loss: 0.0843 - val_accuracy: 0.9820 - lr: 1.0000e-04
Epoch 26/50
470/470 [=====] - 3s 5ms/step - loss: 0.0014 - accuracy: 1.0000
- val_loss: 0.0839 - val_accuracy: 0.9820 - lr: 1.0000e-05
Epoch 27/50
470/470 [=====] - 3s 6ms/step - loss: 0.0014 - accuracy: 1.0000
- val_loss: 0.0861 - val_accuracy: 0.9818 - lr: 1.0000e-05
Epoch 28/50
463/470 [=====>.] - ETA: 0s - loss: 0.0013 - accuracy: 1.0000
Epoch 28: ReduceLROnPlateau reducing learning rate to 1.0000000656873453e-06.
470/470 [=====] - 3s 7ms/step - loss: 0.0013 - accuracy: 1.0000
- val_loss: 0.0846 - val_accuracy: 0.9820 - lr: 1.0000e-05
Epoch 29/50
470/470 [=====] - 3s 5ms/step - loss: 0.0012 - accuracy: 1.0000
- val_loss: 0.0851 - val_accuracy: 0.9820 - lr: 1.0000e-06
Epoch 30/50
470/470 [=====] - 3s 5ms/step - loss: 0.0012 - accuracy: 1.0000
- val_loss: 0.0854 - val_accuracy: 0.9819 - lr: 1.0000e-06
Epoch 31/50
468/470 [=====>.] - ETA: 0s - loss: 0.0012 - accuracy: 1.0000
Epoch 31: ReduceLROnPlateau reducing learning rate to 1.0000001111620805e-07.
470/470 [=====] - 3s 6ms/step - loss: 0.0012 - accuracy: 1.0000
- val_loss: 0.0856 - val_accuracy: 0.9819 - lr: 1.0000e-06
Epoch 32/50
470/470 [=====] - 3s 6ms/step - loss: 0.0012 - accuracy: 1.0000
- val_loss: 0.0856 - val_accuracy: 0.9819 - lr: 1.0000e-07
Epoch 32: early stopping

```

In [56]:

```

results = model_CNN.evaluate(X_test , Y_test, verbose=0)

print("    Test Loss: {:.5f}".format(results[0]))
print("Test Accuracy: {:.2f}%".format(results[1] * 100))

```

```

    Test Loss: 0.12440
Test Accuracy: 97.79%

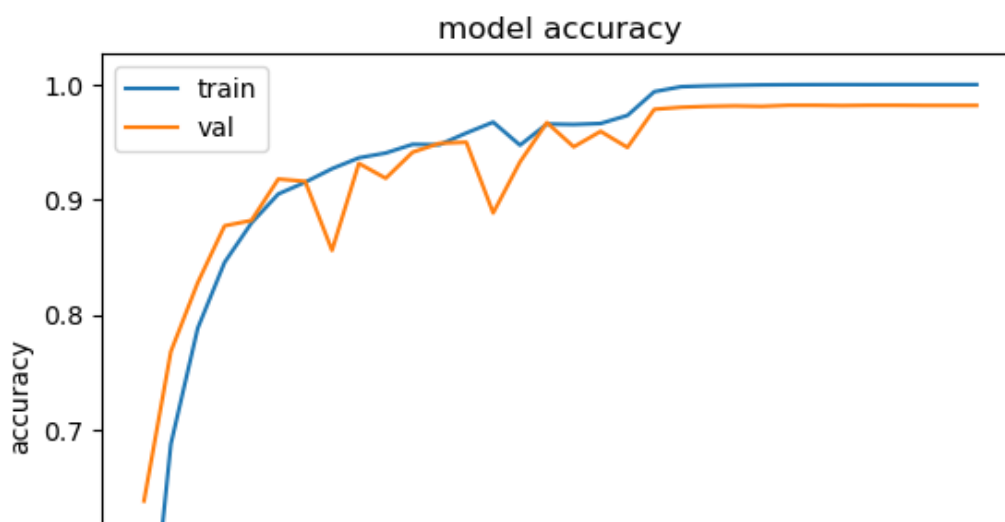
```

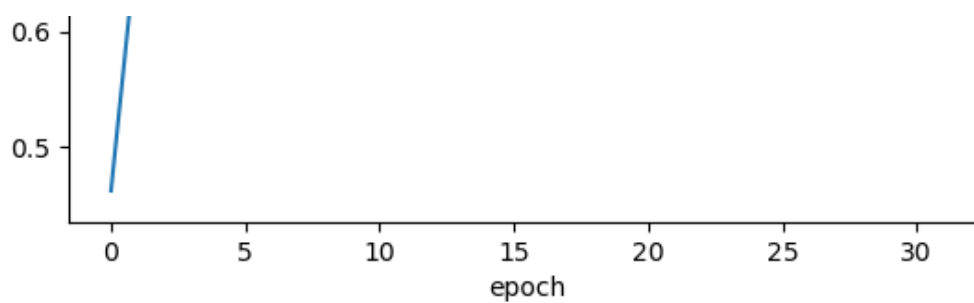
In [57]:

```

plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'val'], loc='upper left')
plt.show()

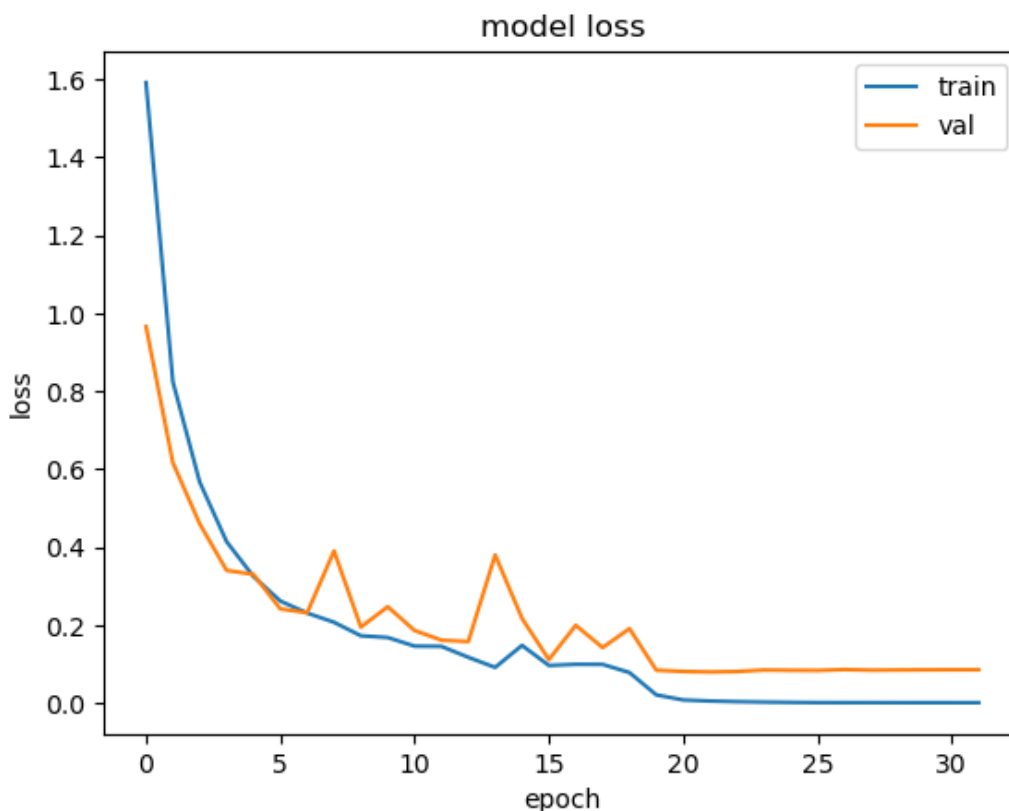
```





In [58]:

```
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'val'], loc='upper right')
plt.show()
```



In [59]:

```
from sklearn.metrics import confusion_matrix , classification_report

y_true_CNN = list(Y_test)
y_pred_CNN = model_CNN.predict(X_test)
y_pred_CNN = list(map(lambda x: np.argmax(x), y_pred_CNN))
print('Y Actual Values : ' , y_true_CNN[0:10])
print('Y Predicted Values : ' , y_pred_CNN[0:10])
```

```
294/294 [=====] - 1s 2ms/step
Y Actual Values : [5, 1, 4, 0, 5, 0, 2, 0, 3, 2]
Y Predicted Values : [5, 1, 4, 0, 5, 0, 2, 0, 3, 2]
```

In [60]:

```
cm_CNN = confusion_matrix(y_true_CNN,y_pred_CNN,labels=classes_labels)
print(confusion_matrix(y_true_CNN,y_pred_CNN,labels=classes_labels))
sns.heatmap(cm_CNN, annot = True, fmt='')
```

```
[[1247   12    0    1    0    1    1]
 [  61 1202    4   92    4   10    1]
 [    0    0 1351    0    0    0    0]]
```

```
#predicting
y_pred_CNN = model_CNN.predict(X_test)
```



```
target_names = [f"{classes[i]}" for i in range(7)]
print(len(Y_test) , " ", len(y_pred_CNN))
y_pred_CNN = list(map(lambda x: np.argmax(x), y_pred_CNN))
print(classification_report(Y_test , y_pred_CNN, target_names=target_names))
```

294/294 [=====] - 1s 2ms/step

	precision	recall	f1-score	support
akiec	1.00	1.00	1.00	1359
bcc	0.99	1.00	1.00	1318
bkl	0.95	0.99	0.97	1262
df	1.00	1.00	1.00	1351
nv	0.98	0.87	0.92	1374
vasc	1.00	1.00	1.00	1358
mel	0.94	0.99	0.96	1365
accuracy			0.98	9387
macro avg	0.98	0.98	0.98	9387
weighted avg	0.98	0.98	0.98	9387

In [63]:

```
# Layers definitions
from keras import backend as K
for l in range(len(model_CNN.layers)):
    print(l, model_CNN.layers[l])
```

```
0 <keras.layers.convolutional.conv2d.Conv2D object at 0x7f3f0d63be90>
1 <keras.layers.pooling.max_pooling2d.MaxPooling2D object at 0x7f3f0d63bad0>
2 <keras.layers.convolutional.conv2d.Conv2D object at 0x7f3f0d65db90>
3 <keras.layers.pooling.max_pooling2d.MaxPooling2D object at 0x7f3f0d622c50>
4 <keras.layers.convolutional.conv2d.Conv2D object at 0x7f3f0d76ab50>
5 <keras.layers.pooling.max_pooling2d.MaxPooling2D object at 0x7f3f0d622a50>
6 <keras.layers.convolutional.conv2d.Conv2D object at 0x7f3f0d64e450>
7 <keras.layers.pooling.max_pooling2d.MaxPooling2D object at 0x7f3f0d651090>
8 <keras.layers.resizing.flatten.Flatten object at 0x7f3f0f419190>
9 <keras.layers.core.dense.Dense object at 0x7f3dffffbec90>
10 <keras.layers.core.dense.Dense object at 0x7f3f0d4cbe50>
11 <keras.layers.core.activation.Activation object at 0x7f3f0d66d410>
12 <keras.layers.core.dense.Dense object at 0x7f3f0d51cc10>
13 <keras.layers.core.activation.Activation object at 0x7f3f0d49b450>
```

In [64]:

```
import os
os.environ["KERAS_BACKEND"] = "tensorflow"
kerasBKED = os.environ["KERAS_BACKEND"]
print(kerasBKED)
```

tensorflow

In [79]:

```
import tensorflow as tf
# feature_extractor = tf.keras.Model(inputs=model_CNN.input,
#                                     outputs=model_CNN.get_layer(-2).output)
# output_layers_model =tf.keras.Model(inputs=model_CNN.input, outputs=model_CNN.output)
cnn_layer_output = model_CNN.layers[-2].output
cnn_model_features = tf.keras.Model(inputs=model_CNN.input, outputs=cnn_layer_output)
```

In [81]:

```
# Extract features from input data using the CNN model
X_train_cnn = cnn_model_features.predict(X_train)
X_test_cnn = cnn_model_features.predict(X_test)
```

1174/1174 [=====] - 2s 2ms/step
294/294 [=====] - 1s 2ms/step

In [82]:

```
import numpy as np
from sklearn.svm import SVC
from sklearn.model_selection import GridSearchCV

parameters = {'kernel': ['rbf'],
              'C': [1, 10, 100, 1000],
              'gamma': [1e-3, 1e-4]}
clf = GridSearchCV(SVC(), parameters)
clf.fit(X_train_cnn, Y_train)
# Evaluate the combined CNN-SVM model on a test dataset
svm_accuracy = clf.score(X_test_cnn, Y_test)
print('SVM Accuracy:', svm_accuracy*100)
```

SVM Accuracy: 98.08245445829338

In [83]:

```
svm_accuracy = clf.score(X_test_cnn, Y_test)
print('SVM Accuracy:', svm_accuracy*100)
svmclf = clf.best_estimator_
print(svmclf)
svmclf.fit(X_train_cnn, Y_train)
```

SVM Accuracy: 98.08245445829338
SVC(C=1, gamma=0.001)

Out[83]:

SVC(C=1, gamma=0.001)

In [84]:

```
y_testSVM = svmclf.predict(X_test_cnn)
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score

print(classification_report(Y_test, y_testSVM))
print("Accuracy: {0}".format(accuracy_score(Y_test, y_testSVM)*100))
```

	precision	recall	f1-score	support
0	1.00	1.00	1.00	1359
1	0.99	1.00	1.00	1318
2	0.96	0.99	0.97	1262
3	1.00	1.00	1.00	1351
4	0.98	0.89	0.93	1374
5	1.00	1.00	1.00	1358
6	0.95	0.99	0.97	1365
accuracy			0.98	9387
macro avg	0.98	0.98	0.98	9387
weighted avg	0.98	0.98	0.98	9387

Accuracy: 98.08245445829338

In [85]:

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV

parameters = {"max_depth": [3],
              "max_features": [1],
              "min_samples_split": [3],
              "min_samples_leaf": [3],
              "bootstrap": [True],
              "criterion": ["entropy"],
              "n_estimators": [10]}
rclf = RandomForestClassifier()
rgclf = GridSearchCV(rclf, param_grid=parameters)
rgclf.fit(X_test_cnn, Y_test)
RFC_accuracy = clf.score(X_test_cnn, Y_test)
print('Random Forest Classifier Accuracy:', RFC_accuracy*100)
```

Random Forest Classifier Accuracy: 98.08245445829338

In [86]:

```
RFclf = rgclf.best_estimator_  
RFclf.fit(X_test_cnn, Y_test)  
print(RFclf)  
y_testRFC = RFclf.predict(X_test_cnn)  
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score  
  
print(classification_report(Y_test, y_testRFC))  
print("Accuracy: {0}".format(accuracy_score(Y_test, y_testRFC)*100))
```

```
RandomForestClassifier(criterion='entropy', max_depth=3, max_features=1,  
                        min_samples_leaf=3, min_samples_split=3,  
                        n_estimators=10)
```

	precision	recall	f1-score	support
0	1.00	1.00	1.00	1359
1	0.97	1.00	0.99	1318
2	0.84	0.98	0.91	1262
3	1.00	1.00	1.00	1351
4	0.98	0.58	0.73	1374
5	0.99	1.00	0.99	1358
6	0.81	0.99	0.89	1365
accuracy			0.93	9387
macro avg	0.94	0.94	0.93	9387
weighted avg	0.94	0.93	0.93	9387

Accuracy: 93.42708000426121

In [88]:

```
from sklearn.neighbors import KNeighborsClassifier  
from sklearn.model_selection import GridSearchCV  
  
parameters = {"n_neighbors": [1, 5, 10, 30],  
              "weights": ['uniform', 'distance'],  
              "metric": ['minkowski', 'euclidean', 'manhattan'],  
              "algorithm": ['auto', 'ball_tree', 'kd_tree', 'brute']}  
  
kclf = KNeighborsClassifier()  
kgclf = GridSearchCV(kclf, param_grid=parameters)  
kgclf.fit(X_test_cnn, Y_test)  
KNN_accuracy = kgclf.score(X_test_cnn, Y_test)  
print('KNN Classifier Accuracy:', KNN_accuracy*100)
```

KNN Classifier Accuracy: 100.0

In [89]:

```
y_testKNN = kgclf.predict(X_test_cnn)  
KNNclf = kgclf.best_estimator_  
print(KNNclf)  
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score  
  
print(classification_report(Y_test, y_testKNN))  
print("Accuracy: {0}".format(accuracy_score(Y_test, y_testKNN)*100))
```

```
KNeighborsClassifier(metric='manhattan', n_neighbors=1)
```

	precision	recall	f1-score	support
0	1.00	1.00	1.00	1359
1	1.00	1.00	1.00	1318
2	1.00	1.00	1.00	1262
3	1.00	1.00	1.00	1351
4	1.00	1.00	1.00	1374
5	1.00	1.00	1.00	1358
6	1.00	1.00	1.00	1365
accuracy			1.00	9387
macro avg	1.00	1.00	1.00	9387

macro avg	1.00	1.00	1.00	9387
weighted avg	1.00	1.00	1.00	

Accuracy: 100.0

In [90]:

```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score
```

```
# Create a logistic regression object
lr = LogisticRegression()
```

```
# Define the hyperparameter grid to search over
param_grid = {'C': [0.001, 0.01, 0.1, 1, 10, 100], 'penalty': ['l1', 'l2']}
```

```
# Perform grid search with 5-fold cross-validation
grid_search_LR = GridSearchCV(lr, param_grid, cv=5)
grid_search_LR.fit(X_test_cnn, Y_test)
```

```
# Print the best hyperparameters and the corresponding accuracy score
print("Best hyperparameters: ", grid_search_LR.best_params_)
y_testKNN = grid_search_LR.predict(X_test_cnn)
```

```
print(classification_report(Y_test, y_testKNN))
print("Accuracy: {0}".format(accuracy_score(Y_test, y_testKNN)*100))
```

```
/opt/conda/lib/python3.7/site-packages/sklearn/linear_model/_logistic.py:818: Convergence
Warning: 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

extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,

```
/opt/conda/lib/python3.7/site-packages/sklearn/linear_model/_logistic.py:818: Convergence
Warning: 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

extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,

```
/opt/conda/lib/python3.7/site-packages/sklearn/linear_model/_logistic.py:818: Convergence
Warning: 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

extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,

```
/opt/conda/lib/python3.7/site-packages/sklearn/linear_model/_logistic.py:818: Convergence
Warning: 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

extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,

```
/opt/conda/lib/python3.7/site-packages/sklearn/linear_model/_logistic.py:818: Convergence
Warning: 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>

<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
extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
/opt/conda/lib/python3.7/site-packages/sklearn/linear_model/_logistic.py:818: Convergence
Warning: 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
extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
/opt/conda/lib/python3.7/site-packages/sklearn/linear_model/_logistic.py:818: Convergence
Warning: 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
extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
/opt/conda/lib/python3.7/site-packages/sklearn/linear_model/_logistic.py:818: Convergence
Warning: 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
extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
/opt/conda/lib/python3.7/site-packages/sklearn/linear_model/_logistic.py:818: Convergence
Warning: 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
extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
/opt/conda/lib/python3.7/site-packages/sklearn/linear_model/_logistic.py:818: Convergence
Warning: 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
extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
/opt/conda/lib/python3.7/site-packages/sklearn/linear_model/_logistic.py:818: Convergence
Warning: 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
extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
/opt/conda/lib/python3.7/site-packages/sklearn/linear_model/_logistic.py:818: Convergence
Warning: 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
extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
/opt/conda/lib/python3.7/site-packages/sklearn/linear_model/_logistic.py:818: Convergence
Warning: 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>

<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
extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
/opt/conda/lib/python3.7/site-packages/sklearn/linear_model/_logistic.py:818: Convergence
Warning: 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
extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
/opt/conda/lib/python3.7/site-packages/sklearn/linear_model/_logistic.py:818: Convergence
Warning: 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
extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
/opt/conda/lib/python3.7/site-packages/sklearn/linear_model/_logistic.py:818: Convergence
Warning: 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
extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
/opt/conda/lib/python3.7/site-packages/sklearn/linear_model/_logistic.py:818: Convergence
Warning: 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
extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
/opt/conda/lib/python3.7/site-packages/sklearn/linear_model/_logistic.py:818: Convergence
Warning: 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
extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
/opt/conda/lib/python3.7/site-packages/sklearn/linear_model/_logistic.py:818: Convergence
Warning: 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
extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
/opt/conda/lib/python3.7/site-packages/sklearn/linear_model/_logistic.py:818: Convergence
Warning: 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
extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
/opt/conda/lib/python3.7/site-packages/sklearn/linear_model/_logistic.py:818: Convergence
Warning: 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>

<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
extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
/opt/conda/lib/python3.7/site-packages/sklearn/linear_model/_logistic.py:818: Convergence
Warning: 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
extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
/opt/conda/lib/python3.7/site-packages/sklearn/linear_model/_logistic.py:818: Convergence
Warning: 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
extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
/opt/conda/lib/python3.7/site-packages/sklearn/linear_model/_logistic.py:818: Convergence
Warning: 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
extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
/opt/conda/lib/python3.7/site-packages/sklearn/linear_model/_logistic.py:818: Convergence
Warning: 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
extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
/opt/conda/lib/python3.7/site-packages/sklearn/linear_model/_logistic.py:818: Convergence
Warning: 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
extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
/opt/conda/lib/python3.7/site-packages/sklearn/linear_model/_logistic.py:818: Convergence
Warning: 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
extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
/opt/conda/lib/python3.7/site-packages/sklearn/linear_model/_logistic.py:818: Convergence
Warning: 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
extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
/opt/conda/lib/python3.7/site-packages/sklearn/linear_model/_logistic.py:818: Convergence
Warning: 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>

```
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
extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
/opt/conda/lib/python3.7/site-packages/sklearn/linear_model/_logistic.py:818: Convergence
Warning: 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
`extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,`
`/opt/conda/lib/python3.7/site-packages/sklearn/model_selection/_validation.py:372: FitFailedWarning:`
30 fits failed out of a total of 60.
The score on these train-test partitions for these parameters will be set to nan.
If these failures are not expected, you can try to debug them by setting `error_score='raise'`.

Below are more details about the failures:

```
-----
30 fits failed with the following error:
Traceback (most recent call last):
  File "/opt/conda/lib/python3.7/site-packages/sklearn/model_selection/_validation.py", line 680, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "/opt/conda/lib/python3.7/site-packages/sklearn/linear_model/_logistic.py", line 1461, in fit
    solver = _check_solver(self.solver, self.penalty, self.dual)
  File "/opt/conda/lib/python3.7/site-packages/sklearn/linear_model/_logistic.py", line 449, in _check_solver
    % (solver, penalty)
ValueError: Solver lbfgs supports only 'l2' or 'none' penalties, got l1 penalty.

warnings.warn(some_fits_failed_message, FitFailedWarning)
/opt/conda/lib/python3.7/site-packages/sklearn/model_selection/_search.py:972: UserWarning: One or more of the test scores are non-finite: [
88945      nan 0.9827417      nan 0.9774149      nan 0.981
      nan 0.9828482      nan 0.9828482      nan 0.98274165]
category=UserWarning,
```

Best hyperparameters: {'C': 1, 'penalty': 'l2'}

	precision	recall	f1-score	support
0	1.00	1.00	1.00	1359
1	1.00	1.00	1.00	1318
2	0.96	0.99	0.97	1262
3	1.00	1.00	1.00	1351
4	0.97	0.91	0.94	1374
5	1.00	1.00	1.00	1358
6	0.96	0.98	0.97	1365
accuracy			0.98	9387
macro avg	0.98	0.98	0.98	9387
weighted avg	0.98	0.98	0.98	9387

Accuracy: 98.30616810482582

```
/opt/conda/lib/python3.7/site-packages/sklearn/linear_model/_logistic.py:818: Convergence
Warning: 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
extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
```

In []:

In []:

In []:

In []: