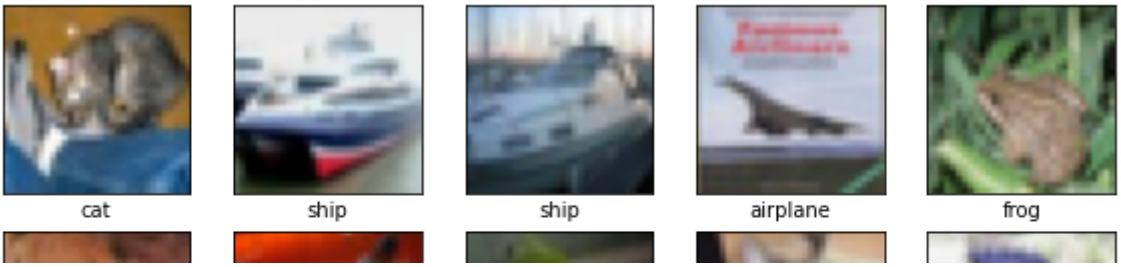
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import sklearn.metrics as sm


from tensorflow import keras
from keras.models import Sequential
from keras.optimizers import Adam
from keras.layers import Dense, Conv2D, Dropout, MaxPooling2D, Flatten


from keras.datasets import cifar10
(x_train,y_train), (x_test,ytest) = cifar10.load_data()
```

```
x_train,y_train
```

```
     (array([[[[ 59,  62,  63],
               [ 43,  46,  45],
               [ 50,  48,  43],
               ...,
               [158, 132, 108],
               [152, 125, 102],
               [148, 124, 103]],

              [[ 16,  20,  20],
               [  0,   0,   0],
               [ 18,   8,   0],
               ...,
               [123,  88,  55],
               [119,  83,  50],
               [122,  87,  57]],

              [[ 25,  24,  21],
               [ 16,   7,   0],
               [ 49,  27,   8],
               ...,
               [118,  84,  50],
               [120,  84,  50],
               [109,  73,  42]],

              ...,

              [[208, 170,  96],
               [201, 153,  34],
               [198, 161,  26],
               ...,
               [160, 133,  70],
               [ 56,  31,   7],
               [ 53,  34,  20]],

              [[180, 139,  96],
               [173, 123,  42],
```

```
                 [186, 144,  30],
                 ...,
                 [184, 148,  94],
                 [ 97,  62,  34],
                 [ 83,  53,  34]],

                [[177, 144, 116],
                 [168, 129,  94],
                 [179, 142,  87],
                 ...,
                 [216, 184, 140],
                 [151, 118,  84],
                 [123,  92,  72]]],


               [[[154, 177, 187],
                 [126, 137, 136],
                 [105, 104,  95],
                 ...,
                 [ 91,  95,  71],
                 [ 87,  90,  71],
                 [ 79,  81,  70]],
```

```
plt.figure(figsize=(10,10))
for i in range(25):
    plt.subplot(5,5,i+1)
    classNames = ['airplane', 'automobile', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse',
    plt.xticks([])
    plt.yticks([])
    plt.imshow(x_test[i])
    plt.xlabel(classNames[ytest[i][0]])
plt.show()
```




```
# Converting input image data into float
x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
```


```
x_train = (x_train-x_train.mean())/x_train.max()
x_test = (x_test-x_test.mean())/x_test.max()
y_train = keras.utils.to_categorical(y_train,10)
y_test = keras.utils.to_categorical(ytest,10)
```

```
# Difference between ytest and y_test
print(ytest)
print(y_test)
```

```
     [[3]
      [8]
      [8]
      ...
      [5]
      [1]
      [7]]
     [[0. 0. 0. ... 0. 0. 0.]
      [0. 0. 0. ... 0. 1. 0.]
      [0. 0. 0. ... 0. 1. 0.]
      ...
      [0. 0. 0. ... 0. 0. 0.]
      [0. 1. 0. ... 0. 0. 0.]
      [0. 0. 0. ... 1. 0. 0.]]
```

```
List = [x_train.shape,x_test.shape,y_train.shape,y_test.shape]
print(List)
```

```
     [(50000, 32, 32, 3), (10000, 32, 32, 3), (50000, 10), (10000, 10)]
```

```
model = Sequential()

model.add(Conv2D(32, (3, 3), padding='same',activation='relu', input_shape=(32,32,3)))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(64, (3, 3), padding='same',activation='relu', input_shape=(32,32,3)))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Flatten())

model.add(Dense(256, activation='relu'))
```

```
model.add(Dropout(0.5))

model.add(Dense(10, activation='softmax'))

model.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_4 (Conv2D)           (None, 32, 32, 32)        896

 max_pooling2d_4 (MaxPooling  (None, 16, 16, 32)       0
 2D)

 conv2d_5 (Conv2D)           (None, 16, 16, 64)        18496

 max_pooling2d_5 (MaxPooling  (None, 8, 8, 64)         0
 2D)

 dropout_4 (Dropout)         (None, 8, 8, 64)          0

 flatten_2 (Flatten)         (None, 4096)              0

 dense_4 (Dense)             (None, 256)               1048832

 dropout_5 (Dropout)         (None, 256)               0

 dense_5 (Dense)             (None, 10)                2570

=================================================================
Total params: 1,070,794
Trainable params: 1,070,794
Non-trainable params: 0
_________________________________________________________________
```

```
model.compile(loss='categorical_crossentropy', optimizer=Adam(lr=1.0e-4), metrics = ['accu
```

```
model.fit(x_train, y_train, batch_size=256, epochs=200)
```

```
196/196 [==============================] - 75s 382ms/step - loss: 0.6281 - accurac
Epoch 73/200
196/196 [==============================] - 76s 385ms/step - loss: 0.6286 - accurac
Epoch 74/200
196/196 [==============================] - 75s 385ms/step - loss: 0.6207 - accurac
Epoch 75/200
196/196 [==============================] - 75s 384ms/step - loss: 0.6182 - accurac
Epoch 76/200
196/196 [==============================] - 75s 381ms/step - loss: 0.6125 - accurac
Epoch 77/200
196/196 [==============================] - 74s 377ms/step - loss: 0.6054 - accurac
Epoch 78/200
196/196 [==============================] - 75s 381ms/step - loss: 0.6003 - accurac
Epoch 79/200
196/196 [==============================] - 75s 381ms/step - loss: 0.5978 - accurac
Epoch 80/200
196/196 [==============================] - 75s 380ms/step - loss: 0.5924 - accurac
Epoch 81/200
196/196 [==============================] - 74s 377ms/step - loss: 0.5880 - accurac
```

```
Epoch 82/200
196/196 [==============================] - 74s 380ms/step - loss: 0.5829 - accurac
Epoch 83/200
196/196 [==============================] - 75s 383ms/step - loss: 0.5776 - accurac
Epoch 84/200
196/196 [==============================] - 75s 384ms/step - loss: 0.5782 - accurac
Epoch 85/200
196/196 [==============================] - 76s 385ms/step - loss: 0.5665 - accurac
Epoch 86/200
196/196 [==============================] - 75s 381ms/step - loss: 0.5675 - accurac
Epoch 87/200
196/196 [==============================] - 75s 385ms/step - loss: 0.5614 - accurac
Epoch 88/200
196/196 [==============================] - 76s 386ms/step - loss: 0.5573 - accurac
Epoch 89/200
196/196 [==============================] - 75s 385ms/step - loss: 0.5531 - accurac
Epoch 90/200
196/196 [==============================] - 75s 385ms/step - loss: 0.5439 - accurac
Epoch 91/200
196/196 [==============================] - 75s 382ms/step - loss: 0.5436 - accurac
Epoch 92/200
196/196 [==============================] - 76s 386ms/step - loss: 0.5367 - accurac
Epoch 93/200
196/196 [==============================] - 76s 385ms/step - loss: 0.5345 - accurac
Epoch 94/200
196/196 [==============================] - 75s 385ms/step - loss: 0.5329 - accurac
Epoch 95/200
196/196 [==============================] - 75s 385ms/step - loss: 0.5216 - accurac
Epoch 96/200
196/196 [==============================] - 75s 381ms/step - loss: 0.5188 - accurac
Epoch 97/200
196/196 [==============================] - 75s 385ms/step - loss: 0.5186 - accurac
Epoch 98/200
196/196 [==============================] - 75s 384ms/step - loss: 0.5149 - accurac
Epoch 99/200
196/196 [==============================] - 75s 383ms/step - loss: 0.5132 - accurac
Epoch 100/200
  9/196 [>.............................] - ETA: 1:11 - loss: 0.4929 - accuracy: 0.
```

```
yhat = model.predict_classes(x_test)
print(sm.classification_report(ytest,yhat))
print(f'Accuracy of test data: {sm.accuracy_score(ytest,yhat)*100}%')


cm = sm.confusion_matrix(ytest,yhat)
plt.clf()
plt.imshow(cm,cmap=plt.cm.autumn_r)
plt.title('Confusion Matrix - Test Data')
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
tick_marks = np.arange(len(classNames))
plt.xticks(tick_marks,classNames,rotation=90)
plt.yticks(tick_marks,classNames)
for i in range(len(classNames)):
  for j in range(len(classNames)):
    plt.text(i,j,cm[i][j])
plt.show()
```

45m 54s completed at 3:50 AM