B.SABARI 18MIS0416
K VIJAY 18MIS0172

Source code:

```
import os import tensorflow as
tf import pandas from pandas
import DataFrame from pandas
import DataFrame
Categories_Dictionary={'001': 'Danaus_plexippus','002': 'Heliconius_ch
aritonius',
'003': 'Heliconius_erato','004': 'Junonia_coenia','005': 'Lycaena_phlaeas',
'006': 'Nymphalis_antiopa','007': 'Papilio_cresphontes','008': 'Pieris_rapa
e',
'009': 'Vanessa_atalanta','010': 'Vanessa_cardui'}
Categories = []
Filenames = os.listdir('/content/drive/My Drive/leedsbutterfly/images/')
for Filename in Filenames:
       Category = Filename.split(".")[0]
Categories.append(Categories_Dictionary[Category[0:3]])
DF=DataFrame({'Filename':Filenames,'Category':Categories})
DF.head(5)

os.chdir('/content/drive/My Drive/') if
os.path.exists('NewDataset'):
! rm -r NewDataset
os.makedirs('NewDataset')

os.chdir('/content/drive/My Drive/NewDataset')
Directories=('train','valid','test') for i in
Directories:   os.makedirs(i) import shutil for i
in range(len(X_train)):
  Source='/content/drive/My Drive/leedsbutterfly/images/'+X_train.iloc[i
]
  Destination='/content/drive/My Drive/NewDataset/train/'+Y_train.iloc[
i]+'/'+X_train.iloc[i]
  print('Processing image: ',i+1,'/',len(X_train))
shutil.copy(Source,Destination)

X_valid , x_test , Y_valid , y_test = train_test_split(x_test, y_test,test_si
ze = 0.1)
print('Size of Validation set = ',len(X_valid)) print('Size
of Test set = ',len(x_test))

for i in range(len(X_valid)):
  Source='/content/drive/My Drive/leedsbutterfly/images/'+X_valid.iloc[ i]
  Destination='/content/drive/My Drive/NewDataset/valid/'+Y_valid.iloc
[i]+'/'+X_valid.iloc[i]    print('Processing
image: ',i+1,'/',len(X_valid))
shutil.copy(Source,Destination)

for i in range(len(x_test)):
  Source='/content/drive/My Drive/leedsbutterfly/images/'+x_test.iloc[i]
  Destination='/content/drive/My Drive/NewDataset/test/'+y_test.iloc[i]+
'/'+x_test.iloc[i]
  print('Processing image: ',i+1,'/',len(x_test))
  shutil.copy(Source,Destination)
```

Modle training:

```
model = Sequential()
model.add(Conv2D(32, (5,5), activation = 'relu', input_shape=(128,128,
3)))
model.add(MaxPooling2D((2,2)))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D((2,2)))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D((2,2)))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D((2,2)))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D((2,2)))
model.add(Flatten()) model.add(Dropout(0.25))
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.3)) model.add(Dense(10,
activation='softmax'))
model.compile(loss='categorical_crossentropy',
optimizer=RMSprop(lr=0.0001,decay=1e-6),
            metrics=['accuracy','mae','mse']) model.summary()
```

Image Training:

```
IDG_Train=ImageDataGenerator(rotation_range=40,width_shift_range=
0.2,         height_shift_range=0.2,
rescale=1./255,shear_range=0.2,zoom_ran
ge=0.2,horizontal_flip=True,fill_mode='nearest')
IDG_Valid=ImageDataGenerator(rescale=1/255)
Train_Data=IDG_Train.flow_from_directory(Train_Path,target_size=(1
28,128),batch_size=Batch_Size_Train,class_mode='categorical')
```

```
Valid_Data=IDG_Valid.flow_from_directory(Valid_Path,target_size=(128,128),batch_size=Batch_Size_Valid,class_mode='categorical')
```

Fitting model

```
model.fit(Train_Data,steps_per_epoch=Steps_Per_Epoch_Train,epochs=4,validation_data=Valid_Data,validation_steps=Steps_Per_Epoch_Valid)
```

filed to which images belong

```
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
Label_Encoding=LabelEncoder().fit_transform(Categories).reshape(1,1)
One_Hot_Encoding=OneHotEncoder().fit_transform(Label_Encoding).toarray()
Categories_Dictionary={tuple(One_Hot_Encoding[i]):Categories[i] for i in range(len(Categories))}
print(Categories_Dictionary)
```

To show images:

```
Original=[]
plt.figure(figsize=(15,20)) for i
in range(len(Test_Images)):
plt.subplot(5,5,i+1)
plt.imshow(Test_Images[i])
  Original.append(Categories_Dictionary[tuple(Test_Labels[i])])
plt.xlabel(Original[i])
plt.show()
```

Errors:

```
print('Accuracy: ',Accuracy)
print('Loss: ',Loss) print('MSE: ',MSE)
print('MAE: ',MAE)
```

Accuracy:

```
Accuracy=0
for  i  in  range(len(Original)):       if  Original[i]==Predicted[i]:
Accuracy+=1      print("Correctly classified image: ",i+1)    else:
print("Wrongly   classified   "+Original[i]+'   as   '+Predicted[i])
plt.figure()                                          plt.imshow(Test_Images[i])
plt.xlabel("Wrongly classified
"+Original[i]+' as '+Predicted[i]) print('\nAccuracy =
',Accuracy,'/',len(Predicted),' = ',Accuracy/len(Predi cted)*100,'%')
```