```
#content google drive
from google.colab import drive
drive.mount('/content/drive', force_remount=True)
```

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

準備

```
[ ] ↳ 5 個のセルが非表示
```

ロード

```
from tensorflow import keras
model1 = keras.models.load_model('/content/drive/MyDrive/result11/model1.h5/')
model2 = keras.models.load_model('/content/drive/MyDrive/result11/model2.h5/')
model3 = keras.models.load_model('/content/drive/MyDrive/result11/model3.h5/')
```

アンサンブル

```
from sklearn.base import is_classifier, is_regressor
models_names=['model1','model2','model3']
print("model name\t estimator name\t is_regressor\t is_classifier")
for estimator , model_name in zip([model1,model2,model3],models_names):
        print("{}\t   {}     \t {}      \t {}".format(model_name,estimator.__class__.__name__,
                                                    is_regressor(estimator),
                                                    is_classifier(estimator)
                                                    ))
```

正常終了

```
model name       estimator name  is_regressor    is_classifier
model1     Sequential            False           False
model2     Sequential            False           False
model3     Sequential            False           False
```

```
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import classification_report
import numpy as np
models=[model1,model2,model3]
model_names=["model1","model2",'model3']
for model,model_name in zip(models,model_names):

    y_prob = model.predict(x_test)
    y_pred1 = y_prob.argmax(axis=-1)
    y_test1=np.argmax(y_test, axis=1)
    # accuracy
    print(model_name+" accuracy: ",accuracy_score(y_test1,y_pred1))
```

```
    del model , y_pred1, y_test1
    #Precision
    #print('Precision:', precision_score(y_test,y_pred1))
    #Recall
    #print('Recall:', recall_score(y_test1,y_pred1))
    #F-measure
    #print("Classification report")
    #print(classification_report(y_test1, y_pred1))
```

```
      1/1 [==============================] - 109s 109s/step
      model1 accuracy:  0.65625
      1/1 [==============================] - 105s 105s/step
      model2 accuracy:  0.6875
      1/1 [==============================] - 104s 104s/step
      model3 accuracy:  0.5625
```

```
# valid_set
x_test, y_test = valid_set.next()

#VotingClassifier(hard)
estimators = [model1,model2,model3]
vc=votingClassifer(estimators=estimators,mode="hard",show_info="percent")
index_classes, class_names, probs=vc.predict(x_test)
```

```
1/1 [==============================] - 14s 14s/step
WARNING:tensorflow:5 out of the last 5 calls to <function Model.make_predict_function.<locals>.predict_function at 0x7ff18065b9d0> triggered tf.function retracing. Tracing is expensive and the excessive number of tr
1/1 [==============================] - 13s 13s/step
WARNING:tensorflow:6 out of the last 6 calls to <function Model.make_predict_function.<locals>.predict_function at 0x7ff18065b430> triggered tf.function retracing. Tracing is expensive and the excessive number of tr
1/1 [==============================] - 13s 13s/step
1/1 [==============================] - 7s 7s/step
1/1 [==============================] - 6s 6s/step
1/1 [==============================] - 7s 7s/step
1/1 [==============================] - 7s 7s/step
1/1 [==============================] - 7s 7s/step
1/1 [==============================] - 7s 7s/step
1/1 [==============================] - 7s 7s/step
1/1 [==============================] - 7s 7s/step
1/1 [==============================] - 7s 7s/step
1/1 [==============================] - 7s 7s/step
1/1 [==============================] - 6s 6s/step
1/1 [==============================] - 6s 6s/step
1/1 [==============================] - 6s 6s/step
1/1 [==============================] - 7s 7s/step
1/1 [==============================] - 7s 7s/step
1/1 [==============================] - 6s 6s/step
1/1 [==============================] - 6s 6s/step
1/1 [==============================] - 7s 7s/step
1/1 [==============================] - 6s 6s/step
1/1 [==============================] - 7s 7s/step
1/1 [==============================] - 7s 7s/step
1/1 [==============================] - 7s 7s/step
1/1 [==============================] - 7s 7s/step
1/1 [==============================] - 8s 8s/step
1/1 [==============================] - 7s 7s/step
1/1 [==============================] - 7s 7s/step
1/1 [==============================] - 7s 7s/step
1/1 [==============================] - 6s 6s/step
1/1 [==============================] - 7s 7s/step
1/1 [==============================] - 7s 7s/step
```

正常終了

```
1/1 [==============================] - 7s 7s/step
1/1 [==============================] - 6s 6s/step
1/1 [==============================] - 7s 7s/step

 Each estimator predict a different class
1/1 [==============================] - 7s 7s/step
1/1 [==============================] - 7s 7s/step
1/1 [==============================] - 7s 7s/step
1/1 [==============================] - 7s 7s/step
1/1 [==============================] - 6s 6s/step
1/1 [==============================] - 7s 7s/step

 Each estimator predict a different class
1/1 [==============================] - 7s 7s/step
1/1 [==============================] - 7s 7s/step
1/1 [==============================] - 7s 7s/step
1/1 [==============================] - 7s 7s/step
1/1 [==============================] - 8s 8s/step
1/1 [==============================] - 7s 7s/step
1/1 [==============================] - 7s 7s/step
1/1 [==============================] - 7s 7s/step
1/1 [==============================] - 7s 7s/step
```

```
# inverse_to_categorical inverser format binary to format indexation
# datagenerator use the methode to_categorical for labelsation to frmat binary
# the method inverce of to_categorical is argmax
import numpy as np
y_test1=np.argmax(y_test, axis=1)
y_pred1=index_classes

# accuracy
from sklearn.metrics import accuracy_score
print("votingClassifer(hard) accuracy : ",accuracy_score(y_test1,y_pred1))
```

```
votingClassifer(hard) accuracy :  0.625
```

正常終了

✓ 0 秒　完了時間: 21:53

正常終了