Lab Report – 2

Goutham Krishnan

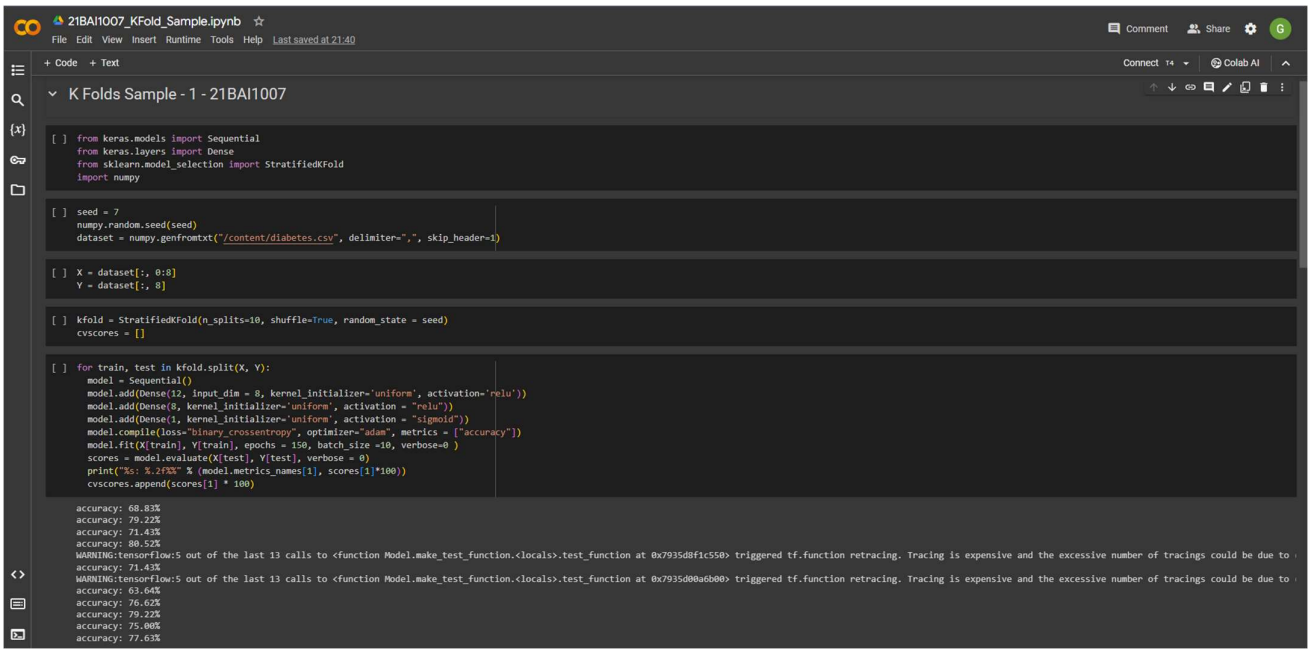
21BAI1007

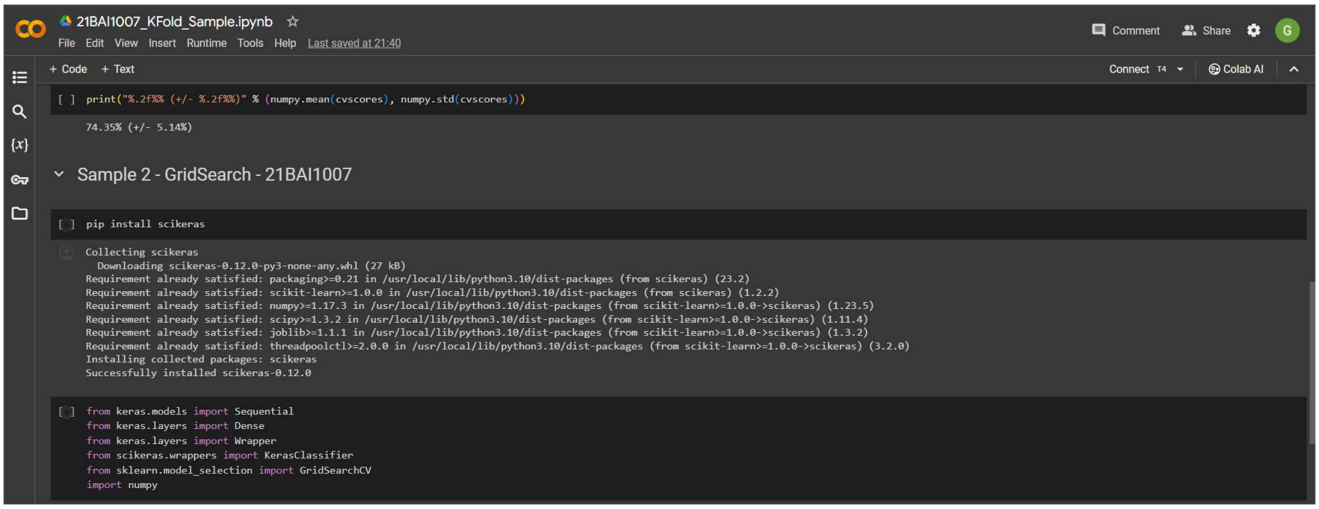
Aim

1. Perform classification on the PIMA diabetes dataset using K folds and Grid search autotuning.
2. Perform MNIST dataset classification using K Folds and Grid Search autotuning.
3. Perform Fashion MNIST classification using K Folds and Grid Search autotuning.
4. Perform classification on the Cats and Dogs dataset using K folds and Grid search autotuning.
5. Perform classification on the PCAM dataset using K folds and Grid Search autotuning.

Code and Output

Sample Code on K folds and Grid Search





21BAI1007_KFold_Sample.ipynb

File Edit View Insert Runtime Tools Help Last saved at 21:40

Comment Share

+ Code + Text

Connect T4 Colab AI

```
[ ] def create_model(optimizer="rmsprop", init = "glorot_uniform"):
      model = Sequential()
      model.add(Dense(12, input_dim = 8, kernel_initializer=init, activation = "relu"))
      model.add(Dense(8, kernel_initializer = init, activation="relu"))
      model.add(Dense(1, kernel_initializer=init, activation="sigmoid"))
      model.compile(loss="binary_crossentropy", optimizer=optimizer, metrics=["accuracy"])
      return model
```

```
[ ] seed = 7
    numpy.random.seed(seed)
    dataset = numpy.genfromtxt("/content/diabetes.csv", delimiter=",", skip_header=1)
```

```
[ ] X=dataset[:, 0:8]
    Y=dataset[:,8]
```

```
[ ] model = KerasClassifier(model=create_model, verbose=0)

    optimizers = ["rmsprop", "adam"]
    epochs = [50, 100, 150]
    batches = [5, 10, 20]
    param_grid = dict(optimizer=optimizers, epochs=epochs, batch_size=batches)

    grid = GridSearchCV(estimator=model, param_grid=param_grid)
    grid_result = grid.fit(X, Y)
```

```
[ ] # summarize results
    print("Best: %f using %s" % (grid_result.best_score_, grid_result.best_params_))
```

```
Best: 0.726670 using {'batch_size': 5, 'epochs': 150, 'optimizer': 'rmsprop'}
```

```
[ ] Start coding or generate with AI.
```

PIMA Dataset classification using autotuning

pima-21BAI1007.ipynb

File Edit View Insert Runtime Tools Help Last edited on 22 January

Comment Share

+ Code + Text

Connect T4 Colab AI

Pima Indian Diabetes Classification - 21BAI1007

```
[ ] import pandas as pd
    import numpy as np
    import matplotlib.pyplot as plt
```

```
[ ] from google.colab import drive
    drive.mount('/content/drive')
```

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

```
[ ] df = pd.read_csv('/content/drive/MyDrive/Datasets/diabetes.csv')
    df.head()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

pima-21BAI1007.ipynb

File Edit View Insert Runtime Tools Help Last edited on 22 January

Comment Share

+ Code + Text

Connect T4 Colab AI

EDA and Preprocessing - 21BAI1007

```
[ ] display(df.info(),df.head())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Pregnancies               768 non-null    int64
 1   Glucose                   768 non-null    int64
 2   BloodPressure             768 non-null    int64
 3   SkinThickness             768 non-null    int64
 4   Insulin                   768 non-null    int64
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64
 8   Outcome                   768 non-null    int64
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
None
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

pima-21BAI1007.ipynb

File Edit View Insert Runtime Tools Help Last edited on 22 January

Comment Share G

+ Code + Text

Connect T4 Colab AI

```
# Checking for null values if any
df.isna().sum()
```

```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

```
## Extracting the features
x = df.iloc[:, :-1]
y = df.iloc[:, -1:]
```

```
x.head()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
0	6	148	72	35	0	33.6	0.627	50
1	1	85	66	29	0	26.6	0.351	31
2	8	183	64	0	0	23.3	0.672	32
3	1	89	66	23	94	28.1	0.167	21
4	0	137	40	35	168	43.1	2.288	33

pima-21BAI1007.ipynb

File Edit View Insert Runtime Tools Help Last edited on 22 January

Comment Share G

+ Code + Text

Connect T4 Colab AI

Creating the model - 21BAI1007

```
from sklearn.model_selection import StratifiedKFold
kfold = StratifiedKFold(n_splits=10, shuffle=True, random_state=7)
cvscores = []
```

```
from keras.models import Sequential
from keras.layers import Dense, Dropout
```

```
for train, test in kfold.split(x, y):

  model = Sequential()
  model.add(Dense(512, input_dim=8, kernel_initializer='uniform' , activation= 'relu' ))
  model.add(Dense(256, kernel_initializer='uniform'))
  model.add(Dense(128, kernel_initializer= 'uniform'))
  model.add(Dense(32))
  model.add(Dense(1, activation='sigmoid'))

  model.compile(loss= "binary_crossentropy" , optimizer= "adam" , metrics=[ "accuracy" ])
  # Fit the model
  model.fit(x[train], y[train], epochs=150, verbose=0) # evaluate the model
  scores = model.evaluate(x[test], y[test], verbose=0)
  print("%s: %.2f%%" % (model.metrics_names[1], scores[1]*100))
  cvscores.append(scores[1] * 100)
```

```
accuracy: 75.32%
accuracy: 75.32%
accuracy: 71.43%
accuracy: 79.22%
WARNING:tensorflow:5 out of the last 13 calls to <function Model.make_test_function.<locals>.test_function at 0x7ad4307bab00> triggered tf.function retracing. Tracing is expensive and the excessive number c
accuracy: 68.83%
WARNING:tensorflow:5 out of the last 13 calls to <function Model.make_test_function.<locals>.test_function at 0x7ad43031c820> triggered tf.function retracing. Tracing is expensive and the excessive number c
accuracy: 67.53%
accuracy: 76.62%
accuracy: 75.32%
```

pima-21BAI1007.ipynb

File Edit View Insert Runtime Tools Help Last edited on 22 January

Comment Share G

+ Code + Text

Connect T4 Colab AI

accuracy: 75.32%
accuracy: 75.32%
accuracy: 71.43%
accuracy: 79.22%
WARNING:tensorflow:5 out of the last 13 calls to <function Model.make_test_function.<locals>.test_function at 0x7ad4307bab00> triggered tf.function retracing. Tracing is expensive and the excessive number c
accuracy: 68.83%
WARNING:tensorflow:5 out of the last 13 calls to <function Model.make_test_function.<locals>.test_function at 0x7ad43031c820> triggered tf.function retracing. Tracing is expensive and the excessive number c
accuracy: 67.53%
accuracy: 76.62%
accuracy: 75.32%
accuracy: 77.63%
accuracy: 82.89%

```
print("%.2f%% (+/- %.2f%%)" % (np.mean(cvscores), np.std(cvscores)))
```

```
75.01% (+/- 4.43%)
```

Prediciting the test output - 21BAI1007

```
probs = model.predict(x[test])
```

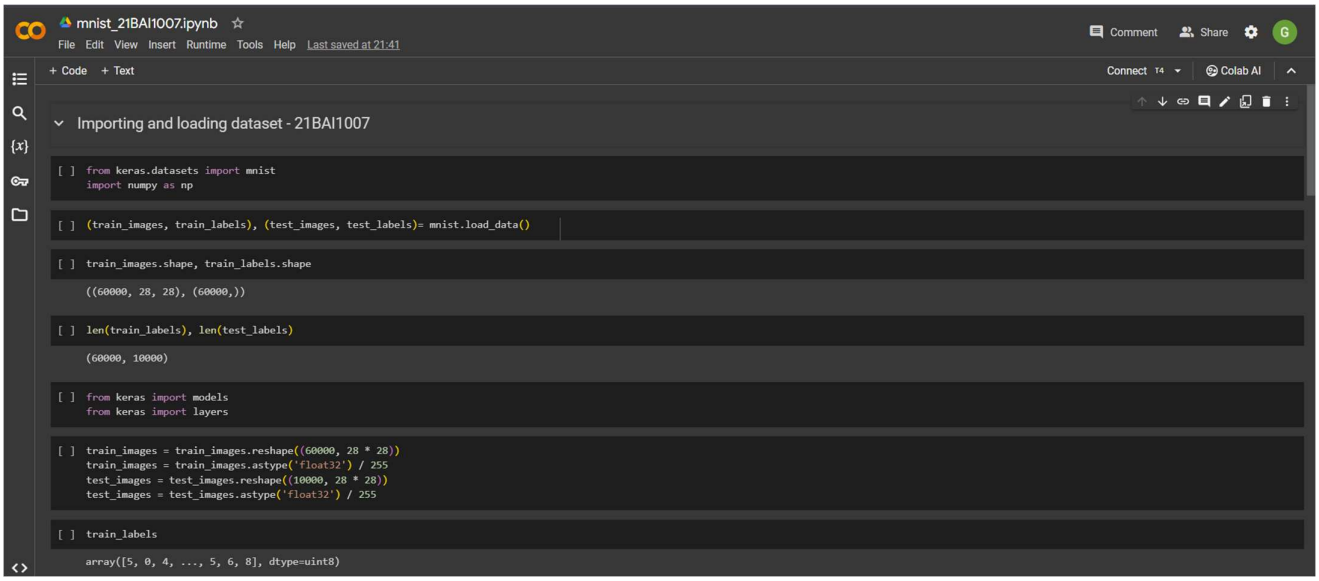
```
3/3 [==============================] - 0s 3ms/step
```

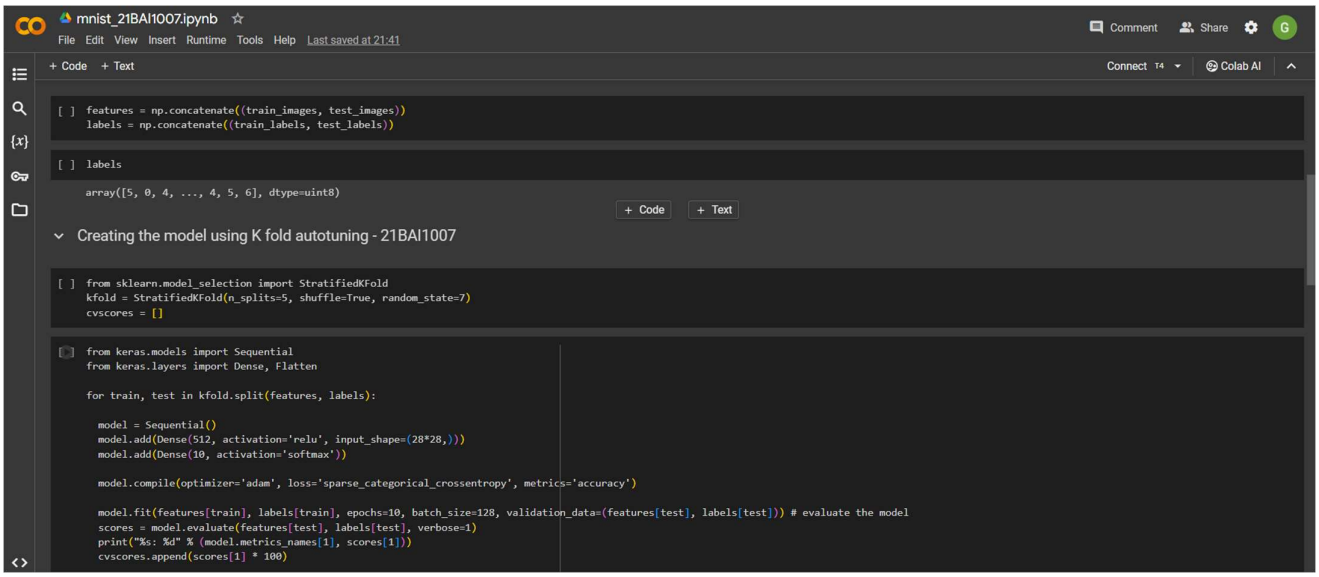
```
th = 0.5
y_pred = np.where(probs > th, 1, 0)
```

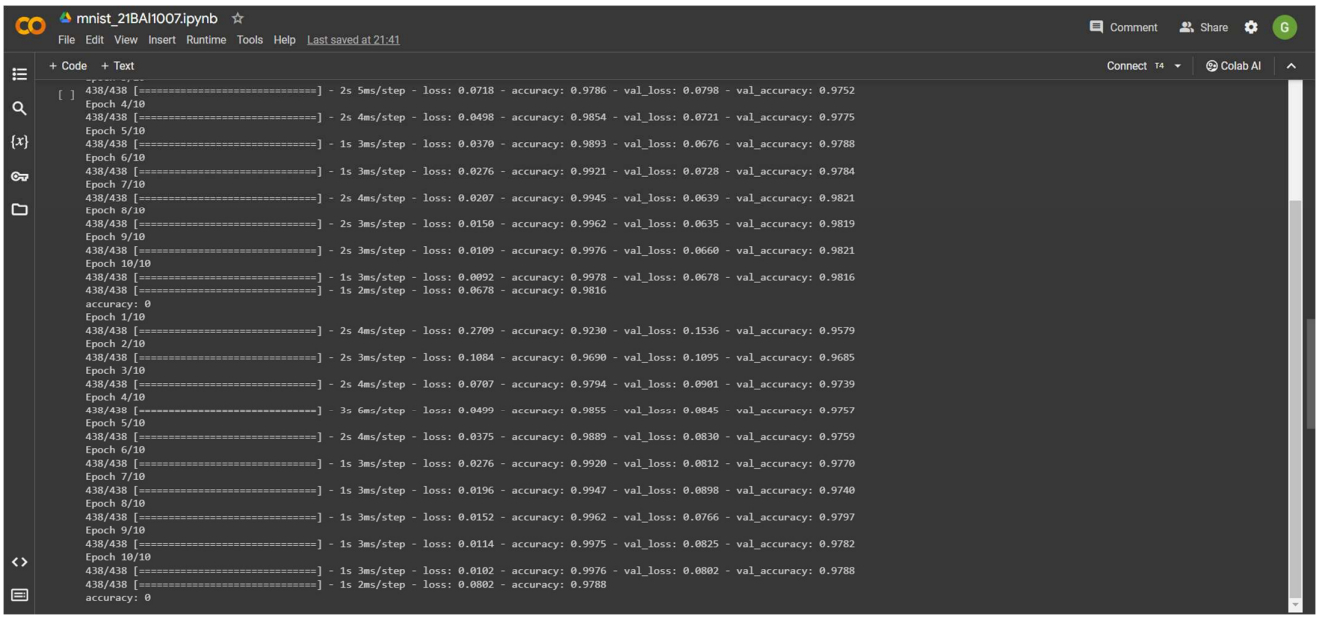
```
from sklearn.metrics import accuracy_score
print(accuracy_score(np.array(y[test]), y_pred)*100, "%")
```

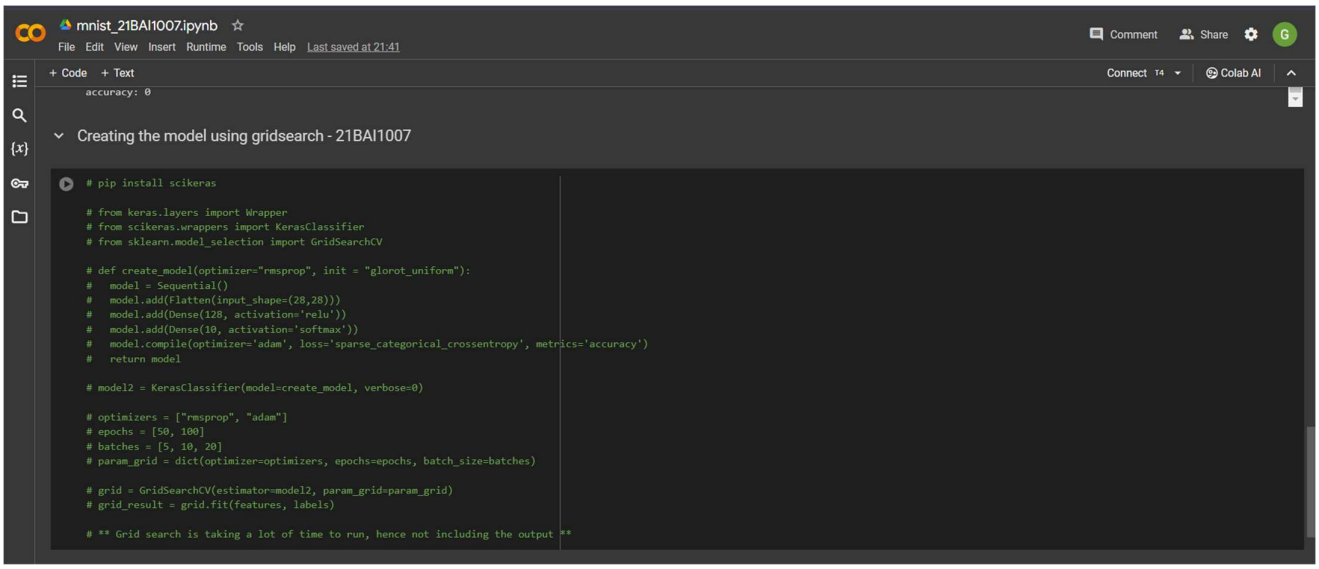
```
82.89473684210526 %
```

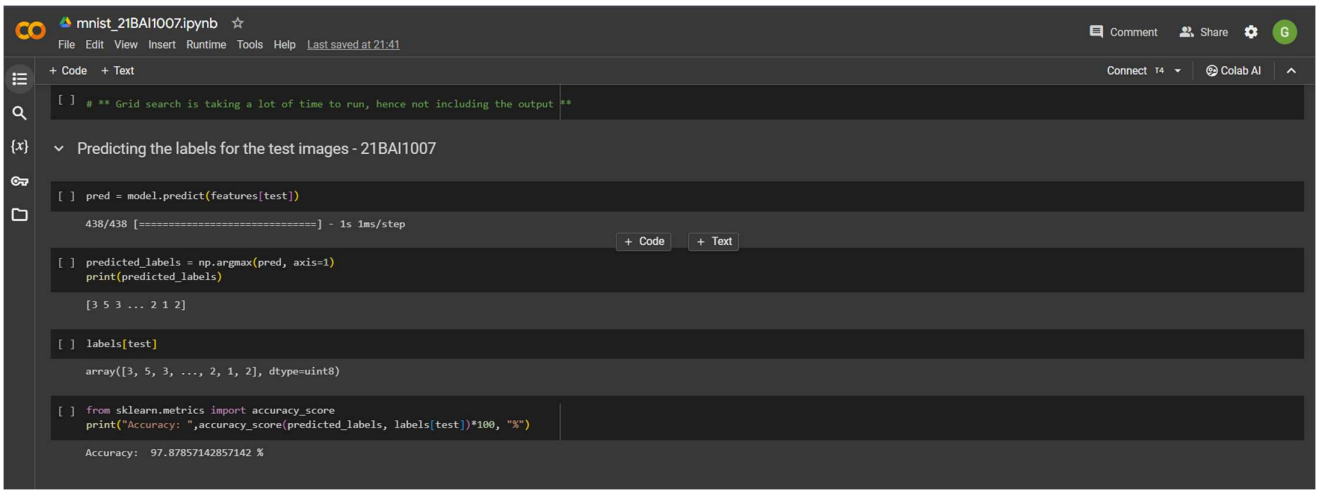
MNIST Dataset Classification using Autotuning



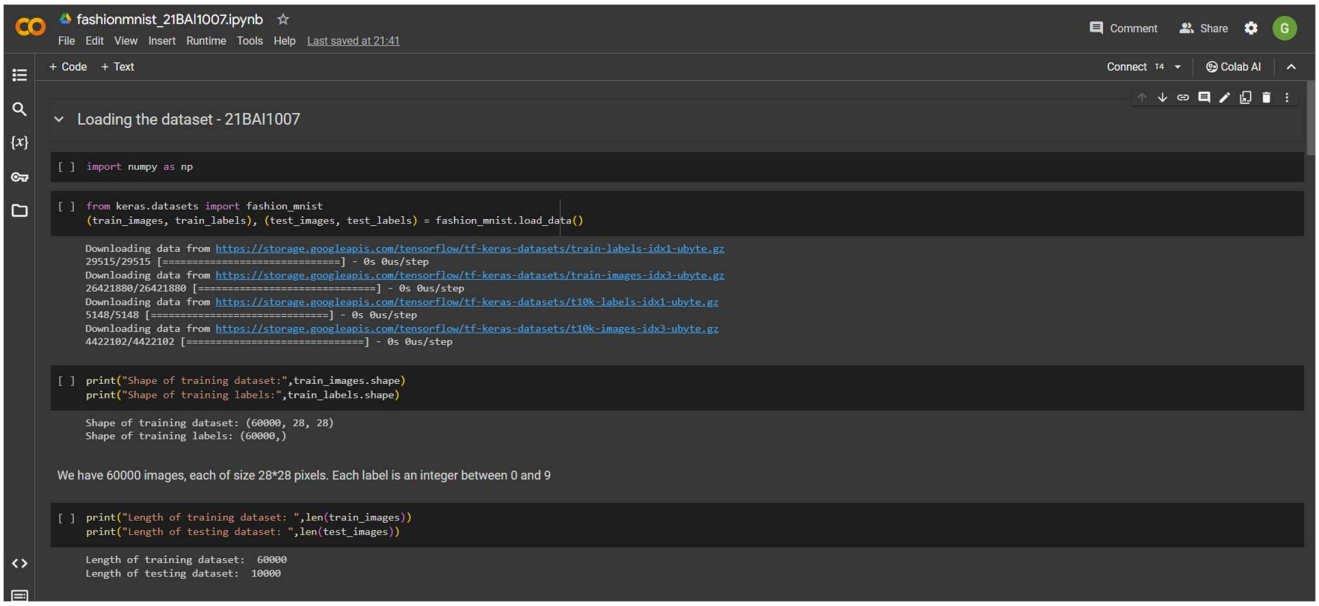








Fashion MNIST dataset classification using autotuning



fashionmnist_21BAI1007.ipynb

File Edit View Insert Runtime Tools Help Last saved at 21:41

Comment Share G

+ Code + Text

Connect T4 Colab AI

We have 60000 images, each of size 28*28 pixels. Each label is an integer between 0 and 9

```
[ ] print("Length of training dataset: ",len(train_images))
    print("Length of testing dataset: ",len(test_images))
```

```
Length of training dataset:  60000
Length of testing dataset:  10000
```

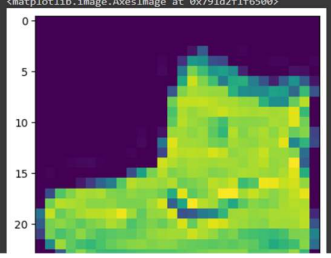
Normalizing the dataset - 21BAI1007

```
[ ] import matplotlib.pyplot as plt
```

+ Code + Text

```
[ ] ## Plotting the image of the first object in the dataset
    plt.figure()
    plt.imshow(train_images[0])
```

```
<matplotlib.image.AxesImage at 0x791d2f1f6500>
```



fashionmnist_21BAI1007.ipynb

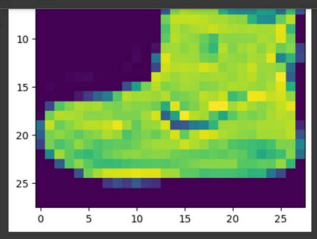
File Edit View Insert Runtime Tools Help Last saved at 21:41

Comment Share G

+ Code + Text

Connect T4 Colab AI

```
[ ]
```



+ Code + Text

```
[ ] ## To normalize the dataset, we divide each value by 255 so that the data is minimized to a value between 0 and 1
    train_images = train_images/255.0
    test_images = test_images/255.0
```

Build the model using K Fold auto tuning- 21BAI1007

```
[ ] features = np.concatenate((train_images, test_images))
    labels = np.concatenate((train_labels, test_labels))
```

```
[ ] from keras.models import Sequential
    from keras.layers import Dense, Flatten
```

```
[ ] from sklearn.model_selection import StratifiedKFold
    kfold = StratifiedKFold(n_splits=5, shuffle=True, random_state=7)
    cvscores = []
```

fashionmnist_21BAI1007.ipynb

File Edit View Insert Runtime Tools Help Last saved at 21:41

Comment Share G

+ Code + Text

Connect T4 Colab AI

```
[ ] for train, test in kfold.split(features, labels):
      model = Sequential()
      model.add(Flatten(input_shape=(28,28)))
      model.add(Dense(128, activation='relu'))
      model.add(Dense(10, activation='softmax'))
      model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics='accuracy')
      model.fit(features[train], labels[train], validation_data=(features[test], labels[test]), epochs=10)
      scores = model.evaluate(features[test], labels[test], verbose=1)
      print("%s: %d" % (model.metrics_names[1], scores[1]))
      cvscores.append(scores[1] * 100)
```

```
Epoch 1/10
1750/1750 [==============================] - 6s 3ms/step - loss: 0.5051 - accuracy: 0.8213 - val_loss: 0.4371 - val_accuracy: 0.8414
Epoch 2/10
1750/1750 [==============================] - 6s 3ms/step - loss: 0.3822 - accuracy: 0.8621 - val_loss: 0.3690 - val_accuracy: 0.8704
Epoch 3/10
1750/1750 [==============================] - 5s 3ms/step - loss: 0.3413 - accuracy: 0.8752 - val_loss: 0.3557 - val_accuracy: 0.8739
Epoch 4/10
1750/1750 [==============================] - 6s 3ms/step - loss: 0.3162 - accuracy: 0.8833 - val_loss: 0.3426 - val_accuracy: 0.8788
Epoch 5/10
1750/1750 [==============================] - 5s 3ms/step - loss: 0.3008 - accuracy: 0.8875 - val_loss: 0.3243 - val_accuracy: 0.8862
Epoch 6/10
1750/1750 [==============================] - 5s 3ms/step - loss: 0.2842 - accuracy: 0.8934 - val_loss: 0.3135 - val_accuracy: 0.8920
Epoch 7/10
1750/1750 [==============================] - 6s 4ms/step - loss: 0.2715 - accuracy: 0.8985 - val_loss: 0.3690 - val_accuracy: 0.8701
Epoch 8/10
1750/1750 [==============================] - 5s 3ms/step - loss: 0.2602 - accuracy: 0.9020 - val_loss: 0.3179 - val_accuracy: 0.8930
Epoch 9/10
1750/1750 [==============================] - 6s 3ms/step - loss: 0.2501 - accuracy: 0.9057 - val_loss: 0.3292 - val_accuracy: 0.8851
Epoch 10/10
1750/1750 [==============================] - 5s 3ms/step - loss: 0.2413 - accuracy: 0.9088 - val_loss: 0.3057 - val_accuracy: 0.8937
438/438 [==============================] - 1s 2ms/step - loss: 0.3057 - accuracy: 0.8937
accuracy: 0
Epoch 1/10
1750/1750 [==============================] - 7s 3ms/step - loss: 0.5110 - accuracy: 0.8207 - val_loss: 0.4377 - val_accuracy: 0.8391
Epoch 2/10
1750/1750 [==============================] - 5s 3ms/step - loss: 0.3892 - accuracy: 0.8602 - val_loss: 0.3626 - val_accuracy: 0.8702
Epoch 3/10
```

fashionmnist_21BAI1007.ipynb

File Edit View Insert Runtime Tools Help Last saved at 21:41

Comment Share G

+ Code + Text Connect T4 Colab AI

```
Epoch 1/10
1750/1750 [==============================] - 7s 3ms/step - loss: 0.5112 - accuracy: 0.8203 - val_loss: 0.4079 - val_accuracy: 0.8551
Epoch 2/10
1750/1750 [==============================] - 5s 3ms/step - loss: 0.3818 - accuracy: 0.8626 - val_loss: 0.3761 - val_accuracy: 0.8627
Epoch 3/10
1750/1750 [==============================] - 5s 3ms/step - loss: 0.3424 - accuracy: 0.8767 - val_loss: 0.3324 - val_accuracy: 0.8795
Epoch 4/10
1750/1750 [==============================] - 6s 3ms/step - loss: 0.3194 - accuracy: 0.8812 - val_loss: 0.3373 - val_accuracy: 0.8769
Epoch 5/10
```

Creating model using grid search - 21BAI1007

```
# pip install scikeras

# from keras.layers import Wrapper
# from scikeras.wrappers import KerasClassifier
# from sklearn.model_selection import GridSearchCV

# def create_model(optimizer="rmsprop", init = "glorot_uniform"):
#   model = Sequential()
#   model.add(Flatten(input_shape=(28,28)))
#   model.add(Dense(128, activation='relu'))
#   model.add(Dense(10, activation='softmax'))
#   model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics='accuracy')
#   return model

# model2 = KerasClassifier(model=create_model, verbose=0)

# optimizers = ["rmsprop", "adam"]
# epochs = [50, 100]
# batches = [5, 10, 20]
# param_grid = dict(optimizer=optimizers, epochs=epochs, batch_size=batches)

# grid = GridSearchCV(estimator=model2, param_grid=param_grid)
# grid_result = grid.fit(features, labels)

# ** Grid search is taking a lot of time to run, hence not including the output **
```

fashionmnist_21BAI1007.ipynb

File Edit View Insert Runtime Tools Help Last saved at 21:41

Comment Share G

+ Code + Text Connect T4 Colab AI

Predicting the classes of the test dataset - 21BAI1007

```
pred = model.predict(test_images)
```

```
313/313 [==============================] - 1s 1ms/step
```

```
# The predictions are a probability distribution of each class. To convert it into the required class
# Predicting the class of the first element in the testing dataset
np.argmax(pred[0])
```

```
9
```

```
# Checking if the prediction is correct
test_labels[0]
```

```
9
```

```
class_names = ['T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat', 'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle boot']
```

Predciting a random image from the test dataset - 21BAI1007

```
image = test_images[1] ## An image of a pullover
print(image.shape)
```

```
(28, 28)
```

fashionmnist_21BAI1007.ipynb

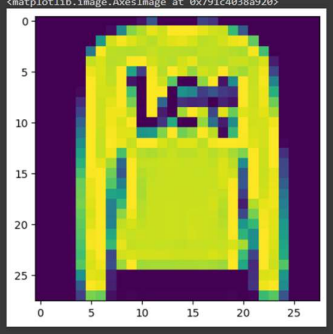
File Edit View Insert Runtime Tools Help Last saved at 21:41

Comment Share G

+ Code + Text Connect T4 Colab AI

```
plt.imshow(test_images[1])
```

```
<matplotlib.image.AxesImage at 0x791c4038a920>
```




```
# To process the image in the model, we have to reshape it no include an extra dimension
image = (np.expand_dims(image, 0))
print(image.shape)
```

```
(1, 28, 28)
```

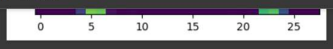
fashionmnist_21BAI1007.ipynb

File Edit View Insert Runtime Tools Help Last saved at 21:41

Comment Share

+ Code + Text

Connect T4 Colab AI




```
[ ] # To process the image in the model, we have to reshape it no include an extra dimension
    image = (np.expand_dims(image, 0))
    print(image.shape)
```

```
(1, 28, 28)
```

```
[ ] # Function to map the class number to the class name
    def get_item_name(item_number, class_names):
        try:
            item_name = class_names[item_number]
            return item_name
        except IndexError:
            return "Item not found"
```

```
[ ] final_prediction = model.predict(image)
    ans = np.argmax(final_prediction)
    item_name = get_item_name(ans, class_names)
    item_name
```

```
1/1 [==============================] - 0s 69ms/step
'Pullover'
```

Hence we correctly predicted the class of the image

Cats Vs Dogs Dataset Classification using Autotuning

CatsDogs_KFold_21BAI1007.ipynb

File Edit View Insert Runtime Tools Help Last saved at 21:40

Comment Share

+ Code + Text

Connect T4 Colab AI

Cats vs Dogs Classification - 21BAI1007

Importing data From Kaggle

```
[ ] !mkdir -p ~/.kaggle
    !cp kaggle.json ~/.kaggle/
```

```
[ ] !kaggle datasets download -d salader/dogs-vs-cats
```

```
Warning: Your Kaggle API key is readable by other users on this system! To fix this, you can run 'chmod 600 /root/.kaggle/kaggle.json'
Downloading dogs-vs-cats.zip to /content
 99% 1.05G/1.06G [00:07<00:00, 93.0MB/s]
100% 1.06G/1.06G [00:07<00:00, 155MB/s]
```

```
[ ] # Data Loaded from kaggle is in zip format. Need to unzip it
    import zipfile
    zip_ref = zipfile.ZipFile('/content/dogs-vs-cats.zip', 'r')
    zip_ref.extractall('/content')
    zip_ref.close()
```

CatsDogs_KFold_21BAI1007.ipynb

File Edit View Insert Runtime Tools Help Last saved at 21:40

Comment Share

+ Code + Text

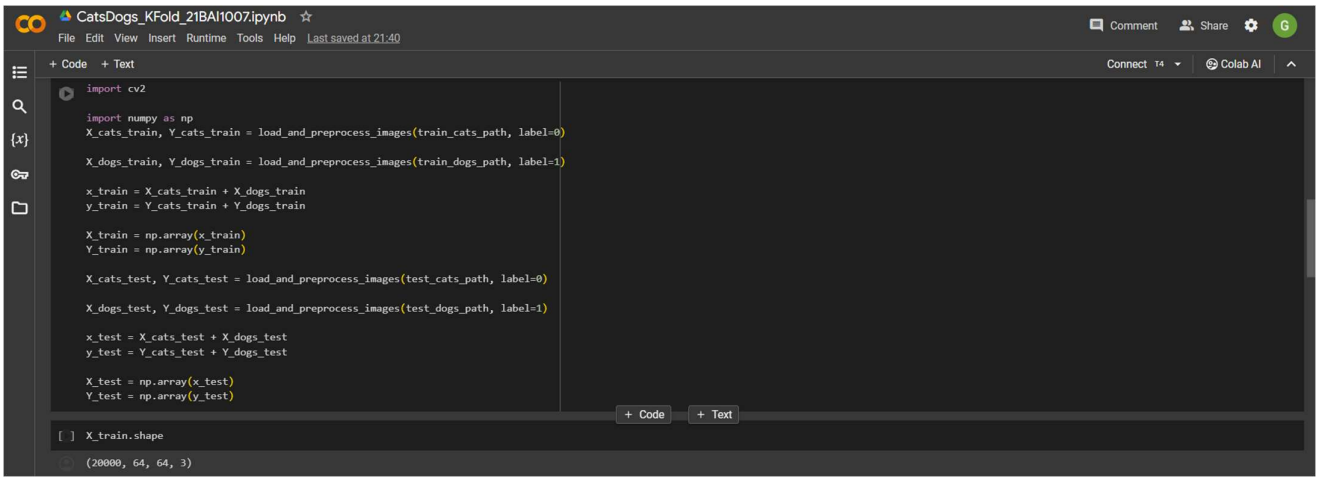
Connect T4 Colab AI

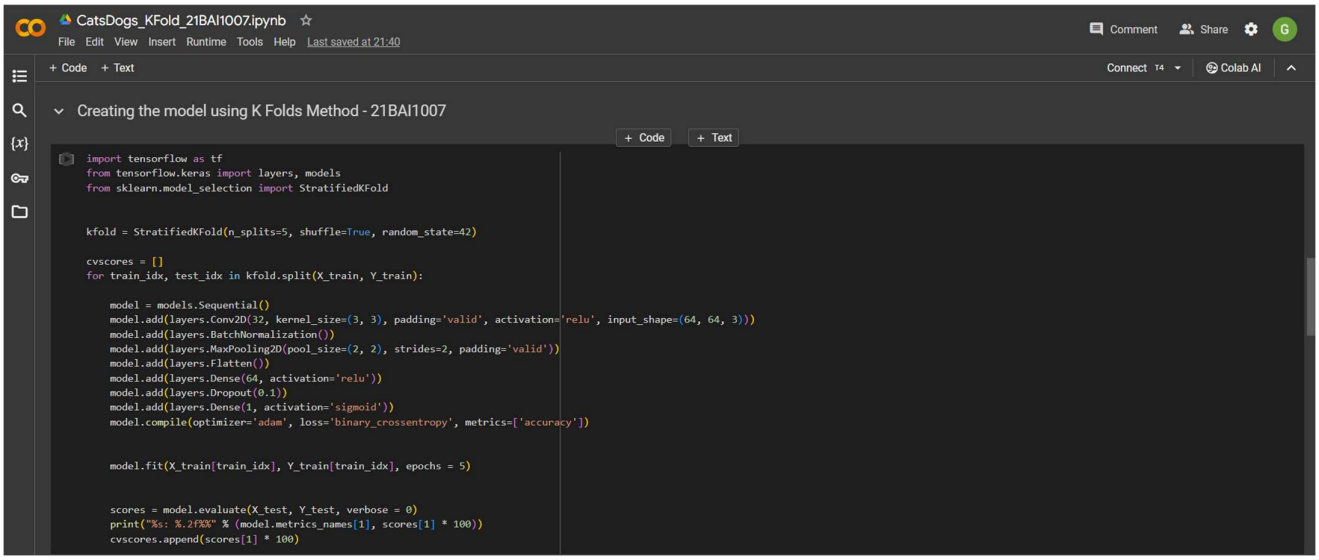
Loading the data - 21BAI1007

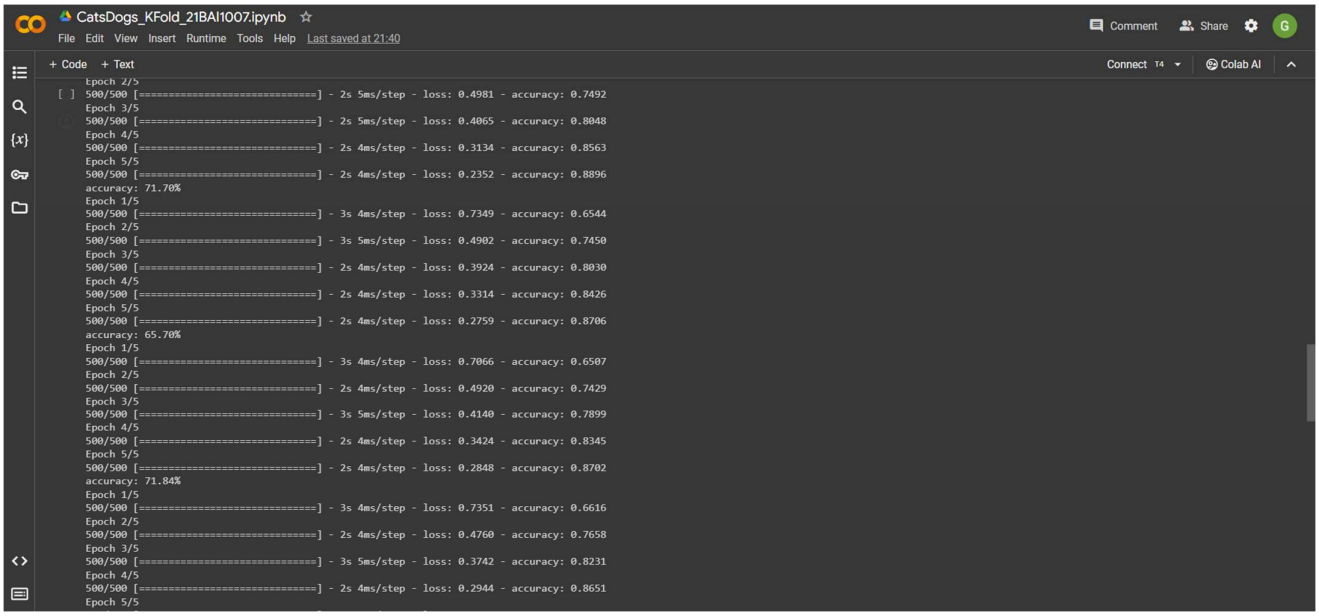
```
[ ] import tensorflow as tf
    import keras
    from keras.models import Sequential
    from keras.layers import Dense, Conv2D, MaxPooling2D, Flatten, BatchNormalization, Dropout
```

```
[ ] train_cats_path = '/content/train/cats'
    train_dogs_path = '/content/train/dogs'
    test_cats_path = '/content/test/cats'
    test_dogs_path = '/content/test/dogs'
```

```
[ ] import os
    def load_and_preprocess_images(directory, label, target_size=(64, 64)):

        x = []
        y = []

        for filename in os.listdir(directory):
            if filename.endswith('.jpg') or filename.endswith('.png'):
                filepath = os.path.join(directory, filename)
                img = cv2.imread(filepath)
                if img is not None:
                    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
                    img = cv2.resize(img, target_size)
                    x.append(img)
                    y.append(label)

        return x,y
```


CatsDogs_KFold_21BAI1007.ipynb
File Edit View Insert Runtime Tools Help Last saved at 21:40
Comment
Share
+ Code + Text
Connect T4
Colab AI
import cv2
import numpy as np
X_cats_train, Y_cats_train = load_and_preprocess_images(train_cats_path, label=0)
X_dogs_train, Y_dogs_train = load_and_preprocess_images(train_dogs_path, label=1)
x_train = X_cats_train + X_dogs_train
y_train = Y_cats_train + Y_dogs_train
X_train = np.array(x_train)
Y_train = np.array(y_train)
X_cats_test, Y_cats_test = load_and_preprocess_images(test_cats_path, label=0)
X_dogs_test, Y_dogs_test = load_and_preprocess_images(test_dogs_path, label=1)
x_test = X_cats_test + X_dogs_test
y_test = Y_cats_test + Y_dogs_test
X_test = np.array(x_test)
Y_test = np.array(y_test)
+ Code + Text
[ ] X_train.shape
(20000, 64, 64, 3)


CatsDogs_KFold_21BAI1007.ipynb
File Edit View Insert Runtime Tools Help Last saved at 21:40
Comment
Share
+ Code + Text
Connect T4
Colab AI
Creating the model using K Folds Method - 21BAI1007
+ Code + Text
import tensorflow as tf
from tensorflow.keras import layers, models
from sklearn.model_selection import StratifiedKFold
kfold = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
cvscores = []
for train_idx, test_idx in kfold.split(X_train, Y_train):
model = models.Sequential()
model.add(layers.Conv2D(32, kernel_size=(3, 3), padding='valid', activation='relu', input_shape=(64, 64, 3)))
model.add(layers.BatchNormalization())
model.add(layers.MaxPooling2D(pool_size=(2, 2), strides=2, padding='valid'))
model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dropout(0.1))
model.add(layers.Dense(1, activation='sigmoid'))
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
model.fit(X_train[train_idx], Y_train[train_idx], epochs = 5)
scores = model.evaluate(X_test, Y_test, verbose = 0)
print("%s: %.2f%%" % (model.metrics_names[1], scores[1] * 100))
cvscores.append(scores[1] * 100)


CatsDogs_KFold_21BAI1007.ipynb
File Edit View Insert Runtime Tools Help Last saved at 21:40
Comment
Share
+ Code + Text
Connect T4
Colab AI
Epoch 2/5
[ ] 500/500 [==========] - 2s 5ms/step - loss: 0.4981 - accuracy: 0.7492
Epoch 3/5
500/500 [==========] - 2s 5ms/step - loss: 0.4065 - accuracy: 0.8048
Epoch 4/5
500/500 [==========] - 2s 4ms/step - loss: 0.3134 - accuracy: 0.8563
Epoch 5/5
500/500 [==========] - 2s 4ms/step - loss: 0.2352 - accuracy: 0.8896
accuracy: 71.70%
Epoch 1/5
500/500 [==========] - 3s 4ms/step - loss: 0.7349 - accuracy: 0.6544
Epoch 2/5
500/500 [==========] - 3s 5ms/step - loss: 0.4902 - accuracy: 0.7450
Epoch 3/5
500/500 [==========] - 2s 4ms/step - loss: 0.3924 - accuracy: 0.8030
Epoch 4/5
500/500 [==========] - 2s 4ms/step - loss: 0.3314 - accuracy: 0.8426
Epoch 5/5
500/500 [==========] - 2s 4ms/step - loss: 0.2759 - accuracy: 0.8706
accuracy: 65.70%
Epoch 1/5
500/500 [==========] - 3s 4ms/step - loss: 0.7066 - accuracy: 0.6507
Epoch 2/5
500/500 [==========] - 2s 4ms/step - loss: 0.4920 - accuracy: 0.7429
Epoch 3/5
500/500 [==========] - 3s 5ms/step - loss: 0.4140 - accuracy: 0.7899
Epoch 4/5
500/500 [==========] - 2s 4ms/step - loss: 0.3424 - accuracy: 0.8345
Epoch 5/5
500/500 [==========] - 2s 4ms/step - loss: 0.2848 - accuracy: 0.8702
accuracy: 71.84%
Epoch 1/5
500/500 [==========] - 3s 4ms/step - loss: 0.7351 - accuracy: 0.6616
Epoch 2/5
500/500 [==========] - 2s 4ms/step - loss: 0.4760 - accuracy: 0.7658
Epoch 3/5
500/500 [==========] - 3s 5ms/step - loss: 0.3742 - accuracy: 0.8231
Epoch 4/5
500/500 [==========] - 2s 4ms/step - loss: 0.2944 - accuracy: 0.8651
Epoch 5/5

CatsDogs_KFold_21BAI1007.ipynb

File Edit View Insert Runtime Tools Help Last saved at 21:40

Comment Share G

+ Code + Text

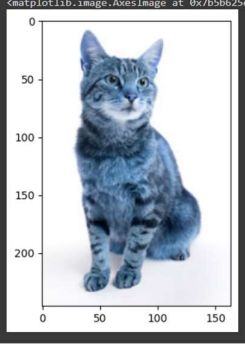
Connect T4 Colab AI

Predicting a random image - 21BAI1007

```
[ ] import cv2
    import matplotlib.pyplot as plt
    test_image = cv2.imread('/content/cat.jpg')
```

```
[ ] plt.imshow(test_image)
```

```
<matplotlib.image.AxesImage at 0x7b5b625e90c0>
```



CatsDogs_KFold_21BAI1007.ipynb

File Edit View Insert Runtime Tools Help Last saved at 21:40

Comment Share G

+ Code + Text

Connect T4 Colab AI

```
[ ] test_image = cv2.resize(test_image,(64, 64))
    test_input = test_image.reshape((1,64, 64,3))
```

```
[ ] predictions_probs = model.predict(test_input)
    predictions = np.argmax(predictions_probs)
    predictions
```

```
1/1 [==============================] - 0s 18ms/step
0
```

Array[0] -> Cat

Hence, the model has correctly predicted the given image of a cat

CatsDogs_KFold_21BAI1007.ipynb

File Edit View Insert Runtime Tools Help Last saved at 21:40

Comment Share G

+ Code + Text

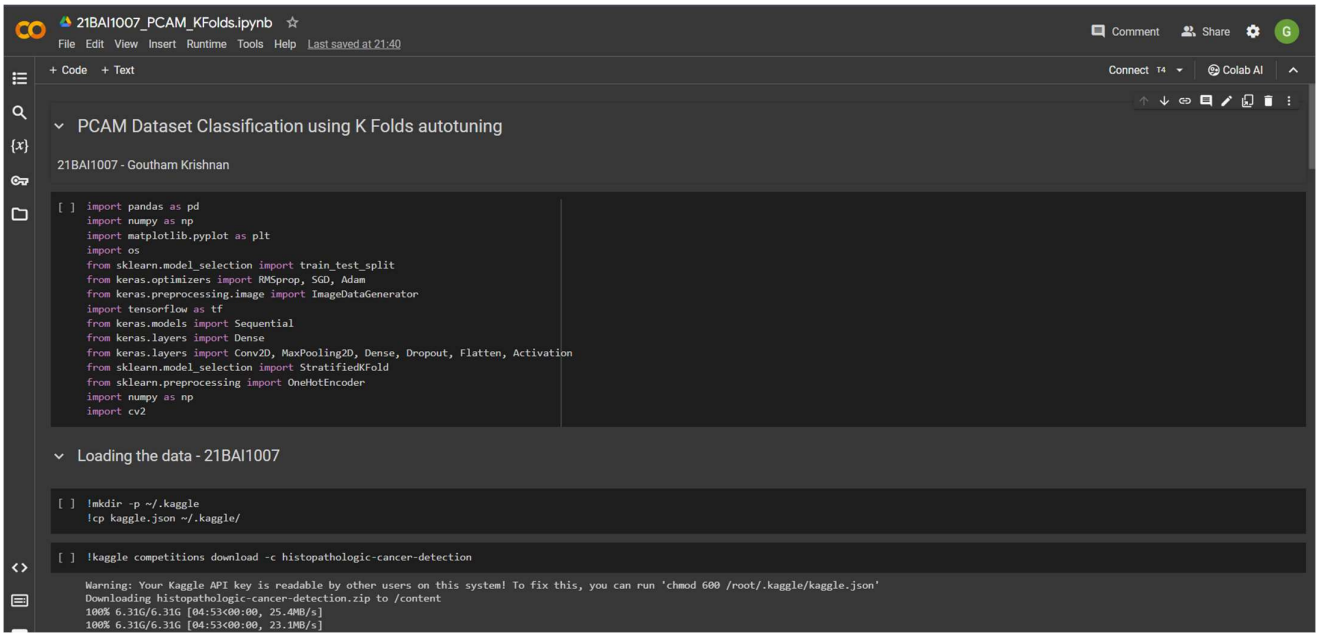
Connect T4 Colab AI

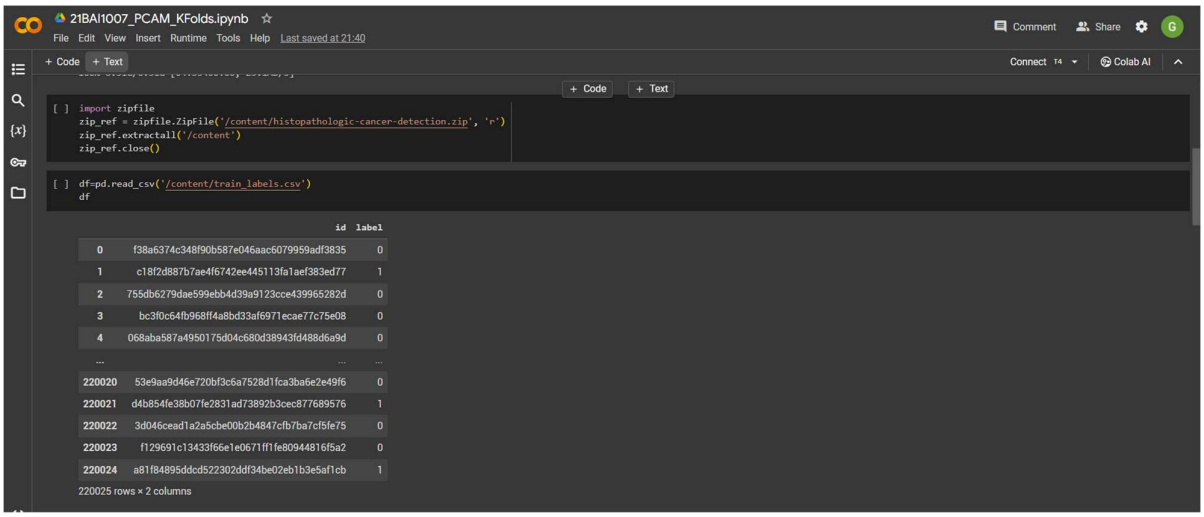
Creating model using grid search - 21BAI1007

** Grid search is taking a lot of time to run, hence not including the output **

```
[ ] # pip install scikeras

    # from keras.layers import Wrapper
    # from scikeras.wrappers import KerasClassifier
    # from sklearn.model_selection import GridSearchCV

    # def create_model(optimizer="rmsprop", init = "glorot_uniform"):
    #     model = models.Sequential()
    #     model.add(layers.Conv2D(32, kernel_size=(3, 3), padding='valid', activation='relu', input_shape=(64, 64, 3)))
    #     model.add(layers.BatchNormalization())
    #     model.add(layers.MaxPooling2D(pool_size=(2, 2), strides=2, padding='valid'))
    #     model.add(layers.Flatten())
    #     model.add(layers.Dense(64, activation='relu'))
    #     model.add(layers.Dropout(0.1))
    #     model.add(layers.Dense(1, activation='sigmoid'))
    #     model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
    #     return model

    # model2 = KerasClassifier(model=create_model, verbose=0)

    # optimizers = ["rmsprop", "adam"]
    # epochs = [50, 100]
    # batches = [5, 10, 20]
    # param_grid = dict(optimizer=optimizers, epochs=epochs, batch_size=batches)

    # grid = GridSearchCV(estimator=model2, param_grid=param_grid)
    # grid_result = grid.fit(X_train, Y_train)
```

PCAM Dataset Classification using Autotuning.





	id	label
0	f38a6374c348f90b587e046aac6079959adf3835	0
1	c18f2d887b7ae4f6742ee445113fa1aef383ed77	1
2	755db6279dae599ebb4d39a9123cce439965282d	0
3	bc3f0c64fb968ff4a8bd33af6971ecae77c75e08	0
4	068aba587a4950175d04c680d38943fd488d6a9d	0
...	...	...
220020	53e9aa9d46e720bf3c6a7528d1fca3ba6e2e49f6	0
220021	d4b854fe38b07fe2831ad73892b3cec877689576	1
220022	3d046cead1a2a5cbe00b2b4847cfb7ba7cf5fe75	0
220023	f129691c13433f66e1e0671ff1fe80944816f5a2	0
220024	a81f84895ddcd522302ddf34be02eb1b3e5af1cb	1

220025 rows × 2 columns

21BAI1007_PCAM_KFolds.ipynb

File Edit View Insert Runtime Tools Help Last saved at 21:40

Comment Share G

+ Code + Text

Connect T4 Colab AI

```
[ ] test_dir = "test_data"
    if os.path.exists(test_dir):
        shutil.rmtree(test_dir)
    os.makedirs(test_dir)
    os.makedirs(os.path.join(test_dir, "0"))
    os.makedirs(os.path.join(test_dir, "1"))
    for file_path, label in zip(test_file_paths, test_labels):
        name = file_path.split("/")[-1]
        if label == "0":
            shutil.copy2(file_path, os.path.join(test_dir, "0", name))
        else:
            shutil.copy2(file_path, os.path.join(test_dir, "1", name))
```

```
[ ] label_0_path_train = '/content/train_data/0'
    label_1_path_train = '/content/train_data/1'
```

```
[ ] def load_images(file_paths, label):
        images = []
        labels = []
        for file_path in file_paths:
            image = cv2.imread(file_path, cv2.IMREAD_UNCHANGED)
            image = cv2.resize(image, (64, 64))
            images.append(image)
            labels.append(label)
        return np.array(images), np.array(labels)
```

```
[ ] label_0_file_paths = [os.path.join(label_0_path_train, file) for file in os.listdir(label_0_path_train)]
    X_label_0, Y_label_0 = load_images(label_0_file_paths, 0)

    label_1_file_paths = [os.path.join(label_1_path_train, file) for file in os.listdir(label_1_path_train)]
    X_label_1, Y_label_1 = load_images(label_1_file_paths, 1)
```

21BAI1007_PCAM_KFolds.ipynb

File Edit View Insert Runtime Tools Help Last saved at 21:40

Comment Share G

+ Code + Text

Connect T4 Colab AI

```
[ ] x = np.concatenate((X_label_0, X_label_1), axis=0)
    y = np.concatenate((Y_label_0, Y_label_1), axis=0)
```

Creating the model - 21BAI1007

```
    seed = 7
    np.random.seed(seed)

    kfold = StratifiedKFold(n_splits=5, shuffle=True, random_state=seed)
    cvscores = []
```

```
[ ] from sklearn.preprocessing import StandardScaler
    scaler = StandardScaler()
    x = scaler.fit_transform(x.reshape(x.shape[0], -1)).reshape(x.shape)
```

21BAI1007_PCAM_KFolds.ipynb

File Edit View Insert Runtime Tools Help Last saved at 21:40

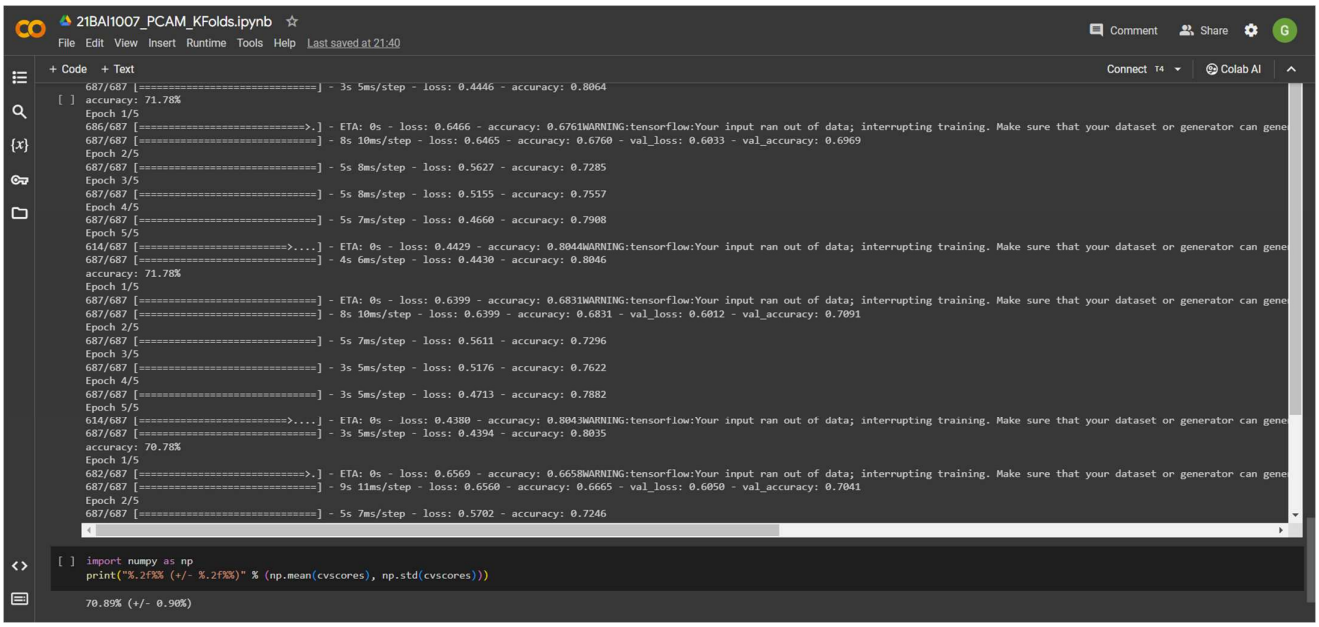
Comment Share G

+ Code + Text

Connect T4 Colab AI

```
for train, test in kfold.split(x, y):

    model = Sequential()

    model.add(Conv2D(filters=16, kernel_size=(3,3)))
    model.add(Conv2D(filters=16, kernel_size=(3,3)))
    model.add(MaxPooling2D(pool_size=(2,2)))

    model.add(Conv2D(filters=32, kernel_size=(3,3)))
    model.add(Conv2D(filters=32, kernel_size=(3,3)))
    model.add(Flatten())
    model.add(Dense(1, activation='sigmoid'))

    model.build(input_shape=(32, 64, 64, 3))

    model.compile(loss='binary_crossentropy', metrics=['accuracy'])

    model.fit(x[train],y[train], steps_per_epoch=687,epochs = 5,validation_data = (x[test],y[test]),validation_steps=171,verbose=1)

    scores = model.evaluate(x[test], y[test], verbose = 0)
    print("%s: %.2f%%" % (model.metrics_names[1], scores[1]*100))
    cvscores.append(scores[1] * 100)
```

```
Epoch 1/5
684/687 [============================>.] - ETA: 0s - loss: 0.6426 - accuracy: 0.6825WARNING:tensorflow:Your input ran out of data; interrupting training. Make sure that your dataset or generator can gene
687/687 [==============================] - 12s 8ms/step - loss: 0.6420 - accuracy: 0.6830 - val_loss: 0.5749 - val_accuracy: 0.7184
Epoch 2/5
687/687 [==============================] - 3s 5ms/step - loss: 0.5617 - accuracy: 0.7304
Epoch 3/5
687/687 [==============================] - 4s 6ms/step - loss: 0.5161 - accuracy: 0.7607
Epoch 4/5
687/687 [==============================] - 5s 7ms/step - loss: 0.4701 - accuracy: 0.7890
Epoch 5/5
619/687 [==========================>...] - ETA: 0s - loss: 0.4416 - accuracy: 0.8054WARNING:tensorflow:Your input ran out of data; interrupting training. Make sure that your dataset or generator can gene
687/687 [==============================] - 5s 7ms/step - loss: 0.4410 - accuracy: 0.8056
accuracy: 70.81%
Epoch 1/5
```



Results

Successfully performed classification on the above datasets and the corresponding models have been improvised by using K folds and Grid search.