

# image-classfication-randomforrest

November 7, 2023

```
[ ]: !pip install kaggle --upgrade --quiet
```

```
[ ]: import os
from getpass import getpass
os.environ['KAGGLE_USERNAME'] = getpass('Enter KAGGLE_USERNAME secret value: ')
os.environ['KAGGLE_KEY'] = getpass('Enter KAGGLE_KEY secret value: ')
↪ #169fffc674d7757499da3c90e3a33933
```

Enter KAGGLE\_USERNAME secret value: .....

Enter KAGGLE\_KEY secret value: .....

```
[ ]: !kaggle datasets download -d paultimothymooney/chest-xray-pneumonia
```

Downloading chest-xray-pneumonia.zip to /content

100% 2.29G/2.29G [00:22<00:00, 25.9MB/s]

100% 2.29G/2.29G [00:22<00:00, 109MB/s]

```
[ ]: import zipfile
import os

zip_file_path = '/content/chest-xray-pneumonia.zip'

extract_folder = '/content/chest-xray-pneumonia'

with zipfile.ZipFile(zip_file_path, 'r') as zip_ref:
    zip_ref.extractall(extract_folder)

extracted_files = os.listdir(extract_folder)
print("Extracted files:", extracted_files)
```

Extracted files: ['chest\_xray']

```
[ ]: import cv2
import numpy as np
```

```

labels = ['PNEUMONIA', 'NORMAL']
img_size = 150
def get_training_data(data_dir):
    data = []
    for label in labels:
        path = os.path.join(data_dir, label)
        class_num = labels.index(label)
        for img in os.listdir(path):
            try:
                img_arr = cv2.imread(os.path.join(path, img), cv2.
↳IMREAD_GRAYSCALE)
                resized_arr = cv2.resize(img_arr, (img_size, img_size))
                data.append([resized_arr, class_num])
            except Exception as e:
                print(e)
    return np.array(data)

```

```

[ ]: train = get_training_data('/content/chest-xray-pneumonia/chest_xray/train')
test = get_training_data('/content/chest-xray-pneumonia/chest_xray/test')
val = get_training_data('/content/chest-xray-pneumonia/chest_xray/val')

```

<ipython-input-5-9725ff751ebe>:18: VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences (which is a list-or-tuple of lists-or-tuples-or ndarrays with different lengths or shapes) is deprecated. If you meant to do this, you must specify 'dtype=object' when creating the ndarray.

```

    return np.array(data)

```

```

[ ]: x_train = []
y_train = []

x_val = []
y_val = []

x_test = []
y_test = []

for feature, label in train:
    x_train.append(feature)
    y_train.append(label)

for feature, label in test:
    x_test.append(feature)
    y_test.append(label)

for feature, label in val:
    x_val.append(feature)

```

```
y_val.append(label)
```

```
[ ]: from sklearn.model_selection import train_test_split
      from sklearn.metrics import accuracy_score
      from sklearn.preprocessing import StandardScaler
```

```
scaler = StandardScaler()
```

```
x_train = np.array(x_train) / 255
```

```
x_val = np.array(x_val) / 255
```

```
x_test = np.array(x_test) / 255
```

```
x_train = x_train.reshape(x_train.shape[0], -1)
```

```
y_train = np.array(y_train)
```

```
x_val = x_val.reshape(x_val.shape[0], -1)
```

```
y_val = np.array(y_val)
```

```
x_test = x_test.reshape(x_test.shape[0], -1)
```

```
y_test = np.array(y_test)
```

```
[ ]: from sklearn.model_selection import GridSearchCV
      from sklearn.ensemble import RandomForestClassifier
      from sklearn.preprocessing import StandardScaler
      from sklearn.pipeline import Pipeline
      from sklearn.metrics import accuracy_score
```

```
pipeline = Pipeline([
    ('scaler', StandardScaler()),
    ('random_forest', RandomForestClassifier())
])
```

```
param_grid = {
    'random_forest__n_estimators': [50, 100, 150],
    'random_forest__criterion': ['gini', 'entropy'],
    'random_forest__max_depth': [None, 10, 20, 30]
}
```

```
grid_search = GridSearchCV(pipeline, param_grid, cv=5)
```

```
grid_search.fit(x_train, y_train)
```

```
best_params = grid_search.best_params_
```

```
best_model = grid_search.best_estimator_
```

```

print("Best Parameters:", best_params)

predictions = best_model.predict(x_test)

accuracy = accuracy_score(y_test, predictions)
print("Accuracy:", accuracy)

```

Best Parameters: {'random\_forest\_\_criterion': 'gini',  
 'random\_forest\_\_max\_depth': 20, 'random\_forest\_\_n\_estimators': 100}  
 Accuracy: 0.7516025641025641

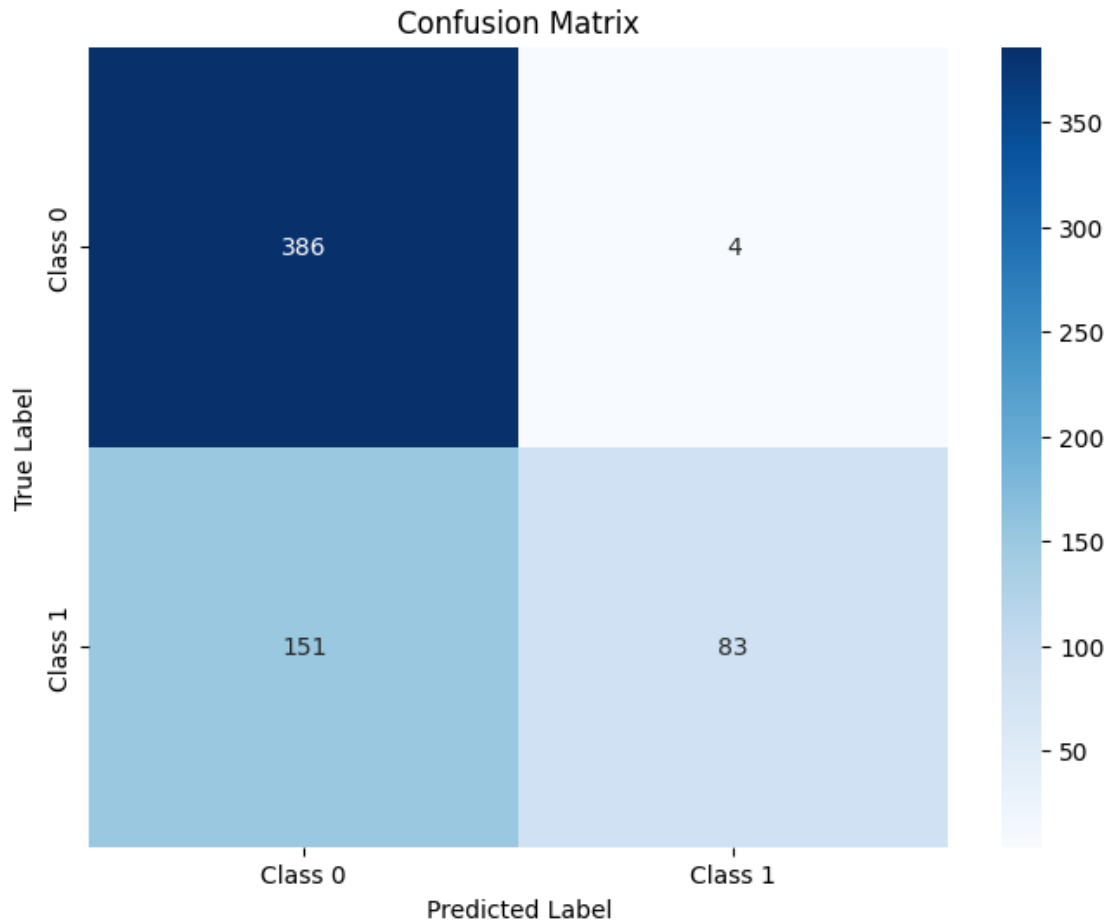
```

[ ]: from sklearn.metrics import confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt

conf_matrix = confusion_matrix(y_test, predictions)

plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=['Class_
↪0', 'Class 1'], yticklabels=['Class 0', 'Class 1'])
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.title('Confusion Matrix')
plt.show()

```



```
[ ]: from sklearn.metrics import roc_curve, auc
import matplotlib.pyplot as plt

fpr, tpr, _ = roc_curve(y_test, predictions)
roc_auc = auc(fpr, tpr)

plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, color='darkorange', lw=2, label='ROC curve (area = {:.2f})'.
        format(roc_auc))
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc='lower right')
```

```
plt.show()
```

