

Pattern recognition project

Lung and Colon Cancer Histopathological

Done by : karim ali

Supervised by:DR.Mohamed mahfouz

Eng.Abdelrahman ezzeldin

Dataset description:

This dataset contains 10,000 histopathological images with 2 classes. All images are 768 x 768 pixels in size and are in jpeg file format.

There are two classes in the dataset, each with 5,000 images, being:

- Lung adenocarcinoma
- Colon adenocarcinoma

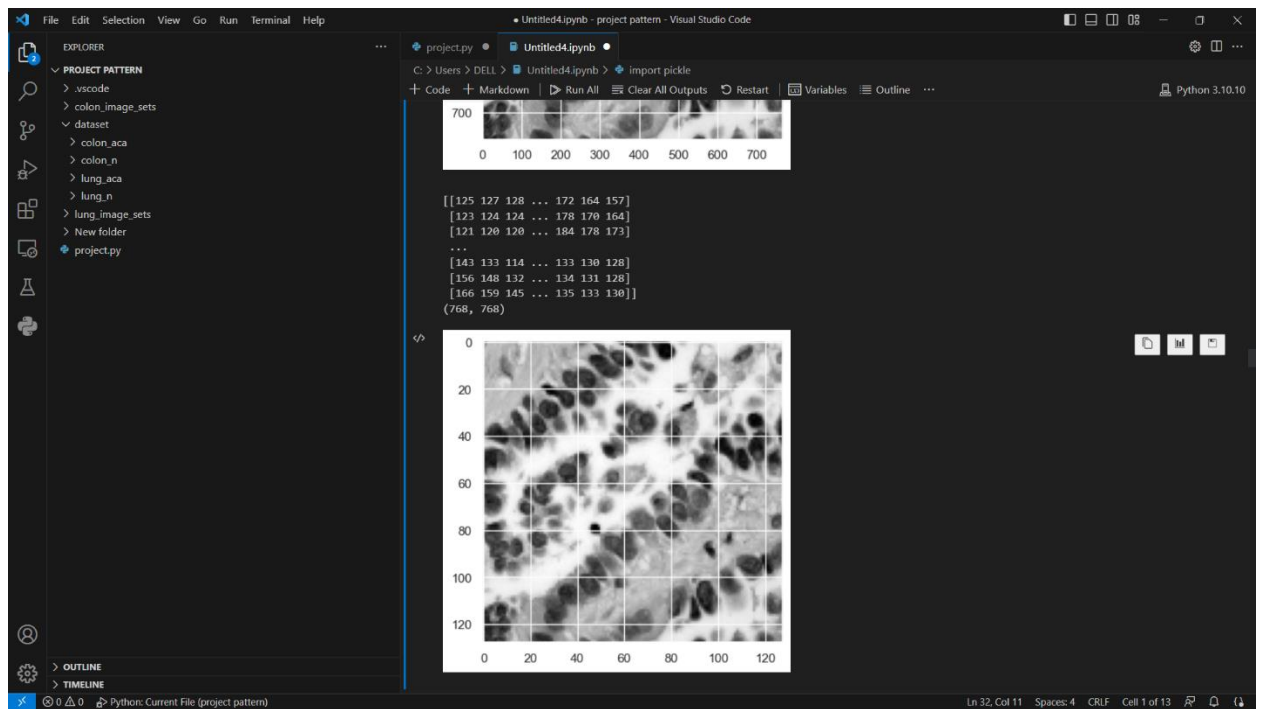
Results:

1)Displayed the first 10 images from each category in the dataset and put them in array after that resized them.

The image array represents the pixel values of the image in grayscale.

The shape of the image array is also printed, indicating the dimensions of the image (height, width).

the output:



2) Training dataset:

1-it iterates over categories, reads images, resizes them, and adds them along with their class labels to the **training data** list.

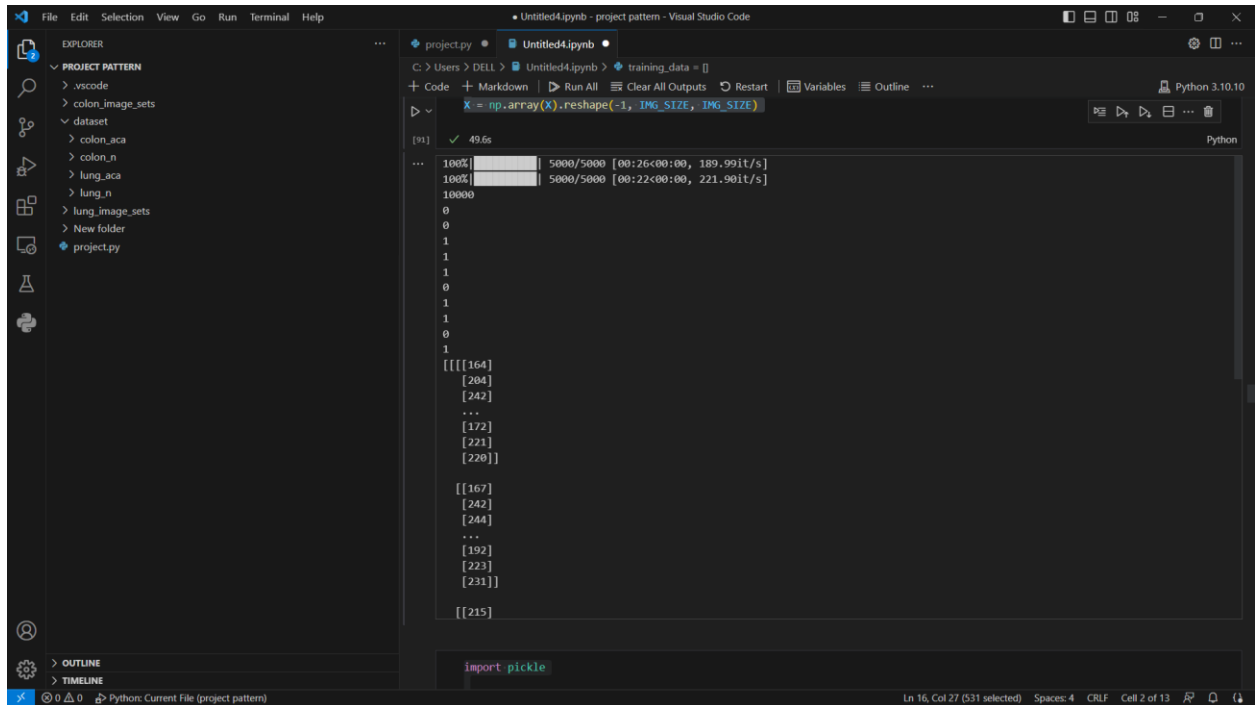
2-The length of the **training data** list is printed to determine the total number of samples.

3-The **training data** list is shuffled randomly, and the class labels of the first 10 samples are printed.

4-The features (resized images) and labels are extracted from the **training data** list and stored in separate lists, **X** and **y**.

5-The **X** list is reshaped to match the expected input shape for a machine learning model.

The output:



The screenshot shows a Jupyter Notebook interface within Visual Studio Code. The Explorer panel on the left displays a project structure with folders like 'dataset', 'lung_image_sets', and 'lung_n'. The main editor area shows a code cell with the following code:

```
X = np.array(X).reshape(-1, IMG_SIZE, IMG_SIZE)
```

The output of the cell is displayed below the code, showing a progress bar at 100% and a list of data points. The first few data points are:

```
[[[164  
[204  
[242  
...  
[172  
[221  
[220]]  
  
[[167  
[242  
[244  
...  
[192  
[223  
[231]]  
  
[[215]]
```

The model is trained using the training data (X train, y train) and validated using the validation data (X value, y value) for 10 epochs.

The training history, including loss and accuracy values, is stored in the history variable.

```
model.compile(optimizer='adam',loss='sparse_categorical_crossentropy',metrics=['accuracy'])
```

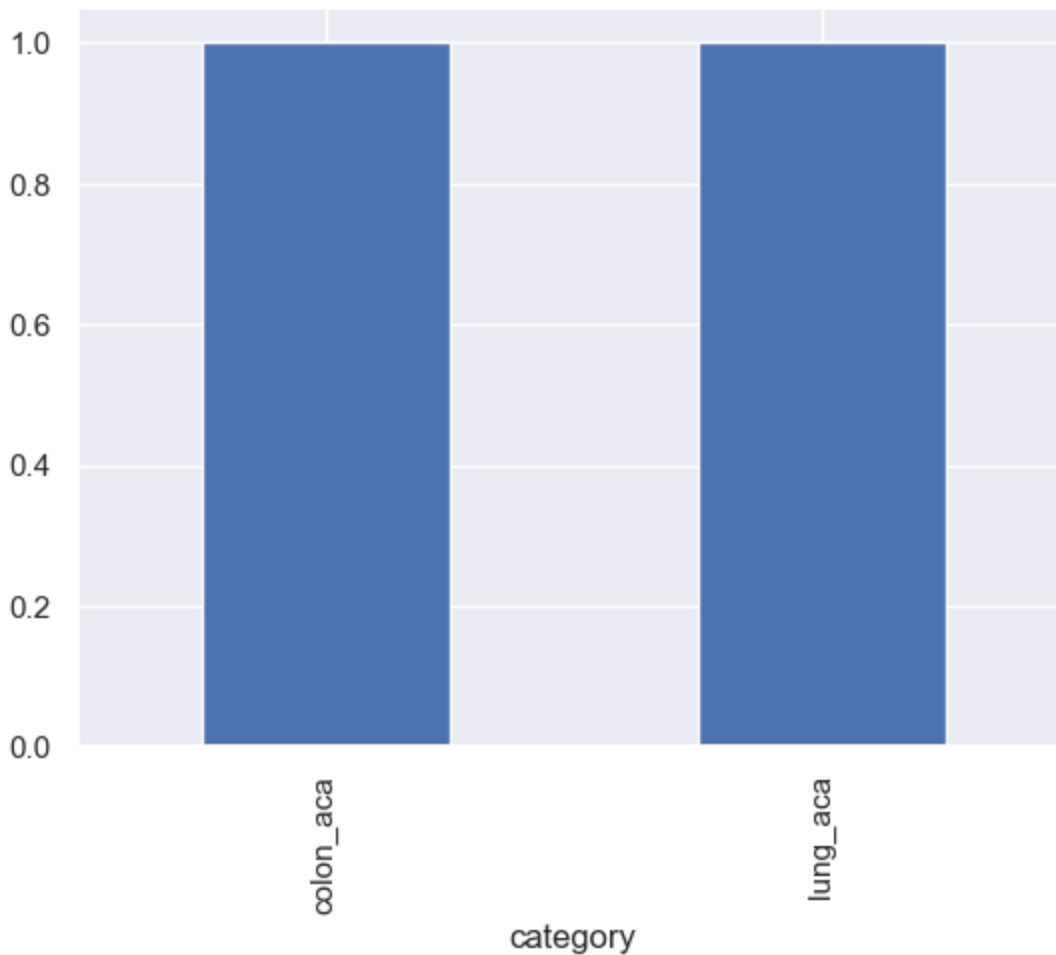
```
history = model.fit(X_train, y_train, validation_data=(X_val, y_val), epochs=10)
```

```
ch 1/10 [=====] - 11s 41ms/step - loss: 0.7014 - accuracy: 0.4992 - val_loss: 0.6931 - val_accuracy: 0.5035
ch 2/10 [=====] - 10s 39ms/step - loss: 0.6990 - accuracy: 0.4969 - val_loss: 0.7027 - val_accuracy: 0.5035
ch 3/10 [=====] - 10s 39ms/step - loss: 0.6973 - accuracy: 0.5131 - val_loss: 0.6952 - val_accuracy: 0.5035
ch 4/10 [=====] - 10s 39ms/step - loss: 0.6980 - accuracy: 0.5084 - val_loss: 0.7032 - val_accuracy: 0.5035
ch 5/10 [=====] - 10s 39ms/step - loss: 0.6977 - accuracy: 0.4949 - val_loss: 0.6971 - val_accuracy: 0.4965
ch 6/10 [=====] - 10s 39ms/step - loss: 0.7004 - accuracy: 0.4941 - val_loss: 0.6936 - val_accuracy: 0.4965
ch 7/10 [=====] - 10s 39ms/step - loss: 0.6960 - accuracy: 0.4946 - val_loss: 0.6981 - val_accuracy: 0.4965
ch 8/10 [=====] - 10s 39ms/step - loss: 0.6963 - accuracy: 0.5021 - val_loss: 0.6948 - val_accuracy: 0.4965
ch 9/10 [=====] - 10s 39ms/step - loss: 0.6950 - accuracy: 0.5036 - val_loss: 0.6931 - val_accuracy: 0.5035
ch 10/10 [=====] - 10s 39ms/step - loss: 0.6959 - accuracy: 0.5049 - val_loss: 0.6959 - val_accuracy: 0.4965
```

```
import pandas as pd
df = pd.DataFrame({
    'filename': DATADIR,
    'category': CATEGORIES
```

Then displayed plot bar graph to visualize the distribution of categories. Each category will be represented on the x-axis, and the count of images in each category will be represented on the y-axis.

The output:



Then done the feature extraction classification with CNN model with several convolutional layers, pooling layers, and fully connected layers for image classification. The model architecture is designed to extract features from the input images and make predictions on the classes.

Output:

Model: "sequential_17"

Layer (type)

Output Shape

Param #

=====

conv2d_6 (Conv2D) (None, 126, 126, 32) 896

batch_normalization_8 (Batch Normalization) (None, 126, 126, 32) 128

max_pooling2d_6 (MaxPooling2D) (None, 63, 63, 32) 0

dropout_8 (Dropout) (None, 63, 63, 32) 0

conv2d_7 (Conv2D) (None, 61, 61, 64) 18496

batch_normalization_9 (Batch Normalization) (None, 61, 61, 64) 256

max_pooling2d_7 (MaxPooling2D) (None, 30, 30, 64) 0

dropout_9 (Dropout)	(None, 30, 30, 64)	0
conv2d_8 (Conv2D)	(None, 28, 28, 128)	73856
batch_normalization_10 (Batch Normalization)	(None, 28, 28, 128)	512
max_pooling2d_8 (MaxPooling2D)	(None, 14, 14, 128)	0
dropout_10 (Dropout)	(None, 14, 14, 128)	0
flatten_16 (Flatten)	(None, 25088)	0
dense_46 (Dense)	(None, 512)	12845568
batch_normalization_11 (Batch Normalization)	(None, 512)	2048
dropout_11 (Dropout)	(None, 512)	0

dense_47 (Dense) (None, 2) 1026

=====

=====

Total params: 12,942,786

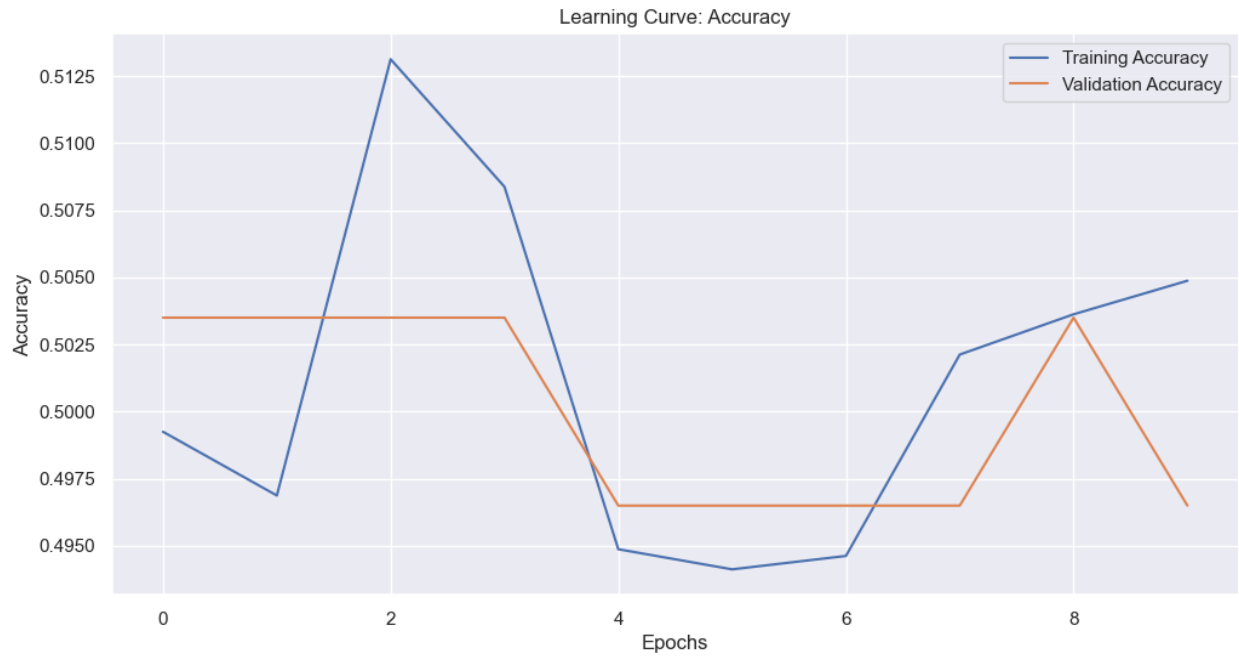
Trainable params: 12,941,314

Non-trainable params: 1,472

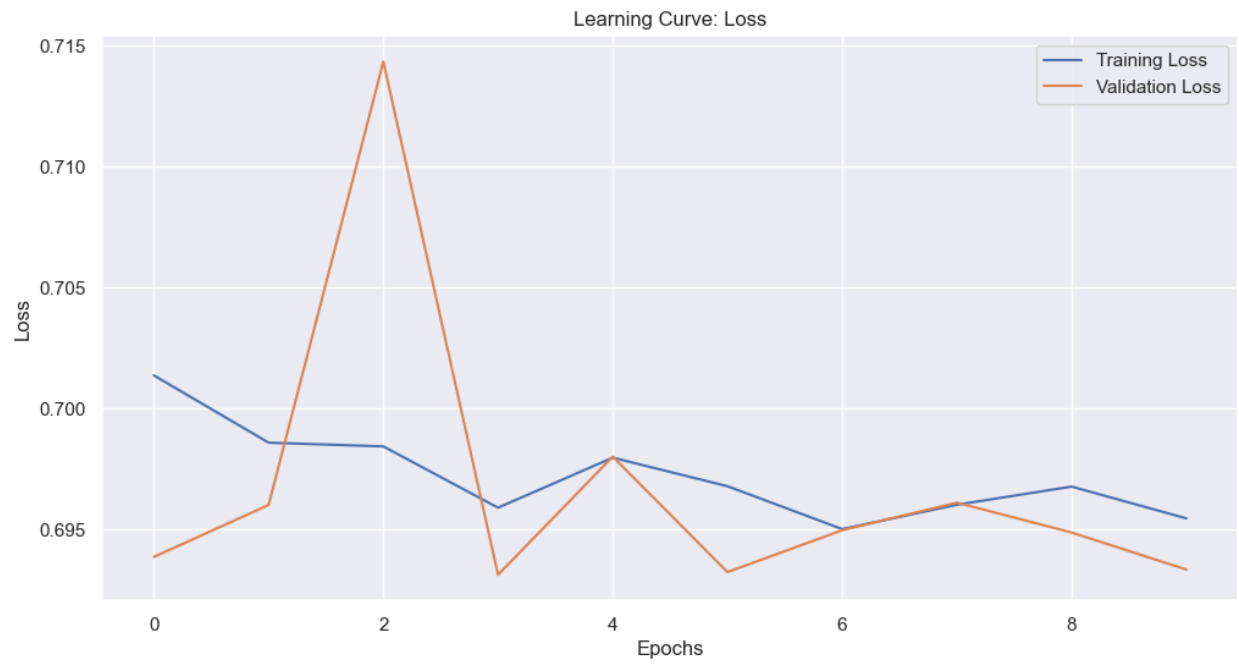
Then used to plot the learning curves for the training and validation loss, as well as the training and validation accuracy.

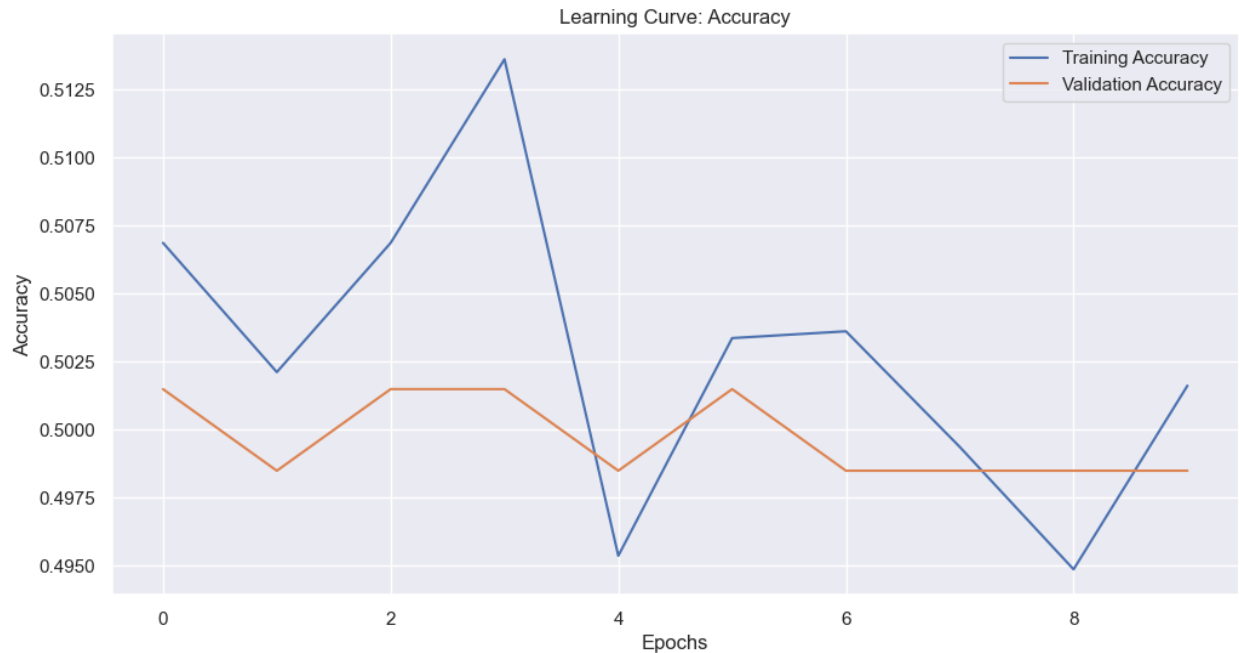
Output: **model 1**





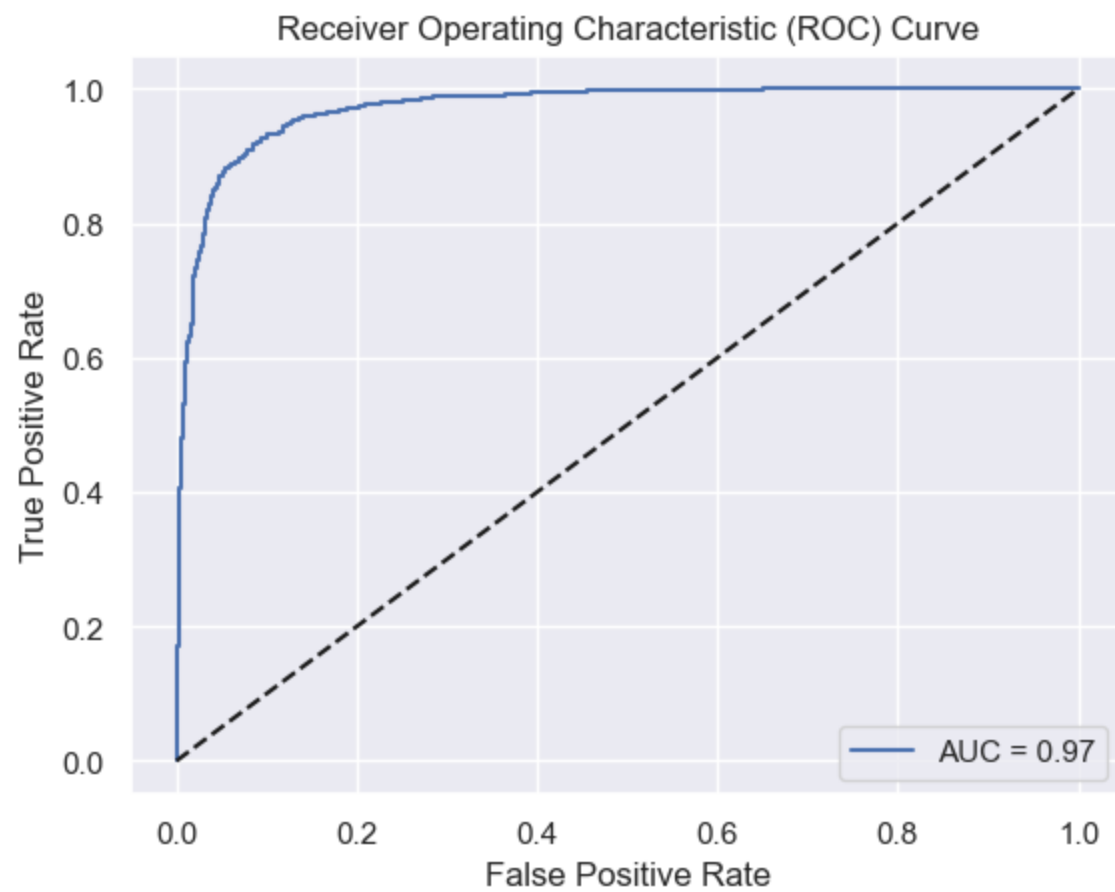
Model 2:



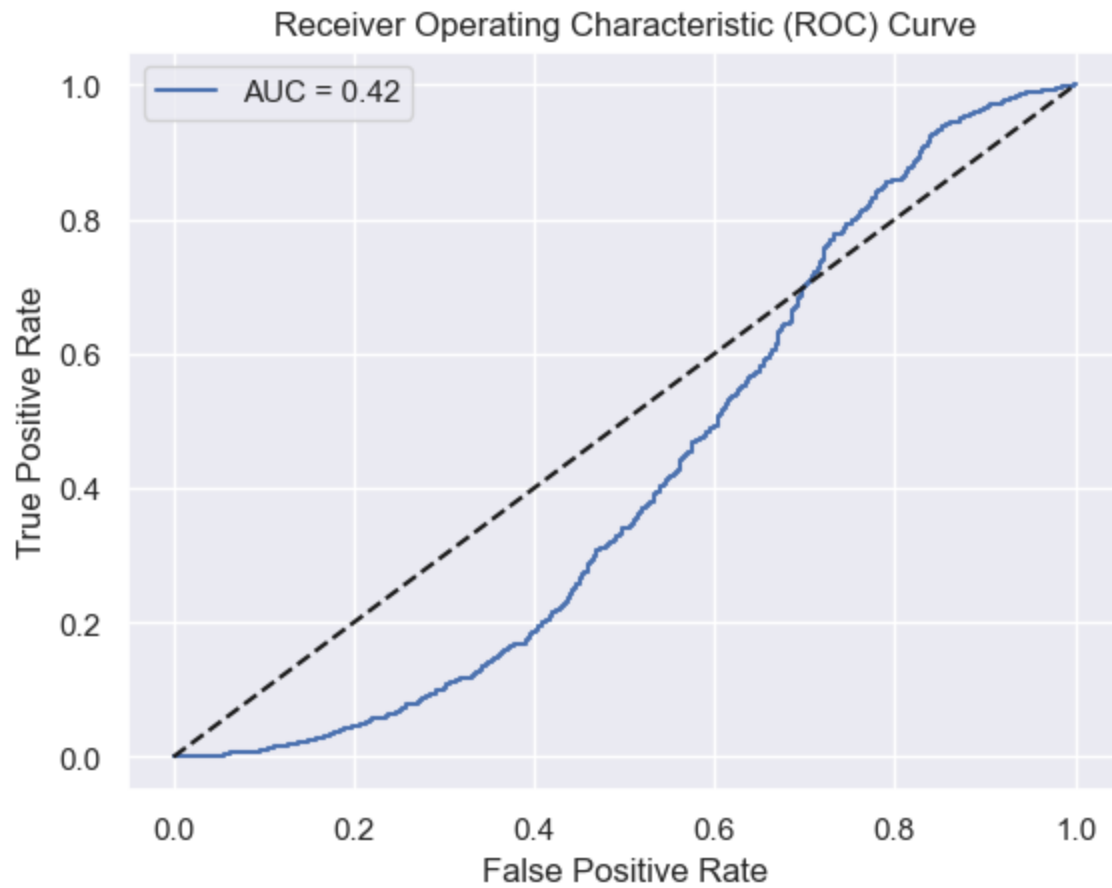


Then evaluated the performance of a binary classification model using the Receiver Operating Characteristic (ROC) curve and the Area Under the Curve (AUC) metric, the curve illustrates the trade-off between the true positive rate and the false positive rate at different classification threshold.

Model 1:

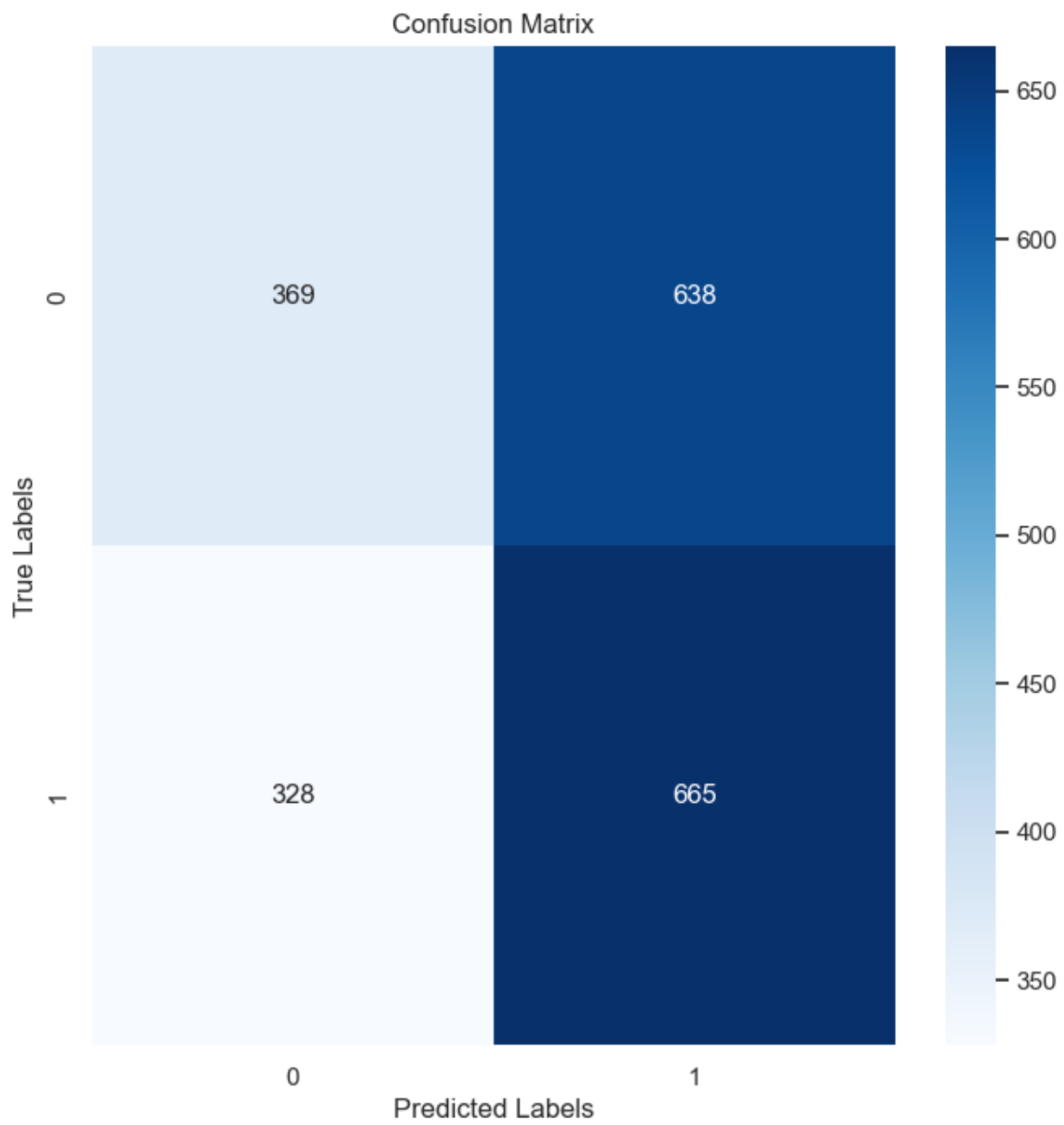


Model 2:



Then used confusion matrix for the performance of a classification algorithm.

Model 1:



Model 2:

