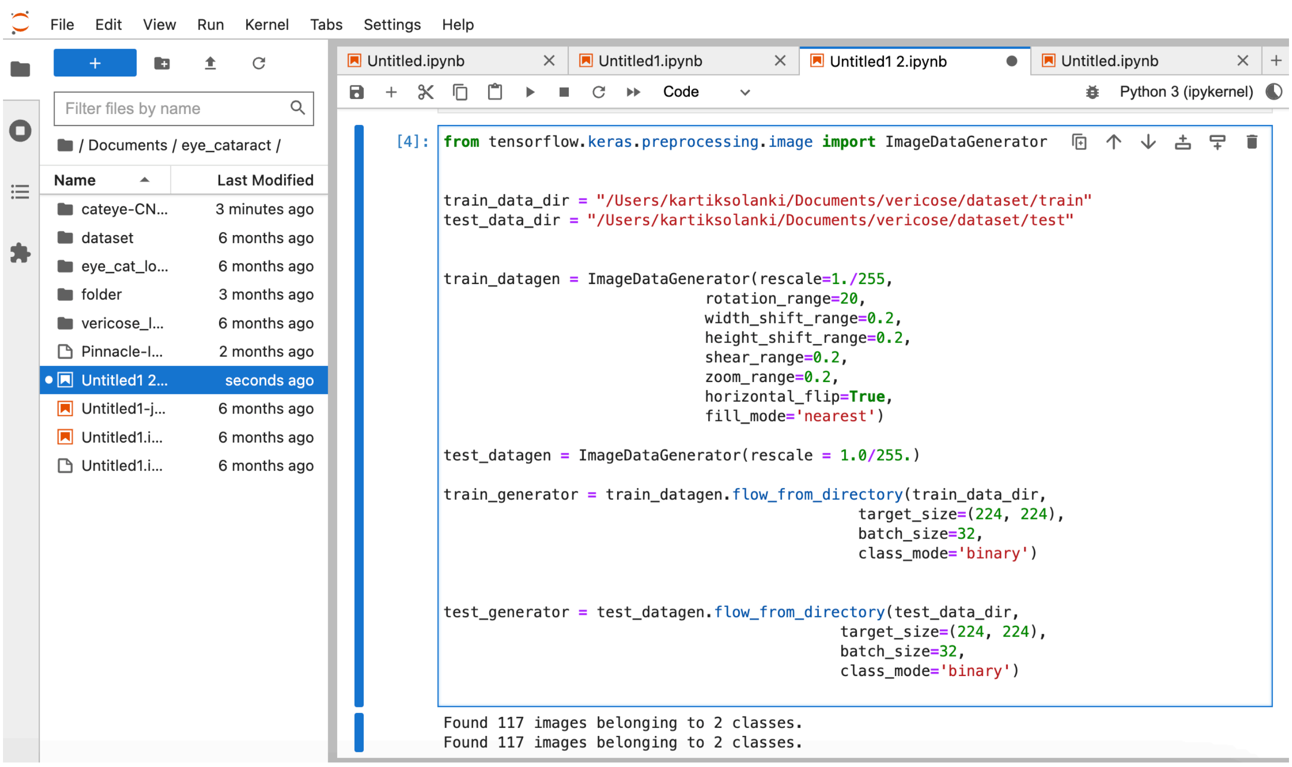
VARICOSE VEINS DETECTION USING CNN

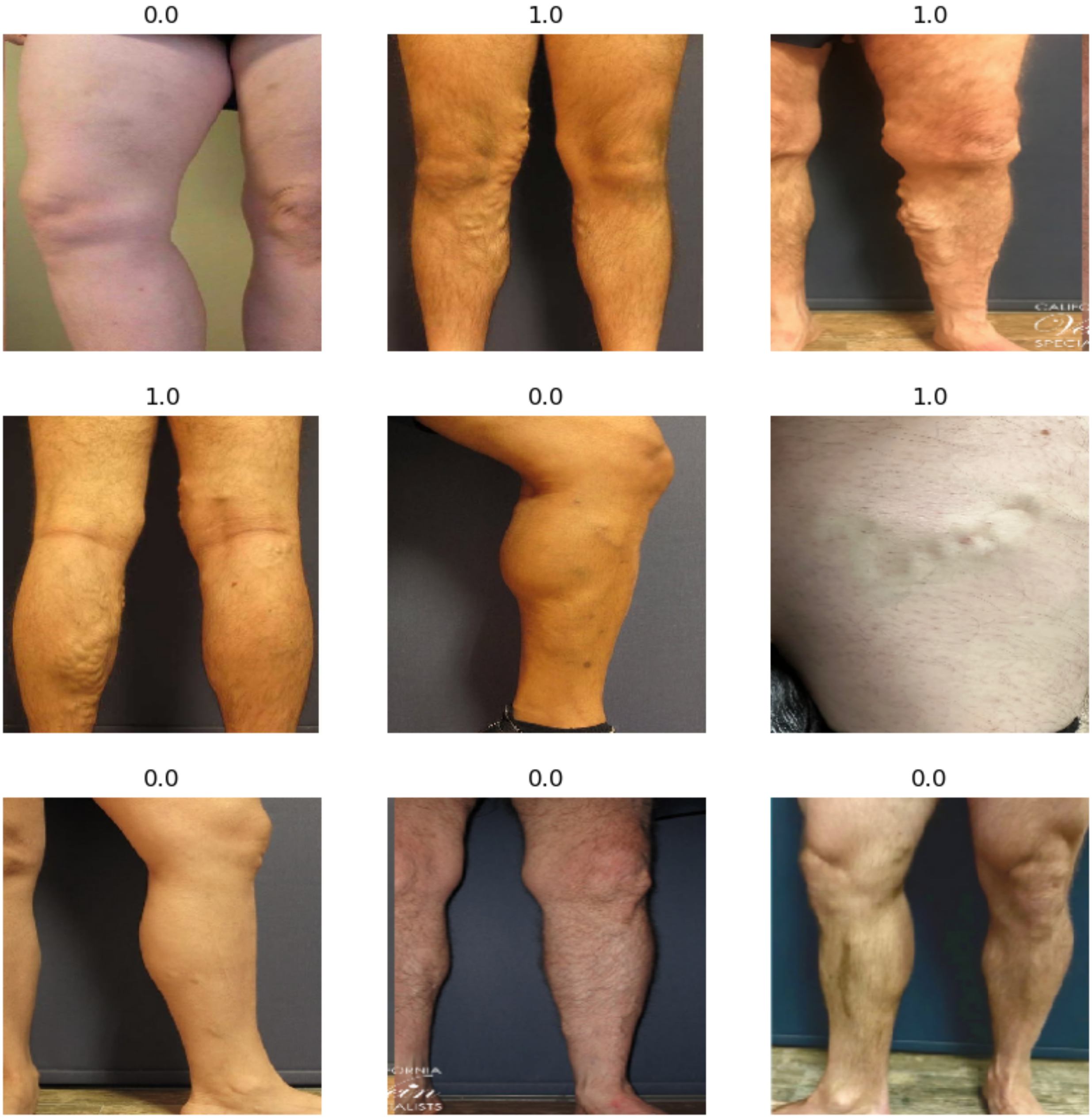
```
import numpy as np
import matplotlib.pyplot as plt
import os
import cv2
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Dense, Flatten
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.preprocessing import image_dataset_from_directory

DATADIR = "/Users/kartiksolanki/Documents/vericose/dataset/Train"
```


```
CATEGORIES = ["normal", "vericose"]

fig, axes = plt.subplots(1, len(CATEGORIES), figsize=(10, 5))  # Create
subplots

for i, category in enumerate(CATEGORIES):
    path = os.path.join(DATADIR, category)

    for img in os.listdir(path):
        img_array = cv2.imread(os.path.join(path, img))
        img_rgb = cv2.cvtColor(img_array, cv2.COLOR_BGR2RGB)  # Convert
BGR to RGB

        axes[i].imshow(img_rgb)
        axes[i].set_title(f"First image from {category}")
```

```
        axes[i].axis('off')  # Hide axis

        break  # Break the loop once the first image is printed

plt.show()



def create_data(path):
    data = []
    for category in CATEGORIES:
        path = os.path.join(DATADIR, category)
        class_num = CATEGORIES.index(category)
        for img in os.listdir(path):
            try:
                files = glob.glob(path+"/"+category+"/*")
                for f in files:
                    img_array = cv2.imread(f)
                    new_array = cv2.resize(img_array, (IMG_SIZE,
IMG_SIZE))
                    data.append([np.array(img_array),
CATEGORIES.index(category)])
            except Exception as e:
                pass
    np.random.shuffle(data)
    return data
```



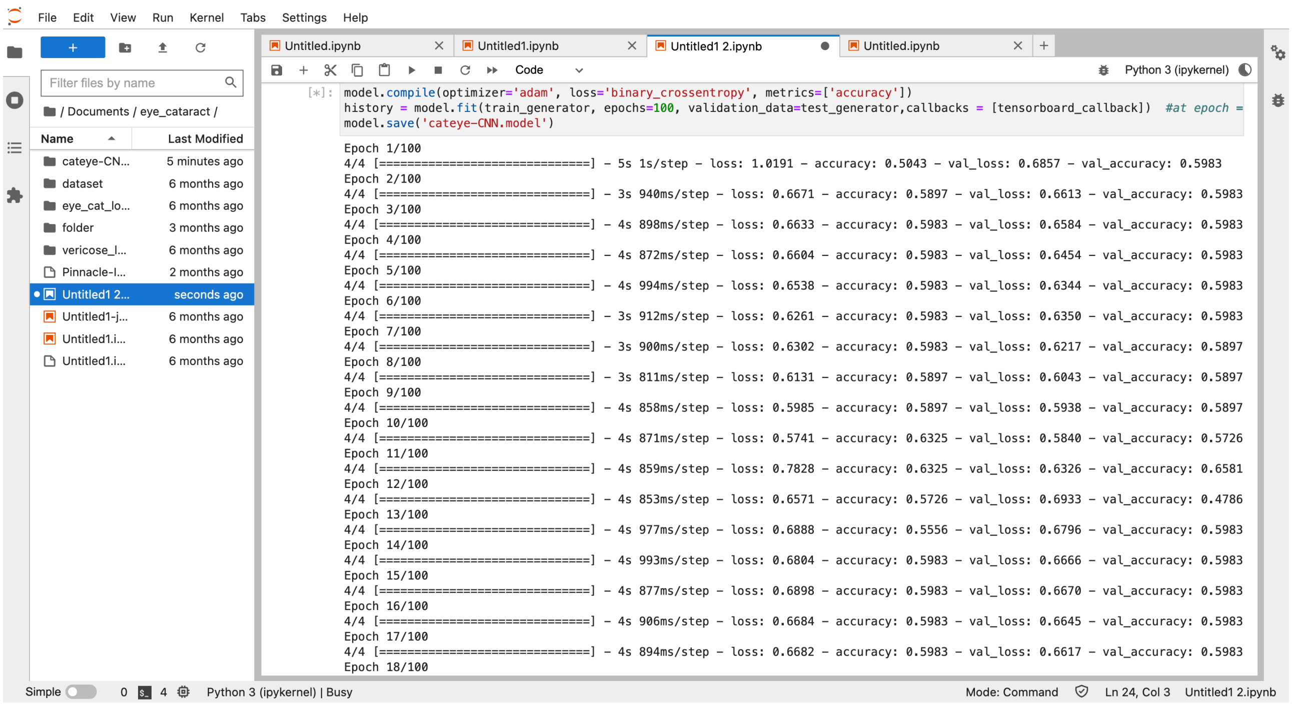

```
import matplotlib.pyplot as plt
```




```
images, labels = next(test_generator)

plt.figure(figsize=(10, 10))

for i in range(9):
    plt.subplot(3, 3, i + 1)
    plt.imshow(images[i])
    plt.title(str(labels[i]) if labels[i] == 0 else str(labels[i]))
    plt.axis('off')

plt.show()
```

```
#Implementing convulational Neural Network
model = keras.Sequential([
```

```
    layers.Conv2D(32, (3, 3), input_shape=(224, 224, 3),
activation='relu'),
    layers.MaxPooling2D(2, 2),

    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D(2, 2),

    layers.Conv2D(128, (3, 3), activation='relu'),
    layers.MaxPooling2D(2, 2),

    layers.Conv2D(256, (3, 3), activation='relu'),#added a new
convulational layer
    layers.MaxPooling2D(2, 2),

    layers.Flatten(),
    layers.Dense(256, activation='relu'),#raised the dense layers to 256

    layers.Dropout(0.5),
    layers.Dense(1, activation='sigmoid')


])
```

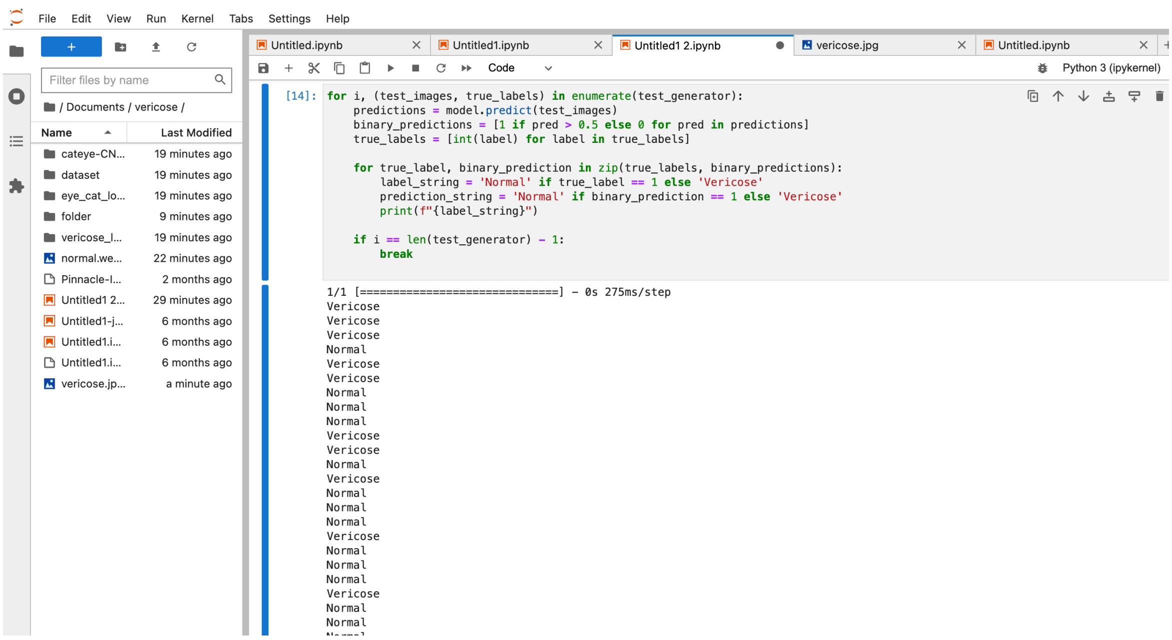



```
test_loss, test_acc = model.evaluate(test_generator)
print(f"Test accuracy: {test_acc}")
```

```
4/4 [==============================] - 1s 206ms/step - loss: 0.5581 - accuracy: 0.6838
Test accuracy: 0.6837607026100159
```

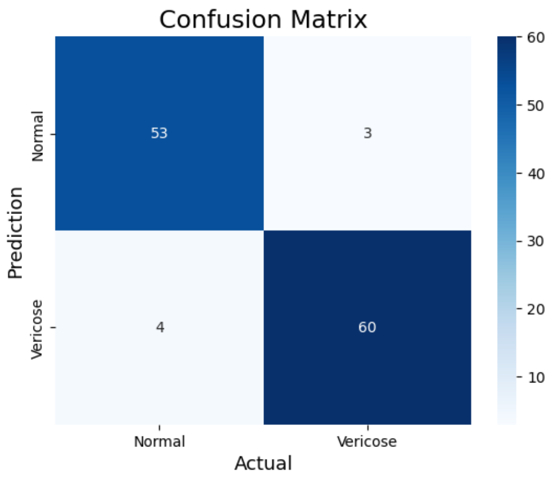
```
import matplotlib.pyplot as plt

epochs = range(1, 101)
plt.figure(figsize=(10, 5))
plt.title("Loss vs Accuracy of Model")
plt.plot(epochs, history.history['loss'][:100], label='Loss')
plt.plot(epochs, history.history['accuracy'][:100], label='Accuracy')
plt.grid()
plt.xlabel("Epochs")
plt.grid()
plt.legend()
plt.show()
```

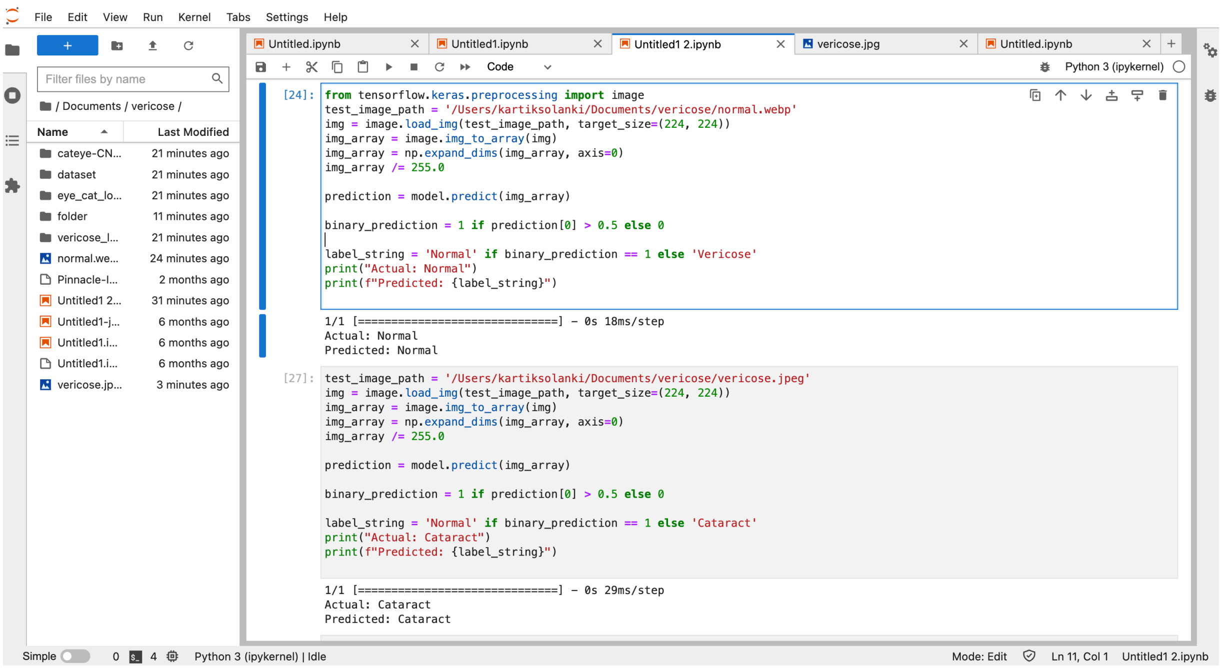
```
import matplotlib.pyplot as plt

epochs = range(1, 101)
plt.figure(figsize=(10, 5))
plt.title("Loss vs Accuracy of Model")
plt.plot(epochs, history.history['loss'][:100], label='Loss')
plt.plot(epochs, history.history['accuracy'][:100], label='Accuracy')
plt.grid()
plt.xlabel("Epochs")
plt.grid()
plt.legend()
plt.show()
```



```
#Import the necessary libraries
import numpy as np
from sklearn.metrics import confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt

#Create the NumPy array for actual and predicted labels.
actual = np.array(

['Vericose','Vericose','Vericose','Vericose','Vericose','Vericose','Veric
ose','Vericose','Vericose','Vericose','Vericose','Vericose','Vericose','V
ericose','Vericose','Vericose','Vericose','Vericose','Vericose','Vericose
','Vericose','Vericose','Vericose','Vericose','Vericose','Vericose','Veri
cose','Vericose','Vericose','Vericose','Vericose','Vericose','Vericose','
Vericose','Vericose','Vericose','Vericose','Vericose','Vericose','Vericos
e','Vericose',
'Vericose','Vericose','Vericose','Vericose','Vericose','Vericose','Verico
se','Vericose','Vericose','Vericose','Vericose','Vericose','Vericose','Ve
ricose','Vericose','Vericose','Vericose','Vericose','Vericose','Vericose'
,'Vericose','Vericose','Vericose','Normal','Normal','Normal','Normal','No
rmal','Normal','Normal','Normal','Normal','Normal','Normal','Normal','Nor
mal','Normal','Normal','Normal','Normal','Normal','Normal','Normal',
'Normal','Normal','Normal','Normal', 'Normal','Normal','Normal','Normal',
'Normal','Normal','Normal','Normal','Normal','Normal','Normal','Normal','
Normal','Normal','Normal','Normal','Normal','Normal','Normal','Normal','N
ormal','Normal','Normal','Normal','Normal','Normal','Normal','Normal',
'Normal','Normal','Normal','Normal'])
pred= np.array(

['Vericose','Vericose','Vericose','Vericose','Vericose','Vericose','Veric
ose','Vericose','Vericose','Vericose','Vericose','Vericose','Vericose','V
ericose','Vericose','Vericose','Vericose','Vericose','Vericose','Vericose
','Vericose','Vericose','Vericose','Vericose','Vericose','Vericose','Veri
cose','Vericose','Vericose','Vericose','Vericose','Vericose','Vericose','
Vericose','Vericose','Vericose','Vericose','Vericose','Vericose','Vericos
e','Vericose',
'Vericose','Vericose','Vericose','Vericose','Vericose','Vericose','Verico
se','Vericose','Vericose','Vericose','Vericose','Vericose','Vericose','Ve
ricose','Vericose','Vericose','Vericose','Vericose','Vericose','Normal','
Normal','Normal','Normal','Normal','Normal','Normal','Normal','Normal','N
ormal','Normal','Normal','Normal','Normal','Normal','Normal','Normal','No
rmal','Normal','Normal','Normal','Normal','Normal','Normal',
'Normal','Normal','Normal','Normal', 'Normal','Normal','Normal','Normal',
'Normal','Normal','Normal','Normal','Normal','Normal','Normal','Normal','
Normal','Normal','Normal','Normal','Normal','Normal','Normal','Normal','N
ormal','Normal','Normal','Normal','Normal','Normal','Normal','Normal',
'Normal','Vericose','Vericose','Vericose'])
#compute the confusion matrix.
cm = confusion_matrix(actual,pred)

#Plot the confusion matrix.  ''
sns.heatmap(cm,
            annot=True,
           fmt='d', cmap='Blues' ,
            xticklabels=['Normal','Vericose'],
            yticklabels=['Normal','Vericose'])
```

```
plt.ylabel('Prediction',fontsize=13)
plt.xlabel('Actual',fontsize=13)
plt.title('Confusion Matrix',fontsize=17)
plt.show()
```





(varicose.jpeg)

(normal.webp)