


Image Classification Program for Acute Lymphoblastic Leukemia

By Sayaka Minegishi
For the Advanced Data Science Capstone
Project

A dark blue diagonal gradient bar that starts from the bottom left corner and extends towards the top right corner, covering the lower half of the slide.

Section 1: To the Stakeholders

This section explores:

The Dataset, Use Case, Solution to the Use Case

A dark blue diagonal gradient bar that starts from the bottom left and extends towards the top right, covering the lower half of the slide.

The Data Set

- Dataset consists of bone marrow aspirate smear images from individuals who test positive for acute lymphoblastic leukemia and those who test negative for it.
- Open dataset from kaggle.com, which has been uploaded by the user Nikhil sharma.
- The image data is created by Fabio Scott from the Department of Information Technology at the University of Milan.

The Use Case

- The project utilizes the Leukemia Dataset from Kaggle to make an image classification program that classifies whether a patient has acute lymphoblastic leukemia, given an image of their bone marrow aspirate smear.

The Solution to the Use Case

- The solution to my use case is documented in my Jupyter Notebook, where I built, tested and deployed my program.
- I formed two image classification models: one using deep learning and one that uses support vector machine (SVM).
- My classification program has been deployed as a PDF report of my Jupyter notebook, which will be handed to the stakeholders.

How the Deployed Product Appears

6/14/2021

capstoneproject.model_deployment.Python.v3

Advanced Data Science Capstone Project - Leukemia Image Classification

BY Sayaka Minegishi June 2021

Use Case

This project will utilize the Leukemia Dataset uploaded to Kaggle by Nikhil Sharma to make an image classification program that classifies whether a patient has acute lymphoblastic leukemia, given an image of their bone marrow aspirate smear.

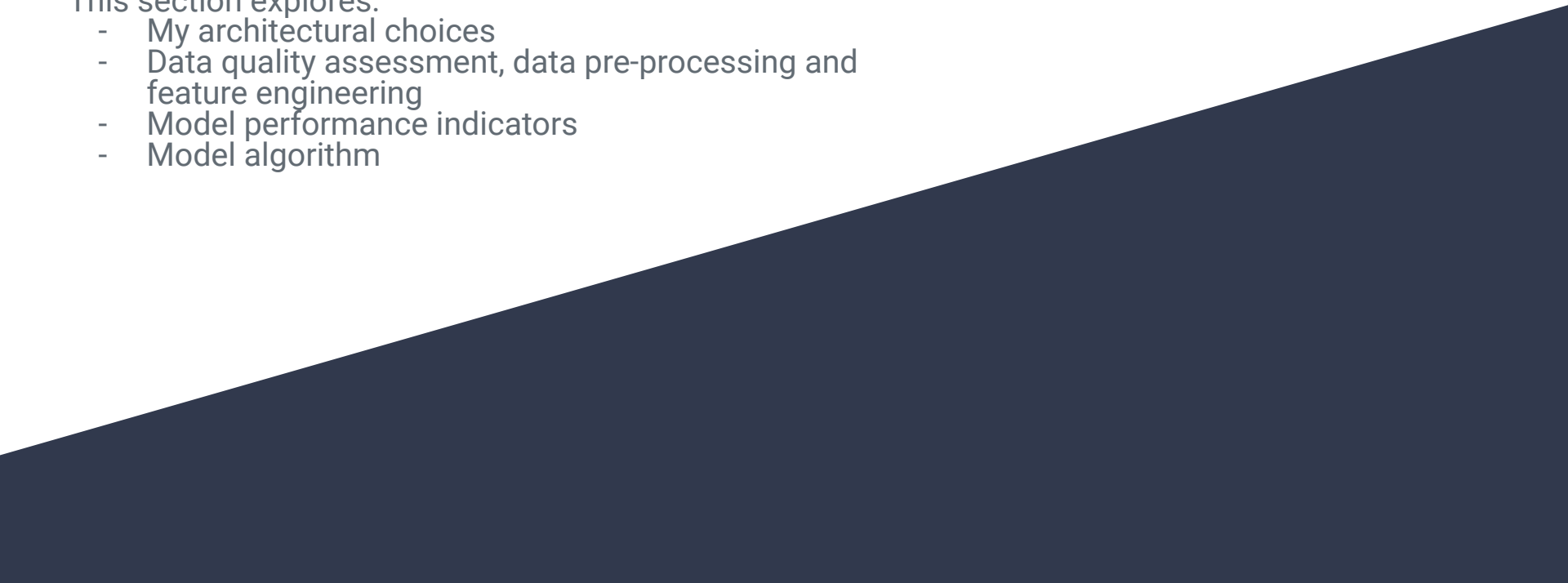
First, we will install the relevant packages

```
In [1]: pip install opencv-python-headless

/opt/conda/envs/Python-3.7-main/lib/python3.7/site-packages/secretstorage/dhcrypto.py:16: CryptographyDeprecationWarning: int_from_bytes is deprecated, use int.from_bytes instead
  from cryptography.utils import int_from_bytes
/opt/conda/envs/Python-3.7-main/lib/python3.7/site-packages/secretstorage/util.py:25: CryptographyDeprecationWarning: int_from_bytes is deprecated, use int.from_bytes instead
  from cryptography.utils import int_from_bytes
Requirement already satisfied: opencv-python-headless in /opt/conda/envs/Python-3.7-main/lib/python3.7/site-packages (4.5.2.54)
Requirement already satisfied: numpy>=1.14.5 in /home/wsuser/.local/lib/python3.7/site-packages (from opencv-python-headless) (1.19.5)
Note: you may need to restart the kernel to use updated packages.
```

Section 2: To Data Science Peers

This section explores:

- My architectural choices
 - Data quality assessment, data pre-processing and feature engineering
 - Model performance indicators
 - Model algorithm
- 
- A dark blue diagonal gradient bar that starts from the bottom left and extends towards the top right, covering the lower half of the slide.

Architectural Choices

- 1. Data Source**
 - I imported my data from Kaggle as they offer open datasets.
- 2. Enterprise Data**
 - Not necessary for my project as I am using open datasets.
- 3. Streaming Analytics**
 - Not necessary for my project as the project does not incorporate real-time stream processing.
- 4. Data Integration**
 - I used Python libraries such as os and cv2.
 - I used os to access my data as the os library allows me to access my data by specifying their directories.
 - I used the cv2 library as it has functions to transform image files into appropriate formats that can later be used as inputs.
- 5. Data Repository**
 - I stored my data in IBM's Cloud Pak Object Storage as the service is free and is large enough to store my full dataset.

Architectural Choices

6. Discovery and Exploration

- I utilized the Matplotlib's pyplot library for visualizing statistical data.
- I used Pillow's image module to visualize my raw data

7. Actionable Insights

- For my deep learning model, I used the evaluate() function to get an accuracy score. For my SVM model, I used F1 score and accuracy_score from sklearn.metrics for model evaluation.

8. Applications/Data Products

- I used IBM Watson Studio's Jupyter Notebook to build my program

9. Security, Information Governance and Systems Management

- The dataset used in this project is stored in IBM Cloud Pak's Object Storage, and hence data is not easily accessible by a third party.

Data Quality Assessment

The dataset contains images of bone marrow aspirate smears from individuals, which depict their lymphocytes.

- Originally, the dataset contained one file that was not an image file ('.DS_Store'). Since this program is an image classification program and should contain only image files, I took note to remove this file from the final dataset.
- There were no missing values or wrong measurements and images in the training set showed properly. All the files are in the the JPEG format.

Data Pre-Processing

- I removed a non-image file called '.DS_Store' from my dataset.
- I also checked that no duplicates images were in the dataset in the data cleansing stage.

We remove the '.DS_Store' file from each directory

```
|: #clean data by removing 'DS_Store' file from each directory

for entryname in os.listdir('./input/training/positive'):
    if (entryname == '.DS_Store'):
        os.remove('./input/training/positive/.DS_Store')

for entryname in os.listdir('./input/training/negative'):
    if (entryname == '.DS_Store'):
        os.remove('./input/training/negative/.DS_Store')

for entryname in os.listdir('./input/validation/positive'):
    if (entryname == '.DS_Store'):
        os.remove('./input/validation/positive/.DS_Store')

for entryname in os.listdir('./input/validation/negative'):
    if (entryname == '.DS_Store'):
        os.remove('./input/validation/negative/.DS_Store')
```

Feature Engineering

1. First, I formed a numpy array that can store the image in an array form and its corresponding label (positive or negative for leukemia) in integer form. I made two of such arrays: one for our training data and one for our validation data.
2. I then stored the image arrays and their labels in the arrays `x_train`, `y_train`, `x_val` and `y_val`. `x_train` and `y_train` stored the images and the labels for the training data, and `x_val` and `y_val` were for the validation data.

Model Algorithm

- I used a Sequential model for my deep learning model. I trained my mode with 12 epochs and a batch size of 32. I included a dropout layer with a rate of 0.4.
- For my non-deep learning model, I created a support vector machine model with a linear kernel.

Model Algorithm from Jupyter Notebook

Sequential model:

```
#model definition
num_classes = 2 #there are 2 classes: one for positive and one for negative

model = Sequential()
model.add(Conv2D(16,3,padding="same", activation="relu", input_shape=(224,224,3)))
model.add(MaxPool2D())
model.add(Conv2D(16, 3, padding="same", activation="relu"))
model.add(MaxPool2D())
model.add(Conv2D(64, 3, padding="same", activation="relu"))
model.add(MaxPool2D())
model.add(Dropout(0.4))
model.add(Flatten())
model.add(Dense(128,activation="relu"))
model.add(Dense(num_classes))
```

We will now compile our model using the 'Adam' optimizer.

```
#compile our model
model.compile(optimizer = 'Adam',
              loss = tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics = ['accuracy'])
```

SVM model:

```
image_data_flattened =[] #input array
target_array = [] #output array
data_directory_training = "./input/training"

for i in labels:
    path_training = os.path.join(data_directory_training, i)
    for img in os.listdir(path_training):
        img_array = imread(os.path.join(path_training, img))
        img_resized = resize(img_array, (224,224,3)) #resize input image
        image_data_flattened.append(img_resized.flatten()) #flatten image
        target_array.append(labels.index(i))
    print(f'loaded category: {i} successfully')

flattened_data = np.array(image_data_flattened)
target_data = np.array(target_array)

df = pd.DataFrame(flattened_data)
df['Target'] = target_data #add column for target (the labels)
x_train_supervised = df.iloc[:,1:-1] #input
y_train_supervised = df.iloc[:,1:-1] #output

loaded category: positive successfully
loaded category: negative successfully
```

```
[32]: #validation data
image_data_flattened_v =[] #input array
target_array_v = [] #output array
data_dir_validation = "./input/validation"

for i in labels:
    path_validation = os.path.join(data_dir_validation, i)
    for img in os.listdir(path_validation):
        img_array = imread(os.path.join(path_validation, img))
        img_resized = resize(img_array, (224,224,3)) #resize input image
        image_data_flattened_v.append(img_resized.flatten()) #flatten image
        target_array_v.append(labels.index(i))
    print(f'loaded category: {i} successfully')

flattened_data_v = np.array(image_data_flattened_v)
target_data_v = np.array(target_array_v)

df_v = pd.DataFrame(flattened_data_v)
df_v['Target'] = target_data_v #add column for target (the labels)
x_val_supervised = df_v.iloc[:,1:-1] #input
y_val_supervised = df_v.iloc[:,1:-1] #output

loaded category: positive successfully
loaded category: negative successfully
```

```
[33]: from sklearn import svm
clf = svm.SVC(kernel = 'linear') #initialize a SVM classifier model
```

Model Performance Indicators

- For my deep-learning model, I evaluated its performance using accuracy score using the `.evaluate()` function. Obtained accuracy of 1.0
- For my SVM model, I used F1 score and `accuracy_score`, which are both imported from `sklearn.metrics`.

F1 score for the SVM model = 0.582;

Accuracy score for the SVM model = 0.585

Model performance indicators from Jupyter Notebook

With Deep Learning Model ¶

we will evaluate the performance of our deep learning model using accuracy score.

```
: score = model.evaluate(x_val, y_val, verbose = 0) #evaluate the model on the validation data
print("accuracy: ", score[1]) #get the accuracy of the model

accuracy: 1.0
```

With non-deep learning model (SVM)

We will evaluate the performance of our image classification model built using SVM with f1 score and accuracy.

```
: from sklearn.metrics import f1_score
f1 = f1_score(y_val_supervised, y_predicted, average = 'weighted')
print("The F1 score for our SVM model is ", f1)
```

The F1 score for our SVM model is 0.5820567492209283

```
: from sklearn.metrics import accuracy_score
#evaluate the SVM model with accuracy score
accuracy_score = accuracy_score(y_val_supervised, y_predicted)

print("Our model has the accuracy score of ", accuracy_score )
```

Our model has the accuracy score of 0.5846153846153846