Data_Mining_Project

March 20, 2025

#Blood Cells Classification for Cancer

This project aims to develop a predictive model for detecting Acute Lymphoblastic Leukemia (ALL) from blood cell images.

0.1 Set up the environment and dataset

[ ]:
```
# Install all needed modules
!pip install ImageHash
!pip install opencv-python
!pip install tensorflow
!pip install imagehash Pillow
!pip install scikit-learn
```

[ ]:
```
# Load modules
import os
import time
import shutil
import pathlib
import itertools
from PIL import Image
import imagehash

import cv2
import numpy as np
import pandas as pd
import seaborn as sns
sns.set_style('darkgrid')
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, classification_report,␣
↪precision_score, recall_score, f1_score, accuracy_score,␣
↪ConfusionMatrixDisplay
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.svm import SVC

import tensorflow as tf
from tensorflow import keras
```

```python
from tensorflow.keras.models import Sequential
from tensorflow.keras.optimizers import Adam, Adamax
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense,␣
↪Activation, Dropout, BatchNormalization
from tensorflow.keras import regularizers
from collections import defaultdict

import warnings
warnings.filterwarnings("ignore")

print ('modules loaded')
```

```
modules loaded
```

[ ]:
```python
# Upload Kaggle API key
from google.colab import files
files.upload()
```

```
<IPython.core.display.HTML object>
```

```
Saving kaggle.json to kaggle.json
```

[ ]:
```
{'kaggle.json':
b'{"username":"jouchihuang","key":"f764f23e82c57f072a05a4e3b72eb84f"}'}
```

[ ]:
```python
# Create a folder for API key
!mkdir -p ~/.kaggle
!mv kaggle.json ~/.kaggle/
!chmod 600 ~/.kaggle/kaggle.json
```

[ ]:
```python
!kaggle datasets download -d mohammadamireshraghi/blood-cell-cancer-all-4class
```

```
Warning: Looks like you're using an outdated API Version, please consider
updating (server 1.7.4.2 / client 1.6.17)
Dataset URL: https://www.kaggle.com/datasets/mohammadamireshraghi/blood-cell-
cancer-all-4class
License(s): Attribution-NonCommercial 4.0 International (CC BY-NC 4.0)
Downloading blood-cell-cancer-all-4class.zip to /content
100% 1.68G/1.68G [01:16<00:00, 23.8MB/s]
100% 1.68G/1.68G [01:16<00:00, 23.6MB/s]
```

[ ]:
```python
# Unzip the datasetn
!unzip blood-cell-cancer-all-4class.zip -d dataset/
```

[ ]:
```python
data_path = "dataset/Blood cell Cancer [ALL]"

images = []
```

```
labels = []

for subfolder in os.listdir(data_path):

    subfolder_path = os.path.join(data_path, subfolder)
    if not os.path.isdir(subfolder_path):
        continue

    for image_filename in os.listdir(subfolder_path):
        image_path = os.path.join(subfolder_path, image_filename)
        images.append(image_path)

        labels.append(subfolder)

data = pd.DataFrame({'image': images, 'label': labels})
```

[ ]:
```
data.head()
```

[ ]:
```
                                               image              label
0  dataset/Blood cell Cancer [ALL]/[Malignant] Pr…  [Malignant] Pro-B
1  dataset/Blood cell Cancer [ALL]/[Malignant] Pr…  [Malignant] Pro-B
2  dataset/Blood cell Cancer [ALL]/[Malignant] Pr…  [Malignant] Pro-B
3  dataset/Blood cell Cancer [ALL]/[Malignant] Pr…  [Malignant] Pro-B
4  dataset/Blood cell Cancer [ALL]/[Malignant] Pr…  [Malignant] Pro-B
```

[ ]:
```
data.tail()
```

[ ]:
```
                                                  image  \
3237  dataset/Blood cell Cancer [ALL]/[Malignant] ea…
3238  dataset/Blood cell Cancer [ALL]/[Malignant] ea…
3239  dataset/Blood cell Cancer [ALL]/[Malignant] ea…
3240  dataset/Blood cell Cancer [ALL]/[Malignant] ea…
3241  dataset/Blood cell Cancer [ALL]/[Malignant] ea…

                         label
3237  [Malignant] early Pre-B
3238  [Malignant] early Pre-B
3239  [Malignant] early Pre-B
3240  [Malignant] early Pre-B
3241  [Malignant] early Pre-B
```

[ ]:
```
# Get unique class labels, ensuring we have all 4 classes
class_labels = sorted(os.listdir(data_path))
print("Classes and their corresponding indices:")
for idx, label in enumerate(class_labels):
    print(f"Class {idx}: {label}")
```

```
Classes and their corresponding indices:
Class 0: Benign
Class 1: [Malignant] Pre-B
Class 2: [Malignant] Pro-B
Class 3: [Malignant] early Pre-B
```

Print out the number of each class to see if there's data imbalance.

[ ]:
```
# Print the number of samples in each class
class_counts = data['label'].value_counts()
print(class_counts)
```

```
label
[Malignant] early Pre-B    979
[Malignant] Pre-B          955
[Malignant] Pro-B          796
Benign                     512
Name: count, dtype: int64
```

0.2 Apply Hash Algorithm to speed up + Data preprocessing

This experiment investigates the effectiveness of hash algorithms in identifying and removing duplicate or similar images, with the goal of enhancing prediction efficiency.

[ ]:
```
# Use a perceptual hash to generate a unique hash for each image
def get_image_hash(image_path):
    try:
        img = Image.open(image_path)
        return str(imagehash.phash(img))
    except Exception as e:
        print(f"Error processing {image_path}: {e}")
        return None
```

[ ]:
```
# Generate hash and integrate hash to data frame
data_path = "dataset/Blood cell Cancer [ALL]"
images = []
labels = []
hashes = []

for subfolder in os.listdir(data_path):
    subfolder_path = os.path.join(data_path, subfolder)
    if not os.path.isdir(subfolder_path):
        continue

    for image_filename in os.listdir(subfolder_path):
        image_path = os.path.join(subfolder_path, image_filename)
        img_hash = get_image_hash(image_path)

        if img_hash:
```

```
            images.append(image_path)
            labels.append(subfolder)
            hashes.append(img_hash)

# Create new DataFrame with image hashes
data = pd.DataFrame({'image': images, 'label': labels, 'hash': hashes})

print(data.head())
```

```
                                                 image               label  \
0  dataset/Blood cell Cancer [ALL]/[Malignant] Pr…  [Malignant] Pro-B
1  dataset/Blood cell Cancer [ALL]/[Malignant] Pr…  [Malignant] Pro-B
2  dataset/Blood cell Cancer [ALL]/[Malignant] Pr…  [Malignant] Pro-B
3  dataset/Blood cell Cancer [ALL]/[Malignant] Pr…  [Malignant] Pro-B
4  dataset/Blood cell Cancer [ALL]/[Malignant] Pr…  [Malignant] Pro-B

               hash
0  811e330fee0fe662
1  81e15a7310de677c
2  d4454cdc40fa73bc
3  84f044fcfa437b32
4  d75217eaa98d5286
```

Remove duplicate images based on hash values. Additionally, validate whether the algorithm is recognizing and distinguishing identical images correctly.

```
[ ]: # Print out all the images that has same hash values
duplicates = data[data.duplicated(subset=['hash'], keep=False)]
print("Duplicate images found:")
print(duplicates[['image', 'hash']])
```

```
Duplicate images found:
                                                    image               hash
2     dataset/Blood cell Cancer [ALL]/[Malignant] Pr…  d4454cdc40fa73bc
15    dataset/Blood cell Cancer [ALL]/[Malignant] Pr…  d25b70485b8b13af
22    dataset/Blood cell Cancer [ALL]/[Malignant] Pr…  95cf4458d4f204f7
32    dataset/Blood cell Cancer [ALL]/[Malignant] Pr…  c45b45d430760fed
40    dataset/Blood cell Cancer [ALL]/[Malignant] Pr…  d45853be9d0d4999
…                                                   …                …
3069  dataset/Blood cell Cancer [ALL]/[Malignant] ea…  c5551695927b22fa
3092  dataset/Blood cell Cancer [ALL]/[Malignant] ea…  a58b52c3593ee1a6
3125  dataset/Blood cell Cancer [ALL]/[Malignant] ea…  c9c23534cf7912ba
3137  dataset/Blood cell Cancer [ALL]/[Malignant] ea…  d295196624dbdd64
3237  dataset/Blood cell Cancer [ALL]/[Malignant] ea…  e88d635b856e29c9

[132 rows x 2 columns]
```

Group images with the same hash value to systematically visualize.

[ ]:
```python
hash_groups = defaultdict(list)

# Group images by their hash values
for _, row in duplicates.iterrows():
    hash_groups[row['hash']].append(row['image'])

# Print images with the same hash
for h, img_list in hash_groups.items():
    print(f"\nImages with hash {h}:")
    for img in img_list:
        print(f"  - {img}")
```

Print out the images have same hash values to see if it's actually duplicate images.

[ ]:
```python
def show_duplicate_images(duplicate_hashes):
    for h, img_list in hash_groups.items():
        fig, axes = plt.subplots(1, len(img_list), figsize=(12, 4))
        fig.suptitle(f"Images with hash: {h}")

        for ax, img_path in zip(axes, img_list):
            img = cv2.imread(img_path)[:, :, ::-1]  # Convert BGR to RGB
            ax.imshow(img)
            ax.set_title(img_path.split('/')[-1])
            ax.axis('off')

        plt.show()

show_duplicate_images(duplicates)
```

```
Output hidden; open in https://colab.research.google.com to view.
```

Remove the duplicate images to improve the prediction and traning efficiency.

[ ]:
```python
# Drop the duplicate images
# Print out the row difference to see how many data we dropped
print("Total rows before removing duplicates:", len(data))

data.drop_duplicates(subset=['hash'], inplace=True)

print("Total rows after removing duplicates:", len(data))
```

```
Total rows before removing duplicates: 3242
Total rows after removing duplicates: 3176
```

0.3 Split the data

```
[ ]: strat = data['label']
train_df, dummy_df = train_test_split(data,  train_size= 0.85, shuffle= True,␣
 ↪random_state= 123, stratify= strat)

strat = dummy_df['label']
valid_df, test_df = train_test_split(dummy_df,  train_size= 0.5, shuffle= True,␣
 ↪random_state= 123, stratify= strat)
```

```
[ ]: print("Training set shape:", train_df.shape)
print("Validation set shape:", valid_df.shape)
print("Test set shape:", test_df.shape)
```

```
Training set shape: (2699, 3)
Validation set shape: (238, 3)
Test set shape: (239, 3)
```

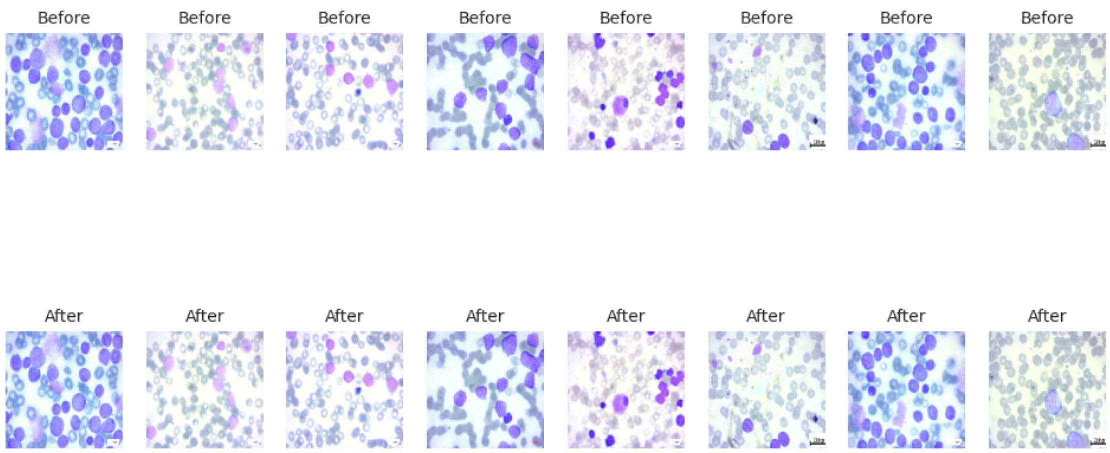
```
[ ]: batch_size = 32
img_size = (150, 150)
channels = 3
img_shape = (img_size[0], img_size[1], channels)

tr_gen = ImageDataGenerator()
ts_gen = ImageDataGenerator()

train_gen = tr_gen.flow_from_dataframe(train_df, x_col='image', y_col='label',␣
 ↪target_size=img_size, class_mode='categorical', color_mode='rgb',␣
 ↪shuffle=True, batch_size=batch_size)

valid_gen = ts_gen.flow_from_dataframe(valid_df, x_col='image', y_col='label',␣
 ↪target_size=img_size, class_mode='categorical', color_mode='rgb',␣
 ↪shuffle=True, batch_size=batch_size)

test_gen = ts_gen.flow_from_dataframe(test_df, x_col='image', y_col='label',␣
 ↪target_size=img_size, class_mode='categorical', color_mode='rgb',␣
 ↪shuffle=False, batch_size=batch_size)
```

```
Found 2699 validated image filenames belonging to 4 classes.
Found 238 validated image filenames belonging to 4 classes.
Found 239 validated image filenames belonging to 4 classes.
```

0.4 Normalize the pixels in training images

```
[ ]: # Show the comparison between before and after normalizing images
g_dict = train_gen.class_indices
classes = list(g_dict.keys())
```

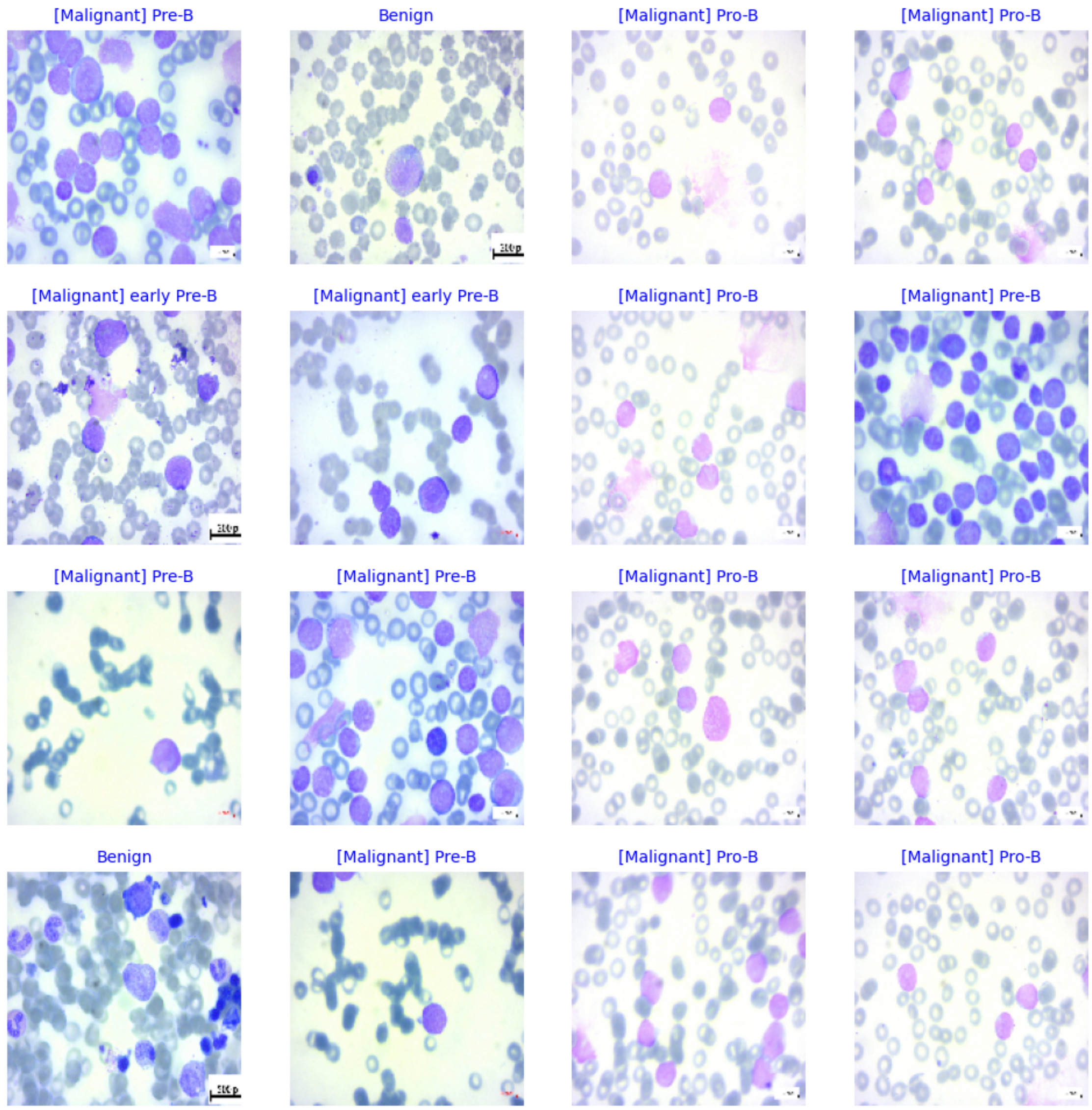
```
images, labels = next(train_gen)

plt.figure(figsize=(12, 6))

for i in range(8):
    plt.subplot(2, 8, i + 1)
    plt.imshow(images[i].astype('uint8'))  # Original (0-255)
    plt.title("Before", fontsize=10)
    plt.axis('off')

    plt.subplot(2, 8, i + 9)
    plt.imshow(images[i] / 255)  # Normalized (0-1)
    plt.title("After", fontsize=10)
    plt.axis('off')

plt.show()
```



Visually, there is no noticeable difference between the images before and after normalization. However, the final model evaluations differ depending on whether hashing was used, as shown below.

[ ]:
```
g_dict = train_gen.class_indices
classes = list(g_dict.keys())
images, labels = next(train_gen)

plt.figure(figsize= (12, 12))

for i in range(16):
    plt.subplot(4, 4, i + 1)
    image = images[i] / 255 # Normalize pixels to 0-1 range
    plt.imshow(image)
    index = np.argmax(labels[i])
```

```
    class_name = classes[index]
    plt.title(class_name, color= 'blue', fontsize= 10)
    plt.axis('off')
plt.show()
```



##Squeeze and Excitation in CNN

This technique is aimed at enhancing important features while suppressing less useful ones, **allowing the network to focus on the most relevant information during training.**

[ ]:
```
import tensorflow as tf
from tensorflow.keras import layers, models
from tensorflow.keras.applications import ResNet50
```

```python
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

Squeeze: This step **compresses the spatial information of each feature map into a single descriptor**, using global average pooling.

Excitation: Excite the feature maps by learning **a set of channel-wise weights** that will be used to scale the channels.

Recalibration: The recalibrated feature map is obtained by **multiplying each original feature map by its corresponding weight** from the excitation step.

[ ]:
```python
def squeeze_excite_block(input, ratio=16):
    # Get number of channels in the input
    channel_axis = -1
    channels = input.shape[channel_axis]

    # Squeeze: Global Average Pooling
    se = layers.GlobalAveragePooling2D()(input)
    se = layers.Reshape((1, 1, channels))(se)

    # Excitation: Fully Connected layer with ReLU and Sigmoid activations
    se = layers.Dense(channels // ratio, activation='relu')(se)
    se = layers.Dense(channels, activation='sigmoid')(se)

    # Scale the input by the output of the SE block
    se = layers.Multiply()([input, se])
    return se
```

[ ]:
```python
def create_senet_model(input_shape, num_classes):
    input = layers.Input(shape=input_shape)

    # Initial Convolution Layer
    x = layers.Conv2D(32, (3, 3), padding='same', activation='relu')(input)
    x = layers.MaxPooling2D((2, 2))(x)

    # SENet Block 1
    x = squeeze_excite_block(x)

    # More Convolution Layers
    x = layers.Conv2D(64, (3, 3), padding='same', activation='relu')(x)
    x = layers.MaxPooling2D((2, 2))(x)

    # SENet Block 2
    x = squeeze_excite_block(x)

    # Global Average Pooling for final feature reduction
    x = layers.GlobalAveragePooling2D()(x)
```

```
    # Fully Connected Layer
    x = layers.Dense(128, activation='relu')(x)

    # Output Layer
    output = layers.Dense(num_classes, activation='softmax')(x)

    # Create the Model
    model = models.Model(inputs=input, outputs=output)

    return model
```

[ ]:
```
model = create_senet_model(input_shape=(150, 150, 3), num_classes=len(train_gen.
 ↪class_indices))

model.compile(optimizer='adam', loss='categorical_crossentropy',␣
 ↪metrics=['accuracy'])

# Print the model summary to confirm the architecture
model.summary()
```

Model: "functional_2"

```
 Layer (type)                 Output Shape                  Param #  Connected␣
↪to

 input_layer_2                (None, 150, 150, 3)                 0  -           ␣
↪
 (InputLayer)                                                                    ␣
↪

 conv2d_4 (Conv2D)            (None, 150, 150, 32)              896 ␣
↪input_layer_2[0][0]

 max_pooling2d_4              (None, 75, 75, 32)                  0 ␣
↪conv2d_4[0][0]
 (MaxPooling2D)                                                                  ␣
↪

 global_average_pooling2d…    (None, 32)                          0 ␣
↪max_pooling2d_4[0][0]
 (GlobalAveragePooling2D)                                                        ␣
↪

 reshape_4 (Reshape)          (None, 1, 1, 32)                    0 ␣
↪global_average_poolin…
```

```
dense_12 (Dense)              (None, 1, 1, 2)                  66 ␣
↪reshape_4[0][0]

dense_13 (Dense)              (None, 1, 1, 32)                 96 ␣
↪dense_12[0][0]

multiply_4 (Multiply)         (None, 75, 75, 32)                0 ␣
↪max_pooling2d_4[0][0],
                                                                  ␣
↪dense_13[0][0]

conv2d_5 (Conv2D)             (None, 75, 75, 64)           18,496 ␣
↪multiply_4[0][0]

max_pooling2d_5               (None, 37, 37, 64)                0 ␣
↪conv2d_5[0][0]
(MaxPooling2D)                                                              ␣
↪

global_average_pooling2d…     (None, 64)                        0 ␣
↪max_pooling2d_5[0][0]
(GlobalAveragePooling2D)                                                    ␣
↪

reshape_5 (Reshape)           (None, 1, 1, 64)                  0 ␣
↪global_average_poolin…

dense_14 (Dense)              (None, 1, 1, 4)                 260 ␣
↪reshape_5[0][0]

dense_15 (Dense)              (None, 1, 1, 64)                320 ␣
↪dense_14[0][0]

multiply_5 (Multiply)         (None, 37, 37, 64)                0 ␣
↪max_pooling2d_5[0][0],
                                                                  ␣
↪dense_15[0][0]

global_average_pooling2d…     (None, 64)                        0 ␣
↪multiply_5[0][0]
(GlobalAveragePooling2D)                                                    ␣
↪

dense_16 (Dense)              (None, 128)                   8,320 ␣
↪global_average_poolin…
```

```
 dense_17 (Dense)            (None, 4)                          516 ␣
↪dense_16[0][0]
```

```
Total params: 28,970 (113.16 KB)

Trainable params: 28,970 (113.16 KB)

Non-trainable params: 0 (0.00 B)
```

Train the model while also tracking the duration of the training process to evaluate whether the hash algorithm contributes to faster performance.

[ ]:
```python
import time

# Start time
start_time = time.time()

# Train the model
history = model.fit(
    train_gen,
    steps_per_epoch=train_gen.samples // batch_size,
    epochs=10,
    validation_data=valid_gen,
    validation_steps=valid_gen.samples // batch_size
)

# End time
end_time = time.time()

# Calculate the training duration
training_duration = end_time - start_time
print(f"Training took {training_duration:.2f} seconds.")
```

```
Epoch 1/10
84/84                 38s 398ms/step -
accuracy: 0.4574 - loss: 2.5867 - val_accuracy: 0.6116 - val_loss: 0.7603
Epoch 2/10
84/84                 3s 31ms/step -
accuracy: 0.5625 - loss: 0.9456 - val_accuracy: 0.6027 - val_loss: 0.7291
Epoch 3/10
84/84                 30s 362ms/step -
accuracy: 0.7310 - loss: 0.7255 - val_accuracy: 0.8036 - val_loss: 0.5947
Epoch 4/10
84/84                 3s 30ms/step -
```

```
accuracy: 0.8750 - loss: 0.4388 - val_accuracy: 0.8036 - val_loss: 0.5983
Epoch 5/10
84/84              31s 366ms/step -
accuracy: 0.7706 - loss: 0.6054 - val_accuracy: 0.8661 - val_loss: 0.4423
Epoch 6/10
84/84              3s 30ms/step -
accuracy: 0.9688 - loss: 0.3277 - val_accuracy: 0.7545 - val_loss: 0.5396
Epoch 7/10
84/84              31s 363ms/step -
accuracy: 0.8230 - loss: 0.4421 - val_accuracy: 0.8259 - val_loss: 0.3839
Epoch 8/10
84/84              3s 32ms/step -
accuracy: 0.8438 - loss: 0.2904 - val_accuracy: 0.8348 - val_loss: 0.3928
Epoch 9/10
84/84              30s 359ms/step -
accuracy: 0.8502 - loss: 0.3857 - val_accuracy: 0.8482 - val_loss: 0.4201
Epoch 10/10
84/84              2s 30ms/step -
accuracy: 0.7812 - loss: 0.5003 - val_accuracy: 0.8616 - val_loss: 0.3805
Training took 173.66 seconds.
```

[ ]:
```
test_loss, test_acc = model.evaluate(test_gen, steps=test_gen.samples //␣
 ↪batch_size)
print(f"Test accuracy: {test_acc}")
```

```
7/7              3s 365ms/step -
accuracy: 0.8518 - loss: 0.4009
Test accuracy: 0.84375
```

[ ]:
```
model.save('senet_blood_cell_model.h5')
```

```
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or
`keras.saving.save_model(model)`. This file format is considered legacy. We
recommend using instead the native Keras format, e.g.
`model.save('my_model.keras')` or `keras.saving.save_model(model,
'my_model.keras')`.
```

[ ]:
```
# Get predictions for the test set
predictions = model.predict(test_gen, steps=test_gen.samples // batch_size)
```

```
WARNING:tensorflow:5 out of the last 24 calls to <function
TensorFlowTrainer.make_predict_function.<locals>.one_step_on_data_distributed at
0x7e33502e2de0> triggered tf.function retracing. Tracing is expensive and the
excessive number of tracings could be due to (1) creating @tf.function
repeatedly in a loop, (2) passing tensors with different shapes, (3) passing
Python objects instead of tensors. For (1), please define your @tf.function
outside of the loop. For (2), @tf.function has reduce_retracing=True option that
can avoid unnecessary retracing. For (3), please refer to
```

```
https://www.tensorflow.org/guide/function#controlling_retracing and
https://www.tensorflow.org/api_docs/python/tf/function for  more details.
```

```
7/7              3s 296ms/step
```

[ ]:
```
# Get the predicted class labels
predicted_classes = np.argmax(predictions, axis=1)

# Get the true class labels from the test generator
true_classes = test_gen.classes
```

[ ]:
```
true_classes = test_gen.classes
print(len(true_classes))
```

```
239
```

[ ]:
```
predictions = model.predict(test_gen)
predicted_classes = np.argmax(predictions, axis=1)
print(len(predicted_classes))
```

```
8/8              3s 425ms/step
239
```

[ ]:
```
# Calculate Accuracy
accuracy = accuracy_score(true_classes, predicted_classes)
print(f"Accuracy: {accuracy:.4f}")

# Calculate Precision
precision = precision_score(true_classes, predicted_classes,␣
 ↪average='weighted')  # weighted averages for multi-class classification
print(f"Precision: {precision:.4f}")

# Calculate Recall
recall = recall_score(true_classes, predicted_classes, average='weighted')
print(f"Recall: {recall:.4f}")

# Calculate F1-Score
f1 = f1_score(true_classes, predicted_classes, average='weighted')
print(f"F1-Score: {f1:.4f}")
```

```
Accuracy: 0.8452
Precision: 0.8523
Recall: 0.8452
F1-Score: 0.8323
```

[ ]:
```
# Confusion Matrix tovisualize classification performance)
conf_matrix = confusion_matrix(true_classes, predicted_classes)
print("Confusion Matrix:")
```

```
print(conf_matrix)
```

```
Confusion Matrix:
[[15  2  5 16]
 [ 0 65  2  5]
 [ 0  0 56  0]
 [ 3  0  4 66]]
```

Takeaways: 1. Pre-B, Pro-B and early Pre-B have relatively high accuracy, with most predictions landing in the correct category. 2. **Benign has a high misclassification rate to early Pre-B, indicating potential confusion between these two classes.** 3. The diagonal values indicate correct predictions for each class. **Correct predictions are still majority cases.**

0.5 Squeeze and Excitation in CNN Without Hash Algorithm

[ ]:
```
# Reload the data just in case hash is not applied
data_path = "dataset/Blood cell Cancer [ALL]"

images = []
labels = []

for subfolder in os.listdir(data_path):

    subfolder_path = os.path.join(data_path, subfolder)
    if not os.path.isdir(subfolder_path):
        continue

    for image_filename in os.listdir(subfolder_path):
        image_path = os.path.join(subfolder_path, image_filename)
        images.append(image_path)

        labels.append(subfolder)

data = pd.DataFrame({'image': images, 'label': labels})
print(len(data)) # Match the original data length = 3242
```

```
3242
```

[ ]:
```
strat = data['label']
train_df, dummy_df = train_test_split(data,  train_size= 0.85, shuffle= True,␣
 ↪random_state= 123, stratify= strat)

strat = dummy_df['label']
valid_df, test_df = train_test_split(dummy_df,  train_size= 0.5, shuffle= True,␣
 ↪random_state= 123, stratify= strat)
```

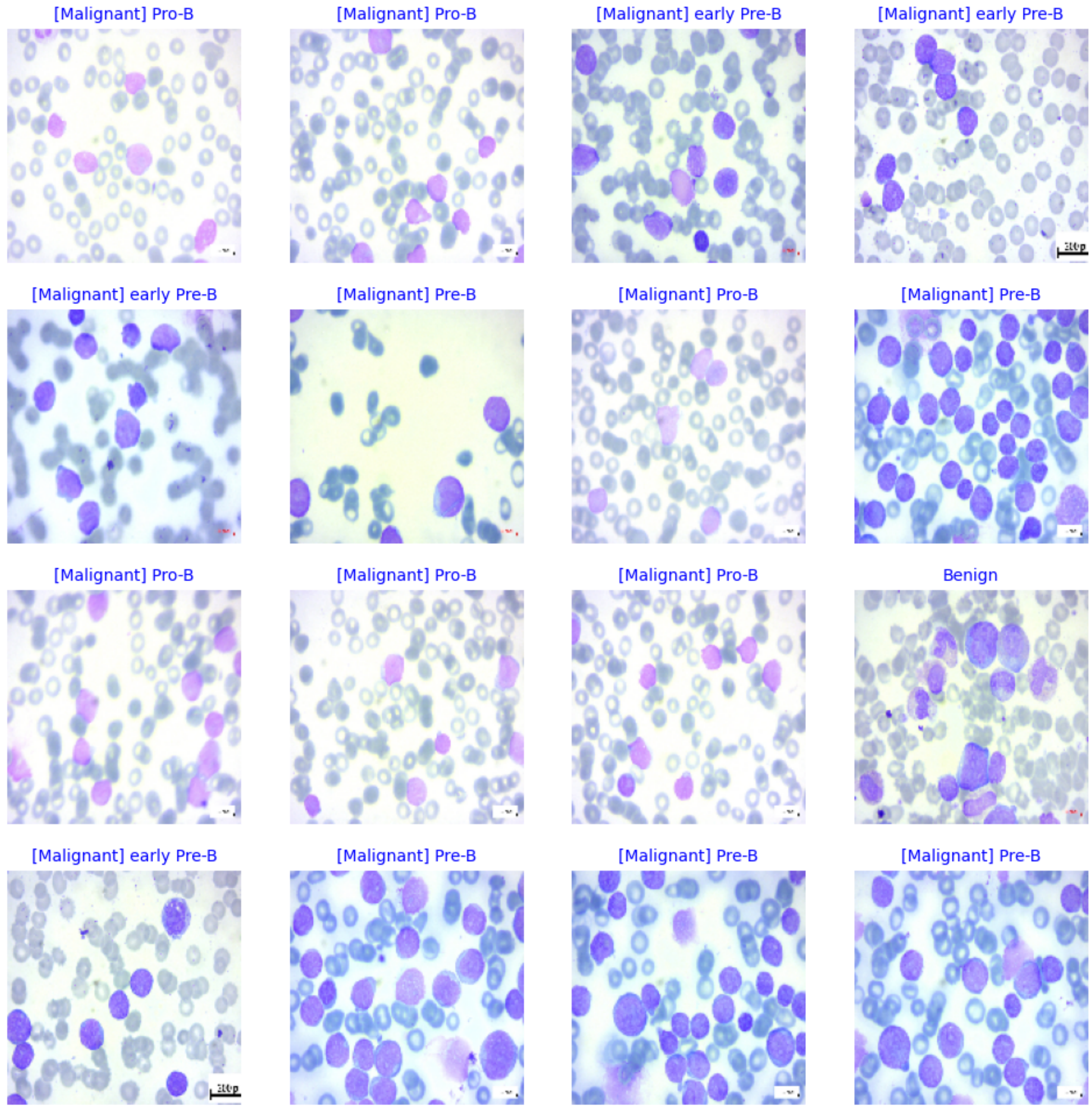
[ ]:
```
print("Training set shape:", train_df.shape)
print("Validation set shape:", valid_df.shape)
print("Test set shape:", test_df.shape)
```

```
Training set shape: (2755, 2)
Validation set shape: (243, 2)
Test set shape: (244, 2)
```

[ ]:
```
batch_size = 32
img_size = (150, 150)
channels = 3
img_shape = (img_size[0], img_size[1], channels)

tr_gen = ImageDataGenerator()
ts_gen = ImageDataGenerator()

train_gen = tr_gen.flow_from_dataframe(train_df, x_col='image', y_col='label',␣
↪target_size=img_size, class_mode='categorical', color_mode='rgb',␣
↪shuffle=True, batch_size=batch_size)

valid_gen = ts_gen.flow_from_dataframe(valid_df, x_col='image', y_col='label',␣
↪target_size=img_size, class_mode='categorical', color_mode='rgb',␣
↪shuffle=True, batch_size=batch_size)

test_gen = ts_gen.flow_from_dataframe(test_df, x_col='image', y_col='label',␣
↪target_size=img_size, class_mode='categorical', color_mode='rgb',␣
↪shuffle=False, batch_size=batch_size)
```

```
Found 2755 validated image filenames belonging to 4 classes.
Found 243 validated image filenames belonging to 4 classes.
Found 244 validated image filenames belonging to 4 classes.
```

[ ]:
```
g_dict = train_gen.class_indices
classes = list(g_dict.keys())
images, labels = next(train_gen)

plt.figure(figsize= (12, 12))

for i in range(16):
    plt.subplot(4, 4, i + 1)
    image = images[i] / 255 # Normalize pixels to 0-1 range
    plt.imshow(image)
    index = np.argmax(labels[i])
    class_name = classes[index]
    plt.title(class_name, color= 'blue', fontsize= 10)
    plt.axis('off')
plt.show()
```



[ ]:
```
model = create_senet_model(input_shape=(150, 150, 3), num_classes=len(train_gen.class_indices))

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

# Print the model summary to confirm the architecture
model.summary()
```

```
Model: "functional_3"
```

```
Layer (type)                 Output Shape                 Param #  Connected
↪to

input_layer_3                (None, 150, 150, 3)                0  -
↪
(InputLayer)
↪

conv2d_6 (Conv2D)            (None, 150, 150, 32)             896
↪input_layer_3[0][0]

max_pooling2d_6              (None, 75, 75, 32)                 0
↪conv2d_6[0][0]
(MaxPooling2D)
↪

global_average_pooling2d…    (None, 32)                         0
↪max_pooling2d_6[0][0]
(GlobalAveragePooling2D)
↪

reshape_6 (Reshape)          (None, 1, 1, 32)                   0
↪global_average_poolin…

dense_18 (Dense)             (None, 1, 1, 2)                   66
↪reshape_6[0][0]

dense_19 (Dense)             (None, 1, 1, 32)                  96
↪dense_18[0][0]

multiply_6 (Multiply)        (None, 75, 75, 32)                 0
↪max_pooling2d_6[0][0],

↪dense_19[0][0]

conv2d_7 (Conv2D)            (None, 75, 75, 64)            18,496
↪multiply_6[0][0]

max_pooling2d_7              (None, 37, 37, 64)                 0
↪conv2d_7[0][0]
(MaxPooling2D)
↪

global_average_pooling2d…    (None, 64)                         0
↪max_pooling2d_7[0][0]
```

```
(GlobalAveragePooling2D)                                                          ␣
↪

reshape_7 (Reshape)         (None, 1, 1, 64)                  0  ␣
↪global_average_poolin…

dense_20 (Dense)            (None, 1, 1, 4)                 260  ␣
↪reshape_7[0][0]

dense_21 (Dense)            (None, 1, 1, 64)                320  ␣
↪dense_20[0][0]

multiply_7 (Multiply)       (None, 37, 37, 64)                0  ␣
↪max_pooling2d_7[0][0],
                                                                 ␣
↪dense_21[0][0]

global_average_pooling2d…   (None, 64)                        0  ␣
↪multiply_7[0][0]
(GlobalAveragePooling2D)                                                          ␣
↪

dense_22 (Dense)            (None, 128)                   8,320  ␣
↪global_average_poolin…

dense_23 (Dense)            (None, 4)                       516  ␣
↪dense_22[0][0]
```

Total params: 28,970 (113.16 KB)

Trainable params: 28,970 (113.16 KB)

Non-trainable params: 0 (0.00 B)

[ ]:
```
import time

# Start time
start_time = time.time()

# Train the model
history = model.fit(
    train_gen,
    steps_per_epoch=train_gen.samples // batch_size,
```

```
    epochs=10,
    validation_data=valid_gen,
    validation_steps=valid_gen.samples // batch_size
)

# End time
end_time = time.time()

# Calculate the training duration
training_duration = end_time - start_time
print(f"Training took {training_duration:.2f} seconds.")
```

```
Epoch 1/10
86/86                 38s 390ms/step -
accuracy: 0.3620 - loss: 2.5406 - val_accuracy: 0.5759 - val_loss: 0.9875
Epoch 2/10
86/86                 3s 31ms/step -
accuracy: 0.4688 - loss: 1.2092 - val_accuracy: 0.5893 - val_loss: 0.9156
Epoch 3/10
86/86                 31s 357ms/step -
accuracy: 0.7057 - loss: 0.7591 - val_accuracy: 0.8036 - val_loss: 0.4978
Epoch 4/10
86/86                 3s 30ms/step -
accuracy: 0.8125 - loss: 0.4971 - val_accuracy: 0.8393 - val_loss: 0.4746
Epoch 5/10
86/86                 31s 360ms/step -
accuracy: 0.8032 - loss: 0.4777 - val_accuracy: 0.8527 - val_loss: 0.3801
Epoch 6/10
86/86                 3s 32ms/step -
accuracy: 0.7812 - loss: 0.4297 - val_accuracy: 0.8080 - val_loss: 0.4550
Epoch 7/10
86/86                 31s 361ms/step -
accuracy: 0.8314 - loss: 0.4203 - val_accuracy: 0.8438 - val_loss: 0.3907
Epoch 8/10
86/86                 3s 31ms/step -
accuracy: 0.9062 - loss: 0.3215 - val_accuracy: 0.8527 - val_loss: 0.4127
Epoch 9/10
86/86                 31s 368ms/step -
accuracy: 0.8256 - loss: 0.4401 - val_accuracy: 0.7812 - val_loss: 0.5663
Epoch 10/10
86/86                 3s 31ms/step -
accuracy: 0.6250 - loss: 1.0245 - val_accuracy: 0.8795 - val_loss: 0.2876
Training took 175.77 seconds.
```

[ ]:
```
test_loss, test_acc = model.evaluate(test_gen, steps=test_gen.samples //␣
 ↪batch_size)
print(f"Test accuracy: {test_acc}")
```

```
7/7              2s 353ms/step -
accuracy: 0.9057 - loss: 0.2416
Test accuracy: 0.9017857313156128
```

[ ]:
```
model.save('senet_model_no_hash.h5')
```

```
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or
`keras.saving.save_model(model)`. This file format is considered legacy. We
recommend using instead the native Keras format, e.g.
`model.save('my_model.keras')` or `keras.saving.save_model(model,
'my_model.keras')`.
```

[ ]:
```
# Get true labels and predictions
y_true = test_gen.classes  # True labels
y_pred_probs = model.predict(test_gen)  # Predicted probabilities
y_pred = np.argmax(y_pred_probs, axis=1)  # Convert probabilities to class␣
 ↪indices
```

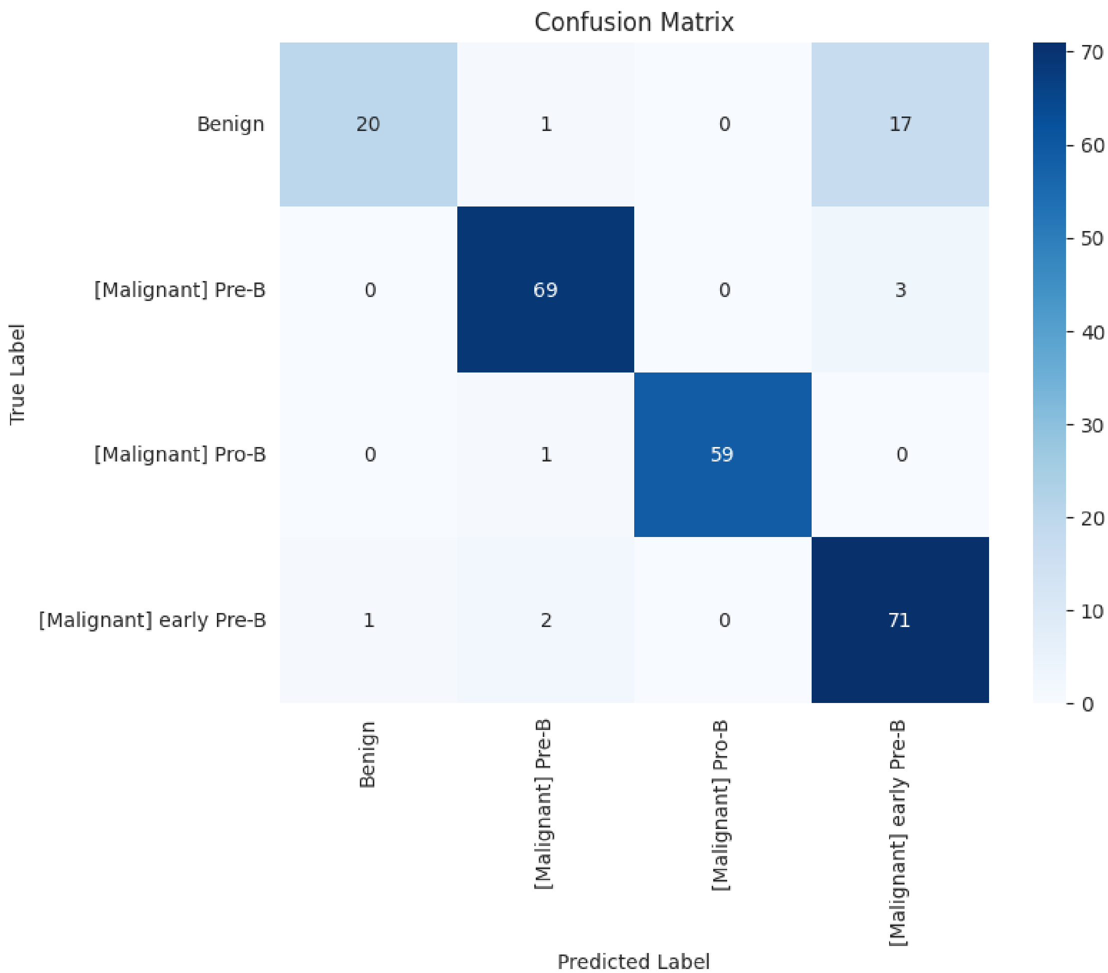
```
8/8              4s 381ms/step
```

[ ]:
```
# Compute accuracy
accuracy = accuracy_score(y_true, y_pred)

# Compute precision, recall, and F1-score (weighted for class imbalance)
precision = precision_score(y_true, y_pred, average='weighted')
recall = recall_score(y_true, y_pred, average='weighted')
f1 = f1_score(y_true, y_pred, average='weighted')

# Display results
print(f"Accuracy: {accuracy:.4f}")
print(f"Precision: {precision:.4f}")
print(f"Recall: {recall:.4f}")
print(f"F1-score: {f1:.4f}")
```

```
Accuracy: 0.8975
Precision: 0.9098
Recall: 0.8975
F1-score: 0.8913
```

[ ]:
```
# Compute confusion matrix
cm = confusion_matrix(y_true, y_pred)

# Plot the confusion matrix
plt.figure(figsize=(8,6))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", xticklabels=class_labels,␣
 ↪yticklabels=class_labels)
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
```

```
plt.title("Confusion Matrix")
plt.show()
```



Takeaways: 1. Evaluation of Hashing Impact: To assess the effect of hashing, **I replicated the model, maintained the same data split, and applied consistent image normalization across all experiments.** 2. Performance Without Hashing: **The results indicated that omitting the hash algorithm led to a slight increase in performance.** 3. Classification Challenges: Similar to the results with hashing applied, the model occasionally struggles to differentiate between benign and early Pre-B categories. **Notably, there is a significant increase in benign cases being misclassified as early Pre-B when hashing is not used.**

##Support Vector Machines (SVM)

The following results were obtained by training the **SVM model using various performance-enhancing methods, including feature scaling, PCA, SMOTE, and GridSearch.**

[ ]:
```
# Load the data again ensuring it's the original data
# Define image
```

```python
IMG_SIZE = (150, 150)

image_paths = []
labels = []

data_path = "dataset/Blood cell Cancer [ALL]"
for subfolder in os.listdir(data_path):
    subfolder_path = os.path.join(data_path, subfolder)
    if not os.path.isdir(subfolder_path):
        continue

    for image_filename in os.listdir(subfolder_path):
        image_path = os.path.join(subfolder_path, image_filename)
        image_paths.append(image_path)
        labels.append(subfolder)

# Convert labels to numerical format
label_encoder = LabelEncoder()
labels_encoded = label_encoder.fit_transform(labels)

# Load and preprocess images
features = []
for img_path in image_paths:
    img = cv2.imread(img_path, cv2.IMREAD_COLOR)  # Read image in color
    img = cv2.resize(img, IMG_SIZE)  # Resize to a fixed size
    img = img.flatten()  # Convert image to 1D feature vector
    features.append(img)

features = np.array(features)  # Convert to NumPy array
labels_encoded = np.array(labels_encoded)  # Convert labels to NumPy array
```

Feature Scaling - Standardization

I conducted experiments using both standard scaling and MinMax scaling methods, **but observed no significant difference in performance.**

[ ]:
```python
from sklearn.preprocessing import StandardScaler, MinMaxScaler
# Use StandardScaler
scaler = StandardScaler()
features_scaled = scaler.fit_transform(features)

# For normalization
# scaler = MinMaxScaler()
# features_normalized = scaler.fit_transform(features)
```

Apply PCA to reduce dimensionality

[ ]:
```python
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA

# Dimensionality Reduction (PCA)
pca = PCA(n_components=100)  # Reduce to 100 principal components
features_pca = pca.fit_transform(features_scaled)
```

Apply SMOTE to balance classes

I noticed that the number of benign samples in the dataset is relatively small, which may explain why the model struggled to capture the characteristics of the benign category and often misclassified it as early-Pre B. To address this issue, I applied SMOTE to mitigate the impact of the class imbalance.

[ ]:
```python
from imblearn.over_sampling import SMOTE

# Handle Class Imbalance using SMOTE
smote = SMOTE()
features_resampled, labels_resampled = smote.fit_resample(features_pca,␣
↪labels_encoded)
```

Split the data

[ ]:
```python
X_train, X_test, y_train, y_test = train_test_split(features_resampled,␣
↪labels_resampled, test_size=0.2, random_state=42)
```

Use GridSeach to find the best parameter

[ ]:
```python
from sklearn.model_selection import GridSearchCV

# Hyperparameter Tuning using GridSearchCV
param_grid = {'C': [0.1, 1, 10], 'gamma': [0.1, 1, 10]}
grid_search = GridSearchCV(SVC(), param_grid, cv=5)  # 5-fold cross-validation
grid_search.fit(X_train, y_train)

print(f"Best parameters from GridSearchCV: {grid_search.best_params_}")

# Train SVM model using the best parameters
best_svc = grid_search.best_estimator_
best_svc.fit(X_train, y_train)
```

```
Best parameters from GridSearchCV: {'C': 10, 'gamma': 0.1}
```

[ ]:
```
SVC(C=10, gamma=0.1)
```

[ ]:
```python
# Evaluate on test data
y_pred = best_svc.predict(X_test)

# Evaluation Metrics without weighted average
```

```
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average='weighted')  # Weighted to␣
 ↪handle multi-class
recall = recall_score(y_test, y_pred, average='weighted')        # Weighted to␣
 ↪handle multi-class
f1 = f1_score(y_test, y_pred, average='weighted')                # Weighted to␣
 ↪handle multi-class

print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1-Score:", f1)
```

```
Accuracy: 0.29591836734693877
Precision: 0.8211252068394926
Recall: 0.29591836734693877
F1-Score: 0.19948910942196046
```

[ ]:
```
import joblib

# Save the trained SVM model
joblib.dump(best_svc, 'svm_model.pkl')
print("Model saved successfully!")
```

```
Model saved successfully!
```

Takeaways: 1. SVM Performance Comparison: **When compared to the CNN model, the performance of the SVM model significantly dropped.** This suggests that either hashing needs to be applied to the SVM model as well, or that SVM may not be well-suited for this particular task. 2. Precision vs. Recall: The model exhibits a relatively high precision but a low recall, resulting in a low F1-Score. This indicates that the model is more cautious, effectively minimizing false positives, but struggles to identify true positives, leading to a high number of false negatives. 3. Next Steps: Moving forward, **I plan to experiment with allowing the CNN to first learn features from the images, followed by the application of the hashing algorithm.** The objective is to **determine whether reversing the order of operations can enhance overall performance** when combined with an SVM model.

0.6 CNN Extract Features - Hashing - SVM Model

[ ]:
```
import os
import numpy as np
import cv2
import imagehash
from PIL import Image
from sklearn.svm import SVC
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
```

```python
# Define image size
IMG_SIZE = (150, 150)

# Load image paths and labels
image_paths = []
labels = []

data_path = "dataset/Blood cell Cancer [ALL]"
for subfolder in os.listdir(data_path):
    subfolder_path = os.path.join(data_path, subfolder)
    if not os.path.isdir(subfolder_path):
        continue

    for image_filename in os.listdir(subfolder_path):
        image_path = os.path.join(subfolder_path, image_filename)
        image_paths.append(image_path)
        labels.append(subfolder)

# Convert labels to numerical format
label_encoder = LabelEncoder()
labels_encoded = label_encoder.fit_transform(labels)

# Perceptual Hash Function
def get_image_hash(image_path):
    try:
        img = Image.open(image_path).convert("L")  # Convert to grayscale
        hash_value = imagehash.phash(img)  # Compute perceptual hash
        return np.array(hash_value.hash, dtype=np.uint8).flatten()  # Convert␣
 ↪to NumPy array
    except Exception as e:
        print(f"Error processing {image_path}: {e}")
        return None

# Compute perceptual hashes for all images
features = []
for img_path in image_paths:
    hash_vector = get_image_hash(img_path)
    if hash_vector is not None:
        features.append(hash_vector)

features = np.array(features)  # Convert to NumPy array
labels_encoded = np.array(labels_encoded)  # Convert labels to NumPy array

# Split into train and test sets
X_train, X_test, y_train, y_test = train_test_split(features, labels_encoded,␣
 ↪test_size=0.2, random_state=42)
```

```
# Train an SVM Classifier
svm_model = SVC(kernel='rbf')  # You can also try 'rbf' kernel
svm_model.fit(X_train, y_train)

# Make Predictions
y_pred = svm_model.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average='weighted')  # Weighted to␣
↪handle multi-class
recall = recall_score(y_test, y_pred, average='weighted')        # Weighted to␣
↪handle multi-class
f1 = f1_score(y_test, y_pred, average='weighted')                # Weighted to␣
↪handle multi-class

print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1-Score:", f1)
```

```
Accuracy: 0.423728813559322
Precision: 0.3622125111192439
Recall: 0.423728813559322
F1-Score: 0.3893547722260043
```

[ ]:
```
# Print Precision, Recall, F1-Score
print("\nClassification Report:")
print(classification_report(y_test, y_pred, target_names=label_encoder.
↪classes_))
```

```
Classification Report:
                        precision    recall  f1-score   support

                 Benign       0.00      0.00      0.00       104
      [Malignant] Pre-B       0.42      0.55      0.48       184
      [Malignant] Pro-B       0.53      0.54      0.53       187
[Malignant] early Pre-B       0.34      0.42      0.38       174

               accuracy                           0.42       649
              macro avg       0.32      0.38      0.35       649
           weighted avg       0.36      0.42      0.39       649
```

[ ]:
```
import joblib
```

```
# Save the SVM model to a file
joblib.dump(svm_model, 'svm_model_hashing.pkl')
```

[ ]: ['svm_model_hashing.pkl']

Takeaways: 1. The model, which applies CNN for feature extraction, followed by perceptual hashing and SVM training, **demonstrates performance similar to the previous SVM-based approach.** 2. However, **the significant drop in precision** suggests that the model is making random predictions rather than effectively distinguishing between classes. 3. Next Steps: I will implement a Random Forest model and incorporate techniques such as PCA and data augmentation to evaluate potential performance improvements.

##Random Forest

[ ]:
```
# Reload the data

data_path = "dataset/Blood cell Cancer [ALL]"

images = []
labels = []

for subfolder in os.listdir(data_path):
    subfolder_path = os.path.join(data_path, subfolder)
    if not os.path.isdir(subfolder_path):
        continue

    for image_filename in os.listdir(subfolder_path):
        image_path = os.path.join(subfolder_path, image_filename)
        images.append(image_path)
        labels.append(subfolder)

# Convert to DataFrame
data = pd.DataFrame({'image': images, 'label': labels})
print(len(data)) # Match the original data length = 3242
```

3242

[ ]:
```
# Convert labels to numerical format
label_encoder = LabelEncoder()
labels_encoded = label_encoder.fit_transform(data['label'])

# Read and process original image data
features = []
for img_path in data['image']:
    img = cv2.imread(img_path)
    img = cv2.resize(img, IMG_SIZE)
    img = img.flatten()  # Convert the image to a 1D feature vector
    features.append(img)
```

```
features = np.array(features)
```

Apply data augmentation

Try to apply data augmentation to increase the diversity of training set.

[ ]:
```
# **Define ImageDataGenerator for data augmentation**
data_gen = ImageDataGenerator(
    rotation_range=20,  # Rotation range
    width_shift_range=0.2,  # Horizontal shift range
    height_shift_range=0.2,  # Vertical shift range
    shear_range=0.2,  # Shear transformation range
    zoom_range=0.2,  # Zoom transformation range
    horizontal_flip=True,  # Whether to flip horizontally
    fill_mode="nearest"  # Filling mode
)

# **Define augmentation function**
def augment_image(image_path):
    img = cv2.imread(image_path)
    img = cv2.resize(img, IMG_SIZE)
    img = np.expand_dims(img, axis=0)  # Reshape for ImageDataGenerator input
    augmented_img = data_gen.flow(img, batch_size=1).__next__()[0]  # Generate␣
 ↪augmented image
    return augmented_img.flatten()  # Convert to 1D feature vector

# **Apply augmentation to each image**
features_augmented = np.array([augment_image(img_path) for img_path in␣
 ↪data['image']])

# **Combine original and augmented data**
features_combined = np.vstack((features, features_augmented))  # Stack original␣
 ↪and augmented features
labels_combined = np.hstack((labels_encoded, labels_encoded))  # Duplicate␣
 ↪labels for augmented images
```

Apply PCA to reduce image dimensionality

Reduce dimensionality after ensuring that principal components are learned from a richer dataset.

[ ]:
```
import os
import cv2
import numpy as np
import pandas as pd
from sklearn.decomposition import PCA
from sklearn.preprocessing import LabelEncoder
```

```
features = np.array(features)
label_encoder = LabelEncoder()
labels_encoded = label_encoder.fit_transform(labels)  # Encode labels␣
↪numerically

# Print original feature shape before PCA
print("Original feature shape:", features.shape)

# Apply PCA to reduce dimensions
pca = PCA(n_components=100)  # Adjust the number of components as needed
features_pca = pca.fit_transform(features)
```

```
Original feature shape: (3242, 67500)
```

Print out variance to decide how many components to keep.

Aim for maximize variance in PCA.

[ ]:
```
# Print the transformed feature shape after PCA
print("Transformed feature shape after PCA:", features_pca.shape)

# Print the explained variance ratio
print("Explained variance ratio by each PCA component:", pca.
↪explained_variance_ratio_)

# Optionally, print cumulative variance to see how much total variance is␣
↪captured
cumulative_variance = np.cumsum(pca.explained_variance_ratio_)
print("Cumulative explained variance:", cumulative_variance)
```

```
Transformed feature shape after PCA: (3242, 100)
Explained variance ratio by each PCA component: [0.14109165 0.02448972
0.01148891 0.00660007 0.00636416 0.0061342
 0.00581092 0.00575734 0.00548945 0.00528904 0.00513959 0.0051223
 0.0049624  0.00483812 0.00481057 0.00468573 0.00464008 0.00458454
 0.00448526 0.00443179 0.0044074  0.00433549 0.00426078 0.00424075
 0.00418621 0.00409506 0.00406601 0.00402646 0.00399933 0.00394168
 0.00392052 0.00386564 0.00384733 0.0038029  0.00377592 0.00374218
 0.00365849 0.00360302 0.00358121 0.00356901 0.00354576 0.00349275
 0.00347186 0.00342403 0.00337234 0.00336686 0.00333136 0.00331455
 0.0033066  0.00328966 0.00322935 0.00319854 0.00317572 0.00317157
 0.00313607 0.00310865 0.0030857  0.00307564 0.0030238  0.00299645
 0.00295966 0.00293287 0.0029136  0.00288877 0.00287741 0.00284059
 0.00282418 0.00280513 0.00277466 0.0027507  0.00273404 0.00272384
 0.00270374 0.00266138 0.00264275 0.00263117 0.00260615 0.00258941
 0.00256957 0.00252312 0.0025023  0.00248444 0.00244772 0.00243182
 0.00242117 0.00240851 0.00239467 0.00234497 0.00233864 0.00232603
 0.00231166 0.00230016 0.002271   0.00225883 0.0022216  0.00219037
 0.00217277 0.00214614 0.00213718 0.0021307 ]
```

```
Cumulative explained variance: [0.14109165 0.16558137 0.17707027 0.18367035
0.19003451 0.19616871
 0.20197963 0.20773697 0.21322642 0.21851546 0.22365505 0.22877736
 0.23373975 0.23857787 0.24338844 0.24807417 0.25271425 0.2572988
 0.26178405 0.26621584 0.27062324 0.27495873 0.27921951 0.28346026
 0.28764647 0.29174153 0.29580754 0.299834   0.30383333 0.30777501
 0.31169552 0.31556116 0.31940849 0.32321139 0.32698731 0.33072949
 0.33438798 0.33799101 0.34157222 0.34514122 0.34868698 0.35217973
 0.35565159 0.35907562 0.36244797 0.36581483 0.36914618 0.37246074
 0.37576733 0.37905699 0.38228633 0.38548488 0.3886606  0.39183217
 0.39496824 0.39807689 0.40116259 0.40423823 0.40726203 0.41025848
 0.41321814 0.41615101 0.41906461 0.42195338 0.42483079 0.42767139
 0.43049557 0.4333007  0.43607536 0.43882606 0.44156009 0.44428393
 0.44698767 0.44964905 0.4522918  0.45492297 0.45752912 0.46011854
 0.46268811 0.46521123 0.46771353 0.47019797 0.47264569 0.47507751
 0.47749867 0.47990719 0.48230186 0.48464684 0.48698548 0.48931151
 0.49162316 0.49392333 0.49619433 0.49845316 0.50067476 0.50286513
 0.50503789 0.50718403 0.50932121 0.51145191]
```

Apply SMOTE to solve classes imbalanced.

Check if we have imbalance problem(benign has less samples) + ensure afterwards we have balanced or not oversampled.

[ ]:
```python
from imblearn.over_sampling import SMOTE
from collections import Counter

# Check class distribution before SMOTE
print("Class distribution before SMOTE:", Counter(labels_encoded))

# Apply SMOTE
smote = SMOTE(random_state=42)
features_resampled, labels_resampled = smote.fit_resample(features_pca,
 ↪labels_encoded)

# Check class distribution after SMOTE
print("Class distribution after SMOTE:", Counter(labels_resampled))
```

```
Class distribution before SMOTE: Counter({np.int64(3): 979, np.int64(1): 955,
np.int64(2): 796, np.int64(0): 512})
Class distribution after SMOTE: Counter({np.int64(2): 979, np.int64(0): 979,
np.int64(1): 979, np.int64(3): 979})
```

Train Random Forest Model

[ ]:
```python
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report
from sklearn.model_selection import train_test_split
```

```python
# Split into train and test sets
X_train, X_test, y_train, y_test = train_test_split(features_resampled,␣
    ↪labels_resampled, test_size=0.2, random_state=42)

# Train Random Forest model
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)

# Make predictions
y_pred = rf_model.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average='weighted')  # Weighted to␣
    ↪handle multi-class
recall = recall_score(y_test, y_pred, average='weighted')        # Weighted to␣
    ↪handle multi-class
f1 = f1_score(y_test, y_pred, average='weighted')                # Weighted to␣
    ↪handle multi-class

print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1-Score:", f1)

# Evaluate the model
print("Accuracy:", accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred, target_names=label_encoder.
    ↪classes_))
```

```
Accuracy: 0.8992346938775511
Precision: 0.9010776907620733
Recall: 0.8992346938775511
F1-Score: 0.8989155212546884
Accuracy: 0.8992346938775511
                         precision    recall  f1-score   support

                 Benign       0.89      0.85      0.87       197
      [Malignant] Pre-B       0.93      0.84      0.88       188
      [Malignant] Pro-B       0.95      0.98      0.96       201
[Malignant] early Pre-B       0.83      0.92      0.88       198

               accuracy                           0.90       784
              macro avg       0.90      0.90      0.90       784
           weighted avg       0.90      0.90      0.90       784
```

[ ]:
```
# Save the model to a file
joblib.dump(rf_model, 'random_forest_model.joblib')
```

[ ]: ['random_forest_model.joblib']

Takeaways: 1. The performance of the model without using hashing is close to that of the CNN model, suggesting that hashing may not significantly contribute to improving classification accuracy in this dataset. 2. The high performance achieved highlights **the importance of data augmentation, PCA, and SMOTE techniques in enhancing model performance.** These preprocessing methods play a crucial role in addressing class imbalance and improving feature representation.

[7]:
```
!apt-get install texlive texlive-xetex texlive-latex-extra pandoc
!pip install pypandoc
```

```
Reading package lists… Done
Building dependency tree… Done
Reading state information… Done
The following additional packages will be installed:
  dvisvgm fonts-droid-fallback fonts-lato fonts-lmodern fonts-noto-mono fonts-
texgyre
  fonts-urw-base35 libapache-pom-java libcmark-gfm-extensions0.29.0.gfm.3
libcmark-gfm0.29.0.gfm.3
  libcommons-logging-java libcommons-parent-java libfontbox-java libfontenc1
libgs9 libgs9-common
  libidn12 libijs-0.35 libjbig2dec0 libkpathsea6 libpdfbox-java libptexenc1
libruby3.0 libsynctex2
  libteckit0 libtexlua53 libtexluajit2 libwoff1 libzzip-0-13 lmodern pandoc-data
poppler-data
  preview-latex-style rake ruby ruby-net-telnet ruby-rubygems ruby-webrick ruby-
xmlrpc ruby3.0
  rubygems-integration t1utils teckit tex-common tex-gyre texlive-base texlive-
binaries
  texlive-fonts-recommended texlive-latex-base texlive-latex-recommended
texlive-pictures
  texlive-plain-generic tipa xfonts-encodings xfonts-utils
Suggested packages:
  fonts-noto fonts-freefont-otf | fonts-freefont-ttf libavalon-framework-java
  libcommons-logging-java-doc libexcalibur-logkit-java liblog4j1.2-java texlive-
luatex
  pandoc-citeproc context wkhtmltopdf librsvg2-bin groff ghc nodejs php python
libjs-mathjax
  libjs-katex citation-style-language-styles poppler-utils ghostscript fonts-
japanese-mincho
  | fonts-ipafont-mincho fonts-japanese-gothic | fonts-ipafont-gothic fonts-
arphic-ukai
  fonts-arphic-uming fonts-nanum ri ruby-dev bundler debhelper gv | postscript-
viewer perl-tk xpdf
```

```
  | pdf-viewer xzdec texlive-fonts-recommended-doc texlive-latex-base-doc
python3-pygments
  icc-profiles libfile-which-perl libspreadsheet-parseexcel-perl texlive-latex-
extra-doc
  texlive-latex-recommended-doc texlive-pstricks dot2tex prerex texlive-
pictures-doc vprerex
  default-jre-headless tipa-doc
The following NEW packages will be installed:
  dvisvgm fonts-droid-fallback fonts-lato fonts-lmodern fonts-noto-mono fonts-
texgyre
  fonts-urw-base35 libapache-pom-java libcmark-gfm-extensions0.29.0.gfm.3
libcmark-gfm0.29.0.gfm.3
  libcommons-logging-java libcommons-parent-java libfontbox-java libfontenc1
libgs9 libgs9-common
  libidn12 libijs-0.35 libjbig2dec0 libkpathsea6 libpdfbox-java libptexenc1
libruby3.0 libsynctex2
  libteckit0 libtexlua53 libtexluajit2 libwoff1 libzzip-0-13 lmodern pandoc
pandoc-data
  poppler-data preview-latex-style rake ruby ruby-net-telnet ruby-rubygems ruby-
webrick ruby-xmlrpc
  ruby3.0 rubygems-integration t1utils teckit tex-common tex-gyre texlive
texlive-base
  texlive-binaries texlive-fonts-recommended texlive-latex-base texlive-latex-
extra
  texlive-latex-recommended texlive-pictures texlive-plain-generic texlive-xetex
tipa
  xfonts-encodings xfonts-utils
0 upgraded, 59 newly installed, 0 to remove and 29 not upgraded.
Need to get 202 MB of archives.
After this operation, 728 MB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu jammy/main amd64 fonts-droid-fallback all
1:6.0.1r16-1.1build1 [1,805 kB]
Get:2 http://archive.ubuntu.com/ubuntu jammy/main amd64 fonts-lato all 2.0-2.1
[2,696 kB]
Get:3 http://archive.ubuntu.com/ubuntu jammy/main amd64 poppler-data all
0.4.11-1 [2,171 kB]
Get:4 http://archive.ubuntu.com/ubuntu jammy/universe amd64 tex-common all 6.17
[33.7 kB]
Get:5 http://archive.ubuntu.com/ubuntu jammy/main amd64 fonts-urw-base35 all
20200910-1 [6,367 kB]
Get:6 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 libgs9-common
all 9.55.0~dfsg1-0ubuntu5.10 [752 kB]
Get:7 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 libidn12 amd64
1.38-4ubuntu1 [60.0 kB]
Get:8 http://archive.ubuntu.com/ubuntu jammy/main amd64 libijs-0.35 amd64
0.35-15build2 [16.5 kB]
Get:9 http://archive.ubuntu.com/ubuntu jammy/main amd64 libjbig2dec0 amd64
0.19-3build2 [64.7 kB]
```

```
Get:10 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 libgs9 amd64
9.55.0~dfsg1-0ubuntu5.10 [5,031 kB]
Get:11 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 libkpathsea6
amd64 2021.20210626.59705-1ubuntu0.2 [60.4 kB]
Get:12 http://archive.ubuntu.com/ubuntu jammy/main amd64 libwoff1 amd64
1.0.2-1build4 [45.2 kB]
Get:13 http://archive.ubuntu.com/ubuntu jammy/universe amd64 dvisvgm amd64
2.13.1-1 [1,221 kB]
Get:14 http://archive.ubuntu.com/ubuntu jammy/universe amd64 fonts-lmodern all
2.004.5-6.1 [4,532 kB]
Get:15 http://archive.ubuntu.com/ubuntu jammy/main amd64 fonts-noto-mono all
20201225-1build1 [397 kB]
Get:16 http://archive.ubuntu.com/ubuntu jammy/universe amd64 fonts-texgyre all
20180621-3.1 [10.2 MB]
Get:17 http://archive.ubuntu.com/ubuntu jammy/universe amd64 libapache-pom-java
all 18-1 [4,720 B]
Get:18 http://archive.ubuntu.com/ubuntu jammy/universe amd64 libcmark-
gfm0.29.0.gfm.3 amd64 0.29.0.gfm.3-3 [115 kB]
Get:19 http://archive.ubuntu.com/ubuntu jammy/universe amd64 libcmark-gfm-
extensions0.29.0.gfm.3 amd64 0.29.0.gfm.3-3 [25.1 kB]
Get:20 http://archive.ubuntu.com/ubuntu jammy/universe amd64 libcommons-parent-
java all 43-1 [10.8 kB]
Get:21 http://archive.ubuntu.com/ubuntu jammy/universe amd64 libcommons-logging-
java all 1.2-2 [60.3 kB]
Get:22 http://archive.ubuntu.com/ubuntu jammy/main amd64 libfontenc1 amd64
1:1.1.4-1build3 [14.7 kB]
Get:23 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 libptexenc1
amd64 2021.20210626.59705-1ubuntu0.2 [39.1 kB]
Get:24 http://archive.ubuntu.com/ubuntu jammy/main amd64 rubygems-integration
all 1.18 [5,336 B]
Get:25 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 ruby3.0 amd64
3.0.2-7ubuntu2.8 [50.1 kB]
Get:26 http://archive.ubuntu.com/ubuntu jammy/main amd64 ruby-rubygems all
3.3.5-2 [228 kB]
Get:27 http://archive.ubuntu.com/ubuntu jammy/main amd64 ruby amd64 1:3.0~exp1
[5,100 B]
Get:28 http://archive.ubuntu.com/ubuntu jammy/main amd64 rake all 13.0.6-2 [61.7
kB]
Get:29 http://archive.ubuntu.com/ubuntu jammy/main amd64 ruby-net-telnet all
0.1.1-2 [12.6 kB]
Get:30 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 ruby-webrick
all 1.7.0-3ubuntu0.1 [52.1 kB]
Get:31 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 ruby-xmlrpc all
0.3.2-1ubuntu0.1 [24.9 kB]
Get:32 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 libruby3.0
amd64 3.0.2-7ubuntu2.8 [5,113 kB]
Get:33 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 libsynctex2
amd64 2021.20210626.59705-1ubuntu0.2 [55.6 kB]
```

```
Get:34 http://archive.ubuntu.com/ubuntu jammy/universe amd64 libteckit0 amd64
2.5.11+ds1-1 [421 kB]
Get:35 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 libtexlua53
amd64 2021.20210626.59705-1ubuntu0.2 [120 kB]
Get:36 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 libtexluajit2
amd64 2021.20210626.59705-1ubuntu0.2 [267 kB]
Get:37 http://archive.ubuntu.com/ubuntu jammy/universe amd64 libzzip-0-13 amd64
0.13.72+dfsg.1-1.1 [27.0 kB]
Get:38 http://archive.ubuntu.com/ubuntu jammy/main amd64 xfonts-encodings all
1:1.0.5-0ubuntu2 [578 kB]
Get:39 http://archive.ubuntu.com/ubuntu jammy/main amd64 xfonts-utils amd64
1:7.7+6build2 [94.6 kB]
Get:40 http://archive.ubuntu.com/ubuntu jammy/universe amd64 lmodern all
2.004.5-6.1 [9,471 kB]
Get:41 http://archive.ubuntu.com/ubuntu jammy/universe amd64 pandoc-data all
2.9.2.1-3ubuntu2 [81.8 kB]
Get:42 http://archive.ubuntu.com/ubuntu jammy/universe amd64 pandoc amd64
2.9.2.1-3ubuntu2 [20.3 MB]
Get:43 http://archive.ubuntu.com/ubuntu jammy/universe amd64 preview-latex-style
all 12.2-1ubuntu1 [185 kB]
Get:44 http://archive.ubuntu.com/ubuntu jammy/main amd64 t1utils amd64
1.41-4build2 [61.3 kB]
Get:45 http://archive.ubuntu.com/ubuntu jammy/universe amd64 teckit amd64
2.5.11+ds1-1 [699 kB]
Get:46 http://archive.ubuntu.com/ubuntu jammy/universe amd64 tex-gyre all
20180621-3.1 [6,209 kB]
Get:47 http://archive.ubuntu.com/ubuntu jammy-updates/universe amd64 texlive-
binaries amd64 2021.20210626.59705-1ubuntu0.2 [9,860 kB]
Get:48 http://archive.ubuntu.com/ubuntu jammy/universe amd64 texlive-base all
2021.20220204-1 [21.0 MB]
Get:49 http://archive.ubuntu.com/ubuntu jammy/universe amd64 texlive-fonts-
recommended all 2021.20220204-1 [4,972 kB]
Get:50 http://archive.ubuntu.com/ubuntu jammy/universe amd64 texlive-latex-base
all 2021.20220204-1 [1,128 kB]
Get:51 http://archive.ubuntu.com/ubuntu jammy/universe amd64 texlive-latex-
recommended all 2021.20220204-1 [14.4 MB]
Get:52 http://archive.ubuntu.com/ubuntu jammy/universe amd64 texlive all
2021.20220204-1 [14.3 kB]
Get:53 http://archive.ubuntu.com/ubuntu jammy/universe amd64 libfontbox-java all
1:1.8.16-2 [207 kB]
Get:54 http://archive.ubuntu.com/ubuntu jammy/universe amd64 libpdfbox-java all
1:1.8.16-2 [5,199 kB]
Get:55 http://archive.ubuntu.com/ubuntu jammy/universe amd64 texlive-pictures
all 2021.20220204-1 [8,720 kB]
Get:56 http://archive.ubuntu.com/ubuntu jammy/universe amd64 texlive-latex-extra
all 2021.20220204-1 [13.9 MB]
Get:57 http://archive.ubuntu.com/ubuntu jammy/universe amd64 texlive-plain-
generic all 2021.20220204-1 [27.5 MB]
```

```
Get:58 http://archive.ubuntu.com/ubuntu jammy/universe amd64 tipa all 2:1.3-21
[2,967 kB]
Get:59 http://archive.ubuntu.com/ubuntu jammy/universe amd64 texlive-xetex all
2021.20220204-1 [12.4 MB]
Fetched 202 MB in 14s (14.1 MB/s)
Extracting templates from packages: 100%
Preconfiguring packages …
Selecting previously unselected package fonts-droid-fallback.
(Reading database … 125044 files and directories currently installed.)
Preparing to unpack …/00-fonts-droid-fallback_1%3a6.0.1r16-1.1build1_all.deb
…
Unpacking fonts-droid-fallback (1:6.0.1r16-1.1build1) …
Selecting previously unselected package fonts-lato.
Preparing to unpack …/01-fonts-lato_2.0-2.1_all.deb …
Unpacking fonts-lato (2.0-2.1) …
Selecting previously unselected package poppler-data.
Preparing to unpack …/02-poppler-data_0.4.11-1_all.deb …
Unpacking poppler-data (0.4.11-1) …
Selecting previously unselected package tex-common.
Preparing to unpack …/03-tex-common_6.17_all.deb …
Unpacking tex-common (6.17) …
Selecting previously unselected package fonts-urw-base35.
Preparing to unpack …/04-fonts-urw-base35_20200910-1_all.deb …
Unpacking fonts-urw-base35 (20200910-1) …
Selecting previously unselected package libgs9-common.
Preparing to unpack …/05-libgs9-common_9.55.0~dfsg1-0ubuntu5.10_all.deb …
Unpacking libgs9-common (9.55.0~dfsg1-0ubuntu5.10) …
Selecting previously unselected package libidn12:amd64.
Preparing to unpack …/06-libidn12_1.38-4ubuntu1_amd64.deb …
Unpacking libidn12:amd64 (1.38-4ubuntu1) …
Selecting previously unselected package libijs-0.35:amd64.
Preparing to unpack …/07-libijs-0.35_0.35-15build2_amd64.deb …
Unpacking libijs-0.35:amd64 (0.35-15build2) …
Selecting previously unselected package libjbig2dec0:amd64.
Preparing to unpack …/08-libjbig2dec0_0.19-3build2_amd64.deb …
Unpacking libjbig2dec0:amd64 (0.19-3build2) …
Selecting previously unselected package libgs9:amd64.
Preparing to unpack …/09-libgs9_9.55.0~dfsg1-0ubuntu5.10_amd64.deb …
Unpacking libgs9:amd64 (9.55.0~dfsg1-0ubuntu5.10) …
Selecting previously unselected package libkpathsea6:amd64.
Preparing to unpack …/10-libkpathsea6_2021.20210626.59705-1ubuntu0.2_amd64.deb
…
Unpacking libkpathsea6:amd64 (2021.20210626.59705-1ubuntu0.2) …
Selecting previously unselected package libwoff1:amd64.
Preparing to unpack …/11-libwoff1_1.0.2-1build4_amd64.deb …
Unpacking libwoff1:amd64 (1.0.2-1build4) …
Selecting previously unselected package dvisvgm.
Preparing to unpack …/12-dvisvgm_2.13.1-1_amd64.deb …
```

```
Unpacking dvisvgm (2.13.1-1) …
Selecting previously unselected package fonts-lmodern.
Preparing to unpack …/13-fonts-lmodern_2.004.5-6.1_all.deb …
Unpacking fonts-lmodern (2.004.5-6.1) …
Selecting previously unselected package fonts-noto-mono.
Preparing to unpack …/14-fonts-noto-mono_20201225-1build1_all.deb …
Unpacking fonts-noto-mono (20201225-1build1) …
Selecting previously unselected package fonts-texgyre.
Preparing to unpack …/15-fonts-texgyre_20180621-3.1_all.deb …
Unpacking fonts-texgyre (20180621-3.1) …
Selecting previously unselected package libapache-pom-java.
Preparing to unpack …/16-libapache-pom-java_18-1_all.deb …
Unpacking libapache-pom-java (18-1) …
Selecting previously unselected package libcmark-gfm0.29.0.gfm.3:amd64.
Preparing to unpack …/17-libcmark-gfm0.29.0.gfm.3_0.29.0.gfm.3-3_amd64.deb …
Unpacking libcmark-gfm0.29.0.gfm.3:amd64 (0.29.0.gfm.3-3) …
Selecting previously unselected package libcmark-gfm-
extensions0.29.0.gfm.3:amd64.
Preparing to unpack …/18-libcmark-gfm-
extensions0.29.0.gfm.3_0.29.0.gfm.3-3_amd64.deb …
Unpacking libcmark-gfm-extensions0.29.0.gfm.3:amd64 (0.29.0.gfm.3-3) …
Selecting previously unselected package libcommons-parent-java.
Preparing to unpack …/19-libcommons-parent-java_43-1_all.deb …
Unpacking libcommons-parent-java (43-1) …
Selecting previously unselected package libcommons-logging-java.
Preparing to unpack …/20-libcommons-logging-java_1.2-2_all.deb …
Unpacking libcommons-logging-java (1.2-2) …
Selecting previously unselected package libfontenc1:amd64.
Preparing to unpack …/21-libfontenc1_1%3a1.1.4-1build3_amd64.deb …
Unpacking libfontenc1:amd64 (1:1.1.4-1build3) …
Selecting previously unselected package libptexenc1:amd64.
Preparing to unpack …/22-libptexenc1_2021.20210626.59705-1ubuntu0.2_amd64.deb
…
Unpacking libptexenc1:amd64 (2021.20210626.59705-1ubuntu0.2) …
Selecting previously unselected package rubygems-integration.
Preparing to unpack …/23-rubygems-integration_1.18_all.deb …
Unpacking rubygems-integration (1.18) …
Selecting previously unselected package ruby3.0.
Preparing to unpack …/24-ruby3.0_3.0.2-7ubuntu2.8_amd64.deb …
Unpacking ruby3.0 (3.0.2-7ubuntu2.8) …
Selecting previously unselected package ruby-rubygems.
Preparing to unpack …/25-ruby-rubygems_3.3.5-2_all.deb …
Unpacking ruby-rubygems (3.3.5-2) …
Selecting previously unselected package ruby.
Preparing to unpack …/26-ruby_1%3a3.0~exp1_amd64.deb …
Unpacking ruby (1:3.0~exp1) …
Selecting previously unselected package rake.
Preparing to unpack …/27-rake_13.0.6-2_all.deb …
```

```
Unpacking rake (13.0.6-2) …
Selecting previously unselected package ruby-net-telnet.
Preparing to unpack …/28-ruby-net-telnet_0.1.1-2_all.deb …
Unpacking ruby-net-telnet (0.1.1-2) …
Selecting previously unselected package ruby-webrick.
Preparing to unpack …/29-ruby-webrick_1.7.0-3ubuntu0.1_all.deb …
Unpacking ruby-webrick (1.7.0-3ubuntu0.1) …
Selecting previously unselected package ruby-xmlrpc.
Preparing to unpack …/30-ruby-xmlrpc_0.3.2-1ubuntu0.1_all.deb …
Unpacking ruby-xmlrpc (0.3.2-1ubuntu0.1) …
Selecting previously unselected package libruby3.0:amd64.
Preparing to unpack …/31-libruby3.0_3.0.2-7ubuntu2.8_amd64.deb …
Unpacking libruby3.0:amd64 (3.0.2-7ubuntu2.8) …
Selecting previously unselected package libsynctex2:amd64.
Preparing to unpack …/32-libsynctex2_2021.20210626.59705-1ubuntu0.2_amd64.deb
…
Unpacking libsynctex2:amd64 (2021.20210626.59705-1ubuntu0.2) …
Selecting previously unselected package libteckit0:amd64.
Preparing to unpack …/33-libteckit0_2.5.11+ds1-1_amd64.deb …
Unpacking libteckit0:amd64 (2.5.11+ds1-1) …
Selecting previously unselected package libtexlua53:amd64.
Preparing to unpack …/34-libtexlua53_2021.20210626.59705-1ubuntu0.2_amd64.deb
…
Unpacking libtexlua53:amd64 (2021.20210626.59705-1ubuntu0.2) …
Selecting previously unselected package libtexluajit2:amd64.
Preparing to unpack
…/35-libtexluajit2_2021.20210626.59705-1ubuntu0.2_amd64.deb …
Unpacking libtexluajit2:amd64 (2021.20210626.59705-1ubuntu0.2) …
Selecting previously unselected package libzzip-0-13:amd64.
Preparing to unpack …/36-libzzip-0-13_0.13.72+dfsg.1-1.1_amd64.deb …
Unpacking libzzip-0-13:amd64 (0.13.72+dfsg.1-1.1) …
Selecting previously unselected package xfonts-encodings.
Preparing to unpack …/37-xfonts-encodings_1%3a1.0.5-0ubuntu2_all.deb …
Unpacking xfonts-encodings (1:1.0.5-0ubuntu2) …
Selecting previously unselected package xfonts-utils.
Preparing to unpack …/38-xfonts-utils_1%3a7.7+6build2_amd64.deb …
Unpacking xfonts-utils (1:7.7+6build2) …
Selecting previously unselected package lmodern.
Preparing to unpack …/39-lmodern_2.004.5-6.1_all.deb …
Unpacking lmodern (2.004.5-6.1) …
Selecting previously unselected package pandoc-data.
Preparing to unpack …/40-pandoc-data_2.9.2.1-3ubuntu2_all.deb …
Unpacking pandoc-data (2.9.2.1-3ubuntu2) …
Selecting previously unselected package pandoc.
Preparing to unpack …/41-pandoc_2.9.2.1-3ubuntu2_amd64.deb …
Unpacking pandoc (2.9.2.1-3ubuntu2) …
Selecting previously unselected package preview-latex-style.
Preparing to unpack …/42-preview-latex-style_12.2-1ubuntu1_all.deb …
```

```
Unpacking preview-latex-style (12.2-1ubuntu1) …
Selecting previously unselected package t1utils.
Preparing to unpack …/43-t1utils_1.41-4build2_amd64.deb …
Unpacking t1utils (1.41-4build2) …
Selecting previously unselected package teckit.
Preparing to unpack …/44-teckit_2.5.11+ds1-1_amd64.deb …
Unpacking teckit (2.5.11+ds1-1) …
Selecting previously unselected package tex-gyre.
Preparing to unpack …/45-tex-gyre_20180621-3.1_all.deb …
Unpacking tex-gyre (20180621-3.1) …
Selecting previously unselected package texlive-binaries.
Preparing to unpack …/46-texlive-
binaries_2021.20210626.59705-1ubuntu0.2_amd64.deb …
Unpacking texlive-binaries (2021.20210626.59705-1ubuntu0.2) …
Selecting previously unselected package texlive-base.
Preparing to unpack …/47-texlive-base_2021.20220204-1_all.deb …
Unpacking texlive-base (2021.20220204-1) …
Selecting previously unselected package texlive-fonts-recommended.
Preparing to unpack …/48-texlive-fonts-recommended_2021.20220204-1_all.deb …
Unpacking texlive-fonts-recommended (2021.20220204-1) …
Selecting previously unselected package texlive-latex-base.
Preparing to unpack …/49-texlive-latex-base_2021.20220204-1_all.deb …
Unpacking texlive-latex-base (2021.20220204-1) …
Selecting previously unselected package texlive-latex-recommended.
Preparing to unpack …/50-texlive-latex-recommended_2021.20220204-1_all.deb …
Unpacking texlive-latex-recommended (2021.20220204-1) …
Selecting previously unselected package texlive.
Preparing to unpack …/51-texlive_2021.20220204-1_all.deb …
Unpacking texlive (2021.20220204-1) …
Selecting previously unselected package libfontbox-java.
Preparing to unpack …/52-libfontbox-java_1%3a1.8.16-2_all.deb …
Unpacking libfontbox-java (1:1.8.16-2) …
Selecting previously unselected package libpdfbox-java.
Preparing to unpack …/53-libpdfbox-java_1%3a1.8.16-2_all.deb …
Unpacking libpdfbox-java (1:1.8.16-2) …
Selecting previously unselected package texlive-pictures.
Preparing to unpack …/54-texlive-pictures_2021.20220204-1_all.deb …
Unpacking texlive-pictures (2021.20220204-1) …
Selecting previously unselected package texlive-latex-extra.
Preparing to unpack …/55-texlive-latex-extra_2021.20220204-1_all.deb …
Unpacking texlive-latex-extra (2021.20220204-1) …
Selecting previously unselected package texlive-plain-generic.
Preparing to unpack …/56-texlive-plain-generic_2021.20220204-1_all.deb …
Unpacking texlive-plain-generic (2021.20220204-1) …
Selecting previously unselected package tipa.
Preparing to unpack …/57-tipa_2%3a1.3-21_all.deb …
Unpacking tipa (2:1.3-21) …
Selecting previously unselected package texlive-xetex.
```

```
Preparing to unpack …/58-texlive-xetex_2021.20220204-1_all.deb …
Unpacking texlive-xetex (2021.20220204-1) …
Setting up fonts-lato (2.0-2.1) …
Setting up fonts-noto-mono (20201225-1build1) …
Setting up libwoff1:amd64 (1.0.2-1build4) …
Setting up libtexlua53:amd64 (2021.20210626.59705-1ubuntu0.2) …
Setting up libijs-0.35:amd64 (0.35-15build2) …
Setting up libtexluajit2:amd64 (2021.20210626.59705-1ubuntu0.2) …
Setting up libfontbox-java (1:1.8.16-2) …
Setting up rubygems-integration (1.18) …
Setting up libzzip-0-13:amd64 (0.13.72+dfsg.1-1.1) …
Setting up fonts-urw-base35 (20200910-1) …
Setting up poppler-data (0.4.11-1) …
Setting up tex-common (6.17) …
update-language: texlive-base not installed and configured, doing nothing!
Setting up libfontenc1:amd64 (1:1.1.4-1build3) …
Setting up libjbig2dec0:amd64 (0.19-3build2) …
Setting up libteckit0:amd64 (2.5.11+ds1-1) …
Setting up libapache-pom-java (18-1) …
Setting up ruby-net-telnet (0.1.1-2) …
Setting up xfonts-encodings (1:1.0.5-0ubuntu2) …
Setting up t1utils (1.41-4build2) …
Setting up libidn12:amd64 (1.38-4ubuntu1) …
Setting up fonts-texgyre (20180621-3.1) …
Setting up libkpathsea6:amd64 (2021.20210626.59705-1ubuntu0.2) …
Setting up ruby-webrick (1.7.0-3ubuntu0.1) …
Setting up libcmark-gfm0.29.0.gfm.3:amd64 (0.29.0.gfm.3-3) …
Setting up fonts-lmodern (2.004.5-6.1) …
Setting up libcmark-gfm-extensions0.29.0.gfm.3:amd64 (0.29.0.gfm.3-3) …
Setting up fonts-droid-fallback (1:6.0.1r16-1.1build1) …
Setting up pandoc-data (2.9.2.1-3ubuntu2) …
Setting up ruby-xmlrpc (0.3.2-1ubuntu0.1) …
Setting up libsynctex2:amd64 (2021.20210626.59705-1ubuntu0.2) …
Setting up libgs9-common (9.55.0~dfsg1-0ubuntu5.10) …
Setting up teckit (2.5.11+ds1-1) …
Setting up libpdfbox-java (1:1.8.16-2) …
Setting up libgs9:amd64 (9.55.0~dfsg1-0ubuntu5.10) …
Setting up preview-latex-style (12.2-1ubuntu1) …
Setting up libcommons-parent-java (43-1) …
Setting up dvisvgm (2.13.1-1) …
Setting up libcommons-logging-java (1.2-2) …
Setting up xfonts-utils (1:7.7+6build2) …
Setting up libptexenc1:amd64 (2021.20210626.59705-1ubuntu0.2) …
Setting up pandoc (2.9.2.1-3ubuntu2) …
Setting up texlive-binaries (2021.20210626.59705-1ubuntu0.2) …
update-alternatives: using /usr/bin/xdvi-xaw to provide /usr/bin/xdvi.bin
(xdvi.bin) in auto mode
update-alternatives: using /usr/bin/bibtex.original to provide /usr/bin/bibtex
```

```
(bibtex) in auto mode
Setting up lmodern (2.004.5-6.1) …
Setting up texlive-base (2021.20220204-1) …
/usr/bin/ucfr
/usr/bin/ucfr
/usr/bin/ucfr
/usr/bin/ucfr
mktexlsr: Updating /var/lib/texmf/ls-R-TEXLIVEDIST…
mktexlsr: Updating /var/lib/texmf/ls-R-TEXMFMAIN…
mktexlsr: Updating /var/lib/texmf/ls-R…
mktexlsr: Done.
tl-paper: setting paper size for dvips to a4:
/var/lib/texmf/dvips/config/config-paper.ps
tl-paper: setting paper size for dvipdfmx to a4:
/var/lib/texmf/dvipdfmx/dvipdfmx-paper.cfg
tl-paper: setting paper size for xdvi to a4: /var/lib/texmf/xdvi/XDvi-paper
tl-paper: setting paper size for pdftex to a4: /var/lib/texmf/tex/generic/tex-
ini-files/pdftexconfig.tex
Setting up tex-gyre (20180621-3.1) …
Setting up texlive-plain-generic (2021.20220204-1) …
Setting up texlive-latex-base (2021.20220204-1) …
Setting up texlive-latex-recommended (2021.20220204-1) …
Setting up texlive-pictures (2021.20220204-1) …
Setting up texlive-fonts-recommended (2021.20220204-1) …
Setting up tipa (2:1.3-21) …
Setting up texlive (2021.20220204-1) …
Setting up texlive-latex-extra (2021.20220204-1) …
Setting up texlive-xetex (2021.20220204-1) …
Setting up rake (13.0.6-2) …
Setting up libruby3.0:amd64 (3.0.2-7ubuntu2.8) …
Setting up ruby3.0 (3.0.2-7ubuntu2.8) …
Setting up ruby (1:3.0~exp1) …
Setting up ruby-rubygems (3.3.5-2) …
Processing triggers for man-db (2.10.2-1) …
Processing triggers for mailcap (3.70+nmu1ubuntu1) …
Processing triggers for fontconfig (2.13.1-4.2ubuntu5) …
Processing triggers for libc-bin (2.35-0ubuntu3.8) …
/sbin/ldconfig.real: /usr/local/lib/libumf.so.0 is not a symbolic link

/sbin/ldconfig.real: /usr/local/lib/libtbb.so.12 is not a symbolic link

/sbin/ldconfig.real: /usr/local/lib/libtbbbind_2_0.so.3 is not a symbolic link

/sbin/ldconfig.real: /usr/local/lib/libur_loader.so.0 is not a symbolic link

/sbin/ldconfig.real: /usr/local/lib/libtcm.so.1 is not a symbolic link

/sbin/ldconfig.real: /usr/local/lib/libtbbbind.so.3 is not a symbolic link
```

```
/sbin/ldconfig.real: /usr/local/lib/libur_adapter_level_zero.so.0 is not a
symbolic link

/sbin/ldconfig.real: /usr/local/lib/libtbbmalloc.so.2 is not a symbolic link

/sbin/ldconfig.real: /usr/local/lib/libhwloc.so.15 is not a symbolic link

/sbin/ldconfig.real: /usr/local/lib/libtcm_debug.so.1 is not a symbolic link

/sbin/ldconfig.real: /usr/local/lib/libtbbbind_2_5.so.3 is not a symbolic link

/sbin/ldconfig.real: /usr/local/lib/libur_adapter_opencl.so.0 is not a symbolic
link

/sbin/ldconfig.real: /usr/local/lib/libtbbmalloc_proxy.so.2 is not a symbolic
link

Processing triggers for tex-common (6.17) …
Running updmap-sys. This may take some time… done.
Running mktexlsr /var/lib/texmf … done.
Building format(s) --all.
        This may take some time… done.
Collecting pypandoc
  Downloading pypandoc-1.15-py3-none-any.whl.metadata (16 kB)
Downloading pypandoc-1.15-py3-none-any.whl (21 kB)
Installing collected packages: pypandoc
Successfully installed pypandoc-1.15
```

[8]:
```
from google.colab import drive
drive.mount('/content/drive')
```

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

[16]:
```
!cp drive/MyDrive/Colab Notebooks/Data_Mining_Project.ipynb ./
```

```
cp: cannot stat 'drive/MyDrive/Colab': No such file or directory
cp: cannot stat 'Notebooks/Data_Mining_Project.ipynb': No such file or directory
```

[ ]:
```
!jupyter nbconvert --to PDF "Data_Mining_Project.ipynb"
```