# **Final Project:**

# Image Recognition Applied to Waste Classification

IEOR242

Group Members:

Pedro Errazuriz
Fellipe Marcellino
Carlos Núñez
Maria Jesus Perez
Francesco Piccoli

## Table of Contents

## Introduction

The last century has seen a significant rise in population and production levels, which has led to an increase in environmental impact and waste. Although global awareness of environmental impact has been increasing in the last decade, actions of limited impact have caused effective recycling rates to remain stagnant. According to the Environmental Protection Agency (EPA), the difference in 2014 and 2015 effective recycling rates was less than one percent [1]. Also, just 35% of the total waste production in the US was recycled or composted in 2015.

We attribute the slowdown in recycling to the fact that recycling is an effortful and time-consuming task. This assumption is based on the results of a survey we conducted with 125 responses to people of all ages across the globe. Most people do not take the time to properly segment their trash into specific recycling bins. For example, in the survey we conducted, more than 46% of respondents said they have thrown trash into the landfill bin even if the item could be recycled. On the other hand, even if people want to recycle, the appropriate bin can be confusing and lead to contaminated bins and ineffective recycling. The question comes naturally: can we improve our current recycling rates using machine learning?

## Data Processing

For our project, we chose to utilize a dataset of 2,527 waste images from a GitHub spanning six classes: plastic (482), metal (410), glass (501), cardboard (403), trash (137), and paper (594) [2]. A sample of this dataset can be seen in **Figure 1** in the appendix. The choice of images as our data type and implementation of multiple types of models meant onerous data processing. First, we split our data into 50% training, 25% validation and 25% test. We chose to perform data augmentation which transformed our dataset to reduce overfitting and increase predictive power. These transformations included width and height shift range, flipping, zooming, and brightness changes to increase the data set ten times. In order to increase computational speed, we also resized the images from 512x384 to 200x200 pixels.

The first four models including CART, random forest, gradient boosting, and XGBoost required vectorized versions of the images. For this purpose we wrote a for loop that opens the images, creates arrays with columns corresponding to red, green, and blue layers, and then vectorizes these matrices. We then combined these vectors into a 25,270 by 120,000 dataframe whose rows correspond to images and columns to features. We realized 120,000 features is a very significant number that would lead to computational problems later on. Therefore, we finished by performing principal component analysis on the data set to reduce the number of features from 120,000 to 25,270. The maximum number of principal components must be less than the rank of the matrix which in this case is limited by the number of observations. We also converted our data labels into integers from 1 through 6 corresponding to the order mentioned above.

For the other models, including the neural network (NN), convolutional neural network (CNN), and CNN with transfer learning, we took advantage of built-in functions to handle the data processing more effectively. For example, we used python functions from Keras and Pytorch that handled processing such as conversion from image to vector for the neural network or tensor for the convolutional neural network and scaling in the range from zero to one.

## Implemented Models

Once our data was processed we decided on a series of models to fit our dataset and accuracy whose formula is shown below as the best measure of predictive performance. This is because our data set was fairly balanced and our specific application for waste classification calls for equal balancing of the categories. In fact, we are equally interested in maximizing correct predictions and minimizing wrong predictions for each of the six different types of waste.

$$Accuracy = \frac{True\ Positives + True\ Negatives}{Total\ Predicted}$$

### CART

The first model we explored was a simple classification tree. The initial implementation, with default parameters including the Gini impurity function, no minimum impurity decrease, and a lower limit for the number of observations in each bucket of 2 yielded an accuracy of 45.11%. We then performed 3-fold cross validation on the technical parameters and obtained an accuracy of 46.24% with no minimum impurity decrease and a lower limit for the number of observations of 6.

### Random Forest

The second model implemented was a random forest, which with default parameters of 10 CART trees built, sqrt(features) variables examined at each split, and one minimum number of observations in each terminal node obtained an accuracy of 60.87%. We then performed a 3-fold cross validation procedure to obtain technical parameters of 1100 CART trees built, 200 features examined at each split, and 5 minimum number of observations in each terminal node and obtained an increased accuracy of 68.90%. **Figures 2 and 3** in the appendix show the confusion matrices for the cross validated CART and random forest models respectively.

### Boosting

The next models analyzed were two types of ensemble boosting models that rely on slow learning rather than building parallel complete tree models. The gradient boosting model with achieved an accuracy of 66.79% while the extreme gradient boosting obtained an accuracy of 66.27%. Although XGBoost utilizes second order gradients and regularization to improve computation time, the multiple hour runtimes of the default models led us to avoid cross validation. Seeing the accuracies for models that require a vectorized image format stagnated below 70% even as model complexity increased, we chose to shift to models that could accept regular images. **Figures 4 and 5** in the appendix show the confusion matrices for the Gradient Boosting and XGBoost models respectively.

### Neural Network

We started by building a simple Neural Network using Keras. We initially obtained oddly low results on the NN. The model was always predicting the same class for each observation because the number of parameters was too large. By tuning the number of epochs and dense layers we were not able to obtain an accuracy better than 47.43%. This was accomplished using a batch size of 32, 250 epochs and 4 dense layers with Relu and Softmax as activation functions.

## CNN

Secondly, we moved to a Convolutional Neural Network (CNN) model that historically has better performances in classifying images. By changing the number of epochs and exploring different structures of the network, we obtained the best performances after 80 epochs, for a CNN with a batch size of 32, 3 2D convolution layers, 3 pooling layers, and 3 dense layers. We used Relu as the activation function for each layer, apart from the last one where we used the Softmax. The obtained accuracy was 71.54% which is higher than the ones obtained with any other previous model.

## CNN with Transfer Learning

Lastly, we implemented a CNN with Transfer Learning using Pytorch and the Fast.ai library [3]. For this model, we used Resnet34, a famous deep residual network pre-trained on ImageNet, a dataset with 14 million images and 22 thousand different classes. This pre-trained model can be very good with low level features related to images, but not necessarily with specific images of waste. Therefore, we added a final layer and trained it to make the model perform well on waste classification. With a batch size of 16, 20 epochs and a tuned learning rate, we obtained an accuracy of 91.29%, which makes this final model outperforming any previous one. **Figures 6, 7, and 8** in the appendix show the confusion matrices for the NN, CNN, and CNN with Transfer Learning models respectively.

The below summarizes the predictive performances of our models in order of increasing accuracy. We chose to compute 95% confidence intervals to ensure the quality and reproducibility of our results.

***Table 1**: Summary of Models and Accuracies*

| **Model** | **Accuracy** | **95% Confidence Interval** |
|---|---|---|
| CART | 46.24% | (42.7%,50%) |
| Neural Network | 47.43% | (43.4%,51.1%) |
| XGBoost | 66.27% | (63.0%,69.8%) |
| Gradient Boosting | 66.79% | (62.6%,70.4%) |
| Random Forest | 68.90% | (65.3%,72.6%) |
| CNN | 71.54% | (68.0%,74.9%) |
| CNN with transfer learning | 91.29% | (89.3%,93.4%) |

## Conclusion & Impact

Our best model gives us an accuracy of 91.3% on new observations. Compared with the actual recycling rates in the US (35%), the high accuracy reveals promising results and potential for adaptability. Our idea is to implement these results into an automatic recycling bin that recognizes, classifies, and separates trash into different types of recycling bins instantly. Using computer vision, the bin will be able to recognize if the trash is recyclable or not and store it accordingly. The person will throw trash away in the same fashion as today, and the bin will do the rest. If a successful waste sorting occurs in the beginning of the recycling chain, there will be less contamination and recycling facilities are able to drastically increase their recycling rates leading to a significantly lower environmental impact for humankind. According to EPA, 52.5% of the total 2015 waste production in the US went into landfill. With our solution, this number has the potential to be significantly decreased.

## Next steps

Our final model has an accuracy of 91.3%, which is very promising. However, as we can see in **Figure 8** in the appendix, the model performed better for some classes than others. For example, the model performed the best with images of paper and the worst with images of trash, which had the smallest sample size. This means that we still have room for improvement regarding some classes. If we perform error analysis, we can identify some patterns in the wrong predictions and add more images to the training set. For example, in **Figure 9** we see that images with shiny elements were wrongly predicted as metal, probably because the model interprets luminosity in pixels as a strong predictor for metal.

Moreover, our model performed very well on our test set, but we don't know how it will perform in real life. Then, it would be useful to test the model with real-world images that we collect with a camera installed in the bin, for example. We should also decide if we focus on one object at a time or if we perform multiple object detection in case someone throws more than one type of waste at the same time. Regarding the classes, if we are to apply the model in real-life, we need to adapt them to the local law. For example, in California our classes would be Landfill, Compost, Cans & Bottles and Mixed Paper. Finally, we can replace accuracy by average loss as our evaluation metric. For that, we need to estimate the loss of incorrectly classifying our classes, which can be linked to the potential revenue we could make by correctly sorting the waste.

## References

[1] "National Overview: Facts and Figures on Materials, Wastes and Recycling." *EPA*, Environmental Protection Agency, 3 Dec. 2019, www.epa.gov/facts-and-figures-about-materials-waste-and-recycling/national-overview-facts-and-figures-materials.

[2] Garythung. "Garythung/Trashnet." *GitHub*, 10 Apr. 2017, github.com/garythung/trashnet.

[3] "Welcome to Fastai." *Fastai*, docs.fast.ai/.

# Appendix

## Part 1: Figures

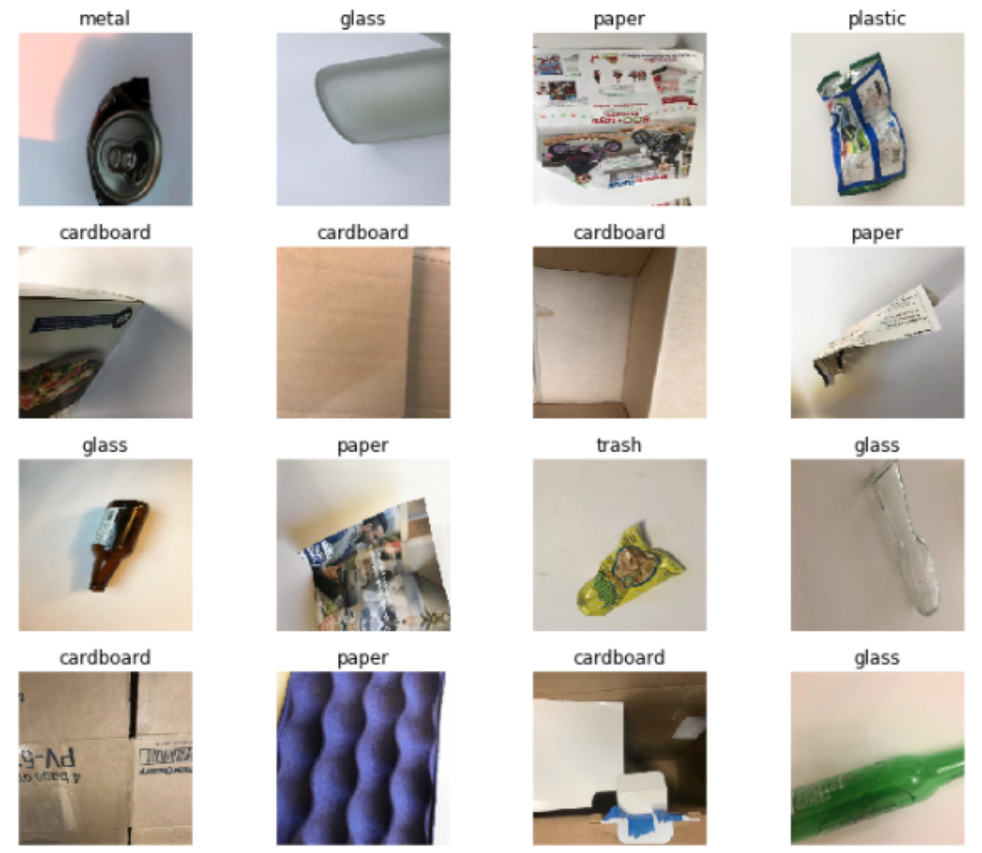

Figure 1: Sample of the image dataset

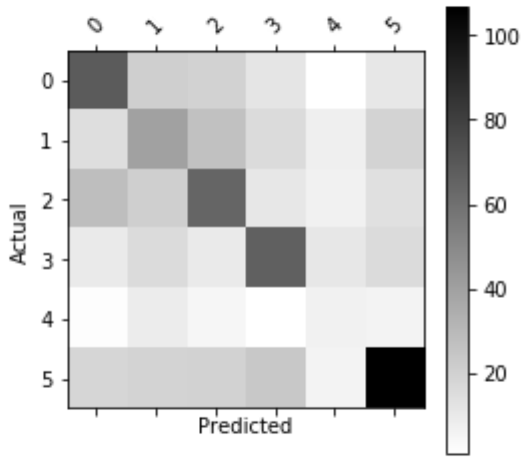

Figure 2: CART confusion matrix

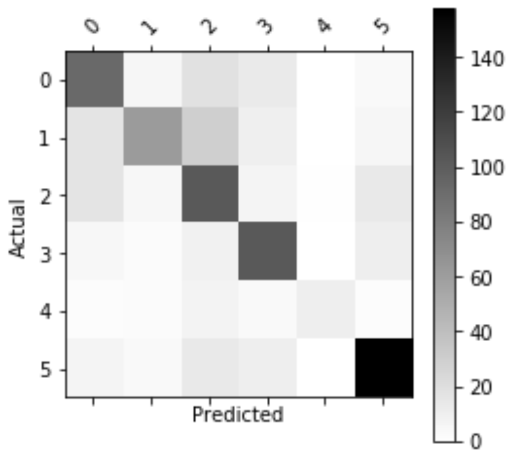

Figure 3: Random Forest confusion matrix

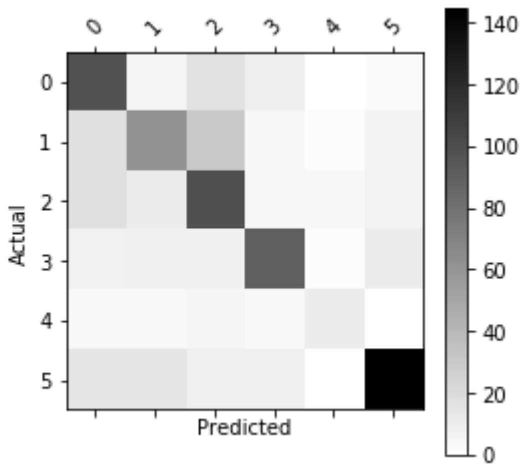

Figure 4: Gradient Boosting confusion matrix

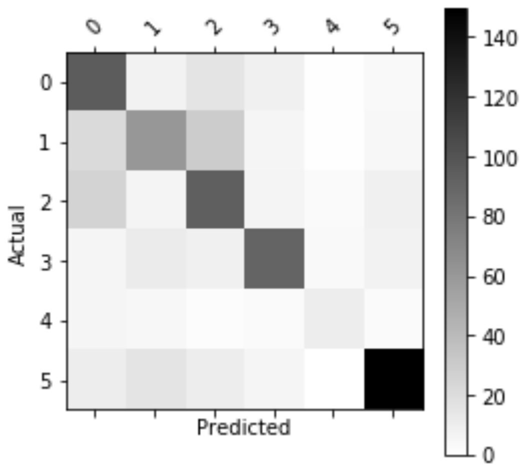

Figure 5: XGBoost confusion matrix

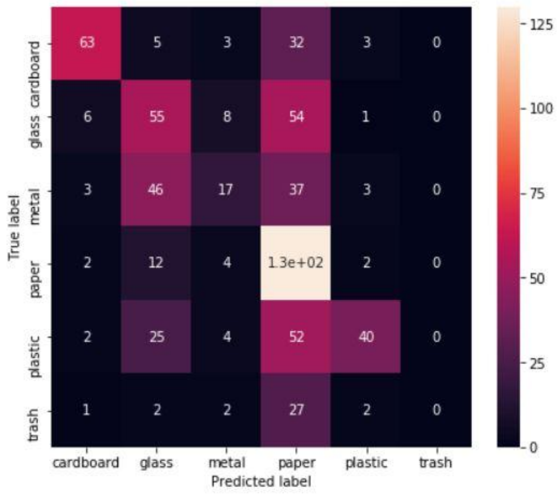

Figure 6: Neural Network confusion matrix

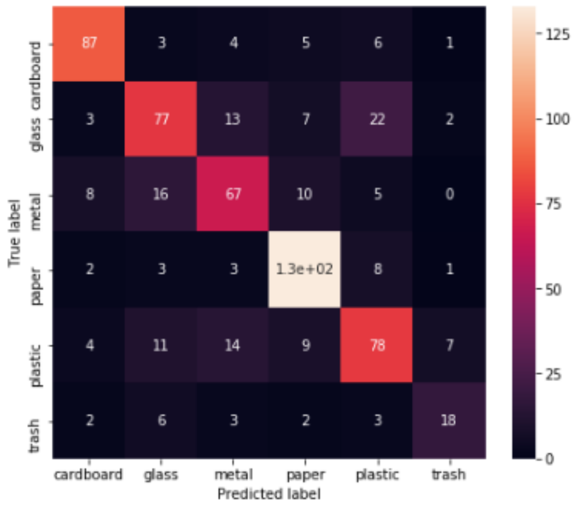

Figure 7: CNN matrix

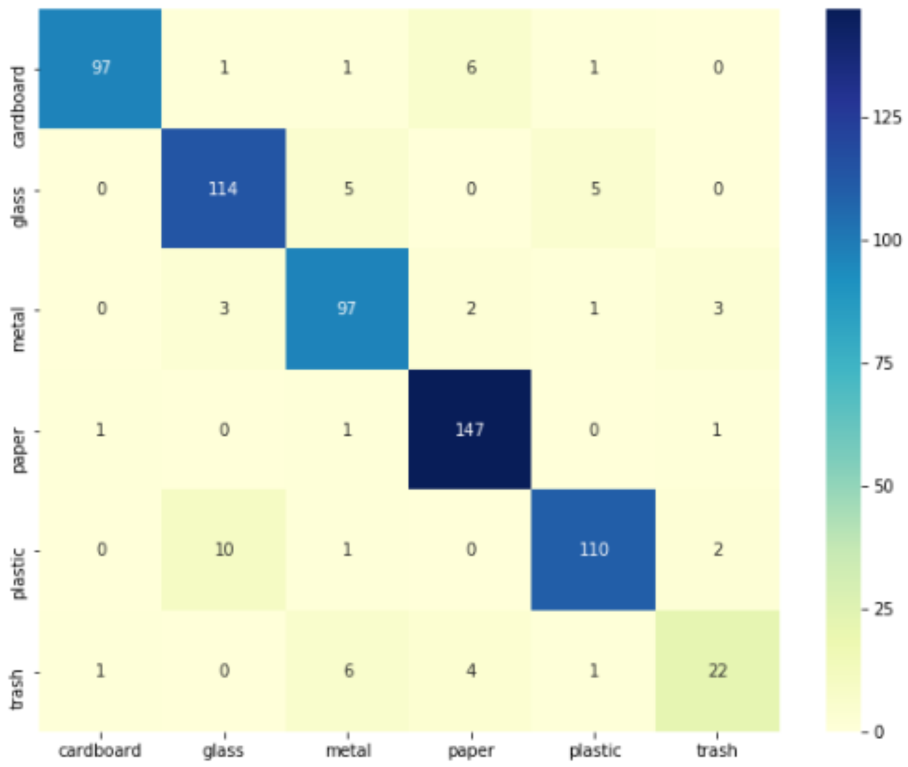

Figure 8: CNN with Transfer Learning matrix

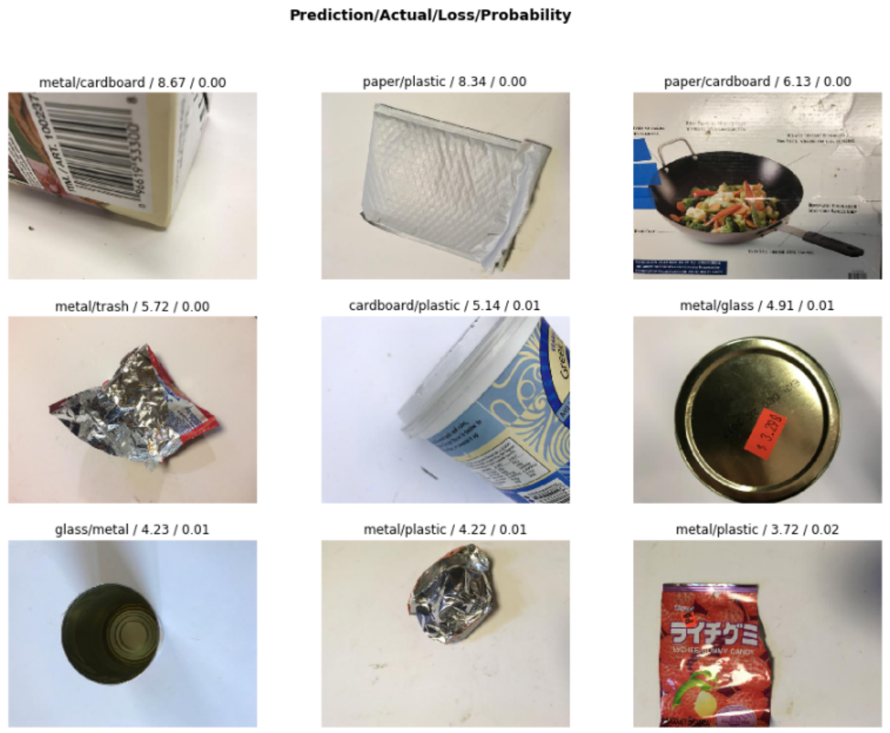
Prediction/Actual/Loss/Probability
metal/cardboard / 8.67 / 0.00
paper/plastic / 8.34 / 0.00
paper/cardboard / 6.13 / 0.00
metal/trash / 5.72 / 0.00
cardboard/plastic / 5.14 / 0.01
metal/glass / 4.91 / 0.01
glass/metal / 4.23 / 0.01
metal/plastic / 4.22 / 0.01
metal/plastic / 3.72 / 0.02

Figure 9: Error analysis for final model

## Part 2: Code

All our models were built in Python.

### *Code for Image Alteration*

```python
import numpy as np
from keras.preprocessing.image import load_img
from keras.preprocessing.image import img_to_array
from keras.preprocessing.image import ImageDataGenerator
import matplotlib.pyplot as plt
from PIL import *
import os

for folderName, subfolders, filenames in os.walk('C:\\Users\\Carlos\\PycharmProjects\\Data_Augmentation\\dataset-resized'):
    print('The current folder is ' + folderName)

    for subfolder in subfolders:
        print('SUBFOLDER OF ' + folderName + ': ' + subfolder)
    for filename in filenames:
        print('FILE INSIDE ' + folderName + ': '+ filename)
        img = load_img(folderName + '\\' + filename)
        data = img_to_array(img)
        samples = np.expand_dims(data, 0)
        datagen = ImageDataGenerator(width_shift_range=[-200, 200], height_shift_range=0.5, horizontal_flip=True,
                                     rotation_range=180, brightness_range=[0.2, 1.0], zoom_range=[0.5, 1.0])
        it = datagen.flow(samples, batch_size=1)
        for i in range(9):
            batch = it.next()
            image = batch[0].astype('uint8')
            word = filename.split(sep='.')
            image_real = Image.fromarray(image, 'RGB')
            image_real.save(folderName + '\\' + word[0] + '_' + str(i) + '.jpg')

    print('')
```

### *Code for Train/Validation/Test split*

```python
import numpy as np
import os
from google.colab import drive

drive.mount('/content/drive', force_remount = True)
labels = os.listdir("drive/My Drive/dataset-resized")
path = "drive/My Drive/"
for label in labels:
    origin = path + "dataset-resized/" + label + "/"
    folder_size = len(os.listdir(origin))
    print(folder_size)
    seq = range(1,folder_size+1)
    sets = np.random.choice(["train","validation","test"], folder_size, p=[0.5,
0.25, 0.25])
    print(label)
    for sq, st in zip(seq, sets):
      destination = path + "data-split/" + st + "/" + label + "/"
      filename = label + str(sq) + ".jpg"
      os.popen("cp '" + origin + filename + "' '" + destination + filename + "'")
```

## *Code for Image Vectorization*

```
#Defining Formula to Vectorize Images
def create_features(img):
    # flatten three channel color image
    color_features = img.flatten()
    # convert image to greyscale
    grey_image = rgb2grey(img)
    # get HOG features from greyscale image
    hog_features = hog(grey_image, block_norm='L2-Hys', pixels_per_cell=(8, 8))
    # combine color and hog features into a single array
    flat_features = np.hstack(color_features)
    return flat_features

#Creating Image Features
labels=[]
features=[]

for j,folder in enumerate(os.listdir(directory)):
For i, filename in enumerate(os.listdir(os.path.join(directory,str(folder)))):
        img_path = os.path.join(directory,str(folder),str(filename))
        img = Image.open(img_path)
        img = img.resize((200,200), Image.ANTIALIAS)
        newi = np.array(img)
        labels.append(str(folder))
        features.append(create_features(newi))

#Formatting Features and Labels
#Converting labels to integer values
integer_labels=[]
for i in range(len(labels)):
  if labels[i] == 'plastic':
    integer_labels.append(1)
  if labels[i] == 'metal':
    integer_labels.append(2)
  if labels[i] == 'glass':
    integer_labels.append(3)
  if labels[i] == 'cardboard':
    integer_labels.append(4)
  if labels[i] == 'trash':
    integer_labels.append(5)
  if labels[i] == 'paper':
    integer_labels.append(6)
```

```
feature_matrix = np.array(features)
feature_matrix = pd.DataFrame(feature_matrix)
integer_labels = pd.DataFrame(integer_labels)

#PCA
from sklearn.decomposition import PCA
pca = PCA(n_components = 25270)
principalComponents = pca.fit_transform(feature_matrix)
```

## *Code for CART*

```
#Building CART model
#We utilize the default model

clf = DecisionTreeClassifier()
clf = clf.fit(X_train,y_train)

#Predict the response for test dataset
y_pred = clf.predict(X_test)
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))

#Optimizing CART parameters
#Doing Cross Validation with default k = 3
parameters = {'max_depth':[4, 8, 16, 20]}
clf = GridSearchCV(DecisionTreeClassifier(), parameters, cv=3, verbose = 2)
clf.fit(X=X_train, y=y_train)
tree_model = clf.best_estimator_
print (clf.best_score_, clf.best_params_)
y_pred = tree_model.predict(X_test)
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))

#Building CART model confusion matrix
df_confusion = sklearn.metrics.confusion_matrix(y_test, y_pred)
df_confusion = pd.DataFrame(df_confusion)
def plot_confusion_matrix(df_confusion, title='Confusion matrix',
cmap=plt.cm.gray_r):
    plt.matshow(df_confusion, cmap=cmap) # imshow
    plt.colorbar()
    tick_marks = np.arange(len(df_confusion.columns))
    plt.xticks(tick_marks, df_confusion.columns, rotation=45)
    plt.yticks(tick_marks, df_confusion.index)
    plt.ylabel('Actual')
    plt.xlabel('Predicted')

plot_confusion_matrix(df_confusion)
```

## *Code for Random Forest*

```
#Building Random Forest
#We utilize the default model
Classifier =
RandomForestClassifier(n_estimators=100,max_features='log2',max_depth=20)
classifier.fit(X_train, y_train)

#Predict the response for test dataset
y_pred_rf = classifier.predict(X_test)
print(classifier)
print("Accuracy:",metrics.accuracy_score(y_test, y_pred_rf))

#Optimizing Random Forest parameters
start = datetime.datetime.now()
random_grid = {'n_estimators': [int(x) for x in np.linspace(start = 200, stop =
2000, num = 3)],
               'max_features': [int(x) for x in np.linspace(start = 200, stop =
500, num = 3)],
               'min_samples_leaf': [5, 10, 15]}
rf_random = RandomizedSearchCV(estimator = classifier, param_distributions =
random_grid, n_iter = 100, cv = 3, verbose=2, random_state=42)
rf_random.fit(X_train, y_train)
best_random = rf_random.best_estimator_
print (rf_random.best_score_, rf_random.best_params_)
y_pred_rf = best_random.predict(X_test)
print("Accuracy:",metrics.accuracy_score(y_test, y_pred_rf))
end= datetime.datetime.now()
elapsed= end-start
print ('Time: ', elapsed)
#Building random forest model confusion matrix
df_confusion = sklearn.metrics.confusion_matrix(y_test, y_pred_rf)
df_confusion = pd.DataFrame(df_confusion)
plot_confusion_matrix(df_confusion)
```

## *Code for Gradient Boosting*

```
#Building initial gradient boosting model
#We utilize the default model
classifier = GradientBoostingClassifier()
classifier.fit(X_train, y_train)

#Predict the response for test dataset
y_pred_boost = classifier.predict(X_test)
print(classifier)
print("Accuracy:",metrics.accuracy_score(y_test, y_pred_boost))

#Building Gradient Boosting confusion matrix
df_confusion = sklearn.metrics.confusion_matrix(y_test, y_pred_boost)
df_confusion = pd.DataFrame(df_confusion)
plot_confusion_matrix(df_confusion)
```

### *Code for XGBoost*

```
#Building initial XGB boosting model
#We utilize the default model
classifier = xgboost.XGBClassifier()
classifier.fit(X_train, y_train)

#Predict the response for test dataset
y_pred_XGboost = classifier.predict(X_test)
print(classifier)
print("Accuracy:",metrics.accuracy_score(y_test, y_pred_XGboost))

#Building XGB Boosting confusion matrix
df_confusion = sklearn.metrics.confusion_matrix(y_test, y_pred_XGboost)
df_confusion = pd.DataFrame(df_confusion)
plot_confusion_matrix(df_confusion)
```

### *Code for Neural Network*

```
import numpy as np
import cv2
import os
from keras.callbacks import ModelCheckpoint,EarlyStopping
from keras.layers import Conv2D, Flatten,
MaxPooling2D,Dense,Dropout,SpatialDropout2D
from keras.models  import Sequential, load_model
from keras.preprocessing.image import ImageDataGenerator, img_to_array,
load_img, array_to_img
import random,os,glob
import matplotlib.pyplot as plt
from google.colab import drive

drive.mount('/content/gdrive', force_remount=True)
```

```
train_dir = "gdrive/My Drive/IEOR SQUAD/242/Final Project/242 Final
Models/data/train/"
test_dir = 'gdrive/My Drive/IEOR SQUAD/242/Final Project/242 Final
Models/data/test/'
valid_dir = 'gdrive/My Drive/IEOR SQUAD/242/Final Project/242 Final
Models/data/valid/'

train_list = glob.glob(os.path.join(train_dir, '*/*.jpg'))
test_list = glob.glob(os.path.join(test_dir, '*/*.jpg'))
valid_list = glob.glob(os.path.join(valid_dir, '*/*.jpg'))

train_datagen = ImageDataGenerator(horizontal_flip=True,
                         vertical_flip=True,
                         rescale=1./255,
                         zoom_range = 0.1,
                         width_shift_range = 0.1,
                         height_shift_range = 0.1)

test_datagen = ImageDataGenerator(rescale=1./255)

valid_datagen = ImageDataGenerator(rescale=1./255)

train_generator = train_datagen.flow_from_directory(
        train_dir,
        target_size=(200, 200),
        batch_size=32,
        class_mode='categorical')

validation_generator = test_datagen.flow_from_directory(
        valid_dir,
        target_size=(200, 200),
        batch_size=32,
        class_mode='categorical',
        shuffle = False)

test_generator = test_datagen.flow_from_directory(
        test_dir,
        target_size=(200, 200),
        batch_size=32,
        class_mode='categorical',
        shuffle = False)

model = Sequential()

model.add(Dense(32,input_shape=(200,200,3),activation='relu'))
model.add(Flatten())
model.add(Dense(64,activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(32,activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(32,activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(6,activation='softmax'))

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

history = model.fit_generator(
        train_generator,
        steps_per_epoch=len(train_list) // 32,
        epochs=5,
        validation_data=validation_generator)

Y_pred = model.predict_generator(test_generator, len(test_list) // 32 +1)
y_pred = np.argmax(Y_pred, axis=1)
target_names = ['cardboard', 'glass', 'metal', 'paper', 'plastic',
'trash']

from sklearn.metrics import classification_report, confusion_matrix
from mlxtend.plotting import plot_confusion_matrix
import pandas as pd
import seaborn as sns
cf = confusion_matrix(test_generator.classes, y_pred)

cm_df = pd.DataFrame(cf,
                     index = target_names,
                     columns = target_names)

correct = 0
```

```
for r in range(len(cf)):
  for c in range(len(cf)):
    if(r==c):
      correct += cf[r,c]

accuracy = correct/sum(sum(cf))
print(accuracy)

correct = 0
for r in range(len(cf)):
  for c in range(len(cf)):
    if(r==c):
      correct += cf[r,c]

accuracy = correct/sum(sum(cf))
print(accuracy)
```

## *Code for Convolutional Neural Network*

```
import numpy as np
import cv2
import os
from keras.callbacks import ModelCheckpoint,EarlyStopping
from keras.layers import Conv2D, Flatten,
MaxPooling2D,Dense,Dropout,SpatialDropout2D
from keras.models  import Sequential, load_model
from keras.preprocessing.image import ImageDataGenerator, img_to_array,
load_img, array_to_img
import random,os,glob
import matplotlib.pyplot as plt
from google.colab import drive
drive.mount('/content/gdrive', force_remount=True)

train_dir = "gdrive/My Drive/IEOR SQUAD/242/Final Project/242 Final
Models/data/train/"
test_dir = 'gdrive/My Drive/IEOR SQUAD/242/Final Project/242 Final
Models/data/test/'
valid_dir = 'gdrive/My Drive/IEOR SQUAD/242/Final Project/242 Final
Models/data/validation/'
train_list = glob.glob(os.path.join(train_dir, '*/*.jpg'))
```

```
test_list = glob.glob(os.path.join(test_dir, '*/*.jpg'))
valid_list = glob.glob(os.path.join(valid_dir, '*/*.jpg'))
'''Image Pre-Proccesing'''

train_datagen = ImageDataGenerator(horizontal_flip=True,
                         vertical_flip=True,
                         rescale=1./255,
                         zoom_range = 0.1,
                         width_shift_range = 0.1,
                         height_shift_range = 0.1)

test_datagen = ImageDataGenerator(rescale=1./255)

valid_datagen = ImageDataGenerator(rescale=1./255)

train_generator = train_datagen.flow_from_directory(
        train_dir,
        target_size=(200, 200),
        batch_size=32,
        class_mode='categorical')

validation_generator = test_datagen.flow_from_directory(
        valid_dir,
        target_size=(200, 200),
        batch_size=32,
        class_mode='categorical',
        shuffle = False)

test_generator = test_datagen.flow_from_directory(
        test_dir,
        target_size=(200, 200),
        batch_size=32,
        class_mode='categorical',
        shuffle = False)

model=Sequential()
#Convolution blocks

#Three Pooling layers
# Three 2D convolution layer
```

```
model.add(Conv2D(32,(3,3),
padding='same',input_shape=(200,200,3),activation='relu'))
model.add(MaxPooling2D(pool_size=2))


model.add(Conv2D(64,(3,3), padding='same',activation='relu'))
model.add(MaxPooling2D(pool_size=2))

model.add(Conv2D(32,(3,3), padding='same',activation='relu'))
model.add(MaxPooling2D(pool_size=2))

#Classification layers
model.add(Flatten())

model.add(Dense(64,activation='relu'))
model.add(Dense(32,activation='relu'))
model.add(Dense(6,activation='softmax'))

#Three fully Connected (dense) Layers

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

history = model.fit_generator(
        train_generator,
        steps_per_epoch=len(train_list) // 32,
        epochs=80,
        validation_data=validation_generator)

from sklearn.metrics import classification_report, confusion_matrix
from mlxtend.plotting import plot_confusion_matrix
import pandas as pd
import seaborn as sns

############
#########
#testing
Y_pred = model.predict_generator(test_generator, len(test_list) // 32 +1)
y_pred = np.argmax(Y_pred, axis=1)
```

```
target_names = ['cardboard', 'glass', 'metal', 'paper', 'plastic',
'trash']
cf = confusion_matrix(test_generator.classes, y_pred)

cm_df = pd.DataFrame(cf,
                     index = target_names,
                     columns = target_names)

print('Classification Report')
print(classification_report(test_generator.classes, y_pred,
target_names=target_names))

plt.figure(figsize=(7,6))
sns.heatmap(cm_df, annot=True)
plt.ylabel('True label')
plt.xlabel('Predicted label')
plt.ylim(len(cf), 0)
plt.show()
```

***Code for CNN with transfer learning***

```
import pandas as pd
from fastai.vision.data import *
from fastai.vision.transform import *
from fastai.vision.learner import *
from fastai.vision.learner import ClassificationInterpretation
from fastai.vision.learner import load_learner
from fastai.vision.data import DatasetType
from fastai.vision.image import open_image
from fastai.vision import models
from fastai.metrics import *
from google.colab import drive
import re
from sklearn.metrics import confusion_matrix
from matplotlib import pyplot as plt
import seaborn as sns
import torch

drive.mount('/content/drive')
```

```
path = "/content/drive/My Drive/data/"
tfms = get_transforms(do_flip=True, flip_vert=True)
data = ImageDataBunch.from_folder(path, test="test", bs=16, size =
(200,200))

data.show_batch(rows=4, figsize=(10,8))

learn = create_cnn(data,models.resnet34, metrics=error_rate)
learn.lr_find(start_lr=1e-6, end_lr=1e1)
learn.recorder.plot(suggestion=True)
min_grad_lr = learn.recorder.min_grad_lr
learn.fit_one_cycle(20,max_lr = min_grad_lr)

interp = ClassificationInterpretation.from_learner(learn)
losses,idxs = interp.top_losses()
interp.plot_top_losses(9, figsize=(15,11))

preds = learn.get_preds(ds_type= DatasetType.Test)
waste_types = data.classes
max_idxs = np.asarray(np.argmax(preds[0], axis=1))
yhat = []
for max_idx in max_idxs:
  yhat.append(data.classes[max_idx])
y = []
for label_path in data.test_ds.items:
  y.append(str(label_path))
pattern = re.compile("([a-z]+)[0-9]+")
for i in range(len(y)):
  y[i] = pattern.search(y[i]).group(1)

cm = confusion_matrix(y, yhat)
df_cm = pd.DataFrame(cm, waste_types, waste_types)

plt.figure(figsize=(10,8))
sns.heatmap(df_cm, annot=True, fmt="d", cmap="YlGnBu")

correct = 0

for r in range(len(cm)):
  for c in range(len(cm)):
    if(r==c):
      correct += cm[r,c]
accuracy = correct/sum(sum(cm))
accuracy
```

### *Code for Confidence Intervals*

```
import numpy as np
from scipy.stats import norm

def confidence_interval(preds, actual):
  pred_actual = (preds == actual)*1
  np.random.seed(100)
  bootstrap = np.random.choice(pred_actual, size=(len(pred_actual),100),
replace=True)
  test_acc = np.mean(pred_actual)
  bootstrap_dist = bootstrap.mean(axis=0) - test_acc
  qu = norm.ppf(0.975)*bootstrap_dist.std() + bootstrap_dist.mean()
  ql = norm.ppf(0.025)*bootstrap_dist.std() + bootstrap_dist.mean()
  lower_bound = test_acc - qu
  upper_bound = test_acc - ql
  return (lower_bound, upper_bound)
```