Introduction

Project Team Members: Abhishek Labade, Atharva Wagh

Research Question: How can the analysis of product and packaging photos to forecast supply chain requirements and stock levels aid in the optimization of inventory management through automated image identification and processing?
Problem Statement: Accurate inventory management and stock level prediction are essential in the fields of supply chain and inventory management. The inefficiencies and error-proneness of traditional manual inventory management systems make them unsuitable for the demands of contemporary e-commerce. A viable answer is provided by the application of automated image recognition and processing algorithms, which allow for real-time stock analysis and product recognition from pictures of the products and their packaging.
Background Information: In the domains of supply chain and inventory management, accuracy in stock level forecast and inventory management is crucial. Traditional manual inventory management systems are not up to the rigors of modern e-commerce due to their inefficiency and error-proneness. The use of automated image recognition and processing algorithms, which enable real-time stock analysis and product recognition from images of the products and their packaging, offers a workable solution.
Importance of the Study: Improving customer happiness, cutting waste, and operating expenses all depend on optimizing inventory management. The large number of products and frequent stock changes are too much for traditional inventory management systems to handle effectively. Automated image identification provides notable advancements in several domains. Businesses may more accurately forecast stock levels, expedite inventory procedures, and identify any problems in the supply chain by evaluating product photos. Thanks to the study's insights, assuring product availability and condition can result in significant cost savings, decreased waste, and improved customer experiences.

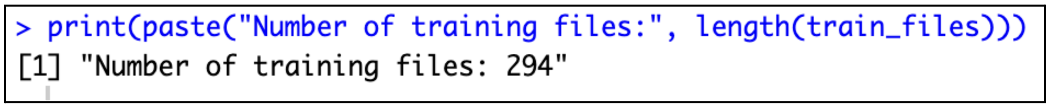
Challenges in Solving the Problem:
Supply chain image analysis presents a number of difficulties.

- Diverse Image Content: It is difficult to standardize image preprocessing due to the large range of products with various sizes, shapes, and packing.
- Complicated Backgrounds: Products are frequently taken in settings with complex backgrounds and uneven illumination, which makes segmentation and analysis more difficult.
- High Dimensionality: Due to the inherent high dimensionality of image data, feature extraction and model training need a lot of computation.
- Class Imbalance: It can be difficult to create accurate and balanced models when some products are overrepresented in datasets.

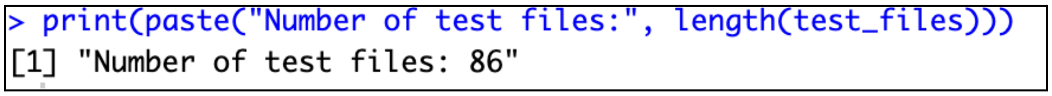
To guarantee precise and scalable solutions, these problems require complex image processing methods and strong machine-learning models.

Dataset

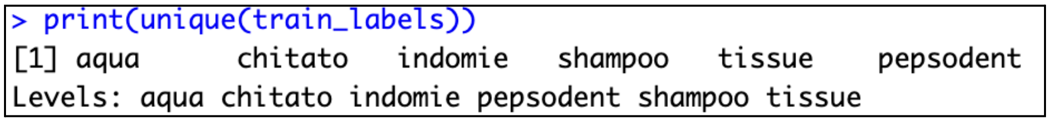
The dataset picked for this particular project is the Retail Product Dataset from Kaggle. The link to the dataset is -
https://www.kaggle.com/datasets/hafizyusufheraldi/retail-product-dataset?resource=download
The collection is composed of training and testing subsets of photos of products and packaging from different categories. The photographs depict various lighting conditions and product packaging features to replicate real-world situations. There are 6 products in total and 294 train images and 86 test images.
In the dataset, the labels are present in annotation files(.xml). They were extracted in R and processed to generate the train and test labels.



```
> print(paste("Number of training files:", length(train_files)))
[1] "Number of training files: 294"
```



```
> print(paste("Number of test files:", length(test_files)))
[1] "Number of test files: 86"
```



```
> print(unique(train_labels))
[1] aqua      chitato   indomie   shampoo   tissue    pepsodent
Levels: aqua chitato indomie pepsodent shampoo tissue
```

Dimensionality:
The representation of any image is a high-dimensional array of pixels:

- Resolution: Although images are downsized to a standard dimension (e.g., 256x256) for analysis, they have various resolutions.
- Color Channels: A single grayscale or three RGB color channels may be present in each image.
- Features: Each image is represented by a vector of features used for classification following preprocessing and feature extraction.

Variables:

- Images (‘X’): The unprocessed pixel data, in either RGB or grayscale, that correspond to each image.
- Labels (‘y’): The ground truth labels that correspond to the conditions or product categories.
- Features (‘F’): Features extracted to represent each image, including colors, forms, and edges.

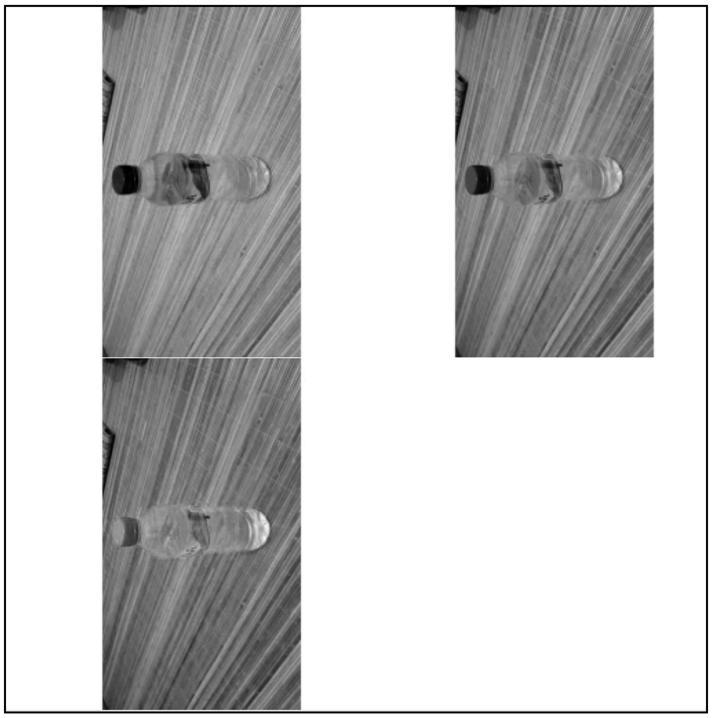
Data Collection: The photos were probably taken by taking pictures of or scanning actual objects in warehouses, retail stores, or other controlled settings. The photographs capture the variances and difficulties that arise in actual situations, such as various backgrounds, lighting settings, and product configurations.
Using the Dataset to Answer the Research Question:
1. Standardization and Preprocessing For consistent analysis, resize and convert photos to grayscale or RGB standard.

2. Feature extraction: To identify products and examine package conditions, extract significant features including edges (using Sobel or Prewitt operators), textures, and colors.
3. Instruction and Evaluation: Using the collected characteristics from the training subset, train machine learning models to identify items and packaging circumstances. Determine how effectively the model generalizes to previously unseen photos by assessing its accuracy on the test subset.
4. Inventory Management Insights: Make use of the trained models to instantly assess product photos in order to forecast stock levels and spot possible problems with the product's condition. Businesses can improve their inventory management systems with more accurate product detection by using automated image recognition and processing algorithms on this dataset. This will improve stock predictions and streamline operations.

Methodology

Implementation:

1. **Pre-processing of data:** We process the image in a step-by-step manner, starting with displaying the original, turning it into grayscale, scaling it to the desired size, normalizing it, and then adding augmentations like a 15-degree rotation and horizontal flip. Every image that has been processed is shown. Ultimately, it compiles these variants into a list, records the quantity handled, and gives back this list.



2. **Extraction of Features:** A Sobel filter is used to find the image's edges. Utilizing a pre-established 3x3 matrix, the Sobel filter highlights horizontal edges by measuring the

gradient of the image's intensity at each pixel. The original image's pixel values' mean and standard deviation is also determined. These statistical measurements give an overview of the distribution of pixel intensities in the image and reveal information about the brightness and contrast in general. A complete feature vector is then created by combining the flattened edge-detected image (converted into a vector) with the mean and standard deviation. This vector includes:

1. The edge strength at each location is represented by a series of pixel values that make up the edge information.
2. The average intensity is indicated by the mean pixel value.
3. The standard deviation, which illustrates the variation in pixel brightness.

Thus, the total number of features extracted from one image would be:
65,536 features from the Sobel filter results (one feature per pixel),
1 feature from the mean of the image,
1 feature from the standard deviation of the image.
So, the total number of features extracted per image is
65,536+1+1=65,538

This image results from applying the sobel filter.

3. Dimensionality Reduction: The dataset is then subjected to Principal Component Analysis (PCA) to reduce dimensionality while preserving important variance. It uses the prcomp function to compute PCA, standardizes the data, and extracts the top n_components principal components. This helps handle high-dimensional data more effectively.

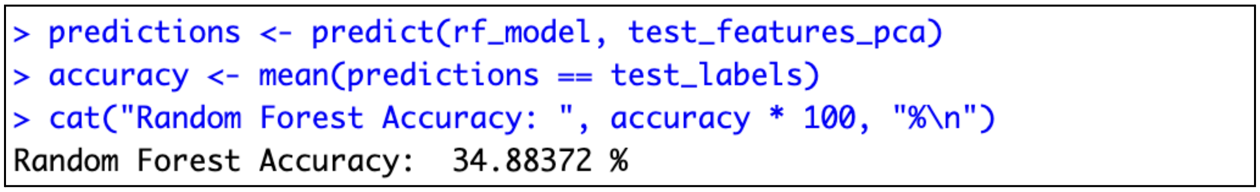
4. **Running the Random Forest Classifier**

The randomForest library is first loaded in order to take advantage of its powerful features for group learning. First, a baseline model with 200 trees is trained on the model known as rf_model using principal component analysis (PCA)-reduced features (train_features_pca) and corresponding labels (train_labels).

The hyperparameters were adjusted multiple times to improve the model:

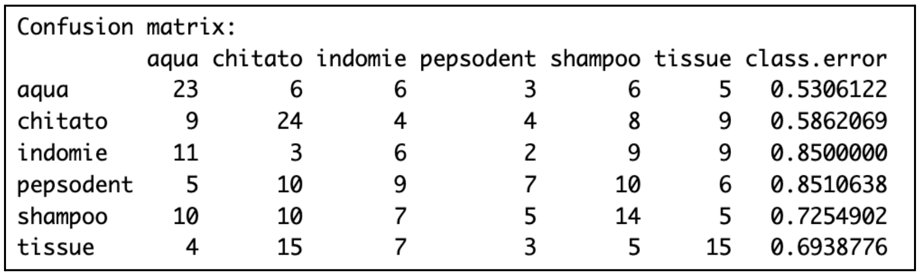
- To boost prediction precision and model resilience, the number of decision trees (ntree) is raised to 300 from 100.
- In line with standard procedures for classification tasks, the square root of the number of features is assigned to the mtry parameter, which indicates the number of variables taken into consideration for splitting at each tree node.
- The model is set up to improve data integrity and interpretability by calculating variable importance and handling missing values by omission.

The accuracy obtained from the model is as follows:



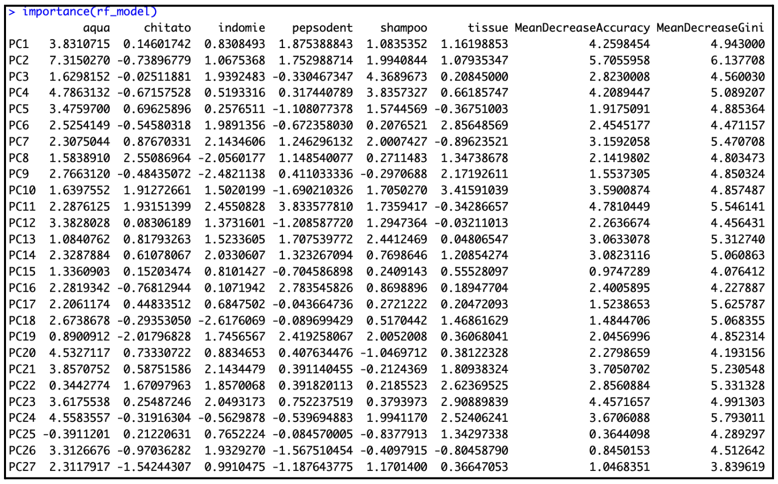
```
> predictions <- predict(rf_model, test_features_pca)
> accuracy <- mean(predictions == test_labels)
> cat("Random Forest Accuracy: ", accuracy * 100, "%\n")
Random Forest Accuracy:  34.88372 %
```

The confusion matrix obtained from the predictions is as follows:



```
Confusion matrix:
          aqua chitato indomie pepsodent shampoo tissue class.error
aqua        23       6       6         3       6      5   0.5306122
chitato      9      24       4         4       8      9   0.5862069
indomie     11       3       6         2       9      9   0.8500000
pepsodent    5      10       9         7      10      6   0.8510638
shampoo     10      10       7         5      14      5   0.7254902
tissue       4      15       7         3       5     15   0.6938776
```

We then check variable importance which is as follows:



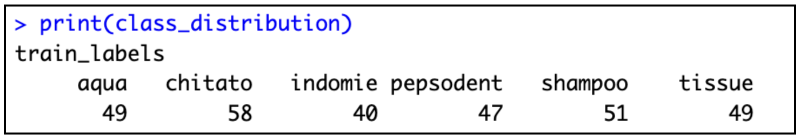
```
> importance(rf_model)
          aqua     chitato    indomie    pepsodent    shampoo      tissue MeanDecreaseAccuracy MeanDecreaseGini
PC1   3.8310715  0.14601742  0.8308493  1.875388843  1.0835352  1.16198853            4.2598454         4.943000
PC2   7.3150270 -0.73896779  1.0675368  1.752988714  1.9940844  1.07935347            5.7055958         6.137708
PC3   1.6298152 -0.02511881  1.9392483 -0.330467347  4.3689673  0.20845000            2.8230008         4.560030
PC4   4.7863132 -0.67157528  0.5193316  0.317440789  3.8357327  0.66185747            4.2089447         5.089207
PC5   3.4759700  0.69625896  0.2576511 -1.108077378  1.5744569 -0.36751003            1.9175091         4.885364
PC6   2.5254149 -0.54580318  1.9891356 -0.672358030  0.2076521  2.85648569            2.4545177         4.471157
PC7   2.3075044  0.87670331  2.1434606  1.246296132  2.0007427 -0.89623521            3.1592058         5.470708
PC8   1.5838910  2.55086964 -2.0560177  1.148540077  0.2711483  1.34738678            2.1419802         4.803473
PC9   2.7663120 -0.48435072 -2.4821138  0.411033336 -0.2970688  2.17192611            1.5537305         4.850324
PC10  1.6397552  1.91272661  1.5020199 -1.690210326  1.7050270  3.41591039            3.5900874         4.857487
PC11  2.2876125  1.93151399  2.4550828  3.833577810  1.7359417 -0.34286657            4.7810449         5.546141
PC12  3.3828028  0.08306189  1.3731601 -1.208587720  1.2947364 -0.03211013            2.2636674         4.456431
PC13  1.0840762  0.81793263  1.5233605  1.707539772  2.4412469  0.04806547            3.0633078         5.312740
PC14  2.3287884  0.61078067  2.0330607  1.323267094  0.7698646  1.20854274            3.0823116         5.060863
PC15  1.3360903  0.15203474  0.8101427 -0.704586898  0.2409143  0.55528097            0.9747289         4.076412
PC16  2.2819342 -0.76812944  0.1071942  2.783545826  0.8698896  0.18947704            2.4005895         4.227887
PC17  2.2061174  0.44833512  0.6847502 -0.043664736  0.2721222  0.20472093            1.5238653         5.625787
PC18  2.6738678 -0.29353050 -2.6176069 -0.089699429  0.5170442  1.46861629            1.4844706         5.068355
PC19  0.8900912 -2.01796828  1.7456567  2.419258067  2.0052008  0.36068041            2.0456996         4.852314
PC20  4.5327117  0.73330722  0.8834653  0.407634476 -1.0469712  0.38122328            2.2798659         4.193156
PC21  3.8570752  0.58751586  2.1434479  0.391140455 -0.2124369  1.80938324            3.7050702         5.230548
PC22  0.3442774  1.67097963  1.8570068  0.391820113  0.2185523  2.62369525            2.8560884         5.331328
PC23  3.6175538  0.25487246  2.0493173  0.752237519  0.3793973  2.90889839            4.4571657         4.991303
PC24  4.5583557 -0.31916304 -0.5629878 -0.539694883  1.9941170  2.52406241            3.6706088         5.793011
PC25 -0.3911201  0.21220631  0.7652224 -0.084570005 -0.8377913  1.34297338            0.3644098         4.289297
PC26  3.3126676 -0.97036282  1.9329270 -1.567510454 -0.4097915 -0.80458790            0.8450153         4.512642
PC27  2.3117917 -1.54244307  0.9910475 -1.187643775  1.1701400  0.36647053            1.0468351         3.839619
```

Challenges:

1. **Various Image Content:**

- Applying a consistent preprocessing technique can be challenging due to the dataset's potential diversity of product photos.To standardize the photos, methods like augmentation, color normalization, and scaling were implemented during preprocessing.

2. **Complex Backgrounds:**
- Accurate segmentation and classification may be hampered by the photos' complex backgrounds. To separate goods from the backdrop, we have utilized sophisticated segmentation techniques like edge detection using a Sobel Filter.
3. **Class Imbalance:**
- Problem: It's possible that the dataset includes a variety of product categories, with a disproportionately high number of samples in some classes compared to others.



```
> print(class_distribution)
train_labels
     aqua   chitato   indomie pepsodent   shampoo    tissue 
       49        58        40        47        51        49 
```

The class distributions were examined and there was no imbalance evident. However, if there was class imbalance, strategies such as weighted classification models, undersampling, and oversampling could be employed.
4. **Feature Extraction:**
- Difficulty: It takes domain knowledge and trial and error with various image processing methods to extract the most pertinent features for product identification and classification. To determine the best strategy, test various feature extraction techniques, including texture analysis, edge detection, and color histograms. In this project, features were extracted using the Sobel Filter, mean and Standard Deviation.
5. **Low Accuracy:**
- Difficulty: The Random Forest's initial configurations, including the number of trees (ntree) and features taken into account at each split (mtry), might not have been the best ones. The model's capacity to generalize is significantly influenced by these parameters. We can perform Hyperparameter Tuning to find optimal parameters.

Solutions to the Challenges

How these challenges were solved:

1. **Variety in Image Content:** Preparation and Normalization: Consistency across the dataset was achieved by downsizing photographs to a consistent resolution and converting them to grayscale to minimize color variances.
2. **Complicated Historical Backgrounds:** Image Segmentation: To distinguish the product from the background, we have applied a Sobel Filter for edge detection. This aids in separating the product for analysis that is more precise.
3. **Feature Detection:** Edge Detection was performed to extract shape information using Sobel operators. These characteristics aid in product differentiation.
4. **Low Accuracy** Multiple strategies were tried to increase the accuracy of the model.
- The number of trees in the classifier was increased iteratively from 100, 150, 200 to 300.
- By lowering the variance in the predictions, more trees can improve the stability and accuracy of the model by allowing the errors of each individual tree to be averaged out.
- The mtry parameter was also set to the square root of the number of features in order to balance the bias and variance of the model.

- By changing na.action to na.omit, improper handling of missing data from skewing or biasing the model training was prevented, which can have an impact on model accuracy.
- Finding out which features are most important for the prediction was aided by enabling the variable importance calculation (importance = TRUE). This knowledge was applied to improve the feature selection procedure even more, possibly improving model performance by emphasizing the most pertinent features.

Training and Testing Strategies

There are 294 photos in the training set and 86 images in the test set of the dataset. With this distribution, we can assess the model's performance on a test set of a manageable size and train it with sufficient data.The aforementioned testing and training approaches were applied in the code provided:
Training Strategy
1. Data Preparation: Loading and Preprocessing: All images from the training dataset are loaded, preprocessed (converted to grayscale and resized), and converted into feature vectors. Each image is associated with a label extracted from its filename.
2. Feature Extraction: The images are converted to a vectorized form suitable for training (e.g., edge detection applied on binary images).
3. Model Training: A Random Forest classifier is used for training on the extracted features. The model learns to associate the features with their corresponding labels..
Testing Strategy
1.Data Preparation: All images from the testing dataset are loaded and preprocessed in the same way as the training images.
2. Feature Extraction: Similar to the training process, feature extraction is applied to the preprocessed test images to obtain feature vectors.
3. Model Testing: The trained Random Forest model is used to predict labels for the test dataset based on the extracted features.
4. Evaluation: The predictions are compared against the true labels of the test data, and the accuracy is calculated to assess model performance.
The purpose of these tactics is to measure the model's ability to generalize to unseen images by making sure it is evaluated on data that it hasn't seen during training.

Mathematical notations associated with dataset variables

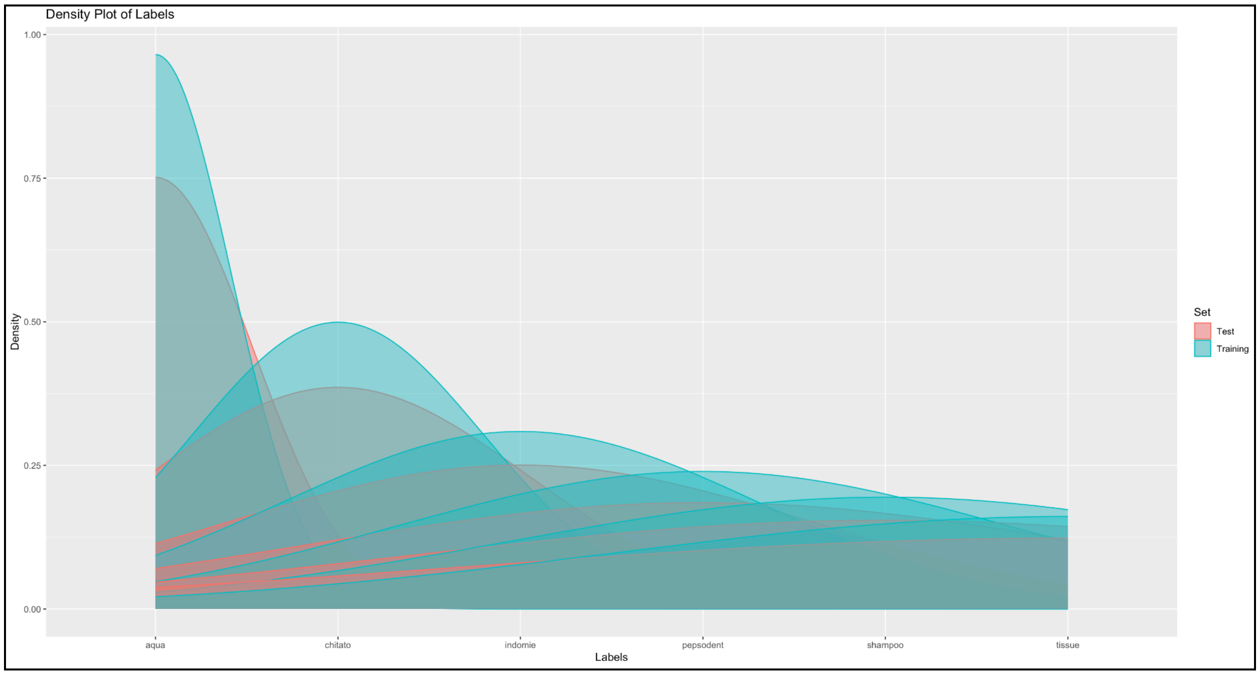
Here's how the mathematical notations used in the PDF relate to the dataset variables in the code:
1. Image Representation (`f(x1,x2)`):
 `f(x1,x2)` represents the pixel intensity of a grayscale image at coordinates `(x1, x2)`.
 - Dataset Variable in Code: `img`
 - Meaning: The grayscale intensity values of the image loaded from the dataset.
2. Binary Image (`B`):
 `g(x,y) = T(f(x,y))`, where `T` is the threshold function.
 - Dataset Variable in Code: `img_binary`

- Meaning: A binary image created by applying a threshold to the grayscale image, representing whether each pixel is above or below the threshold.

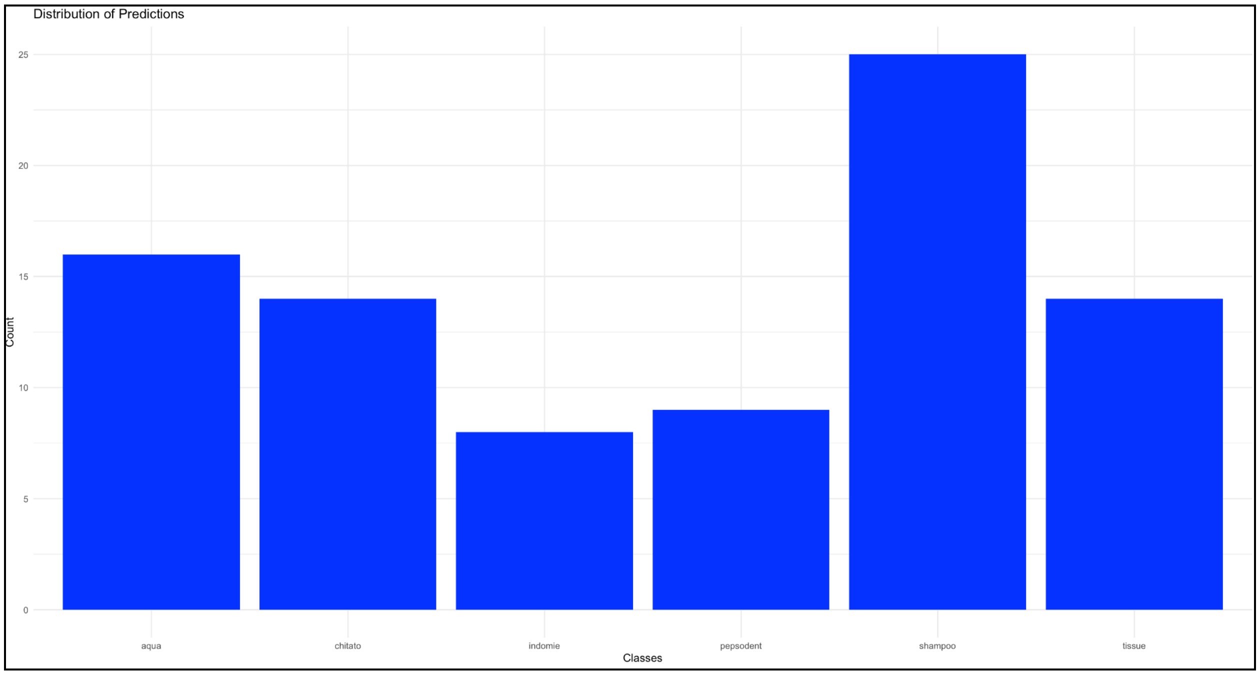
3. Edge Detection (`fx`, `fy`, `E`):
   - `fx`, `fy` represent gradients in the x and y directions, respectively, with edge detection combining them: `E = sqrt(fx^2 + fy^2)`.
   - Dataset Variable in Code:** `img_edges`
   - Meaning: Edge detection result, obtained by convolving the grayscale image with masks (like Sobel) to find edges.

Results and Conclusion

Density Plot for distribution of training and test labels can be seen in the image below:



The predictions and their counts obtained over the test dataset are as follows:

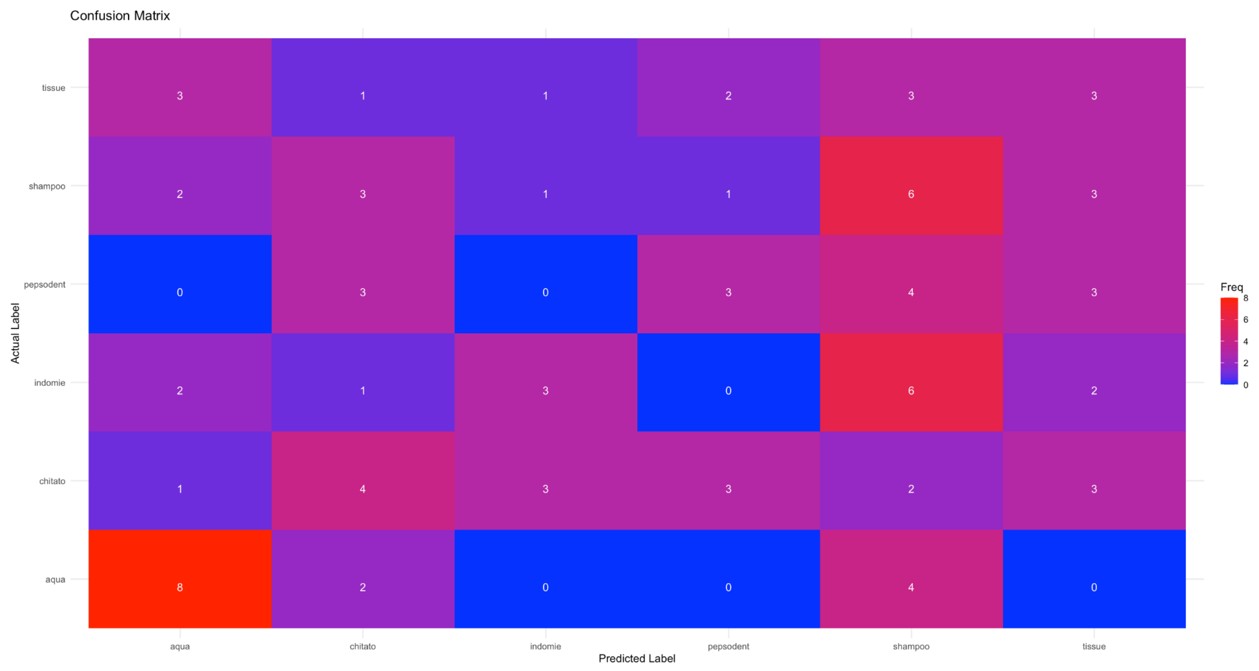


- Aqua: The class with the highest prediction count is the one that the model predicts the most often.
- Shampoo: The second most common, suggesting that this category also has important forecasts.
- Chitato and Tissue: The model appears to identify these classes quite frequently, based on their moderately high prediction counts.
- Among the six classes, Indomie and Pepsodent have the lowest prediction frequencies, indicating that the model predicts them less frequently.

Insights from Predictions distribution:

- Model Bias: The difference in prediction counts may indicate that the model has a bias in favor of some classes, such as "shampoo" and "aqua". This could be because these classes are more common in the training set or because their characteristics stand out more.
- Model Performance: Analyzing the distribution of predictions can also point to possible areas where the model may be performing poorly, especially when it comes to predicting less common classes like "indomie" and "pepsodent." These classes may benefit from additional research as well as possibly resampling methods or modified class weights during training.

The confusion matrix obtained from the model’s predictions is as follows:



The presented confusion matrix shows how well the classification model performs over a number of categories. The predicted labels by the model are represented by each column in the matrix, while the actual class labels are represented by each row. The number of predictions for each class is represented by the numbers in the matrix.
Key observations from the matrix include:

- True Positives (Diagonal Cells): These cells show examples of when the model correctly identified the class, like the 'Aqua' cell in figure 8. For a given class, high numbers here indicate good model performance.
- False Positives and False Negatives (Off-Diagonal Cells): Error-prone areas on the diagonal are indicated by cells. For instance, the model frequently misclassified "Tissue" as "Aqua" three times, pointing to a systematic error where the two classes are confused. Likewise, the misclassification of 'Indomie' as 'Chitato' implies feature overlap between these classes, which results in inaccurate predictions.
- Model Weaknesses: Classifications such as 'Chitato' and 'Pepsodent' are often confused with one another; this is indicated by the counts of three in each corresponding off-diagonal cell, indicating that the model has difficulty differentiating between these features.
- Strategic Insights for Improvement: In order to lower these particular kinds of errors, analyzing the pattern of misclassifications can aid in improving the feature engineering procedure, changing class weights, or fine-tuning model parameters.

In summary, the study has effectively demonstrated that machine learning has great potential to change inventory management, especially when combined with sophisticated image processing methods and Random Forest classifiers. Businesses can greatly improve the accuracy, efficiency, and overall responsiveness of their inventory to market dynamics by further refining these models and integrating them with current business systems.