Universitat de les Illes Balears

escola politècnica superior

Treball Fi de Grau

Computer Engineering

Classification of shoe images using techniques of computer vision and machine learning

IVAN KOMATINA

Tutors
Dr. Antoni Jaume-i-Capó
Dr. Gabriel Moyà

Escola Politècnica Superior
Universitat de les Illes Balears
Palma, 31 de gener de 2019

Contents

1 Introduction 1
- 1.1 Problem 1
- 1.2 Previous solutions 1
- 1.3 Solution 2
- 1.4 Objectives 3

2 Artificial Intelligence and Machine Learning: How Computers Learn 5
- 2.1 Artificial Intelligence 5
- 2.2 Machine Learning 6
- 2.3 Supervised learning 9
  - 2.3.1 Labeled data 9
  - 2.3.2 Tree 10
  - 2.3.3 Decision Tree 11
  - 2.3.4 Random Forest 15
  - 2.3.5 Support-vector machine 17
- 2.4 Computer Vision 18
- 2.5 Technologies 19
  - 2.5.1 Python 19
  - 2.5.2 scikit-learn 19
  - 2.5.3 OpenCV 19
  - 2.5.4 Computer Vision Algorithms 19
- 2.6 The conclusion of the chapter 20

3 Datasets 21
- 3.1 Scrapping the website 21
- 3.2 Image files dataset 22
- 3.3 Feature dataset 23
  - 3.3.1 Removing background 24
  - 3.3.2 Shoe segmentation algorithm 26
- 3.4 Improving dataset 26
  - 3.4.1 Color features 26

3.4.2 Texture features....27
3.5 The conclusion of the chapter....28
4 Classification of dataset....29
4.1 Tuning hyperparameters....29
4.2 Feature importance....30
4.3 Feature similarity....30
4.4 Selecting features....31
4.5 Interpreting Random Forest....33
4.5.1 Visualizing Decision Trees....33
4.5.2 Partial dependence....34
4.5.3 Tree Interpreter....36
4.6 The conclusion of the chapter....37
5 Results....39
5.1 Classification metrics....41
5.2 The Confusion Matrix....42
5.3 Changing the number of categories....44
5.4 Binary classification....44
5.5 Saving shoe predictor....45
6 Discussion and conclusion....47
7 References....49

Table of Figures

Figure 1: How Apple process user's voice commands [9]. 6

Figure 2: Example of overfitting [37]. ....

Figure 3: Bias-variance tradeoff [12]. ....

Figure 4: reCAPTCHA prompt to verify that the user is human..

Figure 5: Labeled binary tree [15]..

Figure 6: Iterations of a Decision Tree [17]. 12

Figure 7: Visualization of ten different predictors and mean predictor (red color) [20]. ....

Figure 8: CPU activity 16

Figure 9: Histogram before and after equalization ....

Figure 10: Original image. Image with equalized contrast. Image on which CLAHE is applied [27]. 20

Figure 11: Disproportional number of shoes in each category 21

Figure 12: Folder structure of the dataset...

Figure 13: Successful background removal cv2.createCLAHE(clipLimit=8.0, tileGridSize=(32,32)) 24

Figure 14: Successful background removal cv2.createCLAHE(clipLimit=8.0, tileGridSize=(32,32)) 25

Figure 15: Unsuccessful background removal. cv2.createCLAHE(clipLimit=16.0, tileGridSize=(4,4)) 25

Figure 16: Acceptable background removal 25

Figure 17: HSV cylinder ....

Figure 18: Input picture. The shoe with the removed background. Visual representation of dominant colors. Names of 4 features for each dominant color. 27

Figure 19: Dendrogram of the feature similarity..31

Figure 20: Accuracy of the model which uses 54 features, on this particular train and test dataset, is 41.613%. 32

Figure 21: The accuracy of the model which uses 24 features, on this particular train and test dataset, is 43.226% 32

Figure 22: Accuracy compared to the increasing number of best features 33

Figure 23: 5th out of 10 decision trees in Random Forest ensemble.....................................33

Figure 24: 1st out of 10 decision trees in Random Forest ensemble 33

Figure 25: 6th out of 10 decision trees in Random Forest ensemble.....................................34

Figure 26: Partial dependence plot for ch_area and bigcategory 35
Figure 27: Partial dependence plot for “ch_center_y” and “bigcategory” features..............35
Figure 28: Correct classification of the object 36
Figure 29: Incorrect classification of the object 36
Figure 30: Authors implementation of tree interpreter..37
Figure 31: Comparison of the average accuracy out of 100 tests 39
Figure 32: Comparison of the maximum accuracy out of 100 tests.......................................40
Figure 33: Comparison of the minimum accuracy out of 100 tests 40
Figure 34: True Positive, True Negative, False Positive, False Negative, Recall, Specificity, Precision, Accuracy, and F1 score..41
Figure 35: Confusion matrix (without normalization) 42
Figure 36: Normalized confusion matrix..43
Figure 37: Accuracy score compared to the number of categories in training dataset..........44
Figure 38: Binary classification 45

Table 1: Texture features 28

Acronyms

GPU	Graphics processing unit
CPU	Central processing unit
AI	Artificial Intelligence
ML	Machine Learning
RF	Random Forest
SVM	Support-vector Machine
CV	Computer Vision
XML	Extensible Markup Languag
CSV	Comma-separated values

-

Abstract

Classification of products is a challenging task which requires both Computer Vision and Machine Learning techniques to create a working system with an acceptable level of accuracy. Local shoemaking company, Camper, provided the dataset for this project and the task was to classify shoes based on single top-view images. There are 16 possible categories of shoes with a disproportional number of shoes in each category.

In this paper, the reader will learn about basic Machine Learning terms: what they mean and why they are used. Artificial Intelligence as a concept will be explained as well as supervised learning, with emphasizing on Decision Tree and Random Forest. Used technologies will briefly be explained: *Python*, *Open-CV* and *scikit-learn* library.

After downloading images from the website and creating a folder structure suitable for ML, features were extracted using the *UIB - V Features* library. Before extraction of features, it was necessary to remove the background because the physical shape of the shoe was the input for the mentioned library. Algorithm for that purpose has been created. Color and texture features were also added.

Since dataset was relatively small, Random Forest was chosen instead of Deep Learning methods. Random Forest is a supervised learning classifier, and after extracting the features, it correctly predicts categories for about 45% shoes in the training datasets. Other results have also been explored: binary classification and classification with a reduced number of categories. Former correctly classifies approximately 85% of the training dataset while the latter does 65% of the training dataset (3 and 4 categories).

Random Forest model is also used to explore the data using Tree Interpreter and Partial Dependence Plots. After analyzing feature importance, half of the feature could be removed while retaining almost the same accuracy and as a result of that time and space complexity of Random Forest Classifier model has been lowered.

1 Introduction

1.1 Problem

The recognition of products is a challenging area offering many benefits. From the user's standpoint, it would be convenient if they could automatically get recommendations for similar items they already have- users wearing running shoes would benefit if they could get recommendations to try similarly looking items from different makers. For sellers, it offers excellent financial promises. By automatically detecting products via a smartphone app, website or terminal in a physical store they have a basis for cross-selling (selling additional products or services) and upselling (persuading customers to buy a more expensive product).

However, product recognition is a difficult task because images of the same product can be taken from different angles and different lighting conditions, with varying levels of noise and other factors can influence the final photo. Also, different features may prove difficult to distinguish visually- like colors or material's texture. Fortunately, adequately tuned convolutional neural networks can effectively resolve these problems, but they require an enormous amount of data for training in order to be more accurate.

Camper[1] is a modern footwear brand from the island of Mallorca, Spain. They are the company which provided the dataset, and their problem is how to classify which shoe belongs to which category correctly. Camper organizes its footwear collection into different categories or classes. Since there are no strict differences between many shoe categories, to which one will newly designed shoe model belong? Does category choice depend on the shape of shoe, color, texture or any other feature?

1.2 Previous solutions

It is worth mentioning that even though all following solutions use neural networks and have good results, they were not applicable to the obtained dataset. In the last mentioned example [1], Conde Nast used deep learning classifier, and it took them around one day to train the model on around 25 times more powerful GPU than I have. Also, the main advantage of the last solution is that they had thousands of images per class while dataset which was used for this project had at most 140 images per class.

Startup GOAT uses AI to verify if shoes are genuine products using deep learning. Using NVIDIA TITAN Xp GPUs and NVIDIA Tesla GPUs on the Amazon Web Services Cloud with the cuDNN-accelerated PyTorch deep learning framework the company trained their neural networks on 75,000 images of authentic sneakers. The company also collects data such as color, suppleness of the sole, hardness of certain rubbers, texture, and the quality of the seams [2].

[1] https://www.camper.com

ELSE.ai is a machine learning and artificial intelligence oriented framework for product design personalization, providing an AI-driven personal style and individual size learning and recommendation services [3]. What is unique about this company is that they used synthetic and augmented 3D & CAD data. That allows generating almost infinite variations of shoe images by changing lighting conditions, textures, colors, background, and other variables.

Shoegazer is proof of concept which uses image recognition and transfer learning technologies to identify brands and models of trainers in real-time with 95% accuracy. Certain features of shoes can be identified like logo, shape, and style [4].

Deepsense.ai solved kaggle.com competition "*iMaterialist Challenge at FGVC 2017 - Can you assign accurate description labels to images of apparel products?*". Dataset had 50000 pictures of 576 classes. They achieved 60% accuracy using 20 convolutional neural networks. They used the following architectures in several variants: DenseNet, ResNet, Inception, and VGG [5]. All of them were initialized with weights pre-trained on the ImageNet dataset. Their models also differed regarding the data preprocessing (cropping, normalizing, resizing, switching of color channels) and augmentation applied (random flips, rotations, color perturbations from Krizhevsky's AlexNet paper). All the neural networks were implemented using the PyTorch framework [5].

Moodstocks was a French startup which was bought by Google in 2016. It used to build image and object recognition software using deep learning techniques and offered an Android app and visual search API that could recognize certain kinds of object. By analyzing video from a smartphone camera, and correlating it with accelerometer readings to determine how the camera is moving around, the software can infer information about the three-dimensional shape of objects in the video, facilitating their recognition [6].

Conde Nast made a handbag classifier [1]. Even though handbags are different compared to shoes, for computers, they are pretty much the same: pixels on images. They used seven brands (classes), each having around 1400 images while 200 images were used for testing and achieved 80-90% accuracy.

1.3 Solution

The idea was to use some machine learning techniques to solve this problem. Deep learning is currently a very promising technique for solving classification problems, but it requires an enormous amount of data- primarily labeled images of shoes from different angles and lighting conditions. Since dataset could be made having only one image from five different angles of each shoe, that was not enough for deep learning.

Another robust machine learning algorithm for classification is Random Forest. It requires fewer images in a dataset, but more numerical features have to be extracted from those pictures compared to deep learning models. When the model is trained on the training set (80% of the total data), the average accuracy of correct predictions of the test set is 45%.

1.4 Objectives

The primary objective of the project was to make a shoe classifier. In this project, Computer Vision and Artificial Intelligence techniques are used because the goal was to create a system that could automatically extract features from all images and learn what features are important in order to learn how to classify of the shoes correctly. The aim of this project was to achieve the following objectives:

1. Learn how to scrape information from websites and how to manipulate XML data.
2. Create an image database from scratch and then extract features using preexisting and custom algorithms.
3. Learn the basics of *OpenCV* and computer vision algorithms in order to edit images.
4. Learn about supervised machine learning, decision trees and how to improve decision trees.
5. Apply Random Forest learning method to predict categories of the shoes.

2 Artificial Intelligence and Machine Learning: How Computers Learn

In this chapter, the reader will learn: what artificial intelligence is and its field of application, different terms used in machine learning, and what a train, validation and test dataset are, what over and underfitting are, and how do companies get labeled data. After this introduction, the Tree and Decision Tree will be defined. The reader will know how to improve decision trees with techniques likes bootstrapping, bagging and boosting. In the end, examples of how Random Forest is used in business.

Computer Vision algorithms used in this project and Python implementations of Machine Learning and Computer Vision algorithms will also be described.

2.1 Artificial Intelligence

An intelligent agent is an independent, self-governing entity or device which perceives its environment and takes actions that maximize its chance of successfully achieving its goals. It has inputs and outputs. Input signals are frequently sensors and output is action derived from agent's purpose to achieve a goal. Intelligent agents can learn and use knowledge to achieve predefined goals. Their complexity can be high or low. Example of the simplex reflex machine is a thermostat.

Artificial intelligence is a field which studies these intelligent agents. Any system can be classified as AI if it can correctly interpret external data, learn from such data, and to use those learnings to achieve specific goals and tasks. In simple terms, AI is a system which mimics the human mind; particularly processes of learning and problem-solving. The end goal of AI is to reach artificial general intelligence (AGI) which can successfully perform any intellectual task that human being can do [7].

Learning process of AI agents was coined in 1956 in Dartmouth workshop: *"Every aspect of learning or any other feature of intelligence can be so precisely described that a machine can be made to simulate it. An attempt will be made to find how to make machines use language, form abstractions and concepts, solve kinds of problems now reserved for humans, and improve themselves”* [8].

Examples or real-world use-cases of AI are:

- Self-driving cars: nearly every major carmaker, but most notably Waymo, part of Google company.
- Strategic games: AlphaGo and AlphaZero both won against best human players and chess engines in the world.
- Speech recognition: Siri, Alexa, and Google Assistant. All voice assistants use ML to process user’s commands, and in Figure 1 the reader can see how Siri works on Apple devices.

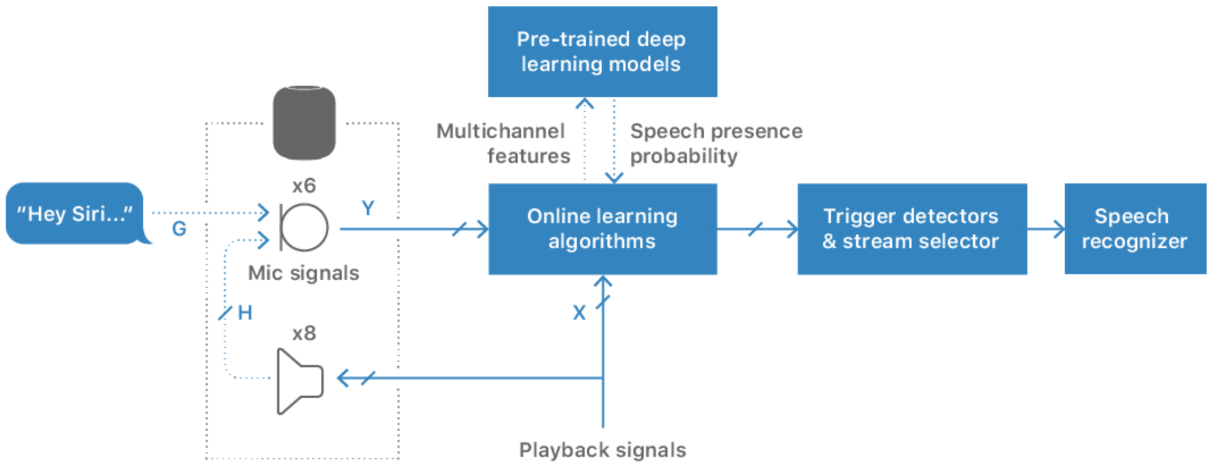

Pre-trained deep learning models
Multichannel features
Speech presence probability
"Hey Siri..."
G
x6
Y
Mic signals
H
x8
Online learning algorithms
X
Playback signals
Trigger detectors & stream selector
Speech recognizer

Figure 1: How Apple process user's voice commands [9].

2.2 Machine Learning

Machine Learning (ML) is the subfield of computer science and a branch of artificial intelligence, whose objective is to develop techniques that allow computers to learn [10]. More specifically, it is about creating programs capable of generalizing behaviors from information provided in the form of examples. In many cases, machine learning concepts overlap with that of computational statistics since the two disciplines are based on data analysis. However, machine learning also focuses on the study of the computational complexity of problems. Many problems are extremely computationally intensive, so much of the research done in machine learning is focused on designing feasible solutions and optimizations to those problems. Machine learning can be seen as an attempt to automate some parts of the scientific method by mathematical methods. Machine learning has a wide range of applications, including search engines, medical diagnostics, fraud detection in the use of credit cards, stock market analysis, classification of DNA sequences, recognition of speech and written language, games and robotics [10].

Terms used in machine learning are:

- Independent variable: a variable that can be changed; the property of an object (examples: temperature, air humidity, a day of the year, etc.)
- Dependent variable: a variable which depends on the values of independent values. We are trying to predict the value of this variable (examples: chance of rain)
- Regression: output (dependent) variable has continuous value, one of infinitely many, uncountable values (example: 43,329%)
- Classification: classify to which class does dependent variable belong to. The depended variable can only be one of the known discrete values (example: ”no rain”/”0”, “rain”/”1”)

2.2.1.1 Training, validation and test datasets

To train any machine learning algorithm data has to be provided. There are several types of dataset. One example of folder structure of the dataset which the author made in chapter *"Image files dataset"* is suitable for machine learning algorithms when inputs are image files, is to have a "*data"* folder in which there are several folders which are different classes (what we are trying to predict) like "*shoe_category_1", "shoe_category_2",* etc. In each class folder, there are subfolders of different shoe models, and each different model has several pictures.

Another example of the dataset is the one used for Random Forest. It is described in "*Feature dataset*" chapter. This dataset is a CSV file and contains a large number of rows and columns. Each row corresponds to one item (object, like a shoe). Columns are features each object has like: identification number, creation date, length, width, or price. Usually the more columns and rows there is, the better model will be (thousands of rows with dozens of columns yield good results).

Datasets are split into train and test dataset, or train, validation, and test datasets. Their sizes are usually: 80% of the original dataset is the training dataset, and the remaining 20% is the test set [11]. These datasets are have randomly chosen parts of the original dataset. However, if we are doing time series (list of data points captured in equally spaced time intervals) analysis, like stock price prediction, it would not make sense to randomly choose data point because we would lose connection and dependencies between previous and following data points. In this case, train set should contain, for example, data from January to July, and test set data for August, and we would use this final, trained, model to predict prices for September.

If we have a sufficiently large dataset and we are trying several different algorithms or models data should be split into three parts:

- Training set (60% of the original dataset): multiple algorithms use this same dataset to train. We tweak (hyper) parameters (weights of connections between neurons in artificial neural networks or maximum depth of trees in Random Forests).
- Validation set (20%): performance of all competing algorithms is measured on this dataset. Choose the algorithm which has the best performance. However, this dataset may have uncommon values, noise or unwanted data relationship. We don't know that, so that is why the next phase is essential.
- Test set (20%): since we are trying to predict upcoming, previously unseen data, we use our final, tweaked model to get an idea how the model is going to work in real life. The model must not be tuned anymore. If performance is low everything should be repeated from the beginning.

2.2.1.2 Over and underfitting

While training a machine learning algorithm, problems such as over and under fit can happen so it is necessary to recognize them and take appropriate measure.

Overfitting is the result of overtraining a learning algorithm with some data when the desired result is known so that it cannot generalize beyond the training set. The learning algorithm (model) needs to generalize well both on the train and test datasets. However, when a system trains too much (is overtrained) or trains on irrelevant data, the learning algorithm can be adjusted to particular characteristics of training data that have no causal relationship with the objective function. During the overfit phase, the accuracy of training samples continues to increase while its performance with test samples gets worse.

The most important consequence of overfitting is poor accuracy when testing on test or validation datasets. Other consequences are the increased complexity while testing the validation set because the function is more likely to need additional data. Overfitting can be minimized by preprocessing data, parameters, and hyperparameters. Also, techniques like minimum spanning tree or lifetime of correlation can be used.

The opposite problem of overfitting is underfitting. Underfitting happens when the underlying structure of the data cannot be generalized by a machine learning algorithm or statistical model. The most likely cause of this is that training data does not have enough data. Underfit has low variance and high bias while overfit has high variance and low bias.

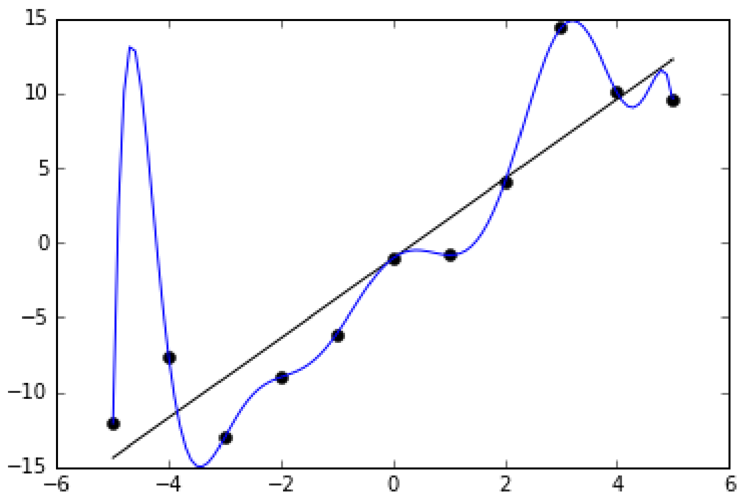


Figure 2: Example of overfitting [37].

In Figure 2 we can see the example of overfitting: blue line represents the data better than the straight line, but straight line generalizes better.

These problems are also known as bias-variance tradeoff [12]:

- The bias is an error from erroneous assumptions in the learning algorithm. High bias can cause an algorithm to miss the proper relations between features and target outputs (underfitting). In Figure 3, bias is shown as the distance of the clustered dots from the red center.

- The variance is an error from sensitivity to small fluctuations in the training set. High variance can cause an algorithm to model the random noise in the training data, rather than the intended outputs (overfitting) [12]. In Figure 3, the variance is shown as the distance of the scattered dots from the red center.

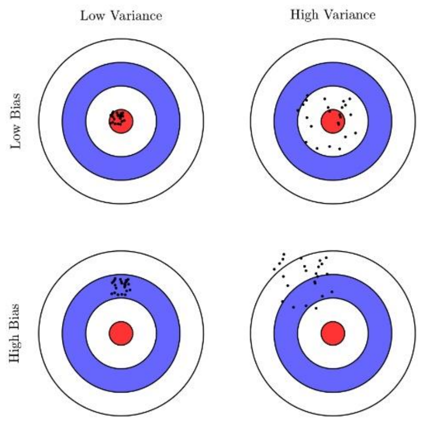


Figure 3: Bias-variance tradeoff [12].

2.3 Supervised learning

In machine learning and data mining, supervised learning is a technique for deducting a function from training data. The training data consist of pairs of objects (usually vectors): one component of the pair is the input data and the other, the desired results. The output of the function can be a numeric value (as in regression problems) or a class label (as in classification). The goal of supervised learning is to create a function capable of predicting the value corresponding to any valid input object after having seen a series of examples, training data [13]. For this, it has to generalize from the data presented to the situations not previously seen.

2.3.1 Labeled data

In order to train neural networks, particularly recurrent and deep neural networks, an enormous amount of data is needed since those systems automatically discover data representation of features needed for classification, as opposed to manual feature engineering (Chapter "*Feature dataset*" on page 23).

Google acquired reCAPTCHA in 2009. In the beginning goal of the service was to help in OCR (optical character recognition) while digitalizing books. Google defines it as: "*Hundreds of millions of CAPTCHAs are solved by people every day. reCAPTCHA makes positive use of this human effort by channeling the time spent solving CAPTCHAs into digitizing text, annotating images, building machine learning datasets. This, in turn, helps to preserve books, improve maps, and solve hard AI problems*" [14].

In recent times Google started using reCAPTCHA for classification of cars, buses, bicycles, fire hydrants, storefronts and other objects self-driving car may see and one example of that is shown in Figure 4.

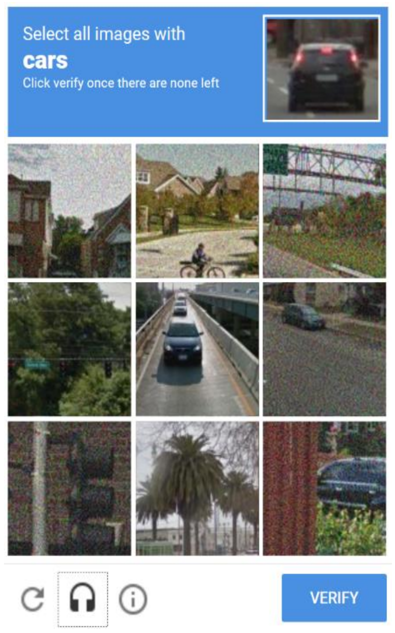


Figure 4: reCAPTCHA prompt to verify that the user is human

2.3.2 Tree

The tree is a data structure which has nodes, implemented as linked lists, in a parent-children relationship. Each parent node has zero or more children nodes. Node is a simple data structure which has a value and list of its (children) nodes. In Figure 5, we can see an example of the unordered tree. The node labeled 7 has two children, labeled 2 and 6, and one parent, labeled 2. The root node, at the top, has no parent [15].

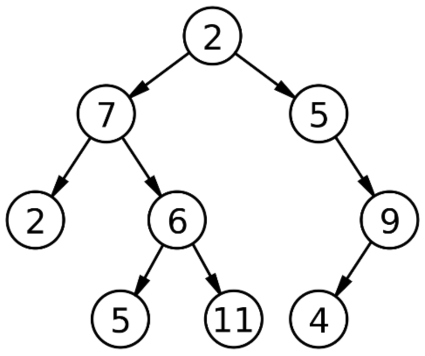


Figure 5: Labeled binary tree [15].

2.3.3 Decision Tree

A decision tree is a prediction model used in various fields ranging from artificial intelligence to economics. Given a set of data, diagrams of logical constructions are produced, very similar to the rules-based prediction systems, which serve to represent and categorize a series of conditions that occur successively.

A decision tree always consists of a root node and any number of inner nodes and at least two leaves. Each node represents a logical rule, and each leaf responds to the decision problem.

The complexity and semantics of the rules are not limited. For binary decision trees, each rule expression can take only one of two values. All decision trees can be transformed into binary decision trees.

To classify a single data object, go down the root node along the tree. At each node, an attribute is queried, and a decision is made about the selection of the following node. This procedure continues until you reach a leaf.

The error rate of a decision tree is the number of incorrectly classified data objects compared to the whole data set. This number is determined regularly both on train and test dataset. Depending on the field of application, it may be of particular importance to either keep down the number of false positive or false negative classified objects in particular. For example, in emergency medicine, it is much less harmful to treat a healthy patient than not to treat a sick patient. The effectiveness of decision trees is therefore always a context-dependent variable.

Decision trees can either be manually written down by experts or automatically induced from lessons learned using machine learning techniques. There are several competing algorithms for this. Creation of decision trees is usually done recursively in a top-down manner. For this purpose, it is crucial that a suitable data set with reliable and relevant empirical values for the decision problem (training data set) is available. This means that the classification of the target attribute must be known for each object of the training data record. For each step, the attribute with which the training data can best be split is searched for. To determine the best splitting point GINI index can be used. Gini impurity (GINI index) is a measure of how often a randomly chosen element from the set would be incorrectly labeled if it was randomly labeled according to the distribution of labels in the subset [16]. The procedure is applied recursively to the resulting subsets until only one object with classification is contained in each subset. In the end, a decision tree has been created which describes the empirical knowledge of the training data set in formal rules. The trees can now be used to automatically classify other records (test set or real-word data) or to interpret the resulting rules (feature importance and confidence). ID3 is an example of an algorithm which generates a decision tree from the dataset.

The big advantage of decision trees is that they are easy to explain and understand. This allows the user to evaluate the result and recognize major attributes. This is especially useful if feature importance is not known upfront.

The disadvantage of the decision trees is the relatively low classification accuracy in classification task on real-world data. For example, due to their discrete set of rules, trees perform slightly worse on most real-world classification problems than other classification techniques such as artificial neural networks or support vector machines (SVM). This means that although the trees can produce easily comprehensible rules for humans, these understandable rules often do not have the best possible quality statistically for real-world classification problems.

Another disadvantage is the possible size of the decision trees if no simple rules can be induced from the training data. On the one hand, a human observer quickly loses the overall view of the relationship between the many rules; on the other hand, such large trees usually also lead to overfitting of the training data set, so that new data sets are only automatically classified incorrectly. Therefore, so-called pruning methods have been developed, which reduce the decision trees to a reasonable size. For example, you can limit the maximum depth of trees or set a minimum number of objects per node.

By increasing the depth of the tree (number of splitting points), the overall accuracy of classification is increased. In Figure 6 there are only two features- X and Y coordinates are independent variables while the color of the dots is the dependent variable.

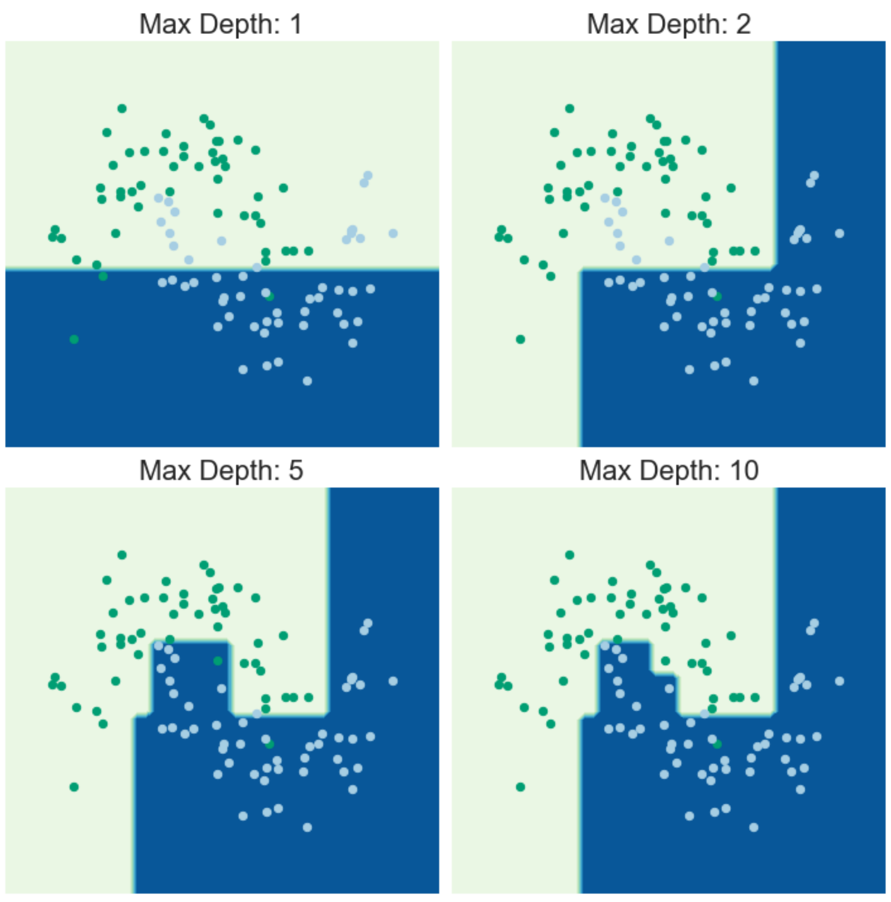


Figure 6: Iterations of a Decision Tree [17].

2.3.3.1 Improving Decision Trees

"Can a set of weak learners create a single strong learner?" [18]

One way to increase the classification quality when using decision trees is to use sets of decision trees instead of individual trees. Since a large decision tree tends to have high error rates and a high variance because there are many decision nodes from the root to the leaves, all of which are traversed under uncertainty. For example, bagging would calculate many small decision trees and use the average of their results, significantly reducing the variance (and error rate).

The disadvantage of decision-making forests is that it is no longer easy for humans to interpret the rules contained in all trees easily, as would be possible with individual trees.

Standard methods of producing suitable forests are boosting and bagging or bootstrapping. Bagging is a process of creating *n* bootstrapped samples of original dataset and training each classifier separately on a different dataset. New data is classified based on majority vote.

Documentation for *scikit-learn* defines bagging as: "In ensemble algorithms, bagging methods form a class of algorithms which build several instances of a black-box estimator on random subsets of the original training set and then aggregate their individual predictions to form a final prediction. These methods are used as a way to reduce the variance of a base estimator (e.g., a decision tree), by introducing randomization into its construction procedure and then making an ensemble out of it. In many cases, bagging methods constitute a very simple way to improve with respect to a single model, without making it necessary to adapt the underlying base algorithm. As they provide a way to reduce overfitting, bagging methods work best with strong and complex models (e.g., fully developed decision trees), in contrast with boosting methods which usually work best with weak models (e.g., shallow decision trees)". [19]

Bootstrap is a way to lower Random Forest from overfitting. It works by giving each tree random subset of the training data. Selection of subsets must be made in a way to resemble distribution and variability of the whole dataset.

Figure 7 shows visual difference between 10 different predictions. They overfit and do not generalize well. However, when taking the average of all predictors, each fitted to a subset of the original data set, we arrive at one bagged predictor (red line). The mean predictor is more stable, and there is less overfit compared to the individual predictors [20].

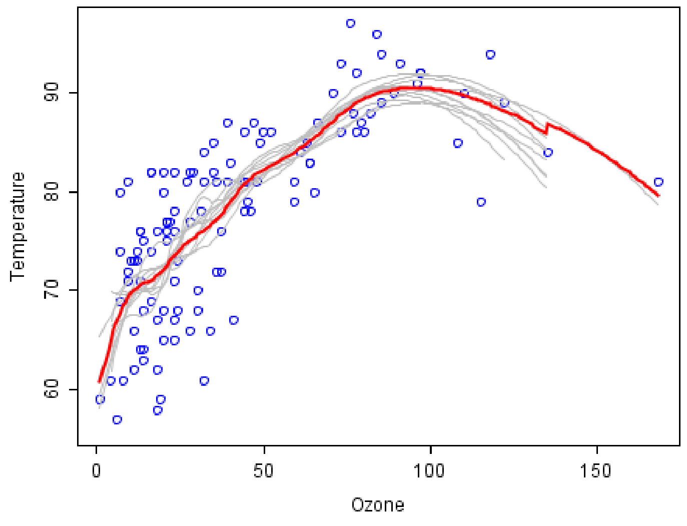


Figure 7: Visualization of ten different predictors and mean predictor (red color) [20].

Boosting specifies a series of objects and a series of weak classifiers. We are looking for a classifier that divides the objects into two classes with least errors. Boosting combines the existing weak classifiers so that the resulting new classifier makes as few mistakes as possible. Weak classifiers are very simple in structure and usually only consider a few features of the objects. They provide poor results on their own, but on the other hand, they can be evaluated very quickly. Boosting combines all weak classifiers into one that pays particular attention to the stronger ones among the weak classifiers, ignoring the really weak classifiers.

AdaBoost is an excellent example of boosting. It uses decision trees as weak learners and works by tweaking next weak learners in a way to emphasize those classifications misclassified by previous decision trees.

Decision tree algorithms are deterministic meaning that for the same data they will always create the same decision tree. If we run this same algorithm we will always get the same splitting points, and in the end, same classification or regression. There are two ways to randomized decision trees:

1. Randomization of data: uses the bootstrap method (sampling with replacement). Each tree is trained on the different training dataset.
2. The random sampling of possible split variables: each node is choosing among all predictors (independent variables) which predictor will give the best split and where should it be split (at which value). If we restrict possible predictors, for example by removing "best" one, we change the greediness of the algorithm to find the best split right now. In this way, we may get worse split in the next level (depth) of Random Forest, but next levels may be better. It can also happen that prediction is worse all the way to the end and that is why we are creating ensembles of trees. In the *scikit-learn* library, this value has the default value of *max_features=sqrt(n_features)* meaning it will take *n* random variables (where *n* is a square root of the number of all

possible independent variables) and decide splitting points upon those variable. Examples of other values are *log2(n_features)* or *max_features=n_features.*

2.3.4 Random Forest

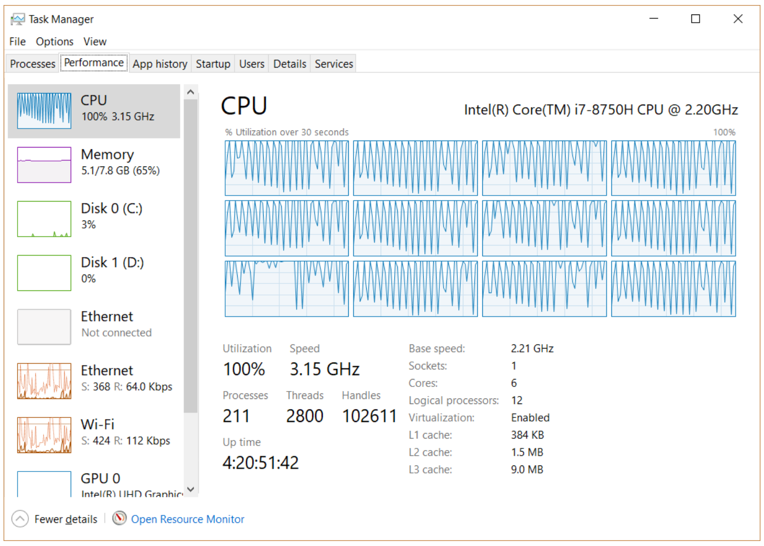
Random Forest is a classification method that consists of several uncorrelated decision trees. All decision trees have grown under a certain kind of randomization during the learning process. Randomization is done with bagging or boosting. For a classification, every tree in that forest is allowed to make a decision, and the class with the most votes decides the final classification. Random Forests can also be used for regression (predicting a numerical, continuous value).

Random Forest has a lot of advantages over other classification methods like Support Vector Machine (SVM):

- The classifier trains very fast. This advantage results from the short training or setup time of a single decision tree and the fact that the training time for a random forest increases linearly with the number of trees. Random Forest is embarrassingly parallel (simple to scale) because it can utilize multi-core CPUs. In Figure 8, CPU activity can be seen when grid search is run to find the best hyperparameters.
- The evaluation of a test example happens on each tree individually and is therefore parallelizable. Evaluation is also swift.
- It is very efficient for large amounts of data (many classes, many training examples, many features).
- Important features can be recognized.
- The connection between features can be recognized.
- Can handle hundreds of input variables.

However, there are some disadvantages:

- Random forests can easily over-fit.
- Unlike decision trees, the classification made by random forest is difficult to interpret. Random Forest is often considered as black-box type classifier but in the chapter "*Visualizing Random Forest*" I show an example of how a single simplified tree can be visualized to get a better sense of data.
- Random forest is not useful for predicting linear relationships between predictor (independent variable) and response (depended variable). In this case, something like linear regression may be better.



CPU Intel(R) Core(TM) i7-8750H CPU @ 2.20GHz

Utilization 100% Speed 3.15 GHz
Processes 211 Threads 2800 Handles 102611
Up time 4:20:51:42

Base speed:	2.21 GHz
Sockets:	1
Cores:	6
Logical processors:	12
Virtualization:	Enabled
L1 cache:	384 KB
L2 cache:	1.5 MB
L3 cache:	9.0 MB

Figure 8: CPU activity

2.3.4.1 *Curse of dimensionality*

Term curse of dimensionality is defined as when the dimensionality increases (the number of features, columns), the volume of the space increases so fast that the available data becomes sparse [21]. However, this is not a problem for Random Forest, and it is encouraged to add the maximum amount of features. Single feature like date and time can be expanded to several columns: “*is_8_pm*”, “*is_weekend*”, “*is_winter*”, “*is_public_holliday*”, etc. If any of these additional features are making model predictions worse they can simply be removed. See “*Selecting features*” chapter for a description of that process.

2.3.4.2 *Training Random Forest model*

This is the algorithm for training Random Forest ensemble method made from *N* Decision Trees on given training dataset with *M* features:

1. Create *N* bootstrapped (shuffled) training sets, where *N* is the number of Decision Trees.
2. Out of *M* features (dimensions of the dataset), choose *m* (*m* << *M*) features randomly. They are considered as criteria for the cut (split).
3. Train the tree with one of *N* training sets and use the rest as the test set to estimate the error.
4. For each node of the tree, randomly choose m variables on which to base the decision (split). Calculate the best partition of the training set from the *m* variables.
5. The tree is fully expanded until limits are reached (ex. depth of tree) or until there are no each leaf has one possible value (class of object).

To classify an input evaluate it in each tree. An outcome is calculated by a majority vote.

2.3.4.3 Usage of Random Forest predictive models in business

From the business point of view, predictive machine learning models can help us answer questions like "when and what products to stock in the inventory" and "when to promote people". However, predictive machine learning models don't function by just dumping data into them and seeing what happens.

Featuring engineering the dataset is extremely important. Best practices for training Random Forests are [22]:

- Data scientists should use domain expertise or business knowledge as a guide. There is no point of reinventing the wheel. Don’t try to refine the model to predict something which people in management or sales know for years. Find out client’s intuition how business and world works. Use those ideas and link them back to features to confirm or deny their existence.
- Interaction effects as new predictors should be explicitly defined.
- Multiple metrics should be used (like MAPE- Mean absolute percentage error). Don’t just use numerical difference, for example, the number of euros spent in marketing campaign this and last year. Also, use the difference in percentage because additional feature won’t hurt the Random Forest model, but they may be beneficial. It also removes outliers: 10000% increase prediction does not have the same importance if last year’s budget was €10 and it was increased to only €1000.

In business concept, it is critical to understand how the model works and how it makes decisions. If nobody understands it why would anyone allow the model to make decisions instead of humans? Best practices for interpreting random forests:

- Confirm which features are actually adding predictive value. Double check with the people in the industry.
- If features expected to be important are not then they may lead to data or model issues.
- Can reveal new business insights to change or confirm business strategy.

`Also, the directional relationship between top predictors and response should be checked. People who run business have to be convinced that the model will work when you put new data in it is going to lead to the right business decision. This can be done by manually reviewing and going to first few steps of the trees to detect patterns. Also, the model can be tested with synthetic data which is the same as the original one, except for a single predictor (feature) is varied.

2.3.5 Support-vector machine

Support-vector machine (SVM) is another popular supervised learning model which is often compared to Random Forest. “No free lunch theorem” states that there will always be datasets when one classifier is better than another. However, these are the reasons why Random Forest is the classifier of choice for this project:

1. Random Forest is better compared to any other classifier. Researchers compared 179 classifiers on 121 datasets and concluded that two best classifiers are: RF (94.1% accuracy) and SVM (92.3%) [23].
2. Faster to train. SVM cannot be trained in parallel and a lot of time is required to find out the perfect non-linear kernel. The computational complexity of SVM is much higher than for RF. This means that SVM will take longer to train compared to RF when the size of the training data is higher.
3. The scale of variables doesn’t matter.
4. SVM is better for binary classification (when the dataset has only two classes) and when data is clean and outlier free. Dataset used in this project has 16 classes.
5. For classification problems, Random Forest gives a probability of belonging to a class. SVM gives the distance to the boundary, but it is up to the author to convert it to the probability.
6. Random Forest is easier to grasp and there are more resources online. Occam's razor is a problem-solving principle that essentially states that simpler solutions are more likely to be correct than complex ones [24].

2.4 Computer Vision

Computer vision is a field which combines engineering, math, and programming with a goal to make computers better at understanding images. Some tasks are extremely easy for humans but extremely difficult for computers (for example object classification). Computer vision gathers and analyzes images to gain some useful information or to make a decision. Data on which computer vision acts on can have many forms: single grayscale image, a colored sequence of images, video, views from multiple cameras or angles or multi-dimensional data from some external device.

Areas in which computer vision is heavily applied are:

- recognition (object classification, facial or fingerprint recognition, handwritten digits identification, detection of cancer cells, content-based image retrieval, pose estimation, optical character recognition)
- motion analysis (tracking objects)
- scene reconstruction (generation 3D model of the scene)
- image restoration (removal of noise and motion blur)

Images can be made with various types of camera and result can be a 2D image, image sequence, 3D model or 3D point cloud. After this images need to be pre-processed (removal of noise and contrast enhancement). Feature extraction extracts different features like edges, corners, blobs, textures. If we are interested in a particular region of the image or want to extract or subtract a specific part of the image, then we are talking about segmentation methods. In chapter “*3.3.2 Shoe segmentation*” you can see the description of the implemented algorithm which removes background from an image of a shoe. After these steps, several useful things can be done as image recognition, image matching or flagging. [25]

2.5 Technologies

2.5.1 Python

Python is a popular language used in machine learning because it is free, very readable and has a great variety of machine learning libraries, primarily *numpy* for numerical linear algebra operations on multi-dimensional arrays and matrices, OpenCV for computer vision, *scikit-learn* for machine learning algorithms and *fastai* for more comfortable work with RF. It allows programmers to not overthink about strict syntax because Python is a high-level language where users don't have to think about memory management. The code is very readable, and because it is interpreted language, it allows quick modifications and code evaluations without needing to compile the whole program.

Work in Python was mostly done in Anaconda Jupyter Notebook. When installed, this package adds almost all python libraries. Included browser IDE is excellent for saving visualizations and code outputs and because it allows running code in separate cell blocks while saving the variables.

2.5.2 scikit-learn

Scikit-learn is a free ML library for the Python programming language. It features many classification, regression and clustering algorithms including SVMs, Random Forests, and k-means, and is designed to work with the Python numerical and scientific libraries NumPy and SciPy [26]. This library allows for easy creation and usage of Random Forest models as *RandomForestClassifier* objects (explained in "*Classification of dataset"* chapter) and k-means clustering method as *KMeans* object (explained in "*Color features*" chapter).

2.5.3 OpenCV

OpenCV is a standard library for every developer who works on computer vision problems. Most common areas of OpenCV applications are facial and object recognition, motion tracking, mobile robotics, and many others. OpenCV is available in C++, Python and Java interfaces and supports Windows, Linux, Mac OS, iOS and Android. OpenCV is highly efficient for real-time applications because it is written in C/C++. Support for CUDA and OpenCL enables hardware acceleration.

2.5.4 Computer Vision Algorithms

2.5.4.1 Histogram equalization

Histogram equalization is a method of contrast adjustment using the image's histogram. In Figure 9, *x* coordinate represents colors ranging from 0 to 255 and *y* coordinate number of pixels with *x* value. The right plot has intensities equality distributed on all range of values.

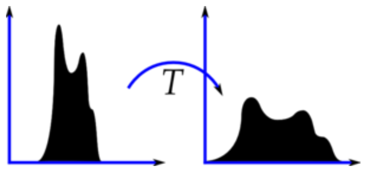


Figure 9: Histogram before and after equalization

2.5.4.2 CLAHE

CLAHE stands for Contrast Limited Adaptive Histogram Equalization. It is used because it improves edge detection in the process of background removal. In Figure 10, we can see the original image, an image with equalized contrast and image with CLAHE applied. CLAHE works by computing histogram for each patch of 8x8 pixels and then redistributing the lightness values of that particular patch. This method is better compared to histogram equalization on the whole image because it improves local contrast and enhances edges in each region (head is overexposed in the 2nd picture, while on third details are still seen).

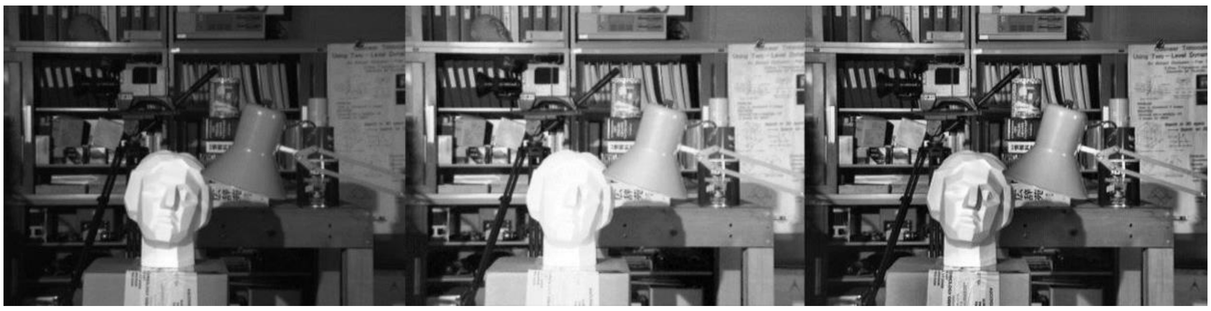


Figure 10: Original image. Image with equalized contrast. Image on which CLAHE is applied [27].

2.6 The conclusion of the chapter

In this chapter, the reader has been introduced to AI and ML concepts and abbreviations. Common terms used in Machine Learning have been defined and explained so that readers who are new to this field of computer science can gently be introduced and can learn new concepts. It is worth mentioning that the rest of this project is done using Random Forest Classifier and reasons for using it have been explained in the chapter "*2.3.5 Support-vector machine*".

3 Datasets

In this chapter, the reader will find out how author downloaded shoe images by parsing provided XML file and by scrapping Camper website. Then shoe segmentation will be explained and what features can be extracted from a shoe image.

In order to train any machine learning dataset is needed. Bigger dataset means better accuracy- more data on which model can generalize. The current dataset used for machine learning is a CSV file which has 617 rows with 57 columns- meaning 617 pictures have an identification number, big category (the dependent variable which we are trying to predict) and 55 features which describe a single picture of one shoe.

Figure 11 shows a disproportional number of objects in each category- five categories have more than 60 objects but remaining 10 have a few dozens. In chapter "*5.3 Changing the number of categories*" you will see how accuracy improves with a decrease in the number of class labels (shoe categories).

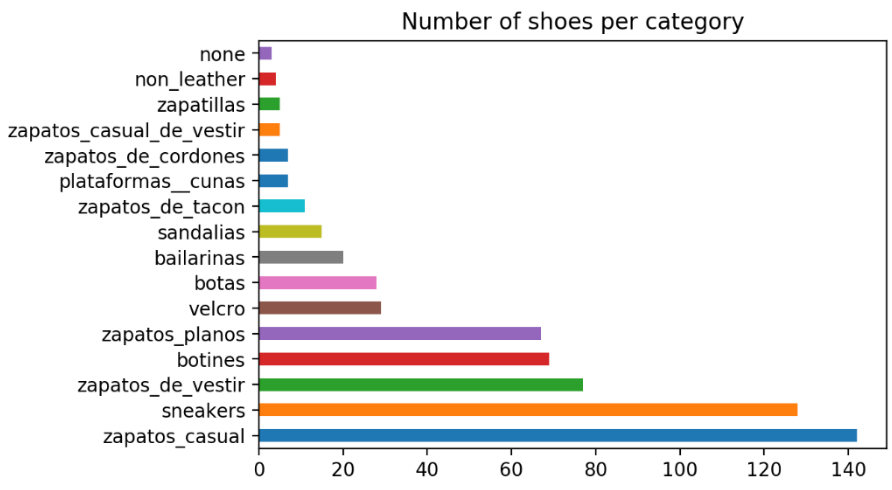


Figure 11: Disproportional number of shoes in each category

3.1 Scrapping the website

The Camper company provided the XML file. It had a link to the webpage of each shoe model. The author made a Python script to visit each webpage and extract direct URL links of shown images of shoes. The script needs up to 60 minutes to finish creating the dataset *data.csv*. That file contains these columns:

- *option* is a unique ID (for example "*K100277-002*")
- *link* points to shoe model webpage
- *bigcategory* is the category of the shoe (e.g., *"Sneakers"*)
- *smallcategory* is subcategory (e.g., *"Runners"*)
- *imagelink1* is a direct link to an image of both shoes from the side (**_F.jpg* filename)
- *imagelink2* is a direct link to an image of both shoes from the other side (**_T.jpg*)

- *imagelink3* is a direct link to top view image of both shoes (**_C.jpg*)
- *imagelink4* is a direct link to bottom view image of shoe sole (**_P.jpg*)
- *imagelink5* is a direct link to a sideways image of one shoe(**_L.jpg*)

imagelink1	*imagelink2*	*imagelink3*	*imagelink4*	*imagelink5*

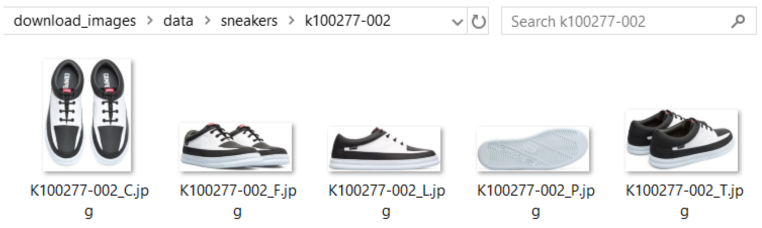
Dataset created from scrapping the website looks like this:

0	option,link,bigcategory,smallcategory,imagelink1,imagelink2,imagelink3,imagelink4,imagelink5,ava ilable
1	K100277-002,https://www.camper.com/es_ES/hombre/zapatos/runner/camper-runner-K100277-002,Sneakers,Runner,https://cloud.camper.com/is/image/ZXNob3AwMQ==/K100277-002_F.jpg, https://cloud.camper.com/is/image/ZXNob3AwMQ==/K100277-002_T.jpg, https://cloud.camper.com/is/image/ZXNob3AwMQ==/K100277-002_C.jpg, https://cloud.camper.com/is/image/ZXNob3AwMQ==/K100277-002_P.jpg, https://cloud.camper.com/is/image/ZXNob3AwMQ==/K100277-002_L.jpg,false
2	…

3.2 Image files dataset

After the website has been scrapped, and the dataset of direct image links has been created, it was necessary to download those images. Even though there were 750 individual shoes in the XML file, not all images could be download because there was a lot of missing web pages (*404 Not Found* error messages). That is why there are 617 folders (one for each shoe model) inside 16 folders representing different categories of shoes.

```
data
  shoe_category_1
    shoe_id_1
      1_C.jpg
      1_F.jpg
      1_L.jpg
      1_P.jpg
      1_T.jpg
    shoe_id_2
      ...
    ...
  shoe_category_2
    shoe_id_56
      ...
    shoe_id_78
      ...
    ...
  ...
```



download_images > data > sneakers > k100277-002

Search k100277-002

K100277-002_C.jpg K100277-002_F.jpg K100277-002_L.jpg K100277-002_P.jpg K100277-002_T.jpg

Figure 12: Folder structure of the dataset

In Figure 12: Folder structure of the dataset, we can see the folder structure of the image dataset: inside "*data*" folder there are 16 folders representing different shoe categories (e.g. "*bailarinas*", "*botas*"…). Inside each category, there are folders representing individual shoe models, named by their *option* (ID), like "*k100277-002*" and those folders contain five pictures.

3.3 Feature dataset

These features are based on shoe mask (physical shape of the shoe) and in order to get that shape first background has to be removed. This process is explained in the chapter *"Shoe segmentation algorithm"*. Following features are from *UIB - V Features*[2] library.

- `solidity` is a proportion between the area of the object in the mask and the convex-hull
- `ch_perimeter` is a perimeter of the convex-hull
- `ch_area` is the area of the convex hull from the contour, using the same function as the area of an object
- `bb_area` is area of the bounding box of the contour.
- `rectangularity` is area of the bounding box of the contour.
- `min_r` is a minor radius of the ellipse from the contour
- `max_r` is a radius of the enclosing circle of the contour
- `ferret` is a major diagonal of the enclosing ellipse of the contour
- `breadth` is a minor diagonal of the ellipse from the contour
- `circularity` is a likeliness of an object to a circle
- `roundness` is circularity corrected by the aspect ratio (explanation can be found in the paper "*New parameter of roundness R: circularity corrected by aspect ratio*")
- `feret_angle` is the angle between the ferret and the horizontal
- `eccentricity` shows how much the conic section deviates from being circular. For any point of a conic section, the distance between a fixed point F and a fixed straight line l is always equal to a positive constant, the eccentricity. Is calculated by the relation between the two diagonals of the ellipse.
- `center_x` is X value of centroid of a contour. The centroid of a plane figure is the arithmetic mean of all the point in the figure.
- `center_y` is Y value of centroid of contour.
- `sphericity` is the proportion between the major and the minor ferret
- `aspect_ratio` is the proportional relationship between its width and its height
- `area_equivalent` is the diameter of the real area of the contour
- `perimeter_equivalent` is the diameter of the real perimeter of the contour
- `equivalent_elipse_area` is the area of the equivalent ellipse
- `compactness` is the proportion between area and the shape of the ellipse

[2] https://pypi.org/project/uib-vfeatures/ and https://gitlab.com/miquelca32/features

- `area` is the area of the object of the mask
- `convexity` is convexity of the contour. The convexity is a measure of the curvature of an object. It is the relation between the perimeter of the convex hull and the perimeter of the object.
- `shape` is the relation between perimeter and area. Calculates the elongation of an object.
- `perimeter` is the perimeter of the object in the mask.

UIB - V Features library is used because researchers from the UGIVIA[3] laboratory have made it and it has shown good results for previous projects done in the laboratory.

3.3.1 Removing background

In order to extract features from the image, a computer needs to know what precisely on input picture a shoe is. The task was to create an algorithm which would separate shoe from the background, returning image mask as a black and white image. That mask is used for feature extraction.

On this picture we see 5 images: original, black and white, black and white with CLAHE applied, filled contours, the shoe with a background changed with green color.

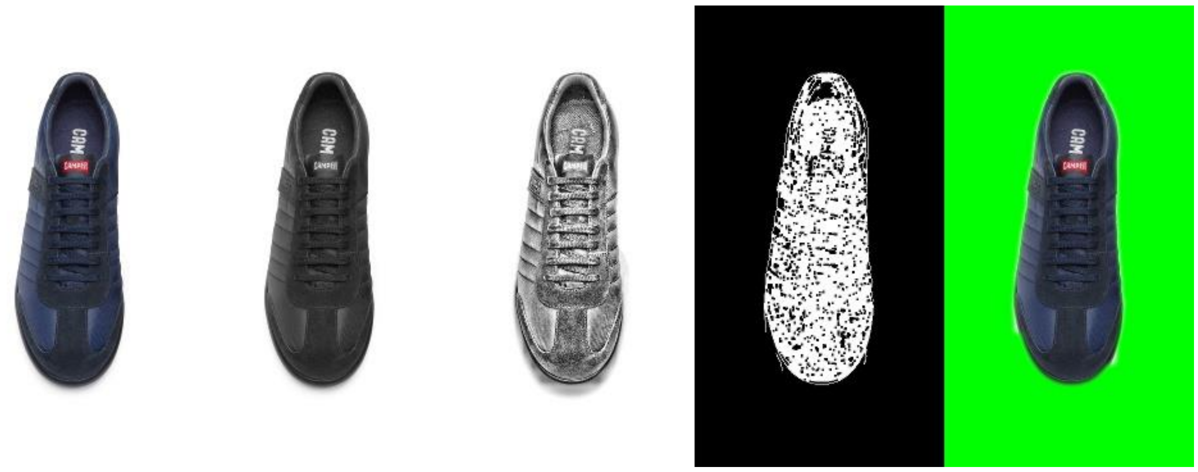


Figure 13: Successful background removal cv2.createCLAHE(clipLimit=8.0, tileGridSize=(32,32))

[3] http://ugivia.uib.es/

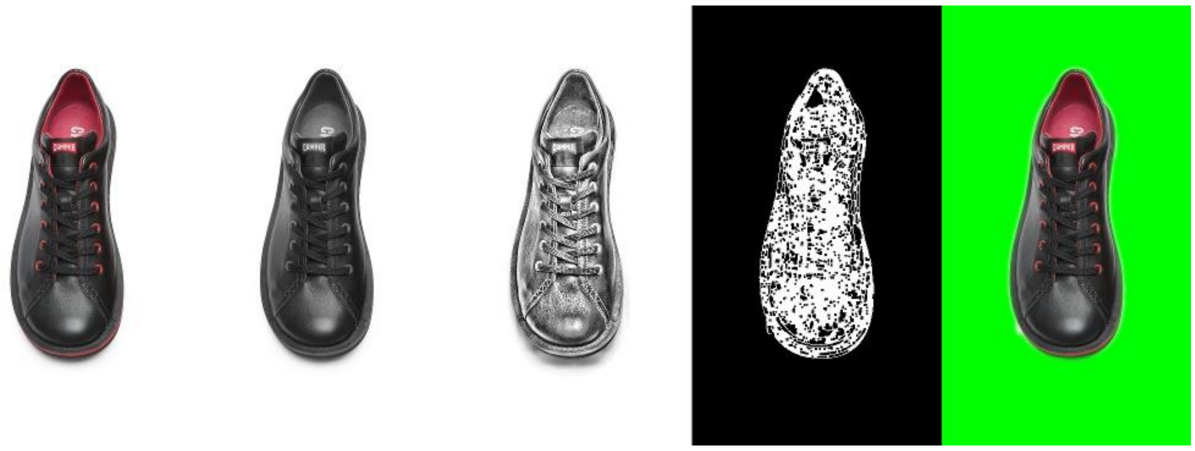


Figure 14: Successful background removal
cv2.createCLAHE(clipLimit=8.0, tileGridSize=(32,32))

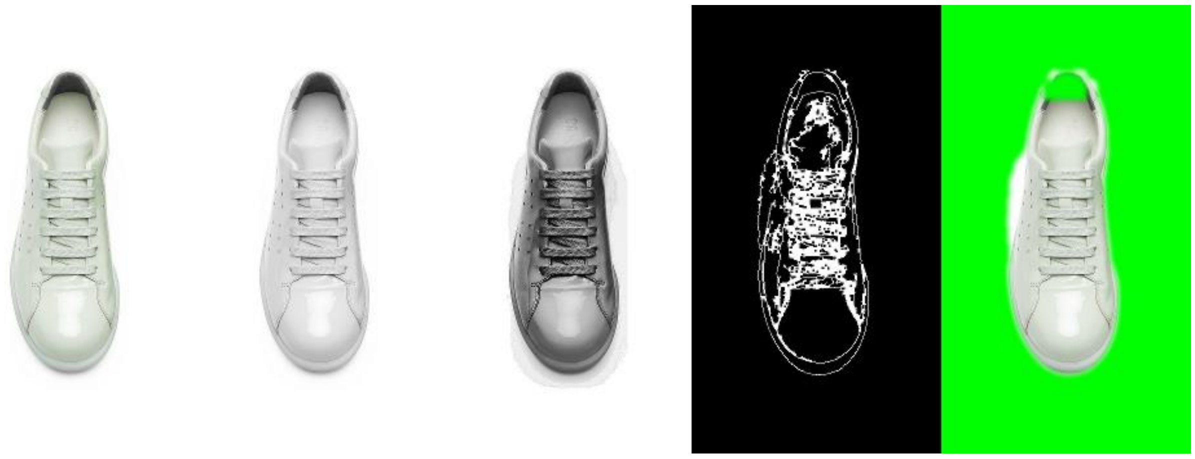


Figure 15: Unsuccessful background removal.
cv2.createCLAHE(clipLimit=16.0, tileGridSize=(4,4))

Artifacts can be seen in step when CLAHE is applied in Figures 16 and 17. Artifacts are made only when a shoe or outer part of the shoe is the same color as a background, in this case, white.

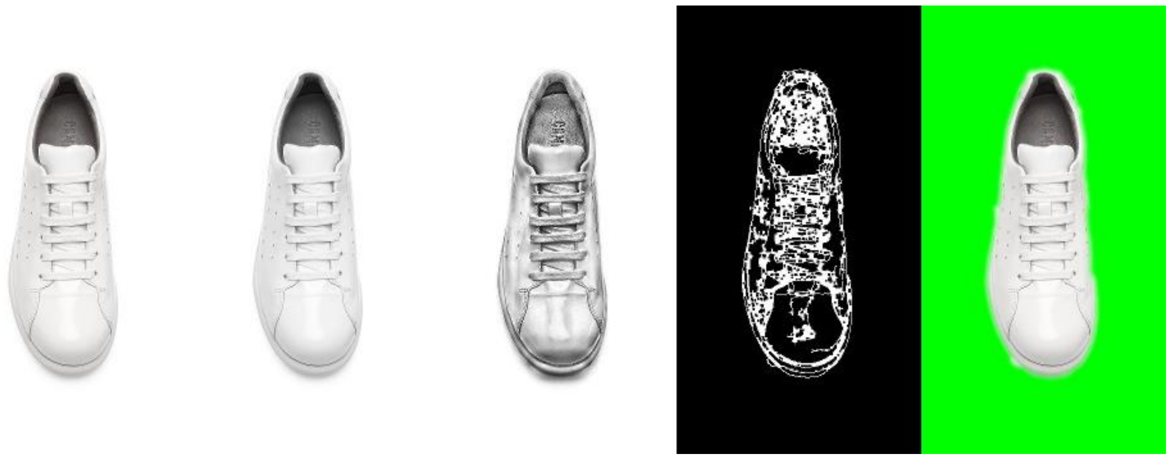


Figure 16: Acceptable background removal

cv2.createCLAHE(clipLimit=8.0, tileGridSize=(32,32))

However, if we know what CLAHE parameters to tweak artifacts can be minimized and thus results can be better (Figure 17).

3.3.2 Shoe segmentation algorithm

1. Cropping the left shoe out of `*_C.jpg` type image.
    a. Input image consists of the image of the left and right shoe taken from the top view.
2. Adding a border to the image.
3. Resizing the image to the height of 720 pixels.
    a. Smaller images give better results but there is a tradeoff between successful removal of background and image quality.
4. Increasing the contrast using CLAHE (3rd picture in Figure 13 - Figure 16).
    a. `clahe_clip_limit` and `clahe_grid_size` are two customizable parameters which are responsible for the output result. The grid size of 32x32 pixels is the default.
5. Detecting edges using Canny Edge Detector[4] (4th picture in Figure 13 - Figure 16).
6. Finding largest contour.
    a. Once we largest contour by area is found, it is dilated and eroded multiple times and then blurred so that there is less inconsistency of the shape.
7. Separating the shoe from the background using the mask.
8. Removing the border and adding transparency.

3.4 Improving dataset

Once feature dataset has been made, work on Random Forest model had begun. After tweaking hyperparameters and experimenting with various things, the author wanted to see if it is possible to add more features. Since previous features were only based on the shape of a shoe, no feature was based on color or texture. In the following chapters, you will see how those features can be extracted and used.

3.4.1 Color features

More features can be added if we take into the account the color. Perhaps some categories have distinctive colors- boots are maybe made out of leather, and maybe they have a black sole. The only way to figure that out is to extract those features and then test them to see if they improve the results. These features were calculated: the average color of the whole image and five dominant colors. Each one of them is represented in the dataset as three different features- hue, saturation, and value.

HSV (hue, saturation, value) is alternative representations of the RGB color model. Colors of each hue are arranged in a radial slice, around a central axis of neutral colors which ranges from black at the bottom to white at the top. The HSV representation models the way

[4] https://docs.opencv.org/2.4/doc/tutorials/imgproc/imgtrans/canny_detector/canny_detector.html

paints of different colors mix together, with the *saturation* dimension resembling various shades of brightly colored paint, and the *value* dimension resembling the mixture of those paints with varying amounts of black or white paint [28].

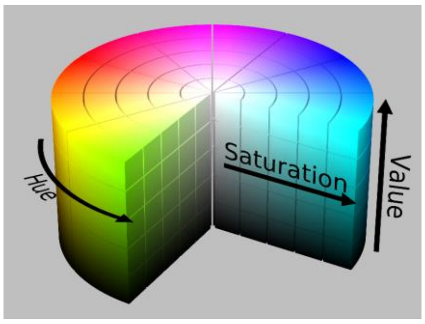


Figure 17: HSV cylinder

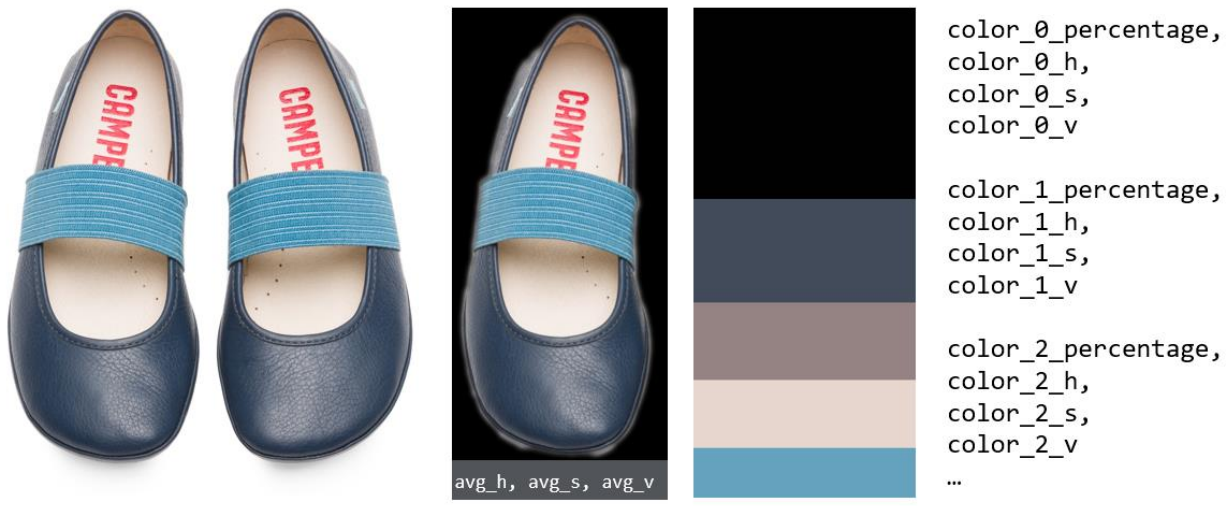


Figure 18: Input picture. The shoe with the removed background. Visual representation of dominant colors. Names of 4 features for each dominant color.

In Figure 19, we can see left shoe with the removed background. Colored rectangle under it shows the average color of the whole image, and from it we can get average hue, average saturation and the average value of the whole image. On the right side, you can see the visual representation of the percentage of the five most dominant colors.

3.4.2 Texture features

Once we input grayscale picture of a shoe without background, we can get several properties which serve as a compact summary of the matrix: contrast, dissimilarity, homogeneity, ASM, energy, and correlation (Table 1: Texture features). First, grey-level co-occurrence matrix has to be calculated (histogram of co-occurring greyscale values at a given offset over an image). The input image should contain integers in [0, levels-1], where $levels$ indicate the number of grey-levels counted (typically 256 for an 8-bit image). The maximum value is 256 [29].

Table 1: Texture features

Contrast	$\sum_{i,j=0}^{levels-1} P_{i,j}(i-j)^2$
Dissimilarity	$\sum_{i,j=0}^{levels-1} P_{i,j}\lvert i-j \rvert$
Homogeneity	$\sum_{i,j=0}^{levels-1} \frac{P_{i,j}}{1+(i-j)^2}$
ASM	$\sum_{i,j=0}^{levels-1} P_{i,j}^2$
Energy	$\sqrt{ASM}$
Correlation	$\sum_{i,j=0}^{levels-1} P_{i,j}\left[\frac{(i-\mu_i)(j-\mu_j)}{\sqrt{(\sigma_i^2)(\sigma_j^2)}}\right]$

3.5 The conclusion of the chapter

In this chapter the reader has been introduced to the dataset: how were image files obtained, how images look and how they were stored. Removal of background has been explained in detail, along with used features. This process is common task before training any other supervised machine learning algorithm and is often used: acquire the data, label the data, clean the data and extract features.

4 Classification of dataset

For classification of the dataset, Random Forest was chosen machine learning algorithm. In the previous chapter “*2.3.4 Random Forest*” you can read the detailed description of how it works and in chapter “*2.3.5 Support-vector machine*” it is explained why SVM was not used. Random Forest is an ensemble method which, compared to SVM, is more accurate on multi-class classification datasets, faster to train, easier to grasp and implement, and that is why it was used in this project.

The goal of ensemble methods is to combine the predictions of several base estimators built with a given learning algorithm to improve generalizability or robustness over a single estimator. Combining is done by building several estimators independently and then averaging their predictions. On average, the combined estimator is usually better than any of the single base estimator because its variance is reduced. [19]

4.1 Tuning hyperparameters

Hyperparameter is a parameter whose value is set before the learning process begins. By contrast, the values of other parameters are derived via training. Given some hyperparameters, the training algorithm learns the parameters from the data. The time required to train and test a model can depend upon the choice of its hyperparameters. Hyperparameter optimization finds a tuple of hyperparameters that yields an optimal model which minimizes a predefined loss function on given test data [30]. The traditional way of performing hyperparameter optimization has been grid search which is an exhaustive search through a manually specified subset of the hyperparameter space of a learning algorithm. A grid search algorithm must be guided by some performance metric, typically measured by cross-validation on the training set.

Even though the grid search is embarrassingly parallel, meaning it is easily split to separate CPU cores if too many parameters were set to be tested, time needed to find optimal hyperparameters can span several days.

This is the example of possible hyperparameters for Random Forest model.

```
param_grid = {
            "n_estimators" : [10, 100, 256, 512, 1024],
            "max_depth" : [None, 8, 16],
            "min_samples_leaf" : [1, 2, 4, 8],
            "max_features": ['sqrt','log2', 1.0, 0.5, 0.25]}
```

Explanation:

- `n_estimators` is the number of decision trees in Random Forest ensemble. 256 give good results.
- `max_depth` limits the depth of each tree. In “Visualizing Random Forest” chapter there is an example of a single tree with a depth of size 2. We don’t want to limit the depth because it gives better results if not limited.

- `min_samples_leaf` means that each decision tree should split samples until there is 1 or more sample on each leaf.
- `max_features` limits the number of features each decision tree uses when consideration for a split. Since there is no a huge dataset in term or both rows and columns (features), it is better to use all feature (`1.0`)

4.2 Feature importance

Feature importance works by randomly shuffling a column (rows inside a column), each column one at a time, then see how accurate the pre-trained model is when you pass in all the data as before but with one column shuffled [31].

Important features are the features that are more closely related to the dependent variable and contribute more for variation of the dependent variable.

How feature importance is calculated [32]:

1. Train random forest model with optimal hyper-parameters.
2. Find prediction score of the model (accuracy of correctly guessed shoe categories).
3. Find prediction scores *p* more times where *p* is number of features, each time randomly shuffling the column of *i*-th feature
4. Compare all p scores with the benchmark score. If randomly shuffling some *i*-th column is lowering the score, which means that our model is worse without that feature.
5. Remove the features that do not hurt the benchmark score and retrain the model with a reduced subset of features.

4.3 Feature similarity

In our dataset, there are some variables which basically measure the same thing. They are only confusing feature importance and they also make our random forest slightly less good because the model then requires more computation because there are more columns to check. Obvious goal is to remove redundant features.

The dendrogram is a useful graph which shows a similarity between different features: features which are more similar will join sooner (Figure 19).

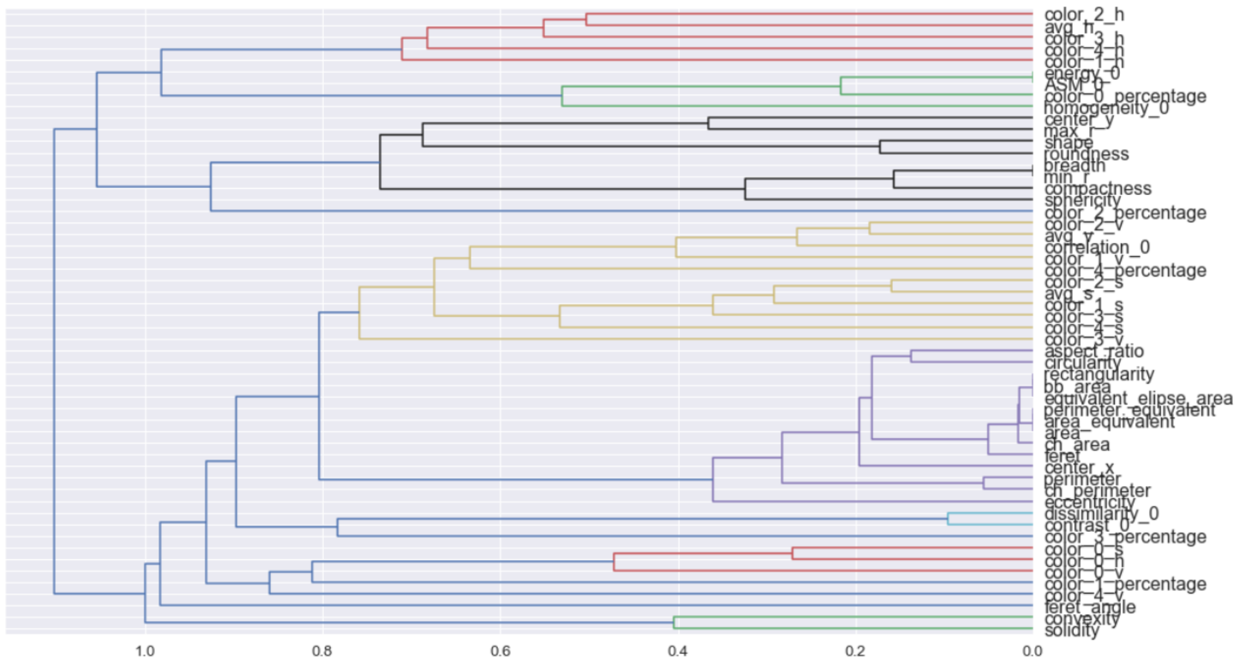


Figure 19: Dendrogram of the feature similarity

The dendrogram illustrates how each cluster is composed by drawing a “-[“ shaped link between a non-singleton cluster and its children. The left side of the “-[“ link indicates a cluster merge. The two legs of the “-[“ link indicate which clusters were merged. The length of the two legs of the “-[“ link represents the distance (similarity) between the child clusters [33].

4.4 Selecting features

Using images in the image dataset, new feature dataset has been created with 56 extracted features. That is not an exceptionally high number, but maybe our model can predict with better accuracy if we reduce the number of features. Even if the model predicts with the same accuracy, dimensionality reduction will help significantly in terms of time needed to train the model.

Features importances in Figure 20 and Figure 21 are hugely depended on the train and test set so they are a little bit different on every run.

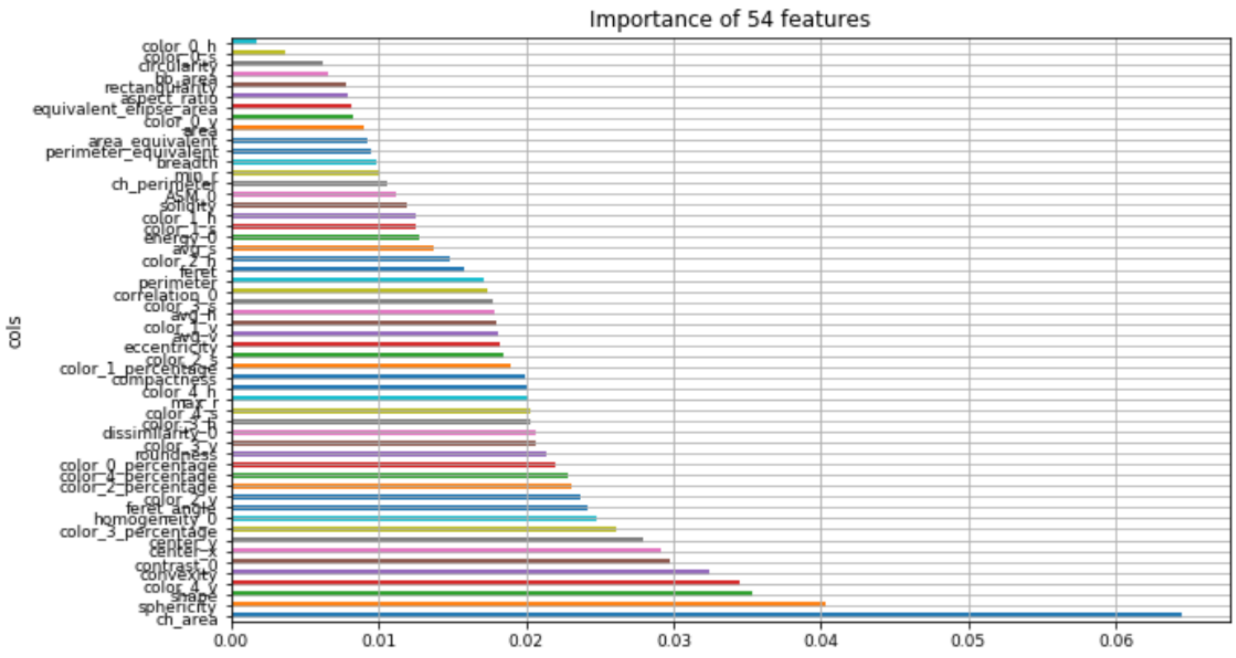


Figure 20: Accuracy of the model which uses 54 features, on this particular train and test dataset, is 41.613%.

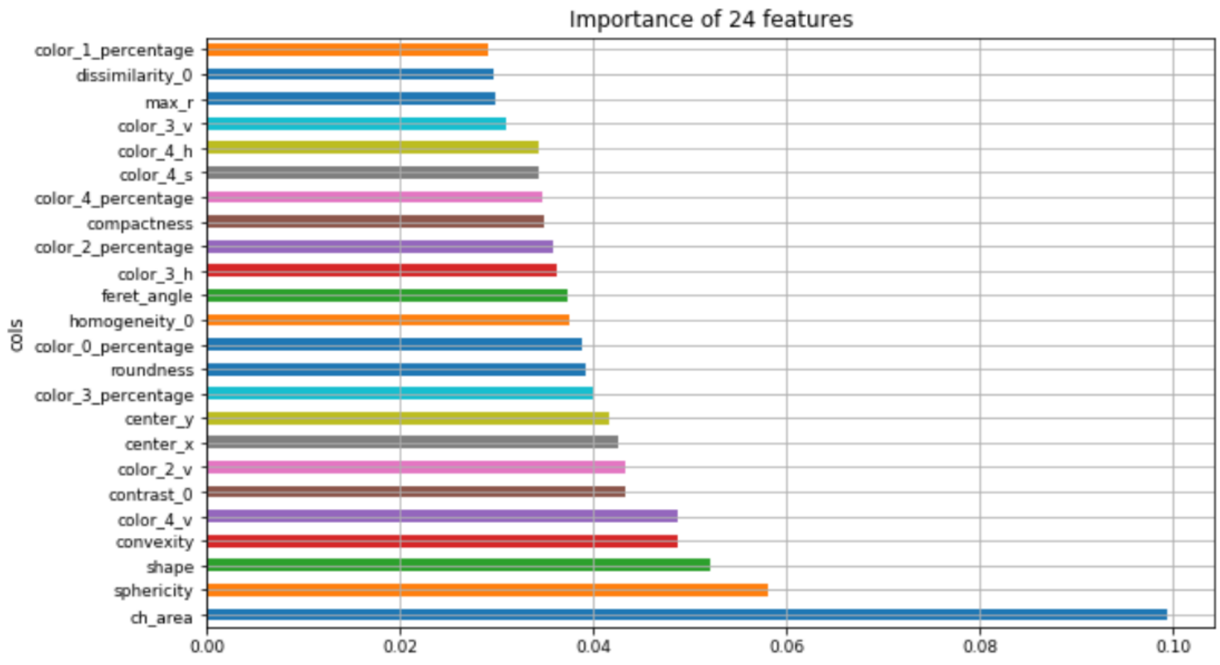


Figure 21: The accuracy of the model which uses 24 features, on this particular train and test dataset, is 43.226%

As we can see in this particular case dimensionality (feature) reduction, the model accuracy has increased when the number of features was decreased. Next thing which can be done is to plot the model accuracy when in dataset there is only first best feature, then first and second and so on. As we can this in Figure 23, there is no point of using more than 20-30 features because model accuracy does not increase significantly, but time complexity does.

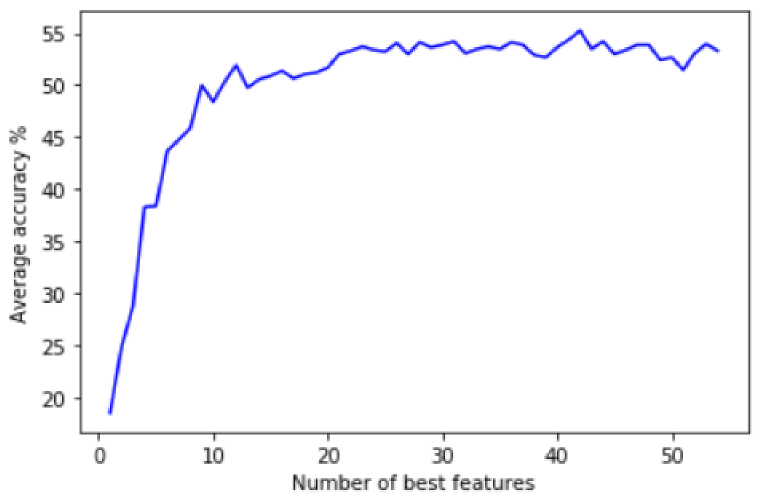


Figure 22: Accuracy compared to the increasing number of best features

4.5 Interpreting Random Forest

4.5.1 Visualizing Decision Trees

Since Random Forest is an ensemble of decision trees, where each tree has a randomized dataset, it is difficult to visualize it. However, we can make a *RandomForestClassifer* with ten trees and limit its maximum depth to two. In this particular dataset, the fifth decision tree uses *ch_area* as the first splitting point (Figure 23). Afterward, this decision tree uses two other features to do more splits.

```
RandomForestClassifier(n_estimators=10, max_depth=2, bootstrap=False)
```

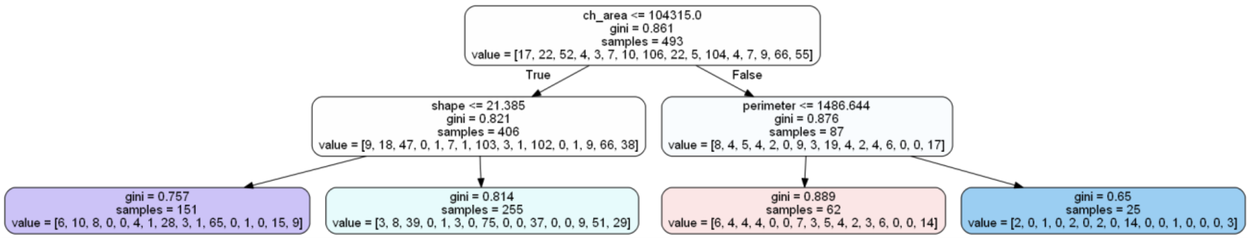

ch_area <= 104315.0
gini = 0.861
samples = 493
value = [17, 22, 52, 4, 3, 7, 10, 106, 22, 5, 104, 4, 7, 9, 66, 55]
True
False
shape <= 21.385
gini = 0.821
samples = 406
value = [9, 18, 47, 0, 1, 7, 1, 103, 3, 1, 102, 0, 1, 9, 66, 38]
perimeter <= 1486.644
gini = 0.876
samples = 87
value = [8, 4, 5, 4, 2, 0, 9, 3, 19, 4, 2, 4, 6, 0, 0, 17]
gini = 0.757
samples = 151
value = [6, 10, 8, 0, 0, 4, 1, 28, 3, 1, 65, 0, 1, 0, 15, 9]
gini = 0.814
samples = 255
value = [3, 8, 39, 0, 1, 3, 0, 75, 0, 0, 37, 0, 0, 9, 51, 29]
gini = 0.889
samples = 62
value = [6, 4, 4, 4, 0, 0, 7, 3, 5, 4, 2, 3, 6, 0, 0, 14]
gini = 0.65
samples = 25
value = [2, 0, 1, 0, 2, 0, 2, 0, 14, 0, 0, 1, 0, 0, 0, 3]

Figure 23: 5th out of 10 decision trees in Random Forest ensemble

However, if we visualize first and sixth decision trees, we can see that they are completely different. Even though they both use *area_equivalent* feature as the first decision for splitting the data, other splitting points differ.

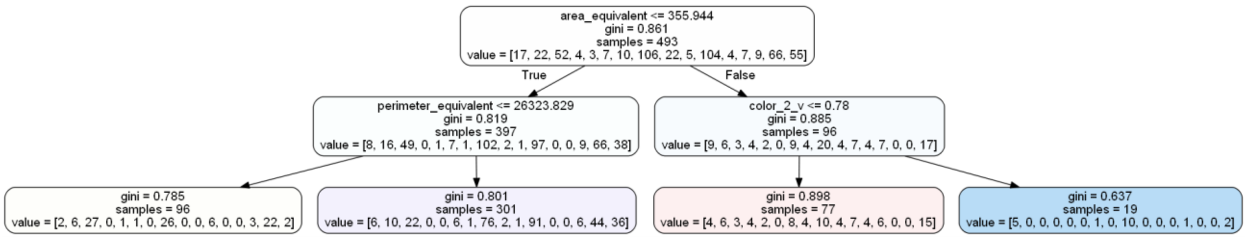

area_equivalent <= 355.944
gini = 0.861
samples = 493
value = [17, 22, 52, 4, 3, 7, 10, 106, 22, 5, 104, 4, 7, 9, 66, 55]
True
False
perimeter_equivalent <= 26323.829
gini = 0.819
samples = 397
value = [8, 16, 49, 0, 1, 7, 1, 102, 2, 1, 97, 0, 0, 9, 66, 38]
color_2_v <= 0.78
gini = 0.885
samples = 96
value = [9, 6, 3, 4, 2, 0, 9, 4, 20, 4, 7, 4, 7, 0, 0, 17]
gini = 0.785
samples = 96
value = [2, 6, 27, 0, 1, 1, 0, 26, 0, 0, 6, 0, 0, 3, 22, 2]
gini = 0.801
samples = 301
value = [6, 10, 22, 0, 0, 6, 1, 76, 2, 1, 91, 0, 0, 6, 44, 36]
gini = 0.898
samples = 77
value = [4, 6, 3, 4, 2, 0, 8, 4, 10, 4, 7, 4, 6, 0, 0, 15]
gini = 0.637
samples = 19
value = [5, 0, 0, 0, 0, 0, 1, 0, 10, 0, 0, 0, 1, 0, 0, 2]

Figure 24: 1st out of 10 decision trees in Random Forest ensemble

In this particular decision tree (Figure 24), “*area_equivalent”* feature is the first splitting point. Afterward, data is split based on “*perimeter_equivalent”* and “*color_2_v”* features.

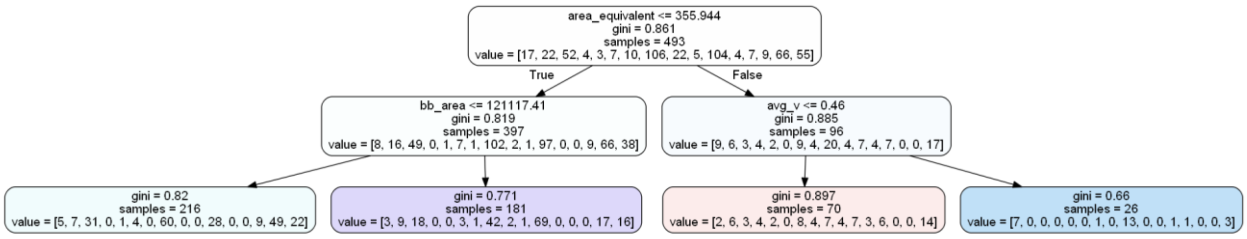

area_equivalent <= 355.944
gini = 0.861
samples = 493
value = [17, 22, 52, 4, 3, 7, 10, 106, 22, 5, 104, 4, 7, 9, 66, 55]
True
False
bb_area <= 121117.41
gini = 0.819
samples = 397
value = [8, 16, 49, 0, 1, 7, 1, 102, 2, 1, 97, 0, 0, 9, 66, 38]
avg_v <= 0.46
gini = 0.885
samples = 96
value = [9, 6, 3, 4, 2, 0, 9, 4, 20, 4, 7, 4, 7, 0, 0, 17]
gini = 0.82
samples = 216
value = [5, 7, 31, 0, 1, 4, 0, 60, 0, 0, 28, 0, 0, 9, 49, 22]
gini = 0.771
samples = 181
value = [3, 9, 18, 0, 0, 3, 1, 42, 2, 1, 69, 0, 0, 0, 17, 16]
gini = 0.897
samples = 70
value = [2, 6, 3, 4, 2, 0, 8, 4, 7, 4, 7, 3, 6, 0, 0, 14]
gini = 0.66
samples = 26
value = [7, 0, 0, 0, 0, 0, 1, 0, 13, 0, 0, 1, 1, 0, 0, 3]

Figure 25: 6th out of 10 decision trees in Random Forest ensemble

In this particular decision tree (Figure 25), “*area_equivalent”* feature is the first splitting point. Afterward, data is split based on “*bb_area”* and “*avg_v”* features.

4.5.2 Partial dependence

Once we have trained our Random Forest Classifier model, we can use it to understand the data better. The Partial Dependence Plot (short PDP or PD plot) shows the marginal effect one or two features have on the predicted outcome of a machine learning model [34]. We are going to find out how a few most important features relate to the dependent variable (*bigcategory*).

Following charts (Figure 26 and Figure 27) show how changing one variable and keeping all values of all other features the same influences the depended variable. However, since dataset has many features which are probably dependent on each other, these charts cannot be used on their own predict classification. Instead, they show us insight how our Random Forest model classifies data.

Explanation of Figure 26: if values of all other features are the same, and we only change *ch_area* feature (*ch_area* is the area of the convex hull from the contour) with values ranging from 85000 to 100000, on X-axis we can see different classification results. If *ch_area* has the value of 95000, then the object will most likely be classified as class 1 or 2 (“*bailarinas”* or” *botas”*). If *ch_area* has the value of bellow 90 000 then object is most likely class 12 or 13 (“*zapatos_de_cordones”* or “*zapatos_de_tacon”*).

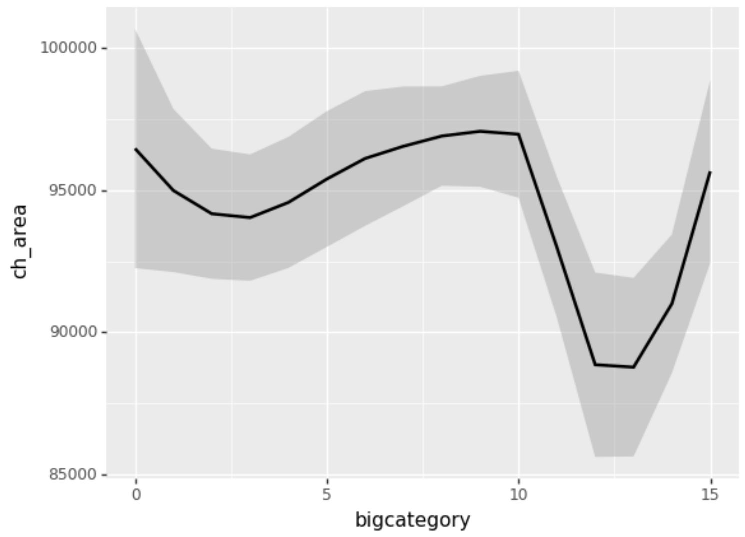


Figure 26: Partial dependence plot for ch_area and bigcategory

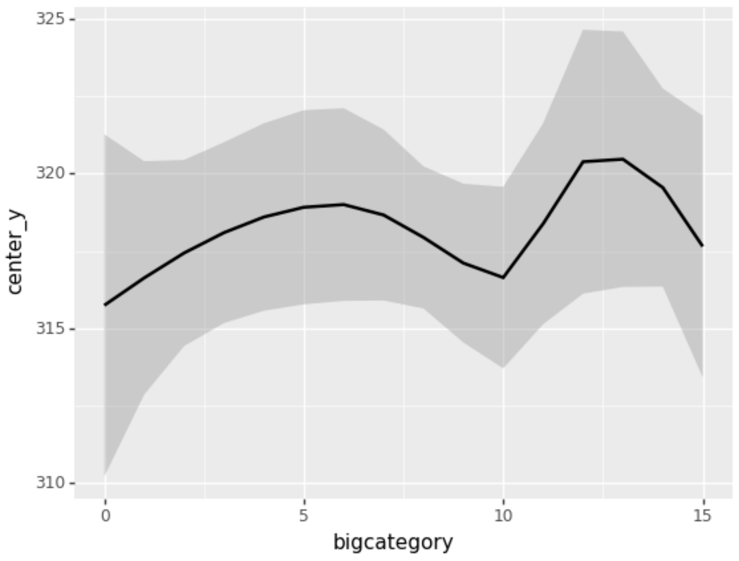


Figure 27: Partial dependence plot for "ch_center_y" and "bigcategory" features

Category of shoes is not shown in a textual format like "*bailarinas*" and "*botas*" but by numerical representation, since *RandomForestClassifier* only works with numerical data.

```
0-bailarinas
1-botas
2-botines
3-none
4-non_leather
5-plataformas__cunas
6-sandalias
7-sneakers
8-velcro
9-zapatillas
10-zapatos_casual
11-zapatos_casual_de_vestir
12-zapatos_de_cordones
13-zapatos_de_tacon
14-zapatos_de_vestir
15-zapatos_plan
```

4.5.3 Tree Interpreter

We can use the TreeInterpreter[5] Python library for interpreting *scikit-learn*'s decision tree and random forest predictions. It allows decomposing each prediction into bias and feature contribution components [35]. For a dataset with `n` features, each prediction on the dataset is decomposed as:

```
Prediction = bias + feature_1_contribution + ... + feature_n_contribution.
```

Let's take any row from the training set and use already trained Random Forest classifier to predict its label (class, category).

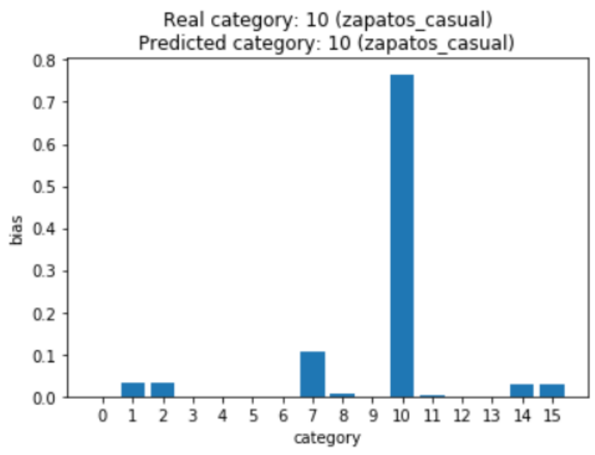


Figure 28: Correct classification of the object

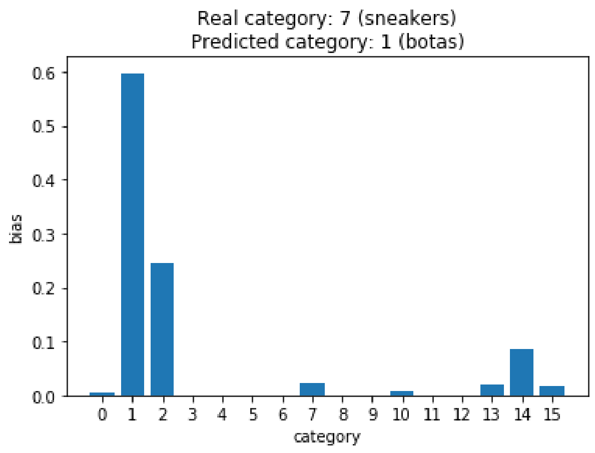


Figure 29: Incorrect classification of the object

In Figure 29: Incorrect classification of the object, the model strongly believes that object is class 1 and less likely class 2 or maybe even class 14. However, the model is class 7.

[5] https://github.com/andosa/treeinterpreter

Interpreting individual interpreters (outputs of individual decision trees) is quite easy, and no library is needed- first let each tree in ensemble classify a single, random, row. Then take those classifications and make a histogram.

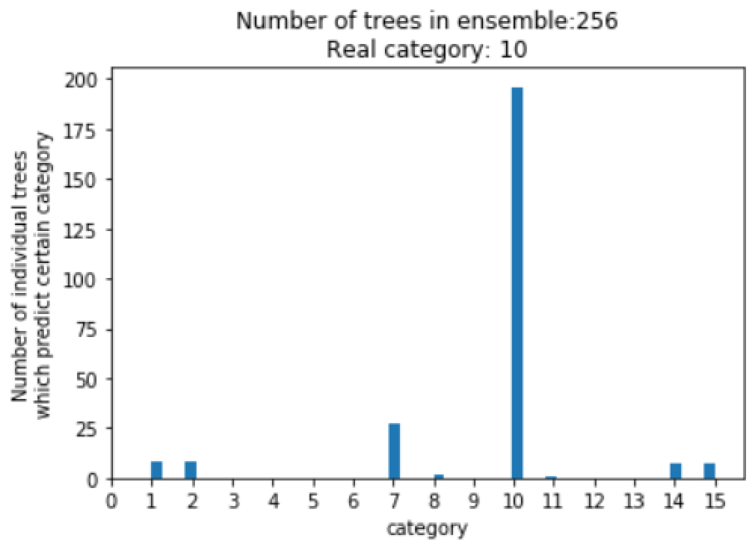


Figure 30: Authors implementation of tree interpreter

4.6 The conclusion of the chapter

In this chapter, the reader has been introduced to several important characteristics of Random Forest: hyperparameters, feature importance, feature similarity, and how to interpret Random Forest. Interpretation is made by showing several individual decision trees and their splitting points, and by using Partial Dependence Plot to visualize how values of one feature influence the classification outcome.

5 Results

By training *RandomForestClassifier* on the training set, the accuracy of correctly assigned categories to rows in the test set is 35-55%, with average being 45%. Considering how small dataset is and how most of the features are based on the shape of a shoe sole (mask), results are acceptable.

We can change a number of decision trees in Random Forest ensemble. By increasing them, computation times also increases, but accuracy score doesn't increase so much. For example, by having 1024 trees compared to having 100, accuracy is increased by a few percents, but the time it takes to train the model is drastically increased (around six times).

The author has retrained Random Forest model with the same optimal hyperparameters 100 times on different train, and test datasets and an only number of estimators (decision trees) were changed. Results for average, maximum and minimum accuracy are snown for four different models on Figure 31: Comparison of the average accuracy out of 100 tests, Figure 32: Comparison of the maximum accuracy out of 100 tests and Figure 33: Comparison of the minimum accuracy out of 100 tests. The average value is most important since it represents the best classification accuracy of the model. Overall model with 1024 trees is better, but even if the model with 100 trees is used results will only be slightly worse.

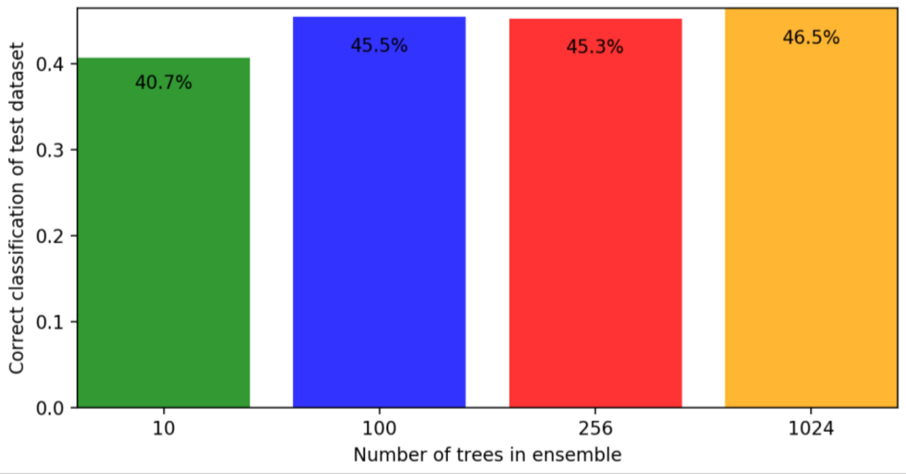


Figure 31: Comparison of the average accuracy out of 100 tests

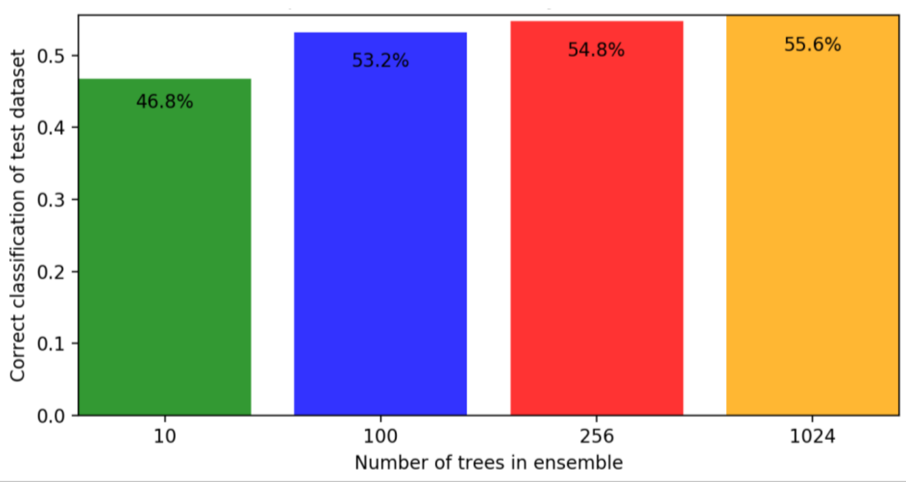


Figure 32: Comparison of the maximum accuracy out of 100 tests

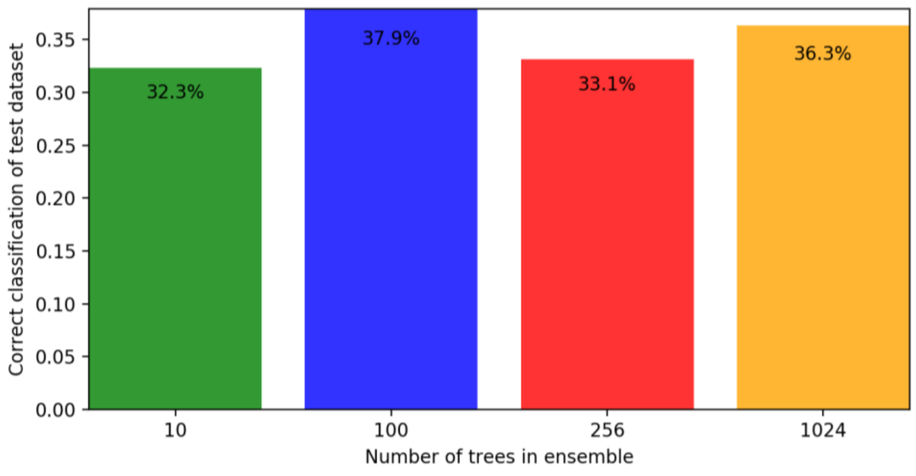
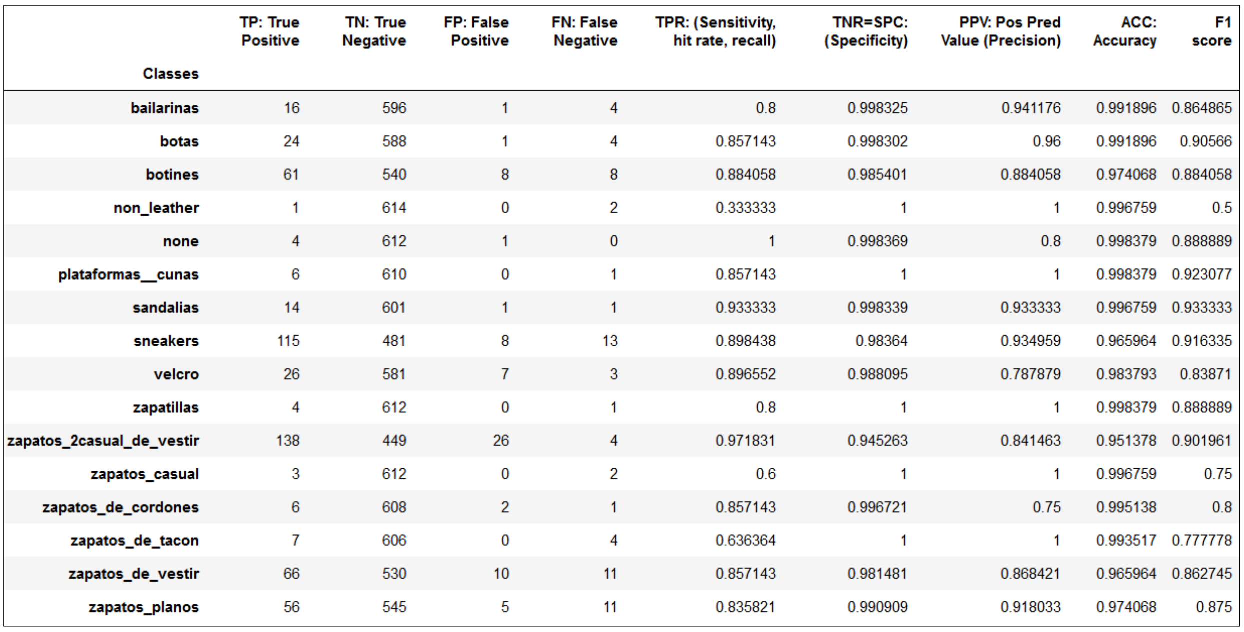


Figure 33: Comparison of the minimum accuracy out of 100 tests

5.1 Classification metrics

Once the predictive model has been created, it is essential to find out is how good it is.



Classes	TP: True Positive	TN: True Negative	FP: False Positive	FN: False Negative	TPR: (Sensitivity, hit rate, recall)	TNR=SPC: (Specificity)	PPV: Pos Pred Value (Precision)	ACC: Accuracy	F1 score
bailarinas	16	596	1	4	0.8	0.998325	0.941176	0.991896	0.864865
botas	24	588	1	4	0.857143	0.998302	0.96	0.991896	0.90566
botines	61	540	8	8	0.884058	0.985401	0.884058	0.974068	0.884058
non_leather	1	614	0	2	0.333333	1	1	0.996759	0.5
none	4	612	1	0	1	0.998369	0.8	0.998379	0.888889
plataformas__cunas	6	610	0	1	0.857143	1	1	0.998379	0.923077
sandalias	14	601	1	1	0.933333	0.998339	0.933333	0.996759	0.933333
sneakers	115	481	8	13	0.898438	0.98364	0.934959	0.965964	0.916335
velcro	26	581	7	3	0.896552	0.988095	0.787879	0.983793	0.83871
zapatillas	4	612	0	1	0.8	1	1	0.998379	0.888889
zapatos_2casual_de_vestir	138	449	26	4	0.971831	0.945263	0.841463	0.951378	0.901961
zapatos_casual	3	612	0	2	0.6	1	1	0.996759	0.75
zapatos_de_cordones	6	608	2	1	0.857143	0.996721	0.75	0.995138	0.8
zapatos_de_tacon	7	606	0	4	0.636364	1	1	0.993517	0.777778
zapatos_de_vestir	66	530	10	11	0.857143	0.981481	0.868421	0.965964	0.862745
zapatos_planos	56	545	5	11	0.835821	0.990909	0.918033	0.974068	0.875

Figure 34: True Positive, True Negative, False Positive, False Negative, Recall, Specificity, Precision, Accuracy, and F1 score

- **True Positives (TP)** - These are the correctly predicted positive values which mean that the value of the actual class is *x* and the value of the predicted class is also *x*, where *x* is the correct category of the shoe.
- **True Negatives (TN)** - These are the correctly predicted negative values which mean that the value of the actual class is *not x* and the value of the predicted class is also *not x*.
- **False Positives (FP)** – When the actual class is *not x* and predicted class is *x*.
- **False Negatives (FN)** – When the actual class is *x* but predicted class is *not x*.

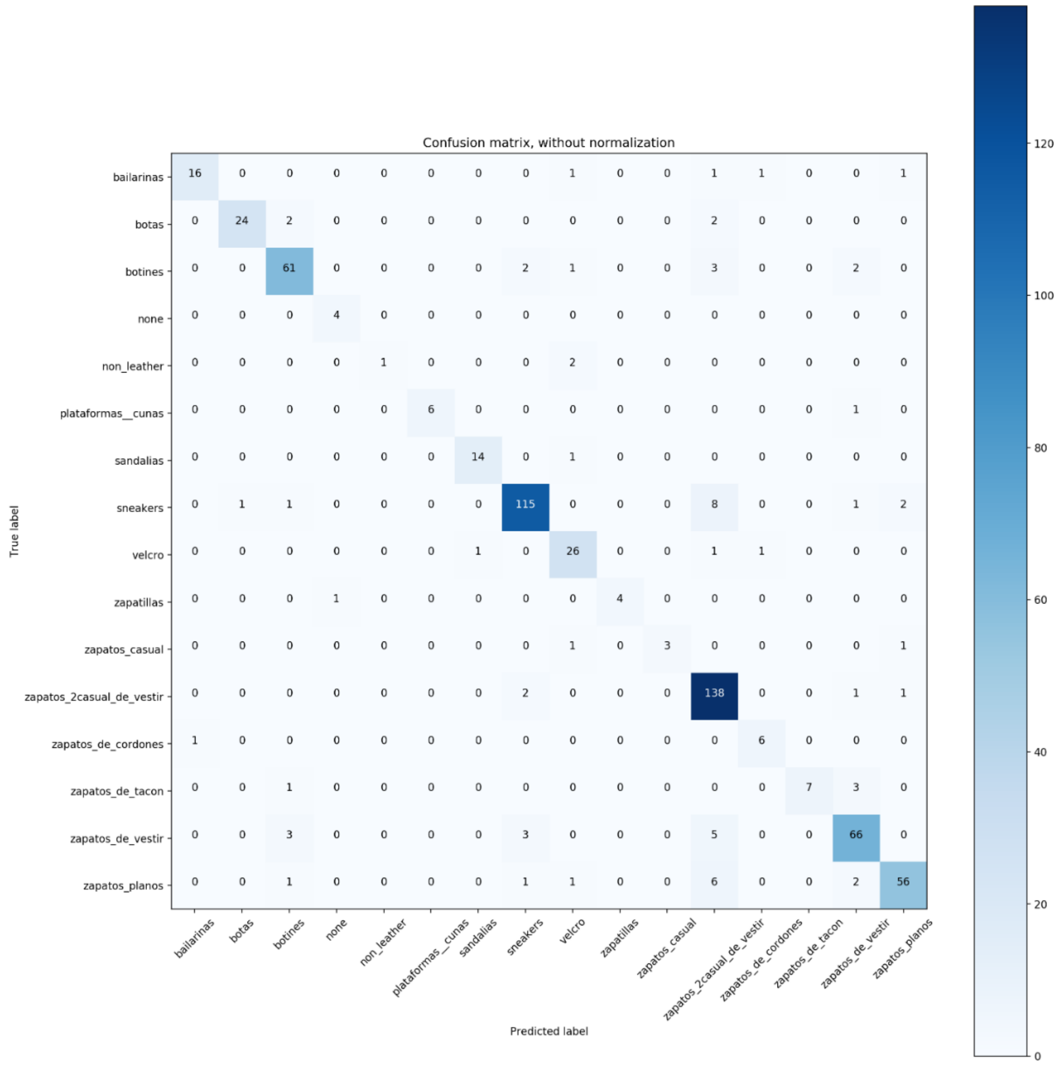
Explanation for the "*sneakers*" category: TP shows that 115 shoes are correctly classified as "*sneakers*", TN that 481 shoes are not "*sneakers*", FP that 8 shoes which are not "*sneakers*" are predicted to be "*sneakers*", and FN shows that 13 shoes which are indeed "*sneakers*" were predicted to be some other category.

- **Accuracy** is the overall correctness of the model and is calculated as the sum of correct classifications divided by the total number of classifications.
- **Precision** is a measure of the accuracy provided that a specific class has been predicted.
  - Precision = TP/(TP + FP)
- **Recall** is a measure of the ability of a prediction model to select instances of a certain class from a data set. It is commonly also called sensitivity and corresponds to the true positive rate.
  - Recall = Sensitivity = TP/(TP+FN)

   - TP + FN is the total number of test examples of the considered class.
- Recall/sensitivity is related to **specificity**, which is a measure that is commonly used in two class problems where one is more interested in a particular class. Specificity corresponds to the true-negative rate.
   - Specificity = TN/(TN+FP)
- **F1 Score** is the weighted average of Precision and Recall. F1 is usually more useful than accuracy, especially if you have an uneven class distribution. Accuracy works best if false positives and false negatives have a similar cost. If the cost of false positives and false negatives are very different, it is better to look at both Precision and Recall.
   - F1 Score = 2*(Recall * Precision) / (Recall + Precision)

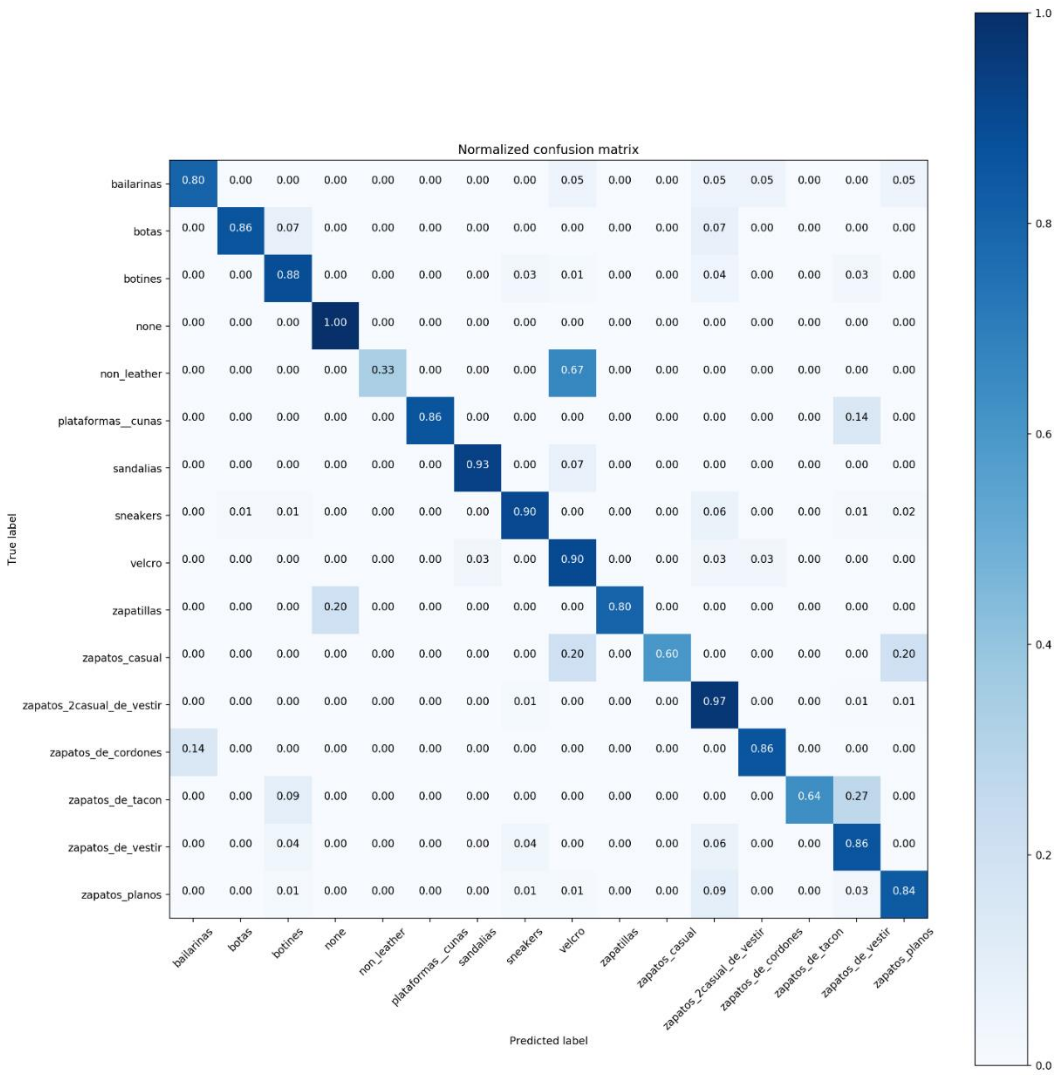
5.2 The Confusion Matrix

When referring to the performance of a classification model, we are interested in the model's ability to correctly predict the classes. When looking at the errors made by a classification model, the confusion matrix gives the full picture (Figure 35 and Figure 36).



	bailarinas	botas	botines	none	non_leather	plataformas__cunas	sandalias	sneakers	velcro	zapatillas	zapatos_casual	zapatos_2casual_de_vestir	zapatos_de_cordones	zapatos_de_tacon	zapatos_de_vestir	zapatos_planos
bailarinas	16	0	0	0	0	0	0	0	1	0	0	1	1	0	0	1
botas	0	24	2	0	0	0	0	0	0	0	0	2	0	0	0	0
botines	0	0	61	0	0	0	0	2	1	0	0	3	0	0	2	0
none	0	0	0	4	0	0	0	0	0	0	0	0	0	0	0	0
non_leather	0	0	0	0	1	0	0	0	2	0	0	0	0	0	0	0
plataformas__cunas	0	0	0	0	0	6	0	0	0	0	0	0	0	0	1	0
sandalias	0	0	0	0	0	0	14	0	1	0	0	0	0	0	0	0
sneakers	0	1	1	0	0	0	0	115	0	0	0	8	0	0	1	2
velcro	0	0	0	0	0	0	1	0	26	0	0	1	1	0	0	0
zapatillas	0	0	0	1	0	0	0	0	0	4	0	0	0	0	0	0
zapatos_casual	0	0	0	0	0	0	0	0	1	0	3	0	0	0	0	1
zapatos_2casual_de_vestir	0	0	0	0	0	0	0	2	0	0	0	138	0	0	1	1
zapatos_de_cordones	1	0	0	0	0	0	0	0	0	0	0	0	6	0	0	0
zapatos_de_tacon	0	0	1	0	0	0	0	0	0	0	0	0	0	7	3	0
zapatos_de_vestir	0	0	3	0	0	0	0	3	0	0	0	5	0	0	66	0
zapatos_planos	0	0	1	0	0	0	0	1	1	0	0	6	0	0	2	56

Figure 35: Confusion matrix (without normalization)



	bailarinas	botas	botines	none	non_leather	plataformas__cunas	sandalias	sneakers	velcro	zapatillas	zapatos_casual	zapatos_2casual_de_vestir	zapatos_de_cordones	zapatos_de_tacon	zapatos_de_vestir	zapatos_planos
bailarinas	0.80	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.05	0.00	0.00	0.05	0.05	0.00	0.00	0.05
botas	0.00	0.86	0.07	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.07	0.00	0.00	0.00	0.00
botines	0.00	0.00	0.88	0.00	0.00	0.00	0.00	0.03	0.01	0.00	0.00	0.04	0.00	0.00	0.03	0.00
none	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
non_leather	0.00	0.00	0.00	0.00	0.33	0.00	0.00	0.00	0.67	0.00	0.00	0.00	0.00	0.00	0.00	0.00
plataformas__cunas	0.00	0.00	0.00	0.00	0.00	0.86	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.14	0.00
sandalias	0.00	0.00	0.00	0.00	0.00	0.00	0.93	0.00	0.07	0.00	0.00	0.00	0.00	0.00	0.00	0.00
sneakers	0.00	0.01	0.01	0.00	0.00	0.00	0.00	0.90	0.00	0.00	0.00	0.06	0.00	0.00	0.01	0.02
velcro	0.00	0.00	0.00	0.00	0.00	0.00	0.03	0.00	0.90	0.00	0.00	0.03	0.03	0.00	0.00	0.00
zapatillas	0.00	0.00	0.00	0.20	0.00	0.00	0.00	0.00	0.00	0.80	0.00	0.00	0.00	0.00	0.00	0.00
zapatos_casual	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.20	0.00	0.60	0.00	0.00	0.00	0.00	0.20
zapatos_2casual_de_vestir	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.97	0.00	0.00	0.01	0.01
zapatos_de_cordones	0.14	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.86	0.00	0.00	0.00
zapatos_de_tacon	0.00	0.00	0.09	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.64	0.27	0.00
zapatos_de_vestir	0.00	0.00	0.04	0.00	0.00	0.00	0.00	0.04	0.00	0.00	0.00	0.06	0.00	0.00	0.86	0.00
zapatos_planos	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.01	0.01	0.00	0.00	0.09	0.00	0.00	0.03	0.84

Figure 36: Normalized confusion matrix

The confusion matrix shows how the model makes the predictions. The rows correspond to the known class of the data, i.e. the labels in the data. The columns correspond to the predictions made by the model. The value of each element in the matrix is the number of predictions made with the class corresponding to the column for examples with the correct value as represented by the row. Thus, the diagonal elements show the number of correct classifications made for each class, and the off-diagonal elements show the errors made.

Normalization means that numbers are scaled to the total number of elements in each class. Explanation: there are 128 shoes in "*sneakers"* category and Figure 35: Confusion matrix (without normalization) shows that 115 shoes are "*sneakers*" and they are indeed in that category. Figure 36: Normalized confusion matrix shows 0.90 for the same "*sneakers*" category because 115/128=0.90, meaning that 90% of the "*sneakers*" were correctly classified as "*sneakers*".

5.3 Changing the number of categories

Since each category has a different number of shoes, we can see how the accuracy of *RandomForestClassifier* changes when dataset only has a limited number of categories: one, two, three and so forth.

```
Dataset has following categories: ['zapatos_casual']
Dataset shape: (142, 56)
Accuracy: {'min': 1.0, 'avg': 1.0, 'max': 1.0}
--------------------------------
Dataset has following categories: ['zapatos_casual', 'sneakers']
Dataset shape: (270, 56)
Accuracy: {'min': 0.685, 'avg': 0.804, 'max': 0.889}
--------------------------------
Dataset    has    following    categories:    ['zapatos_casual',    'sneakers',
'zapatos_de_vestir']
Dataset shape: (347, 56)
Accuracy: {'min': 0.557, 'avg': 0.671, 'max': 0.743}
...
```

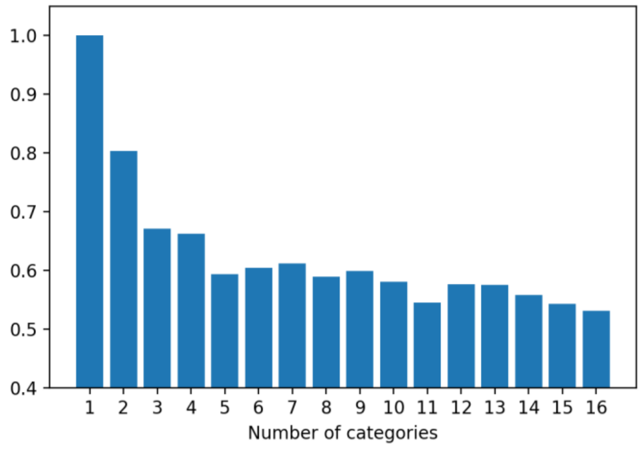


Figure 37: Accuracy score compared to the number of categories in training dataset

In Figure 37: Accuracy score compared to the number of categories in training datasetwhen the dataset has only one category of shoes, "*zapatos_casual*" accuracy score on the training set is 100% because there is nothing to predict and depended variable always has the same value. However, when we only have to guess if a shoe is of category "*zapatos_casual*" or "*sneakers*", where each category has 142 and 128 shoes in its category, respectively, then accuracy score increases to 80%. Quite an increase compared to previously mentioned 45% on the final model and 16 categories. When additional categories are added, accuracy score drops, as it is visible on the graph above.

5.4 Binary classification

Figure 38: Binary classification represents the results of binary classifications- if we are trying only to predict if shoes belong to a single category. Training dataset has all previously mentioned features, except *bigcategory* is removed and replaced with *are_zapatos_casual* in the first test, then with *are_sneakers* in the second test, and so forth. There are 142 shoes in

zapatos_casual category, an 475 which are not *zapatos_casual* and that is enough for a good classification accuracy- almost 80% percent.

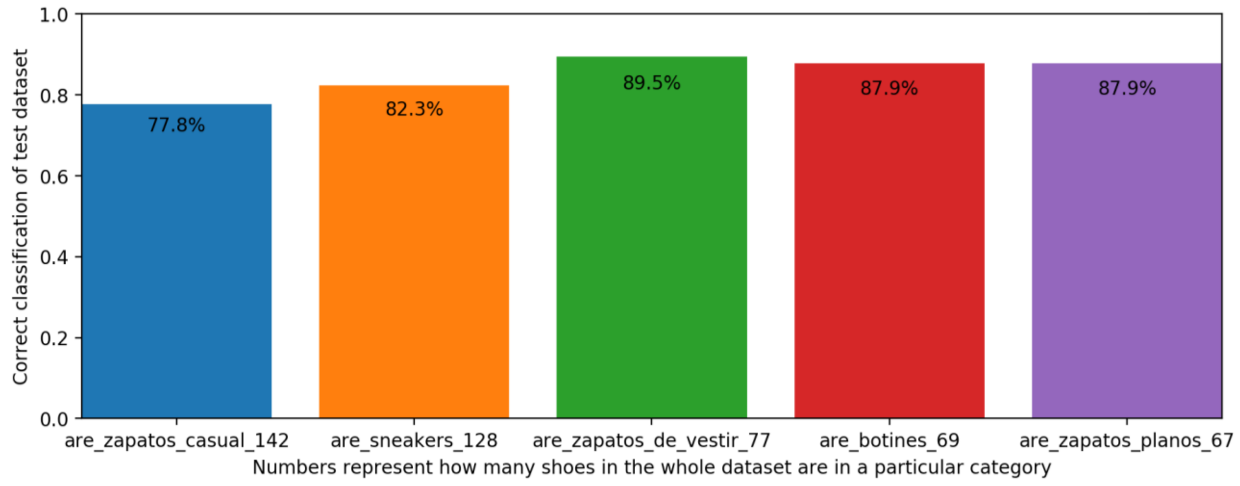


Figure 38: Binary classification

Figure 38: shows the accuracy score for five different datasets. Datasets only have one class. The dependent variable is a column representing if a shoe is or isn't the class. Dataset doesn't have "*bigcategory*" feature but has, for example, "*are_sneakers*" which has two values: 1 (shoes are in "*sneakers*" category) or 0 (shoes are in that category).

5.5 Saving shoe predictor

Once training of random forest is complete, it is easy to save trained model to permanent storage. Training on shoe dataset is quick (a couple of seconds), but it is useful in scenarios when we only want to load to model and use it in production quickly. Once a computer is shut down, any code which was running, but not explicitly saved, is lost. Then you have to repeat all the steps in order to get to the same model. Keep in mind that training on a dataset which has millions of rows can take several minutes, or even hours to train, depending on selection hyperparameters. Python library *pickle* allows to load and save models quickly. Loading and saving take a couple of seconds.

6 Discussion and conclusion

Random Forest is a powerful and robust machine learning algorithm which has many applications in both classification and regression tasks. It is an excellent choice for almost any machine learning problem because it can handle different kinds of data (numerical, categorical, datetime, etc.); it makes implicit feature selection and provides a pretty good indicator of feature importance; can be tuned and trained very quickly; is free and straightforward; easy to run in parallel. Because it is difficult to create "bad" RF predictor they are often used as a benchmark for other ML models. The negative side of the random forest is that it is difficult to analyze because it is considered as "black-box", although single trees can be visualized [36].

Current RF model correctly classifies 45% of the training datasets. Even though product classification is a problematic area, those results are acceptable because of significant number of categories (15) and a disproportional number of shoes in each category. When the number of categories is lowered, the model can correctly predict up to 89.5% of the training datasets for binary classification, and around 60% when datasets have 3 or 4 categories. This indicates that if all categories had a sufficient number of images, overall results could be improved.

Correct prediction of shoe category can probably be improved by adding more features to the dataset. If current solution is used in real world, it would be highly depended on lighting conditions and background because the user would have to replicate the image of the top view of two images on www.camper.com website in order to get current accuracy.

Another crucial component is the process of extracting shoe image against the background. Current algorithm for shoe separation works perfectly on a white background but would have to be redone if the goal is to use it against a background in real-world-texturized, non-white background with non-uniform lightning.

All objectives mentioned in the chapter "*1.4 Objectives*" were successfully accomplished: scraping data from the website and parsing XML, create image dataset extracting features from images, using Computer Vision algorithms to edit images, and using Random Forest to predict categories of the shoes.

For future work, implementation of extremely accurate and robust classifier would require 3D scans of each shoe so that the massive amount of labeled images can be used by some deep learning algorithm for training and testing. It would improve results because manual extraction of features would not be needed. Expected results of deep leaning classifiers are in the range of 60 to 95% if there is large enough image dataset (as mentioned in the chapter "*1.2 Previous solutions*").

7 References

[1] J. Edvinsson, "Machine Learning: Handbag Brand and Color Detection using Deep Neural Networks. | Condé Nast Technology," 6 November 2017. [Online]. Available: https://technology.condenast.com/story/handbag-brand-and-color-detection. [Accessed 30 January 2019].

[2] "Startup GOAT uses AI to Verify Shoe Authenticity," 31 August 2018. [Online]. Available: https://news.developer.nvidia.com/startup-goat-uses-ai-to-verify-shoe-authenticity/. [Accessed 30 January 2019].

[3] ELSE Corp, "ELSE.ai, a project by ELSE Corp," [Online]. Available: http://www.else.ai/. [Accessed 30 January 2019].

[4] "Happy Finish | Shoegazer AI Prototype Application," [Online]. Available: https://www.happyfinish.com/work/shoegazer-artifical-intelligence/. [Accessed 30 January 2019].

[5] "How to create a product recognition solution | deepsense.ai," 22 August 2017. [Online]. Available: https://deepsense.ai/how-to-create-a-product-recognition-solution/.

[6] P. Sayer, "Google buys sneaker-scanning machine learning company Moodstocks," 6 July 2016. [Online]. Available: https://www.pcworld.com/article/3091881/google-buys-sneaker-scanning-machine-learning-company-moodstocks.html. [Accessed 30 January 2019].

[7] W. contributors, "Artificial intelligence," Wikipedia, The Free Encyclopedia., 14 January 2019. [Online]. Available: https://en.wikipedia.org/wiki/Artificial_intelligence. [Accessed 30 January 2019].

[8] J. McCarthy, M. L. Minsky, N. Rochester and C. E. Shannon, "A Proposal For The Dartmouth Summer Research Project On...," 31 August 1995. [Online]. Available: http://www-formal.stanford.edu/jmc/history/dartmouth/dartmouth.html. [Accessed 30 January 2019].

[9] Audio Software Engineering team and Siri Speech Team, "Optimizing Siri on HomePod in Far-Field Settings," Apple, December 2018. [Online]. Available: https://machinelearning.apple.com/2018/12/03/optimizing-siri-on-homepod-in-far-field-settings.html. [Accessed 29 January 2019].

[10] S. Hawking, "The Basic Concepts of Machine Learning - Give it a Spin," 2 July 2018. [Online]. Available: https://giveitaspin.net/2018/07/02/basics-of-machine-learning/. [Accessed 30 January 2019].

[11] B. Allison, "Machine learning - Is there a rule-of-thumb for how to divide a dataset into training and validation sets? - Stack Overflow," Stack Exchange Inc, 29 November 2012. [Online]. Available: https://stackoverflow.com/a/13623707. [Accessed 29 January 2019].

[12] W. contributors, "Bias–variance tradeoff," Wikipedia, The Free Encyclopedia., 30 December 2018. [Online]. Available: https://en.wikipedia.org/wiki/Bias%E2%80%93variance_tradeoff. [Accessed 30 January 2019].

[13] "Technology Services - Graviton - T Solutions," [Online]. Available: http://www.gravitonsolutions.com/technology-services/. [Accessed 30 January 2019].

[14] "reCAPTCHA | Google Developers," Google, [Online]. Available: https://developers.google.com/recaptcha/. [Accessed 30 January 2019].

[15] W. contributors, "Tree (data structure)," Wikipedia, The Free Encyclopedia, 8 January 2019. [Online]. Available: https://en.wikipedia.org/wiki/Tree_(data_structure). [Accessed 30 January 2019].

[16] W. contributors, "Decision tree learning," Wikipedia, The Free Encyclopedia., 10 January 2019. [Online]. Available: https://en.wikipedia.org/wiki/Decision_tree_learning#Gini_impurity. [Accessed 29 January 2019].

[17] G. Tam, "Interpreting Decision Trees and Random Forests - Pivotal Engineering Journal," 19 September 2017. [Online]. Available:

http://engineering.pivotal.io/post/interpreting-decision-trees-and-random-forests/. [Accessed 30 January 2019].

[18] W. contributors, "Boosting," Wikipedia, The Free Encyclopedia., 9 November 2018. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Boosting_(machine_learning). [Accessed 30 January 2019].

[19] scikit-learn developers, "1.11. Ensemble methods - scikit-learn 0.20.2 documentation," [Online]. Available: https://scikit-learn.org/stable/modules/ensemble.html. [Accessed 30 January 2019].

[20] W. contributors, "Bootstrap aggregating," Wikipedia, The Free Encyclopedia., 5 November 2018. [Online]. Available: https://en.wikipedia.org/wiki/Bootstrap_aggregating. [Accessed 30 January 2019].

[21] R. Bellman, Dynamic programming, Princeton Univ Pr, 1957.

[22] G. Shklovsky, "Gabby Shklovsky - Random Forests Best Practices for the Business World," PyData, 21 December 2017. [Online]. Available: https://www.youtube.com/watch?v=E7VLE-U07x0. [Accessed 30 January 2019].

[23] M. Fernández-Delgado, M. Fernández-Delgado, M. Fernández-Delgado and M. Fernández-Delgado, "Do we Need Hundreds of Classifiers to Solve Real World Classification Problems?," *The Journal of Machine Learning Research,* vol. Volume 15, no. Issue 1, pp. 3133-3181, 2014.

[24] W. contributors, "Occam's razor," Wikipedia, The Free Encyclopedia., 22 January 2019. [Online]. Available: https://en.wikipedia.org/wiki/Occam's_razor. [Accessed 30 January 2019].

[25] W. contributors, "Computer vision," Wikipedia, The Free Encyclopedia., 21 December 2018. [Online]. Available: https://en.wikipedia.org/wiki/Computer_vision. [Accessed 30 January 2019].

[26] "scikit-learn: machine learning in Python - scikit-learn 0.20.2 documentation," [Online]. Available: https://scikit-learn.org/. [Accessed 30 January 2019].

[27] "OpenCV: Histograms - 2: Histogram Equalization," 18 December 2015. [Online]. Available: https://docs.opencv.org/3.1.0/d5/daf/tutorial_py_histogram_equalization.html. [Accessed 29 January 2019].

[28] W. contributors, "HSL and HSV," Wikipedia, The Free Encyclopedia., 14 January 2019. [Online]. Available: https://en.wikipedia.org/wiki/HSL_and_HSV. [Accessed 30 January 2019].

[29] The scikits-image team, "Module: feature.texture - skimage v0.7.0 docs," [Online]. Available: http://scikit-image.org/docs/0.7.0/api/skimage.feature.texture.html. [Accessed 29 January 2019].

[30] W. contributors, "Hyperparameter (machine learning)," Wikipedia, The Free Encyclopedia., 20 June 2018. [Online]. Available: https://en.wikipedia.org/wiki/Hyperparameter_(machine_learning). [Accessed 30 January 2019].

[31] H. Suenaga, "Machine Learning 1: Lesson 4," Medium, 9 August 2018. [Online]. Available: https://medium.com/@hiromi_suenaga/machine-learning-1-lesson-4-a536f333b20d. [Accessed 30 January 2019].

[32] P. Grover, "Intuitive Interpretation of Random Forest," Medium, 25 November 2017. [Online]. Available: https://medium.com/usf-msds/intuitive-interpretation-of-random-forest-2238687cae45. [Accessed 30 January 2019].

[33] "scipy.cluster.hierarchy.dendrogram - SciPy v1.2.0 Reference Guide," The SciPy community, 17 December 2018. [Online]. Available: https://docs.scipy.org/doc/scipy/reference/generated/scipy.cluster.hierarchy.dendrogram.html. [Accessed 27 January 2019].

[34] J. H. Friedman, "Greedy function approximation: a gradient boosting machine," *Annals of statistics,* pp. 1189--1232, 2001.

[35] "Interpreting random forests | Diving into data," 14 October 2014. [Online]. Available: http://blog.datadive.net/interpreting-random-forests/. [Accessed 30 January 2019].

[36] A. E. Deeb, "The Unreasonable Effectiveness of Random Forests – Rants on Machine Learning – Medium," Medium, 17 Jul 2015. [Online]. Available: https://medium.com/rants-on-machine-learning/the-unreasonable-effectiveness-of-random-forests-f33c3ce28883. [Accessed 30 January 2019].

[37] W. contributors, "Overfitting," Wikipedia, The Free Encyclopedia., 18 January 2019. [Online]. Available: https://en.wikipedia.org/wiki/Overfitting. [Accessed 29 January 2019].