



University of Essex

School of Mathematics, Statistics
and Actuarial Science

MA981 DISSERTATION

A STUDY ON THE VISUAL ELEMENTS
OF BOOK COVERS USING MACHINE
LEARNING METHODS

Reethu Mathew (Student ID: 2213186)

Supervisor: **Dr. Xinan Yang**

November 24, 2023
Colchester

Abstract

This research focuses on the relationship between the visual elements of book covers and book categories. Using several machine learning methods, the colour, text, and object elements of the book covers are examined with respect to their category. The book cover images are subjected to feature extraction methods to gather the required data elements from the book cover images in machine readable format. The extracted data is then used to train different machine learning and neural network models which are then compared to uncover the best model for each visual element. The results obtained indicated that the tested machine learning models alone are not sufficient for the category-based prediction of visual elements. The regression models used were not suitable for the categorical prediction of the numerical data and the classification models were excessively affected by the biased data. In order to build an artificial intelligence model that is capable of assisting designers in creative decision making processes, datasets of higher quality need to be studied with far more advanced models and superior resources.

Contents

1	Introduction	7
2	Related Work	10
3	Dataset	15
3.1	Dataset Description	15
3.2	Data Preprocessing	16
3.2.1	Dataset Encoding	16
3.2.2	Column Headers	16
3.2.3	Null values	18
3.2.4	Duplicates	18
3.2.5	Cleaning the data	18
3.2.6	Stopwords Removal	19
3.2.7	Type conversion	19
3.2.8	Label Encoding	20
3.2.9	One-Hot Encoding	20
3.2.10	Tokenizer	21
3.2.11	TfidfVectorizer	21
3.2.12	Train-Test Split	22
3.3	Feature Extraction	23
3.3.1	Colour Extraction using K-Means Clustering	23
3.3.2	Text Area Calculation with EasyOCR	24
3.3.3	Object Recognition with Google Vision API	26
4	Methodology	27
4.1	Regression Models	27

4.1.1	Linear Regression	27
4.1.2	Decision Tree Regression	28
4.1.3	Random Forest Regression	29
4.1.4	Support Vector Regression	30
4.2	Classification Models	32
4.2.1	Logistic Regression	33
4.2.2	Decision Tree Classification	33
4.2.3	Random Forest Classification	34
4.2.4	Support Vector Machine	35
4.2.5	Multinomial Naive Bayes Model	36
4.3	Neural Networks	38
4.3.1	MLP Classifier	39
4.3.2	Tensorflow Sequential	40
4.4	Experimental Setup	40
4.4.1	Category Prediction from Book Titles	41
4.4.2	Colour Prediction from Categories	43
4.4.3	Text Area Prediction from Categories	44
4.4.4	Object Prediction from Categories	45
4.5	Evaluation Metrics	46
4.5.1	Classification Metrics	46
4.5.2	Regression Metrics	48
5	Results and Discussion	49
6	Conclusion	57
A	Code	58
B	Additional Figures	97
C	Additional Tables	99

List of Figures

3.1	First five rows of the original dataset	15
3.2	An Example of One-Hot Encoding	20
3.3	EasyOCR Framework	25
4.1	Linear Regression	28
4.2	Decision Tree Regression	29
4.3	Overfitting in Decision Tree Regression Model	30
4.4	Random Forest Regression	31
4.5	Support Vector Regression	32
4.6	Comparison between Linear and Logistic Regression	34
4.7	Decision Tree Classification	35
4.8	Random Forest Classification	36
4.9	Support Vector Machine Classification	37
4.10	Neural Network Structure	39
4.11	Confusion Matrix	47
5.1	Category Prediction from Sample Book Title	50
5.2	RGB Colours predicted by the selected model for two categories	51
5.3	HSL Colours predicted by the selected model for two categories	52
5.4	Distribution of Colour Labels for Children's Books Category	55
A.1	Colour 1 Prediction	72
B.1	Classification Report - Logistic Regression	97
B.2	Confusion Matrix - Logistic Regression	97
B.3	Distribution of Colour Labels for Romance Category	98

List of Tables

3.1	Book categories in the dataset	17
4.1	Parameters of TfidfVectorizer	41
4.2	Parameter of MLPClassifier	42
4.3	Criteria for colour labelling	44
5.1	Evaluation Metrics of models for Category Prediction from Book Titles .	49
5.2	Evaluation Metrics for RGB Regression Models	50
5.3	Evaluation Metrics for HSL Regression Models	50
5.4	Evaluation Metrics for Colour Label Classification Models	51
5.5	Evaluation Metrics for Percentage Text Area Prediction	52
5.6	Categories with Highest Percentage of Text Area Predicted	53
5.7	Object Predictions For All Categories	53
C.1	RGB Colour Generated For All Categories	99
C.2	HSL Values Generated For All Categories	105
C.3	Predicted Colour Labels For All Categories	108
C.4	Predicted Text Area Percentage For All Categories	110
C.5	Classification Report - Object Classification Model	111

Introduction

The saying "Don't judge a book by its cover" means that one should not judge something or someone by their appearance alone [26]. But when it comes to the actual books, the cover is one of the key factors that grabs the attention of the reader. Especially for a casual reader who is not familiar with the author of the book or the general category to which the book belonged, the cover of the book is the first interaction, and it creates an impression on the reader [56]. Even with the narrow likelihood of an in-person book buyer actually referring to the contents of the book to determine whether they hold true to the hints provided by the cover [83], a luxury that rarely comes by for online book buyers, the book cover is often the only thing that decides whether the book is purchased [30]. While the synopsis on the back cover helps the buyer decide if the book is interesting enough for them to spend money on, the front cover's job is to make sure the book gets the attention of the readers while they are pondering their choices in a library or bookshop. In an experimental study on the factors influencing the reader's interest in new book releases conducted by d'Astous et al., there are four key factors suggested based on the semi-structured interviews conducted as part of their preliminary studies: how famous the author is, how established the publisher is, how attractive the book cover is, and how true the book cover is to the actual content of the book. The results of the study concluded that the first three components put forward in the study were most significant in terms of the impact they made on the reader, and especially for a new release, an attractive book cover piques the readers' interest [22].

There has been a substantial growth in the number of books being released to the

public. Publishers are in constant need of being aware of trends, genre expectations, cultural presses, reader preferences, and the circulation of the book in the community [30]. With the competition growing between the publishers [44] and digital publishing becoming more popular because of its accessibility and convenience, looking into what exactly consumers find appealing to pick up the book or click on the link is crucial [132] for the publishers. Readers are exposed to an enormous number of books belonging to several different genres each year [22]. Piters et al. hypothesised that when a book is identified by its cover, the typicality of a specific genre affects the preference for the book. So it is important to study the relationship of a book cover with its genre and how it affects the readers' preferences [89], which will help the publishers design a successful cover that is tailored towards the readers' liking in that particular genre, aiding in better sales [22]. The more a cover identifies attributes of a particular genre, the more the book will be considered of that specific genre [89].

Every element on a book cover, whether it's a physical or digital book, is the culmination of artistic and marketing choices that focus on the potential of a book to be noticed, picked up, purchased, [30] liked, and recommended among the community for the benefit of the author and the publisher. The creative decisions taken in the course of the book cover design must be focused on getting the book to stand out in the marketplace and generate curiosity, interest, and engagement in potential readers [30]. Besides key information such as the title of the book, the name of the author, and the publisher, the visual aesthetics and additional components such as blurbs, straplines, and illustrations add to the appeal to the reader and should not be overlooked or prioritised during design [30]. Through a classification of book covers according to their genre, Uchida et al. identified three core design areas concerning book covers: colour, objects, and text. Images being the first thing that catches the eye of potential readers, the context and layout of objects, along with the typography, play a big role in associating a book with its content or genre. In the absence of pictorial representation, they found that dominant colours alone helped categorise a book by its genre [56]. While working on a project, designers are up-to-date with the current trends in the industry to keep up with the social demands and preferences of the audiences while keeping in mind the guidelines that have proven to be successful [30]. Considering the growth of technology and the rise of digital publishing, designers will also have to adapt their designs to suit

electronic book covers, which can be quite different from physical copies [132]. The goal of this study is to extract information about the visual aspects of a book cover and help the design process better suit the genre.

Related Work

Machine learning and artificial intelligence can support the design process by providing suggestions for layout options, generating new ideas, and automating repetitive workloads using data and algorithms [133]. Machine learning has the potential to support designers in the following three processes: insight, prototyping, and evaluation. It can help in identifying and analysing handwritten annotations, recognising patterns unnoticed by the human eye, and providing suggestions for colours and layouts [126]. Artificial intelligence enables organisations to overcome the limitations of human-intensive designs by improving the scalability of the process, widening its scope, and enhancing its ability to learn and adapt [133], thus improving creativity and innovation. The advances in technology and the widespread use of the internet have generated a huge amount of picture data that has overwhelmed the traditional analytical resources, which mostly involved human evaluation. To counter this massive inflow of image data, deep learning is gradually being applied to deal with two-dimensional data to automatically extract its features and the corresponding information. Even though deep learning models need more training and computing resources, they can achieve better accuracy [67].

Machine learning approaches are taking over the field of medicine and are becoming progressively successful in image-based diagnosis, disease forecasting, and risk assessment [75]. Clinical medicine has always required medical practitioners to handle huge amounts of data in the form of clinical records, examination notes, and medical images [127]. Radiologists are tasked with the differential diagnosis of each patient's

medical images, determining the presence or absence of an entity to identify the type of malignancy [65]. Introducing computer-assisted interventions to assist with interpreting this large volume of data enabled precise, real-time predictions of diseases, faster adoption of necessary preventive and therapeutic solutions, and automatic execution of time-consuming complex tasks. The NiftyNet infrastructure, a modular deep learning pipeline that provides a range of medical imaging applications including image segmentation, image regression, image generation, and image model representation, plays a critical role at several stages in a clinical workflow, from population screening and diagnosis to treatment delivery and monitoring [41].

One of the most recent and relevant implementations of deep learning in medicine was during the COVID-19 pandemic, which continues to spread and present unprecedented challenges in many health systems globally [78]. Accurate and timely diagnosis and effective prognosis of the disease are important to provide the best care for patients with COVID-19 and reduce the burden on the health care system [82]. The main challenge when using deep learning for medical imaging is the scarcity of correctly labelled data available for training and testing. But with the help of transfer learning by fine-tuning and feature extraction from the available data, better models with high accuracy and avoid overfitting are developed [47]. Deep learning-based approaches such as deep feature extraction, fine-tuned pretrained convolutional neural networks, and end-to-end training of developed models have been used to classify COVID-19-infected and normal healthy chest X-ray images, while keeping an eye out for other lung conditions as well [55]. A non-medical application of deep learning during the COVID-19 pandemic was used to detect the use of face masks. The continuous upsurge of the infection and emerging variants caused many regulatory authorities to institute the mandatory use of face masks, especially in public places [78]. Mbunge et al. conducted a study of face mask-detecting mechanisms that were adopted using artificial intelligence models and the shortcomings of such models due to the lack of real-world datasets.

As an important method in the field of artificial intelligence, machine learning has been widely used for traffic identification research in recent years. Vehicle detection is an important part of the effective operation of the traffic monitoring system and has greatly promoted transportation modernization. Image recognition technologies reinforced by machine learning and artificial intelligence are used to correctly segment

the vehicle and obtain the target area, as well as aid with license plate recognition with high accuracy [70]. Another application of image recognition in road networks is urban road monitoring using surveillance cameras, in-vehicle cameras, or smartphones to recognise the specific types of road damages in order to plan maintenance resources efficiently [6]. Extracting road features from remote sensing images is one of the most challenging research subjects in remote sensing, as it influences multiple scenes such as map updating, traffic management, road monitoring, and emergency tasks [2]. Wang et al. proposed a neural-dynamic tracking framework to extract road networks from very-high-resolution aerial and satellite imagery based on deep convolutional neural networks and a finite state machine [59]. While the approach was able to work efficiently when the roads were clearly distributed on the images, it struggled to perform well on more complex cases of aerial images where it was difficult to tell the roads and buildings apart.

A study on recognising and detecting cassava diseases even in low-quality images using a trained neural network for image classification used a data augmentation procedure to train a network on artificially generated images with image colour space modified using the convolution of the probability density functions of colour histograms with the Chebyshev orthogonal functions [1]. While these results of the study reinforce the fact that using high-quality images for classification will produce better prediction results, they also shine a light on the vital need to consider low-quality images and their easy integration into mobile applications deployed on lower-end smartphones. thereby focusing on the target users, such as rural farmers. Developing and improving AI-based smart agricultural applications is advantageous for the early detection of plant diseases and boosting crop yields. Another focus point was the urgency to develop new benchmarks for testing and comparing the efficiency of neural network models on imperfect or corrupted test data.

The satellite data captured from long distances for its various applications in agriculture, defence, navigation, etc. is affected by the presence of unwanted noises, which affect the quality of the image and also affect future analyses of the images [9]. Remote sensing data collected by researchers and governments to analyse the earth's changes around the world is useful in climate change estimation, environmental monitoring, disaster management, and land development. Gathering accurate information about

landslides as soon as possible is a prime focus area in remote sensing for disaster prevention and relief. Due to the suddenness of landslide occurrences, image classification algorithms deployed on the remote sensing images extract information about landslides using the reflectivity differences of different surfaces [73]. A deep learning network (MSLWENet) used for lake water body extraction from Google remote sensing images is mainly used for spatial geographic analysis that plays an important role in the prevention of natural disasters, resource utilisation, and water quality monitoring [136]. The automatic target recognition (ATR) approach suggested for the sonar on-board unmanned underwater vehicles extracts the target features using a convolutional neural network operating on sonar images and then classifies them using a support vector machine that is trained on manually labelled data [140]. ATR is trained to extract information from low-quality images, as is expected of underwater sonar images, and removes the manual classification, which is slow, costly, and inefficient.

Another application of deploying image extraction and classification techniques is the proposal for semi-automatic processing of images using convolutional neural networks from the unmanned aerial vehicles (UAV) used for animal censuses in wildlife [138]. Kellenberg et al. study how to scale CNNs for large wildlife census tasks, especially in remote areas like the African savanna that was explored for the paper. A different study on the Snapshot Serengeti dataset containing 3.2 million images from camera traps using deep neural network architectures resulted in greater than 93.8% accuracy in detecting animals automatically [63]. Further study on camera trap images using convolutional neural networks with ResNet-18 architecture trained the model on 3,367,383 images obtained from five different states in the USA with the highest recorded accuracy in classifying wildlife using machine learning of 97.6% and then tested on out-of-sample datasets from Canada and Tanzania [121]. As evident from all the previous works in the field of image processing using deep learning discussed so far, it is evident that deep neural network classifiers can successfully extract the semantics regardless of the structure or context [57]. As deep learning thrives on large neural networks and large datasets, in order to create more useful applications harvesting this power of deep learning, we need to harness the abundance of image data that is available now. With the enormous amount of digital information being produced in the form of images each second, proper mechanisms and tools are required to retrieve the

images and transform them into suitable forms for learning.

One of the main challenges while trying to build deep learning models is the large volume of labelled data that is needed to properly train the model. At the rate at which data is being generated each second around the world, manually labelling the required data for each deep learning problem is costly and error-prone [74]. So a common practice that is currently being followed to ensure adequate training of deep learning models is generating synthetic images using GAN models and using them to train the model. The GAN series can generate different types of samples to boost the performance of classification models using deep learning [57]. While several applications using CNN were introduced in medical imaging, high-quality, balanced datasets were rare as medical images are mostly imbalanced and time-consuming to obtain their labels [65]. Since the introduction of GAN in medical imaging, it has been used in many applications to generate various image types. Despite the recent progress on unsupervised image generation, conditional models are far superior [74], as we have no control over the modes of data being generated in an unconditioned generative model [80]. By conditioning the model, we can direct the data generation to our requirements and necessitate the generation of synthesised images that belong to the intended class [85]. Generating images conditioned over text descriptions is getting more attention as it brings computer vision and natural language processing closer, but it is still challenging because of the inability to keep the generated image holistically and semantically consistent with the input text [68].

Dataset

3.1 Dataset Description

The BookCover30 dataset contains the details of 57,000 books that are sourced from the Amazon.com, Inc. Marketplace [129] and is accompanied by a folder containing the images of the book covers. The 57,000 books are equally distributed into 30 categories, each containing 1,900 books. The original study performed on the dataset classified the books by their cover using Convolutional Neural Networks (CNN) built on the concepts of transfer learning [56]. The original data was split into two files for the purpose of the research carried out by Uchida et al., which were combined into one dataset for this study.

	Index	Filename	URL	Title	Author	Category_ID	Category
0	1404803335	1404803335.jpg	http://ecx.images-amazon.com/images/I/51UJnL3T...	Magnets: Pulling Together, Pushing Apart (Amaz...	Natalie M. Rosinsky	4	Children's Books
1	1446276082	1446276082.jpg	http://ecx.images-amazon.com/images/I/51MGUKhk...	Energy Security (SAGE Library of International...	NaN	10	Engineering & Transportation
2	1491522666	1491522666.jpg	http://ecx.images-amazon.com/images/I/51qKvjsi...	An Amish Gathering: Life in Lancaster County	Beth Wiseman	9	Christian Books & Bibles
3	970096410	0970096410.jpg	http://ecx.images-amazon.com/images/I/51qoUENb...	City of Rocks Idaho: A Climber's Guide (Region...	Dave Bingham	26	Sports & Outdoors
4	8436808053	8436808053.jpg	http://ecx.images-amazon.com/images/I/41aDW5pz...	Como vencer el insomnio. Tecnicas, reglas y co...	Choliz Montanes	11	Health, Fitness & Dieting

Figure 3.1: First five rows of the original dataset

The first five samples of the original dataset are shown in Figure 3.1. The dataset contains seven columns.

- Index: The Amazon index number that is unique for each book

- **Filename:** The name of the book cover image file in the images folder accompanying the dataset.
- **URL:** The links to the book cover images in the Amazon repository.
- **Title:** The title of the book
- **Author:** Author of the book
- **Category_ID:** The number assigned to each category
- **Category:** The category to which the book belonged.

The thirty categories in which the books are equally distributed and their labels are displayed in Table 3.1.

3.2 Data Preprocessing

3.2.1 Dataset Encoding

The ASCII character encoding that represented numbers 0–9, lowercase and uppercase English letters, some punctuation and symbols, whitespace characters, and some non-printable characters such as ‘\b’ [135] was not accommodating enough for the variety of languages around the world and their dialects, symbols, and glyphs [94]. Unicodes like UTF-8, UTF-16, and UTF-32 are used to make sure a wide range of characters can be converted to bytes of information that can be processed and stored by computers [34]. If the encoding is not specified or is incorrect, it can cause problems where some characters may not read or display correctly. The encoding used while loading the separate datasets into respective dataframes is ‘ISO-8859-1’ (Latin-1), which is popular for its large number of western characters and is also the default for the Hypertext Transfer Protocol (HTTP).

3.2.2 Column Headers

Column headers are necessary for understanding the context of the data, as they provide a name for each column in the dataframe [106]. The original data only contained the details of the 57,000 books; hence, the column headers had to be hard-coded into the

Table 3.1: Book categories in the dataset

Label	Category Name	Size
0	Arts & Photography	1900
1	Biographies & Memoirs	1900
2	Business & Money	1900
3	Calendars	1900
4	Children's Books	1900
5	Comics & Graphic Novels	1900
6	Computers & Technology	1900
7	Cookbooks, Food & Wine	1900
8	Crafts, Hobbies & Home	1900
9	Christian Books & Bibles	1900
10	Engineering & Transportation	1900
11	Health, Fitness & Dieting	1900
12	History	1900
13	Humour & Entertainment	1900
14	Law	1900
15	Literature & Fiction	1900
16	Medical Books	1900
17	Mystery, Thriller & Suspense	1900
18	Parenting & Relationships	1900
19	Politics & Social Sciences	1900
20	Reference	1900
21	Religion & Spirituality	1900
22	Romance	1900
23	Science & Math	1900
24	Science Fiction & Fantasy	1900
25	Self-Help	1900
26	Sports & Outdoors	1900
27	Teen & Young Adult	1900
28	Test Preparation	1900
29	Travel	1900

dataframes before they were combined into one. Two entries from the datasets were removed while adding the column headers, which reduced the shape of the dataset to (56998, 7) and the number of books in the categories 'Biographies & Memoirs' and 'Medical Books' to 1899.

3.2.3 Null values

The presence of missing data can cause problems in a model as it can bias the results [12]. So it is important to take care of the missing values before any modification or analysis is done on the data. The dataset is checked to see if it contains any null values [107]. A column-wise inspection revealed that the 'Author' column contained 378 null values. These were ignored as the 'Author' column was not relevant to the current study, and removing these would create additional instances that the model could train on. The empty strings that resulted as part of the object recognition were converted to null values and filtered out before building the learning model.

3.2.4 Duplicates

Duplicates in the dataset can cause several problems, such as overfitting, inflated accuracy, and distorted representation, all of which will produce faulty outcomes. Removing the duplicates makes the results more reliable. The dataset was scanned for duplicate values using the duplicated() function and revealed to have none [40]. The duplicated function returns a boolean series that indicates whether the row is duplicated or not [98], and the sum() function returns the sum of all true values if there are any. The duplicated() function can be modified to check the duplicates in specific columns as well [19]. Even though the 'Title' column had 378 duplicates, these were allowed as the same title could have different book covers, which can add to the colour analysis of categories.

3.2.5 Cleaning the data

Dealing with text data is always a hurdle for computers because understanding the context and semantics is very difficult in digital terms. Even though many models are being developed that can handle textual data as well as humans, the pre-requisites

to using these models involve a lot of pre-processing. The most basic step out of all is removing the special characters that could cause the models confusion and only retaining the alphanumeric values and white space. Converting all the text to lowercase can help in following preprocesses by avoiding the mismatch due to uppercase and lowercase characters or in later stages of natural language processing [125].

3.2.6 Stopwords Removal

Stopwords are commonly used words that are required for grammatical accuracy but not for context. These are often removed by language processing models in order to increase the efficiency of the process. This is done with the help of an existing list of stop words. NLTK (Natural Language Toolkit) in Python contains lists of stopwords in several languages [38]. Some of the stopwords for English language stored in NLTK are: I, me, my, we, our, you, you're, your, he, him, his, she, her, herself, it, it's, its, they, them, their, what, which, who, this, that, these, those, am, is, are, was, were, be, have, has, had, do, does, did, a, an, the, and, but, if, or, because, as, while, of, at, by, for, with, about, against, between, through, before, after, to, from, up, down, in, out, on, off, again, then, here, there, when, where, why, how, all, any, both, few, most, no, nor, only, same, so, too, can, will, now etc. and the contractions of some of these words like isn't. As helpful as this is to language processing, it is not mandatory to remove the stopwords in every sentimental analysis, as these can sometimes provide context to the matter under inspection. One can choose to remove or keep the stopwords based on the type of data and the analysis required of them, or use another list of stopwords that is more suitable.

3.2.7 Type conversion

One of the most important factors to notice while dealing with data is whether the data type is compatible with the procedure. In most machine learning models, if the data is not fed in the correct format, it won't be able to process the data and will end in error. When the data is extracted from other data elements using other methods, the output can be different from what we expect. So it's crucial to convert the data to the required type. In two out of three feature extraction models, the data is formatted so that it comes out as required for the model, but in the case of object recognition, the list that resulted was converted into a string type in order for the model to process it.

3.2.8 Label Encoding

Most datasets that are subjected to machine learning contain different types of data. In the majority of cases, these datasets containing textual, categorical, and numeric data will need to be trained by the same model. But machine learning algorithms cannot process non-numerical data directly, so it will need to be converted into a numeric type. `LabelEncoder` from the `scikit-learn` library helps normalise labels so that they only contain values between 0 and $n-1$ [109]. It assigns an integer value for each category based on its alphabetical order [103]. Table 3.1, which contains the category name and the category ID, is an example of label encoding where the categories are assigned an integer as category ID depending on their alphabetical order.

3.2.9 One-Hot Encoding

Label encoding is very straightforward in converting non-numerical values into numerical values, but it has the disadvantage of algorithms assuming some sort of hierarchy within them [139]. One Hot Encoding is a representation of categorical variables as binary vectors [14]. It essentially creates dummy variables based on the number of unique values in the categorical feature [113]. While manual one-hot encoding is possible, it is not practical in the case of large datasets. `OneHotEncoder` from `scikit-learn` automatically maps categories to integers and integers to binary vectors. Another way to encode is to use the function `get_dummies()` from the `Pandas` library. For example, if a categorical variable has five labels: white, red, green, blue, and black, the one hot encoding on the labels is demonstrated in Figure 3.2.

	0	1	2	3	4
white	1	0	0	0	0
red	0	1	0	0	0
green	0	0	1	0	0
blue	0	0	0	1	0
black	0	0	0	0	1

Figure 3.2: An Example of One-Hot Encoding

3.2.10 Tokenizer

The objective of tokenizers is to convert textual data into a form that's readable by algorithms without losing context [10]. It converts an unstructured string into a numerical data structure that's machine-readable. Tokenizer takes a stream of characters as input and produces a stream of tokens as output after breaking the input into individual tokens [31]. By converting them into tokens, algorithms can easily identify patterns, which will help machines understand and respond to human input [10]. It can segregate sentences, words, and characters into tokens [79]. A token is a sample of a sequence of characters that are grouped together as a machine-readable semantic unit [27]. Before converting the words into tokens, the space-separated words are stripped of all punctuation, with the exception of apostrophes in some cases, which will be vectorized before they are subjected to computer processing [43]. One of the most popular tokenizers in use is the `text_to_sequence` method from the `keras.preprocessing.text.Tokenizer()` class, which has the additional advantage of converting the words into lowercase before tokenization, significantly reducing runtime.

3.2.11 TfidfVectorizer

Vectorization is the process of turning a collection of texts into numerical vectors [108]. `TfidfVectorizer` converts raw documents into a matrix of TF-IDF features [58]. TF-IDF (term frequency-inverse document frequency) is a measure of the importance of words within a document and across a corpus, providing valuable insights to machine learning tasks like information retrieval, document clustering, and text classification [86]. Term Frequency (TF) is specific to a document as it is the number of times a word appears in that document [102]. It can be calculated as in equation 3.1.

$$tf(w, d) = \frac{\text{occurrence of } w \text{ in document } d}{\text{total number of words in document } d} \quad (3.1)$$

Inverse Document Frequency (IDF) is a measure of how common a term is across the entire corpus of documents and is calculated as equation 3.2 [61].

$$idf(w, D) = \ln \left(\frac{\text{total number of documents } (N) \text{ in corpus } D}{\text{number of documents containing } w} \right) \quad (3.2)$$

$$idf(w, D) = \log \left(\frac{N}{f(w, D)} \right) \quad (3.3)$$

The combination of these two factors, the TF-IDF, is calculated as equation 3.4.

$$tfidf(w, d, D) = tf(w, d) \times idf(w, D) \quad (3.4)$$

The relevancy of a term increases with an increase in the TF-IDF value. Scikit-learn's TF-IDF Vectorizer (Term Frequency-Inverse Document Frequency) combines the CountVectorizer and TF-IDF Transformer [123]. Compared to the TfidfTransformer, where the word counts had to be computed using the CountVectorizer and then the TF-IDF scores were computed separately, the TfidfVectorizer automatically converted a set of documents into a matrix of TF-IDF features [33]. It can be further modified for better performance by adding parameters such as:

- `stop_words` to remove stopwords from the output,
- `max_df` and `min_df`, which set thresholds that decide if a term is to be considered by the vectorizer,
- `max_features` can limit the number of features that can appear in the final output [124].

3.2.12 Train-Test Split

The majority of the machine learning models follow the same procedural flow, which consists mainly of four steps:

1. Arrange the data: Arrange the data into a format acceptable for a train test split.
2. Split the data: splitting the data into training and testing sets
3. Train the model: training the model on the training set
4. Test the model: Evaluate the performance of the model using the testing set [52].

A model evaluated on the data it was trained on is bound to bias the results. In order to produce unbiased results, the model needs to be tested on unfamiliar data, which can be accomplished by splitting the dataset [93]. The `train_test_split()` function from the scikit-learn library splits the data into training and testing sets based on the `test_size` parameter and the `random_state` parameter, which sets the seed for reproducibility. This ensures that the model is not overfitted and can also respond well to new data [20].

Before splitting the dataset into training and testing sets, it is crucial to make sure the two subsets are large enough to represent the problem. If the data is not enough, the model performance could be falsely portrayed as either too good or too bad [16].

3.3 Feature Extraction

Feature extraction is the process of transforming raw data, such as texts or images, into a format supported by machine learning algorithms. It is part of the dimensionality reduction process, which divides the data into more manageable groups [18] so that the processing is made easier, especially for large datasets. One of the major applications of feature extraction is in image processing, as it is one of the toughest data types for machine learning algorithms to handle. For this study, we are carrying out three feature extraction tasks on the book cover images focused on the three core design areas identified by Uchida et al.: colour, text, and object [56].

3.3.1 Colour Extraction using K-Means Clustering

Clustering is a type of unsupervised learning that aggregates data into different groups based on their similarities and dissimilarities. They are grouped in such a way that all the data points in a group are similar to each other, and those from different groups are as different as possible [116]. K-means Clustering is a widely used clustering technique that separates the data into a predefined number of clusters. It is an iterative process with the objective of minimising the sum of distances between the data points and the cluster centroid while identifying which group each data point should belong to [115]. The process consists of four stages:

1. Choosing the K number of clusters into which the data is grouped.
2. Initialising the centroids randomly for each cluster as the exact centre of the data points is unknown
3. Assign data points to the nearest cluster by first calculating the distance between data point X and centroid C using the Euclidean Distance metric and then choosing the cluster for data points where the distance between the data point and the centroid is minimum [115].

4. Re-initialising the centroids based on the data points of that cluster

Steps iii and iv are repeated until the model finalises the centroid for each cluster and the assigned data points to clusters are no longer changing.

An image is a collection of pixels that contains three values ranging from 0 to 255, representing the red, blue, and green components that define its colour [35]. We can use K-means clustering to automatically group pixels with similar colours by clustering them into groups. This can be very advantageous in extracting the colour palette out of images, especially when there are many colours present in the image [69]. For this study, K-means clustering is used to produce a list of the top 5 colours on each book cover based on their occurrence. The function `extract_colours()` takes an image as input and converts it into a collection of pixels by reshaping it according to the size of the image. The `KMeans()` method from the `sklearn.cluster` class then creates 5 clusters and saves the centroids to the `colour_palette` variable. This variable is then formatted into a list, which is returned to the function call. The `apply_extraction()` function converts the book cover image, which is in .jpg format, into a Matplotlib image object before sending it to the `extract_colours()` function. This is applied to each row in the dataset and saves the output from the `extract_colours()` function to a new column named 'Colours'.

3.3.2 Text Area Calculation with EasyOCR

Optical Character Recognition (OCR) converts text data in an image into machine-readable format [8] using pattern recognition and feature extraction. There are various types of OCR technologies based on the type of data they capture.

- Optical Character Recognition (OCR) that can recognise handwritten or typed characters based on an existing internal database
- OWR Word Recognition (OWR) targets typewritten, space-separated text, one specific word at a time.
- Optical Mark Recognition (OMR) analyses watermarks, logos, symbols, marks, and patterns on a paper document.
- Intelligent Character Recognition (ICR) uses data capture tools to read handwritten or cursive text using machine learning and AI technology [48].

The OCR technology already has several applications in the banking, healthcare, and logistics industries [21]. With advances in the fields of machine learning and artificial intelligence, OCR technologies are evolving to the point where they can understand the content of the text that is recognized. They enable Natural Language Processing models to decipher text and understand the context, and they also enable translations into several different languages. Deep learning technologies built on neural networks can extract a large number of features from enormous datasets [11]. The Tesseract library in Python is an open-sourced OCR engine that is used in various operating systems for character recognition [11]. But it can be a bit difficult to work with, especially for people starting out in the field of OCR.

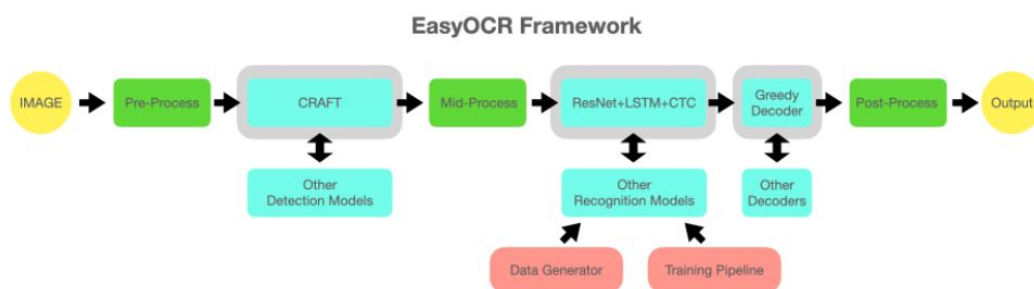


Figure 3.3: EasyOCR Framework. Source: [5]

EasyOCR, a ready-to-use OCR Python package developed and maintained by Jaided AI, can support 80+ languages [4] and can effortlessly perform OCR functions. Figure 3.3 shows the EasyOCR [5]. Installing in a single command and after import, it can conduct OCR with simply two lines of code: one to initialise the Reader class and the other to extract the text using the `readtext()` function [101]. EasyOCR is implemented using pyTorch library. A CUDA-capable-capable speed up text detection and OCR if it's available [66]. For this study, the EasyOCR package is used to extract the bounding box coordinates of the recognised text and calculate the area of those bounding boxes. The function `calculate_area` takes the image input from the `read_text()` function and returns the percentage of text detected in the image. Applying the `read_text()` function to each row saves the percentage of the text detected in a new column, 'Text_Area'.

3.3.3 Object Recognition with Google Vision API

Object Recognition is a computer vision technique for identifying objects in images or videos and is an output of deep learning and machine learning algorithms. It is the building block of technology behind a variety of applications, like self-driving cars, medical imaging, and security detection systems. Popular machine learning models for object recognition are HOG feature extraction with an SVM machine learning model, Bag-of-words models with SURF and MSER features, and the Viola-Jones algorithm. Deep learning models are either trained from scratch or an existing network is combined with a pre-trained model such as AlexNet, and they offer high accuracy to make up for the requirement of a large amount of training data. They also work faster with a GPU and produce better results [77]. Widely used deep learning models are YOLO (You Only Look Once), Single-shot Detector (SSD), and Region-based convolutional neural networks (R-CNNs) [45]. Running these highly advanced deep learning models will require some experience on the developer's end and a lot of computational power with an advanced system. So object recognition API services like Google Cloud Vision API, Chooch AI, and Visua AI are becoming popular for swift and small-scale object recognition requirements [3].

The Google Vision API utilises pre-trained machine learning models to detect objects, texts, and faces [122]. It can extract multiple objects from an image with Object Localization and provide a LocalizedObjectAnnotation [42]. While the current annotations are only in English, they can be translated into several languages with the help of Cloud Translation. The Object Localization output contains the textual description of the object identified, the confidence score, and the normalised vertices of the bounding polygon around the object [120]. The Google Vision API is used for object detection and tracking in automated manufacturing, robotics, retail, surveillance systems, and smart cities because of its efficiency in identifying people, animals, vehicles, and other objects [32]. The Google Vision API is very accessible for the images we train on our personal devices and can be utilised through API requests. For this study, the objects in the book cover images were recognised using the Google Vision API services in the `detect_objects()` function. The `localized_object_annotations` obtained from each image were saved under the new column 'Objects'.

Methodology

4.1 Regression Models

Regression is a supervised machine learning model that predicts the output of a continuous numerical variable [114]. It is often practiced in finance and investing to learn the relationships within the data at hand and make predictions based on the patterns that are identified from the data. Common applications of machine learning regression models are:

- Forecasting house prices, stock prices, or sales [17].
- Predicting the success of retail marketing campaigns
- Predicting user trends
- Time series visualisations

The four regression algorithms used for this study are: Linear Regression, Decision Tree, Support Vector Regression, and Random Forest.

4.1.1 Linear Regression

In statistics, linear regression analysis is used to predict the value of a continuous dependent variable from the value of an independent variable under the assumption of a linear relationship between the two variables. This is done by calculating the

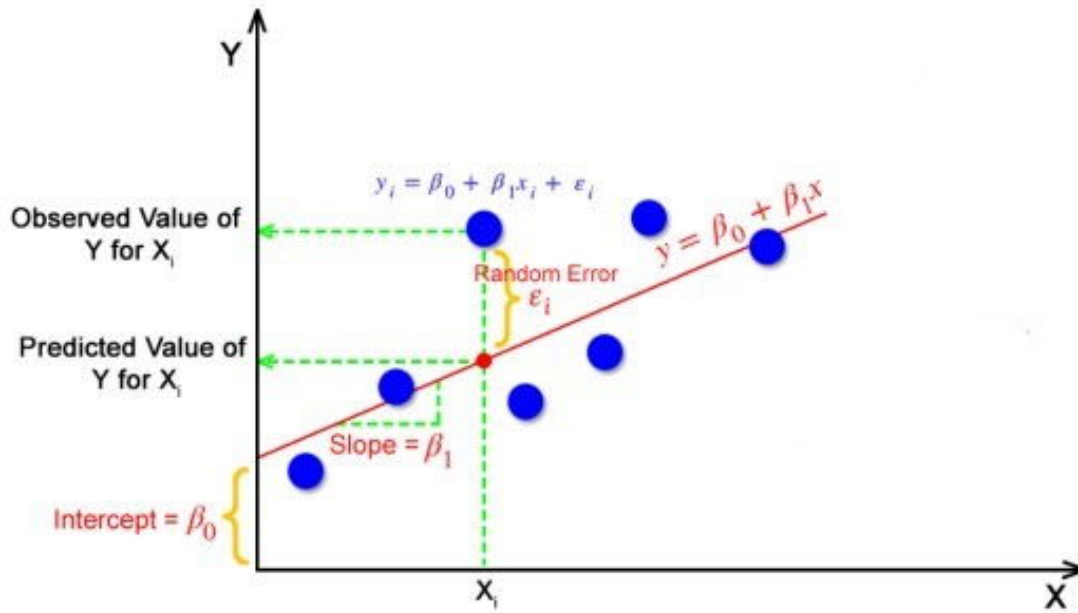


Figure 4.1: Graphical Representaion of Linear Regression. Source [76]

coefficients of a linear equation from the available independent and dependent variables and then using the coefficients to predict the dependent variable. The goal of linear regression is to find the best-fitting line (Figure 4.1) that describes the relationship between the variables by minimising the sum of squared differences between predicted and actual values [76]. The representation of simple linear regression in mathematical form is as follows:

$$y = b_0 + b_1 x \quad (4.1)$$

Linear regression is the simplest statistical regression method used in predictive analysis [24] and is applied in price elasticity analysis, risk assessment in insurance companies, and sports analysis [49]. Linear regression is also adopted in the field of machine learning to build prediction models. The LinearRegression model from the sklearn.linear_model library fits a linear model with coefficients to optimise the residual sum of squares between the observed targets in the dataset and the predicted targets by linear approximation [111].

4.1.2 Decision Tree Regression

Decision trees are a non-parametric supervised machine learning algorithm that can be used for both regression and classification tasks [105]. It focuses on providing a

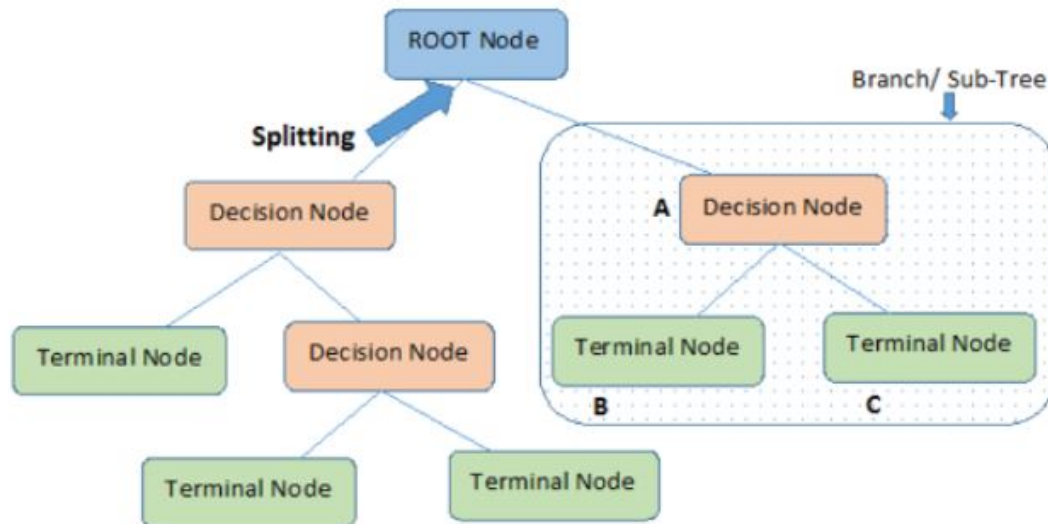


Figure 4.2: Graphical Representaion of Decision Tree Model. Source [105]

prediction result for a target variable based on simple decision rules deduced from data features [110]. It has a hierarchical tree structure with three nodes:

- Root Node: The initial node represents the entire sample.
- Interior/Decision Node: represents the features of the dataset, and branches represent decision rules [60].
- Leaf/Terminal Node: The outcome

A graphical representation of nodes is in Figure 4.2. Decision trees are simple to understand and visualise and also require very little time for preparing the model. But it can also be very unstable and susceptible to overfitting and bias. A regression tree is used when a decision tree is used to predict continuous outputs [91]. It is executed using the `sklearn.tree.DecisionTreeRegressor` class. The decision trees fitted with a sine curve with additional noisy observation learn the local linear regressions approximating the sine curve [110]. In figure 4.3, it is evident that when the maximum depth of the tree is too high, the tree starts to learn from the noises, which results in overfitting.

4.1.3 Random Forest Regression

Random Forest Regression is a supervised ensemble learning method that has multiple decision trees constructed as base learning models [37]. These decision trees run in

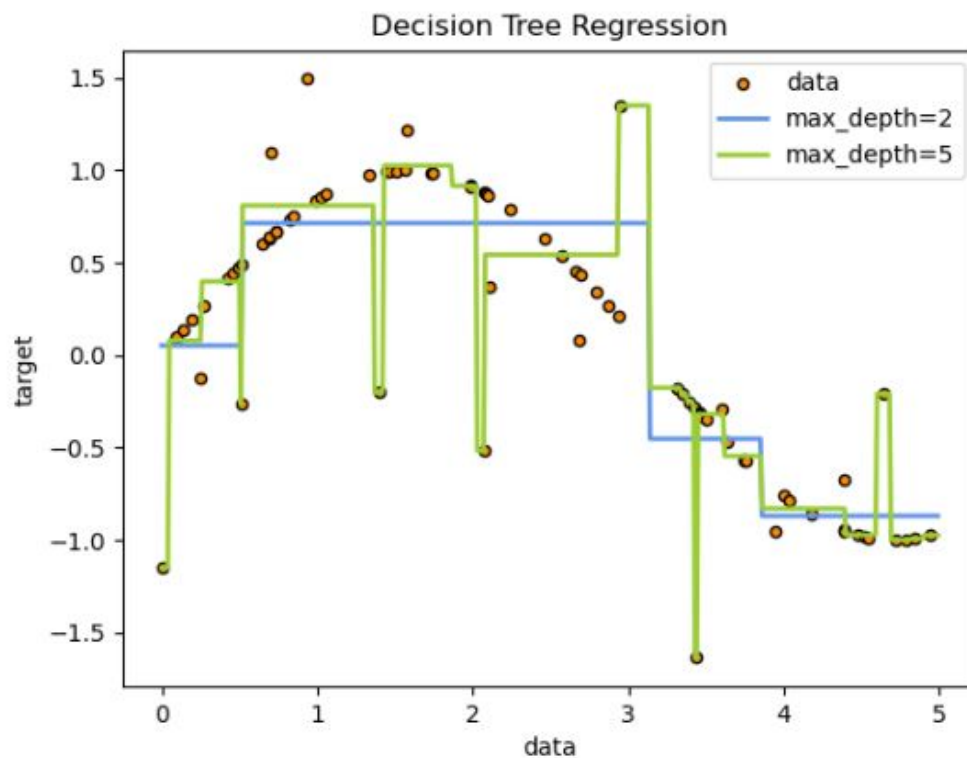


Figure 4.3: Graphical Representaion of Overfitting in Decision Tree Regression Model.
Source [110]

parallel with no interaction with each other and produce their own output, which is then aggregated into a single result. Figure 4.4 shows the basic structure of a random forest. It can handle large datasets and produce accurate results, compared to many other learning algorithms. It also has an effective method for handling missing data and producing unbiased data while maintaining accuracy [51]. But random forests are prone to overfitting in the presence of noisy datasets. Random forest models are widely used in banking, health care, the stock market, and marketing to identify risks or predict behaviours [112]. In Python, the `sklearn.ensemble` package produces the `RandomForestRegressor` class to build random forest regression models.

4.1.4 Support Vector Regression

A Support Vector Machine (SVM) is a supervised machine learning algorithm used for both classification and regression tasks [113]. The regression model SVR (Support

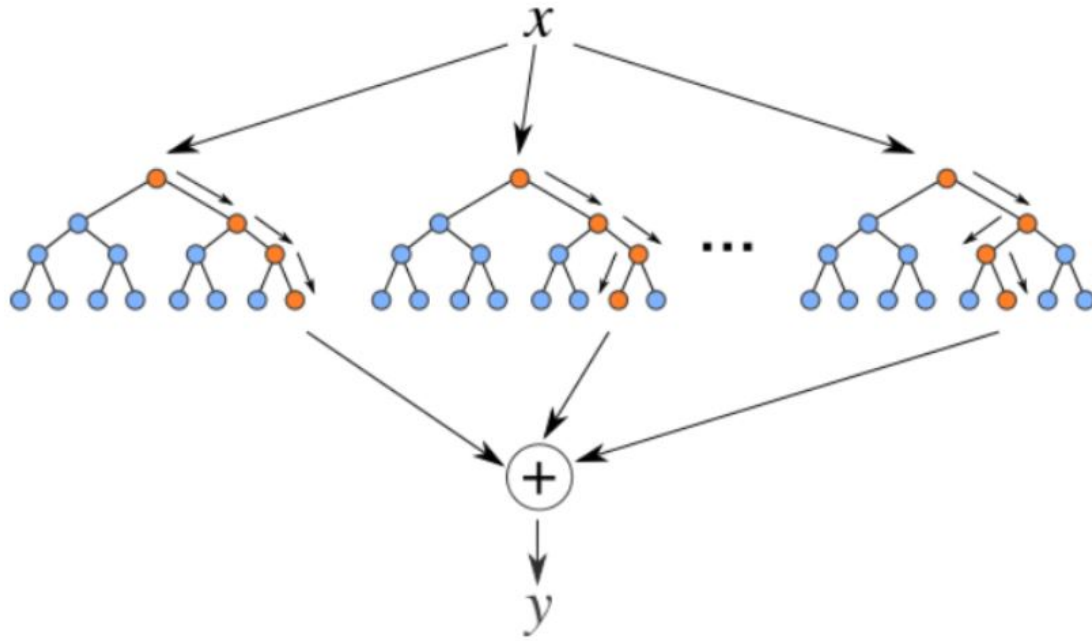


Figure 4.4: Graphical Representaion of Random Forest Model. Source [114]

Vector Regression) aims to find a function that approximates the relationship between the input variables and a continuous target variable. SVR maps the input variables to a high-dimensional feature space and finds a hyperplane that maximises the margin between the hyperplane and the closest datapoint while minimising prediction error. Consider the two dotted lines in Figure 4.5 as the decision boundaries that create a margin between the datapoint and the blue line as the hyperplane, which helps predict the output value. If the equation of hyperplace is as follows:

$$y = wx + b \quad (4.2)$$

Then the boundary line equations are:

$$wx + b = +\epsilon \quad (4.3)$$

$$wx + b = -\epsilon \quad (4.4)$$

Thus, any hyperplane that satisfies the SVR should also satisfy:

$$-\epsilon < Y - wx + b < +\epsilon \quad (4.5)$$

The best-fit line is the hyperplane that has the maximum number of data points without violating the margin. As the regression is performed at higher dimensions, a linear or

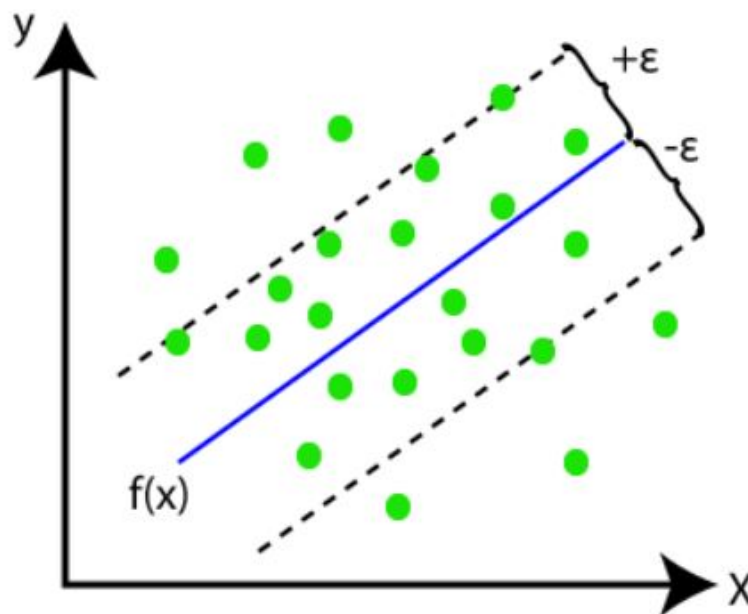


Figure 4.5: Graphical Representaion of Support Vector Regression. Sourec [90]

non-linear kernel function is used to map the data points, such as the Sigmoidal kernel, the Polynomial kernel, and the Gaussian kernel [29]. In the scikit-learn Python package, the SVR class with a linear or non-linear kernel is used for regression based on the complexity of the task and the properties of the data [39]. While the model is immune to outliers, the decision models are highly vulnerable to noise in the dataset [96].

4.2 Classification Models

Classification is a supervised machine learning model that can predict the class labels of input data from the problem domain [13]. Then the model is fully trained on the training data and then evaluated on the testing data before it is used for classification [62]. There are mainly four types of classification methods [81], which are:

- Binary classification
- Multi-Label Classification
- Multi-Class Classification
- Imbalanced Classification

These different types of classification methods are adopted in the fields of healthcare, education, transportation, agriculture, [62] retail, and finance to build machine learning models for image classification, fraud detection, document classification, spam filtering, malware detection, facial and voice recognition, customer behaviour prediction, and product categorization [97]. Five classification models are explored for this study: Logistic Regression, Decision Trees, Random Forests, Support Vector Machine, and Multinomial Naive-Bayes Classifier.

4.2.1 Logistic Regression

While the name is deceiving, logistic regression is a classification algorithm that predicts the probability of certain classes based on dependent variables. There are three types of logistic regression: Binomial, Multinomial and Ordinal. But it is most widely used for binomial classifications, as the output of logistic regression is always one of the two possibilities, 0 or 1. If we try to fit a linear regression model to a binary classification problem, the model will fit in a straight line and forecast continuous results [54]. But the logistic regression, which uses a sigmoid function as shown in equation 4.6, takes any real input x and outputs a probability value between 0 and 1.

$$\sigma(x) = \frac{1}{1 + e^{-x}} \quad (4.6)$$

Here, theta is the parameter we want to learn, train, or optimize. Figure 4.6 shows a comparison between the linear and logistic regression models fitted to the same data, and we can see evidently how logistic regression classifies the data. The sigmoid function squeezes the straight line into an S-curve [104].

4.2.2 Decision Tree Classification

As a supervised learning algorithm, the Decision Tree classifier is trained on pre-processed data. It is built through binary partitioning [95], which is an iterative process of splitting the data into different nodes in a top-down approach. At the end of training, an optimised decision tree is returned that is qualified with categorical prediction ability [88], as shown in Figure 4.7. Entropy is a very important factor in decision tree classification as it is a measure of uncertainty in a particular dataset and describes the

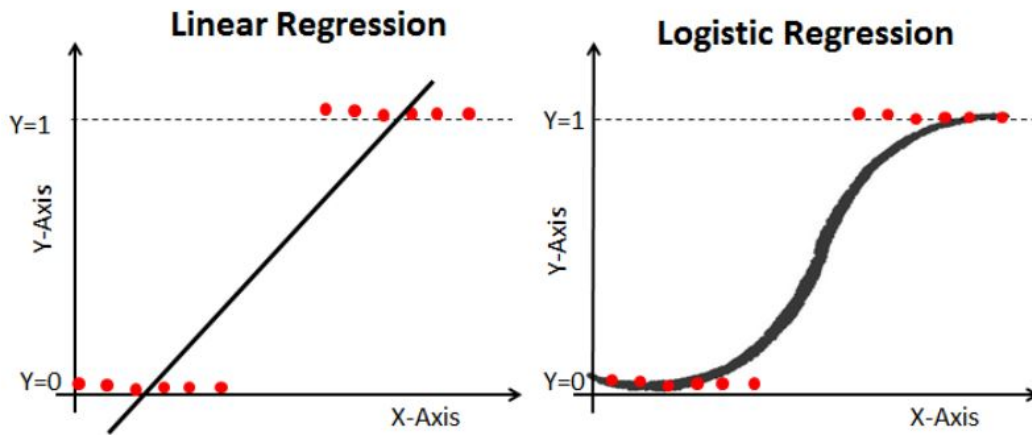


Figure 4.6: Comparison between Linear and Logistic Regression. Source [104]

degree of randomness of a specific node [95]. It is calculated using equation 4.7.

$$Entropy = \sum_{i=1}^c -p * \log_2(p_i) \quad (4.7)$$

As high values of entropy indicate higher randomness in the data, it is important to aim for lower entropies while training decision trees. Decision Tree models are extremely fast and efficient at classification tasks and are comparatively inexpensive. But it possesses higher chances of overfitting and can be complicated when it comes to large trees. Decision Tree Classifiers are used in:

- Biomedical engineering for identifying features used in implantable devices
- Financial Analysis: Customer Satisfaction
- Physics: Particle detection
- Astronomy: Galaxy classification
- Manufacturing and production: quality control [53]

4.2.3 Random Forest Classification

Random forest classification is used to determine which class a target variable is most likely to be a part of [72]. Multiple decision trees are created from random subsets of data and features, which are separately progressed and then combined together to vote

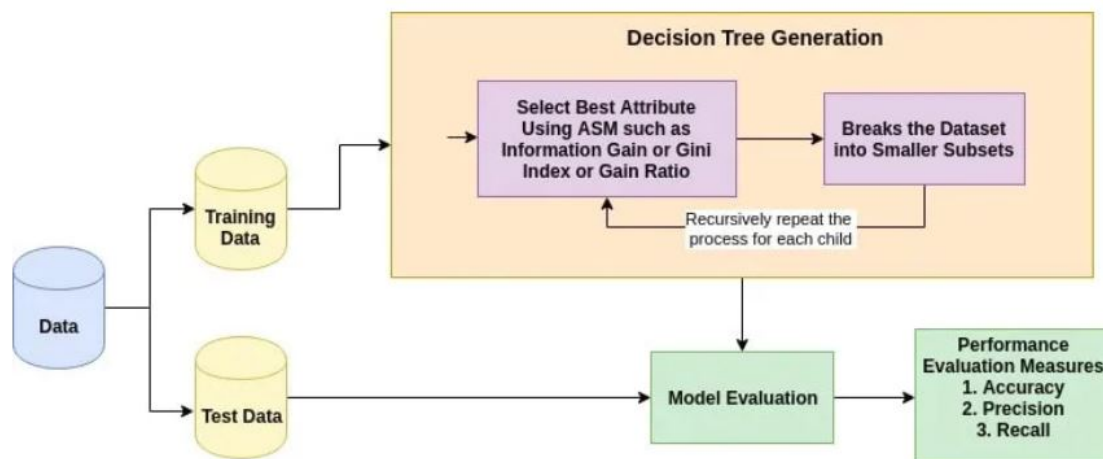


Figure 4.7: Decision Tree Classification Framework. Source [88]

on the highest occurring result. Figure 4.8 is a graphical representation of the random forest classification. As portrayed in the figure, the working of a random forest consists of the following steps:

1. Selecting random samples from the dataset
2. constructing a decision tree for every sample and obtaining the prediction results from each tree.
3. Voting was performed on every output.
4. Selecting the most-voted prediction as the final result.

Random forest classification models work well with large sets of data and overcome overfitting by combining the results. They are very flexible and possess very high accuracy without scaling or even in the presence of missing data. But the models tend to be rather complex and time-consuming, which requires more computing resources [128].

4.2.4 Support Vector Machine

Support Vector Machine is a highly scalable classification algorithm that assigns data to a particular category based on the training data [134]. This is a well-suited model

Random Forest Classifier

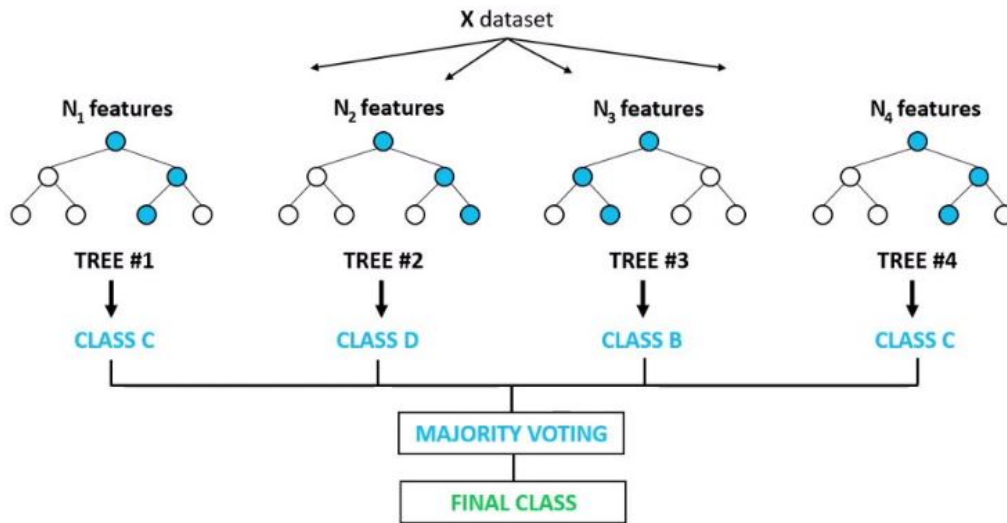


Figure 4.8: Random Forest Classification. Source [92]

for linearly separable data. The model aims to find a hyperplane that best separates the different classes of data points and chooses the hyperplane with the maximum margin. This is demonstrated in figure 4.9. Support Vector Machines are very effective in high-dimensional spaces with the support of non-linear kernels. While they are versatile and memory-efficient, they are also computationally expensive to train and highly sensitive to the choice of parameters [118].

4.2.5 Multinomial Naive Bayes Model

The Multinomial Naive Bayes Classifier is a supervised machine learning algorithm that is widely used for text classification [25]. Based on the Bayes Theorem in statistics, this classifier calculates the probability of each tag for a given sample and gives the highest probability as output. The Bayes theorem evaluates the probability of an event occurring based on prior knowledge of conditions related to the event [130]. The probability of event A occurring given that B is true can be written in statistical terms as:

$$P(A|B) = \frac{P(A) * P(B|A)}{P(B)} \quad (4.8)$$

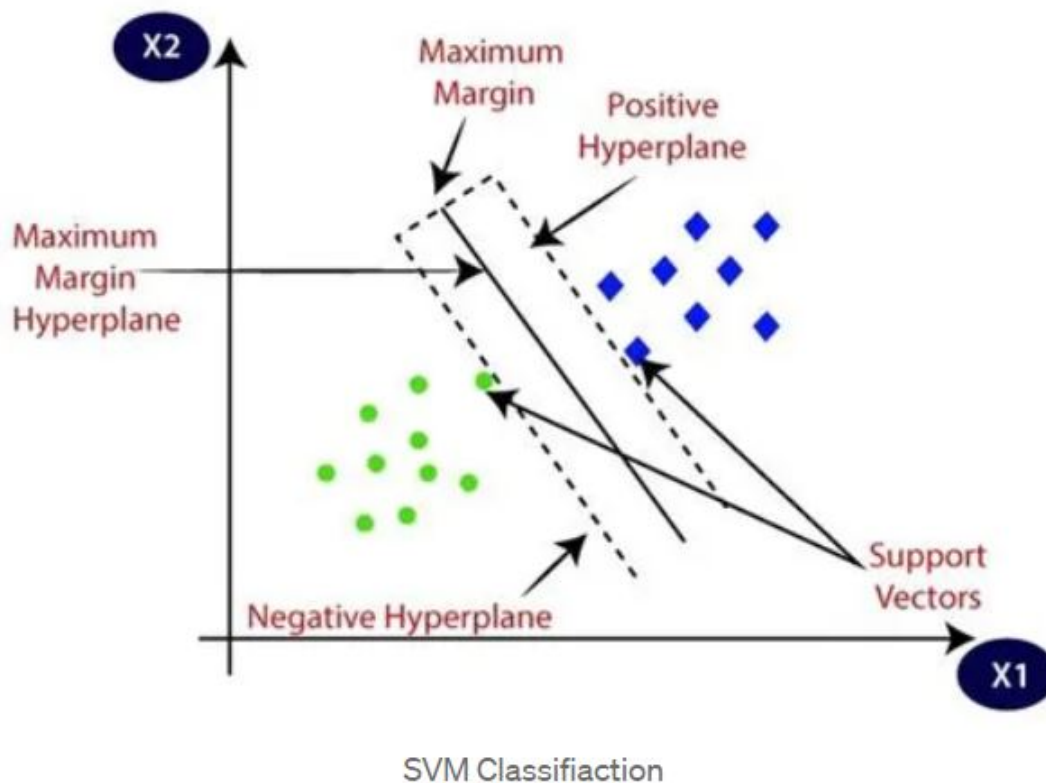


Figure 4.9: Support Vector Machine Classification. Source [118]

where, $P(B)$ = independent probability of B

$P(A)$ = independent probability of class A

$P(B|A)$ = probability of event B given A is true [18].

The Multinomial NB model is most popular for assigning documents to classes based on the likelihood of that document belonging to a class of other documents having similar content. This model tests the hypothesis that a document's words already appear in other documents from that particular class [99]. The 'naive' factor in the model comes from the assumptions that predictors in the model are conditionally independent and that all features contribute equally to the outcome. Even though this assumption doesn't work well in every real-world situation, it makes the classification process easier, especially for small datasets [25]. The Multinomial NB Classification is also used in weather prediction, spam detection, sentimental analysis, language identification, news classification, authorship identification, face recognition, age/gender identification, and medical diagnosis [130].

4.3 Neural Networks

Neural networks are a part of artificial intelligence that teach computers to process data like human brains perceive data. With the help of deep learning, existing neural network models can make intelligent decisions with limited human involvement [7]. Neural network models are made of nodes that are the software equivalent of neurons in the human brain. A basic neural network model has three layers of interconnected nodes, as depicted in Figure 4.10.

1. Input layer: that processes the data fed to the model and passes on to the next layer.
2. Hidden layer: The incoming data from the input layer or other hidden layers is analysed and processed before passing on to the next hidden layer or the output layer. Neural network models running on deep learning technologies have several hidden layers between the input and output layers with interconnected nodes.
3. Output layer: gives the final result of the entire data processing by the neural network.

There are three main types of neural networks in popular practice: Feedforward Neural Networks or Multi-Layer Perceptrons (MLPs), Convolutional Neural Networks and Recurrent Neural Networks [50]. MLPs are the foundation for computer vision, natural language processing, and other neural networks. CNNs are usually utilised in image recognition, pattern recognition, and computer vision, while RNNs are used for time series data to make predictions. These different types of neural networks are used in several industries. A few applications are:

- Medical diagnosis aided by image classification
- Targeted marketing through social media filtering and behavioural analysis
- Financial forecasting
- Energy demand forecasting
- Process and quality control

Deep neural network

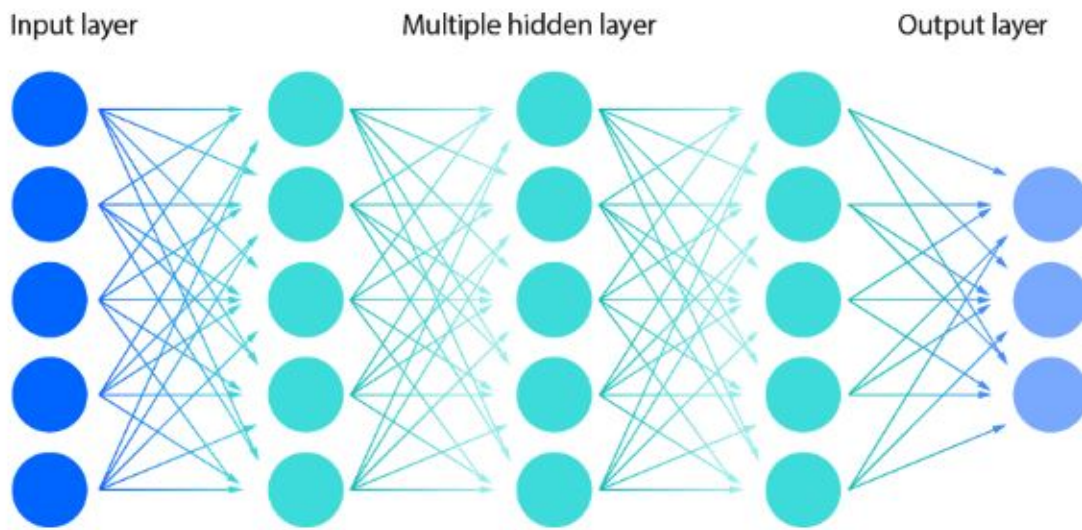


Figure 4.10: Neural Network Structure. Source [50]

- Self-driving cars
- Speech and face recognition
- Automated virtual assistants
- Hate-comment filtering in social media
- Recommendation engines

For this study, we are using two feedforward neural network models: MLPClassifier and Tensorflow Sequential.

4.3.1 MLP Classifier

The Multi-Layer Perceptron (MLP) Classifier from the `sklearn.neural_network` package is Scikit-learn's deep learning model, which relies on its underlying neural networks to perform complex classification tasks. Compared to other more complex neural network models, MLPClassifier is very easy to implement, just like other machine learning classifiers from the scikit-learn library [84]. An MLP classifier consists of multiple layers of interconnected artificial neurons called perceptrons. Each layer receives the input

signals, applies a non-linear activation function, and passes the output to the next layer. The MLP classifier is trained in two steps: the forward propagation, where the input data is pushed through the network and the output of each neuron is computed, and the backpropagation, where the error is propagated backward through the network and the weights of the neurons are adjusted using gradient descent optimisation [117]. This loop is continued until the network converges to its optimum accuracy. MLP classifiers can handle complex non-linear relationships between the variables and labels. It can handle numerical, categorical, and text data, which allows them to carry out multi-class classification problems and also learn relevant features from raw data, eliminating the need for feature engineering. MLP classifiers are used in image and object recognition, natural language processing, financial analysis, and medical diagnosis.

4.3.2 Tensorflow Sequential

Tensorflow is an open-source deep learning framework developed and maintained by Google. Using TensorFlow can be a bit challenging, especially for less experienced coders with limited computing resources [15]. The Tensorflow sequential model is an API that enables the user to create complex neural networks easily [23]. The tensorflow Sequential available in Keras contains a stack of linear formats that consists of various layers of the library package `tf.keras.Model` and is inherited from the `Module`, `Layer`, and `Model` classes [28]. It creates a sequential model of tensorflow as per the specified arguments and attributes and executes and creates each model layer one by one. The most frequently used sequential architecture models are VGGNet, AlexNet, and LeNet [100].

4.4 Experimental Setup

The study is split into four parts. For each part, different models were tested on the data, and the model with the best evaluation metrics was selected for the prediction task. The first and fourth parts are classification models, and the other two are regression models. The models tested on the data are machine learning and neural network models discussed in earlier sections.

Table 4.1: Parameters of TfidfVectorizer

Parameter	Value
min_df	2
max_features	70000
strip_accents	unicode
lowercase	True
analyzer	word
token_pattern	\w+
use_idf	True
smooth_idf	True
sublinear_tf	True
stop_words	english

4.4.1 Category Prediction from Book Titles

This prediction study is carried out on the 'Title' and 'Category' columns. The category labels in the 'Category' column are encoded using LabelEncoder() from the sklearn.preprocessing package, which is set as the target variable. The English stop-words were removed from the features variable 'Title' using the stopwords package from nltk before passing them through the TfidfVectorizer(). The parameters for the TfidfVectorizer are provided in Table 4.1. The output of the vectorizer and encoded categories is split into training and testing sets in an 80/20 ratio. The training sets of X and y are first run through a Logistic regression model with a 'sag' (Stochastic Average Gradient) solver, a max_iter value of 200, and a random state of 450. The model is tested on the test_x subset, and the F1 score and accuracy score are captured. The second prediction model is a neural network classification model called MLPClassifier (Multi-Layer Perceptron). Table 4.2 lists the parameters that were used to train this model on the X and y train sets. The F1-score and accuracy score are captured from the test_y values and the values predicted by the model for the test_X values. The evaluation metrics of the two models are compared.

Table 4.2: Parameter of MLPClassifier

Parameter	Value
activation	logistic
alpha	0.00003
batch_size	auto
beta_1	0.9
beta_2	0.999
early_stopping	False
epsilon	1e-08
hidden_layer_sizes	(20,)
learning_rate	constant
learning_rate_init	0.003
max_iter	200
momentum	0.9
nesterovs_momentum	True
power_t	0.5
random_state	450
shuffle	True
solver	adam
tol	0.0001
validation_fraction	0.1
verbose	False
warm_start	False

4.4.2 Colour Prediction from Categories

This study is carried out by transforming the top five colours into different types. The output of the colour extraction function is a list of the five colours in RGB. The first set of machine learning models was tested on these values. As a first step, five colours that were in a list in a single column were split into five separate columns named 'Colour1', 'Colour2', 'Colour3', 'Colour4', and 'Colour5'. As I could not find any models to process the RGB values as they are, the predictions were carried out for each of the components in all five colours. As the colour extraction sorted the list based on the occurrence of the colour pixels in the images, the individual columns were ranked, as the first column contained the most occurring colours for each image and the last column contained the least occurring colours out of the five. The models used for this study are Linear Regression, Decision Tree Regression, Random Forest Regression and Support Vector Regression. The red value of the first colour was tested with the models as the target variable. As it is a numeric value, it did not need any preprocessing. The features column 'Category' was encoded using one-hot encoding before subjecting to `train_test_split`. All four regression models were trained on the X and y training sets. The trained models were tested on the test_X and test_y sets, and the evaluation metrics were recorded. The evaluation metrics used for these models are the Mean Squared Error (MSE), the Mean Absolute Error (MAE), and the R-Squared values. Based on the evaluation metrics, the most suitable model was selected for further analysis.

The second part of this study focused on the HSL (Hue, Saturation, Lightness) values of the colours. The RGB values in all five columns were converted into HSL values with the `rgb_to_hsl()` function. The columns 'hsl_1', 'hsl_2', 'hsl_3', 'hsl_4', and 'hsl_5' now contain the HSL values of their respective RGB values. The HSL values were also subjected to machine learning in the same way RGB values were. All four models were tested against the Hue value of the first model, and based on the evaluation metrics, the chosen machine learning model was used for predicting the rest of the values.

For the third part of the study on colours, the HSL values were converted into colour labels. The colours were converted from a tuple of three numbers into a text label. The criteria for the conversion are provided in Table 4.3. The values of red, green, and blue on the hue colour wheel are 0, 120, and 240. Based on these values, the ranges were assigned to either side of the points, with 60 degrees on both sides. If the lightness value

Table 4.3: Criteria for colour labelling

	Hue (°)	Lightness (%)
White	-	≤ 20
Red	0 - 60	21 - 79
	300 - 360	
Green	60 - 180	21 - 79
Blue	180 - 300	21 - 79
Black	-	≥ 80

went below 20, it was considered white, and if it was above 80, it was considered black regardless of the hue values. The saturation values were omitted from this criteria as the effect they have on the colour is significantly lower compared to the other two. So the colour values that contained numerical information are now categorised into five levels in columns 'Colour1_labels', 'Colour2_labels', 'Colour3_labels', 'Colour4_labels', and 'Colour5_labels'. These are now subjected to classification models separately. For classification, the feature column 'Category' is encoded using the `get_dummies()` method. The target variable is the column containing the labels of the first colour. After splitting into training and testing sets, the training X and y values were used to build classification models. The classification models used are Logistic Regression, Decision Tree Classifier, Random Forest Classifier, and Support Vector Machine. The `train_test_split` and the classification models were all run at a random state of 42. The logistic regression model used 'multinomial' as the multi-class parameter and 'lgfsgs' (Limited-memory Broyden–Fletcher–Goldfarb–Shanno) as the solver. The Support Vector Machine model adopted 'ovr' as the `decision_function_shape` parameter. After the models were trained, they were evaluated over the testing sets based on their performance using the following evaluation metrics: Accuracy, Classification Report and Confusion Matrix. Based on these values, the best model was chosen and used to classify the rest of the columns.

4.4.3 Text Area Prediction from Categories

This is a very simple regression model, as we only have a single column containing continuous values that need to be predicted. Similar to the case of colour-category

prediction, the category column is encoded using a one-hot encoder before being subjected to the `train_test_split()` function along with the target variable 'Text_Area'. The models used are the same as in the case of the RGB and HSL values (Logistic Regression, Decision Tree Regression, Random Forest Regression, and Support Vector Regression). The model chosen from the evaluation metrics (MSE, MAE, and R2 score) was chosen for further analysis.

4.4.4 Object Prediction from Categories

The 'Objects' column is currently containing a list of objects that were identified from the images using the Google Vision API. To avoid missing values causing problems in the machine learning models, the dataset is filtered to remove rows that contain null values in the Objects column. After that, the list is converted into a string, and in rows that contain more than one object, the first object is retained while the others are removed. The Google Vision API sorts the objects based on their confidence score; hence, the first object with the highest confidence score is fed into the classification models. The classification models used for this problem are Multinomial Naive Bayes, Random Forest Classifier, and TensorFlow Sequential API. The category column was subjected to `TfidfVectorizer` after the `train_test_split`. The Random Forest Classifier has the `n_estimators` parameter set to 100 and a random state of 450.

For the neural network classification, the X values in the category column are tokenized using `tf.keras.preprocessing.Tokenizer()`. The values were also padded using the `tf.keras.preprocessing.sequence.pad_sequences()` method to make sure the input had a fixed length. Next, the training and testing target variables in the Objects column are converted into numerical labels. The Sequential model has three layers: Embedding, `GlobalAveragePooling1D` and Dense. The Embedding Layer is responsible for converting the input integer-coded words into dense vectors of fixed size. The 'input_dim' parameter is set to the total number of unique words, which is calculated by `len(tokenizer.index) + 1`, and the 'output_dim' is set to 50. The next layer, `GlobalAveragePooling1D` computes the average of all the vectors in the sequence. The first dense layer has 128 artificial neurons with a 'ReLU' activation, and the final dense layer has the same number of neurons as the number of unique items in the Objects column. The activation function 'softmax' used for this layer is suitable for multi-class

classification problems. The output from the Sequential model was compiled with 'adam' optimizer, 'categorical_crossentropy' loss, and accuracy as the evaluation metrics. The neural network model is then fitted with the training datasets with 5 epochs, a batch size of 32, and a validation split of 20%. The model is evaluated over the test datasets, and the accuracy is recorded and analysed.

4.5 Evaluation Metrics

Evaluation metrics are quantitative measures used to assess the predictive performance and effectiveness of a statistical or machine learning model. These provide an idea of how a model is performing and also aid in comparing different models. It is important to make sure that the metrics chosen consider the predictive capacity, generalisation potential, and overall quality of the algorithm. The adoption of metrics also depends on the specific problem domain, data type, and outcome [119].

4.5.1 Classification Metrics

1. **Confusion Matrix** summarises the performance of a machine learning model on test set data. Usually used in the case of classification models, it displays the number of true positives (TP = number of samples correctly predicted as positive), true negatives (TN = number of samples correctly predicted as negative), false positives (FP = number of samples wrongly predicted as positive), and false negatives (FN = number of samples wrongly predicted as negative). The structure of a confusion matrix is shown in Figure 4.11.
2. **F1 Score** In statistical analysis, the F-score is a measure of a test's accuracy [137]. In machine learning, the F1-score combines the precision and recall scores and measures a model's accuracy. It computes how many times the model made a

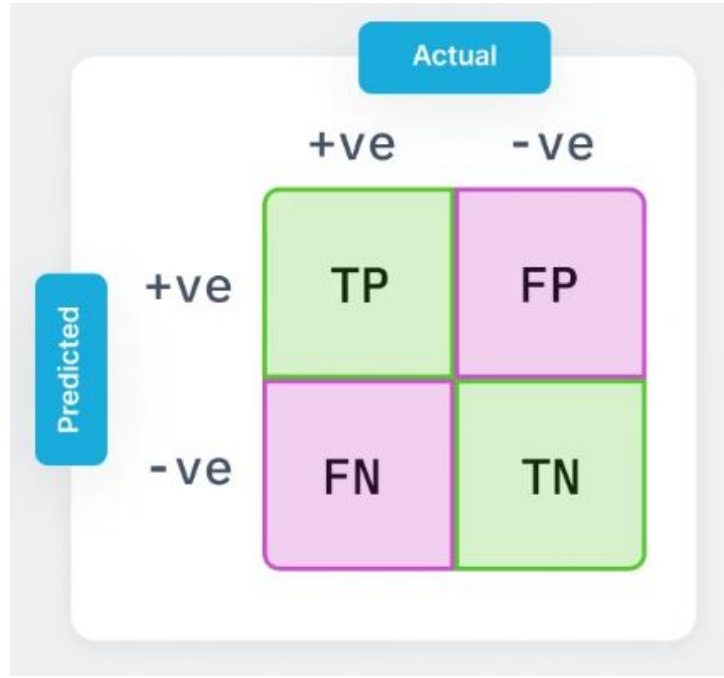


Figure 4.11: Confusion Matrix. Source [131]

correct prediction across the entire data [131]. This is how the F1 score is calculated:

$$Precision = \frac{TP}{TP + FP} \quad (4.9)$$

$$Recall = \frac{TP}{TP + FN} \quad (4.10)$$

$$F1Score = \frac{2}{\frac{1}{Precision} + \frac{1}{Recall}} \quad (4.11)$$

$$= \frac{2 \times Precision \times Recall}{Precision + Recall} \quad (4.12)$$

$$F1Score = \frac{TP}{TP + \frac{1}{2}(FP + FN)} \quad (4.13)$$

Generally, an F1 score higher than 0.7 is considered good. Low values of the F1 score could be due to imbalanced or insufficient data, inappropriate model selection, or inadequate features [71].

3. **Accuracy Score** The accuracy score in machine learning is a classification metric that is the fraction of the predictions that a model got right [46]. The higher the accuracy score, the better the predictions made by the model. The formula for the accuracy score is as follows:

$$Accuracy = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}} \quad (4.14)$$

In terms of the confusion matrix, it can be represented as:

$$Accuracy = \frac{TN + TP}{TP + FP + TN + FN} \quad (4.15)$$

4. **Classification Report** A classification report is a combination of several classification metrics. The `classification_report()` method from the `sklearn.metrics` package lists the precision, recall, F1 scores, and support scores of the model. We have already discussed the first three factors in the F1 score section. The support score is the number of actual occurrences of the class in the dataset. It stays the same for different models and diagnoses the performance evaluation process [64].

4.5.2 Regression Metrics

1. **Mean Squared Error** The Mean Squared Error (MSE) measures the amount of error in a statistical model by computing the mean squared difference between the observed and predicted variables [87]. The equation for the MSE is:

$$MSE = \frac{1}{N} \sum_{i=1}^N (y_i - \hat{y}_i)^2 \quad (4.16)$$

Every model aims to have the lowest MSE value possible, which means fewer errors in prediction. The MSE having a value of zero means there are no errors in the model.

2. **Mean Absolute Error** The Mean Absolute Error (MAE) is similar to the MSE but does not consider the direction of the value when the difference between the predictions and the original values is calculated. The formula is as follows:

$$MSE = \frac{1}{n} \sum_{j=1}^n |y_i - \hat{y}_i| \quad (4.17)$$

3. **R-Squared** The R-Squared is a statistical measure of the fit of a regression model. It ranges from 0 to 1 [36]. A R-squared value of 1 means a perfect fit of the model with not even a single error. The higher the R-Squared the better the fit of the model is, where a value higher than 0.9 means a good fit of the model. The following formula shows the calculation of the R-squared value based on the variance values. Equation in figure 8.

$$R^2 = 1 - \frac{Unexplained\ Variation}{Total\ Variation} \quad (4.18)$$

Results and Discussion

In this section, the output from each model for each part of the study is revealed.

Category Prediction from Book Titles

The two models employed for this study are Logistic Regression, which is a machine learning model, and MLPClassifier, which is a neural network learning model. A comparison of the F1 score and the accuracy is shown in Table 5.1.

Table 5.1: Evaluation Metrics of models for Category Prediction from Book Titles

	F1 Score	Accuracy
Logistic Regression	0.5625435745013103	0.5615789473684211
MLPClassifier	0.4697479614483681	0.4663157894736842

Out of the two models, the logistic regression model was chosen for category prediction from the book title as it exhibited better prediction capabilities with higher F1 score and accuracy. The chosen model was put to the test with a book title that was not present in the dataset the model was trained on, and the prediction is captured in figure 5.1.

Predicting Colours From Category

To find a suitable model that could suggest colours for a particular category, different types of colour values were tested against different machine learning models. RGB

```
# Test book title
text = ['The Ballad of Songbirds and Snakes']
# Vectorizing the title
s = (vectorizer.transform(text))
# Predicting the category
d = (title_model_lr.predict(s))

# Reverse encoding to reveal the category
le.inverse_transform(d)[0]

"Children's Books"
```

Figure 5.1: Category Prediction from Sample Book Title

values and HSL values were trained and tested on regression machine learning models, and classification models were used for colour labels. The evaluation metrics obtained for the RGB and HSL values of colour are listed in Tables 5.2 and 5.3 respectively.

Table 5.2: Evaluation Metrics for RGB Regression Models

Model	MSE	MAE	R2
Linear	6711.5280956799525	72.53719570031346	0.023250524932545424
Decision Tree	6711.527970490774	72.53719514495111	0.023250543151714376
Random Forest	6711.521561138089	72.53735923923054	0.02325147592466592
Support Vector	6808.948167996394	72.10558257941642	0.009072680015623114

Even though there were no significant differences between the performance metrics of the models, the decision tree regression model, which displayed slightly lower MSE and MAE values and a somewhat higher R-squared value, was selected for the prediction part. The RGB values predicted by the decision tree regression model for the

Table 5.3: Evaluation Metrics for HSL Regression Models

Model	MSE	MAE	R2
Linear	11278.784000486581	93.46191977802047	0.010853298567334302
Decision Tree	11278.783848781537	93.46191923825417	0.010853311871830917
Random Forest	11279.45443776487	93.47481874267409	0.010794501375872012
Support Vector	13486.110226778916	88.16086598155208	0.182728691798709

RGB values predicted for Children's Books category: (148, 140, 125), (150, 140, 125), (146, 135, 117), (151, 135, 115) and (145, 132, 115)



RGB values predicted for History category: (133, 122, 111), (141, 129, 119), (141, 127, 115), (140, 125, 113) and (139, 122, 108)

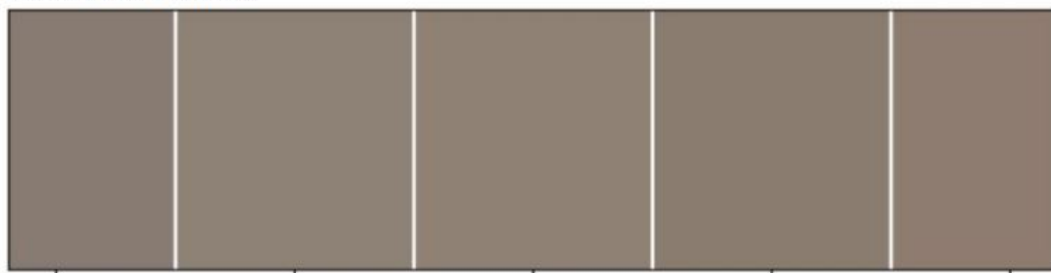


Figure 5.2: RGB Colours predicted by the selected model for two categories

“Children’s Books” and “History” categories are displayed in figure 5.2. The studies on the HSL values were conducted because the predictions for different categories resulted in very similar colours with trivial differences in values, but the models trained with HSL values had higher MSE and MAE values and lower R-squared values, and they also predicted the same shades of colour for different categories. Figure 5.3 shows the colours predicted for the “Romance” and “Self-Help” categories. The RGB (Table C.1) and HSL (Table C.2) colours predicted for all thirty categories are attached in Appendix C.

Table 5.4: Evaluation Metrics for Colour Label Classification Models

Classification Model	Accuracy
Decision Tree	0.36210526315789476
Random Forest	0.36210526315789476
Logistic	0.36210526315789476
Support Vector	0.36210526315789476

The accuracy values resulting from the classification of the colour labels are listed in

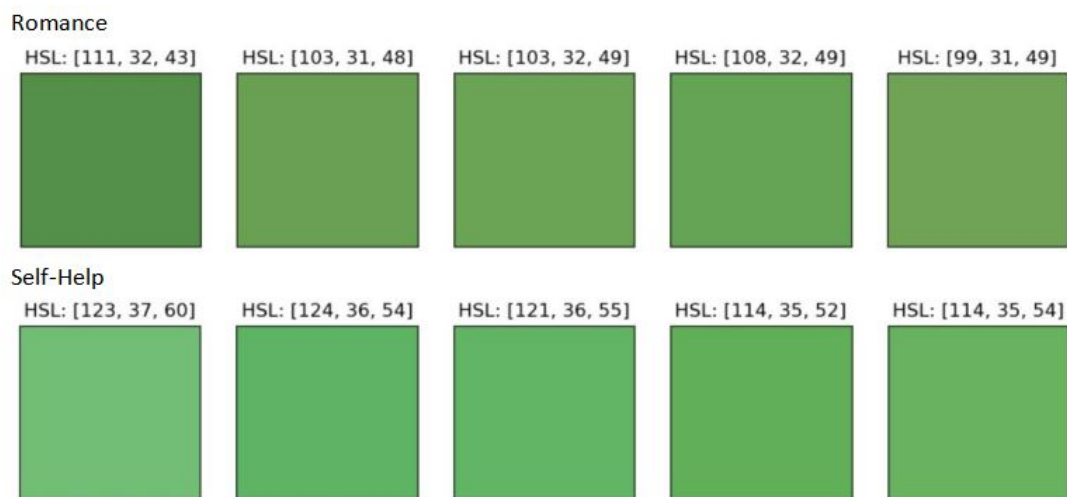


Figure 5.3: HSL Colours predicted by the selected model for two categories

Table 5.4. All four classification models resulted in the exact same accuracy, classification report and confusion matrix; hence, the Logistic Regression model was chosen for the label prediction. The classification report and confusion matrix for logistic regression model are attached in Appendix B. The prediction of colour labels for “Children’s Books and “Health, Fitness & Dieting” categories gave [‘Red’, ‘Red’, ‘Red’, ‘Red’, ‘Red’] as output. Appendix C contains the colour label predictions for all thirty categories (Table C.3).

Text Area Prediction from Categories

As the percentage text area predicted by the easyOCR from the images are numerical values, the same four regression models used to predict the RGB and HSL values are used to predict the percentage text area for a particular category. Table 5.5 lists the evaluation metrics (MSE, MAE, and R-squared) for the models, and based on these, the

Table 5.5: Evaluation Metrics for Percentage Text Area Prediction

Model	MSE	MAE	R2
Linear	241.89482165316807	11.692525177698153	0.0919687147846251
Decision Tree	241.89482960163525	11.692525221855957	0.09196868494745514
Random Forest	241.8905026004703	11.693400013107814	0.09198492776076583
Support Vector	250.96561865469792	11.448904870825885	0.05791851311872653

Random Forest Regression model was finalised. Table 5.6 shows ten categories that were predicted to have the largest percentage of text area. The predicted values for all thirty categories are included in Appendix C (Table C.4).

Table 5.6: Categories with Highest Percentage of Text Area Predicted

Category	Predicted Text Area (%)
Mystery, Thriller & Suspense	37.98
Self-Help	31.70
Parenting & Relationships	30.03
Romance	29.89
Test Preparation	29.73
Health, Fitness & Dieting	29.24
Business & Money	29.07
Humor & Entertainment	26.01
Reference	25.93
Biographies & Memoirs	25.46

Object Prediction from Categories

Two machine learning classification models (Multinomial Naive Bayes and Random Forest) and a neural network classification model (TensorFlow Sequential) are used for object prediction from categories. All the models displayed an accuracy of 44%, so one of the machine learning models, the Multinomial NB, and the TensorFlow Sequential neural network models were used for prediction. The object predictions for each category made by both models are shown in Table 5.7 and Table C.5 in Appendix C is the classification report.

Table 5.7: Object Predictions For All Categories

Category	Multinomial NB	TF Sequential
Children's Books	Person	Person
Sports & Outdoors	Person	Person

Health, Fitness & Dieting	Person	Person
Medical Books	Person	Person
Travel	Person	Person
Business & Money	Person	Person
Cookbooks, Food & Wine	Food	Food
Politics & Social Sciences	Person	Person
Crafts, Hobbies & Home	Plant	Person
Religion & Spirituality	Person	Person
Literature & Fiction	Person	Person
Science & Math	Animal	Animal
Humor & Entertainment	Person	Person
Computers & Technology	Animal	Animal
Biographies & Memoirs	Person	Person
Arts & Photography	Person	Person
Christian Books & Bibles	Person	Person
Romance	Person	Person
Comics & Graphic Novels	Person	Person
Reference	Person	Person
History	Person	Person
Teen & Young Adult	Person	Person
Parenting & Relationships	Person	Person
Law	Person	Person
Self-Help	Person	Person
Science Fiction & Fantasy	Person	Person
Test Preparation	Person	Person
Engineering & Transportation	Car	Car
Calendars	Person	Person
Mystery, Thriller & Suspense	Person	Person

Discussion

The machine learning and neural network models adopted for this study produced unsatisfactory results across all three visual elements, with the exception of the model for category prediction from the book title. With an accuracy of 56.16%, this logistic regression model had significantly better performance compared to the neural network model that was used for the same prediction. Even though the predictions were not completely accurate, the predicted category was contextually similar. Training the model with more book titles and categories can increase the predictive power and improve the performance.

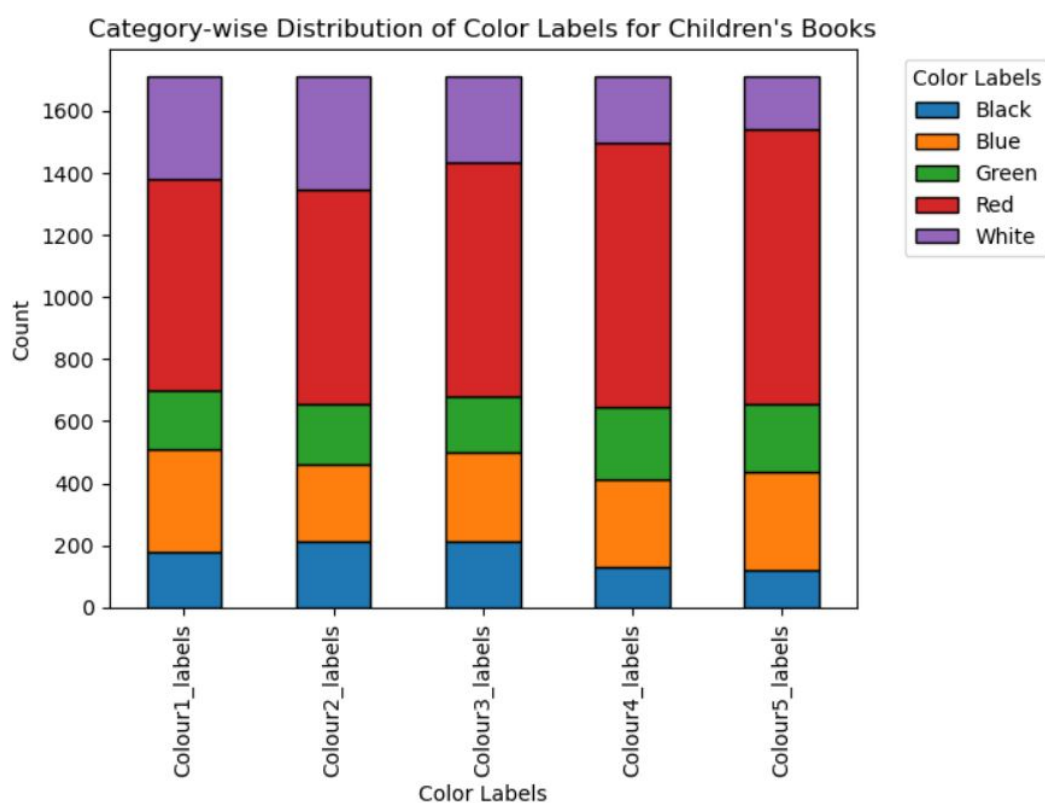


Figure 5.4: Distribution of Colour Labels for Children's Books Category

The study of categorical influences on colour was the most extensive yet under-performing study beside object prediction. While we could clearly see the biased modelling that resulted in the poor implementation of the object prediction models, the reason for the inefficiency of the colour prediction models is not clear considering the colour value feature extraction was most accurate compared to the other two extraction

models. Different modes of colour values were tested on different types of machine learning models utilising regression and classification principles but failed to produce a decent outcome. The regression models generated slightly different rgb and hsl values that gave the same colour with slight variations for any category that was fed into the model. At the same time, the classification model was heavily biased towards the 'red' colour label with the slight influence of the 'white' colour label and without a single occurrence of the other three labels in the predictions. The actual distribution of colour labels for the category "Children's Books" is shown in Figure 5.4. The explicit dominance of the colour label 'Red' has caused a bias in the data which led to faulty classification models. The distribution of another category 'Romance' in Appendix B (Figure B.3) displays similarly biased data.

Out of the three focus areas of the visual elements of the book covers, the percentage text area prediction performed better than the other two components. However, the validation of these results is difficult due to the ambiguity in the variable 'Text_Area', which is the percentage of all text identified through the OCR technology. Because of the possibility of the OCR technology not extracting all the text and the presence of additional textual elements such as blurbs or straplines beside the book title, author's name, and publisher, further analysis is required in order to obtain concrete results. As I've mentioned earlier, the object prediction is heavily biased by two labels, 'animal' and 'person', with two other labels, 'car' and 'food', also present. As part of the training, rows containing multiple identified objects were stripped to a single object with the highest confidence score. The removed objects could have been more suited to the correlation with the categorical labels. The order forced upon the identified objects by the image recognition technology imposed an additional bias that could have affected the contextual relationship between the category and the objects.

In the study of all three visual components, we can see that different machine learning models produced comparable results. This could be caused by data leakage, overfitting, or limited variability in the dataset. The problems causing the poor performance of machine learning and neural network models need to be investigated and rectified before far more advanced models and more data are brought into this research.

Conclusion

For this research, I have studied the three visual elements of book covers—colour, text, and objects—and their influence on the category to which the book belonged to. Several machine learning and neural network models were utilised to train the data, predict the category from the book title, and then predict the colour, percentage text area, and objects based on the category. While the prediction of category from the book title and the percentage text area from category did comparatively better, the prediction of colours and objects was heavily biased and failed to distinguish between different categories. One other notable point was the indifference of the data to different models. These two issues need to be investigated further before moving on to advanced models or trying a bigger dataset.

Future Scope

Even with the underperforming models, the possible influence of the category of the book on the colour, text elements, and objects cannot be completely overwritten, as this research lightly explores the correlation between the category and the visual elements of the book cover only using machine learning and neural networks. With better models and more data, these insights can be the building blocks of an artificial intelligence model that can help make creative decisions in book cover design.



Code

Link to the Dataset

The dataset and the folder containing the images are sourced from [GitHub](#).

Loading the data

```
# Importing Required Libraries
import pandas as pd

# Loading the data
df_1 = pd.read_csv("D:/DATA SCIENCE AND ITS APPLICATIONS/
DISSERTATION/Data/book30-listing-train.csv",
encoding = "ISO-8859-1")
df_2 = pd.read_csv("D:/DATA SCIENCE AND ITS APPLICATIONS/
DISSERTATION/Data/book30-listing-test.csv",
encoding = "ISO-8859-1")

# Creating column headers
df_1.columns = ['Index', 'Filename', 'URL', 'Title', 'Author',
'Category_ID', 'Category']
df_2.columns = ['Index', 'Filename', 'URL', 'Title', 'Author',
'Category_ID', 'Category']

# combining the separate data
```

```
df = pd.concat([df_1, df_2], axis=0)
# Folder containing the images
images_directory = r"D:\DATA SCIENCE AND ITS APPLICATIONS\
DISSERTATION\Data\Images\224x224"
```

Dataset Description

```
df.shape
```

Output : (56998, 7)

```
df.head()
```

Output : Figure [3.1](#)

```
df.info()
```

Output : <class 'pandas.core.frame.DataFrame'>

Int64Index: 56998 entries, 0 to 5698

Data columns (total 7 columns):

#	Column	Non-Null	Count	Dtype
—	—	—	—	
0	Index	56998	non-null	object
1	Filename	56998	non-null	object
2	URL	56998	non-null	object
3	Title	56998	non-null	object
4	Author	53165	non-null	object
5	Category_ID	56998	non-null	int64
6	Category	56998	non-null	object

dtypes: int64(1), object(6)

memory usage: 3.5+ MB

Checking Missing Values

```
df.isna().sum()
```

Output :

```
Index          0
Filename        0
URL             0
Title           0
Author         3833
Category_ID     0
Category        0
dtype: int64
```

Checking for Duplicates

```
# Check for duplicate rows
duplicate_rows = df[df.duplicated()]
# Display the duplicate rows
print("Duplicate Rows except first occurrence:")
print(duplicate_rows)
```

Output:

Duplicate Rows except first occurrence:

Empty DataFrame

Columns: [Index, Filename, URL, Title, Author, Category_ID, Category]

Index: []

```
# Checking duplicate values in Title column
# Check for duplicates in the specified column
duplicates_in_column = df['Title'].duplicated().sum()
# Display the number of duplicates in the specified column
print(f"\nNumber of duplicates in column 'Title':
{duplicates_in_column}")
```

Output :

Number of duplicates in column 'Title': 378

COLOUR EXTRACTION

```
# Importing required libraries
import os
import numpy as np
import matplotlib.image as img
from sklearn.cluster import KMeans
# Create a function to extract colors
def extract_colours(image):
    # Get the dimensions
    w, h, d = tuple(image.shape)
    # Reshape the image into an array where each row
    represents a pixel
    pixel = np.reshape(image, (w * h, d))
    # Set the desired number of colors in the image
    n_colours = 5
    # Create a KMeans model with the specified number
    of clusters and fit it to the pixels
    model = KMeans(n_clusters=n_colours,
        random_state=42).fit(pixel)
    # Get the cluster centers from the model
    colour_palette = np.uint8(model.cluster_centers_)
    # Get the RGB codes from the clusters
    rgb_colours = colour_palette.tolist()
    return rgb_colours
def apply_extraction(row):
    image_path = os.path.join(images_directory,
        row['Filename'])
    image = img.imread(image_path)
```

```
    colours = extract_colours(image)
    return colours
# Applying the colour extraction to all the rows
df['Colours'] = df.apply(apply_extraction, axis=1)
```

Code Reference : [GeeksforGeeks.org](https://www.geeksforgeeks.org/)

PERCENTAGE TEXT AREA CALCULATION

```
# Importing required libraries
import easyocr
from PIL import Image
def read_text(image):
    reader = easyocr.Reader(['en'], gpu=False)
    result = reader.readtext(image)
    return result
def calculate_area(row):
    area = 0.0
    image_path = os.path.join(images_directory, row['Filename'])
    text_output = read_text(image_path)
    img = Image.open(image_path)
    w, h = img.size
    for detection in text_output:
        top_left = tuple([int(val) for val in detection[0][0]])
        x1, y1 = top_left
        bottom_right = tuple([int(val) for val in detection[0][2]])
        x2, y2 = bottom_right
        a = (((x2-x1)*(y2-y1))/(h*w))*100
        area += a
    return round(area, 2)
df['Text_Area'] = df.apply(calculate_area, axis=1)
```

Reference : [Youtube](https://www.youtube.com/watch?v=...)

OBJECT RECOGNITION

```
# Importing required libraries
import io
from google.cloud import vision
from google.cloud.vision_v1 import types
os.environ['GOOGLE_APPLICATION_CREDENTIALS'] = r"D:\
DATA SCIENCE AND ITS APPLICATIONS\
DISSERTATION\Vision_API_credentials.json"
client = vision.ImageAnnotatorClient()
def detect_objects(row):
    image_path = os.path.join(images_directory , row['Filename'])
    with io.open(image_path , 'rb') as image_file:
        content = image_file.read()
    image = types.Image(content=content)
    objects = client.object_localization(image=image)
    obj = objects.localized_object_annotations
    objects_in_image = []
    for o in obj:
        objects_in_image.append(o.name)
    return list(set(objects_in_image))
df['Objects'] = df.apply(detect_objects , axis=1)
# Replacing the empty strings with NaN
df['Objects'] = df['Objects'].apply(lambda x:
np.nan if len(x) == 0 else x)

Reference: Youtube

# Creating a copy of the dataframe for machine learning methods
main_df = df.copy()
```

PREDICTING CATEGORY FROM BOOK TITLE

```
# Importing required libraries
from sklearn.preprocessing import LabelEncoder
import nltk
from nltk.corpus import stopwords
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn import linear_model
from sklearn import metrics
from sklearn.neural_network import MLPClassifier
books = pd.DataFrame(main_df[ ' Title ' ])
genre = pd.DataFrame(main_df[ ' Category ' ])
# Unique values in the category column
genre[ ' Category ' ].unique()
```

Output: array(["Children's Books", 'Engineering & Transportation', 'Christian Books & Bibles', 'Sports & Outdoors', 'Health, Fitness & Dieting', 'Medical Books', 'Science & Math', 'Travel', 'Business & Money', 'Cookbooks, Food & Wine', 'Politics & Social Sciences', 'Crafts, Hobbies & Home', 'Religion & Spirituality', 'Literature & Fiction', 'Humor & Entertainment', 'Law', 'Computers & Technology', 'Test Preparation', 'Biographies & Memoirs', 'Arts & Photography', 'Parenting & Relationships', 'Romance', 'History', 'Comics & Graphic Novels', 'Reference', 'Teen & Young Adult', 'Self-Help', 'Calendars', 'Science Fiction & Fantasy', 'Mystery, Thriller & Suspense'], dtype=object)

```
# Encoding the category column
feat = [ ' Category ' ]
for x in feat:
    le = LabelEncoder()
    le.fit(list(genre[x].values))
    genre[x] = le.transform(list(genre[x]))
# Category values after encoding
genre[ ' Category ' ].unique()
```



```
Output : array([ 4, 10, 5, 26, 11, 16, 23, 29, 2, 8, 19, 9, 21, 15, 13, 14, 7, 28, 1, 0, 18, 22,
12, 6, 20, 27, 25, 3, 24, 17])
```

```
# downloading stopwords
nltk.download('stopwords')
stop = list(stopwords.words('English'))
stop[:10]
# function to remove stopwords
def change(t):
    t = t.split()
    return ' '.join([(i) for (i) in t if i not in stop])
# Removig stopwords from the title column
main_df['Title'] = main_df['Title'].apply(change)
# Vectorizing the title column
vectorizer = TfidfVectorizer(min_df=2, max_features=70000,
strip_accents='unicode', lowercase=True, analyzer='word',
token_pattern=r'\w+', use_idf=True, smooth_idf=True,
sublinear_tf=True, stop_words='english')
vectors = vectorizer.fit_transform(main_df['Title'])
vectors.shape
# Splitting the data into testing and training datasets
X_train, X_test, y_train, y_test = train_test_split(vectors,
genre['Category'], test_size=0.2)
```

Model 1: Logistic Regression

```
title_model_lr = linear_model.LogisticRegression(solver='sag',
max_iter=200, random_state=450)
title_model_lr.fit(X_train, y_train)
pred = title_model_lr.predict(X_test)
print(metrics.f1_score(y_test, pred, average='macro'))
print(metrics.accuracy_score(y_test, pred))
title_model_lr
```

Output :

0.5625435745013103

0.5615789473684211

LogisticRegression(max_iter=200, random_state=450, solver='sag')

Model 2 : Neural Network MLPClassifier

```
title_model_nn = MLPClassifier(activation='logistic',
alpha=0.00003, batch_size='auto', beta_1=0.9, beta_2=0.999,
early_stopping=False, epsilon=1e-08, hidden_layer_sizes=(20,),
learning_rate='constant', learning_rate_init=0.003,
max_iter=200, momentum=0.9, nesterovs_momentum=True,
power_t=0.5, random_state=450, shuffle=True, solver='adam',
tol=0.0001, validation_fraction=0.1, verbose=False,
warm_start=False)
title_model_nn.fit(X_train, y_train)
pred = title_model_nn.predict(X_test)
print(metrics.f1_score(y_test, pred, average='macro'))
print(metrics.accuracy_score(y_test, pred))
title_model_nn
```

Output:

0.4697479614483681

0.4663157894736842

```
MLPClassifier(activation='logistic', alpha=3e-05,
hidden_layer_sizes=(20,), learning_rate_init=0.003,
random_state=450)
```

Results compared in Table 5.1 and the prediction captured in Figure 5.1

Code Reference : [Github](#)

PREDICTING COLOURS FROM CATEGORY

```
# Importing required libraries
from sklearn.preprocessing import OneHotEncoder
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error, r2_score
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.svm import SVR, SVC
import matplotlib.pyplot as plt
import colorsys

# Splitting the Colours column into 5 separate columns
main_df[['Colour1', 'Colour2', 'Colour3', 'Colour4',
'Colour5']] = pd.DataFrame(main_df['Colours'].tolist(),
index=main_df.index)
```

The five colours are sorted in descending order based on their occurrence in the image. So for each colours the red, green and blue values will be predicted. First we'll take a look at different models we can use for this prediction.

X is the category and y is the components of each color

```
# Encoding the Category values
onehotencoder = OneHotEncoder()
```

```
X = onehotencoder.fit_transform(pd.DataFrame(main_df['Category']))

# Colour1: RED component
y = main_df['Colour1'].apply(lambda x: x[0])

# Splitting into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=450)
```

Model 1: Linear Regression

```
colour_model_lr = LinearRegression()
colour_model_lr.fit(X_train, y_train)
pred = colour_model_lr.predict(X_test)
mse_lr = mean_squared_error(y_test, pred)
mae_lr = mean_absolute_error(y_test, pred)
r2_lr = r2_score(y_test, pred)
print(f'Mean Squared Error (MSE): {mse_lr}')
print(f'Mean Absolute Error (MAE): {mae_lr}')
print(f'R-squared (R2) Score: {r2_lr}')
```

Model 2: Decision Tree Regression

```
colour_model_dt = DecisionTreeRegressor()
colour_model_dt.fit(X_train, y_train)
pred = colour_model_dt.predict(X_test)
mse_dt = mean_squared_error(y_test, pred)
mae_dt = mean_absolute_error(y_test, pred)
r2_dt = r2_score(y_test, pred)
print(f'Mean Squared Error (MSE): {mse_dt}')
print(f'Mean Absolute Error (MAE): {mae_dt}')
print(f'R-squared (R2) Score: {r2_dt}')
```

Model 3: Random Forest Regression

```
colour_model_rf = RandomForestRegressor()
colour_model_rf.fit(X_train, y_train)
pred = colour_model_rf.predict(X_test)
mse_rf = mean_squared_error(y_test, pred)
mae_rf = mean_absolute_error(y_test, pred)
r2_rf = r2_score(y_test, pred)
print(f'Mean Squared Error (MSE): {mse_rf}')
print(f'Mean Absolute Error (MAE): {mae_rf}')
print(f'R-squared (R2) Score: {r2_rf}')
```

Model 4: Support Vector Regression

```
colour_model_svr = SVR()
colour_model_svr.fit(X_train, y_train)
pred = colour_model_svr.predict(X_test)
mse_svr = mean_squared_error(y_test, pred)
mae_svr = mean_absolute_error(y_test, pred)
r2_svr = r2_score(y_test, pred)
print(f'Mean Squared Error (MSE): {mse_svr}')
print(f'Mean Absolute Error (MAE): {mae_svr}')
print(f'R-squared (R2) Score: {r2_svr}')
```

The comparison between the MSE, MAE and R-Square values are captured in Table 5.2. While comparing the evaluation metrics, it is preferred to have lower MSE and MAE values and a higher R-Squared values. Even though all four models have similar values in all three metrics, model 4 seem to be least effective. So, out of the first three regression models, the decision tree regression model is chosen as the prediction model.

Category as input

```
new_X = pd.Series(["Children's Books"])
```

Encoding the category value

```
new_X_encoded = onehotencoder.transform(pd.DataFrame(new_X))
```

```
# Prediction with the chosen model
red_pred_1 = colour_model_dt.predict(new_X_encoded)
print(f'Predicted R value for {new_X.iloc[0]}:
{int(red_pred_1[0].round())}')
```

Output:

Predicted R value for Children's Books: 148

We can now train the model on the blue and green values of Colour1 and display the colour.

```
# Green values of Colour1
y = main_df['Colour1'].apply(lambda x: x[1])
# Splitting into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=450)
# Training the model
colour_model_dt = DecisionTreeRegressor()
colour_model_dt.fit(X_train, y_train)
pred = colour_model_dt.predict(X_test)
mse_dt_g = mean_squared_error(y_test, pred)
mae_dt_g = mean_absolute_error(y_test, pred)
r2_dt_g = r2_score(y_test, pred)
print(f'Mean Squared Error (MSE): {mse_dt_g}')
print(f'Mean Absolute Error (MAE): {mae_dt_g}')
print(f'R-squared (R2) Score: {r2_dt_g}')
# Prediction
new_X = pd.Series(["Children's Books"])
new_X_encoded = onehotencoder.transform(pd.DataFrame(new_X))
green_pred_1 = colour_model_dt.predict(new_X_encoded)
print(f'Predicted G value for {new_X.iloc[0]}:
{int(green_pred_1[0].round())}')
```

Output:

Mean Squared Error (MSE): 6175.7077189622805

Mean Absolute Error (MAE): 68.85760759304638

R-squared (R2) Score: 0.024146794260467397

Predicted G value for Children's Books: 140

```
# Blue values of Colour1
y = main_df['Colour1'].apply(lambda x: x[2])
# Splitting into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=450)
# Training the model
colour_model_dt = DecisionTreeRegressor()
colour_model_dt.fit(X_train, y_train)
pred = colour_model_dt.predict(X_test)
mse_dt_b = mean_squared_error(y_test, pred)
mae_dt_b = mean_absolute_error(y_test, pred)
r2_dt_b = r2_score(y_test, pred)
print(f'Mean Squared Error (MSE): {mse_dt_b}')
print(f'Mean Absolute Error (MAE): {mae_dt_b}')
print(f'R-squared (R2) Score: {r2_dt_b}')
# Prediction
new_X = pd.Series(["Children's Books"])
new_X_encoded = onehotencoder.transform(pd.DataFrame(new_X))
blue_pred_1 = colour_model_dt.predict(new_X_encoded)
print(f'Predicted B value for {new_X.iloc[0]}:
{int(blue_pred_1[0].round())}')
```

Output:

Mean Squared Error (MSE): 6137.7047889677315

Mean Absolute Error (MAE): 68.56471939639908

R-squared (R2) Score: 0.02232162652181935

Predicted B value for Children's Books: 125

```
# Displaying the colour
Colour1_prediction = (int(red_pred_1[0].round()),
int(green_pred_1[0].round()), int(blue_pred_1[0].round()))
colour1_image = [[Colour1_prediction]]
plt.imshow(colour1_image)
plt.title(f'RGB: {Colour1_prediction}')
plt.show()
```

Output :

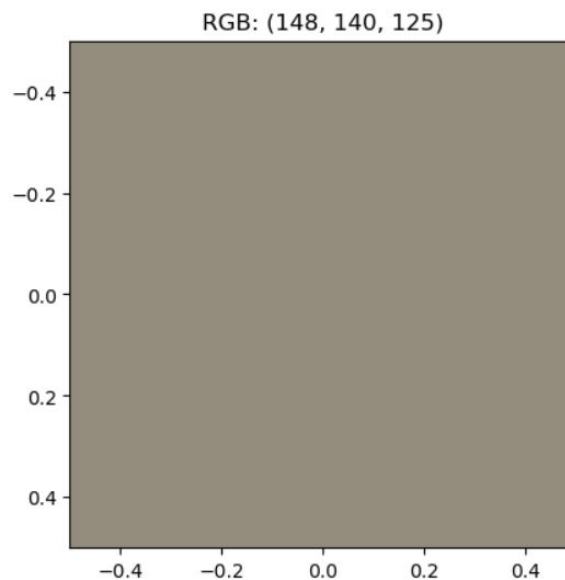


Figure A.1: Colour 1 Prediction

```
# Code to input a column and predict the RGB values for a
category based on that column
colour2_pred = []
for i in range(3):
    y = main_df['Colour2'].apply(lambda x: x[i])
    X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=450)
    colour_model_dt = DecisionTreeRegressor()
    colour_model_dt.fit(X_train, y_train)
    new_X = pd.Series(["Children's Books"])
```

```

new_X_encoded = onehotencoder.transform(pd.DataFrame(new_X))
new_prediction = colour_model_dt.predict(new_X_encoded)
colour2_pred.append(int(new_prediction[0].round()))
print(colour2_pred)

```

Output: [150, 140, 125]

```

# Code to input a Category and predict the five colours
based on each five columns
predicted_top_5_colours = []
for j in range(5):
    column_name = f"Colour{j+1}"
    colour_pred = []
    for i in range(3):
        y = main_df[column_name].apply(lambda x: x[i])
        X_train, X_test, y_train, y_test = train_test_split(X, y,
            test_size=0.2, random_state=450)
        colour_model_dt = DecisionTreeRegressor()
        colour_model_dt.fit(X_train, y_train)
        new_X = pd.Series(["Children's Books"])
        new_X_encoded = onehotencoder.transform(pd.DataFrame(new_X))
        new_prediction = colour_model_dt.predict(new_X_encoded)
        colour_pred.append(int(new_prediction[0].round()))
    predicted_top_5_colours.append(colour_pred)
predicted_top_5_colours

```

Output :

```

[[148, 140, 125],
 [150, 140, 125],
 [146, 135, 117],
 [151, 135, 115],
 [145, 132, 115]]

```

```
# Displaying all five colours
fig, ax = plt.subplots(1, 1, figsize=(8,2))
for i, colour in enumerate(predicted_top_5_colours):
    ax.axvline(x=i, color=np.array(colour)/255.0,
               linewidth=100)
ax.set_yticks([])
ax.set_xticks(range(len(predicted_top_5_colours)))
ax.set_xticklabels([])
plt.show

# Trying another category
predicted_top_5_colours = []
for j in range(5):
    column_name = f"Colour{j+1}"
    colour_pred = []
    for i in range(3):
        y = main_df[column_name].apply(lambda x: x[i])
        X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                             test_size=0.2, random_state=450)
        colour_model_dt = DecisionTreeRegressor()
        colour_model_dt.fit(X_train, y_train)
        new_X = pd.Series(["History"])
        new_X_encoded = onehotencoder.transform(pd.DataFrame(new_X))
        new_prediction = colour_model_dt.predict(new_X_encoded)
        colour_pred.append(int(new_prediction[0].round()))
    predicted_top_5_colours.append(colour_pred)
print(predicted_top_5_colours)
fig, ax = plt.subplots(1, 1, figsize=(8,2))
for i, colour in enumerate(predicted_top_5_colours):
    ax.axvline(x=i, color=np.array(colour)/255.0,
               linewidth=100)
```

```

ax.set_yticks([])
ax.set_xticks(range(len(predicted_top_5_colours)))
ax.set_xticklabels([])
plt.show

```

The colours predicted for categories "Children's Books" and "History" are shown in Figure 5.2.

```

# Getting all the unique categories into a list
categories = main_df['Category'].unique().tolist()
# Predicting colours for all thirty categories
for category in categories:
    predicted_top_5_colours = []
    for j in range(5):
        column_name = f"Colour{j+1}"
        colour_pred = []
        for i in range(3):
            y = main_df[column_name].apply(lambda x: x[i])
            X_train, X_test, y_train, y_test = train_test_split(X,
            y, test_size=0.2, random_state=450)
            colour_model_dt = DecisionTreeRegressor()
            colour_model_dt.fit(X_train, y_train)
            new_X = pd.Series([category])
            new_X_encoded = onehotencoder.transform(
            pd.DataFrame(new_X))
            new_prediction = colour_model_dt.predict(new_X_encoded)
            colour_pred.append(int(new_prediction[0].round()))
        predicted_top_5_colours.append(colour_pred)
    print(category)
    print(predicted_top_5_colours)
    fig, ax = plt.subplots(1, 1, figsize=(8,2))
    for i, colour in enumerate(predicted_top_5_colours):
        ax.axvline(x=i, color=np.array(colour)/255.0,
        linewidth=100)

```

```

ax.set_yticks([])
ax.set_xticks(range(len(predicted_top_5_colours)))
ax.set_xticklabels([])
plt.show

```

The output is consolidated in Table C.1 in Appendix C. Regardless of the category, the output is very similar in colour. We can now try another form for colour representation.

HSL (Hue, Saturation, Lightness) values of colours are a different way of representing colours. We will now convert the RGB colours into their HSL values and train the models with those values.

```

# Function to convert RGB to HSL
def rgb_to_hsl(rgb):
    r, g, b = rgb[0] / 255.0, rgb[1] / 255.0,
    rgb[2] / 255.0

    cmax = max(r, g, b)
    cmin = min(r, g, b)
    delta = cmax - cmin

    # Calculate Hue
    if delta == 0:
        h = 0
    elif cmax == r:
        h = 60 * (((g - b) / delta) % 6)
    elif cmax == g:
        h = 60 * (((b - r) / delta) + 2)
    elif cmax == b:
        h = 60 * (((r - g) / delta) + 4)

    # Calculate Lightness
    l = (cmax + cmin) / 2

    # Calculate Saturation

```

```

    s = delta / (1 - abs(2 * l - 1)) if
    delta != 0 else 0

    return int(h), round(s * 100, 1),
    round(l * 100, 1)

# Applying to all the five colour columns
main_df['hsl_1'] = main_df['Colour1'].apply(rgb_to_hsl)
main_df['hsl_2'] = main_df['Colour2'].apply(rgb_to_hsl)
main_df['hsl_3'] = main_df['Colour3'].apply(rgb_to_hsl)
main_df['hsl_4'] = main_df['Colour4'].apply(rgb_to_hsl)
main_df['hsl_5'] = main_df['Colour5'].apply(rgb_to_hsl)

# Hue values of Colour1
y = main_df['hsl_1'].apply(lambda x: x[0])

# Splitting into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=450)

```

Model 1: Linear Regression

```

hsl_model_lr = LinearRegression()
hsl_model_lr.fit(X_train, y_train)
pred = hsl_model_lr.predict(X_test)
mse_lr = mean_squared_error(y_test, pred)
mae_lr = mean_absolute_error(y_test, pred)
r2_lr = r2_score(y_test, pred)
print(f'Mean Squared Error (MSE): {mse_lr}')
print(f'Mean Absolute Error (MAE): {mae_lr}')
print(f'R-squared (R2) Score: {r2_lr}')

```

Model 2: Decision Tree Regression

```
hsl_model_dt = DecisionTreeRegressor()  
hsl_model_dt.fit(X_train, y_train)  
pred = hsl_model_dt.predict(X_test)  
mse_dt = mean_squared_error(y_test, pred)  
mae_dt = mean_absolute_error(y_test, pred)  
r2_dt = r2_score(y_test, pred)  
print(f'Mean Squared Error (MSE): {mse_dt}')print(f'Mean Absolute Error (MAE): {mae_dt}')print(f'R-squared (R2) Score: {r2_dt}')
```

Model 3: Random Forest Regression

```
hsl_model_rf = RandomForestRegressor()  
hsl_model_rf.fit(X_train, y_train)  
pred = hsl_model_rf.predict(X_test)  
mse_rf = mean_squared_error(y_test, pred)  
mae_rf = mean_absolute_error(y_test, pred)  
r2_rf = r2_score(y_test, pred)  
print(f'Mean Squared Error (MSE): {mse_rf}')print(f'Mean Absolute Error (MAE): {mae_rf}')print(f'R-squared (R2) Score: {r2_rf}')
```

Model 4: Support Vector Regression

```
hsl_model_svr = SVR()  
hsl_model_svr.fit(X_train, y_train)  
pred = hsl_model_svr.predict(X_test)  
mse_svr = mean_squared_error(y_test, pred)  
mae_svr = mean_absolute_error(y_test, pred)  
r2_svr = r2_score(y_test, pred)
```

```
print(f'Mean Squared Error (MSE): {mse_svr} ')
print(f'Mean Absolute Error (MAE): {mae_svr} ')
print(f'R-squared (R2) Score: {r2_svr} ')
```

The results are compared in Table 5.3. Out of the four models, the decision tree regressor has the lowest MSE and MAE values and highest R-squared values. So we can use this model to predict the values based on the category.

```
predicted_top_5_colours = []
for j in range(5):
    column_name = f"hsl_{j+1}"
    colour_pred = []
    for i in range(3):
        y = main_df[column_name].apply(lambda x: x[i])
        X_train, X_test, y_train, y_test = train_test_split(X,
            y, test_size=0.2, random_state=450)
        hsl_model_dt = DecisionTreeRegressor()
        hsl_model_dt.fit(X_train, y_train)
        new_X = pd.Series(["History"])
        new_X_encoded = onehotencoder.transform(
            pd.DataFrame(new_X))
        new_prediction = hsl_model_dt.predict(new_X_encoded)
        colour_pred.append(int(new_prediction[0].round()))
    predicted_top_5_colours.append(colour_pred)
predicted_top_5_colours
```

Output:

```
[[99, 30, 48], [96, 27, 51], [95, 29, 50], [93, 28, 50],
[94, 28, 49]]
```

```
def convert_colour(h, s, l):
    r, g, b = colorsys.hls_to_rgb(h, l, s)
    return int(r * 255), int(g * 255), int(b * 255)
```

```

fig, ax = plt.subplots(1, len(predicted_top_5_colours),
figsize=(12,2))
for i, hsl in enumerate(predicted_top_5_colours):
    rgb = convert_colour(hsl[0]/360, hsl[1]/100, hsl[2]/100)
    ax[i].add_patch(plt.Rectangle((0, 0), 1, 1,
    color=[comp/255 for comp in rgb]))
    ax[i].set_title(f'HSL: {hsl}')
    ax[i].set_xticks([])
    ax[i].set_yticks([])
plt.show()

# Trying another category
predicted_top_5_colours = []
for j in range(5):
    column_name = f"hsl_{j+1}"
    colour_pred = []
    for i in range(3):
        y = main_df[column_name].apply(lambda x: x[i])
        X_train, X_test, y_train, y_test = train_test_split(X,
        y, test_size=0.2, random_state=450)
        hsl_model_dt = DecisionTreeRegressor()
        hsl_model_dt.fit(X_train, y_train)
        new_X = pd.Series(["Romance"])
        new_X_encoded = onehotencoder.transform(
        pd.DataFrame(new_X))
        new_prediction = hsl_model_dt.predict(new_X_encoded)
        colour_pred.append(int(new_prediction[0].round()))
    predicted_top_5_colours.append(colour_pred)
fig, ax = plt.subplots(1, len(predicted_top_5_colours),
figsize=(12,2))
for i, hsl in enumerate(predicted_top_5_colours):
    rgb = convert_colour(hsl[0]/360, hsl[1]/100,

```



```

    hsl[2]/100)
    ax[i].add_patch(plt.Rectangle((0, 0), 1, 1,
    color=[comp/255 for comp in rgb]))
    ax[i].set_title(f'HSL: {hsl}')
    ax[i].set_xticks([])
    ax[i].set_yticks([])
plt.show()

```

The results of the HSL value predictions for these two categories are captured in Figure 5.3.

```

# Predicting HSL values for all 30 categories
for category in categories:
    print(category)
    predicted_top_5_colours = []
    for j in range(5):
        column_name = f"hsl_{j+1}"
        colour_pred = []
        for i in range(3):
            y = main_df[column_name].apply(lambda x: x[i])
            X_train, X_test, y_train, y_test = train_test_split(X,
            y, test_size=0.2, random_state=450)
            hsl_model_dt = DecisionTreeRegressor()
            hsl_model_dt.fit(X_train, y_train)
            new_X = pd.Series([category])
            new_X_encoded = onehotencoder.transform(
            pd.DataFrame(new_X))
            new_prediction = hsl_model_dt.predict(
            new_X_encoded)
            colour_pred.append(int(new_prediction[0].round()))
        predicted_top_5_colours.append(colour_pred)
    fig, ax = plt.subplots(1, len(predicted_top_5_colours),
    figsize=(12,2))
    for i, hsl in enumerate(predicted_top_5_colours):

```

```

    rgb = convert_colour(hsl[0]/360, hsl[1]/100,
    hsl[2]/100)
    ax[i].add_patch(plt.Rectangle((0, 0), 1, 1,
    color=[comp/255 for comp in rgb]))
    ax[i].set_title(f'HSL: {hsl}')
    ax[i].set_xticks([])
    ax[i].set_yticks([])
plt.show()

```

The HSL value predictions for all thirty categories are compiled in Table C.2. This is also showing very similar colours for different categories. Since regression models are not working properly we will try some classification models. But before that we will convert these HSL values into categorical values. Each entry is assigned a dominant colour based on the hue and lightness. The saturation is omitted for this classification. Table 4.3 contains the criteria under which the colours were categorised.

Reference: [w3schools](#)

```

# Converting into labels
def label_value(values):
    hue, _, lightness = values

    # Lightness Check
    if lightness >= 0 and lightness <= 20:
        return "Black"
    elif lightness >= 80 and lightness <= 100:
        return "White"

    # Hue Check
    if (hue >= 0 and hue <= 60) or (hue >= 300
    and hue <= 360):
        return "Red"
    elif hue >= 60 and hue < 180:
        return "Green"
    elif hue >= 180 and hue < 300:

```

```

        return "Blue"

# Apply the label_value function to the 'colors' column
main_df['Colour1_labels'] = main_df['hsl_1'].apply(label_value)
main_df['Colour2_labels'] = main_df['hsl_2'].apply(label_value)
main_df['Colour3_labels'] = main_df['hsl_3'].apply(label_value)
main_df['Colour4_labels'] = main_df['hsl_4'].apply(label_value)
main_df['Colour5_labels'] = main_df['hsl_5'].apply(label_value)

# Encoding the Category values
onehotencoder = OneHotEncoder()
X = onehotencoder.fit_transform(pd.DataFrame(main_df['Category']))

y = main_df['Colour1_labels']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    test_size=0.2, random_state=450)

```

Model 1: Decision Tree Classifier

```

label_model_dt = DecisionTreeClassifier()
label_model_dt.fit(X_train, y_train)
pred = label_model_dt.predict(X_test)
print("Accuracy:", accuracy_score(y_test, pred))
print("Classification Report:\n",
      classification_report(y_test, pred))
print("Confusion Matrix:\n",
      confusion_matrix(y_test, pred))

```

Model 2: Random Forest Classifier

```

label_model_rf = RandomForestClassifier()

```

```
label_model_rf.fit(X_train, y_train)
pred = label_model_rf.predict(X_test)
print("Accuracy:", accuracy_score(y_test, pred))
print("Classification Report:\n",
      classification_report(y_test, pred))
print("Confusion Matrix:\n",
      confusion_matrix(y_test, pred))
```

Model 3: Logistic Regression

```
label_model_lr = LogisticRegression(multi_class='multinomial',
solver='lbfgs', random_state=450)
label_model_lr.fit(X_train, y_train)
pred = label_model_lr.predict(X_test)
print("Accuracy:", accuracy_score(y_test, pred))
print("Classification Report:\n",
      classification_report(y_test, pred))
print("Confusion Matrix:\n",
      confusion_matrix(y_test, pred))
```

Model 4: Support Vector Machines

```
label_model_svm = SVC(decision_function_shape='ovr',
random_state=450)
label_model_svm.fit(X_train, y_train)
pred = label_model_svm.predict(X_test)
print("Accuracy:", accuracy_score(y_test, pred))
print("Classification Report:\n",
      classification_report(y_test, pred))
print("Confusion Matrix:\n",
      confusion_matrix(y_test, pred))
```

The evaluation metrics are compared in Table 5.4 and the confusion matrix and classification report is attached in Appendix B (Figures B.1 & B.2). Since all the models have the exact same accuracy, we'll use the logistic regression model for prediction.

```
new_X = pd.Series(["Children's Books"])
new_X_encoded = onehotencoder.transform(pd.DataFrame(new_X))
label1_pred = label_model_lr.predict(new_X_encoded)
print(f'Predicted label: {label1_pred[0]}')
```

Output:

Predicted label: Red

```
# Predicting colours from all five columns
predicted_labels = []
for j in range(5):
    column_name = f"Colour{j+1}_labels"
    y = main_df[column_name]
    X_train, X_test, y_train, y_test = train_test_split(X,
    y, test_size=0.2, random_state=450)
    label_model_lr = LogisticRegression(multi_class='multinomial',
    solver='lbfgs', random_state=450)
    label_model_lr.fit(X_train, y_train)
    new_X = pd.Series(["Children's Books"])
    new_X_encoded = onehotencoder.transform(pd.DataFrame(new_X))
    new_prediction = label_model_lr.predict(new_X_encoded)
    predicted_labels.append(new_prediction[0])
predicted_labels
```

Output: ['Red', 'Red', 'Red', 'Red', 'Red']

```
# Trying another category
predicted_labels = []
for j in range(5):
```

```

column_name = f"Colour{j+1}_labels"
y = main_df[column_name]
X_train, X_test, y_train, y_test = train_test_split(X,
y, test_size=0.2, random_state=450)
label_model_lr = LogisticRegression(multi_class='multinomial',
solver='lbfgs', random_state=450)
label_model_lr.fit(X_train, y_train)
new_X = pd.Series(["Health, Fitness & Dieting"])
new_X_encoded = onehotencoder.transform(pd.DataFrame(new_X))
new_prediction = label_model_lr.predict(new_X_encoded)
predicted_labels.append(new_prediction[0])
predicted_labels

```

Output: ['Red', 'Red', 'Red', 'Red', 'Red']

Both the models output the same label(red). We'll compare the output with the actual distribution of the labels in these two categories.

```

category_ = 'Health, Fitness & Dieting'
category_subset = main_df[main_df['Category'] == category_]

# Count the occurrences of each color label in the subset
color_counts = category_subset[['Colour1_labels',
'Colour2_labels', 'Colour3_labels', 'Colour4_labels',
'Colour5_labels']].apply(pd.value_counts).fillna(0)

# Plot the distribution of color labels for the
specified category
plt.figure(figsize=(10, 6))
color_counts.T.plot(kind='bar', stacked=True,
edgecolor='black')

```

```
plt.title(f'Category-wise Distribution of Color Labels for
{category_of_interest}')
plt.xlabel('Color Labels')
plt.ylabel('Count')
plt.legend(title='Color Labels', bbox_to_anchor=(1.05, 1),
loc='upper left')
plt.show()

category_ = "Children's Books"
category_subset = main_df[main_df['Category'] == category_]

# Count the occurrences of each color label in the subset
color_counts = category_subset[['Colour1_labels',
'Colour2_labels', 'Colour3_labels', 'Colour4_labels',
'Colour5_labels']].apply(pd.value_counts).fillna(0)

# Plot the distribution of color labels for the
specified category
plt.figure(figsize=(10, 6))
color_counts.T.plot(kind='bar', stacked=True,
edgecolor='black')
plt.title(f'Category-wise Distribution of Color Labels
for {category_}')
plt.xlabel('Color Labels')
plt.ylabel('Count')
plt.legend(title='Color Labels', bbox_to_anchor=(1.05, 1),
loc='upper left')
plt.show()
```

The distribution of labels are shown in Figures [5.4](#) and [B.3](#).

```
# Predicting colour labels for all thirty categories
for category in categories:
    predicted_labels = []
```

```

for j in range(5):
    column_name = f"Colour{j+1}_labels"
    y = main_df[column_name]
    X_train, X_test, y_train, y_test = train_test_split(X,
    y, test_size=0.2, random_state=450)
    label_model_lr = LogisticRegression(multi_class='multinomial',
    solver='lbfgs', random_state=450)
    label_model_lr.fit(X_train, y_train)
    new_X = pd.Series([category])
    new_X_encoded = onehotencoder.transform(pd.DataFrame(new_X))
    new_prediction = label_model_lr.predict(new_X_encoded)
    predicted_labels.append(new_prediction[0])
print(f"{category}: {predicted_labels}")

```

The predictions are listed in Table [C.3](#).

PREDICTING TEXT AREA FROM CATEGORY

Here the X value remains the same. y is the percentage area of text detected by easyOCR.

```
y = main_df['Text_Area']
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=450)
```

Model 1: Linear Regression

```

area_model_lr = LinearRegression()
area_model_lr.fit(X_train, y_train)
pred = area_model_lr.predict(X_test)
mse_lr = mean_squared_error(y_test, pred)
mae_lr = mean_absolute_error(y_test, pred)

```



```
r2_lr = r2_score(y_test , pred)
print(f'Mean Squared Error (MSE): {mse_lr} ')
print(f'Mean Absolute Error (MAE): {mae_lr} ')
print(f'R-squared (R2) Score: {r2_lr} ')
```

Model 2: Decision Tree Regression

```
area_model_dt = DecisionTreeRegressor()
area_model_dt.fit(X_train , y_train)
pred = area_model_dt.predict(X_test)
mse_dt = mean_squared_error(y_test , pred)
mae_dt = mean_absolute_error(y_test , pred)
r2_dt = r2_score(y_test , pred)
print(f'Mean Squared Error (MSE): {mse_dt} ')
print(f'Mean Absolute Error (MAE): {mae_dt} ')
print(f'R-squared (R2) Score: {r2_dt} ')
```

Model 3: Random Forest Regression

```
area_model_rf = RandomForestRegressor()
area_model_rf.fit(X_train , y_train)
pred = area_model_rf.predict(X_test)
mse_rf = mean_squared_error(y_test , pred)
mae_rf = mean_absolute_error(y_test , pred)
r2_rf = r2_score(y_test , pred)
print(f'Mean Squared Error (MSE): {mse_rf} ')
print(f'Mean Absolute Error (MAE): {mae_rf} ')
print(f'R-squared (R2) Score: {r2_rf} ')
```

Model 4: Support Vector Regression

```
area_model_svr = SVR()
area_model_svr.fit(X_train, y_train)
pred = area_model_svr.predict(X_test)
mse_svr = mean_squared_error(y_test, pred)
mae_svr = mean_absolute_error(y_test, pred)
r2_svr = r2_score(y_test, pred)
print(f'Mean Squared Error (MSE): {mse_svr}')
print(f'Mean Absolute Error (MAE): {mae_svr}')
print(f'R-squared (R2) Score: {r2_svr}')
```

The models are compared in Table 5.5. Out of the three better performing models, we'll choose the Random Forest regression model for the text area prediction.

```
new_X = pd.Series(["Comics & Graphic Novels"])
new_X_encoded = onehotencoder.transform(pd.DataFrame(new_X))
new_prediction = area_model_rf.predict(new_X_encoded)
print(f'Predicted text area for {new_X.iloc[0]}:
{new_prediction[0].round(2)}')
```

Output:

Predicted text area for Comics & Graphic Novels: 16.91

```
# Testing another category
new_X = pd.Series(["Health, Fitness & Dieting"])
new_X_encoded = onehotencoder.transform(pd.DataFrame(new_X))
new_prediction = area_model_rf.predict(new_X_encoded)
print(f'Predicted text area for {new_X.iloc[0]}:
{new_prediction[0].round(2)}')
```

Output:

Predicted text area for Health, Fitness & Dieting: 29.24

```

# Predicting percentage text area for all thirty categories.
category_area = []
for category in categories:
    new_X = pd.Series([category])
    new_X_encoded = onehotencoder.transform(pd.DataFrame(new_X))
    new_prediction = area_model_rf.predict(new_X_encoded)
    category_area.append(new_prediction[0].round(2))
category_area

area_df = pd.DataFrame({
    'Category': categories,
    'Predicted Text Area': category_area
})
area_df

# sorting dataframe based on the predicted text area
area_df = area_df.sort_values(by='Predicted Text Area',
ascending=False).reset_index(drop=True)
area_df

```

The dataframe containing the predicted text areas for all thirty categories is attached in Appendix C (Table C.4) while Table 5.6 contains the categories with highest predicted area.

PREDICTING OBJECTS FROM CATEGORY

```

# Importing required libraries
from sklearn.naive_bayes import MultinomialNB
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Embedding
from tensorflow.keras.layers import GlobalAveragePooling1D

```

```
# Filtering out rows that does not contain any
identified objects
main_df = main_df[main_df['Objects'].apply(lambda x:
isininstance(x, list) and any(pd.notna(i) for i in x)))]

# Convert list values to strings
main_df['Objects'] = main_df['Objects'].apply(
lambda x: ', '.join(x))
```

For rows containing more than one objects, only the first one is kept and others are removed as this has the highest confidence score by the vision API.

```
# Keeping only the first object identified as it had
the most score
main_df['Objects'] = main_df['Objects'].apply(
lambda x: x.split(',')[0])
```

Model 1: Multinomial Naive Bayes Classification

```
# Split the data into training and testing sets
train_data, test_data = train_test_split(main_df,
test_size=0.2, random_state=450)
# Extract features using TF-IDF vectorizer
tfidf_vectorizer = TfidfVectorizer(max_features=5000,
stop_words='english')
X_train = tfidf_vectorizer.fit_transform(train_data['Category'])
X_test = tfidf_vectorizer.transform(test_data['Category'])
# Use Multinomial Naive Bayes as the classifier
object_model_nb = MultinomialNB()
object_model_nb.fit(X_train, train_data['Objects'])
# Make predictions on the test set
```

```
predictions = object_model_nb.predict(X_test)
# Evaluate the model
accuracy = accuracy_score(test_data['Objects'], predictions)
print(f'Accuracy: {accuracy:.2f}')
# Display classification report
print('\nClassification Report:\n',
      classification_report(test_data['Objects'], predictions))
```

Output : Accuracy: 0.44

Model 2: Random Forest Classifier

```
# Use Random Forest as the classifier
object_model_rf = RandomForestClassifier(n_estimators=100,
random_state=450)
object_model_rf.fit(X_train, train_data['Objects'])
# Make predictions on the test set
pred = object_model_rf.predict(X_test)
# Evaluate the Random Forest model
rf_accuracy = accuracy_score(test_data['Objects'], pred)
print(f'Random Forest Accuracy: {rf_accuracy:.2f}')
# Display classification report
print('\nRandom Forest Classification Report:\n',
      classification_report(test_data['Objects'], pred))
```

Output: Random Forest Accuracy: 0.44

Model 3: TensorFlow Sequential

```
# Tokenize and pad sequences
tokenizer = tf.keras.preprocessing.text.Tokenizer()
tokenizer.fit_on_texts(train_data['Category'])
```

```
X_train = tokenizer.texts_to_sequences(train_data['Category'])
X_test = tokenizer.texts_to_sequences(test_data['Category'])
X_train = tf.keras.preprocessing.sequence.pad_sequences(X_train)
X_test = tf.keras.preprocessing.sequence.pad_sequences(X_test,
maxlen=X_train.shape[1])

# Convert the 'Object' column to numerical labels
label_mapping = {label: idx for idx, label in
enumerate(main_df['Objects'].unique())}
train_data['Objects'] = train_data['Objects'].map(label_mapping)
test_data['Objects'] = test_data['Objects'].map(label_mapping)
# Define the model
embedding_dim = 50
model_tf = Sequential([
    Embedding(input_dim=len(tokenizer.word_index) + 1,
    output_dim=embedding_dim, input_length=X_train.shape[1]),
    GlobalAveragePooling1D(),
    Dense(128, activation='relu'),
    Dense(len(main_df['Objects'].unique()),
    activation='softmax')
])
# Compile the model
model_tf.compile(optimizer='adam',
loss='sparse_categorical_crossentropy',
metrics=['accuracy'])
# Train the model
model_tf.fit(X_train, train_data['Objects'], epochs=5,
batch_size=32, validation_split=0.2)
# Evaluate the model on the test set
_, nn_accuracy = model_tf.evaluate(X_test,
test_data['Objects'])
print(f'Neural Network Accuracy: {nn_accuracy:.2f}')
```

Output: Neural Network Accuracy: 0.44

Since all three models have the same accuracy, we'll test prediction from two models: MultinomialNB and TensorFlow Sequential.

```
category_input = ["Humour & Entertainment"]
# Transforming the input category using the same TF-IDF
vectorizer used during training
category_input_transformed = tfidf_vectorizer.transform(
category_input)
# Using the Naive Bayes model to predict the object
nb_prediction = object_model_nb.predict(
category_input_transformed)
print(f"Naive Bayes Prediction: {nb_prediction[0]}")
```

Output: Naive Bayes Prediction: Person

```
category_input = ["Humour & Entertainment"]
# Tokenizing and padding the input category
category_input_tokenized = tokenizer.texts_to_sequences(
category_input)
category_input_padded = tf.keras.preprocessing.sequence.pad_sequences(
category_input_tokenized, maxlen=X_train.shape[1])
# Using the Neural Network model to predict the object
nn_prediction = model_tf.predict(category_input_padded)
predicted_class_index = tf.argmax(nn_prediction,
axis=1).numpy()[0]
predicted_object = list(label_mapping.keys())
[list(label_mapping.values()).index(predicted_class_index)]
print(f"Neural Network Prediction: {predicted_object}")
```

Output: Neural Network Prediction: Person

```
# Predicting objects for all the categories by both
naive bayes and tensorflow sequential model
nb_objects = []
tf_objects = []
for category in categories:
    category_input = [category]
    category_input_transformed =
    tfidf_vectorizer.transform(category_input)
    nb_prediction = object_model_nb.predict(
    category_input_transformed)
    nb_objects.append(nb_prediction[0])
    category_input_tokenized =
    tokenizer.texts_to_sequences(category_input)
    category_input_padded =
    tf.keras.preprocessing.sequence.pad_sequences(
    category_input_tokenized, maxlen=X_train.shape[1])
    nn_prediction = model_tf.predict(category_input_padded)
    predicted_class_index = tf.argmax(nn_prediction,
    axis=1).numpy()[0]
    predicted_object = list(label_mapping.keys())
    [list(label_mapping.values()).index(predicted_class_index)]
    tf_objects.append(predicted_object)
object_df = pd.DataFrame({
    'Category': categories,
    'MultinomialNB Predictions': nb_objects,
    'TF Sequential Predictions': tf_objects
})
```

The predictions made by both models are listed in Table 5.7 and the classification report generated for the Multinomial Naive Bayes classification is in Table C.5.



Additional Figures

Classification Report:				
	precision	recall	f1-score	support
Black	0.00	0.00	0.00	1940
Blue	0.00	0.00	0.00	1979
Green	0.00	0.00	0.00	910
Red	0.36	0.97	0.53	4129
White	0.35	0.05	0.09	2442
accuracy			0.36	11400
macro avg	0.14	0.20	0.12	11400
weighted avg	0.21	0.36	0.21	11400

Figure B.1: Classification Report - Logistic Regression

```
Confusion Matrix:
[[ 0  0  0 1914  26]
 [ 0  0  0 1923  56]
 [ 0  0  0  887  23]
 [ 0  0  0 4004 125]
 [ 0  0  0 2318 124]]
```

Figure B.2: Confusion Matrix - Logistic Regression

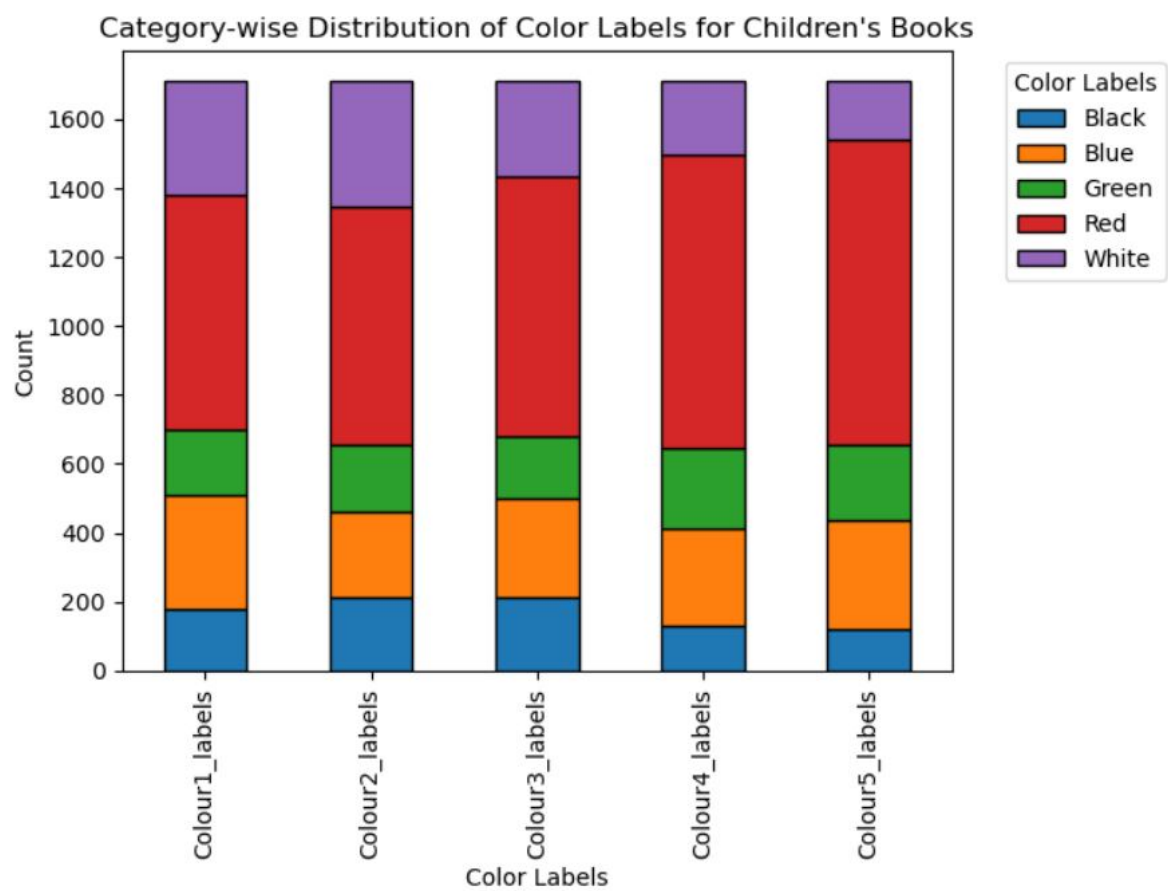

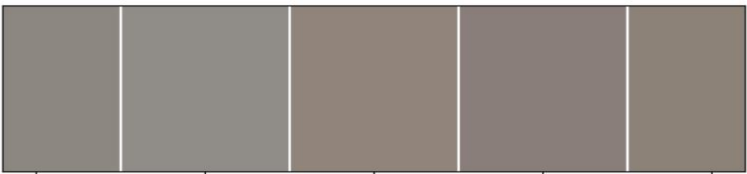
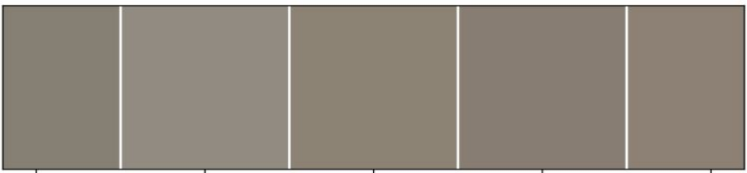


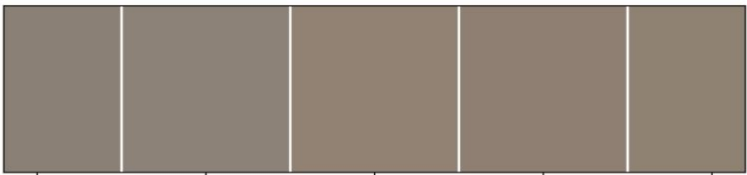


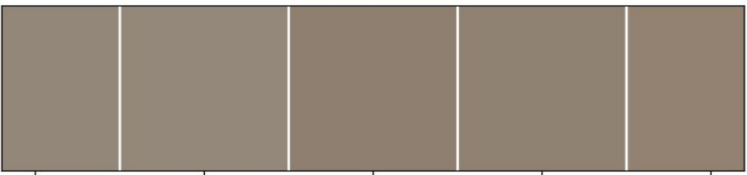




Figure B.3: Distribution of Colour Labels for Romance Category

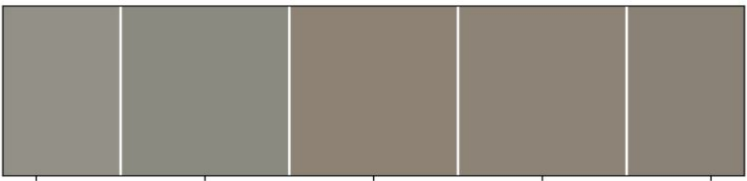
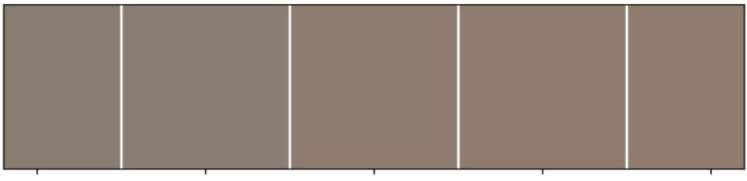


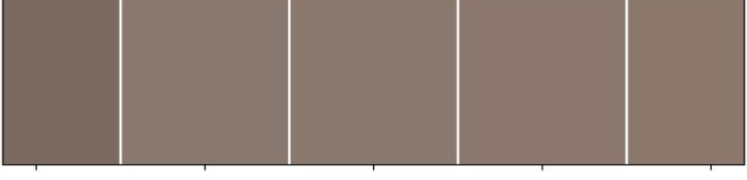

Additional Tables

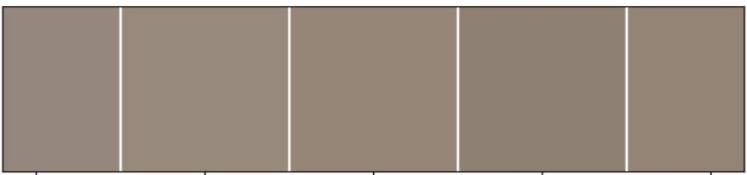




Table C.1: RGB Colour Generated For All Categories

Category	Predicted Colour
Children’s Books	<div> <div>(150, 142, 127), (150, 139, 123), (146, 134, 116), (149, 133, 114), (146, 133, 116)</div> <div> <div></div> <div></div> <div></div> <div></div> <div></div> </div> </div>
Sports & Outdoors	<div> <div>(132, 123, 117), (141, 134, 124), (142, 129, 117), (135, 123, 113), (141, 126, 114)</div> <div> <div></div> <div></div> <div></div> <div></div> <div></div> </div> </div>
Health, Fitness & Dieting	<div> <div>(161, 149, 137), (151, 137, 127), (148, 131, 119), (150, 131, 117), (148, 132, 119)</div> </div>

	
Medical Books	<p>(141, 135, 130), (144, 140, 135), (145, 133, 123), (138, 126, 122), (141, 130, 120)</p> 
Travel	<p>(134, 127, 115), (145, 139, 129), (141, 131, 117), (135, 125, 114), (140, 129, 116)</p> 
Business & Money	<p>(143, 138, 129), (139, 132, 123), (145, 133, 121), (133, 125, 115), (142, 130, 118)</p> 
Cookbooks, Food & Wine	<p>(160, 142, 121), (157, 137, 117), (156, 132, 109), (153, 128, 105), (158, 133, 108)</p> 
Politics & Social Sciences	<p>(139, 128, 118), (141, 130, 120), (145, 130, 116), (143, 127, 114), (144, 130, 115)</p> 

Crafts, Hobbies & Home	<p>(146, 135, 120), (148, 136, 123), (143, 127, 112), (144, 130, 115), (147, 130, 113)</p> 
Religion & Spirituality	<p>(147, 135, 122), (142, 129, 117), (145, 130, 116), (140, 126, 112), (146, 130, 115)</p> 
Literature & Fiction	<p>(136, 123, 112), (143, 128, 114), (141, 127, 113), (144, 128, 112), (144, 127, 110)</p> 
Science & Math	<p>(127, 123, 113), (143, 137, 127), (138, 129, 119), (136, 127, 114), (134, 128, 116)</p> 
Humor & Entertainment	<p>(141, 128, 117), (146, 135, 122), (146, 126, 113), (145, 127, 112), (147, 127, 111)</p> 
Computers & Technology	<p>(147, 145, 135), (138, 138, 128), (141, 130, 116), (141, 132, 119), (138, 130, 119)</p>

	
Biographies & Memoirs	<p>(137, 125, 114), (138, 126, 117), (143, 124, 112), (143, 124, 110), (144, 125, 112)</p> 
Arts & Photography	<p>(141, 130, 121), (140, 129, 120), (143, 130, 120), (138, 124, 113), (144, 129, 117)</p> 
Christian Books & Bibles	<p>(139, 128, 116), (147, 133, 121), (143, 128, 115), (139, 125, 111), (143, 130, 115)</p> 
Romance	<p>(124, 106, 98), (137, 120, 111), (139, 121, 111), (140, 119, 110), (141, 120, 109)</p> 
Comics & Graphic Novels	<p>(147, 132, 119), (143, 127, 116), (148, 130, 117), (142, 123, 109), (147, 128, 113)</p> 

Reference	<p>(148, 136, 126), (149, 138, 124), (150, 134, 119), (143, 128, 116), (148, 132, 117)</p> 
History	<p>(133, 122, 111), (141, 129, 119), (141, 127, 115), (140, 125, 113), (139, 122, 108)</p> 
Teen & Young Adult	<p>(133, 122, 114), (140, 129, 117), (144, 129, 117), (140, 124, 112), (143, 127, 115)</p> 
Parenting & Relationships	<p>(168, 157, 147), (157, 145, 134), (148, 135, 124), (153, 134, 119), (153, 139, 126)</p> 
Law	<p>(138, 130, 122), (138, 132, 123), (144, 131, 121), (136, 124, 114), (135, 124, 114)</p> 
Self-Help	<p>(164, 150, 141), (150, 138, 127), (155, 138, 125), (148, 131, 119), (151, 136, 123)</p>

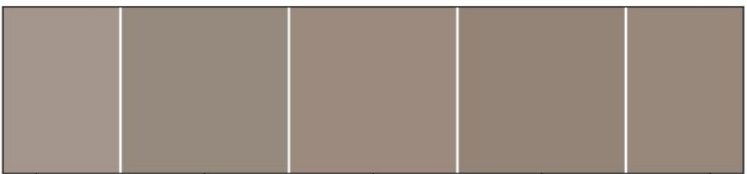














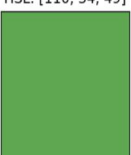












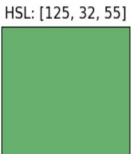
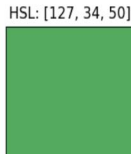
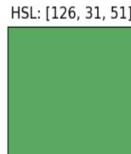
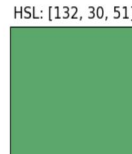
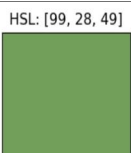
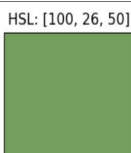
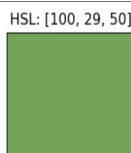
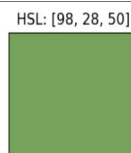
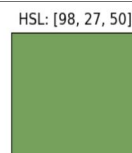
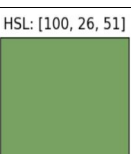
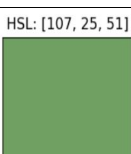
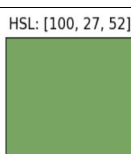
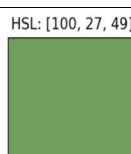
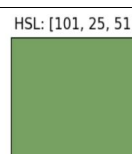




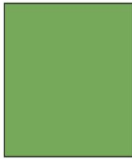
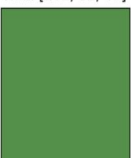




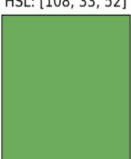



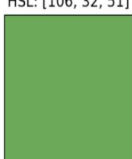
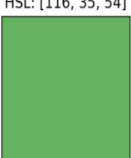

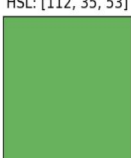

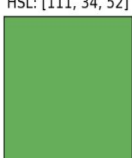
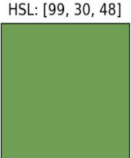
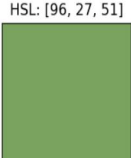
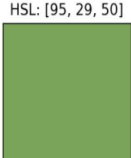

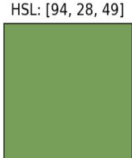
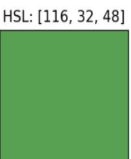
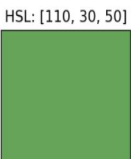
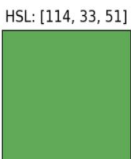
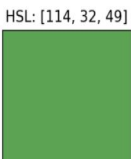
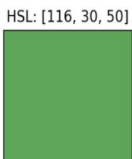
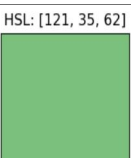
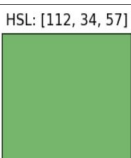
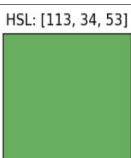
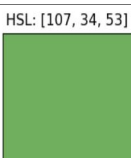
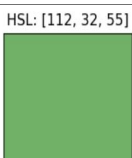
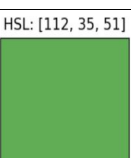
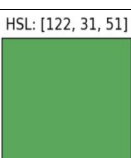
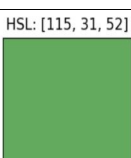
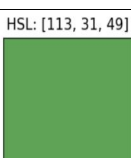
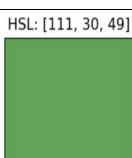
	
Science Fiction & Fantasy	<p>(110, 101, 93), (132, 122, 110), (136, 124, 113), (132, 119, 106), (134, 119, 106)</p> 
Test Preparation	<p>(145, 142, 137), (154, 150, 145), (147, 132, 120), (142, 130, 119), (149, 136, 122)</p> 
Engineering & Transportation	<p>(132, 126, 120), (139, 134, 126), (140, 128, 118), (136, 124, 115), (143, 130, 115)</p> 
Calendars	<p>(140, 129, 117), (143, 132, 120), (146, 132, 117), (140, 127, 112), (146, 132, 118)</p> 
Mystery, Thriller & Suspense	<p>(124, 110, 100), (136, 125, 114), (136, 119, 107), (141, 122, 109), (142, 124, 110)</p> 

Table C.2: HSL Values Generated For All Categories

Category	Predicted Colours				
Children's Books	HSL: [118, 36, 54] 	HSL: [108, 35, 54] 	HSL: [109, 36, 52] 	HSL: [102, 36, 52] 	HSL: [108, 36, 52] 
Sports & Outdoors	HSL: [119, 31, 49] 	HSL: [119, 30, 52] 	HSL: [114, 31, 51] 	HSL: [115, 31, 49] 	HSL: [116, 30, 50] 
Health, Fitness & Dieting	HSL: [115, 37, 58] 	HSL: [112, 35, 55] 	HSL: [118, 35, 52] 	HSL: [105, 34, 52] 	HSL: [110, 34, 52] 
Medical Books	HSL: [132, 37, 53] 	HSL: [136, 35, 55] 	HSL: [127, 34, 53] 	HSL: [132, 35, 51] 	HSL: [125, 32, 51] 
Travel	HSL: [119, 36, 49] 	HSL: [117, 35, 54] 	HSL: [114, 36, 51] 	HSL: [112, 34, 49] 	HSL: [110, 33, 50] 
Business & Money	HSL: [121, 33, 54] 	HSL: [123, 31, 52] 	HSL: [122, 33, 52] 	HSL: [122, 31, 49] 	HSL: [120, 31, 51] 
Cookbooks, Food & Wine	HSL: [80, 35, 55] 	HSL: [81, 36, 54] 	HSL: [73, 36, 52] 	HSL: [74, 37, 51] 	HSL: [73, 36, 52] 
Politics & Social Sciences	HSL: [115, 32, 50] 	HSL: [110, 30, 51] 	HSL: [102, 30, 51] 	HSL: [104, 30, 50] 	HSL: [108, 29, 51] 

Crafts, Hobbies & Home	HSL: [102, 30, 52] 	HSL: [104, 30, 53] 	HSL: [100, 31, 50] 	HSL: [95, 30, 51] 	HSL: [95, 30, 51] 
Religion & Spirituality	HSL: [107, 35, 53] 	HSL: [109, 34, 51] 	HSL: [106, 34, 51] 	HSL: [109, 32, 50] 	HSL: [104, 33, 51] 
Literature & Fiction	HSL: [110, 34, 49] 	HSL: [100, 32, 51] 	HSL: [105, 32, 50] 	HSL: [97, 32, 50] 	HSL: [96, 32, 50] 
Science & Math	HSL: [121, 34, 47] 	HSL: [119, 32, 53] 	HSL: [116, 32, 51] 	HSL: [112, 32, 49] 	HSL: [113, 31, 49] 
Humor & Entertainment	HSL: [110, 32, 50] 	HSL: [106, 31, 53] 	HSL: [111, 33, 51] 	HSL: [108, 32, 50] 	HSL: [105, 32, 51] 
Computers & Technology	HSL: [125, 32, 55] 	HSL: [135, 31, 53] 	HSL: [127, 34, 50] 	HSL: [126, 31, 51] 	HSL: [132, 30, 51] 
Biographies & Memoirs	HSL: [99, 28, 49] 	HSL: [100, 26, 50] 	HSL: [100, 29, 50] 	HSL: [98, 28, 50] 	HSL: [98, 27, 50] 
Arts & Photography	HSL: [100, 26, 51] 	HSL: [107, 25, 51] 	HSL: [100, 27, 52] 	HSL: [100, 27, 49] 	HSL: [101, 25, 51] 

Christian Books & Bibles	HSL: [102, 34, 50] 	HSL: [101, 33, 53] 	HSL: [102, 31, 50] 	HSL: [101, 31, 49] 	HSL: [99, 32, 51] 
Romance	HSL: [111, 32, 43] 	HSL: [103, 31, 48] 	HSL: [103, 32, 49] 	HSL: [108, 32, 49] 	HSL: [99, 31, 49] 
Comics & Graphic Novels	HSL: [108, 33, 52] 	HSL: [109, 31, 51] 	HSL: [111, 32, 52] 	HSL: [110, 32, 49] 	HSL: [106, 32, 51] 
Reference	HSL: [116, 35, 54] 	HSL: [110, 35, 54] 	HSL: [112, 35, 53] 	HSL: [113, 33, 51] 	HSL: [111, 34, 52] 
History	HSL: [99, 30, 48] 	HSL: [96, 27, 51] 	HSL: [95, 29, 50] 	HSL: [93, 28, 50] 	HSL: [94, 28, 49] 
Teen & Young Adult	HSL: [116, 32, 48] 	HSL: [110, 30, 50] 	HSL: [114, 33, 51] 	HSL: [114, 32, 49] 	HSL: [116, 30, 50] 
Parenting & Relationships	HSL: [121, 35, 62] 	HSL: [112, 34, 57] 	HSL: [113, 34, 53] 	HSL: [107, 34, 53] 	HSL: [112, 32, 55] 
Law	HSL: [112, 35, 51] 	HSL: [122, 31, 51] 	HSL: [115, 31, 52] 	HSL: [113, 31, 49] 	HSL: [111, 30, 49] 



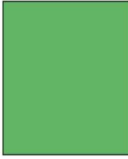






















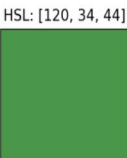
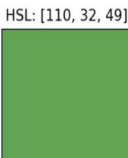
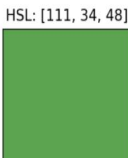
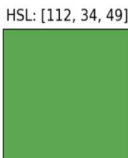
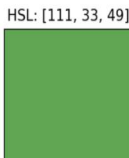
Self-Help	HSL: [123, 37, 60] 	HSL: [124, 36, 54] 	HSL: [121, 36, 55] 	HSL: [114, 35, 52] 	HSL: [114, 35, 54] 
Science Fiction & Fantasy	HSL: [120, 29, 40] 	HSL: [113, 29, 48] 	HSL: [109, 30, 49] 	HSL: [113, 30, 47] 	HSL: [109, 30, 47] 
Test Preparation	HSL: [143, 37, 56] 	HSL: [137, 32, 59] 	HSL: [137, 38, 52] 	HSL: [137, 35, 51] 	HSL: [131, 35, 53] 
Engineering & Transportation	HSL: [125, 29, 50] 	HSL: [128, 27, 52] 	HSL: [131, 30, 51] 	HSL: [129, 29, 49] 	HSL: [124, 31, 51] 
Calendars	HSL: [109, 32, 50] 	HSL: [107, 30, 52] 	HSL: [102, 32, 52] 	HSL: [103, 31, 50] 	HSL: [106, 31, 52] 
Mystery, Thriller & Suspense	HSL: [120, 34, 44] 	HSL: [110, 32, 49] 	HSL: [111, 34, 48] 	HSL: [112, 34, 49] 	HSL: [111, 33, 49] 

Table C.3: Predicted Colour Labels For All Categories

Category	Predicted Colours
Children's Books	['Red', 'Red', 'Red', 'Red', 'Red']
Sports & Outdoors	['Red', 'Red', 'Red', 'Red', 'Red']
Health, Fitness & Dieting	['Red', 'Red', 'Red', 'Red', 'Red']
Medical Books	['Red', 'Red', 'Red', 'Red', 'Red']
Travel	['Red', 'Red', 'Red', 'Red', 'Red']

Business & Money	['Red', 'Red', 'Red', 'Red', 'Red']
Cookbooks, Food & Wine	['Red', 'Red', 'Red', 'Red', 'Red']
Politics & Social Sciences	['Red', 'Red', 'Red', 'Red', 'Red']
Crafts, Hobbies & Home	['Red', 'Red', 'Red', 'Red', 'Red']
Religion & Spirituality	['Red', 'Red', 'Red', 'Red', 'Red']
Literature & Fiction	['Red', 'Red', 'Red', 'Red', 'Red']
Science & Math	['Red', 'Red', 'Red', 'Red', 'Red']
Humor & Entertainment	['Red', 'Red', 'Red', 'Red', 'Red']
Computers & Technology	['White', 'Red', 'Red', 'Red', 'Red']
Biographies & Memoirs	['Red', 'Red', 'Red', 'Red', 'Red']
Arts & Photography	['Red', 'Red', 'Red', 'Red', 'Red']
Christian Books & Bibles	['Red', 'Red', 'Red', 'Red', 'Red']
Romance	['Red', 'Red', 'Red', 'Red', 'Red']
Comics & Graphic Novels	['Red', 'Red', 'Red', 'Red', 'Red']
Reference	['Red', 'Red', 'Red', 'Red', 'Red']
History	['Red', 'Red', 'Red', 'Red', 'Red']
Teen & Young Adult	['Red', 'Red', 'Red', 'Red', 'Red']
Parenting & Relationships	['White', 'Red', 'Red', 'Red', 'Red']
Law	['Red', 'Red', 'Red', 'Red', 'Red']
Self-Help	['Red', 'Red', 'Red', 'Red', 'Red']
Science Fiction & Fantasy	['Red', 'Red', 'Red', 'Red', 'Red']
Test Preparation	['Red', 'White', 'Red', 'Red', 'Red']
Engineering & Transportation	['Red', 'Red', 'Red', 'Red', 'Red']
Calendars	['Red', 'Red', 'Red', 'Red', 'Red']
Mystery, Thriller & Suspense	['Red', 'Red', 'Red', 'Red', 'Red']

Table C.4: Predicted Text Area Percentage For All Categories

Category	Predicted Text Area (%)
Children's Books	21.44
Sports & Outdoors	24.14
Health, Fitness & Dieting	29.24
Medical Books	23.26
Travel	17.48
Business & Money	29.07
Cookbooks, Food & Wine	23.92
Politics & Social Sciences	22.62
Crafts, Hobbies & Home	19.50
Religion & Spirituality	21.24
Literature & Fiction	22.45
Science & Math	20.32
Humor & Entertainment	26.01
Computers & Technology	23.50
Biographies & Memoirs	25.46
Arts & Photography	17.71
Christian Books & Bibles	24.45
Romance	29.89
Comics & Graphic Novels	16.91
Reference	25.93
History	20.98
Teen & Young Adult	21.74
Parenting & Relationships	30.03
Law	19.52
Self-Help	31.70
Science Fiction & Fantasy	24.80
Test Preparation	29.73

Engineering & Transportation	21.29
Calendars	16.65
Mystery, Thriller & Suspense	37.98

Table C.5: Classification Report - Object Classification
Model

	precision	recall	f1-score	support
1D barcode	0.0	0.0	0.0	9
2D barcode	0.0	0.0	0.0	2
Airplane	0.0	0.0	0.0	20
Alarm clock	0.0	0.0	0.0	1
Animal	0.34	0.09	0.14	1501
Apple	0.0	0.0	0.0	9
Backpack	0.0	0.0	0.0	3
Bag	0.0	0.0	0.0	1
Baked goods	0.0	0.0	0.0	28
Ball	0.0	0.0	0.0	47
Balloon	0.0	0.0	0.0	4
Banana	0.0	0.0	0.0	3
Baseball	0.0	0.0	0.0	1
Basket	0.0	0.0	0.0	2
Basketball	0.0	0.0	0.0	4
Beaker	0.0	0.0	0.0	1
Bed	0.0	0.0	0.0	2
Beetle	0.0	0.0	0.0	2
Bench	0.0	0.0	0.0	2
Bicycle	0.0	0.0	0.0	5
Billiard table	0.0	0.0	0.0	1
Binoculars	0.0	0.0	0.0	1
Bird	0.0	0.0	0.0	53

Bladeless fan	0.0	0.0	0.0	1
Boat	0.0	0.0	0.0	48
Book	0.0	0.0	0.0	66
Boot	0.0	0.0	0.0	6
Bottle	0.0	0.0	0.0	5
Bowl	0.0	0.0	0.0	13
Bowtie	0.0	0.0	0.0	1
Box	0.0	0.0	0.0	37
Boxed packaged goods	0.0	0.0	0.0	6
Bracelet	0.0	0.0	0.0	12
Bread	0.0	0.0	0.0	4
Bridge	0.0	0.0	0.0	6
Broccoli	0.0	0.0	0.0	1
Bronze sculpture	0.0	0.0	0.0	2
Building	0.0	0.0	0.0	112
Bus	0.0	0.0	0.0	4
Bust	0.0	0.0	0.0	3
Butterfly	0.0	0.0	0.0	6
Cabbage	0.0	0.0	0.0	1
Cake	0.0	0.0	0.0	7
Calculator	0.0	0.0	0.0	1
Camel	0.0	0.0	0.0	4
Camera	0.0	0.0	0.0	1
Camera lens	0.0	0.0	0.0	1
Canary	0.0	0.0	0.0	1
Candle	0.0	0.0	0.0	3
Canoe	0.0	0.0	0.0	2
Car	0.2	0.64	0.3	81
Cart	0.0	0.0	0.0	5
Cassette deck	0.0	0.0	0.0	1
Castle	0.0	0.0	0.0	3

Cat	0.0	0.0	0.0	38
Cattle	0.0	0.0	0.0	1
Ceiling fan	0.0	0.0	0.0	2
Cello	0.0	0.0	0.0	1
Chair	0.0	0.0	0.0	17
Chandelier	0.0	0.0	0.0	2
Cheese	0.0	0.0	0.0	1
Chest of drawers	0.0	0.0	0.0	2
Chicken	0.0	0.0	0.0	3
Christmas tree	0.0	0.0	0.0	8
Clock	0.0	0.0	0.0	22
Clothing	0.0	0.0	0.0	2
Coat	0.0	0.0	0.0	3
Coffee cup	0.0	0.0	0.0	7
Coffee table	0.0	0.0	0.0	1
Coin	0.0	0.0	0.0	12
Computer keyboard	0.0	0.0	0.0	6
Computer monitor	0.0	0.0	0.0	1
Container	0.0	0.0	0.0	29
Couch	0.0	0.0	0.0	3
Cowboy hat	0.0	0.0	0.0	1
Crab	0.0	0.0	0.0	1
Cream	0.0	0.0	0.0	1
Crown	0.0	0.0	0.0	3
Cufflink	0.0	0.0	0.0	1
Deer	0.0	0.0	0.0	1
Dessert	0.0	0.0	0.0	8
Dice	0.0	0.0	0.0	3
Dinosaur	0.0	0.0	0.0	2
Dog	0.0	0.0	0.0	29
Dolphin	0.0	0.0	0.0	1

Door	0.0	0.0	0.0	7
Doughnut	0.0	0.0	0.0	3
Dragonfly	0.0	0.0	0.0	1
Dress	0.0	0.0	0.0	3
Drill	0.0	0.0	0.0	1
Drink	0.0	0.0	0.0	17
Drum	0.0	0.0	0.0	1
Eagle	0.0	0.0	0.0	2
Earrings	0.0	0.0	0.0	2
Egg	0.0	0.0	0.0	5
Elephant	0.0	0.0	0.0	3
Envelope	0.0	0.0	0.0	1
Falcon	0.0	0.0	0.0	2
Fedora	0.0	0.0	0.0	1
Fire hydrant	0.0	0.0	0.0	1
Fireplace	0.0	0.0	0.0	3
Fish	0.0	0.0	0.0	2
Flag	0.0	0.0	0.0	14
Flower	0.0	0.0	0.0	36
Flowerpot	0.0	0.0	0.0	8
Flute	0.0	0.0	0.0	1
Food	0.2	0.51	0.28	133
Football	0.0	0.0	0.0	3
Football helmet	0.0	0.0	0.0	1
Footwear	0.0	0.0	0.0	1
Fork	0.0	0.0	0.0	3
Fountain	0.0	0.0	0.0	1
Frog	0.0	0.0	0.0	1
Fruit	0.0	0.0	0.0	39
Furniture	0.0	0.0	0.0	31
Giraffe	0.0	0.0	0.0	1

Glasses	0.0	0.0	0.0	17
Glove	0.0	0.0	0.0	11
Goldfish	0.0	0.0	0.0	1
Golf ball	0.0	0.0	0.0	3
Gondola	0.0	0.0	0.0	1
Grape	0.0	0.0	0.0	3
Grapefruit	0.0	0.0	0.0	2
Greeting card	0.0	0.0	0.0	3
Grooming trimmer	0.0	0.0	0.0	6
Guacamole	0.0	0.0	0.0	1
Hamburger	0.0	0.0	0.0	2
Hamster	0.0	0.0	0.0	2
Handbag	0.0	0.0	0.0	3
Harp	0.0	0.0	0.0	1
Harpsichord	0.0	0.0	0.0	1
Hat	0.0	0.0	0.0	39
Hedgehog	0.0	0.0	0.0	2
Helicopter	0.0	0.0	0.0	1
Helmet	0.0	0.0	0.0	7
High heels	0.0	0.0	0.0	3
Hippopotamus	0.0	0.0	0.0	1
Home appliance	0.0	0.0	0.0	27
Horse	0.0	0.0	0.0	17
House	0.0	0.0	0.0	28
Houseplant	0.0	0.0	0.0	1
Ice	0.0	0.0	0.0	2
Insect	0.0	0.0	0.0	26
Jeans	0.0	0.0	0.0	3
Juice	0.0	0.0	0.0	1
Kangaroo	0.0	0.0	0.0	1
Kitchen Appliance	0.0	0.0	0.0	1

Kitchen appliance	0.0	0.0	0.0	1
Kitchenware	0.0	0.0	0.0	1
Kite	0.0	0.0	0.0	1
Ladder	0.0	0.0	0.0	1
Lamp	0.0	0.0	0.0	1
Lantern	0.0	0.0	0.0	1
Laptop	0.0	0.0	0.0	7
Lemon	0.0	0.0	0.0	6
Leopard	0.0	0.0	0.0	1
License plate	0.0	0.0	0.0	8
Light bulb	0.0	0.0	0.0	5
Light fixture	0.0	0.0	0.0	11
Lighthouse	0.0	0.0	0.0	15
Lighting	0.0	0.0	0.0	9
Limousine	0.0	0.0	0.0	1
Lion	0.0	0.0	0.0	4
Lipstick	0.0	0.0	0.0	2
Lizard	0.0	0.0	0.0	4
Lock	0.0	0.0	0.0	1
Loudspeaker	0.0	0.0	0.0	1
Luggage & bags	0.0	0.0	0.0	24
Magazine	0.0	0.0	0.0	2
Mango	0.0	0.0	0.0	1
Mechanical fan	0.0	0.0	0.0	1
Menu	0.0	0.0	0.0	2
Microphone	0.0	0.0	0.0	3
Mirror	0.0	0.0	0.0	1
Mobile phone	0.0	0.0	0.0	5
Moths and butterflies	0.0	0.0	0.0	3
Motorcycle	0.0	0.0	0.0	14
Mouse	0.0	0.0	0.0	3

Mule	0.0	0.0	0.0	1
Musical instrument	0.0	0.0	0.0	9
Musical keyboard	0.0	0.0	0.0	1
Necklace	0.0	0.0	0.0	13
Notebook	0.0	0.0	0.0	16
Orange	0.0	0.0	0.0	1
Outerwear	0.0	0.0	0.0	5
Owl	0.0	0.0	0.0	1
Packaged goods	0.0	0.0	0.0	464
Painting	0.0	0.0	0.0	10
Pancake	0.0	0.0	0.0	1
Pants	0.0	0.0	0.0	1
Parachute	0.0	0.0	0.0	2
Parrot	0.0	0.0	0.0	3
Peach	0.0	0.0	0.0	1
Pear	0.0	0.0	0.0	3
Pen	0.0	0.0	0.0	10
Penguin	0.0	0.0	0.0	2
Person	0.47	0.95	0.63	3415
Piano	0.0	0.0	0.0	2
Picture frame	0.0	0.0	0.0	67
Pig	0.0	0.0	0.0	1
Pillow	0.0	0.0	0.0	18
Pineapple	0.0	0.0	0.0	2
Pizza	0.0	0.0	0.0	9
Plant	0.0	0.0	0.0	92
Plate	0.0	0.0	0.0	5
Polar bear	0.0	0.0	0.0	2
Pomegranate	0.0	0.0	0.0	1
Post-it note	0.0	0.0	0.0	3
Poster	0.0	0.0	0.0	152

Pumpkin	0.0	0.0	0.0	5
Rabbit	0.0	0.0	0.0	2
Racket	0.0	0.0	0.0	42
Raven	0.0	0.0	0.0	2
Ring binder	0.0	0.0	0.0	3
Robotic vacuum cleaner	0.0	0.0	0.0	1
Rocket	0.0	0.0	0.0	3
Rugby ball	0.0	0.0	0.0	3
Ruler	0.0	0.0	0.0	1
Salad	0.0	0.0	0.0	6
Sandal	0.0	0.0	0.0	1
Sandwich	0.0	0.0	0.0	2
Scale	0.0	0.0	0.0	1
Scarf	0.0	0.0	0.0	3
Scissors	0.0	0.0	0.0	2
Sculpture	0.0	0.0	0.0	10
Shark	0.0	0.0	0.0	4
Sheep	0.0	0.0	0.0	3
Ship	0.0	0.0	0.0	4
Shoe	0.0	0.0	0.0	12
Shorts	0.0	0.0	0.0	1
Single-lens reflex camera	0.0	0.0	0.0	1
Skateboard	0.0	0.0	0.0	1
Skirt	0.0	0.0	0.0	4
Slow cooker	0.0	0.0	0.0	1
Snack	0.0	0.0	0.0	3
Snail	0.0	0.0	0.0	1
Snake	0.0	0.0	0.0	1
Sombrero	0.0	0.0	0.0	1
Spoon	0.0	0.0	0.0	1
Squash	0.0	0.0	0.0	14

Squirrel	0.0	0.0	0.0	2
Starfish	0.0	0.0	0.0	1
Stethoscope	0.0	0.0	0.0	3
Stop sign	0.0	0.0	0.0	1
Straw hat	0.0	0.0	0.0	1
Strawberry	0.0	0.0	0.0	3
Street light	0.0	0.0	0.0	4
Stretcher	0.0	0.0	0.0	2
Submarine	0.0	0.0	0.0	1
Sunglasses	0.0	0.0	0.0	7
Surfboard	0.0	0.0	0.0	1
Swan	0.0	0.0	0.0	1
Sword	0.0	0.0	0.0	6
Table	0.0	0.0	0.0	7
Table tennis racket	0.0	0.0	0.0	1
Table top	0.0	0.0	0.0	4
Tablet computer	0.0	0.0	0.0	7
Tableware	0.0	0.0	0.0	62
Taco	0.0	0.0	0.0	1
Tank	0.0	0.0	0.0	4
Teapot	0.0	0.0	0.0	2
Teddy bear	0.0	0.0	0.0	3
Telephone	0.0	0.0	0.0	1
Television	0.0	0.0	0.0	2
Tennis ball	0.0	0.0	0.0	5
Tent	0.0	0.0	0.0	1
Throw pillow	0.0	0.0	0.0	4
Tie	0.0	0.0	0.0	32
Tiger	0.0	0.0	0.0	3
Tire	0.0	0.0	0.0	10
Toaster	0.0	0.0	0.0	1

Tomato	0.0	0.0	0.0	3
Toothbrush	0.0	0.0	0.0	1
Top	0.0	0.0	0.0	15
Tortoise	0.0	0.0	0.0	1
Toy	0.0	0.0	0.0	40
Toy vehicle	0.0	0.0	0.0	1
Traffic sign	0.0	0.0	0.0	6
Train	0.0	0.0	0.0	12
Truck	0.0	0.0	0.0	11
Trumpet	0.0	0.0	0.0	1
Tubed packaged goods	0.0	0.0	0.0	2
Turtle	0.0	0.0	0.0	33
Umbrella	0.0	0.0	0.0	5
Van	0.0	0.0	0.0	4
Vase	0.0	0.0	0.0	9
Vegetable	0.0	0.0	0.0	15
Vinyl record	0.0	0.0	0.0	2
Violin	0.0	0.0	0.0	1
Volleyball	0.0	0.0	0.0	10
Wallet	0.0	0.0	0.0	2
Watch	0.0	0.0	0.0	9
Watermelon	0.0	0.0	0.0	1
Weapon	0.0	0.0	0.0	1
Wheel	0.0	0.0	0.0	48
Whiteboard	0.0	0.0	0.0	1
Window	0.0	0.0	0.0	5
Wine glass	0.0	0.0	0.0	2
Wine rack	0.0	0.0	0.0	1
Wok	0.0	0.0	0.0	1
Wrench	0.0	0.0	0.0	1
Zebra	0.0	0.0	0.0	1

accuracy			0.44	7923
macro avg	0.0	0.01	0.0	7923
weighted avg	0.27	0.44	0.3	7923

Bibliography

- [1] Olusola Oluwakemi Abayomi-Alli et al. "Cassava disease recognition from low-quality images using enhanced data augmentation model and deep learning". In: *Expert Systems* 38.7 (2021), e12746. DOI: <https://doi.org/10.1111/exsy.12746>. eprint: <https://onlinelibrary.wiley.com/doi/pdf/10.1111/exsy.12746>. URL: <https://onlinelibrary.wiley.com/doi/abs/10.1111/exsy.12746>.
- [2] Abolfazl Abdollahi et al. "Deep Learning Approaches Applied to Remote Sensing Datasets for Road Extraction: A State-Of-The-Art Review". In: *Remote Sensing* 12.9 (2020). ISSN: 2072-4292. DOI: [10.3390/rs12091444](https://doi.org/10.3390/rs12091444). URL: <https://www.mdpi.com/2072-4292/12/9/1444>.
- [3] Eden AI. *Top 10 object detection apis*. Nov. 2023. URL: <https://www.edenai.co/post/top-10-object-detection-apis>.
- [4] Jaied AI. *EasyOCR demo*. Nov. 2023. URL: <https://www.jaied.ai/easyocr/>.
- [5] Jaied AI. *Jaiedai/EasyOCR: Ready-to-use OCR*. Oct. 2023. URL: <https://github.com/JaiedAI/EasyOCR>.
- [6] Abdullah Alfarrarjeh et al. "A Deep Learning Approach for Road Damage Detection from Smartphone Images". In: *2018 IEEE International Conference on Big Data (Big Data)*. 2018, pp. 5201–5204. DOI: [10.1109/BigData.2018.8621899](https://doi.org/10.1109/BigData.2018.8621899).
- [7] Amazon Amazon. *What is a Neural Network?* Nov. 1978. URL: <https://aws.amazon.com/what-is/neural-network/#:~:text=A%20neural%20network%20is%20a,that%20resembles%20the%20human%20brain..>

- [8] Amazon Amazon. *What is OCR (Optical Character Recognition)?* Nov. 1978. URL: [https://aws.amazon.com/what-is/ocr/#:~:text=Optical%20Character%20Recognition%20\(OCR\)%20is,scan%20as%20an%20image%20file..](https://aws.amazon.com/what-is/ocr/#:~:text=Optical%20Character%20Recognition%20(OCR)%20is,scan%20as%20an%20image%20file..)
- [9] Anju Asokan and J Anitha. "Machine Learning based Image Processing Techniques for Satellite Image Analysis -A Survey". In: *2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon)*. 2019, pp. 119–124. DOI: [10.1109/COMITCon.2019.8862452](https://doi.org/10.1109/COMITCon.2019.8862452).
- [10] Abid Ali Awan. *What is tokenization? types, use cases, implementation*. Sept. 2023. URL: <https://www.datacamp.com/blog/what-is-tokenization>.
- [11] Hmrishav Bandyopadhyay. *Optical character recognition (OCR): Definition how to guide*. July 2021. URL: <https://www.v7labs.com/blog/ocr-guide>.
- [12] Pritha Bhandari. *Missing data: Types, explanation, imputation*. Oct. 2022. URL: <https://www.scribbr.co.uk/stats/missing-values/>.
- [13] Jason Brownlee. *4 types of classification tasks in machine learning*. Aug. 2020. URL: <https://machinelearningmastery.com/types-of-classification-in-machine-learning/>.
- [14] Jason Brownlee. *How to one hot encode sequence data in python*. Aug. 2019. URL: <https://machinelearningmastery.com/how-to-one-hot-encode-sequence-data-in-python/>.
- [15] Jason Brownlee. *Tensorflow 2 tutorial: Get started in deep learning with tf.keras*. Aug. 2022. URL: <https://machinelearningmastery.com/tensorflow-tutorial-deep-learning-with-tf-keras/>.
- [16] Jason Brownlee. *Train-test split for Evaluating Machine Learning Algorithms*. Aug. 2020. URL: <https://machinelearningmastery.com/train-test-split-for-evaluating-machine-learning-algorithms/>.

- [17] Dianne Castillo. *Machine learning regression explained*. Nov. 2023. URL: <https://www.seldon.io/machine-learning-regression-explained>.
- [18] Sampriti Chatterjee. *What is feature extraction? feature extraction in image processing*. July 2022. URL: <https://www.mygreatlearning.com/blog/feature-extraction-in-image-processing/>.
- [19] Saturn Cloud. *How to check for duplicate values in pandas DataFrame column*. Oct. 2023. URL: <https://saturncloud.io/blog/how-to-check-for-duplicate-values-in-pandas-dataframe-column/#:~:text=To%20check%20for%20duplicate%20values%20in%20a%20Pandas%20DataFrame%20column,is%20a%20duplicate%20or%20not..>
- [20] CodePal CodePal. *Python test train split*. Oct. 2023. URL: <https://codepal.ai/code-generator/query/1OUNXKtI/python-test-train-split>.
- [21] TechTarget Contributor. *What is OCR (optical character recognition)?: Definition from TechTarget*. Nov. 2022. URL: <https://www.techtarget.com/searchcontentmanagement/definition/OCR-optical-character-recognition>.
- [22] Alain d'Astous, Francois Colbert, and Imene Mbarek. "Factors influencing readers' interest in new book releases: An experimental study". In: *Poetics* 34.2 (2006), pp. 134–147. ISSN: 0304-422X. DOI: <https://doi.org/10.1016/j.poetic.2005.12.001>. URL: <https://www.sciencedirect.com/science/article/pii/S0304422X05000781>.
- [23] Andrea D'Agostino. *Get started with tensorflow 2.0 - introduction to Deep Learning*. Feb. 2023. URL: <https://towardsdatascience.com/a-comprehensive-introduction-to-tensorflows-sequential-api-and-model-for-deep-learning-c5e31aee49fa>.

- [24] A Deepanshi. *All you need to know about your first machine learning model - linear regression*. July 2023. URL: <https://www.analyticsvidhya.com/blog/2021/05/all-you-need-to-know-about-your-first-machine-learning-model-linear-regression/>.
- [25] IBM Developer. *Classifying data using the Multinomial Naive Bayes algorithm*. Nov. 2023. URL: <https://developer.ibm.com/tutorials/awb-classifying-data-multinomial-naive-bayes-algorithm/>.
- [26] *Don't judge a book by its cover*. Aug. 2023. URL: https://en.wikipedia.org/wiki/Don%27t_judge_a_book_by_its_cover.
- [27] Stanford Edu. Nov. 2008. URL: <https://nlp.stanford.edu/IR-book/html/htmledition/tokenization-1.html>.
- [28] EDUCBA. *Tensorflow sequential: Complete Guide on Tensorflow Sequential*. Aug. 2023. URL: <https://www.educba.com/tensorflow-sequential/>.
- [29] EDUCBA EDUCBA. *Support vector regression: Learn the working and advantages of SVR*. Mar. 2023. URL: <https://www.educba.com/support-vector-regression/>.
- [30] Athena L Edwards. *The paratexts of audience engagement: Cover matter that draws in and keeps readers*. July 2022. URL: https://nsuworks.nova.edu/hcas_etd_all/92/.
- [31] Elastic. *Tokenizer reference: Elasticsearch guide [8.11]*. Nov. 2023. URL: <https://www.elastic.co/guide/en/elasticsearch/reference/current/analysis-tokenizers.html>.
- [32] Marcin Frackiewicz. *Applying google vision API to object detection and tracking*. Apr. 2023. URL: <https://ts2.space/en/applying-google-vision-api-to-object-detection-and-tracking/#gsc.tab=0>.
- [33] Kavita Ganesan. *How to use TFIDFTRANSFORMER and Tfidfvectorizer - A Short tutorial*. Aug. 2020. URL: <https://kavita-ganesan.com/tfidftransformer-tfidfvectorizer-usage-differences/>.

- [34] Garry. *Why is encoding and decoding needed for any programming language/database?* Apr. 2023. URL: <https://stackoverflow.com/questions/14822323/why-is-encoding-and-decoding-needed-for-any-programming-language-database>.
- [35] GeeksforGeeks. *Extract dominant colors of an image using Python*. Sept. 2023. URL: <https://www.geeksforgeeks.org/extract-dominant-colors-of-an-image-using-python/>.
- [36] GeeksforGeeks. *R-squared in regression analysis in machine learning*. May 2023. URL: <https://www.geeksforgeeks.org/ml-r-squared-in-regression-analysis/>.
- [37] GeeksforGeeks. *Random Forest regression in python*. June 2023. URL: <https://www.geeksforgeeks.org/random-forest-regression-in-python/>.
- [38] GeeksforGeeks. *Removing stop words with NLTK in python*. May 2023. URL: <https://www.geeksforgeeks.org/removing-stop-words-nltk-python/>.
- [39] GeeksforGeeks. *Support vector regression (SVR) using linear and non-linear kernels in Scikit learn*. Jan. 2023. URL: <https://www.geeksforgeeks.org/support-vector-regression-svr-using-linear-and-non-linear-kernels-in-scikit-learn/>.
- [40] GeeksforGeeks GeeksforGeeks. *Find duplicate rows in a Dataframe based on all or selected columns*. Feb. 2022. URL: <https://www.geeksforgeeks.org/find-duplicate-rows-in-a-dataframe-based-on-all-or-selected-columns/>.
- [41] Eli Gibson et al. "NiftyNet: a deep-learning platform for medical imaging". In: *Computer Methods and Programs in Biomedicine* 158 (2018), pp. 113–122. ISSN: 0169-2607. DOI: <https://doi.org/10.1016/j.cmpb.2018.01.025>. URL: <https://www.sciencedirect.com/science/article/pii/S0169260717311823>.
- [42] Google. *Detect multiple objects, cloud vision API, google cloud*. Nov. 2023. URL: <https://cloud.google.com/vision/docs/object-localizer>.

- [43] TensorFlow Google. *Tf.keras.preprocessing.text.tokenizer tensorflow V2.14.0*. Nov. 2023. URL: https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/text/Tokenizer.
- [44] Arūnas Gudinaičius and Andrius Šuminas. *Choosing a book by its cover: Analysis of a reader's choice*. Nov. 2017. URL: <https://www.emerald.com/insight/content/doi/10.1108/JD-09-2016-0111/full/html>.
- [45] Ulrik Stig Hansen. *Object detection: Models, use cases, examples*. Apr. 2023. URL: <https://encord.com/blog/object-detection/>.
- [46] Hasty.ai. *Accuracy*. Nov. 2023. URL: <https://wiki.cloudfactory.com/docs/mp-wiki/metrics/accuracy#:~:text=To%20define%20the%20term%2C%20in,performance%20with%20a%20single%20value..>
- [47] Shen D;Wu G;Suk HI; *Deep learning in medical image analysis*. Mar. 2017. URL: <https://pubmed.ncbi.nlm.nih.gov/28301734/>.
- [48] Hyland. *What is optical character recognition (OCR) technology?* Nov. 2023. URL: <https://www.hyland.com/en/resources/terminology/data-capture/what-is-optical-character-recognition-ocr>.
- [49] IBM. *About linear regression*. Nov. 2023. URL: <https://www.ibm.com/topics/linear-regression#:~:text=Resources-,What%20is%20linear%20regression%3F,is%20called%20the%20independent%20variable..>
- [50] IBM. *What are neural networks?* Nov. 2023. URL: <https://www.ibm.com/topics/neural-networks>.
- [51] Built In. *Random Forest regression in Python explained*. Nov. 2023. URL: <https://builtin.com/data-science/random-forest-python>.
- [52] Built In. *Understanding train test split*. Oct. 2023. URL: <https://builtin.com/data-science/train-test-split>.

- [53] Built In. *What is decision tree classification?* Nov. 2023. URL: <https://builtin.com/data-science/classification-tree>.
- [54] Built In. *Why is logistic regression a classification algorithm?* Nov. 2023. URL: <https://builtin.com/machine-learning/logistic-regression-classification-algorithm>.
- [55] Aras M. Ismael and Abdulkadir Şengür. "Deep learning approaches for COVID-19 detection based on chest X-ray images". In: *Expert Systems with Applications* 164 (2021), p. 114054. ISSN: 0957-4174. DOI: <https://doi.org/10.1016/j.eswa.2020.114054>. URL: <https://www.sciencedirect.com/science/article/pii/S0957417420308198>.
- [56] Brian Kenji Iwana et al. *Judging a Book By its Cover*. 2017. arXiv: [1610.09204](https://arxiv.org/abs/1610.09204) [cs.CV].
- [57] Licheng Jiao and Jin Zhao. "A Survey on the New Generation of Deep Learning in Image Processing". In: *IEEE Access* 7 (2019), pp. 172231–172263. DOI: [10.1109/ACCESS.2019.2956508](https://doi.org/10.1109/ACCESS.2019.2956508).
- [58] Jobin John. *TfidfVectorizer: TF-IDF vectorizer scikit-learn*. July 2022. URL: <https://www.egochi.com/tfidfvectorizer/>.
- [59] Mingquan Chen Jun Wang Jingwei Song and Zhi Yang. "Road network extraction: a neural-dynamic framework based on deep learning and a finite state machine". In: *International Journal of Remote Sensing* 36.12 (2015), pp. 3144–3169. DOI: [10.1080/01431161.2015.1054049](https://doi.org/10.1080/01431161.2015.1054049). eprint: <https://doi.org/10.1080/01431161.2015.1054049>. URL: <https://doi.org/10.1080/01431161.2015.1054049>.
- [60] Gurucharan M K. *Machine learning basics: Decision tree regression*. July 2020. URL: <https://towardsdatascience.com/machine-learning-basics-decision-tree-regression-1d73ea003fda>.
- [61] KDnuggets. *Convert text documents to a TF-IDF matrix with tfidfvectorizer*. Nov. 2023. URL: <https://www.kdnuggets.com/2022/09/convert-text-documents-tfidf-matrix-tfidfvectorizer.html>.

- [62] Zoumana Keita. *Classification in machine learning: A guide for beginners*. Sept. 2022. URL: <https://www.datacamp.com/blog/classification-machine-learning>.
- [63] Benjamin Kellenberger, Diego Marcos, and Devis Tuia. "Detecting mammals in UAV images: Best practices to address a substantially imbalanced dataset with deep learning". In: *Remote Sensing of Environment* 216 (2018), pp. 139–153. ISSN: 0034-4257. DOI: <https://doi.org/10.1016/j.rse.2018.06.028>. URL: <https://www.sciencedirect.com/science/article/pii/S0034425718303067>.
- [64] Aman Kharwal. *Classification report in Machine Learning: Aman Kharwal*. July 2021. URL: <https://thecleverprogrammer.com/2021/07/07/classification-report-in-machine-learning/>.
- [65] Mingyu Kim et al. *Deep learning in medical imaging*. Dec. 2019. URL: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6945006/>.
- [66] D L. *Optical character recognition (OCR) with easyocr: Pytorch*. Aug. 2023. URL: <https://medium.com/@ilaslanduzgun/optical-character-recognition-ocr-with-easyocr-pytorch-27232272ab38>.
- [67] Yunfei Lai. "A Comparison of Traditional Machine Learning and Deep Learning in Image Recognition". In: *Journal of Physics: Conference Series* 1314.1 (Oct. 2019), p. 012148. DOI: [10.1088/1742-6596/1314/1/012148](https://doi.org/10.1088/1742-6596/1314/1/012148). URL: <https://dx.doi.org/10.1088/1742-6596/1314/1/012148>.
- [68] Wentong Liao et al. "Text to Image Generation With Semantic-Spatial Aware GAN". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2022, pp. 18187–18196.
- [69] PhD Lihi Gur Arie. *Image color segmentation by K-means clustering algorithm*. Feb. 2023. URL: <https://towardsdatascience.com/image-color-segmentation-by-k-means-clustering-algorithm-5792e563f26e>.

- [70] Lijuan Liu, Yanping Wang, and Wanle Chi. "Image Recognition Technology Based on Machine Learning". In: *IEEE Access* (2020), pp. 1–1. DOI: [10 . 1109/ACCESS.2020.3021590](https://doi.org/10.1109/ACCESS.2020.3021590).
- [71] Inna Logunova. *A guide to F1 score*. July 2023. URL: <https://serokell.io/blog/a-guide-to-f1-score>.
- [72] Inna Logunova. *Random forest classifier: Basic principles and applications*. June 2022. URL: <https://serokell.io/blog/random-forest-classification>.
- [73] Heng Lu et al. "Landslides Information Extraction Using Object-Oriented Image Analysis Paradigm Based on Deep Learning and Transfer Learning". In: *Remote Sensing* 12.5 (2020). ISSN: 2072-4292. DOI: [10 . 3390 / rs12050752](https://doi.org/10.3390/rs12050752). URL: <https://www.mdpi.com/2072-4292/12/5/752>.
- [74] Mario Lučić et al. "High-Fidelity Image Generation With Fewer Labels". In: *Proceedings of the 36th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR, June 2019, pp. 4183–4192. URL: <https://proceedings.mlr.press/v97/lucic19a.html>.
- [75] de Bruijne M; *Machine learning approaches in medical image analysis: From detection to diagnosis*. June 2016. URL: <https://pubmed.ncbi.nlm.nih.gov/27481324/>.
- [76] Kavita Mali. *Everything you need to know about linear regression!* Sept. 2023. URL: <https://www.analyticsvidhya.com/blog/2021/10/everything-you-need-to-know-about-linear-regression/>.
- [77] Simulink MATLAB. *Object recognition*. Oct. 2023. URL: <https://www.mathworks.com/solutions/image-video-processing/object-recognition.html#:~:text=Object%20recognition%20is%20a%20computer,%2C%20scenes%2C%20and%20visual%20details..>

- [78] Elliot Mbunge et al. "Application of deep learning and machine learning models to detect COVID-19 face masks - A review". In: *Sustainable Operations and Computers* 2 (Apr. 2021), pp. 235–245. ISSN: 2666-4127. DOI: <https://doi.org/10.1016/j.susoc.2021.08.001>. URL: <https://www.sciencedirect.com/science/article/pii/S2666412721000325>.
- [79] Amal Menzli. *Tokenization in NLP: Types, challenges, examples, tools*. Aug. 2023. URL: <https://neptune.ai/blog/tokenization-in-nlp>.
- [80] Mehdi Mirza and Simon Osindero. *Conditional Generative Adversarial Nets*. 2014. arXiv: [1411.1784](https://arxiv.org/abs/1411.1784) [cs.LG].
- [81] Arnab Mondal. *A complete guide to understand classification in Machine Learning*. Apr. 2023. URL: <https://www.analyticsvidhya.com/blog/2021/09/a-complete-guide-to-understand-classification-in-machine-learning/>.
- [82] Mahdieh Montazeri et al. "Machine Learning Models for Image-Based Diagnosis and Prognosis of COVID-19: Systematic Review". In: *JMIR Med Inform* 9.4 (Apr. 2021), e25181. ISSN: 2291-9694. DOI: [10.2196/25181](https://doi.org/10.2196/25181). URL: <http://www.ncbi.nlm.nih.gov/pubmed/33735095>.
- [83] Brian Mossop. "Judging a translation by its cover". In: *The Translator* 24.1 (2018), pp. 1–16. DOI: [10.1080/13556509.2017.1287545](https://doi.org/10.1080/13556509.2017.1287545). eprint: <https://doi.org/10.1080/13556509.2017.1287545>. URL: <https://doi.org/10.1080/13556509.2017.1287545>.
- [84] Amal Nair. *A beginner's guide to scikit-learn's mlpclassifier*. Nov. 2020. URL: <https://analyticsindiamag.com/a-beginners-guide-to-scikit-learns-mlpclassifier/>.
- [85] Augustus Odena, Christopher Olah, and Jonathon Shlens. "Conditional Image Synthesis with Auxiliary Classifier GANs". In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. PMLR, Aug. 2017, pp. 2642–2651. URL: <https://proceedings.mlr.press/v70/odena17a.html>.

- [86] Capital One. *Scikit-Learn: Implementation of TF-IDF for NLP*. Sept. 2023. URL: <https://www.capitalone.com/tech/machine-learning/scikit-tfidf-implementation/>.
- [87] Shiksha Online. *Mean squared error in machine learning - shiksha online*. Nov. 2022. URL: [https://www.shiksha.com/online-courses/articles/mean-squared-error-in-machine-learning/#:~:text=Mean%20Squared%20Error%20\(MSE\)%20measures,as%20the%20model%20error%20increases..](https://www.shiksha.com/online-courses/articles/mean-squared-error-in-machine-learning/#:~:text=Mean%20Squared%20Error%20(MSE)%20measures,as%20the%20model%20error%20increases..)
- [88] Priyanka Parashar. *Decision tree classification and it's mathematical implementation*. June 2020. URL: <https://medium.com/@priyankaparashar54/decision-tree-classification-and-its-mathematical-implementation-c27006caefbb>.
- [89] Ronald A. M. P. Piter and Mia J. W. Stokmans. "Genre Categorization and its Effect on Preference for Fiction Books". In: *Empirical Studies of the Arts* 18.2 (2000), pp. 159–166. DOI: [10.2190/0VJF-Y04E-H5NU-VL5B](https://doi.org/10.2190/0VJF-Y04E-H5NU-VL5B). eprint: <https://doi.org/10.2190/0VJF-Y04E-H5NU-VL5B>. URL: <https://doi.org/10.2190/0VJF-Y04E-H5NU-VL5B>.
- [90] Java Point. *Regression analysis in machine learning - javatpoint*. Nov. 2023. URL: <https://www.javatpoint.com/regression-analysis-in-machine-learning>.
- [91] Ashwin Prasad. *Regression trees: Decision tree for regression: Machine Learning*. Aug. 2021. URL: <https://medium.com/analytics-vidhya/regression-trees-decision-tree-for-regression-machine-learning-e4d7525d8047>.
- [92] prashant111. *Random forest classifier tutorial*. Mar. 2020. URL: <https://www.kaggle.com/code/prashant111/random-forest-classifier-tutorial>.
- [93] Real Python. *Split your dataset with scikit-learn's $\text{train_test_split}()$* . Oct. 2023. URL: <https://realpython.com/train-test-split-python-data/>.

- [94] Real Python. *Unicode Character Encodings in python: A painless guide*. Oct. 2023. URL: <https://realpython.com/python-encodings-guide/#enter-unicode>.
- [95] Ashwin Raj. *An exhaustive guide to decision tree classification in python* 3.X. Dec. 2021. URL: <https://towardsdatascience.com/an-exhaustive-guide-to-classification-using-decision-trees-8d472e77223f>.
- [96] Ashwin Raj. *Unlocking the true power of support vector regression*. Oct. 2020. URL: <https://towardsdatascience.com/unlocking-the-true-power-of-support-vector-regression-847fd123a4a0>.
- [97] Manasa Ramakrishnan. *What is classification in Machine Learning and why is it important?* July 2023. URL: <https://emeritus.org/blog/artificial-intelligence-and-machine-learning-classification-in-machine-learning/>.
- [98] Gabriel Ramuglia. *Using pandas to drop duplicates: A detailed walkthrough*. Aug. 2023. URL: <https://ioflood.com/blog/pandas-drop-duplicates/#:~:text=One%20common%20issue%20is%20duplicate,based%20on%20unique%2C%20reliable%20data..>
- [99] Arthur V. Ratz. *Multinomial NAIVE Bayes' for documents classification and Natural Language Processing (NLP)*. Apr. 2022. URL: <https://towardsdatascience.com/multinomial-na%C3%AFve-bayes-for-documents-classification-and-natural-language-processing-nlp-e08cc848ce6>.
- [100] Adrian Rosebrock. *3 ways to create a Keras model with tensorflow 2.0 (sequential, functional, and model subclassing)*. June 2023. URL: <https://pyimagesearch.com/2019/10/28/3-ways-to-create-a-keras-model-with-tensorflow-2-0-sequential-functional-and-model-subclassing/>.
- [101] Adrian Rosebrock. *Getting started with easyocr for optical character recognition*. June 2023. URL: <https://pyimagesearch.com/2020/09/14/>

- getting-started-with-easyocr-for-optical-character-recognition/.
- [102] Unnikrishnan C S. *How SKLEARN's Tfidfvectorizer calculates TF-IDF values*. Nov. 2021. URL: <https://www.analyticsvidhya.com/blog/2021/11/how-sklearns-tfidfvectorizer-calculates-tf-idf-values/>.
- [103] Zach S. *Label encoding vs. One hot encoding: What's the difference?* Aug. 2022. URL: <https://www.statology.org/label-encoding-vs-one-hot-encoding/>.
- [104] Anshul Saini. *A beginner's guide to logistic regression*. Oct. 2023. URL: <https://www.analyticsvidhya.com/blog/2021/08/conceptual-understanding-of-logistic-regression-for-data-science-beginners/>.
- [105] Anshul Saini. *Decision tree algorithm - A complete guide*. Sept. 2023. URL: <https://www.analyticsvidhya.com/blog/2021/08/decision-tree-algorithm/#:~:text=A%20decision%20tree%20algorithm%20is,each%20node%20of%20the%20tree..>
- [106] SaturnCloud. *Adding headers to a DataFrame in pandas: A guide*. Nov. 2023. URL: <https://saturncloud.io/blog/adding-headers-to-a-dataframe-in-pandas-a-comprehensive-guide/>.
- [107] SaturnCloud. *How to count nan values in a pandas DataFrame column*. Oct. 2023. URL: <https://saturncloud.io/blog/how-to-count-nan-values-in-a-pandas-dataframe-column/>.
- [108] scikit. *Feature extraction*. Nov. 2023. URL: https://scikit-learn.org/stable/modules/feature_extraction.html#:~:text=We%20call%20vectorization.
- [109] scikit. *Transforming the prediction target (Y)*. Nov. 2023. URL: https://scikit-learn.org/stable/modules/preprocessing_targets.html#preprocessing-targets.
- [110] scikit learn scikit. *Decision trees*. Nov. 2023. URL: <https://scikit-learn.org/stable/modules/tree.html>.

- [111] scikit learn scikit. *Sklearn.linear_model.linearregression*. Nov. 2023. URL: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html.
- [112] Section Section. *Introduction to random forest in machine learning*. Nov. 2023. URL: <https://www.section.io/engineering-education/introduction-to-random-forest-in-machine-learning/>.
- [113] Alakh Sethi. *One hot encoding vs. label encoding using Scikit-Learn*. June 2023. URL: <https://www.analyticsvidhya.com/blog/2020/03/one-hot-encoding-vs-label-encoding-using-scikit-learn/>.
- [114] Gaurav Sharma. *5 regression algorithms you should know - introductory guide!* July 2023. URL: <https://www.analyticsvidhya.com/blog/2021/05/5-regression-algorithms-you-should-know-introductory-guide/>.
- [115] Natasha Sharma. *K-means clustering explained*. Aug. 2023. URL: <https://neptune.ai/blog/k-means-clustering>.
- [116] Pulkit Sharma. *The Ultimate Guide to K-means clustering: Definition, methods and applications*. Nov. 2023. URL: <https://www.analyticsvidhya.com/blog/2019/08/comprehensive-guide-k-means-clustering/>.
- [117] Ravi Singh. *Understanding MLP classifiers: A powerful tool for machine learning*. June 2023. URL: <https://www.linkedin.com/pulse/understanding-mlp-classifiers-powerful-tool-machine-learning-singh/>.
- [118] Beytullah Soylev. *What is SVM classification?* Aug. 2023. URL: <https://beytullahsoylev.medium.com/what-is-svm-classification-36d9f2ffbcc4>.
- [119] Tavish Srivastava. *12 important model evaluation metrics for Machine Learning Everyone should know (updated 2023)*. July 2023. URL: [https://www.analyticsvidhya.com/blog/2019/08/11-important-model-evaluation-error-metrics/#:~:text=Evaluation%](https://www.analyticsvidhya.com/blog/2019/08/11-important-model-evaluation-error-metrics/#:~:text=Evaluation%20metrics)

- 20metrics%20are%20quantitative%20measures, comparing%20different%20models%20or%20algorithms..
- [120] Dedi Supardi. *Detecting objects on images using google cloud vision API*. Sept. 2022. URL: <https://blogs.embarcadero.com/detecting-objects-on-images-using-google-cloud-vision-api/>.
- [121] Michael A. Tabak et al. "Machine learning to classify animal species in camera trap images: Applications in ecology". In: *Methods in Ecology and Evolution* 10.4 (2019), pp. 585–590. DOI: <https://doi.org/10.1111/2041-210X.13120>. eprint: <https://besjournals.onlinelibrary.wiley.com/doi/pdf/10.1111/2041-210X.13120>. URL: <https://besjournals.onlinelibrary.wiley.com/doi/abs/10.1111/2041-210X.13120>.
- [122] Srujana Takkallapally. *Google vision API for image analysis with python*. Mar. 2021. URL: <https://towardsdatascience.com/google-vision-api-for-image-analysis-with-python-d3a2e45913d4#:~:text=Google%20Vision%20API%20detects%20objects,tool%20and%20get%20its%20contents..>
- [123] Einblick Content Team. *Processing text data with scikit-learn's TfidfVectorizer()*. Nov. 2023. URL: <https://www.einblick.ai/python-code-examples/tfidfvectorizer/>.
- [124] Einblick Content Team. *Processing text data with scikit-learn's TfidfVectorizer()*. Nov. 2023. URL: <https://www.einblick.ai/python-code-examples/tfidfvectorizer/>.
- [125] Jalaj Thanaki. *Python Natural Language Processing*. Nov. 2023. URL: <https://www.oreilly.com/library/view/python-natural-language/9781787121423/9742008f-6384-42a4-9711-2721dd6fd382.xhtml#:~:text=Converting%20all%20your%20data%20to,when%20you%20are%20doing%20parsing..>
- [126] Cristina Trocin, Åsne Stige, and Patrick Mikalef. "Machine Learning (ML) diffusion in the design process: A study of Norwegian design consultancies". In: *Technological Forecasting and Social Change* 194 (2023), p. 122724.

- ISSN: 0040-1625. DOI: <https://doi.org/10.1016/j.techfore.2023.122724>. URL: <https://www.sciencedirect.com/science/article/pii/S0040162523004092>.
- [127] Masayuki Tsuneki. “Deep learning models in medical image analysis”. In: *Journal of Oral Biosciences* 64.3 (2022), pp. 312–320. ISSN: 1349-0079. DOI: <https://doi.org/10.1016/j.job.2022.03.003>. URL: <https://www.sciencedirect.com/science/article/pii/S1349007922000500>.
- [128] tutorialspoint. *Classification algorithms - random forest*. Nov. 2023. URL: https://www.tutorialspoint.com/machine_learning_with_python/machine_learning_with_python_classification_algorithms_random_forest.htm.
- [129] Uchidalab. *Uchidalab/book-dataset: This dataset contains 207,572 books from the Amazon.com, inc. marketplace*. Apr. 2017. URL: <https://github.com/uchidalab/book-dataset>.
- [130] upGrad blog upGrad. *Multinomial naive Bayes explained: Function, Advantages and disadvantages, applications in 2023*. Nov. 2023. URL: <https://www.upgrad.com/blog/multinomial-naive-bayes-explained/>.
- [131] V7. *F1 score in Machine Learning: Intro and Calculation*. Nov. 2023. URL: <https://www.v7labs.com/blog/f1-score-guide>.
- [132] Natalie R. Vaders. *The consequences of e-books and e-readers on the cover design industry and the use of book covers as an effective marketing tool: An exploratory study*. July 2012. URL: https://cdr.lib.unc.edu/concern/masters_papers/tx3lqn540.
- [133] Roberto Verganti, Luca Vendraminelli, and Marco Iansiti. “Innovation and Design in the Age of Artificial Intelligence”. In: *Journal of Product Innovation Management* 37.3 (2020), pp. 212–227. DOI: <https://doi.org/10.1111/jpim.12523>. eprint: <https://onlinelibrary.wiley.com/doi/pdf/10.1111/jpim.12523>. URL: <https://onlinelibrary.wiley.com/doi/abs/10.1111/jpim.12523>.

- [134] Vertica. *SVM (Support Vector Machine) for Classification*. Nov. 2023. URL: <https://www.vertica.com/docs/9.3.x/HTML/Content/Authoring/AnalyzingData/MachineLearning/SVM/SVM.htm>.
- [135] W3Schools. *HTML ASCII Reference*. Nov. 2023. URL: https://www.w3schools.com/charsets/ref_html_ascii.asp#:~:text=ASCII%20is%20a%207%2Dbit,are%20all%20based%20on%20ASCII..
- [136] Zhaobin Wang et al. "MSLWENet: A Novel Deep Learning Network for Lake Water Body Extraction of Google Remote Sensing Images". In: *Remote Sensing* 12.24 (2020). ISSN: 2072-4292. DOI: [10.3390/rs12244140](https://doi.org/10.3390/rs12244140). URL: <https://www.mdpi.com/2072-4292/12/24/4140>.
- [137] Wikipedia. *F-score*. Sept. 2023. URL: <https://en.wikipedia.org/wiki/F-score>.
- [138] Meiyin Wu and Li Chen. "Image recognition based on deep learning". In: *2015 Chinese Automation Congress (CAC)*. 2015, pp. 542–546. DOI: [10.1109/CAC.2015.7382560](https://doi.org/10.1109/CAC.2015.7382560).
- [139] Dinesh Yadav. *Categorical encoding using label-encoding and one-hot-encoder*. Dec. 2019. URL: <https://towardsdatascience.com/categorical-encoding-using-label-encoding-and-one-hot-encoder-911ef77fb5bd>.
- [140] Pingping Zhu et al. "Deep learning feature extraction for target recognition and classification in underwater sonar images". In: *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*. 2017, pp. 2724–2731. DOI: [10.1109/CDC.2017.8264055](https://doi.org/10.1109/CDC.2017.8264055).