DEGREE PROJECT IN INFORMATION AND COMMUNICATION TECHNOLOGY,
SECOND CYCLE, 30 CREDITS
STOCKHOLM, SWEDEN 2020

Combining Learned and Analytical Models for Predicting CO2e Emissions in Textile Products

AREF MORADI

KTH ROYAL INSTITUTE OF TECHNOLOGY
SCHOOL OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

Combining Learned and Analytical Models for Predicting CO2e Emissions in Textile Products

AREF MORADI

Master in ICT Innovation (Data Science)
Date: August 19, 2020
Supervisor: Sina Sheikholeslami
Examiner: Amir H. Payberah
School of Electrical Engineering and Computer Science
Host company: Klarna AB
Swedish title: Att förutsäga CO2-utsläpp från textilprodukter genom att kombinera analytiska modeller med maskininlärningsmodeller

Abstract

In recent years, there has been an increased attention to climate change and the effects it might have on human society. One of the main causes of climate change is increased carbon emissions due to different human activities and global scale industrialization. Various approaches exist to mitigate the effects of climate change. One of the largest sources of carbon emissions is the production of consumer goods. Therefore, by influencing the choices consumers make, we can help mitigate some of the effects of climate change due to carbon emissions.

In this thesis, we focus on carbon emissions during the production phase of textile products. To be able to influence consumers in their choice of textile products, we must be able to provide them with detailed information on the emissions of individual products. Such information will allow consumers to compare the products and potentially influence them to choose environmentally friendly products. Such a product will hopefully push the market towards sustainable production.

Our work focuses on providing methods that can be used to create a machine learning model to predict the carbon emissions of textile products. We collaborate with IVL Swedish Environmental Research Institute (IVL) to build an analytical model to calculate the emissions of textile products. Furthermore, we leverage the analytical model to design and compare three machine learning models. We focus on building models that benefit from the knowledge of the analytical model while being scalable with regards to new data. Moreover, we introduce a method to use knowledge in the form of analytical models to bootstrap a machine learning model when labelled data is not readily available. In this way, the machine learning model can benefit from the existing knowledge of the analytical model while being adaptable to new labelled data.

Finally, we compare the three proposed models and discuss the advantages and disadvantages of each model. We also mention the situations where each model performs best.

Sammanfattning

Intresset för klimatförändringar och dess potentiella effekter på samhället har ökat under de senaste åren. En av de främsta orsakerna till klimatförändringar är ökade koldioxidutsläpp till följd av en globaliserad ekonomi och mänskliga aktiviteter som exempelvis resande. En betydande andel av koldioxidutsläppen kommer från konsumtion av konsumentprodukter. Genom att påverka konsumenternas val så kan en bidra till att minska koldioxidutsläppen och i sin tur mildra klimatförändringarna.

I denna masteruppsats analyseras koldioxidutsläpp från produktion av textiler. För att aktivt kunna påverka konsumenternas val så bör de presenteras med detaljerad information om den specifika textilproduktens miljöpåverkan. Denna information tillåter konsumenterna att jämföra produkter med avseende på miljöeffekter, vilket potentiellt kan få dem att prioritera miljövänligare alternativ. Förhoppningen är att denna lösning i förlängningen kommer att driva textilmarknaden mot en mer hållbar produktion.

Syftet med denna rapport är att tillhandahålla maskininlärnings modeller som kan användas för att förutsäga koldioxidutsläpp från textilprodukter. I rapporten så används en analytisk modell för att beräkna utsläpp från textilprodukter, denna modell är framtagen i samarbete IVL Swedish Environmental Research Institute (IVL). Den analytiska modellen används sedan i utformningen och jämförelsen av tre maskininlärningsmetoder. Dessa maskininlärningsalgoritmer är utformade för att kunna kombinera en analytisk modell med nya datapunkter på ett skalbart sätt. Således föreslår vi även att kunskap från en analytisk modell kan användas för att bootstrapa en maskininlärningsmodell. Detta är särskilt nyttigt då få annoterade datapunkter är tillgängliga. Denna metodik medför att modellen kan utnyttja den analytiska modellens kunskap och sedan anpassas till nya annoterade datapunkter.

Slutligen så ställs de tre maskininlärningsmodellerna mot varandra i en diskussion om var och ens fördelar respektive nackdelar. Vi framhåller även i vilka typer av situationer de enskilda modellernas prestationer var optimala.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In recent years, climate change and the effects it will have on human societies has been garnering increased attention. In their 2018 assessment report, the Intergovernmental Panel on Climate Change (IPCC)[1] has predicted catastrophic consequences in various sectors if greenhouse emissions are not mitigated in the next thirty years.

Furthermore, the Climate Change AI group[2] investigated the benefits of using Machine Learning (ML) for preventing climate change and mentions that addressing climate change issues require both mitigating and adapting to changes [1]. Also, various ways in which the ML field can help solve or reduce issues regarding climate change are discussed. Understanding personal footprint and facilitating behavioural change are two of the aforementioned ways that ML can help affect climate change. Personal behaviours and decisions are one of the strongest driving forces in different industries and consequently, the effects of these industries on climate change.

The textile industry has been responsible for a large share of greenhouse gas production in recent years [2], which is mainly caused by *Fast Fashion*. Fast Fashion is the term used to describe clothing designs that move quickly from the catwalk to stores to meet new trends. The collections are often based on designs presented at Fashion Week events. The efforts to reduce the impact of this industry requires action from both manufacturers and consumers.

One of the main methods of influencing consumer behaviour towards choosing environmental-friendly options is raising awareness about their choices

[1]An intergovernmental organization dedicated to providing the world with objective, scientific information relevant to understanding the scientific basis of the risk of human-induced climate change.

[2]A group of volunteers from academia and industry who believe in using ML, where it is relevant, to help tackle the climate crisis.

and the environmental impact of these choices. In the last few years, various businesses have made progress in being more transparent about the effect of their products on the environment and providing detailed reports regarding environmental issues. For example, Zalando and H&M provide sustainability reports annually [3, 4].

In the textile industry, few Life Cycle Assessments (LCA) on the carbon emissions of textile products exist [5, 6, 7, 8]. These assessments provide accurate information on the different sections in the life-cycle of a product. The major issue with LCA's is that they are expensive and time-consuming. This makes it infeasible to use them for raising awareness on the consumer side where we need the emissions of a large number of products. Therefore, a model that can estimate the carbon emissions during the production phase of textile products (for a large number) can help increase consumer awareness and potentially influence them to choose environmentally-friendly products. Some of the major issues in using ML methods for estimating the carbon emissions of products is the lack of suitable data and a high variance in the data that does. Also, most of the data available for building such methods is unstructured.

1.1 Motivation

The consequences of climate change have garnered increased attention in the past few years. The Climate Change AI group state that addressing climate change issues require both mitigating and adapting to changes [1]. These issues have to be addressed both on the consumer side and the manufacturer side. In recent years, many manufacturers in different industries have been dedicating more resources on producing environmental-friendly products. In the textile industry, many manufacturers have begun providing details about the climate impact their products will have. Moreover, some manufacturers are dedicating special product line-ups to environmental-friendly clothes.

According to section 1.2 of The Climate Change AI group report [9], raising consumer awareness about the impact of their choices on climate change has a beneficial effect on their choice of consumer products. By providing details about the impact of buying different textile products and enabling consumers to take this factor into account when deciding on their purchase, consumers can be influenced to choose environmental-friendly options.

To provide details about the carbon emissions of a textile product during its life-cycle using ML methods, the problems regarding missing data and unlabeled data need to be solved.

1.2 Problem Statement

This thesis primarily focuses on finding a solution to raise consumers' awareness about their purchases in the textile industry. Determining the carbon emissions of textile products on a large scale, depends on having sufficient data and a suitable model to enable estimating the carbon emissions of unseen products. Clothing retail companies have different ways of categorizing their products. For example, a retail store might combine hoodies and sweaters in one category while another might have separate categorization for them.

Furthermore, due to the different categorization methods and different details that are disclosed by textile manufacturers about their products, the data required for training such a model has to be collected by manually scraping different manufacturers and unifying the format of their data. Moreover, the different details provided by manufacturers lead to missing features in some datasets which will need to be accounted for.

Finally, labelled datasets regarding the emissions of textile products are not readily available in a large scale. The available data is in the form of LCA studies which are time consuming and not scalable. The lack of labelled datasets makes building a learned model that can generalize well difficult. With the continuation of such a project and with more resources labelled datasets might become available in the future. Although, until such a dataset is available, our model should be able to benefit from the best estimations that exist.

1.3 Research Question

In this thesis, we mainly focus on the following research question:

How can we combine learned and analytical models in a way to benefit from both the flexibility of learned models and the pre-existing knowledge embedded in the analytical model.

Analytical models have advantages like interpretability and being based on expert human knowledge. Moreover, analytical models do not need large amounts of data to be able to calculate outputs accurately. However, manual interference is needed to add new types of inputs or to adapt the model to accurate labelled data. On the other hand, learned models need a large amount of data during their training phase to start being beneficial. However, they can adapt to new datasets with minimal manual interference. In the case of estimating carbon emissions for textile products, one of the main problems is the lack of labelled data which makes training learned models impossible. On the other hand, the available analytical model requires inputs in a specific format and is limited to some input types.

To investigate the main research question, we propose three models. The first model is a regression model trained on data labelled by the analytical model. To determine the best regression model, we perform experiments between well-known regression algorithms. The second model is an ensemble of the analytical model and several classifiers. The classifiers provide the input of the analytical model by using the product information. Finally, the third model is a novel method for benefiting from the pre-existing knowledge of the analytical model while maintaining the benefits of a learned model. This model is mainly inspired by the work in [10].

We compare the models based on different aspects. This results in the following smaller questions:

- How do each of the models react to feature ablations?
- How do each of the models react to datasets with different sizes?
- Which of the proposed models generalize better to unseen inputs?
- In general, which of the three models estimate the carbon emissions more accurately?

1.4 Contributions

The contributions of this work include comparing the performance of the three proposed models, alongside a proof-of-concept for a carbon emissions estimator tool that will estimate the carbon emissions of producing a textile product.

1.5 Ethics and Sustainability

The work in this thesis is directly related to sustainability in the textile industry. One of the main motivations of this work is providing consumers with information about the effect of their choices on the environment. Also, in direct relation with the work of the Climate Change AI group, this work is concerned with using ML to mitigate the effects of climate change. Furthermore, Fast Fashion has been subject to increasing criticism in recent years and one of the most important ways to reduce the effects of Fast Fashion is influencing consumer behaviour [2].

1.6 Outline

This thesis is organized as follows. In Chapter 2, the necessary background information needed for understanding the methods and contributions in this work is provided. Chapter 3 explains the methodology and methods used to build the carbon estimator. In particular, in Section 3.2 the challenges and solutions used regarding the dataset are explained. In Chapter 4, the results and a comparison between the proposed models are presented. Finally, in Chapter 5 a final discussion of the limitations and potential future work alongside the conclusion is provided.

Chapter 2

Background and Related Work

In the following sections, the necessary background information regarding the different aspects of the thesis is provided.

2.1 Assessing Carbon Emissions

The main problem we focus on is assessing the carbon emissions of the production phase of a textile product. Various steps in the production of a textile product contribute to the carbon emissions. These contributions are in different forms. For example, heat and energy needed for producing such products emit carbon in different ways. In the following section, we explain a method for calculating the carbon emissions of the different stages of a products life-cycle.

2.1.1 Life Cycle Assessment

LCA is a methodology for assessing the environmental impact of the different stages of a products life-cycle. The LCA methodology has different variants such as Cradle-to-Gate, Cradle-to-Grave, etc. The Cradle-to-Gate variation is concerned with the stages in the life-cycle of a product up until it is ready to be purchased. The different stages of the life-cycle of a product can be seen in 2.1. Cradle-to-Gate is concerned with the three steps of Raw Materials, Production and Distribution.

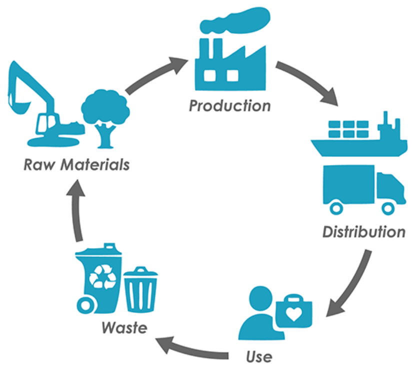

Figure 2.1: Stages of the life-cycle of a product.

The effects of greenhouse gases are not limited to CO_2. Various greenhouse gases other than carbon affect the climate. Although, to make the process of measuring these effects and studying them easier a common unit named CO_2e is defined. CO_2e signifies the amount of CO_2 which would have the equivalent global warming impact. In this thesis, all emissions are measured using the CO_2e.

2.1.2 IVL Swedish Environmental Research Institute

The IVL Swedish Environmental Research Institute AB is an independent and non-profit Swedish research institute in the environmental field and a national knowledge resource for businesses and the state. The purpose of the institute is to develop impartial decision-making bases and provide research with high quality and with application to the needs of society, and to promote ecologically, economically and socially sustainable growth in businesses and society. IVL was jointly founded in 1966 by the Swedish state and national business interests to carry out research on industrial, air and water issues [11].

IVL provided us with their expert knowledge and resources regarding LCA's and textile products. They collaborated with us to develop a carbon emissions calculator for textile products.

2.2 Web crawling

Web crawling is the term used to describe the different steps and methods for automatically visiting different web pages in the World Wide Web and extracting information from them. Web crawling is performed by web crawlers (or web bots) that start with a list of webpage URLs to visit and recursively retrieve and visit other URLs present on that web page.

One of the most common purposes of web crawling is extracting data and building datasets that don't currently exist. For example, in this thesis, we require a large amount of data about textile products which contain the properties that will affect their carbon emissions. Based on our research, no such datasets are publicly available. Therefore, we used a custom web crawler to find, visit and store the data of textile products.

Designing a simple web crawler that visits a small number of URLs and saves the data is a relatively easy task. However, to build a web crawler on a large scale many challenges are encountered.

Most publicly available websites have a relatively high user traffic. To maintain the quality of service, these web pages use preventive measures to stop any fraudulent or abnormal activity. For example, if a high number of requests are made to a web page, the IP of the sender is temporarily blocked. From the point of view of the host web page, a web crawler is the same as a normal user. However, due to the automated nature of a web crawler, a large number of requests for multiple URLs can be sent simultaneously. For a small number of requests, this is not a problem. But, by continuing this type of behaviour, the bot might be marked as "suspicious" and the requests to the web page will be denied. To solve this problem, web crawlers use policies for different aspects of their behaviour. For example, a common policy to prevent them from being marked as "suspicious" is limiting the rate of the requests. Furthermore, as part of the Robots Exclusion Policy (REP)[1], web pages include a robots.txt file on their domain which indicates which type of users and requests are allowed and not allowed on their web page. Web pages with a large number of users tend to have strict rules.

2.3 Machine Learning and Deep Learning

Machine Learning (ML) is used to refer to algorithms that can improve automatically with experience. There are various definitions of ML. In his book, ML [12], Tom M. Mitchell defines ML as follows:

"A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E."

[1] A group of web standards that regulate how robots crawl the web, access and index content

In recent years, ML has been used in various fields and industries. For example, detecting if an email should be considered a spam email or not. ML consists of various methods and algorithms related to automatic learning. Two of the main categories of ML methods are supervised and unsupervised learning. In supervised learning, the algorithm attempts to learn a mapping function from a certain input to an output. The email spam detection mentioned before is one example of supervised learning algorithms. On the other hand, unsupervised learning algorithms attempt to find unseen patterns that exist in a dataset.

In recent years, a specific family of ML methods, Deep Learning, has been garnering increased attention. This method is based on using Artificial Neural Networks to solve problems in the ML domain. The reason for using the keyword *deep* is the use of multiple layers of neural networks. Deep Learning methods can also be used in both a supervised and unsupervised fashion.

2.3.1 Data and Learning

As mentioned previously, ML is mainly concerned with performing a task automatically with minimal human supervision. Any ML algorithm will require data to be able to learn to perform a given task. Going back to the email spam detection example, a supervised model will require a dataset of emails. These emails should be labelled according to the task at hand. In this case, the label will show if the email is spam or not. Similarly, an unsupervised ML algorithm would also need a dataset with enough data to be able to detect meaningful patterns in this dataset. The final end-to-end algorithm that uses the provided data to learn how to perform a specific task is called an ML *model*.

A major part of creating any ML model is having suitable and sufficient data for the given task. The data used in supervised and unsupervised methods have a major difference. As suggested by the name, supervised methods are guided towards the function they must learn. This *supervision* is provided in the form of a label in the dataset. For example, each email in the dataset for building an email spam detection should have a label indicating if it is a spam or normal email. On the other hand, an unsupervised model does not require a label and attempts to learn unknown patterns in the dataset. An example of an unsupervised model is a model that receives a dataset of images with different animals in them and puts the same animals in the same groups.

The process of using a dataset to teach an ML model how to perform a specific task is called training the model. Training a model is an iterative process where the model repeatedly uses the given dataset and tries to improve itself in the given task, until a certain condition is met. The training phase of a model is controlled by parameters called *hyperparameters*. For example, The number of times the model should use each data point for improving itself is a hyperparameter. These parameters are set manually before starting the training phase. Hyperparameters and how they impact the training of a model are discussed in section 2.3.1.

The process of training and improving a model requires some sort of evaluation of how the model is performing. Using this evaluation, the model can decide how it can improve itself. This evaluation is done by using different metrics and a loss function. The choice of these metrics and the loss function usually depends on the task at hand and is discussed thoroughly in the evaluating models subsection. Furthermore, after the training of the model is complete, a final evaluation has to be performed to assess the model according to the specific task it was trained on. This final evaluation is called *testing* the model.

Data splitting

As can be seen in the previous sections, building an ML model consists of different parts. The data used for the different parts of building a model is different. One of the most important reasons for splitting the data for the different phases of training a model is to avoid overfitting. Overfitting is a modelling error that occurs when a function fits a specific dataset too closely and is not able to generalize the given task for unseen data.

To avoid this problem, we split the dataset into two parts. The first part is the training data. This part is used in the training phase of the model and is the data the model uses to learn the mapping between the inputs and outputs of a function. This part of the data is called the *training* set. Usually, a part of the training data is used for finding the best hyperparameters and is called the *validation* set. In the training phase, the model only uses the training and validation sets.

The second part of the dataset is called the *test* set. As the name suggests, the test set is used for testing the model and evaluating its performance. Therefore, the test data is never used in the training phase of the model and is unseen to the model. By doing this, we prevent the model from overfitting on the training set and hence, improve the model's ability to generalize. It is worth mentioning that the process described above is mainly used for splitting the dataset for supervised algorithms. Although, unsupervised algorithms may also need to partition the dataset depending on the problem they are attempting to solve.

Hyperparameters and Optimization

Hyperparameters are parameters set in a model before the training process begins. For example, the number of times the model goes through the training set and uses it to improve itself is called the *epoch* hyperparameter. Different training algorithms have different hyperparameters.

The process of finding the best hyperparameters for a model performing a certain task is called tuning the hyperparameters. As mentioned in the previous section, the validation set is used to tune the hyperparameters. Tuning the hyperparameters is also an iterative process. Different hyperparameters are used to train the model. In the end, the hyperparameters resulting in the best performance on the validation set are used for the final model.

A common tuning method is called GridSearch. In this method, different combinations of all the hyperparameters of the model are considered and used to train the model. The best combination is then used to create the final model. Training a model is equivalent to finding the parameters which minimize the loss function. One of the ways to find these parameters is using optimizers. Various optimizers exist for different models [13].

In this work, we test and train our models using two optimizers, Stochastic Gradient Descent (SGD) [14] and Adam [15]. In SGD, the model calculates the difference between the predictions and the labels of a subset of the data, using the loss function. This difference is then used to calculate the gradient of the error, with regards to each variable. Each variable is then changed in the opposite direction of the gradient. The intuition behind this technique is that the gradient of the error is showing the direction that will make the error rise. Therefore, to reduce the error, we move in the opposite direction. In every step, the variables are changed according to the formula in equation 2.1. SGD is a variation of Gradient Descent. In GD, the gradient is calculated with regard to each single data point. SGD is usually preferred in ML models with a large number of variables because of its efficiency. The variable α is the learning rate and is a hyperparameter. It determines the size of the steps taken after the gradient is calculated.

$$\begin{aligned} &for\ i\ in\ range(m): \\ &\qquad \theta_j = \theta_j - \alpha(\hat{y}^i - {x_j}^i) \end{aligned} \tag{2.1}$$

The second optimizer used in this work is the Adam optimizer. The Adam optimizer improves upon SGD by introducing an adaptive learning rate individually for each parameter.

Evaluating Models

As mentioned in the previous sections, we need to evaluate the performance of a model in both the training and testing phases. The two main aspects of evaluating a model are a loss function and some performance metrics.

A loss function calculates the difference between the outputs predicted by the model and the real outputs. Different functions can be used as the loss function based on the data and the task at hand. The loss function is mainly used in the training phase of the model so the model can understand how it is currently performing and improve itself based on its current performance.

The performance metrics are used to evaluate the final model and also compare models (for example, when tuning the hyperparameters.). Similar to the loss function, various performance metrics exist and they are chosen based on the task at hand [16]. It is worth mentioning that the performance metric for evaluating the model can be the same as the loss function used during training.

Regression and Classification

In this section, we explain regression and classification which are two of the main types of ML models. Regression models are concerned with approximating a mapping function from some inputs to a **continuous** variable. For example, a model that takes the different attributes of a house as input, and outputs the price of the house is a regression model. On the other hand, classification models are concerned with approximating a mapping function from some inputs to a **discrete** label variable. For example, a model which takes an email as an input and outputs a label that determines if the email is spam or not is a classification model. Note that, classification models can output more than two labels. For example, an image classifier might take a picture and predict which kind of animal is present in the input picture. Regression models and classification models have different performance metrics. Some of these metrics are explained in section 4.1.

2.3.2 Artificial Neural Networks

Artificial Neural Networks (ANN) are a type of computing system vaguely inspired by neural networks in the brain. The building blocks of these networks are artificial neurons. Each neuron consists of an input, an activation function and an output. The activation function determines the output of the neuron based on the input.

An important subclass of neural networks, that are responsible for a major part of the AI boom in the recent years, are Deep Neural Networks (DNN). DNNs have three types of layers. The input layer, the output layer and an arbitrary number of layers in between that together are called the hidden layers. Each layer consists of an arbitrary number of neurons. A DNN with a single hidden layer is called a Multilayer Perceptron (MLP).

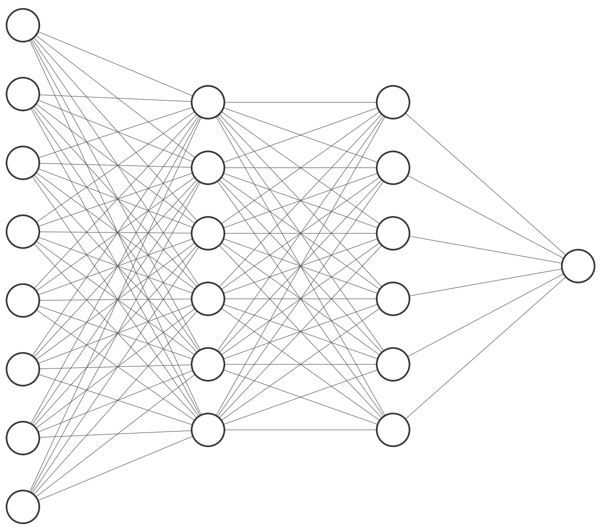

Figure 2.2: A deep neural network.

The way the different layers and neurons are connected together is called the architecture of the neural network. Based on the architecture of the neural network, the networks are used for different ML problems. Similar to classical ML models, ANNs can be used in a supervised or unsupervised fashion. In this thesis, we leverage both supervised and unsupervised ANNs.

2.4 Natural Language Processing

Natural Language Processing (NLP) is a branch of Artificial Intelligence that deals with human-computer communication using natural languages. Any problem regarding computers and natural languages falls under NLP which makes it a broad subject. In this thesis, we focus on a few topics in NLP. Many problems in NLP are solved using Deep Learning [17].

2.4.1 Lemmatisation and Stemming

As mentioned in the previous sections, one of the most important aspects of ML is the data used to train the models. Natural language data is usually in text format, and text data is usually unstructured and varied. For example, words used to refer to similar concepts, might be slightly different. For example, *Jeans* and *Jean* refer to the same thing and it is easy for a human to understand the similarity. However, a computer treats the two words as different words. Two of the main approaches to solve this problem in NLP are lemmatisation and stemming.

Lemmatisation is the process of removing inflectional parts of a word and reducing it to its base dictionary word. The base form of the word is called the *lemma*. For example, the words eat, eating and eaten have the base *eat*. Lemmatisation and stemming are closely related, with the difference being that lemmatisation algorithms consider the context in which a word is used in to deduct its *lemma*.

However, stemming does not consider the context that the word is used in. For example, the result of stemming and lemmatising the word *eating* will be *eat*. However, a lemmatiser would output "good" as the lemma of "better". Whereas, this relation will not be noticed by stemming.

Lemmatisation and stemming are usually performed as a pre-processing for NLP problems and can potentially improve the performance of such models [18].

2.4.2 Text Classification

Classification is one of the most common applications of ML models. Classification is the process of determining a *tag* or *label* for a given subset of data from a pre-defined set of labels.

The process of applying classification to text data is called text classification. For example, the problem of determining if an email is spam or not is an example of text classification.

In recent years, various methods and algorithms have been developed for text classification. Due to a large number of methods for text classification and the scope of this thesis, we will not explain different methods and instead focus on the methods used in this project. However, in [19], the authors provide a survey of the different methods used for text classification in the field.

2.4.3 Word Embeddings

In this thesis, we use word embeddings for solving the problems related to text classification. Word embeddings are a class of techniques used in NLP to map words and sentences to vectors with scalar values. This process allows text data to be used with already existing ML methods such as existing classification methods [20].

The main idea of word embedding is representing text data in a multi-dimensional space in a way that similar words and sentences are close to each other in this space. Various techniques and methods have been developed for acquiring the word embeddings of text data. We only explain the methods used in this thesis. However, in [21], the authors provide an extensive survey on the methods used for creating word embeddings.

This work leverages the FastText library developed by Facebook, to perform text classification [22, 23]. FastText uses a similar method to the Word2Vec method [24], where a hidden layer in an ANN is trained to encode the vocabulary. This layer has fewer dimensions than the vocabulary of the text corpus. However, FastText presents improvements on the Word2Vec model which greatly enhance the performance. These improvements are explained at the end of this section. The Word2Vec method can be trained using two model architectures, Continous Bag of Words (CBOW) and Continuous Skip-Gram [25]. In the CBOW architecture, the model receives the surrounding words of a single word and attempts to predict the word itself.

In the Continuous Skip-Gram model, the model takes a single word as input and outputs the surrounding window of words. Training the Word2Vec model in this way, forces the hidden layer to capture an embedding of those words in n-dimensional space, with n being equal to the number of neurons in the hidden layer. Therefore, we can use the trained Word2Vec model to map words to vectors.

Word n-grams

In the mentioned algorithms, the models can use a sequence of consecutive words instead of individual words. These word sequences are called word n-grams. With n denoting the number of words in the sequence. For example, consider the sentence "Blue tight Jean", the 1-grams would be "Blue", "tight" and "Jean". The 2-grams would be "Blue tight" and "tight Jean".

Furthermore, the vector of a sentence is calculated by combining the vectors of the words in the sentence. The combining can be done by taking the average of the word vectors or by using more complex methods.

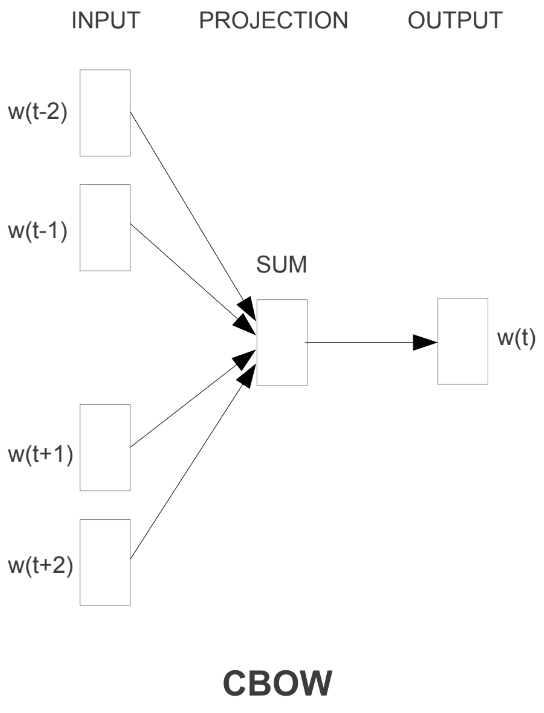

Figure 2.3: The CBOW model architecture. (Source: [24])

These encodings can then be used to perform text classification with widely used classification methods.

FastText

As mentioned previously, FastText introduces improvements to the Word2Vec method. The main improvement is that instead of just using whole words and their neighbours in learning the representations, FastText also leverages smaller parts of each word. The smaller parts of words are called character n-grams. For example, take the word "internet" with a character n-gram of 3, it will be represented as: <"int", "nte", "ter", "ern", "rne", "net">. In this way, FastText extracts much more information from each word. For example, it gains information about the prefixes and suffixes of the words and it handles rare words by having embedded the smaller parts of the word. After each word is split into its character n-grams, the n-grams are used to train the Skip-Gram or CBOW model. Also, when using the trained model to get the representation of a string, FastText first splits all the words into their character n-grams.

FastText also provides a built-in text classifier [22, 23]. The default classi-

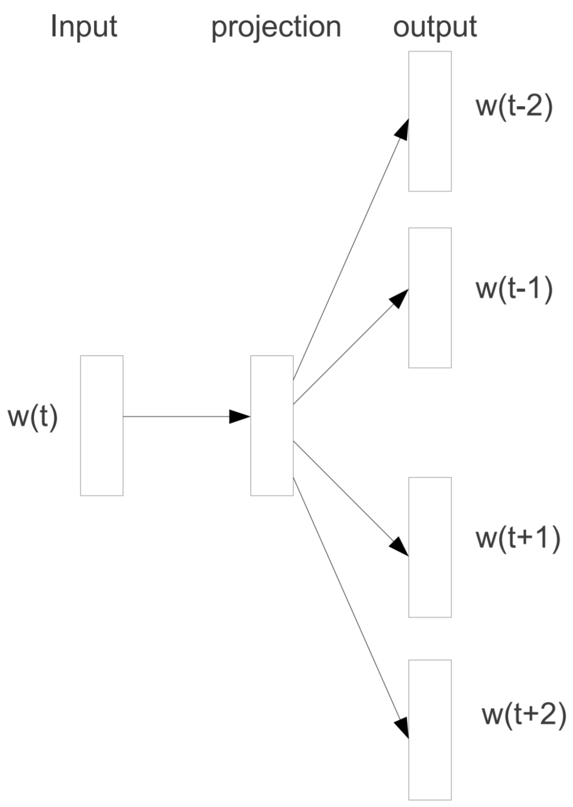

Figure 2.4: The continuous Skip-Gram model architecture. (Source: [24])

fier uses a softmax layer as the output layer of a CBOW model. When learning the word representations, the values of the hidden layer are important. However, for classification, we want the model to learn the relationship between the input words and the outputs (the labels). The output of the softmax layer will be the probabilities of each label.

The FastText model was chosen because of its relatively accurate performance and ease of implementation. The model can be installed using pip in python and includes a classifier out of the box.

2.5 Related Work

2.5.1 Estimating Carbon Emissions

Most of the studies on calculating emissions in textile products are detailed LCA's. These LCA's measure the emissions of each stage in the production of textile products.

Sandin et al, assess the life cycle of six common types of garments in the fashion industry of Sweden [26].

Various other studies provided by either third-party researchers or by manufacturers focus on the LCA of a single product [27, 28, 29].

Levi's provides studies on two specific jeans in their line-up [6].

The North Face has published assessments for two Goretex labelled products, a boot [30] and a jacket [31].

Most of these assessments are concerned with the whole life-cycle of the product and can be used as ground truth data for carbon emission estimation. However, conducting such assessments is usually time-consuming and expensive. Therefore, using them as labelled data is impractical for training ML models.

In this thesis, only the emissions during the Cradle-to-Gate stages of the life-cycle are considered. Using the data gathered through various LCA's and IVL's internal data, an analytical model was developed and provided to us to calculate the carbon emissions of a textile product. This model is a simple linear formula that takes attributes of a garment product as input, and outputs the carbon emissions for that product. The model is explained in detail in section 3.1.

To the best knowledge of the author, no other study has been done that focuses on using ML methods to estimate the carbon-emissions of products from the product's specifications. However, studies have been conducted on imputing missing data in LCA's in general. Ping Hou et al, use a similarity-based approach to impute missing unit process data in LCA's [32]. The results show that missing data can be accurately estimated when less than 5% of the data-points are missing. The estimation performance decreases as the percentage of missing data increases. However, the mentioned study focuses on data during the assessment of the product's life cycle.

None of the studies concerning missing data in LCA's has solely focused on textile products.

2.5.2 Combining Analytical And Learned Models.

As stated in section 1.3, the main research question we focus on is how to combine learned and analytical models to predict the carbon emissions during the production phase of a textile product.

Similar studies in other fields have attempted to use the combination of an analytical model and ML model to maintain the benefits of both.

One of these studies [33] is mainly concerned with the interpretability of ML models which is not directly related to our work. In [33], the authors propose an unsupervised grey-box architecture where they use a black-box model to enlarge a labelled dataset and use the enlarged dataset to train the white-box model. This method is not useful for our work since one of the major challenges is the lack of true labelled data.

Another study uses the method of combining an analytical and learned model to reduce the amount of data needed for predicting the action effect of a robot arm when hitting an object [34]. Traditionally, analytical models are used to predict such effects by using the current physical state as input. In [34], the authors leverage an ML model to predict the physical state according to the analytical model and then proceed to use the analytical model to predict the action effect and also improve the performance of the models when dealing with novel states. In this way, they reduce the amount of training data needed or predicting the action effect. The results show that the white-box model performs better than a pure ML model with the same dataset. The approach used in [34] differs from our work mainly because of the complexity of the analytical model for a physical state. Their analytical model is concerned with an initial state of the object and calculating the effect of different actions. Also, the model relies on visual data to determine the initial state of an object. However, in our case, the analytical model is a much simpler linear model and again in the case of [34] true labelled data exists.

Furthermore, in [35] the authors leverage both analytical and learned methods by using them in an ensemble model. They introduce three ensemble models to combine analytical and learned models. The models are defined as follows:

- K Nearest Neighbors (KNN): during the learning process, this algorithm evaluates the accuracy that can be achieved by the selected analytical model(s) of the target system and by one (or several) black-box ML approaches (e.g., Decision Trees, Artificial Neural Networks, Support Vector Machines) in points of the features' space that were not included in the training sets used to build the ML-based learners (namely, a validation set). When used to predict the performance achievable in a configuration c, the average error achieved by the AM model(s) and by the ML-based learner(s) across the K Nearest Neighbors configurations belonging to the validation set is used to determine which prediction method to choose.

- Hybrid Boosting (HyBoost): in this technique, a chain (possibly of length one) of ML algorithms are used to learn the residual errors of some analytical model. The intuition is that the function that characterizes the error of the analytical model may be learned more easily than the original target function that describes the relationship between input and output variables. With this approach, the actual performance prediction in operative phases is based on the output by the analytical model, adjusted by the error corrector function.

- Probing (PR): The idea at the basis of this algorithm is to use ML to perform predictions exclusively on the regions of the features' space in which the analytical model does not achieve sufficient accuracy (rather than across the whole space). To this end, two learners are exploited. Initially, a classifier is used to learn in which regions of the features' space the AM incurs a prediction error larger than some predetermined threshold. In these regions, a second black-box regressor is trained to learn the desired performance function.

Similar to previous studies, the methods mentioned in [35] rely on having labelled data which is not the case in our study. Therefore, the main benefits of these methods are not present in our problem space. Specifically, The HB method relies on the learned models correcting the analytical model by learning the error residuals, which is not possible when the labelled data is generated with the analytical model itself. Furthermore, the Probing method relies on knowing the feature spaces in which the analytical model does not perform well and to cover those spaces using a learned model, which again requires labelled data to determine these spaces.

Finally, in [10], which is the work most related to this thesis, the authors introduce a new technique called bootstrapping to combine analytical and learning models. The main idea behind this method is to augment data using an existing analytical model and use this data to *bootstrap* (pre-train) the ML model. In this way, the ML model will benefit from the constraints that exist in the analytical model. After the initial bootstrapping phase, the black-box model is used in real-life scenarios to predict the performance of elastic computing services. In this process, more labelled data becomes available to the model. This data can be used to iteratively train the model. In this way, the model slowly diverges from the learned function when trained on the analytical model and goes towards the real distribution function that the data is generated from. Furthermore, the authors provide and discuss different methods to give weights to the training data. Intuitively, it is clear that we would like the ML model to rely more and more on actual real-life data. Therefore, the authors provide various ways to determine how much weight should be given to new data and how much to the augmented data generated by the analytical model.

In this work, we introduce a model inspired by the bootstrapping technique and use it in an ensemble model. Furthermore, we introduce a novel method to assign weights to the real labelled data and augmented data which is specific to our problem space and model.

2.6 Limitations

Based on the scope of the thesis, the goal is to provide a proof-of-concept for such a carbon emissions estimator. Based on the results, building this estimator is feasible.

However, the models presented in this thesis perform on a limited list of fabric types which were defined in collaboration with IVL. For example, the current analytical model can not calculate the carbon emissions for shoes.

Moreover, the problem of not having access to large amounts of ground truth data makes the process of evaluating the models difficult, since the current emissions might be biased in favor of the analytical model.

Finally, as is mentioned in section 3.2, various parts of the dataset are defined and transformed manually. These manual processes should be completely replaced by different models to be able to use such a tool in a production environment. For example, an ML model that can extract the country of production of a product directly from its web page is needed.

Chapter 3

Methods

In this chapter, we discuss the methodology used in this study and describe the dataset and how it was created.

We start by discussing an analytical model developed in collaboration with IVL. The analytical model is the backbone for labelling the datasets and is also directly used in one of the proposed models. The process of gathering and labelling the dataset alongside the features of the dataset is explained in section 3.3. We then proceed to describe three models considered for predicting carbon emissions in section 3.3. Each model performs a data transformation on the constructed dataset and predicts the carbon emissions. The transformed dataset for each model is then used to train each of the models to predict the carbon emissions.

The first model is a deep regression model, from this point on, we refer to this model as the *black-box* model. The second model is an ensemble model and is a combination of two text classifiers and the analytical model, from this point on, we refer to this model as the *white-box* model. Finally, the last model is a combination of bootstrapped regression models and embedding layers. We refer to this model as the *grey-box* model.

We use an empirical approach to compare the performance of the mentioned models. For this purpose, we need a datasets of textile products. The process of creating this dataset is discussed in section 3.2. The dataset is then used to validate the models and compare them. The choice of methodology is explained by the fact that the models consist of several sub-models. This makes it difficult to provide formal proofs. Also, all previous studies, use an empirical approach to compare the final results.

3.1 Analytical Model

The first step for calculating and predicting the carbon emissions of textile products is obtaining an analytical model which can be used for labelling datasets. This model was designed and implemented in collaboration with IVL experts. They provided us with knowledge about the LCA of a textile product and presented us with the features needed to determine the carbon emissions of producing a textile product. This model is a simple linear formula which calculates the Cradle-to-Gate carbon emissions for textile products. The model has six inputs. The influence of the inputs on the output is determined by constant coefficients that are calculated by IVL experts.

Categorization Scheme

As mentioned previously, different merchants might have different categorization schemes for their textile products. The first step to create the analytical model is defining the possible categories so the coefficients associated with those categories can be calculated by the experts. Therefore, we introduce and use a custom categorization scheme to be used by IVL and the models. In the prediction phase of our models, the category provided by the merchant and other details about the product are used to infer a category based on our custom scheme. The inferred category is then used to predict the carbon emissions. The custom scheme provided by us can be seen in Table A.1. The process of creating the analytical model is shown in Figure3.1.

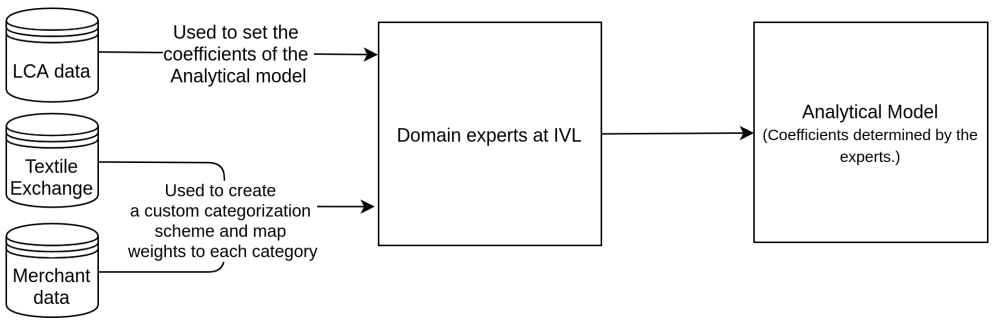
LCA data
Used to set the coefficients of the Analytical model
Textile Exchange
Merchant data
Used to create a custom categorization scheme and map weights to each category
Domain experts at IVL
Analytical Model
(Coefficients determined by the experts.)

Figure 3.1: Process of creating the analytical model.

After creating a custom categorization scheme, we used the data provided by merchant B in their product pages as a base for the possible values of each input. These inputs were used by IVL to design the final analytical model. The analytical model can be divided into 9 separate parts. Each of these parts depend on a subset of the inputs. An overview of the analytical model can be seen in Figure3.2

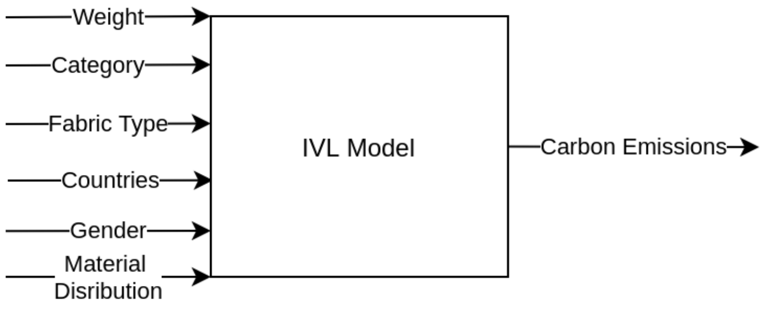

Figure 3.2: The analytical model.

As it can be seen in Figure 3.2, the model takes 6 inputs which are explained below:

- **label:** The category of the product based on the custom categorization presented in this thesis. (text)
- **weights:** The weight of the product based on the category, gender and the weight mappings presented in this thesis. (grams)
- **fabric type:** The type of fabric in the product. This value can be Knitted, Woven or Unwoven. (text)
- **country:** The country the product was produced in. In case this feature is missing from the dataset, China is used. (text)
- **gender:** The gender the product was intended for. (text)
- **main_*:** The features starting with main_ represent the material used in the product. (percentage)

As mentioned previously, the analytical model can be divided into 9 smaller functions which we formally define below:

$$\begin{aligned} carbon\_emissions(&\\ category,&\\ materials,&\\ weight,&\\ country,&\\ fabric\_type,&\\ gender)& \end{aligned} \tag{3.1}$$

Equation 3.1 shows the function of the analytical model. The function is equal to the sum of the following smaller functions.

$$cutting\_sewing\_impact(category, weight, gender, country) \tag{3.2}$$

$$wet\_treatment\_impact(category, weight, gender, country) \tag{3.3}$$

$$fabric\_production\_impact(weight, country, fabric\_type) \tag{3.4}$$

$$yarn\_production\_impact(weight, country, material) \tag{3.5}$$

$$fibre\_production\_impact(weight, country, material) \tag{3.6}$$

$$packaging\_impact(weight) \tag{3.7}$$

$$distribution\_impact(weight, country) \tag{3.8}$$

$$retail\_impact(weight) \tag{3.9}$$

$$transport\_impact(weight) \tag{3.10}$$

An overview of the function can be seen in Figure 3.3.

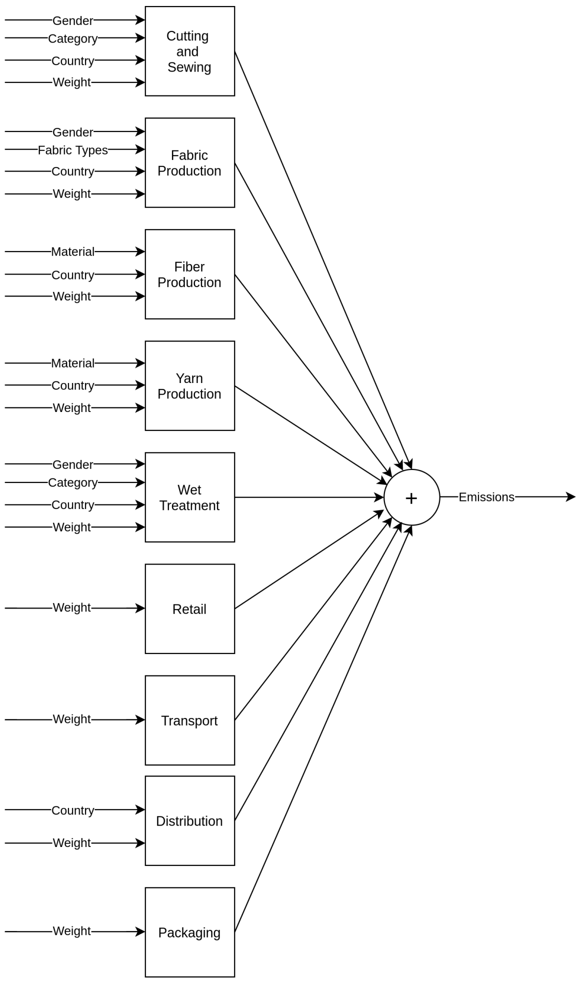

Figure 3.3: Different functions of the analytical model.

3.2 Dataset

The next step of building a carbon estimator is constructing a suitable dataset and labelling it. The dataset used in this work has been scraped from two online markets. We will refer to the two online markets as "merchant A" and "merchant B". The names of the online markets are hidden in this work due to privacy concerns. A custom web crawler was developed and used to scrape the datasets.

First, we obtained a list of URLs corresponding to products in these two manufacturers. The list of product URLs that are used for scraping the datasets is determined by crawling through the URLs in a manufacturers' domain and using a classifier to determine if the URL is a product or not. The classifier for this task was provided by Klarna.

After obtaining the URLs of the product webpages, they are scraped and saved in HTML files. After saving the product pages in HTML files, the files are parsed and the relevant data is extracted in CSV format. For scraping the web pages, a custom MapReduce [36] based framework is used to parallelize the scraping jobs. This framework is also provided by Klarna. Since some of the data provided in online shops are provided dynamically, e.g., a user has to click on a button on the product web page that fetches additional details about the product, a web traversal framework was used to mimic this behaviour and perform the actions needed to retrieve the data. The web traversal framework was also provided by Klarna. The pseudocode of this procedure is shown in algorithm 1.

Algorithm 1 Pseudocode for scraping data from merchant B

```
1: procedure ScrapeURLs($url\_list$)
2:     for each $url$ in $url\_list$ do
3:         while $url.has\_unclicked\_infobutton()$ do
4:             $click(url.next\_button())$
5:         end while
6:         $url.save\_html()$
7:     end for
8: end procedure
```

The gathered webpages are saved in HTML format. After that, the HTML files are parsed and the information is extracted in CSV format. This results in two CSV files. One for merchant A and one for merchant B. We perform a pre-processing step where the name of the columns are unified and extra information from the datasets are left out leaving one dataset containing the data of both merchants.

3.2.1 Data - Features

After saving the HTML files of the product pages, the files are parsed and the extracted information of each merchant is pre-processed and converted to one CSV.

Parsing HTML files

We read and parse the HTML files using BeautifulSoup [37]. The parsing process is slightly different for each manufacturer since the webpages have different layouts and provide different information about the product. When parsing the HTML files, the HTML elements are mapped to one of the columns used in the CSV file for each manufacturer. The mappings for each of the manufacturers are shown in Table 3.1. Note that the HTML elements have been replaced with fake elements to make the real merchants untraceable.

Merchant A	
HTML Element	**Column in Dataset**
<div class="material">	Material Composition
<ol class="category">	Category of Product
<div class="name">	Name of Product
<div class="productdescription">	Text Details 1
Merchant B	
HTML Element	**Column in Dataset**
<dl class="details-list">	Material Composition
<li class="cat-list">	Category of Product
<li class="cat-list">	Name of Product
<ul class="countries-list">	Country of Production
<p class="desc">	Text Details 1

Table 3.1: The mappings of the HTML elements to columns in the CSV files.

Note that these mappings are created manually by looking at the product pages of the two merchants. After reading an HTML file with BeautifulSoup, we use the *find* and *findall* methods provided by the API to find the required HTML elements.

For the list elements, we iterate through the items of the list and add all of them to its respective CSV column. For the other elements, we take the value of the element and insert them in the CSV file.

In the end, each of the mappings will result in one column in the CSV file. The code snippet in Figure 3.4 shows the code for retrieving the "Text Details 1" column. As can be seen in the code, after finding the element with the proper class, we iteratively parse through the inner elements of it to reach the values that are of interest to us. Defining the inner elements is currently a manual process.

The parsing process results in two CSV files. One for merchant A and one

```
def extract_basic_information(soup):
    try:
        base_details_list = soup.findAll("div",
           {"class": "productdescription"})[0]
        category = base_details_list.findAll("p",
           {})[0].text
        details_list = base_details_list.findAll("ul",
           {})[0]
        details = ""
        for detail in details_list:
            if isinstance(detail, NavigableString):
                continue
            details += str(detail.text) + "$$"
        return {"info_features_names": ["category",
           "details"], "info_features_values":
           [category, details]}
    except Exception as e:
        return{"info_features_names": [],
           "info_features_values": []}
```

Figure 3.4: Code snippet for extracting the text details in an merchant A product page.

for merchant B. The columns for the two CSV files are shown in Table A.2.

Pre-Processing the CSV files

At first, the columns for the two CSV files are slightly different. Therefore, we perform a pre-processing step where the useless text data is ignored and the columns of the two CSV files are merged into a final dataset. The pre-processing step for each column is shown below:

- **Material Composition**: The material compositions provided by both merchants are in a key-value format. For example, the materials of a product might be: Cotton: 50%, Polyester: 50%. To be able to use these values in our models, we separate and convert the values to columns. We do this by adding a column for each possible material and assigning the percentage of the material to the corresponding column. The material columns are in this format: "main_*". For example, "main_cotton".

- **Category of Product**: The categories provided by the merchants follow different schemes. For this column, we map each original category to a category in our custom categorization scheme. Also, the categories are Hierarchical. For example, the category of a t-shirt may be: Men > T-shirt > Polo. We split these categories and insert each of them in one of the following columns based on their order: cat0, cat1, cat2 and cat3. The category mappings are first done on the cat3 column and the products that still have an undefined final category are mapped once again using the cat2 column. The cat1 column is then concatenated to the text column of the final dataset. Finally, the gender column is created by making a copy of the cat1 column which contains the gender.

- **Name of Product**: The names of the products are trimmed and leading and trailing white spaces are removed. The name of the product is then concatenated to the text column of the final dataset.

- **Country of Production**: The names of the products are trimmed and the leading and trailing white-spaces are removed. Also, missing countries are replaced with "unknown". Note that merchant A does not provide the country of production, so all products from merchant A have "unknown" as their country. Merchant B provides more than one country for some products and in these cases, we take the first country on the list as the country.

- **Text Details 1**: The text details of the products are trimmed and the white-spaces are removed.

In addition to the steps mentioned above, all of the text data in the CSV files are converted to lowercase letters, they are trimmed and the extra white spaces are removed. Also, all variations of "t-shirt" is manually replaced with "tshirt". After this, the stop words are removed and all words are stemmed to their root word. Finally, the rows with "nan" values are dropped.

Due to the vast variety of information provided by e-commerce websites about their products, the information might be missing some important features for calculating the carbon emissions. For example, one of the most important features is the country of production. The country of production is absent from the details provided by many of the textile manufacturers, including merchant A. The statistics of the most common categorical features of our dataset is shown in table 3.2.1

Variable	Outcome	Count	Percentage
country	unknown	8399	67.22
	china	1338	10.71
	bangladesh	1220	9.76
	cambodia	321	2.57
	turkey	298	2.38
garment_type	woven	4587	36.71
	uknown	4124	33.01
	knitted	3784	30.28
gender	women	7631	61.07
	men	4864	38.93
label	dress	2360	18.89
	tshirt	1582	12.66
	shirt	1336	10.69
	sweater	1217	9.74
	pants	943	7.55

Table 3.2: Statistics of the top 5 elements in each category.

The missing data and the unstructured format of the data need to be accounted for when training an estimator model. More details about the method each model uses to deal with this problem are explained in each models data subsection.

The most important missing features that impact carbon emissions are the weight and fabric type.

Below we explain each feature and its type in the final dataset in detail:

- **name:** The name of the product as presented by the manufacturer. (text)
- **text:** The concatenation of the different unstructured text information of the product that is available on the product page. (text)
- **label:** The category of the product based on the custom categorization scheme presented in this thesis. This feature is added manually and hence cannot be used directly as training data as we cannot perform manual labelling for unseen data. (text)
- **weight in grams:** The weight of the product based on the category, gender and the weight mapping presented in this thesis. This feature is added manually and hence cannot be used directly as training data as we cannot perform manual labelling for unseen data. (float)
- **fabric type:** How the fabric of the product was formed. This value can be Knitted, Woven or Unwoven. This feature is added manually and hence cannot be used directly as training data as we cannot perform manual labelling for unseen data. (text)
- **country:** The country the product was produced in. (text)
- **gender:** The gender the product is intended for. (text)
- **main_*:** The features starting with main_ represent the material used in the product (excluding none main parts of the product, such as Linen). (percentage)

Note that the materials feature is represented by a column for each possible material type in the *main_*:* columns. However, on the website, the material composition is provided in text format. We parse each material and its amount and then transpose them to have a column for each material. This results in a sparse dataset when considering the *main_*:* columns.

Finally, to be able to reliably evaluate the different models, we split the dataset into three subsets. First, a dataset where all features are available and no feature is missing. This is effectively a subset of the merchant B data since the products in merchant A are missing the country feature. The second and third datasets are the data points where the category and fabric type features could be labelled successfully. In this way, we have larger datasets for training the category and fabric type classifiers.

Exploratory Data Analysis

In this section, we conduct exploratory analysis on the features, focusing on the correlation between the features.

A heatmap of the continuous data that shows the correlations between the materials can be seen in figure 3.5. It can be seen that the materials used in the garments have little correlation with each other.

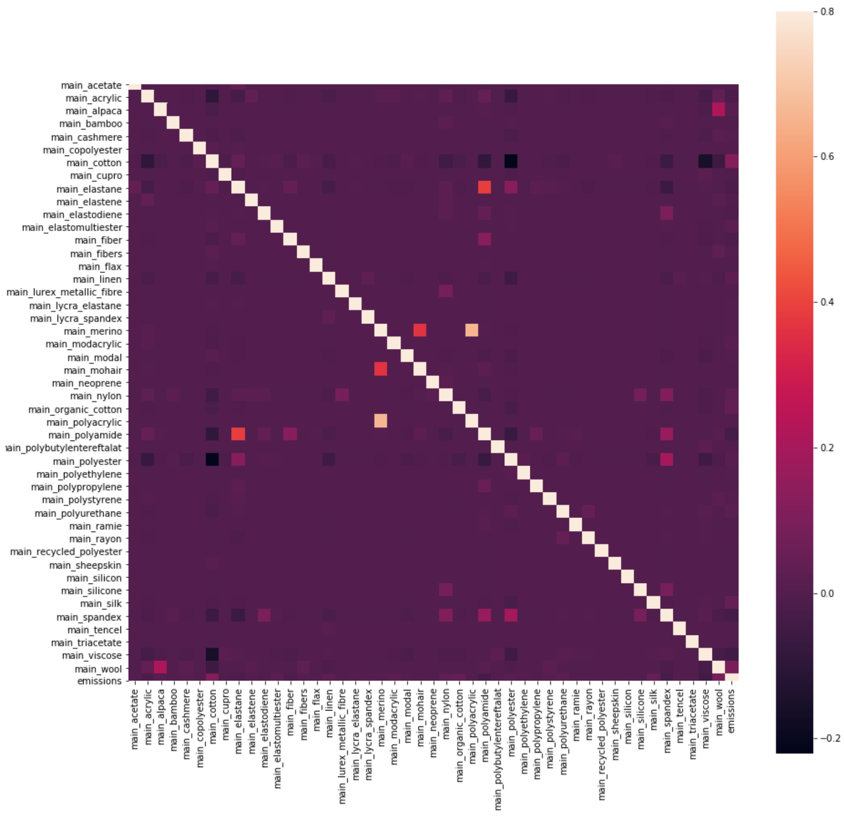

Figure 3.5: Heatmap of the correlation of different materials.

Next, we analyse the correlation between the category of a product and its material composition. To analyse this, we calculate the average percentage of each material, in regards to the category. The results are shown in Figure 3.6. Based on the heatmap, Polyester and Cotton are present in a lot of different categories which match our expectations. A lot of the materials presented have a low average share in the products which points to the fact that the data is not balanced well.

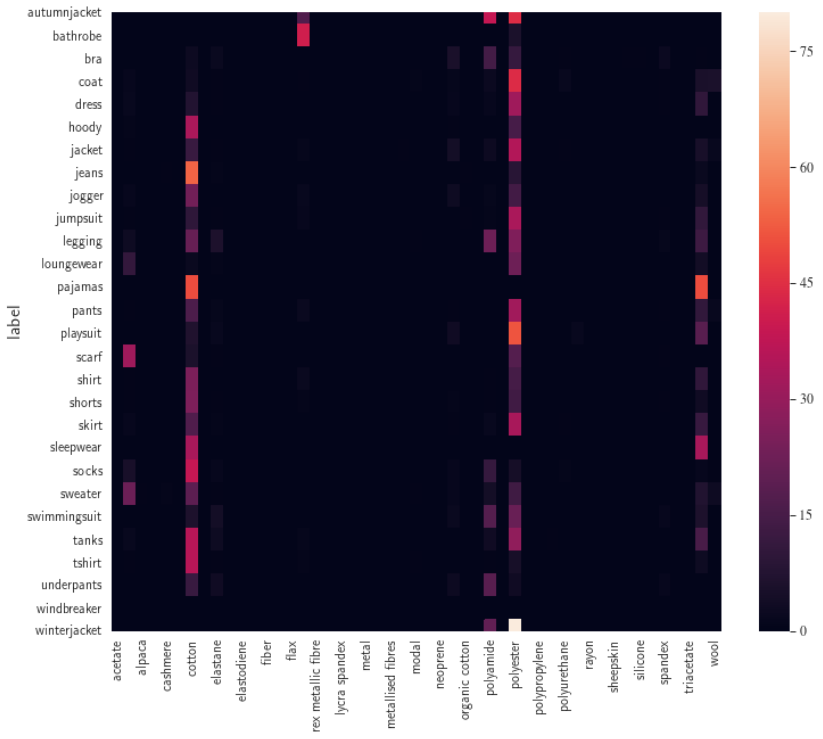

Figure 3.6: Heatmap of the average percentage for the materials based on each category.

Another interesting aspect of the data to look into is if the different materials come from specific countries. To analyse this, we calculate the conditional probabilities of a country, given the material. The results are shown in Figure 3.7.

A large portion of the materials have an unknown country of origin and we can ignore them. It can be seen that China and Bangladesh provide a major share of the materials. Also, some materials in the dataset are provided by specific countries. For example, Cashmere is mainly from India.

Similar to the previous heatmap, We calculate the conditional probabilities of a country given the category of the garment. This can potentially tell us if specific garment categories are usually made in specific countries. The results are shown in Figure 3.8. We ignore the unknown country again and it can be

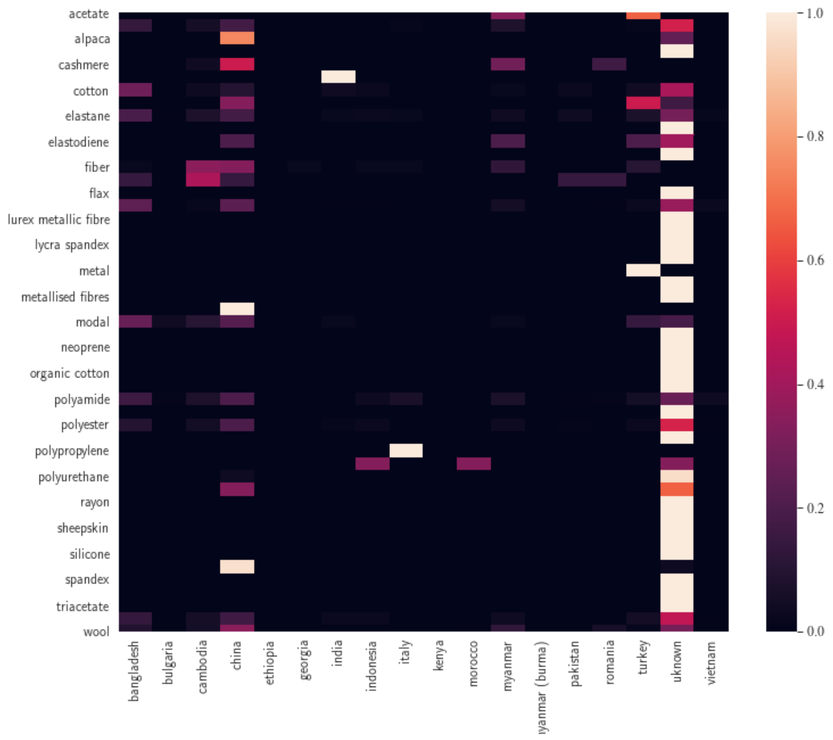

Figure 3.7: Heatmap of the conditional probability of a country given the material.

seen that many of the categories are from China. Some categories do come from specific countries. For example, the "pajamas" category is mostly from Bangladesh.

3.2.2 Data - Labels

A major challenge in estimating carbon emissions is the lack of labelled data. The existing labelled datasets are in the form of LCA's. LCA's are expensive and time-consuming to conduct and only a few of them exist for common garments. Therefore, the data had to be labelled manually.

To generate labelled data, we use the model provided by IVL and the data scraped from merchant B and merchant A. To the best knowledge of the author, merchant B provides the most complete set of features required by the analytical model. The merchant B data includes the country of production but merchant A does not include the country.

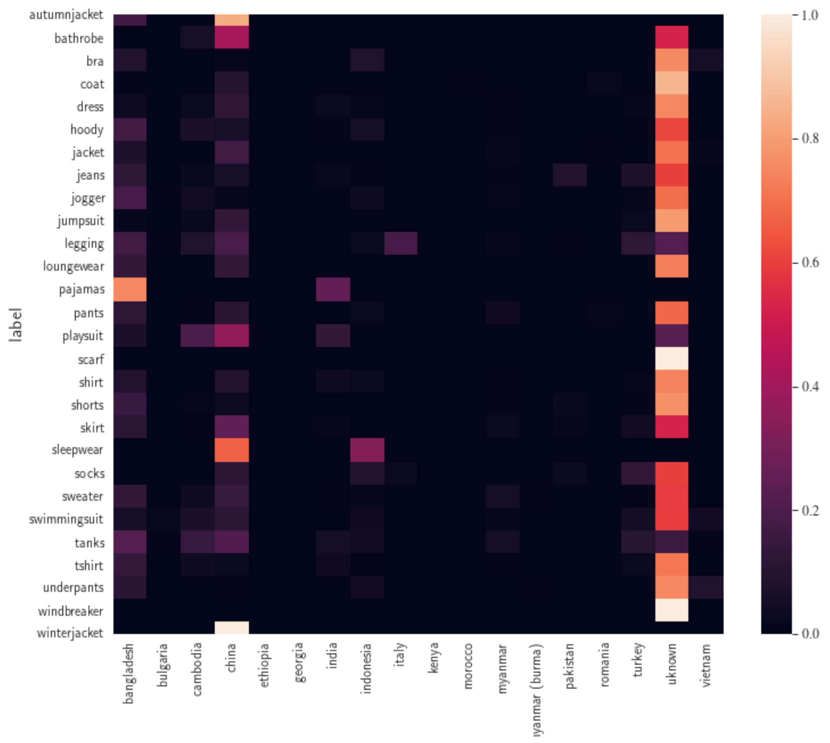

Figure 3.8: Heatmap of the conditional probability of a country given the label.

This data is manually transformed into the format needed by the analytical model. The analytical model requires the following inputs to calculate the carbon emissions of a product:

- **Materials**
- **fabric type**
- **Country**
- **Weight**
- **Gender**
- **Category**

Since e-commerce websites do not follow a unified categorization method the categories in different manufacturers are not identical. To solve this problem we must introduce a customized categorization scheme to unify the category of the products for different manufacturers. This categorization scheme is presented in Table A.1. This custom categorization is defined manually based on input from experts at IVL and a guideline for calculating weights of textile products taken from Textile Exchange [38].

The categories provided by each manufacturer is then mapped to the new categories. The new categories are inserted into the "label" column in the dataset and also mapped to a weight. The weight mapping is done by using data from shipping companies and the Textile Exchange guide that provides an average weight for different categories of clothing products.

Furthermore, the analytical model expects the fabric type of the product as an input to calculate the carbon emissions. The fabric type is manually labelled in the products by checking if the details provided for the product contain specific keywords relating to the type of garment. This data is inserted into the fabric_type column of the dataset. The keywords used for determining each fabric_type are shown in Table 3.3. For garments that are missing any of these keywords, the fabric type is labelled as "unknown".

Woven	**Knitted**	**Non-Woven**
woven	knit	non-woven
weave	jersey	nonwoven
denim	-	-
twill	-	-
satin	-	-

Table 3.3: Keywords used to label the fabric types.

Using the features and the analytical model, the dataset is labelled and the emissions are added to the dataset. The emissions data is inserted in the emissions column. The type and description of the column is shown below:

- **emissions in CO_2e per kg:** The Cradle-to-Gate emissions of the product. (float)

Notes on the Final Dataset

This dataset is separated into a test and train set to be used by the models. The test set is 20% of the whole dataset. All mentions of the data in a training context refer to the training data. Furthermore, to create more accurate performance metrics we only use data points where all the inputs needed for assigning a label are available. This means that for training and evaluating models which rely on the category label, we only use data points which don't have a category of unknown. The dataset containing labelled categories consists of both merchant A and merchant B. Both merchants provided sufficient information to label the categories. Similarly for training models which rely on the fabric type, we only use data points where we could label the fabric type. Again, the dataset with labelled fabric types consists of products from both merchant A and merchant B where the fabric type could be labelled using the text details. Finally, for training and evaluating end-to-end models for the carbon emissions data we only use data points where all inputs needed to calculate the final emissions were available. This effectively means we only use the merchant B data points for the final evaluations as they were the only data points which included the country. In the end, we have three subsets of the dataset. One for models relying on the category. One for models relying on the fabric type. And finally, one for the carbon emissions themselves which are basically data points that don't have a value of unknown for the country, fabric type and category fields.

3.2.3 Augmented Dataset

As mentioned previously, the grey-box model proposed in this work uses augmented data to bootstrap deep regression models. To create augmented datasets, we use the analytical model and all the different combinations of the values of each input. The number of possibilities for each input is as follows:

- **Materials:** There are 31 possible materials in the current analytical model and each material can have a percentage between 0 and 100 in each garment.
- **Fabric type:** There are 6 possible fabric types defined in the analytical model and each garment can have one fabric type.
- **Country:** There are 20 possible countries possible for a garment and each garment can have one country of production.
- **Weight:** In the current version of the model, the weights are tied to the categories of a garment so they do not affect the number of possibilities.
- **Gender:** There are 2 possible genders in the current version of the model.

- **Category:** There are 35 possible categories for a garment and each garment can have 1 category.

As can be seen in the equations of the analytical model in section 3, the analytical model can be divided into 9 smaller functions. As was mentioned before, in the current project, the weight of a garment is tied to the category and gender of the product. This is because the merchants do not provide the weight of the garments. Therefore, when generating the augmented datasets, instead of the weight input we generate all combinations of the category and gender which can then be statically mapped to the weight.

It can be shown that the number of different combinations for the dataset is 394,506,000. Obviously, this dataset is too large to be practically used to bootstrap a regression model. However, as mentioned before the analytical model can be divided into 9 smaller functions where each of the functions depends only on a subset of these inputs. These functions and the number of input combinations for each of the them are shown below:

- $cutting\_sewing\_impact(category, country) : 750$
- $wet\_treatment\_impact(category, country) : 750$
- $fabric\_production\_impact(category, weight, gender, country, fabric\_type) : 4,200$
- $yarn\_production\_impact(category, weight, gender, country, material) : 1,643,775$
- $fibre\_production\_impact(category, weight, gender, country, material) : 1,643,775$
- $retail\_impact(category, weight, gender) : 70$
- $distribution\_impact(category, weight, gender, country) : 700$
- $transport\_production\_impact(category, weight, gender) : 70$
- $packaging\_production\_impact(category, weight, gender) : 70$

It should be noted that, for the $yarn\_production\_impact$ and $fibre\_production\_impact$ functions, where the material feature is used to calculate the emissions, we assume that each category can have at most 2 materials. Therefore, when augmenting the datasets, we select two materials out of 31 and distribute 100 (the total share of the materials) between them.

Each of the regression models used in the grey-box model represents one of these functions and consequently uses its own augmented dataset. As it was shown, dividing the functions and the regression models in this way reduces the number of augmented data points drastically and makes it possible to train the regression models on the datasets for multiple epochs.

We also considered using sub-sampling to bootstrap the model. This would solve the issue of having a large dataset. However, the output of the functions which rely on the material of the product, change monotonically to the percentage. We suspected that sub-sampling the dataset would make the regression model become biased on some percentages and not be able to generalize well to other percentages. Therefore, sub-sampling was not performed.

3.3 Models

In this section, we describe the models and their sub-models, alongside how the dataset is used differently for each model.

The white-box model consists of a combination of the analytical model and two text classifiers.

The black-box model is a deep regression model that will be trained on the manually labelled dataset.

Finally, the grey-box model is an ensemble of bootstrapped regression models alongside embedding layers.

All models take the information provided on the product page as input, and output the carbon emissions for that product. The models are described in detail in the following sections.

3.3.1 White-box Model

The white-box model consists of two parts. The first part includes multiple classifiers which will predict the inputs required by the second part. The second part is the analytical model. The high-level architecture of the white-box model can be seen in Figure 3.9.

The main idea behind this model is that by including the analytical model alongside the ML classifiers, we estimate the carbon emissions more accurately. Also, we will be able to tell the customers where the difference between the carbon emissions of two products come from.

The category, fabric type and weight features are currently predicted by the text classifiers and the remaining inputs are passed directly from the page data to the analytical model. The analytical model then calculates and outputs the

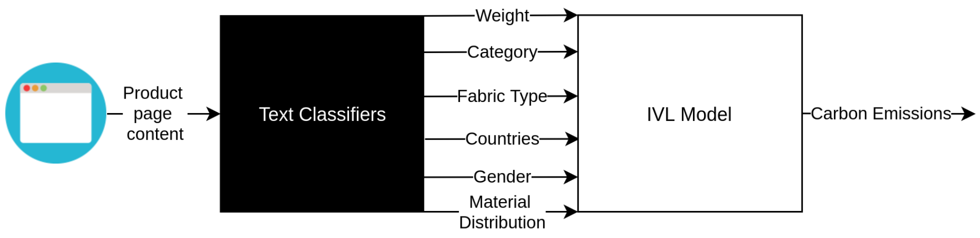

Figure 3.9: Architecture of the white-box model.

estimated carbon emission of the product. In this way, this model intrinsically captures the prior knowledge provided by the analytical model while being able to infer the category and fabric type using the text classifiers. Note that, in cases that both classifiers predict the category and fabric type correctly, the model predicts the emissions accurately.

Data For The white-box Model

In the white-box model, the analytical part (analytical model) expects certain inputs to be able to calculate the carbon emissions. As seen in section 3.2.2, three of the inputs for the analytical model are assumed to be directly extracted from the product page, namely, country, gender and material. For the remaining three features, the fabric type and category are added in the final dataset manually as explained in section 3.2.2. The weight is also calculated by mapping each category to a weight for that category. This mapping function is developed by us, and uses the average weight of each category for assigning the weights. However, we cannot perform the mentioned manual processing in the prediction phase. Therefore, in the prediction phase, the fabric type, category and weight have to be predicted using the text classifiers. To achieve this, the fabric type and category columns are used as labels during the training phase to train the category and fabric type classifiers.

The two text classifiers are explained in the next section.

Text Classifiers

The first part of the white-box model is the text classifiers used for predicting the inputs of the analytical model. The text classifiers take the unstructured text information extracted from the web page as input, and predict the category and fabric type inputs needed by the analytical model.

The final design of this part consists of two text classifiers. The detailed view of the text classifiers can be seen in figure 3.10.

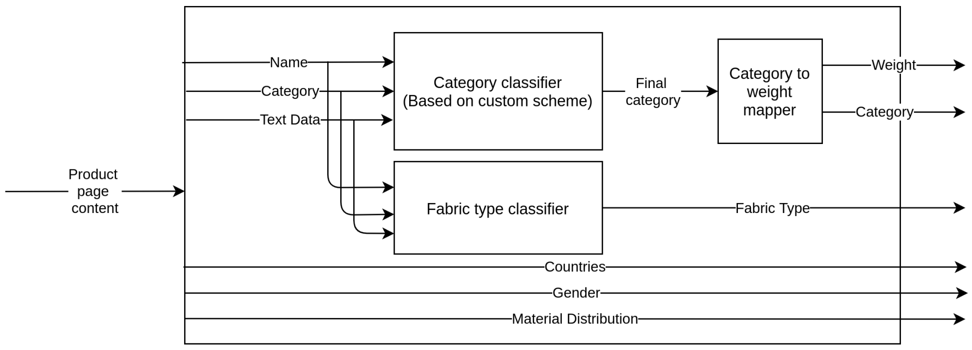

Figure 3.10: Detailed view of the text classifiers in the white-box model.

One classifier will classify the product category based on the name and text features. The possible labels of this classifier are the same as the custom categorization scheme presented in this thesis. As mentioned in section 3.2.2, a mapper is used to map each category to a weight for the product. Therefore, this classifier outputs the weight and category features needed for the analytical model.

The second classifier predicts the fabric type of the product, and is also based on the name and text features. The classes for this classifier are the same values present in the garment_type column.

The rest of the inputs needed for the analytical model (countries, gender and material distribution) are provided as structured data directly from the web page.

The two classifiers shown in Figure 3.10 are FastText supervised classifiers. FastText provides an easy to use python interface to train and use text classifiers. As seen in the figure, the inputs used for the classifiers are the name, category (as defined by the manufacturer) and text details of the product. These three features are all in string format and are concatenated before being given to the classifiers. The text data is lemmatized and stemmed before being used in the model in both the training and prediction phases of the model.

We tune the FastText supervised classifiers based on three hyperparameters. Epoch, Learning Rate and WordGrams. The epoch determines how many times the model will see each data point during training. The learning rate determines the rate at which the gradients are applied to the weights of the model. Finally, the WordGrams hyperparameter determines the maximum length of the word n-grams.

To be able to determine the best hyperparameters for the text classifiers, we take 20% of the train data as validation data and perform a grid search on multiple parameters. The hyperparameters tested for both classifiers can be seen in Table 3.4. To compare the models, the precision metric (which in this case is equal to the recall) is used.

Epoch	**Learning Rate**	**WordGrams**
100	0.5	1
200	1	2
300	2	2

Table 3.4: Hyperparameters tested for text classifiers.

The FastText text classifier uses the precision and recall metrics to evaluate models. The hyperparameters that yielded the best results for the category classifier alongside their respective precision and recall are shown in Table 3.5.

Epochs	Learning Rate	WordNgrams	Precision	Recall
200	2	2	0.96	0.96

Table 3.5: Category classifier parameters and the test set evaluation metrics.

To show the effect of the number of samples present from each category in the training dataset on the performance metrics, we plot the precision (equal to recall in this case) of each label over the count of that label in Figure 3.11.

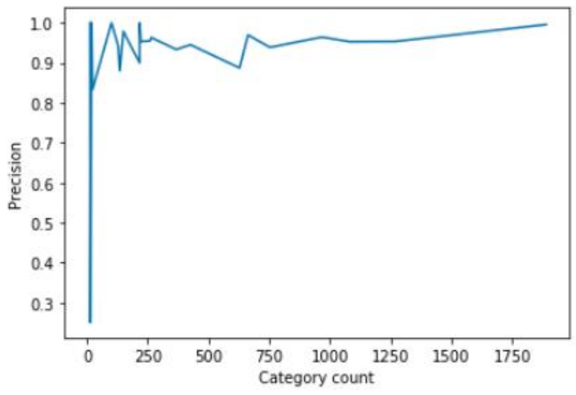

Figure 3.11: Precision of each category over the count of the category in the training data.

Furthermore, in Figure 3.12, we see the heat-map of the predicted and actual labels. As can be seen, most of the products are labelled correctly. Also, for the products that have been misclassified, the wrong label is still close to the true label. For example, Jeans are mostly misclassified as pants. Furthermore, winter jackets, autumn jackets and coats are usually misclassified as jackets. These cases show that the model captures the information provided by the text details of the product quite well.

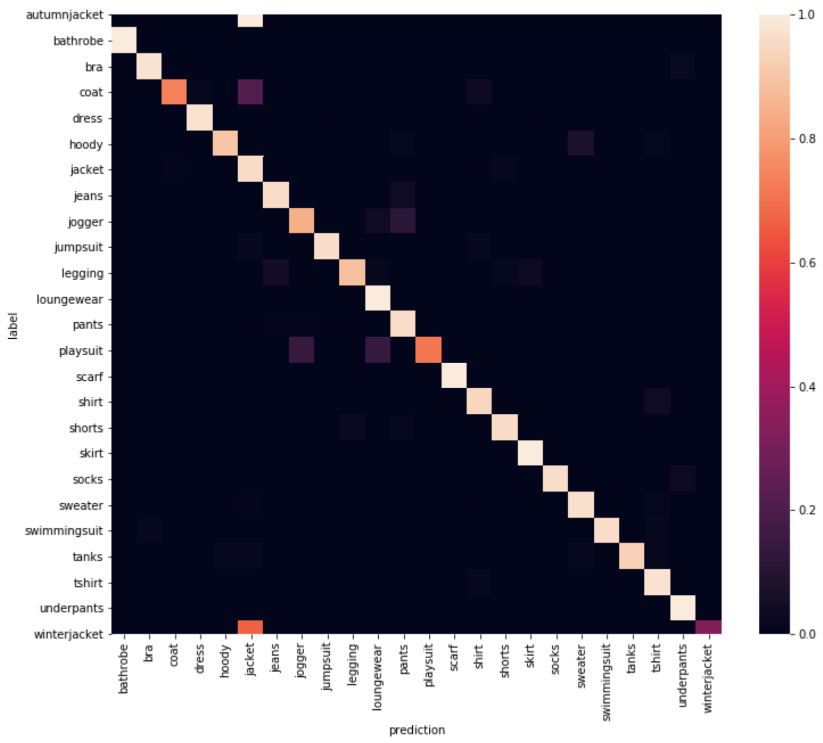

Figure 3.12: Heat-map of the predictions and actual values for the product categories.

The hyperparameters that yielded the best results for the fabric type classifier alongside their respective precision and recall are shown in Table 3.6. Also, it is worth mentioning that the text details that yielded the best results for the classifiers, were the name, category (provided by merchant) and the product details. Adding the care and washing details or specific information about the garment (e.g. recycled, waterproof) reduce the performance. Also, first predicting the category and then predicting the fabric type using the predicted category yielded the best results.

Epochs	Learning Rate	WordNgrams	Precision	Recall
300	2	4	0.974	0.974

Table 3.6: Fabric type classifier parameters and the test set evaluation metrics.

Note that, by default, the text classifiers output the three labels with the highest probability. However, we only take the label with the highest probability and use that as the predicted label. Also, micro-averaging is used for calculating the precision and recall in a k-class setting. As mentioned in section 4.1.2, the micro-averages of the precision and recall are equal in a multi-class setting where one label is considered for each data point.

It is worth mentioning that for this part of the white-box model, we first considered using imputation methods to input the missing features needed for the analytical model. However, as the inputs do not have meaningful correlations between each other, imputation methods do not perform well. We tested an imputation method based on Generative Adversarial Networks (GAN) on our dataset and based on the results this method was not used.

Analytical Model

The second part of the white-box model is the analytical model described in section 3.1.

As mentioned in the previous section, the fabric type and category inputs are provided by the text classifiers and the rest of the inputs are directly extracted from the product page.

3.3.2 Black-box Model

The black-box model is a deep regression model. This model takes the features of the dataset as input, and predicts the carbon-emissions directly.

In addition to the black-box model explained in this section, we compared various other regression algorithms and compared them using the validation dataset. The deep regression model showed the best performance and was chosen as the main black-box model. We experimented with the full dataset and the unsupervised embeddings to first select the regression model. The best regression model was then used as the black-box model for the rest of the experiments. The results of the different regression models can be seen in Table 3.3.2.

Regression Model	MSE	MAE	R2
Lasso Regression	6.47	1.61	0.71
Ridge Regression	4.88	1.39	0.78
Random Forest	3.84	0.94	0.83
XGBoost	3.28	0.80	0.85
Deep Regressor	1.94	0.54	0.89

Table 3.7: Comparison of different learned models for regression.

As it can be seen, the deep regressor has the best performance and is chosen.

Data for the black-box Model

Since some of the features like the name and text are strings, they cannot be directly used as the input of the deep regressor. Therefore, we use the 100-dimensional word embeddings of these features.

The embeddings are produced by training the unsupervised model in FastText. The text and name features of all the products are concatenated into one document and used as an input for the FastText unsupervised model. FastText uses the Skipgram model by default and the developers mention superior performance when using the Skipgram model with the default parameters on the English language. Therefore, we do not change these parameters. We reduce the dimensions of the embeddings to 100 since the vocabulary in our data is limited to clothing products. After the training is complete, FastText has learned a 100-dimensional representation of each of the words in the vocabulary. To get the representations of a sentence, FastText averages over the representations of the words in the sentence.

Finally, FastText outputs a trained model that takes a string as an input and outputs the 100-dimensional representation of the string. Another method to obtain the vector representation of the text features is to use the embeddings learned by the text classifiers that were mentioned in the white-box model. In this way, since the representations are learned with regards to the labels, they might contain information that will improve the performance.

As mentioned in section 3.2.2, the fabric_type, label and weight columns are added manually and therefore cannot be directly used in the training phase of the black-box model. The black-box model excludes these columns and does not use the custom categorization scheme introduced in this thesis. Since this would make the model dependant on the extra information provided in the form of the categorization scheme. However, this model uses the word embeddings of the text and name features that are also used to create the fabric_type and category columns. Therefore, The regression model might be able to capture the correlation between these features and the carbon emissions.

This is the main difference that the black-box and grey-box models have with the white-box model in the way they use the dataset. In the white-box model, the fabric_type and category columns are used as labels to train the classifiers. On the other hand, in the black-box and grey-box models, the embeddings of the same text data used to train the classifiers are used to train the models.

The architecture of the black-box model is shown in Figure 3.13.

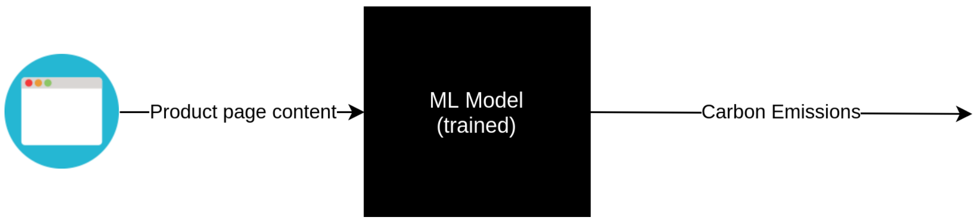

Figure 3.13: Overview of the black-box model.

Architecture of the deep regression model

The final deep regression model is a fully-connected sequential model which consists of an input layer with 128 neurons and three hidden layers with 512 neurons. The Output layer contains one neuron with a linear activation function. The model is trained using the stochastic gradient descent optimizer and an MAE loss function. For the final model, a batch size of 16 and an epoch of 400 was used. A detailed view of the black-box model is shown in Figure 3.14.

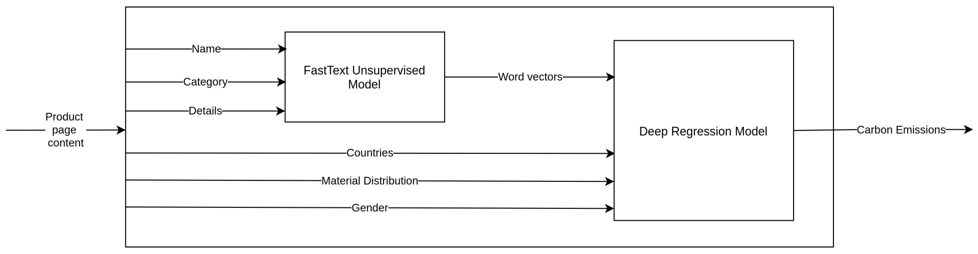

Figure 3.14: Detailed view of black-box model.

The black-box model was implemented using Keras on top of TensorFlow. The detailed architecture of the model can be seen in Table 3.3.2.

	Layer Type	Number of Neurons	Activation	Initializer
0	Input	128	ReLU	normal
1	Fully Connected	512	ReLU	normal
2	Fully Connected	512	ReLU	normal
3	Fully Connected	512	ReLU	normal
4	Output	1	Linear	normal

Table 3.8: Architecture of deep regression model.

Furthermore, to choose the best hyperparameters a grid search was done on the parameters shown in table3.3.2.

Optimiser	**Batch Size**	**Hidden Layer Nodes**	**Epochs**
SGD	16	128	300
ADAM	32	256	400
-	512	512	-

Table 3.9: Parameters tested for the black-box model.

3.3.3 Grey-box Model

As mentioned previously, the architecture of the grey-box model is inspired by [10] and leverages the bootstrapping technique to capture the pre-existing knowledge of the analytical model. In [10], the authors train an ML model on augmented datasets of different sizes and show how increasing the size of the augmented dataset increases the amount of knowledge captured from the analytical model. However, in our case, because of the techniques mentioned in section 3.2.3, we can reduce the size of the augmented data. By introducing some assumptions, we are able to produce all possible input and output pairs in a practical size. In this way, we can in a way overfit the function of the analytical model. By doing this, we benefit from the existing knowledge in the form of the analytical model while having a fully trainable model that can be used with other neural network layers and be trained on new datasets. The main novelty in this model is using the bootstrapping technique inside an ensemble method built by analysing the existing constraints in the analytical model.

Data for the grey-box Model

The grey-box model uses both the augmented dataset and the datasets used by the other models.

In the bootstrapping phase, each small regression model is trained on its respective augmented dataset for multiple epochs. In this way the regression models fully capture the function being represented by the analytical model.

After each model is bootstrapped, they are connected as described in the next section. After this, the end-to-end model is trained on the scraped dataset to fine-tune the model.

Architecture of the grey-box Model

The grey-box model consists of two main parts. The first part is an ensemble of deep regression models meant to represent and capture the analytical model in a neural network form.

There are 9 regression models. One for each smaller function of the analytical model. The architecture of this part of the grey-box model can be seen in Figure 3.15

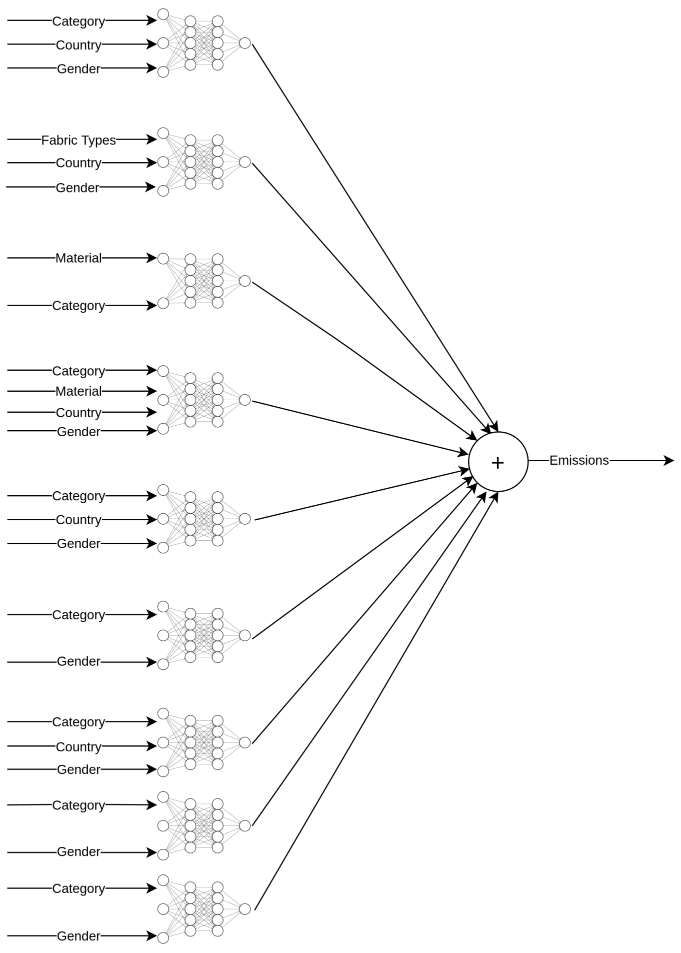

Figure 3.15: Ensemble of regression models that will replicate the analytical model.

Each of the regression models is a multi-layer perceptron. All of the regression models have the same architecture as shown in Table 3.3.3

The outputs of these 9 models are then connected together using an addition layer to represent the whole analytical model. The architecture is shown in Figure 3.15

Furthermore, two embedding layers with a softmax activation function are connected to the functions. The reason behind these layers is that the bootstrapped models are trained on one hot encoded versions of the category and fabric type features and the final end-to-end model will basically have to first

	Layer Type	Number of Neurons	Activation	Initializer
0	Input	64	ReLU	normal
1	Fully Connected	64	ReLU	normal
2	Fully Connected	64	ReLU	normal
4	Output	1	Linear	normal

Table 3.10: Architecture of each small regression model.

classify the category and fabric type (similar to the white-box model). An embedding layer with a softmax activation effectively performs the classification part.

The rest of the features are directly connected to each bootstrapped layer as they are assumed to be directly available from the garment page. The architecture of the final model can be seen in Figure 3.16. As it can be seen, unlike the analytical model, the weight is not being inputted to the fake analytical model part. This is because the category and weight features are 100% correlated in the current dataset and a learned model will not benefit from having both features.

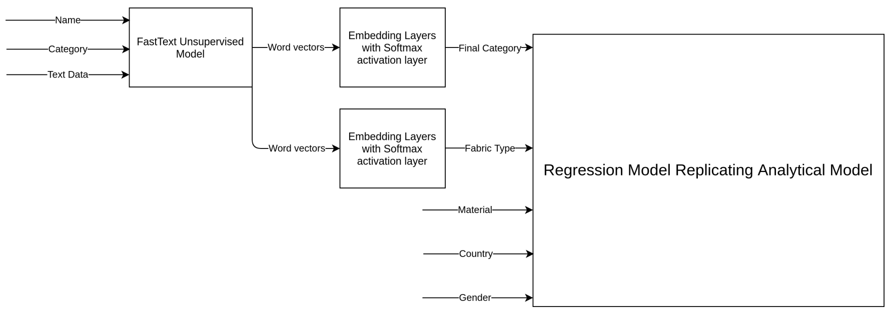
Name
Category
Text Data
FastText Unsupervised Model
Word vectors
Word vectors
Embedding Layers with Softmax activation layer
Embedding Layers with Softmax activation layer
Final Category
Fabric Type
Material
Country
Gender
Regression Model Replicating Analytical Model

Figure 3.16: Full grey-box model.

Training the grey-box model.

As mentioned previously, the grey-box model has two training phases. The first phase is only performed on the 9 regression models and the models are trained using the augmented datasets. In this phase, each model basically overfits a small part of the analytical model so that, in combination, these models can in effect replicate the analytical model.

Note that for the models which relied on the materials of a garment, we did not perform hyperparameter optimization. Because the number of data points were large and this would result in longer training times.

After the bootstrapping phase is completed, the end-to-end model is created as shown in Figure 3.16. The model is then trained on the datasets scraped from merchant B.

As mentioned previously, in [10], the authors discuss different methods to give weights to augmented datasets and real datasets. In our case, we can introduce a novel method to give weights to the augmented and scraped datasets. As our current labelled dataset is labelled with the analytical model itself, it would not make sense for the analytical part of the grey-box model to be trainable when training on the scraped dataset. Therefore, we can lock the weights of this part of the model and only allow the weights of the embedding layers to be trainable. This in effect means that the category and fabric type classifiers are being trained with respect to the final predicted emissions. We perform the training with both locked and unlocked weights and compare the results in chapter 4. In this phase of the training, the model is fine-tuned on the scraped datasets and the embedding layers will effectively learn to classify the category and fabric type.

Chapter 4

Results and Experiments

In this chapter, we describe the metrics considered and used for evaluating the models. Furthermore, we present the results and compare the performance of the models.

4.1 Performance Metrics

Performance metrics are metrics used to evaluate and compare ML models. Depending on the type of the ML model, different metrics are used to evaluate the performance of the models. In the following sections, we briefly explain and compare some of the most common performance metrics for regression and classification.

4.1.1 Regression

The metrics used for evaluating the performance of regression models deal with continuous variables. These performance metrics quantify the difference between the predicted and real value of a given variable. Performance metrics can be used in both the training and testing phase of the model.

Mean Absolute Error

Mean Absolute Error (MAE) calculates the absolute difference between the real value and predicted value of a variable. The MAE is dependant on the values of the variable and therefore can not be directly used to compare the performance of two models when the data have different scales. Moreover, the MAE is robust to outliers. The formula for MAE is seen in equation 4.1.

$$MAE = \frac{1}{n}\sum_{j=1}^{n}(y_j - \hat{y_j}) \tag{4.1}$$

Mean Squared Error

Mean Squared Error (MSE) is one of the most common performance metrics used for regression. It is defined as the average of the squared difference between the predicted values and real values. Since the errors (distances) are squared, this metric penalizes small errors and therefore is more sensitive to how bad a model is. It is also sensitive to outliers in the dataset. The formula to calculate the MSE is seen in equation 4.2.

$$MSE = \frac{1}{n}\sum_{j=1}^{n}(y_j - \hat{y_j})^2 \tag{4.2}$$

Root Mean Squared Error

Root Mean Squared Error (RMSE) is defined as the square root of the averaged square differences. Similar to the MAE, the RMSE is on the same scale as the data. However, in the RMSE, large errors are penalized heavily as the average is calculated on the squared differences. RMSE is the most widely used metric for regression tasks. The formula for calculating the RMSE can be seen in equation 4.3.

$$RMSE = \sqrt{\frac{1}{n}\sum_{j=1}^{n}(y_j - \hat{y_j})^2} \tag{4.3}$$

Coefficient of Determination (R2 score)

The Coefficient of Determination is a performance metric that is scale-free and unlike the previous metrics is not dependant on the scale of the data. The R2 metric compares a model to a baseline model and shows how much better it is. The baseline model is defined as the mean of the data. It is worth mentioning that the R2 metric is always below 1 and can also have a negative value which means the current model is even worse than naively taking the mean of the data. The formula of the R2 score can be seen in equation 4.4.

$$R^2 = 1 - \frac{MSE(model)}{MSE(baseline)} \tag{4.4}$$

4.1.2 Classification

The metrics used for classification are concerned with categorical values. To better understand the metrics for classification, we first explain the confusion matrix. In any classification problem, we have a real label and a predicted label. The predicted and real values can have four different states with regards to each other. These state can be shown intuitively using a confusion matrix. A confusion matrix of the states is shown in Figure 4.1.

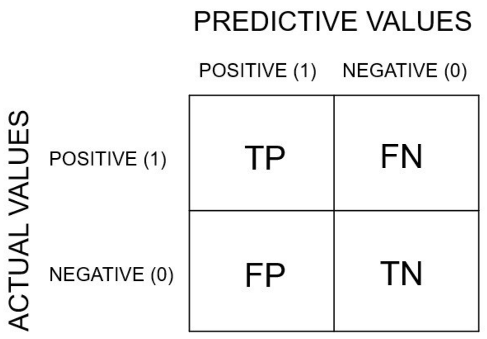

Figure 4.1: Confusion matrix.

Based on the confusion matrix, we proceed to explain some of the classification metrics.

Accuracy

The accuracy metric is defined as the number of correct predictions over all of the predictions made by the model. The accuracy metric is useful for cases where the labels in the dataset are balanced. In cases where the labels of the data are not balanced, the accuracy metric should not be used. For example, assume a dataset about cancer where 99% of the labels are not cancerous. A model that naively classifies every input with the label not cancerous will have an accuracy of 99% which is not a good representation of the model's performance. The formula of the accuracy metric is shown in equation 4.5.

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{4.5}$$

Precision

Precision is defined as how many times the model predicts the target label correctly over the total number of times it has predicted the target value. As the name suggests, precision is concerned with being precise about the target value. Precision becomes more important when we want the model to be sure about labelling a certain data point as the target label. Using the cancer example, precision will tell us how many of the patients labelled as having cancer actually had cancer. The formula of precision based can be seen in equation 4.6.

$$precision = \frac{TP}{TP + FP} \tag{4.6}$$

Recall

The recall metric is defined as the number of times the model captured a target label over the number of times that target label actually existed in the dataset. In the cancer example, this would translate to the number of patients the model labelled as having cancer over the actual number of patients with cancer. The formula of recall is shown in equation 4.7.

$$recall = \frac{TP}{TP + FN} \tag{4.7}$$

Multi-Class Recall and Precision

The explained recall and precision metrics and their respective formulas are for binary classification (having two possible labels) cases. However, in our case, both text classifiers deal with more than two labels. In multi-class scenarios, the average of the metrics is used. Two methods exist to calculate the average of the precision and recall metrics. Macro-averaging and micro-averaging. In micro-averaging, we calculate the metrics from the individual true positives, false positives and false negatives. The micro-average formulas for the precision and recall metrics in an example with k-class's are shown in equations 4.8 and 4.9.

$$precision_{micro} = \frac{TP_1 + \cdots + TP_k}{TP_1 + \cdots + TP_k + FP_1 + \cdots + FP_k} \tag{4.8}$$

$$recall_{micro} = \frac{TP_1 + \cdots + TP_k}{TP_1 + \cdots + TP_k + FN_1 + \cdots + FN_k} \tag{4.9}$$

In macro-averaging, we average the performance of each class separately. The macro-average formulas for the precision and recall metrics in an example with k-class's are shown in equations 4.10 and 4.11.

$$precision_{macro} = \frac{precision_1 + \cdots + precision_k}{k} \tag{4.10}$$

$$recall_{macro} = \frac{recall_1 + \cdots + recall_k}{k} \tag{4.11}$$

We use the micro-average metric in this thesis. Note that if we only take the label with the highest probability as the classified label, the precision and recall are equal in the case of the micro-average. This is because each false negative in the recall formula will correspond to a false positive in the precision formula. For example, assume a 3-class setting with the 3 classes *cat*, *dog* and *chicken*. Take a *dog* that has been misclassified as a *chicken*. This example will be a false positive for the *chicken* class and a false negative for the *dog* class. This is repeated for any misclassification. Therefore, the sum of the false negatives is equal to the sum of the false positives.

4.1.3 Chosen Performance Metrics

As mentioned in the previous sections, we evaluate both classification and regression models. The Regression models (The black-box and grey-box models) use the MAE both as the loss function during training and for the final evaluation.

On the other hand, the white-box model consists of two text classifiers. As mentioned previously, we use the FastText tool for building these text classifiers. FastText uses the precision and recall metrics for evaluating the text classifiers. Also, the labels of the datasets used for the text classifiers are not balanced. Therefore, the accuracy metric is not used. Furthermore, the complete model is a regression model and similar to the black-box and grey-box models we use the MAE to evaluate the performance. We also report the R2 score for each model.

It is worth noting that since all models are evaluated on the same data, the MAE can directly be used to compare the models without scaling issues.

Missing Feature	**MSE**	**MAE**	**R2**
Country	1.03	0.56	0.94
Materials	3.5	0.89	0.80
-	0.07	0.02	0.99

Table 4.1: Effect of missing features in the white-box model.

4.2 Experiments

In this section, we experiment with the models based on different aspects. We start with simple feature ablation studies where we leave out features in the training phases and examine their effect on the performance of the models. Then we move on to experimenting the size of the datasets and how they affect the performance of the models.

4.2.1 Feature Ablation Experiments

In these experiments, we simply hide each of the features in the training phase. The feature ablations do not have an effect on the classifiers in the white-box model. The analytical model replaces a missing feature with the most common value of that feature.

The absence of the category and fabric type features can not be meaningfully analysed on the models since they are inferred from the text data. Therefore, we consider the effect of hiding the country and material features. Note that the weight, gender and category are tied together.

Table 4.1 shows the performance of the white-box model while hiding each of the features.

As it can be seen in Table 4.1, hiding each of these features results in a decrease in performance. This is what we expected, especially from the white-box model since it relies on the analytical model and each input is highly informative. It can be seen that the drop in performance is much higher when the material feature is missing. This shows that the country variable does not account for much of the variance in the datasets. This can be explained by the fact that the functions of the analytical model which are related to the logistics, have extreme assumptions. Also, the distribution and retail functions assume the destination country is always Sweden.

In the experiments for the black-box and grey-box models, we examine the effect of hiding each feature in two cases. In one case, we vectorize the text features using the embeddings generated by an unsupervised model. In the

Missing Feature	**MSE**	**MAE**	**R2**
Country	2.87	0.87	0.84
Materials	3.47	1.00	0.80
-	1.94	0.54	0.89

Table 4.2: Effect of missing features in the black-box model with unsupervised embeddings.

Missing Feature	**MSE**	**MAE**	**R2**
Country	3.44	1.20	0.80
Materials	1.86	0.68	0.91
-	0.46	0.28	0.97

Table 4.3: Effect of missing features in the black-box model with supervised embeddings.

second case, we vectorize the text features using the embeddings generated by a supervised model with the categories as labels. In this way, the embeddings are created with respect to the category label allowing the black-box and grey-box models to also make use of the information of the larger category dataset. For both the unsupervised and supervised models we use the FastText model.

In Table 4.2, we see the performance of the black-box model when hiding each feature while using the unsupervised text embeddings.

In Table 4.3, we see the effect of hiding the features while using the supervised text embeddings from the category classifier.

As can be seen, using the supervised embeddings worsens the performance of the model when the country category is missing and improves the performance when the materials feature is missing. This might point to the fact that the category and material features are correlated and therefore, when we hide the country feature the models overfit on similar features. This would also explain the improvement gain when the material features are missing as some of the missing information caused by removing the material feature is regained by using the supervised embeddings.

Finally, we examine the effect of hiding the features on the grey-box model while using the unsupervised embeddings. The results are shown in Table 4.4.

In Table 4.5, we see the effect of hiding the features while using the supervised text embeddings.

As can be seen from the results, hiding features reduces the performance of the grey-box model similar to the previous models. The model performs similarly with both locked and unlocked weights. In the grey-box model, using

Missing Feature	**Locked Weights**			**Trainable Weights**		
	MSE	**MAE**	**R2**	**MSE**	**MAE**	**R2**
Country	1.91	0.74	0.91	1.83	0.74	0.91
Material	1.52	0.52	0.93	1.60	0.53	0.92
-	1.56	0.46	0.93	1.71	0.52	0.92

Table 4.4: Effect of missing features in the grey-box model with unsupervised embeddings.

Missing Feature	**Locked Weights**			**Trainable Weights**		
	MSE	**MAE**	**R2**	**MSE**	**MAE**	**R2**
Country	1.30	0.64	0.94	1.33	0.64	0.94
Material	2.88	0.64	0.87	2.34	0.55	0.89
-	0.83	0.36	0.96	0.76	0.36	0.96

Table 4.5: Effect of missing features in the grey-box model with supervised embeddings.

the supervised embeddings worsens the performance of the model when the material feature is missing. This is true with both locked and unlocked weights. In contrast, the supervised embeddings improve the performance of the model when the country feature is missing. This result is surprising as the black-box model behaved in the opposite way. Therefore, it creates doubt that the models are actually capturing a correlation between the category and material features. However, the difference of the performance metrics is relatively small and can be considered as normal variance in the results in both models.

4.2.2 Data Size Experiments

In this section, we examine the effect of the size of the training data on the models. In this way, we can examine the sensitivity of each model on the training data size. Before running the experiments, we expect the black-box model to perform much worse than the white-box and grey-box models. Because, they contain the knowledge of the analytical model. Between the grey-box and white-box model, we suspect the white-box model to perform better as it uses the analytical model directly. We examine 4 subsets of the original dataset for this experiment. In Table 4.6, we see the performance of the classifiers when using the smaller datasets.

Both classifiers perform surprisingly well even with 20% of the original dataset. In Table 4.7, we see the performance of the end-to-end white-box model with the smaller datasets.

Fraction of the whole dataset	fabric type Classifier	Category Classifier
100%	0.97	0.96
80%	0.96	0.95
60%	0.96	0.94
40%	0.96	0.93
20%	0.95	0.90

Table 4.6: Text Classifier performance with smaller datasets.

Fraction of the whole dataset	MSE	MAE	R2
100%	0.07	0.02	0.99
80%	0.29	0.10	0.98
60%	0.40	0.14	0.97
40%	0.65	0.19	0.96
20%	1.09	0.29	0.90

Table 4.7: Effect of size of the dataset on the white-box model.

As expected, the model works relatively well considering the smaller datasets. This is mainly because the white-box model's performance relies mainly on the performance of the classifiers. As t can be seen the classifiers perform well even with small datasets. Therefore, the white-box model also performs well even with smaller datasets.

Similar to the feature ablation experiment, we examine the black-box and grey-box models in two cases. One case where the text vectors are generated using the unsupervised FastText model and one where the text vectors are generated using the supervised FastText model trained on the category labels.

Similar to how the white-box model was resilient towards the data size because of the text classifiers, we expect the black-box and grey-box models to perform better on smaller datasets when using the embeddings from the supervised model.

We suspect that the black-box model will be affected the most from the smaller datasets as the grey-box model has been bootstrapped using the augmented data. We see the performance of the black-box model when using the unsupervised embeddings in Table 4.8. In Table 4.9, we see the effect of smaller datasets when using the supervised embeddings.

As suspected, the black-box model is affected more intensively than the white-box model and performs worse than the white-box model in all cases by a relatively large margin. However, using the text embeddings that are generated by the category classifier creates a large improvement on the datasets

Fraction of the whole dataset	MSE	MAE	R2
100%	1.94	0.54	0.89
80%	2.53	0.60	0.85
60%	2.95	0.86	0.83
40%	2.67	0.73	0.85
20%	4.55	1.22	0.74

Table 4.8: Effect of size of the dataset on the black-box model when using unsupervised embeddings.

Fraction of the whole dataset	MSE	MAE	R2
100%	0.46	0.28	0.97
80%	1.00	0.39	0.95
60%	0.69	0.40	0.96
40%	2.61	1.03	0.84
20%	3.44	1.20	0.80

Table 4.9: Effect of size of the dataset on the black-box model when using supervised embeddings.

with 80% and 60% of the original dataset size. This result shows that even in the black-box model, we can benefit from garment datasets which are not labelled based on their carbon emissions but at least can be labelled based on the category columns. Although, when more than 50% of the original dataset is missing most of the improvement is lost.

Finally, in Table 4.10, we can see the performance of the grey-box model on the smaller datasets while using the unsupervised embeddings. In Table 4.11, the performance when using supervised embeddings is shown.

From Table 4.10, it can be seen that letting the bootstrapped weights be

Data Size	Locked Weights			Trainable Weights		
	MSE	MAE	R2	MSE	MAE	R2
100%	1.56	0.46	0.93	1.71	0.52	0.92
80%	0.63	0.43	0.96	0.36	0.34	0.97
60%	1.08	0.48	0.93	0.95	0.46	0.94
40%	2.57	0.94	0.85	3.20	1.02	0.81
20%	13.89	2.75	0.21	12.88	2.64	0.27

Table 4.10: Effect of size of the dataset on the grey-box model using unsupervised embeddings.

Data Size	Locked Weights			Trainable Weights		
	MSE	MAE	R2	MSE	MAE	R2
100%	0.83	0.36	0.96	0.76	0.36	0.96
80%	0.47	0.28	0.97	0.43	0.28	0.97
60%	0.43	0.5	0.97	0.44	0.37	0.97
40%	2.57	0.85	0.85	2.10	0.77	0.87
20%	7.85	2.04	0.55	7.65	1.99	0.57

Table 4.11: Effect of size of the dataset on the grey-box model using supervised embeddings.

trainable, slightly improves the performance of the model. However, this might be because of the model slightly overfitting on the merchant B dataset and should be analysed when new labelled datasets become available. The model performs well when less than 50% of the dataset is missing and manages to keep the error low. This is similar to the results from the black-box model.

From Table 4.11, we see that using the supervised embeddings improves the performance of the model which is also consistent with the experiments on the black-box model. The improvement is noticeable with smaller datasets. Which again, shows the benefit of using the supervised embeddings and the importance of the information in the categories. Furthermore, similar to the unsupervised embeddings case, the performance of the model slightly improves when the bootstrapped weights are trainable. However, the same cautions that were mentioned above should be considered.

4.3 Comparison

In the following sections, we present the final result for each of the models when using the final test data. We also discuss the results of the experiments and compare them.

4.3.1 Feature Ablation

First, we examine the effect of the missing features on each of the models. As expected, all models suffer from a drop in performance when a feature is missing. As mentioned previously, when no feature is missing, the white-box model has the best performance followed by the black-box model and the grey-box model. However, the performance difference between the black-box and grey-box models is small.

Material Feature

When the material feature is hidden, the performance of the white-box model worsens drastically as can be seen in table 4.1. However, the black-box model does not see a drastic drop in performance like the white-box model and manages to outperform the white-box model when using supervised embeddings. This shows that the black-box model is more resilient to missing features and also that the model has managed to capture some of the information provided by the material feature by leveraging the other features. The grey-box model proves to be even more resilient to the missing features and outperforms both the black-box and white-box models.

Country Feature

When the material feature is missing, the performance of the white-box model suffers from a drastic drop. However, in this case, the black-box model also suffers from a drastic drop in performance and performs worse than the white-box model. The grey-box model also sees a drop in performance. However, relative to the case where the material feature was missing, the drop in performance is not as drastic as the other two models and the grey-box model performs similar to the white-box model.

4.3.2 Data Size

In this section, we compare the performance of the models with regards to data size.

The white-box model manages to perform well even when we reduce the size of the training sets. This is because the text classifiers that provide the input of the analytical model, manage to achieve high precision and recall even with 20% of the original dataset.

The performance of the black-box model worsens rapidly when using the unsupervised embeddings and performs worse than the white-box model in every case. However, when using the supervised embeddings, the performance does not drop as rapidly as long as less than 50% of the dataset is missing. This again shows the benefit of using the supervised embeddings.

The grey-box model performs better than the black-box model with both supervised and unsupervised embeddings. However, this model suffers from a sudden drop of performance when the dataset is 20% of the original dataset and performs even worse than the black-box model in this case. We suspect this to be due to the fact that the model overfits the small dataset. Especially when considering that it has a larger degree of freedom compared to the black-box model when the bootstrapped weights are trainable. In the case where the bootstrapped weights are locked, we suspect that the amount of data is not enough for the model to find the relation between the output of the softmax layers (which are effectively classifiers) and the carbon emissions in the output. In general, the white-box model is the model most resilient to reducing the size of the dataset.

4.3.3 Generalization

As mentioned previously, the main challenges in this problem space are the lack of labelled data and a lack of merchants that provide all the required inputs for determining the carbon emissions of a product. Therefore, testing the generalization of the models through experimentation was not possible. Consequently, we mainly discuss how we expect the models to generalize to new datasets and provide some predictions about the generalization of the models.

In the previous experiments, it is obvious that the grey-box model performs better than the black-box model, and the white-box model outperforms both the grey-box and black-box models. However, if such a project is continued and the model is used more data will become available. The newly available data will have two cases. In one case, the data is similar to the data we scraped from merchant B, and has all the required features needed to label it by using the analytical model. In another case, the dataset is truly labelled by the manufacturer (or any other source) and it contains true labelled data.

In the first case, all three models can be improved using the new datasets. However, in the second case, the white-box model will need to be manually changed and its weights will need to be updated by human interference. Therefore, benefiting from labelled datasets would require manual work. But, the black-box and grey-box models can be automatically trained on such datasets. And between these two models, the grey-box model shows superior performance. This superior performance is because some of the knowledge in the analytical model has been successfully captured by the grey-box model in the bootstrapping phase. Moreover, as mentioned in section 3.3.3, the grey-box model can be easily trained on both types of new datasets, without becoming biased on the datasets labelled by the analytical model. Whenever the datasets are labelled by the analytical model, the bootstrapped weights can be locked and only the softmax layers are trained. And when labelled data is available, the weights are set to be trainable so the model can adapt the previous knowledge gained from the analytical model to the new knowledge present in the new datasets.

Chapter 5

Conclusions and Discussion

As mentioned in this work, various ways exist to combine analytical and learned models to gain the benefits of both. However, in previous research, labelled datasets were available. The labelled datasets created the opportunity to use analytical models as base models and either improve them with learned models or, use learned models to compensate for situations where the analytical model fails to produce good results. In our work, we did not have access to labelled data and the labels produced by the analytical model was the closest we could get to true labels. However, for such a project to be used practically, we should make a scalable model that will be able to adapt to potential new data while relying on the best estimates possible along the way.

As was seen, we design and compare three different models for this purpose. The most naive approach is to label available data using the analytical model and use that to train an ML model. However, we would need to gather large amounts of data that contain all the required features and label them to train the ML models sufficiently.

In the second experimented model, we overcome this problem by using an ensemble of the analytical model with learned models. In this way, we can benefit from the knowledge of the analytical model and take advantage of the flexibility of learned models that are used for classifying the inputs needed by the analytical model. Furthermore, the different classifiers in the white-box model can be trained separately with datasets readily available. They can also be replaced with better classifiers if they become available. However, the white-box model falls short in the case of potential labelled data. The white-box model relies on the analytical part to calculate the emissions which can not be trained (in an automated way) based on new labelled data. We would want to adapt the model to new labelled data points in cases that the analytical model calculates a different carbon emission.

The grey-box model attempts to solve this problem by first training an ML model on augmented data created using the analytical model to replicate the analytical model. This results in a model that can adapt to cases where labelled data become available still benefiting from the knowledge of the analytical model. Also, this model can easily be connected to other neural networks and trained at the same time.

Finally, We conclude that although the white-box model performs better than the grey-box model in the short term and in cases that all features are present, this model is not scalable. Therefore, the grey-box model provides a suitable trade-off between the scalability of a learned model and the robustness of an analytical model. However, to fully leverage the robustness of the analytical model, the grey-box model should be bootstrapped on a more complete version of the augmented dataset. Also, the black-box model might outperform the grey-box model in case labelled datasets with a complete set of features become available. This will need further experimentation.

5.1 Future Work

As mentioned in the thesis, one of the most challenging problems with developing such a model has been the lack of ground truth data and sufficient data for other sections, for example, weight classifications. One of the potential solutions for solving this issue in the white-box model is providing all of the inputs required by the analytical model using other classification or regression models. For example, currently, the fabric types are directly mapped to a coefficient that is used for converting the weight of a garment to electricity usage. These static mappers can be replaced by predictive models which can also handle new inputs. Additionally, training each smaller prediction model separately would result in not needing a complete dataset which represents the relations between all features. For example, training a classifier which will predict the weight of a garment will not require a dataset that includes the carbon emissions which makes it easier to find a training dataset for a weight classifier.

Also, in order for this model to be used by consumers, a proper user interface should be developed which allows the consumer to interact with this model. Such an interface might query the user for extra information to help with predicting the carbon emissions of a product.

Furthermore, as also mentioned in section 2.6, a model that can classify which section of a web page contains the different inputs required by the model has to be created. Although, in the case of the regression model, the architecture can be changed so the features include the different HTML elements of a product page and directly predict the carbon emissions from the webpage. In this case, a classifier that can classify if a web page is a product page or not will suffice.

Moreover, the grey-box model does not take the weight feature as input, since all the categories are statically mapped to a specific weight and the two features would be 100% correlated and the model would not benefit from using both features. However, the grey-box model will benefit from being trained on a range of possible weights for a category. This will result in large augmented datasets that will make the bootstrapping phase lengthy and possibly impractical. Although, this issue might be solved in another way if datasets with the weight of the products are obtained.

Finally, with the right amount of data and the appropriate models, such a carbon emissions estimator can be extended to calculate the carbon emissions consumer products other than textile products.

5.2 Conclusions

Based on the work of this thesis, we conclude that using the analytical model during the prediction phase of the carbon estimation model improves the performance of the carbon estimator for textile products drastically. Also, to provide a scalable alternative, we present the grey-box model inspired by the bootstrapping technique.

Finally, We have provided a proof-of-concept that can hopefully provide the basis of a carbon emissions estimator for textile products and can be leveraged to potentially affect consumer behaviour towards choosing sustainable products. However, the limitations, especially the lack of true labelled data, need to be overcome for such a product to be used by consumers.

Bibliography

[1] David Rolnick et al. “Tackling Climate Change with Machine Learning”. In: (June 2019). URL: http://arxiv.org/abs/1906.05433.

[2] *The price of fast fashion*. Jan. 2018. DOI: 10.1038/s41558-017-0058-9.

[3] *Zalando. The Starting Point for Fashion*. Tech. rep. URL: https://www.imwf.de/pressemitteilung/.

[4] HM. *HM 2018 Sustainability Report*. https://about.hm.com/content/dam/hmgroup/groupsite/documents/masterlanguage/CSR/reports/2018_Sustainability_report/HM_Group_SustainabilityReport_2018_\%20FullReport.pdf. 2018.

[5] Jaehun Sim and Vittaldas Prabhu. “The life cycle assessment of energy and carbon emissions on wool and nylon carpets in the United States”. In: *Journal of Cleaner Production* 170 (Jan. 2018), pp. 1231–1243. ISSN: 09596526. DOI: 10.1016/j.jclepro.2017.09.203.

[6] *THE LIFE CYCLE Understanding the environmental impact of a pair of Levi’s ® 501 ® jeans*. Tech. rep. 2015.

[7] Daniel Thylmann et al. *Title of the Study: Life Cycle Assessment (LCA) of Organic Cotton-A global average*. Tech. rep. 2014. URL: www.pe-international.com.

[8] Llorenç Milà et al. “Application of life cycle assessment to footwear”. In: *International Journal of Life Cycle Assessment* 3.4 (1998), pp. 203–208. ISSN: 09483349. DOI: 10.1007/BF02977570.

[9] David Rolnick et al. “Tackling Climate Change with Machine Learning”. In: *arXiv e-prints*, arXiv:1906.05433 (June 2019), arXiv:1906.05433. arXiv: 1906.05433 [cs.CY].

[10] Diego Didona and Paolo Romano. "Using analytical models to bootstrap machine learning performance predictors". In: *Proceedings of the International Conference on Parallel and Distributed Systems - ICPADS*. Vol. 2016-Janua. IEEE Computer Society, Jan. 2016, pp. 405–413. ISBN: 9780769557854. DOI: 10.1109/ICPADS.2015.58.

[11] *IVL Swedish Environmental Research Institute*. 1966.

[12] Tom M. Mitchell. *Machine Learning*. New York: McGraw-Hill, 1997. ISBN: 978-0-07-042807-2.

[13] Claudio Gambella, Bissan Ghaddar, and Joe Naoum-Sawaya. "Optimization Models for Machine Learning: A Survey". In: *arXiv e-prints*, arXiv:1901.05331 (Jan. 2019), arXiv:1901.05331. arXiv: 1901.05331 [math.OC].

[14] J. Kiefer and J. Wolfowitz. "Stochastic Estimation of the Maximum of a Regression Function". In: *The Annals of Mathematical Statistics* 23.3 (1952), pp. 462–466. ISSN: 00034851. URL: http://www.jstor.org/stable/2236690.

[15] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *arXiv e-prints*, arXiv:1412.6980 (Dec. 2014), arXiv:1412.6980. arXiv: 1412.6980 [cs.LG].

[16] Lukas Ryll and Sebastian Seidens. "Evaluating the Performance of Machine Learning Algorithms in Financial Market Forecasting: A Comprehensive Survey". In: *arXiv e-prints*, arXiv:1906.07786 (June 2019), arXiv:1906.07786. arXiv: 1906.07786 [q-fin.CP].

[17] Daniel W. Otter, Julian R. Medina, and Jugal K. Kalita. "A Survey of the Usages of Deep Learning in Natural Language Processing". In: *arXiv e-prints*, arXiv:1807.10854 (July 2018), arXiv:1807.10854. arXiv: 1807.10854 [cs.CL].

[18] Jose Camacho-Collados and Mohammad Taher Pilehvar. "On the Role of Text Preprocessing in Neural Network Architectures: An Evaluation Study on Text Categorization and Sentiment Analysis". In: *arXiv e-prints*, arXiv:1707.01780 (July 2017), arXiv:1707.01780. arXiv: 1707.01780 [cs.CL].

[19] Kamran Kowsari et al. "Text Classification Algorithms: A Survey". In: *Information (Switzerland)* 10.4 (Apr. 2019). DOI: 10.3390/info10040150. URL: http://arxiv.org/abs/1904.08067%20http://dx.doi.org/10.3390/info10040150.

[20] Yoon Kim. “Convolutional Neural Networks for Sentence Classification”. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing* (Aug. 2014). DOI: 10.3115/v1/D14-1181.

[21] Yang Li and Tao Yang. “Word Embedding for Understanding Natural Language: A Survey”. In: vol. 26. May 2017. ISBN: 978-3-319-53817-4. DOI: 10.1007/978-3-319-53817-4.

[22] Armand Joulin et al. “Bag of Tricks for Efficient Text Classification”. In: *15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017 - Proceedings of Conference* 2 (July 2016), pp. 427–431. URL: http://arxiv.org/abs/1607.01759.

[23] Piotr Bojanowski et al. “Enriching Word Vectors with Subword Information”. In: *Transactions of the Association for Computational Linguistics* 5 (Dec. 2017), pp. 135–146. ISSN: 2307-387X. DOI: 10.1162/tacl{_}a{_}00051. URL: http://arxiv.org/abs/1607.04606.

[24] Tomas Mikolov et al. “Efficient estimation of word representations in vector space”. In: *1st International Conference on Learning Representations, ICLR 2013 - Workshop Track Proceedings*. International Conference on Learning Representations, ICLR, Jan. 2013.

[25] Tomas Mikolov et al. “Efficient Estimation of Word Representations in Vector Space”. In: *arXiv e-prints*, arXiv:1301.3781 (Jan. 2013), arXiv:1301.3781. arXiv: 1301.3781 [cs.CL].

[26] Gustav Sandin et al. *environmental assessment of Swedish clothing consumption-six garments, sustainable futures*. ISBN: 9789189049055. URL: www.ri.se.

[27] Altun Sule. *Cotton t-shirt lifecycle assessement*. Tech. rep. 2. 2012, pp. 87–89. URL: www.isca.in.

[28] You Zhang et al. “Life cycle assessment of cotton T-shirts in China”. In: *International Journal of Life Cycle Assessment* 20.7 (July 2015), pp. 994–1004. ISSN: 16147502. DOI: 10.1007/s11367-015-0889-4.

[29] Engr Ayub Nabi Khan et al. “Lifecycle Analysis (LCA) of a White Cotton T-shirt and Investigation of Sustainability Hot Spots: A Case Study”. In: (2018). ISSN: 2631-8504.

[30] *LCA of a pair of GORE-TEX ® branded waterproof and breathable hiking boots 2 1 GOAL 1.1 Main goal*. Tech. rep.

[31] *LCA of a GORE NorthFace Jacket*. Tech. rep.

[32] Ping Hou et al. "Estimating Missing Unit Process Data in Life Cycle Assessment Using a Similarity-Based Approach". In: *Environmental Science and Technology* 52.9 (May 2018), pp. 5259–5267. ISSN: 15205851. DOI: `10.1021/acs.est.7b05366`.

[33] Emmanuel Pintelas, Ioannis E. Livieris, and Panagiotis Pintelas. "A Grey-Box Ensemble Model Exploiting Black-Box Accuracy and White-Box Intrinsic Interpretability". In: *Algorithms* 13.1 (Jan. 2020), p. 17. ISSN: 1999-4893. DOI: `10.3390/a13010017`. URL: `https://www.mdpi.com/1999-4893/13/1/17`.

[34] Alina Kloss, Stefan Schaal, and Jeannette Bohg. "Combining learned and analytical models for predicting action effects". In: (Oct. 2017). URL: `http://arxiv.org/abs/1710.04102`.

[35] Diego Didona et al. "Enhancing performance prediction robustness by combining analytical modeling and machine learning". In: *ICPE 2015 - Proceedings of the 6th ACM/SPEC International Conference on Performance Engineering*. Association for Computing Machinery, Inc, Jan. 2015, pp. 145–156. ISBN: 9781450332484. DOI: `10.1145/2668930.2688047`.

[36] Jeffrey Dean and Sanjay Ghemawat. "MapReduce: Simplified Data Processing on Large Clusters". In: *OSDI'04: Sixth Symposium on Operating System Design and Implementation*. San Francisco, CA, 2004, pp. 137–150.

[37] *BeautifulSoup*. `https://www.crummy.com/software/BeautifulSoup/`. 2020.

[38] Textile Exchange. *CFMB 2019 Fiber Uptake Calculations Guide*. `https://textileexchange.org/wp-content/uploads/2019/06/CFMB_2019_Fiber-Uptake-Calculations-Guide.pdf`. 2019.

Appendix A

Appendix

tshirt	shorts	scarf	boots	skirt	jogger	pants
Jeans	hoody	gloves	shoes	sweater	jumpsuit	hat
dress	coat	slipper	bra	pajamas	loungewear	sneakers
jacket	autumnjacket	boot	swimmingsuit	tanks	sleepwear	legging
shirt	winterjacket	sandal	suit	windbreaker	playsuit	bathrobe

Table A.1: Custom product categorization scheme.

Merchant B Columns	**Merchant A Columns**
description	name
Composition	category
Care instructions	details
name	cat0
cat0	cat1
cat1	details
cat2	-
cat3	-
cat4	-
country of production	-

Table A.2: The columns of the CSV files before merging them.