metals

MDPI

Article

Texture-Based Metallurgical Phase Identification in Structural Steels: A Supervised Machine Learning Approach

Dayakar L. Naik, Hizb Ullah Sajid and Ravi Kiran *

Department of Civil & Environmental Engineering, North Dakota State University, ND 58105, USA; dayakarnaik.lavadiya@ndsu.edu (D.L.N.); hizbullah.sajid@ndsu.edu (H.U.S.)

* Correspondence: ravi.kiran@ndsu.edu; Tel.: +01-701-231-7878

Received: 19 March 2019; Accepted: 8 May 2019; Published: 10 May 2019

Abstract: Automatic identification of metallurgical phases based on thresholding methods in microstructural images may not be possible when the pixel intensities associated with the metallurgical phases overlap and, hence, are indistinguishable. To circumvent this problem, additional visual information about the metallurgical phases, referred to as textural features, are considered in this study. Mathematically, textural features are the second order statistics of an image domain and can be distinct for each metallurgical phase. Textural features are evaluated from the gray level co-occurrence matrix (GLCM) of each metallurgical phase (ferrite, pearlite, and martensite) present in heat-treated ASTM A36 steels in this study. The dataset of textural features and pixel intensities generated for the metallurgical phases is used to train supervised machine learning classifiers, which are subsequently employed to predict the metallurgical phases in the microstructure. Naïve Bayes (NB), k-nearest neighbor (K-NN), linear discriminant analysis (LDA), and decision tree (DT) classifiers are the four classifiers employed in this study. The performances of all four classifiers were assessed prior to their deployment, and the classification accuracy was found to be >97%. The proposed technique has two unique advantages: (1) unlike pixel intensity-based methods, the proposed method does not misclassify the grain boundaries as a metallurgical phase, and (2) the proposed method does not require the end-user to input the number of phases present in the microstructure.

Keywords: gray level co-occurrence matrix (GLCM); ASTM A36; steel microstructure; textural features; machine learning classifiers

1. Introduction

It is well known that the macroscopic mechanical properties of any material are governed by its underlying microstructure [1]. Most of the engineering metallic alloys like dual-phase steels, α-β brass, α-β titanium, etc., possess multiphase polycrystalline microstructures [2]. Such microstructures are characterized by grain sizes, distinct phases and their volume fractions, and morphology [3]. Under different mechanical and thermal manufacturing and operating conditions, these microstructural features undergo changes that result in modified bulk properties of the metal [4]. Analysis of such information results in the establishment of a relationship between microstructural features and the bulk material properties that will guide engineers in designing components for specialized applications (e.g., Hall–Petch relation [5,6]). In general, material characterization techniques such as X-ray, neutron and electron diffraction, light optical microscopy, and electron and ion beam microscopy are employed to investigate and quantify the microstructural features of metals at various length scales [1]. Some of these techniques are time-consuming and expensive; hence, researchers often resort to light optical microscopy for performing tasks such as metallurgical phase identification and evaluation of grain

sizes. The images obtained from the light-optical microscope are then analyzed manually following the standard protocols provided by the American Society for Testing and Materials (ASTM) standards E114 [7] and E562 [8]. However, this process is labor-intensive and a subjective process prone to poor repeatability and interpretation of results [9]. Therefore, automated digital image processing-based techniques have been developed in recent years to overcome these issues and accurately quantify the microstructural features for better design of engineering components.

Image segmentation is a digital image processing technique that is widely used in the fields of engineering, medicine, food science, remote sensing, etc., to identify the distinct regions/objects in an image that possess distinguishable visual characteristics or features [10]. Grayscale level, color, contrast, spectral values, or textural features are some examples of such distinguishing features [11]. In general, two types of approaches are employed for segmentation of images: a discontinuity approach and a similarity approach [12]. While a discontinuity approach involves computation of abrupt changes or discontinuity of some object (e.g., edges) in the image to identify distinct regions, a similarity approach involves extraction and a one-to-one comparison of similar features (e.g., pixel intensity) for identification of distinct regions. Techniques of discontinuity approach include Sobel operator [13], Laplacian of Gaussian (LoG) operator [14], Laplacian operator [15], and canny operator [15]. Techniques of similarity approach include histogram-based thresholding [16], region splitting and merging [17], level-set [18], clustering, and water shedding [11]. Among these techniques, histogram-based thresholding (or Otsu's method [16]) is extensively used for image analysis or segmentation of microstructures [19]. In this technique, threshold-based criteria are established from the multimodal histogram of pixel intensities, which is used for the segmentation of distinct phases. Implementation of such a technique results in an accurate segmentation of metallurgical phases whose pixel intensity distributions are distinct (multimodal) and do not overlap significantly (see Figure 1). However, when there are multiple metallurgical phases whose pixel intensity distributions overlap with each other (see Figure 1), employing histogram-based thresholding may lead to misclassification of phases.

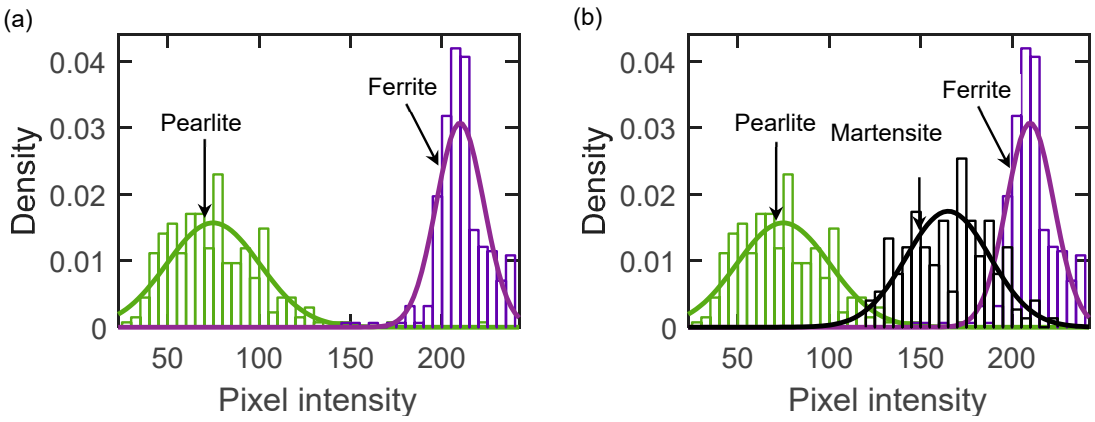

Figure 1. Probability density function of pixel intensities of metallurgical phases: (**a**) well-discriminated ferrite and pearlite phases, and (**b**) overlapped ferrite, pearlite, and martensite phases.

Automated image segmentation procedures have been proposed and developed in recent years by numerous researchers to identify and quantify microstructural features [20]. Zhang and Liu [21] implemented the canny edge algorithm for identification of phases in Ti-6Al-4V titanium alloy, which involved a series of preprocessing steps such as noise removal and uneven illumination. Campbell et al. [9] developed a watershed algorithm-based technique with pre- and post-processing steps to segment the touching grains in a titanium (Ti-6Al-4V) microstructure. An automated microstructural characterization software MiPAR® was developed by Sosa et al. [19] that included a segmentation module built based on thresholding methods. Other notable works that employed threshold-based techniques include segmentation of ferritic-martensitic dual phase steel by

Burikova et al. [22] and identification of bainite in Fe-C-Mo steel by Ontman et al. [23]. In addition to these studies, artificial intelligence-based image segmentation is also found in the literature, which includes clustering [24], neural networks [25], fuzzy logic [26], and support vector machine (SVM) [27] methods for identification of microstructures in various metals. A sophisticated image processing technique that accounts for additional distinguishing features is required for accurate phase identification in a microstructure with phases that have overlapping pixel intensities.

The goal of this paper is to identify the distinct metallurgical phases of heat-treated ASTM A36 steels based on the textural features and pixel intensities of individual phases. To this end, the microstructural images of metallographic specimens are acquired using an optical microscope, and the textural features and pixel intensities of distinct metallurgical phases are extracted from a gray level co-occurrence matrix (GLCM) of each phase. Supervised machine learning classifiers are employed for identification of metallurgical phases, and the following four classifiers are employed for this purpose: (1) naïve Bayes, (2) k-nearest neighbor, (3) linear discriminant analysis, and (4) decision tree. All four classifiers are trained with the extracted textural features and pixel intensities (of distinct metallurgical phases) and then deployed to identify the unknown phases in the microstructure. The rest of the manuscript is organized as follows: the textural feature extraction method is explained in Section 2, a brief overview of supervised machine learning classifiers is provided in Section 3, the materials and methodology adopted in this study are described in Section 4, the feature selection method is explained in Section 5, the validation of results are discussed in Section 6, and a summary of the study is provided in Section 7.

2. Texture

An object is considered to possess texture if its appearance is composed of patterns defined by variations in brightness and color. Objects can have natural textured, humanmade textures, or a combination of both. While wood, soil, grass, etc., are some examples of natural texture-possessing objects, carpet, brick walls, concrete, etc., are some examples of humanmade objects with distinctive textures. Human vision perceives the texture of different objects by sensing the variations in brightness and color. This information is used to discriminate one object from the other. However, in computer vision, a set of metrics are required to quantify the texture of an object. These sets of metrics are often referred to as image-textural features and are extracted from a given digital image through image processing techniques. Image textural features are widely used to segment different objects in an image for object/image identification and/or classification purposes.

Two methods are commonly employed for quantifying the image texture: (1) a structural approach and (2) a statistical approach. While the primitive or repetitive elements and their placement rules are obtained in a structural approach to describe the texture, nondeterministic properties obtained from the distribution of grayscale levels of a region of an image are used in a statistical approach [28]. These nondeterministic properties are referred to as textural features. The structural approach is more suitable for regular textural patterns (e.g., checkerboard patterns, carpet textures, etc.) as it considers the hierarchy of spatial arrangement of primitives, and the statistical approach is more suitable for arbitrary textures (e.g., sand, concrete, etc.). In the context of this study, a statistical approach was adopted, owed to the unstructured visual pattern (nonrepetitive pattern) exhibited by the steel microstructure, to identify the constituent metallurgical phases. In this study, the metallurgical phases of ASTM A36 steel, namely ferrite, pearlite, and martensite, were assumed to possess unique textures. The statistical features quantifying the textures of each metallurgical phase were determined in this study by employing the gray-level co-occurrence matrix [29]. A detailed description of the GLCM method and extraction of textural features is provided next.

Gray Level Co-occurrence Matrix (GLCM) and Textural Features

Let us consider an image domain ($\mathbf{\Omega}$) that consists of N_x and N_y number of pixels in x and y directions, respectively. A pixel (short form for "picture element") is the basic logic unit in a digital

image. It has a rectangular or a square shape and has a unique location attached to it. Location of a pixel in the domain $\boldsymbol{\Omega}$ is denoted by ω_{ij}, where $i = 1 \ldots N_y$, and $j = 1 \ldots .N_x$ represents the corresponding row and column numbers of the pixel image grid, respectively. Each pixel in an image is associated with an intensity $I\left(\omega_{ij}\right) = I_{ij}$ where $I_{ij} \in \mathbb{Z}_{+}^{N_y \times N_x}$. In an 8-bit grayscale image, which was the case in the current study, any pixel can have $2^8 = 256$ intensity levels $\left(N_g = 256\right)$. In fact, for the problem at hand, it was not necessary to consider 256 intensity levels. Instead, eight intensity levels or grayscale levels were sufficient. The process of converting 256 grayscale levels to eight grayscale levels is referred to as quantization. In this study, this operation was accomplished through an in-built command 'imquantize' available in MATLAB®. The original $\left(N_g = 256\right)$ and modified/quantized images $\left(N_g = 8\right)$ of the microstructure are presented in Figure 2. Using $N_g = 8$, instead of $N_g = 256$, will lead to substantial savings in computational time without loss of visual information, as demonstrated in Figure 2.

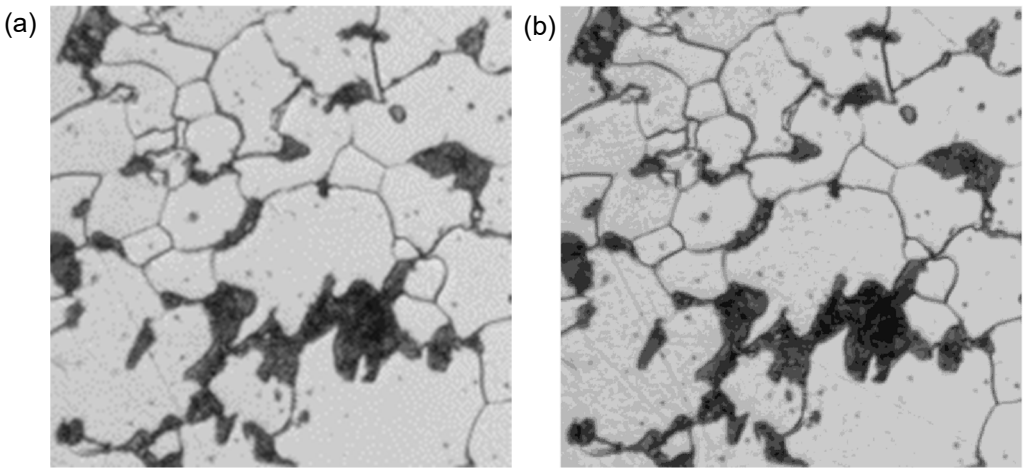

Figure 2. Illustration of (**a**) an original image with 256 greyscale levels, and (**b**) a quantized image with 8 grayscale levels.

GLCM ($\boldsymbol{G}$) is a square matrix whose size is determined by the number of grayscales present in an image ($\boldsymbol{G} \in \mathbb{Z}_{+}^{N_g \times N_g}; N_g = 8$) and is independent of the number of pixels in the x and y directions in the image domain ($\boldsymbol{\Omega}$). Each element of the GLCM (G_{mn}) represents the count of pixel pairs that are separated by d number of pixels in the θ direction, wherein one pixel has an intensity m and the other pixel has an intensity n. The mathematical definition of a typical element G_{mn} in a GLCM ($\boldsymbol{G}$) for $\theta = 0^\circ$ or 180° is given as follows:

$$G_{mn} = \#(\omega_{ab}, \omega_{ef} \in \Omega \Big| \, a - e = 0, \left|b - f\right| = d, \left(I(\omega_{ab}), I(\omega_{ef})\right) = (m, n) \; or \; (n, m)), \tag{1}$$

where $a, e \in \left\{1, 2, 3 \ldots . N_y\right\}$; $b, f \in \{1, 2, 3 \ldots . N_x\}$; and $m, n \in \left\{1, 2, 3, \ldots N_g\right\}$. The condition $a - e = 0$ signifies the fact that pixels ω_{ab} and ω_{ef} are in the same row (i.e., $\theta = 0^\circ$ or 180°), and the condition $\left|b - f\right| = d$ has two consequences: (1) the pixels ω_{ab} and ω_{ef} are separated by d number of pixels, and (2) the order of the pixels (bidirectional) does not have any impact on the value of G_{mn} ($\theta = 0^\circ$ or 180°). The definitions of GLCM for $\theta = 45^\circ, 90^\circ, 135^\circ, 225^\circ, 270^\circ$, and 315° can be found elsewhere [29]. In the current study, the value of d and θ were fixed to be 1 and $0^\circ/180^\circ$, respectively, in order to estimate the GLCM for various metallurgical phases present in the microstructural image (ASTM A36 steel).

The GLCM ($\boldsymbol{G}$) evaluated using Equation (1) is referred to as unnormalized GLCM, and a typical element G_{ij}^N in the normalized GLCM ($\boldsymbol{G}^N$) is defined as follows:

$$G_{ij}^N(\theta, d) = \frac{G_{ij}(\theta, d)}{\sum_{i=1}^{8} \sum_{j=1}^{8} G_{ij}(\theta, \, d)}. \tag{2}$$

Each element (G_{ij}^N) of the normalized GLCM is a measure of the joint probability occurrence of pixel pairs that are separated by distance d in the θ direction such that the grayscale levels of the first and second pixels match with row and column numbers of GLCM, respectively. Detailed examples for the evaluation of normalized and unnormalized GLCM is provided elsewhere [29].

As previously mentioned, the image texture is quantified by the textural features. In the statistical approach, textural features are the second order statistics extracted from the normalized GLCM $\left(\boldsymbol{G}^N\right)$. In total, there are 19 textural features that can be extracted from the GLCM proposed by various researchers in the past. Among the 19 textural features, 14 were proposed by Haralick et al. [29], and five other textural features were proposed by Soh et al. [30]. The mathematical definitions of these textural features are summarized in Table 1. While some of these textural features are easy to interpret (e.g., homogeneity, entropy, contrast, and correlation), the rest of them are purely mathematical in nature and are difficult to interpret. In this study, for a given digital image domain $\boldsymbol{\Omega}$, textural features were evaluated at every pixel. In fact, a pixel does not possess texture. Hence, a window of $S \times S$ pixels, where S is the number of pixels $\left(S \leq N_x \text{ and } N_y\right)$, was used to evaluate the textural features, and these textural features were assigned to the pixel located at the center of this window. The calculated textural features were then used to classify a pixel into one of the metallurgical phases. This classification was performed using machine learning algorithms, which are discussed next.

Table 1. Textural features from the gray level co-occurrence matrix (GLCM) [31].

Notation	Texture Feature	Equation
T_1	Autocorrelation	$\sum_i \sum_j (i.j) p(i,j)$
T_2	Contrast	$\sum_i \sum_j \left\|i-j\right\|^2 p(i,j)$
T_3	Cluster prominence	$\sum_i \sum_j \left(i+j-\mu_x-\mu_y\right)^4 p(i,j)$
T_4	Cluster shade	$\sum_i \sum_j \left(i+j-\mu_x-\mu_y\right)^3 p(i,j)$
T_5	Dissimilarity	$\sum_i \sum_j \left\|(i-j)\right\| p(i,j)$
T_6	Energy	$\sum_i \sum_j p(i,j)^2$
T_7	Entropy	$-\sum_i \sum_j p(i,j).\log(p(i,j))$
T_8	Homogeneity	$\sum_i \sum_j \frac{p(i,j)}{1+\left\|i-j\right\|}$
T_9	Maximum probability	$\max p(i,j)$
T_{10}	Sum of squares	$\sum_i \sum_j (1-v)^2 p(i,j)$
T_{11}	Sum average	$\sum_{i=2}^{2L} i.p_{x+y}(i)$
T_{12}	Sum entropy	$-\sum_{i=2}^{2L} p_{x+y}(i).\log(p_{x+y}(i))$
T_{13}	Sum variance	$\sum_{i=2}^{2L} (i-T_{12})^2.p_{x+y}(i)$
T_{14}	Difference variance	$\sum_{i=0}^{L-1} i^2.p_{x-y}(i)$
T_{15}	Difference entropy	$-\sum_{i=0}^{L-1} p_{x-y}(i).\log(p_{x-y}(i))$
T_{16}	Information measure of correlation I	$\frac{H_{XY}-H_{XY1}}{\max(H_X,H_Y)}$
T_{17}	Inverse difference normalized	$\sum_i \sum_j \frac{p(i,j)}{1+\frac{\left\|i-j\right\|}{L}}$
T_{18}	Inverse difference moment normalized	$\sum_i \sum_j \frac{p(i,j)}{1+\frac{\left\|i-j\right\|^2}{L^2}}$

Note: L denotes quantization levels of grayscale; $p(i,\ j)$ denotes the $(i,\ j)$th entry of co-occurrence probability matrix; $\sum_i(.)$ and $\sum_j(.)$ are $\sum_{i=1}^{L}(.)$ and $\sum_{j=1}^{L}(.)$, respectively; $\mu_x = \sum_{i=1}^{L}\sum_{j=1}^{L} i.p(i,j)$; $\mu_y = \sum_{i=1}^{L}\sum_{j=1}^{L} j.p(i,j)$; v = mean value of $p(i,\ j)$; $p_{x+y} = \sum_{i=1}^{L}\sum_{j=1}^{L} p(i,j)$ for $i+j=k$; $p_{x-y} = \sum_{i=1}^{L}\sum_{j=1}^{L} p(i,j)$ for $\left|i-j\right| = k$; $p_x(i) = \sum_{j=1}^{L} p(i,j)$; $p_y(j) = \sum_{i=1}^{L} p(i,j)$; H_X = Entropy of p_x; H_Y = Entropy of p_y; H_{XY} = Entropy of $p(i,j)$; and $H_{XY1} = \sum_{i=1}^{L}\sum_{j=1}^{L} p(i,j).\log\left(p_x(i)p_y(j)\right)$.

3. Supervised Machine Learning

Supervised machine learning is a branch of machine learning (ML) that is used for performing classification and regression tasks on labeled data [32]. Labeled data are the data gathered from experiments or observations whose outcomes are known. The factors that govern the outcome of

an observation or an experiment are referred to as descriptive features, and the variable(s) that quantify the outcome is/are referred to as response/target variable(s). The values of descriptive features and response/target variables in a dataset can be either categorical (nominal/ordinal) or numerical (discrete/continuous). Supervised machine learning involves three important steps: (1) gathering the data, (2) training the machine learning algorithm, and (3) testing and deploying the machine learning algorithm for the intended purpose. Data are the workhorses of machine learning algorithms, and gathering high-quality and high-quantity labeled data is the first step in supervised machine learning. The quality and quantity of labeled data are very crucial for improving the predictive power of ML algorithms. An elaborate discussion on the influence of the quality and quantity of data on the predictive power of ML algorithms can be found in [33]. The gathered data is partitioned into two datasets, namely training dataset and test dataset, at the beginning of the second step. Choosing a partition ratio of 80:20 is very common, where the numbers 80 and 20 represent the percentage of gathered data used for training and testing purposes, respectively. The other methods of partitioning the data and associated issues are discussed in reference [34]. Followed by data partition, a machine learning algorithm is employed to learn the patterns, relationships, and/or dependencies from the obtained training dataset in the second step. The efficacy of the trained algorithm is then tested on the testing data (also called validation dataset) in the third step. If the prediction accuracy is satisfactory, then the trained algorithm will be deployed for performing classification or regression tasks on the new data. From this point of discussion in this article, the term supervised machine learning will be replaced by machine learning for the sake of brevity. In the context of this study, machine learning-based classification algorithms were used to learn the textures of metallurgical phases of ASTM A36 steel, and they were then deployed to classify them accurately into different phases. To this end, four machine learning algorithms were employed: (1) naïve Bayes (NB) classifier, (2) k-nearest neighbor (K-NN) algorithm, (3) linear discriminant analysis (LDA), and (4) decision tree (DT) classifier. Coding of these algorithms was carried out in an in-house MATLAB®-based machine learning software. The basic nomenclature employed for describing the data is provided next.

The master dataset was denoted by $\boldsymbol{D} \in \mathbb{R}^{p \times r}$, where p was the number of observations (or in this context, the number of gathered image domains of metallurgical phases), and $r = q + 1$, where q was the number of descriptive textural features ($q = 20 -$ pixel intensity along with 19 textural features); for definitions of textural features, see Table 1. The last $\left(r^{th}\right)$ column of $\boldsymbol{D}$ had the outcome of the experiment or the target variable, which in this case was the metallurgical phase (ferrite, pearlite, or martensite). Each p^{th} row of $\boldsymbol{D}$ was denoted by an instance vector $\boldsymbol{x}_j = \left(x_{j1}, x_{j2}, \ldots, x_{jq}, x_{jr}\right)$, where j took the values from 1 to p, and $p = 735$ in this study. Here, $x_{j1}, x_{j2}, \ldots, x_{jq}$ were the descriptive textural features corresponding to the metallurgical phase identified by x_{jr}. In this study, 80% of the $p = 735$ number of observations were randomly chosen for training purposes, and 20% were chosen for testing purposes. The instance vectors used for training purposes were designated as $\boldsymbol{x}_j^*$, and the vectors used for testing (validation) purposes were designated as $\boldsymbol{x}_j^\#$. Note that the range of j in the training and testing sets was not same as that of the master dataset $\boldsymbol{D}$ and depended on the fractions of data used for training and testing purposes.

3.1. Naïve Bayes

Naïve Bayes is a probabilistic classifier that is derived from Bayes' theorem [35]. Bayes' theorem defines the conditional probability of an event A given event B as follows:

$$P(A|B) = \frac{P(B|A) \times P(A)}{P(B)}, \tag{3}$$

where $P(A|B)$ and $P(B|A)$ are conditional or posterior probabilities, and $P(A)$ and $P(B)$ are prior probabilities. Using this definition, for any given i^{th} observation in the training dataset:

$$P\left(x_{ir} = C_j \middle| x_{i1}, x_{i2}, \ldots, x_{iq}\right) = \frac{P\left(x_{i1}, x_{i2}, \ldots, x_{iq} \middle| x_{ir} = C_j\right) \times P\left(x_{ir} = C_j\right)}{P\left(x_{i1}, x_{i2}, \ldots, x_{iq}\right)}, \tag{4}$$

where $C_{j=1,2,3}$ is the class label for the given descriptive textural features; C_1, C_2, and C_3 are ferrite, pearlite, and martensite, respectively. The joint conditional probability term in the numerator of Equation (4) can be rewritten using the chain rule of probability [36,37] as follows:

$$\begin{aligned} P\left(x_{i1}, \ldots, x_{iq} \middle| x_{ir} \right. & \left. = C_j\right) \\ & = P\left(x_{i1} \middle| x_{ir} = C_j\right) \times P\left(x_{i2} \middle| x_{i1}, x_{ir} = C_j\right) \times \ldots . \\ & \quad \times P\left(x_{iq} \middle| x_{i1}, \ldots, x_{iq}, x_{ir} = C_j\right) \end{aligned} \tag{5}$$

Similarly, the denominator can also be written as:

$$P\left(x_{i1}, x_{i2}, \ldots, x_{iq}\right) = P(x_{i1}) \times P(x_{i2}|x_{i1}) \times \ldots . \times P\left(x_{iq} \middle| x_{i1}, x_{i2}, \ldots, x_{iq-1}\right). \tag{6}$$

Substituting Equations (5) and (6) in Equation (4), the following expression is obtained:

$$P\left(x_{ir} = C_j \middle| x_{i1}, x_{i2}, \ldots, x_{iq}\right) = \frac{P\left(x_{i1} \middle| x_{ir}=C_j\right) \times P\left(x_{i2} \middle| x_{i1}, x_{ir}=C_j\right) \times \ldots . \times P\left(x_{iq} \middle| x_{i1}, \ldots, x_{iq}, x_{ir}=C_j\right) \times P(x_{ir}=C_j)}{P(x_{i1}) \times P(x_{i2}|x_{i1}) \times \ldots . \times P\left(x_{iq} \middle| x_{i1}, x_{i2}, \ldots, x_{iq-1}\right)}. \tag{7}$$

Here, it is important to note that the denominator is independent of the class label $\left(C_j\right)$ and will only serve as a normalizing constant. Hence, the denominator can be dropped without loss of any classification information. Now the left hand side of Equation (7) represents a measure of conditional probability denoted by $M\left(x_{ir} = C_j \middle| x_{i1}, x_{i2}, \ldots, x_{iq}\right)$.

$$\begin{aligned} M(x_{ir} & = C_j \middle| x_{i1}, x_{i2}, \ldots, x_{iq}\right) \\ & = P\left(x_{i1} \middle| x_{ir} = C_j\right) \times P\left(x_{i2} \middle| x_{i1}, x_{ir} = C_j\right) \times \ldots . \\ & \quad \times P\left(x_{iq} \middle| x_{i1}, \ldots, x_{iq-1}, x_{ir} = C_j\right) \times P\left(x_{ir} = C_j\right) \end{aligned} \tag{8}$$

With an increase in the number of descriptive features, the evaluation of conditional probabilities in Equation (8) becomes computationally expensive. To circumvent this computational issue, the naïve Bayes algorithm assumes that the descriptive features are conditionally independent. As a consequence of conditional independence, Equation (8) can be rewritten as follows:

$$M\left(x_{ir} = C_j \middle| x_{i1}, x_{i2}, \ldots, x_{iq}\right) = P\left(x_{i1} \middle| x_{ir} = C_j\right) \times P\left(x_{i2} \middle| x_{ir} = C_j\right) \times \ldots . \times P\left(x_{iq} \middle| x_{ir} = C_j\right) \times P\left(x_{ir} = C_j\right). \tag{9}$$

The output class label $\left(C_{j=1,2,3}\right)$ for a given instance $(x_{i1}^{\#}, x_{i2}^{\#}, \ldots, x_{iq}^{\#})$ belongs to the class which maximizes the value of M. The posterior probabilities and the prior probability of a class were evaluated from the training dataset and were used directly in the above equation to find the class label (metallurgical phase) when the textural features for an unknown metallurgical phase were provided. It was important to note that the textural features of the metallurgical phases extracted in this study were continuous variables. Therefore, to evaluate the conditional probabilities of these textural features, conditional probability density functions were used as proxy measures in place of actual conditional probabilities in Equation (9). The probability density functions considered in this study were a normal distribution and a Weibull distribution.

3.2. K-Nearest Neighbor

K-nearest neighbor (K-NN) classifier is a nonparametric, instance-based classifier that determines the class label of $\boldsymbol{x}^{\#}$ based on the assumption that the instances belonging to the same class are found in close proximity to each other when a consistent measure of proximity is employed. In other words, the class label of $\boldsymbol{x}^{\#}$ will be the same as that of the class label shared by its nearest neighboring instances/observations $\boldsymbol{x}_j^*$. In order to quantify the proximity between the training instance $\boldsymbol{x}_j^*$ and test instance $\boldsymbol{x}^{\#}$ and identify the nearest neighbor instances, generally a distance metric $\lceil$ is employed. Note that this distance $\lceil$ is different from the one that is used to represent the pixel distance in the GLCM matrix evaluation. Although there are numerous distance metrics available (see [38]), Euclidean distance $\lceil$ was used in this study as it is the most commonly used distance metric for continuous descriptive features. It is defined as follows:

$$\lceil(\boldsymbol{x}_j^*, \boldsymbol{x}^{\#}) = \| \boldsymbol{x}_j^* - \boldsymbol{x}^{\#} \|_2 \, , \forall \, \mathrm{j} = 1, 2, \ldots, \mathrm{p}, \tag{10}$$

where $\| \, . \, \|_2$ is the l_2 norm, and p is the number of instances in the training set.

The class label $x_{ir}^{\#}$ corresponding to an instance $\boldsymbol{x}^{\#}$ is then determined as the most frequent class label among the K nearest neighboring instances. In this study, a general rule of thumb, $K = \sqrt{\mathrm{p}}$, was used to estimate the value of K, where p was the number of training instances, and the ceil operator . rounded any positive number to the nearest integer which is greater than the number on which the operator was used.

3.3. Linear Discriminant Analysis

The linear discriminant analysis (LDA) classifier employs a discriminant score function $L_j\left(\boldsymbol{x}^{\#}\right)$ to predict the class labels of a given instance $\boldsymbol{x}^{\#}$. The discriminant score for each class is derived based on the Bayes' theorem provided in Equations (3) and (4). Ignoring the denominator in Equation (4), as it is independent of the class label, we get a measure of conditional probability that can be written as:

$$M\left(x_{ir} = C_j \middle| x_{i1}, x_{i2}, \ldots, x_{iq}\right) = P\left(x_{i1}, x_{i2}, \ldots, x_{iq} \middle| x_{ir} = C_j\right) \times P\left(x_{ir} = C_j\right). \tag{11}$$

The class that maximizes the value of M for a given $\boldsymbol{x}$ is the class of $\boldsymbol{x}$, and the above equation is referred to as Bayes' classifier. However, evaluating the conditional probability in the Bayes' classifier is challenging. In the case of linear discriminant analysis, all the instances that belong to a class C_j are assumed to be sampled from a multivariate normal distribution $\mathcal{N}\left(\boldsymbol{\Sigma}, \boldsymbol{\mu}_j\right)$, where $\boldsymbol{\Sigma}$ and $\boldsymbol{\mu}_j$ are the covariance matrix and mean vector of all features in the instances that belong to class C_j. The probability density function is given as:

$$f\left(\boldsymbol{x} \middle| C_j\right) = \frac{1}{\sqrt{(2\pi)^q |\boldsymbol{\Sigma}|}} \exp\left(-\frac{1}{2}\left(\boldsymbol{x} - \boldsymbol{\mu}_j\right)' \boldsymbol{\Sigma}^{-1} \left(\boldsymbol{x} - \boldsymbol{\mu}_j\right)\right).$$

By substituting $f(\boldsymbol{x}|C_j)$ in the place of $P(\boldsymbol{x}|C_j)$ as a proxy measure, we get:

$$M\left(x_{ir} = C_j \middle| x_{i1}, x_{i2}, \ldots, x_{iq}\right) = \frac{1}{\sqrt{(2\pi)^q |\boldsymbol{\Sigma}|}} \exp\left(-\frac{1}{2}\left(\boldsymbol{x} - \boldsymbol{\mu}_j\right)' \boldsymbol{\Sigma}^{-1} \left(\boldsymbol{x} - \boldsymbol{\mu}_j\right)\right) \times P\left(x_{ir} = C_j\right). \tag{12}$$

Note that $P(\boldsymbol{x}|C_j)$ could be replaced by its proxy value $f\left(\boldsymbol{x}\middle|C_j\right)$, as we were interested in discriminating instances into classes and were not interested in evaluating the actual probabilities. By applying a logarithm on both sides, we get the discriminant score function for class C_j as:

$$L_j(\boldsymbol{x}) = -\frac{1}{2} \log((2\pi)^q |\boldsymbol{\Sigma}|) - \frac{1}{2}\left(\boldsymbol{x} - \boldsymbol{\mu}_j\right)' \boldsymbol{\Sigma}^{-1} \left(\boldsymbol{x} - \boldsymbol{\mu}_j\right) + \log P\left(C_j\right). \tag{13}$$

Noting that $\mathbf{\Sigma}^{-1}$ is symmetric (i.e., $\boldsymbol{x}'\mathbf{\Sigma}^{-1}\boldsymbol{\mu}_j = \boldsymbol{\mu}_j'\mathbf{\Sigma}^{-1}\boldsymbol{x}$) we can simplify the discriminant score function as follows:

$$L_j(\boldsymbol{x}) = -\frac{1}{2}\log((2\pi)^q|\mathbf{\Sigma}|) - \frac{1}{2}\boldsymbol{\mu}_j'\mathbf{\Sigma}^{-1}\boldsymbol{\mu}_j - \frac{1}{2}\boldsymbol{x}'\mathbf{\Sigma}^{-1}\boldsymbol{x} + \boldsymbol{\mu}_j'\mathbf{\Sigma}^{-1}\boldsymbol{x} + \log P\big(C_j\big).$$

By ignoring the terms that are independent of the class (as they do not improve the discriminative power of the algorithm), we obtain the discriminant score function for a class as:

$$L_j(\boldsymbol{x}) = -\frac{1}{2}\boldsymbol{\mu}_j'\mathbf{\Sigma}^{-1}\boldsymbol{\mu}_j + \boldsymbol{\mu}_j'\mathbf{\Sigma}^{-1}\boldsymbol{x} + \log P\big(C_j\big). \tag{14}$$

In the case of linear discriminant analysis, the covariance matrices ($\mathbf{\Sigma}_1, \mathbf{\Sigma}_2$, and $\mathbf{\Sigma}_3$) for all classes are assumed to be equal. In order to better capture the variances in the available dataset, a pooled covariance matrix defined as:

$$\mathbf{\Sigma}_{pl} = \frac{1}{\mathrm{p} - m}\sum_{i=1}^{m}\big(\mathrm{p}_i - 1\big)\mathbf{\Sigma}_i$$

is used in the place of $\mathbf{\Sigma}$ to modify the discriminant score function as follows:

$$L_j(\boldsymbol{x}) = -\frac{1}{2}\boldsymbol{\mu}_j'\mathbf{\Sigma}_{pl}^{-1}\boldsymbol{\mu}_j + \boldsymbol{\mu}_j'\mathbf{\Sigma}_{pl}^{-1}\boldsymbol{x} + \log P\big(C_j\big). \tag{15}$$

To predict the class label $x_{jr}^{\#}$ of a given test instance $\boldsymbol{x}^{\#}$, the discriminant scores $L_{j=1:m}\big(\boldsymbol{x}^{\#}\big)$ are first evaluated for all the m class labels, and then the index of the class label j that yields the maximum L_j value is assigned as the class label for the test instance $\boldsymbol{x}^{\#}$. In the context of this study, the class label $j \in \{C_1,\ C_2, C_3\}$.

3.4. Decision Tree

Decision tree classifier is a nonparametric classifier that employs a hierarchical tree structure to predict the class label of an instance $\boldsymbol{x}^{\#}$. This tree structure is obtained as a result of recursive partitioning of the training dataset, which yields a class label as an outcome at the end [36]. Basic decision tree architecture consists of a root node at the top, intermediate nodes in between, and leaf nodes at the bottom. Each node (root and intermediate) in a tree represents a feature, each descending branch represents a criterion or a decision rule, and the leaf nodes at the bottom represent the final outcomes or the classification labels. In this study, an iterative dichotomiser 3 (ID3) decision tree algorithm [39] was used, and the continuous textural features were treated according to the procedure employed in the C4.5 algorithm [40] to create a decision tree. This algorithm employs entropy and information gain measures for partitioning the data into a decision tree. Entropy is a measure that characterizes the (im)purity or uncertainty of collected observations. Given a column vector of class labels $\boldsymbol{s}$ (i.e., r^{th} column of training dataset), the entropy of the distribution of class labels denoted by $H(\boldsymbol{s})$ is evaluated as:

$$H(\boldsymbol{s}) = -\sum\nolimits_{i=1}^{m} P_i \log_2 P_i, \tag{16}$$

where P_i is the probability that an observation belongs to a class label C_i. Entropy is zero when all classification labels belong to the same class, and it is one when the classification labels are equally proportioned into all classes. In order to evaluate the root node, the unique values of all textural features are sorted into ascending order. The entropy reduction is calculated for every feature choosing each of its unique values. The feature ($\boldsymbol{x}_j = \big(x_{1j}, x_{2j}, \ldots x_{kj}, \ldots, x_{pj}\big)$; p is the number of observations) and the unique value of that feature (x_{kj}), about which the partition is made that maximizes information gain, are taken as the root node to determine the corresponding decision rules for the first two branches of the decision tree. The information gain is defined as the expected reduction in entropy caused by partitioning the class label vector $\boldsymbol{s}$ with respect to the given j^{th} attribute vector

$\boldsymbol{x}_j = \left(x_{1j}, x_{2j}, \ldots x_{kj}, \ldots, x_{pj}\right)$, whose unique values are already sorted into ascending order, where x_{kj} is the partition value. The information gain is defined as:

$$U\left(\boldsymbol{s}, \boldsymbol{x}_j, x_{kj}\right) = H(\boldsymbol{s}) - \frac{|\boldsymbol{s}_l|}{|\boldsymbol{s}|} H(\boldsymbol{s}_l) - \frac{|\boldsymbol{s}_h|}{|\boldsymbol{s}|} H(\boldsymbol{s}_h), \tag{17}$$

where $\boldsymbol{s}_l$ and $\boldsymbol{s}_h$ are subsets of $\boldsymbol{s}$ given as $\boldsymbol{s}_l = \left\{x_{1j}, x_{2j}, \ldots x_{kj}\right\}$ and $\boldsymbol{s}_h = \left\{x_{k+1j}, x_{k+2j}, \ldots x_{pj}\right\}$. and $|*|$ in this context is the cardinality of the subsets. While the first term on right-hand side of the above equation represents the entropy of the distribution of class labels before data partition, the second and the third terms represent the expected decrease in entropy after partitioning the data using attribute $\boldsymbol{x}_j$ about the x_{kj} value. The intermediate nodes and their branches are generated similarly but by excluding the features that were already assigned to nodes previously. This procedure is carried out until each of the branches result in a class label as the output. Decision trees are prone to overfitting, and, hence, tree pruning is recommended [41]. This is specifically true when there is a large number of irrelevant and dependent features. As a feature selection algorithm was used in a preprocessing step, tree pruning was not performed in this study. The ID3 algorithm based on the training data provided a set of decision rules, and a given instance $\boldsymbol{x}^{\#}$ from the test dataset was taken through these decision rules to obtain its unknown class label $x_{ir}^{\#}$.

4. Methodology

The main objective of this study was to identify the distinct metallurgical phases present in ASTM A36 steel using supervised machine learning classifiers. The following step-by-step procedure was adapted in the current study to accomplish this objective (see Figure 3): (1) acquisition of microstructural images of heat treated ASTM A36 steel metallurgical specimens, (2) splitting of the acquired images into training image set and test image set, (3) extraction of textural features for known metallurgical phases from training images, (4) extraction of textural features for unknown metallurgical phases from test images, (5) application of the feature selection algorithm to select the most relevant textural features, (6) training the classifier with the selected relevant textural features, and (7) predicting the metallurgical phases in the test image using a trained classifier. In this section, a detailed description of each step is provided.

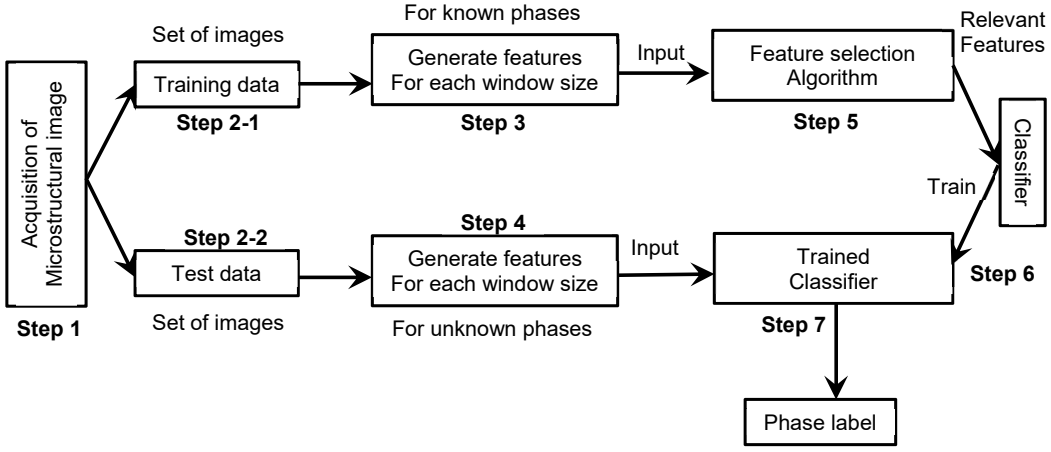

Figure 3. Flowchart of methodology for metallurgical phase identification in ASTM A36 steel using supervised machine learning.

Nine ASTM A36 metallographic specimens cooled from different elevated temperatures were chosen in this study. Of these nine specimens, six were heated to temperatures of 500, 600, 700, 800, 900, and 1000 °C, followed by air cooling; two specimens were heated to temperatures of 900 and 1000 °C, followed by water cooling; and one specimen was extracted from as-received steel. Further

details about the heat treatment of these specimens can be found elsewhere [42,43]. It was important to note that the heat treatment temperatures considered in this study were chosen to obtain various phase compositions in ASTM A36 steel. The air-cooled specimens had a ferrite–pearlite microstructure, and the water-cooled specimens possessed a martensite–ferrite microstructure. All nine metallographic specimens were then examined under an Amscope® optical microscope at 50× magnification to acquire the microstructural images. In total, 45 images were acquired, which included five images each from all nine different metallographic specimens. Note that to obtain these five images from each metallographic specimen, five different locations were chosen on the specimen.

In the second step, the images acquired for each specimen were divided into two sets of images: the training image set and test image set. Out of the five images acquired for every metallographic specimen, three images were allocated to the training image set, the other two images were allocated to the test image set, and this assignment was done randomly. By repeating this exercise for all nine specimens, 27 images in total were generated for the training image set, and 18 images are generated for the test image set. In the third step, the textural features of each metallurgical phase were extracted from the images available in the training image set, and a dataset $\boldsymbol{D}$ was generated. The generated dataset $\boldsymbol{D}$ consisted of 735 data points in total, which included 315 data points corresponding to ferrite, 270 data points corresponding to pearlite, and 150 data points corresponding to martensite. A lower number of data points for pearlite and martensite can be attributed to the lesser number of images available for water-cooled specimens. To extract the textural features of a metallurgical phase from an image, an in-house MATLAB® code was built in this study. This code allowed the user to choose pixels randomly from an image that may correspond to any of the metallurgical phases. A schematic of pixel selection locations adopted in this study is shown in Figure 4. The textural features for each of these selected pixels were then extracted using GLCM (explained in Section 2). To generate dataset $\boldsymbol{D}$ consisting of 735 data points, this exercise was repeated for all 27 images present in the training image set. As mentioned in Section 2, a pixel does not possess a texture, and, hence, a window size of $S \times S$ pixels was used to evaluate the textural features, which were then assigned to the pixel located at the center of this window. As the ideal window size was not known a priori, the following five window sizes were considered in this study: 61×61, 81×81, 101×101, 121×121, and 161×161 pixels. Textural features were computed for all five window sizes.

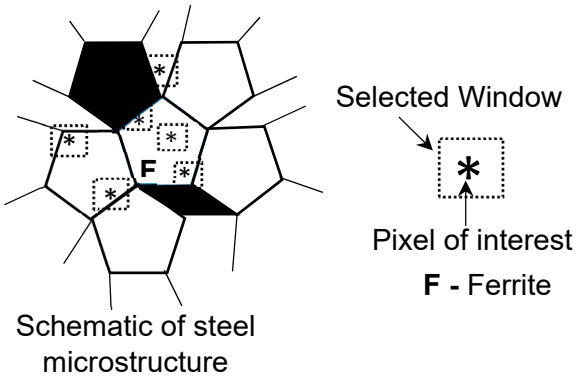

Figure 4. Schematic of pixel locations selected for extraction of textural features.

In the fourth step, the textural features of unknown metallurgical phases in the test images were evaluated using the same procedure described in the third step. These test images were selected from the 18 images that were present in the testing image set and not used for training purposes. However, in this step, every pixel of the test image was selected automatically by the MATLAB® code because the metallurgical phases of these pixels were unknown and should be classified into one of the known phases. The dataset generated in this step was called the test dataset. In the fifth step, feature selection was performed on the dataset $\boldsymbol{D}$ to obtain the subset of the most relevant textural features. In the sixth step, the dataset $\boldsymbol{D}$ consisting of only most relevant textural features was provided as input to the

classifier chosen in this study, and the machine learning algorithm was trained. In the seventh and final step, the test dataset was given as input to the trained algorithm, and unknown phase labels were identified. The flowchart of this methodology is illustrated in Figure 3.

5. Feature Selection

Feature selection is the process of choosing a subset of features from the entire 20 textural features that are ideally necessary and sufficient for predicting the target variable (metallurgical phase) [44]. Feature selection eliminates the redundant and irrelevant features from the evaluated 20 textural features, which otherwise will adversely impact the performance of a given machine learning classifier. In this study, feature selection was carried out using a filter approach that primarily involved two steps: (1) ranking of features and (2) selecting a subset of top-ranked relevant features. A brief description of these two steps is provided below.

5.1. Feature Ranking

Feature ranking is the process of assigning ranks to the descriptive features based on the scores estimated from information measures, distance, similarity, consistency, statistical measures, etc. [45]. These estimated scores represent the correlation/association/relevance between the feature and a target variable or class label. The higher the score, the higher the rank is, and the stronger the correlation is. In this study, a reliefF algorithm [44,46] was employed to estimate the ranking of textural features. This algorithm calculated a proxy statistic for each textural feature, called the feature weight ($W_{i=1:q}$), which was used to estimate the relevance of the textural feature to the target variable or metallurgical phase [45]. To determine the feature weight W_i, the reliefF algorithm employed an iterative updating scheme for weights, which was executed z number of times (z was the user-defined parameter, $z \in \mathbb{Z}_+$) in the following three steps: (1) an instance $\boldsymbol{x}_{t=1:p}$ was sampled at random from the dataset $\boldsymbol{D}$ without replacement in the first step; (2) k (a user-defined parameter, $k \in \mathbb{Z}_+$) number of nearest instances of $\boldsymbol{x}_t$ were determined from each class in the second step; and (3) weights of each textural feature were estimated and updated using Equation (18) in the third step. The set of k nearest instances that were identified in the second step were referred to as nearest-hit instances, $\boldsymbol{h}_j$, if the class label of k instances were the same as that of the class label of $\boldsymbol{x}_t$, and they were referred to as nearest-miss instances, $\boldsymbol{m}_j$, if the class label of the k instances differed from that of the class label of $\boldsymbol{x}_t$, where j took the values from 1 to k. Based on the nearest-hit and nearest-miss instances identified in the second step, this algorithm rewarded or penalized the textural features by updating their weights using Equation (18), which assigned higher weights to features that were strongly correlated to the target variable when compared to the irrelevant features. The number of iterations z was an arbitrary integer and was generally chosen to be $\sqrt{\mathrm{p}}$. The equation to evaluate the weights of the feature is expressed as:

$$W_i = W_i^{old} - \sum_{j=1}^{k} \frac{\mathrm{diff}\left(i, x_{ti}, h_{ji}\right)}{zk} + \sum_{C_o \neq \text{class of } \boldsymbol{x}_t} \frac{\mathrm{P}(C_o)}{1 - \mathrm{P}(\text{class of } \boldsymbol{x}_t)} \sum_{j=1}^{k} \frac{\mathrm{diff}\left(i, x_{ti}, m_{ji}(C_o)\right)}{zk}, \quad (18)$$

where W_i on the left-hand side is the updated weight of the textural feature i; W_i^{old} on the right-hand side is the initial or previous weight of the textural feature i; z is the number of iterations; x_{ti}, h_{ji}, and m_{ji} are the values of textural feature i corresponding to the instance $\boldsymbol{x}_t$, near-hit instance $\boldsymbol{h}_j$, and near-miss instance $\boldsymbol{m}_j$, respectively; C_o is the class label of an instance $\boldsymbol{m}_j$, which is not the same as that of the class label of $\boldsymbol{x}_t$; $\mathrm{P}(C_o)$ is the prior probability of class C_o; and both $\mathrm{diff}\left(i, x_{ti}, h_{ji}\right)$ and $\mathrm{diff}\left(i, x_{ti}, m_{ji}\right)$ evaluate the difference between the normalized values of the i^{th} feature, expressed as:

$$\mathrm{diff}\left(i, x_{ti}, h_{ji}\right) = \frac{\left|x_{ti} - h_{ji}\right|}{\max(i) - \min(i)} \text{ and, } \mathrm{diff}\left(i, x_{ti}, m_{ji}\right) = \frac{\left|x_{ti} - m_{ji}\right|}{\max(i) - \min(i)}.$$

A more detailed description of the algorithm can be found elsewhere [44,46]. The dataset $\boldsymbol{D}$ generated in Section 4 was given as an input to the reliefF algorithm, and the rank of the descriptive features was obtained (see Table 2). This procedure was repeated for all five different window sizes considered in this study.

Table 2. Feature ranking based on the reliefF algorithm.

Rank	61 × 61	81 × 81	101 × 101	121 × 121	161 × 161
1	Intensity	Intensity	Intensity	Intensity	Intensity
2	Maximum probability	Maximum probability	Maximum probability	Maximum probability	Maximum probability
3	Auto-correlation	Sum of squares	Sum of squares	Cluster shade	Sum of squares
4	Sum of squares	Auto-correlation	Auto-correlation	Inverse correlation	Auto-correlation
5	Entropy	Sum variance	Sum variance	Sum of squares	Cluster shade
6	Sum variance	Energy	Inverse correlation	Auto-correlation	Sum variance
7	Cluster shade	Cluster shade	Cluster shade	Energy	Sum average
8	Sum average	Sum average	Energy	Sum variance	Inverse correlation
9	Inverse difference moment	Sum entropy	Sum average	Inverse difference normalized	Inverse difference normalized
10	Sum entropy	Entropy	Inverse difference normalized	Sum average	Energy

5.2. Selection of the Feature Subset

Followed by a ranking of textural features, a subset of relevant textural features was selected in the second step of the feature selection process. In this study, as the number of relevant features was not known a priori, a trial and error approach was used. In this approach, a particular combination of relevant textural features was chosen in each trial, and the resulting misclassification errors for each classifier (from the selected combination) were evaluated. The combination of relevant textural features that minimized the misclassification error was then selected as the most relevant subset of textural features in this study. To select a combination of relevant textural features during each trail, the following procedure was adapted: in the first trial, only one relevant descriptive feature was selected that had the highest relevance index or rank of one (see Table 2); in the second trial, the next relevant descriptive feature with a rank of two was included along with the previous one; in the third trial, the third relevant feature was included with the previous two descriptive features, and this procedure was repeated for each trial. In other words, for each trial, the next most relevant descriptive feature was added successively. Note that these combinations were found to differ for different window sizes.

5.3. Performance Assessment of the Classifier

To evaluate the misclassification error or assess the performance of the classifier, dataset $\boldsymbol{D}$ was split into two subsets: one subset ($\boldsymbol{S}_1$) containing 80% of the observations and the other subset ($\boldsymbol{S}_2$) containing the rest (20%) of the observations. To obtain the subset $\boldsymbol{S}_1$, the observations available in dataset $\boldsymbol{D}$ were randomly sampled without replacement using the 'datasample' function available in MATLAB®. To avoid bias between class labels, sampling was carried out such that 80% of observations from each metallurgical phase were chosen from dataset $\boldsymbol{D}$. The rest of the observations were grouped into subset $\boldsymbol{S}_2$. Here, subset $\boldsymbol{S}_1$ was referred to as the training dataset (for the classifier), and subset $\boldsymbol{S}_2$ was referred to as the validation dataset. In other words, the classifier was first trained using subset $\boldsymbol{S}_1$ and then deployed to predict the class labels on subset $\boldsymbol{S}_2$. As the class labels in subset $\boldsymbol{S}_2$ were already known, the performance of the classifier on subset $\boldsymbol{S}_2$ was assessed by cross-validating the known class labels. For this, the counts of both correctly and incorrectly classified class labels were obtained and summarized in the form of a matrix, called a confusion matrix (see Table 3).

Table 3. Confusion matrix ($\boldsymbol{C}$).

-		**Predicted Class Label**			
		$\boldsymbol{C_1}$	$\boldsymbol{C_2}$	$\cdots$	$\boldsymbol{C_m}$
Actual Class Label	C_1	c_{11}	c_{12}	$\cdots$	c_{1m}
	C_2	c_{21}	c_{22}	$\cdots$	c_{2m}
	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
	C_m	c_{m1}	$\cdots$	$\cdots$	c_{mm}

Note: C_1—Ferrite, C_2—Pearlite, and C_3—Martensite.

A confusion matrix ($\boldsymbol{C}$) is a square matrix of size $m \times m$, where m is the number of class labels or metallurgical phases, and each element c_{ij} of the matrix represents the frequency of instances from the validation dataset that are assigned class j by the classifier, which in reality belongs to class i [47]. In other words, a confusion matrix provides the summary of correct and incorrect classifications predicted by the classifier. Here, $i, j \in \{C_1, C_2, C_3\}$, where C_1, C_2, and C_3 denoted ferrite, pearlite, and martensite, respectively. While the summation of each row of the confusion matrix (see Table 3) represented the number of instances actually belonging to class i in the validation dataset, the summation of each column of the confusion matrix (see Table 3) represented the number of instances that were assigned to class i by the classifier.

Accuracy (A) is often used to quantify the predictive power of classifiers. It is the ratio of a total number of observations whose class labels are correctly predicted to the total number of observations present in the validation dataset. Mathematically, A is defined as:

$$A = \frac{\sum_{i=1}^{m} c_{ii}}{\sum_{i=1}^{m} \sum_{j=1}^{m} c_{ij}} \times 100\%. \tag{19}$$

However, the accuracy estimated from the above equation may be misleading when the dataset is imbalanced (i.e., the number of data points corresponding to each class label is not the same [48]. The training dataset $\boldsymbol{D}$ consisted of 43.7% of data points corresponding to ferrite, 37.5% of data points corresponding to pearlite, and 20.8% of data points corresponding to martensite. To assess the performance of such imbalanced datasets, often the F-measure (F_m) is used instead of accuracy (A). F-measure (F_m) is the harmonic mean of two other accuracy measures, namely precision (O) and recall (R). Precision is defined as the ratio of the number of observations whose class label i is correctly predicted by the classifier to the total number of observations that are assigned to the class i by the classifier, and recall is defined as the proportion of observations of class i (in the validation dataset) that are correctly predicted as class i by the classifier [49]. Provided a confusion matrix $\boldsymbol{C}$, the precision with which a class i instance is classified is the ratio of the number of correctly classified class i instances to the total number of instances that are assigned to class i (i.e., the summation of corresponding column elements $\sum_{j=1}^{m} c_{ji}$) by the classifier. The recall of a class i is evaluated as the ratio of the correctly classified number of instances with class label i to the total number of instances that actually belong to class i (i.e., the summation of corresponding row elements $\sum_{j=1}^{m} c_{ij}$). The overall precision (O) and overall recall (R) are then computed as the average of the precision and recall of all classes, respectively, and are given as follows [49]:

$$\mathrm{O} = \frac{1}{m} \sum\nolimits_{i=1}^{m} \frac{c_{ii}}{\sum_{j=1}^{m} c_{ji}} \text{ and } \mathrm{R} = \frac{1}{m} \sum_{i=1}^{m} \frac{c_{ii}}{\sum_{j=1}^{m} c_{ij}}.$$

Here, m represents the number of class labels. While overall precision and overall recall are also used as the measures of performance assessment for classifiers, F-measure (F_m) combines the trade-off between both overall precision and overall recall [48] and is evaluated as follows:

$$\mathrm{F}_m = \frac{2 \times \mathrm{O} \times \mathrm{R}}{\mathrm{O} + \mathrm{R}} \times 100\%. \tag{20}$$

6. Results

The results of textural feature ranking and the subsets of most relevant textural features for all classifiers is presented in this section. In addition to this, the performance assessment of all four classifiers for the selected subset of relevant textural features is summarized. A numerical example to demonstrate the importance of choosing relevant textural features is also provided in this section. Finally, the unknown metallurgical phases are identified in test images of ASTM A36-500AC, ASTM A36-900AC, and ASTM A36-900WC for the sake of visual validation.

6.1. Feature Ranking

The textural features of the dataset $\boldsymbol{D}$ obtained in Section 4 were ranked using the reliefF algorithm, and the top 10 relevant features were provided in Table 2. From Table 2, it was clear that the rank of textural features depended on the choice of window size (i.e., rank of the textural features changed with the change in window size). This change was attributed to the fact that the image textures obtained for different window sizes were not similar. A smaller window failed to capture the richness of the texture of the metallurgical phase at a given magnification, and it misranked the textural features. On the other hand, larger windows either included too much redundant textural information or incorporated textural information from neighboring metallurgical phases that resulted in the evaluation of texture features not representative of the metallurgical phase under consideration. It was interesting to note that 'pixel intensity' and 'maximum probability' remained the top relevant features for all five window sizes considered in this study. While 'pixel intensity' is perceivable to the human eye as the relative brightness/grayscale, 'maximum probability' is a statistical measure evaluated from the GLCM matrix and has no physical meaning. Mathematically, the 'maximum probability' textural feature is the maximum value of the joint probability occurrence of pixel pairs that are separated by distance d in the θ direction such that the quantized grayscale levels of the first pixel is i and the second pixel is j (see Section 2 and Table 1). 'Pixel intensity' is undoubtedly a very important feature to identify the metallurgical phase. Indeed, many commercially available thresholding methods rely solely on pixel intensity. However, the robustness of the method proposed in this study relied on adding textural features that may not be perceivable by an ordinary human eye but could be measured mathematically and used by a machine learning algorithm to classify/identify unknown metallurgical phases. Besides 'pixel intensity' and 'maximum probability', the other textural features that were identified as the most relevant features for all window sizes included 'auto-correlation', 'sum of squares', 'cluster shade', 'sum variance', 'sum average', and 'energy'. Note that most of these textural features were purely mathematical in nature and did not have physical meaning.

6.2. Feature Subset Selection

The subset of the most relevant textural features obtained by adapting the procedure explained in the previous section is summarized in Table 4. From Table 4, it was observed that the combinations of textural features differed for all five window sizes (see Table 2). This could be attributed to the fact that order of the ranks of the textural features changed with the change in window size as explained in the previous section. In this study, the ideal window size and the subset of relevant textural features (see Table 4) that maximized the performance of each classifier were determined. Accuracy and F-measure were evaluated for this purpose, and the performances assessed for each classifier were provided in Tables 5–8. Additionally, the confusion matrices were also summarized in Tables 5–8.

From Tables 5–8 it was clear that the window size of 161 × 161 pixels yielded a higher accuracy (>97%) and F-measure (>97%) for all four classifiers—naïve Bayes, K-NN, LDA, and decision tree. Note that window sizes larger than 161 × 161 pixels were not considered in this study because larger window sizes may include redundant or conflicting (neighboring phase) texture information and will increase the computational cost, as mentioned previously. Confusion matrices provided in Tables 5–8 aided in inferring the misclassifications or wrong predictions made by the classifier. From the confusion matrices summarized in Tables 5–8, the following two inferences were drawn: (1) the martensite phase was often misclassified into ferrite when only 'pixel intensity' was considered, and (2) misclassification was reduced when relevant textural features were considered in addition to pixel intensity. The first inference can be attributed to the overlap of pixel intensities of ferrite and martensite phases, which was evident from the pixel intensity histogram provided in Figure 1. However, with the addition of textural features to the pixel intensity, misclassification was reduced for all classifiers because of the distinctiveness of textural features for each metallurgical phase which do not overlap. Among all four classifiers considered in this study, naïve Bayes, LDA, and decision tree classifiers were observed to exhibit, more or less, the same robustness with respect to the misclassifications (i.e., there were no significant variations in the confusion matrices with various subsets of textural features considered in this study). On the other hand, K-NN misclassified a number of martensitic phase pixels as ferrite phase pixels, compared to other classifiers, when pixel intensity alone was used. However, with the addition of relevant textural features, misclassification was reduced in the case of the K-NN classifier.

In conjunction with the ideal window size, a combination of top-ranked textural features that resulted in higher F-measures for all four classifiers were determined and were provided in Table 9. The ideal window size and the combination of textural features determined for naïve Bayes classifier were 161 × 161 pixels and ϱ_3; for the K-NN classifier they were 161 × 161 pixels and ρ_5; for the LDA classifier they were 101 × 101 pixels and ρ_6; and for the decision tree classifier they were 161 × 161 pixels and 'All features'. Although the LDA classifier was observed to yield a higher accuracy for a 161 × 161 pixel window size, there were other combinations of window sizes and textural features that yielded similar accuracies. Considering the confusion matrices provided in Table 7, a window size of 101 × 101 pixels was chosen in this study, as it was observed to produce a more reliable classification of metallurgical phases along with the feature combination ρ_6 (see the 3rd row of Table 4). Unlike naïve Bayes, K-NN, and LDA classifiers, feature selection was not carried out for the decision tree classifier in the current study. The decision tree classifier is a low bias/high variance classifier and is sensitive to small fluctuations in the training dataset. While the inclusion of redundant and irrelevant features will decrease the performance of a high bias/low variance classifier (e.g., Naïve Bayes and LDA), it may decrease the performance of a decision tree classifier when there is a significant amount of noise in the training and test data [50], which was not the case in the current study. Therefore, no feature selection was performed for the decision tree classifier. Bias represents error resulting from the limitations of mathematical assumptions made in formulating a machine learning classifier, and variance represents sensitivity to small fluctuations in the training dataset resulting in a model overfit.

Table 4. Combinations of features for each window size.

Window Size	Combination					
	ϱ_1	ϱ_2	ϱ_3	ϱ_4	ϱ_5	ϱ_6
161 × 161	Intensity	$\varrho_1 + T_9$	$\varrho_1 + T_{10}$	$\varrho_2 + T_1$	$\varrho_3 + T_4$	$\varrho_4 + T_{13}$
121 × 121	Intensity	$\varrho_1 + T_9$	$\varrho_1 + T_4$	$\varrho_2 + T_{16}$	$\varrho_3 + T_{10}$	$\varrho_4 + T_1$
101 × 101	Intensity	$\varrho_1 + T_9$	$\varrho_1 + T_{10}$	$\varrho_2 + T_1$	$\varrho_3 + T_{13}$	$\varrho_4 + T_{16}$
81 × 81	Intensity	$\varrho_1 + T_9$	$\varrho_1 + T_{10}$	$\varrho_2 + T_1$	$\varrho_3 + T_{13}$	$\varrho_4 + T_6$
61 × 61	Intensity	$\varrho_1 + T_9$	$\varrho_1 + T_1$	$\varrho_2 + T_{10}$	$\varrho_3 + T_7$	$\varrho_4 + T_{13}$

Table 5. Performance of the naïve Bayes classifier for different combinations of textural features.

Size		C for ϱ_2		F_m		C for ϱ_3		F_m		C for ϱ_4		F_m		C for ϱ_5		F_m		C for ϱ_6		F_m
	0.976	0.000	0.024		0.992	0.000	0.008		0.968	0.016	0.016		0.810	0.016	0.175		0.849	0.016	0.135	
161	0.009	0.954	0.037	95.22	0.000	1.000	0.000	97.70	0.000	0.991	0.009	95.62	0.000	1.000	0.000	91.55	0.009	0.991	0.000	90.32
	0.067	0.000	0.933		0.067	0.033	0.900		0.033	0.067	0.900		0.000	0.017	0.983		0.050	0.050	0.900	
Acc.	-	95.91	-	-	-	97.62	-	-	-	96.26	-	-	-	91.50	-	-	-	91.16	-	-
	0.976	0.008	0.016		0.849	0.016	0.135		0.881	0.000	0.119		0.913	0.000	0.087		0.929	0.008	0.063	
121	0.000	0.991	0.009	94.69	0.000	1.000	0.000	93.00	0.009	0.981	0.009	92.61	0.009	0.991	0.000	94.29	0.000	1.000	0.000	95.79
	0.133	0.017	0.850		0.017	0.000	0.983		0.033	0.017	0.950		0.050	0.000	0.950		0.033	0.000	0.967	
Acc.	-	95.58	-	-	-	93.19	-	-	-	93.19	-	-	-	94.89	-	-	-	96.26	-	-
	0.968	0.000	0.032		0.952	0.000	0.048		0.944	0.008	0.048		0.913	0.000	0.087		0.944	0.000	0.056	
101	0.000	0.972	0.028	95.05	0.000	0.981	0.019	94.69	0.000	0.991	0.009	92.59	0.000	1.000	0.000	94.10	0.000	1.000	0.000	95.99
	0.067	0.017	0.917		0.067	0.017	0.917		0.133	0.033	0.833		0.033	0.033	0.933		0.033	0.017	0.950	
Acc.	-	95.92	-	-	-	95.60	-	-	-	93.88	-	-	-	94.89	-	-	-	96.59	-	-
	0.960	0.000	0.040		0.968	0.000	0.032		0.952	0.008	0.040		0.881	0.008	0.111		0.849	0.000	0.151	
81	0.000	0.963	0.037	95.18	0.000	0.981	0.019	94.99	0.000	1.000	0.000	94.71	0.000	0.981	0.019	90.24	0.000	0.991	0.009	91.21
	0.033	0.017	0.950		0.067	0.033	0.900		0.083	0.033	0.883		0.100	0.033	0.867		0.017	0.050	0.933	
Acc.	-	95.92	-	-	-	95.92	-	-	-	95.58	-	-	-	91.49	-	-	-	91.84	-	-
	0.984	0.000	0.016		0.952	0.000	0.048		0.921	0.000	0.079		0.865	0.000	0.135		0.889	0.000	0.111	
61	0.000	0.963	0.037	94.09	0.000	0.981	0.019	93.85	0.000	1.000	0.000	92.61	0.000	0.981	0.019	92.46	0.000	0.991	0.009	92.75
	0.100	0.033	0.867		0.117	0.000	0.883		0.017	0.117	0.867		0.033	0.000	0.967		0.017	0.050	0.933	
Acc.	-	95.24	-	-	-	94.89	-	-	-	93.88	-	-	-	92.86	-	-	-	93.54	-	-

Note: The term "window size" is replaced by "Size", whose dimensions are denoted by S instead of $S \times S$ pixels; "Acc." here denotes the accuracy.

Table 6. Performance of the k-nearest neighbor (K-NN) classifier for different combinations of textural features.

Size	C for ϱ_2			F_m	C for ϱ_3			F_m	C for ϱ_4			F_m	C for ϱ_5			F_m	C for ϱ_6			F_m
	0.944	0.000	0.056		0.968	0.000	0.032		0.976	0.000	0.024		0.992	0.000	0.008		0.960	0.000	0.040	
161	0.000	0.944	0.056	91.05	0.000	0.954	0.046	93.83	0.000	0.972	0.028	94.57	0.009	0.981	0.009	97.23	0.000	0.972	0.028	94.71
	0.133	0.017	0.850		0.083	0.017	0.900		0.100	0.017	0.883		0.067	0.000	0.933		0.083	0.000	0.917	
Acc.	-	92.52	-	-	-	94.89	-	-	-	95.58	-	-	-	97.62	-	-	-	95.58	-	-
	0.968	0.000	0.032		0.968	0.000	0.032		0.984	0.000	0.016		0.960	0.000	0.040		0.968	0.000	0.032	
121	0.000	0.972	0.028	93.35	0.000	0.972	0.028	94.21	0.000	0.944	0.056	93.79	0.009	0.981	0.009	94.36	0.000	0.963	0.037	92.95
	0.150	0.000	0.850		0.117	0.000	0.883		0.117	0.000	0.883		0.117	0.000	0.883		0.150	0.000	0.850	
Acc.	-	94.56	-	-	-	95.24	-	-	-	94.89	-	-	-	95.24	-	-	-	94.22	-	-
	0.944	0.000	0.056		0.968	0.000	0.032		0.976	0.000	0.024		0.952	0.000	0.048		0.976	0.000	0.024	
101	0.000	0.963	0.037	90.94	0.000	0.963	0.037	93.78	0.000	0.963	0.037	91.18	0.009	0.963	0.028	92.31	0.000	0.972	0.028	93.75
	0.183	0.000	0.817		0.100	0.017	0.883		0.233	0.000	0.767		0.150	0.000	0.850		0.150	0.000	0.850	
Acc.	-	92.52	-	-	-	94.89	-	-	-	92.86	-	-	-	93.54	-	-	-	94.89	-	-
	0.952	0.000	0.048		0.968	0.000	0.032		0.976	0.000	0.024		0.976	0.000	0.024		0.944	0.000	0.056	
81	0.000	0.954	0.046	92.22	0.000	0.981	0.019	93.30	0.000	0.972	0.028	92.85	0.000	0.954	0.046	92.07	0.000	0.963	0.037	92.72
	0.117	0.017	0.867		0.150	0.017	0.833		0.167	0.017	0.817		0.183	0.000	0.817		0.117	0.000	0.883	
Acc.	-	93.54	-	-	-	94.56	-	-	-	94.22	-	-	-	93.54	-	-	-	93.88	-	-
	0.968	0.000	0.032		0.960	0.000	0.040		0.976	0.000	0.024		0.976	0.000	0.024		0.976	0.000	0.024	
61	0.000	0.935	0.065	90.90	0.000	0.954	0.046	90.41	0.000	0.972	0.028	93.71	0.000	0.972	0.028	93.75	0.009	0.981	0.009	96.42
	0.183	0.000	0.817		0.217	0.000	0.783		0.133	0.017	0.850		0.150	0.000	0.850		0.067	0.000	0.933	
Acc.	-	92.52	-	-	-	92.18	-	-	-	94.89	-	-	-	94.89	-	-	-	96.94	-	-

Note: The term "window size" is replaced by "Size", whose dimensions are denoted by S instead of $S \times S$ pixels; "Acc." here denotes the accuracy.

Table 7. Performance of the linear discriminant analysis (LDA) classifier for different combinations of textural features.

Size		C for ϱ_2		F_m		C for ϱ_3		F_m		C for ϱ_4		F_m		C for ϱ_5		F_m		C for ϱ_6		F_m
	0.952	0.000	0.048		0.984	0.000	0.016		0.897	0.000	0.103		0.992	0.000	0.008		0.968	0.000	0.032	
161	0.000	0.944	0.056	95.04	0.000	0.963	0.037	96.73	0.000	0.972	0.028	92.89	0.019	0.981	0.000	97.79	0.009	0.972	0.019	96.49
	0.017	0.000	0.983		0.033	0.000	0.967		0.050	0.000	0.950		0.050	0.000	0.950		0.033	0.000	0.967	
Acc.	-	95.58	-	-	-	97.28	-	-	-	93.54	-	-	-	97.96	-	-	-	96.94	-	-
	0.921	0.000	0.079		0.944	0.000	0.056		0.960	0.000	0.040		0.960	0.000	0.040		0.968	0.000	0.032	
121	0.000	0.963	0.037	93.51	0.000	0.963	0.037	94.96	0.000	0.972	0.028	96.46	0.009	0.991	0.000	97.33	0.000	0.991	0.009	97.17
	0.050	0.000	0.950		0.033	0.000	0.967		0.017	0.000	0.983		0.017	0.000	0.983		0.033	0.000	0.967	
Acc.	-	94.22	-	-	-	95.58	-	-	-	96.94	-	-	-	97.62	-	-	-	97.62	-	-
	0.952	0.000	0.048		0.897	0.000	0.103		0.976	0.000	0.024		0.944	0.000	0.056		0.960	0.000	0.040	
101	0.000	0.954	0.046	94.05	0.000	0.963	0.037	93.03	0.000	0.963	0.037	95.49	0.009	0.963	0.028	95.93	0.000	0.991	0.009	97.23
	0.067	0.000	0.933		0.033	0.000	0.967		0.067	0.000	0.933		0.000	0.000	1.000		0.017	0.000	0.983	
Acc.	-	94.89	-	-	-	93.54	-	-	-	96.26	-	-	-	96.26	-	-	-	97.62	-	-
	0.944	0.000	0.056		0.968	0.000	0.032		0.960	0.000	0.040		0.984	0.000	0.016		0.944	0.000	0.056	
81	0.000	0.944	0.056	94.71	0.000	0.981	0.019	97.21	0.000	0.981	0.019	97.28	0.000	0.954	0.046	96.78	0.000	0.981	0.019	95.69
	0.017	0.000	0.983		0.017	0.000	0.983		0.000	0.000	1.000		0.017	0.000	0.983		0.033	0.000	0.967	
Acc.	-	95.24	-	-	-	97.62	-	-	-	97.62	-	-	-	97.28	-	-	-	96.26	-	-
	0.952	0.000	0.048		0.944	0.000	0.056		0.921	0.000	0.079		0.944	0.000	0.056		0.913	0.000	0.087	
61	0.000	0.926	0.074	93.89	0.000	0.954	0.046	94.61	0.000	0.981	0.019	94.66	0.000	0.981	0.019	96.14	0.009	0.981	0.009	95.30
	0.033	0.000	0.967		0.033	0.000	0.967		0.033	0.000	0.967		0.017	0.000	0.983		0.000	0.000	1.000	
Acc.	-	94.56	-	-	-	95.24	-	-	-	95.24	-	-	-	96.60	-	-	-	95.58	-	-

Note: The term "window size" is replaced by "Size", whose dimensions are denoted by S instead of $S \times S$ pixels; "Acc." here denotes the accuracy.

Table 8. Performance of the decision tree (DT) classifier.

Size	*C* for All Features			F_m
	0.976	0.000	0.024	
161	0.009	0.954	0.037	97.39
	0.067	0.000	0.933	
Acc.	-	97.39	-	-
	0.976	0.008	0.016	
121	0.000	0.991	0.009	91.11
	0.133	0.017	0.850	
Acc.	-	91.02	-	-
	0.968	0.000	0.032	
101	0.000	0.972	0.028	96.07
	0.067	0.017	0.917	
Acc.	-	96.02	-	-
	0.960	0.000	0.040	
81	0.000	0.963	0.037	93.49
	0.033	0.017	0.950	
Acc.	-	93.20	-	-
	0.984	0.000	0.016	
61	0.000	0.963	0.037	92.92
	0.100	0.033	0.867	
Acc.	-	92.82	-	-

Note: The term "window size" is replaced by "Size" whose dimensions are denoted by S instead of $S \times S$ pixels; "Acc." here denotes the accuracy.

Table 9. Window size and subset of features.

Classifier	Window Size	Feature Combination
Naïve Bayes	161 × 161	ϱ_3
K-NN	161 × 161	ϱ_5
LDA	101 × 101	ϱ_6
DT	161 × 161	All

6.3. Example Problem

A numerical example (microstructure) is provided in this section to demonstrate the importance of choosing the most relevant textural features for the prediction of metallurgical phases. This microstructure (see Figure 5a) consisted of two distinct metallurgical phases: ferrite and pearlite. A K-NN classifier with an ideal window size of 161×161 pixels (see Table 9) was chosen, and classification was performed with the following textural features: (a) 'pixel intensity'; (b) 'pixel intensity' and the top three textural features, ρ_4 (see Table 4); and (c) 'pixel intensity' and the top four textural features, ρ_5 (see Table 4). Results obtained for each selected set of relevant features is shown in Figure 5b–d. From Figure 5b, it was inferred that the K-NN classifier predicted a significant portion of martensite in the microstructure when only 'pixel intensity' was considered. This prediction was not true and could be attributed to the fact that a considerable portion of martensite pixel intensities overlapped with ferrite and pearlite, which resulted in such a prediction (see Figure 1b). However, considering the top three relevant textural features in addition to 'pixel intensity' (i.e., ρ_4), there was a reduction in martensite (see Figure 5c). Further addition of one more textural feature to ρ_4 resulted in the elimination of martensite misclassification and grain boundaries from the microstructure (see Figure 5d). The slight amount of martensite and low number of grain boundaries observed in the final microstructure were attributed to the error from the classifier. With this, it was concluded that the addition of relevant textural features to 'pixel intensity' improved the prediction accuracy of the K-NN classifier. This was also proved for other classifiers but was omitted to avoid repetition. In other words, 'pixel intensity' alone was inadequate for phase identification, especially when pixel intensity of different metallurgical phases overlapped. Note that further addition of features to the optimal subset of features provided in

Table 9 would degrade the prediction accuracy of the classifier. In this study, a reliefF algorithm was employed to rank the features in decreasing order of relevance. Hence, when more than the first few relevant features were chosen, prediction accuracies were observed to decrease, as the low-ranked features were irrelevant [51].

For comparison purposes, the same microstructure was also processed in ImageJ, an image processing software, to identify the phases and segmented images, as shown in Figure 5e. In Figure 5e, two distinct phases were present in the segmented image of the microstructure. At this juncture, it was important to note that this segmentation process required the end-user to input the number of phases present in the microstructure, which was not necessary for the procedure proposed in the manuscript. Based on the provided input and a single threshold level, distinct metallurgical phases were identified when ImageJ was used. Unlike the proposed method, this segmentation process failed to identify the grain boundary and categorized it into pearlite. While ImageJ predicted a pearlite volume fraction of 15.3%, the trained classifier predicted a pearlite volume fraction of 12.2% (i.e., a comparatively lower volume fraction of pearlite pixels (3%) were predicted by the trained classifier, reducing the misclassification of grain boundaries into pearlite). This was attributed to the similar pixel intensities exhibited by the both grain boundary and pearlite.

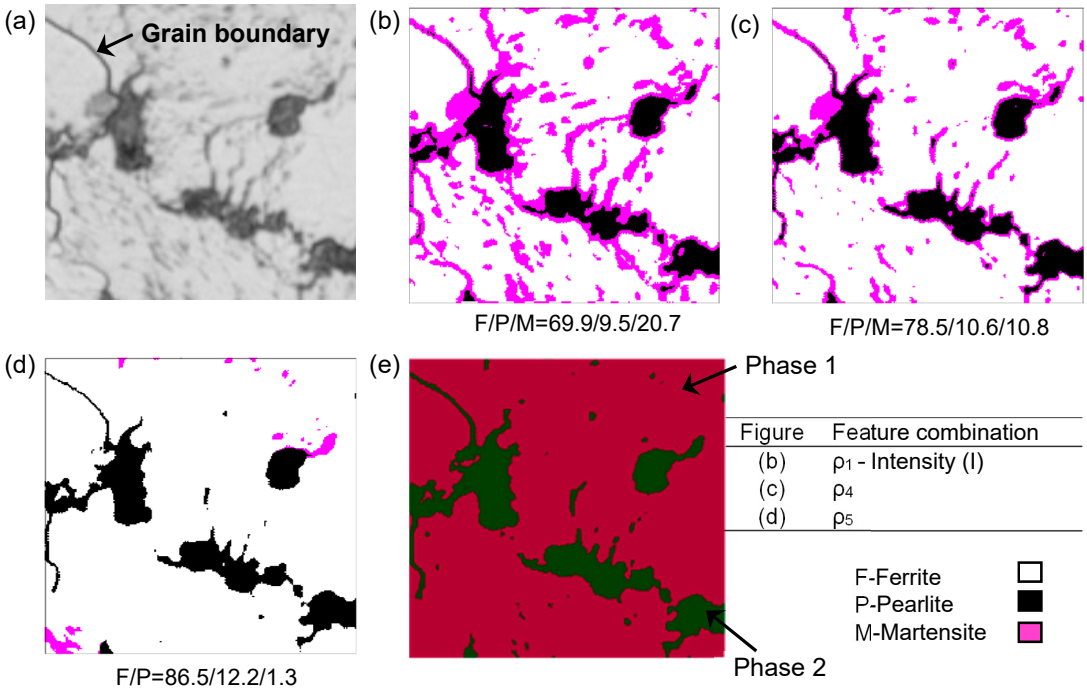

Figure 5. A numerical example illustrating the importance selecting the most relevant textural features: (**a**) original image, (**b**) only 'pixel intensity', (**c**) 'pixel intensity' and top 3 textural features, (**d**) 'pixel intensity' and top 4 textural features, and (**e**) segmentation in ImageJ. It is observed from subfigure (**d**) that the accuracy of classifying metallurgical phases has increased with the addition of more relevant textural features.

6.4. Validation

In this study, three different microstructural images were randomly chosen from the test image set to validate the proposed texture-based phase identification in a microstructure. These microstructural images corresponded to ASTM A36-500 air-cooled (500AC), ASTM A36-900 air-cooled (900AC), and ASTM A36-900 water-cooled (900WC) specimens, which are shown in Figures 6–8. Textural features were extracted using the procedure explained in Section 4, which served as the test data. Based on the combinations of textural features and ideal window sizes provided in Table 9, the metallurgical phases of all three test images were identified, and they are shown in Figures 6–8. In Figures 6–8, all four

classifiers were able to predict the metallurgical phases accurately for both air-cooled (ferrite–pearlite) (Figures 6 and 7) and water-cooled (martensite and ferrite) microstructures (Figure 8). Besides the identification of metallurgical phases, the volume fractions of the phases were also evaluated for all four classifiers, and they are summarized in Table 10. The volume fractions of ferrite and pearlite predicted for (1) ASTM A36-500AC were 81.6% and 17.7%, respectively, and for (2) ASTM A36-900AC they were 78.8% and 19.8%, respectively. For the ASTM A36-900WC specimen, the volume fractions of ferrite and martensite were predicted to be 36% and 64%, respectively.

Table 10. Volume fractions (%) of distinct metallurgical phases.

Classifier	Microstructure								
	500AC			900AC			900WC		
	Ferrite	Pearlite	Martensite	Ferrite	Pearlite	Martensite	Ferrite	Pearlite	Martensite
NB	79.2	20.8	0	77	23	0	39	0	61
K-NN	81.2	18.2	0.6	76.9	20.6	2.5	37	0	63
LDA	81	16.8	2.2	80.2	18.8	1	33	0	67
DT	85	15	0	81.1	17.5	1.4	31	1.5	67.5

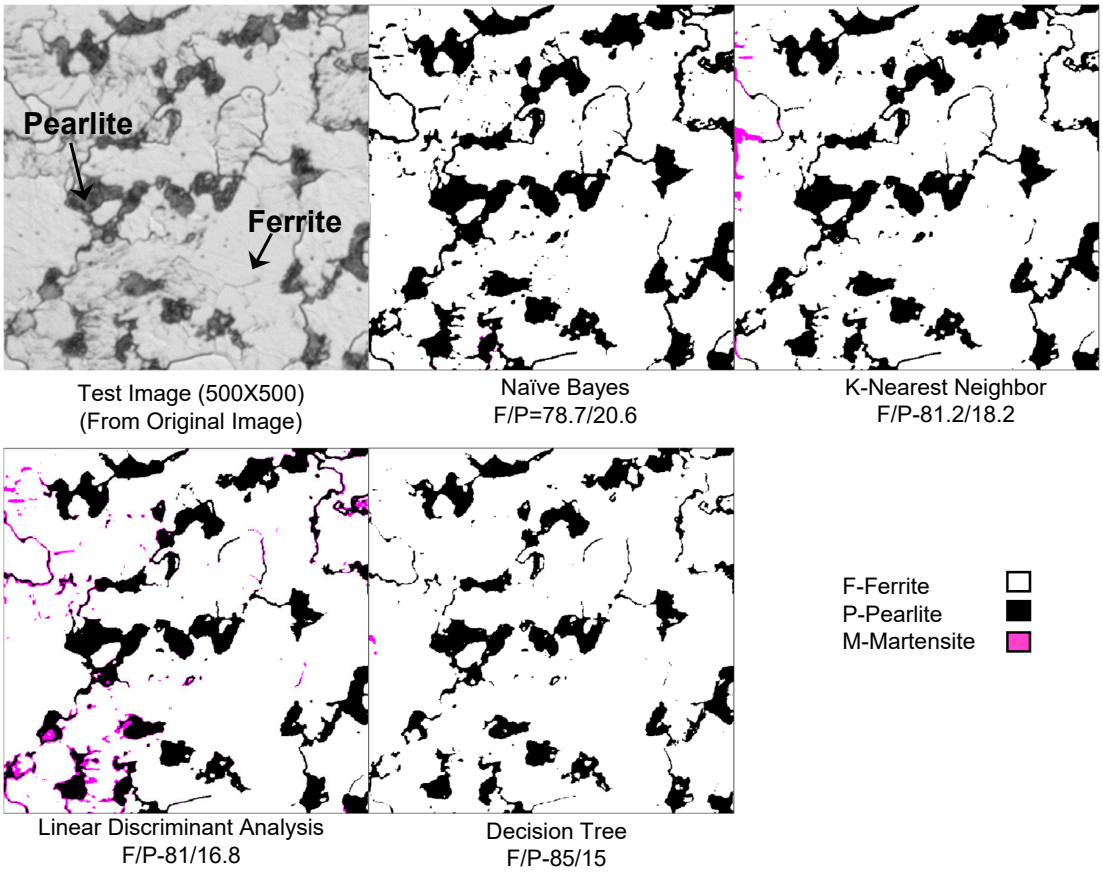

Figure 6. The microstructure of ASTM A36-500AC and machine learning-based metallurgical phase identification. All four classifiers predicted the phases accurately.

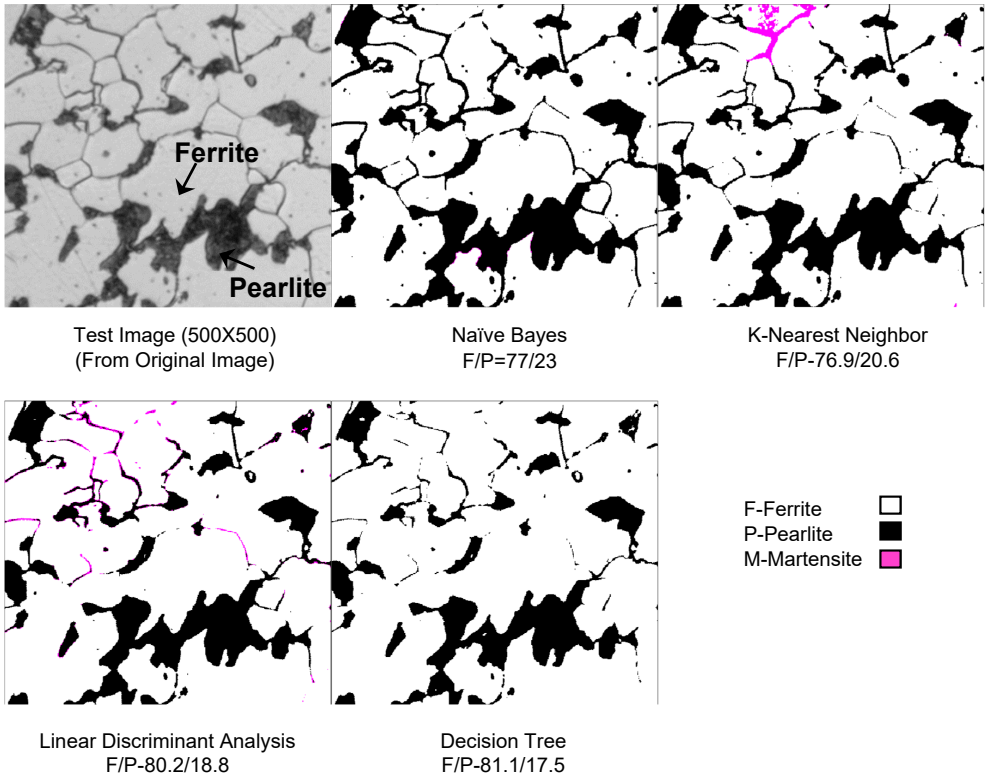

Figure 7. The microstructure of ASTM A36-900AC and machine learning-based metallurgical phase identification. All four classifiers predicted the phases accurately.

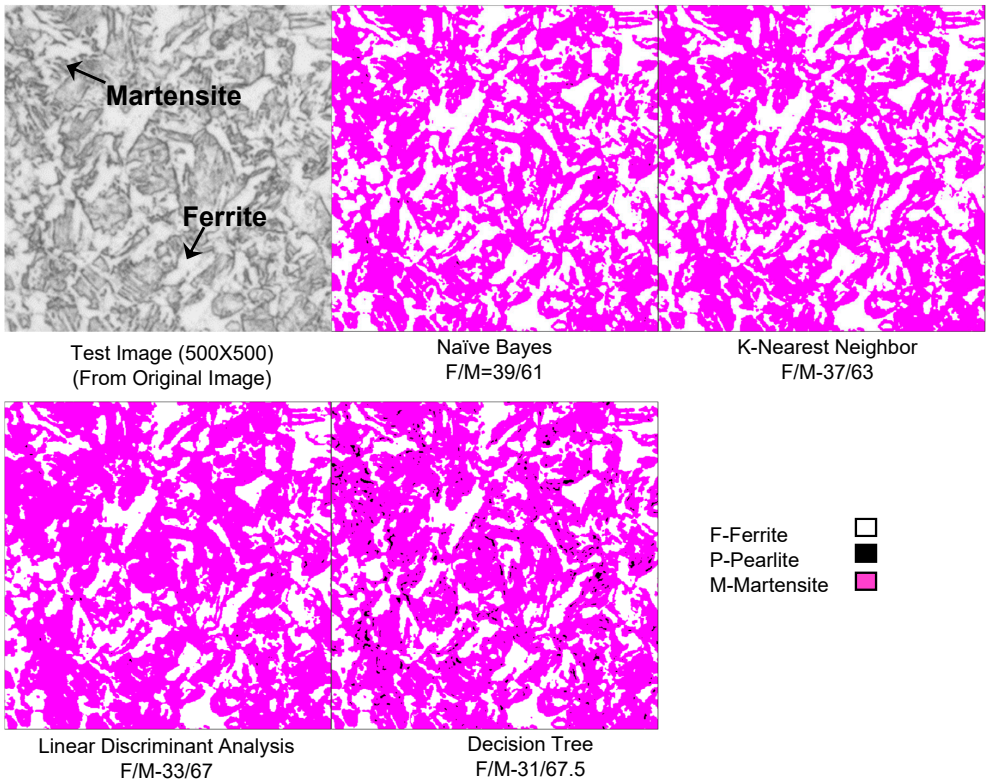

Figure 8. The microstructure of ASTM A36-900WC and machine learning-based metallurgical phase identification. All four classifiers predicted the phases accurately.

7. Summary and Recommendations

In this study, a supervised machine learning approach was proposed to identify the metallurgical phases in ASTM A36 heat-treated steel, namely ferrite, pearlite, and martensite. To identify or classify metallurgical phases in the microscopic images, both the pixel intensities and textural features were extracted from the images for individual phases, which were then used as the descriptive features for machine learning classifiers. To extract textural features, GLCMs of each metallurgical phase were evaluated, and to perform classification, naïve Bayes, K-NN, LDA, and decision tree classifiers were employed. Microstructural images corresponding to nine different heat-treated metallographic specimens were acquired using an optical microscope, and descriptive (textural) features were generated for all three metallurgical phases and stored in a dataset $\boldsymbol{D}$. As the ideal window size for extraction of textural features was not known a priori, window sizes of 61×61, 81×81, 101×101, 121×121, and 161×161 pixels were considered to extract the textural features, which were allocated to the center pixel of each window. Feature selection was performed on dataset $\boldsymbol{D}$ using a reliefF algorithm, and the most relevant features were obtained. Among the 20 descriptive features, 'pixel intensity', 'maximum probability', 'auto-correlation', 'sum of squares', 'cluster shade', 'sum variance', 'sum average', and 'energy' were found to be most relevant features for all five window sizes. The performances of all four classifiers were assessed, and the ideal window size and a combination of the most relevant features that minimized classification error were determined. A numerical example was provided to demonstrate the importance of choosing the most relevant features, and validation of the proposed approach was carried out on three different microstructural images that were not part of the training data. Unlike the threshold-based segmentation approach, the proposed approach avoided misclassification of grain boundaries into pearlite. Further, the proposed approach did not require the end-user to input the number of metallurgical phases present in the microstructure, which is advantageous when investigating new microstructures.

Based on the current study, the following two recommendations can be made for researchers or engineers interested in using machine vision for identifying metallurgical phases: (1) a sufficient number of data points must be acquired (under consistent illumination conditions) to train the classifiers, and (2) an optimal window size must be determined in conjunction with the subset of the most relevant textural features for an accurate prediction of metallurgical phases.

Author Contributions: Conceptualization, D.L.N. and R.K.; Formal analysis, D.L.N.; Funding acquisition, R.K.; Investigation, R.K.; Methodology, D.L.N., H.U.S. and R.K.; Project administration, R.K.; Resources, H.U.S. and R.K.; Software, D.L.N.; Supervision, R.K.; Validation, D.L.N.; Writing—original draft, D.L.N.; Writing—review & editing, D.L.N. and R.K.

Funding: This research received financial support from the North Dakota Established Program to Stimulate Competitive Research (ND EPSCoR).

Acknowledgments: The authors would like to acknowledge the financial support from the North Dakota Established Program to Stimulate Competitive Research (ND EPSCoR).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Clemens, H.; Mayer, S.; Scheu, C. Microstructure and Properties of Engineering Materials. In *Neutrons and Synchrotron Radiation in Engineering Materials Science: From Fundamentals to Applications*; Wiley: Hoboken, NJ, USA, 2017; pp. 1–20.
2. Fan, Z. Microstructure and Mechanical Properties of Multiphase Materials. Ph.D. Thesis, University of Surrey, Guildford, UK, February 1993.
3. Bales, B.; Pollock, T.; Petzold, L. Segmentation-free image processing and analysis of precipitate shapes in 2D and 3D. *Modell. Simul. Mater. Sci. Eng.* **2017**, *25*, 045009. [CrossRef]
4. Beddoes, J.; Bibby, M. *Principles of Metal Manufacturing Processes*; Butterworth-Heinemann: Oxford, UK, 1999.
5. Hall, E. The deformation and ageing of mild steel: III discussion of results. *Proc. Phys. Soc. Sect. B* **1951**, *64*, 747. [CrossRef]

6. Petch, N. The influence of grain boundary carbide and grain size on the cleavage strength and impact transition temperature of steel. *Acta Metall.* **1986**, *34*, 1387–1393. [CrossRef]
7. American Society for Testing and Materials. *ASTM E112-96 (2004) e2: Standard Test Methods for Determining Average Grain Size*; ASTM: West Conshohocken, PA, USA, 2004.
8. ASTM E562-08. *Standard Test Method for Determining Volume Fraction by Systematic Manual Point Count*; ASTM International: West Conshohocken, PA, USA, 2008.
9. Campbell, A.; Murray, P.; Yakushina, E.; Marshall, S.; Ion, W. New methods for automatic quantification of microstructural features using digital image processing. *Mater. Des.* **2018**, *141*, 395–406. [CrossRef]
10. Ayech, M.B.H.; Amiri, H. Texture description using statistical feature extraction. In Proceedings of the 2016 2nd International Conference on Advanced Technologies for Signal and Image Processing (ATSIP), Monastir, Tunisia, 21–23 March 2016; IEEE: Piscataway Township, NJ, USA, 2016; pp. 223–227.
11. Acharya, T.; Ray, A.K. *Image Processing: Principles and Applications*; John Wiley & Sons: Hoboken, NJ, USA, 2005.
12. Pakhira Malay, K. *Digital Image Processing and Pattern Recognition*; PHI Learning Private Limited: Delhi, India, 2011.
13. Sobel, I. *Camera Models and Machine Perception*; Computer Science Department, Technion: Haifa, Israel, 1972.
14. Dougherty, G. *Digital Image Processing for Medical Applications*; Cambridge University Press: Cambridge, UK, 2009.
15. Jahne, B. *Digital Image Processing: Concepts, Algorithms, and Scientific Aplications*; Springer-Verlag: Heildelberg/Berlin, Germany, 1997; p. 570.
16. Otsu, N. A threshold selection method from gray-level histograms. *IEEE Trans. Syst. Man Cybern.* **1979**, *9*, 62–66. [CrossRef]
17. Jain, A.K. *Fundamentals of Digital Image Processing*; Englewood Cliffs, N.J., Prentice Hall: Upper Saddle River, NJ, USA, 1989.
18. Jiang, X.; Zhang, R.; Nie, S. Image segmentation based on level set method. *Phys. Procedia* **2012**, *33*, 840–845. [CrossRef]
19. Sosa, J.M.; Huber, D.E.; Welk, B.; Fraser, H.L. Development and application of MIPAR™: A novel software package for two-and three-dimensional microstructural characterization. *Integrating Mater. Manuf. Innov.* **2014**, *3*, 10. [CrossRef]
20. Bonnet, N. Some trends in microscope image processing. *Micron* **2004**, *35*, 635–653. [CrossRef]
21. Yang, D.; Liu, Z. Quantification of microstructural features and prediction of mechanical properties of a dual-phase Ti-6Al-4V alloy. *Materials* **2016**, *9*, 628. [CrossRef] [PubMed]
22. Buriková, K.; Rosenberg, G. Quantification of microstructural parameter ferritic-martensite dual phase steel by image analysis. *Metal* **2009**, *5*, 19–21.
23. Ontman, A.Y.; Shiflet, G.J. Microstructure segmentation using active contours—Possibilities and limitations. *JOM* **2011**, *63*, 44–48. [CrossRef]
24. Coverdale, G.; Chantrell, R.; Martin, G.; Bradbury, A.; Hart, A.; Parker, D. Cluster analysis of the microstructure of colloidal dispersions using the maximum entropy technique. *J. Magn. Magn. Mater.* **1998**, *188*, 41–51. [CrossRef]
25. Azimi, S.M.; Britz, D.; Engstler, M.; Fritz, M.; Mücklich, F. Advanced Steel Microstructural Classification by Deep Learning Methods. *Sci. Rep.* **2018**, *8*, 2128. [CrossRef]
26. Prakash, P.; Mytri, V.; Hiremath, P. Fuzzy Rule Based Classification and Quantification of Graphite Inclusions from Microstructure Images of Cast Iron. *Microsc. Microanal.* **2011**, *17*, 896–902. [CrossRef]
27. DeCost, B.L.; Holm, E.A. A computer vision approach for automated analysis and classification of microstructural image data. *Comput. Mater. Sci.* **2015**, *110*, 126–133. [CrossRef]
28. Haralick, R.M. Statistical and structural approaches to texture. *Proc. IEEE* **1979**, *67*, 786–804. [CrossRef]
29. Haralick, R.M.; Shanmugam, K. Textural features for image classification. *IEEE Trans. Syst. Man Cybern.* **1973**, *6*, 610–621. [CrossRef]
30. Soh, L.-K.; Tsatsoulis, C. Texture analysis of SAR sea ice imagery using gray level co-occurrence matrices. *IEEE Trans. Geosci. Remote Sens.* **1999**, *37*, 780–795. [CrossRef]
31. Gómez, W.; Pereira, W.; Infantosi, A.F.C. Analysis of co-occurrence texture statistics as a function of gray-level quantization for classifying breast ultrasound. *IEEE Trans. Med. Imaging* **2012**, *31*, 1889–1899. [CrossRef] [PubMed]

32. Camastra, F.; Vinciarelli, A. *Machine Learning for Audio, Image and Video Analysis: Theory and Applications*; Springer-Verlag: London, UK, 2015.
33. Sessions, V.; Valtorta, M. The Effects of Data Quality on Machine Learning Algorithms. *ICIQ* **2006**, *6*, 485–498.
34. James, G.; Witten, D.; Hastie, T.; Tibshirani, R. *An Introduction to Statistical Learning*; Springer-Verlag: New York, NY, USA, 2013; Volume 112.
35. Kelleher, J.D.; Mac Namee, B.; D'Arcy, A. *Fundamentals of Machine Learning for Predictive Data Analytics: Algorithms, Worked Examples, and Case Studies*; MIT Press: Cambridge, MA, USA, USA 2015.
36. Aggarwal, C.C. *Data Classification: Algorithms and Applications*; CRC Press: New York, NY, USA, 2014.
37. Naik, D.L.; Kiran, R. Naïve Bayes classifier, multivariate linear regression and experimental testing for classification and characterization of wheat straw based on mechanical properties. *Ind. Crops Prod.* **2018**, *112*, 434–448. [CrossRef]
38. Kononenko, I.; Kukar, M. *Machine Learning and Data Mining*; Horwood Publishing: West Sussex, UK, 2007.
39. Mitchell, T.M. *Machine Learning*; McGraw-Hill, Inc.: New York, NY, USA, 1997; p. 432.
40. Quinlan, J.R. *C4.5: Programs for Machine Learning*; Morgan Kaufmann Publishers: San Mateo, CA, USA, 2014.
41. Bramer, M. *Principles of Data Mining*; Springer-Verlag: London, UK, 2007; Volume 180.
42. Sajid, H.U.; Kiran, R. Influence of stress concentration and cooling methods on post-fire mechanical behavior of ASTM A36 steels. *Constr. Build. Mater.* **2018**, *186*, 920–945. [CrossRef]
43. Sajid, H.U.; Kiran, R. Post-fire mechanical behavior of ASTM A572 steels subjected to high stress triaxialities. *Eng. Struct.* **2019**, *191*, 323–342. [CrossRef]
44. Kira, K.; Rendell, L.A. The feature selection problem: Traditional methods and a new algorithm. In Proceedings of the Tenth National Conference on Artificial Intelligence, San Jose, CA, USA, 12–16 July 1992; pp. 129–134.
45. Urbanowicz, R.J.; Meeker, M.; La Cava, W.; Olson, R.S.; Moore, J.H. Relief-based feature selection: introduction and review. *J. Biomed. Inf.* **2018**. [CrossRef]
46. Robnik-Šikonja, M.; Kononenko, I. Theoretical and empirical analysis of ReliefF and RReliefF. *Mach. Learn.* **2003**, *53*, 23–69. [CrossRef]
47. Müller, M.E. *Relational Knowledge Discovery*; Cambridge University Press: Cambridge, UK, 2012.
48. Akosa, J. Predictive Accuracy: A Misleading Performance Measure for Highly Imbalanced Data. In Proceedings of the SAS Global Forum 2017, Orlando, FL, USA, 2–5 April 2017.
49. Sokolova, M.; Lapalme, G. A systematic analysis of performance measures for classification tasks. *Inf. Process. Manag.* **2009**, *45*, 427–437. [CrossRef]
50. Ratanamahatana, C.A.; Gunopulos, D. Scaling up the naive Bayesian classifier: Using decision trees for feature selection. 2002.
51. Kohavi, R.; John, G.H. Wrappers for feature subset selection. *Artif. Intel.* **1997**, *97*, 273–324. [CrossRef]