Martian surface characterization using supervised machine learning

Utkarsh Mali,[1,2]★
[1]Department of Physics, University of Toronto, 60 St. George Street, Toronto ON M5S 1C6, Canada
[2]Canadian Institute of Theoretical Astrophysics, University of Toronto, 27 King's College Cir, Toronto M5S 3H8, Canada

Accepted XXX. Received YYY; in original form ZZZ

ABSTRACT

Sublimation in the polar ice regions occur due to a buildup of carbon dioxide (CO2) under the winter surface ice. In the spring, solar radiation causes the ice to rupture, sending the CO2 and eroded dust airborne. The dust is blown by surface winds, gradually forming detectable topographic features. I explore the use of supervised machine learning methods in detecting and characterizing these surface features. I begin by preprocessing the data using principal component analysis along with other preprocessing methods. I then implement a suite of supervised machine learning methods using the k-fold cross validation technique. In general, the accuracy scores fall between 60% to 70%. The best performing model was a Gaussian naive Bayes with an accuracy of 71.8%. Upon further analysis I find the model to skew towards falsely predicting positive features. Over 20% of the items that were not features or "fans" were predicted as a feature. After discussing this issue, I conclude by providing potential followup studies. Overall, this project aims to highlight the potential for machine learning to aid the study of Martian surface recognition. By involving citizen scientists in the classification of these features we, as a community, aim to further engage the people in the study of Mars. In doing so, foster a greater understanding of the planet among the general public.

Key words: methods: statistical – planets and satellites: surfaces – planets and satellites: atmospheres

1 INTRODUCTION

The planet Mars has long been a subject of fascination for the both the public, and the scientific community (David & Howard 2016). Studying its properties has occupied many great researchers. Modern instruments have greatly enhances our ability to study the planet and its various features (Cohen et al. 2019). One area of particular interest is the study of cold jets in the southern region of Mars, which has been shown to have an impact on the planet's atmosphere (Kieffer 2007). These cold jets arise from the sublimation of carbon dioxide in the polar ice cap. The thickness of the Martian surface ice can exceed 1 meter (Kieffer et al. 2006). During the winter, carbon dioxide forms under the thick surface ice causing high-pressure gas to build up. Once the temperature rises, solar radiation impacting the gas under the ice increases, eventually leading to ruptures in the thick ice. Eventually, sublimation occurs. One the ice ruptures, it triggers a disruptive jet spraying the sub-ice carbon dioxide up into the air as a strong wind. These winds have been shown to be affected by the atmosphere. (Kaufmann & Hagermann 2017). As the CO2 jet releases, the jet pressure erodes the Martian surface dust and cases it to be blown up along with the CO2. The airbone dust slowly settles onto the Martian surface, it is affected by the surface currents and winds as it settles (Aye et al. 2019). The imprint that is left on the surface carries information about the weather pattern at the time at which the sublimation occurred. These surfaces features can be used to analyse the local and seasonal wind providing insight into the Martian weather patterns.

In recent years, there has been an increase in the number of satellites and surveys observing Mars (Sharma et al. 2021; Fisher et al. 2005; Zurek & Smrekar 2007; Balme et al. 2006; Montmessin et al. 2017). This greatly accelerates the study of the planet, and its various features. This has lead to an increase in the amount of data that needs to be processed by scientists (Pan et al. 2017; Maltagliati et al. 2011). The surveys generate vasts amounts of data about the surface and atmosphere presenting both challenges and opportunities. The sheer volume of data can be difficult to manage and analyse. On the other hand, it provides a wealth of information which can be used to infer new insights from. One way to address the challenge is to involve citizen scientists (Bird et al. 2018; Banerji et al. 2010; Mali & Rogers Mali & Rogers). Doing this speeds up the process of data analysis and makes it more efficient. Citizen scientist projects like Galaxy Zoo and WISE have already demonstrated the potential for involving the public in academic research (Peng et al. 2018; Nguyen et al. 2018; Raddick et al. 2009, 2013; Fortson et al. 2012; Lintott et al. 2008). The Planet Four Collaboration is another citizen science project. First initiated in 2014 by a team of researchers at the University of Arizona. It focuses on the topographic features of the Martian surface, such as fans, streaks and blotches that appear during the sublimation process (Schwamb et al. 2018). By recruiting volunteers to classify these features, the collaboration was able to generate large amount of data which can be used to train machine learning models (Sprinks et al. 2019). In doing this, the collaboration will be able to greatly speed up the process of data analysis. In addition, off loading the predictive task to a machine learning model will enable scientist to focus on the

★ E-mail: utkarsh.mali@utoronto.ca

identifying macroscopic patterns in the data. This is more likely to provide valuable insight about new science.

In my exploratory study, I will explore the extent to which methods in supervised machine learning (ML) can aid the classification of Martian surface features. To accomplish this, I begin by formatting the images into a standard format. I then clean and preprocess the image data to enable better training of the ML models. This includes data augmentation, gamma correction, principal component analysis (PCA) and histogram oriented gradients (HOG). Once cleaned, I apply multiple supervised ML models to train the data. Some of the models are frequentist; logistic regression (LR), support vector machines (SVM), k-nearest neighbours (KNN), while others are Bayesian; linear/quadratic discriminant analysis (LDA/QDA), naive Bayes (NB), Gaussian process (GP). Finally, I explore the possibility of applying unsupervised methods such as stochastic gradient descent (SGD). Once trained I evaluate the model performance with with accuracy scores and study the best model in dept through a confusion matrix.

2 DATA

The data for this project was measured on the Mars Reconnaissance Orbiter (MRO), a spacecraft launched by NASA in 2005. Its mission was to map important regions on Martian surface. It contained many cameras and feature detection devices, one of which is the *High Resolution Imaging Science Experience* or HiRISE. It was used to study the geology and surface features on Mars with a resolution of up to 0.3 meters per pixel (McEwen et al. 2007; Simpson et al. 2014). Over its commission the HiRISE instrument focused on a few regions of interest in the south polar region, mapping a large portion of the polar martian surface as it did. These images capture the surface features such as fans, rivers, cracks and blotches highlighted in Section 1, I will focus on fans. The photos taken are open source and can be found on the Zooniverse Planet Four Collaboration. The catalogue includes images of the Martian surface along with the classifications of features marked by citizen scientists. The catalogue metadata includes the observations of each general photo with its location along the Martian surface `P4_catalog_v1.1_metadata.csv`. The table `P4_catalog_v1.1_tile_coords_final.csv` represents how the larger images are broken down into image tiles. Finally `P4_catalog_v1.1_tile_coords_final.csv` contain the marking identification numbers of each feature classification along with the vote percentage or ratio of if humans voted for an object being a fan or not. A classification is considered "true" is more than 50% of the human scientists mark the item as a specific feature. They are available for open source download on their website and in their database, with the ESP identification numbers highlighted in both the metadata and fan identification files. Using these files, I am able to draw boxes around each marking. This is shown in a sample tile Fig 1. Each tile is uniquely identifiable with many classifications per tile, representing multiple fans per image. In supervised machine learning, the data is split into training and testing data. The training data is used to generate reference samples for a model while the testing data is used to evaluate the models performance. Doing this enables a model to perform well on new, unseen data. For each tile, random boxes are generated within the image, these random boxes, both random in size and position are known as "notfans". The user-marked features are labeled "fans". Together, they create ground truth and false categories. In my model, I aim to predict if the feature is a fan given a random box. The train/test split, highlighted in Fig 2, consists of 1400 train and 1400 test photos. I skew the dataset towards "notfan" classifications with the aim of eliminating false positives. With the data evenly split between training and testing, I preprocess the data in order to denoise and improve train time.

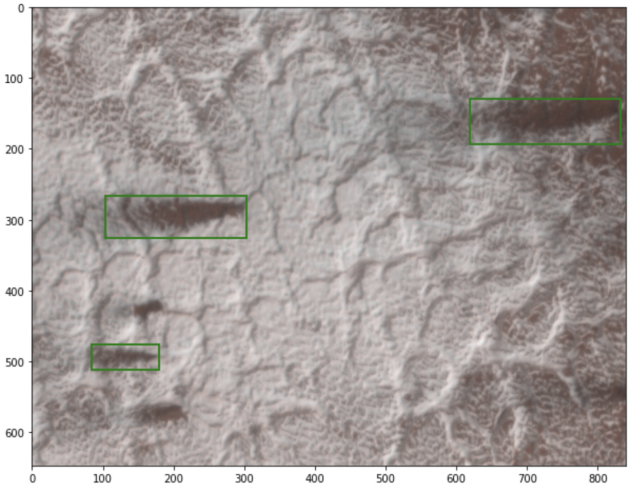

Figure 1. Sample image of a single "tile" in the data taken from a larger image with multiple tiles. The image highlights topographical features on the Martian surface. Each green box represents a marked item on the surface with a specified classification id. A box is considered "marked" if more than 50% of humans classify it as a fan.

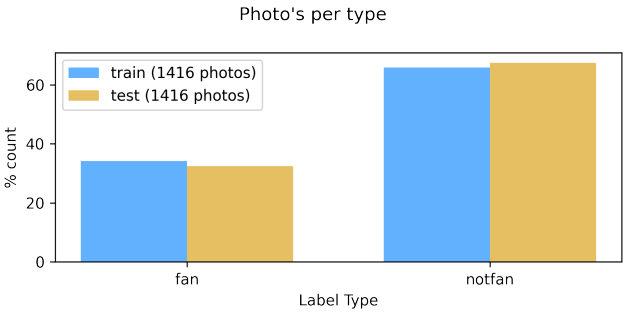

Figure 2. Representation split of the dataset into training and testing data. The size of the data was reduced due to limitations in training time.

3 PREPROCESSING

Preprocessing images involve using a set of methods to convert the image into a standardized format. It also includes feature extraction, which identifies important characteristics from the images. The goal is to improve the performance of the model. In this section, I will outline the steps taken achieve this goal.

3.1 Image Augmentation and Gamma Correction

I begin the analysis pipeline with data augmentation. I use this to increase the size and diversity of the training data. As our dataset is limited, data augmentation may help prevent over-fitting and improve the general performance of the model. I do this by translating, scaling and flipping "fans" to create a larger data space with new

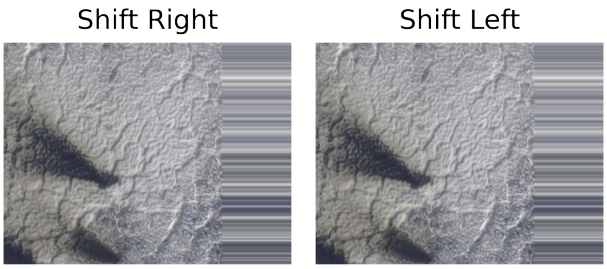

Figure 3. An example of a typical data augmentation process. In this case, both images are taken from a single reference image. The outputs contains multiple images with translated, flipped and inverted version of the reference image.

augmented versions of the fan images. A sample of the data augmentation pipeline is highlighted in Fig 3 in which a same image is translated. In order to further improve the preprocessing pipeline I apply gamma correction to the images used in the training set. Doing this allows me to improve the brightness and contrast of the images. The equation for the correction is given by the following:

$$I_c = I_o^{\frac{1}{\gamma}} \tag{1}$$

Here, I_c is the corrected fan image, I_o is the original fan image. This non-linear equation maps the pixel values in the original image to new values in the corrected image according to a power function, with the gamma value determining the exponent of the function. The gamma value controls the amount of correction applied, with higher values resulting in brighter fans and lower values resulting in darker fans, in our case $\gamma = 2$. An example of this process is demonstrated in Fig 4.

3.2 Principal Component Analysis

After preprocessing the data, I reduce its size to improve the efficiency of our machine learning model. I explored using general data reduction techniques such as principal component analysis (PCA) and histogram oriented gradients (HOG). PCA is a mathematical technique in linear algebra that offers versatility and flexibility, while HOG is a feature extraction method. It divides an image into cells that generate gradients of orientations for each cell. These gradients are combined into a feature vector which represents the entire image. I ultimately choose to use PCA for its ability to denoise images and its superior predictive performance. I will discuss this technique in more detail below.

When applying PCA I use the single vector decomposition (SVD) method, a mathematical method to reduce the original matrix into a set of diagonal and unitary matrices. I then make an orthogonal projection of the components onto a diagonal matrix order by the variance. The diagonal elements of the decomposition are known as the principal components. I apply PCA on the x-y image domain. (Mudrova & Procházka 2005)

$$\mathbf{I}(x, y) = \mathbf{V}\mathbf{\Sigma}\mathbf{U}^\top \tag{2}$$

Here $\mathbf{I}$ is the image. $\mathbf{V}$ is a unitary matrix and is known as the left singular matrix. I use $\mathbf{V}$ to project vectors from the image to feature (component) basis. $\mathbf{\Sigma}$ is a diagonal matrix ordered by decreasing

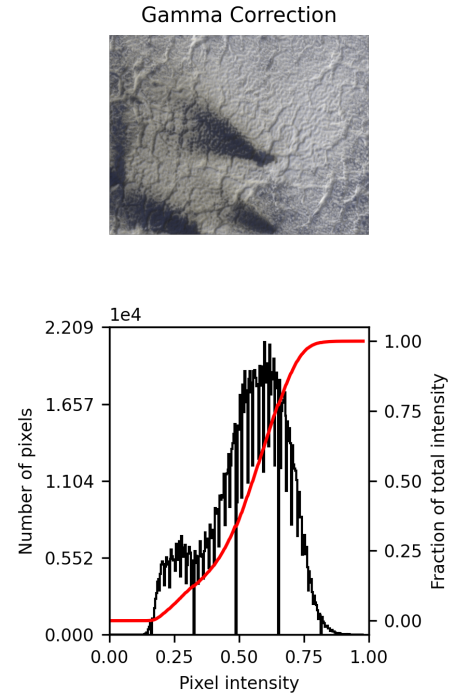

Figure 4. Image correction applied to the reference image in order to simplify further preprocessing. The x-axis represents pixel intensity and the y-axis represents count data. The red line represents the corrective fix applied to the image.

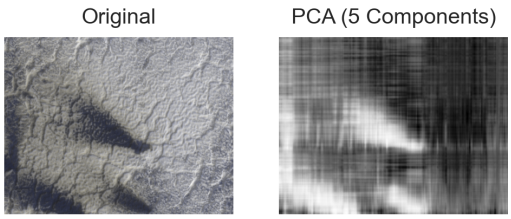

Figure 5. Original (left) vs reconstructed (right) image with the application of principal component analysis (PCA). The reconstructed image contains most of the features surrounding areas of interest while greatly reducing the size of the data. Some ringing effects are observed in the reconstructed image

magnitude (i.e. $\sigma_1 >> \sigma_2 >> ... >> \sigma_n$). Here σ_i is known as a singular value which holds information about the data vector. $\mathbf{U}$ is also a unitary matrix. It is known as the right singular matrix. Using this I form the feature vector. I do this by projecting the image vector $\mathbf{i_i}$ onto the change of basis vector $\mathbf{V}$ to get the principle components of the input data. The feature vector is also known as the principal components of the transformation. An example of the reconstructed image is shown in Fig 5. The reconstructed images reduce noise and make the fan images easier to train on. PCA has been known to cause ringing effects which on how the features and centered and their relative sizes. I alleviate this problem through the use of data augmentation. This will allow for a more general model predicting on a diverse range of data.

4 MODEL SELECTION

When selecting a model, factors such as size, complexity and data quality must be taken into consideration. I study the impact of model choice through k-fold cross validation. This occurs by diving the dataset into k subsets (folds). The model is then trained ($k = 10$) times using a unique data subset for validation set each time. The model is training on the remaining subsets. Comparisons between each model are then made. I also use multi-threading to speed up the compute time. The final performance metrics are calculated using an average of the model performance over each ($k = 10$) subset. I use this to tune the model hyperparameters and aid my decision in model selection. I restrict my search to mainly supervised learning methods (Cady 2017). In an attempt to keep the language accessible to both an astronomy and statistics audience, I relate the parameters back to the data in an astrophysical context.

4.1 Model: Support Vector Machines

A support vector machine (SVM) is a classification algorithm that splits the training data over a decision boundary. This boundary maximally separates the two classes, fans and not fans. It is represented mathematically in the following form:

$$\min_{w,b} \frac{1}{2} w^T w + C \sum_{i=1}^{n} \max\left(0, 1 - y_i \left(w^T \phi(x_i) + b\right)\right) \tag{3}$$

The decision boundary is represented as a hyperplane of PCA components with w being the weight vector normal to the hyperplane. Here, b is the bias term and C is the penalty intensity parameter. C controls the trade-off between a large classification penalty and no classification penalty. The former represents data over-fitting while the latter represents under-fitting. The equation is solved as a quadratic optimization problem with linear constrains. It can be solved using the Lagrange multiplier method. Once the decision model has been trained, it can make new predictions by computing the value of the decision function $f(x) = w^T x + b$ for each new image. By doing this, the model assigns a fan or notfan class to the new image. Since I am solving a binary classification problem. Images with $f(x) > 0$ are classified as one class while images with $f(x) < 0$ are classified as the opposite class. This algorithm has a run-time complexity of $O(n^3)$, it takes very long to run with scaling data. Lower values of C have been shown to reduce this. Relating this back to the data, we expect the decision boundary to split features that uniquely characterize fans and notfans, the features are the PCA components that characterise the most important features of each image (Hearst et al. 1998).

4.2 Model: k-Nearest-Neighbours

k-Nearest-Neighbours (kNN) is a simple classification algorithm. New images are predicted based on the classification of k nearby labeled images. The algorithm finds k examples in the training set which are the nearest (have the closest PCA vector) to the new example. When computing the distance, Euclidien distance (L2 norm) is used. Formally:

$$y^* = \max_{t^{(z)} \in \text{ class labels}} \sum_{i=1}^{k} \mathbb{I}\left(t^{(z)} = t^{(i)}\right) \tag{4}$$

In which I have found k examples $\left\{\mathbf{x}^{(i)}, t^{(i)}\right\}$ nearest to the test instance $\mathbf{x}$. The KNN algorithm is simple and easy to implement, it requires no training phase (i.e. is used directly on the dataset). However, KNN runs into some computational complexity problems. The requirement in which it must compute the distance between a new example and all examples in the training set makes it struggle with high-dimensional data. The distance between each data-point approaches infinity as the data dimension, or number of PCA components (dataset size) is increased. This is formally known as the curse of dimensionality problem (CSC311 2022). In the context of the fan/notfan data, consider the dataset to have 10 images, 5 of which are fans and the other 5 are notfans. These images are transformed into feature space using PCA. A test image is then transformed into feature space. Now support $k = 3$, the algorithm will identify 3 training images with the smallest difference in features to the test image. The class prediction is then computer as the average of these 3 images.

4.3 Model: Logistic Regression

Logistic regression (LR) is an algorithm which is typically used for binary classification tasks, when the data is not linearly separable. The model learns a set of parameters that define the logistic function. A function which splits the fan/notfan classes in binary classification. Once trained the function maps the PCA input space to the classification output space using a monotonically increasing function. The logistic function is defined as the following.

$$f(x) = \frac{1}{1 + e^{-k(x - x_0)}} \tag{5}$$

Here, x_o is the midpoint value and k is the logistic growth rate. The logistic function outputs a probability between 0 and 1 that the test image belongs to a certain class. I set the classification threshold to 0.5. The model is optimized by implementing a penalty between probability of a certain prediction and the ground truth value of that image. This is written formally below:

$$\begin{cases} -\ln p_k & \text{if } y_k = 1 \\ -\ln(1 - p_k) & \text{if } y_k = 0 \end{cases} \tag{6}$$

This equation can be turned into a loss function by combining the terms to obtain the *cross-entropy*.

$$\mathcal{L}_{\text{CE}}(p_k, y) = -y \log p_k - (1 - y) \log(1 - p_k) \tag{7}$$

Then, I maximizing the inverse log-likelihood through gradient descent. This can be considered equivalent to minimizing the penalty term written above. The benefit to using LR is that the threshold may be changed above/below 0.5 in order to eliminate false positives/negatives. Once again, consider the same example as before in which there are 10 train images, split evenly, and a test image. The logistic regression model with use the features (PCA components) of the test image to make a prediction about its class, fan/notfan based on the training data. It will do this be computing a probability that the image belongs to each class, using the logistic function to classify and gradient descent to optimize its weights. The prediction class is selected as the class with higher probability (CSC311 2022).

4.4 Model: Naïve Bayes

Naïve Bayes (NB) is a probabilistic classifier that applies Bayes theorem with a strong assumption that the feature vectors are independent. Only a small amount of data is required for parameter classification. The conditional independence assumption allows us

to rewrite a version of Bayes theorem.

$$\hat{y} = \arg\max_{y} P(y) \prod_{i=1}^{n} P\left(x_i \mid y\right) \tag{8}$$

Here, y, is the class fan or notfan, and x is the PCA vector space. The probability $P(y)$ is given by the training ground truth.

$$P\left(x_i \mid y\right) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{\left(x_i - \mu_y\right)^2}{2\sigma_y^2}\right) \tag{9}$$

My version implements Gaussian Naïve Bayes which uses a Gaussian likelihood $P\left(x_i \mid y\right)$ in its approach. Here σ_y and μ_y are used in estimating the maximum likelihood. In this approach, I exploit the fact that NB is a quick yet simple classifier.

Referring back to the 10 images example. Suppose once again, I would like to predict a test image. Once transforming it to the PCA feature space, the NB classifier will assume features of each PCA component are distributed according to a Gaussian distribution. The probability of the test images features given these distributions are used to determine the class of the image. More explicitly. If I give the NB classifier a fan image, it will search its feature space of known fan images, and rank the probability that the test feature space matches the training data, through a Gaussian distribution. The degree of match between the Gaussian distribution of fan's PCA components and the test PCA component will determing the probability of the test image being classified as fan (Kaur & Oberai 2014).

4.5 Model: Gaussian Processes

A Gaussian process (GP) is a stochastic method used for non-parametric regression and classification. Since it does not make any assumptions about the underlying data distribution, it is robust when trained on smaller data sets. It is able to model the relationship between the input, PCA components and output, classification fan or notfan using a Gaussian distribution. This continuous distribution is defined by its mean and covariance. The GP model uses a mean function to model the conditional mean of the fan/notfan classification given the PCA components. In random variable notation, it has the following form:

$$y(x) \sim GP(X, K(X, X')) \tag{10}$$

Here X is the mean function and $K(X, X')$ is the covariance kernel. The key predictive equations are shown below.

$$\bar{\mathbf{f}}_* \triangleq \mathbb{E}\left[\mathbf{f}_* \mid X, \mathbf{y}, X_*\right] = K\left(X_*, X\right)\left[K(X, X) + \sigma_n^2 I\right]^{-1}\mathbf{y} \tag{11}$$

$$\operatorname{cov}\left(\mathbf{f}_*\right) = K\left(X_*, X_*\right) - K\left(X_*, X\right)\left[K(X, X) + \sigma_n^2 I\right]^{-1} K\left(X, X_*\right) \tag{12}$$

The former equation represents the functional form of the mean maximum a posteriori (MAP) while the latter equation represents the optimal covariance. Essentially, they are the mathematical versions of the optimized Gaussian's over the training data. In my case, the functional forms of how to fit the principal components of a new test image along the the error or covariance associated with any deviations from the mean. Together they are used to make new classifications on new datasets (PCA components). As with logistic regression, I am able to tune the threshold probability to prevent false positives from arising. A benefit to GP classification is that I am able to propagate the errors through the training set. One drawback to GP classification is the $O(n^3)$ run time. Similar to support vector machines, the inversion of the covariance matrix results in a large computational cost added to it. Its popularity in astronomy and planetary science has made it a useful candidate to investigate (Rasmussen 2003).

4.6 Model: Linear Discriminant Analysis (and Quadratic Discriminant Analysis)

Both Linear Discriminant Analysis (LDA) and Quadratic Discriminant Analysis (QDA) are classification algorithms which attempt to split the data into the fan/notfan classes. LDA does this by finding the best linear combination which maximize the separation between fan and notfan. This is done by optimizing the direction in which the data varies the most and projecting the data onto that direction. QDA is similar to LDA, but instead of linear boundaries between the fan/notfan class, quadratic boundaries are allowed. This enables QDA to find more complex patterns in the data. As a result, QDA may also result in data overfitting.

I am explicitly ommiting the math of LDA simply because it does not provide much insight into the properties of how the optimization takes place. Simply put, LDA assumes the class conditional distributions of the PCA components are Gaussian and have equal covariance. LDA then finds the linear combination that maximally separates the different classes from each other. It does this by maximizing the variance between classes and minimizing the variance within a class. Referring back to the astrophysics at hand, LDA will attempt to maximize the variance between fan and notfan while concurrently minimize the variance within the fan class and within the notfan class separately. QDA, similar to LDA will do the same. Except, QDA does not assume the covariance matricies of different classes are equal, it does this separately and uniquely.

While LDA is generally preferred in the ML community due to its efficiency and non-overfitting robustness. QDA has the ability to choose priors. This makes it more robust with followup surveys. I choose to apply both LDA and QDA. Our feature space is small (only a few PCA components are used) compared to the large number of data points, as a result, over-fitting should not occur.

4.7 Model: Stochastic Gradient Descent (Unsupervised)

As an extension to the supervised methods used, Stochastic Gradient Descent (SGD) is an unsupervised algorithm designed to find the minimum value of a function. It is an iterative algorithm that starts with an initial guess, and takes small steps in the direction of the steepest negative gradient. The size of these steps are known as the learning rate and the determination of what the value should be is called the loss function. The minimization of this loss function provides the classification of the fan/notfan. In our case, I implement a hinge loss function.

$$\ell(y) = \max(0, 1 - t \cdot y) \tag{13}$$

In this equation, t represents the class similarity ans is usually ± 1, y represents the classifier score. I choose this due to its similarity to SVMs which are well understood in the literature. I expect the SGD to be a quicker version of the SVM classifier. While slightly outside the scope of this study, it is useful to study the SGDs performance in relation to that of the supervised methods listed above. As a result, its discussion will be kept brief.

5 RESULTS

The results of the model training is presented in this section. I tested different models with varying hyperparameters for each proposed model. As an exploratory study, I prioritized understanding the pipeline, and proving the ability to use different models over the direct model performance.

The first method used was logistic regression, in this model I applied the L2 penalty with an saga optimizer. I was able to achieve a test accuracy of 69.4%. The model took 12 seconds to train. The version I applied was regularized logistic regression.

Next I applied Gaussian Process classification. In this classifier, I used an RBF covariance matrix (Rasmussen 2003) with a length-scale of 1, the Laplace approximation was used to approximate a non-Gaussian posterior into a functional normal distribution. The test dataset resulted in an accuracy of 60.5%. The inversion of the covariance kernel resulted in a long training time of 168 seconds.

The next model used was Support Vector Machines. I applied the linear kernel version due to its convex quadratic optimization, I set the penalty term $C = 1$. The model performance resulted in an accuracy of 66.9% with a run time of 58 seconds.

Both Linear Discriminant Analysis and Quadratic Discriminant Analysis were applied with similar success, the former had a test accuracy of 62.2% while the latter had an accuracy of 65.8%. Both took approximately 6 seconds to complete training. LDA used singular value decomposition (SVD) to propagate the vectors into the diagonal space. There were no priors used in QDA, although this may be a possible extension to consider. As expected, QDA performed better than LDA.

The KNN classifier was set-up using 5 neighbours with a uniform weight on each neighbour. The distance measured was Euclidean (L2 norm). The resulting test accuracy was 65.7% with a train time of 5 seconds. This model returned the highest variability in accuracy between k-folds of training.

Stochastic Gradient Descent was the only non-supervised approach used in training. As mentioned before, a hinge-loss model was used to imitate the performance of a SVM model. A maximum of 1000 iterations of gradient descant were set with a stopping criteria tolerance of 0.01 between steps. The model trained in less than a second with a predictive performance of 65.9%. Similar to KNN, this model had relatively high variability between the k-folds of training.

Naive Bayes was the final process used. Similar to QDA, it is a Bayesian classification method. Our model performed efficiently training is less than a second and providing the best accuracy of 71.8%. Gaussian Naive Bayes was used with small variational smoothing = `1e-9`.

The model was trained in python using mainly the `sklearn` package. Some dependencies occur in C and FORTRAN. The use of multi-threading played a large role in speeding up both the data wrangling and model training. This occurred through `joblib`. The plots were generated using `matplotlib` and `seaborn`.

A tabular representation of the results is shown in Table 1, with its corresponding figure comparison in Fig 8. A sample image of the best Gaussian NB model performance can be seen in Fig 6. Finally, the confusion matrix representing the Gaussian NB performance for recall, precision and accuracy is shown in Fig 7. The model performs well at predicting the True Positive, and True Negative. However, it is also skewed towards over-predicting false positives. I.e. predicting that an image is a fan when a human has classified it as not a fan.

Table 1. Test accuracy's of each model performance averaged over k-folds ($k = 10$) of cross-validation. The best performing model was the Gaussian Naive Bayes (NB) with an accuracy of 72% and runtime of under 1s. The worse performing model, in both time and accuracy complexity, was the Gaussian Process Classifier (GP) with an accuracy of 61% and runtime of just under 3 minutes. All the remaining models has approximately similar performances. For reference, LR (Logistic Regression), SVM-l (Support Vector Machine - Linear), LDA/QDA (Linear/Quadratic Discriminant Analysis), KNN (K-Nearest Neighbours) and SGD (Stochastic Gradient Descent)

Model	Test Accuracy (out of 1)	Runtime (seconds)
LR	0.694	12s
GP	0.605	168s
SVM-l	0.669	58s
LDA	0.622	7s
QDA	0.658	6s
KNN	0.657	5s
SGD	0.659	1s
NB	0.718	1s

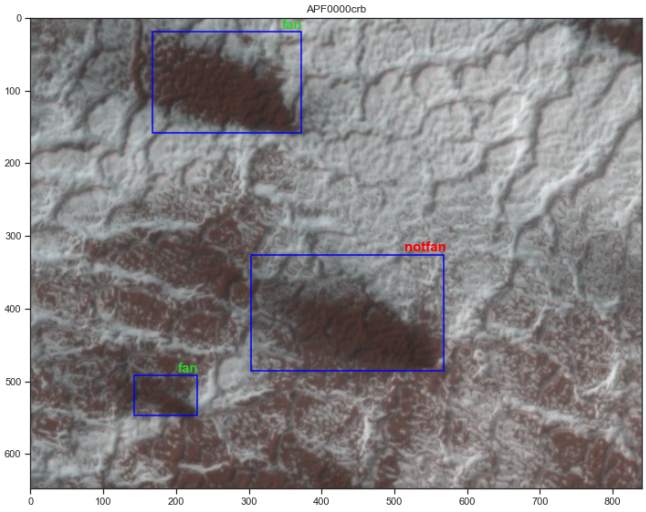

Figure 6. Sample tile with over-plotted results containing model predictions of the trained model. Green text corresponds to correctly predicted features, while red text corresponds to incorrectly predicted features.

6 DISCUSSION

I aim to explore some of the key issues that arose in my analysis. I will begin the discussion with the challenges and limitations of our implemented models. I will then move onto some changes that I recommend. Finally, I will highlight possibilities for future work.

Beginning with first an overall discussion of the model per-

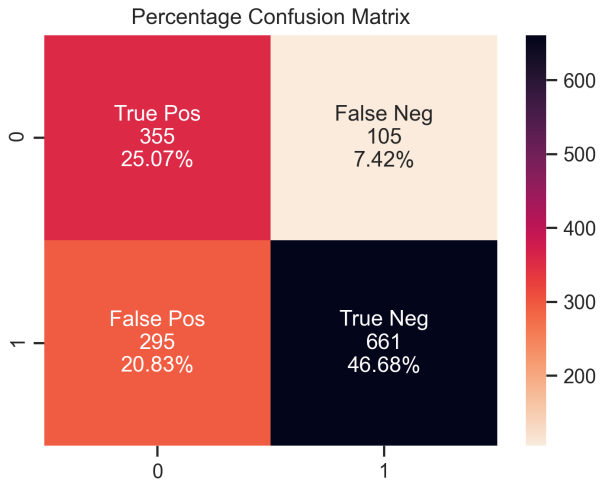

Figure 7. Confusion matrix showing a visual representation of the classification algorithm. The four entries of the matrix, in both percentage and count represent the True Positive (TP), False Negative (FN), False Positive (FP) and True Negative (TN) values. A False Positive corresponds to the model predicting a fan while the human classification is that the feature is not a fan. A False Negative corresponds to the model predicting that a feature is not a fan while the human classification is fan. While the model performs well in predicting true positives and true negatives, it has a skew towards predicting false positives.

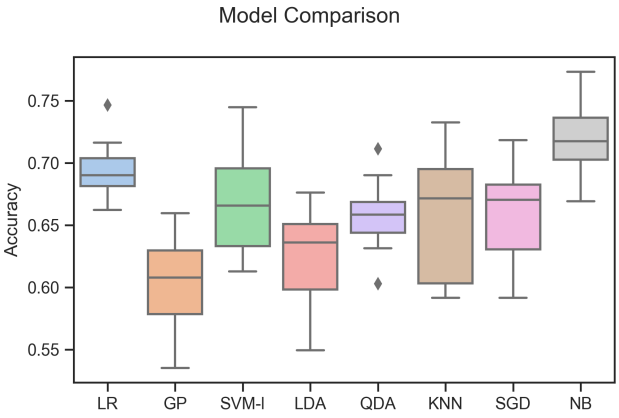

Figure 8. Pictorial representation of the tabular results highlighted in Table 1. As before, Naive Bayes was the best performing model with Gaussian Process being the worse performing model. While the rest of the models had similar performance, their spread greatly varies. Quadratic Discriminant Analysis and Logistic Regression both had small spread in their test accuracy's. Models such as K-Nearest Neighbours and Support Vector Machines, had a larger spread in its accuracy.

formance. The results of the study clearly indicate that Gaussian naive Bayes performed the best among the models tested, Gaussian processes performed the worst. This can be attributed to the benefits of using naive Bayes on a small dataset. It is known to perform well with a limited feature space. This was the case since only 5 PCA components were used. With more data, it is expected that the other models would eventually outperform NB.

GP and LDA were the worst performing models. The poor performance of the GP was likely due to the improper weighting of data. GP is not sparse and thus gives equal weight to every PCA component. This leads to sub-optimal results since the first and more important PCA component is not given sufficient importance. In the case of LDA, the poor performance could have been caused by a lack of linear separability over the feature space. In addition to this, a normal distribution over the PCA components is assumed which may not likely be the case.

Most of the models displayed Gaussian behaviour through their trails with an even spread with over the k-folds of cross-validation. As a result, I am, to some degree, able to interpret the Gaussianity of the models. In order to make this interpretation mode reliable, the model would need to be validated over a much larger dataset and over a longer period of time.

Referring to the the difference in performance of QDA vs LDA. QDA is a generalization of the popular LDA classification algorithm. It was found in Fig 8 that QDA outperformed LDA, as expected. This was partly due to the nature of the data, but was also due to the overfitting bais of QDA. QDA modeling the covariance of each fan/notfan class separately resulted in it better capturing the complexity of each class, thus better predictive performance. On the other hand, LDA assumes identical covariance between each class resulting in a limitation which is not necessarily true. Trivially, random boxes will have different covariance to fans. Overall, the results indicate that QDA is marginally more effective than LDA. This will further improve with the addition of class specific priors. A possible extension for future study.

Referring to Table 1, it can be observed that SGD was surprisingly outperformed by SVM. This was unexpected, especially since they used analogous loss-optimization functions. Upon further examination, I conclude that the use of strict learning rates, a small number of iterations and early stopping may have resulting in its poor performance. Additionally, the PCA components used may have been sub-optimal for gradient descent learning. Overall, our results demonstrate the importance of carefully choosing the SGD hyperparameters as well as considering the training time and number of PCA components.

The choice of PCA components varies the complexity of the dataset being used. Increasing the number of components greatly changes the model performance, and reduces the chance of PCA causing the artifacts to ring. However, training the model on more features results in including unwanted noise. After testing multiple models between 5-200 PCA components, I concluded that using as few as possible (5 components) resulted in the best performance. In doing so, I was able to maintain most of the images key features while denoising the feature space. An example of the PCA is shown in Fig 5. Ultimately, the choice of PCA components is fluid and should be studied concurrently with model/hyper-parameter variations.

PCA, however, is not the only method of dimensional reduction I explored. The HOG (Histogram Oriented Gradients) transform is widely used in computer vision when extracting features from fan images. However, when I attempted to use this transform in our pipeline for modeling training, I found that it its performance was not a great improvement over PCA. This could be due to a variety of reasons, such as the quality of the input fan image, the nature of the

fans themselves or the hyperparameters used in the HOG transform. Experimenting with this is a possible avenue of further study.

Another approach that was considered was using edge detection with a Fourier transform to extract the fan structure from the image. By doing this, I expect a peak in Fourier space around the edge of the fan structure. The difficulty in this method lies in the conversion of the pixel values of the edge to vector values. As a result, the preprocessing pipeline would need to be supplemented with the HOG transform before applying the Fourier transform. While this approach does increase the preprocessing complexity, it could in theory provide a large performance improvement.

The results in the confusion matrix in Fig 7 clearly indicate a model skew towards classifying items that are not fans as fans (i.e. in a false positive manner). This is particularly harmful when trying to predict the Martian weather patterns. A false classification would result in an incorrect weather vector, deteriorating the weather predictive precision. It is possible to alleviate this issue by adjusting the some of the models threshold for predicting positive outcomes. For example, one may increase the threshold from 0.5 to 0.8 in both LR and GP, or use a different functional decision boundary in SVM. This would encourage the model to skew towards false negatives which are much preferred over false positives. Additionally, using different NB classifiers may further improve the predictive performance.

The high false positive rate also demonstrates the need to increase the size of the fan dataset. While I have already translated and flipped the images, simple methods like these have their limitations. One could consider using more advanced models such as creating new images using generative models or synthesizing a combination of images using computer vision. Additionally, one may consider using different data augmentation techniques separately for the fan and notfan classes. As they have inherently different data composition, they may benefit from separate treatment. Overall, the extension of different data augmentation methods should further improve the false positive rate along with making the model more robust.

Commonly used in both industry and academia, one may consider using a convolution neural network (CNN) to classify fans instead of supervised methods. A potential advantage to this is that CNN's are able to extract and combine feature vectors in a hierarchical manner. This is particularly useful in image classification. In addition to this CNN's are transitional invariant and train well under increasing large amounts of data. Combined with better data augmentation this would greatly improve predictive performance and is an avenue that should be seriously considered.

This work, once streamlined, will be able to help predict Martian weather patterns. This work can be expanded to include other data sources such as rover observations, and spectroscopic samples. In doing so we may be able predict when and where the ice may rupture, thus know where to point the satellites to measure cleaner data. Furthermore, the results of my work can inform decision making about possible landing sites. Overall, this work will improve our understanding of the polar ice sublimation and how to plan future missions around it.

7 CONCLUSION

The use of machine learning, both supervised and unsupervised has the potential to greatly improve our understanding of Martian topographical features. By studying them we will be able to better understand the Martian surface. I implement a suite of machine learning models, attempting to detect and characterize these Martian surface features. In doing so, I implement a preprocessing pipeline that implements data augmentation, gamma correction and principal component analysis. I then train on multiple different machine learning models, mainly supervised (both Bayesian and frequentist) and compare their performance. The Gaussian Naive Bayes classifier produced the best accuracy and time efficiency metrics making 71.8% of its predictions correctly after training in under 1 second. Upon further analysis, I determined the main are of concern being the high false positive rate (FP = 20.83%). I discuss possible ways of mitigating this, such as higher thresholds for our models and the implementation of mode robust preprocessing algorithms. I then highlight extensions to this work with unsupervised methods such as convolution neural networks. This project has demonstrates the feasibility of using supervised learning methods to identify Martian surface features. Further research in this area has the potential to yield valuable insights into the polar ice patterns as well as aid future missions.

ACKNOWLEDGEMENTS

I acknowledge Professor Joshua Speagle for his useful discussion and valuable insight into statistical methods and Alexander Laroche for his discussion about Fourier transforms during my presentation. I acknowledge the NASA Mars Orbiter (MRO) and its HiRISE instrument with which the photos were taken and the Planet Four Collaboration which processed the data which I used. Finally I wish to acknowledge this land on which the University of Toronto operates. For thousands of years it has been the traditional land of the Huron-Wendat, the Seneca, and the Mississaugas of the Credit. Today, this meeting place is still the home to many Indigenous people from across Turtle Island and we are grateful to have the opportunity to work on this land.

DATA AVAILABILITY

The data is part of the Planet Four Collaboration and is open source. The specific training images are available to download from the results page which contain both the fan catalogue and the observational metadata. More details about the files can be found on this information link.

REFERENCES

Aye K.-M., et al., 2019, Icarus, 319, 558

Balme M., Mangold N., Baratoux D., Costard F., Gosselin M., Masson P., Pinet P., Neukum G., 2006, Journal of Geophysical Research: Planets, 111

Banerji M., et al., 2010, Monthly Notices of the Royal Astronomical Society, 406, 342

Bird R., et al., 2018, Muon Hunter: a Zooniverse project, doi:10.48550/ARXIV.1802.08907, https://arxiv.org/abs/1802.08907

CSC311 C., 2022, CSC 311 fall 2022: Introduction to Machine Learning, https://www.cs.toronto.edu/~rahulgk/courses/csc311_f22/index.html
Cady F., 2017, The data science handbook. John Wiley & Sons
Cohen B. A., Malespin C. A., Farley K. A., Martin P. E., Cho Y., Mahaffy P. R., 2019, Astrobiology, 19, 1303
David L., Howard R., 2016, Mars: Our Future on the Red Planet. National Geographic, https://books.google.ca/books?id=Dy0kDQAAQBAJ
Fisher J. A., et al., 2005, Journal of Geophysical Research: Planets, 110
Fortson L., Masters K., Nichol R., Edmondson E., Lintott C., Raddick J., Wallin J., 2012, Advances in machine learning and data mining for astronomy, 2012, 213
Hearst M. A., Dumais S. T., Osuna E., Platt J., Scholkopf B., 1998, IEEE Intelligent Systems and their applications, 13, 18
Kaufmann E., Hagermann A., 2017, Icarus, 282, 118
Kaur G., Oberai E. N., 2014, International Journal of Computer Science and Mobile Computing, 3, 864
Kieffer H. H., 2007, Journal of Geophysical Research: Planets, 112
Kieffer H. H., Christensen P. R., Titus T. N., 2006, Nature, 442, 793
Lintott C. J., et al., 2008, Monthly Notices of the Royal Astronomical Society, 389, 1179
Mali U., Rogers K. K.,
Maltagliati L., Titov D. V., Encrenaz T., Melchiorri R., Forget F., Keller H. U., Bibring J.-P., 2011, Icarus, 213, 480
McEwen A. S., et al., 2007, Journal of Geophysical Research: Planets, 112
Montmessin F., et al., 2017, Icarus, 297, 195
Mudrova M., Procházka A., 2005, in Proceedings of the MATLAB technical computing conference, Prague.
Nguyen T., Pankratius V., Eckman L., Seager S., 2018, Astronomy and computing, 23, 72
Pan L., Ehlmann B. L., Carter J., Ernst C. M., 2017, Journal of Geophysical Research: Planets, 122, 1824
Peng T., English J. E., Silva P., Davis D. R., Hayes W. B., 2018, Monthly Notices of the Royal Astronomical Society, 479, 5532
Raddick M. J., Bracey G., Gay P. L., Lintott C. J., Murray P., Schawinski K., Szalay A. S., Vandenberg J., 2009, arXiv preprint arXiv:0909.2925
Raddick M. J., et al., 2013, arXiv preprint arXiv:1303.6886
Rasmussen C. E., 2003, in Summer school on machine learning. pp 63–71
Schwamb M. E., et al., 2018, Icarus, 308, 148
Sharma M., Gupta A., Gupta S. K., Alsamhi S. H., Shvetsov A. V., 2021, Drones, 6, 4
Simpson R., Page K. R., De Roure D., 2014, in Proceedings of the 23rd international conference on world wide web. pp 1049–1054
Sprinks J., Houghton R., Bamford S., Morley J., 2019, Meteoritics & Planetary Science, 54, 1325
Zurek R. W., Smrekar S. E., 2007, Journal of Geophysical Research: Planets, 112

This paper has been typeset from a TEX/LATEX file prepared by the author.