Understanding the problem as a classification problem with labelled data is important as limits our model selection to classifications algorithms. This is in opposition to unsupervised clustering-based algorithms, dimensionality reduction, or regression models (although regression models provide a probability and can hence be separated into classes by a threshold or activation function). Since we are dealing with a classification problem, our model simplifies to one of the following: Linear SVC (support-vector model), stochastic gradient decent (SGD) classifiers, K-Nearest-Neighbors (KNN) classifier, non-linear SVC, and Naïve Bayes. In an ideal world choosing the best model is easiest when comparing their results after implementation. However due to time restrictions the problem was approached using Ockham's razor's principle. Since our dataset size was small I opted for Linear SVC as starting model since SGD's work better with larger sample sizes. After implementing linear SVC, I then implemented KNN as it was also simple to implement. KNN was chosen over Naïve Bayes after researching use cases of Naïve Bayes and shown to be mostly for text data. However, for future work comparisons of results with Naïve Bayes, non-linear SVC's, and ensemble classifiers would be ideal.

Training and testing on the same data will cause and overly complex model i.e. overfitting and hence will not generalize to new data. To combat this the model should be split into ratio of training data to test data such that the model can be trained and tested on different data. The ratio must be chosen such that there is enough training samples as well as having low variance to prevent overfitting, in this problem the data is split in a 80:20 ratio. For this dataset K-fold cross-validation was used to provide an even better estimate of out-of-sample performance, despite running "k" times slower than simply splitting the data in the 80:20 ratio. This method can be implemented since the sample size of the data set is relatively small and performance is not hindered.

The next step is to understand the model's evaluation metrics. For this implementation the following metrics were used:

- Classification accuracy – percentage of correct predictions
- Null accuracy – percentage of correct by always predicting the most frequent class

By comparing the classification accuracy to the null accuracy, we can see the underlying distribution of response values. A further metric that can be used that was not implemented in this solution would be to use a confusion matrix. For future improvements a confusion matrix can give a more complete picture of how the model is performing and hence help us choose a better classifier, i.e. by providing more evaluation metrics such as precision.

Linear SVMs can be explained by creating the widest distance between support vectors (i.e. datapoints on the margin created by a linear hyperplane). In the code provided these points are labelled as 0 if the image is good, 1 if the image contains a lens flare and 2 if the image is blurry. The samples are generated by calculating the histogram-oriented gradients of a scaled down, greyscale version of the image (for faster processing). Similarly, in the code KNN also uses these methods for labelling the classes as well as generating the samples. The KNN algorithm finds k closest data points given a new sample. For future work iterating through k to find the ideal k would be ideal, with consideration of over/under fitting. However, since this was not implemented, a k value of 4 was used since 3 is a multiple of the number of 3 classes we can avoid ties. The main drawback of KNN is the complexity in searching the nearest neighbors for each sample.

Comparing the two models, SVM won by the metric comparisons. Image of results has been provided.