Virtual Traveling

Jannik Sheikh, 3873496
/graded

Submission by 15th March 2021

1 Introduction

For the first part of the final project of the course «Introduction to Machine Learning»we had to classify given red, green and blue images, with the use of various machine learning algorithms, to their corresponding labels. Since images can have multiple labels we can categorize this machine learning task as supervised learning, multi-label classification. The used models are Random Forest Classifier (RFC), Multi-label K-Nearest neighbor (ML-kNN) as well as Logistic Regression (LogReg) and Multinomial Naive Bayes (MNB) with the use of MultiOutputClassifer (MOC). The F-score was chosen as an evaluation metric. Furthermore different preproccessing techniques as well as cross validation for model selection and hyperparameter tuning has been used.

2 Data Set

The data set used for this project holds information on 1186 landscape images and their corresponding labels. The following label names are **Snow/Ice**, **Mountains/Rocks**, **Plants/Forest**, **Stars** and **Sandy Desert**. In the following we refer to those labels by only using the first name mentioned. Figure 1, 2, 3, 4 and 5 are examples for the mentioned labels in the given order. As mentioned above, images also can have multiple labels, like **Snow/Ice** and **Mountains/Rocks**, see Figure 6. Table 1 shows how many images in total were assigned to the labels mentioned. Furthermore Table 9 shows a detailed enumeration of how the 1186 images were labeled. Based on the referred tables a class imbalance can be observed. For about 52.01% of the images **Mountains** is part of the label. In comparison the label **Stars** only occurce about 3.75%.

Label	Occurrence	%
Snow	248	13.27
Mountains	972	52.01
Plants	371	19.85
Stars	70	3.75
Sandy Desert	208	11.13
Sum	1869	100[1]

Table 1: Occurrence of each label

3 Models

In this section we give a short introduction to the four models we trained.

3.0.1 Random Forest Classifier

A random forest is an ensemble method of multiple decision trees, that are estimated with bootstrap aggregation, also called Bagging. Here, each tree in a random forest learns from a random sample of the data as well as considering only a subset of the given features for splitting each node in each decision tree [1]. Like normal decision trees, a random forest can overfit the data and we have to carefully select our hyperparameters when

[1] Percentages may not total 100 due to rounding

optimizing the model. A big advantage of the RFC is that we can specify the parameter *class_ weight* to *balance* which then uses the values of y to automatically adjust weights inversely proportional to class frequencies in the input data [2].

3.0.2 ML-kNN

The ML-kNN is an adaption of the kNN classifier for multi-label classification. It uses k-NearestNeighbors to find the nearest examples to a test class and then uses Bayesian inference to select assigned labels [3]. Here, Bayes'theorem is used to update the probability for a hypothesis as more evidence or information becomes available [4]. Since kNN is not influenced in any way by the size of the class, it will not favor any majority classes.

3.0.3 Logistic Regression

Logistic Regression uses the sigmoid function to predict the output

$$h = \frac{1}{(1 + exp^{(-z)})},$$

where z is the input feature multiplied by a randomly initialized value denoted as θ [5]. Through the cost function we can measure how close or far our predicition is from the original output. Smaller values imply a better fit. θ will be updated through gradient descent in each iteration by

$$\theta = \theta - \alpha \sum (h^i - y^i) * X_j^i,$$

where y is the original output label and h is the predicted output. Using the Logistic Regression model from *sklearn* we have the option to choose from different penalties for regularization as well as different solvers. For our implementation we choose the **elasticnet** penalty, with the **saga** solver. The saga solver is a variant of **sag** solver which uses Stochastic Average Gradient descent. Saga is also the only solver that supports the elasticnet penalty. Elasticnet regularization is a combination of the L1 and L2 regularization. It's minimizes the following cost function

$$\min_{w,c} \frac{1-p}{2} * w^T * w + p||w||_1 + C \sum_{i=1}^{n} log(exp(-y_i(x_i^T * w + c)) + 1),$$

where p controls the strengt of L1 regularization vs. L2 regularization. For further information we refer the reader to the *sklearn* documentation. Finally, it should be mentioned that in the Logistic Regression model we can also set *class_ weights* as in RFC. For more information please visit subsection 3.0.1.

3.0.4 Mulinomial Naive Bayes

Multinomial Naive Bayes classifier is a specific instance of a Naive Bayes classifier which uses a multinomial distribution for each of the features. Naive Bayes is based on the Bayes'theorem, with the assumption that the features in the dataset are mutually independent. The final decision rule is [6]

$$\hat{y} = \arg\max_k (lnP(C_k) + \sum_{i=1}^{n} ln \frac{N_{ki} + \alpha}{N_k + \alpha n}).$$

3.0.5 MultiOutputClassifier

Since Logsitic Regression and Multinomial Naive Bayes don't support multi-label classification, we used MultiOutputClassifer from *sklearn* which simply fits one classifier per target. This approach treats each label independently where as multi-label classifiers, like RFC or ML-kNN treat the multiple classes simultaneously, accounting for correlated behavior among them.

4 Evaluation

For the evaluation of our models we used 5-fold cross validation. As evaluation metric we chose the F1 score, also known as balanced F score. The F score can be interpreted as a weighted average of the precision and recall [7].

$$F1 = 2 * \frac{precision * recall}{precision + recall}.$$

Because we are dealing with a multi-label classification task, an average parameter has to be set. Our F1 score will be applied to each label independently and later gets averaged, depending on the chosen parameter. Since our data is imbalanced, we chose the *weighted* parameter, which calculate metrics for each label, and find their average weighted by support the number of true instances for each label. The formula for precision will be

$$\frac{1}{\sum_{l \in L} |\hat{y}_l|} \sum_{l \in L} |\hat{y}_l| P(y_l, \hat{y}_l),$$

and the formula for recall will be

$$\frac{1}{\sum_{l \in L} |\hat{y}_l|} \sum_{l \in L} |\hat{y}_l| R(y_l, \hat{y}_l),$$

where y will be the set of predicted (sample,label) pairs, $\hat{y}$ will be the set of true (sample, label) pairs, L the set of labels, y_l the subset of y with label l and $\hat{y}_l$ the subset of $\hat{y}$ with label l. Furthermore $P(A, B) := \frac{|A \cap B|}{|A|}$ and $R(A, B) := \frac{|A \cap B|}{|B|}$. All this information and more can be found here [8]. The best value for the F1 score is 1 and the worst score is 0.

5 Preprocessing and Feature Extraction

Since our data was already of type float64 and therefore between 0 and 1, further normalization or data type changes weren't necessary, since floats are faster and more precise when applying mathematics. The first step in our preprocessing pipeline was to resize every image to the shape of 128 to 128, which increases the computational speed for later techniques and still don't lose much information. Following we transformed our input image from the RGB color space to the HSV color space, which is just a alternative representation of the given image. This color space describes the colors (hue(H)) in terms of their shade (saturation(S)) and their brightness value(V) [9]. We chose this transformation, because of the significant color differences and color brightness between the images, depending on the labels. For example images labeled as **Snow** are much brighter than images labeled as **Plants**. Further the RGB components are correlated which can then, by analysing each channel separately, produce false colors. Subsequently we applied a non-linear denoising filter, called **Median filter**, on each HSV channel. Here the filter replaces each entry of the signal with the median of neighboring entries. The number of considered neighbours has to be predefined. Due to the fact, that we have different textures in the images, we then applied an edge detector, called **Sobel**, on each individual median filtered HSV channel. Edge detectors like Sobel are spatial filters, that perform some sort of convolution with the given image. Sobel applies two 3x3 Kernels, one for horizontal changes, and one for vertical, around a central pixel, for all pixels in the image. It is designed to approximate the gradient of an image by computing the sum of differences between the diagonally adjacent pixels. By that its highlight the edges. Afterwards we merged the three channels back together to one Median and Sobel filtered HSV image. Furthermore we used only the median filtered HSV image, after converting it back to an RGB image, as an input for our Gabor filter to generate even more features. Gabor filters can be used for texture analysis as well as edge detection. We decided to transform the image back to the RGB color space, because in [10], the authors concluded that the use of either color space is sufficient. Gabor filters are convolution filters representing a combination of a gaussian and a sinusoidal term. The gaussian part provides the weights and the sine component provides the direction. The Gabor kernel is a function of the pixel postion, x and y, the wavelength of the sine component λ, the orientation of the filter θ, Phase offset ϕ, standard deviation of the gaussian envelope σ and the spatial aspect ratio γ. By changing any paramter, we can create different features, for the same position.

6 Experiments

6.1 Set-Up

In the beginning, we splitted our data into training and testing sets. 30% were hold out for validation. We then applied the mentioned preprocessing steps in section 5, for each training and validation set. Afterwards we ended up with six features for our algorithms. The features including the original pixel values, the pixel values for the Median and Sobel filtered image, as well as four Gabor filters. It is to mention that more Gabor features could be produced and would probably lead to an improvement of the models. Due to the fact that we had limited computing power, we were limited in the number of features we could create. Subsequently we trained our four models with hyperparameter tuning, using RandomizedSearchCV with 5-fold cross validation. Depending on the given model, different parameter grids had to be passed. For more information about the

specific hyperparameters for each model, we refer the reader to the *sklearn* documentation of each model. For evaluation we used 5-fold cross validation to calculate our F1 score for each model. After that we used the best classifier of each model to predict on our validation set. We used 5-fold cross validation for evaluation.

6.2 Results

Table 2 shows the results of the chosen models on the training set, Table 3 represents the results on the validation set. The RFC performs best on the training set, followed by the LogReg with MOC. MNB with MOC is, like expected, due to the fact of being a simpler model, last with an F1 score of 0.643. On the evaluation set, LogReg with MOC performs best, in terms of the F1 score and achieves 0.828. Table 4, 5, 6, 7 and 8 show the confusion matrices for each individual label on the validation set.

Model	F1 Score
RFC	0.841
ML-kNN	0.788
MNB with MOC	0.643
LogReg with MOC	0.825

Table 2: Evaluation on Training Set

Model	F1 Score
RFC	0.814
ML-kNN	0.772
MNB with MOC	0.617
LogReg with MOC	0.828

Table 3: Evaluation on Test Set

Model	TP	FP	FN	TN
RFC	273	12	34	37
ML-kNN	221	64	26	45
MNB with MOC	166	119	22	49
LogReg with MOC	249	36	17	54

Table 4: Confusion Matrix for Snow

Model	TP	FP	FN	TN
RFC	37	31	12	276
ML-kNN	21	47	12	276
MNB with MOC	42	26	89	199
LogReg with MOC	36	32	30	258

Table 5: Confusion Matrix for Mountains

Model	TP	FP	FN	TN
RFC	211	34	36	75
ML-kNN	181	64	25	86
MNB with MOC	164	81	42	69
LogReg with MOC	217	28	24	87

Table 6: Confusion Matrix for Plants

Model	TP	FP	FN	TN
RFC	331	1	1	23
ML-kNN	331	1	1	23
MNB with MOC	271	61	6	18
LogReg with MOC	325	7	5	19

Table 7: Confusion Matrix for Stars

Model	TP	FP	FN	TN
RFC	293	2	28	33
ML-kNN	284	11	30	31
MNB with MOC	195	100	24	37
LogReg with MOC	286	9	11	50

Table 8: Confusion Matrix for Desert

7 Analysis and Discussion

Overall the results are sufficient. All our models fit the training and validation set pretty good, since we see no big difference between the training and test score. The performance increase of the LogReg is pretty small and could be explained by the fact that 32.16% of all occurred labels on the validation set are **plants** and **desert**. By analysing the confusion matrices, we can see that LogReg performed better than RFC when dealing with these labeled images. Our initial guess, that the RFC would perform better than any other model has been partly refuted. Due to the fact that we had limited computing power, an increase of the features, as well as a more detailed hyperparamter tuning and higher k for the k-fold cross validation could result in a bigger separation between the models. Nevertheless it should be mentioned that training time for LogReg with MOC and hyperparameter tuning with only a small grid, by changing only the Parameter C, takes incredibly

much longer than train a RFC and try different parameters for *n_ estimators*, *max_ depth*, *min_ samples_ leaf*, *min_ samples_ split* and *bootstrap: True or Talse*. ML-kNN trying different parameters for k and s, as well as hyperparameter tuning for the MNB with MOC by testing different *alpha* values and test *fit_ prior: True or False*, are much less time complex than LogReg with MOC, but a bit slower than RFC. For more informations about the specific hyperparametesrs we refer the reader to read the *sklearn* documentation about each model.

8 Appendix

Label	Assigned	%
Only Snow	10	0.84
Only Mountains	341	28.75
Only Plants	146	12.31
Only Stars	43	3.63
Only Sandy Desert	8	0.67
Snow and Mountains	196	16.53
Snow and Plants	5	0.42
Mountains and Plants	181	15.26
Mountains and Stars	19	1.60
Mountains and Desert	190	16.02
Plants and Desert	2	0.17
Snow, Mountains and Plants	37	3.12
Mountains, Stars and Desert	8	0.67
Sum	1186	100[2]

Table 9: Assigned Labels and Combinations

Figure 1: Image labeled only Snow

Figure 2: Image labeled only Mountains

[2]Percentages may not total 100 due to rounding

Figure 3: Image labeled only Plants

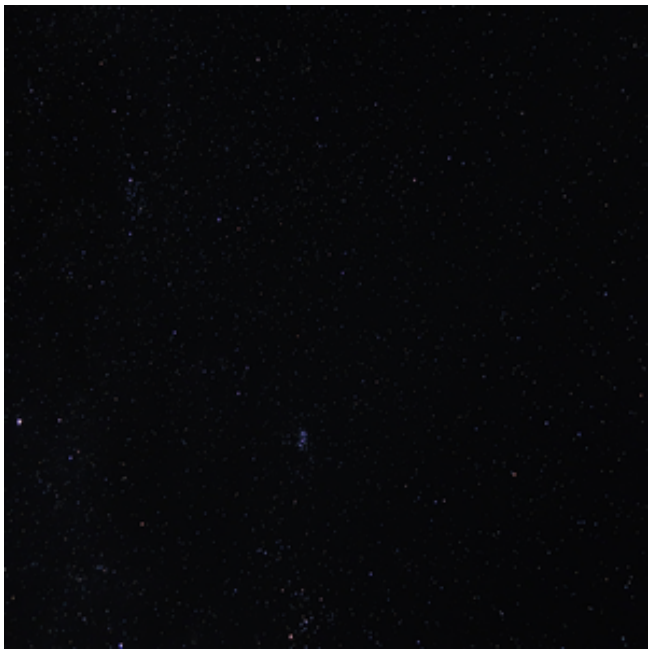

Figure 4: Image labeled only Stars

Figure 5: Image labeled only Desert

Figure 6: Image labeled Snow and Mountains

References

[1] W. Koehrsen, "An implementation and explanation of the random forest in python." Website, 2018. https://towardsdatascience.com/an-implementation-and-explanation-of-the-random-forest-in-python-77bf308a9b76; accessed March 09, 2021.

[2] "sklearn.ensemble.randomforestclassifier." https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html. accessed March 10, 2021.

[3] M.-L. Zhang and Z.-H. Zhou, "Ml-knn: A lazy learning approach to multi-label learning," *Pattern recognition*, vol. 40, no. 7, pp. 2038–2048, 2007.

[4] "Bayesian inference - wikipedia." https://en.wikipedia.org/wiki/Bayesian_inference. accessed March 10, 2021.

[5] R. N. Sucky, "Multiclass classification using logistic regression from scratch in python: Step by step guide." Website, 2020. https://towardsdatascience.com/multiclass-classification-algorithm-from-scratch-with-a-project-in-python-step-by-step-guide-485a8 accessed March 10, 2021.

[6] S. Smetanin, "Sentiment analysis of tweets using multinomial naive bayes." Website, 2018. https://towardsdatascience.com/sentiment-analysis-of-tweets-using-multinomial-naive-bayes-1009ed24276b; accessed March 10, 2021.

[7] "sklearn.metrics.f1score." https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html. accessed March 10, 2021.

[8] "Metrics and scoring: quantifying the quality of predictions." https://scikit-learn.org/stable/modules/model_evaluation.html#precision-recall-f-measure-metrics. accessed March 10, 2021.

[9] J. H. Bear, "The hsv color model in graphic design." https://www.lifewire.com/what-is-hsv-in-design-1078068, 2019. accessed March 11, 2021.

[10] C. Palm, D. Keysers, T. Lehmann, and K. Spitzer, "Gabor filtering of complex hue/saturation images for color texture classification," 07 2000.