# APPLIED MACHINE LEARNING SYSTEM ELEC0132 20/21 REPORT

*SN: 19162034*

## ABSTRACT

This report shows the working and findings of four tasks in image processing. The first, detecting gender of people in pictures, solved by a voting estimator of three algorithms to try and maximize accuracy. Second, detect if these people are smiling, minimizing resource utilization while not affecting accuracy. Third, detect eye color of cartoons, which provides great results except for the fact that 15% of cartoons are wearing sunglasses, making it impossible to see their eye color. The problem could be solved by inferring the eye color from other features, but for this a massive number of variables is required. Fourth, detect the face shape of these cartoons, which achieved perfect results with barely any hyperparameter tuning, so the resources utilized were minimized.
Link to GitHub code:
https://github.com/sebbarpar/AMLS_assignment20_21

## 1. INTRODUCTION

Machine learning algorithms are to be compared to determine which one fits a certain image processing task. The algorithms will be tested in two datasets:

- Celeba [1]: 5000 pictures of people. Two tasks are defined:
  1. Predict gender of the person in the picture.
  2. Detect if the person in the picture is smiling.
- Cartoon_set: 10000 pictures of cartoon faces with randomly generated features. Again, two tasks are defined:
  1. Detect eye color of the cartoon.
  2. Detect face shape (4 types).

A summary of all considered algorithms and methods is outlined. The proposed models are described, the implementation detailed and finally the results analyzed.
*Note: Only very relevant figures are included in the document itself. Others have been added as additional material*

## 2. LITERATURE SURVEY

The algorithms considered for every task were mainly supervised learning algorithms. The algorithms considered were:
1. Logistic regression [2]: the variables are used to create a regression curve and, using a sigmoid function, they are classified. Classical regression is not used due to the fact that all tasks are classification problems, not continuous estimations.
2. Ridge regression [3]: Takes into account a regularization parameter to avoid overfitting. Similar to logistic regression, but with a defined parameter specifically to deal with over/underfitting.
3. K-nearest neighbors [4]: Every instance of the dataset is plotted in a n-dimensional plane (n being the number of variables). The labels of the k-closest points are studied to make predictions about the label of the point in question.
4. Decision trees/ random forest [5]: A tree is created and the labels for every instance is filtered by the tree. A random forest is an ensemble algorithm in which many trees are created and the final label is chosen by majority voting.
5. Bagging [6]: Similar to a random forest, but with smoother decision boundaries. It uses a training set of many examples to train various models which are then aggregated.
6. Boosting [6]: Combine many weaker estimators to obtain a stronger one.
7. Naïve-Bayes [7]: Assumes conditional independence between every pair of features if the class of the variables is known. Only works with binary classification problems

Apart from these, a consideration was made to construct a neural network to estimate the labels. These mimic the functioning of a human brain, using neurons to estimate labels and back-propagation to calculate the error associated with every preceding layer. However, this will be avoided whenever possible. Although a neural network can obtain very good results, all tasks can be solved by copying the same neural network and changing the labels and dataset. Therefore, it is believed that showcasing other algorithms and different ways to preprocess the dataset will be more beneficial to the learning experience, and neural networks will only be utilized when no other option is available.
Kernel learning [8] is another algorithm that can be used, in which a set of hyperplanes are constructed to try and separate the labels of the dataset by using kernels.
Another ensemble algorithm that is considered is voting different algorithms to combine their results and estimate labels with a higher accuracy score.
An unsupervised machine learning algorithm that could also be applied is clustering [9], which groups together similar points in the data and then the labels could be estimated for each of them.
A state-of-the-art algorithm that was used was the extraction of features of a face [10] by defining certain points that are meant to exist in every face. Other considered algorithms were rotating the images [11] to train the model to better detect these features even when they are not in the expected position, the Viola-Jones face detector [12], which uses the

AdaBoost algorithm to detect where the faces are in a picture; the canny edge detection [13] algorithm and k-fold validation [14] in order to obtain an unbiased accuracy score.

## 3. DESCRIPTION OF MODELS

For every task, a different algorithm was defined and implemented. In this section, the reasoning behind the algorithm choice is outlined.

### 3.1. Task A1: Predict gender

The objective of this task is to determine the binary gender (male or female) of the person in the picture. As opposed to other tasks, there is not a single unique identifier that could determine whether the person is male of female. In task A2, where a smile must be detected, the features around the mouth are much more important than perhaps the other ones around the face, but in this case, there is no way to limit the number of features immediately. For this reason, an ensemble method was used in which three different models were tested and then the result voted by weights. The three used are:

1. Logistic regression:
   One of the simplest algorithms, classifies following a sigmoid (Figure 1) function to determine if the output will be 1 or 0.

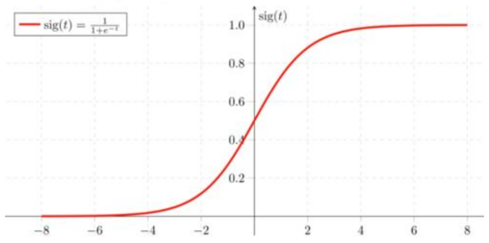

**Figure 1:** Sigmoid function

2. Ridge regression:
   Takes into account overfitting by varying the regularization parameter. By changing the value of this parameter, the fitting of the model to the dataset can be configured.
3. SVM:
   Support Vector Machine, separates the data by constructing a set of hyperplanes to be used for classification. To find the best one kernels are used, that is, positive semi-definite symmetric functions.

The flow chart can be seen in Figure 2.

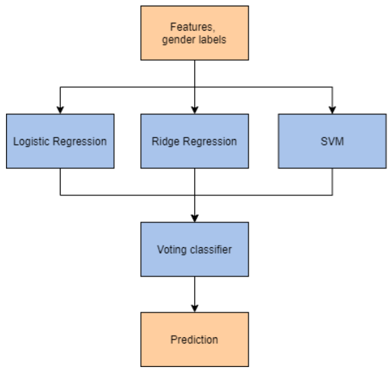

**Figure 2:** Flow chart for task A1

These three algorithms were chosen because the objective for this task will be to maximize the accuracy of the algorithm. Logistic regression might be simple, but it achieved very promising results with slight overfitting, which is why ridge regression was also introduced, as a way to balance this. SVM is a much more sophisticated algorithm, brough in to try and serve as a tiebreaker in case the two previous algorithms did not agree. The choice for this to be SVM and not some other algorithm is that it maximizes the margin, allowing for the decision boundary between classes to be as far away from them as possible, allowing for smoother decision boundaries and providing better results.

### 3.2. Task A2: Detect smile

To design the model for this task a different approach was used. While in the previous one many different models were used and then combined into one, in this case one single model was picked and then processes to achieve the best result. A random forest was generated, using many trees which later, by majority voting, pick the final class of the output. This is therefore an ensemble method by itself. However, this method uses too many resources to justify the results obtained. For this reason, some of the features will be deleted and not taken into account. The reasoning for this is that the way the images are being processed is by extracting 68 features from the face and then using these to train the algorithm (see Section 4). Out of these 68 features it is quite likely that the most important ones will be the ones of the mouth, since the position of these will most likely detect the smile. Therefore, many features can be deleted in order to utilize less resources. In section 4 it is explained how the features to be deleted are chosen.
However, after testing on the validation dataset, it was determined that this algorithm caused overfitting, and after it proved difficult to combat, a different method was implemented using the reduced features extracted from the random forest. This algorithm was logistic regression, due to its simplicity and reasonable results, keeping in line with the objective of reducing resource utilization. The flowchart for this task can be found in Figure 3.

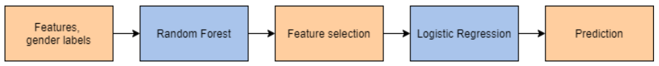

**Figure 3:** Flow chart for task A2

### 3.3. Task B1: Detect eye color

This dataset is quite different from the previous one, and the task requires different preprocessing as well. While previously what mattered most was the position of different features on the face, in this case it is the color that matters. Not only that, but the color in a very specific part of the face: the eye. Therefore, the algorithm used previously to detect features is not useful anymore, and the image must be imported a different way. Since the color is important, the image must be imported as RGB, which means every pixel will be represented as an array with 3 values. This creates a gigantic array (10000x500x500x3), which obviously is too big to use it as the dataset. At this point it is important to study how these images are generated. There is a list of attributes apparently independent from one another, one of them being the eye color, and therefore there is no reason to take into account any other part of the face. There is no attribute that sets the position of the eyes (only if the eye is tilted inwards or outwards), and therefore it is safe to assume that there is a point *(x,y)* in the image that is always on the iris of the eye. There are many of these points, so if one is found, its 3 RGB values will become the only variables needed to estimate the color of the eye.

Any algorithm should provide good results with only 3 variables and only 5 outcomes, but the chosen one was the bagging classifier. The reasoning for this was that the colors are not entirely constant (they can vary by a few values in R, G or B), and therefore smoothing the decision boundaries of the tree will prevent overfitting and provide better results. The flow chart of the bagging algorithm can be found in Figure 4.

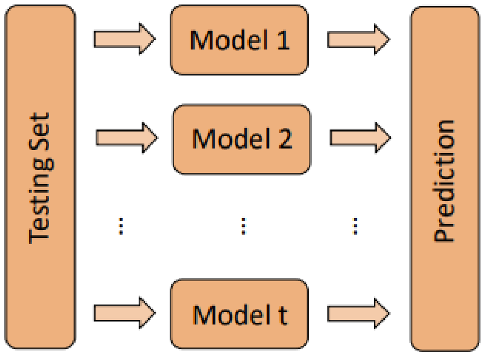

**Figure 4:** Bagging flow chart

### 3.4. Task B2: the task name

Since this is the task with the largest number of variables to be taken into account when training and testing the model, all methods that are computationally complex should be discarded, such as K-nearest neighbors, SVC or random forest. A simple algorithm should work, since so many variables will allow for easier classification. The algorithms considered as choices for this will be decision tree and simple logistic regression. At first, the thought was to use a voting algorithm, as in Task 1, and bring in another algorithm to serve as a tiebreaker. However, after seeing the results, it was determined that this was unnecessary and one algorithm is enough to obtain near perfect results. Instead of this, the objective now became to try and reduce resource utilization. It was considered that resources could be reduced more effectively with the logistic regression algorithm, so this was the final choice.

## 4. IMPLEMENTATION

The data that is available is simply pictures and a file with the labels associated with these pictures. For the first dataset (celeba), the labels were ordered in this format:

| | | |
|---|---|---|
| Name | Gender | Smile |

Meanwhile, in the second dataset (cartoon) the labels were slightly different:

| | | |
|---|---|---|
| Eye Color | Face Shape | Name |

Therefore, a different function must be used to extract these features and put them into a numpy array. This function will read each of the lines and split them in three parts: Feature1, Feature2 and Path. The first feature will be either the gender (first dataset) or the eye color (second dataset). The second will be smile (first) or face shape (second). The Path will be the path to get to the image from the folder in which the code being executed is.

This way, the labels have been extracted from the file, but they are in format “dict” and it is necessary to convert them to a NumPy array to be able to utilize them. For this the dict will be read line by line and added to the corresponding column of the NumPy array. An interesting observation is that the order in which the labels are extracted and the images read is following the path and increasing its value by 1. Therefore, the order of the images that will be processed is:

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 10 | 100 | 1000 | 1001 | 1002 | 1003 | 1004 | … |

Clearly, this order is not the order the loop function follows, and therefore the index of the row where the features for these images will be stored does not simply increase by one every iteration of the loop. For this reason, the index for picking the labels must be extracted from the image name, and not from the iteration of the loop.

For the dataset used in Task A, the preprocessing to be done is extracting the features of the image. To do this the library *dlib* is used. Inside this library there is a function which allows to extract 68 features from a face. These features are supposed to exist in every face, and they correspond to specific places in a face (Figure 5)

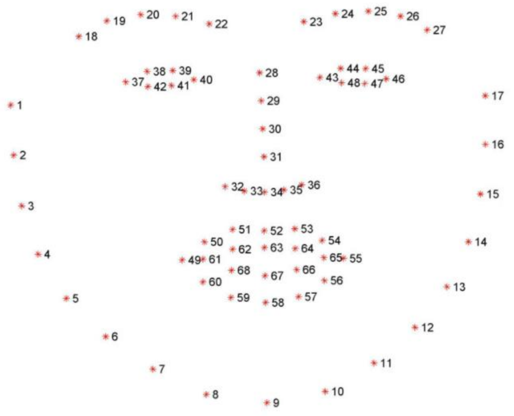

**Figure 5:** 68 features of a face

The function returns all 68 features by a 2-dimensional point for each one. This way every one of the 68 points is represented as a pair of numbers which represent its position in the image. The values of these must now be simplified into a 1-dimensional array to be able to input it into the model.

For dataset B, the images will be imported in their entirety, each as an array of 500x500x3 integers (3 from RGB).

Before any actual training or calculations, outliers and missing values should be taken into account, as they could greatly affect the performance of the algorithm:

1. Outliers:
   Task A: Since the data does not have any numeric values and simply is pictures, the outliers that could exist are after a bit of preprocessing of the data. After features are extracted, the z-score is compiled for all features and checked if any of the value exceed the threshold set as 3. In testing, no outliers were detected.
   Task B: The images are converted to matrices with depth 3 (RGB). There is no reason to expect any outliers, since the images are simply being imported with the cv2 library.
2. Missing values:
   Missing values must be dealt with. There are two possible places for these to be: in the labels file or after the features have been extracted. If no feature values were found, the image is immediately dropped and is not used in the algorithm. The z score didn't find any outliers and if the values were missing, they would be substituted with a 0, which would have been found an outlier. Therefore, it can be assumed that there are no missing values after the feature extraction. A check of the labels file for both datasets reveals that no missing values are present there either.

Once the datasets are ready for processing, the next step will be to define the training, validation and test datasets. This is achieved by splitting the original dataset into 2 parts and keeping the smallest (20% of the original dataset) as test set. The other part is split again into training and validation [15].

Finally, the libraries used were:

- Sklearn: implement the machine learning models and algorithms.
- Numpy: operations with matrixes
- Cv2: Process image.
- Dlib: Perform feature extraction from the images

## 4.1. Task A1: Predict gender

As explained before, three different models were trained and their results voted. They are explained here in detail:

*4.1.1 Logistic regression*

The first and perhaps simplest model, in which simple regression is used to classify whether the person is male or female. To use this model the maximum number of iterations had to be increased, since the dataset was too big for the default number (100).
Another parameter changed originally was the regularization strength, which controls the overfitting of the algorithm. As seen in Figure 12, this value was 0.1.

*4.1.2 Ridge regression*

Ridge regression takes into account overfitting and regularizes the model by penalizing some large parameter values. This is done by choosing the value of the regularization parameter. Initially, the values tried were in base 10 ($10^{-2}$, $10^{-1}$, $10^{0}$, $10^{1}$, $10^{2}$, $10^{3}$), and after further testing the value which maximized both accuracy scores was 30. Figure 13 shows the results.

*4.1.3 SVM*

Defining hyperplanes to classify values through kernels. The kernel chosen was the option 'poly', since it provided the best results. The C parameter, which is the same as in logistic regression, was chosen following Figure 14 as 1.5. The degree of the polynomial kernel function was kept at the default value of 3, as seen in Figure 15.

By putting these three models to a vote, in which their weights were based on their accuracy score, the final model is calculated. Since all three models had similar accuracy scores, the value of all weights is 1. The learning curve for the final algorithm can be found in Figure 6.

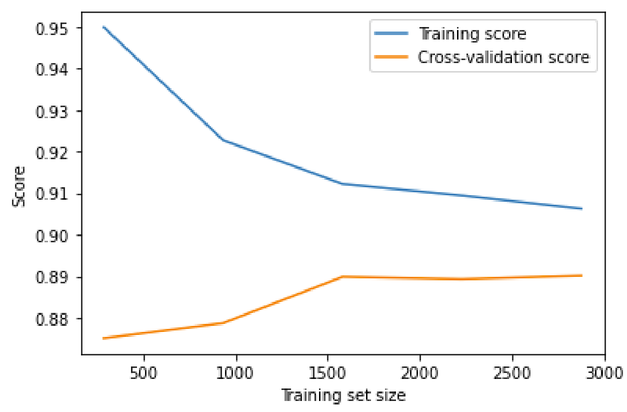

**Figure 6:** Learning curve task A1

By observing this curve, it is possible to estimate that the training and cross-validation score are converging, and if the training set was larger, they would probably converge completely. However, since the training set size was about 1600, the curves converge much slower. Therefore, if computing power was a limitation, the dataset could be decreased to this value and the overall performance of the algorithm would not be as affected greatly.

## 4.2. Task A2: Detect smile

The first thing to do will be to determine the parameters when calling RandomForestClassifier. For this, the first one will be the number of trees. The optimum value should be between 68 and 128 [16], so the possible values have been plotted in Figure 16. As expected, the value is between 68 and 128. Since the objective of this algorithm will be to achieve good results using few resources, 68 trees will be the best number, since 108's higher accuracy does not make up for its usage of resources. The plotted accuracy is the validation dataset, since the training had 100% accuracy for all values of trees, which shows overfitting. By varying other hyperparameters this will be combated.

The first parameter to be set is the maximum depth of the tree. If left blank, the nodes will continue expanding. Since the other parameters are also being set to try and minimize resource utilization, the accuracy has been plotted in Figure 17. There is still overfitting, as the validation accuracy does not increase when the training accuracy decreases. The same happens for the minimum samples split in Figure 18.

Overfitting is proving difficult to solve with the random forest algorithm, but there is still something it can offer. As explained in section 3.2, some of the features of the image will not be taken into account. The way to choose them is with the built-in function of RandomForestClassifier, which assigns a numerical value to the importance of the feature. If the value is below a certain threshold, that feature will be dropped. To find the value of this threshold a few are tested and the accuracy scores compared in Figure 7, along with the number of features left. It is important to say that the accuracy score was calculated with k-fold variation in order to have a constant measure that will not change if the dataset were split differently.

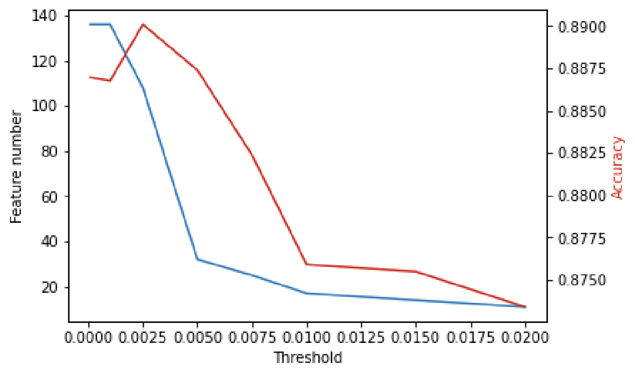

**Figure 7:** Features vs accuracy task A2

As it can be seen, the accuracy has a drop that starts after the drop of the feature number. The objective will be to find a point where the feature number is greatly diminished without having accuracy too far below the maximum. It is quite clear that it is the point with threshold 0.005, since the drop in features is massive, while the accuracy only falls by 0.25% from the maximum value. Therefore, this will be the chosen operating point of the curve. Although other algorithms were tested and provided similar results, this one was chosen because the solution seemed relevant to the case in question, since only some of the features were relevant. The learning curve for this task can be found in Figure 8.

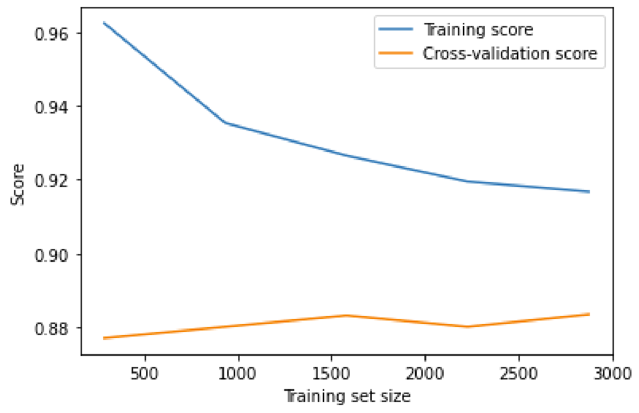

**Figure 8:** Learning curve random forest task A2

From this graph, it is extracted that the scores are not converging as fast as they should, and a 4-point difference is quite big. This shows clear overfitting, as will be tested in section 5. For this reason, another algorithm was used with the new reduced features. The chosen algorithm was simple logistic regression, which does not utilize massive resources and provides reasonable results. The hyperparameter to be trained was C, as seen in Figure 19.

The chosen value could be 0.7 or 1.5. Keeping in line with reducing resource utilization, the value chosen will be 1.5. The learning curve is as seen in Figure 9.

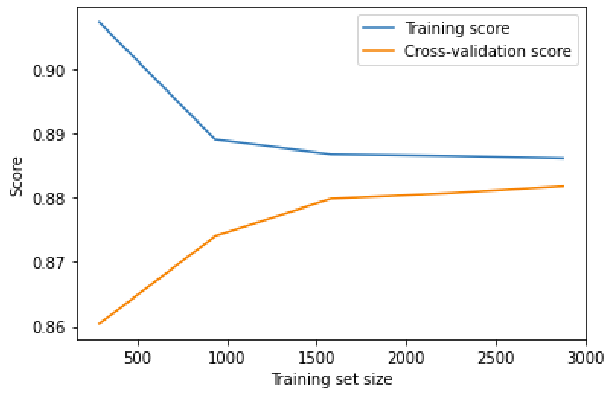

**Figure 9:** Learning curve logistic regression task A2

Clearly, much better results as far as overfitting is concerned, with the two curves converging. The stopping criterion allows for the training set size to be set to 1600 and not affect the outcome of the algorithm.

### 4.3. Task B1: Detect eye color

The first hyperparameter to be set will be the number of estimators used in bagging. Figure 20 shows the training and validation accuracy. Reasonably, the value chosen will be 30 estimators. For the maximum number of samples, Figure 21 shows the accuracy results. Following this, 0.5 is the chosen parameter. Finally, the hyperparameter max_features will be chosen, which by default is 1. Figure 22 shows that the maximum accuracy is obtained when the parameter is 2. The learning curve for this task can be found in Figure 10.

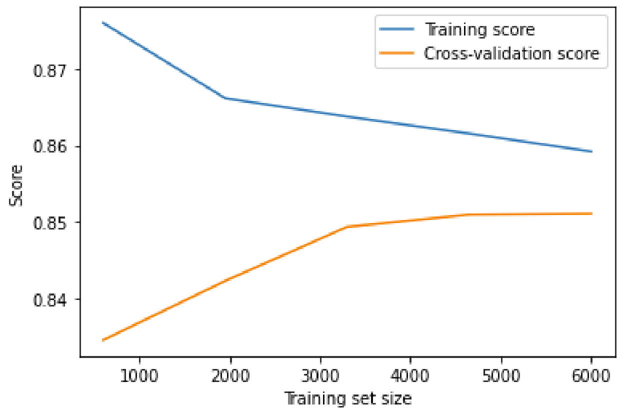

**Figure 10:** Learning curve task B1

The results are quite satisfactory, since both curves converge at very similar values. As for the stopping criterion, the converge does not improve drastically since the size of around 4700, and the algorithm could be stopped here to reduce computational complexity without having an important effect on performance.

### 4.4. Task B2: Detect face shape

Data preprocessing is crucial. The approach chosen here was to import the image and later use Canny edge detection. The point of this is to decrease the amount of data to process. If the original image was an array of 500x500x3 integers (RGB), the converted image, after implementing edge detection, is an array of 500x500x1 uint-8. Since every pixel can be represented with one bit (1-white or 0-black), the size of the variables to process is decreased greatly. Even more so when cropping the images to cut out the blank space on the sides. After cropping, the final image will be 300x300x1 bits. This is still computationally expensive, although when testing both proposed algorithms (logistic regression and decision tree), the accuracy seems to be 100% for both, even without hyperparameter selection. Therefore, the objective will be to make the model less computationally expensive.

One of the approaches was to try and decrease the number of variables that are used for the model, similarly to task A2. However, this proved to be very computationally expensive, so the idea was discarded. Ultimately, the chosen algorithm will be logistic regression. To be able to make the tree less computationally expensive, if not be decreasing the features, the method would be to select the max_depth, min_samples_split, min_samples_leaf…etc. However, for logistic regression by simply adjusting the tolerance for stopping criteria the algorithm will be less expensive. After testing different values on the validation dataset, the results are in Figure 23. As it can be seen, when adjusting the tolerance to 0.1, the accuracy only falls by 0.05%, which is an acceptable result considering how much less computationally expensive the algorithm is. The learning curve can be found in Figure 11.

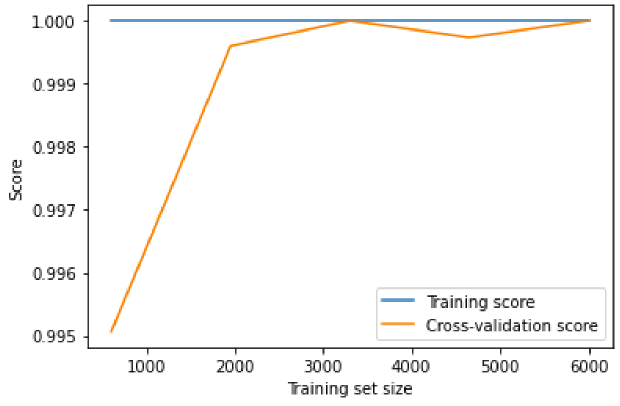

**Figure 11:** Learning curve task B2

Clearly the best learning curve obtained, in which the convergence happens very early, and for the stopping criterion, even with as little as 1700 samples would be enough to have near-perfect results. Just in case, 2000 samples is the number that was chosen to train the algorithm.
A note is to be made that a neural network would fit this task well, since the number of features is very high. However, as explained in section 3, this algorithm is avoided due to

the fact that it is believed to be more beneficial to the student to work out different algorithms and preprocessing techniques.

## 5. EXPERIMENTAL RESULTS AND ANALYSIS

For every task, a different model was created. Therefore, the performance of each task will be evaluated separately. In every task, the original dataset was separated into three parts: training, validation and testing. The accuracy will be tested for each. There is another dataset given which will only be tested on. The accuracy values should be around the same as they are for testing in the first dataset.

### 5.1. Task A1: Predict gender

Since two datasets were given, both will be evaluated. The table below compared how each algorithm individually scores and the total.

| Model | Dataset 1 | | | Dataset 2 |
|---|---|---|---|---|
| | Training | Validation | Test | |
| Logistic regression | 93.9% | 91.1% | 90.9% | 90.2% |
| Ridge regression | 93.1% | 91.1% | 91.2% | 90.9% |
| SCV | 93% | 91.2% | 91.8% | 91% |
| Final model | 93.6% | 91.4% | 92% | 91.4% |

The results are very satisfactory, since a high accuracy is reached, and the new dataset has a similar accuracy to the testing of the original one. However, the question could be raised if there is a reason to put these three models to a vote, since for example SVC offers very similar results to the final model, and if the additional utilization of resources justifies the improvement. The improvement in accuracy of an algorithm is not linear, and it is not equally easy to go from 60% to 70% as it is to go from 90% to 100%. A small improvement as achieved by voting all three algorithms is a reasonable result, as to achieve a better result would be very complicated given the high accuracy to begin with. In some of the following tasks the objective was to decrease the resource utilization, while in this one the desired result is to showcase that a slight improvement might entail larger resource utilization. Therefore, the choice has been made to submit al three algorithms to a vote. If this were a scenario in which resources are scarce, only one of the three algorithms would be used.

### 5.2. Task A2: Detect smile

Again, both datasets are compared and tested.

| Model | Dataset 1 | | | Dataset 2 |
|---|---|---|---|---|
| | Training | Validation | Test | |
| Random forest with all features | 99% | 87% | 88.9% | 89.9% |
| Random forest with selected features | 97.9% | 86.7% | 88.7% | 89.7% |
| Logistic regression | 88.6% | 87.7% | 87.2% | 87.8% |

As it can be seen, the accuracy score is slightly lower when selecting a limited number of features. While the difference in accuracy is quite small, the number of features evaluated is much lower, so the results are very satisfactory. As opposed to the previous algorithm, the objective is now to decrease resource utilization to showcase a different way the problem could be presented. If the results wanted to be maximized, more algorithms could have been brough in and then voted. Instead, it was considered more beneficial to show a different approach when training a machine learning algorithm. As mentioned before, the final algorithm used to estimate with the new reduced features is logistic regression, which does not have overfitting, and the total accuracy could be improved by applying a similar method as task 1, with many algorithms subject to a vote.

### 5.3. Task B1: Detect eye color

| Model | Dataset 1 | | | Dataset 2 |
|---|---|---|---|---|
| | Training | Validation | Test | |
| Bagging | 85.9% | 85.1% | 85.4% | 84.8% |

Before the testing was conducted, a very high accuracy was expected, but instead of that what is received are reasonable results, but not exceptional. After further examination, this is due to the fact that about 15% of the images are wearing sunglasses, making it impossible to determine their eye color. A look at the origin of the dataset will show that eye color is dependent of other parameters, "*The mapping from attribute to artwork is also defined by the artist such that any random selection of attributes produces a visually appealing cartoon without any misaligned artwork; this sometimes involves handling interaction between attributes.*" [17]. This, however, is problematic. There is no guarantee that the eye color is dependent upon anything else, since it could produce a "visually appealing cartoon" being set randomly, and even if it was dependent, it is impossible to know of which parameters. Therefore, to correctly estimate it and raise its accuracy, the training set should be composed of the pictures in their entirety, causing for unreasonable memory usage. Another way would be to implement a neural network.

As a note, after testing task B2, the canny images were input as variables to try and detect eye color. There are 5 different eye colors, so a random guess should produce an accuracy of about 20%. However, every algorithm tested produced the same accuracy: 50%. This leads to believe that the other features of the face do impact the eye color, and with if the dataset chosen as input to the machine learning algorithm was the entirety of the pictures, an accuracy of almost 100%

would be reached, same as in the next task. When trying to test this, a memory error message appeared, as the RAM could not commit that much memory. Overcommitting memory on Windows is not possible, and therefore this theory was never tested in practice. However, there is a strong reason to believe that with this dataset as input, an almost perfect accuracy would have been reached. To do this, the most appropriate algorithm would be to implement a convolutional neural network.

### 5.4. Task B2: Detect face shape

| Model | Dataset 1 | | | Dataset 2 |
|---|---|---|---|---|
| | Training | Validation | Test | |
| Logistic regression | 100% | 99.96% | 99.9% | 99.84% |

Almost perfect results, which could be turned to 100% accuracy if more resources were utilized (stricter tolerance, stopping criterion looser). As in task A2, the objective now was to minimize resource utilization without decreasing accuracy greatly. With this in mind, this is the task with the best results. The reason for this was the unambiguity of the variables. There are 4 different face shapes which will always be the same, and if the canny function detects edges at specific points, it detects one face shape, and if not, it detects another. This kind of accuracy would be very difficult to achieve in a real-world dataset, for example the one used in the first two tasks.

## 6. CONCLUSION

All four algorithms provided satisfactory results. In the first and third, the objective was to improve accuracy, while in the second and fourth it was to reduce resource utilization while not affecting the accuracy, in order to showcase a different type of approach. An interesting observation is the almost perfect accuracy of the last task, which is achieved due to the fact that the dataset is artificially generated. This kind of result would very difficultly have been achieved on the first dataset. The only reason that this does not happen in task B1 is that the eyes of 15% of cartoons are covered by sunglasses, making it impossible to determine their eye color.

The main problems with the implementation came from two sources:

1. The feature extraction from the first task could not find any features in an image and then the image will not be counted in the training or testing. To improve this, either a different algorithm should be used to extract features or be improved in order to train it to extract them better.
2. In task B1, memory should be allocated to test the theory that an accuracy of almost 100% could be achieved if all pixels of the image were evaluated, since with the algorithm chosen does not provide good results for cartoons with sunglasses. Another approach would be to program a convolutional neural network.

## 7. REFERENCES

[1] (S. Yang, P. Luo, C. C. Loy, and X. Tang, "From facial parts responses to face detection: A Deep Learning Approach", in IEEE International Conference on Computer Vision (ICCV), 2015

[2] Bewick, Viv & Cheek, Liz & Ball, Jonathan. (2005). Statistics review 14: Logistic regression. Critical care (London, England). 9. 112-8. 10.1186/cc3045.

[3] Arthur E. Hoerl & Robert W. Kennard (1970) Ridge Regression: Biased Estimation for Nonorthogonal Problems, Technometrics, 12:1, 55-67, DOI: 10.1080/00401706.1970.10488634

[4] Cunningham, Padraig & Delany, Sarah. (2007). k-Nearest neighbour classifiers. Mult Classif Syst.

[5] Breiman, L. Random Forests. *Machine Learning* **45,** 5–32 (2001). https://doi.org/10.1023/A:1010933404324

[6] Bühlmann, Peter. (2012). Bagging, Boosting and Ensemble Methods. Handbook of Computational Statistics. 10.1007/978-3-642-21551-3_33.

[7] Hand, D. J., & Yu, K. (2001). Idiot's Bayes—not so stupid after all?. *International statistical review*, **69**(3), 385-398.

[8] Lê, Linh & Hao, Jie & Xie, Ying & Priestley, Jennifer. (2016). Deep kernel: learning kernel function from data using deep neural network. 1-7. 10.1145/3006299.3006312.

[9] Bano, Saima & Khan, Naeem. (2018). A Survey of Data Clustering Methods. International Journal of Advanced Science and Technology. 113. 10.14257/ijast.2018.113.14.

[10] King, Davis. (2009). Dlib-ml: A Machine Learning Toolkit. Journal of Machine Learning Research. 10. 1755-1758. 10.1145/1577069.1755843.

[11] Zhang, Xin & He, Guojin & Yuan, Jiying. (2009). A Rotation Invariance Image Matching Method Based on Harris Corner Detection. 10.1109/CISP.2009.5304497.

[12] Viola, Paul & Jones, Michael. (2001). Rapid Object Detection using a Boosted Cascade of Simple Features. IEEE Conf Comput Vis Pattern Recognit. 1. I-511. 10.1109/CVPR.2001.990517.

[13] J. Canny, "A Computational Approach to Edge Detection," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. PAMI-8, no. 6, pp. 679-698, Nov. 1986, doi: 10.1109/TPAMI.1986.4767851.

[14] Rodríguez, Juan & Pérez, Aritz & Lozano, Jose. (2010). Sensitivity Analysis of k-Fold Cross Validation in Prediction Error Estimation. Pattern Analysis and Machine Intelligence, IEEE Transactions on. 32. 569 - 575.

[15] Xu, Y., Goodacre, R. On Splitting Training and Validation Set: A Comparative Study of Cross-Validation, Bootstrap and Systematic Sampling for Estimating the Generalization Performance of Supervised Learning. *J. Anal. Test.* **2,** 249–262 (2018). https://doi.org/10.1007/s41664-018-0068-2

[16] Oshiro, Thais & Perez, Pedro & Baranauskas, José. (2012). How Many Trees in a Random Forest?. Lecture notes in computer science. 7376. 10.1007/978-3-642-31537-4_13.

[17] Google.github.io. 2020. *Cartoon Set: An Image Dataset Of Random Cartoons.* [online] Available at: <https://google.github.io/cartoonset/download.html> [Accessed 18 December 2020].

## ADDITIONAL MATERIAL

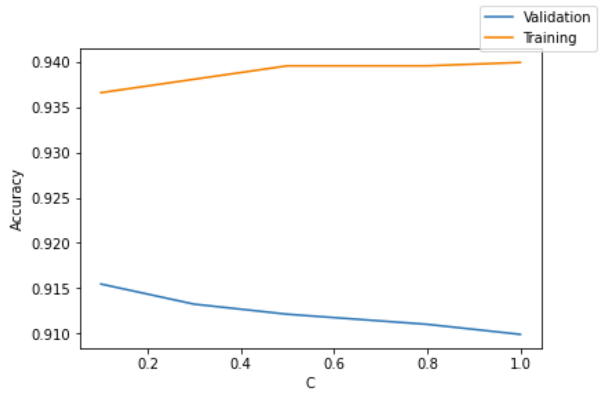

**Figure 12:** Regularization strength logistic regression task A1

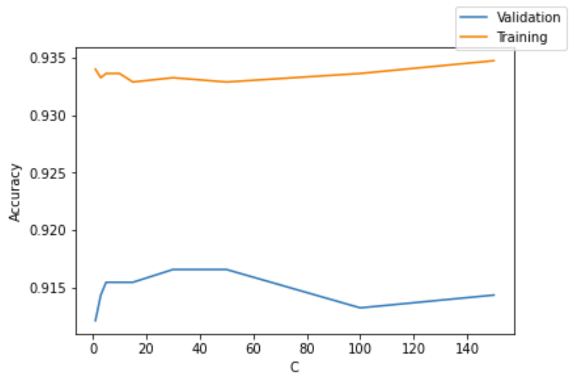

**Figure 13:** Regularization parameter ridge regression task A1

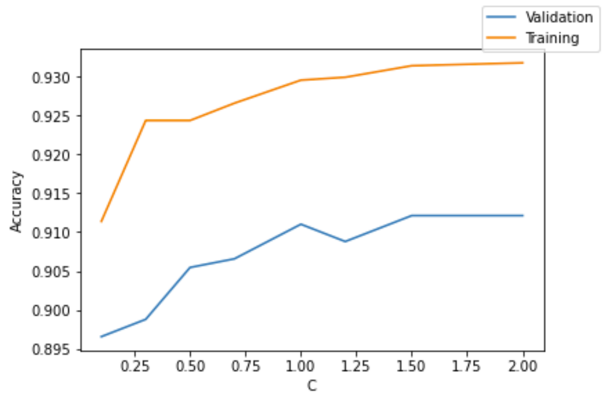

**Figure 14:** Regularization strength SVM task A1

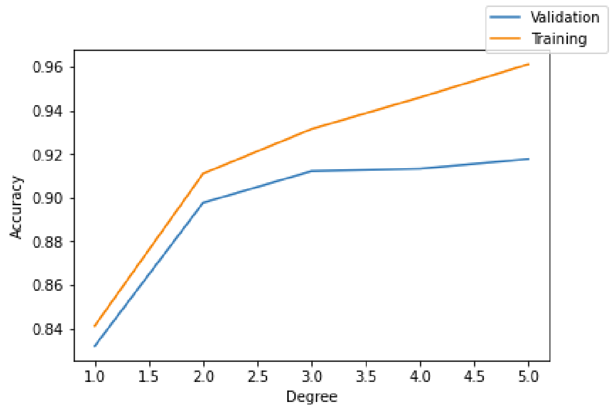

**Figure 15:** Degree of SVM task A1

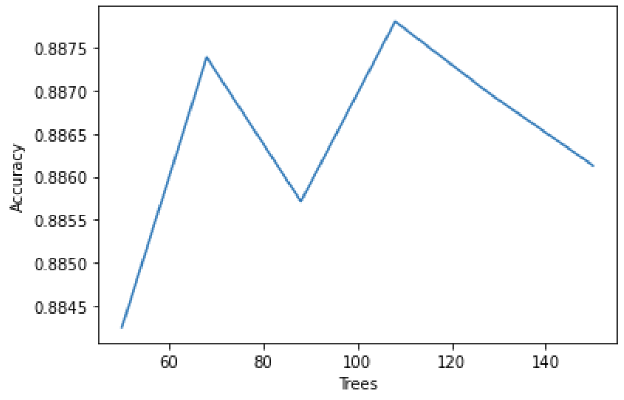

**Figure 16:** Number of trees task A2

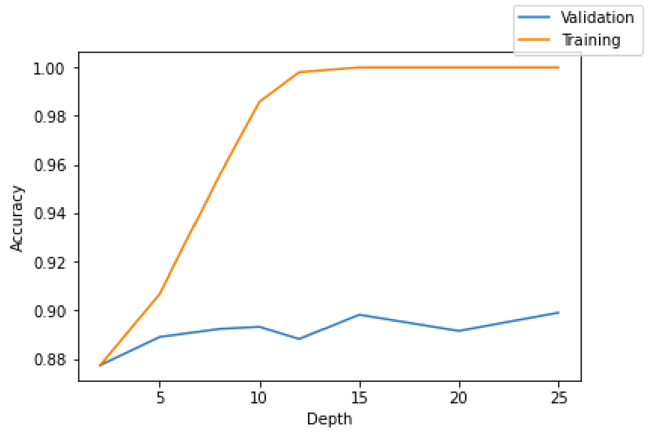

**Figure 17:** Depth of tree task A2

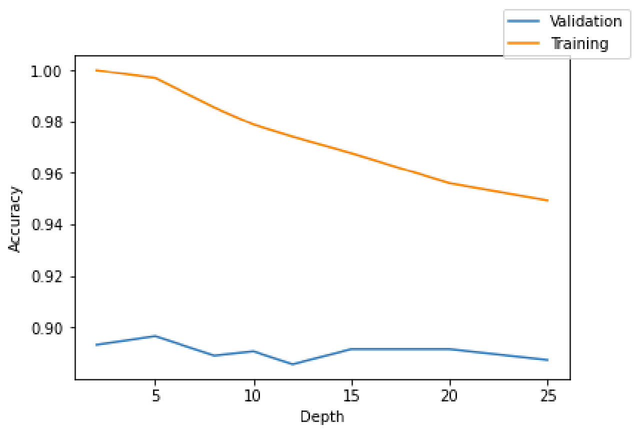

**Figure 18:** Minimum samples split task A2

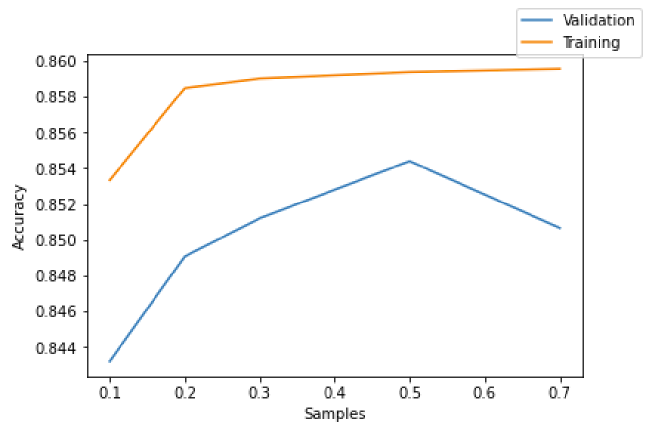

**Figure 21:** Max samples task B1

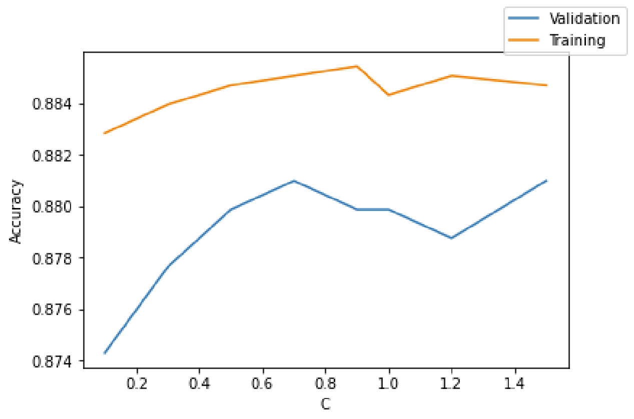

**Figure 19:** Regularization strength task A2

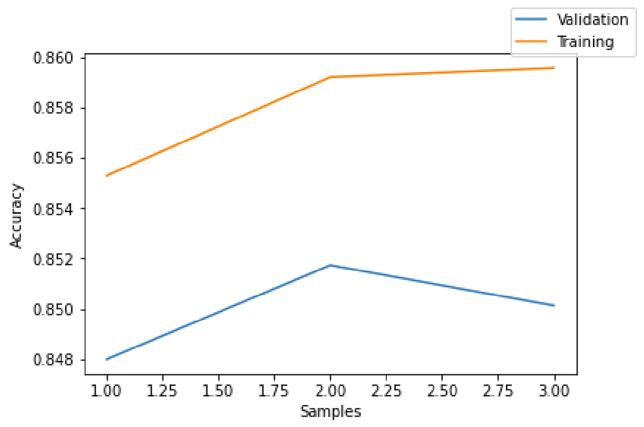

**Figure 22:** Max features task B1

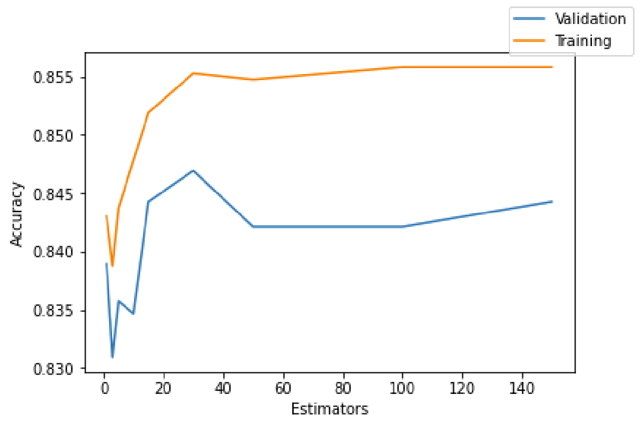

**Figure 20:** Estimators task B1

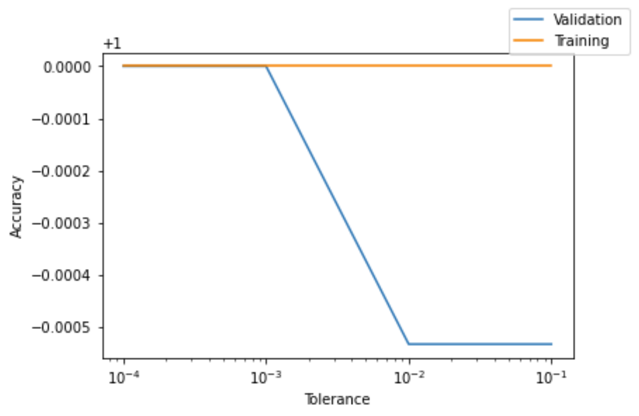

**Figure 23:** Tolerance task B2