

A COURSE WORK ON MACHINE LEARNING (CS5014)
PRACTICAL 2: LEARNING VISUAL ATTRIBUTES
STUDENT ID: 200015689.

PROJECT AIM

The aim of this practical is to apply machine learning on a recent open-ended research.

In this report, a machine learning classifier was developed to help predict color and texture.

This project accounts for

- How to select and train a suitable classification model
- How to evaluate and compare the performance of different models with the objective of choosing which is the best.
- How to explain and justify a work in a technical report.

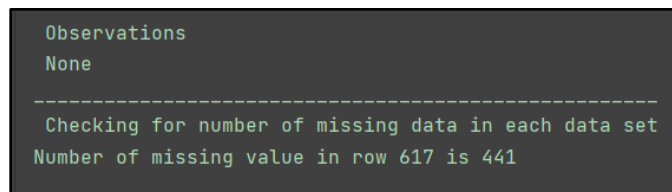
The dataset used in this report work is based on a subset of the GQA dataset for learning about relation. The dataset was split into two; the train and test dataset. The train dataset provides a ground for the model to learn on how to correctly classifier the targets. Whereas, the test dataset served as a real-world data with targets unknown. The dataset consists of images annotated in terms of boundary boxes and contains relevant attributes and relations.

QUESTION 1

Loading and Inspecting the data

The data (data_train and data_test) which was presented in CSV format was loaded using the pandas python library. On loading, the data was investigated to ascertain some information about the data which would be useful to construct a model for the classification (colour and texture) problem. Information such as the total observation, number of features, data type of each features, missing values and non-sensical data were critically analyzed. The reason behind this, is to determine the necessary steps to be taken in preprocessing the data, in order to develop an effective model. After critically examining the data, the following were observed;

- I. The data contained 1335 observations and 449 features.
- II. 447 features were found to be Numeric values (comprising of both float and integers), whereas, 2 features were found to be categorical (texture and color respectively), which serves as the target or class columns.
- III. Some columns contained 1 missing result. A total of 441 missing values was found on a specific row as shown in figure 1.0.



```
Observations
None
-----
Checking for number of missing data in each data set
Number of missing value in row 617 is 441
```

Fig 1a: A figure showing the total number of missing values as found in the train data set.

On realizing the independent or explanatory variables were all numeric, there wasn't any need for it to be encoded. Also, one hot encoding was ignored for the target variable, since the model to be employed (precisely, the Support vector Machine) works well even if the target variable is categorical. However, there was need to handle missing value as it affects the model, if not handled properly. To handle missing value for the affected column, the median value for the column was used since we had just 1 missing value for the affected columns. Nevertheless, this process was done after splitting the data, to avoid data leakage. Furthermore, the class label proportion for each target was ascertain to understand the distribution (either balanced or unbalanced) of the class in the dataset. The result is as shown in fig 1b and 1c for texture and color dataset respectively. This shows that the classes were imbalanced.

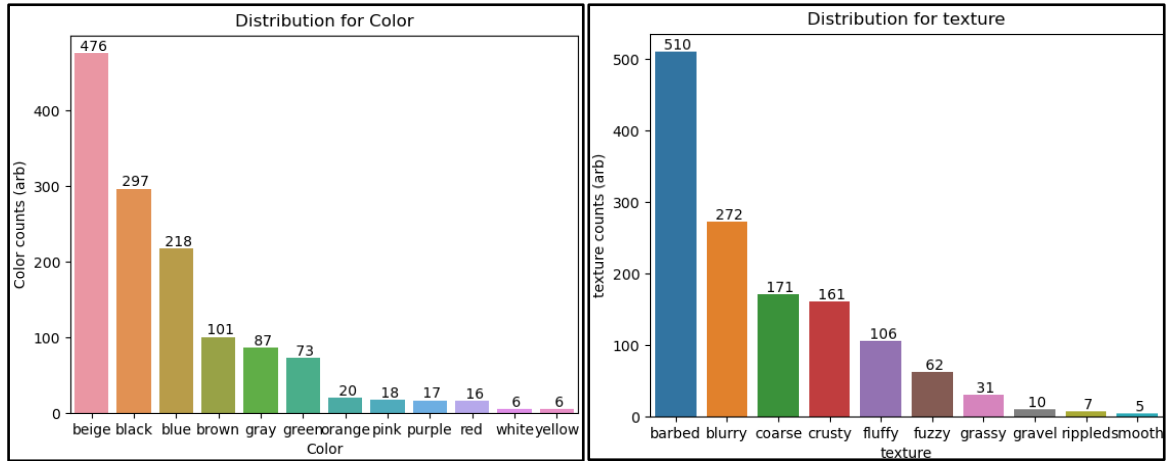


Fig 1b, 1c depicting the distribution of Class Labels across color and texture data. This shows that both classes are imbalanced and highly skewed.

Data Splitting

After thoroughly investigating the dataset, it was split using the `train_test_split` function as found in the `sklearn` library. Splitting for both color and texture classes was done by setting `test_size` parameter to 0.2. This allows for 80% of the dataset to be used for training, while the remaining 20% was used as validation data, to test the performance of the model. Aside setting `test_size` to 0.2, `stratify` was set to the texture and color classes for each split respectively. This is to ensure there is equal proportion of class labels across each split (train and validation). Also, `random state` was set to a number (123 was used here) to ensure reproducibility of splits. Upon successful splitting, the training data set was visualized in to understand how the data are arrange in space. The essence of this was to determine the best model which suites the classification task. The result is as shown in fig 1d and e for color and texture distribution respectively. The result shows that the data was non-linearly separable.

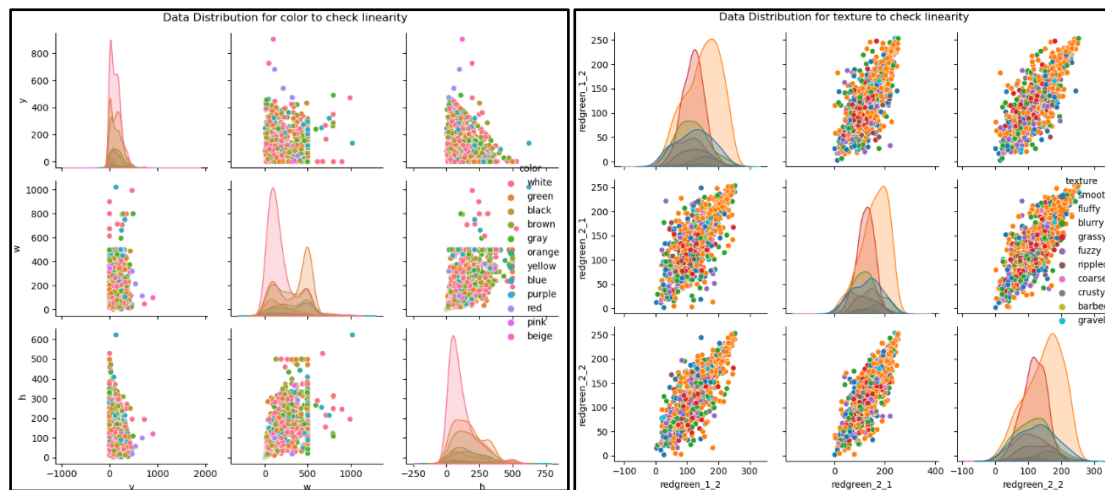


Fig 1d, 1e showing 2d visualization of the data, to check the nature of its distribution for both classes. The gaussian curve was also generated to show the relationship of the data across each class.

Feature Selection

Feature selection was done using the ensemble method. In this method, the DecisionTreeClassifier was used to select features which have high impact on the class. This process was done separately for both the texture and class classification, as they are dependent on different features. The reason of using the Decision Tree Classifier is because it works well on identifying features even in a non-linear space. Scaling was not done at this stage, since the decision tree is not sensitive to scaling. Fig 1f and g shows 10 features out of the total features selected.

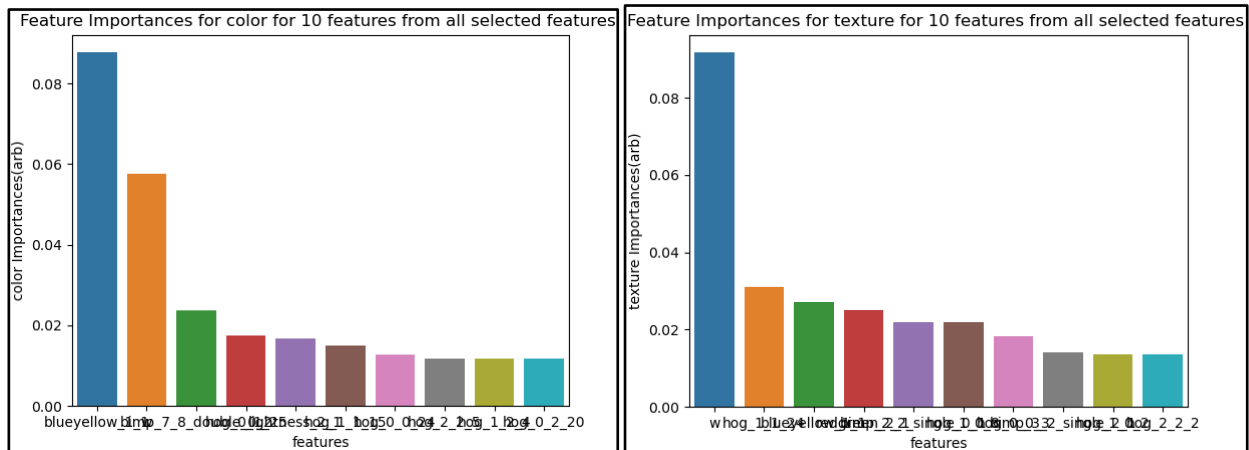


Fig 1f, 1g showing the feature selected and their impact on the class (for color or texture). Only 10 feature were displayed, however, more features were selected.

Feature Scaling

Upon selecting the important features, feature scaling was done using the StandardScaler (reference) which scales the selected features within a mean of 0 and Standard Deviation of 1. This was necessary because the model (Support vector machine) is highly sensitive to scaling.

QUESTION 2:

For this task, the support vector Machine (the support Machine Classifier) algorithm (a supervised learning algorithm) was used for both the colour and texture classification problem. The reason behind this is because it performs well on both linear and non-linear separable data set (reference). As seen in fig 1d and 1e, the data set was found to be non-linearly separable, as such requires a model that works well on fitting non-linear data. The Support vector machine has the ability to optimally fit in the non-linear through the use of kernel function, which helps to create an N dimensional space to map our features in a bid to create a hyperplane to linearly separate the class labels. This makes it a useful model for this classification task.

QUESTION 3

The support vector machine makes use of several hyper-parameter such as Class_weight, kernel, gamma, C, degree, and many more, to help solve a given classification. However, in this task, only few of the hyper-parameters were used, since it was observed to work best for the model. The Hyper-parameters used include;

- **C** – This hyper-parameter represents the regularization parameter which controls the amount of regularization. It serves as the penalty of the error terms. This tell the learning algorithm (support vector machine) how much to avoid in mis-classification of the classes.

The C hyper-parameter was set to **1.0** for both the texture and color models.

- **Gamma** – This is a hyper-parameter for non-linear plane. It determines how far the influence of a single data (training example) reaches.

The Gamma parameter was set to **0.001**, and **0.01** for texture and color classification models respectively.

- **Kernel** – This hyper-parameter selects the type of hyper-plane use to separate the data in a given space. It helps to map out features into N dimensional space in order to find a best hyperplane to separate each class present in the data set. Kernel can be linear, Gaussian rabial basial function (rbf), poly, sigmoid, or any user defined kernel function.

The Kernel parameter was set to **“rbf”** for both classification models (texture and color classification) respectively.

- **Class_weight** – This hyperparameter assigns weight to both minority and majority classes of the target data. It can be None, auto, balanced, or user defined (using a dictionary with values for each class).

The class_weight parameter was set to **balanced**, for both texture and colour classification model. Doing this automatically assigns class_weight inversely proportional to the frequency of each class, to avoid bias towards the minority class.

This hyperparameters were tuned and optimally selected using the GridSearchCV (ref), which selected the best values based on a combination of different parameter values with the best-balanced accuracy. In doing this, random_state parameter was set to 12, to ensure reproducibility.

QUESTION 4

The type of regularization employed in the model is the **l2-norm regularization**. This seeks to penalize the model, in a bid to control and reduce mis-classification.

QUESTION 5

The use of the Lagrange dual function optimization strategy was employed in the models for both texture and color classification. The optimization strategy allows the support vector model to model a complex non-linear boundary, by using basic expansion to build new feature space, through the use of kernels. The lagrange dual optimization strategy uses the dot product method, which can also be represented by the kernel function expressed as $K(h(x), h(x^i))$, and as a dot product $(\phi(x), \phi(x^i))$. In the support vector machine models, the basic expansion is integrated into the optimization method for the kernel function (in the case is the gaussian kernel (rbf)), which makes it easy to optimize the model, by setting optimal values for the parameter (known as slack variables), which represents the lagrange multiplier. The optimal values were gotten using the GridSearchCV function.

Slack variables include the C hyper-parameter, where $C = \alpha + \mu$, and gamma.

I prefer this optimization process over others because it uses the dot product approach, which can be easily substituted by the kernel function employed in the model. Also, it works well on constraint dual-primal optimization problems.

The strength of the Lagrange optimization method, is that it can be use to optimize both linear and non-linear optimization problems with complex equality constraint. It works effectively on constraint optimization problems, by using the dot product approach, and expansion strategy.

Nevertheless, one of it **weaknesses** is that it may not be effective in optimizing complex inequality constraint. It is practical for only solving small inequality constraint problem.

QUESTION 6

Although accuracy score can be used to evaluate the performance of model, it wasn't used to assess the performance of the model as used in used in this task. Accuracy score represents the percentage of correctly classified class in the dataset. Since, the data was imbalance (see fig 1b and c), the accuracy score is likely be highly, where it may correctly classify labels for the majority class, and maybe fail to classify for the minority class. As such, it wasn't used. Owing to the imbalanced nature of the dataset, the evaluation metrics employed to access the performance is the `balanced_accuracy_score`.

The balanced accuracy is based on the arithmetic mean on commonly used metrics (sensitivity and specificity) (**ref**). This represents a proportion of how well the model was able to classify both majority and minority classes correctly.

The model performance on both the texture and color classification are 0.2936 and 0.2037 respectively. Also, the accuracy of the model when `class_weight` was set to balanced is 0.4701 and 0.4341 for the color

and texture classification respectively. This is above the worst accuracy; 0.3566 and 0.3820 for color and texture classification respectively.

To ensure the result is repeatable and not a statical fluke, hyper-parameter for both models (color and texture) were consistent (based on the optimal values gotten), also, `random_state` was used for reproducibility when training the model. Aside from this, upon splitting, and hyper-parameter selection, stratified process was used (`stratify` for splitting, and `stratifiedKfold` for `GridsearchCV`).

QUESTION 7

Yes, some classes for both colour and texture were easier than the other in the dataset. This is as a result of class imbalance. Thus, the model tends to correctly classify classes with higher proportions compared to those with lower proportion. For example, as shown in fig 1b, it's easier for the color model to identify beige better than white which is very less. Likewise in fig 1c, the texture model is able to identify barbed easily than smooth. However, this tradeoff is solved by assigning `class_weight=balanced`.

QUESTION 8

The model (color svm and texture svm) was compared with a simple logistic regression classifier from `sklearn` (ref). The essence of this is to ascertain which model performs better for deployment. The simple logistic regression was assigned a hyperparameter of `class_weight` and `penalty`. The `class_weight` was set to `balanced` and the `penalty` was set to `l2` (ridge regularization) to handle mis-classification.

The models, both svm and logistic classifier for color and texture performance were compared using the `balanced_accuracy` score evaluation metrics as described in **question 6 above**. As shown in Table 1.0 and Table 2.0, the support vector machine(svm) model for both colour and texture classification had a better performance than the logistic regression. This can be justified by a higher `balanced_accuracy` of the Support vector machine as when compared with the Logistic regression. This is as result of the support vector machine being able to create a better hyper plane (using the kernel function “rbf”) in N dimensional space to map or separate each class. Although the logistic regression classifier works well on classification problem, its under-performs in cases where the classes are not linearly separable.

Table 1.0: A table showing the balanced accuracy score of the support vector machine model (main model) and logistic regression on color classification.

MODEL	BALANCED ACCURACY SCORE
Support Vector Machine	0.2037
Logistic Regression Classifier	0.1674

Table 2.0: A table showing the balanced accuracy score of the support vector machine model (main model) and logistic regression on texture classification.

MODEL	BALANCED ACCURACY SCORE
Support Vector Machine	0.2936
Logistic Regression Classifier	0.2214

ADVANCED SESSION

QUESTION 9

The main model (for color and texture) was further compared with the K nearest Neighbor (KNN) classifier to assess the performance of both models on the data, and finally justify the differences between the both algorithms. The KNN classifier algorithm is a supervised algorithm that is often used in classification problem. It works by calculating the distance of the test observation from all the observation it memorizes from the training dataset, and then finding the K nearest neighbor of it. To compute the distance, it uses the best or optimal distance metric, selected either from the Euclidean distance, Manhattan distance etc.

Technically, the KNN classifier classifies an unknown item by looking at K of it's already classified nearest neighbor item, computing and finding out the majority votes from the nearest neighbors.

As compared to the main model (support vector machine), it is also a non-parametric model, where the unknown parameter is K, and mostly likely the metric distance. For an optimal classification, an optimal value of K should be computed, along as using the best and appropriate distance metrics. Just like the main model used in this task, the KNN also performs well on non-linearly separable dataset. However, it tends to perform very low in cases where the class are not noise free, and highly skewed (imbalanced). Unlike the support vector machine (main algorithm) which performs relatively better than the KNN in cases of highly skewed class.

The KNN predict a class by comparing the distance (either Euclidean, manhattan or any assigned metric) between the K(computed) nearest data point. It performs best when the data is small. However, as the data and dimension increases, as there is hardly any difference between the nearest and farthest neighbor. Unlike

the KNN, the main model (support vector machine) uses a kernel function which projects the classes to N dimensional space, in a bid to find a best hyperplane to evenly separate classes in the dataset.

The KNN doesn't have a loss function, and hence doesn't use any optimization strategy since there is likely nothing to minimize. However, for best accuracy, an optimal value of K can be computed to find help predict new data point. Whereas, the main model (support vector machine), contains hinge loss, and employs optimization strategy such as lagrange dual function, sub gradient descent method to minimize the loss.

To have an overall view of the performance, the KNN Classifier was generated for the dataset, and its balanced accuracy score were computed, and compared with the balance accuracy score of the main model. The result is as shown in Table 3.0, and 4.0. From the result, it can be observed that the support vector models performs better than the K Nearest Neighbour Classifier. From the dataset (see fig 1d, 1e), it can be seen that the class were high, and clustered together, making it difficult for the KNN, to differentiate between nearest and farthest neighbor between each point. Also, as seen in fig(1c, 1d), it can be seen that the classes for both color and texture classification were imbalanced.

Table 3.0: A table showing the balanced accuracy of the support vector machine model (main model) and KNN Classifier on Color Classification.

MODEL	BALANCED ACCURACY SCORE
Support Vector Machine	0.2037
K Nearest Neighbor (KNN)	0.1381

Table 4.0: A table showing the balanced accuracy of the support vector machine model (main model) and KNN Classifier on Texture Classification.

MODEL	BALANCED ACCURACY SCORE
Support Vector Machine	0.2936
K Nearest Neighbor (KNN)	0.1695

QUESTION 10

The cost function of the model can be given expressed mathematically as

$$\text{Min } J(\theta) = \frac{1}{N} \sum_{l=1}^N \left(1 - y_l (X_l^T \theta + \theta_0) \right) + \lambda ||\theta||_2^2$$

Where $\left(1 - y_i (X_i^T \theta + \theta_0)\right)$ is known as the hinge loss. Which is the loss function used in the support vector machine.

Optimization strategy employed is the lagrange dual optimization strategy, which is represented as;

$$L_p = \frac{1}{2} ||\theta||^2 + c \sum_{i=1}^N \varepsilon - \sum_{i=1}^N (\alpha_i [y_i (x_i^T \theta + \theta_0) - (1 - \varepsilon_i)] - \sum_{i=1}^N \mu_i \varepsilon_i$$

Where c , ε and μ are known as the lagrange multiplier used in optimizing the model.