Report Writing

Title: Classifying the predominant colour from the `new_flag` dataset

Author's Name: Rokshana Pervin

Affiliations: RMIT University, Melbourne.

Contact details: Email: s3923324@student.rmit.edu.au

Date of report: 28/11/2021

Table of contents:

An abstract/ Introduction:

Flag plays a vital role for every country in this world. The first identity to recognize any country is flag. If the colour of the flag is not different then one country cannot be separated from another. In this case, a report is written about the `flag` dataset to classify the predominant colours of the flag which are crucial for every country and their population. Prior classifying to depth analysis on predominant colours, it is great important to pre-process the dataset to get clean and tidier once for further analysis. This report will emphasize on pre-processing the dataset and it's all features.

In initial stage, the `flag` dataset has been retrieved using the `pandas` library. Next, the data types have been checked and cleaned up using some techniques and strategies. After tidying up the `flag` dataset, a new and cleaned dataset is obtained which named as `new_flag` dataset.

After that, the `new_flag` dataset has been explored using it's features into different types of graphs where maximum features are formed their relationships with each other.

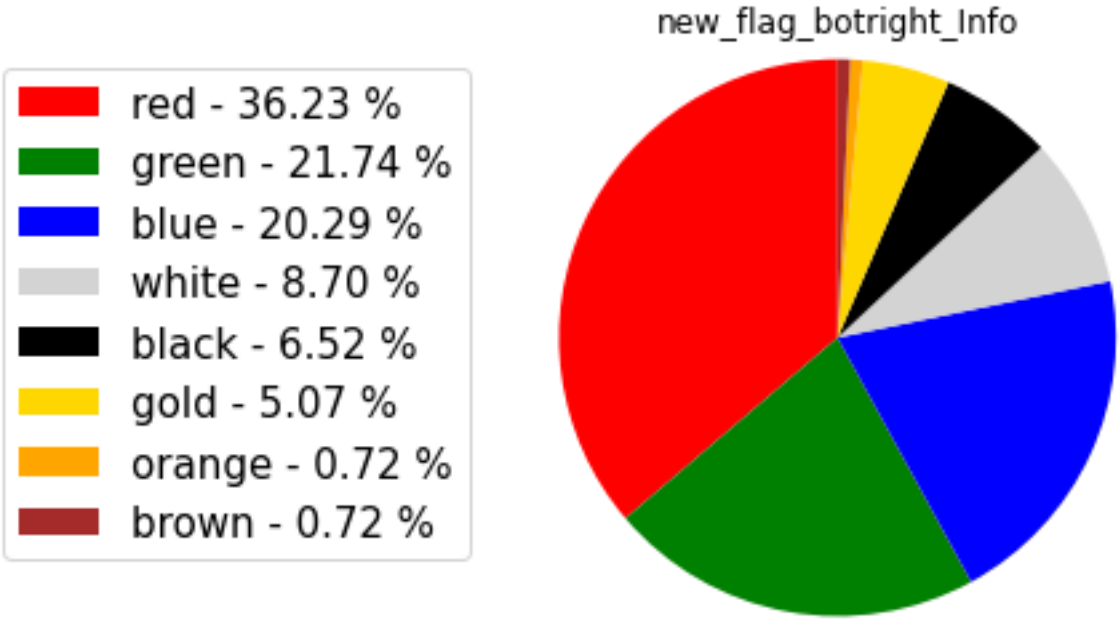
In the data modelling step, the two models have been selected known as `KNN(KNearesNeighbors) and Decision Tree model to classify the predominant colours from the `new_flag` dataset. In this step, different techniques and methods are used to create the models' accuracy adequately. Finally, the decision tree model has been visualized for the `new_flag` dataset.

Methodology:

To pre-process the `flag` dataset, few methods have been used to make the dataset nice and tidy. During this manipulation, this dataset has been retrieved using the `pandas` library and `read_csv` method. As this dataset was text able, the `read_csv` method has been applied to read the dataset with all features explicitly. In this pre-processing step, the data-types are checked using the `flag.types` command and the `flag.info()` function. After that, some data types has been changed using `astype()` function and checked their sanity and validity applying the `str.strip()`, `dropna()` and ` isnull().values.any()` functions.

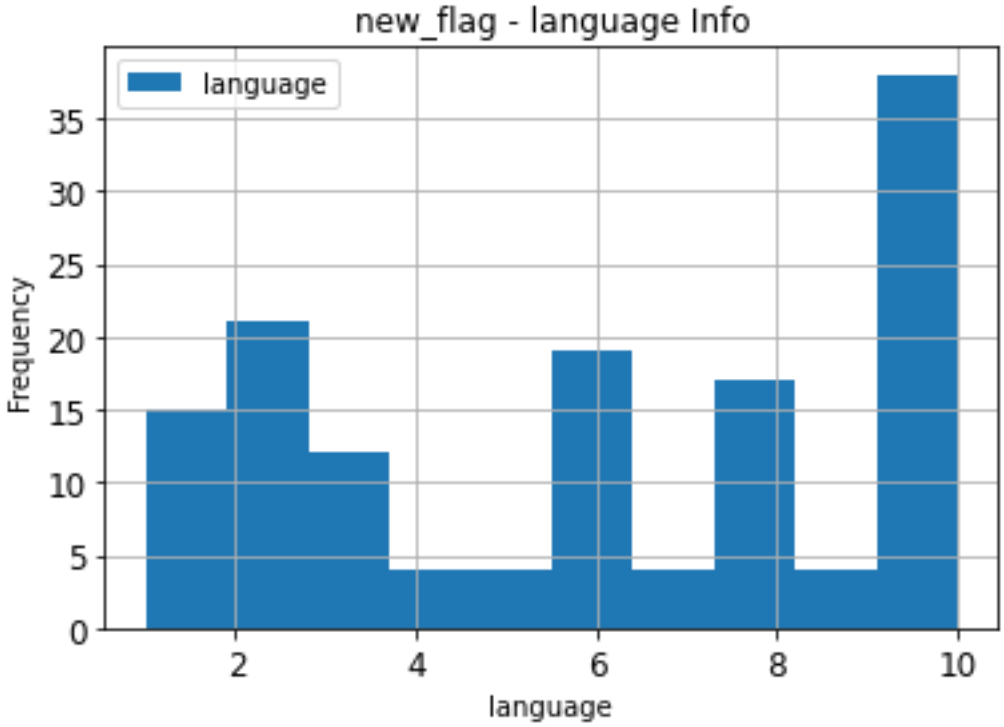
In data exploration, some libraries are used such as numpy, matplotlib and matplotlib.pyplot and seaborn to get the plots and explore the relationships of all variables appropriately. To demonstrate the variables' attributes and their relationships, pie chart, bar plot, boxplot, scatter plot and heatmap chart have been used. Here, few graphs' figures are given as below to explore the features attributes and relationships:

Pie chart:



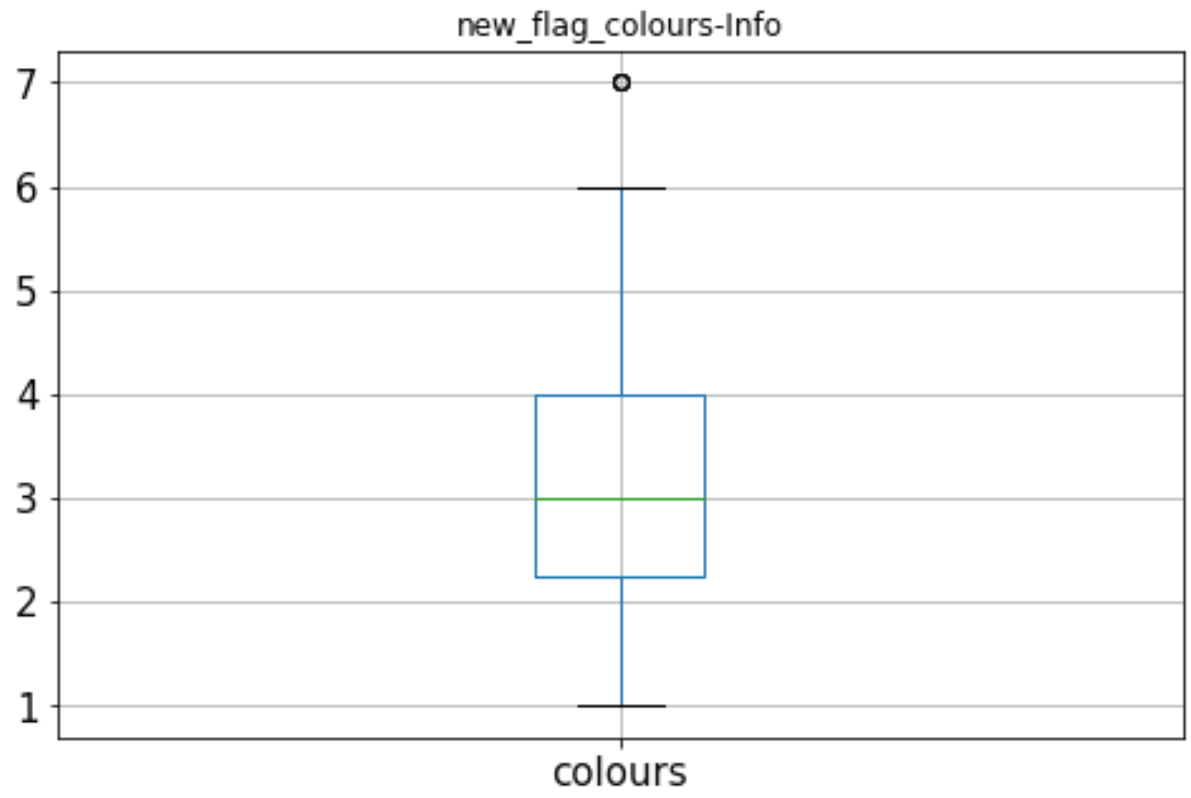
Here, this pie chart has been chosen to explore the categorical variable's statistical a nd graphical condition with proportionally which named as `botright`. The above step' s explanation is given in the jupyter notebook.

Histogram:



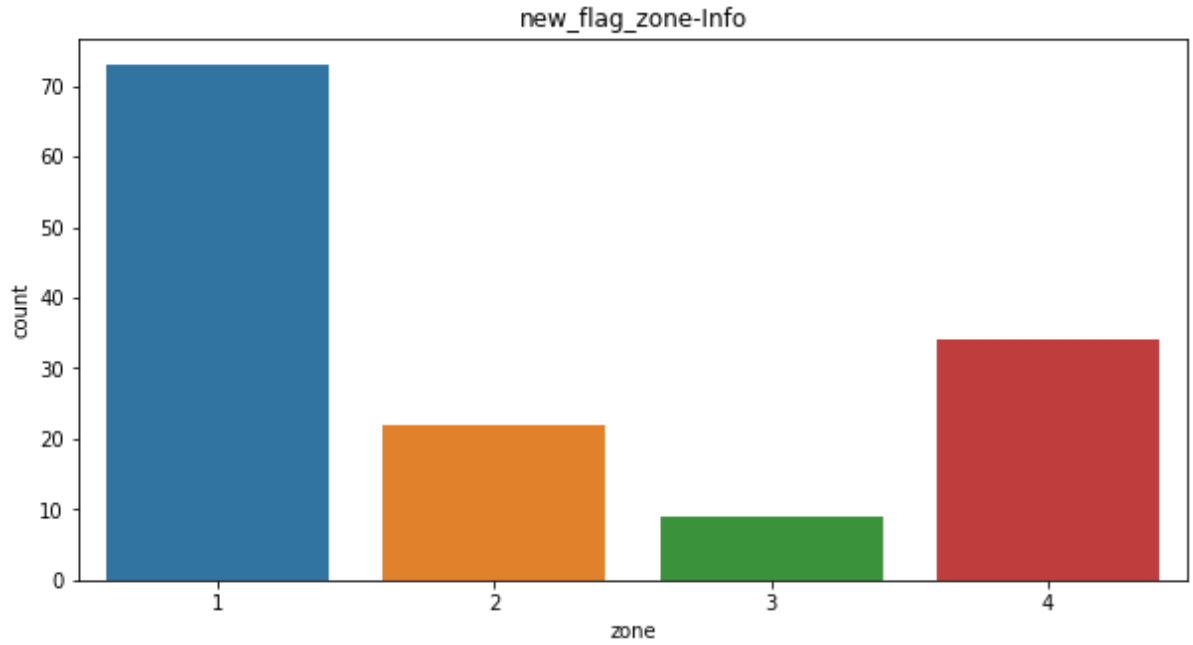
Here, this histogram chart has chosen to explore the numerical variable's statistical a nd graphical condition which named as `language`. The histogram chart is given as a bove.

Boxplot:



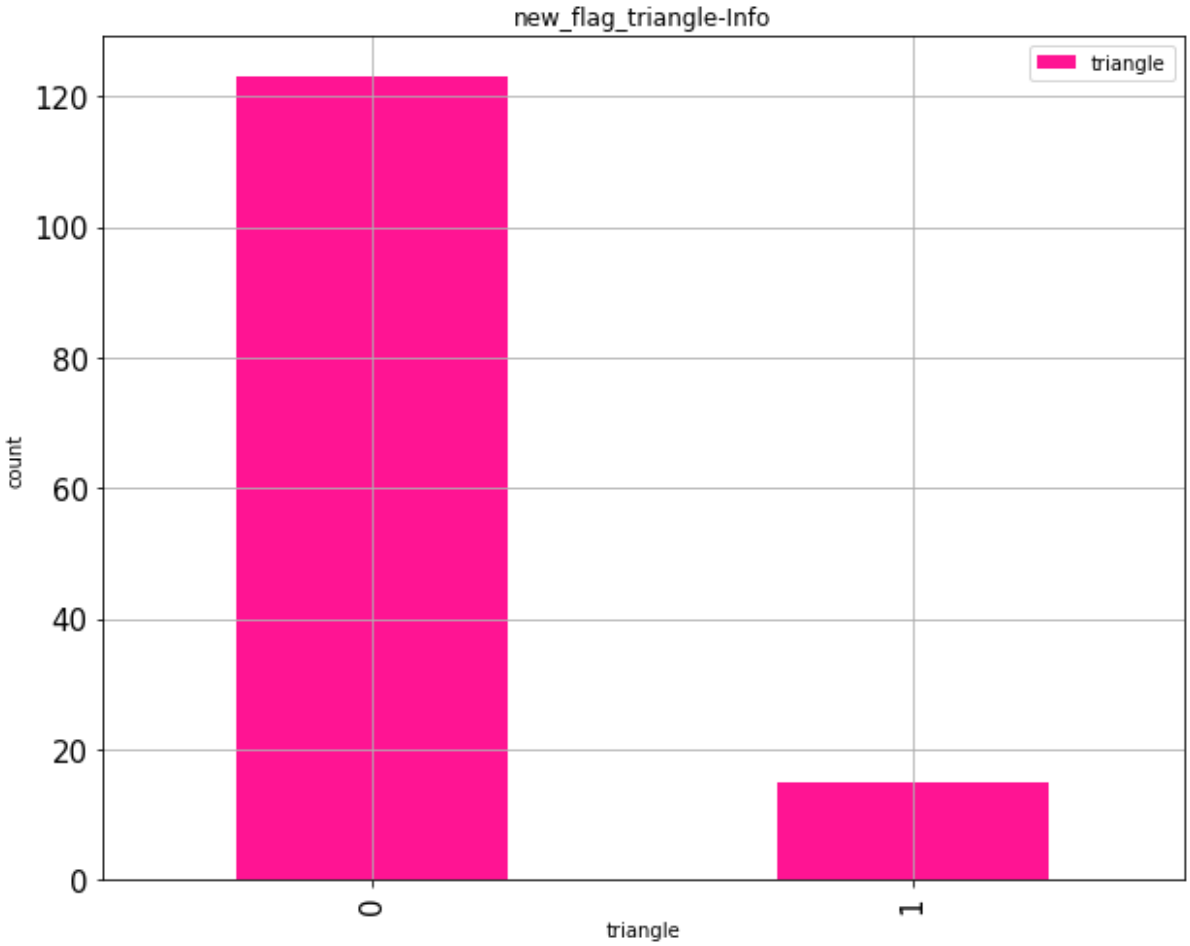
Here, the boxplot has been chosen to explore the numerical variable's key figure whi ch named as `colours` within the distribution which can also help to detect the outlier s. The boxplot figure is given above.

Count plot:



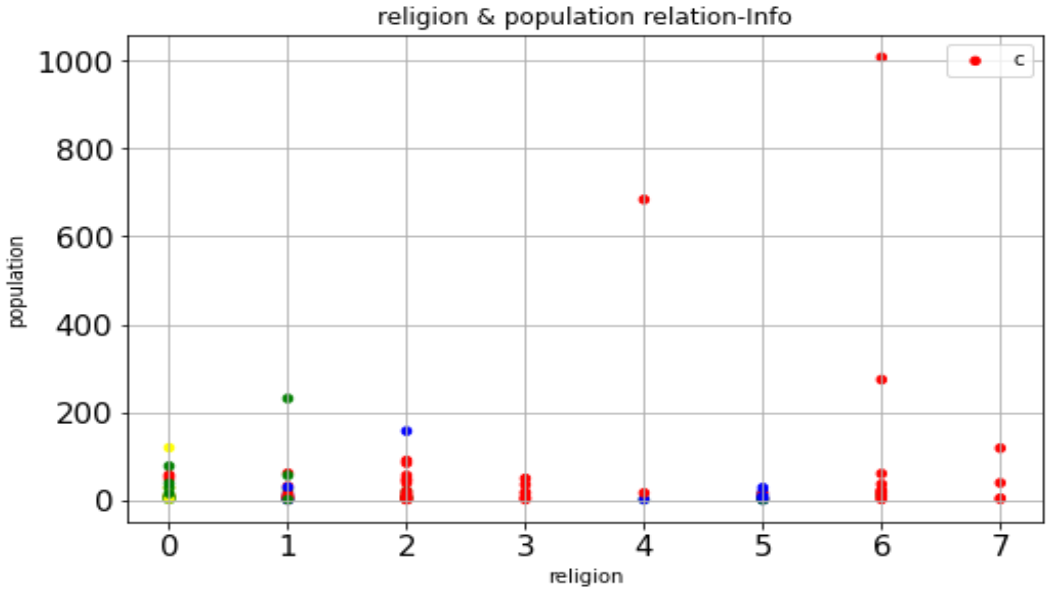
Here, this count plot has chosen to explore the numerical variable's statistical and gr aphical condition which named as `zone`. The count plot has given as above.

Bar plot:



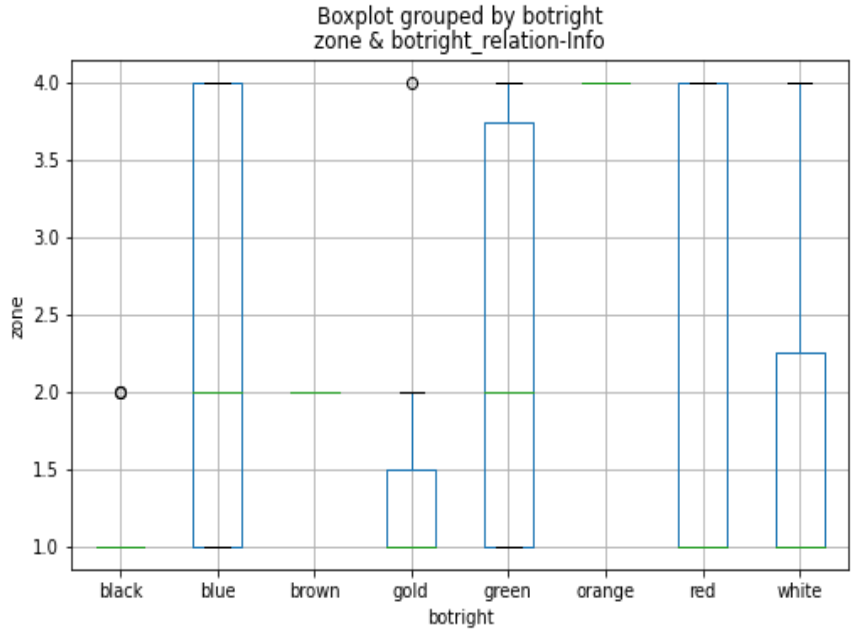
Here, this bar plot has chosen to explore the numerical variable's statistical and grap hical condition which named as `triangle`. The bar plot has given as above.

Scatter plot (Relationship with features):



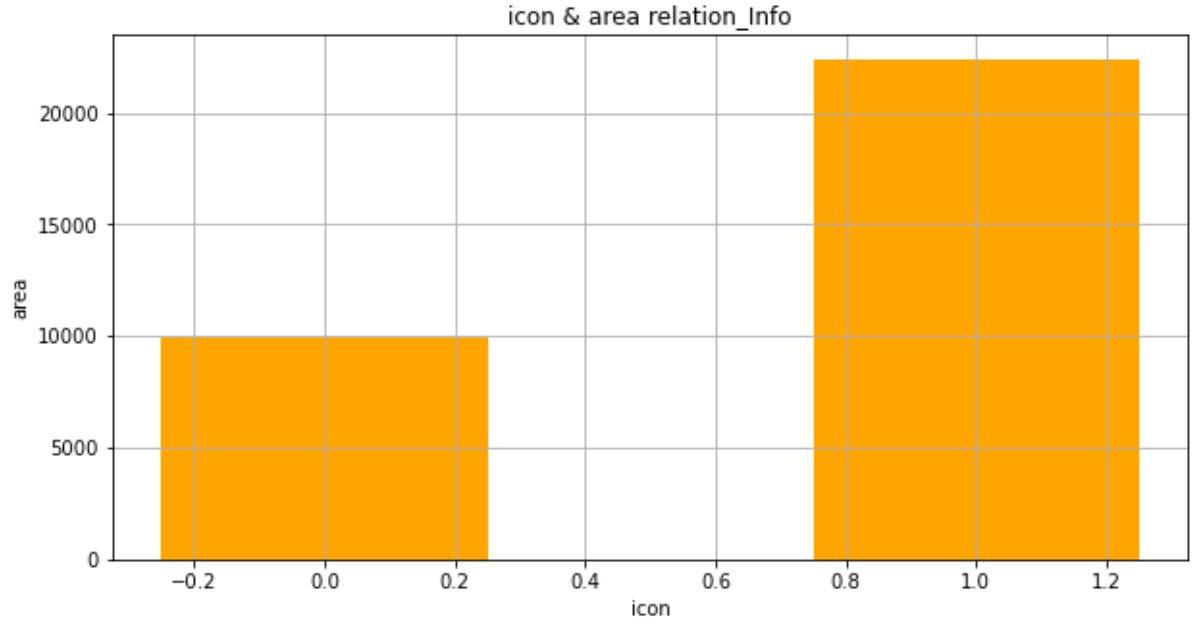
In the above scatter plot, the relationship has been found with three features such as ‘religion’, ‘population’ and ‘zone’ which is highlighting in the above scatter plot.

Box plot (Relationship with features):



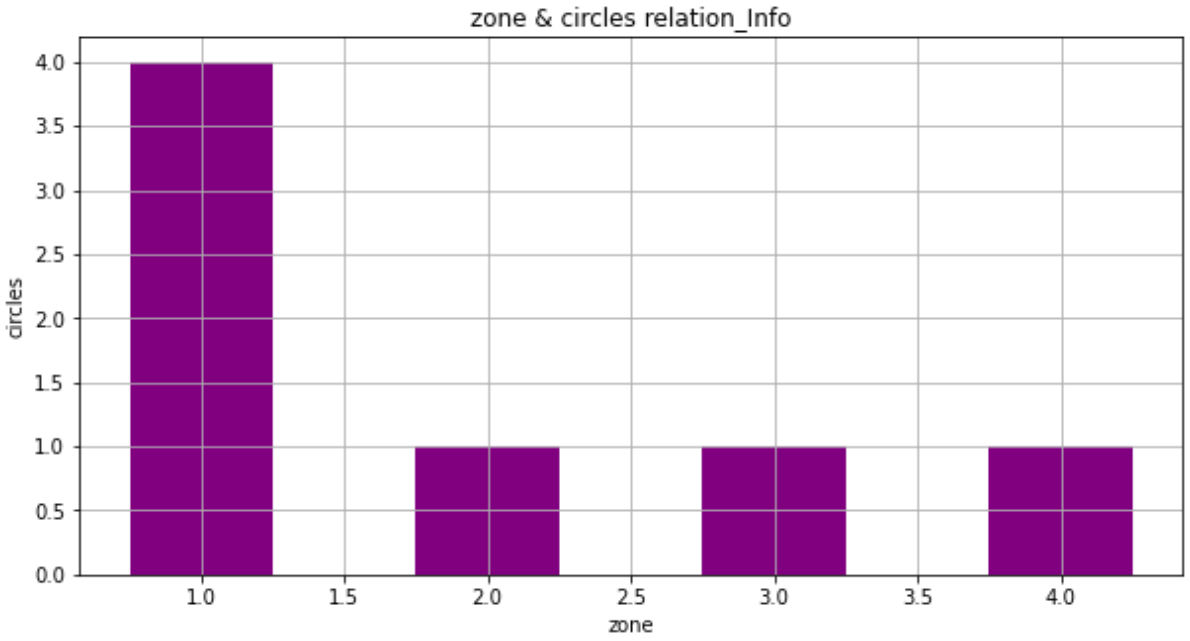
In the above box plot, the relationship has been found with two features such as ‘zon e’, and ‘botright’ which is highlighting in the above boxplot. The above step’s explana tion is given in the jupyter notebook.

Bar plot (Relationship with features):



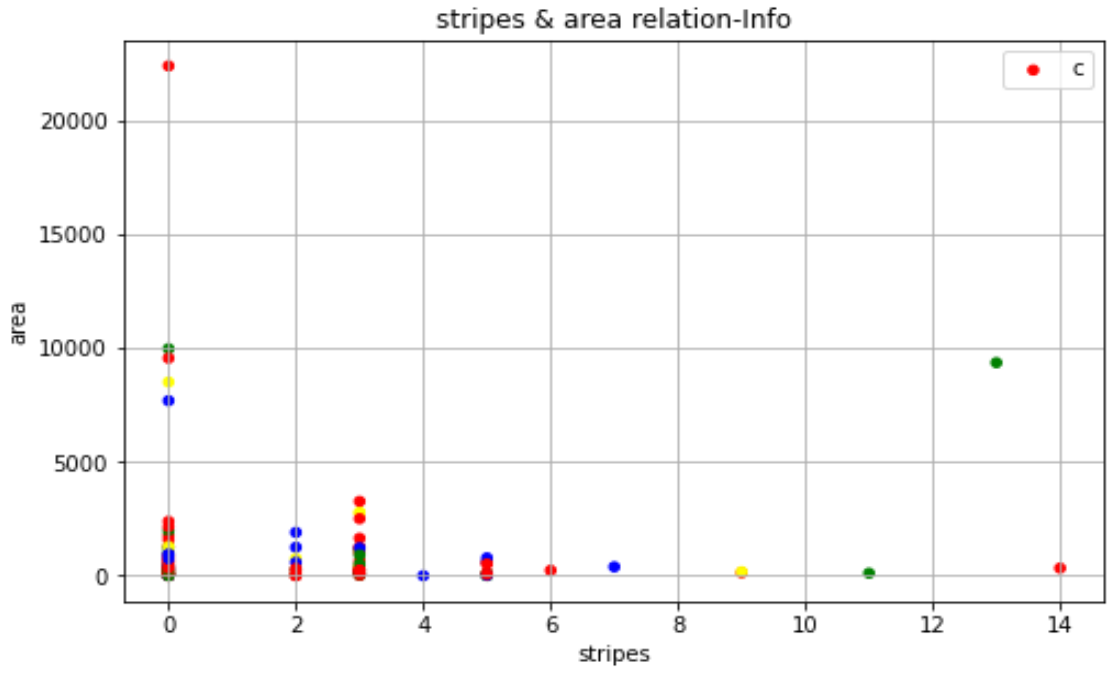
In the above bar plot, the relationship has been found with two features such as ‘icon ’, and ‘area’ which is highlighting in the above bar plot. The above step’s explanation is given in the jupyter notebook.

Bar plot (Relationship with features):



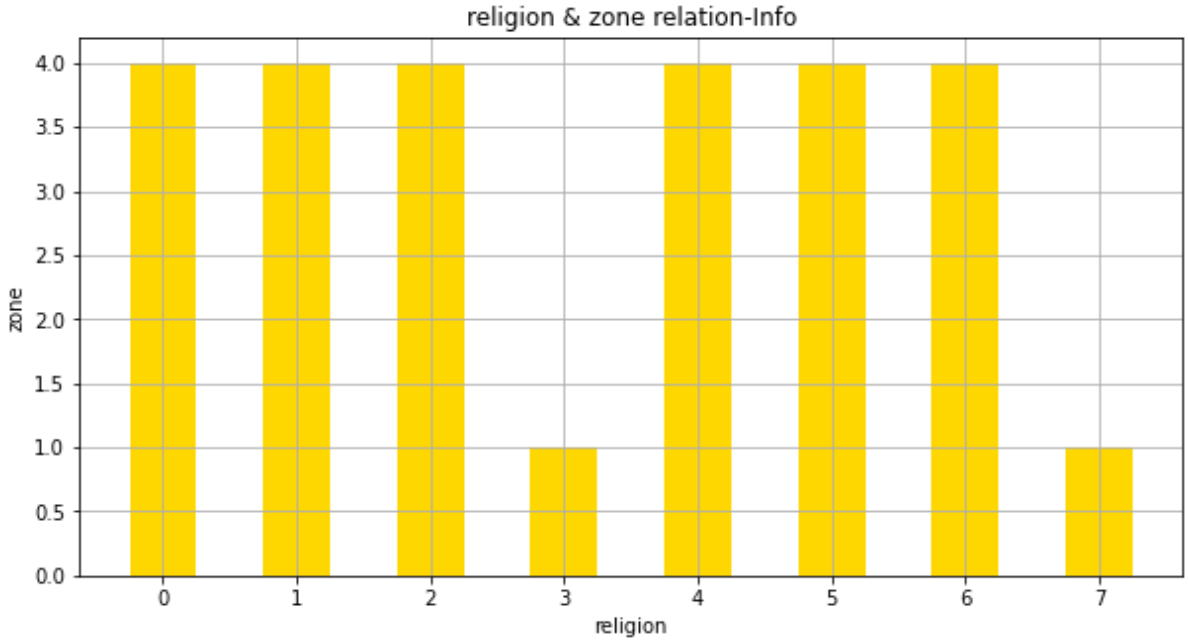
In the above bar plot, the relationship has been found with two features such as ‘zon e’, and ‘circles’ which is highlighting in the above bar plot. The above step’s explanati on is given in the jupyter notebook.

scatter plot (Relationship with features):



In the above scatter plot, the relationship has been found with three features such as ‘stripes’, ‘area’ and ‘zone’ which is highlighting in the above scatter plot.

Bar plot (Relationship with features):



In the above bar plot, the relationship has been found with two features such as ‘relig ion’, and ‘zone’ which is highlighting in the above bar plot. The above step’s explanat ion is given in the jupyter notebook.

In data modelling step, two models have been chosen to do the classification algorithm. One is KNN(KNearestNeighbors) and another one is Decision tree model. Here, few methods have been used to get the two models’(KNN and Decision tree) best accuracy. In this step, all categorical are encoded using the ` astype('category').cat.codes` function. Next, the ` train_test_split()` function is applied to split the dataset accurately. Here, target variables has been set as `mainhue` to classify the predominant colours of the flag from the `new_flag` dataset. To select the KNearestNeighbors and evaluate the model adequately, the `KNeighborsClassifier()` function has been used. Apart from this, fitted the data using the '.fit()' function and predicted the data appropriately. Here, predicted on unseen data using the `predict()` function which is added with the `fit` object and also created another `object` as `predicted`. The ` confusion_matrix()` function has been used to test the accuracy where precision, recall and f1-score are measured accurately. During this operation, the classification error rate has been found using the `Elbow method`. To get the accuracy, the dataset has been splitted for three times such as 50% and 50% for training and test, 60% and 40% for training and test,

and 80% and 20% for training and test. But, unfortunately, there were no good accuracy rather than acquired high classification error rate. Therefore, another model is chosen and tested the accuracy of this model which known as `Decision Tree` model. Here, same techniques and process are applied which are also done for `KNN` model in the above description. Only, the `default parameters` had to take instead of `K-value` which are generated the decision tree model's accuracy and performance very high.

Here, to generate the decision tree and evaluate the model adequately, the DecisionTreeClassifier library has been imported from the sklearn.tree package. Here, the parameters have been added for decision tree classifier with gini criterion and parallelly other parameters have been checked to change their values which results the decision tree model's accuracy down. But, as the default parameters are generated the decision tree models accuracy is high, the default parameters are used in here appropriately. During this operation, the DecisionTreeClassifier() function has been used and clf is created as object to explore the DecisionTreeClassifier value adequately. The default parameters such as criterion='gini', max_depth=None, min_samples_split=2, min_samples_leaf=1, max_features=None, and max_leaf_nodes=None have been explored using the `DecisionTreeClassifier()` function which are created the best accuracy for the decision tree model. Finally, the decision tree model has been visualized using this function, with open("new_flag.dot","w") as f:

```
 f = tree.export_graphviz(clf,
                          out_file=f,
                         feature_names=feature_names,
                         class_names=target_names,
                          filled=True,
                           rounded=True,
                          special_characters=True)
```

where predominant colors and their attributes have been explored adequately.

Results:

In this step, for 50% and 50% splitting,

The `KNN` model's classification error rate is obtained by subtracting the accuracy result with `1` and the result has been found as `0.6521739130434783` whereas the decision tree models classification error rate is obtained as 0.08695652173913049.

In this step, for 60% and 40% splitting,

The `KNN` model's classification error rate is obtained by subtracting the accuracy re sult with `1` and the result has been found as ` 0.7321428571428572` whereas the d ecision tree models classification error rate is obtained as 0.0892857142857143.

In this step, for 80% and 20% splitting,

The `KNN` model’s classification error rate is obtained by subtracting the accuracy result with `1` and the result has been found as ` 0.8214285714285714 ` whereas the decision tree models classification error rate is obtained as ` 0.1071428571428571.
After completing these above operations, it is also observed that in 50% training and 50% testing splitting process, the KNN model's accuracy is 0.35 whereas the Decision Tree model's accuracy is 0.90 In 60% training and 40% testing splitting process, the `KNN` model's accuracy is 0.27 whereas the `Decision Tree` model's accuracy is 0.91 In 80% training and 20% testing splitting process, the `KNN` model's accuracy is 0.18 whereas the Decision Tree model's accuracy is 0.89.

Discussion:

In the modelling step, the classification error rate was very high which is made the `KNN(KNearestNeighbors)` model’s performance very low. On the other hand, it is observed that the decision tree model is highlighting very good performance with its accuracy after splitting three times with different percentages. Apart from, it is also analysed that, the KNN model has been splitted three times with different percentages but it doesn’t get any better accuracy rather than the classification error rate high. During, these operations times, K value has been changed many times from 1 to 18 to get the good result of it’s accuracy whereas only using the default the parameters the decision tree model has shown it’s better performance and accuracy.

Conclusion:

In conclusion, it is represented that the two (KNN and Decision Tree) models' accuracy have been compared which are done in the above steps such as in 50% training and 50% testing splitting process, the KNN model's accuracy is 0.35 whereas the Decision Tree model's accuracy is 0.90 In 60% training and 40% testing splitting process, the KNN model's accuracy is 0.27 whereas the Decision Tree model's accuracy is 0.91 In 80% training and 20% testing splitting process, the KNN model's accuracy is 0.18 whereas the Decision Tree model's accuracy is 0.89.
Here, it may decide that as the KNN model's accuracy is lower than the Decision tree model, therefore, the Decision Tree model would be perfect to use in the real world for the particular new_flag dataset.
After completing above all tasks, it is classified that the predominant colours of the flag has been demonstrated and visualised with the decision tree model from the particular `new_flag` dataset which can obtained from the jupyter notebook.