

Cats and Dogs Image Classification Using Machine Learning with Python

The Ultimate Guide for building Binary Image Classifier by applying Image Analysis and Pre-processing techniques on unseen photos of Dogs and Cats.



**Blog By: Chaithanya Vamshi Sai**

**Student Id: 21152797**

# **Introduction**

There’s a strong belief that when it comes to working with unstructured data, especially image data, deep learning models are the best. Deep learning algorithms undoubtedly perform extremely well, but is that the only way to work with images?

Have you ever wondered what if there was a way, we could classify the images of cats and dogs using Machine Learning? That would be great right?

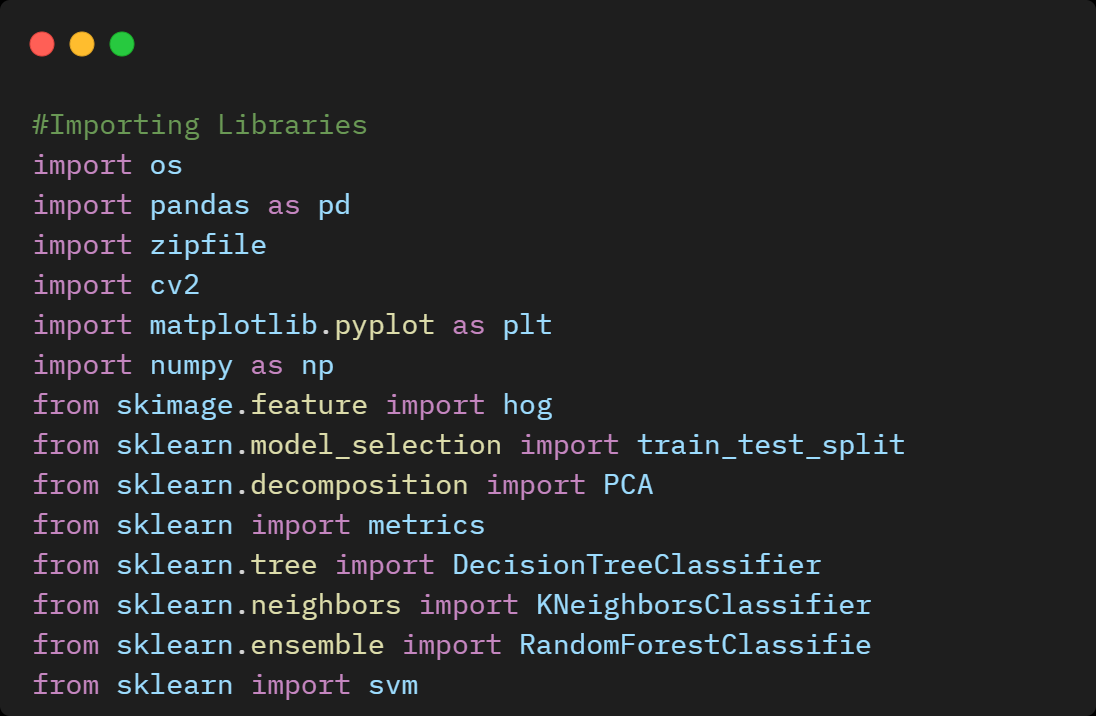
If we provide the right data and features, Machine learning models can perform adequately well and can even be used as an automated solution.

In this blog, I will demonstrate and show how we can harness the power of Machine Learning and perform image classification using four popular machine learning algorithms like Random Forest Classifier, K-Nearest Neighbour (KNN) Classifier, Decision Tree Classifier, and Support Vector Machine (SVM).

# **Problem Statement**

The main objective of this task is to apply Machine learning algorithms to build Binary Image Classifier by applying Image Analysis and Pre-processing techniques and making predictions on unseen photos of Dogs and Cats.

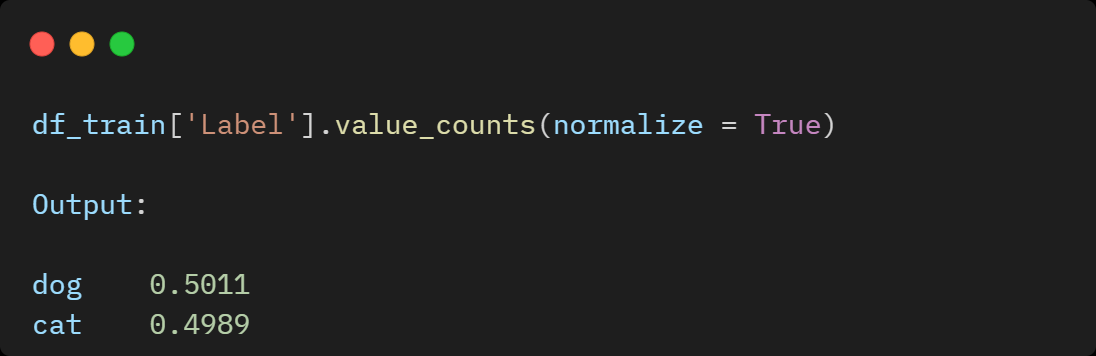
# **Importing Libraries**

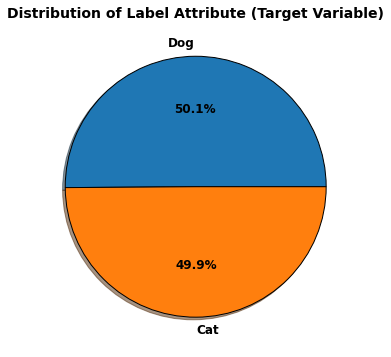


# **Unzipping and Reading Train and Test Dataset**

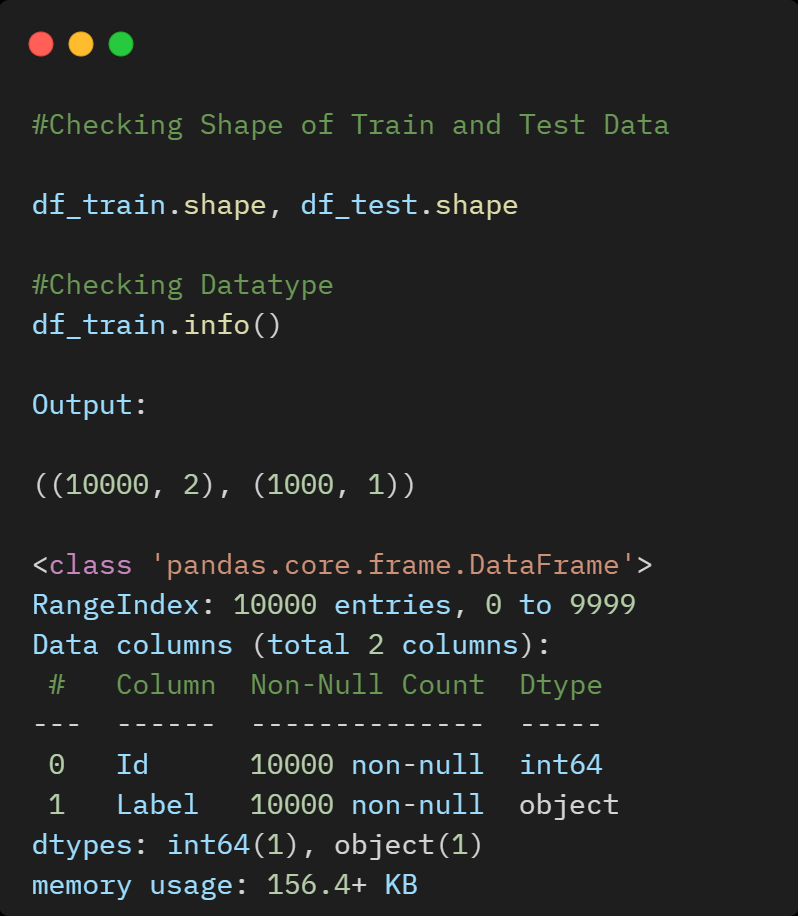


# **Checking Distribution of Label (Target Variable) in Train Data**



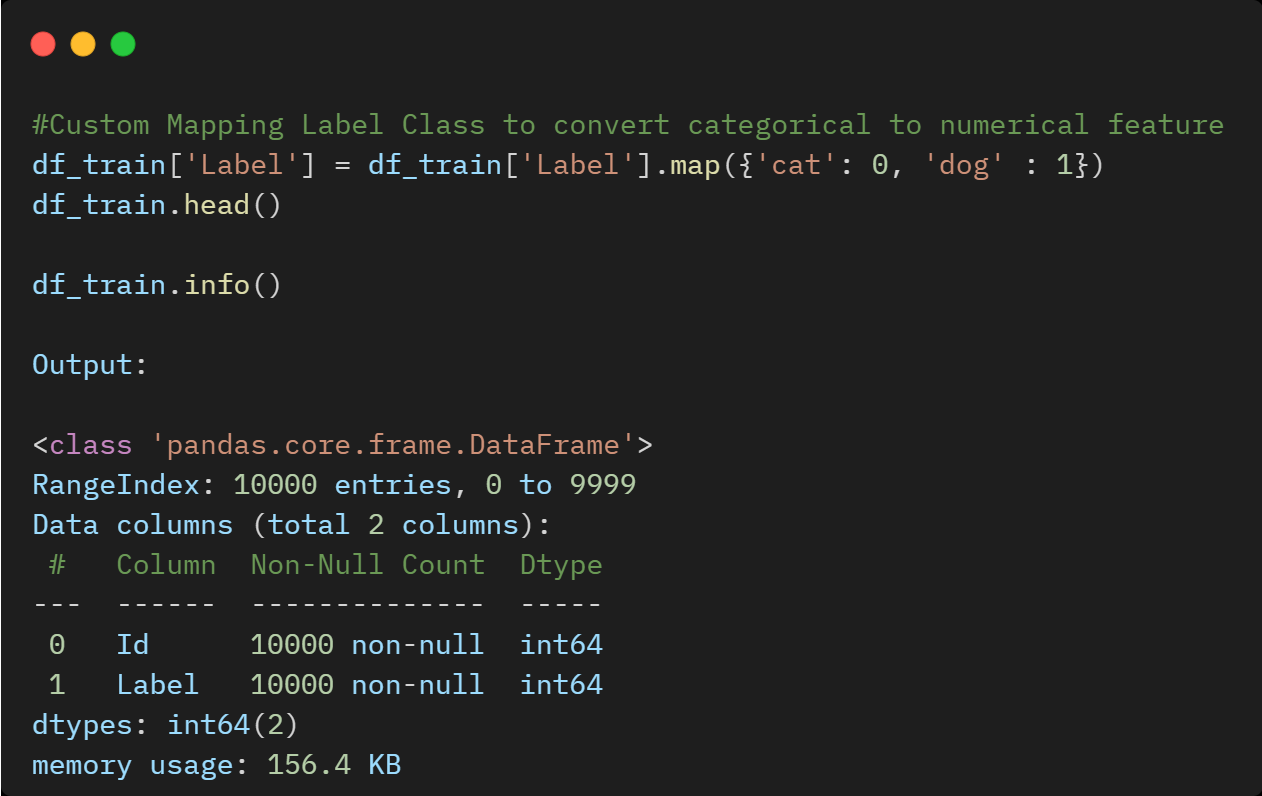


# **Checking Shape and Data Types of Train and Test Data**

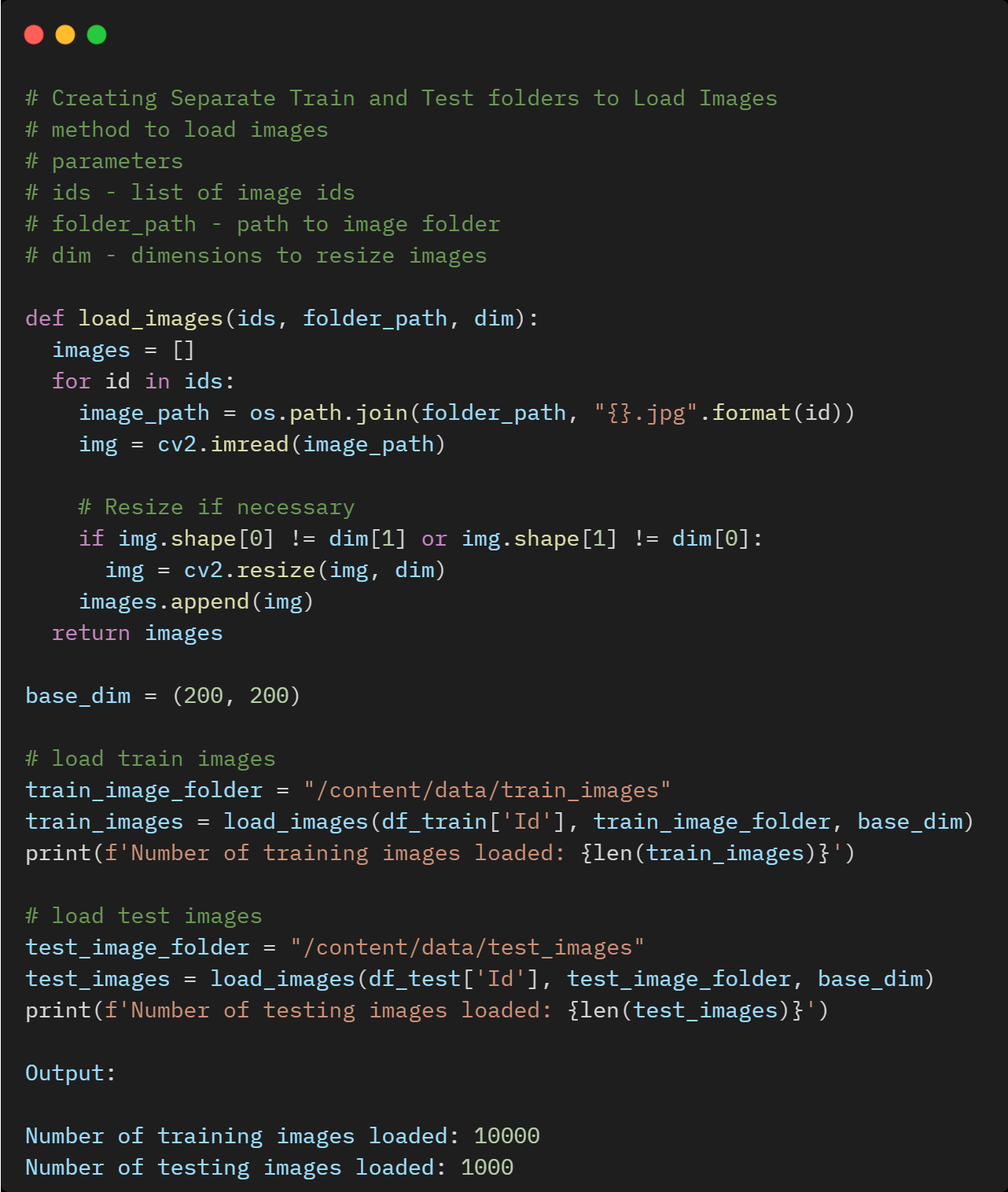


# **Feature Encoding – Custom Mapping “Label” Target Attribute**

To apply Machine Learning algorithms, we need to encode the features from categorical to numerical data types. Hence, I will apply one of the Feature Encoding methods on the “Label” target column by applying Custom Mapping labelling Cat as 0 and Dog as 1.



# **Creating Separate Train and Test folders to Load Images**



# **Image Pre-processing on Train Data**

## **Image Manipulation**

* **Gray Scaling**: Grayscale is a range of monochromatic shades from black to white. Gray scaling in converting images to grayscale.
* **Resizing:** Image resizing refers to either enlarging or shrinking images. Machine learning algorithms require the same size images during the learning and prediction phases. To convert all images to a common size, we need to define a base image size and resize images.
* **Smoothing/Blurring:** Smoothing is commonly used for noise reduction in images. It reduces irrelevant details such as pixel regions that are small for the filter kernel size. In this task, I have applied the Gaussian Filtering technique which assigns the Gaussian weighted average of all the pixels under the kernel area as the central element value.

## **Feature Extraction**

* **Image Vectorisation:** Method to convert an image to a vector is matrix vectorisation. Colour image vectorisation results in very long vectors which leads to a curse of dimensionality. Hence, the simplest solution is image gray scaling when compared to vectorisation.
* **Edge Detection:** Edge detection is an image processing technique that finds the boundaries of objects within images. In this task, I have applied Canny edge detection is an optimal algorithm for edge detection which helps in Noise reduction, finding intensity gradient of the image, filters out non-real edges and removing pixels that may not constitute edges
* **HOG Feature Descriptor:** Histogram of Oriented Gradients feature descriptor counts the occurrences of gradient orientation in localised portions of an image as to features. It mainly focuses on the shape or the structure of objects.
* **Principle Component Analysis (PCA)**: PCA is a dimensionality reduction technique that uses Singular Value Decomposition of the data to project it to a lower-dimensional space.

# **Model Building**

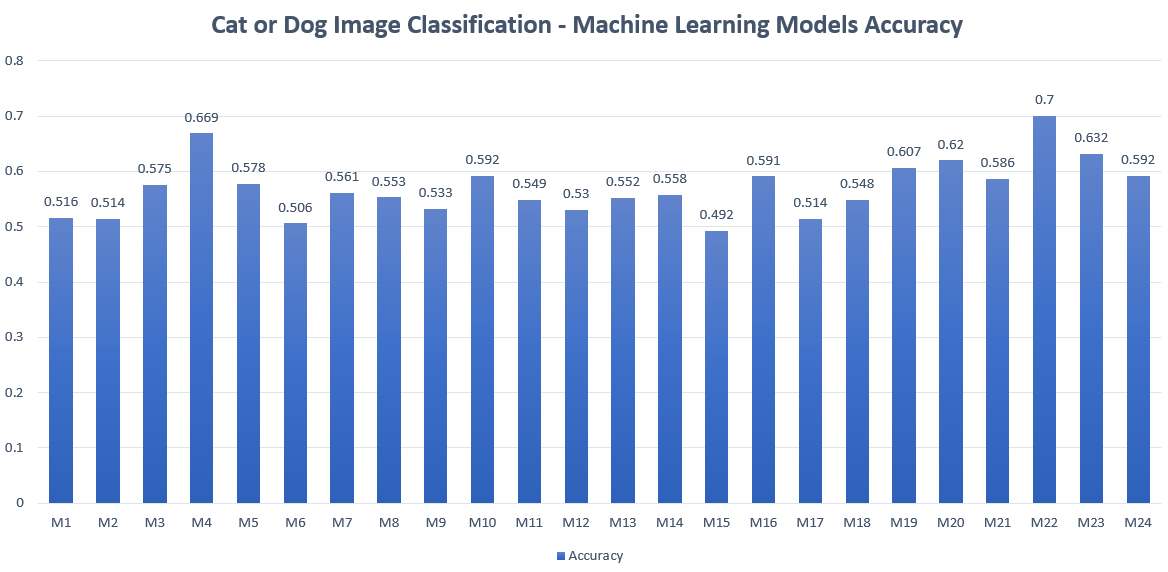
Python Implementation Source code of Machine Learning Algorithms is available on [**GitHub**](https://github.com/ChaithanyaVamshi/Binary_Image_Classification_ML/blob/main/Image_Classification.ipynb)

1. Support Vector Machine (SVM) Classifier (M1 - M6)
2. Decision Tree Classifier (M7 - M12)
3. K-Nearest Neighbour (KNN) Classifier (M13 - M18)
4. Random Forest Classifier (M19 - M24)

## **Summary of Machine Learning Model Accuracy on Train and Validation Data**

From all the Machine Learning Models, **Random Forest Classifier (M22)** with Gray scaling and HOG Feature has achieved the best Accuracy of **70%** followed by **Support Vector Machine (SVM) Classifier** with Gray scaling and HOG Feature with **67%** Accuracy.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ML Algorithm** | **Model** | **Pre-processing** | **Feature Extraction** | **Accuracy** |
| Support Vector Machine | M1 | Gray Scaling | Image Vector | 0.516 |
| Support Vector Machine | M2 | Gray Scaling + Gaussian Filtering | Image Vector | 0.514 |
| Support Vector Machine | M3 | Gray Scaling | Canny Edge Detection | 0.575 |
| Support Vector Machine | M4 | Gray Scaling | HOG Feature | 0.669 |
| Support Vector Machine | M5 | Gray Scaling | Image Vector + Canny Edge Detection | 0.578 |
| Support Vector Machine | M6 | Gray Scaling | Image Vector + PCA | 0.506 |
| Decision Tree Classifier | M7 | Gray Scaling | Image Vector | 0.561 |
| Decision Tree Classifier | M8 | Gray Scaling + Gaussian Filtering | Image Vector | 0.553 |
| Decision Tree Classifier | M9 | Gray Scaling | Canny Edge Detection | 0.533 |
| Decision Tree Classifier | M10 | Gray Scaling | HOG Feature | 0.592 |
| Decision Tree Classifier | M11 | Gray Scaling | Image Vector + Canny Edge Detection | 0.549 |
| Decision Tree Classifier | M12 | Gray Scaling | Image Vector + PCA | 0.53 |
| KNN Classifier | M13 | Gray Scaling | Image Vector | 0.552 |
| KNN Classifier | M14 | Gray Scaling + Gaussian Filtering | Image Vector | 0.558 |
| KNN Classifier | M15 | Gray Scaling | Canny Edge Detection | 0.492 |
| KNN Classifier | M16 | Gray Scaling | HOG Feature | 0.591 |
| KNN Classifier | M17 | Gray Scaling | Image Vector + Canny Edge Detection | 0.514 |
| KNN Classifier | M18 | Gray Scaling | Image Vector + PCA | 0.548 |
| Random Forest Classifier | M19 | Gray Scaling | Image Vector | 0.607 |
| Random Forest Classifier | M20 | Gray Scaling + Gaussian Filtering | Image Vector | 0.62 |
| Random Forest Classifier | M21 | Gray Scaling | Canny Edge Detection | 0.586 |
| Random Forest Classifier | M22 | Gray Scaling | HOG Feature | 0.7 |
| Random Forest Classifier | M23 | Gray Scaling | Image Vector + Canny Edge Detection | 0.632 |
| Random Forest Classifier | M24 | Gray Scaling | Image Vector + PCA | 0.592 |



## **Model Evaluation of Training and Validation Data Using Confusion Matrix**

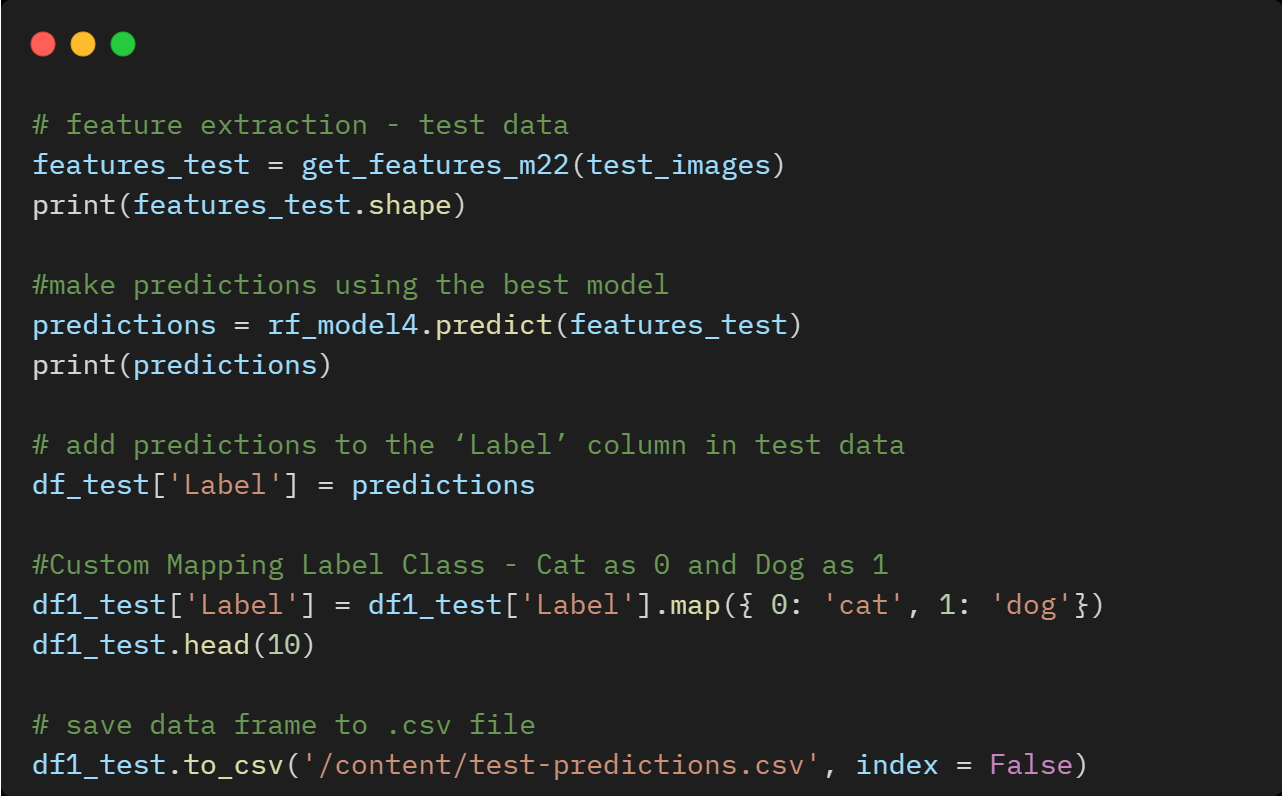
|  |  |
| --- | --- |
| **M1** | **M2** |
|  |  |
| **M3** | **M4** |
|  |  |
| **M5** | **M6** |
|  |  |
| **M7** | **M8** |
|  |  |

|  |  |
| --- | --- |
| **M9** | **M10** |
|  |  |
| **M11** | **M12** |
|  |  |
| **M13** | **M14** |
|  |  |
| **M15** | **M16** |
|  |  |

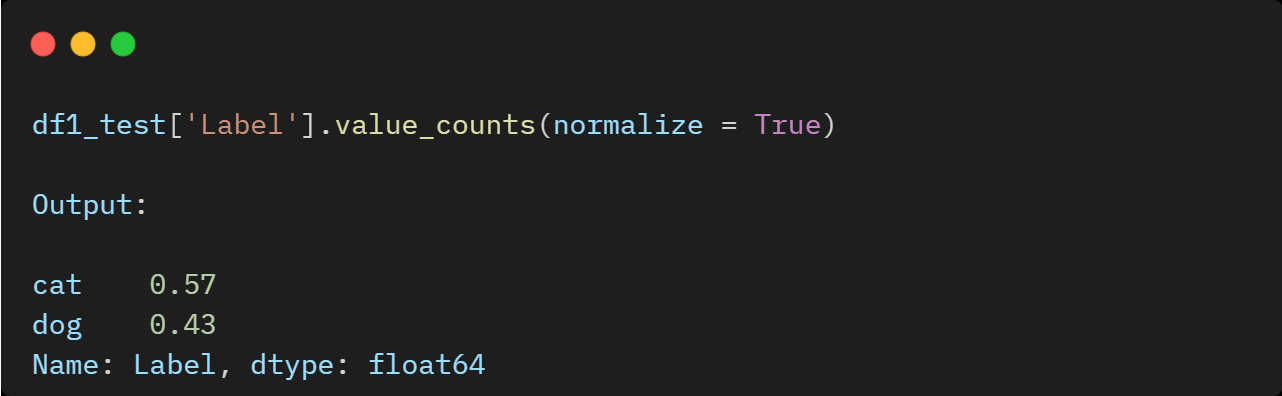
|  |  |
| --- | --- |
| **M17** | **M18** |
|  |  |
| **M19** | **M20** |
|  |  |
| **M21** | **M22** |
|  |  |
| **M23** | **M24** |
|  |  |

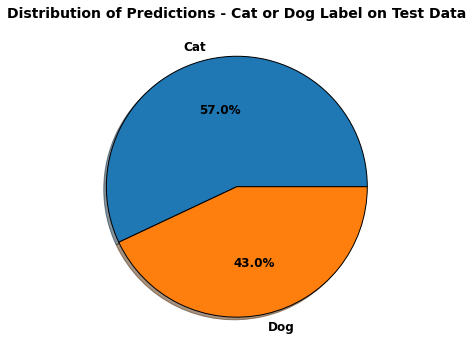
# **Predictions on Test Dataset**

Using the best ML Model Random Forest Classifier (M22), we will make predictions on test data and save predictions on .csv file name “test-predictions.csv”



## **Distribution of Predictions - Label Class on Test Data**





# **Conclusion**

In this blog, we discussed how to approach the binary Image classification problem by implementing four ML algorithms including Random Forest, KNN, Decision Tree, and Support Vector Machines. Out of these 4 ML Algorithms, Random Forest Classifier with HOG Feature extraction shows the best performance with 70% accuracy.

We can explore this work further by trying to improve the Machine Learning Image classification using hyperparameter tuning and Deep learning algorithms.

# **References**

Brownlee, J., 2020a. Parametric and Nonparametric Machine Learning Algorithms. [online] Machine Learning Mastery. Available at: [https://machinelearningmastery.com/parametric-andnonparametric-machine-learning-algorithms/](https://machinelearningmastery.com/parametric-andnonparametric-machine-learning-algorithms/%20)

Dataturks, 2018. Understanding Svms’: For Image Classification. [online] Medium. Available at: <https://medium.com/@dataturks/understanding-svms-for-image-classification-cf4f01232700>

Navlani, A., 2019. (Tutorial) Support Vector Machines (SVM) In Scikit-Learn. [online] DataCamp Community. Available at[: https://www.datacamp.com/community/tutorials/svm-classification-scikitlearn-python](:%20https:/www.datacamp.com/community/tutorials/svm-classification-scikitlearn-python%20)

Srivastava, T., 2018. K Nearest Neighbor | KNN Algorithm | KNN In Python & R. [online] Analytics Vidhya. Available at: [https://www.analyticsvidhya.com/blog/2018/03/introduction-k-neighboursalgorithm-clustering/](https://www.analyticsvidhya.com/blog/2018/03/introduction-k-neighboursalgorithm-clustering/%20)