

# Documentation of Machine Learning Project

**hackerearth**

Machine Learning challenge

-By Pritimoy Sarkar

# Adopt a Buddy



Pritimoy Sarkar

From Guru Nanak Institute of Technology

❖ About the Dataset .....	2
❖ Data Pre-processing .....	3
• Redundant feature removal .....	3
• Univariate Analysis .....	3
• Bivariate Analysis .....	7
• Class Imbalance .....	7
❖ Model Building .....	9
• Multioutput Regression Method .....	9
▪ Logistic Regression Model .....	9
▪ K Nearest Neighbors Model .....	10
▪ Decision Tree Regressor Model .....	11
• Chained Multioutput Regression Method .....	12
▪ Gaussian Naive Bayes .....	12
▪ Linear Support Vector Machine .....	13
▪ Decision Tree Classifier .....	13
▪ Logistic Regression .....	14
▪ Random Forest Classifier .....	14
▪ Multi-Layer Perception Classifier .....	15
❖ Performance Analyzing.....	16
• Multioutput Regression Method (Chart 1) .....	16
• Chained Multioutput regression (Chart 2).....	17
• Decision Tree Classifier (Performance report label 2 output).....	18
• Compare prediction accuracy for both of the output feature .....	19
❖ At a glance .....	20
❖ Thank You and personal details.....	21

## Features

The dataset provided by Hackerearth had 18834 instances and 11 columns. Among those 11 columns 9 column was supposed to be input feature and 2 columns was supposed to be label

- Input Features:
  - 1) Pet\_id
  - 2) Issue\_date
  - 3) Listing\_date
  - 4) Condition
  - 5) Color\_type
  - 6) Length(m)
  - 7) Height (cm)
  - 8) X1
  - 9) X2
- Output Feature (label):
  - 1) Breed\_category
  - 2) Pet\_category

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 18834 entries, 0 to 18833
Data columns (total 11 columns):
#   Column                Non-Null Count  Dtype
---  ---
0   pet_id                18834 non-null  object
1   issue_date            18834 non-null  object
2   listing_date          18834 non-null  object
3   condition             17357 non-null  float64
4   color_type            18834 non-null  object
5   length(m)             18834 non-null  float64
6   height(cm)            18834 non-null  float64
7   X1                    18834 non-null  int64
8   X2                    18834 non-null  int64
9   breed_category        18834 non-null  float64
10  pet_category          18834 non-null  int64
dtypes: float64(4), int64(3), object(4)
memory usage: 1.6+ MB
None
```

**18834** instances with 11 Colum (9 features 2 label)

```
pet_id                0
issue_date            0
listing_date          0
condition             1477
color_type            0
length(m)             0
height(cm)            0
X1                    0
X2                    0
breed_category        0
pet_category          0
dtype: int64
```

**1477** NaN value in 'condition' feature

## Redundant feature removal

Redundant features like

- 1) Pet\_id
- 2) Issue\_date
- 3) Listing\_date

Was removed, as these features were irrelevant to predict the output in our case

	condition	color_type	length(m)	height(cm)	X1	X2	breed_category	pet_category
0	2.0	Brown Tabby	0.80	7.78	13	9	0.0	1
1	1.0	White	0.72	14.19	13	9	0.0	2
2	NaN	Brown	0.15	40.90	15	4	2.0	4
3	1.0	White	0.62	17.82	0	1	0.0	2
4	2.0	Black	0.50	11.06	18	4	0.0	1

The dataset, after removing redundant features

From Now on, further work will be done on these 6 input feature and 2 output feature (label) would

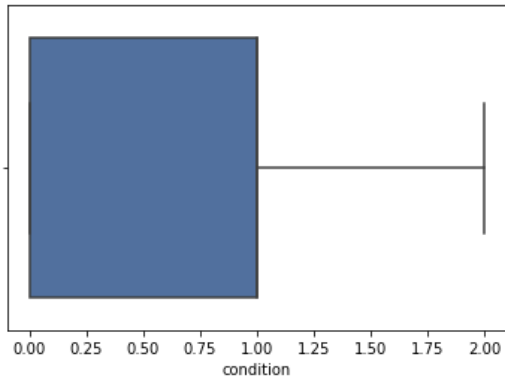
## Univariate Analysis

After Numeric Encoding of classification features NaN value treatment and skewness and outliers checking and removal was done

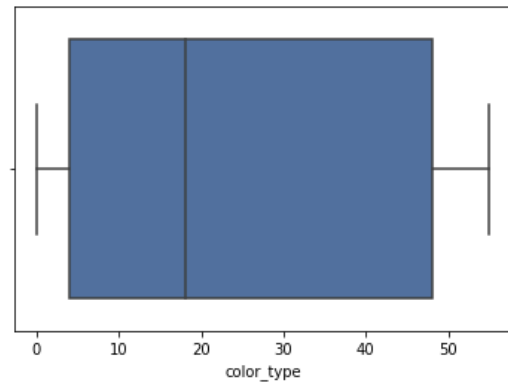
Below we will check some box plot to visualize any existence of outliers:

# Data pre-processing

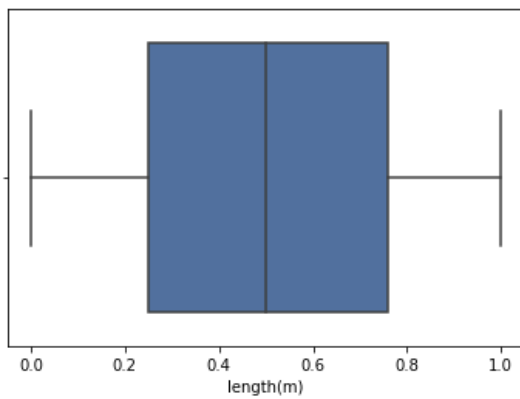
4



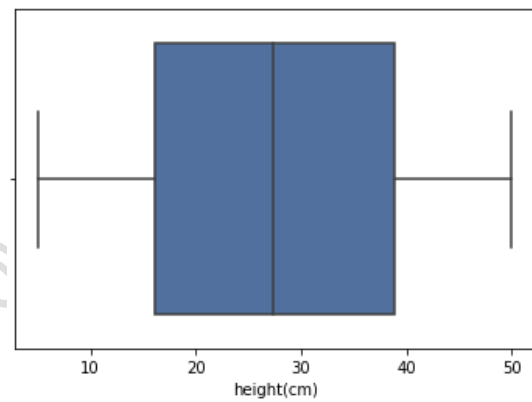
Boxplot of 'Condition' colum



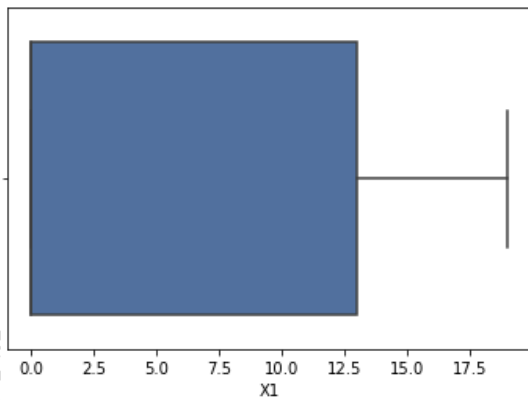
Boxplot of 'color\_type' colum



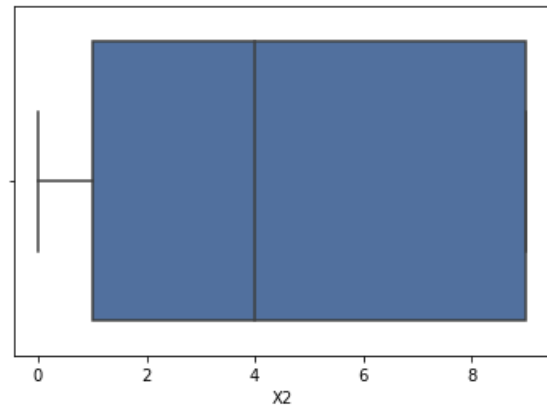
Boxplot of 'length(m)' colum



Boxplot of 'height(cm)' colum



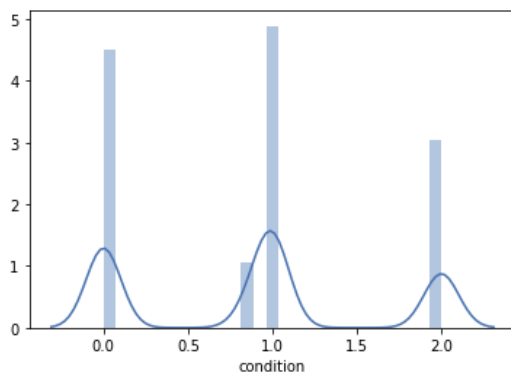
Boxplot of 'x1' colum



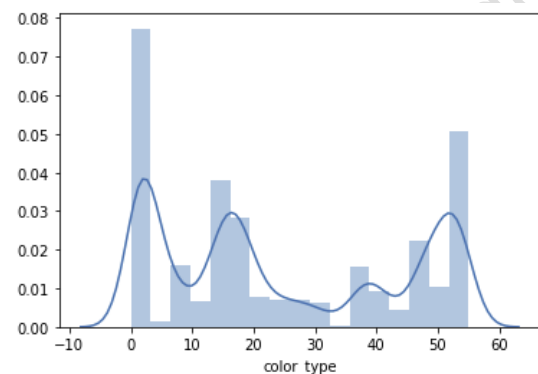
Boxplot of 'x2' colum

As we can see there is no outliers in any of the 6 features, we will move forward to check skewness of the input features

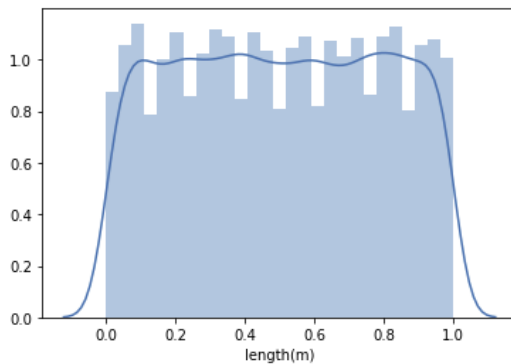
**Below we will check some distribution plot to visualize any existence of outliers:**



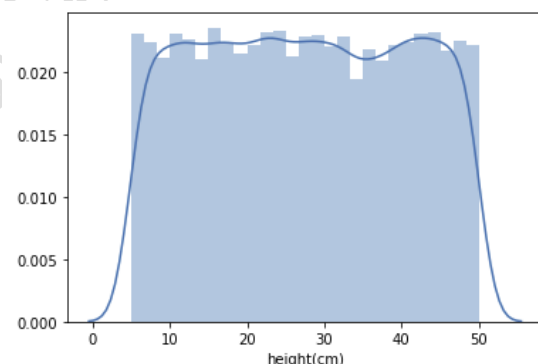
Distribution plot of 'Condition' column



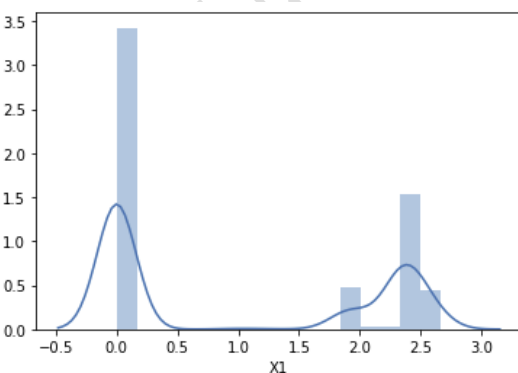
Distribution plot of 'Condition' column



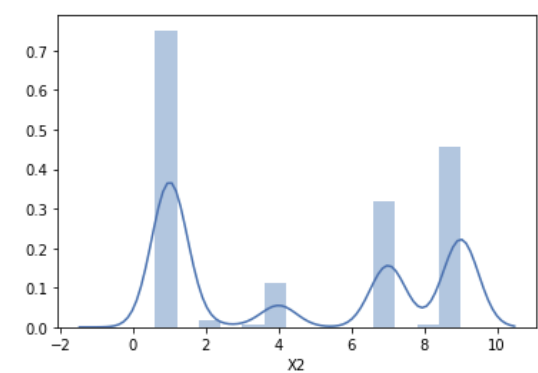
Distribution plot of 'Condition' column



Distribution plot of 'Condition' column



Distribution plot of 'Condition' column



Distribution plot of 'Condition' column

From these distribution plot its hard to visualize minor skewness as the data seems more or less equally distributed.

For fine tuning the skewness is checked manually by the skew values

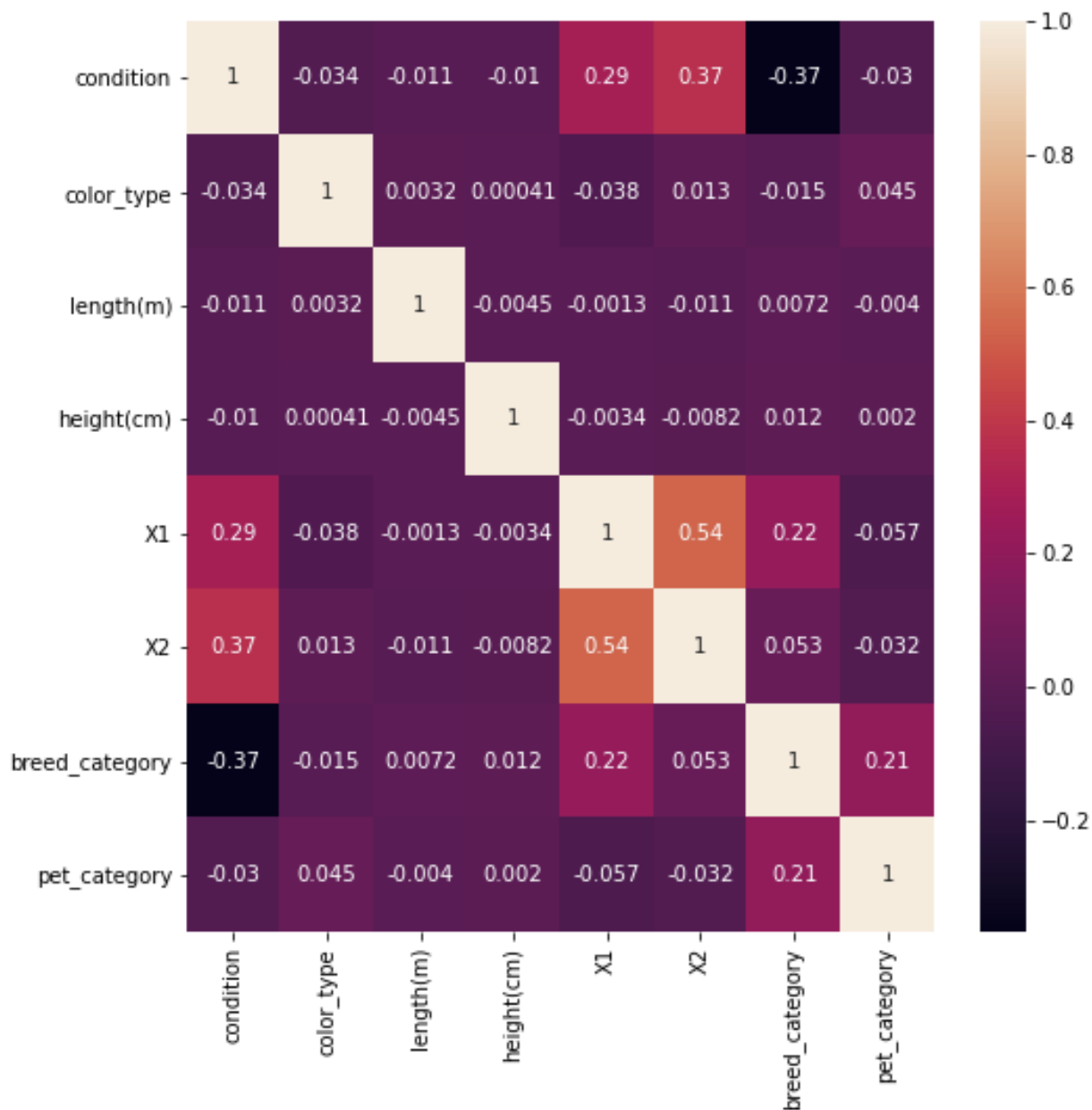
```
condition          0.210992
color_type         0.297405
length(m)         -0.002447
height(cm)         0.008525
X1                 0.563746
X2                 0.129107
breed_category     0.559098
pet_category       1.230389
dtype: float64
```

Skew value of all features of the dataset

As our dataset doesn't contains highly skewed input feature, tampering the input data is avoided here

## Bivariate Analysis

Visualize the relation between the features

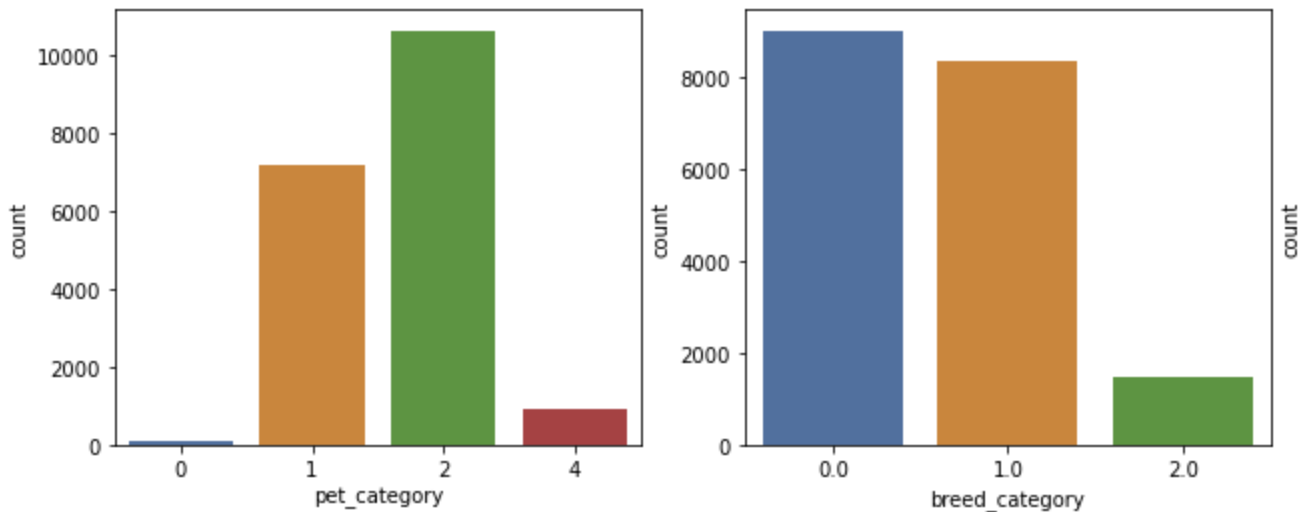


Having a smaller number of input features, reduction of curse of dimensionality is avoided



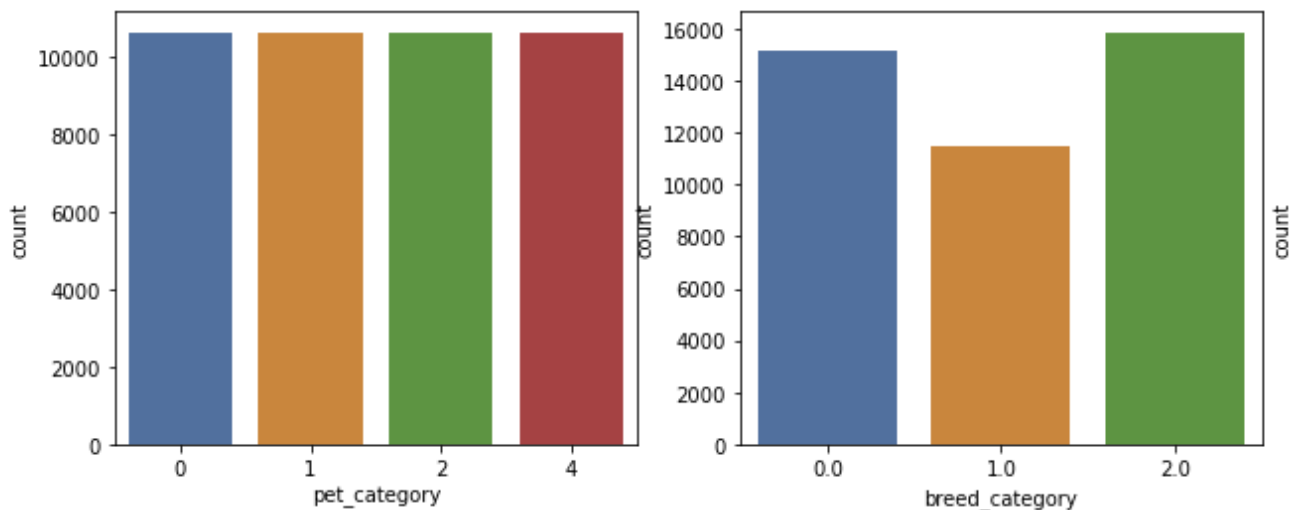
## Class Imbalance

Count-plot is used to visualize the class imbalance in the two output columns (label)



- Using upsampling the class imbalance in the output columns is removed

Count plot is reused to visualize the effect of resampling to remove class imbalance and the class imbalance is gone from 'pet\_catagory' and in breed category the class imbalance is reduced significantly



## Multioutput Regression Method

- Some models which are able to make multioutput prediction are used here and their performance report is shown here

### Logistic Regression Model

```
Confusion Matrix and Classification report for 'breed_category' column
[[ 11 3540 2410  26]
 [  5  895 3452 365]
 [  0 1304 4079 907]
 [  0   0   0   0]]
Value of 'breed_category' column predicted with accuracy: 29.333882546781215 %
      precision    recall  f1-score   support

    0.0         0.00     0.69     0.00         16
    1.0         0.19     0.16     0.17        5739
    2.0         0.65     0.41     0.50       9941
    3.0         0.00     0.00     0.00       1298

 accuracy         0.29       16994
 macro avg         0.21     0.31     0.17       16994
weighted avg         0.44     0.29     0.35       16994

Confusion Matrix and Classification report for 'pet_category' column
[[1144 2350  788   0]
 [ 247 3343  684   0]
 [ 879 2967  362   0]
 [ 119 1434 2677   0]]
Value of 'pet_category' column predicted with accuracy: 28.53360009415088 %
      precision    recall  f1-score   support

    0.0         0.27     0.48     0.34        2389
    1.0         0.78     0.33     0.47       10094
    2.0         0.09     0.08     0.08        4511
    4.0         0.00     0.00     0.00           0

 accuracy         0.29       16994
 macro avg         0.28     0.22     0.22       16994
weighted avg         0.52     0.29     0.35       16994
```

### Performance result of Logistic Regressor Model

Actual accuracy from both output column: 2.724490996822408 %

Actual accuracy is when both of the outputs are predicted correctly at once

## K Nearest Neighbors Model

Confusion Matrix and Classification report for 'breed\_category' column

```
[[1504 1912 1930 180 461]
 [ 408 2006 1746 150 407]
 [2442 271 60 100 3417]
 [ 0 0 0 0 0]
 [ 0 0 0 0 0]]
```

Value of 'breed\_category' column predicted with accuracy: 21.007414381546425 %

	precision	recall	f1-score	support
0.0	0.25	0.35	0.29	4354
1.0	0.43	0.48	0.45	4189
2.0	0.01	0.02	0.01	3736
3.0	0.00	0.00	0.00	430
4.0	0.00	0.00	0.00	4285
accuracy			0.21	16994
macro avg	0.14	0.17	0.15	16994
weighted avg	0.17	0.21	0.19	16994

Confusion Matrix and Classification report for 'pet\_category' column

```
[[1456 385 2441 0]
 [1939 2132 203 0]
 [2164 1900 144 0]
 [ 372 379 3479 0]]
```

Value of 'pet\_category' column predicted with accuracy: 21.96069200894433 %

	precision	recall	f1-score	support
0.0	0.34	0.25	0.29	5931
1.0	0.50	0.44	0.47	4796
2.0	0.03	0.02	0.03	6267
4.0	0.00	0.00	0.00	0
accuracy			0.22	16994
macro avg	0.22	0.18	0.20	16994
weighted avg	0.27	0.22	0.24	16994

## Performance result of K Neighbors Regressor Model

Actual accuracy from both output column: 16.270448393550666 %

## Accuracy result of K Neighbors Regressor Model

## Decision Tree Regressor Model

Confusion Matrix and Classification report for 'breed\_category' column

```
[[1467 2082 2017 421]
 [ 402 2131 1768 416]
 [2447 305 11 3527]
 [ 0 0 0 0]]
```

Value of 'breed\_category' column predicted with accuracy: 21.236907143697774 %

	precision	recall	f1-score	support
--	-----------	--------	----------	---------

0.0	0.25	0.34	0.28	4316
1.0	0.45	0.47	0.46	4518
2.0	0.00	0.00	0.00	3796
4.0	0.00	0.00	0.00	4364

accuracy			0.21	16994
macro avg	0.17	0.20	0.19	16994
weighted avg	0.18	0.21	0.20	16994

Confusion Matrix and Classification report for 'pet\_category' column

```
[[1456 385 2441 0]
 [1948 1992 334 0]
 [2221 1957 30 0]
 [ 376 358 3496 0]]
```

Value of 'pet\_category' column predicted with accuracy: 20.466046840061196 %

	precision	recall	f1-score	support
--	-----------	--------	----------	---------

0.0	0.34	0.24	0.28	6001
1.0	0.47	0.42	0.44	4692
2.0	0.01	0.00	0.01	6301
4.0	0.00	0.00	0.00	0

accuracy			0.20	16994
macro avg	0.20	0.17	0.18	16994
weighted avg	0.25	0.20	0.22	16994

## Performance result of Decision Tree Regressor Model

Actual accuracy from both output column: 18.74779333882547 %

## Accuracy result of Decision Tree Regressor Model

As we can see the Decision Tree Classifier is giving us best accuracy but this value is not up to the mark so another process is followed to get multioutput prediction

## Chained Multioutput Regression Method

In this method only one output feature is predicted at once.

After predicting one output column, all previous taken columns and this predicted output column is taken as input features to predict another output feature

To determine the best algorithm for predicting the output one column is chosen and some relevant model are created to determine the best algorithm by observing their performance in this dataset

## Gaussian Naive Bayes

Confusion Matrix and Classification report for 'breed\_category' column

```
[[1966  446 1007  863]
 [ 725 1817 1539  193]
 [ 733  809 2597   69]
 [ 363  451  546 2870]]
0.5443097563846063
```

	precision	recall	f1-score	support
0	0.46	0.52	0.49	3787
1	0.43	0.52	0.47	3523
2	0.62	0.46	0.52	5689
4	0.68	0.72	0.70	3995
accuracy			0.54	16994
macro avg	0.54	0.55	0.54	16994
weighted avg	0.56	0.54	0.54	16994

## Linear Support Vector Machine

Confusion Matrix and Classification report for 'breed\_category' column

```
[[1966 446 1007 863]
 [ 725 1817 1539 193]
 [ 733 809 2597 69]
 [ 363 451 546 2870]]
```

0.5025891491114511

	precision	recall	f1-score	support
0	0.53	0.49	0.50	4634
1	0.15	0.43	0.22	1497
2	0.64	0.47	0.54	5714
4	0.70	0.57	0.63	5149
accuracy			0.50	16994
macro avg	0.50	0.49	0.48	16994
weighted avg	0.58	0.50	0.53	16994

## Decision Tree Classifier

Confusion Matrix and Classification report for 'breed\_category' column

```
[[4282  0  0  0]
 [  6 3854 371 43]
 [ 15 652 3468 73]
 [  0  0  0 4230]]
```

0.931740614334471

	precision	recall	f1-score	support
0	1.00	1.00	1.00	4303
1	0.90	0.86	0.88	4506
2	0.82	0.90	0.86	3839
4	1.00	0.97	0.99	4346
accuracy			0.93	16994
macro avg	0.93	0.93	0.93	16994
weighted avg	0.93	0.93	0.93	16994

## Logistic Regression

Confusion Matrix and Classification report for 'breed\_category' column

```
[[2158 391 816 917]
 [ 977 1395 1114 788]
 [ 812 900 2284 212]
 [ 395 393 540 2902]]
0.514240320112981
```

	precision	recall	f1-score	support
0	0.50	0.50	0.50	4342
1	0.33	0.45	0.38	3079
2	0.54	0.48	0.51	4754
4	0.69	0.60	0.64	4819
accuracy			0.51	16994
macro avg	0.51	0.51	0.51	16994
weighted avg	0.53	0.51	0.52	16994

## Random Forest Classifier

Confusion Matrix and Classification report for 'breed\_category' column

```
[[3894 107 104 177]
 [ 115 3101 996 62]
 [ 92 727 3369 20]
 [ 406 321 390 3113]]
0.7930446039778746
```

	precision	recall	f1-score	support
0	0.91	0.86	0.89	4507
1	0.73	0.73	0.73	4256
2	0.80	0.69	0.74	4859
4	0.74	0.92	0.82	3372
accuracy			0.79	16994
macro avg	0.79	0.80	0.79	16994
weighted avg	0.80	0.79	0.79	16994

## Multi-Layer Perception Classifier

Confusion Matrix and Classification report for 'breed\_category' column

```
[[ 0 386 2904 992]
 [ 0 1676 2239 359]
 [ 0 629 3445 134]
 [ 0 352 1012 2866]]
0.46998940802636224
```

	precision	recall	f1-score	support
0	0.00	0.00	0.00	0
1	0.39	0.55	0.46	3043
2	0.82	0.36	0.50	9600
4	0.68	0.66	0.67	4351
accuracy			0.47	16994
macro avg	0.47	0.39	0.41	16994
weighted avg	0.71	0.47	0.53	16994

From these reports above its clear that Decision Tree Classifier perform the best in our dataset. To visualize the model's accuracy in one place a chart is given below

### Chart 1:

Accuracy, precision and recall of all model used in **Multioutput Regressor Method** is plotted in this chart

### Chart 2:

Accuracy, precision and recall of all model used in **Chained Multioutput Regression Method** is plotted in this chart



Chart I

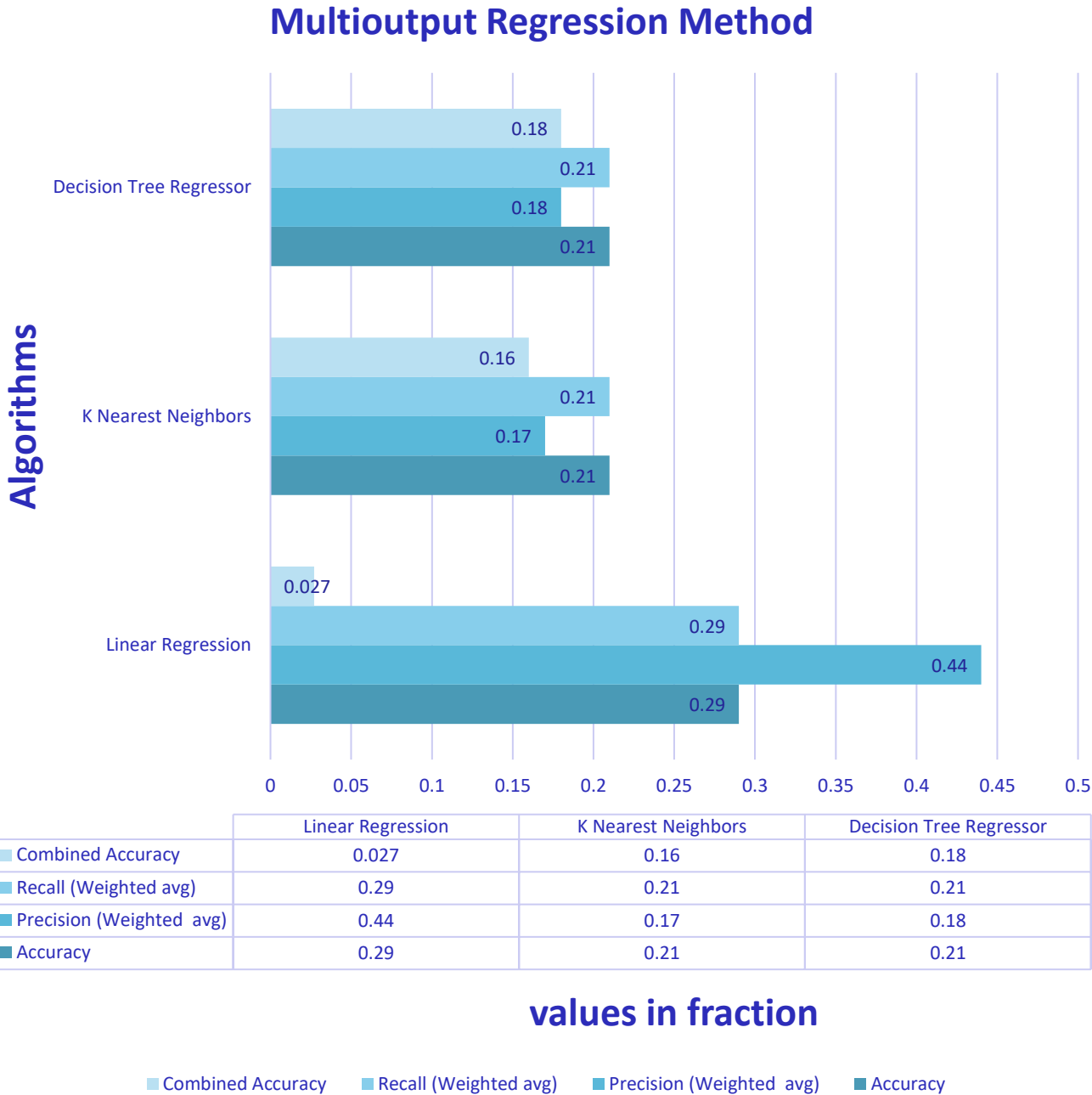
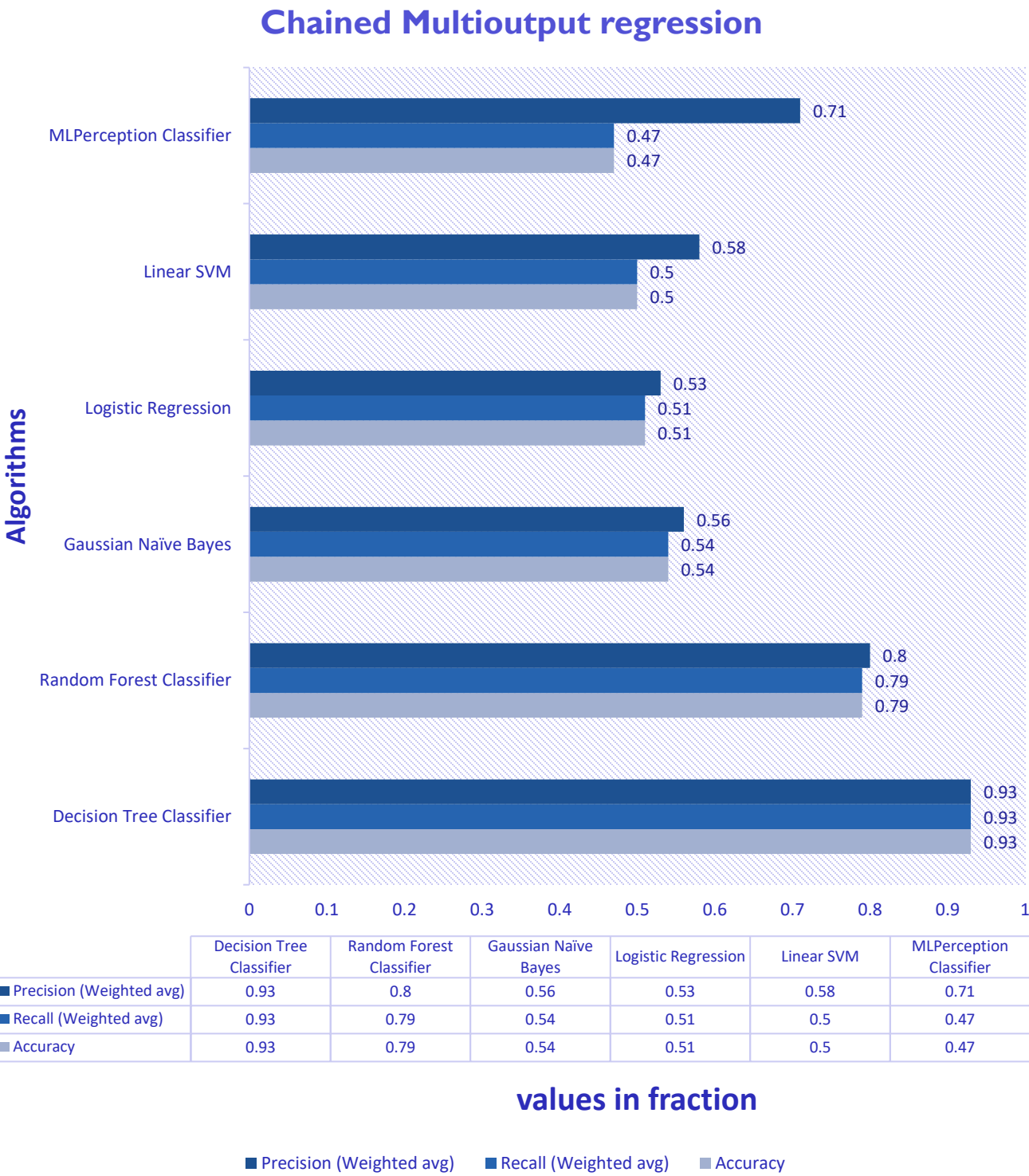


Chart 2



# Performance Analyzing

From these charts above it's clear that **Decision Tree Classifier** model gives the best output in our case so, **Decision Tree Classifier** is used as the final model to predict this feature as well as to predict the next output feature.

## Decision Tree Classifier (Performance report label 2 output)

Confusion Matrix and Classification report for 'breed\_category' column

```
[[5681  306    0]
 [ 319 4398    0]
 [    0    0 6290]]
0.9632223137577969
```

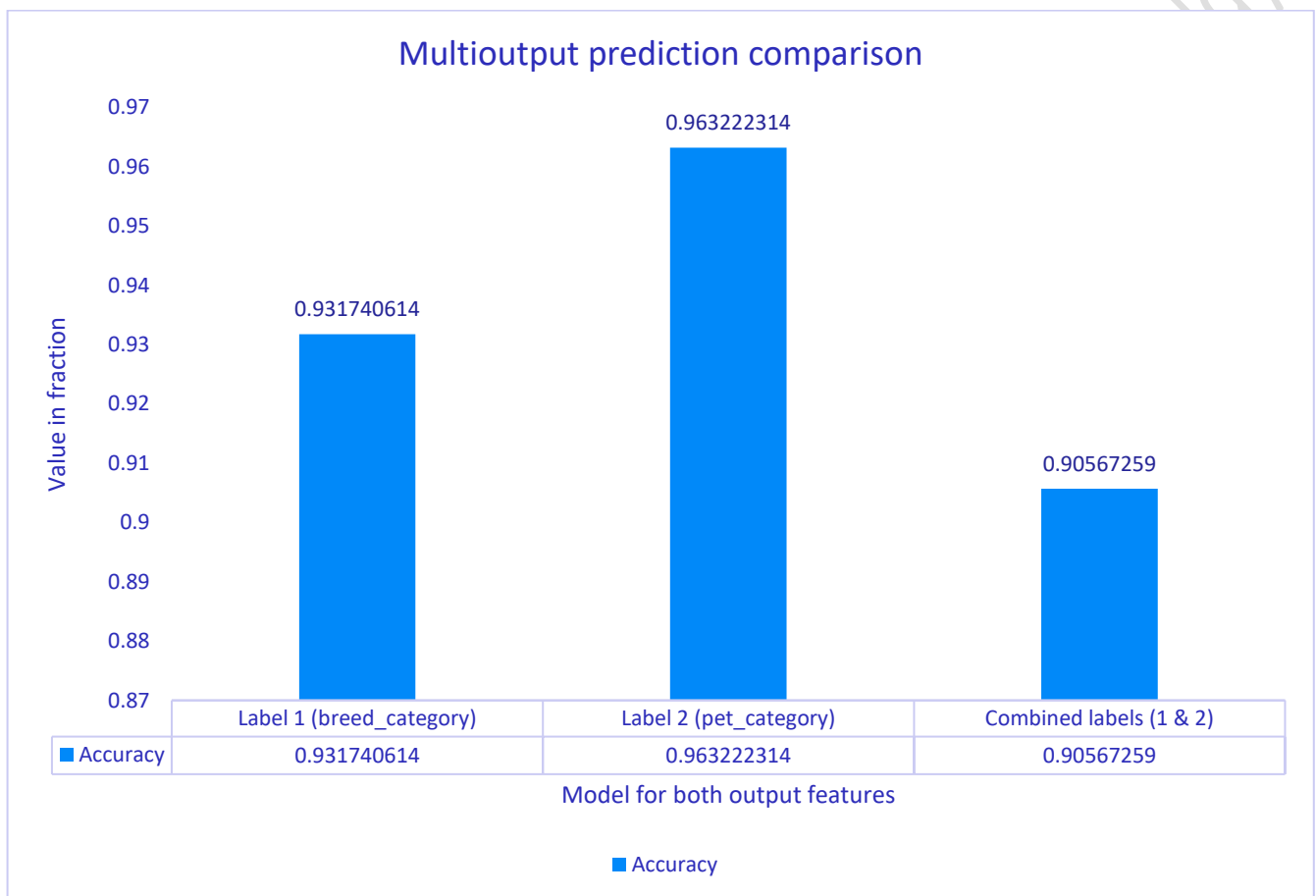
	precision	recall	f1-score	support
0.0	0.95	0.95	0.95	6000
1.0	0.93	0.93	0.93	4704
2.0	1.00	1.00	1.00	6290
accuracy			0.96	16994
macro avg	0.96	0.96	0.96	16994
weighted avg	0.96	0.96	0.96	16994

## Actual accuracy for two combined output features

Actual accuracy from both output column: 90.56725903259975 %

# Performance Analyzing

## Compare prediction accuracy for both of the output feature



Finally a clean .ipynb file was created with only essential codes to get the output as .csv file

The new new ipynb file named as 'Final Output.ipynb'

# At a glance

- Data pre-processing
  - ❖ Redundant feature removal
  - ❖ NaN value treatment
  - ❖ Outliers and skewness check
  - ❖ Class Imbalance removal from output column
- Model Building
  - ❖ Multioutput Regression Method
  - ❖ Chained Multioutput Regression Method
- Choosing best algorithm
- Applying the algorithm to predict both column one by one

## Final Step

- Create a final .ipynb file named 'Final Output.ipynb' to create a .csv file from train data with both of the output features and 'pet\_id column'

[The Decision Tree Classifier Algorithm is used and the predicted output's accuracy is 93% and 96% in the two output feature respectively and 90% accuracy in two combined output features]

# Thank You

My CV



Project by Pritimoy Sarkar


Student of Guru Nanak Institute of Technology

From Computer Science and Engineering branch

Email: [pritimoy.sarkar@gmail.com](mailto:pritimoy.sarkar@gmail.com)

 [Social Handle](#)



 GitHub Profile