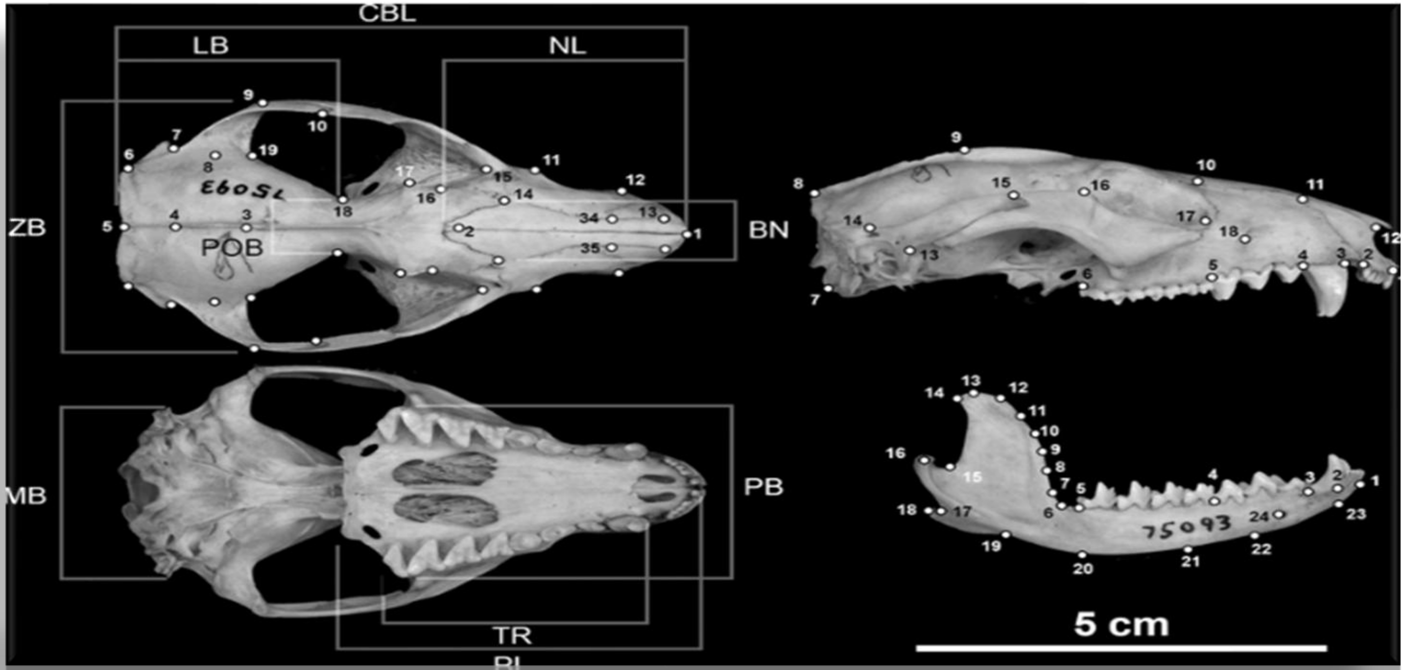
Predictive Modeling and Analysis of Possum Morphometrics:

A Regression Approach



GROUP NUMBER 2
BHAVAN'S VIVEKANANDA COLLEGE.
<u>GROUPMEMBERS:</u>
B.HARSHIT KUMAR,
SRIKANTH YADAV,
CHAITANYA JADAV.

Abstract:

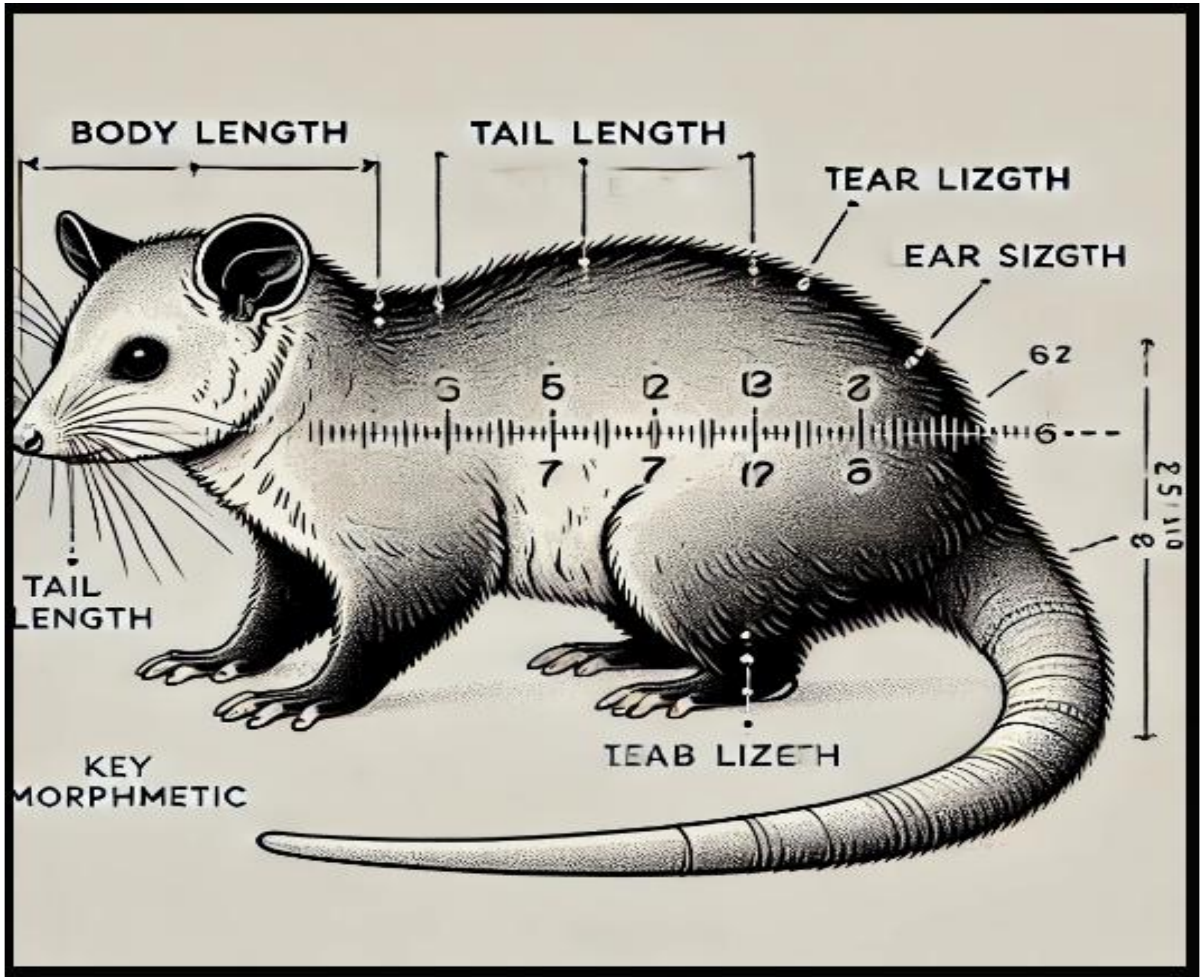
This study applies regression analysis to the possum dataset to explore relationships between morphological features, such as head length, skull width, and tail length. The goal is to develop predictive models that reveal how these characteristics are interrelated, providing insights useful for species identification and ecological research

Objectives:

1. **To identify key relationships** between various morphological features of possums using regression analysis.
2. **To develop predictive models** that estimate specific possum characteristics (e.g., body or head length) based on other physical attributes.
3. **To evaluate model accuracy** and determine the best predictors among the variables in the dataset.

CONTENT

- Introduction
- Literature Review
- Data Preprocessing
- Exploratory Data Analysis
- Data Modeling and Evaluation
- Summary
- Appendix



INTODUCTION:

Possums exhibit diverse physical traits that vary across species and habitats. Using the possum dataset, this study applies regression analysis to explore relationships among features like head length, skull width, and tail length. The goal is to identify key predictors and develop models for estimating specific traits based on others. This analysis provides insights into possum morphology, supporting species classification and ecological research.

LITERATURE REVIEW

LITRATURE REVIEW-1

McDonald and Rose (2000) examined age-related changes in skeletal morphology of possums, documenting how certain features, like skull width and body length, increase with age. Their findings are often used to create growth models for wild populations.

LITRATURE REVIEW-2

-Tyndale-Biscoe and MacKenzie (1976) examined morphological differences in possums from coastal versus inland habitats, concluding that environmental factors like temperature and food resources significantly impact body size and growth patterns.

DATA PREPROCESSING

DATA

DATASET: The data consists of 14 variables and 105 records
SOURCE:https://drive.google.com/drive/folders/1vGSRCnhqSxEH53BgLqhh32F1qNKqfOim?usp=drive_link

1	case	site	Pop	sex	age	hdlngth	skullw	totlngth	taill	footlgth	earconch	eye	chest	belly
2	1	1	Vic	m	8	94.1	60.4	89	36	74.5	54.5	15.2	28	36
3	2	1	Vic	f	6	92.5	57.6	91.5	36.5	72.5	51.2	16	28.5	33
4	3	1	Vic	f	6	94	60	95.5	39	75.4	51.9	15.5	30	34
5	4	1	Vic	f	6	93.2	57.1	92	38	76.1	52.2	15.2	28	34
6	5	1	Vic	f	2	91.5	56.3	85.5	36	71	53.2	15.1	28.5	33
7	6	1	Vic	f	1	93.1	54.8	90.5	35.5	73.2	53.6	14.2	30	32
8	7	1	Vic	m	2	95.3	58.2	89.5	36	71.5	52	14.2	30	34.5
9	8	1	Vic	f	6	94.8	57.6	91	37	72.7	53.9	14.5	29	34
10	9	1	Vic	f	9	93.4	56.3	91.5	37	72.4	52.9	15.5	28	33
11	10	1	Vic	f	6	91.8	58	89.5	37.5	70.9	53.4	14.4	27.5	32
12	11	1	Vic	f	9	93.3	57.2	89.5	39	77.2	51.3	14.9	31	34
13	12	1	Vic	f	5	94.9	55.6	92	35.5	71.7	51	15.3	28	33
14	13	1	Vic	m	5	95.1	59.9	89.5	36	71	49.8	15.8	27	32
15	14	1	Vic	m	3	95.4	57.6	91.5	36	74.3	53.7	15.1	28	31.5
16	15	1	Vic	m	5	92.9	57.6	85.5	34	69.7	51.8	15.7	28	35
17	16	1	Vic	m	4	91.6	56	86	34.5	73	51.4	14.4	28	32
18	17	1	Vic	f	1	94.7	67.7	89.5	36.5	73.2	53.2	14.7	29	31
19	18	1	Vic	m	2	93.5	55.7	90	36	73.7	55.4	15.3	28	32
20	19	1	Vic	f	5	94.4	55.4	90.5	35	73.4	53.9	15.2	28	32
21	20	1	Vic	f	4	94.8	56.3	89	38	73.8	52.4	15.5	27	36
22	21	1	Vic	f	3	95.9	58.1	96.5	39.5	77.9	52.9	14.2	30	40
23	22	1	Vic	m	3	96.3	58.5	91	39.5	73.5	52.1	16.2	28	36

VARIABLES

The variables are of two types:
continuous and categorical data, where the continuous variables are in integer and float format and categorical variables are in object format

Integer	Float	Object
case	age	pop
site	hdlngth,eye	sex
	skullw totlngth	
	taill	
	footlngth	
	eaarconch	
	Chest,belly	

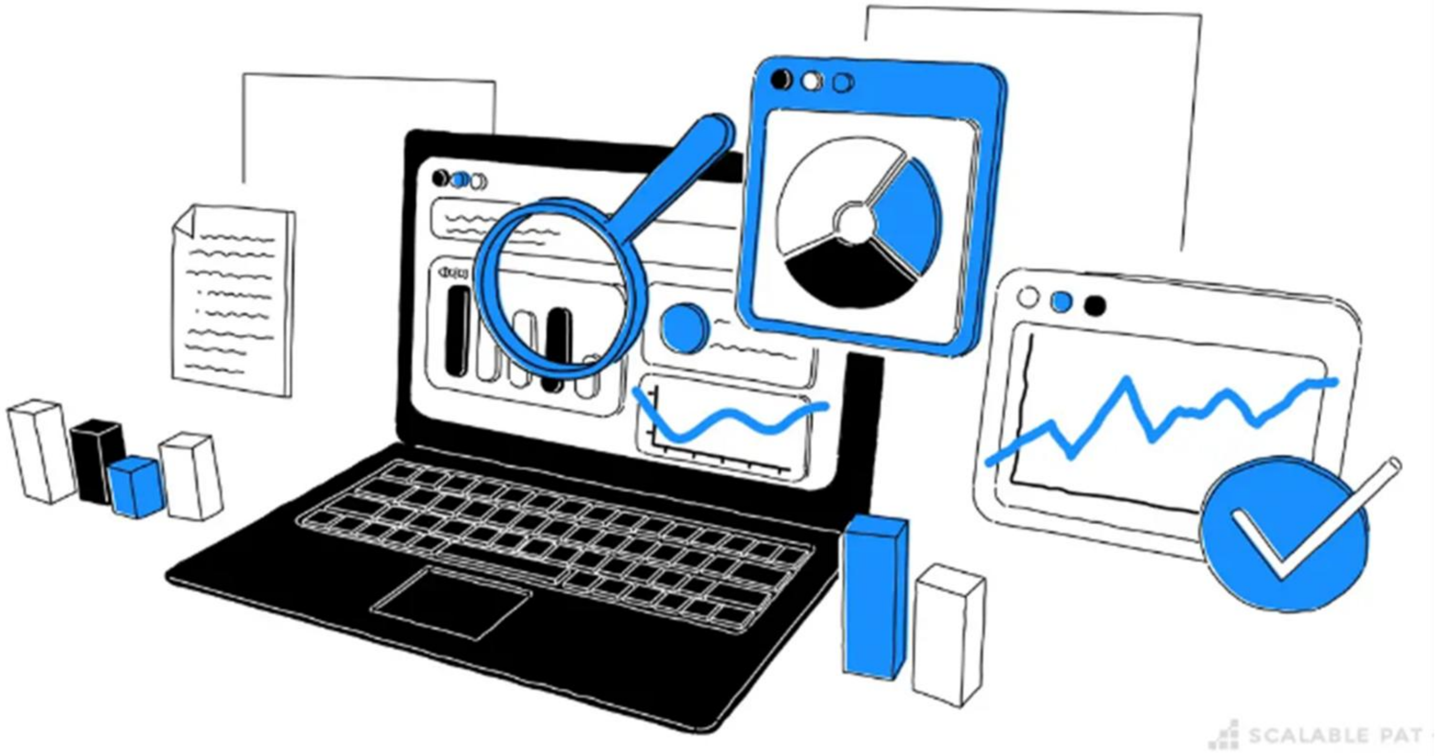
DATA CLEANING

- Checking the Unique Values: The unique values of all the columns are identified.
- Identifying Null Values: The data is analysed for any missing value or null value.
- Replacing the Null Values: The null values which are found are then replaced with suitable values like mean/

- To perform dummy variable encoding we divided the data into two sets, which are independent variables and dependent variables
- The categorical variables are encoded into continuous variables using the dummy variables
- Renaming the columns, the dummy columns are

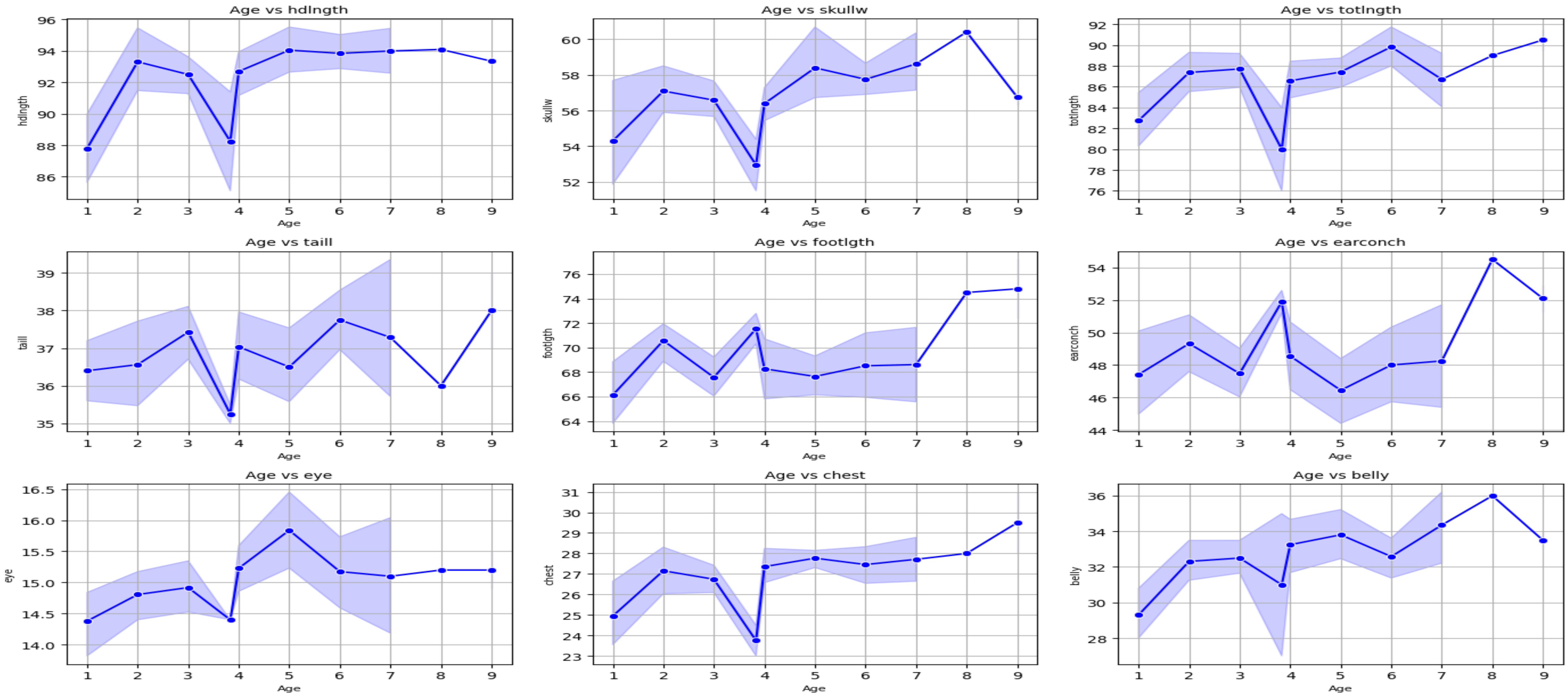
ORGINAL VARIABLE NAMES	RENAMED VARIABLE NAMES
pop	pop_other
sex	Sex_m

EXPLORATORY DATA ANALYSIS



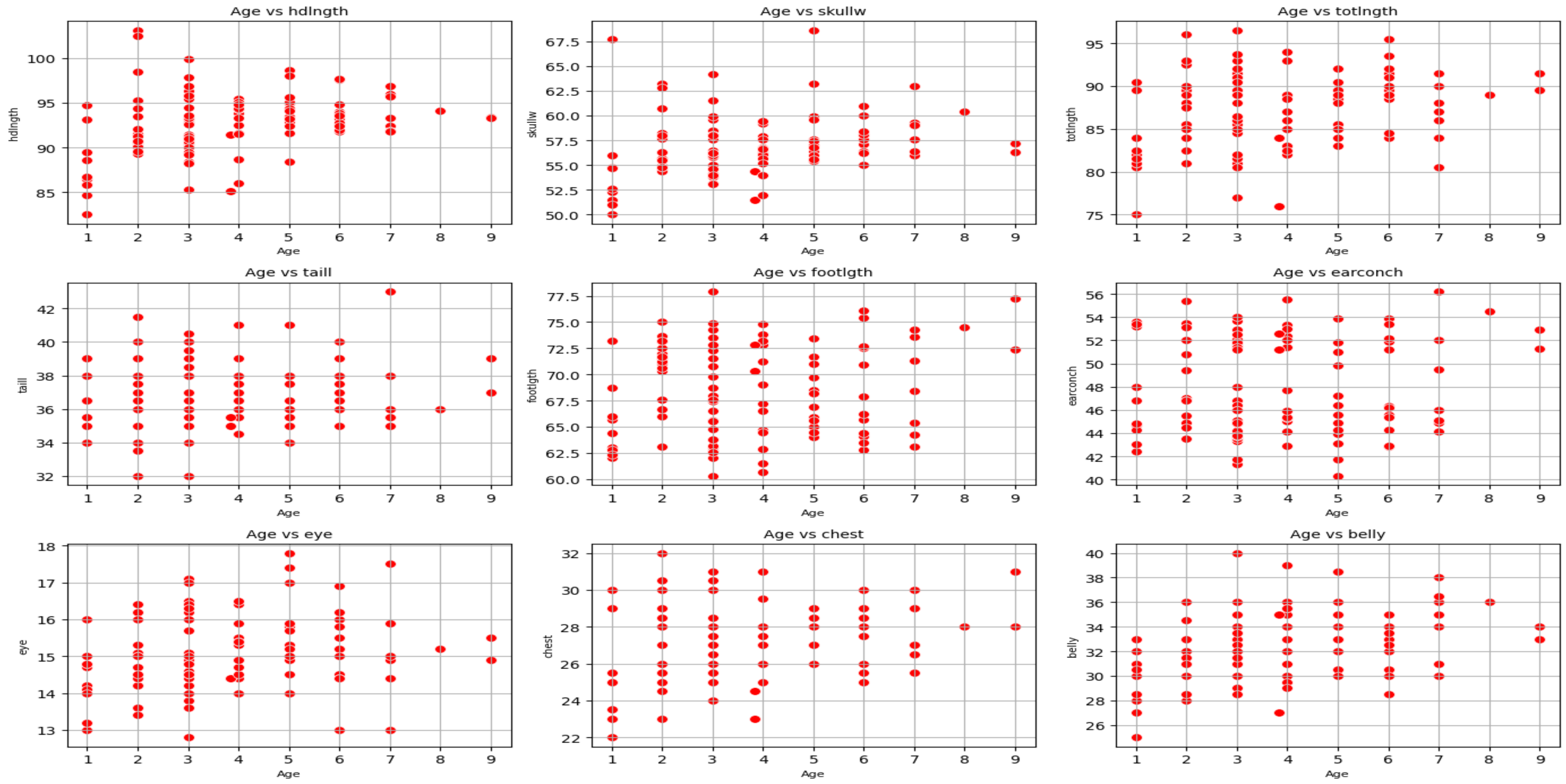
LINEPLOT

The target variable "age" is plotted against all the independent variables in the line plots as shown below.



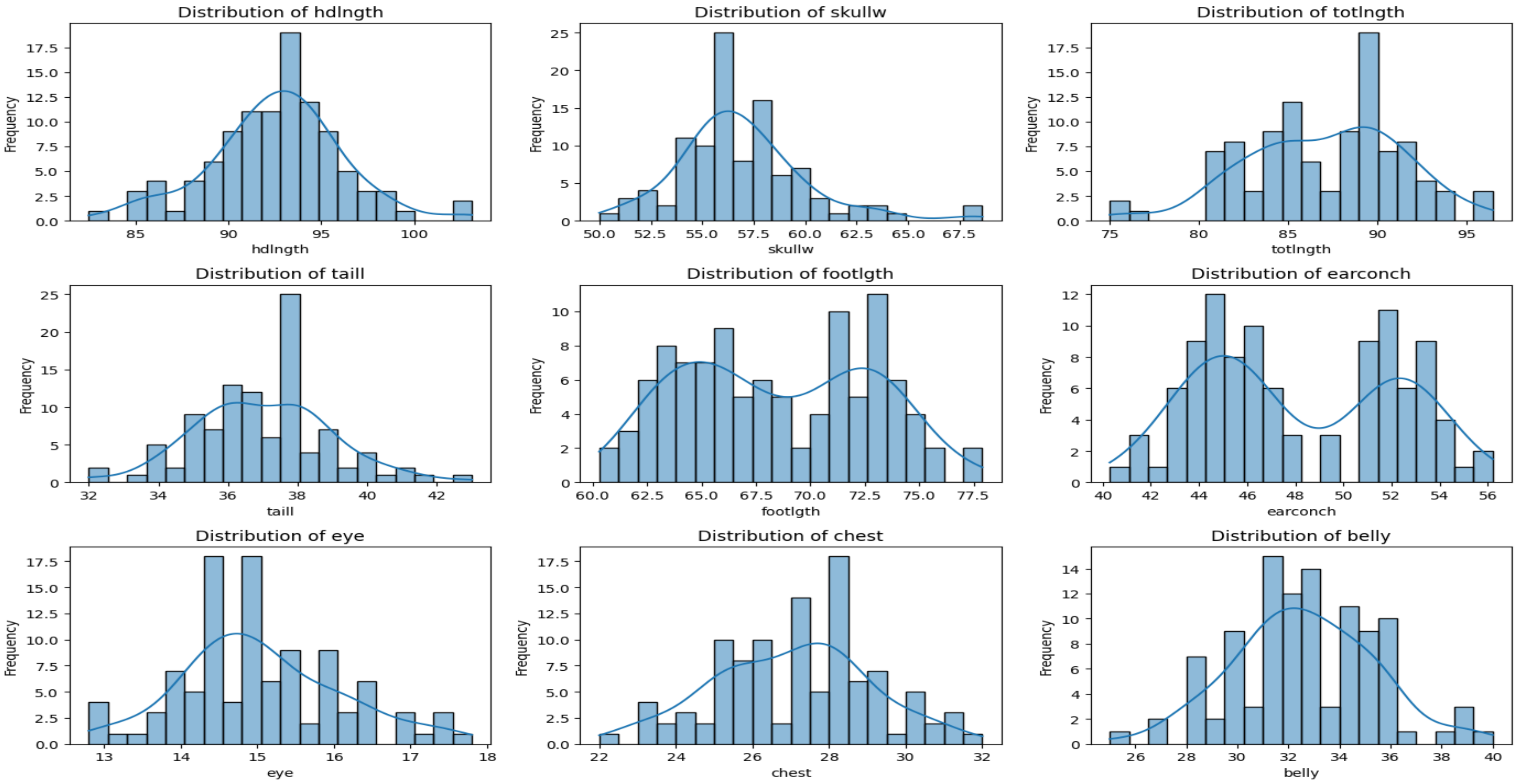
SCATTERPLOT

The target variable “age” is plotted against all the independent variables in the scatter plot shown below.



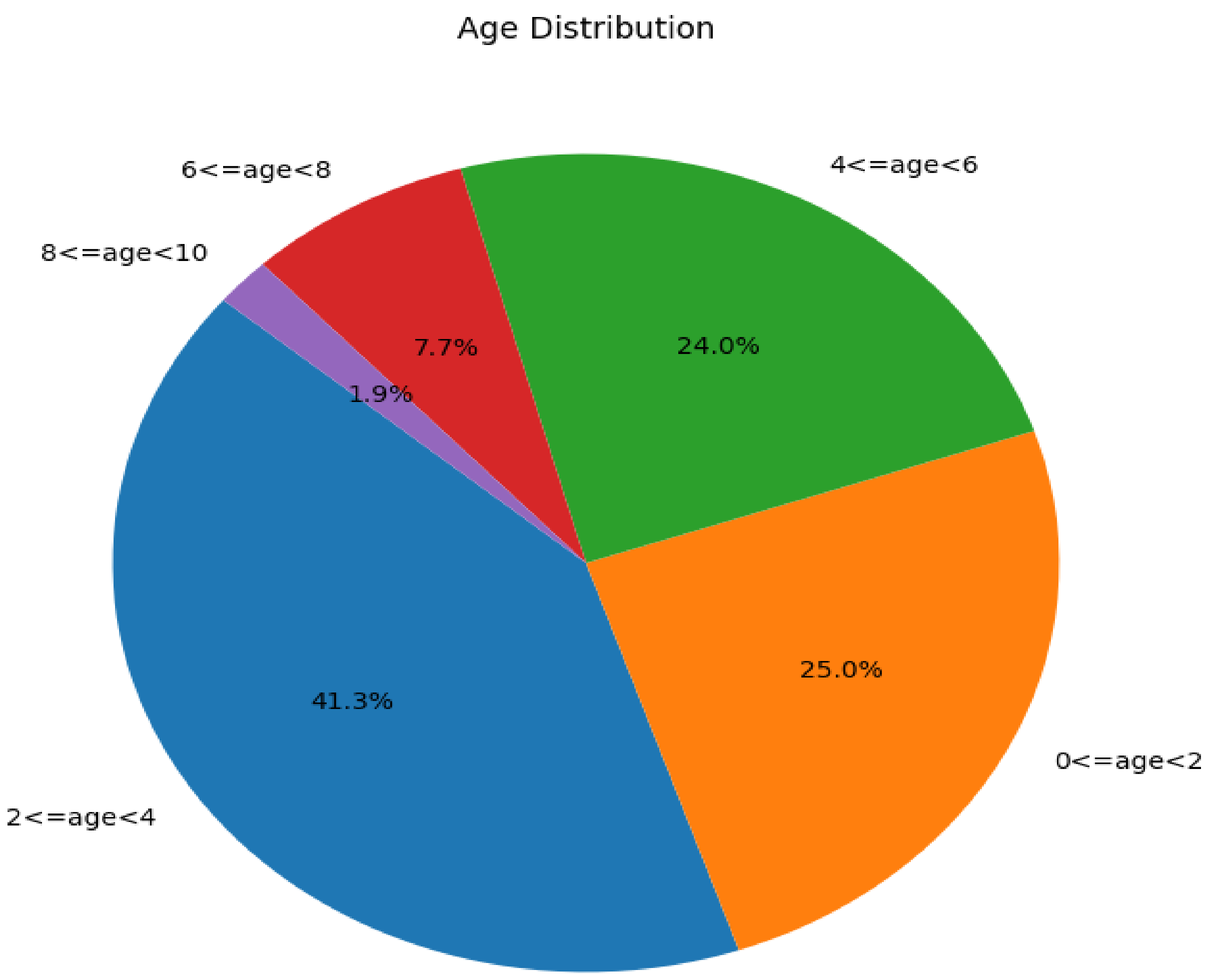
HISTOGRAM

The target variable "age" is plotted against all the independent variables in the histogram shown below.



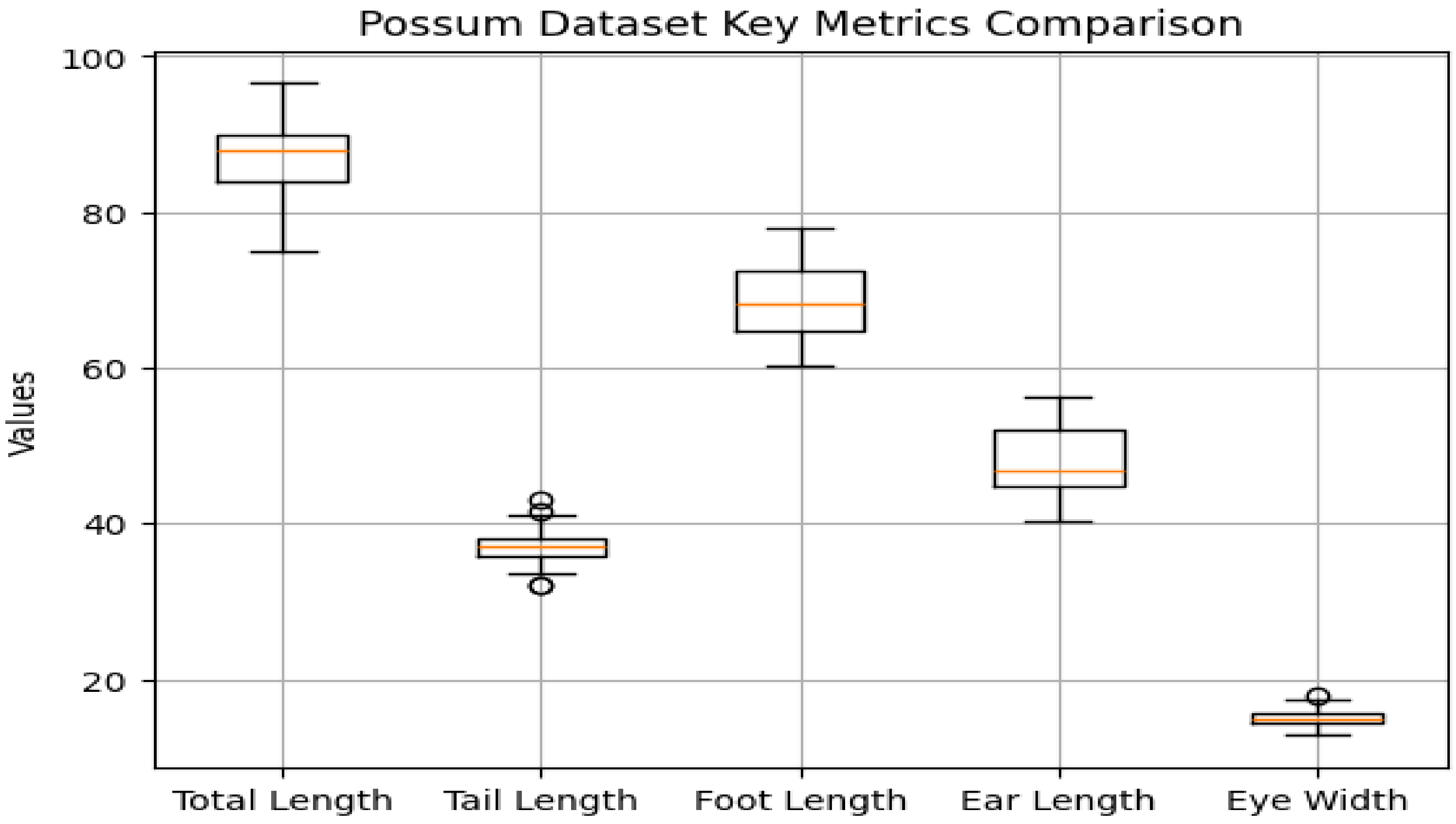
PIECHART

The pie chart below shows the distribution in the "age" column.



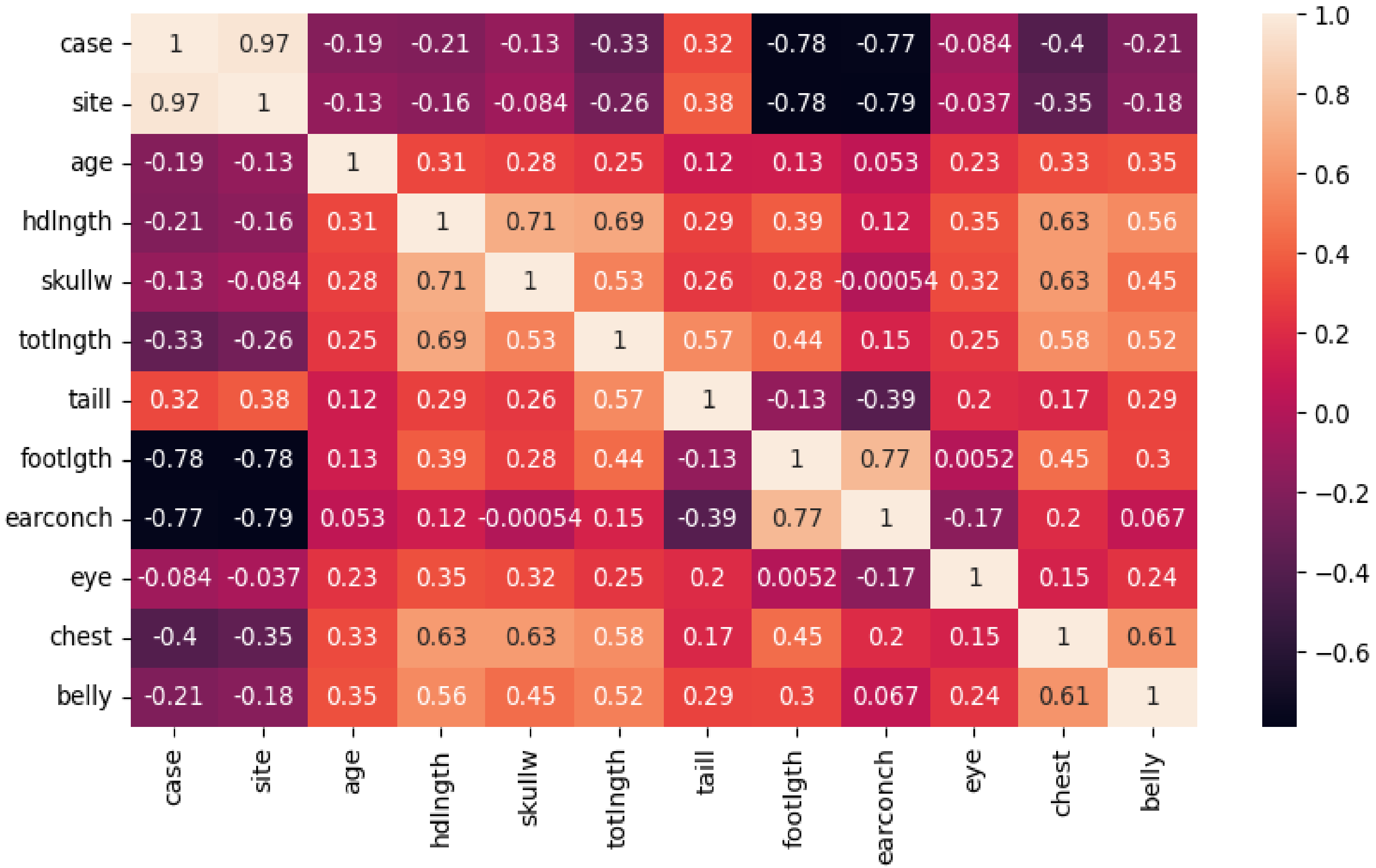
BOXPLOT

The boxplot below shows the possum dataset key metrics comparison.



CORRELATION MATRIX

Here we can observe that case and site have high positive correlation.
The next most positively correlated are headlength(hlngth) and skullwidth(skullw).



MULTICOLLINEARITY CHECK

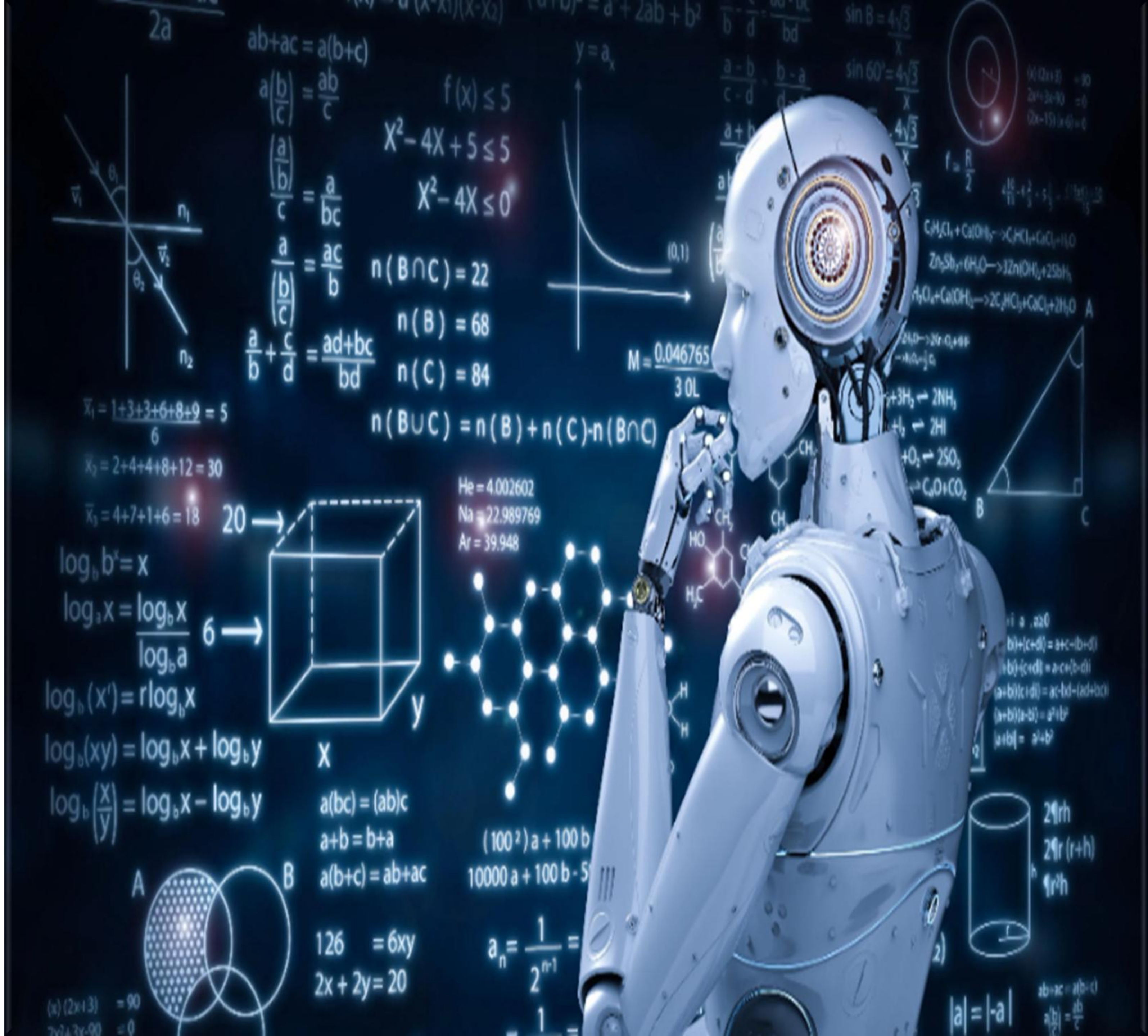
The variables in this dataset are highly correlated with each other as we are only left with 2 variables at last after performing the Multicollinearity Test. Hence performing Multicollinearity Test for this dataset is not suitable

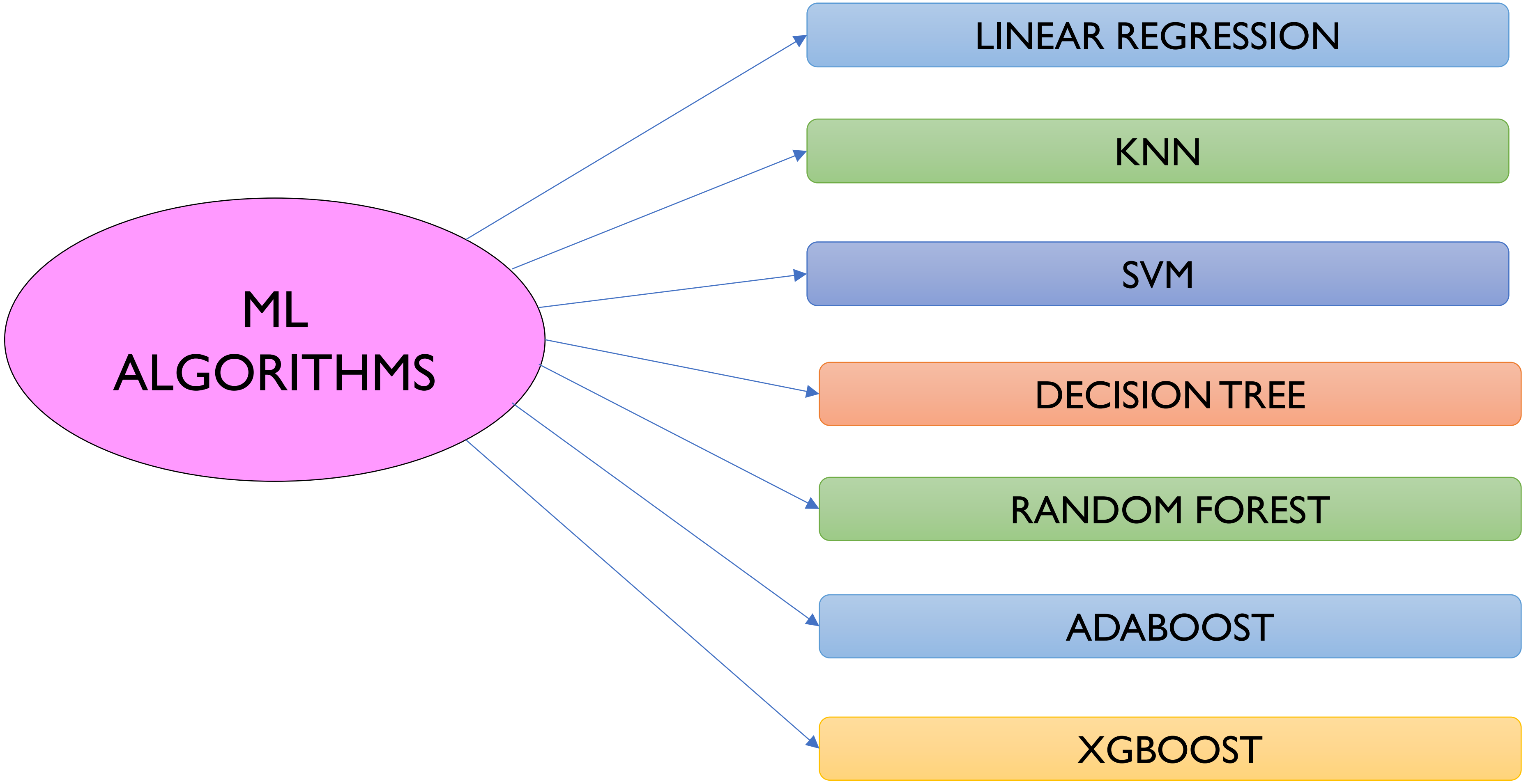
	variables	VIF
0	case	85.9
1	site	80.3
2	hdlngth	2111.3
3	skullw	937.0
4	totlngth	1967.4
5	taill	975.7
6	footlgth	1036.3
7	earconch	384.4
8	eye	230.7
9	chest	486.9
10	belly	274.8

	variables	VIF
0	case	79.2
1	site	75.4
2	taill	421.1
3	earconch	179.6
4	eye	174.7
5	chest	296.8
6	belly	258.9

	variables	VIF
0	site	3.0
1	earconch	3.0

MACHINE LEARNING ALGORITHMS




ML
ALGORITHMS
LINEAR REGRESSION
KNN
SVM
DECISION TREE
RANDOM FOREST
ADABOOST
XGBOOST

60:40 Train-Test Split

Algorithm	MAE	R2_score
Linear Regression	1.530	0.072
KNN	1.622	0.037
SVM	1.659	-0.063
Decision Tree	1.591	0.014
Random Forest	2.047	0.204
AdaBoost	1.461	0.098
XGBoost	1.398	0.272

70:30 Train-Test Split

Algorithm	MAE	R2_score
Linear Regression	1.345	0.150
KNN	1.520	0.035
SVM	1.401	0.118
Decision Tree	1.307	0.147
Random Forest	2.759	0.273
AdaBoost	0.975	0.429
XGBoost	1.258	0.283

75:25 Train-Test Split

Algorithm	MAE	R2_score
Linear Regression	1.449	0.148
KNN	1.617	0.033
SVM	1.543	0.149
Decision Tree	1.339	0.235
Random Forest	1.339	0.239
AdaBoost	1.588	0.044
XGBoost	1.403	0.139

80:20 Train-Test Split

Algorithm	MAE	R2_score
Linear Regression	1.525	0.104
KNN	1.665	0.016
SVM	1.582	0.124
Decision Tree	2.047	-0.599
Random Forest	1.522	0.122
AdaBoost	1.413	0.108
XGBoost	1.530	0.003

ALGORITHM COMPARISION

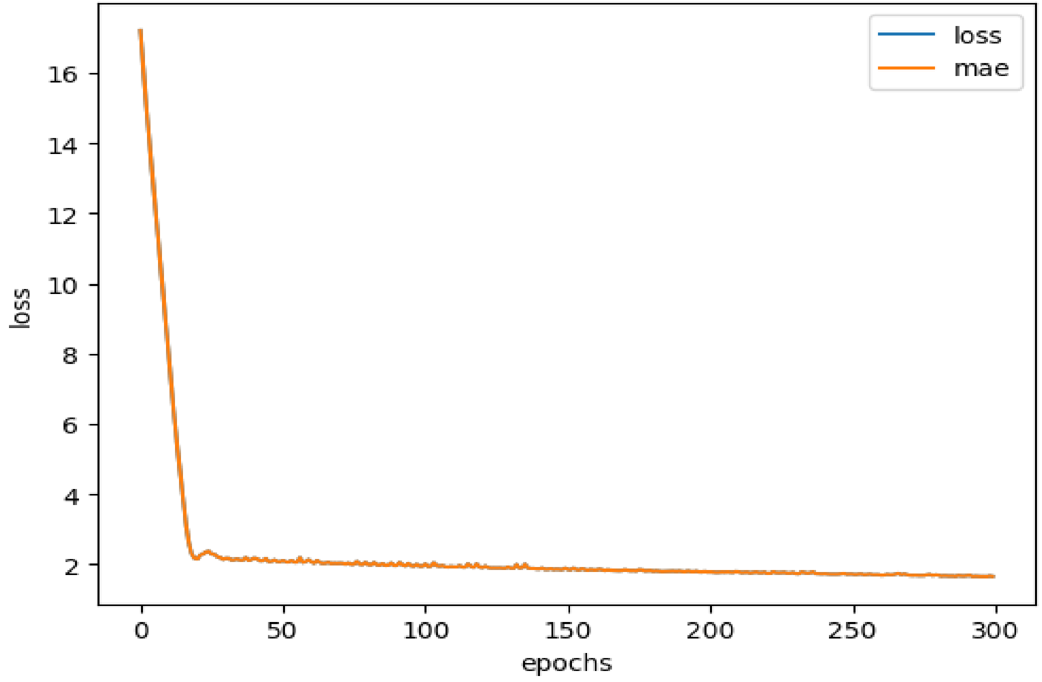
ADABOOST FOR 70:30 SPLIT

Algorithm	MAE	R2_SCORE
Linear Regression	1.345	0.150
KNN	1.520	0.035
SVM	1.401	0.118
Decision Tree	1.307	0.014
Random Forest	2.759	0.273
AdaBoost	0.975	0.429
XGBoost	1.258	0.283

ARTIFICIAL NEURAL NETWORK:

SLPIT	ARCHITECTURE	OPTIMIZER	EPOCHS	MAE
60-40	10-7-5-1	Adam	300	1.47
70-30	10-7-5-1	Adam	250	1.46
75-25	10-7-5-1	Adam	300	1.33
80-20	10-7-5-1	Adam	250	1.54

Training Metrics: Loss and MAE for best split 75-25



CONCLUSION

- The primary goal of this research is to predict the age of a possum based on its body measurements and gender.
- In our study, we applied both Machine Learning (ML) models and a Deep Learning model to build a predictive model for possum age based on its morphometric data.
- Among the ML models, the Adaboost algorithm demonstrated the best performance, achieving the lowest Mean Absolute Error (MAE) value of **0.975**.
- For Deep Learning, we used only the Artificial Neural Network (ANN) model, which showed a slightly higher MAE value of **1.33**, indicating relatively lower accuracy compared to the ML model.
- Thus, we conclude that the ML model (Adaboost) outperforms the Deep Learning model (ANN) in terms of predictive accuracy for this dataset

INSIGHTS

•**Key Findings:** Strong correlations observed between head length & skull width;

multicollinearity limited usable predictors.

•**Best Model:** AdaBoost (70:30 split) outperformed other algorithms.

•**Practical Use:** Predictive modeling aids in species identification and ecological research.

•**Challenges:** Multicollinearity and small dataset size impacted results.

•**Future Steps:** Enhance dataset size and apply advanced feature selection

FUTURE SCOPE

•Expand dataset size and include more features.

•Address multicollinearity using advanced techniques.

•Optimize models with ensemble methods and tuning.

•Apply models for species identification in ecology.

•Develop tools for real-time predictions

Work Distribution

Team Member 1 (Chaitanya)	Collecting Information about credit card approvals and Literature Review
Team Member 2 (Srikanth)	Data Pre-processing and EDA
Team Member 3 (Harshit)	Implementing ML Algorithms

COLAB LINK

THANK YOU

B.HARSHIT KUMAR
SRIKANTH YADAV
CHAITANYA JADAV

APPENDIX

LOADING DATASET

```
[7] data=pd.read_csv('/content/possum.csv')
    data.head()
```

	case	site	Pop	sex	age	hdlngth	skullw	totlngth	taill	footlgth	earconch	eye	chest	belly
0	1	1	Vic	m	8.0	94.1	60.4	89.0	36.0	74.5	54.5	15.2	28.0	36.0
1	2	1	Vic	f	6.0	92.5	57.6	91.5	36.5	72.5	51.2	16.0	28.5	33.0
2	3	1	Vic	f	6.0	94.0	60.0	95.5	39.0	75.4	51.9	15.5	30.0	34.0
3	4	1	Vic	f	6.0	93.2	57.1	92.0	38.0	76.1	52.2	15.2	28.0	34.0
4	5	1	Vic	f	2.0	91.5	56.3	85.5	36.0	71.0	53.2	15.1	28.5	33.0

Checking Null values

```
data.isna().sum()
```

	0
case	0
site	0
Pop	0
sex	0
age	2
hdlngth	0
skullw	0
totlngth	0
taill	0
footlgth	1
earconch	0
eye	0
chest	0
belly	0

CHECKING DATATYPE

```
[ ] data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 104 entries, 0 to 103
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   case      104 non-null    int64
 1   site      104 non-null    int64
 2   Pop       104 non-null    object
 3   sex       104 non-null    object
 4   age       102 non-null    float64
 5   hdlngth   104 non-null    float64
 6   skullw    104 non-null    float64
 7   totlngth  104 non-null    float64
 8   taill     104 non-null    float64
 9   footlgth  103 non-null    float64
 10  earconch  104 non-null    float64
 11  eye       104 non-null    float64
 12  chest     104 non-null    float64
 13  belly     104 non-null    float64
dtypes: float64(10), int64(2), object(2)
memory usage: 11.5+ KB
```

FINDING UNIQUE VALUES

```
for i in range(data.shape[1]):
  print(data.iloc[:,i].unique())
  print(data.iloc[:,i].value_counts())
```

```
[  1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18
  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36
  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54
  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72
  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90
  91  92  93  94  95  96  97  98  99 100 101 102 103 104]
case
1      1
2      1
77     1
76     1
75     1
      ..
32     1
31     1
30     1
29     1
104    1
Name: count, Length: 104, dtype: int64
[1 2 3 4 5 6 7]
```

```
[11] data.isna().sum()
```

	0
case	0
site	0
Pop	0
sex	0
age	0
hdlngth	0
skullw	0
totlngth	0
taill	0
footlgth	0
earconch	0
eye	0
chest	0
belly	0

Replacing missing values

```
[10] continuous_columns = data.select_dtypes(include=['float64', 'int64']).columns
     # Replace null values in continuous columns with the mean
     for column in continuous_columns:
         mean_value = data[column].mean()
         data[column].fillna(mean_value, inplace=True)
```

Dropping the column 'age' and assigning it as target variable

```python
X=df.drop(['age'],axis=1)
print(X)
y=df['age']
print(y)
```

```
     case  site    Pop sex  hdlngth  skullw  totlngth  taill  footlgth  \
0       1     1    Vic   m     94.1    60.4      89.0   36.0      74.5
1       2     1    Vic   f     92.5    57.6      91.5   36.5      72.5
2       3     1    Vic   f     94.0    60.0      95.5   39.0      75.4
3       4     1    Vic   f     93.2    57.1      92.0   38.0      76.1
4       5     1    Vic   f     91.5    56.3      85.5   36.0      71.0
..    ...   ...    ...  ..      ...     ...       ...    ...       ...
99    100     7  other   m     89.5    56.0      81.5   36.5      66.0
100   101     7  other   m     88.6    54.7      82.5   39.0      64.4
101   102     7  other   f     92.4    55.0      89.0   38.0      63.5
102   103     7  other   m     91.5    55.2      82.5   36.5      62.9
103   104     7  other   f     93.6    59.9      89.0   40.0      67.6

     earconch   eye  chest  belly
0        54.5  15.2   28.0   36.0
1        51.2  16.0   28.5   33.0
2        51.9  15.5   30.0   34.0
3        52.2  15.2   28.0   34.0
4        53.2  15.1   28.5   33.0
..        ...   ...    ...    ...
99       46.8  14.8   23.0   27.0
100      48.0  14.0   25.0   33.0
101      45.4  13.0   25.0   30.0
102      45.9  15.4   25.0   29.0
103      46.0  14.8   28.5   33.5
```

CREATING DUMMY VARIABLE

```python
X=pd.get_dummies(X,dtype='int',drop_first=True)
print(X)
```

```
     case  site  hdlngth  skullw  totlngth  taill  footlgth  earconch   eye  \
0       1     1     94.1    60.4      89.0   36.0      74.5      54.5  15.2
1       2     1     92.5    57.6      91.5   36.5      72.5      51.2  16.0
2       3     1     94.0    60.0      95.5   39.0      75.4      51.9  15.5
3       4     1     93.2    57.1      92.0   38.0      76.1      52.2  15.2
4       5     1     91.5    56.3      85.5   36.0      71.0      53.2  15.1
..    ...   ...      ...     ...       ...    ...       ...       ...   ...
99    100     7     89.5    56.0      81.5   36.5      66.0      46.8  14.8
100   101     7     88.6    54.7      82.5   39.0      64.4      48.0  14.0
101   102     7     92.4    55.0      89.0   38.0      63.5      45.4  13.0
102   103     7     91.5    55.2      82.5   36.5      62.9      45.9  15.4
103   104     7     93.6    59.9      89.0   40.0      67.6      46.0  14.8

     chest  belly  Pop_other  sex_m
0     28.0   36.0          0      1
1     28.5   33.0          0      0
2     30.0   34.0          0      0
3     28.0   34.0          0      0
4     28.5   33.0          0      0
..     ...    ...        ...    ...
99    23.0   27.0          1      1
100   25.0   33.0          1      1
101   25.0   30.0          1      0
102   25.0   29.0          1      1
103   28.5   33.5          1      0

[104 rows x 13 columns]
```

LIBRARIES

```python
import pandas as pd
import seaborn as sns
import statsmodels.formula.api as smf
from sklearn.linear_model import LinearRegression
from sklearn import metrics
from sklearn.model_selection import train_test_split
import numpy as np

import warnings
warnings.filterwarnings("ignore")
```

```python
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
```

```python
[ ] # Import library for VIF
    from statsmodels.stats.outliers_influence import variance_inflation_factor

    def calc_vif(X):

        # Calculating VIF
        vif = pd.DataFrame()
        vif["variables"] = X.columns
        vif["VIF"] = [variance_inflation_factor(X.values, i).round(1) for i in range(X.shape[1])]

        return(vif)

    calc_vif(X)
```

ML Models

LINEAR REGRESSION

```python
### SCIKIT-LEARN ###
#multiple regression
# create X and y
feature_cols = ['case','site','hdlngth', 'skullw', 'totlngth', 'taill', 'footlgth', 'earconch', 'eye', 'chest', 'belly']
X = data[feature_cols]
y = data.age

# instantiate and fit
lm2 = LinearRegression()
lm2.fit(X, y)

# print the coefficients
print(lm2.intercept_)
print(lm2.coef_)
```

```
-5.20235066804844
[-0.05649674  0.5077668   0.0796396   0.04261819 -0.06877445  0.08789572
 -0.0889329   0.01835949  0.13979907  0.05951186  0.14094195]
```

```python
[ ] X = data[['case','site','hdlngth', 'skullw', 'totlngth', 'taill', 'footlgth', 'earconch', 'eye', 'chest', 'belly']]
    y = data.age
    X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=1)
    lm2 = LinearRegression()
    lm2.fit(X_train, y_train)
    y_pred = lm2.predict(X_test)
    print(np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
```

```
1.5766374109418095
```

```python
[ ] X_train1, X_test1, y_train1, y_test1 = train_test_split(X, y, test_size=0.40, random_state=42)
    X_train2, X_test2, y_train2, y_test2 = train_test_split(X, y, test_size=0.30, random_state=42)
    X_train3, X_test3, y_train3, y_test3 = train_test_split(X, y, test_size=0.25, random_state=42)
    X_train4 ,X_test4, y_train4, y_test4 = train_test_split(X, y, test_size=0.20, random_state=42)
```

KNN

60-40

```
from sklearn.neighbors import KNeighborsRegressor
```

```
model=KNeighborsRegressor(n_neighbors=25)
```

```
model.fit(X_train1, y_train1)
```

```
KNeighborsRegressor
KNeighborsRegressor(n_neighbors=25)
```

```
y_pred1 = model.predict(X_test1)
```

```
y_pred1
```

```
array([3.87333333, 4.04      , 4.        , 4.2       , 3.63333333,
       3.72      , 3.72      , 3.87333333, 3.95333333, 3.95333333,
       3.95333333, 3.96      , 3.72      , 3.72      , 3.64      ,
       3.95333333, 3.8       , 3.83333333, 3.95333333, 3.83333333,
       3.72      , 4.2       , 3.95333333, 3.76      , 3.63333333,
       3.8       , 3.95333333, 3.63333333, 3.59333333, 3.95333333,
       3.72      , 3.95333333, 3.72      , 3.64      , 3.83333333,
       3.44      , 3.95333333, 4.        , 4.12      , 3.59333333,
       3.55333333, 3.96      ])
```

```
knn = pd.DataFrame({'Predicted':y_pred1,'Actual':y_test1})
knn
```

SVM

60-40

```
from sklearn.svm import SVR
```

```
model = SVR(kernel='linear')
```

```
model.fit(X_train1, y_train1)
```

```
SVR
SVR(kernel='linear')
```

```
y_pred1 = model.predict(X_test1)
y_pred1
```

```
array([3.78349348, 4.41898078, 3.66060923, 5.09121879, 0.86563047,
       3.10057874, 1.95960846, 4.06685823, 4.37329096, 5.52784653,
       3.42793154, 2.29967115, 3.92479408, 2.55413257, 3.09316111,
       6.03890767, 3.35093645, 2.57073635, 4.88493594, 2.38128164,
       2.84346003, 4.08931164, 3.38687573, 4.16790598, 1.75766679,
       1.81515822, 3.53821012, 1.1654959 , 2.66874514, 5.56223427,
       3.991962  , 4.67050832, 1.96700702, 2.83582736, 1.03266035,
       3.6287386 , 5.15237627, 4.71475792, 5.29207748, 1.97820386,
       1.76123744, 4.38749148])
```

```
svm = pd.DataFrame({'Predicted':y_pred1,'Actual':y_test1})
svm
```

RANDOM FOREST

60-40

```
from sklearn.ensemble import RandomForestRegressor
from sklearn.tree import plot_tree
```

```
rf=RandomForestRegressor()
```

```
rf.fit(X_train1,y_train1)
```

```
RandomForestRegressor
RandomForestRegressor()
```

```
y_pred1=rf.predict(X_test1)
y_pred1
```

```
array([4.045     , 3.3       , 4.62      , 3.34      , 2.51      ,
       4.37      , 2.60666667, 4.19      , 5.24      , 5.37      ,
       3.18      , 3.97      , 4.17      , 3.44      , 2.705     ,
       5.95833333, 3.76      , 2.715     , 3.99      , 2.40833333,
       4.92      , 3.5       , 4.73      , 4.29      , 3.185     ,
       2.66166667, 3.93      , 2.86666667, 3.29      , 4.92      ,
       3.18      , 4.12      , 4.48      , 2.88      , 3.255     ,
       2.945     , 3.83      , 4.28      , 3.56      , 2.875     ,
       3.69666667, 3.79      ])
```

BOOSTING

XGboost

```
import xgboost as xgb
```

60-40

```
model1 = xgb.XGBRegressor()
model2 = xgb.XGBRegressor(n_estimators=100, max_depth=8, learning_rate=0.1, subsample=0.5)

train_model1 = model1.fit(X_train1, y_train1)
train_model2 = model2.fit(X_train1, y_train1)
```

```
pred1 = train_model1.predict(X_test1)
pred2 = train_model2.predict(X_test1)
```

EVALUATION METRICS

```
print("RMSE1:",np.sqrt(metrics.mean_squared_error(y_test1, pred1)))
print("RMSE2:",np.sqrt(metrics.mean_squared_error(y_test1, pred2)))
print("R2 score1:",metrics.r2_score(y_test1,pred1))
print("R2 score2:",metrics.r2_score(y_test1,pred2))
```

```
RMSE1: 1.7318548517013523
RMSE2: 1.9981763626516869
R2 score1: 0.2723557167707735
R2 score2: 0.03135693198142453
```

DEEP LEARNING MODEL(ANN)

AdaBoost

```
[ ] from sklearn.ensemble import AdaBoostRegressor
```

```
[ ] from sklearn.ensemble import AdaBoostRegressor
    from sklearn.tree import DecisionTreeRegressor

    # Replace 'base_estimator' with 'estimator'
    base_estimator = DecisionTreeRegressor(max_depth=3, random_state=0)
    adaboost = AdaBoostRegressor(estimator=base_estimator, # Changed argument name here
                                  n_estimators=3,random_state=0)
```

60-40

```
[ ] adaboost.fit(X_train1, y_train1)
```

```
AdaBoostRegressor
estimator: DecisionTreeRegressor
DecisionTreeRegressor
```

```
[ ] y_pred1 = adaboost.predict(X_test1)
```

EVALUATION METRICS

```
[ ] print("RMSE:",np.sqrt(metrics.mean_squared_error(y_test1,y_pred1)))
    print("R2 score:",metrics.r2_score(y_test1,y_pred1))
```

```
RMSE: 1.9278896842681346
R2 score: 0.09830326007898671
```

NN

```
[ ] import tensorflow as tf
```

60-40 Train Test Split

Epochs=100,optimizer=Adam

```
tf.random.set_seed(42)

# STEP1: Creating the model

model= tf.keras.Sequential([
                            tf.keras.layers.Dense(10),


                            tf.keras.layers.Dense(5),


                            tf.keras.layers.Dense(1)
])

# STEP2: Compiling the model   # optimizer can be SGD, Adam

model.compile(loss= tf.keras.losses.mae,
              optimizer= tf.keras.optimizers.Adam(),  #SGD
              metrics= ["mae"])

# STEP3: Fit the model

history= model.fit(X_train1, y_train1, epochs= 100, verbose=1)
```

```
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 2.2349 - mae: 2.2349
```