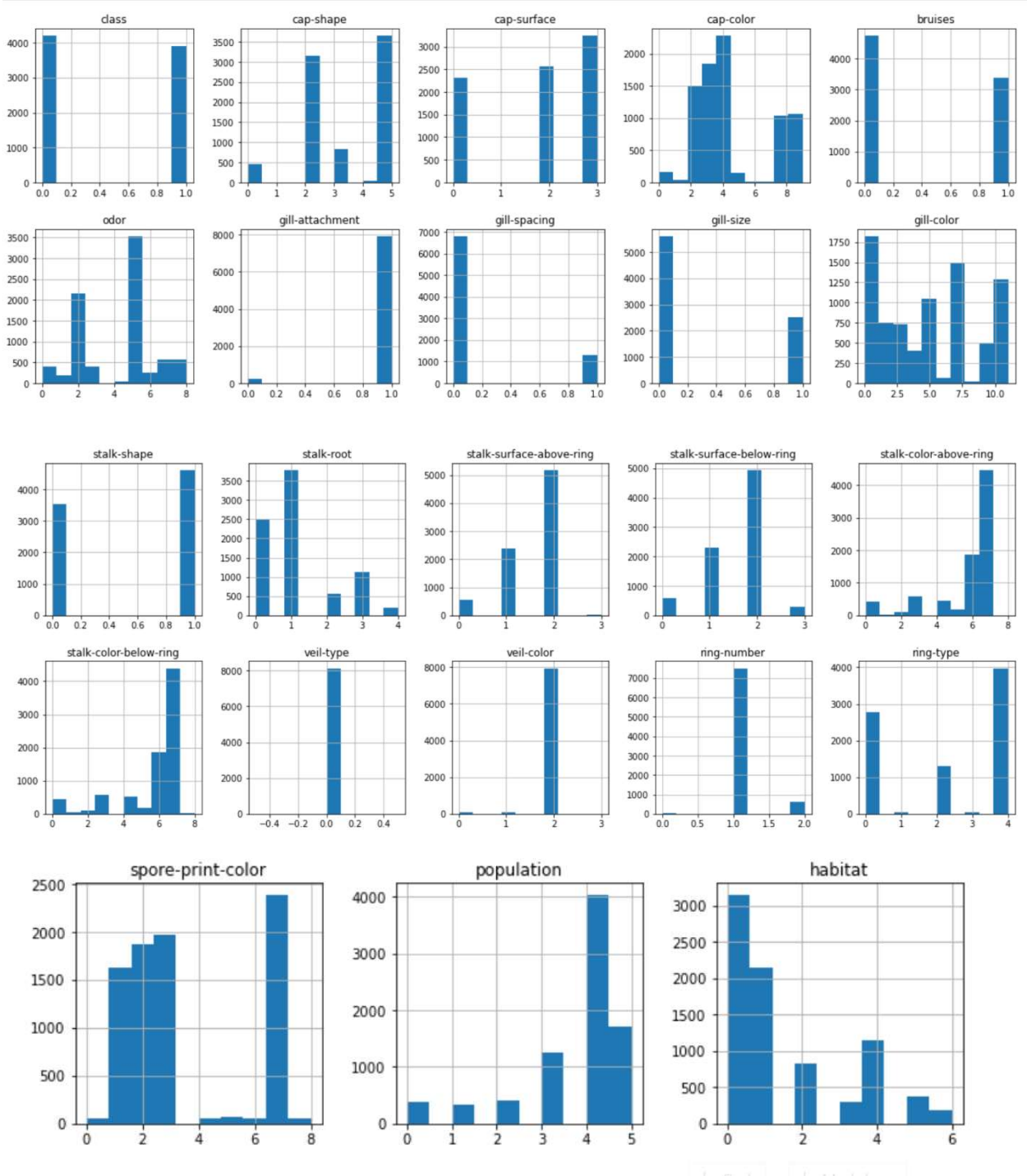
EDA for Mushroom Dataset

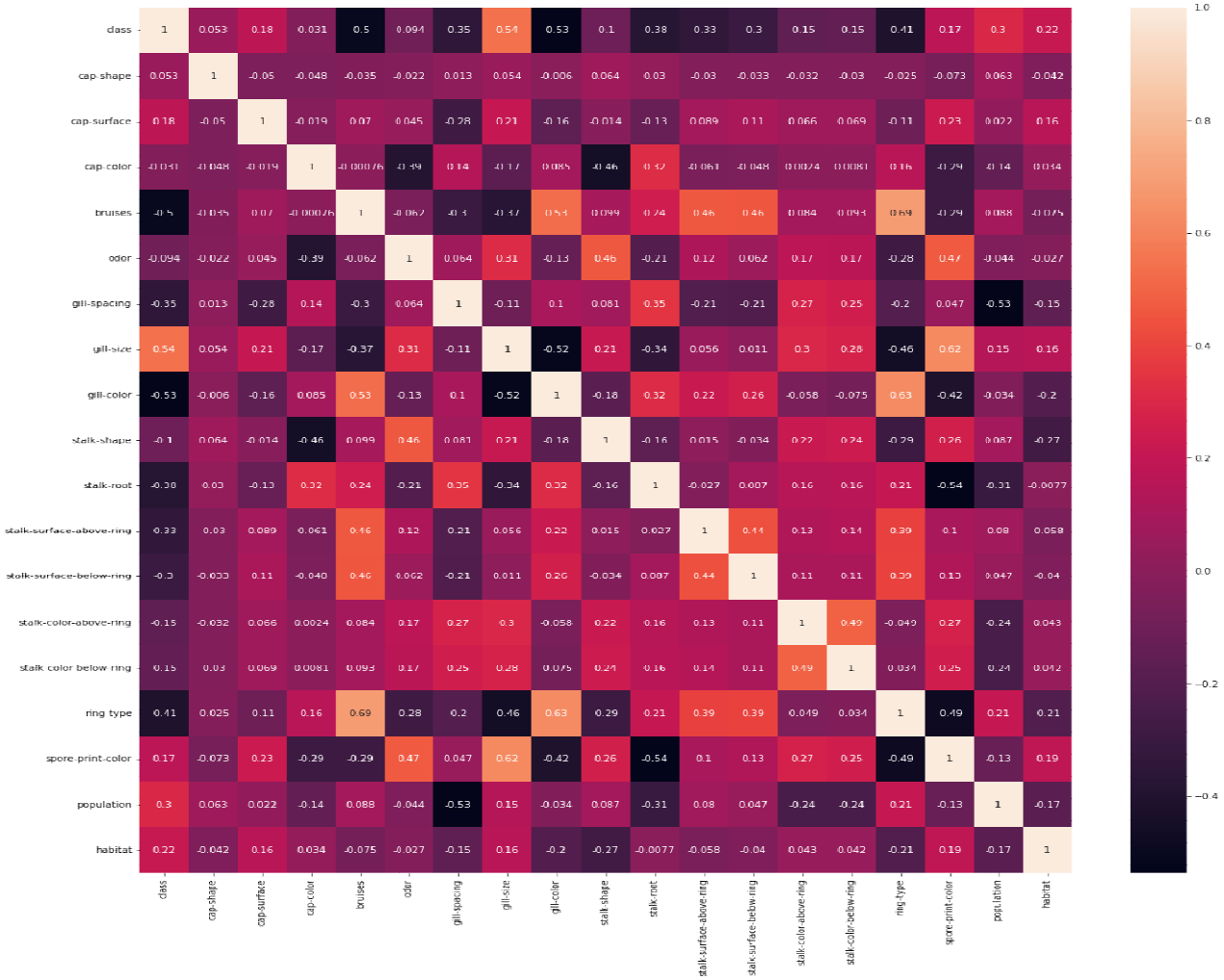
Histogram

```
data.hist(figsize=(20,20));
```



Heatmap

```
plt.figure(figsize = (20,20))
sns.heatmap(data.corr(), annot=True);
```



Classification

Installing and Importing Libraries

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

Reading Data

```
DATA = pd.read_csv('../datasets/mushrooms.csv')
DATA.head()

DATA.shape
```

Checking for missing values

```
DATA.isna().sum().sort_values(ascending=False)
```

```
class                       0
stalk-surface-above-ring    0
population                  0
spore-print-color           0
ring-type                   0
ring-number                 0
veil-color                  0
veil-type                   0
stalk-color-below-ring      0
stalk-color-above-ring      0
stalk-surface-below-ring    0
stalk-root                  0
cap-shape                   0
stalk-shape                 0
gill-color                  0
gill-size                   0
gill-spacing                0
gill-attachment             0
odor                        0
bruises                     0
cap-color                   0
cap-surface                 0
habitat                     0
dtype: int64
```

Label Encoding

```
from sklearn.preprocessing import LabelEncoder
encoder = LabelEncoder()

data = pd.DataFrame()
for col in DATA.columns:
  data[col] = encoder.fit_transform(DATA[col])
```

Feature Selection

```
from feature_engine.selection import SmartCorrelatedSelection,
DropConstantFeatures, DropDuplicateFeatures
from sklearn.pipeline import Pipeline
```

```
pip = Pipeline([('constant', DropConstantFeatures(tol=0.9)), ('duplicate',
DropDuplicateFeatures()), ('correlated', SmartCorrelatedSelection())])
data = pip.fit_transform(data)
data.shape
```

ML Model

```
X = data.drop('class', axis=1)
Y = data['class']
X.shape, Y.shape
```

```
((8124, 18), (8124,))
```

```
from sklearn.model_selection import train_test_split
xTrain, xTest, yTrain, yTest = train_test_split(X, Y)
```

1. Naïve Bayes

```
from sklearn.naive_bayes import GaussianNB
nb = GaussianNB()
nb.fit(xTrain, yTrain)
```

```
print("Training Accuracy: ",nb.score(xTrain, yTrain))
print("Testing Accuracy: ",nb.score(xTest, yTest))
```

```
Training Accuracy:  0.8952896766781553
Testing Accuracy:  0.8862629246676514
```

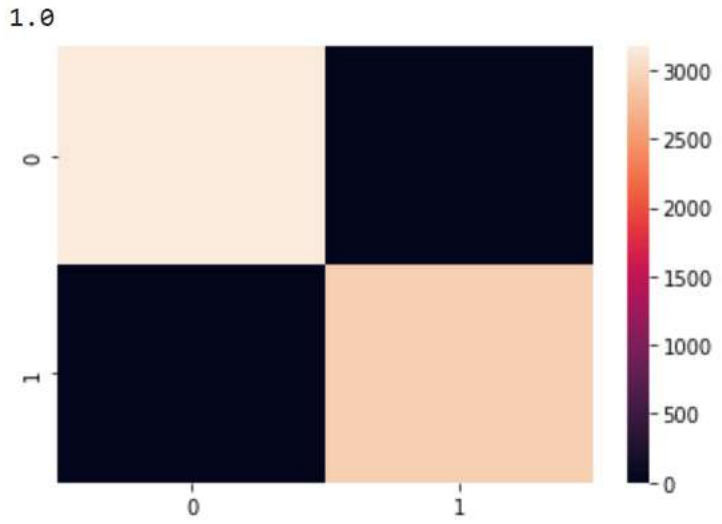
2. KNN

```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=1)
knn.fit(xTrain, yTrain)
yPred1 = knn.predict(xTrain)
yPred2 = knn.predict(xTest)
```

```
from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report
matrix = confusion_matrix(yTrain, yPred1)
print(classification_report(yTrain, yPred1))
sns.heatmap(matrix)
accuracy_score(yTrain, yPred1)
```

Result 1

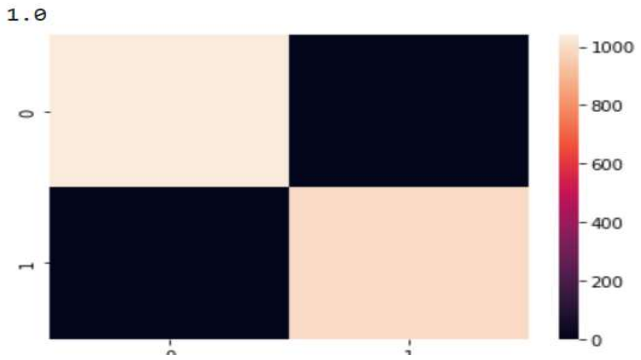
	precision	recall	f1-score	support
0	1.00	1.00	1.00	3169
1	1.00	1.00	1.00	2924
accuracy			1.00	6093
macro avg	1.00	1.00	1.00	6093
weighted avg	1.00	1.00	1.00	6093



Result 2

```
matrix = confusion_matrix(yTest, yPred2)
print(classification_report(yTest, yPred2))
sns.heatmap(matrix)
accuracy_score(yTest, yPred2)
```

	precision	recall	f1-score	support
0	1.00	1.00	1.00	1039
1	1.00	1.00	1.00	992
accuracy			1.00	2031
macro avg	1.00	1.00	1.00	2031
weighted avg	1.00	1.00	1.00	2031

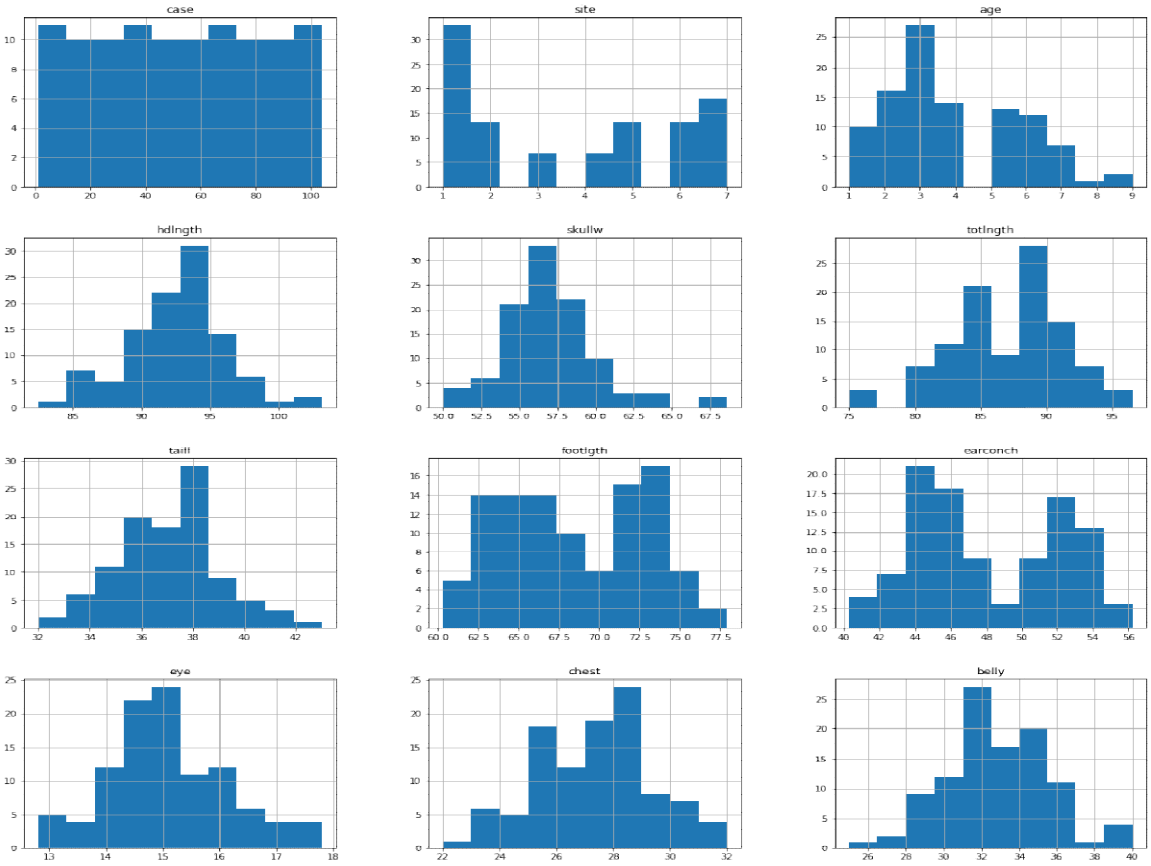


Comparison:

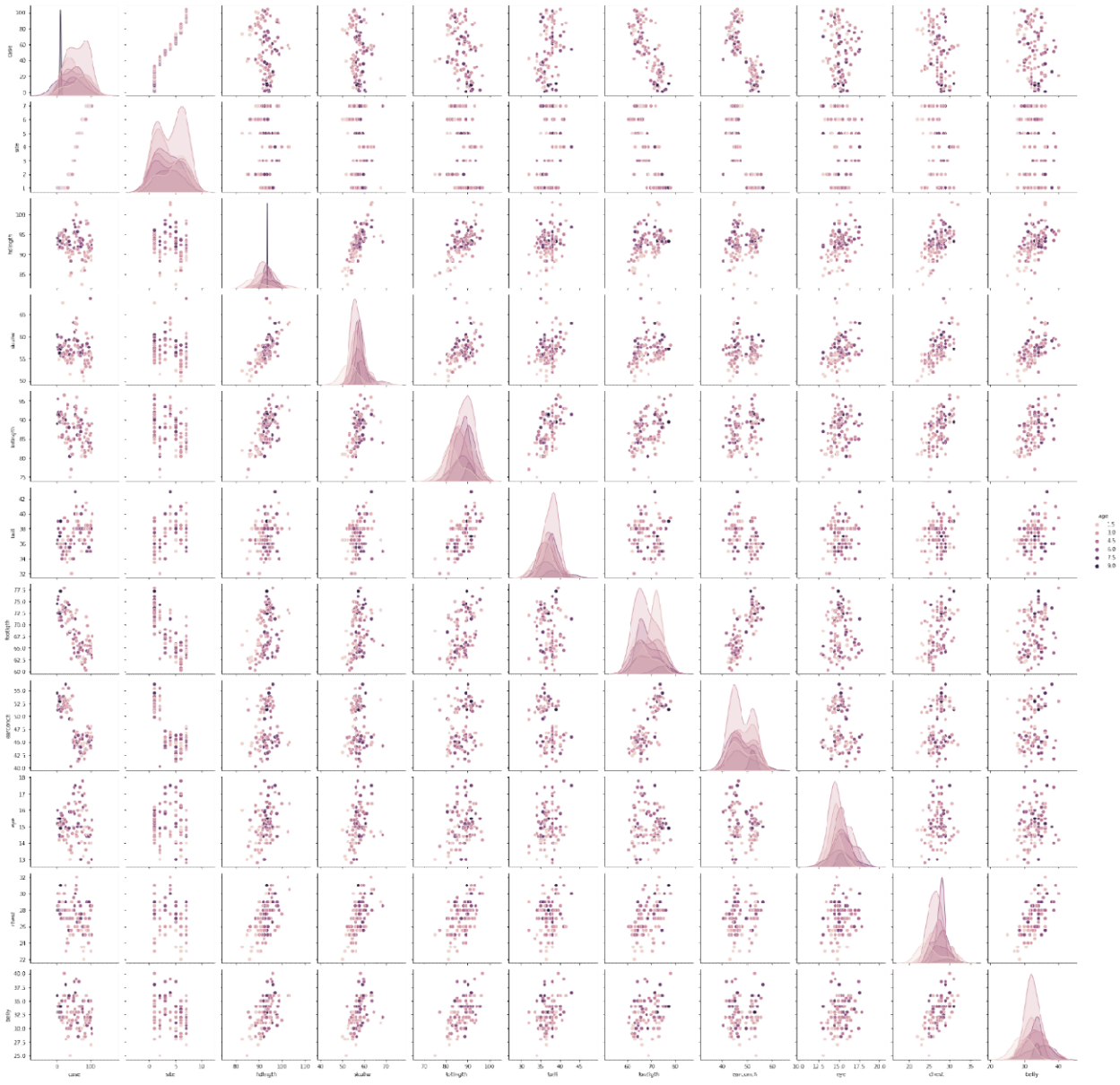
By analyzing all the steps of EDA and doing all the preprocessing, training-testing steps we can observed that both the models can predict the output with the same accuracy.

EDA for Possum Dataset

```
DATA.hist(figsize=(20,20));
```

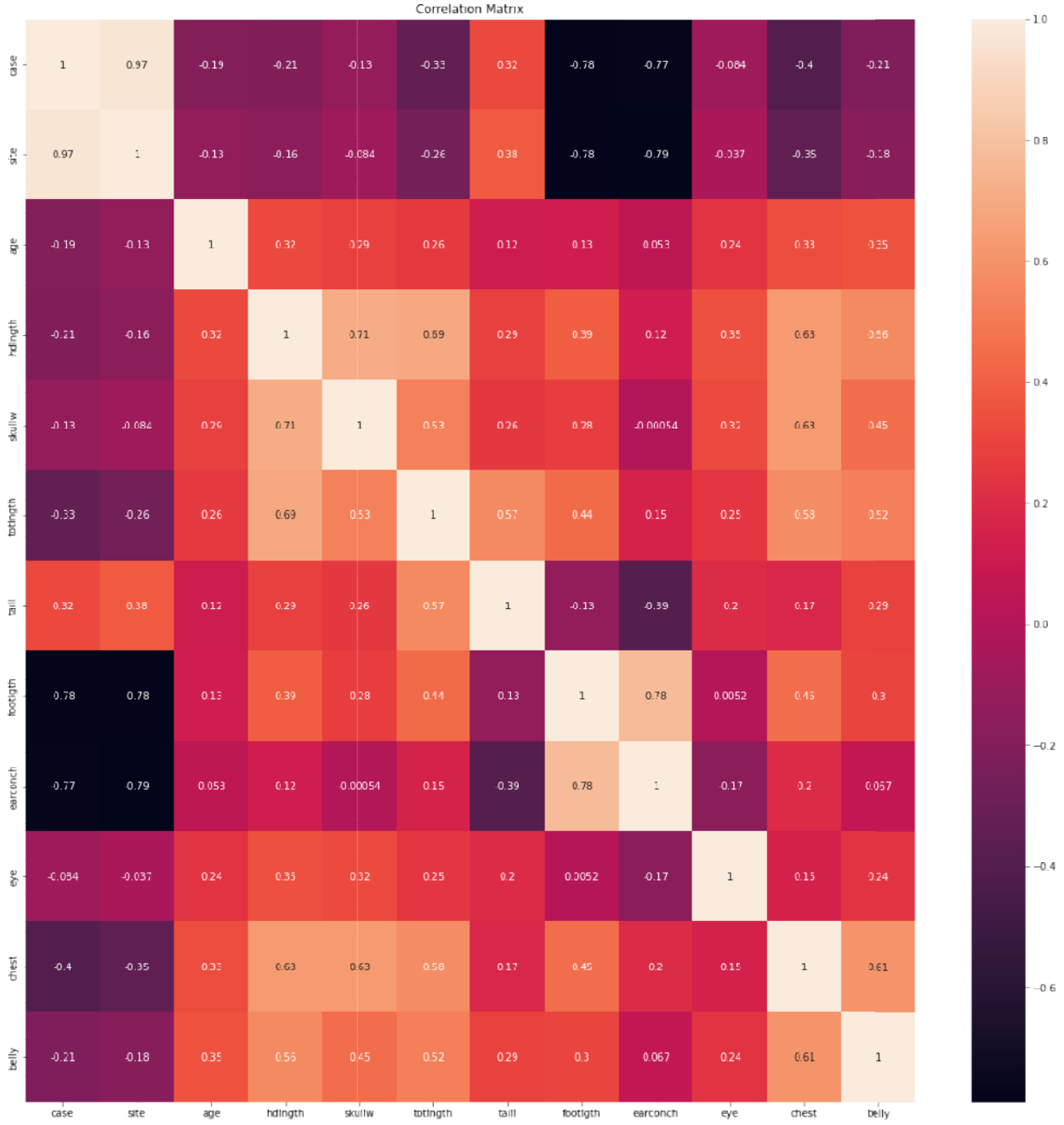



```
sns.pairplot(DATA,hue='age');
```



```
corr = DATA.corr()
plt.figure(figsize = (20,20))
```

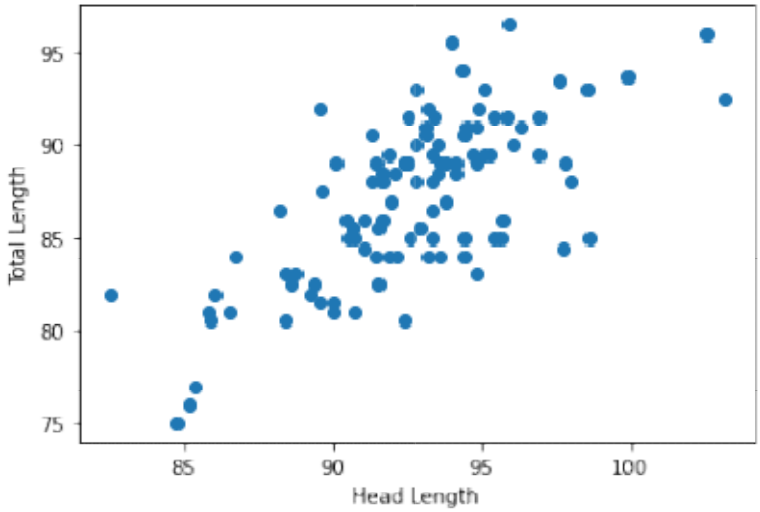
```
sns.heatmap(corr, annot=True);
plt.title("Correlation Matrix");
plt.show()
```



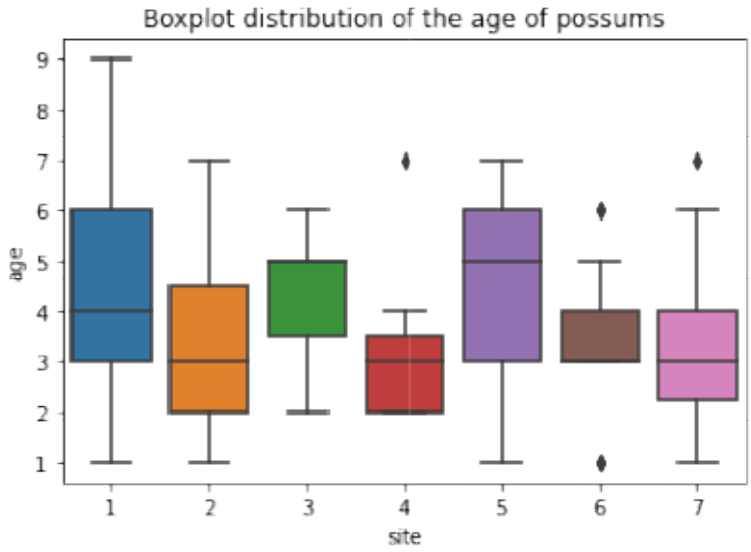

```
plt.scatter(DATA['hdlngth'], DATA['totlngth'])
plt.xlabel('Head Length')
```

```
plt.ylabel('Total Length')
```

```
Text(0, 0.5, 'Total Length')
```




```
sns.boxplot(x="site", y="age", data= DATA);
plt.title("Boxplot distribution of the age of possums");
```



Regression

```
DATA = pd.read_csv('../datasets/possum.csv')
DATA
```

	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
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
99	100	7	other	m	1.0	89.5	56.0	81.5	36.5	66.0	46.8	14.8	23.0	27.0
100	101	7	other	m	1.0	88.6	54.7	82.5	39.0	64.4	48.0	14.0	25.0	33.0
101	102	7	other	f	6.0	92.4	55.0	89.0	38.0	63.5	45.4	13.0	25.0	30.0
102	103	7	other	m	4.0	91.5	55.2	82.5	36.5	62.9	45.9	15.4	25.0	29.0
103	104	7	other	f	3.0	93.6	59.9	89.0	40.0	67.6	46.0	14.8	28.5	33.5

104 rows × 14 columns

Null Values

```
DATA.isna().sum().sort_values(ascending=False)
```

```
age         2
footlgth    1
case        0
site        0
Pop         0
sex         0
hdlngth     0
skullw      0
totlngth    0
taill       0
earconch    0
eye         0
chest       0
belly       0
dtype: int64
```

```
DATA['footlgth'] = DATA['footlgth'].fillna(DATA['footlgth'].mean())
DATA = DATA.dropna()
```

```
DATA.isna().any()
```

```
case        False
site        False
Pop         False
sex         False
age         False
hdlngth     False
skullw      False
totlngth    False
taill       False
footlgth    False
earconch    False
eye         False
chest       False
belly       False
dtype: bool
```

Label Encoding

```
from sklearn.preprocessing import LabelEncoder
encoder = LabelEncoder()

DATA['Pop'] = encoder.fit_transform(DATA['Pop'])
DATA['sex'] = encoder.fit_transform(DATA['sex'])
DATA.head()
```

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

Scaling

```
DATA = DATA.drop('case', axis=1)
X = DATA.drop('age', axis=1)
Y = DATA['age']

from sklearn.preprocessing import StandardScaler
std = StandardScaler()
x = std.fit_transform(X)
```

Feature Engineering

```
from feature_engine.selection import SmartCorrelatedSelection,
DropConstantFeatures, DropDuplicateFeatures
from sklearn.pipeline import Pipeline
```

```
pip = Pipeline([('constant', DropConstantFeatures(tol=0.9)), ('duplicate',
DropDuplicateFeatures()), ('correlated', SmartCorrelatedSelection())])
x = pip.fit_transform(x)
x.shape
```

Model Selection

```
from sklearn.metrics import r2_score, mean_squared_error
def evaluate(y_true, y_hat, label='test'):
    mse = mean_squared_error(y_true, y_hat)
    rmse = np.sqrt(mse)
    variance = r2_score(y_true, y_hat)
    print('{} set RMSE:{}, R2:{}'.format(label, rmse, variance))

from sklearn.model_selection import train_test_split
xTrain, xTest, yTrain, yTest = train_test_split(x, Y)
```

1. Linear Regression

```
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(xTrain, yTrain)

y_hat_train = model.predict(xTrain)
evaluate(yTrain, y_hat_train)
```

Result

```
y_hat_test = model.predict(xTest)
evaluate(yTest, y_hat_test)
```

```
test set RMSE:1.723667912982484, R2:0.2808969374437904
test set RMSE:1.7910153338253658, R2:-0.8176273981344242
```

2. SVM

```
from sklearn.svm import SVR
svr_f = SVR()
svr_f.fit(xTrain, yTrain)

y_hat_train = svr_f.predict(xTrain)
evaluate(yTrain, y_hat_train, 'train')
```

Result

```
y_hat_test = svr_f.predict(xTest)
evaluate(yTest, y_hat_test)
```

```
train set RMSE:1.207361762670402, R2:0.548172539182358
test set RMSE:1.987615324010251, R2:0.13962128829301113
```

Comparison:

It can be observed that both the models can predict the output with high accuracy, but the Linear Regression model produces a bit more error than the SVM model. Hence, SVM has better accuracy.