Usama Arif Rollno 14

Task 1: Read given data into DataFrame in python “Cat_Human.csv”. Perform Data cleaning.

In [62]:
```
import pandas as pd
from sklearn.preprocessing import MinMaxScaler,OneHotEncoder,LabelEncoder
from sklearn.compose import ColumnTransformer
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score
import seaborn as sns
import matplotlib.pyplot as plt

```

In [9]:
```
Cat_human=pd.read_csv('Cat_human.csv')
Cat_human.head()
```

Out[9]:

	Color	Eye_color	Height	Legs	Moustache	Tail	Weight	label
0	No	black	5.14	2	No	No	70.0	human
1	No	brown	6.80	2	No	No	64.4	human
2	Yes	brown	5.00	2	Yes	No	64.8	human
3	No	blue	5.90	2	No	No	78.8	human
4	No	blue	6.56	2	No	No	73.2	human

In [13]:
```

```

In [32]:
```
Cat_human.isna().sum()
```

Out[32]:
```
Color        0
Eye_color    0
Height       0
Legs         0
Moustache    0
Tail         0
Weight       0
label        0
dtype: int64
```

Task 2: After data cleaning, you are required to prepare your dataset for training.

Transformation to numeric data

In [33]:
```
x=Cat_human.drop('label',axis=1)
y=Cat_human['label']
```

In [28]:
```
model=RandomForestClassifier()
one_hot=OneHotEncoder()
features=['Color','Eye_color','Moustache','Tail']
transformer=ColumnTransformer([('one_hot',one_hot,features)],remainder='pa
transformed_x=transformer.fit_transform(x)
#transforming y using label encoder
encoder=LabelEncoder()
y_transformed=encoder.fit_transform(y)
```

In [30]:
```
scaler=MinMaxScaler()
x_scaled=scaler.fit_transform(transformed_x)
x_scaled
```

Out[30]:
```
array([[1.        , 0.        , 0.        , ..., 0.71604938, 0.        ,
        0.87863464],
       [1.        , 0.        , 0.        , ..., 0.97222222, 0.        ,
        0.80783818],
       [0.        , 1.        , 0.        , ..., 0.69444444, 0.        ,
        0.81289507],
       ...,
       [0.        , 0.        , 0.        , ..., 0.00462963, 1.        ,
        0.07414237],
       [0.        , 0.        , 0.        , ..., 0.08179012, 1.        ,
        0.076994  ],
       [0.        , 0.        , 0.        , ..., 0.05092593, 1.        ,
        0.09315324]])
```

In [31]:
```
x_train,x_test,y_train,y_test=train_test_split(x_scaled,y_transformed,test
model.fit(x_test,y_test)
```

Out[31]:
```
RandomForestClassifier()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [34]:
```
x_train
```

Out[34]:
```
array([[0.        , 1.        , 0.        , ..., 0.95987654, 0.        ,
        0.66624526],
       [0.        , 0.        , 0.        , ..., 0.00154321, 1.        ,
        0.0418239 ],
       [1.        , 0.        , 0.        , ..., 0.91975309, 0.        ,
        0.83817952],
       ...,
       [0.        , 1.        , 0.        , ..., 0.91358025, 0.        ,
        0.66118837],
       [0.        , 1.        , 0.        , ..., 0.83641975, 0.        ,
        0.52970923],
       [1.        , 0.        , 0.        , ..., 0.82716049, 0.        ,
        0.67635904]])
```

In [35]:
```
x_test

```

Out[35]:
```
array([[0.        , 0.        , 0.        , 1.        , 0.        ,
        0.        , 0.        , 0.        , 1.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 1.        ,
        0.        , 1.        , 0.0462963 , 1.        , 0.00570326],
       [0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 1.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 1.        , 0.        , 1.        ,
        0.        , 1.        , 0.04938272, 1.        , 0.0636864 ],
       [0.        , 0.        , 0.        , 1.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 1.        , 0.        , 1.        ,
        0.        , 1.        , 0.08950617, 1.        , 0.02091195],
       [1.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 1.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 1.        , 0.        ,
        1.        , 0.        , 0.7345679 , 0.        , 0.99494311],
       [0.        , 1.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 1.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 1.        ,
```

In [36]:
```
y_train
```

Out[36]:
```
array([1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0,
       1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0,
       0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1,
       1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0,
       1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0,
       1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1,
       0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0,
       0, 1, 0, 1, 1, 1])
```

In [37]:
```
    y_test
```

Out[37]:
```
array([0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0,
       1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1])
```

In [39]:
```
1 y_preds=model.predict(x_test)
```

Task 3: Display confusion matrix and generate report of f1-score, recall and precision.

In [40]:
```
1 cm=confusion_matrix(y_test,y_preds)
2 cm
```

Out[40]:
```
array([[20,  0],
       [ 0, 20]], dtype=int64)
```

In [54]:
```
1 f1_score=f1_score(y_test,y_preds)
2 f1_score
```

Out[54]: 1.0

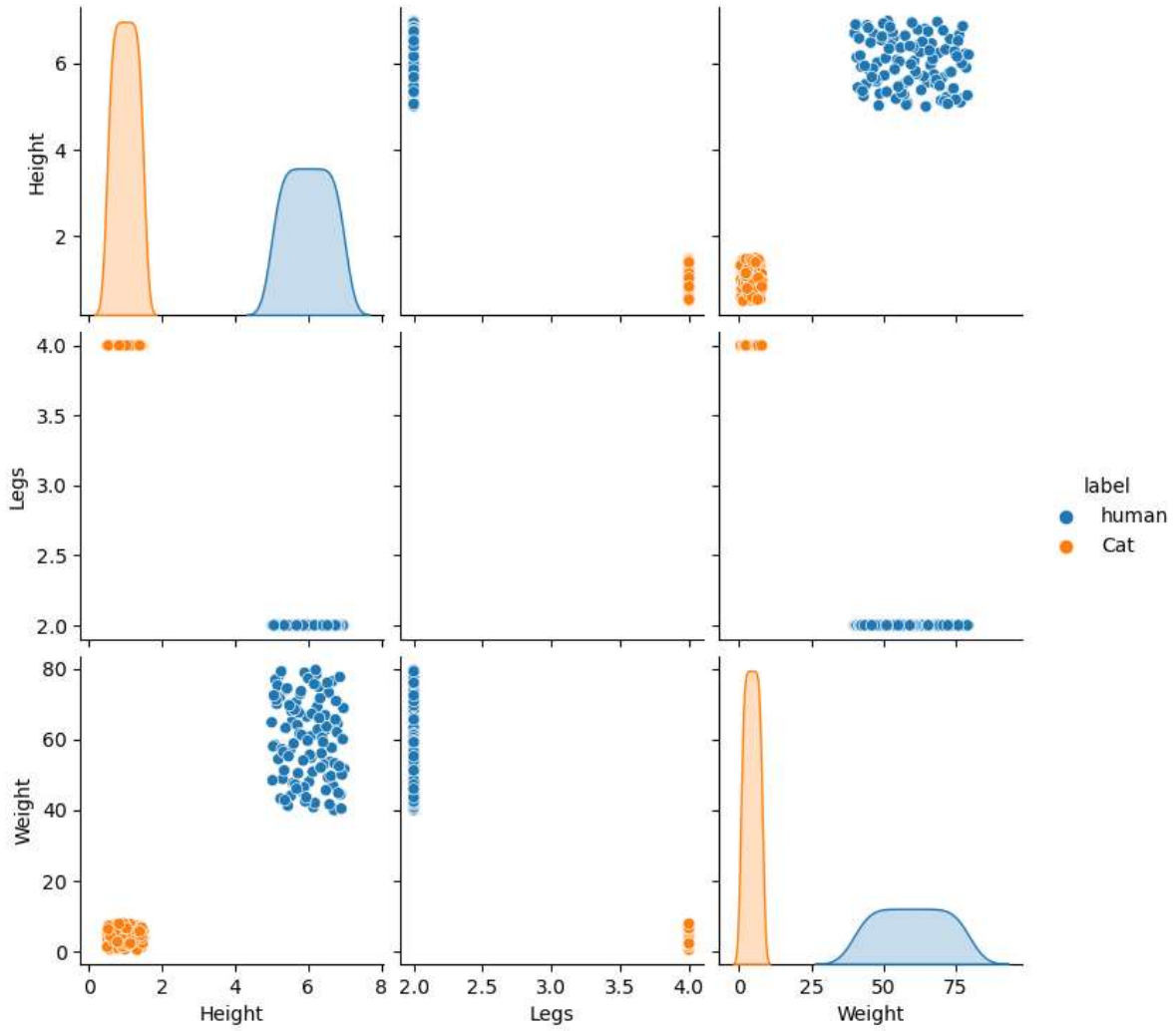
In [50]:
```
1 recall_score=recall_score(y_test,y_preds, average='weighted')
2 recall_score
3
```

Out[50]: 1.0

In [83]:
```
1 p_score=precision_score(y_preds,y_test)
2 p_score
3
```

Out[83]: 1.0

In [63]:
```
1 sns.pairplot(Cat_human, hue='label')
2 plt.show()
```



In [ ]:
```
1
```

In [ ]:
```
1
```