In [1]:
```
import pandas as pd
import numpy as np
```

task 01

In [2]:
```
#reading data
data = pd.read_csv("Cat_human.csv")
data
```

Out[2]:

	Color	Eye_color	Height	Legs	Moustache	Tail	Weight	label
0	No	black	5.14	2	No	No	70.000000	human
1	No	brown	6.80	2	No	No	64.400000	human
2	Yes	brown	5.00	2	Yes	No	64.800000	human
3	No	blue	5.90	2	No	No	78.800000	human
4	No	blue	6.56	2	No	No	73.200000	human
...	...	...	...	...	...	...	...	...
195	brown	gray	1.14	4	Yes	Yes	2.304511	Cat
196	white	yellow	1.39	4	Yes	Yes	5.687970	Cat
197	white	black	0.53	4	Yes	Yes	6.364662	Cat
198	brown	green	1.03	4	Yes	Yes	6.590226	Cat
199	brown_white	blue	0.83	4	Yes	Yes	7.868421	Cat

200 rows × 8 columns

In [3]:
```
#checking if the data is clean or not
data.isna().sum()
```

Out[3]:
```
Color        0
Eye_color    0
Height       0
Legs         0
Moustache    0
Tail         0
Weight       0
label        0
dtype: int64
```

In [5]:
```
#seperatting x and y
x = data.drop("label", axis = 1)
y = data["label"]
```

In [6]:
```
x , y
```

Out[6]:
```
(          Color Eye_color  Height  Legs Moustache Tail     Weight
 0            No     black    5.14     2        No   No  70.000000
 1            No     brown    6.80     2        No   No  64.400000
 2           Yes     brown    5.00     2       Yes   No  64.800000
 3            No      blue    5.90     2        No   No  78.800000
 4            No      blue    6.56     2        No   No  73.200000
 ..          ...       ...     ...   ...       ...  ...        ...
 195       brown      gray    1.14     4       Yes  Yes   2.304511
 196       white    yellow    1.39     4       Yes  Yes   5.687970
 197       white     black    0.53     4       Yes  Yes   6.364662
 198       brown     green    1.03     4       Yes  Yes   6.590226
 199  brown_white      blue    0.83     4       Yes  Yes   7.868421

 [200 rows x 7 columns],
 0      human
 1      human
 2      human
 3      human
 4      human
        ...
 195      Cat
 196      Cat
 197      Cat
 198      Cat
 199      Cat
 Name: label, Length: 200, dtype: object)
```

task 02

In [24]:
```
from sklearn.preprocessing import MinMaxScaler, LabelEncoder
```

In [8]:
```
scaling = MinMaxScaler()
x_scaled = scaling.fit_transform(x)
```

```
    883     ) from complex_warning

File C:\ProgramData\anaconda3\lib\site-packages\sklearn\utils\_array_api.p
y:185, in _asarray_with_order(array, dtype, order, copy, xp)
    182     xp, _ = get_namespace(array)
    183 if xp.__name__ in {"numpy", "numpy.array_api"}:
    184     # Use NumPy API to support order
--> 185     array = numpy.asarray(array, order=order, dtype=dtype)
    186     return xp.asarray(array, copy=copy)
    187 else:

File C:\ProgramData\anaconda3\lib\site-packages\pandas\core\generic.py:207
0, in NDFrame.__array__(self, dtype)
   2069 def __array__(self, dtype: npt.DTypeLike | None = None) -> np.ndarr
ay:
-> 2070     return np.asarray(self._values, dtype=dtype)

ValueError: could not convert string to float: 'No'
```

as MinMaxScaler excpects input as numerical but our features are in cataogorical so we'll use onehot encocoser to make it numerical

In [25]:
```
label_encoder = LabelEncoder()
```

In [30]:
```
c = label_encoder.fit_transform(data["Color"])
data["Color"] = pd.Series(c)

e_c = label_encoder.fit_transform(data["Eye_color"])
data["Eye_color"] = pd.Series(e_c)

m = label_encoder.fit_transform(data["Moustache"])
data["Moustache"] = pd.Series(m)

t = label_encoder.fit_transform(data["Tail"])
data["Tail"] = pd.Series(t)
```

In [31]:
```
data
```

Out[31]:

	Color	Eye_color	Height	Legs	Moustache	Tail	Weight	label
0	0	0	5.14	2	0	0	70.000000	human
1	0	2	6.80	2	0	0	64.400000	human
2	1	2	5.00	2	1	0	64.800000	human
3	0	1	5.90	2	0	0	78.800000	human
4	0	1	6.56	2	0	0	73.200000	human
...	...	...	...	...	...	...	...	...
195	3	3	1.14	4	1	1	2.304511	Cat
196	6	5	1.39	4	1	1	5.687970	Cat
197	6	0	0.53	4	1	1	6.364662	Cat
198	3	4	1.03	4	1	1	6.590226	Cat
199	4	1	0.83	4	1	1	7.868421	Cat

200 rows × 8 columns

In [33]:
```
#seperatting x and y
x = data.drop("label", axis = 1)
y = data["label"]
```

In [34]:
```
x
```

Out[34]:

	Color	Eye_color	Height	Legs	Moustache	Tail	Weight
0	0	0	5.14	2	0	0	70.000000
1	0	2	6.80	2	0	0	64.400000
2	1	2	5.00	2	1	0	64.800000
3	0	1	5.90	2	0	0	78.800000
4	0	1	6.56	2	0	0	73.200000
...	...	...	...	...	...	...	...
195	3	3	1.14	4	1	1	2.304511
196	6	5	1.39	4	1	1	5.687970
197	6	0	0.53	4	1	1	6.364662
198	3	4	1.03	4	1	1	6.590226
199	4	1	0.83	4	1	1	7.868421

200 rows × 7 columns

In [35]:
```
y_encoded = label_encoder.fit_transform(y)
y_encoded
```

Out[35]:
```
array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0])
```

In [38]:
```
x_scaled = scaling.fit_transform(x)
x_scaled
```

Out[38]:
```
array([[0.        , 0.        , 0.71604938, ..., 0.        , 0.        ,
        0.87863464],
       [0.        , 0.4       , 0.97222222, ..., 0.        , 0.        ,
        0.80783818],
       [0.16666667, 0.4       , 0.69444444, ..., 1.        , 0.        ,
        0.81289507],
       ...,
       [1.        , 0.        , 0.00462963, ..., 1.        , 1.        ,
        0.07414237],
       [0.5       , 0.8       , 0.08179012, ..., 1.        , 1.        ,
        0.076994  ],
       [0.66666667, 0.2       , 0.05092593, ..., 1.        , 1.        ,
        0.09315324]])
```

In [40]:
```
from sklearn.model_selection import train_test_split
```

In [41]:
```
np.random.seed(42)
x_train ,x_test ,y_train ,y_test = train_test_split(x_scaled ,y_encoded ,test_s
```

task 3

In [55]:
```
from sklearn.linear_model import LogisticRegression
import matplotlib.pyplot as plt
```

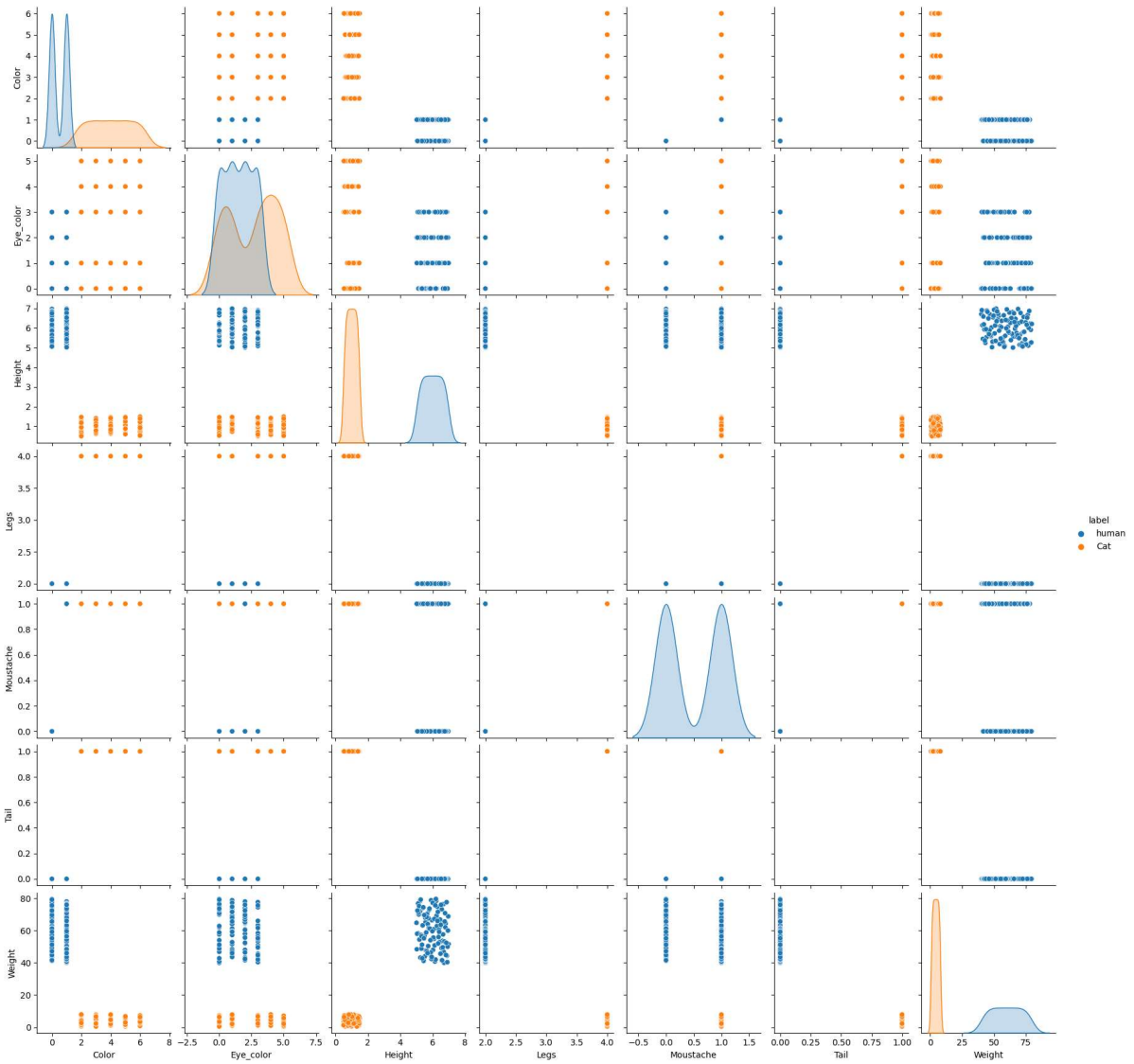
In [43]:
```
model = LogisticRegression()
model.fit(x_train, y_train)
```

Out[43]:
```
LogisticRegression
LogisticRegression()
```

In [52]:
```
import accuracy_score, precision_score, f1_score,recall_score, confusion_matrix
```

In [56]:
```
import seaborn as sns
# Predict test set
y_pred = model.predict(x_test)
#accuracy score
accuracy = accuracy_score(y_test, y_pred)
print('Accuracy:', accuracy)
# Generate confusion matrix
cm = confusion_matrix(y_test, y_pred)
print('Confusion Matrix:')
print(cm)
#precision score
precision = precision_score(y_test, y_pred, average='weighted')
print('Precision:', precision)
#recall score
recall = recall_score(y_test, y_pred, average='weighted')
print('Recall:', recall)
# Calculate F1 score
f1 = f1_score(y_test, y_pred, average='weighted')
print('F1 Score:', f1)
# Visualize the dataset
sns.pairplot(data, hue='label')
plt.show()
```

```
Accuracy: 1.0
Confusion Matrix:
[[19  0]
 [ 0 21]]
Precision: 1.0
Recall: 1.0
F1 Score: 1.0
```


Color
Eye_color
Height
Legs
Moustache
Tail
Weight
label
human
Cat

In [ ]: