In [ ]:

In [3]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler,LabelEncoder
from sklearn.metrics import accuracy_score ,confusion_matrix, precision_score ,recall_score,f1_score ,mean_sq
```

Task 1

In [4]:
```
#Task 1: Read given data into DataFrame in python “Cat_Human.csv”. Perform Data cleaning.
data = pd.read_csv("Cat_human.csv")

data = data.loc[:, ~data.columns.duplicated()]
data = data.drop_duplicates()

data = data.dropna(axis=1, how = 'all')
data =data.dropna(axis = 0,how='all')
data
```

Out[4]:

	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

Task 2

In [6]:
```
#Task 2: After data cleaning, you are required to prepare your dataset for training.
#• Separate features and labels.
#• Feature scaling/Normalization
#• Perform Label Encoding
#• Split dataset into training and testing data

label_encoder = LabelEncoder()

tail = data['Tail']
t = label_encoder.fit_transform(tail)
data['Tail'] = pd.Series(t)

color = data['Color']
c = label_encoder.fit_transform(color)
data['Color'] = pd.Series(c)


eyeColor = data['Eye_color']
eC = label_encoder.fit_transform(eyeColor)
data['Eye_color'] = pd.Series(eC)

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



X = data[['Color','Eye_color','Height', 'Legs','Moustache', 'Weight']]
y = data['label']


#apply minmaxscaler to normalize feature

scaler =MinMaxScaler()
x_scaled= scaler.fit_transform(X)

#label encoder for string values
encoder = LabelEncoder()
y_encoded=encoder.fit_transform(y)

#split the data into x_train ,x_test ,y_train ,y_test
x_train ,x_test ,y_train ,y_test = train_test_split(x_scaled ,y_encoded  ,test_size = 0.2 ,random_state = 20
```

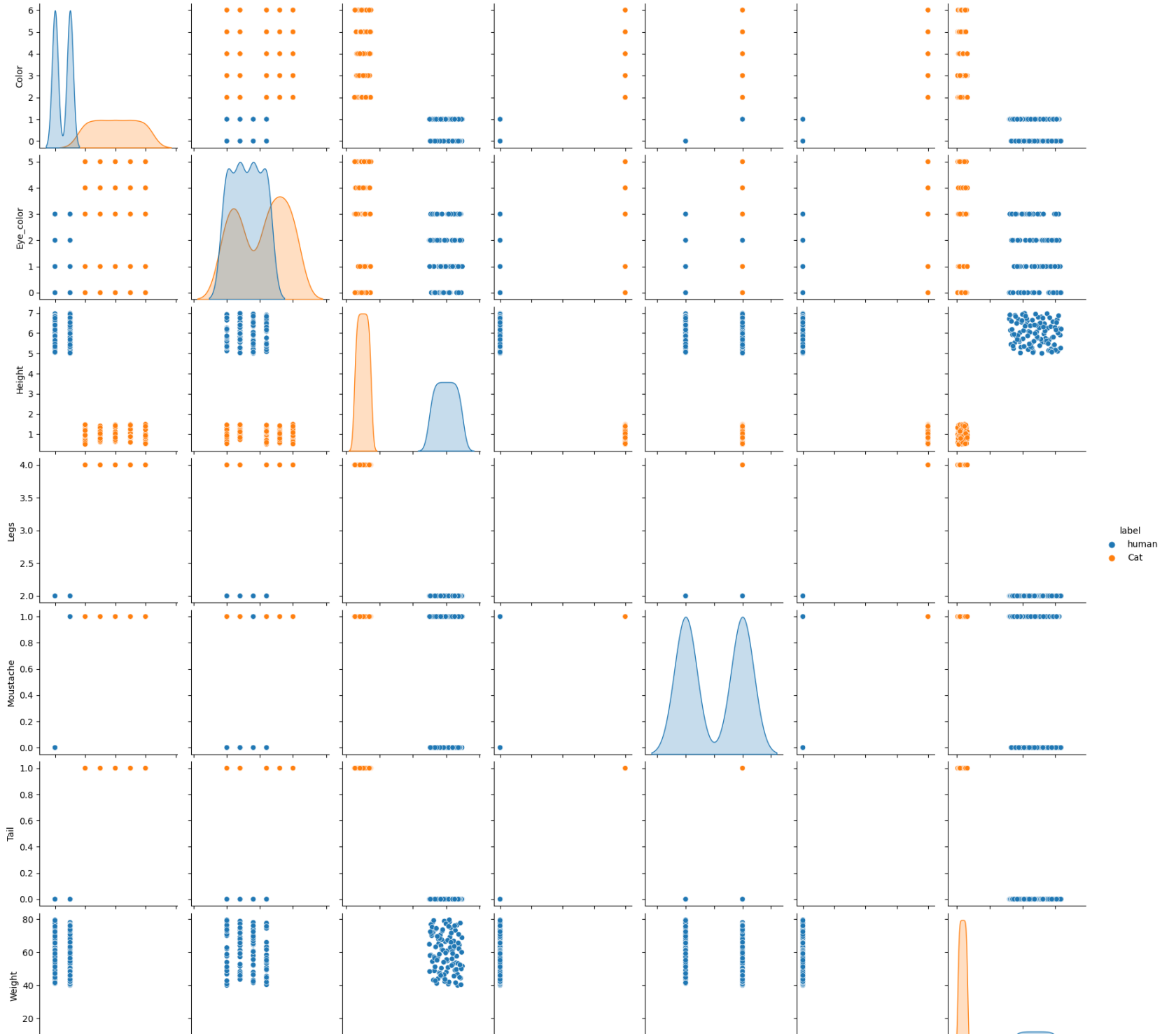
```
print('\n  X_train \n',x_train)
print('\n X_test \n',x_test)
print('\n y_train \n',y_train)
print('\n y_test\n',y_test)
```

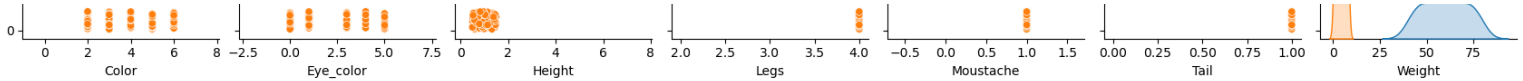
```
[0.16666667 0.6        0.94444444 0.         1.         0.62073325]
[0.         0.2        0.73765432 0.         0.         0.71681416]
[0.5        0.6        0.0308642  1.         1.         0.08269726]
[0.16666667 0.6        0.74382716 0.         1.         0.70670038]
[0.5        0.2        0.04012346 1.         1.         0.09125215]
[0.         0.         0.76234568 0.         0.         0.51453856]
[0.16666667 0.         0.81481481 0.         1.         0.91403287]
[0.5        0.8        0.02777778 1.         1.         0.05323042]]

y_train
[0 1 0 1 1 0 0 0 0 0 1 0 1 0 1 1 1 1 0 1 0 1 1 1 1 0 1 0 1 1 1 0 1 0 0 1 1
1 1 0 1 0 1 1 1 0 1 0 1 0 0 1 1 1 1 1 0 0 1 0 1 1 1 0 1 1 0 0 0 1 1 1 0 0
1 1 1 1 0 0 1 1 0 1 1 0 1 0 1 1 0 0 1 0 0 0 1 0 0 0 0 1 0 0 1 0 1 1 0 0 1
0 0 1 1 1 1 0 1 1 0 0 0 0 1 0 0 1 0 1 1 0 0 1 1 0 1 1 0 1 0 0 1 1 0 0 0 1
1 1 0 0 1 1 1 0 0 0 1 1]

y_test
[0 1 1 0 0 0 0 0 1 0 0 1 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 1 1 1 0 1 1 0 1 0
1 1 0]
```

Task 3

In [7]:
```
#Task 3: Display confusion matrix and generate report of f1-score, recall and precision.
#predict the label on the base of given input variable

#make a model which will take the data x_train and t_train
classification_model = LogisticRegression()
classification_model.fit(x_train, y_train)

# Predict test set
y_pred = classification_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:
[[26  0]
 [ 0 14]]
Precision: 1.0
Recall: 1.0
F1 Score: 1.0
```


Color
Eye_color
Height
Legs
Moustache
Tail
Weight
label
human
Cat


0
0 2 4 6 8
Color
−2.5 0.0 2.5 5.0 7.5
Eye_color
0 2 4 6 8
Height
2.0 2.5 3.0 3.5 4.0
Legs
−0.5 0.0 0.5 1.0 1.5
Moustache
0.00 0.25 0.50 0.75 1.00
Tail
0 25 50 75
Weight

In [ ]: