In [38]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from scipy import optimize
import numpy as np
```

In [2]:
```
dataset = pd.read_csv('Data/dataset.csv')
```

In [3]:
```
dataset.describe()
```

Out[3]:

	Fraction Formatting Tags	Num Formatting Tags	Fraction Words	Num Words	Fraction p Children	Label
count	1211.000000	2119.000000	1211.000000	2119.000000	2119.000000	2119.000000
mean	1.110515	0.036350	0.867396	0.027007	0.004935	0.005663
std	2.314751	0.148286	1.687923	0.108564	0.056375	0.075058
min	0.000000	0.000000	0.030695	0.000132	0.000000	0.000000
25%	0.000000	0.000000	0.276255	0.000432	0.000000	0.000000
50%	0.000000	0.000000	0.551051	0.001061	0.000000	0.000000
75%	1.327158	0.000000	0.828766	0.005684	0.000000	0.000000
max	13.244898	1.000000	40.563294	0.999073	1.000000	1.000000

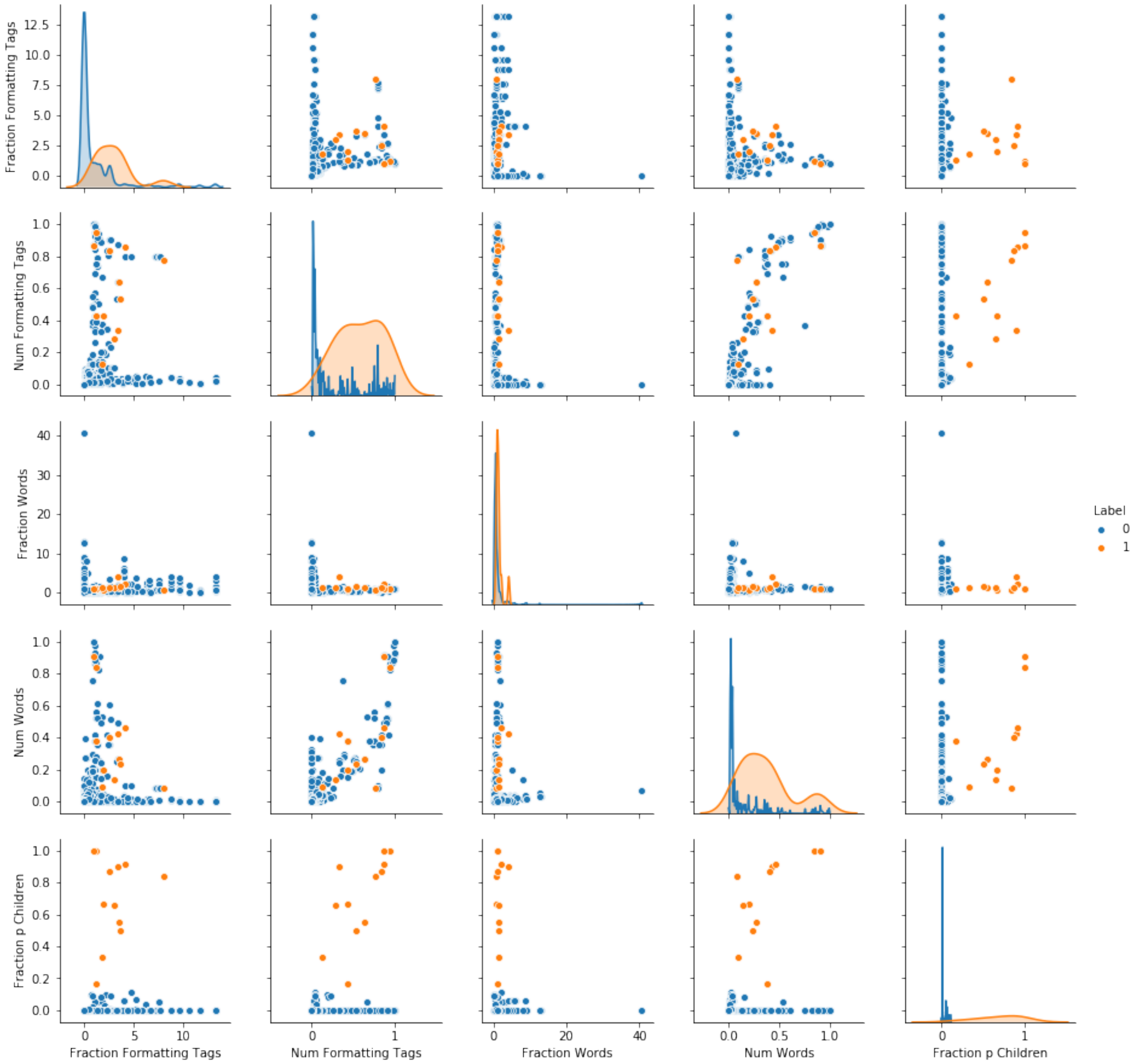
In [5]:
```
dataset['Label'].value_counts()
```

Out[5]:
```
0    2107
1      12
Name: Label, dtype: int64
```

This is a highly imbalanced dataset!

In [8]:
```
sns.pairplot(dataset, vars = ['Fraction Formatting Tags', 'Num Formatting Tags', 'Fraction Words', 'Num Words', 'Fraction p Children']
, hue = 'Label')
```

Out[8]: <seaborn.axisgrid.PairGrid at 0x1a18cf8ed0>



From the above pairplots, it is quite certain that **Fraction p Children** is a quite huge factor in determining the necessity of a tag. The other features don't seem to be necessary in making a decision.

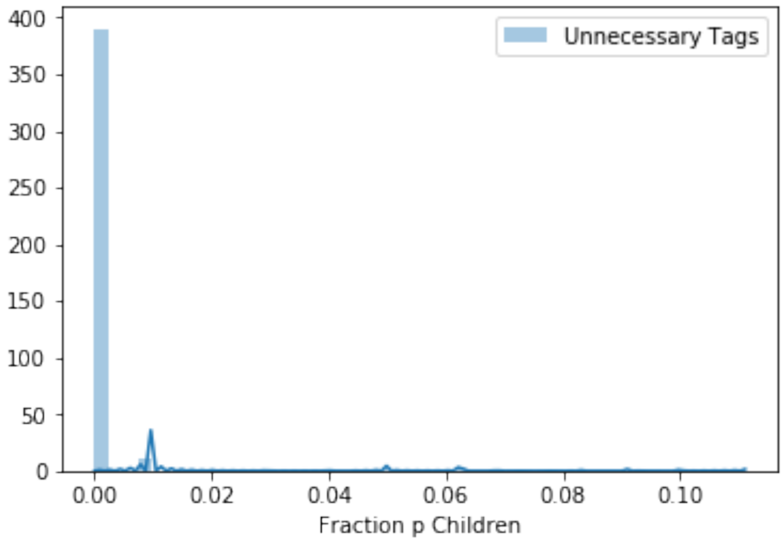
Let's look if there is a clear demarcation between the unnecessary and necessary tags in terms of the feature

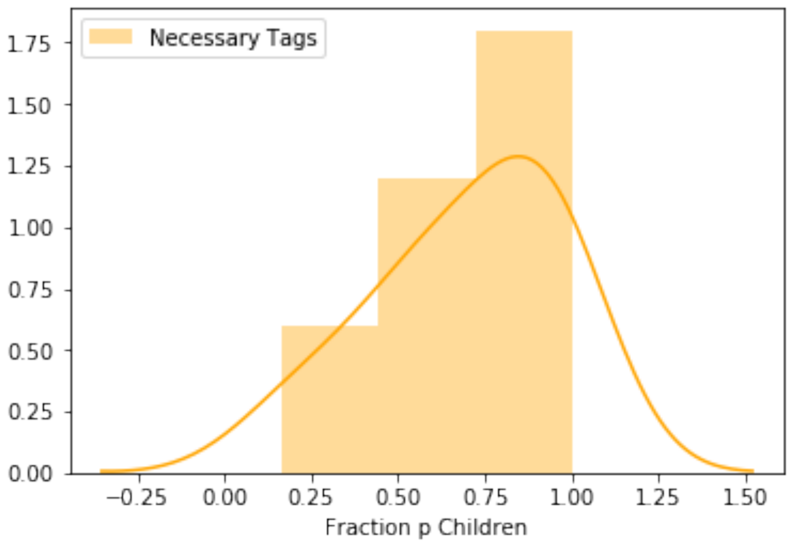
In [18]:
```
fpc_unnecessary = dataset.loc[dataset['Label'] == 0, 'Fraction p Ch
ildren']
fpc_necessary = dataset.loc[dataset['Label'] == 1, 'Fraction p Chil
dren']

print('Max value for fraction in case of unnecessary tags: {}'.form
at(max(fpc_unnecessary)))
print('Min value for fraction in case of necessary tags: {}'.format
(min(fpc_necessary)))

sns.distplot(fpc_unnecessary, label = 'Unnecessary Tags')
plt.legend()
plt.show()
sns.distplot(fpc_necessary, color = 'orange',label = 'Necessary Tag
s')
plt.legend()
plt.show()
```

```
Max value for fraction in case of unnecessary tags: 0.111111111
Min value for fraction in case of necessary tags: 0.166666667
```





This shows a clear demarcation! Let's however analyse the unnecessary tags with relatively high value of **Fraction p Children**.

In [13]:
```
high_fpc_unnecessary = dataset.loc[(dataset['Label'] == 0) & (dataset['Fraction p Children'] > 0.08)]
```

In [19]:
```
high_fpc_unnecessary
```

Out[19]:

	Fraction Formatting Tags	Num Formatting Tags	Fraction Words	Num Words	name	attrs	Fraction p Children	Label
364	4.784483	0.034483	2.179480	0.015708	div	{'class': ['success-messaging', 'ui-hidden']}	0.111111	0
398	0.656126	0.036364	0.486584	0.026967	div	{'id': 'document-footer-content'}	0.100000	0
441	2.219251	0.181818	0.307575	0.025199	div	{'class': ['byline-social']}	0.100000	0
1881	1.742750	0.204082	1.248927	0.146253	section	{'id': 'ref60691', 'data-level': '1'}	0.083333	0
1968	0.782609	0.042553	0.497283	0.027039	div	{'id': 'document-footer-content'}	0.090909	0
2014	2.675676	0.234043	0.304054	0.026596	div	{'class': ['byline-social']}	0.090909	0

In [20]:
```
low_fpc_necessary = dataset.loc[(dataset['Label'] == 1) & (dataset['Fraction p Children'] < 0.2)]
```

In [21]:
```
low_fpc_necessary
```

Out[21]:

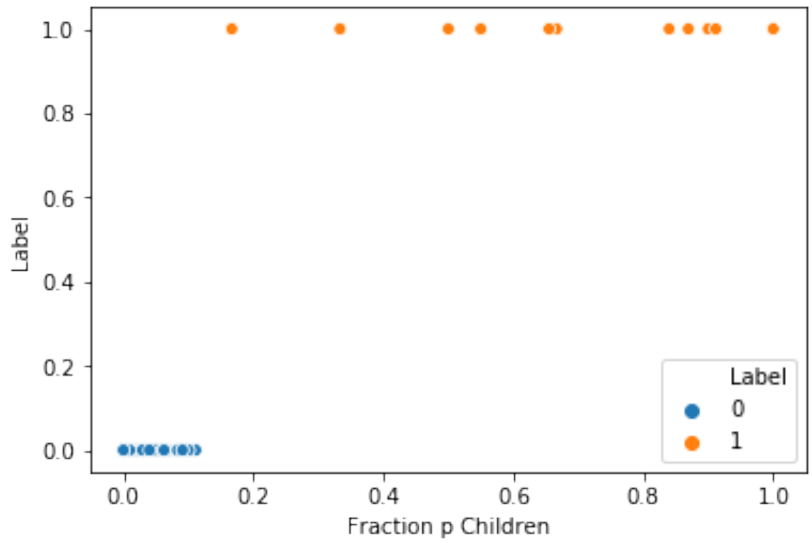
	Fraction Formatting Tags	Num Formatting Tags	Fraction Words	Num Words	name	attrs	Fraction p Children	Label
1056	1.251458	0.430233	1.105749	0.38014	td	{}	0.166667	1

It seems that **Num Formatting Tags** and **Num Words** add appreciable distinction in the border cases of **Fraction p Children**. It can be used in the future, if there is a necessity.

Let's now fit a Logistic Regression model to the data and see the outcome. However, we will verify if the assumption - "**The log odds of success varies linearly with the independent variables**". The above assumption states that the probability of getting on of the class will linearly increase with the independent variables.

In [23]:
```
sns.scatterplot(x = 'Fraction p Children', y = 'Label', hue = 'Label', data = dataset)
```

Out[23]: <matplotlib.axes._subplots.AxesSubplot at 0x1a1bc0c7d0>



This does look like a good dataset for Logistic Regression. Let's go forward with the model fitting.

In [40]:
```
def sigmoid(X, a, b):
    return 1/(1 + np.exp(-a * (X - b)))
```

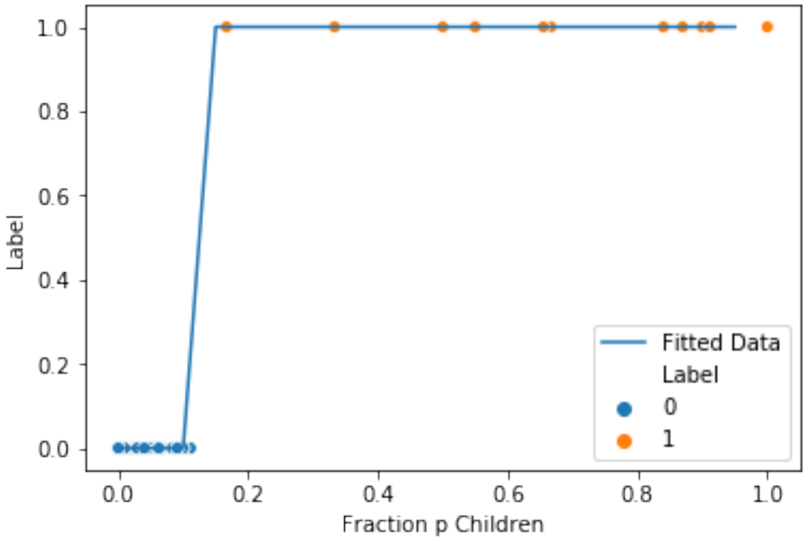
In [46]:
```
X = dataset['Fraction p Children'].values
y = dataset['Label'].values

params_optimal, _ = optimize.curve_fit(sigmoid, xdata = X, ydata =
y, p0 = (1, 0.1))

pseudo_X = np.arange(0, 1, 0.05)
y_proba = sigmoid(pseudo_X, *params_optimal)

sns.scatterplot(x = 'Fraction p Children', y = 'Label', hue = 'Labe
l', data = dataset)
plt.plot(pseudo_X, y_proba, label = 'Fitted Data')
plt.legend()
```

Out[46]: <matplotlib.legend.Legend at 0x1a1bab8450>



In [53]:
```
sigmoid(np.arange(0.15, 0.17, 0.05), *params_optimal)
```

Out[53]: array([0.99990459])