HR Analytic Using Logistic Regression

✏Contents of notebook :-

Importing Libraries

In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
```

Importing Dataset

In [2]:
```
hr = pd.read_csv('HR_comma_sep.csv')
```

Performing Some EDA

In [3]:
```
hr.head()
```

Out[3]:

	satisfaction_level	last_evaluation	number_project	average_montly_hours	time_spend_company	Work_accident	left	promotion_last_5years
0	0.38	0.53	2	157	3	0	1	0
1	0.80	0.86	5	262	6	0	1	0
2	0.11	0.88	7	272	4	0	1	0
3	0.72	0.87	5	223	5	0	1	0
4	0.37	0.52	2	159	3	0	1	0

In [4]:
```
hr.size
```

Out[4]:
```
149990
```

In [5]:
```
hr.describe()
```

Out[5]:

	satisfaction_level	last_evaluation	number_project	average_montly_hours	time_spend_company	Work_accident	left	promotion_
count	14999.000000	14999.000000	14999.000000	14999.000000	14999.000000	14999.000000	14999.000000	1
mean	0.612834	0.716102	3.803054	201.050337	3.498233	0.144610	0.238083	
std	0.248631	0.171169	1.232592	49.943099	1.460136	0.351719	0.425924	
min	0.090000	0.360000	2.000000	96.000000	2.000000	0.000000	0.000000	
25%	0.440000	0.560000	3.000000	156.000000	3.000000	0.000000	0.000000	
50%	0.640000	0.720000	4.000000	200.000000	3.000000	0.000000	0.000000	
75%	0.820000	0.870000	5.000000	245.000000	4.000000	0.000000	0.000000	
max	1.000000	1.000000	7.000000	310.000000	10.000000	1.000000	1.000000	

In [6]:
```
hr.describe(include = 'object')
```

Out[6]:

	Department	salary
count	14999	14999
unique	10	3
top	sales	low
freq	4140	7316

In [7]:
```
hr.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14999 entries, 0 to 14998
Data columns (total 10 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   satisfaction_level     14999 non-null  float64
 1   last_evaluation        14999 non-null  float64
 2   number_project         14999 non-null  int64  
 3   average_montly_hours   14999 non-null  int64  
 4   time_spend_company     14999 non-null  int64  
 5   Work_accident          14999 non-null  int64  
 6   left                   14999 non-null  int64  
 7   promotion_last_5years  14999 non-null  int64  
 8   Department             14999 non-null  object 
 9   salary                 14999 non-null  object 
dtypes: float64(2), int64(6), object(2)
memory usage: 1.1+ MB
```

There is no null values in this dataset so We don't have to perform any data cleaning for null values

In [8]:
```
hr['Department'].unique()
```

Out[8]:
```
array(['sales', 'accounting', 'hr', 'technical', 'support', 'management',
       'IT', 'product_mng', 'marketing', 'RandD'], dtype=object)
```

In [9]:
```
hr['salary'].unique()
```

Out[9]:
```
array(['low', 'medium', 'high'], dtype=object)
```

Department

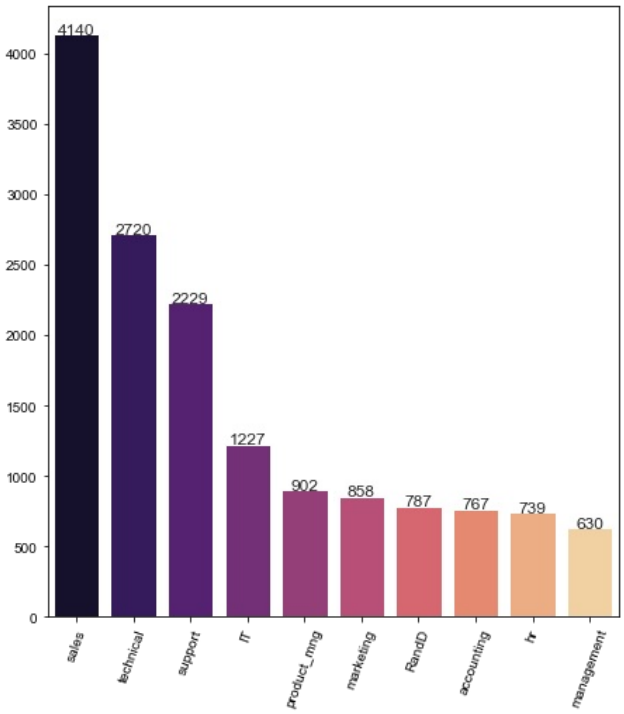
In [10]:
```
hr['Department'].value_counts()/len(hr)*100
```

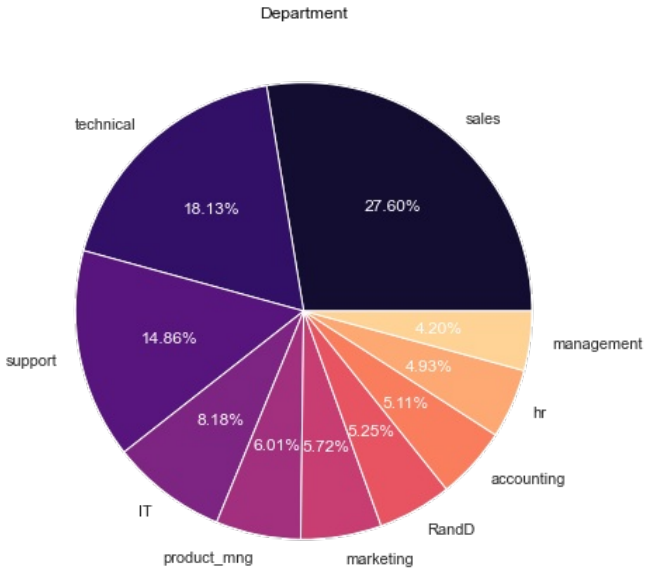
Out[10]:
```
sales          27.601840
technical      18.134542
support        14.860991
IT              8.180545
product_mng     6.013734
marketing       5.720381
RandD           5.247016
accounting      5.113674
hr              4.926995
management      4.200280
Name: Department, dtype: float64
```

Data Visualtization

In [11]:
```
fig, ax = plt.subplots(1,2, figsize = (16,8))
sns.set(style = 'dark' , color_codes = True)
data = hr['Department'].value_counts()
```

```
pal = sns.color_palette("magma" , len(data))

ax[0] = sns.barplot( x = data.index, y = data.values , ax = ax[0] , palette = pal)
for bar in ax[0].patches:
    ax[0].annotate( "{:.0f}".format(bar.get_height()) , ( bar.get_x() + bar.get_width()/2 , bar.get_height()) , h

ax[0].set_xticklabels( ax[0].get_xticklabels() ,  rotation = 70)

_,_, autotexts = ax[1].pie( data.values, labels = data.index , autopct = "%.2f%%" , colors = pal)

for text in autotexts:
    text.set_color('white')

plt.title("Department")



plt.show()
```





In [12]:

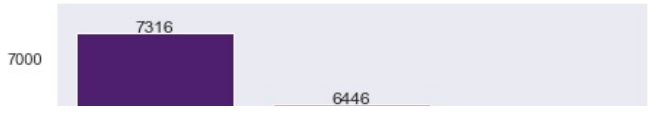
```
hr['salary'].value_counts()/len(hr)*100
```

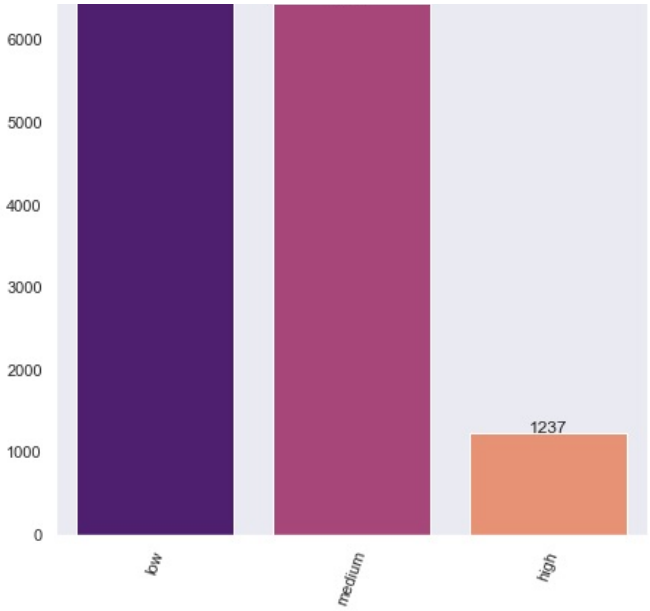
Out[12]:

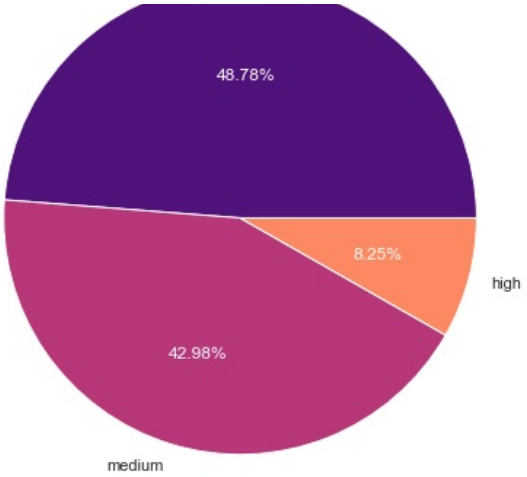
```
low       48.776585
medium    42.976198
high       8.247216
Name: salary, dtype: float64
```

In [13]:

```
fig, ax = plt.subplots(1,2, figsize = (16,8))
sns.set(style = 'dark' , color_codes = True)
data = hr['salary'].value_counts()
pal = sns.color_palette("magma" , len(data))

ax[0] = sns.barplot( x = data.index, y = data.values , ax = ax[0] , palette = pal)
for bar in ax[0].patches:
    ax[0].annotate( "{:.0f}".format(bar.get_height()) , ( bar.get_x() + bar.get_width()/2 , bar.get_height()) , h

ax[0].set_xticklabels( ax[0].get_xticklabels() ,  rotation = 70)

_,_, autotexts = ax[1].pie( data.values, labels = data.index , autopct = "%.2f%%" , colors = pal)

for text in autotexts:
    text.set_color('white')

plt.title("Salary")



plt.show()
```





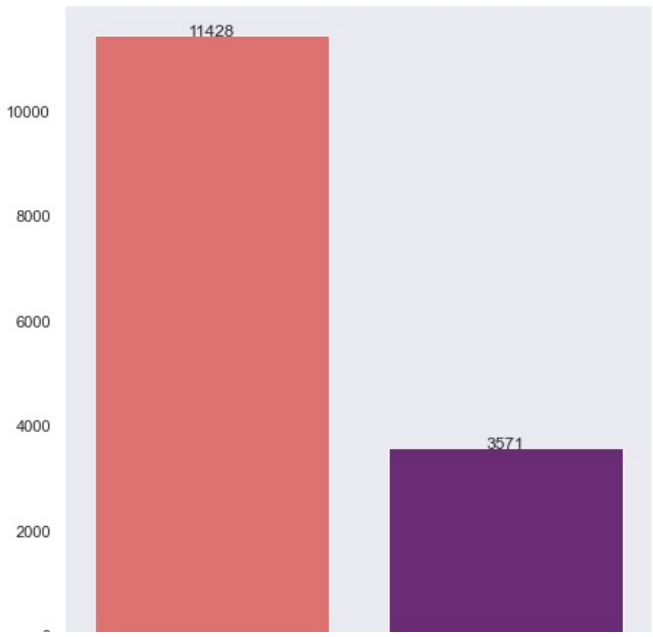


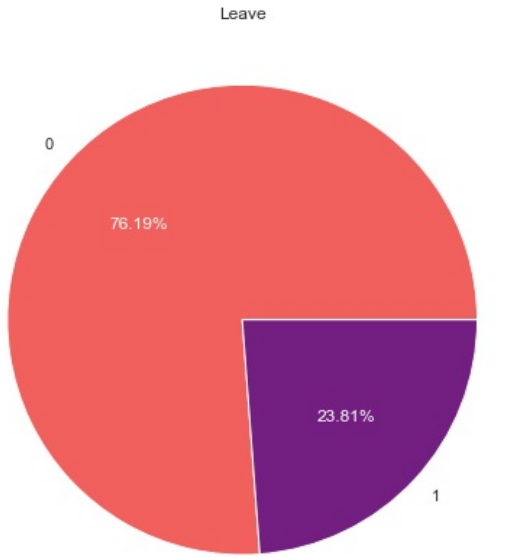
In [14]:
```
hr.left.value_counts()/len(hr)*100
```

Out[14]:
```
0    76.191746
1    23.808254
Name: left, dtype: float64
```

In [15]:
```
hr['left'] = hr.left.astype('object')
```

In [16]:
```
fig, ax = plt.subplots(1,2, figsize = (16,8))
sns.set(style = 'dark' , color_codes = True)
data = hr['left'].value_counts()
pal = sns.color_palette("magma" , len(data))

ax[0] = sns.barplot( x = data.index, y = data.values , ax = ax[0] , palette = pal[::-1])
for bar in ax[0].patches:
    ax[0].annotate( "{:.0f}".format(bar.get_height()) , ( bar.get_x() + bar.get_width()/2 , bar.get_height()) , h

ax[0].set_xticklabels( ax[0].get_xticklabels() ,  rotation = 70)

_,_, autotexts = ax[1].pie( data.values, labels = data.index , autopct = "%.2f%%" , colors = pal[::-1])

for text in autotexts:
    text.set_color('white')

plt.title("Leave")



plt.show()
```





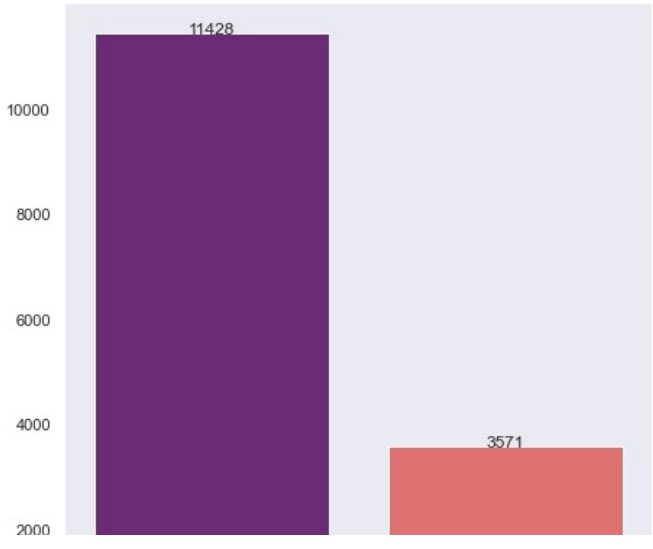
In [17]:

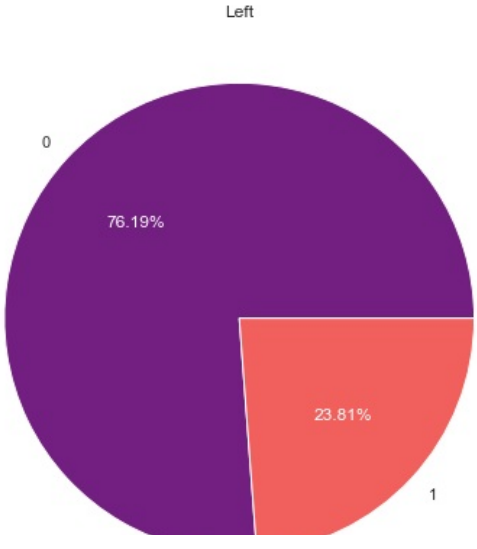
```
ct = pd.crosstab(hr['Department'] , hr['salary'])

ct.div(ct.sum(1).astype(float) , axis = 0).plot(kind = 'bar' , stacked = True)

plt.title('Salary vs Department')

plt.xlabel('Department')

plt.ylabel('Salary')

plt.show()
```

Now find the independent features

In [18]:

```
fig, ax = plt.subplots(1,2, figsize = (16,8))
sns.set(style = 'dark' , color_codes = True)
data = hr['left'].value_counts()
pal = sns.color_palette("magma" , len(data))

ax[0] = sns.barplot( x = data.index, y = data.values , ax = ax[0] , palette = pal)
for bar in ax[0].patches:
    ax[0].annotate( "{:.0f}".format(bar.get_height()) , ( bar.get_x() + bar.get_width()/2 , bar.get_height()) , h

ax[0].set_xticklabels( ax[0].get_xticklabels() ,  rotation = 70)

_,_, autotexts = ax[1].pie( data.values, labels = data.index , autopct = "%.2f%%" , colors = pal)

for text in autotexts:
    text.set_color('white')

plt.title("Left")


plt.show()
```

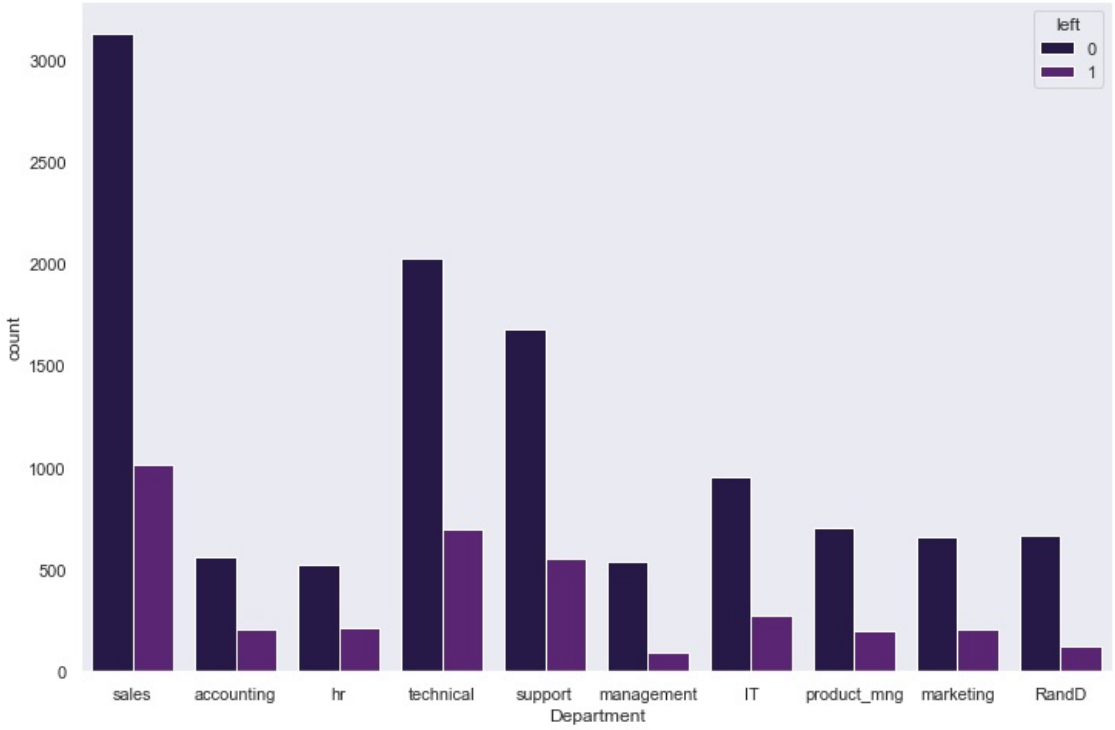




We can see that almost 24% employees leave the company

In [19]:
```
fig = plt.figure( figsize = (12,8))
sns.set(style = 'dark' , color_codes = True)
sns.countplot('Department' , data = hr , palette = sns.color_palette('magma'), hue = 'left')
```

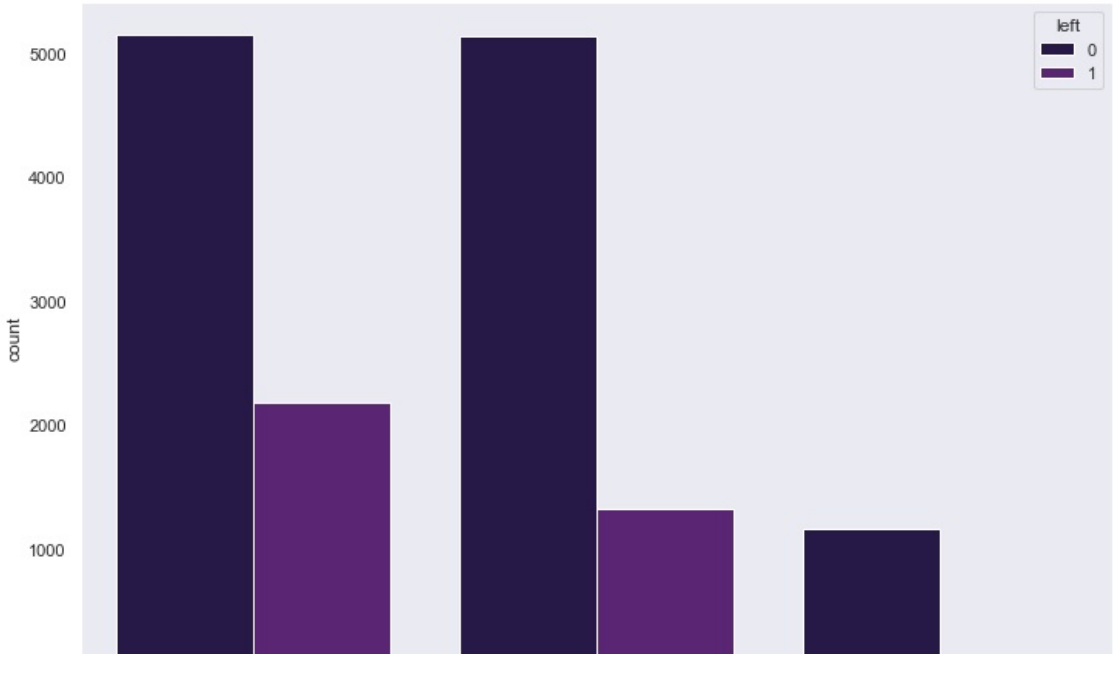
Out[19]:
```
<AxesSubplot:xlabel='Department', ylabel='count'>
```

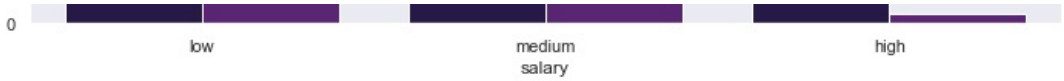


We can see here that there is no such major impact of department on retention of any employee

In [20]:
```
fig = plt.figure( figsize = (12,8))
sns.set(style = 'dark' , color_codes = True)
sns.countplot('salary' , data = hr , palette = sns.color_palette('magma'), hue = 'left')
```

Out[20]:
```
<AxesSubplot:xlabel='salary', ylabel='count'>
```

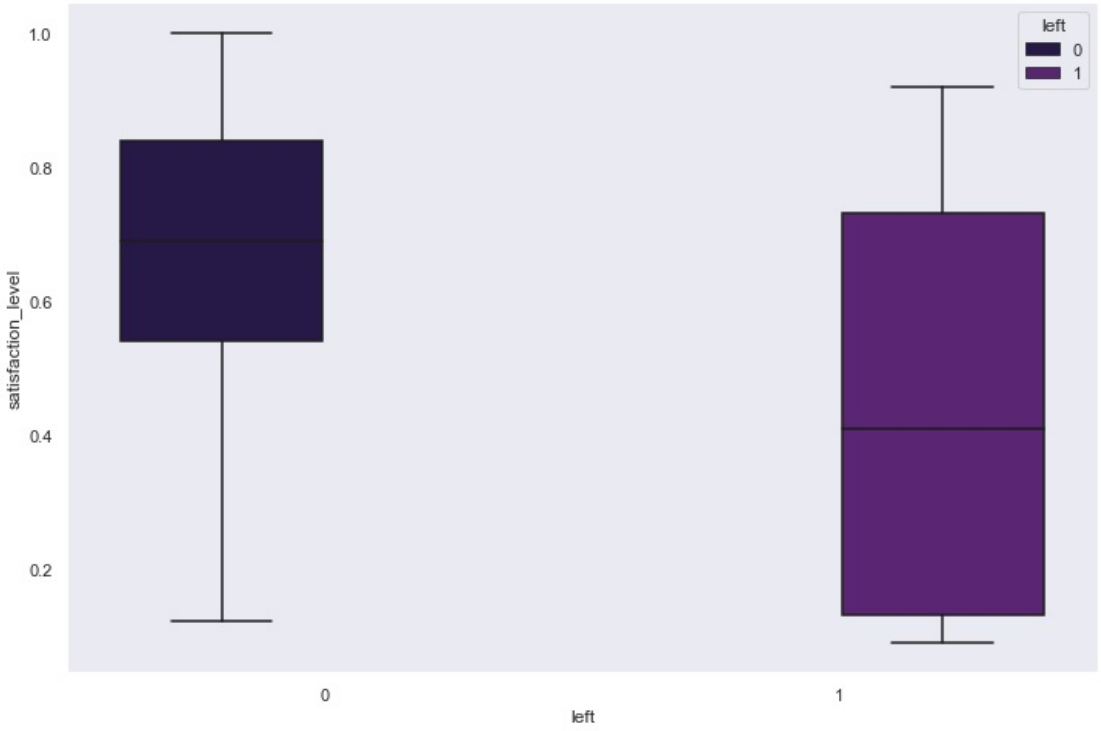




We can clearly see here that employees with higher salaries are not like to leave the company

In [21]:
```
fig = plt.figure( figsize = (12,8))
sns.set(style = 'dark' , color_codes = True)
sns.boxplot( x = 'left' , y = 'satisfaction_level' , data = hr , palette = sns.color_palette('magma'), hue = 'le
```

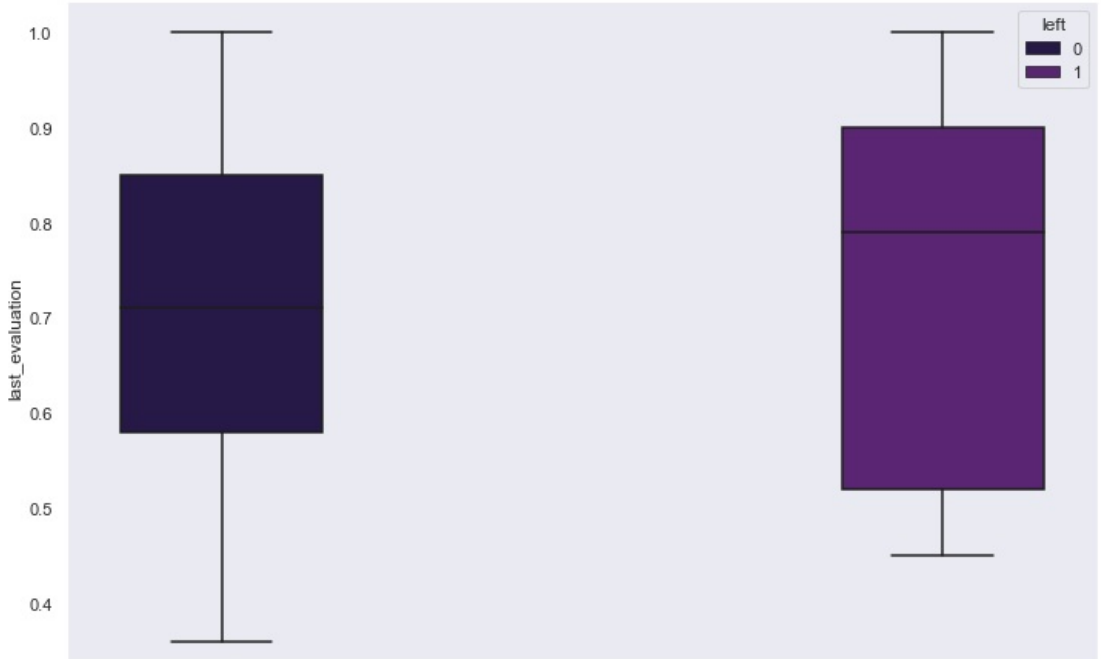
Out[21]:
```
<AxesSubplot:xlabel='left', ylabel='satisfaction_level'>
```



We can see here that satisfaction level is directly impact the leaving chances of the employee

In [22]:
```
fig = plt.figure( figsize = (12,8))
sns.set(style = 'dark' , color_codes = True)
sns.boxplot( x = 'left' , y = 'last_evaluation' , data = hr , palette = sns.color_palette('magma'), hue = 'left')
```

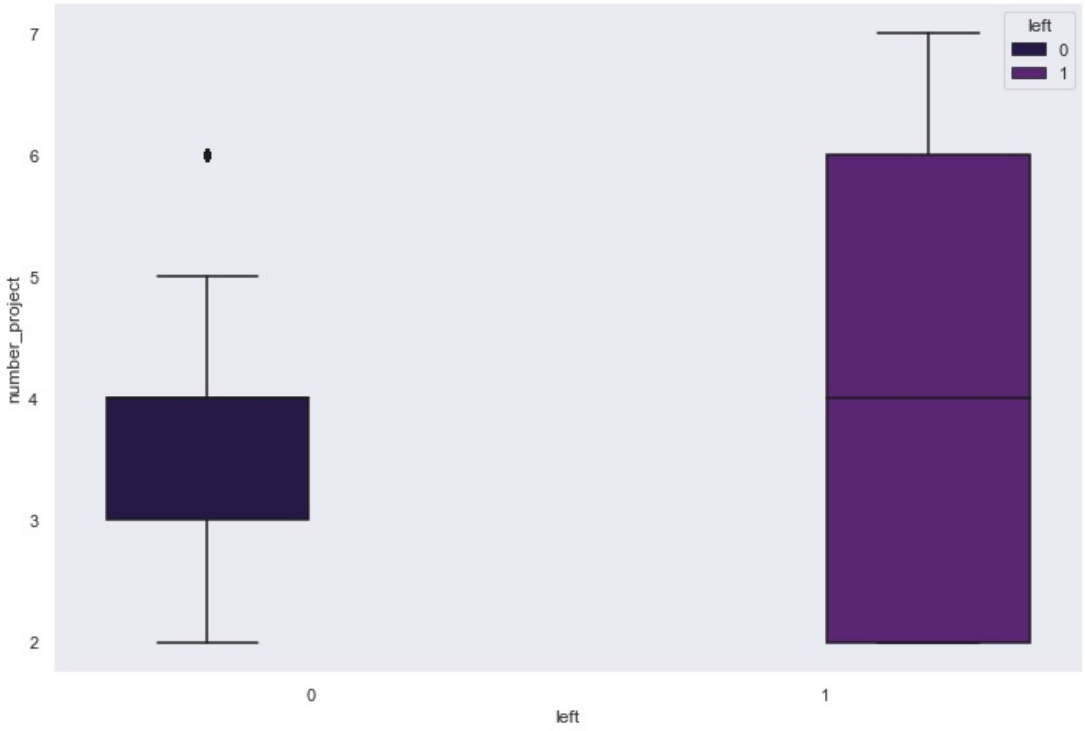
Out[22]:
```
<AxesSubplot:xlabel='left', ylabel='last_evaluation'>
```



From above chart there seem to be no impact of last_evalution on employee retention

In [23]:
```
fig = plt.figure( figsize = (12,8))
sns.set(style = 'dark' , color_codes = True)
sns.boxplot( x = 'left' , y = 'number_project' , data = hr , palette = sns.color_palette('magma'), hue = 'left')
```

Out[23]:
```
<AxesSubplot:xlabel='left', ylabel='number_project'>
```



From above chart there seem to be no impact of number_project on employee retention

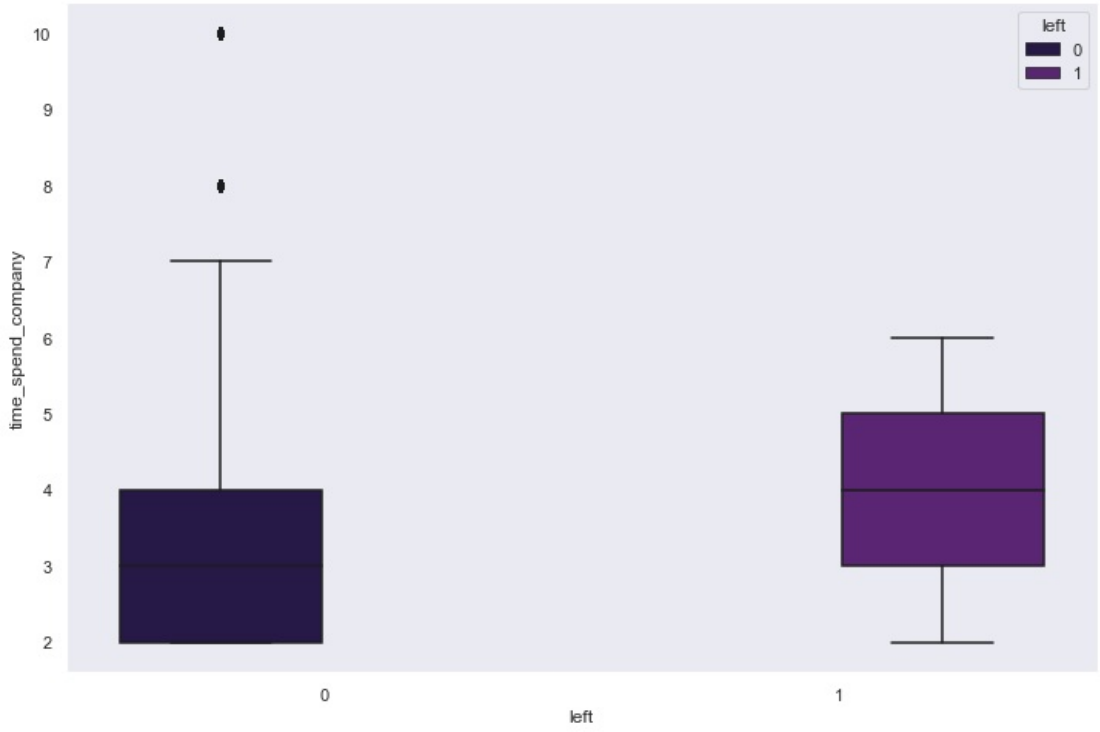
In [24]:
```
fig = plt.figure( figsize = (12,8))
sns.set(style = 'dark' , color_codes = True)
sns.boxplot( x = 'left' , y = 'average_montly_hours' , data = hr , palette = sns.color_palette('magma'), hue = 'l
```

Out[24]:
```
<AxesSubplot:xlabel='left', ylabel='average_montly_hours'>
```

From above chart there seem to be some impact of average_montly_hours on employee retention but it is not too major but we will consider it in our analysis

In [25]:
```
fig = plt.figure( figsize = (12,8))
sns.set(style = 'dark' , color_codes = True)
sns.boxplot( x = 'left' , y = 'time_spend_company' , data = hr , palette = sns.color_palette('magma'), hue = 'lef
```

Out[25]:
```
<AxesSubplot:xlabel='left', ylabel='time_spend_company'>
```



Above bar chart shows employees with low time_spend_compmay are likely to not leave the company

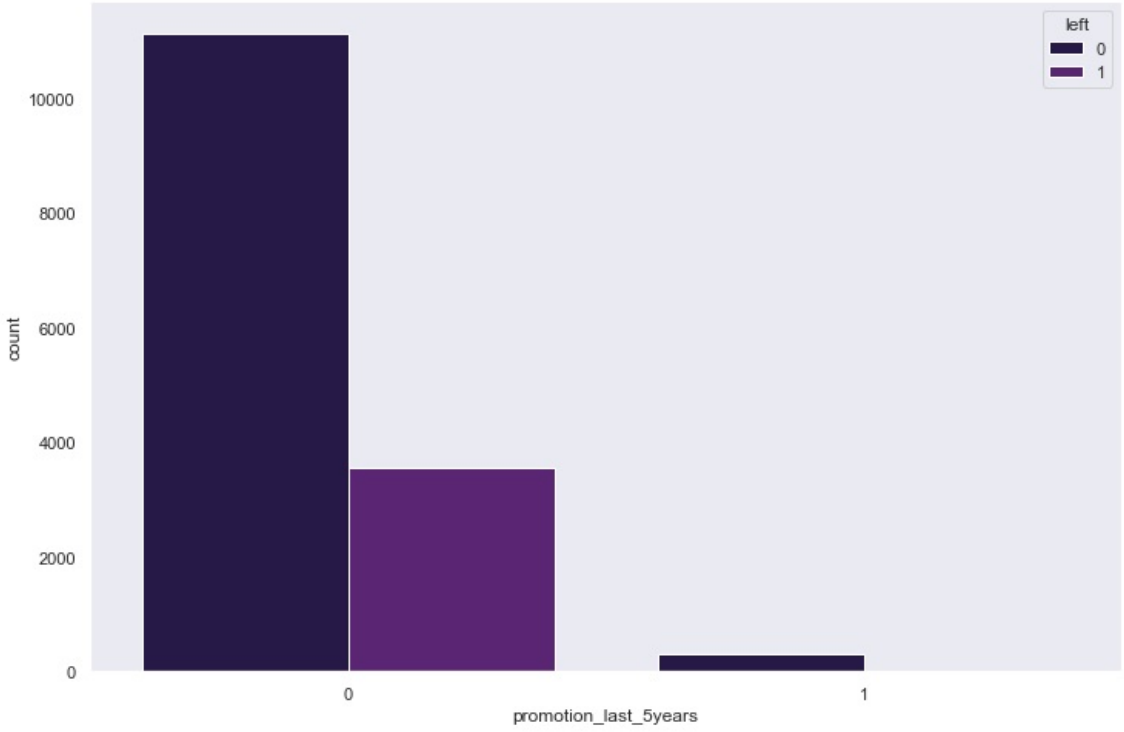
In [26]:
```
fig = plt.figure( figsize = (12,8))
sns.set(style = 'dark' , color_codes = True)
sns.countplot('Work_accident' , data = hr , palette = sns.color_palette('magma'), hue = 'left')
```

Out[26]:
```
<AxesSubplot:xlabel='Work_accident', ylabel='count'>
```

From above chart there seem to be impact of Work_accident on employee retention

In [27]:
```
fig = plt.figure( figsize = (12,8))
sns.set(style = 'dark' , color_codes = True)
sns.countplot('promotion_last_5years' , data = hr , palette = sns.color_palette('magma'), hue = 'left')
```

Out[27]:
```
<AxesSubplot:xlabel='promotion_last_5years', ylabel='count'>
```



From the data analysis so far we can conclude that we will use following variables as independant variables in our model

1. **Satisfaction Level**
2. **Average Monthly Hours**
3. **Promotion Last 5 Years**
4. **Salary**
5. **Work Accident**

Data Preprocessing

In [28]:
```
subdf = hr[['satisfaction_level','average_montly_hours','promotion_last_5years','Work_accident','salary']]
subdf.head()
```

Out[28]:

	satisfaction_level	average_montly_hours	promotion_last_5years	Work_accident	salary
0	0.38	157	0	0	low
1	0.80	262	0	0	medium
2	0.11	272	0	0	medium
3	0.72	223	0	0	low
4	0.37	159	0	0	low

In [29]:
```
salary_dummies = pd.get_dummies(subdf.salary, prefix="salary")
```

In [30]:
```
df_with_dummies = pd.concat([subdf,salary_dummies],axis='columns')
```

In [31]:
```
df_with_dummies.head()
```

Out[31]:

	satisfaction_level	average_montly_hours	promotion_last_5years	Work_accident	salary	salary_high	salary_low	salary_medium
0	0.38	157	0	0	low	0	1	0
1	0.80	262	0	0	medium	0	0	1
2	0.11	272	0	0	medium	0	0	1
3	0.72	223	0	0	low	0	1	0
4	0.37	159	0	0	low	0	1	0

In [32]:
```
df_with_dummies.drop(['salary','salary_low'],axis='columns',inplace=True)
df_with_dummies.head()
```

Out[32]:

	satisfaction_level	average_montly_hours	promotion_last_5years	Work_accident	salary_high	salary_medium
0	0.38	157	0	0	0	0
1	0.80	262	0	0	0	1
2	0.11	272	0	0	0	1
3	0.72	223	0	0	0	0
4	0.37	159	0	0	0	0

In [33]:
```
X = df_with_dummies
X.head()
```

Out[33]:

	satisfaction_level	average_montly_hours	promotion_last_5years	Work_accident	salary_high	salary_medium
0	0.38	157	0	0	0	0
1	0.80	262	0	0	0	1
2	0.11	272	0	0	0	1
3	0.72	223	0	0	0	0
4	0.37	159	0	0	0	0

In [34]:
```
y = hr['left'].astype(str)
y
```

Out[34]:
```
0        1
1        1
2        1
3        1
4        1
       ..
14994    1
14995    1
14996    1
14997    1
14998    1
Name: left, Length: 14999, dtype: object
```

In [35]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.3)
```

Model Building

In [36]:
```
from sklearn.linear_model import LogisticRegression
```

```
model = LogisticRegression()
```

In [37]:
```
model.fit(X_train, y_train)
```

Out[37]:
```
LogisticRegression()
```

In [38]:
```
ypred = model.predict(X_test)
```

In [39]:
```
model.score(X_test,y_test)
```

Out[39]:
```
0.7753333333333333
```

Exciting Milestone: Successfully trained my first logistic regression model, one more step in my journey into data science and predictive analytics. Looking forward to exploring more complex algorithms and applications!