ASSIGNMENT/ TASK 9

GO_STP_12574
SHWETA JHA

Predict retention of an employee within an organization such that whether the employee will leave the company or continue with it. An organization is only as good as its employees, and these people are the true source of its competitive advantage. Dataset is downloaded from Kaggle. Link: https://www.kaggle.com/giripujar/hr-analytics

First do data exploration and visualization, after this create a logistic regression model to predict Employee Attrition Using Machine Learning & Python.

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import sklearn
from sklearn.linear_model import LinearRegression
```

```
dataset=pd.read_csv('/content/HR_comma_sep.csv')
dataset.head()
```

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

```
dataset.shape
```

```
(14999, 10)
```

```
dataset.describe()
```

	satisfaction_level	last_evaluation	number_project	average_montly_hours	ti
count	14999.000000	14999.000000	14999.000000	14999.000000	
mean	0.612834	0.716102	3.803054	201.050337	
std	0.248631	0.171169	1.232592	49.943099	

```
dataset.columns
```

```
Index(['satisfaction_level', 'last_evaluation', 'number_project',
       'average_montly_hours', 'time_spend_company', 'Work_accident', 'left',
       'promotion_last_5years', 'Department', 'salary'],
      dtype='object')
```

max	1.000000	1.000000	7.000000	310.000000

```
dataset.dtypes
```

```
satisfaction_level       float64
last_evaluation          float64
number_project             int64
average_montly_hours       int64
time_spend_company         int64
Work_accident              int64
left                       int64
promotion_last_5years      int64
Department                object
salary                    object
dtype: object
```

```
dataset.corr()
```

	satisfaction_level	last_evaluation	number_project	average_m
satisfaction_level	1.000000	0.105021	-0.142970	
last_evaluation	0.105021	1.000000	0.349333	
number_project	-0.142970	0.349333	1.000000	
average_montly_hours	-0.020048	0.339742	0.417211	
time_spend_company	-0.100866	0.131591	0.196786	
Work_accident	0.058697	-0.007104	-0.004741	
left	-0.388375	0.006567	0.023787	
promotion_last_5years	0.025605	-0.008684	-0.006064	

```
dataset.info()
```

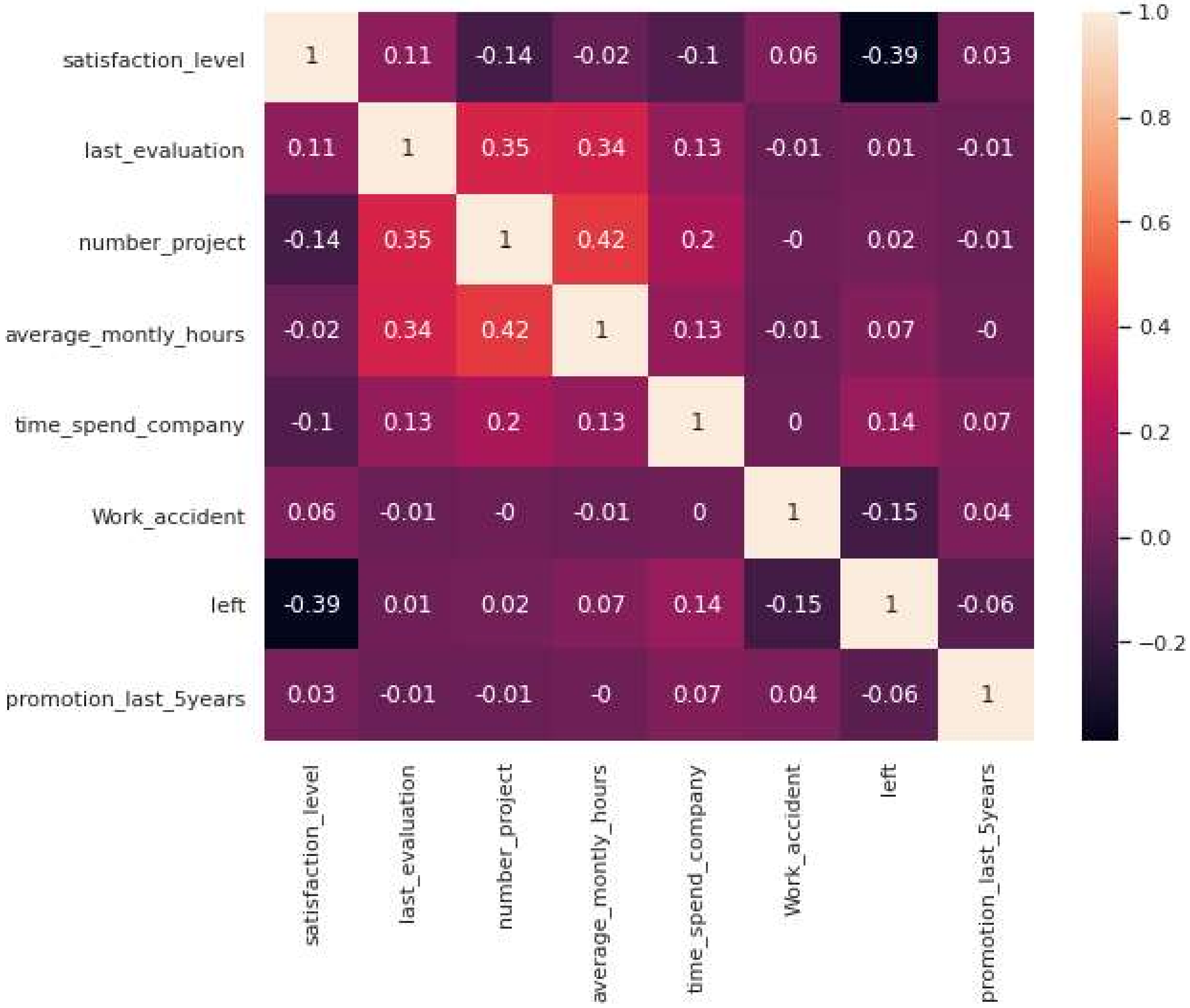
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14999 entries, 0 to 14998
Data columns (total 10 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   satisfaction_level     14999 non-null  float64
```

```
 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
```

```
import seaborn as sns
```

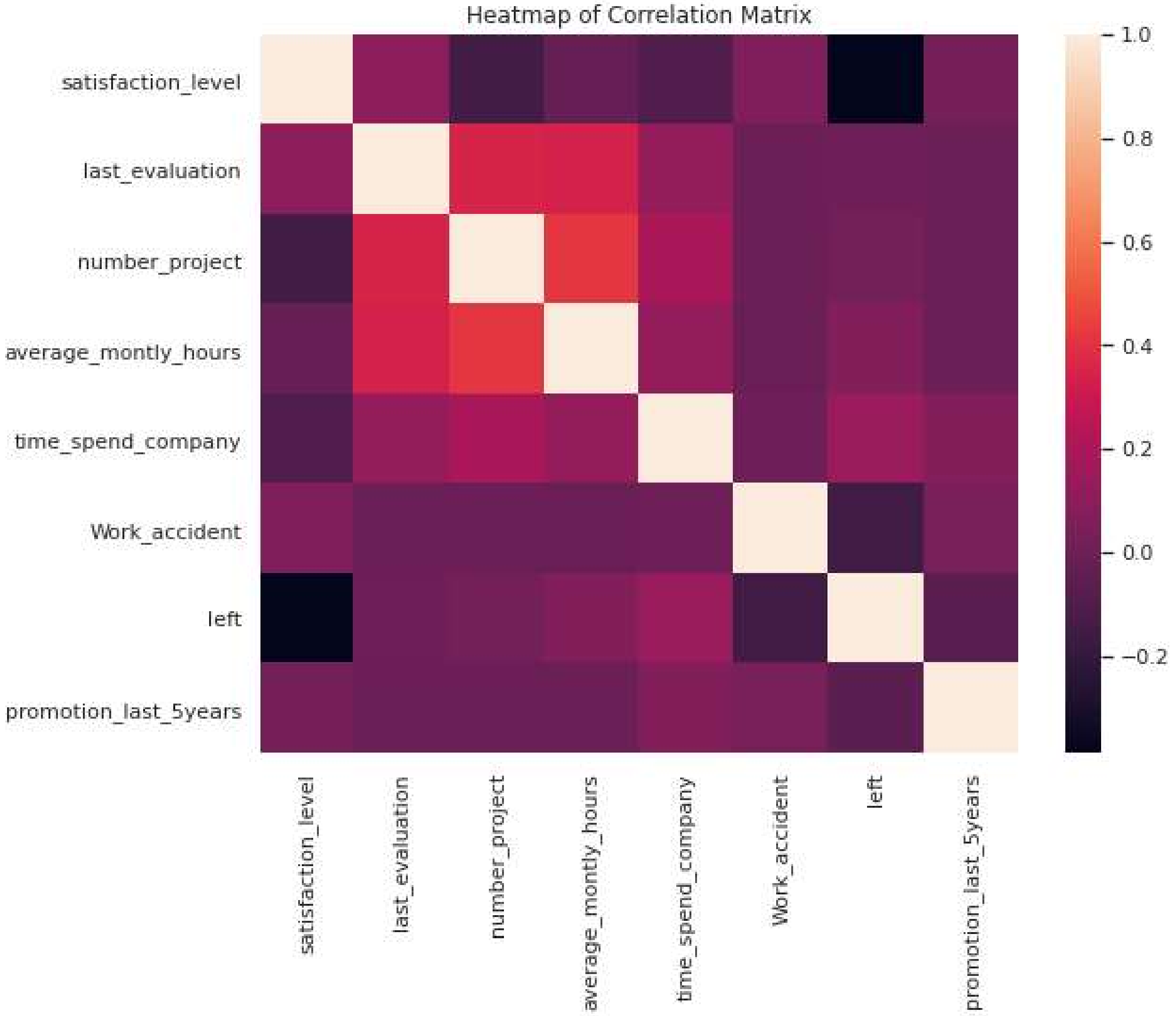
```
sns.set(rc={'figure.figsize':(9,7)})
correlation_matrix = dataset.corr().round(2)
sns.heatmap(data=correlation_matrix, annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5d820e86d0>
```




```
corr=dataset.corr()
corr =(corr)
sns.heatmap(corr, xticklabels=corr.columns.values,yticklabels=corr.columns.values)
plt.title('Heatmap of Correlation Matrix')
```

```
Text(0.5, 1.0, 'Heatmap of Correlation Matrix')
```




```
dataset.groupby('salary').mean()
```

salary	satisfaction_level	last_evaluation	number_project	average_montly_hours
high	0.637470	0.704325	3.767179	199.867421
low	0.600753	0.717017	3.799891	200.996583
medium	0.621817	0.717322	3.813528	201.338349

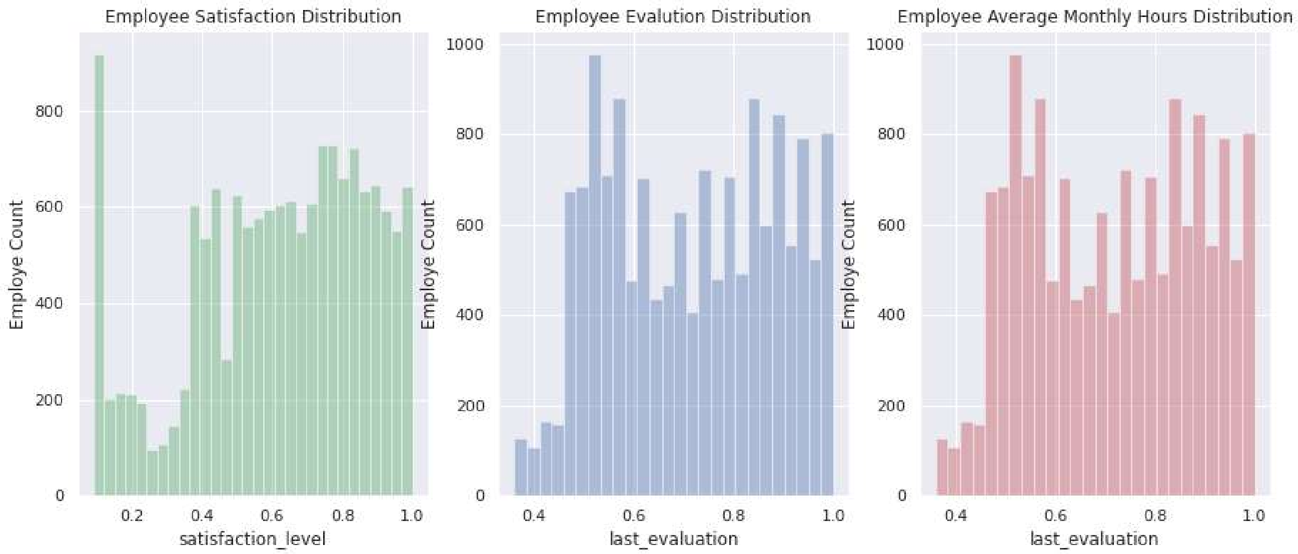
```
emp_population_satisfaction =dataset['satisfaction_level'].mean()
emp_turnover_satisfaction =dataset[dataset['left']==1]['satisfaction_level'].mean()
print('the mean for population is:'+str(emp_population_satisfaction))
print('the mean for the employe is:'+str(emp_turnover_satisfaction))
```

```
the mean for population is:0.6128335222348166
the mean for the employe is:0.44009801176140917
```

```
f, axes=plt.subplots(ncols=3,figsize=(15, 6))
sns.distplot(dataset.satisfaction_level,kde=False,color="g",ax=axes[0]).set_title('Employe
axes[0].set_ylabel('Employe Count')
sns.distplot(dataset.last_evaluation,kde=False,color="b",ax=axes[1]).set_title('Employee E
```

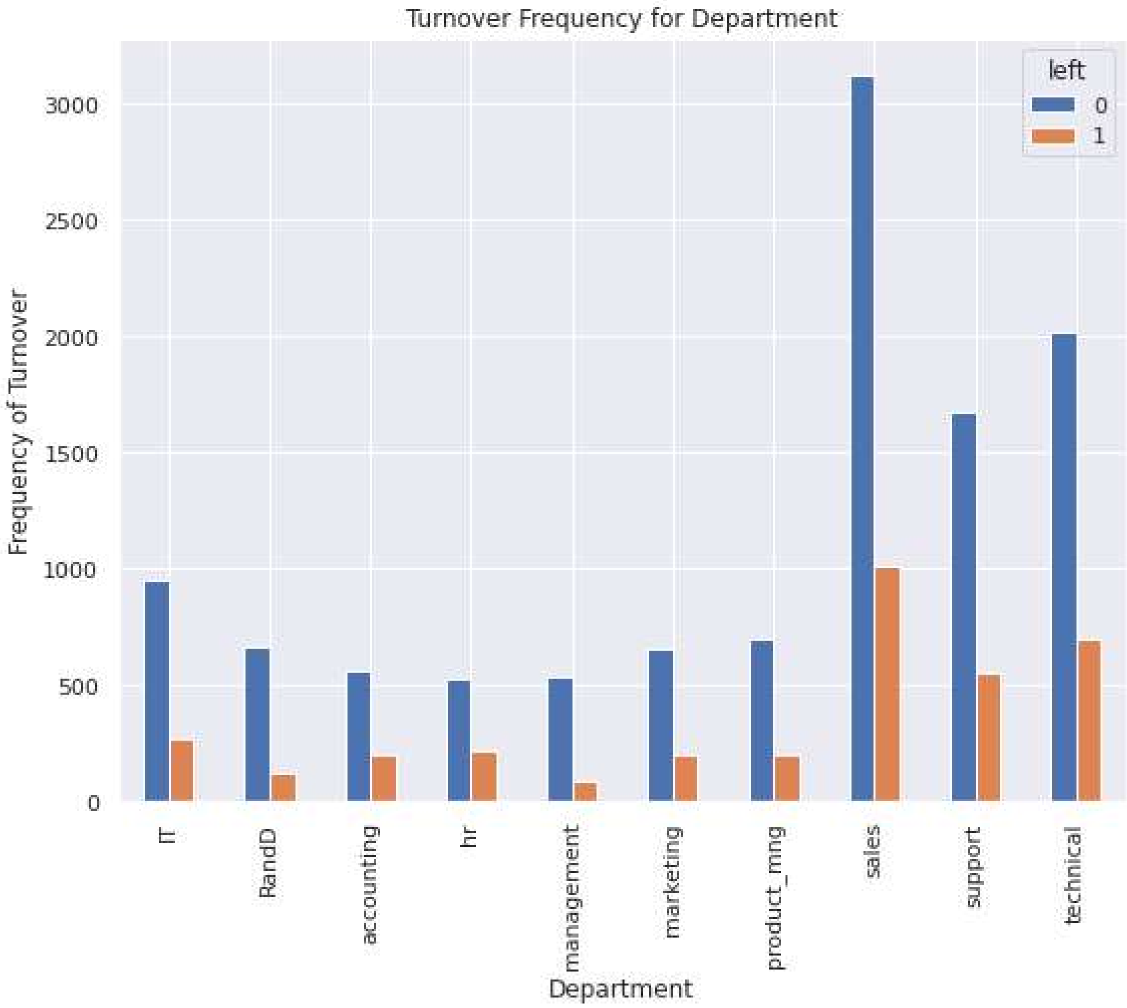
```
axes[1].set_ylabel('Employe Count')
sns.distplot(dataset.last_evaluation,kde=False,color="r",ax=axes[2]).set_title('Employee A
axes[2].set_ylabel('Employe Count')
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2557: FutureWarning:
  warnings.warn(msg, FutureWarning)
Text(0, 0.5, 'Employe Count')
```

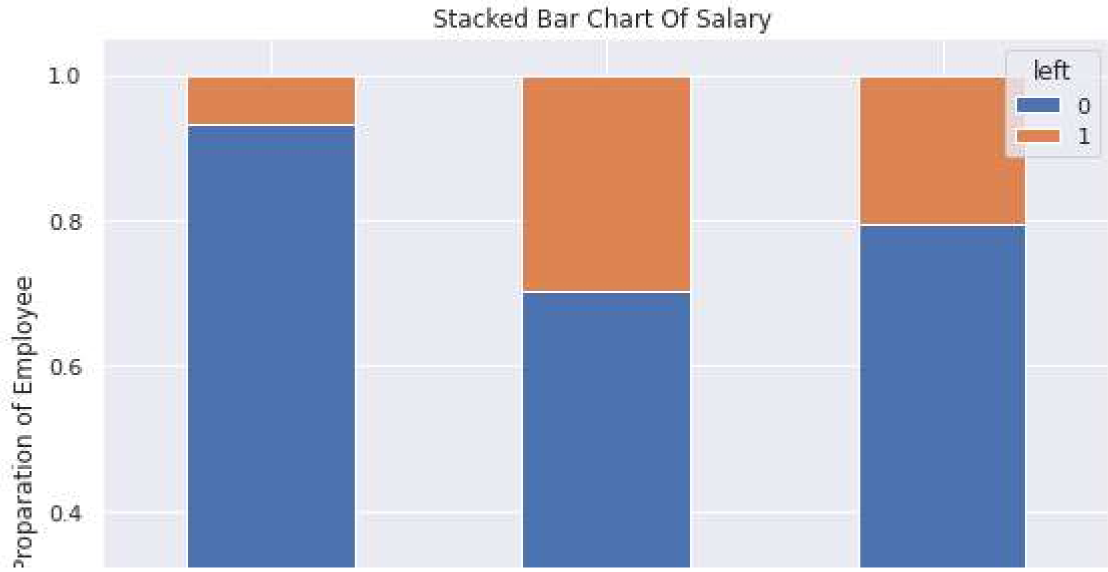



```
sns.countplot(x='Department',data=dataset).set_title("Employe Department");
```


```
pd.crosstab(dataset.Department,dataset.left).plot(kind='bar')
plt.title('Turnover Frequency for Department')
plt.xlabel('Department')
plt.ylabel('Frequency of Turnover')
plt.savefig('Department_bar_chart')
```

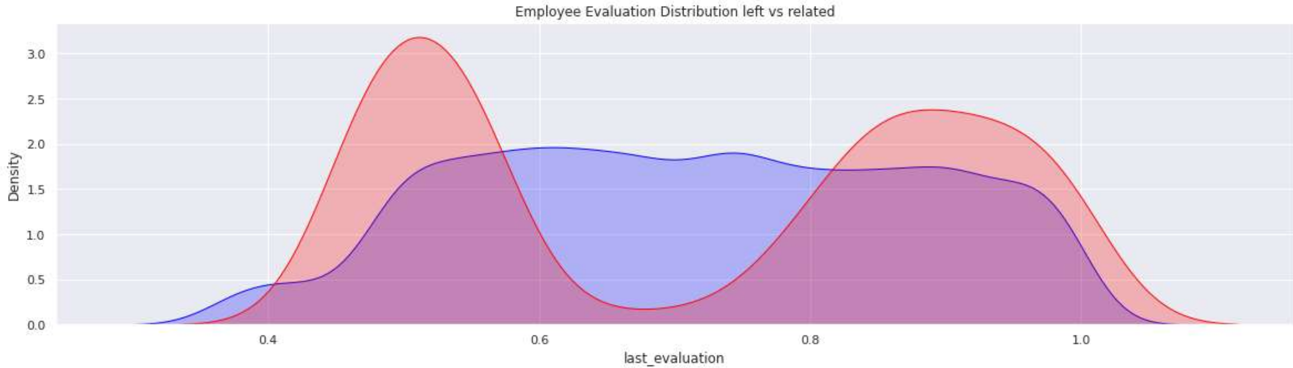



```
table=pd.crosstab(dataset.salary,dataset.left)
table.div(table.sum(1).astype(float),axis=0).plot(kind='bar',stacked=True)
plt.title("Stacked Bar Chart Of Salary ")
plt.xlabel('Salary Level')
plt.ylabel('Proparation of Employee')
plt.savefig('salary_bar_chart')
```




```
fig=plt.figure(figsize=(20,5))
ax=sns.kdeplot(dataset.loc[(dataset["left"]==0),'last_evaluation'],color='blue',shade=True
ax=sns.kdeplot(dataset.loc[(dataset["left"]==1),'last_evaluation'],color='red',shade=True)
plt.title('Employee Evaluation Distribution left vs related')
```

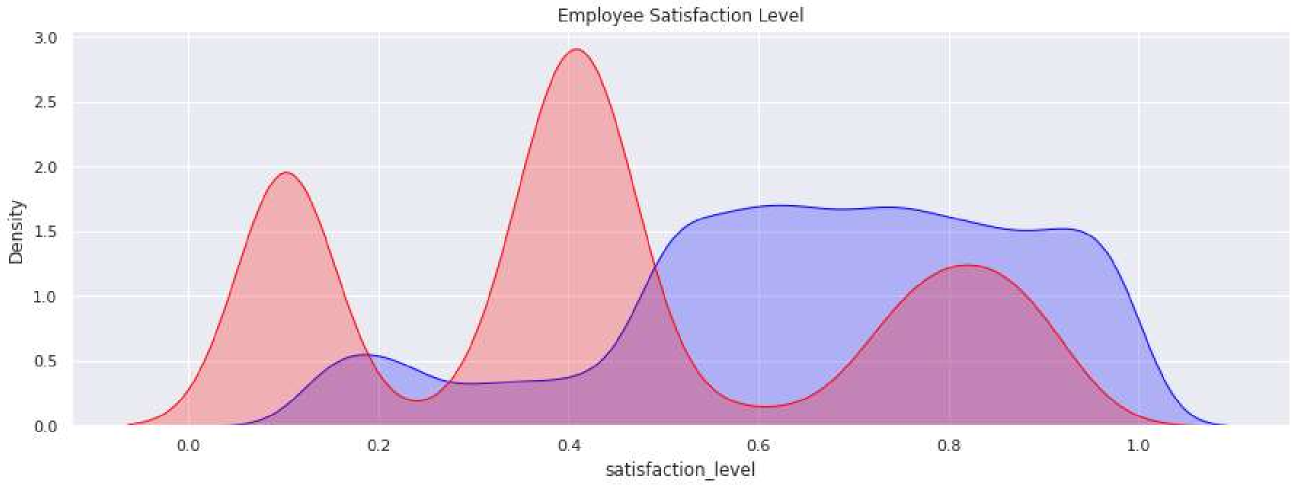
```
Text(0.5, 1.0, 'Employee Evaluation Distribution left vs related')
```




```
fig=plt.figure(figsize=(15,5))
ax=sns.kdeplot(dataset.loc[(dataset["left"]==0),'average_montly_hours'],color='blue',shade
ax=sns.kdeplot(dataset.loc[(dataset["left"]==1),'average_montly_hours'],color='red',shade=
plt.title('Employee Evaluation Distribution left vs relained')
```

```
Text(0.5, 1.0, 'Employee Evaluation Distribution left vs relained')
```


```
fig=plt.figure(figsize=(15,5))
ax=sns.kdeplot(dataset.loc[(dataset["left"]==0),'satisfaction_level'],color='blue',shade=T
ax=sns.kdeplot(dataset.loc[(dataset["left"]==1),'satisfaction_level'],color='red',shade=Tr
plt.title('Employee Satisfaction Level')
```




```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
dataset['Department']=le.fit_transform(dataset['Department'])
```

```
from sklearn.model_selection import train_test_split
x=dataset.Department
y=dataset.salary
xtrain, xtest,ytrain,ytest = train_test_split(x,y,test_size=0.30,random_state=99)
xtrain.shape,xtest.shape,ytrain.shape,ytest.shape
```

```
((10499,), (4500,), (10499,), (4500,))
```

```
from sklearn.linear_model import LogisticRegression
model=LogisticRegression()
print(model)
```

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

0s completed at 2:04 AM