In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
from IPython import get_ipython
import warnings
warnings.filterwarnings("ignore")
```

In [2]:

```
data = pd.read_csv('hr_data.csv')
```

In [3]:

```
data.head()
```

Out[3]:

	satisfaction_level	last_evaluation	number_project	average_montly_hours	time_spend_compan
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	

In [4]:

```
data.tail()
```

Out[4]:

	satisfaction_level	last_evaluation	number_project	average_montly_hours	time_spend_con
14994	0.40	0.57	2	151	
14995	0.37	0.48	2	160	
14996	0.37	0.53	2	143	
14997	0.11	0.96	6	280	
14998	0.37	0.52	2	158	

In [5]:

```
1 data.shape
```

Out[5]:

```
(14999, 10)
```

In [6]:

```
1 data.columns
```

Out[6]:

```
Index(['satisfaction_level', 'last_evaluation', 'number_project',
       'average_montly_hours', 'time_spend_company', 'Work_accident', 'lef
t',
       'promotion_last_5years', 'sales', 'salary'],
      dtype='object')
```

In [7]:

```
1 data.duplicated().sum()
```

Out[7]:

```
3008
```

In [8]:

```
1 data.drop_duplicates()
```

Out[8]:

	satisfaction_level	last_evaluation	number_project	average_montly_hours	time_spend_con
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	
...	...	...	...	...	
11995	0.90	0.55	3	259	
11996	0.74	0.95	5	266	
11997	0.85	0.54	3	185	
11998	0.33	0.65	3	172	
11999	0.50	0.73	4	180	

11991 rows × 10 columns

In [9]:

```
1 data.isnull().sum()
```

Out[9]:

```
satisfaction_level       0
last_evaluation          0
number_project           0
average_montly_hours     0
time_spend_company       0
Work_accident            0
left                     0
promotion_last_5years    0
sales                    0
salary                   0
dtype: int64
```

In [10]:

```
1 data.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   sales                  14999 non-null  object 
 9   salary                 14999 non-null  object 
dtypes: float64(2), int64(6), object(2)
memory usage: 1.1+ MB
```

In [11]:

```
1 data.describe()
```

Out[11]:

	satisfaction_level	last_evaluation	number_project	average_montly_hours	time_spend_com
count	14999.000000	14999.000000	14999.000000	14999.000000	14999.00
mean	0.612834	0.716102	3.803054	201.050337	3.49
std	0.248631	0.171169	1.232592	49.943099	1.46
min	0.090000	0.360000	2.000000	96.000000	2.00
25%	0.440000	0.560000	3.000000	156.000000	3.00
50%	0.640000	0.720000	4.000000	200.000000	3.00
75%	0.820000	0.870000	5.000000	245.000000	4.00
max	1.000000	1.000000	7.000000	310.000000	10.00

In [12]:

```
1 data.nunique()
```

Out[12]:

```
satisfaction_level        92
last_evaluation           65
number_project             6
average_montly_hours     215
time_spend_company         8
Work_accident              2
left                       2
promotion_last_5years      2
sales                     10
salary                     3
dtype: int64
```

In [13]:

```
1 data['number_project'].unique()
```

Out[13]:

```
array([2, 5, 7, 6, 4, 3], dtype=int64)
```

In [14]:

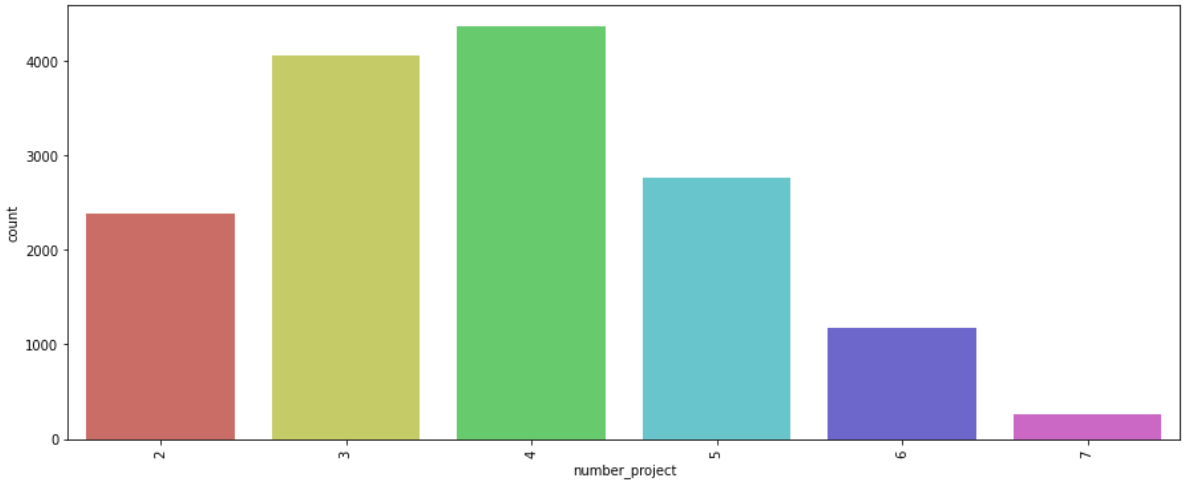
```
1 data['number_project'].value_counts()
```

Out[14]:

```
4    4365
3    4055
5    2761
2    2388
6    1174
7     256
Name: number_project, dtype: int64
```

In [15]:

```
1 plt.figure(figsize=(15,6))
2 sns.countplot('number_project', data = data, palette = 'hls')
3 plt.xticks(rotation = 90)
4 plt.show()
```



In [16]:

```
1 data['time_spend_company'].unique()
```

Out[16]:

```
array([ 3,  6,  4,  5,  2,  8, 10,  7], dtype=int64)
```

In [17]:

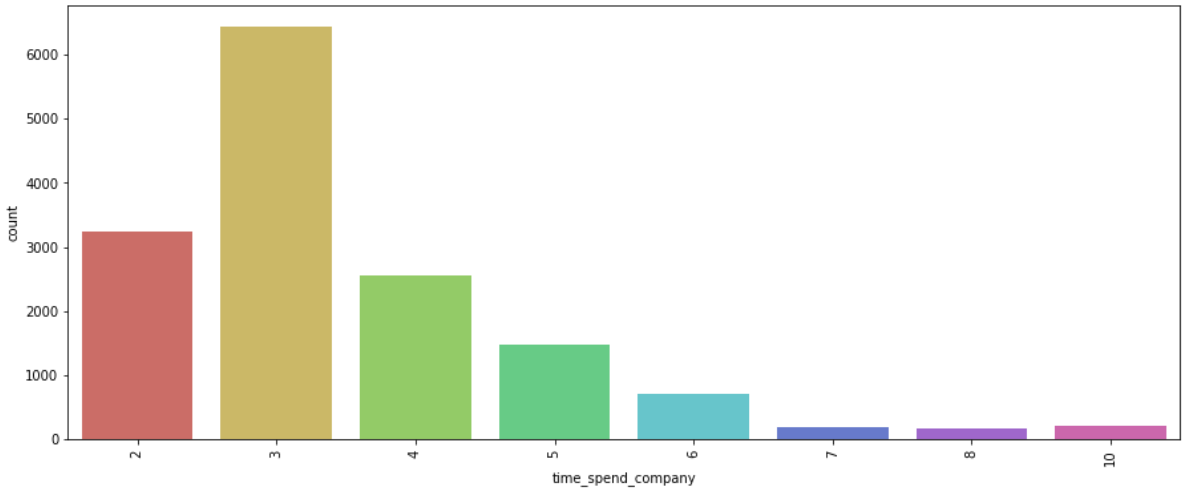
```
1 data['time_spend_company'].value_counts()
```

Out[17]:

```
3     6443
2     3244
4     2557
5     1473
6      718
10     214
7      188
8      162
Name: time_spend_company, dtype: int64
```

In [18]:

```
1 plt.figure(figsize=(15,6))
2 sns.countplot('time_spend_company', data = data, palette = 'hls')
3 plt.xticks(rotation = 90)
4 plt.show()
```



In [19]:

```
1 data['Work_accident'].unique()
```

Out[19]:

```
array([0, 1], dtype=int64)
```

In [20]:

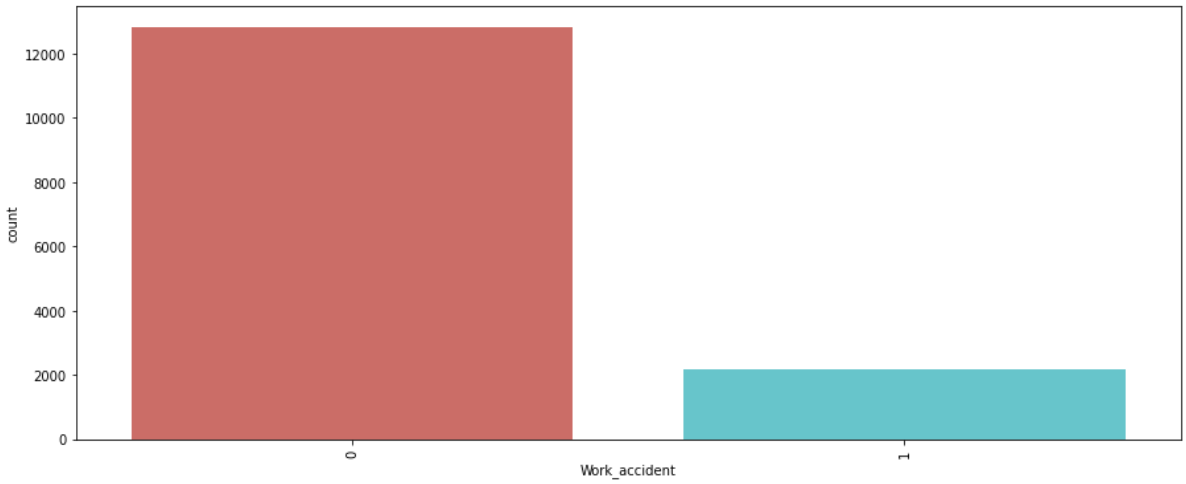
```
1 data['Work_accident'].value_counts()
```

Out[20]:

```
0    12830
1     2169
Name: Work_accident, dtype: int64
```

In [21]:

```
plt.figure(figsize=(15,6))
sns.countplot('Work_accident', data = data, palette = 'hls')
plt.xticks(rotation = 90)
plt.show()
```



In [22]:

```
data['left'].unique()
```

Out[22]:

```
array([1, 0], dtype=int64)
```

In [23]:

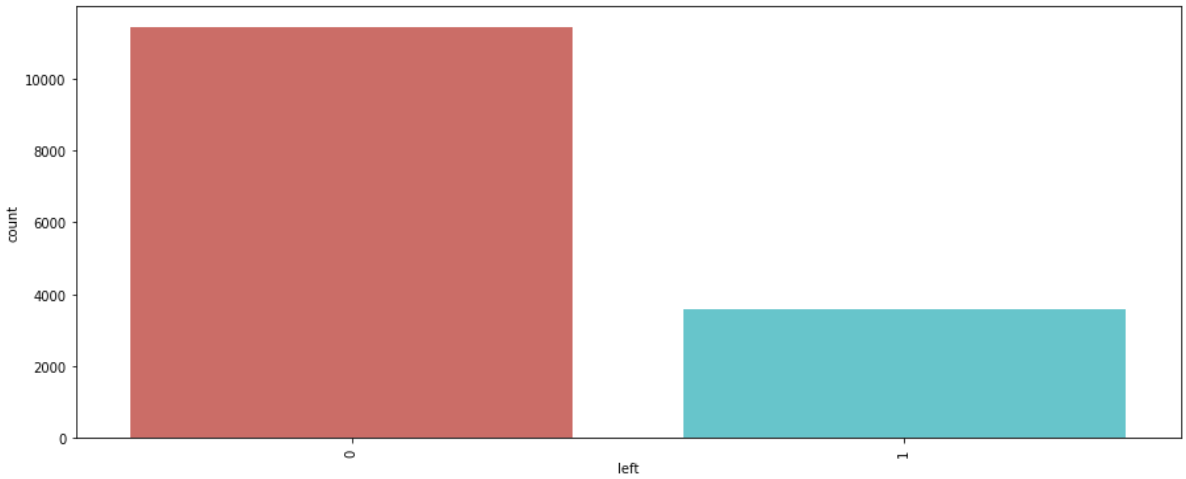
```
data['left'].value_counts()
```

Out[23]:

```
0    11428
1     3571
Name: left, dtype: int64
```

In [24]:

```
plt.figure(figsize=(15,6))
sns.countplot('left', data = data, palette = 'hls')
plt.xticks(rotation = 90)
plt.show()
```



In [25]:

```
data['promotion_last_5years'].unique()
```

Out[25]:

```
array([0, 1], dtype=int64)
```

In [26]:

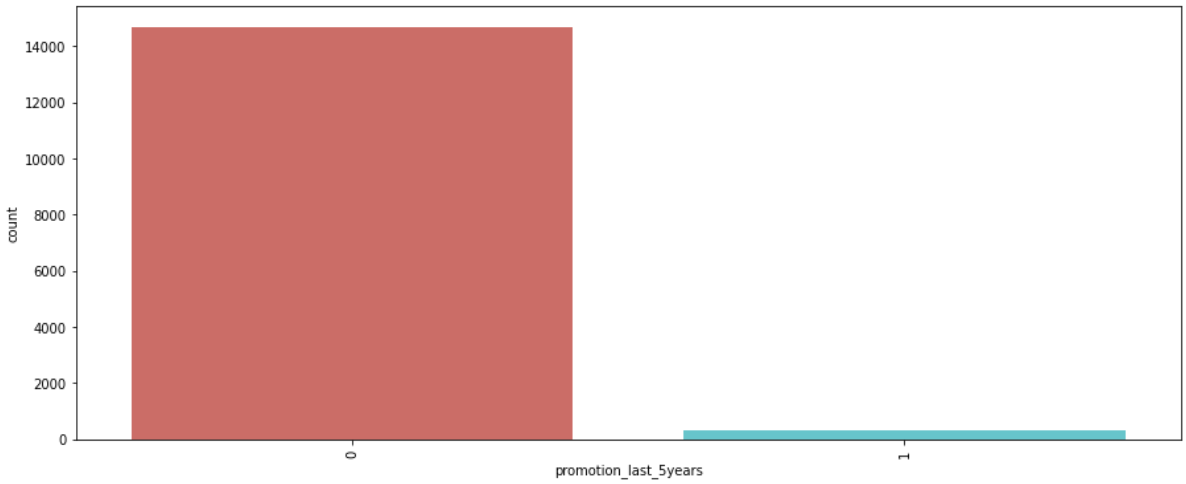
```
data['promotion_last_5years'].value_counts()
```

Out[26]:

```
0    14680
1      319
Name: promotion_last_5years, dtype: int64
```

In [27]:

```
plt.figure(figsize=(15,6))
sns.countplot('promotion_last_5years', data = data, palette = 'hls')
plt.xticks(rotation = 90)
plt.show()
```



In [28]:

```
data['sales'].unique()
```

Out[28]:

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

In [29]:

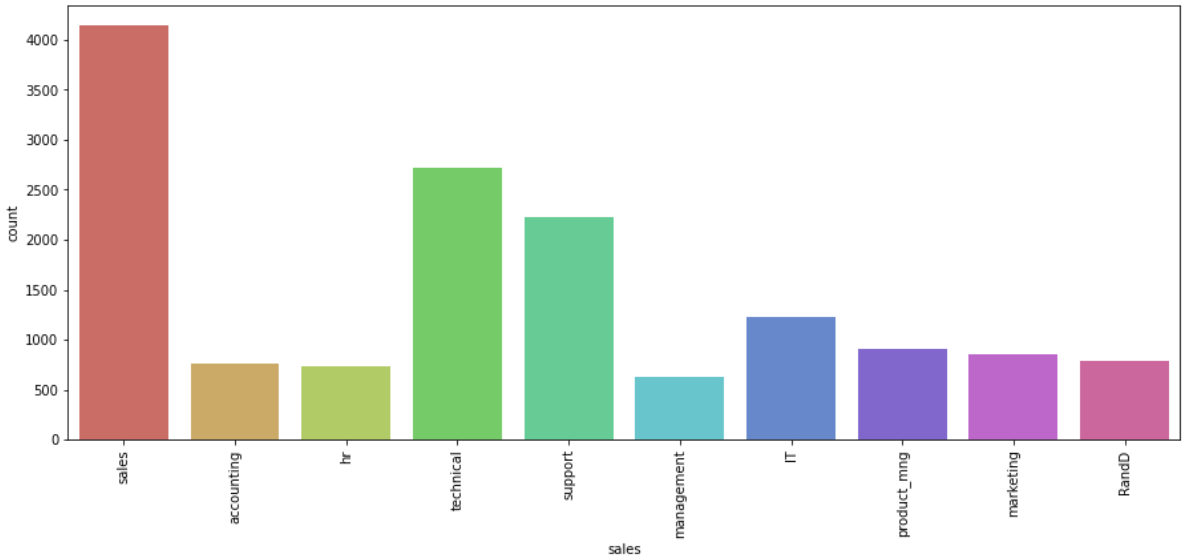
```
data['sales'].value_counts()
```

Out[29]:

```
sales          4140
technical      2720
support        2229
IT             1227
product_mng     902
marketing       858
RandD           787
accounting      767
hr              739
management      630
Name: sales, dtype: int64
```

In [30]:

```
plt.figure(figsize=(15,6))
sns.countplot('sales', data = data, palette = 'hls')
plt.xticks(rotation = 90)
plt.show()
```



In [31]:

```
data['salary'].unique()
```

Out[31]:

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

In [32]:

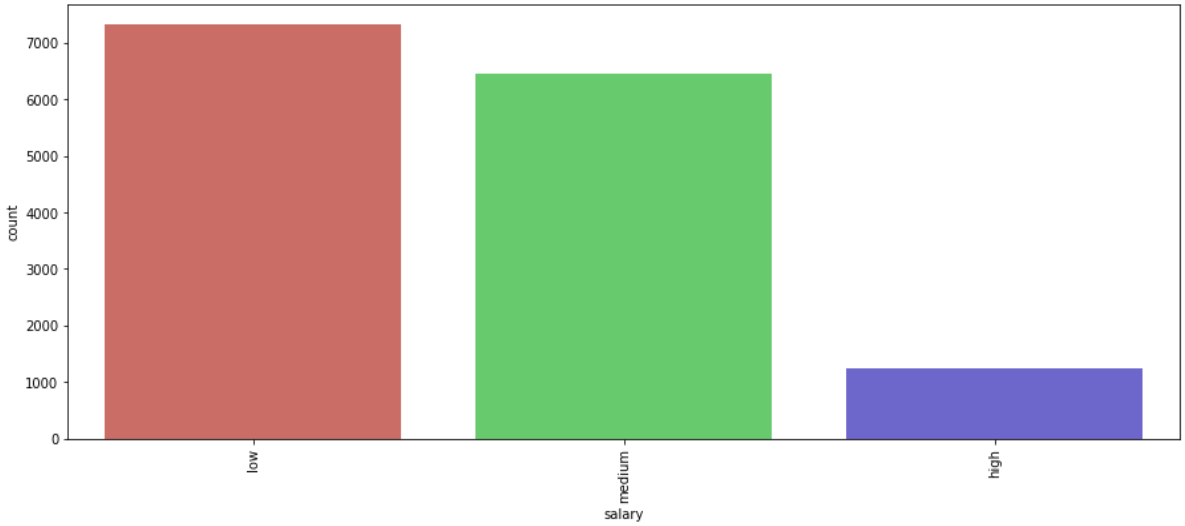
```
data['salary'].value_counts()
```

Out[32]:

```
low       7316
medium    6446
high      1237
Name: salary, dtype: int64
```

In [33]:

```
plt.figure(figsize=(15,6))
sns.countplot('salary', data = data, palette = 'hls')
plt.xticks(rotation = 90)
plt.show()
```



In [34]:

```
data = data.rename(columns = {'sales':'department'})
```

In [35]:

```
data['department']=np.where(data['department'] =='support',
                            'technical', data['department'])
data['department']=np.where(data['department'] =='IT',
                            'technical', data['department'])
```

In [36]:

```
cat_vars=['department','salary']
for var in cat_vars:
    cat_list='var'+'_'+var
    cat_list = pd.get_dummies(data[var], prefix=var)
    data1=data.join(cat_list)
    data=data1
```

In [37]:

```
data.drop(data.columns[[8, 9]], axis=1, inplace=True)
data.columns.values
```

Out[37]:

```
array(['satisfaction_level', 'last_evaluation', 'number_project',
       'average_montly_hours', 'time_spend_company', 'Work_accident',
       'left', 'promotion_last_5years', 'department_RandD',
       'department_accounting', 'department_hr', 'department_management',
       'department_marketing', 'department_product_mng',
       'department_sales', 'department_technical', 'salary_high',
       'salary_low', 'salary_medium'], dtype=object)
```

In [38]:

```
hr_vars=data.columns.values.tolist()
y=['left']
X=[i for i in hr_vars if i not in y]
```

In [41]:

```
cols=['satisfaction_level', 'last_evaluation', 'time_spend_company', 'Work_accident
      'department_RandD', 'department_hr', 'department_management', 'salary_high',
X=data[cols]
y=data['left']
```

In [43]:

```
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,
                                                    random_state=0)
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
logreg = LogisticRegression()
logreg.fit(X_train, y_train)
```

Out[43]:

```
LogisticRegression()
```

In [44]:

```
from sklearn.metrics import accuracy_score
print('Logistic regression accuracy: {:.3f}'.format(accuracy_score(y_test,
                                                                   logreg.predict(X
```

```
Logistic regression accuracy: 0.771
```

In [45]:

```
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier()
rf.fit(X_train, y_train)
```

Out[45]:

```
RandomForestClassifier()
```

In [46]:

```
print('Random Forest Accuracy: {:.3f}'.format(accuracy_score(y_test,
                                                      rf.predict(X_test))))
```

```
Random Forest Accuracy: 0.979
```

In [47]:

```
from sklearn.metrics import classification_report
print(classification_report(y_test, rf.predict(X_test)))
```

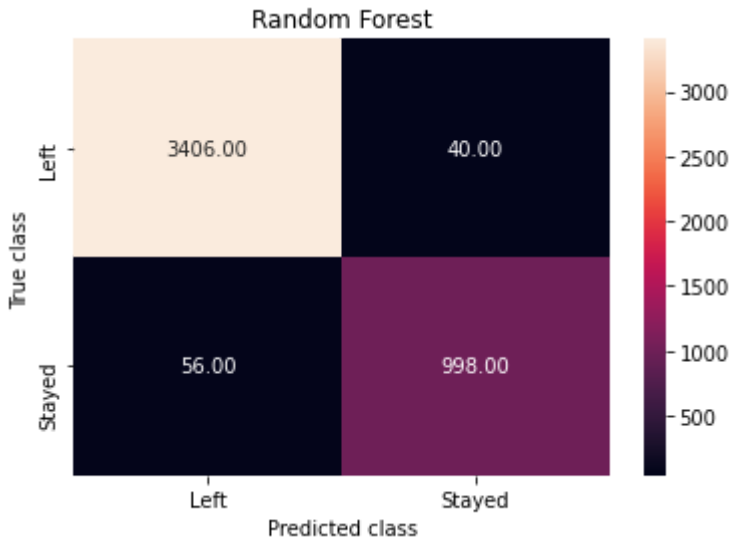
```
              precision    recall  f1-score   support

           0       0.99      0.98      0.99      3462
           1       0.95      0.96      0.95      1038

    accuracy                           0.98      4500
   macro avg       0.97      0.97      0.97      4500
weighted avg       0.98      0.98      0.98      4500
```

In [49]:

```
y_pred = rf.predict(X_test)
from sklearn.metrics import confusion_matrix
import seaborn as sns
forest_cm = metrics.confusion_matrix(y_pred, y_test)
sns.heatmap(forest_cm, annot=True, fmt='.2f',xticklabels = ["Left", "Stayed"] , yti
plt.ylabel('True class')
plt.xlabel('Predicted class')
plt.title('Random Forest')
```

Out[49]:

```
Text(0.5, 1.0, 'Random Forest')
```



In [50]:

```
print(classification_report(y_test, logreg.predict(X_test)))
```

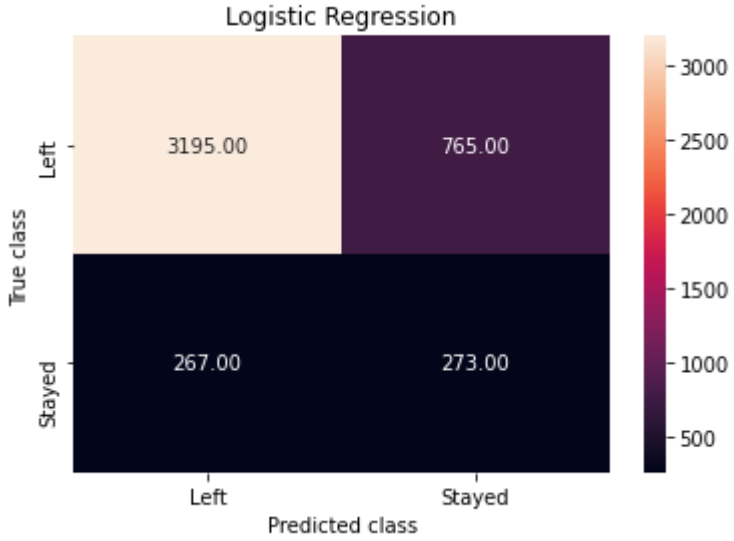
```
              precision    recall  f1-score   support

           0       0.81      0.92      0.86      3462
           1       0.51      0.26      0.35      1038

    accuracy                           0.77      4500
   macro avg       0.66      0.59      0.60      4500
weighted avg       0.74      0.77      0.74      4500
```

In [52]:

```
logreg_y_pred = logreg.predict(X_test)
logreg_cm = metrics.confusion_matrix(logreg_y_pred, y_test)
sns.heatmap(logreg_cm, annot=True, fmt='.2f',
            xticklabels = ["Left", "Stayed"] ,
            yticklabels = ["Left", "Stayed"] )
plt.ylabel('True class')
plt.xlabel('Predicted class')
plt.title('Logistic Regression')
```

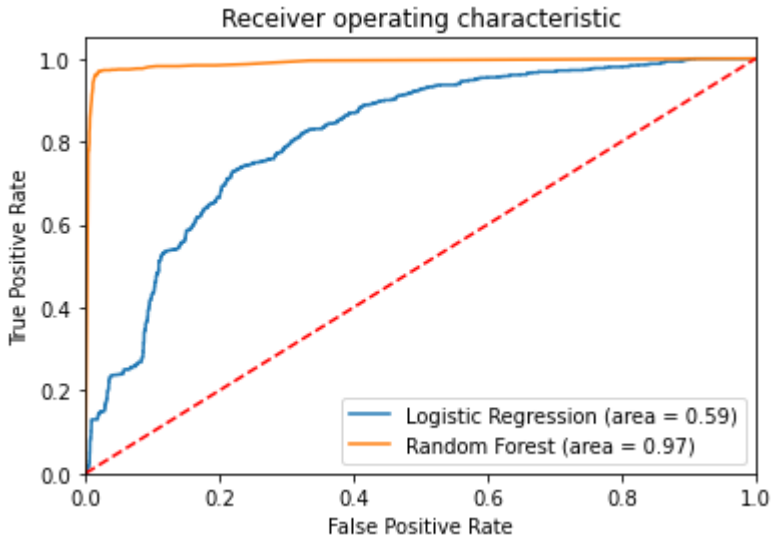
Out[52]:

```
Text(0.5, 1.0, 'Logistic Regression')
```



In [53]:

```
from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve
logit_roc_auc = roc_auc_score(y_test, logreg.predict(X_test))
fpr, tpr, thresholds = roc_curve(y_test, logreg.predict_proba(X_test)[:,1])
rf_roc_auc = roc_auc_score(y_test, rf.predict(X_test))
rf_fpr, rf_tpr, rf_thresholds = roc_curve(y_test, rf.predict_proba(X_test)[:,1])
plt.figure()
plt.plot(fpr, tpr, label='Logistic Regression (area = %0.2f)' % logit_roc_auc)
plt.plot(rf_fpr, rf_tpr, label='Random Forest (area = %0.2f)' % rf_roc_auc)
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic')
plt.legend(loc="lower right")
plt.show()
```



In [54]:

```
feature_labels = np.array(['satisfaction_level', 'last_evaluation', 'time_spend_com
      'department_RandD', 'department_hr', 'department_management', 'salary_high',
importance = rf.feature_importances_
feature_indexes_by_importance = importance.argsort()
for index in feature_indexes_by_importance:
    print('{}-{:.2f}%'.format(feature_labels[index], (importance[index] *100.0)))
```

```
promotion_last_5years-0.22%
department_management-0.26%
department_RandD-0.29%
department_hr-0.30%
salary_high-0.63%
salary_low-1.18%
Work_accident-1.46%
last_evaluation-18.31%
time_spend_company-25.70%
satisfaction_level-51.65%
```