In [1]:

```
import numpy as np
import pandas as pd
```

In [2]:

```
hr_df=pd.read_csv(r'C:\Users\abhin\OneDrive\Documents\Excel\hr_data.csv')
```

In [3]:

```
hr_df
```

Out[3]:

	employee_id	number_project	average_montly_hours	time_spend_company	Work_accide
0	1003	2	157	3	
1	1005	5	262	6	
2	1486	7	272	4	
3	1038	5	223	5	
4	1057	2	159	3	
...	...	...	...	...	
14994	87670	2	151	3	
14995	87673	2	160	3	
14996	87679	2	143	3	
14997	87681	6	280	4	
14998	87684	2	158	3	

14999 rows × 9 columns

In [4]:

```
#numerical analysis
```

In [5]:

```
hr_df.shape
```

Out[5]:

```
(14999, 9)
```

In [6]:

```
hr_df.size
```

Out[6]:

```
134991
```

In [7]:

```
hr_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14999 entries, 0 to 14998
Data columns (total 9 columns):
 #   Column                 Non-Null Count  Dtype 
---  ------                 --------------  ----- 
 0   employee_id            14999 non-null  int64 
 1   number_project         14999 non-null  int64 
 2   average_montly_hours   14999 non-null  int64 
 3   time_spend_company     14999 non-null  int64 
 4   Work_accident          14999 non-null  int64 
 5   left                   14999 non-null  int64 
 6   promotion_last_5years  14999 non-null  int64 
 7   department             14999 non-null  object
 8   salary                 14999 non-null  object
dtypes: int64(7), object(2)
memory usage: 1.0+ MB
```

In [8]:

```
hr_df['department'].unique()
```

Out[8]:

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

In [10]:

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

Out[10]:

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

In [11]:

```
#Loadong our employee satsifaction data
```

In [13]:

```
s_df=pd.read_csv(r'C:\Users\abhin\OneDrive\Documents\Excel\employee_satisfaction_evaluation
```

In [14]:

```
s_df
```

Out[14]:

	EMPLOYEE #	satisfaction_level	last_evaluation
0	1003	0.38	0.53
1	1005	0.80	0.86
2	1486	0.11	0.88
3	1038	0.72	0.87
4	1057	0.37	0.52
...	...	...	...
14994	87670	0.40	0.57
14995	87673	0.37	0.48
14996	87679	0.37	0.53
14997	87681	0.11	0.96
14998	87684	0.37	0.52

14999 rows × 3 columns

In [15]:

```
#merging and joining
```

In [16]:

```
main_df=hr_df.set_index('employee_id').join(s_df.set_index('EMPLOYEE #'))
```

In [18]:

```
main_df=main_df.reset_index()
```

In [19]:

```
main_df
```

Out[19]:

	employee_id	number_project	average_montly_hours	time_spend_company	Work_accide
0	1003	2	157	3	
1	1005	5	262	6	
2	1486	7	272	4	
3	1038	5	223	5	
4	1057	2	159	3	
...	...	...	...	...	
14994	87670	2	151	3	
14995	87673	2	160	3	
14996	87679	2	143	3	
14997	87681	6	280	4	
14998	87684	2	158	3	

14999 rows × 11 columns

In [20]:

```
main_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14999 entries, 0 to 14998
Data columns (total 11 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   employee_id            14999 non-null  int64  
 1   number_project         14999 non-null  int64  
 2   average_montly_hours   14999 non-null  int64  
 3   time_spend_company     14999 non-null  int64  
 4   Work_accident          14999 non-null  int64  
 5   left                   14999 non-null  int64  
 6   promotion_last_5years  14999 non-null  int64  
 7   department             14999 non-null  object 
 8   salary                 14999 non-null  object 
 9   satisfaction_level     14972 non-null  float64
 10  last_evaluation        14972 non-null  float64
dtypes: float64(2), int64(7), object(2)
memory usage: 1.3+ MB
```

In [21]:

```
main_df[main_df.isnull().any(axis=1)]
```

Out[21]:

	employee_id	number_project	average_montly_hours	time_spend_company	Work_acc
18	3794	2	160	3	
19	1140	5	262	5	
33	1230	2	140	3	
53	1340	2	132	3	
72	22316	2	149	3	
92	1581	2	143	3	
107	17376	2	148	3	
120	1739	4	158	4	
137	1847	2	129	3	
175	32923	4	164	2	
191	2160	4	226	6	
352	3150	4	262	6	
376	3250	4	296	2	
402	3405	5	275	5	
427	78130	3	180	4	
442	3635	5	229	5	
468	3755	5	245	5	
543	4150	5	237	5	
892	43615	4	276	5	
1588	42185	5	264	5	
1934	11895	4	225	5	
2343	14170	3	115	2	
2743	16445	5	149	2	
3170	18980	5	186	2	
3609	21580	3	263	2	
3776	22555	4	214	3	
4122	24505	3	192	3	
4740	27950	3	253	3	
5028	29640	4	180	4	
6453	38090	5	166	2	
7005	41535	4	150	3	
7516	44460	4	264	3	
8630	50960	4	167	3	
9455	55770	4	270	3	

	employee_id	number_project	average_montly_hours	time_spend_company	Work_acc
9901	58630	5	252	3	
10647	62595	4	165	3	
10962	64350	5	233	3	
11575	20215	3	192	7	
11967	70005	3	148	3	
12422	72475	5	257	5	
12853	74880	3	136	2	
13482	78780	3	207	7	
13925	81315	3	133	3	

In [24]:

```
main_df.fillna(main_df.mean(),inplace=True)
```

In [25]:

```
main_df[main_df.isnull().any(axis=1)]
```

Out[25]:

number_project	average_montly_hours	time_spend_company	Work_accident	left	promotion_las

In [26]:

```
main_df.loc[main_df['employee_id']==1581]
```

Out[26]:

	employee_id	number_project	average_montly_hours	time_spend_company	Work_accident
92	1581	2	143	3	0

In [27]:

```
main_df.drop(columns='employee_id',inplace=True)
```

In [28]:

```
main_df
```

Out[28]:

	number_project	average_montly_hours	time_spend_company	Work_accident	left	prom
0	2	157	3	0	1	
1	5	262	6	0	1	
2	7	272	4	0	1	
3	5	223	5	0	1	
4	2	159	3	0	1	
...	...	...	...	...	...	
14994	2	151	3	0	1	
14995	2	160	3	0	1	
14996	2	143	3	0	1	
14997	6	280	4	0	1	
14998	2	158	3	0	1	

14999 rows × 10 columns

In [29]:

```
#main_df['department'].values_counts()
```

In [30]:

```
main_df.groupby('department').sum()
```

Out[30]:

department	number_project	average_montly_hours	time_spend_company	Work_accident	le
IT	4683	248119	4256	164	27
RandD	3033	158030	2650	134	12
accounting	2934	154292	2702	96	20
hr	2701	146828	2480	89	21
management	2432	126787	2711	103	9
marketing	3164	171073	3063	138	20
product_mng	3434	180369	3135	132	19
sales	15634	831773	14631	587	101
support	8479	447490	7563	345	55
technical	10548	550793	9279	381	69

In [31]:

```
main_df.groupby('department').mean()
```

Out[31]:

department	number_project	average_montly_hours	time_spend_company	Work_accident	
IT	3.816626	202.215974	3.468623	0.133659	0.2
RandD	3.853875	200.800508	3.367217	0.170267	0.1
accounting	3.825293	201.162973	3.522816	0.125163	0.2
hr	3.654939	198.684709	3.355886	0.120433	0.2
management	3.860317	201.249206	4.303175	0.163492	0.1
marketing	3.687646	199.385781	3.569930	0.160839	0.2
product_mng	3.807095	199.965632	3.475610	0.146341	0.2
sales	3.776329	200.911353	3.534058	0.141787	0.2
support	3.803948	200.758188	3.393001	0.154778	0.2
technical	3.877941	202.497426	3.411397	0.140074	0.2

In [33]:

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

Out[33]:

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

In [34]:

```
#Data Visualization
```

In [36]:

```
import matplotlib.pyplot as plt
import seaborn as sns
```

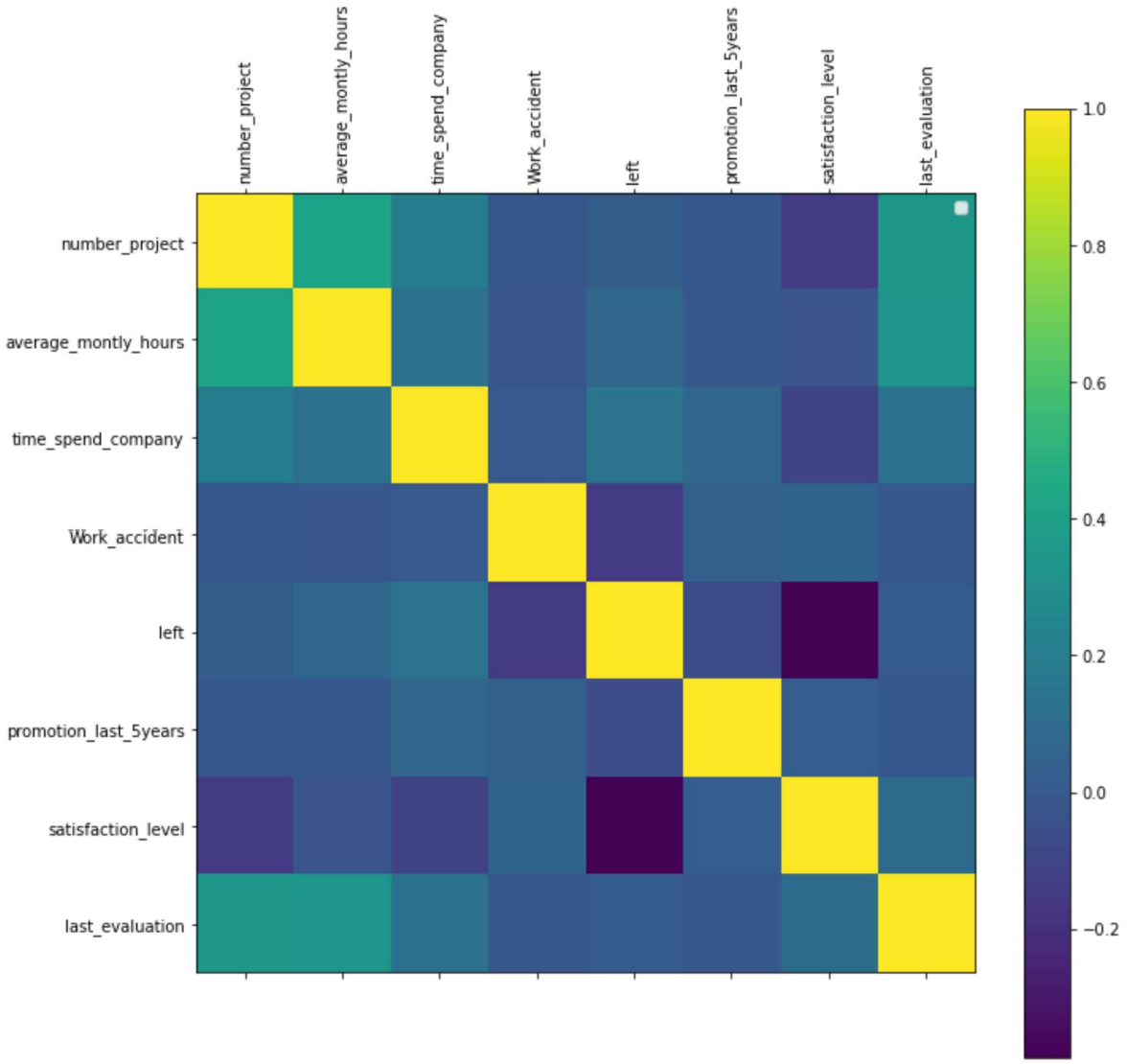
In [38]:

```
def plot_corr(df,size=11):

    corr=df.corr()
    fig,ax=plt.subplots(figsize=(size,size))
    ax.legend()
    cax=ax.matshow(corr)
    fig.colorbar(cax)
    plt.xticks(range(len(corr.columns)), corr.columns, rotation='vertical')
    plt.yticks(range(len(corr.columns)), corr.columns)


plot_corr(main_df)
```

```
No handles with labels found to put in legend.
```

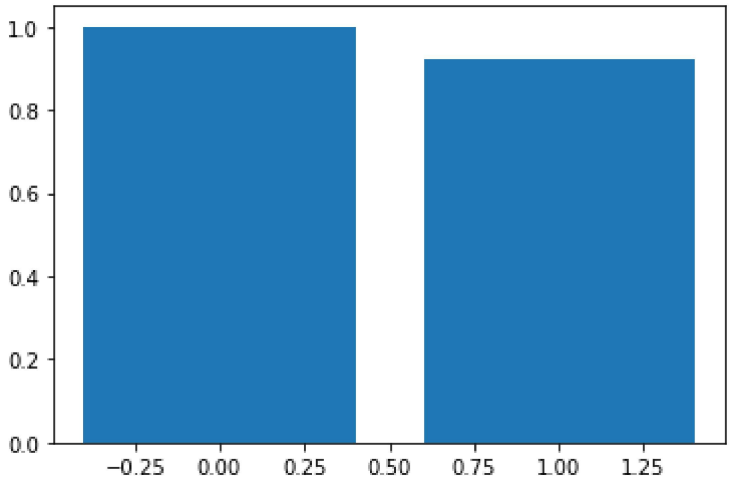


In [40]:

```
plt.bar(x=main_df['left'],height=main_df['satisfaction_level'])
```

Out[40]:

```
<BarContainer object of 14999 artists>
```

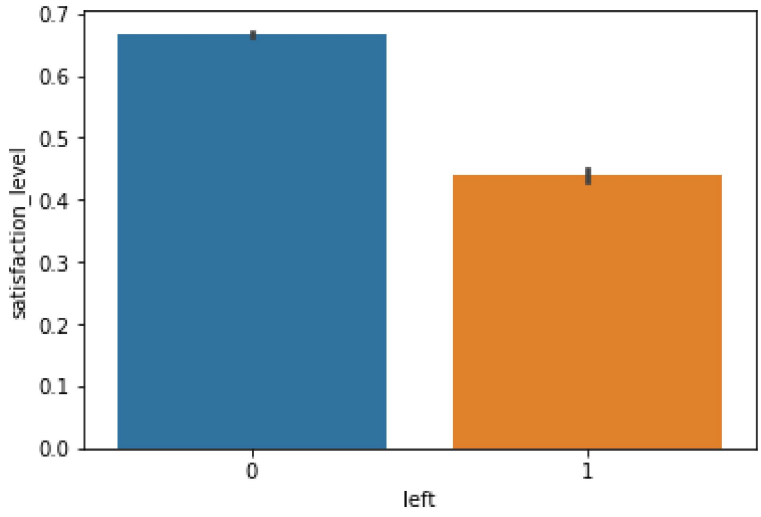


In [41]:

```
sns.barplot(x='left',y='satisfaction_level',data=main_df)
```

Out[41]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1abfa2e3790>
```

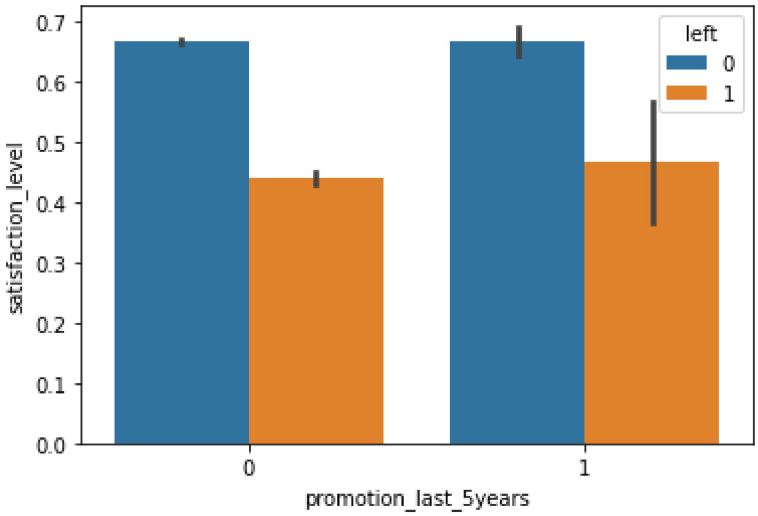


In [42]:

```
sns.barplot(x='promotion_last_5years',y='satisfaction_level',data=main_df,hue='left')
```

Out[42]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1abf9eb3bb0>
```

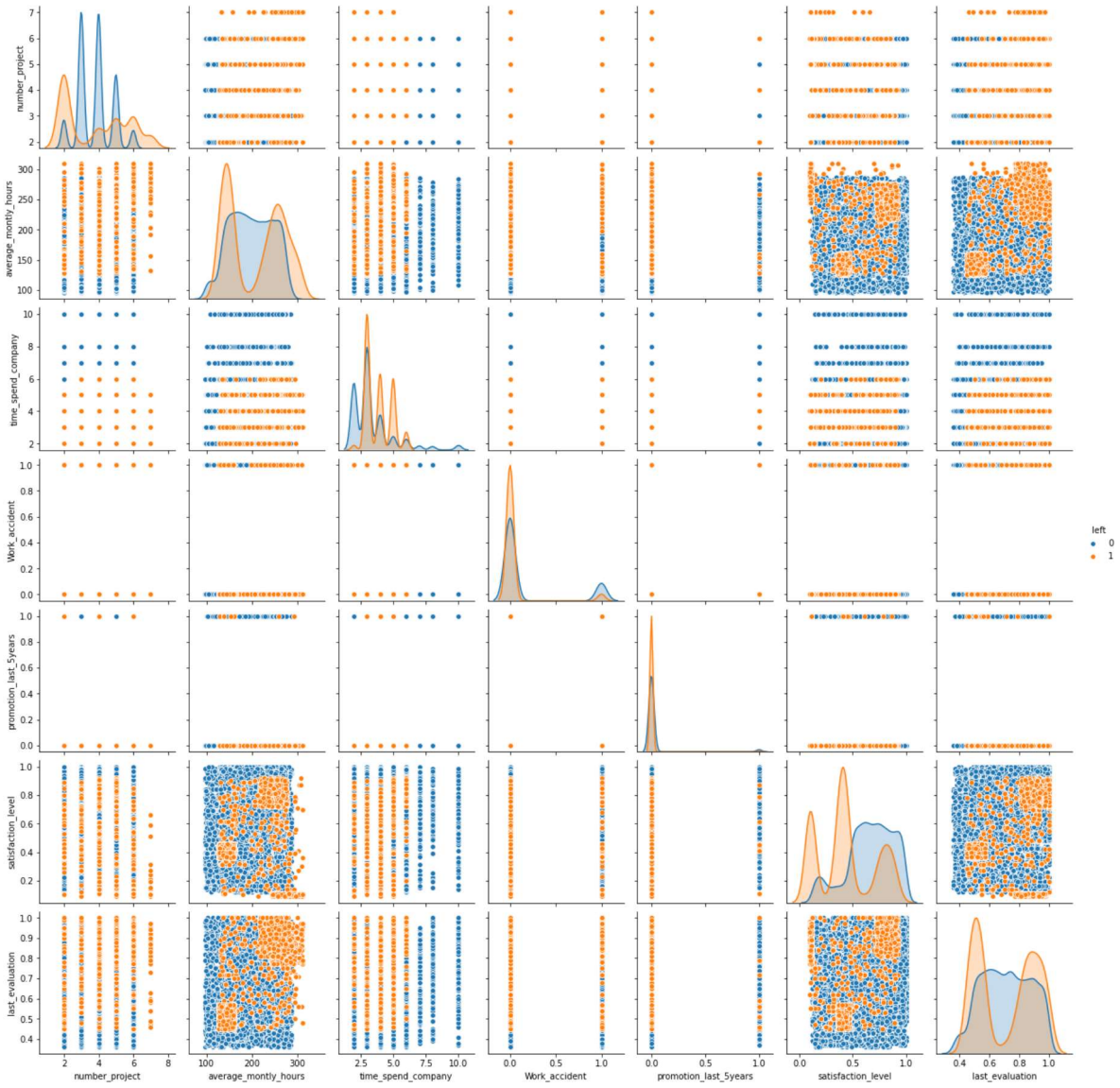


In [43]:

```
sns.pairplot(main_df,hue='left')
```

Out[43]:

```
<seaborn.axisgrid.PairGrid at 0x1abfa00ffd0>
```



In [44]:

```
#data processing
```

In [46]:

```
y=main_df[['department','salary']]
```

In [47]:

```
y
```

Out[47]:

	department	salary
0	sales	low
1	sales	medium
2	sales	medium
3	sales	low
4	sales	low
...	...	...
14994	support	low
14995	support	low
14996	support	low
14997	support	low
14998	support	low

14999 rows × 2 columns

In [48]:

```
from sklearn.preprocessing import LabelEncoder

le=LabelEncoder()

k=le.fit_transform(main_df['salary'])
```

In [50]:

```
main_df['salary_num']=k
```

In [51]:

```
main_df
```

Out[51]:

	number_project	average_montly_hours	time_spend_company	Work_accident	left	prom
0	2	157	3	0	1	
1	5	262	6	0	1	
2	7	272	4	0	1	
3	5	223	5	0	1	
4	2	159	3	0	1	
...	...	...	...	...	...	
14994	2	151	3	0	1	
14995	2	160	3	0	1	
14996	2	143	3	0	1	
14997	6	280	4	0	1	
14998	2	158	3	0	1	

14999 rows × 11 columns

In [52]:

```
main_df.loc[main_df['salary']=='high']
```

Out[52]:

	number_project	average_montly_hours	time_spend_company	Work_accident	left	prom
72	2	149	3	0	1	
111	6	289	4	0	1	
189	2	156	3	0	1	
267	2	129	3	0	1	
306	2	149	3	0	1	
...	...	...	...	...	...	
14829	2	148	3	0	1	
14868	2	130	3	0	1	
14902	2	159	3	0	1	
14941	2	131	3	0	1	
14980	5	238	5	0	1	

1237 rows × 11 columns

In [53]:

```
main_df.drop(['salary'],axis=1,inplace=True)
```

In [54]:

```
main_df
```

Out[54]:

	number_project	average_montly_hours	time_spend_company	Work_accident	left	prom
0	2	157	3	0	1	
1	5	262	6	0	1	
2	7	272	4	0	1	
3	5	223	5	0	1	
4	2	159	3	0	1	
...	...	...	...	...	...	
14994	2	151	3	0	1	
14995	2	160	3	0	1	
14996	2	143	3	0	1	
14997	6	280	4	0	1	
14998	2	158	3	0	1	

14999 rows × 10 columns

In [55]:

```
z=le.fit_transform(main_df['department'])
```

In [56]:

```
z
```

Out[56]:

```
array([7, 7, 7, ..., 8, 8, 8])
```

In [57]:

```
main_df['department_num']=z
```

In [60]:

```
main_df.loc[main_df['department']=='IT']
```

Out[60]:

	number_project	average_montly_hours	time_spend_company	Work_accident	left	prom
61	7	308	4	0	1	
62	6	244	5	0	1	
63	2	132	3	0	1	
64	6	286	4	0	1	
65	6	161	4	0	1	
...	...	...	...	...	...	
14930	6	268	4	0	1	
14931	5	240	5	0	1	
14932	2	127	3	0	1	
14933	7	264	4	0	1	
14938	4	271	5	0	1	

1227 rows × 11 columns

In [63]:

```
main_df.drop(['department'],axis=1,inplace=True)
```

In [64]:

```
main_df
```

Out[64]:

	number_project	average_montly_hours	time_spend_company	Work_accident	left	promotion_last_5ye
0	2	157	3	0	1	
1	5	262	6	0	1	
2	7	272	4	0	1	
3	5	223	5	0	1	
4	2	159	3	0	1	
...	...	...	...	...	...	
14994	2	151	3	0	1	
14995	2	160	3	0	1	
14996	2	143	3	0	1	

In [65]:

```
X=main_df.drop(['left'],axis=1)
```

In [66]:

```
X
```

Out[66]:

	number_project	average_montly_hours	time_spend_company	Work_accident	promotion
0	2	157	3	0	
1	5	262	6	0	
2	7	272	4	0	
3	5	223	5	0	
4	2	159	3	0	
...	...	...	...	...	
14994	2	151	3	0	
14995	2	160	3	0	
14996	2	143	3	0	
14997	6	280	4	0	
14998	2	158	3	0	

14999 rows × 9 columns

In [67]:

```
y=main_df['left']
```

In [68]:

```
y.size
```

Out[68]:

```
14999
```

In [72]:

```
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)
```

In [73]:

```
X_test
```

Out[73]:

	number_project	average_montly_hours	time_spend_company	Work_accident	promotion_
12442	5	229	5	0	
11019	2	285	3	0	
5533	5	268	2	0	
9176	4	184	4	0	
1434	2	140	3	0	
...	...	...	...	...	
4104	4	184	4	0	
13482	3	207	7	0	
6658	3	153	2	0	
10397	4	173	4	0	
13912	3	163	3	1	

4500 rows × 9 columns

In [74]:

```
y_test
```

Out[74]:

```
12442    1
11019    0
5533     0
9176     0
1434     1
        ..
4104     0
13482    0
6658     0
10397    0
13912    0
Name: left, Length: 4500, dtype: int64
```

In [75]:

```
#standard scaler
```

In [76]:

```
#model classification
```

In [77]:

```
#decision tree
```

In [78]:

```
from sklearn.metrics import accuracy_score
```

In [86]:

```
from sklearn.tree import DecisionTreeClassifier

dt=DecisionTreeClassifier()

dt.fit(X_train,y_train)
```

Out[86]:

```
DecisionTreeClassifier()
```

In [87]:

```
prediction_dt=dt.predict(X_test)
```

In [88]:

```
prediction_dt
```

Out[88]:

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

In [89]:

```
y_test
```

Out[89]:

```
3587     0
4688     0
12241    1
1429     1
13784    0
        ..
12851    0
7791     0
5625     0
3574     0
7150     0
Name: left, Length: 4500, dtype: int64
```

In [90]:

```
accuracy_dt=accuracy_score(y_test,prediction_dt)*100
```

In [91]:

```
accuracy_dt
```

Out[91]:

```
97.51111111111112
```

In [92]:

```
X_test
```

Out[92]:

	number_project	average_montly_hours	time_spend_company	Work_accident	promotion
3587	3	142	3	0	
4688	4	255	4	0	
12241	2	156	3	0	
1429	4	255	5	0	
13784	6	263	7	0	
...	...	...	...	...	
12851	4	155	4	0	
7791	6	214	6	0	
5625	3	179	3	0	
3574	4	276	2	0	
7150	3	224	3	0	

4500 rows × 9 columns

In [93]:

```
Catagory=['Employee will stay','Employee will Leave']
```

In [94]:

```
custom_dt=[[1,500,3,6,0,0.90,0.89,1,8]]
```

In [95]:

```
print(int(dt.predict(custom_dt)))
```

```
1
```

In [96]:

```
Catagory[int(dt.predict(custom_dt))]
```

Out[96]:

```
'Employee will Leave'
```

In [97]:

```
dt.feature_importances_
```

Out[97]:

```
array([1.09362544e-01, 1.22716479e-01, 1.30517823e-01, 7.54562356e-04,
       1.70543960e-05, 5.03378095e-01, 1.19375716e-01, 2.07055869e-03,
       1.18071674e-02])
```

In [98]:

```
:ance=pd.DataFrame(dt.feature_importances_,index=X_train.columns,columns=['Importance']).sor
```

In [99]:

```
feature_importance
```

Out[99]:

	Importance
satisfaction_level	0.503378
time_spend_company	0.130518
average_montly_hours	0.122716
last_evaluation	0.119376
number_project	0.109363
department_num	0.011807
salary_num	0.002071
Work_accident	0.000755
promotion_last_5years	0.000017

In [100]:

```
X_train
```

Out[100]:

	number_project	average_montly_hours	time_spend_company	Work_accident	promotion_
13853	3	153	2	0	
11407	4	149	3	0	
11804	3	252	2	0	
8534	4	216	2	1	
8000	3	193	3	0	
...	...	...	...	...	
8803	4	219	4	1	
2545	6	169	3	0	
3132	3	148	3	0	
9332	4	140	2	0	
6267	5	202	6	0	

10499 rows × 9 columns

In [101]:

```
#KNN
```

In [102]:

```
# Data Processing of KNN
```

In [103]:

```
from sklearn.preprocessing import StandardScaler
```

In [104]:

```
sc=StandardScaler().fit(X_train)
X_train_std=sc.transform(X_train)
X_test_std=sc.transform(X_test)
```

In [105]:

```
X_train_std
```

Out[105]:

```
array([[-0.65401011, -0.96913658, -1.02243962, ..., -0.38493855,
         1.03933532, -0.30624781],
       [ 0.15628634, -1.04924243, -0.33983797, ...,  1.36986059,
         1.03933532,  0.39030673],
       [-0.65401011,  1.01348321, -1.02243962, ...,  0.96040746,
        -0.55528422,  0.39030673],
       ...,
       [-0.65401011, -1.06926889, -0.33983797, ...,  0.08300789,
        -0.55528422, -1.3510796 ],
       [ 0.15628634, -1.22948059, -1.02243962, ..., -0.44343185,
        -0.55528422,  0.39030673],
       [ 0.96658279,  0.01216008,  1.70796698, ..., -0.15096533,
        -0.55528422,  0.39030673]])
```

In [106]:

```
X_test_std
```

Out[106]:

```
array([[-0.65401011, -1.18942767, -0.33983797, ..., -1.08685821,
        -0.55528422,  0.39030673],
       [ 0.15628634,  1.0735626 ,  0.34276368, ..., -0.67740507,
        -0.55528422, -1.00280234],
       [-1.46430656, -0.90905719, -0.33983797, ..., -1.49631134,
        -0.55528422,  0.39030673],
       ...,
       [-0.65401011, -0.44844856, -0.33983797, ..., -0.85288499,
         1.03933532,  0.73858399],
       [ 0.15628634,  1.49411831, -1.02243962, ...,  0.60944763,
         1.03933532, -2.04763413],
       [-0.65401011,  0.45274226, -0.33983797, ...,  1.07739407,
         1.03933532,  0.73858399]])
```

In [107]:

```
from sklearn.neighbors import KNeighborsClassifier

knn=KNeighborsClassifier(n_neighbors=3)
knn.fit(X_train_std,y_train)
```

Out[107]:

```
KNeighborsClassifier(n_neighbors=3)
```

In [108]:

```
prediction_knn=knn.predict(X_test_std)
```

In [109]:

```
accuracy_knn=accuracy_score(y_test,prediction_knn)*100
```

In [110]:

```
accuracy_knn
```

Out[110]:

```
95.62222222222222
```

In [111]:

```
prediction_knn
```

Out[111]:

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

In [112]:

```
y_test
```

Out[112]:

```
3587     0
4688     0
12241    1
1429     1
13784    0
        ..
12851    0
7791     0
5625     0
3574     0
7150     0
Name: left, Length: 4500, dtype: int64
```

In [113]:

```
k_range=range(1,26)
scores={}
scores_list=[]


for k in k_range:
    knn=KNeighborsClassifier(n_neighbors=k)
    knn.fit(X_train_std,y_train)
    prediction_knn=knn.predict(X_test_std)
    scores[k]=accuracy_score(y_test,prediction_knn)*100
    scores_list.append(accuracy_score(y_test,prediction_knn))
```

In [114]:

```
scores
```

Out[114]:

```
{1: 96.35555555555555,
 2: 96.13333333333334,
 3: 95.62222222222222,
 4: 95.57777777777777,
 5: 95.04444444444444,
 6: 95.66666666666667,
 7: 95.37777777777777,
 8: 95.39999999999999,
 9: 95.19999999999999,
 10: 95.35555555555555,
 11: 95.08888888888889,
 12: 95.04444444444444,
 13: 94.97777777777779,
 14: 95.04444444444444,
 15: 94.88888888888889,
 16: 94.97777777777779,
 17: 94.82222222222222,
 18: 94.91111111111111,
 19: 94.66666666666667,
 20: 94.64444444444445,
 21: 94.46666666666667,
 22: 94.4888888888889,
 23: 94.31111111111112,
 24: 94.44444444444444,
 25: 94.13333333333334}
```

In [115]:

```
scores_list
```

Out[115]:

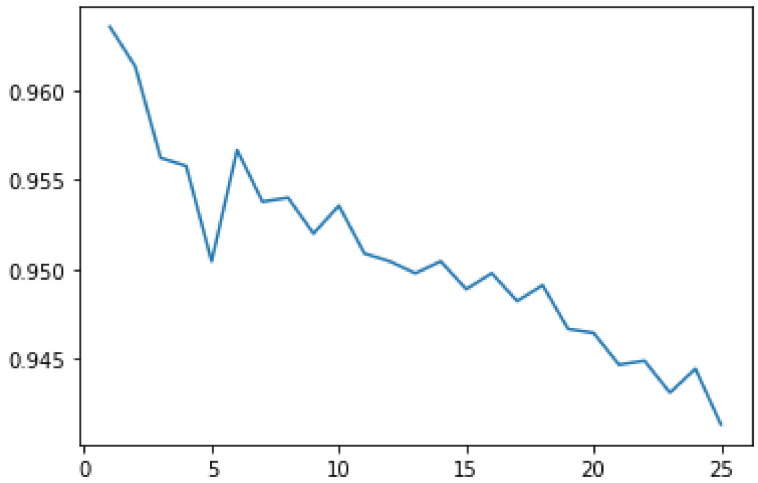
```
[0.9635555555555556,
 0.9613333333333334,
 0.9562222222222222,
 0.9557777777777777,
 0.9504444444444444,
 0.9566666666666667,
 0.9537777777777777,
 0.954,
 0.952,
 0.9535555555555556,
 0.9508888888888889,
 0.9504444444444444,
 0.9497777777777778,
 0.9504444444444444,
 0.9488888888888889,
 0.9497777777777778,
 0.9482222222222222,
 0.9491111111111111,
 0.9466666666666667,
 0.9464444444444444,
 0.9446666666666667,
 0.9448888888888889,
 0.9431111111111111,
 0.9444444444444444,
 0.9413333333333334]
```

In [116]:

```
plt.plot(k_range,scores_list)
```

Out[116]:

```
[<matplotlib.lines.Line2D at 0x1abfff474f0>]
```



In [117]:

```
X_test.head(1)
```

Out[117]:

	number_project	average_montly_hours	time_spend_company	Work_accident	promotio
3587	3	142	3	0	

In [118]:

```
X_knn=np.array([[20,500,10,6,0,0.10,0.30,1,8]])
X_knn_std=sc.transform(X_knn)
```

In [119]:

```
X_knn_std
```

Out[119]:

```
array([[13.12102954,  5.98004591,  4.43837358, 16.79193245, -0.14697495,
        -2.05908203, -2.43220422, -0.55528422,  0.73858399]])
```

In [120]:

```
X_knn_prediction=knn.predict(X_knn_std)
```

In [121]:

```
X_knn_prediction
```

Out[121]:

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

In [122]:

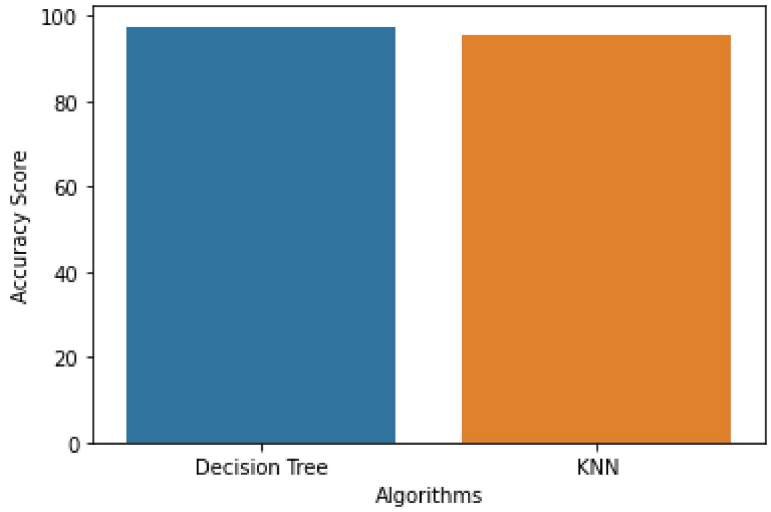
```
Catagory[int(dt.predict(custom_dt))]
```

Out[122]:

```
'Employee will Leave'
```

In [123]:

```
algorithms=['Decision Tree','KNN']
scores=[accuracy_dt,accuracy_knn]
plt.xlabel("Algorithms")
plt.ylabel("Accuracy Score")
sns.barplot(algorithms,scores)
plt.show()
```



In [ ]:

```
```