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

In [2]:
```
dataset=pd.read_csv("HR_comma_sep.csv")
```

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

Out[3]:

	satisfaction_level	last_evaluation	number_project	average_montly_hours	time_spend_company	Work_accident	left	promotion_last_5years	Depa
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]:
```
dataset
```

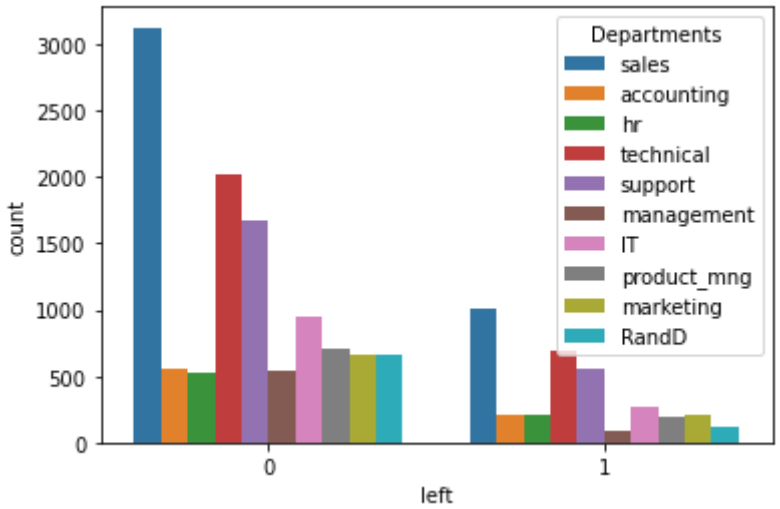
Out[4]:

	satisfaction_level	last_evaluation	number_project	average_montly_hours	time_spend_company	Work_accident	left	promotion_last_5years	I
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	
...	...	...	...	...	...	...	...	...	
14994	0.40	0.57	2	151	3	0	1	0	
14995	0.37	0.48	2	160	3	0	1	0	
14996	0.37	0.53	2	143	3	0	1	0	
14997	0.11	0.96	6	280	4	0	1	0	
14998	0.37	0.52	2	158	3	0	1	0	

14999 rows × 10 columns

In [5]:
```
sns.countplot(x='left',hue='Departments ',data=dataset)
```

Out[5]: <AxesSubplot:xlabel='left', ylabel='count'>



In [6]:
```
sal=pd.get_dummies(dataset['salary'],drop_first=True)
```

In [7]:
```
sal
```

Out[7]:

	low	medium
0	1	0
1	0	1
2	0	1
3	1	0
4	1	0
...	...	...
14994	1	0
14995	1	0
14996	1	0
14997	1	0
14998	1	0

14999 rows × 2 columns

In [8]:
```
dep=pd.get_dummies(dataset['Departments '],drop_first=True)
```

In [9]:
```
dep
```

Out[9]:

	RandD	accounting	hr	management	marketing	product_mng	sales	support	technical
0	0	0	0	0	0	0	1	0	0
1	0	0	0	0	0	0	1	0	0
2	0	0	0	0	0	0	1	0	0
3	0	0	0	0	0	0	1	0	0
4	0	0	0	0	0	0	1	0	0
...	...	...	...	...	...	...	...	...	...
14994	0	0	0	0	0	0	0	1	0
14995	0	0	0	0	0	0	0	1	0
14996	0	0	0	0	0	0	0	1	0
14997	0	0	0	0	0	0	0	1	0
14998	0	0	0	0	0	0	0	1	0

14999 rows × 9 columns

In [10]:
```
dataset.drop(['Departments ','salary'],axis=1,inplace=True)
```

In [11]:
```
dataset
```

Out[11]:

	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
...	...	...	...	...	...	...	...	...
14994	0.40	0.57	2	151	3	0	1	0
14995	0.37	0.48	2	160	3	0	1	0
14996	0.37	0.53	2	143	3	0	1	0
14997	0.11	0.96	6	280	4	0	1	0
14998	0.37	0.52	2	158	3	0	1	0

14999 rows × 8 columns

In [12]:
```
dataset=pd.concat([dataset,sal,dep],axis=1)
```

In [13]:
```
dataset
```

Out[13]:

	satisfaction_level	last_evaluation	number_project	average_montly_hours	time_spend_company	Work_accident	left	promotion_last_5years	I
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	
...	...	...	...	...	...	...	...	...	...
14994	0.40	0.57	2	151	3	0	1	0	
14995	0.37	0.48	2	160	3	0	1	0	
14996	0.37	0.53	2	143	3	0	1	0	
14997	0.11	0.96	6	280	4	0	1	0	
14998	0.37	0.52	2	158	3	0	1	0	

14999 rows × 19 columns

FEATURE SCALING

In [14]:
```
from sklearn.preprocessing import StandardScaler
```

In [15]:
```
scalar=StandardScaler()
scalar.fit(dataset)
```

Out[15]:
```
StandardScaler()
```

In [16]:
```
dataset
```

Out[16]:

	satisfaction_level	last_evaluation	number_project	average_montly_hours	time_spend_company	Work_accident	left	promotion_last_5years	I
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	
...	...	...	...	...	...	...	...	...	
14994	0.40	0.57	2	151	3	0	1	0	
14995	0.37	0.48	2	160	3	0	1	0	
14996	0.37	0.53	2	143	3	0	1	0	
14997	0.11	0.96	6	280	4	0	1	0	
14998	0.37	0.52	2	158	3	0	1	0	

14999 rows × 19 columns

SPLITTING THE DATA

In [17]:
```
x=dataset.drop('left',axis=1).values
x
```

Out[17]:
```
array([[0.38, 0.53, 2.  , ..., 1.  , 0.  , 0.  ],
       [0.8 , 0.86, 5.  , ..., 1.  , 0.  , 0.  ],
       [0.11, 0.88, 7.  , ..., 1.  , 0.  , 0.  ],
       ...,
       [0.37, 0.53, 2.  , ..., 0.  , 1.  , 0.  ],
       [0.11, 0.96, 6.  , ..., 0.  , 1.  , 0.  ],
       [0.37, 0.52, 2.  , ..., 0.  , 1.  , 0.  ]])
```

In [18]:
```
y=dataset['left'].values
y
```

Out[18]:
```
array([1, 1, 1, ..., 1, 1, 1], dtype=int64)
```

In [19]:
```
from sklearn.model_selection import train_test_split
```

In [20]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=1)
```

LOGISTIC REGRESSION

In [21]:
```
from sklearn.linear_model import LogisticRegression
```

In [22]:
```
reg=LogisticRegression()
reg.fit(x_train,y_train)
```

Out[22]:
```
LogisticRegression()
```

In [23]:
```
y_pred=reg.predict(x_test)
y_pred
```

Out[23]:
```
array([0, 0, 0, ..., 0, 0, 0], dtype=int64)
```

In [24]:
```
y_test
```

Out[24]:
```
array([0, 0, 0, ..., 0, 0, 1], dtype=int64)
```

In [25]:
```
model=LogisticRegression(solver='liblinear',random_state=0).fit(x,y)
```

In [26]:
```
model.predict(x)
```

Out[26]:
```
array([1, 0, 1, ..., 1, 1, 1], dtype=int64)
```

CONFUSION MATRIX

In [27]:
```
from sklearn.metrics import confusion_matrix,accuracy_score
cm=confusion_matrix(y_test,y_pred)
```

In [28]:
```
cm
```

Out[28]:
```
array([[3128,  288],
       [ 642,  442]], dtype=int64)
```

In [29]:
```
accuracy_score(y_test,y_pred)
```

Out[29]:
```
0.7933333333333333
```

DECISION TREE

In [30]:
```
from sklearn.tree import DecisionTreeClassifier,plot_tree
```

In [31]:
```
dc=DecisionTreeClassifier(random_state=0)
dc.fit(x_train,y_train)
```

Out[31]:
```
DecisionTreeClassifier(random_state=0)
```

In [32]:
```
y_pred2=dc.predict(x_test)
y_pred2
```

Out[32]:
```
array([0, 0, 0, ..., 0, 0, 1], dtype=int64)
```

In [33]:
```
cm_dt=confusion_matrix(y_test,y_pred2)
cm_dt
```

Out[33]:
```
array([[3367,   49],
       [  41, 1043]], dtype=int64)
```

In [34]:
```
accuracy_score(y_test,y_pred2)
```

Out[34]:
```
0.98
```

In [35]:
```
df=dc.fit(x_test,y_pred2)
plt.figure(figsize=(15,7))
```

Out[35]:
```
<Figure size 1080x504 with 0 Axes>
```

```
<Figure size 1080x504 with 0 Axes>
```

In [36]:
```
plot_tree(df,filled=True)
```

Out[36]:
```
[Text(154.44, 211.4, 'X[0] <= 0.465\ngini = 0.368\nsamples = 4500\nvalue = [3408, 1092]'),
 Text(58.792500000000004, 199.32, 'X[2] <= 2.5\ngini = 0.477\nsamples = 1272\nvalue = [500, 772]'),
 Text(15.66, 187.24, 'X[1] <= 0.575\ngini = 0.211\nsamples = 501\nvalue = [60, 441]'),
 Text(7.5600000000000005, 175.16, 'X[1] <= 0.44\ngini = 0.102\nsamples = 464\nvalue = [25, 439]'),
 Text(5.4, 163.07999999999998, 'gini = 0.0\nsamples = 14\nvalue = [14, 0]'),
 Text(9.72, 163.07999999999998, 'X[0] <= 0.34\ngini = 0.048\nsamples = 450\nvalue = [11, 439]'),
 Text(4.32, 151.0, 'X[4] <= 4.5\ngini = 0.245\nsamples = 7\nvalue = [6, 1]'),
 Text(2.16, 138.92000000000002, 'gini = 0.0\nsamples = 5\nvalue = [5, 0]'),
 Text(6.48, 138.92000000000002, 'X[15] <= 0.5\ngini = 0.5\nsamples = 2\nvalue = [1, 1]'),
 Text(4.32, 126.84, 'gini = 0.0\nsamples = 1\nvalue = [1, 0]'),
 Text(8.64, 126.84, 'gini = 0.0\nsamples = 1\nvalue = [0, 1]'),
 Text(15.120000000000001, 151.0, 'X[3] <= 125.0\ngini = 0.022\nsamples = 443\nvalue = [5, 438]'),
 Text(12.96, 138.92000000000002, 'gini = 0.0\nsamples = 2\nvalue = [2, 0]'),
 Text(17.28, 138.92000000000002, 'X[3] <= 166.0\ngini = 0.014\nsamples = 441\nvalue = [3, 438]'),
 Text(12.96, 126.84, 'X[0] <= 0.375\ngini = 0.005\nsamples = 438\nvalue = [1, 437]'),
 Text(10.8, 114.75999999999999, 'X[3] <= 152.5\ngini = 0.023\nsamples = 86\nvalue = [1, 85]'),
 Text(8.64, 102.67999999999999, 'gini = 0.0\nsamples = 57\nvalue = [0, 57]'),
 Text(12.96, 102.67999999999999, 'X[3] <= 153.5\ngini = 0.067\nsamples = 29\nvalue = [1, 28]'),
 Text(10.8, 90.6, 'X[7] <= 0.5\ngini = 0.444\nsamples = 3\nvalue = [1, 2]'),
```

RANDOM FOREST

In [37]:
```
from sklearn.ensemble import RandomForestClassifier
Classifier=RandomForestClassifier(n_estimators=20,criterion='entropy',random_state=0)
Classifier.fit(x_train,y_train)
```

Out[37]:
```
RandomForestClassifier(criterion='entropy', n_estimators=20, random_state=0)
```

In [38]:
```
y_pred3=Classifier.predict(x_test)
```

In [39]:
```
y_pred3
```

Out[39]:
```
array([0, 0, 0, ..., 0, 0, 1], dtype=int64)
```

In [40]:
```
y_test
```

Out[40]:
```
array([0, 0, 0, ..., 0, 0, 1], dtype=int64)
```

In [41]:
```
cm=confusion_matrix(y_test,y_pred3)
cm
```

Out[41]:
```
array([[3407,    9],
       [  45, 1039]], dtype=int64)
```

In [42]:
```
accuracy_score(y_test,y_pred3)
```

Out[42]:
```
0.988
```

K NEAREST NEIGHBOUR

In [43]:
```
from sklearn.neighbors import KNeighborsClassifier
classifier=KNeighborsClassifier(n_neighbors=5)
classifier.fit(x_train,y_train)
```

Out[43]:
```
KNeighborsClassifier()
```

In [44]:
```
y_pred4=classifier.predict(x_test)
```

In [45]:
```
y_pred4
```

Out[45]:
```
array([0, 0, 1, ..., 0, 0, 1], dtype=int64)
```

In [46]:
```
y_test
```

Out[46]:
```
array([0, 0, 0, ..., 0, 0, 1], dtype=int64)
```

In [47]:
```
cmk=confusion_matrix(y_test,y_pred4)
```

In [48]:
```
cmk
```

Out[48]:
```
array([[3216,  200],
       [  87,  997]], dtype=int64)
```

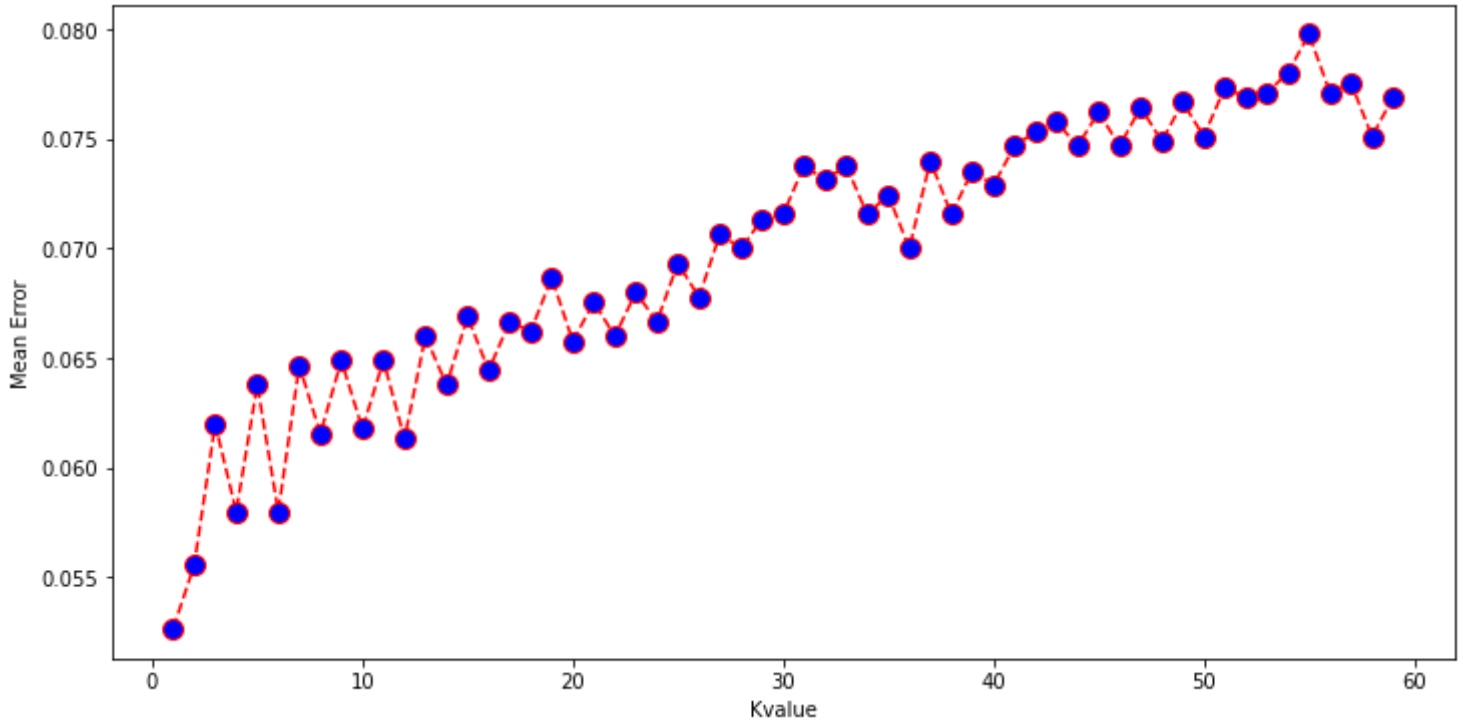
In [49]:
```
accuracy_score(y_test,y_pred4)
```

Out[49]:
```
0.9362222222222222
```

In [50]:
```
error=[]
for i in range(1,60):
    knn=KNeighborsClassifier(n_neighbors=i)
    knn.fit(x_train,y_train)
    Ypred_i=knn.predict(x_test)
    error.append(np.mean(Ypred_i !=y_test))
```

In [51]:
```
plt.figure(figsize=(12,6))
plt.plot(range(1,60),error,color='red',linestyle='dashed',marker='o',markerfacecolor='blue',markersize=10)
plt.xlabel('Kvalue')
plt.ylabel('Mean Error')
```

Out[51]:
```
Text(0, 0.5, 'Mean Error')
```



In [ ]: