In [2]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score,classification_report
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import OrdinalEncoder
```

In [3]:
```
df=pd.read_csv("synthetic_employee_burnout.csv")
df
```

Out[3]:

	Name	Age	Gender	JobRole	Experience	WorkHoursPerWeek	RemoteRatio	Satis
0	Max Ivanov	32	Male	Analyst	3	60	21	
1	Max Wang	40	Female	Engineer	9	47	67	
2	Nina Petrov	33	Female	Engineer	2	44	20	
3	John Ivanov	35	Female	Manager	6	44	70	
4	John Wang	59	Male	Sales	8	38	46	
...	...	...	...	...	...	...	...	...
1995	Leo Brown	41	Female	Manager	4	63	17	
1996	Alex Brown	23	Female	HR	2	39	20	
1997	Nina Wang	31	Female	HR	10	39	4	
1998	Kate Lee	25	Male	HR	0	40	57	
1999	Lily Petrov	49	Female	Engineer	13	65	22	

2000 rows × 10 columns

In [4]:
```
df.drop("Name",axis=1,inplace=True)
```

In [5]:
```
y=df["Burnout"]
y
x=df.drop("Burnout",axis=1)
x
```

Out[5]:

	Age	Gender	JobRole	Experience	WorkHoursPerWeek	RemoteRatio	SatisfactionL
0	32	Male	Analyst	3	60	21	
1	40	Female	Engineer	9	47	67	
2	33	Female	Engineer	2	44	20	
3	35	Female	Manager	6	44	70	
4	59	Male	Sales	8	38	46	
...	...	...	...	...	...	...	
1995	41	Female	Manager	4	63	17	
1996	23	Female	HR	2	39	20	
1997	31	Female	HR	10	39	4	
1998	25	Male	HR	0	40	57	
1999	49	Female	Engineer	13	65	22	

2000 rows × 8 columns

In [6]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.25)
```

In [7]:
```
string_numerical = ["Gender", "JobRole"]
oe = OrdinalEncoder()
x_train[string_numerical] = oe.fit_transform(x_train[string_numerical])
x_test[string_numerical] = oe.transform(x_test[string_numerical])
```

In [8]:
```
model= StandardScaler()
x_train=model.fit_transform(x_train)
x_test=model.transform(x_test)
```

In [9]:
```
model=KNeighborsClassifier(n_neighbors=5)
model.fit(x_train,y_train)
p=model.predict(x_test)
```

In [10]:
```
accuracy_score(y_test,p)*100
```

Out[10]: 96.6

In [11]:
```
print(classification_report(y_test,p))
```

```
              precision    recall  f1-score   support

           0       0.97      0.99      0.98       472
           1       0.76      0.57      0.65        28

    accuracy                           0.97       500
   macro avg       0.87      0.78      0.82       500
weighted avg       0.96      0.97      0.96       500
```

In [ ]: