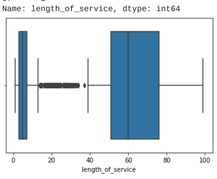
In [1]:
```
import pandas as pd# we import pandas to handle the file and its save our lots of time, it handle large data set efficeintly.
import numpy as np #for creating arrays
import seaborn as sns #for data visualization
import matplotlib.pyplot as plt # for data visualization

df1 = pd.read_csv(r'C:\Users\PC-chetan\Desktop\train.csv') # trian data

df2 = pd.read_csv(r'C:\Users\PC-chetan\Desktop\test.csv') # test data

df1.education.fillna("Bachelor's", inplace=True)

df2.education.fillna("Bachelor's", inplace=True)

df1.previous_year_rating.fillna('3.0',inplace=True)

df2.previous_year_rating.fillna('3.0',inplace=True)

from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()

df1.drop(columns=['employee_id','region','recruitment_channel'], inplace=True)

df2.drop(columns=['employee_id','region','recruitment_channel'], inplace=True)

#lets encode the education in their degree of importance
df1['education'] = df1['education'].replace(("Master's & above", "Bachelor's", "Below Secondary"),
                                            (3, 2, 1))
df2['education'] = df2['education'].replace(("Master's & above", "Bachelor's", "Below Secondary"), (3, 2, 1))

df1.gender = le.fit_transform(df1.gender)

df1.department = le.fit_transform(df1.department)

df2.department = le.transform(df2.department)


df2['gender'] = df2['gender'].replace(("m", "f"),(1,0))
```

In [2]:
```
df1.select_dtypes('number').head()

df2.select_dtypes('number').head()

sns.boxplot(data=df1,x=df1['avg_training_score'])

df1.shape

Q1=df1['avg_training_score'].quantile(0.25)
Q3=df1['avg_training_score'].quantile(0.75)
IQR=Q3-Q1
print(Q1)
print(Q3)
print(IQR)
min_1 = Q1-(1.5)*IQR
max_1 = Q3+(1.5)*IQR
print(min_1)
print( max_1)


df1['avg_training_score'].unique()

df1 = df1[df1['avg_training_score']< max_1]

df1.shape


sns.boxplot(data=df1,x=df1['length_of_service'])

Q2=df1['length_of_service'].quantile(0.25)
Q4=df1['length_of_service'].quantile(0.75)
IQRt=Q4-Q2
print(Q2)
print(Q4)
print(IQRt)
min_2 = Q2-1.5*IQRt
max_2 = Q4+1.5*IQRt
print(max_2,min_2)

df1['length_of_service'].unique()

df1 = df1[df1['length_of_service'] > 13]

df1.shape

df1['length_of_service'].value_counts()
```

Out[2]:
```
51.0
76.0
25.0
13.5
113.5
3.0
7.0
4.0
13.0 -3.0
15    593
14    549
16    548
17    432
18    392
19    329
20    128
21     78
24     70
23     65
22     61
25     51
26     41
27     36
29     30
28     30
31     20
30     12
32     10
33      9
34      4
37      1
Name: length_of_service, dtype: int64
```



In [3]:
```
df1
```

Out[3]:

	department	education	gender	no_of_trainings	age	previous_year_rating	length_of_service	awards_won?	avg_training_score	is_promoted
13	8	3	1	2	39	3.0	16	0	80	0
42	2	2	1	1	59	4.0	26	0	52	0
60	7	3	1	1	50	4.0	17	0	47	1
74	7	2	1	1	50	3.0	14	0	52	0
99	1	3	1	1	60	5.0	17	0	59	0
...	...	...	...	...	...	...	...	...	...	...
54691	0	3	1	1	47	5.0	19	0	86	0
54695	4	2	0	2	52	5.0	18	0	56	1
54697	7	2	1	1	47	5.0	15	0	50	0
54754	8	2	0	1	42	3.0	14	0	79	0
54803	8	2	1	1	48	3.0	17	0	78	0

3489 rows × 10 columns

In [4]:
```
y = df1['is_promoted']
x = df1.drop(columns=['is_promoted'])

#X_train, X_test, y_train, y_test =train_test_split(x,test_size=.3)

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=41)
```

In [6]:
```
from sklearn.tree import DecisionTreeClassifier
dtree = DecisionTreeClassifier()

dtree.fit(x_train,y_train)
```

Out[6]:
```
DecisionTreeClassifier()
```

In [7]:
```
dtree.score(x_test,y_test)
```

Out[7]:
```
0.8911174785100286
```

In [9]:
```
from sklearn.metrics import confusion_matrix
```

In [11]:
```
predict = dtree.predict(df2)

predict

pd.set_option('display.max_rows', 500000) # for getting the max veiw of raws
pd.set_option('display.max_column', 500000) # for getting the max veiw of columns

result = pd.DataFrame(predict, columns=['is_promoted'])
```

In [12]:
```
predict
```

Out[12]:
```
array([0, 0, 0, ..., 0, 0, 1], dtype=int64)
```

In [14]:
```
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier(n_estimators = 51)

rf.fit(x_train,y_train)
```

Out[14]:
```
RandomForestClassifier(n_estimators=51)
```

In [16]:
```
rf.predict(x_test)
```

Out[16]:
```
array([0, 0, 0, ..., 0, 0, 0], dtype=int64)
```

In [17]:
```
rf.score(x_test,y_test)
```

Out[17]:
```
0.938872970391595
```

In [18]:
```
predict2 = rf.predict(df2)
```

In [19]:
```
predict2
```

Out[19]:
```
array([0, 0, 0, ..., 0, 0, 1], dtype=int64)
```

In [ ]:
```
# result improved by doing outleir process
```