In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as pyplt
%matplotlib inline
```

In [2]:

```
customer_churn = pd.read_csv("customer_churn.csv")
```

In [3]:

```
customer_churn.head()
```

Out[3]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	Inte
0	7590-VHVEG	Female	0	Yes	No	1	No	No phone service	
1	5575-GNVDE	Male	0	No	No	34	Yes	No	
2	3668-QPYBK	Male	0	No	No	2	Yes	No	
3	7795-CFOCW	Male	0	No	No	45	No	No phone service	
4	9237-HQITU	Female	0	No	No	2	Yes	No	

5 rows × 21 columns

In [4]:

```
customer_5 = customer_churn.iloc[:, 4]
customer_5.head()
```

Out[4]:

```
0    No
1    No
2    No
3    No
4    No
Name: Dependents, dtype: object
```

In [5]:

```
customer_15 = customer_churn.iloc[:, 14]
customer_15.head()
```

Out[5]:

```
0    No
1    No
2    No
3    No
4    No
Name: StreamingMovies, dtype: object
```

In [6]:

```
senior_male_electronic =customer_churn[(customer_churn['gender']=='Male') & (customer_
senior_male_electronic.head(10)
```

Out[6]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines
20	8779-QRDMV	Male	1	No	No	1	No	No phone service
55	1658-BYGOY	Male	1	No	No	18	Yes	Yes
57	5067-XJQFU	Male	1	Yes	Yes	66	Yes	Yes
78	0191-ZHSKZ	Male	1	No	No	30	Yes	No
91	2424-WVHPL	Male	1	No	No	1	Yes	No
129	2639-UGMAZ	Male	1	No	No	71	No	No phone service
168	3445-HXXGF	Male	1	Yes	No	58	No	No phone service
214	2504-DSHIH	Male	1	Yes	No	23	Yes	Yes
245	0221-WMXNQ	Male	1	No	No	4	Yes	No
247	9947-OTFQU	Male	1	No	No	15	Yes	No

10 rows × 21 columns

In [7]:

```
customer_total_tenure = customer_churn[((customer_churn['tenure']>70) | (customer_chur
customer_total_tenure.head(10)
```

Out[7]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	In
8	7892-POOKP	Female	0	Yes	No	28	Yes	Yes	
12	8091-TTVAX	Male	0	Yes	No	58	Yes	Yes	
13	0280-XJGEX	Male	0	No	No	49	Yes	Yes	
14	5129-JLPIS	Male	0	No	No	25	Yes	No	
15	3655-SNQYZ	Female	0	Yes	Yes	69	Yes	Yes	
17	9959-WOFKT	Male	0	No	Yes	71	Yes	Yes	
28	5248-YGIJN	Male	0	Yes	No	72	Yes	Yes	
30	3841-NFECX	Female	1	Yes	No	71	Yes	Yes	
35	6234-RAAPL	Female	0	Yes	Yes	72	Yes	Yes	
38	5380-WJKOV	Male	0	No	No	34	Yes	Yes	

10 rows × 21 columns

```
In [8]: two_mail_yes= customer_churn[((customer_churn['Contract']=='Two year')
                              & (customer_churn['Churn']=='Yes') &
                              (customer_churn['PaymentMethod']=='Mailed check'))]
        two_mail_yes.head(10)
```

Out[8]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines
268	6323-AYBRX	Male	0	No	No	59	Yes	No
5947	7951-QKZPL	Female	0	Yes	Yes	33	Yes	Yes
6680	9412-ARGBX	Female	0	No	Yes	48	Yes	No

3 rows × 21 columns

```
In [9]: customer_333= customer_churn.sample(n=333)
        customer_333.head()
```

Out[9]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines
657	7838-LAZFO	Male	0	Yes	No	45	Yes	No
938	2692-AQCPF	Female	0	Yes	No	65	Yes	Yes
6309	2169-RRLFW	Female	0	Yes	No	71	Yes	Yes
1936	2239-CGBUZ	Female	0	Yes	No	51	Yes	No
2645	8562-GHPPI	Female	0	No	No	1	Yes	No

5 rows × 21 columns

In [10]:

```
customer_churn['Churn'].value_counts().keys()
```

Out[10]:

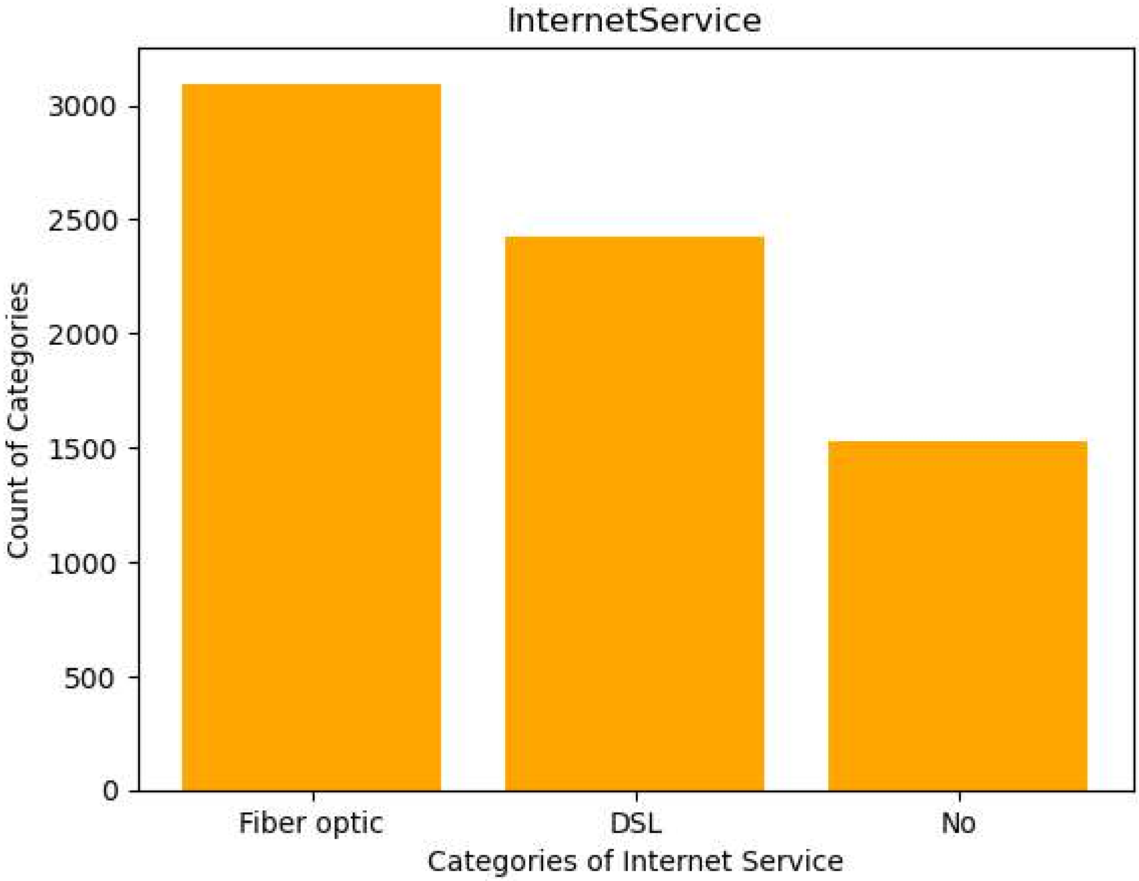
```
Index(['No', 'Yes'], dtype='object')
```

In [11]:

```
x= customer_churn['InternetService'].value_counts().keys()
y= customer_churn['InternetService'].value_counts()
pyplt.bar(x,y,color='orange')
pyplt.xlabel('Categories of Internet Service')
pyplt.ylabel('Count of Categories')
pyplt.title('InternetService')
```

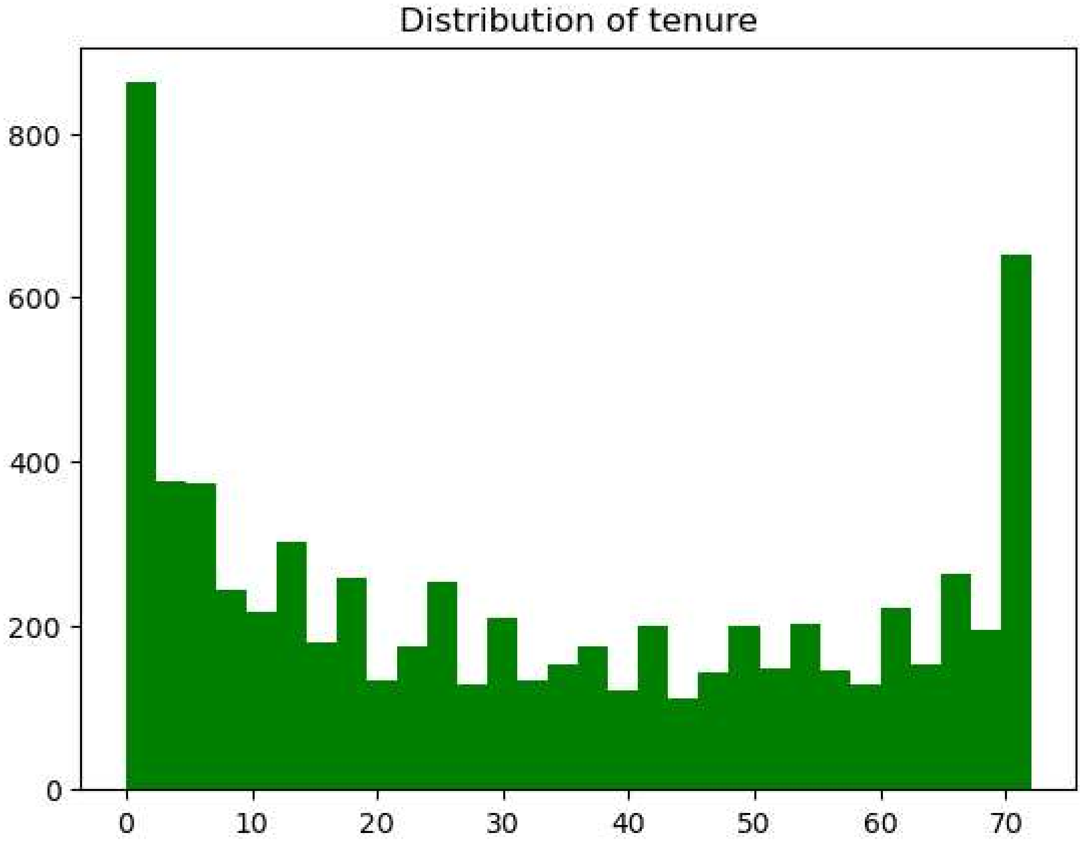
Out[11]:

```
Text(0.5, 1.0, 'InternetService')
```



```
In [12]: pyplt.hist(customer_churn['tenure'],color='green', bins=30)
         pyplt.title('Distribution of tenure')
```

```
Out[12]: Text(0.5, 1.0, 'Distribution of tenure')
```

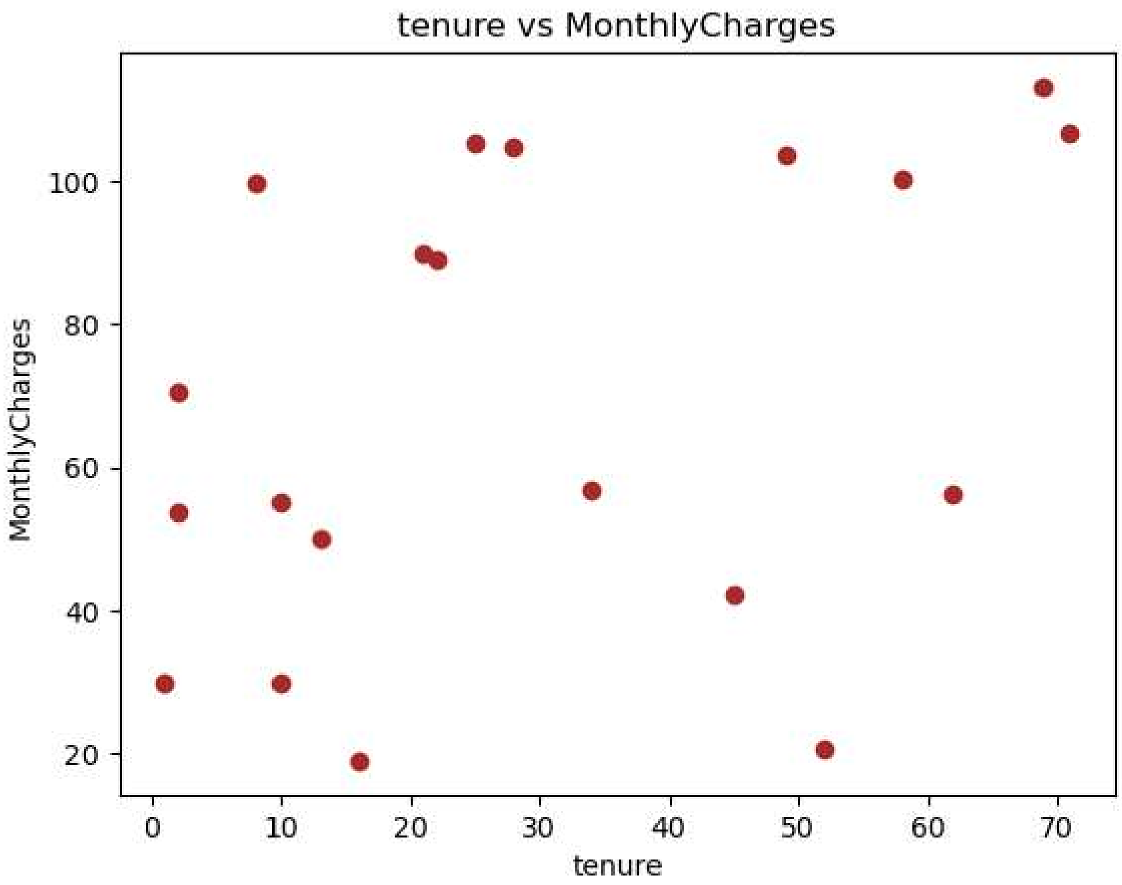


In [13]:

```
pyplt.scatter(x=customer_churn['tenure'].head(20), y=customer_churn['MonthlyCharges'].
pyplt.xlabel('tenure')
pyplt.ylabel('MonthlyCharges')
pyplt.title('tenure vs MonthlyCharges')
```

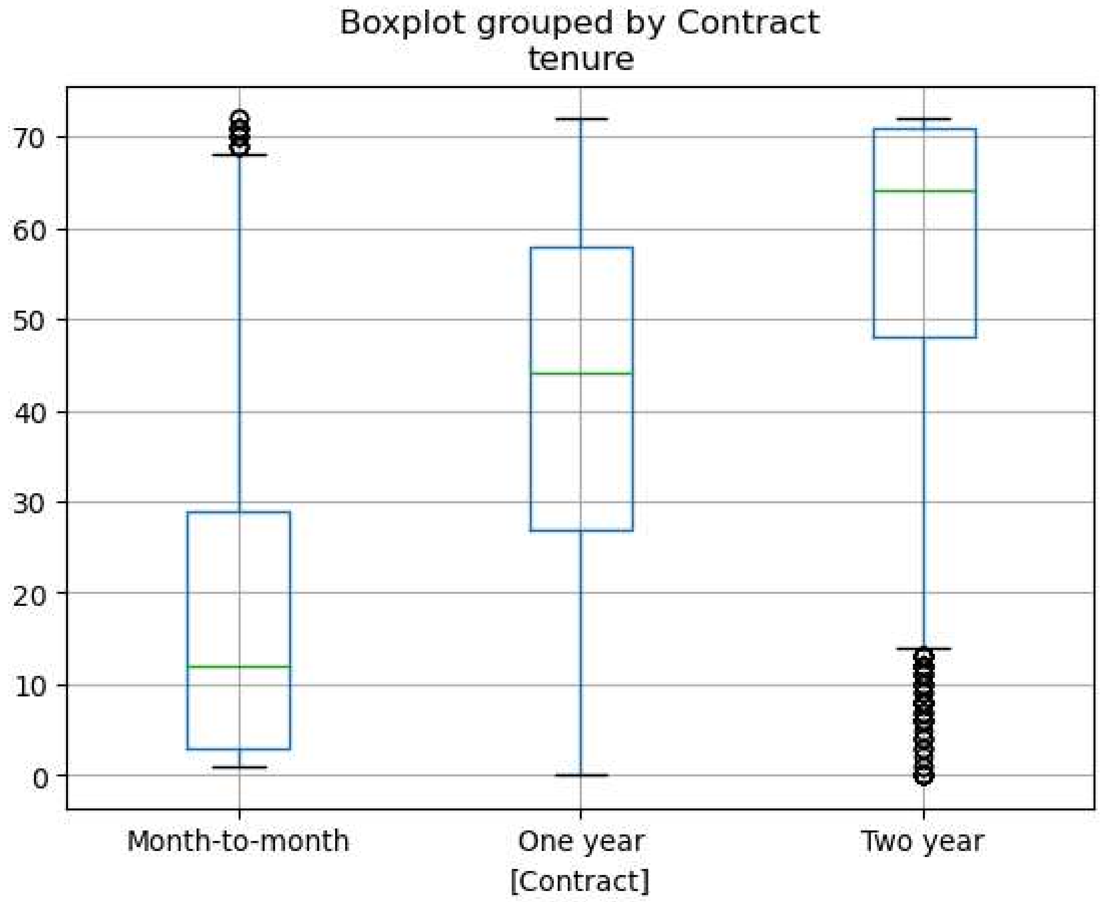
Out[13]:

```
Text(0.5, 1.0, 'tenure vs MonthlyCharges')
```



```
In [14]: customer_churn.boxplot(column='tenure', by=['Contract'])
```

```
Out[14]: <Axes: title={'center': 'tenure'}, xlabel='[Contract]'>
```

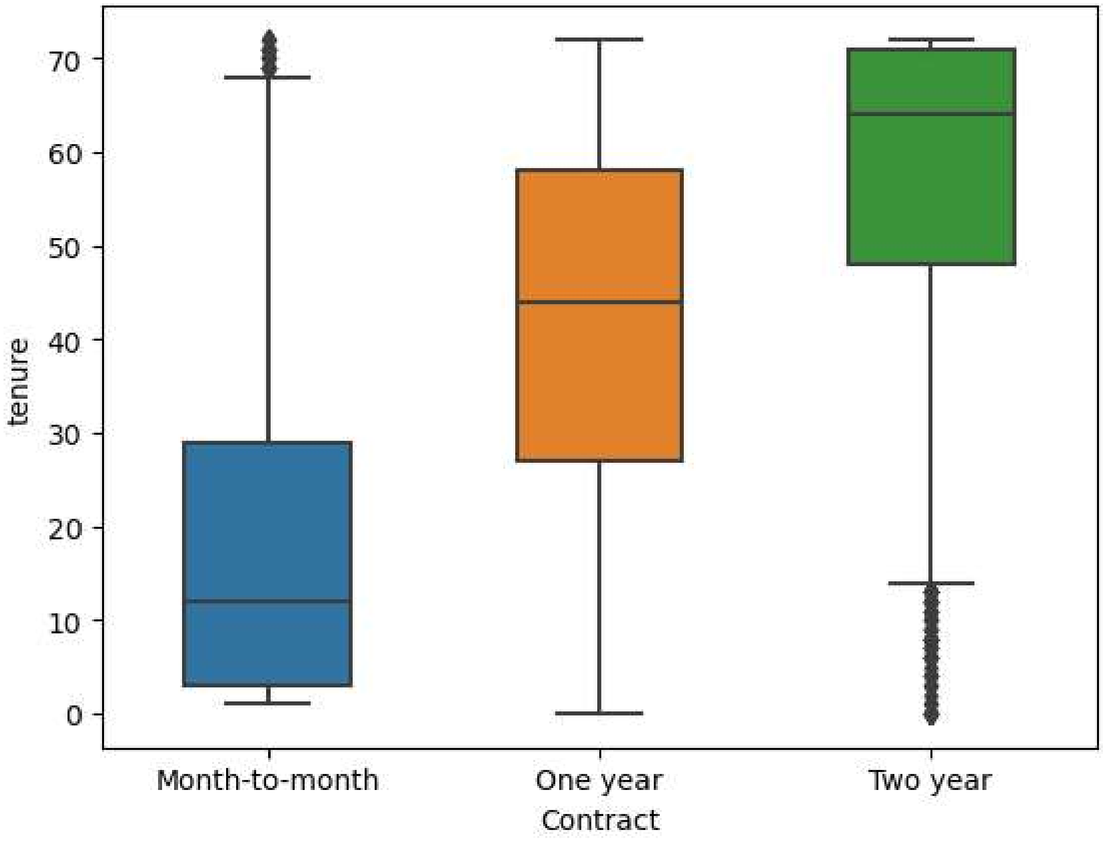


In [15]:

```
import seaborn as sns
sns.boxplot(x='Contract', y='tenure', data =customer_churn, width=0.5)
```

Out[15]:

```
<Axes: xlabel='Contract', ylabel='tenure'>
```



```
In [16]: from sklearn.model_selection import train_test_split
         x=pd.DataFrame(customer_churn['tenure'])
         y=customer_churn['MonthlyCharges']
```

```
In [17]: x
```

Out[17]:

	tenure
0	1
1	34
2	2
3	45
4	2
...	...
7038	24
7039	72
7040	11
7041	4
7042	66

7043 rows × 1 columns

```
In [18]: y
```

Out[18]:

```
0        29.85
1        56.95
2        53.85
3        42.30
4        70.70
         ...  
7038     84.80
7039    103.20
7040     29.60
7041     74.40
7042    105.65
Name: MonthlyCharges, Length: 7043, dtype: float64
```

In [19]:

```
x_train, x_test, y_train, y_test=train_test_split(x, y, test_size=0.3, random_state= 0
```

In [20]:

```
x_train
```

Out[20]:

	tenure
3580	9
2364	14
6813	64
789	72
561	3
...	...
4931	15
3264	10
1653	58
2607	1
2732	4

4930 rows × 1 columns

In [21]:

```
x_test
```

Out[21]:

	tenure
2200	19
4627	60
3225	13
2828	1
3768	55
...	...
4448	30
1231	20
3304	69
4805	52
5843	35

2113 rows × 1 columns

In [22]:

```
y_train
```

Out[22]:

```
3580     72.90
2364     82.65
6813     47.85
789      69.65
561      23.60
         ...  
4931    103.45
3264     91.10
1653     20.75
2607     69.75
2732     20.40
Name: MonthlyCharges, Length: 4930, dtype: float64
```

In [23]:

```
y_test
```

Out[23]:

```
2200     58.20
4627    116.60
3225     71.95
2828     20.45
3768     77.75
         ...  
4448     99.70
1231     64.40
3304    109.95
4805     24.55
5843     81.60
Name: MonthlyCharges, Length: 2113, dtype: float64
```

In [24]:

```
from sklearn.linear_model import LinearRegression
LR= LinearRegression()
LR.fit(x_train, y_train)
```

Out[24]:

```
▾ LinearRegression
LinearRegression()
```

In [25]:

```
y_predict= LR.predict(x_test)
```

In [26]:

```
y_predict
```

Out[26]:

```
array([60.95089608, 72.98096699, 59.1903979 , ..., 75.62171426,
       70.63363608, 65.6455579 ])
```

In [27]:

```
y_test
```

Out[27]:

```
2200     58.20
4627    116.60
3225     71.95
2828     20.45
3768     77.75
         ...
4448     99.70
1231     64.40
3304    109.95
4805     24.55
5843     81.60
Name: MonthlyCharges, Length: 2113, dtype: float64
```

In [28]:

```
from sklearn.metrics import mean_squared_error
mse= mean_squared_error(y_predict, y_test)
rmse=np.sqrt(mse)
rmse
```

Out[28]:

```
29.394584027273893
```

In [29]:

```
#logistic reg
x=pd.DataFrame(customer_churn['MonthlyCharges'])
y=customer_churn['Churn']
```

In [30]:

```
x
```

Out[30]:

	MonthlyCharges
0	29.85
1	56.95
2	53.85
3	42.30
4	70.70
...	...
7038	84.80
7039	103.20
7040	29.60
7041	74.40
7042	105.65

7043 rows × 1 columns

In [31]:

```
y
```

Out[31]:

```
0        No
1        No
2       Yes
3        No
4       Yes
       ... 
7038     No
7039     No
7040     No
7041    Yes
7042     No
Name: Churn, Length: 7043, dtype: object
```

In [32]:

```
x_train, x_test, y_train, y_test=train_test_split(x, y, test_size=0.35, random_state=
```

In [33]:

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix, accuracy_score
LoR= LogisticRegression()
LoR.fit(x_train, y_train)
```

Out[33]:

```
▾ LogisticRegression
LogisticRegression()
```

In [34]:

```
y_predict=LoR.predict(x_test)
y_predict
```

Out[34]:

```
array(['No', 'No', 'No', ..., 'No', 'No', 'No'], dtype=object)
```

In [35]:

```
y_test
```

```
Out[35]: 2200     No
4627     No
3225     No
2828     No
3768     No
        ...
5753     No
4109    Yes
4106    Yes
2760     No
2534     No
Name: Churn, Length: 2466, dtype: object
```

```
In [36]: y_predict[[200]]
```

```
Out[36]: array(['No'], dtype=object)
```

```
In [37]: confusion_matrix(y_predict, y_test), accuracy_score(y_predict, y_test)
```

```
Out[37]: (array([[1815,  651],
        [   0,    0]], dtype=int64),
 0.7360097323600974)
```

```
In [38]: x=pd.DataFrame(customer_churn.loc[:,['MonthlyCharges','tenure']])
y=customer_churn['Churn']
```

```
In [39]: x_train, x_test, y_train, y_test=train_test_split(x, y, test_size=0.2, random_state= 0
x_train
```

Out[39]:

	MonthlyCharges	tenure
2920	85.10	72
2966	46.35	14
6099	24.70	71
5482	73.90	33
2012	98.75	47
...	...	...
4931	103.45	15
3264	91.10	10
1653	20.75	58
2607	69.75	1
2732	20.40	4

5634 rows × 2 columns

```
In [40]: x_test
```

Out[40]:

	MonthlyCharges	tenure
2200	58.20	19
4627	116.60	60
3225	71.95	13
2828	20.45	1
3768	77.75	55
...	...	...
2631	99.25	7
5333	88.35	13
6972	111.95	56
4598	56.25	18
3065	45.80	1

1409 rows × 2 columns

In [41]:

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix, accuracy_score
LoR= LogisticRegression()
LoR.fit(x_train, y_train)
```

Out[41]:

```
▾ LogisticRegression
LogisticRegression()
```

In [42]:

```
y_predict=LoR.predict(x_test)
y_predict
```

Out[42]:

```
array(['No', 'No', 'No', ..., 'No', 'No', 'No'], dtype=object)
```

In [43]:

```
y_test
```

Out[43]:

```
2200     No
4627     No
3225     No
2828     No
3768     No
        ... 
2631    Yes
5333    Yes
6972    Yes
4598     No
3065     No
Name: Churn, Length: 1409, dtype: object
```

In [44]:

```
confusion_matrix(y_predict, y_test), accuracy_score(y_predict, y_test)
```

Out[44]:

```
(array([[934, 212],
        [107, 156]], dtype=int64),
 0.7735982966643009)
```

In [45]:

```
#decission tree
x=pd.DataFrame(customer_churn['tenure'])
y=customer_churn['Churn']
```

In [46]:

```
x_train, x_test, y_train, y_test=train_test_split(x, y, test_size=0.2, random_state= 0
```

In [47]:

```
from sklearn.tree import DecisionTreeClassifier
DecisionTree= DecisionTreeClassifier()
DecisionTree.fit(x_train, y_train)
```

Out[47]:

```
▾ DecisionTreeClassifier
DecisionTreeClassifier()
```

In [48]:

```
y_predict=DecisionTree.predict(x_test)
y_predict
```

Out[48]:

```
array(['No', 'No', 'No', ..., 'No', 'No', 'Yes'], dtype=object)
```

In [49]:

```
y_test
```

Out[49]:

```
2200     No
4627     No
3225     No
2828     No
3768     No
       ... 
2631    Yes
5333    Yes
6972    Yes
4598     No
3065     No
Name: Churn, Length: 1409, dtype: object
```

In [50]:

```
from sklearn.metrics import confusion_matrix, accuracy_score
confusion_matrix(y_predict, y_test), accuracy_score(y_predict, y_test)
```

Out[50]:

```
(array([[965, 281],
        [ 76,  87]], dtype=int64),
 0.7466288147622427)
```

In [51]:

```
#random forest
x=pd.DataFrame(customer_churn.loc[:,['MonthlyCharges','tenure']])
y=customer_churn['Churn']
```

In [52]:

```
x_train, x_test, y_train, y_test=train_test_split(x, y, test_size=0.3, random_state= 0
```

In [53]:

```
from sklearn.ensemble import RandomForestClassifier
RFC=RandomForestClassifier(n_estimators=100)
RFC.fit(x_train, y_train)
```

Out[53]:

```
▾ RandomForestClassifier
RandomForestClassifier()
```

In [54]:

```
y_predict=RFC.predict(x_test)
y_predict
```

Out[54]:

```
array(['No', 'No', 'No', ..., 'No', 'No', 'No'], dtype=object)
```

In [55]:

```
from sklearn.metrics import confusion_matrix, accuracy_score
confusion_matrix(y_predict, y_test), accuracy_score(y_predict, y_test)
```

Out[55]:

```
(array([[1346,  321],
        [ 214,  232]], dtype=int64),
 0.7468054898248935)
```

In [ ]:

```
```