In [1]:

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

In [2]:

```
# Reading the file
customer_churn = pd.read_csv("../input/customer-churn-data-set/customer_churn (3).csv")
```

In [3]:

```
customer_churn.head()
```

Out[3]:

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

5 rows × 21 columns

In [4]:

```
#a.     Extract the 5th column & store it in ‘customer_5’
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]:

```
# b.    Extract the 15th column & store it in ‘customer_15’
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]:

```
#c.Extract all the male senior citizens whose Payment Method is Electronic check & store the result in ‘senior_
senior_male_electronic =customer_churn[(customer_churn['gender']=='Male') & (customer_churn['SeniorCitizen']==1
senior_male_electronic.head(10)
```

Out[6]:

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

10 rows × 21 columns

In [7]:

```
# d.Extract all those customers whose tenure is greater than 70 months or their Monthly charges is more than 10
```

```
customer_total_tenure = customer_churn[((customer_churn['tenure']>70) | (customer_churn['MonthlyCharges']>100))
customer_total_tenure.head(10)
```

Out[7]:

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

10 rows × 21 columns

In [8]:

```
#e.Extract all the customers whose Contract is of two years,
# payment method is Mailed check & the value of Churn is ‘Yes’ &
# store the result in ‘two_mail_yes’

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	InternetService	OnlineSecurity	...	Devic
268	6323-AYBRX	Male	0	No	No	59	Yes	No	No	No internet service	...	
5947	7951-QKZPL	Female	0	Yes	Yes	33	Yes	Yes	No	No internet service	...	
6680	9412-ARGBX	Female	0	No	Yes	48	Yes	No	Fiber optic	No	...	

3 rows × 21 columns

In [9]:

```
#f.Extract 333 random records from the customer_churn dataframe & store the result in ‘customer_333’
customer_333= customer_churn.sample(n=333)
customer_333.head()
```

Out[9]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...	Devic
3831	6946-LMSQS	Male	1	Yes	No	25	Yes	Yes	Fiber optic	Yes	...	
5880	3538-WZPHD	Male	0	No	No	3	No	No phone service	DSL	No	...	
4848	5380-AFSSK	Female	0	No	No	5	Yes	Yes	Fiber optic	No	...	
1752	7801-CEDNV	Male	0	Yes	No	27	Yes	No	DSL	Yes	...	
4892	8875-AKBYH	Male	1	No	No	20	Yes	Yes	Fiber optic	No	...	

5 rows × 21 columns

In [10]:

```
#.Get the count of different levels from the ‘Churn’ column
customer_churn['Churn'].value_counts().keys()
```

Out[10]:

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

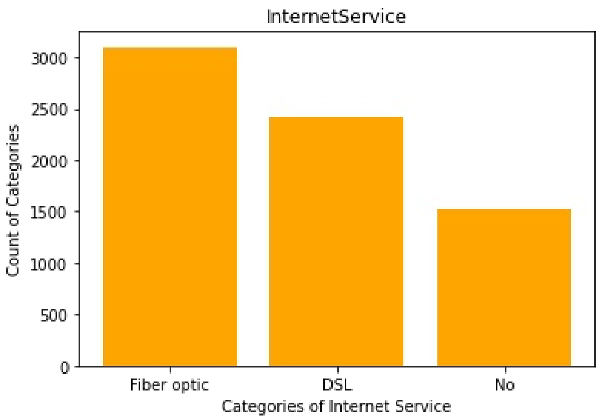
In [11]:

```
#Build a bar-plot for the ’InternetService’ column:
```

```
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')
```



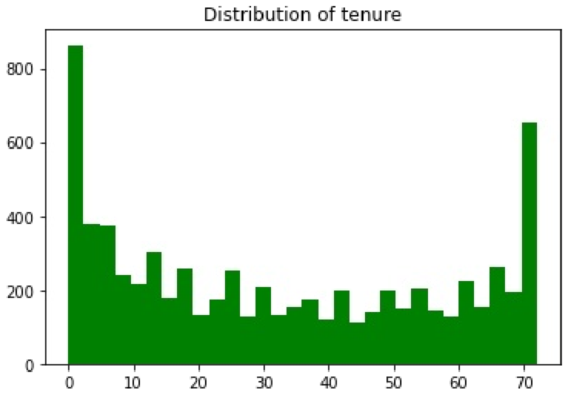
In [12]:

```
#histogram for the ‘tenure’ column:
#b.Build a histogram for the ‘tenure’ column:
#i.Set the number of bins to be 30
#ii.Set the color of the bins  to be ‘green’
#iii.Assign the title ‘Distribution of tenure’
#histogram

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

Out[12]:

```
Text(0.5, 1.0, 'Distribution of tenure')
```

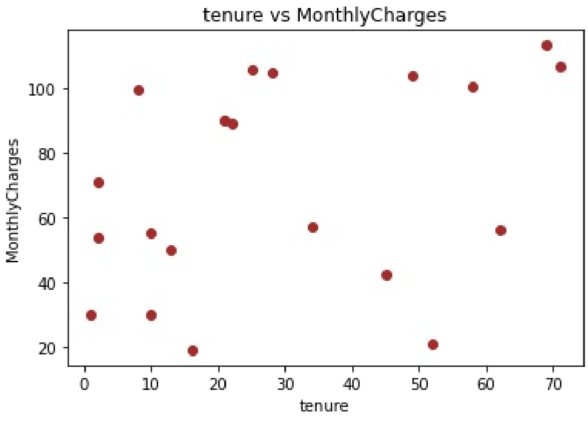


In [13]:

```
#c.     Build a scatter-plot between ‘MonthlyCharges’ & ‘tenure’. Map ‘MonthlyCharges’ to the y-axis & ‘tenure’
pyplt.scatter(x=customer_churn['tenure'].head(20), y=customer_churn['MonthlyCharges'].head(20), color ='Brown')
pyplt.xlabel('tenure')
pyplt.ylabel('MonthlyCharges')
pyplt.title('tenure vs MonthlyCharges')
```

Out[13]:

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

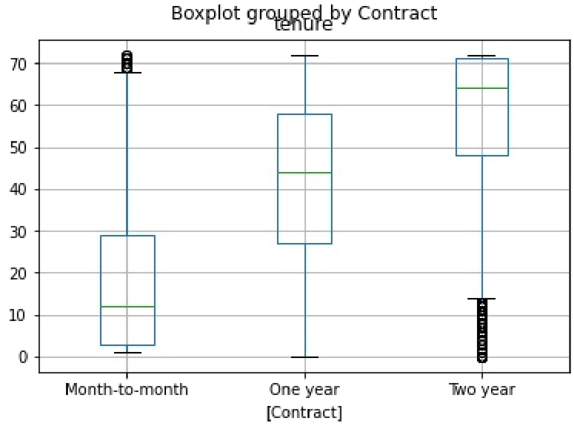


In [14]:

```
#Build a box-plot between ‘tenure’ & ‘Contract’. Map ‘tenure’ on the y-axis & ‘Contract’ on the x-axis.
customer_churn.boxplot(column='tenure', by=['Contract'])
```

Out[14]:

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

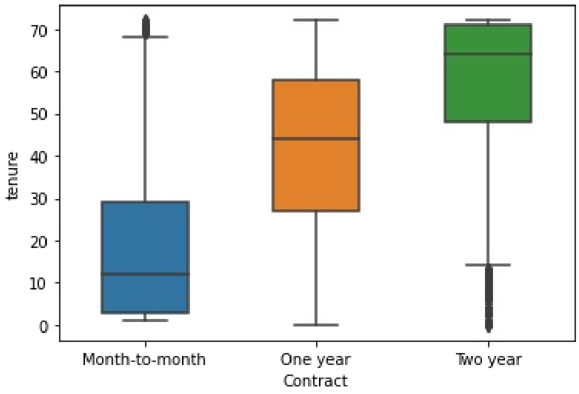


In [15]:

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

Out[15]:

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



In [16]:

```
#Linear Regression:
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 [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()
```

In [25]:
```
# Predicting the values
```

```
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]:

```
#so much of error, if it is close to 1 that means#
```

In [30]:

```
#Logistic Regression:
x=pd.DataFrame(customer_churn['MonthlyCharges'])
y=customer_churn['Churn']
```

In [31]:

```
x
```

Out[31]:

	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 [32]:

```
y
```

Out[32]:

```
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 [33]:

```
x_train, x_test, y_train, y_test=train_test_split(x, y, test_size=0.35, random_state= 0)
```

In [34]:

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

Out[34]:

```
LogisticRegression()
```

In [35]:

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

```
y_predict
```

Out[35]:

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

In [36]:

```
y_test
```

Out[36]:

```
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 [37]:

```
y_predict[[200]]
```

Out[37]:

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

In [38]:

```
confusion_matrix(y_predict, y_test), accuracy_score(y_predict, y_test)
# TP FP
#FN TN
```

Out[38]:

```
(array([[1815,  651],
        [   0,    0]]),
 0.7360097323600974)
```

In [39]:

```
# Multiple Logistic Regression

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

In [40]:

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

Out[40]:

	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 [41]:

```
x_test
```

Out[41]:

	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 [42]:

```
from sklearn.linear_model import LogisticRegression
```

In [42]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix, accuracy_score
LoR= LogisticRegression()
LoR.fit(x_train, y_train)
```

Out[42]:
```
LogisticRegression()
```

In [43]:
```
y_predict=LoR.predict(x_test)
y_predict
```

Out[43]:
```
array(['No', 'No', 'No', ..., 'No', 'No', 'No'], dtype=object)
```

In [44]:
```
y_test
```

Out[44]:
```
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 [45]:
```
confusion_matrix(y_predict, y_test), accuracy_score(y_predict, y_test)
```

Out[45]:
```
(array([[934, 212],
        [107, 156]]),
 0.7735982966643009)
```

In [46]:
```
# Decision Tree

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

In [47]:
```
x_train, x_test, y_train, y_test=train_test_split(x, y, test_size=0.2, random_state= 0)
```

In [48]:
```
from sklearn.tree import DecisionTreeClassifier
DecisionTree= DecisionTreeClassifier()
DecisionTree.fit(x_train, y_train)
```

Out[48]:
```
DecisionTreeClassifier()
```

In [49]:
```
y_predict=DecisionTree.predict(x_test)
y_predict
```

Out[49]:
```
array(['No', 'No', 'No', ..., 'No', 'No', 'Yes'], dtype=object)
```

In [50]:
```
y_test
```

Out[50]:
```
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 [51]:
```
from sklearn.metrics import confusion_matrix, accuracy_score
confusion_matrix(y_predict, y_test), accuracy_score(y_predict, y_test)
```

Out[51]:
```
(array([[965, 281],
        [ 76,  87]]),
 0.7466288147622427)
```

In [52]:
```
# Random Forest
x=pd.DataFrame(customer_churn.loc[:,['MonthlyCharges','tenure']])
y=customer_churn['Churn']
```

In [53]:
```
x_train, x_test, y_train, y_test=train_test_split(x, y, test_size=0.3, random_state= 0)
```

In [54]:
```
from sklearn.ensemble import RandomForestClassifier
RFC=RandomForestClassifier(n_estimators=100)
RFC.fit(x_train, y_train)
```

Out[54]:
```
RandomForestClassifier()
```

In [55]:
```
y_predict=RFC.predict(x_test)
```

```
y_predict
```

Out[55]:

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

In [56]:

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

Out[56]:

```
(array([[1345,  332],
        [ 215,  221]]),
 0.7411263606247042)
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js