Problem Statement

You are the data scientist at the telicom company "Neo" whose customer are churing out to its competitors. You hav to analyse the data of your company and find insight and stop your customer fro churing out to telicom company.

Task to be done

Data manipulation, a.Extract the 5th column & store in 'Customer_5'

```
b.Etract the 15th column & store in 'customer_15

c.Extract all the male seniour citizens whose payment method is electronic check & staore the result
in 'Senior male electronic'

d.Extract all those customers whose tenure is greater than 70 months or their monthly charges is more
than 100$ & store them in 'Customer total tenure'

e.Extract all the customer whose contact is for 2 years,payment method is mailed check & the value of
churn is 'Yes' & store the result in 'two mail yes'

f.Extract 333 random record from customer churn dataframe & store the result in customer_333

g.get the count of diffrent levels from the churn column
```

Data visualization

a.Build a bar plot for the 'internetService' column:

```
1.set x_axis labeled to 'categories of internet service'

2.set y-axis to 'count of categories'

3.set the title of the plot as 'Distrubution of Internet service'
```

```
4.set the color of the bar 'Orange'
```

b.Build a Histogram for 'tenure' column:

```
1.set the no of bins to 30

2.set the color of bin to 'Green'

3.Assign the title as 'Distrubution of tenure'
```

c.Built a scatter plot between 'MonthlyCharges' & 'Tenure'.map 'MonthlyCharges' to y-Axis & 'Tenure' to the 'x-Axis'

```
1.assign the points of color as brown

2.set the axis label to 'tenure customer'

3.set the y-axis to 'monthly charges of customer'

4.set the title to 'tenure vs monthly Charges'
```

d.plot an Box plot between tenure and contract.map 'tenure' on the y-axis and contract on the x-axis

In [1]:
```
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
```

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	InternetService	OnlineSecurity	...	DeviceProtectio

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

5 rows × 21 columns

In [4]:
```
#Extracting the 5th column
c_5=customer_churn.iloc[:,4]
c_5.head()
```

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

In [5]:
```
#Extracting the 15 column
c_15=customer_churn.iloc[:,14]
c_15.head()
```

Out[5]:
```
0    No
1    No
2    No
3    No
4    No
Name: StreamingMovies, dtype: object
```

In [6]:
```
#Extracting 'c'-Extract all the male seniour citizens whose payment method is electronic check & staore the result ir
c_random=customer_churn[(customer_churn['gender']=='Male') & (customer_churn['SeniorCitizen']==1) & (customer_churn[
```

In [7]:
```
c_random.head()
```

Out[7]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...	DeviceProtecti
20	8779-QRDMV	Male	1	No	No	1	No	No phone service	DSL	No	...	Y
55	1658-BYGOY	Male	1	No	No	18	Yes	Yes	Fiber optic	No	...	I
57	5067-XJQFU	Male	1	Yes	Yes	66	Yes	Yes	Fiber optic	No	...	Y
78	0191-ZHSKZ	Male	1	No	No	30	Yes	No	DSL	Yes	...	I
91	2424-WVHPL	Male	1	No	No	1	Yes	No	Fiber optic	No	...	I

5 rows × 21 columns

In [8]:
```
#Extracting 'd'-Extract all those customers whose tenure is greater than 70 months or their monthly charges is more t
c_random=customer_churn[(customer_churn['tenure']>70) | (customer_churn['MonthlyCharges']>100)]
```

In [9]:
```
c_random.head()
```

Out[9]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...	DeviceProtecti
8	7892-POOKP	Female	0	Yes	No	28	Yes	Yes	Fiber optic	No	...	Y
12	8091-TTVAX	Male	0	Yes	No	58	Yes	Yes	Fiber optic	No	...	Y
13	0280-XJGEX	Male	0	No	No	49	Yes	Yes	Fiber optic	No	...	Y
14	5129-JLPIS	Male	0	No	No	25	Yes	No	Fiber optic	Yes	...	Y
15	3655-SNQYZ	Female	0	Yes	Yes	69	Yes	Yes	Fiber optic	Yes	...	Y

5 rows × 21 columns

In [10]:
```
#Extracting 'e'-Extract all the customer whose contact is for 2 years,payment method is mailed check & the value of c
c_random=customer_churn[(customer_churn['Contract']=='Two year') & (customer_churn['PaymentMethod']=='Mailed check')
```

In [11]:
```
c_random
```

Out[11]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...	DeviceProte
268	6323-AYBRX	Male	0	No	No	59	Yes	No	No	No internet service	...	No int se
5947	7951-QKZPL	Female	0	Yes	Yes	33	Yes	Yes	No	No internet service	...	No int se
6680	9412-ARGBX	Female	0	No	Yes	48	Yes	No	Fiber optic	No	...	

3 rows × 21 columns

In [12]:
```
#Extracting 'f'-Extract 333 random record from customer churn dataframe & store the result in customer_333
c_333=customer_churn.sample(n=333)
```

In [13]:
```
c_333.head()
```

Out[13]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...	DeviceProte
1445	3211-AAPKX	Male	0	No	No	20	Yes	Yes	Fiber optic	No	...	
6293	7977-HXJKU	Male	0	No	Yes	21	Yes	No	No	No internet service	...	No int se
1081	1751-NCDLI	Male	1	Yes	No	46	Yes	Yes	Fiber optic	No	...	
851	2252-NKNSI	Male	0	No	Yes	52	Yes	Yes	DSL	Yes	...	

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...	DeviceProte
3938	6629-LADHQ	Female	0	No	No	2	Yes	No	DSL	Yes	...	

5 rows × 21 columns

In [14]:
```
customer_churn['Churn'].value_counts() #get the count of diffrent levels from the churn column
```

Out[14]:
```
No     5174
Yes    1869
Name: Churn, dtype: int64
```

In [15]:
```
customer_churn['Contract'].value_counts()
```

Out[15]:
```
Month-to-month    3875
Two year          1695
One year          1473
Name: Contract, dtype: int64
```

Data visualization

In [16]:
```
#visualization of a,
customer_churn['InternetService'].value_counts().keys().tolist()
```

Out[16]:
```
['Fiber optic', 'DSL', 'No']
```

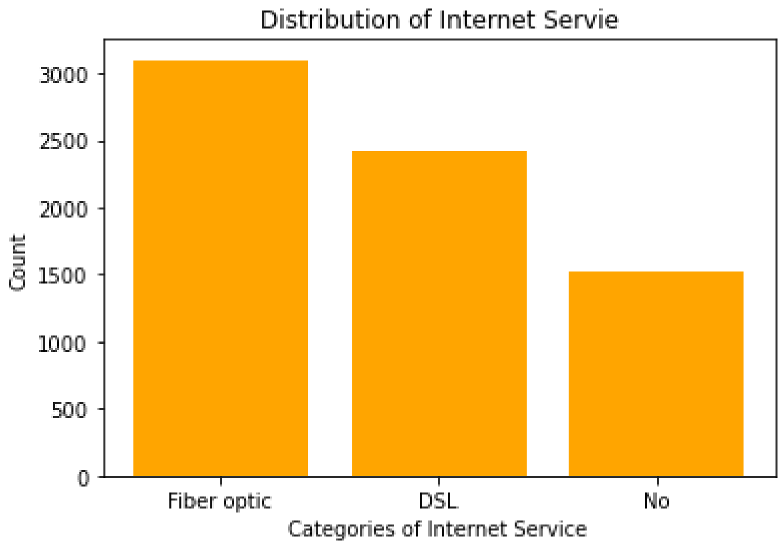
In [17]:
```
customer_churn['InternetService'].value_counts().tolist()
```

Out[17]:
```
[3096, 2421, 1526]
```

In [18]:
```
#bar plot is used when we need categorical column
plt.bar(customer_churn['InternetService'].value_counts().keys().tolist(),customer_churn['InternetService'].value_cour
plt.xlabel("Categories of Internet Service")
plt.ylabel('Count')
plt.title("Distribution of Internet Servie")
```

```
Text(0.5, 1.0, 'Distribution of Internet Servie')
```

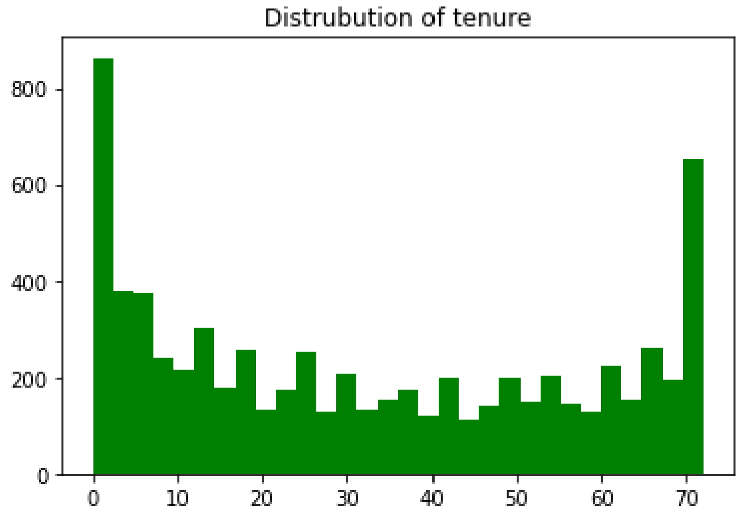
Out[18]:



In [19]:

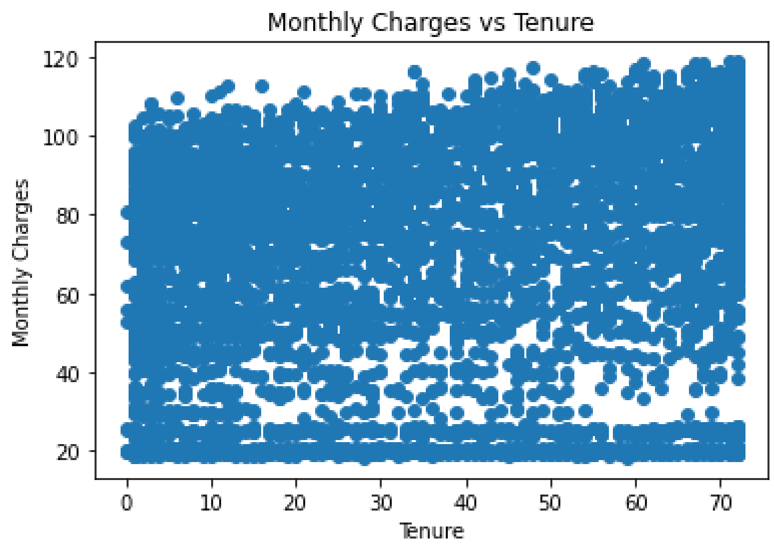
```
#visualtion of b,
#Distrubution of countinous numerical column we go with an historam
plt.hist(customer_churn['tenure'],bins=30,color='green')
plt.title('Distrubution of tenure')
```

Out[19]: Text(0.5, 1.0, 'Distrubution of tenure')



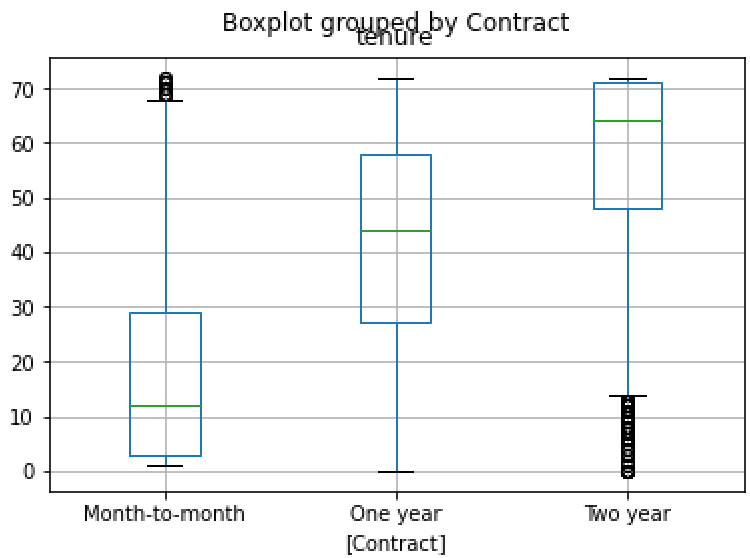
In [20]:
```
#visualizing 'c'
plt.scatter(x=customer_churn['tenure'],y=customer_churn['MonthlyCharges'])
plt.xlabel('Tenure')
plt.ylabel('Monthly Charges')
plt.title('Monthly Charges vs Tenure')
```

Out[20]:
```
Text(0.5, 1.0, 'Monthly Charges vs Tenure')
```



In [21]:
```
#visualizing 'd'
customer_churn.boxplot(column=['tenure'],by=['Contract'])
```

Out[21]: <AxesSubplot:title={'center':'tenure'}, xlabel='[Contract]'>



Machine Learning

A.Linear Regression

```
Buid a simple linear model where dependent variables is 'MonthlyCharges' and indepenent Variables is
'tenure'

1.Divide the dataset into train and test test in 70:30 ratio

2.Buid a model on train set and predict the values on test set

3.After predicting the values, find the root mean square error

4.Find out the error in the prediction & store the result in 'error'

5.Find the root mean square error
```

In [22]:
```
from sklearn import linear_model
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split

y=customer_churn[['MonthlyCharges']]
x=customer_churn[['tenure']]
```

In [23]:
```
y.head(),x.head()
```

Out[23]:
```
(   MonthlyCharges
 0           29.85
 1           56.95
 2           53.85
 3           42.30
 4           70.70,
    tenure
 0       1
 1      34
 2       2
 3      45
 4       2)
```

In [24]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.30,random_state=0)
```

In [25]:
```
x_train.shape,y_train.shape,x_test.shape,y_test.shape
```

Out[25]:
```
((4930, 1), (4930, 1), (2113, 1), (2113, 1))
```

In [26]:
```
regressor=LinearRegression()

regressor.fit(x_train,y_train)
```

Out[26]:
```
LinearRegression()
```

In [27]:
```
y_pred=regressor.predict(x_test)
y_pred[:5],y_test[:5]
```

Out[27]:
```
(array([[60.95089608],
        [72.98096699],
        [59.1903979 ],
        [55.66940154],
        [71.51388517]]),
       MonthlyCharges
 2200           58.20
 4627          116.60
 3225           71.95
 2828           20.45
 3768           77.75)
```

In [28]:
```
from sklearn.metrics import mean_squared_error

np.sqrt(mean_squared_error(y_test,y_pred))
```

Out[28]:
```
29.394584027273893
```

Logistic Regression

Build a multiple logistic regression model where dependent variables is 'Churn' & independent variable are 'tenure' & 'MonthlyCharges'

```
1.Divide the dataset in 80:20 ratio
```

2.Build the model on train set and predict the values on test set

3.Build the Confusion matrix and get the accuracy score

In [29]:
```
x=customer_churn[['MonthlyCharges','tenure']]
y=customer_churn[['Churn']]
```

In [30]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20,random_state=0)
```

In [31]:
```
from sklearn.linear_model import LogisticRegression

log_model=LogisticRegression()

log_model.fit(x_train,y_train)
```

```
C:\Users\win7\anaconda3\lib\site-packages\sklearn\utils\validation.py:73: DataConversionWarning: A column-vector y wa
s passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  return f(**kwargs)
```

Out[31]:
```
LogisticRegression()
```

In [32]:
```
y_pred=log_model.predict(x_test)
```

In [33]:
```
from sklearn.metrics import confusion_matrix,accuracy_score
```

In [34]:
```
confusion_matrix(y_test,y_pred),accuracy_score(y_test,y_pred)
```

Out[34]:
```
(array([[934, 107],
        [212, 156]], dtype=int64),
 0.7735982966643009)
```

In [35]:
```
(935+157)/(935+157+106+211)
```

Out[35]:
```
0.7750177430801988
```

In [36]:
```
#We have got 77% of accuracy with Logistic Regression
```

Decision Tree

Buid a decision tree model where depenent variable is 'churn' & independent variable is 'tenure'

```
1.Divide the dataset in 80:20 ratio

2.Build the model on train set and predict the values on test set

3.Buid the confusion matrix and calculate the accuracy
```

In [37]:
```
x=customer_churn[['tenure']]
y=customer_churn[['Churn']]

from sklearn.tree import DecisionTreeClassifier
```

In [38]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20,random_state=0)
```

In [39]:
```
my_tree=DecisionTreeClassifier()

my_tree.fit(x_train,y_train)
```

Out[39]:
```
DecisionTreeClassifier()
```

In [40]:
```
my_tree.predict(x_test)
```

Out[40]:
```
array(['No', 'No', 'No', ..., 'No', 'No', 'Yes'], dtype=object)
```

In [41]:
```
from sklearn.metrics import confusion_matrix,accuracy_score
```

In [42]:
```
confusion_matrix(y_test,y_pred)
```

Out[42]:
```
array([[934, 107],
       [212, 156]], dtype=int64)
```

In [43]:
```
(934+156)/(965+156+212+107)
```

Out[43]:
```
0.7569444444444444
```

Random Forest Classifier

In [44]:
```
from sklearn.ensemble import RandomForestClassifier

rf=RandomForestClassifier()

rf.fit(x_train,y_train)
```

```
<ipython-input-44-3928cbcdd673>:5: DataConversionWarning: A column-vector y was passed when a 1d array was expected.
Please change the shape of y to (n_samples,), for example using ravel().
  rf.fit(x_train,y_train)
```

Out[44]:
```
RandomForestClassifier()
```

In [45]:
```
rf.predict(x_test)
```

Out[45]:
```
array(['No', 'No', 'No', ..., 'No', 'No', 'Yes'], dtype=object)
```

In [46]:
```
confusion_matrix(y_test,y_pred)
```

Out[46]:
```
array([[934, 107],
       [212, 156]], dtype=int64)
```

In [47]:
```
accuracy_score(y_test,y_pred)
```

Out[47]:
```
0.7735982966643009
```

With the Above data claculation Logistic Regressor Have Highest Accuracy

In [ ]:
```

```