Problem Statement:

You are the Data Scientist at a telecom company “Neo” whose customers are churning out to its competitors. You have to analyse the data of your company and find insights and stop your customers from churning out to other telecom companies. Tasks to be done:

A) Data Manipulation:

a. Extract the 5th column & store it in ‘customer_5’ b. Extract the 15th column & store it in ‘customer_15’ c. Extract all the male senior citizens whose Payment Method is Electronic check & store 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 the result in ‘customer_total_tenure’ 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’ f. Extract 333 random records from the customer_churndataframe& store the result in ‘customer_333’ g. Get the count of different levels from the ‘Churn’ column

In [5]:
```
import os
os.chdir('C:\\Users\\veena\\OneDrive\\Desktop')
```

In [3]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
```

In [6]:
```
cus_chr=pd.read_csv('customer_churn.csv')
cus_chr.head()
```

Out[6]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines
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 [30]:
```
cus_chr.isnull().sum()
```

Out[30]:
```
customerID          0
gender              0
SeniorCitizen       0
Partner             0
Dependents          0
tenure              0
PhoneService        0
MultipleLines       0
InternetService     0
OnlineSecurity      0
OnlineBackup        0
DeviceProtection    0
TechSupport         0
StreamingTV         0
StreamingMovies     0
Contract            0
PaperlessBilling    0
PaymentMethod       0
MonthlyCharges      0
TotalCharges        0
Churn               0
dtype: int64
```

In [13]:
```
customer_5=cus_chr[4:5]
customer_5
```

Out[13]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines
4	9237-HQITU	Female	0	No	No	2	Yes	No

1 rows × 21 columns

In [14]:
```
customer_15=cus_chr[14:15]
customer_15
```

Out[14]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines
14	5129-JLPIS	Male	0	No	No	25	Yes	No

1 rows × 21 columns

In [32]:
```
senior_male_electronic = cus_chr.loc[(cus_chr['SeniorCitizen']==True) & (cus_chr['I
senior_male_electronic.head()
```

Out[32]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines
20	8779-QRDMV	Male	1	No	No	1	No	No phone service
50	8012-SOUDQ	Female	1	No	No	43	Yes	Yes
55	1658-BYGOY	Male	1	No	No	18	Yes	Yes
57	5067-XJQFU	Male	1	Yes	Yes	66	Yes	Yes
75	2673-CXQEU	Female	1	No	No	56	Yes	Yes

5 rows × 21 columns

In [40]:

```
customer_total_tenure = cus_chr.loc[(cus_chr['tenure']>70) & (cus_chr['MonthlyChar
customer_total_tenure.head()
```

Out[40]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLine
17	9959-WOFKT	Male	0	No	Yes	71	Yes	Ye
59	5954-BDFSG	Female	0	No	No	72	Yes	Ye
94	9848-JQJTX	Male	0	No	No	72	Yes	Ye
106	6728-DKUCO	Female	0	Yes	Yes	72	Yes	Ye
193	9680-NIAUV	Female	0	Yes	Yes	72	Yes	Ye

5 rows × 21 columns

In [42]:

```
#e. Extract all the customers whose Contract is of two years, payment method is Ma
#check & the value of Churn is 'Yes' & store the result in 'two_mail_yes'
two_mail_yes = cus_chr.loc[(cus_chr['Contract']=='Two year') & (cus_chr['PaymentMe
two_mail_yes.head()
```

Out[42]:

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

3 rows × 21 columns

In [46]:

```
#f. Extract 333 random records from the customer_churn dataframe& store the result
customer_333=cus_chr.sample(n=3)
customer_333
```

Out[46]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLin
5561	5088-QZLRL	Male	0	No	No	1	Yes	N
30	3841-NFECX	Female	1	Yes	No	71	Yes	Y
308	3167-SNQPL	Male	1	Yes	Yes	38	Yes	Y

3 rows × 21 columns

In [50]:

```
#Get the count of different levels from the 'Churn' column
cus_chr.count(axis=0)
```

Out[50]:

```
customerID          7043
gender              7043
SeniorCitizen       7043
Partner             7043
Dependents          7043
tenure              7043
PhoneService        7043
MultipleLines       7043
InternetService     7043
OnlineSecurity      7043
OnlineBackup        7043
DeviceProtection    7043
TechSupport         7043
StreamingTV         7043
StreamingMovies     7043
Contract            7043
PaperlessBilling    7043
PaymentMethod       7043
MonthlyCharges      7043
TotalCharges        7043
Churn               7043
dtype: int64
```

In [51]:

```
cus_chr.count(axis=1)
```

Out[51]:
```
0       21
1       21
2       21
3       21
4       21
        ..
7038    21
7039    21
7040    21
7041    21
7042    21
Length: 7043, dtype: int64
```

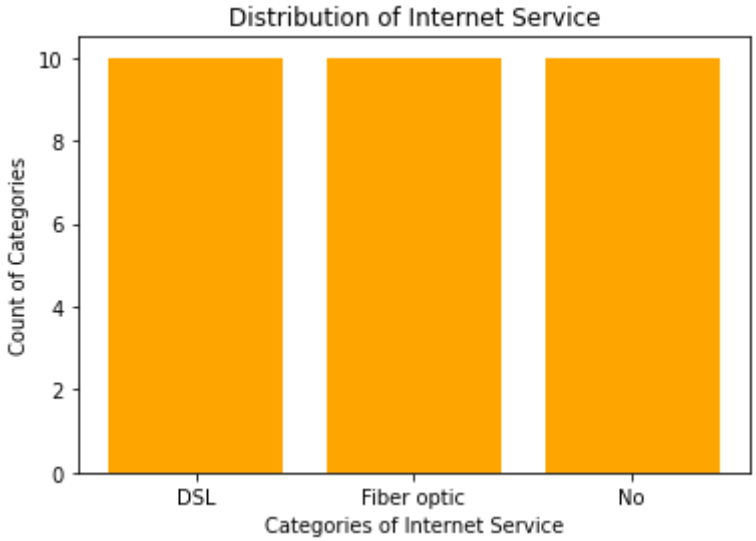
B) Data Visualization:

In [2]:
```
import matplotlib.pyplot as plt
%matplotlib inline
```

In [ ]:
```
#a. Build a bar-plot for the 'InternetService' column:
i. Set x-axis label to 'Categories of Internet Service'
ii. Set y-axis label to 'Count of Categories'
iii. Set the title of plot to be 'Distribution of Internet Service'
iv. Set the color of the bars to be 'orange'
```

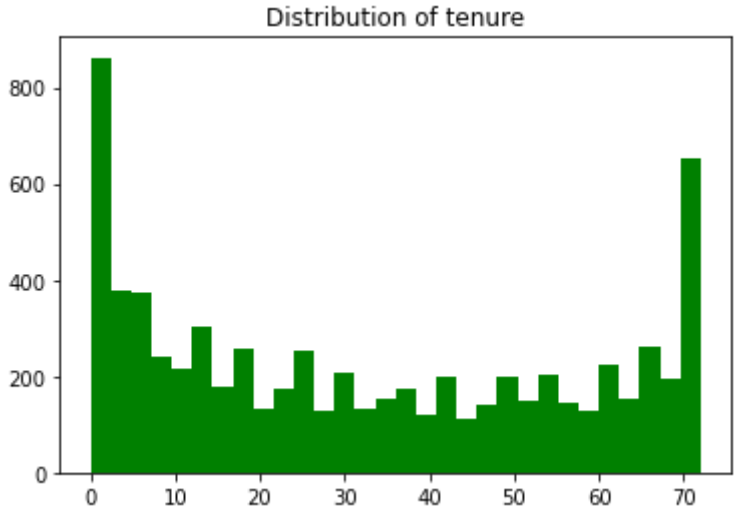
In [65]:
```
x=list(cus_chr.InternetService)
```

In [67]:
```
plt.bar(x, height=10, color='orange')
plt.xlabel ('Categories of Internet Service')
plt.ylabel ('Count of Categories')
plt.title("Distribution of Internet Service")
plt.show()
```



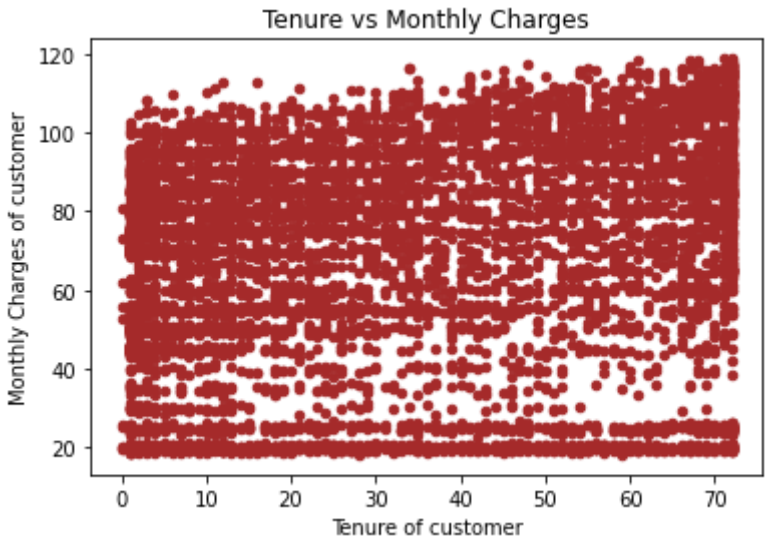
In [ ]:
```
#b. Build a histogram for the 'tenure' column:
i. Set the number of bins to be 30
ii. Set the color of the bins to be 'green'
iii. Assign the title 'Distribution of tenure'
```

In [7]:
```
plt.hist(cus_chr['tenure'], bins=30, color='green')
plt.title('Distribution of tenure')
plt.show()
```



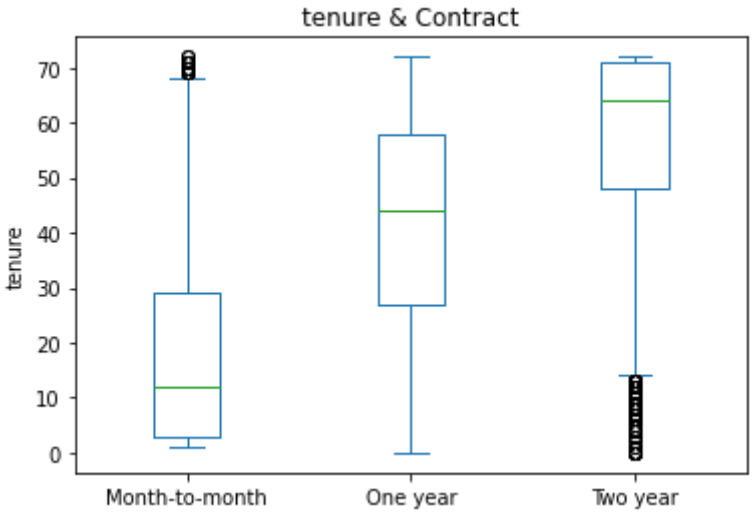
In [ ]:
```
#c. Build a scatter-plot between ‘MonthlyCharges’ & ‘tenure’. Map ‘MonthlyCharges’
i. Assign the points a color of ‘brown’
ii. Set the x-axis label to ‘Tenure of customer’
iii. Set the y-axis label to ‘Monthly Charges of customer’
iv. Set the title to ‘Tenure vs Monthly Charges’
```

In [8]:
```
cus_chr.plot.scatter(x='tenure', y='MonthlyCharges', color='brown')
plt.xlabel('Tenure of customer')
plt.ylabel('Monthly Charges of customer')
plt.title('Tenure vs Monthly Charges')
plt.show()
```



In [ ]:
```
#d. Build a box-plot between ‘tenure’ & ‘Contract’. Map ‘tenure’ on the y-axis &‘Co
```

In [15]:
```
boxplot=cus_chr.plot.box(column='tenure', by='Contract')
plt.ylabel('tenure')
plt.title('tenure & Contract')
plt.show()
```



In [ ]:
```
# C) Linear Regression:

#a. Build a simple linear model where dependent variable is 'MonthlyCharges' and i
#i. Divide the dataset into train and test sets in 70:30 ratio.
#ii. Build the model on train set and predict the values on test set
#iii. After predicting the values, find the root mean square error
#iv. Find out the error in prediction & store the result in 'error'
#v. Find the root mean square error
```

In [28]:
```
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
```

In [107…
```
x=cus_chr[['tenure']]
y=cus_chr[['MonthlyCharges']]
```

In [108…
```
x_train, x_test, y_train, y_test=train_test_split(x,y,test_size=0.3)
```

In [109…
```
lr=LinearRegression()
lr
```

Out[109]:
```
▾ LinearRegression
LinearRegression()
```

In [110…
```
lr.fit(x_train,y_train)
```

Out[110]:
```
▾ LinearRegression
LinearRegression()
```

In [111…
```
y_pred=lr.predict(x_test)
y_pred
```

Out[111]:
```
array([[54.81470613],
       [76.86401174],
       [56.38965653],
       ...,
       [57.33462677],
       [57.64961685],
       [68.98925973]])
```

In [112…
```
from sklearn.metrics import mean_squared_error
from math import sqrt
rmse=sqrt(mean_squared_error(y_test,y_pred))
print(rmse)
```

```
28.87223874368882
```

In [113…
```
error=mean_squared_error(y_test,y_pred)
error
```

Out[113]:
```
833.6061700725656
```

In [ ]:

D) Logistic Regression:

a. Build a simple logistic regression model where dependent variable is 'Churn' & independent variable is 'MonthlyCharges' i. Divide the dataset in 65:35 ratio ii. Build the model on train set and predict the values on test set iii. Build the confusion matrix and get the accuracy score b. Build a multiple logistic regression model where dependent variable is 'Churn' & independent variables are 'tenure' & 'MonthlyCharges' i. Divide the dataset in 80:20 ratio ii. Build the model on train set and predict the values on test set iii. Build the confusion matrix and get the accuracy score

In [48]:
```
x=cus_chr[['MonthlyCharges']]
y=cus_chr[['Churn']]
```

In [51]:
```
x_train, x_test, y_train, y_test=train_test_split(x,y,test_size=0.35)
```

In [52]:
```
from sklearn.linear_model import LogisticRegression
logr=LogisticRegression()
logr
```

Out[52]:
```
▾ LogisticRegression
LogisticRegression()
```

In [53]:
```
logr.fit(x_train,y_train)
```

```
C:\Users\veena\anaconda3\lib\site-packages\sklearn\utils\validation.py:1141: DataC
onversionWarning: A column-vector y was passed when a 1d array was expected. Pleas
e change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

Out[53]:
```
▾ LogisticRegression
LogisticRegression()
```

In [55]:
```
y_pred=logr.predict(x_test)
y_pred
```

Out[55]:
```
array(['No', 'No', 'No', ..., 'No', 'No', 'No'], dtype=object)
```

In [57]:
```
from sklearn.metrics import confusion_matrix
cm=confusion_matrix(y_test,y_pred)
```

```
cm
```

Out[57]:
```
array([[1818,    0],
       [ 648,    0]], dtype=int64)
```

In [ ]:
```
#mulitivariable Logistic Regression
```

In [72]:
```
a=cus_chr[['tenure','MonthlyCharges']]
b=cus_chr[['Churn']]
```

In [73]:
```
a_train,a_test, b_train, b_test = train_test_split(a,b,test_size=0.2)
```

In [74]:
```
from sklearn.linear_model import LogisticRegression
logr=LogisticRegression()
logr
```

Out[74]:
```
▾ LogisticRegression
LogisticRegression()
```

In [75]:
```
logr.fit(a_train,b_train)
```

```
C:\Users\veena\anaconda3\lib\site-packages\sklearn\utils\validation.py:1141: DataC
onversionWarning: A column-vector y was passed when a 1d array was expected. Pleas
e change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

Out[75]:
```
▾ LogisticRegression
LogisticRegression()
```

In [77]:
```
b_pred=logr.predict(a_test)
b_pred
```

Out[77]:
```
array(['No', 'No', 'No', ..., 'No', 'No', 'Yes'], dtype=object)
```

In [78]:
```
from sklearn.metrics import confusion_matrix
cm=confusion_matrix(b_test,b_pred)
cm
```

Out[78]:
```
array([[927, 111],
       [197, 174]], dtype=int64)
```

In [79]:
```
logr.score(a_test,b_test)
```

Out[79]:
```
0.7814052519517388
```

In [84]:
```
from sklearn.metrics import accuracy_score
accuracy_score(b_test, b_pred)
```

Out[84]:
```
0.7814052519517388
```

In [ ]:

E) Decision Tree:

a. Build a decision tree model where dependent variable is 'Churn' & independent variable is 'tenure' i. Divide the dataset in 80:20 ratio ii. Build the model on train set and predict the values on test set iii. Build the confusion matrix and calculate the accuracy

In [85]:
```
p=cus_chr[['tenure']]
q=cus_chr[['Churn']]
```

In [86]:
```
p_train, p_test, q_train, q_test = train_test_split(p,q,test_size=0.2)
```

In [87]:
```
from sklearn import tree
```

In [88]:
```
tr=tree.DecisionTreeClassifier()
tr
```

Out[88]:
```
▾ DecisionTreeClassifier
DecisionTreeClassifier()
```

In [89]:
```
tr.fit(p_train,q_train)
```

Out[89]:
```
▾ DecisionTreeClassifier
DecisionTreeClassifier()
```

In [92]:
```
q_pred=tr.predict(p_test)
q_pred
```

Out[92]:
```
array(['No', 'Yes', 'No', ..., 'No', 'No', 'Yes'], dtype=object)
```

In [94]:
```
from sklearn.metrics import confusion_matrix
cm=confusion_matrix(q_test,q_pred)
cm
```

Out[94]:
```
array([[918,  72],
       [309, 110]], dtype=int64)
```

In [95]:
```
from sklearn.metrics import accuracy_score
accuracy_score(q_test,q_pred)
```

Out[95]:
```
0.7295954577714692
```

In [ ]:

F) Random Forest:

a. Build a Random Forest model where dependent variable is 'Churn' & independent variables are 'tenure' and 'MonthlyCharges' i. Divide the dataset in 70:30 ratio ii. Build the model on train set and predict the values on test set iii. Build the confusion matrix and calculate the accuracy

In [96]:
```
c=cus_chr[['tenure','MonthlyCharges']]
d=cus_chr[['Churn']]
```

In [97]:
```
c_train, c_test, d_train, d_test = train_test_split(c,d,test_size=0.3)
```

In [98]:
```
from sklearn.ensemble import RandomForestClassifier
rfc=RandomForestClassifier()
rfc
```

Out[98]:
```
▾ RandomForestClassifier
RandomForestClassifier()
```

In [99]:
```
rfc.fit(c_train, d_train)
```

```
C:\Users\veena\AppData\Local\Temp\ipykernel_13760\661053459.py:1: DataConversionWa
rning: A column-vector y was passed when a 1d array was expected. Please change th
e shape of y to (n_samples,), for example using ravel().
  rfc.fit(c_train, d_train)
```

Out[99]:
```
▾ RandomForestClassifier
RandomForestClassifier()
```

In [101…
```
d_pred=rfc.predict(c_test)
d_pred
```

Out[101]:
```
array(['No', 'No', 'No', ..., 'No', 'No', 'No'], dtype=object)
```

In [102…
```
from sklearn.metrics import confusion_matrix
cm=confusion_matrix(d_test,d_pred)
cm
```

Out[102]:
```
array([[1301,  238],
       [ 316,  258]], dtype=int64)
```

In [104…
```
accuracy_score(d_test,d_pred)
```

Out[104]:
```
0.7378135352579271
```

In [ ]: