In [7]:

```
# QUESTION
#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_m
#d. Extract all those customers whose tenure is greater than 70 months or their Monthly charges is more than 100$
#e. Extract all the customers whose Contract is of two years, payment method is Mailed check & the value of Churn
#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 [3]:

```
# ANSWER BELLOW :
```

In [3]:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
sns.set(style="darkgrid")
%matplotlib inline
```

In [5]:

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

In [6]:

```
customer
```

Out[6]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...	DeviceP
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	...	
...	...	...	...	...	...	...	...	...	...	...	...	
7038	6840-RESVB	Male	0	Yes	Yes	24	Yes	Yes	DSL	Yes	...	
7039	2234-XADUH	Female	0	Yes	Yes	72	Yes	Yes	Fiber optic	No	...	
7040	4801-JZAZL	Female	0	Yes	Yes	11	No	No phone service	DSL	Yes	...	
7041	8361-LTMKD	Male	1	Yes	No	4	Yes	Yes	Fiber optic	No	...	
7042	3186-AJIEK	Male	0	No	No	66	Yes	No	Fiber optic	Yes	...	

7043 rows × 21 columns

In [7]:

```
customer.head(5)
```

Out[7]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...	DeviceProte
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 [11]:
```
customer.iloc[:,5]
```

Out[11]:
```
0        1
1       34
2        2
3       45
4        2
        ..
7038    24
7039    72
7040    11
7041     4
7042    66
Name: tenure, Length: 7043, dtype: int64
```

In [12]:
```
customer.iloc[:,15]
```

Out[12]:
```
0       Month-to-month
1             One year
2       Month-to-month
3             One year
4       Month-to-month
             ...
7038          One year
7039          One year
7040    Month-to-month
7041    Month-to-month
7042          Two year
Name: Contract, Length: 7043, dtype: object
```

In [14]:
```
seniorcitizen=customer[(customer["gender"]=="Male")&(customer["SeniorCitizen"]==1)&(customer["PaymentMethod"]=="E
```

In [15]:
```
seniorcitizen
```

Out[15]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...	DeviceP
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	...	
...	...	...	...	...	...	...	...	...	...	...	...	
6837	6229-LSCKB	Male	1	No	No	6	Yes	No	Fiber optic	No	...	
6894	1400-MMYXY	Male	1	Yes	No	3	Yes	Yes	Fiber optic	No	...	
6914	7142-HVGBG	Male	1	Yes	No	43	Yes	Yes	Fiber optic	No	...	
6967	8739-WWKDU	Male	1	No	No	25	Yes	Yes	Fiber optic	No	...	
7032	6894-LFHLY	Male	1	No	No	1	Yes	Yes	Fiber optic	No	...	

298 rows × 21 columns

In [16]:
```
seniorcitizen=customer[(customer["tenure"]>70 )|(customer["MonthlyCharges"]<100)]
```

In [17]:
```
seniorcitizen
```

```
seniorcitizen
```

Out[17]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...	DeviceP
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	...	
...	...	...	...	...	...	...	...	...	...	...	...	
7037	2569-WGERO	Female	0	No	No	72	Yes	No	No	No internet service	...	N
7038	6840-RESVB	Male	0	Yes	Yes	24	Yes	Yes	DSL	Yes	...	
7039	2234-XADUH	Female	0	Yes	Yes	72	Yes	Yes	Fiber optic	No	...	
7040	4801-JZAZL	Female	0	Yes	Yes	11	No	No phone service	DSL	Yes	...	
7041	8361-LTMKD	Male	1	Yes	No	4	Yes	Yes	Fiber optic	No	...	

6310 rows × 21 columns

In [ ]:

```
# E)Extract all the customers whose contract is of two years,payment method is mailed check & the value of churn
```

In [18]:

```
seniorcitizen_mailcheck=seniorcitizen[(seniorcitizen['Contract']=='Two year') & (seniorcitizen['PaymentMethod']==
```

In [20]:

```
seniorcitizen
```

Out[20]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...	DeviceP
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	...	
...	...	...	...	...	...	...	...	...	...	...	...	
7037	2569-WGERO	Female	0	No	No	72	Yes	No	No	No internet service	...	N
7038	6840-RESVB	Male	0	Yes	Yes	24	Yes	Yes	DSL	Yes	...	
7039	2234-XADUH	Female	0	Yes	Yes	72	Yes	Yes	Fiber optic	No	...	
7040	4801-JZAZL	Female	0	Yes	Yes	11	No	No phone service	DSL	Yes	...	
7041	8361-LTMKD	Male	1	Yes	No	4	Yes	Yes	Fiber optic	No	...	

6310 rows × 21 columns

In [ ]:

```
## F)Extract 333 random records from the customer_churn dataframe & store the result in 'customer_333'
```

In [21]:

```
customer_333=customer.sample(n=333)
```

In [22]:

```
customer_333
```

Out[22]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...	DeviceP
4590	4884-TVUQF	Female	1	No	No	57	Yes	Yes	Fiber optic	Yes	...	
4797	3892-NXAZG	Male	0	Yes	Yes	72	Yes	Yes	Fiber optic	Yes	...	
3117	9844-FELAJ	Female	1	Yes	Yes	70	Yes	Yes	Fiber optic	No	...	
2685	5781-BKHOP	Female	0	Yes	No	72	Yes	Yes	Fiber optic	Yes	...	
6997	2523-EWWZL	Female	0	Yes	No	27	Yes	No	Fiber optic	No	...	
...	...	...	...	...	...	...	...	...	...	...	...	
4741	6583-KQJLK	Female	1	Yes	No	15	Yes	No	Fiber optic	No	...	
220	9408-SSNVZ	Female	0	No	No	4	Yes	No	Fiber optic	No	...	
647	2391-IPLOP	Male	0	Yes	Yes	50	Yes	Yes	DSL	No	...	
6444	6302-JGYRJ	Male	0	No	Yes	31	Yes	Yes	DSL	No	...	
6337	2696-ECXKC	Female	0	Yes	Yes	55	Yes	Yes	Fiber optic	Yes	...	

333 rows × 21 columns

In [ ]:

```
#g) Get the count of different levels from the 'churn' column
```

In [23]:

```
customer['Churn'].value_counts()
```

Out[23]:

```
No     5174
Yes    1869
Name: Churn, dtype: int64
```

In [24]:

```
# QUESTION
# 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’
#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’
#c. Build a scatter-plot between ‘MonthlyCharges’ & ‘tenure’. Map ‘MonthlyCharges’ to the y-axis & ‘tenure’ to th
#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’
#d. Build a box-plot between ‘tenure’ & ‘Contract’. Map ‘tenure’ on the y-axis & ‘Contract’ on the x-axis
```

In [31]:

```
#i)set x-axis label to 'Categories of internet Service
```

In [26]:

```
import matplotlib.pyplot as plt
```

In [32]:

```
customer['InternetService'].value_counts()
```

Out[32]:

```
Fiber optic    3096
DSL            2421
No             1526
Name: InternetService, dtype: int64
```

In [36]:

```
x=customer['InternetService'].value_counts()
```

In [37]:

```
seniorcitizen
```

Out[37]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...	DeviceP
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	...	
...	...	...	...	...	...	...	...	...	...	...	...	
7037	2569-WGERO	Female	0	No	No	72	Yes	No	No	No internet service	...	N
7038	6840-RESVB	Male	0	Yes	Yes	24	Yes	Yes	DSL	Yes	...	
7039	2234-XADUH	Female	0	Yes	Yes	72	Yes	Yes	Fiber optic	No	...	
7040	4801-JZAZL	Female	0	Yes	Yes	11	No	No phone service	DSL	Yes	...	
7041	8361-LTMKD	Male	1	Yes	No	4	Yes	Yes	Fiber optic	No	...	

6310 rows × 21 columns

In [38]:

```
y=customer['InternetService'].value_counts().tolist()
```

In [39]:

```
y
```

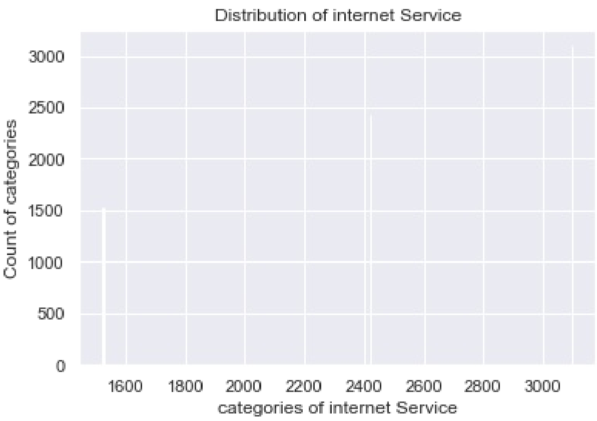
Out[39]:

```
[3096, 2421, 1526]
```

In [43]:

```
import numpy as np
import matplotlib.pyplot as plt

plt.bar(x,y,color='red')
plt.xlabel("categories of internet Service")
plt.ylabel("Count of categories")
plt.title("Distribution of internet Service")
plt.show()
```

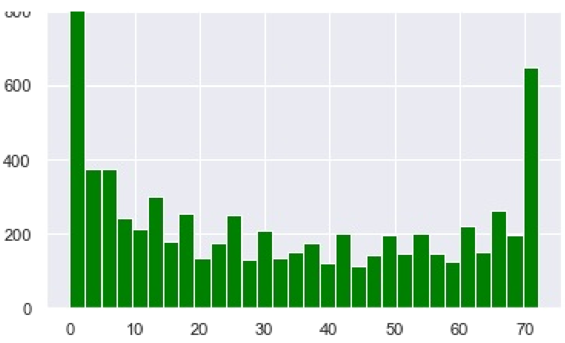


In [44]:

```
# b)Build a histogram for the 'tenure' column:
## i)set num of bins to be 30
## ii)set the color of bins to be 'green'
## iii)Assign the title 'Distribution of tenure'
```

In [45]:

```
plt.hist(customer['tenure'],bins=30,color='green')
plt.title("Distribution of tenure")
plt.show()
```



In [ ]:

```
### Built a scatter-plot between 'MonthlyCharges' & 'tenure'. Map 'MonthlyCharges' to the y-axis & 'tenure' to th
#### 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 the 'Tenure vs Monthly Charges'
```

In [46]:

```
import pandas as pd
```

In [47]:

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

In [48]:

```
data
```

Out[48]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...	DeviceP
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	...	
...	...	...	...	...	...	...	...	...	...	...	...	
7038	6840-RESVB	Male	0	Yes	Yes	24	Yes	Yes	DSL	Yes	...	
7039	2234-XADUH	Female	0	Yes	Yes	72	Yes	Yes	Fiber optic	No	...	
7040	4801-JZAZL	Female	0	Yes	Yes	11	No	No phone service	DSL	Yes	...	
7041	8361-LTMKD	Male	1	Yes	No	4	Yes	Yes	Fiber optic	No	...	
7042	3186-AJIEK	Male	0	No	No	66	Yes	No	Fiber optic	Yes	...	

7043 rows × 21 columns

In [101…

```
data.head(2)
```

Out[101…

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...	DeviceProte
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	...	

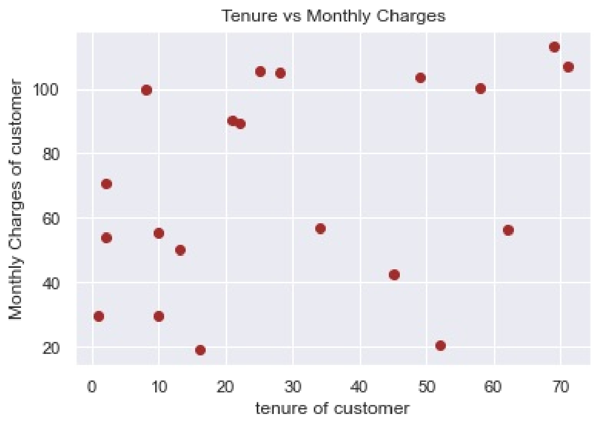
2 rows × 21 columns

In [49]:

```
import matplotlib.pyplot as plt
```

In [50]:

```
plt.scatter(x=data['tenure'].head(20),y=data['MonthlyCharges'].head(20),color='brown')
plt.xlabel('tenure of customer')
plt.ylabel('Monthly Charges of customer')
plt.title('Tenure vs Monthly Charges')
plt.show()
```



In [106…:

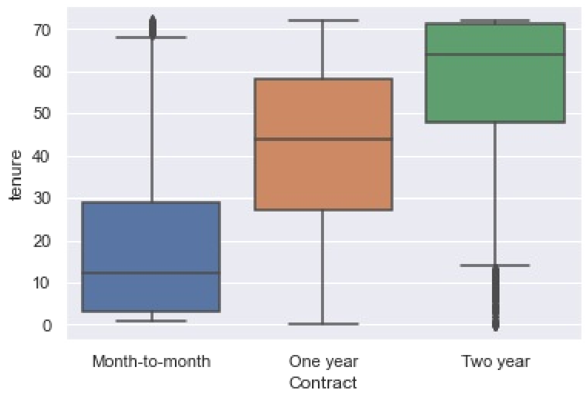
```
## d)built a boxplot between tenure & contract. Map 'tenure' on y-axis and 'Contract' on the x-axis
```

In [51]:

```
import seaborn as sns
```

In [52]:

```
sns.boxplot(x=data['Contract'],y=data['tenure'])
plt.show()
```

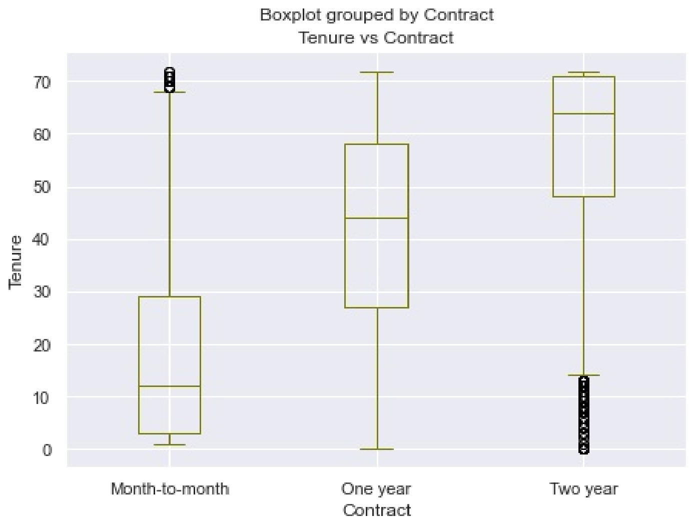


In [53]:

```
import seaborn as sns
import matplotlib.pyplot as plt
```

In [54]:

```
data.boxplot(by=['Contract'],column="tenure",figsize=(7,5),color='olive')
plt.xlabel("Contract")
plt.ylabel("Tenure")
plt.title("Tenure vs Contract")
plt.show()
```



In [ ]:

```
# QUESETION
# C) Linear Regression:
# a. Build a simple linear model where dependent variable is ‘MonthlyCharges’ and independent variable is ‘tenure
#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 [ ]:

```
# i)Divide the dataset into train and test sets is 70:30 ratio
```

In [55]:

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
```

In [56]:

```
data.head(2)
```

Out[56]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...	DeviceProte
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 rows × 21 columns

In [57]:

```
x=pd.DataFrame(data['tenure']) #independent variable
```

In [58]:

```
y=data['MonthlyCharges']        # dependent variable
```

In [59]:

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=0)
print(data.shape)
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(7043, 21)
(4930, 1)
(2113, 1)
(4930,)
(2113,)
```

In [118…

```
import numpy as np
import pandas as pd
```

In [119…

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

In [60]:

```
data
```

Out[60]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...	DeviceP
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	...	
...	...	...	...	...	...	...	...	...	...	...	...	
7038	6840-RESVB	Male	0	Yes	Yes	24	Yes	Yes	DSL	Yes	...	
7039	2234-XADUH	Female	0	Yes	Yes	72	Yes	Yes	Fiber optic	No	...	

7040	4801-JZAZL	Female	0	Yes	Yes	11	No	No phone service	DSL	Yes	...
7041	8361-LTMKD	Male	1	Yes	No	4	Yes	Yes	Fiber optic	No	...
7042	3186-AJIEK	Male	0	No	No	66	Yes	No	Fiber optic	Yes	...

7043 rows × 21 columns

In [122…

```
data.head(3)
```

Out[122…

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...	DeviceProte
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 rows × 21 columns

In [61]:

```
lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[61]:

```
LinearRegression()
```

In [62]:

```
y_pred=lr.predict(x_test)
```

In [63]:

```
y_pred
```

Out[63]:

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

In [64]:

```
y_test.values
```

Out[64]:

```
array([ 58.2 , 116.6 ,  71.95, ..., 109.95,  24.55,  81.6 ])
```

In [ ]:

```
#D)Logistic Regression
# a)Built a simple logistic regressin model where dependent variable is 'churn' & independent variable is 'Month
#i)Divide the dataset into 65:35 ratio
#ii)Build the model on train set and predict the values on test set
# iV)Build the confusion matrix and get the accuracy score¶
```

In [65]:

```
from sklearn.metrics import mean_squared_error
import numpy as np
```

In [66]:

```
msc=mean_squared_error(y_test,y_pred)
```

In [67]:

```
error=np.sqrt(msc)
```

In [68]:

```
error
```

Out[68]:

```
29.394584027273893
```

In [69]:

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

In [70]:
```
x=pd.DataFrame(data['MonthlyCharges'])
```

In [71]:
```
y=data['Churn']
```

In [72]:
```
y_pred
```

Out[72]:
```
array([60.95089608, 72.98096699, 59.1903979 , ..., 75.62171426,
       70.63363608, 65.6455579 ])
```

In [140…
```
y_test.values
```

Out[140…
```
array([ 58.2 , 116.6 ,  71.95, ..., 109.95,  24.55,  81.6 ])
```

In [145…
```
from sklearn.metrics import confusion_matrix,accuracy_score
```

In [146…
```
(1815+0)/(1815+651+0+0)
```

Out[146…
```
0.7360097323600974
```

In [148…
```
x=pd.DataFrame(data.loc[:,['tenure','MonthlyCharges']])
```

In [73]:
```
y=data['Churn']
```

In [74]:
```
y
```

Out[74]:
```
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 [75]:
```
mlr=LogisticRegression()
```

In [155…
```
mlr
```

Out[155…
```
LogisticRegression()
```

In [76]:
```
y_pred
```

Out[76]:
```
array([60.95089608, 72.98096699, 59.1903979 , ..., 75.62171426,
       70.63363608, 65.6455579 ])
```

In [77]:
```
y_test.values
```

Out[77]:
```
array([ 58.2 , 116.6 ,  71.95, ..., 109.95,  24.55,  81.6 ])
```

In [79]:
```
(934+156)/(934+156+212+107)
```

Out[79]:
```
0.7735982966643009
```

In [164…
```
 # QUESTION
# D) Logistic Regression:
# a. Build a simple logistic regression modelwhere dependent variable is ‘Churn’ & independent variable is ‘Month
# 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
```

In [80]:
```
x=pd.DataFrame(data.loc[:,['tenure','MonthlyCharges']])
```

In [81]:
```
y=data['Churn']
```

In [82]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,train_size=0.8,random_state=0)
```

In [83]:
```
mlr=LogisticRegression()
```

In [84]:
```
mlr.fit(x_train,y_train)
```

Out[84]:
```
LogisticRegression()
```

In [85]:
```
y_pred=mlr.predict(x_test)
```

In [86]:
```
y_pred
```

Out[86]:
```
array(['No', 'No', 'No', ..., 'No', 'No', 'No'], dtype=object)
```

In [87]:
```
y_test.values
```

Out[87]:
```
array(['No', 'No', 'No', ..., 'Yes', 'No', 'No'], dtype=object)
```

In [88]:
```
from sklearn.metrics import confusion_matrix,accuracy_score
```

In [89]:
```
print(confusion_matrix(y_pred,y_test))
```

```
[[934 212]
 [107 156]]
```

In [90]:
```
(934+156)/(934+156+212+107)
```

Out[90]:
```
0.7735982966643009
```

In [91]:
```
accuracy_score(y_test,y_pred)
```

Out[91]:
```
0.7735982966643009
```

In [ ]:
```
# QUESTION
# b. Build a multiple logistic regression model where dependent variable is ‘Churn’ & independent variables are ‘
# 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 [165…

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

In [166…

```
x_train,x_test,y_train,y_test=train_test_split(x,y,train_size=0.8,random_state=0)
```

In [167…

```
from sklearn.tree import DecisionTreeClassifier
```

In [168…

```
# Create Decision Tree classifer object
clf = DecisionTreeClassifier()

# Train Decision Tree Classifer
clf = clf.fit(x_train,y_train)

#Predict the response for test dataset
y_pred = clf.predict(x_test)
```

In [169…

```
y_pred
```

Out[169…

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

In [170…

```
y_test
```

Out[170…

```
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 [172…

```
#Import scikit-learn metrics module for accuracy calculation
from sklearn.metrics import confusion_matrix,accuracy_score
from sklearn import metrics
```

In [173…

```
# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

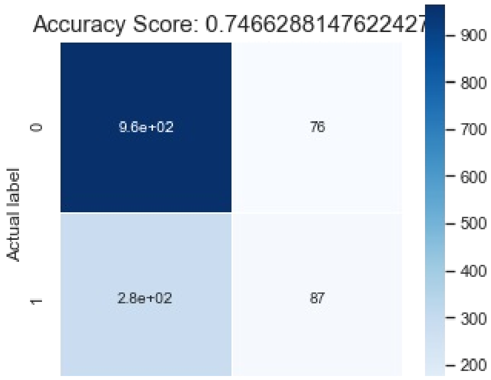
```
Accuracy: 0.7466288147622427
```

In [177…

```
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(5,5))
sns.heatmap(data=cm,linewidths=.5, annot=True,square = True,  cmap = 'Blues')
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
all_sample_title = 'Accuracy Score: {0}'.format(clf.score(x_test, y_test))
plt.title(all_sample_title, size = 15)
```

Out[177…

```
Text(0.5, 1.0, 'Accuracy Score: 0.7466288147622427')
```



In [92]:

```
!pip install graphviz
```

```
Requirement already satisfied: graphviz in /opt/anaconda3/lib/python3.8/site-packages (0.20)
```

In [93]:

```
pip install pydotplus
```

```
Requirement already satisfied: pydotplus in /opt/anaconda3/lib/python3.8/site-packages (2.0.2)
Requirement already satisfied: pyparsing>=2.0.1 in /opt/anaconda3/lib/python3.8/site-packages (from pydotplus) (2
.4.7)
Note: you may need to restart the kernel to use updated packages.
```

In [101…

```
from sklearn.tree import export_graphviz
from sklearn.externals.six import StrinIO
from IPython.display  import pydotplus

dot_data = StringIO
export_graphviz(clf, out_file=dot_data, filled=True, rounded=True, special_characters=True,feature_names = x_trai
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
graph.write_png('diabetes.png')
Image(graph.create_png())
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
<ipython-input-101-d5db378b6599> in <module>
      1 from sklearn.tree import export_graphviz
----> 2 from sklearn.externals.six import StrinIO
      3 from IPython.display  import pydotplus
      4 
      5 dot_data = StringIO

ModuleNotFoundError: No module named 'sklearn.externals.six'
```

In [ ]:

```
# QUESTION
#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 [102…

```
from sklearn.model_selection import train_test_split

X=data[['tenure']]  # Features
y=data['Churn']  # Labels


# Split dataset into training set and test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
```

In [107…

```
#Import Random Forest Model
from sklearn.ensemble import RandomForestClassifier

#Create a Random Forest Classifier
seniorcitizen=RandomForestClassifier(n_estimators=10)

#Train the model using the training sets y_pred=clf.predict(X_test)
seniorcitizen.fit(X_train,y_train)

y_pred=seniorcitizen.predict(X_test)

#Import scikit-learn metrics module for accuracy calculation
from sklearn import metrics

# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

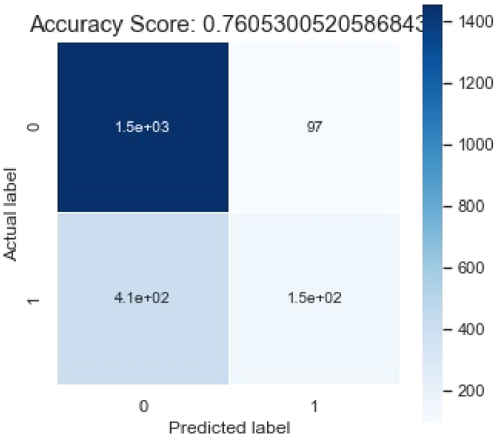
```
Accuracy: 0.7605300520586843
```

In [109…

```
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(5,5))
sns.heatmap(data=cm,linewidths=.5, annot=True,square = True,  cmap = 'Blues')
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
all_sample_title = 'Accuracy Score: {0}'.format(seniorcitizen.score(X_test, y_test))
plt.title(all_sample_title, size = 15)
```

Out[109…

```
Text(0.5, 1.0, 'Accuracy Score: 0.7605300520586843')
```



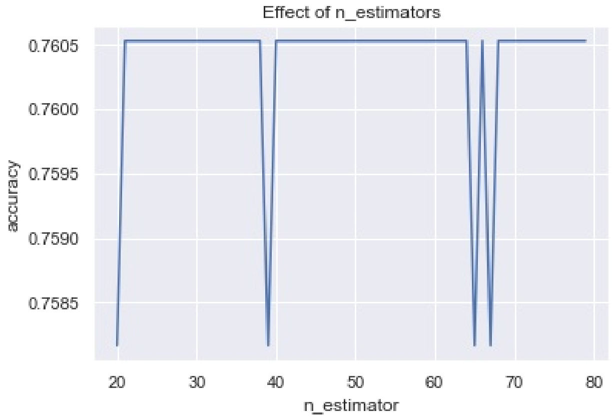
In [116…

```
# Try different numbers of n_estimators
estimators = np.arange(20,80)
accuracy = []

for n in estimators:
    seniorcitizen.set_params(n_estimators=n)
    seniorcitizen.fit(X_train, y_train)
    y_pred=seniorcitizen.predict(X_test)
    accuracy.append(metrics.accuracy_score(y_test, y_pred))
plt.title("Effect of n_estimators")
plt.xlabel("n_estimator")
plt.ylabel("accuracy")
plt.plot(estimators, accuracy)
```

Out[116…

```
[<matplotlib.lines.Line2D at 0x7fa40c950490>]
```



In [117…

```
# QUESTION
#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 [ ]:

In [ ]: