In [9]:

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

In [10]:

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

In [11]:

```python
data
```

Out[11]:

	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 [12]:

```python
data.head()
```

Out[12]:

	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

A)Data Manipulation

a) Extract the 5th column and store it in 'customer_5'

In [13]:

```python
customer_5=data.iloc[:,4]
customer_5
```

Out[13]:

```
0        No
1        No
2        No
3        No
4        No
       ...
7038    Yes
7039    Yes
7040    Yes
```

```
7041     No
7042     No
Name: Dependents, Length: 7043, dtype: object
```

b)extract the 15th column and store it in 'customer_15'

In [14]:

```
customer_15=data.iloc[:,14]
customer_15
```

Out[14]:

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

In [15]:

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

Out[15]:

	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

C)Extract all the male senior citizen whose payment method is electronic check and store the result in 'senior_male_electronic'

In [17]:

```
senior_male_electronic=data[(data['gender']=='Male') & (data['SeniorCitizen']==1) & (data['PaymentMethod']=='Elec
```

In [18]:

```
senior_male_electronic
```

Out[18]:

	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

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'

In [19]:

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

Out[19]:

	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 [20]:

```
customer_total_tenure=data[(data['tenure']>70) | (data['MonthlyCharges'] >100)]
```

In [21]:

```
customer_total_tenure
```

Out[21]:

	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	...	
...	...	...	...	...	...	...	...	...	...	...	...	
7023	1035-IPQPU	Female	1	Yes	No	63	Yes	Yes	Fiber optic	No	...	
7034	0639-TSIQW	Female	0	No	No	67	Yes	Yes	Fiber optic	Yes	...	
7037	2569-WGERO	Female	0	No	No	72	Yes	No	No	No internet service	...	N
7039	2234-XADUH	Female	0	Yes	Yes	72	Yes	Yes	Fiber optic	No	...	
7042	3186-AJIEK	Male	0	No	No	66	Yes	No	Fiber optic	Yes	...	

1259 rows × 21 columns

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'

In [22]:

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

Out[22]:

	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 [24]:

```
two_mail_yes=data[(data['Contract']=='Two year') & (data['PaymentMethod']=='Mailed check') & (data['Churn']=='Yes
```

In [25]:

```
two_mail_yes
```

Out[25]:

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

3 rows × 21 columns

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

In [19]:

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

In [ ]:

```
customer_333
```

Out[ ]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...	DeviceP
1589	7351-KYHQH	Female	1	No	No	7	Yes	No	DSL	No	...	
382	8204-YJCLA	Male	1	Yes	Yes	72	No	No phone service	DSL	Yes	...	
1480	8898-KASCD	Male	0	No	No	39	No	No phone service	DSL	No	...	
5645	4942-VZZOM	Male	0	Yes	No	64	Yes	Yes	DSL	Yes	...	
5379	6284-KMNUF	Female	0	Yes	No	56	Yes	Yes	Fiber optic	Yes	...	
...	...	...	...	...	...	...	...	...	...	...	...	
106	6728-DKUCO	Female	0	Yes	Yes	72	Yes	Yes	Fiber optic	Yes	...	
6521	1092-WPIVQ	Female	0	Yes	Yes	18	Yes	Yes	No	No internet service	...	N
4810	1112-CUNAO	Female	1	No	No	15	Yes	Yes	Fiber optic	No	...	
4194	2961-VNFKL	Female	0	Yes	No	71	Yes	Yes	No	No internet service	...	N
6317	7493-TPUWZ	Male	0	No	No	1	Yes	Yes	Fiber optic	No	...	

333 rows × 21 columns

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

In [26]:

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

Out[26]:

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

Data visualization

A)Build a barplot 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 bars to be 'orange'

In [27]:

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

Out[27]:

	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 [28]:

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

In [ ]:

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

Out[ ]:

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

In [29]:

```
x=data['InternetService'].value_counts().keys()
```

In [30]:

```
x
```

Out[30]:

```
Index(['Fiber optic', 'DSL', 'No'], dtype='object')
```

In [32]:

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

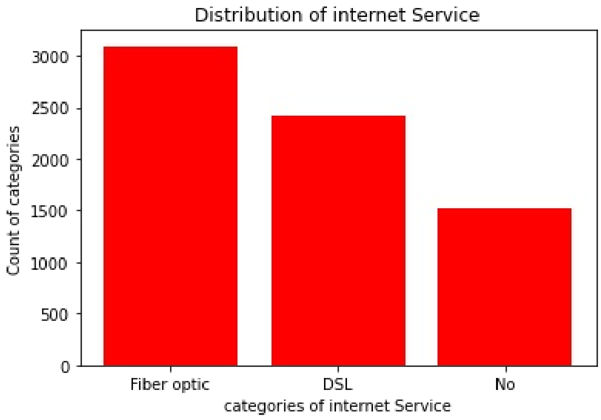
Out[32]:

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

In [33]:

```
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()
```



b)Build a histogram for the 'tenure' column:

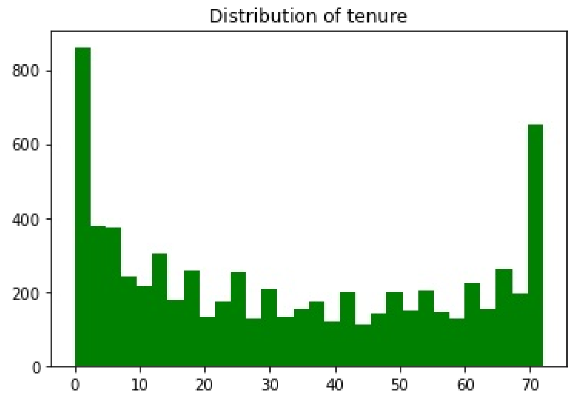
i)set num of bins to be 30

ii)set the color of bins to be 'green'

iii)Assign the title 'Distribution of tenure'

In [34]:

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



Built a scatter-plot between 'MonthlyCharges' & 'tenure'. Map 'MonthlyCharges' to the y-axis & 'tenure' to the 'x-axis'

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 [35]:

```
import pandas as pd
```

In [37]:

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

In [ ]:

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

Out[ ]:

	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	...	

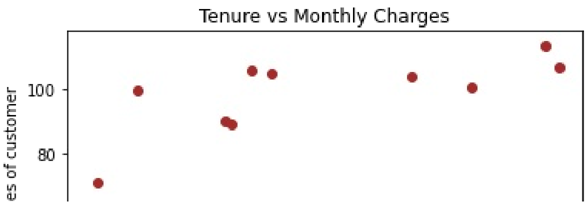
2 rows × 21 columns

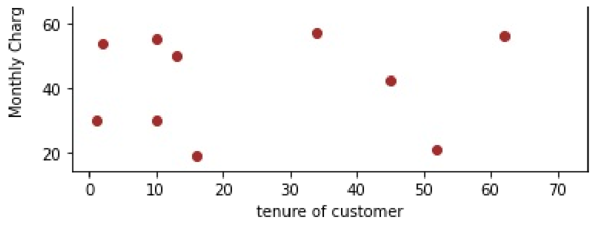
In [38]:

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

In [39]:

```
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()
```





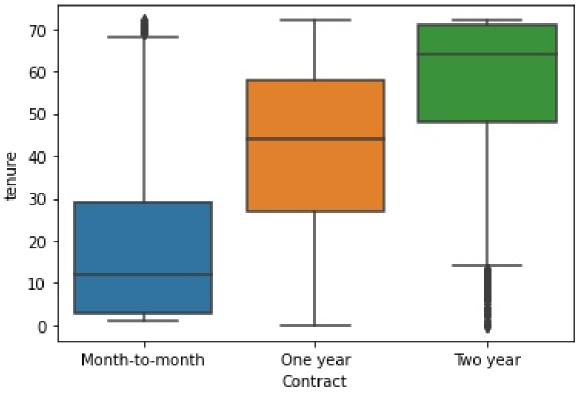
d)built a boxplot between tenure & contract. Map 'tenure' on y-axis and 'Contract' on the x-axis

In [40]:

```
import seaborn as sns
```

In [41]:

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

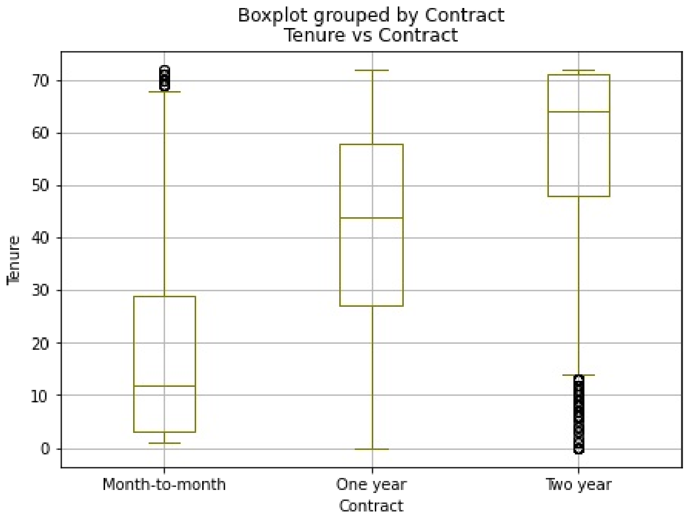


In [4]:

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

In [42]:

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



Linear Regression

A)Built a simple linear model where dependent variable is 'MonthlyCharges' and independent variable is 'tenure' i)Divide the dataset into train and test sets is 70:30 ratio

ii)built the model on train set and predict the values on test set

iii)After predicting the value 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 [43]:

```
from sklearn.model_selection import train_test_split
```

In [44]:

```
from sklearn.linear_model import LinearRegression
```

In [45]:

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

Out[45]:

	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 [46]:

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

In [47]:

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

In [50]:

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

In [51]:

```
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 [52]:

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

In [55]:

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

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]:

```
lr=LinearRegression()
```

In [58]:

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

```
Out[58]: LinearRegression()
```

```
In [60]: y_pred=lr.predict(x_test)
```

```
In [61]: y_pred
```

```
Out[61]: array([60.95089608, 72.98096699, 59.1903979 , ..., 75.62171426,
                70.63363608, 65.6455579 ])
```

```
In [62]: y_test.values
```

```
Out[62]: array([ 58.2 , 116.6 ,  71.95, ..., 109.95,  24.55,  81.6 ])
```

```
In [63]: from sklearn.metrics import mean_squared_error
         import numpy as np
```

```
In [64]: msc=mean_squared_error(y_test,y_pred)
```

```
In [65]: error=np.sqrt(msc)
```

```
In [66]: error
```

```
Out[66]: 29.394584027273893
```

D)Logistic Regression

a)Built a simple logistic regressin model where dependent variable is 'churn' & independent variable is 'MonthlyCharges'

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 [67]: from sklearn.linear_model import LogisticRegression
```

```
In [68]: x=pd.DataFrame(data['MonthlyCharges'])
```

```
In [69]: y=data['Churn']
```

```
In [70]: x_train,x_test,y_train,y_test=train_test_split(x,y,train_size=0.65,random_state=0)
```

```
In [71]: logmodel=LogisticRegression()
```

```
In [72]: logmodel.fit(x_train,y_train)
```

```
Out[72]: LogisticRegression()
```

```
In [73]: y_pred=logmodel.predict(x_test)
```

In [74]:

```
y_pred
```

Out[74]:

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

In [75]:

```
y_test.values
```

Out[75]:

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

In [77]:

```
from sklearn.metrics import confusion_matrix,accuracy_score
```

In [78]:

```
confusion_matrix(y_pred,y_test)
```

Out[78]:

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

In [79]:

```
(1815+0)/(1815+651+0+0)
```

Out[79]:

```
0.7360097323600974
```

In [80]:

```
accuracy_score(y_pred,y_test)
```

Out[80]:

```
0.7360097323600974
```

b)Built a multiple logistic regression model where dependent variable is 'Churn' & independent variable are 'tenure' & 'MonthlyCharges'

i)Divide the dataset in 80:20 ratio

ii)Built the model on train set and predict the values on test set

iii)Built the confusion matrix and get accuracy score

In [81]:

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

In [82]:

```
y=data['Churn']
```

In [83]:

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

In [84]:

```
mlr=LogisticRegression()
```

In [86]:

```
mlr.fit(x_train,y_train)
```

Out[86]:

```
LogisticRegression()
```

In [87]:

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

In [88]:

```
y_pred
```

Out[88]:

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

In [89]:

```
y_test.values
```

Out[89]:

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

In [53]:

```
from sklearn.metrics import confusion_matrix,accuracy_score
```

In [90]:

```
print(confusion_matrix(y_pred,y_test))
```

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

In [91]:

```
(934+156)/(934+156+212+107)
```

Out[91]:

```
0.7735982966643009
```

In [92]:

```
accuracy_score(y_test,y_pred)
```

Out[92]:

```
0.7735982966643009
```

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 [93]:

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

In [94]:

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

In [95]:

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

In [96]:

```
# 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 [97]:

```
y_pred
```

Out[97]:

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

In [ ]:

```
y_test
```

Out[ ]:

```
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 [98]:

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

In [99]:

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

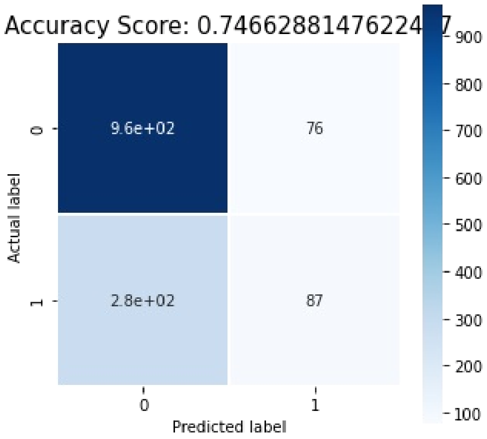
```
Accuracy: 0.7466288147622427
```

In [100…

```
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[100…

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



In [125…

```
!pip install graphviz

!pip install pydotplus
```

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

In [126…

```
pip install.scikit-learn==0.20.3
```

```
ERROR: unknown command "install.scikit-learn==0.20.3"
Note: you may need to restart the kernel to use updated packages.
```

In [129…

```
from sklearn.tree import export_graphviz
from sklearn.externals.six import StringIO
from IPython.display import Image
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')
```

```
Image(graph.create_png())
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
<ipython-input-129-df236d1b4dd6> in <module>
      1 from sklearn.tree import export_graphviz
----> 2 from sklearn.externals.six import StringIO
      3 from IPython.display import Image
      4 import pydotplus
      5 

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

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 [114…

```
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 [115…

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

#Create a Random Forest Classifier
clf1=RandomForestClassifier(n_estimators=10)

#Train the model using the training sets y_pred=clf.predict(X_test)
clf1.fit(X_train,y_train)

y_pred=clf.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))
```

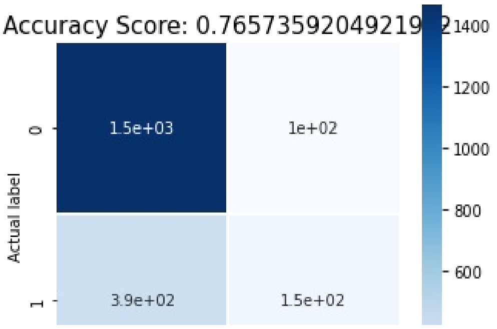
```
Accuracy: 0.7501183151916706
```

In [ ]:

```
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[ ]:

```
Text(0.5, 1.0, 'Accuracy Score: 0.7657359204921912')
```



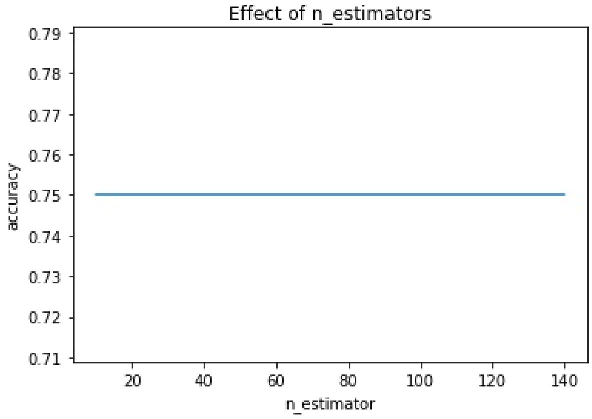
In [116…

```
# Try different numbers of n_estimators
estimators = np.arange(10, 150, 10)
accuracy = []

for n in estimators:
    clf1.set_params(n_estimators=n)
    clf1.fit(X_train, y_train)
    y_pred=clf.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 0x7fcf8d3fbd30>]
```



In [ ]:

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