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 [120]:

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

In [121]:

```
df=pd.read_csv("D:\AA-DATA SCIENCE\DATASET FOR DS IN R\customer_churn.csv")
```

In [122]:

```
df.columns
```

Out[122]:

```
Index(['customerID', 'gender', 'SeniorCitizen', 'Partner', 'Dependents',
       'tenure', 'PhoneService', 'MultipleLines', 'InternetService',
       'OnlineSecurity', 'OnlineBackup', 'DeviceProtection', 'TechSupport',
       'StreamingTV', 'StreamingMovies', 'Contract', 'PaperlessBilling',
       'PaymentMethod', 'MonthlyCharges', 'TotalCharges', 'Churn'],
      dtype='object')
```

In [123]:

```
#. Extract the 5th column & store it in ‘customer_5’
customer_5=df.iloc[:,4]
```

In [124]:

```
#b. Extract the 15th column & store it in ‘customer_15’
customer_15=df.iloc[:,14]
```

In [125]:

```
#c. Extract all the male senior citizens whose Payment Method is Electronic check &
#store the result in ‘senior_male_electronic’
senior_male_electronic=df[(df["gender"]=="Male")& (df['PaymentMethod']=="Electronic check")
```

In [126]:

```
#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’
customer_total_tenure=df[(df["tenure"]>70) | (df['MonthlyCharges']>100)]
```

In [127]:

```
#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=df[(df["Contract"]=="Two year") & (df['PaymentMethod']=="Mailed check") & (df[
```

In [128]:

```
#f. Extract 333 random records from the customer_churndataframe& store the result in'custom
customer_333=df.sample(333)
```

In [129]:

```
#g. Get the count of different levels from the 'Churn' column
df["Churn"].value_counts()
```

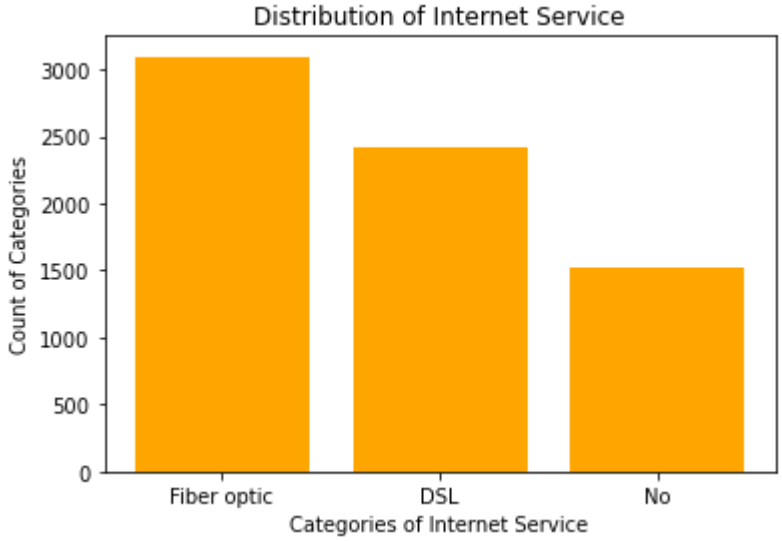
Out[129]:

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

B) Data Visualization: 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 [130]:

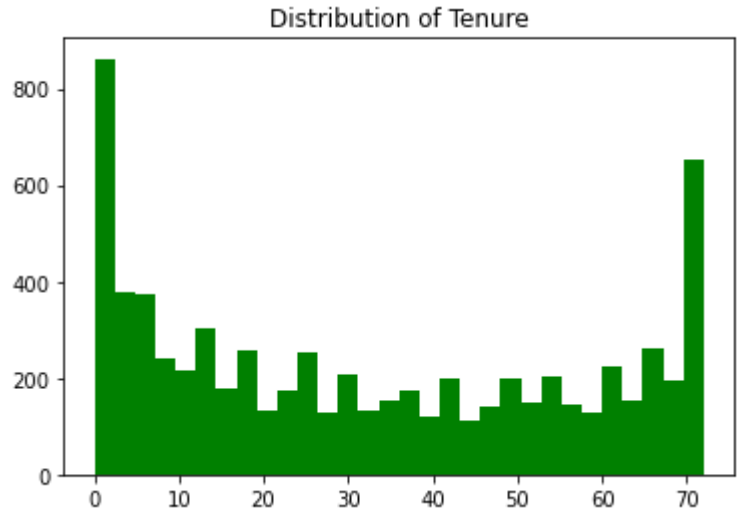
```
plt.bar(df['InternetService'].value_counts().keys().tolist(),df['InternetService'].value_co
plt.xlabel("Categories of Internet Service")
plt.ylabel("Count of Categories")
plt.title("Distribution of Internet Service")
plt.show()
```



b. Build a histogram for the 'tenure' column: i. Set 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 [131]:

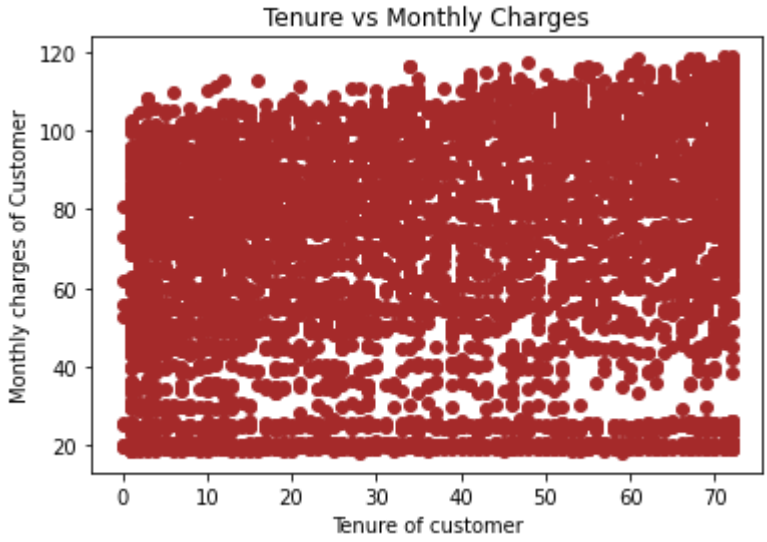
```
plt.hist(df['tenure'],color="green",bins=30)
plt.title("Distribution of Tenure")
plt.show()
```



c. Build 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 'Tenure vs Monthly Charges'

In [132]:

```
plt.scatter(x=df['tenure'],y=df['MonthlyCharges'],color='brown')
plt.xlabel("Tenure of customer")
plt.ylabel("Monthly charges of Customer")
plt.title("Tenure vs Monthly Charges")
plt.show()
```



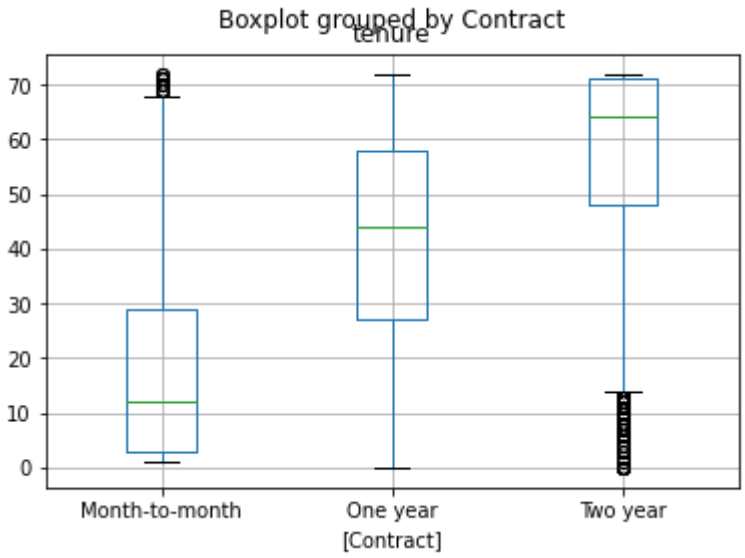
d. Build a box-plot between 'tenure' & 'Contract'. Map 'tenure' on the y-axis & 'Contract' on the x-axis.

In [133]:

```
df.boxplot(column='tenure',by=['Contract'])
```

Out[133]:

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



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 [134]:

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

In [135]:

```
x=pd.DataFrame(df['tenure'])
y=pd.DataFrame(df["MonthlyCharges"])
```

In [136]:

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

In [137]:

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

In [138]:

```
slp=LinearRegression()
```

In [139]:

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

Out[139]:

```
LinearRegression()
```

In [140]:

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

In [141]:

```
y_pred
```

Out[141]:

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

In [142]:

```
from sklearn.metrics import mean_squared_error
mse=mean_squared_error(y_pred,y_test)
rmse=np.sqrt(mse)
```

In [143]:

```
rmse
```

Out[143]:

```
29.394584027273893
```

Logistic Regression: a. Build a simple logistic regression modelwhere 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 [144]:

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

In [145]:

```
X=pd.DataFrame(df["MonthlyCharges"])
Y=pd.DataFrame(df["Churn"])
```

In [146]:

```
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.35,random_state=1)
```

In [147]:

```
print(X_train.size, X_test.size,Y_train.size,Y_test.size)
```

```
4577 2466 4577 2466
```

In [148]:

```
logmodel=LogisticRegression()
logmodel.fit(X_train,Y_train)
```

```
F:\conda\lib\site-packages\sklearn\utils\validation.py:72: DataConversionWar
ning: A column-vector y was passed when a 1d array was expected. Please chan
ge the shape of y to (n_samples, ), for example using ravel().
  return f(**kwargs)
```

Out[148]:

```
LogisticRegression()
```

In [149]:

```
Y_pred=logmodel.predict(X_test)
```

In [150]:

```
Y_pred
```

Out[150]:

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

In [151]:

```
print("Accuracy: ",logmodel.score(X_test,Y_test)*100,"%")
```

```
Accuracy:  74.61476074614761 %
```

In [152]:

```
from sklearn.metrics import confusion_matrix
cf=confusion_matrix(Y_test,Y_pred)
```

In [153]:

```
cf
```

Out[153]:

```
array([[1840,    0],
       [ 626,    0]], dtype=int64)
```

In [154]:

```
acc=1840/(1840+626)
```

In [155]:

```
print("Accuracy of the model =",acc*100,"%")
```

```
Accuracy of the model = 74.61476074614761 %
```

In [156]:

```
X=df[["tenure", "MonthlyCharges"]]
Y=df["Churn"]
```

In [157]:

```
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.20,random_state=1)
```

In [158]:

```
print(X_train.size//2, X_test.size//2,Y_train.size,Y_test.size)
```

```
5634 1409 5634 1409
```

In [159]:

```
logmodel=LogisticRegression()
logmodel.fit(X_train,Y_train)
```

Out[159]:

```
LogisticRegression()
```

In [160]:

```
Y_pred=logmodel.predict(X_test)
```

In [161]:

```
Y_pred
```

Out[161]:

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

In [162]:

```
l=Y_pred==Y_test
```

In [163]:

```
l
```

Out[163]:

```
3381     True
6180     True
4829     True
3737    False
4249     True
        ...
2563     True
2028     True
2899     True
3474     True
5154     True
Name: Churn, Length: 1409, dtype: bool
```

In [164]:

```
import collections
r=collections.Counter(l)
```

In [165]:

```
r
```

Out[165]:

```
Counter({True: 1123, False: 286})
```

In [166]:

```
acc=r[True]/(r[True]+r[False])
```

In [167]:

```
acc*=100
```

In [168]:

```
print("Accuracy of the model= ",acc,"%d")
```

```
Accuracy of the model=  79.70191625266146 %d
```

In [169]:

```
print("Accuracy: %d",logmodel.score(X_test,Y_test)*100)
```

```
Accuracy: %d 79.70191625266146
```

In [170]:

```
from sklearn.metrics import confusion_matrix
cf=confusion_matrix(Y_test,Y_pred)
```

In [171]:

```
cf
```

Out[171]:

```
array([[965,  96],
       [190, 158]], dtype=int64)
```

In [172]:

```
from sklearn.metrics import classification_report
print(classification_report(Y_test,Y_pred))
```

```
              precision    recall  f1-score   support

          No       0.84      0.91      0.87      1061
         Yes       0.62      0.45      0.52       348

    accuracy                           0.80      1409
   macro avg       0.73      0.68      0.70      1409
weighted avg       0.78      0.80      0.79      1409
```

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 [173]:

```
import pandas as pd
import numpy as np
d=pd.read_csv("D:\AA-DATA SCIENCE\DATASET FOR DS IN R\customer_churn.csv")
x=pd.DataFrame(d["tenure"])
y=d["Churn"]
```

In [174]:

```
x
```

Out[174]:

	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 [175]:

```
y
```

Out[175]:

```
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 [176]:

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

In [177]:

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=1)
```

In [178]:

```
y_test.size
```

Out[178]:

```
1409
```

In [179]:

```
from sklearn.tree import DecisionTreeClassifier
reg=DecisionTreeClassifier()
reg.fit(x_train,y_train)
```

Out[179]:

```
DecisionTreeClassifier()
```

In [180]:

```
reg
```

Out[180]:

```
DecisionTreeClassifier()
```

In [181]:

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

In [182]:

```
y_pred
```

Out[182]:

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

In [183]:

```
y_pred.size
```

Out[183]:

```
1409
```

In [184]:

```
from sklearn.metrics import classification_report,confusion_matrix,accuracy_score
print(confusion_matrix(y_test,y_pred))
print("Accuracy of the Model is ",accuracy_score(y_test,y_pred)*100,"%")
#r=accuracy_score(y_test,y_pred)
#print("accuracy of the model=",r)
```

```
[[983  78]
 [254  94]]
Accuracy of the Model is  76.43718949609652 %
```

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 [185]:

```
X=df[["tenure", "MonthlyCharges"]]
Y=df["Churn"]
```

In [186]:

```
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.30)
```

In [187]:

```
from sklearn.ensemble import RandomForestClassifier
```

In [188]:

```
cl=RandomForestClassifier(n_estimators=100)
cl.fit(X_train,Y_train)
```

Out[188]:

```
RandomForestClassifier()
```

In [189]:

```
y_pred=cl.predict(X_test)
```

In [190]:

```
y_pred
```

Out[190]:

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

In [191]:

```
l=y_pred==Y_test
```

In [192]:

```
l
```

Out[192]:

```
5718     True
4638     True
4768     True
6975     True
1470     True
        ...
5580     True
3667     True
4831     True
5311    False
5056     True
Name: Churn, Length: 2113, dtype: bool
```

In [193]:

```
import collections
r=collections.Counter(l)
```

In [194]:

```
r
```

Out[194]:

```
Counter({True: 1581, False: 532})
```

In [110]:

```
acc=r[True]/Y_test.size
```

In [111]:

```
acc
```

Out[111]:

```
0.7487579843860894
```

In [112]:

```
print("Accuracy of the model=",acc*100,"%")
```

```
Accuracy of the model= 74.87579843860894 %
```

In [113]:

```
from sklearn import metrics,confusion_matrix
print("Accuraccy= ",metrics.accuracy_score(Y_test,y_pred)*100,"%")
```

```
Accuraccy=  74.87579843860894 %
```

In [115]:

```
r1=confusion_matrix(Y_test,y_pred)
```

In [116]:

```
r1
```

Out[116]:

```
array([[898, 163],
       [191, 157]], dtype=int64)
```

In [117]:

```
accr=(r1[0][0]+r1[1][1])/r1.sum()
```

In [118]:

```
accr*=100
```

Out[118]:

```
0.7487579843860894
```

In [119]:

```
print("Accuracy of the model= ",accr,"%")
```

```
Accuracy of the model=  0.7487579843860894 %
```

Thank You