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

In [2]:
```python
cust_churn = pd.read_csv('customer_churn.csv')
```

In [3]:
```python
cust_churn.head()
```

Out[3]:

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

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

a. Extract the 5th column & store it in 'customer_5'

In [4]:
```python
c_5=cust_churn.iloc[:, 4]
c_5.head()
```

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

b. Extract the 15th column & store it in 'customer_15'

In [5]:
```python
c_15 = cust_churn.iloc[:, 15]
c_15.head()
```

Out[5]:
```
0    Month-to-month
1          One year
2    Month-to-month
3          One year
4    Month-to-month
Name: Contract, dtype: object
```

c. Extract all the male senior citizens whose Payment Method is Electronic check & store the result in 'senior_male_electronic'

In [6]:
```python
(cust_churn['gender']  == 'male') & (cust_churn['SeniorCitizen'] == 1) & (cust_churn['PaymentMethod'] ==
```

Out[6]:
```
0       False
1       False
2       False
3       False
4       False
        ...
7038    False
7039    False
7040    False
7041    False
7042    False
Length: 7043, dtype: bool
```

In [7]:
```python
c_random = cust_churn[(cust_churn['gender']  == 'Male') & (cust_churn['SeniorCitizen'] == 1) & (cust_chu
```

In [8]:
```python
c_random.head()
```

Out[8]:

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

5 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 [9]:
```python
customer_total_tenure = cust_churn[(cust_churn['tenure'] > 70) | (cust_churn['MonthlyCharges'] > 100)]
```

In [10]:
```python
customer_total_tenure.head()
```

Out[10]:

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

5 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 [11]:
```python
(cust_churn['Contract']  == 'Two year') & (cust_churn['PaymentMethod'] == 'Mailed check') & (cust_churn[
```

Out[11]:
```
0       False
1       False
2       False
3       False
4       False
        ...
7038    False
7039    False
7040    False
7041    False
7042    False
Length: 7043, dtype: bool
```

In [12]:
```python
two_mail_yes = cust_churn[(cust_churn['Contract']  == 'Two year') & (cust_churn['PaymentMethod'] == 'Ma
```

In [13]:
```python
two_mail_yes.head()
```

Out[13]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity
268	6323-AYBRX	Male	0	No	No	59	Yes	No	No	No internet service
5947	7951-QKZPL	Female	0	Yes	Yes	33	Yes	Yes	No	No internet service
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_churndataframe& store the result in 'customer_333'

In [14]:
```python
c_333 = cust_churn.sample(n=333)
```

In [15]:
```python
c_333.head()
```

Out[15]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity
1298	5989-OMNJE	Female	0	Yes	Yes	24	No	No phone service	DSL	Yes
1948	8183-ONMXC	Female	0	No	No	2	Yes	No	Fiber optic	No
4339	6992-TKNYO	Male	0	Yes	No	38	Yes	Yes	DSL	No
194	2146-EGVDT	Male	0	Yes	Yes	59	Yes	No	No	No internet service
2749	9086-YJYXS	Male	0	Yes	Yes	34	Yes	Yes	DSL	No

5 rows × 21 columns

g. Get the count of different levels from the 'Churn' column

In [16]:
```python
cust_churn['Churn'].value_counts()
```

Out[16]:
```
No     5174
Yes    1869
Name: Churn, dtype: int64
```

In [17]:
```python
cust_churn['Contract'].value_counts()
```

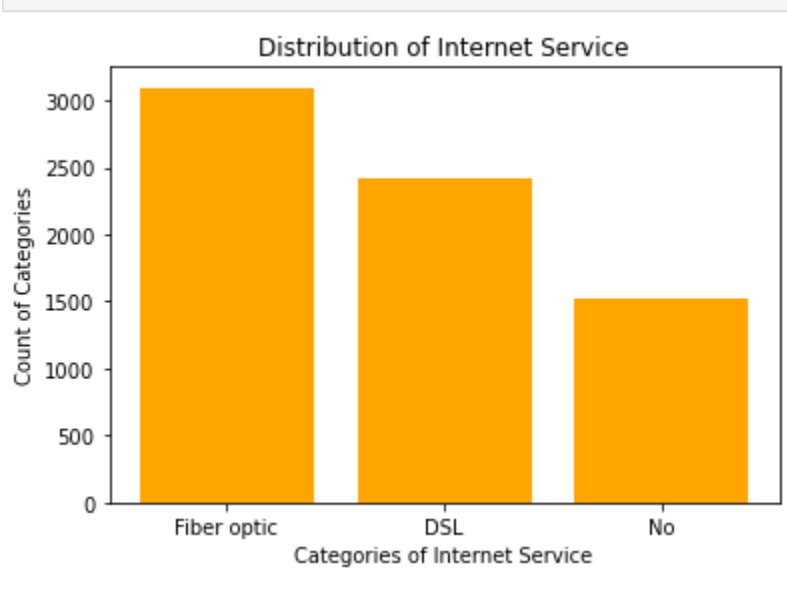
Out[17]:
```
Month-to-month    3875
Two year          1695
One year          1473
Name: Contract, dtype: int64
```

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 [18]:
```python
plt.bar(cust_churn['InternetService'].value_counts().keys().tolist(),cust_churn['InternetService'].value

plt.xlabel('Categories of Internet Service')
plt.ylabel('Count of Categories')
plt.title('Distribution of Internet Service')
plt.show()
```

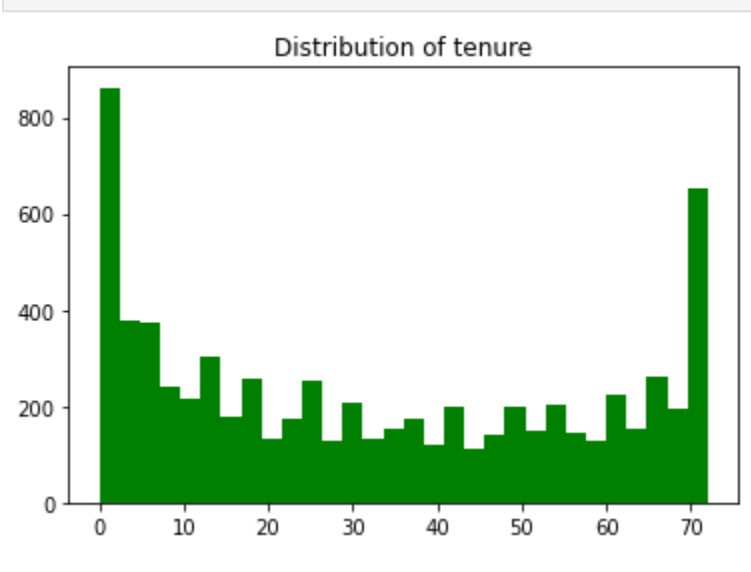


In [19]:
```python
cust_churn['InternetService'].value_counts().tolist()
```

Out[19]:
```
[3096, 2421, 1526]
```

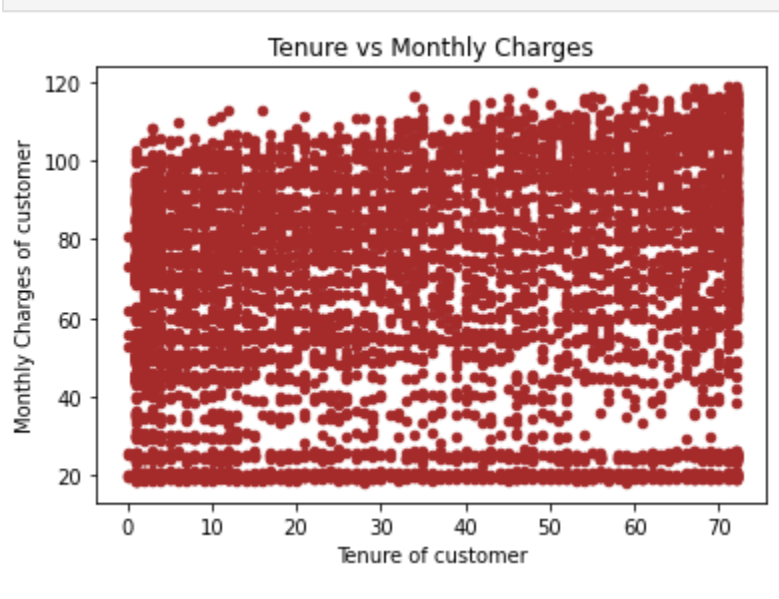
b.Build a histogram for the 'tenure' column: i. Set the number of bins to be 30 ii. Set the color of the bins to be 'green' iii. Assign the title 'Distribution of tenure'

In [20]:
```python
plt.hist(cust_churn['tenure'], bins=30, color = 'Green')
plt.title('Distribution of tenure')
plt.show()
```



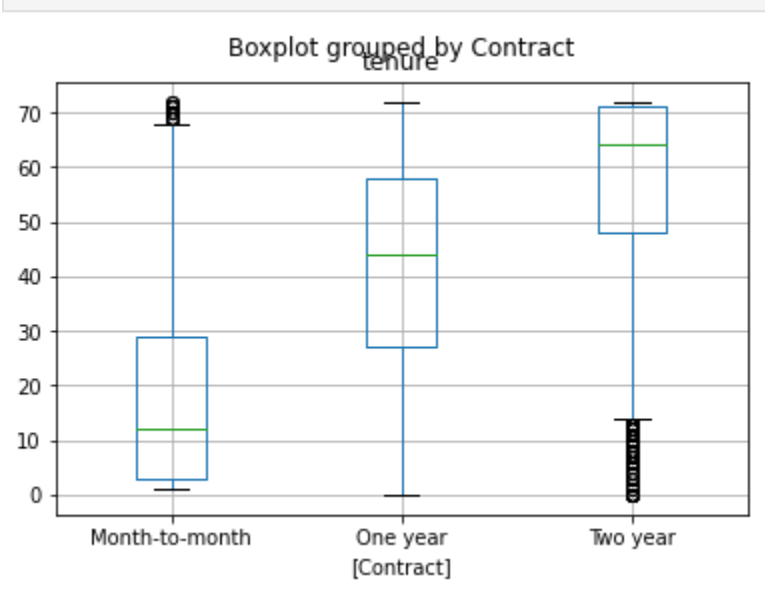
c. Build a scatter-plot between 'MonthlyCharges' & 'tenure'. Map 'MonthlyCharges' to the y-axis & 'tenure' to the 'x-axis': i. Assign the points a color of 'brown' ii. Set the x-axis label to 'Tenure of customer' iii. Set the y-axis label to 'Monthly Charges of customer' iv. Set the title to 'Tenure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

In [21]:
```python
cust_churn.plot(kind='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()
```



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

In [22]:
```python
cust_churn.boxplot(column=['tenure'], by=['Contract'])
plt.show()
```



Machine Learning

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 [23]:
```python
from sklearn import linear_model
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split

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

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

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

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

In [26]:
```python
# fit the model on top of training set
regressor = LinearRegression()

regressor.fit(x_train,y_train)
```

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

In [27]:
```python
#oredicting value on top of x_test
y_pred=regressor.predict(x_test)

# after calculating rmse

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]:
```python
#rmse --> lower the value of rmse better your model
from sklearn.metrics import mean_squared_error

rmse=np.sqrt(mean_squared_error(y_test, y_pred))

rmse
```

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

D) 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. iii. Build the model on train set and predict the values on test set Build the confusion matrix and get the accuracy score

In [30]:
```python
x = cust_churn[['MonthlyCharges']]
y=cust_churn[['tenure']]
```

In [31]:
```python
x_train, x_test, y_train, y_test=train_test_split(x,y, test_size=0.30, random_state=0)
```

In [32]:
```python
from sklearn.linear_model import LogisticRegression

log_model = LogisticRegression()

log_model.fit(x_train,y_train)
```

```
C:\Users\Sanket Hanjage\Downloads\anaconda\lib\site-packages\sklearn\utils\validation.py:993: DataConve
rsionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y t
o (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
C:\Users\Sanket Hanjage\Downloads\anaconda\lib\site-packages\sklearn\linear_model\_logistic.py:814: Con
vergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

Out[32]:
```
LogisticRegression()
```

In [33]:
```python
y_pred = log_model.predict(x_test)
```

In [34]:
```python
from sklearn.metrics import confusion_matrix, accuracy_score
```

In [35]:
```python
confusion_matrix(y_test, y_pred), accuracy_score(y_test, y_pred)
```

Out[35]:
```
(array([[  0,   3,   0, ...,   0,   0,   0],
        [  0, 170,   0, ...,   0,   0,  10],
        [  0,  73,   0, ...,   0,   0,   5],
        ...,
        [  0,  21,   0, ...,   0,   0,  26],
        [  0,  26,   0, ...,   0,   0,  24],
        [  0,  39,   0, ...,   0,   0,  73]], dtype=int64),
 0.11500236630383341)
```

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 [36]:
```python
x = cust_churn[['MonthlyCharges','tenure']]
y=cust_churn[['tenure']]
```

In [37]:
```python
x_train, x_test, y_train, y_test=train_test_split(x,y, test_size=0.20, random_state=0)
```

In [38]:
```python
from sklearn.linear_model import LogisticRegression

log_model = LogisticRegression()

log_model.fit(x_train,y_train)
```

```
C:\Users\Sanket Hanjage\Downloads\anaconda\lib\site-packages\sklearn\utils\validation.py:993: DataConve
rsionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y t
o (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
C:\Users\Sanket Hanjage\Downloads\anaconda\lib\site-packages\sklearn\linear_model\_logistic.py:814: Con
vergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

Out[38]:
```
LogisticRegression()
```

In [39]:
```python
y_pred = log_model.predict(x_test)
```

In [40]:
```python
from sklearn.metrics import confusion_matrix, accuracy_score
```

In [41]:
```python
confusion_matrix(y_test, y_pred), accuracy_score(y_test, y_pred)
```

Out[41]:
```
(array([[  0,   3,   0, ...,   0,   0,   0],
        [  0, 122,   0, ...,   0,   0,   0],
        [  0,  30,   0, ...,   0,   0,   0],
        ...,
        [  0,   0,   0, ...,   0,   0,  26],
        [  0,   0,   0, ...,   0,   0,  29],
        [  0,   0,   0, ...,   0,   0,  77]], dtype=int64),
 0.1596877217885025)
```

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 [42]:
```python
x = cust_churn[['tenure']]
y = cust_churn[['Churn']]
```

In [43]:
```python
from sklearn.tree import DecisionTreeClassifier
x_train, x_test, y_train, y_test=train_test_split(x,y, test_size=0.20, random_state=0)
```

In [44]:
```python
my_tree = DecisionTreeClassifier()

my_tree.fit(x_train, y_train)
```

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

In [45]:
```python
# predict value

y_pred=my_tree.predict(x_test)
```

In [46]:
```python
from sklearn.metrics import confusion_matrix, accuracy_score
```

In [47]:
```python
confusion_matrix(y_test,y_pred)
```

Out[47]:
```
array([[965,  76],
       [281,  87]], dtype=int64)
```

In [48]:
```python
(965+87)/(965+87+76+281)
```

Out[48]:
```
0.7466288147622427
```

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 [49]:
```python
x = cust_churn[['tenure','MonthlyCharges']]
y = cust_churn[['Churn']]
```

In [50]:
```python
from sklearn.ensemble import RandomForestClassifier


rf = RandomForestClassifier()

rf.fit(x_train, y_train)
```

```
C:\Users\Sanket Hanjage\AppData\Local\Temp\ipykernel_23284\1775287623.py:6: DataConversionWarning: A co
lumn-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), fo
r example using ravel().
  rf.fit(x_train, y_train)
```

Out[50]:
```
RandomForestClassifier()
```

In [51]:
```python
rf.predict(x_test)
```

Out[51]:
```
array(['No', 'No', 'No', ..., 'No', 'No', 'Yes'], dtype=object)
```

In [53]:
```python
from sklearn.metrics import confusion_matrix, accuracy_score
```

In [54]:
```python
confusion_matrix(y_test,y_pred)
```

Out[54]:
```
array([[965,  76],
       [281,  87]], dtype=int64)
```

In [55]:
```python
accuracy_score(y_test, y_pred)
```

Out[55]:
```
0.7466288147622427
```

In [ ]: