In [4]:
```
#import the library Pandas
import pandas as pd
```

In [5]:
```
#Put the mtcars.csv dataset on Root Directory
#Read the dataset
data = pd.read_csv('mtcars.csv')
```

In [6]:
```
#Find the head of the dataset.
data.head()
```

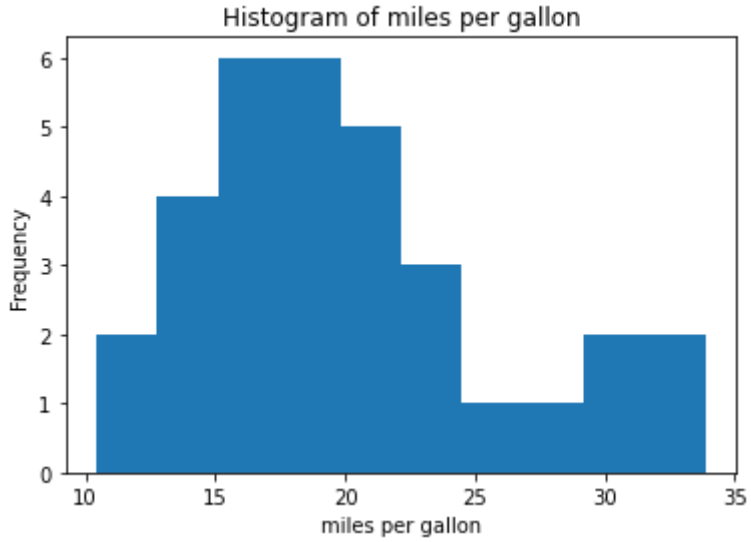
Out[6]:

	model	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
0	Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4
1	Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4	4
2	Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	1
3	Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	1
4	Hornet Sportabout	18.7	8	360.0	175	3.15	3.440	17.02	0	0	3	2

In [7]:
```
#Find the Datatype of Dataset (each column)
datatype = data.dtypes
datatype
```

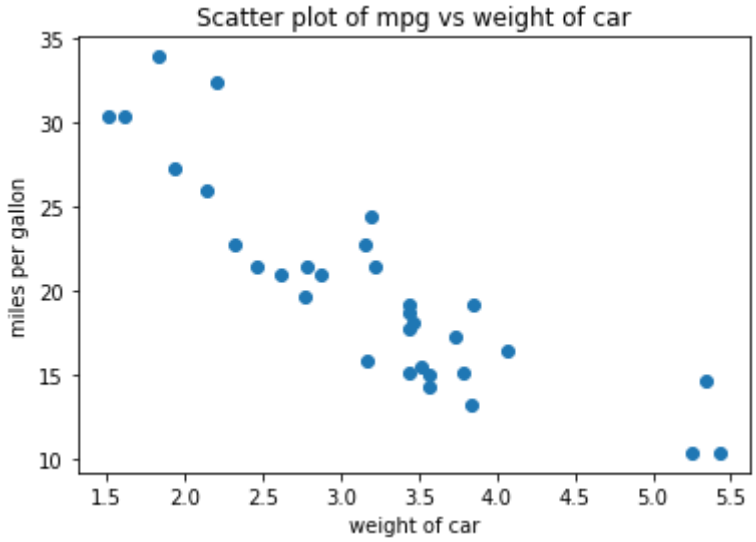
Out[7]:
```
model     object
mpg      float64
cyl        int64
disp     float64
hp         int64
drat     float64
wt       float64
qsec     float64
vs         int64
am         int64
gear       int64
carb       int64
dtype: object
```

In [8]:
```
#From the given dataset ‘mtcars.csv’, plot a histogram to check the freque
#the variable ‘mpg’ (Miles per gallon).
#Find the highest frequency of interval
import matplotlib.pyplot as plt
#=======================================
#         Histogram
#=======================================
plt.hist(data['mpg'],density = False)
plt.title('Histogram of miles per gallon')
plt.xlabel('miles per gallon')
plt.ylabel('Frequency')
plt.show()
```



In [9]:

```
#Which can be inferred from scatter plot of 'mpg' (Miles per gallon) vs 'wt
#=======================================
#           Scatter Plot
#=======================================
plt.scatter(data['wt'],data['mpg'])
plt.title('Scatter plot of mpg vs weight of car')
plt.xlabel('weight of car')
plt.ylabel('miles per gallon')
plt.show()
```



In [ ]:

In [1]:
```
#import the library Pandas
import pandas as pd
```

In [2]:
```
#Put the churn.csv dataset on Root Directory
#Read the dataset
churn = pd.read_csv('churn.csv')
```

In [3]:
```
#Showing the 5 Data of given dataset
churn.head()
```

Out[3]:

	Unnamed: 0	customerID	tenure	Contract	PaperlessBilling	PaymentMethod	MonthlyCharges
0	1	8260-NGFNY	One	Month-to-month	No	Mailed check	25.2
1	2	2359-QWQUL	39	One year	Yes	Credit card (automatic)	104.7
2	3	6598/RFFVI	2	One year	No	Credit card (automatic)	19.3
3	4	IXSTS-8780	6	Month-to-month	Yes	Electronic check	90.1
4	5	2674/MIAHT	Four	Month-to-month	Yes	Mailed check	80.3

5 rows × 22 columns

In [4]:
```
#Find the no. of duplicate records in the churn dataframe based on the cut
duplicate = churn[churn.duplicated(['customerID'],keep='first')]
duplicate.shape[0]
```

Out[4]: 7

In [5]:
```
#In the churn dataframe, what are the total no. of missing values for the
churn.TotalCharges.isnull().sum()
```

Out[5]: 15

In [9]:
```
#From the churn dataframe, what is the average monthly charge paid by a cus
churn.MonthlyCharges.mean()
```

Out[9]: 62.473481781376535

In [12]:
```
#In the churn dataframe, under the variable Dependents how many records hav
pd.crosstab(index=churn.Dependents, columns="count")
```

Out[12]:

col_0	count
Dependents	
1@#	6
No	171
Yes	80

In [10]:

```
#Find the data type of the variable tenure from the churn dataframe.
churn['tenure'].ftypes
```

Out[10]:

```
'object:dense'
```

In [ ]:

In [1]:
```
#import the library Pandas
import pandas as pd
```

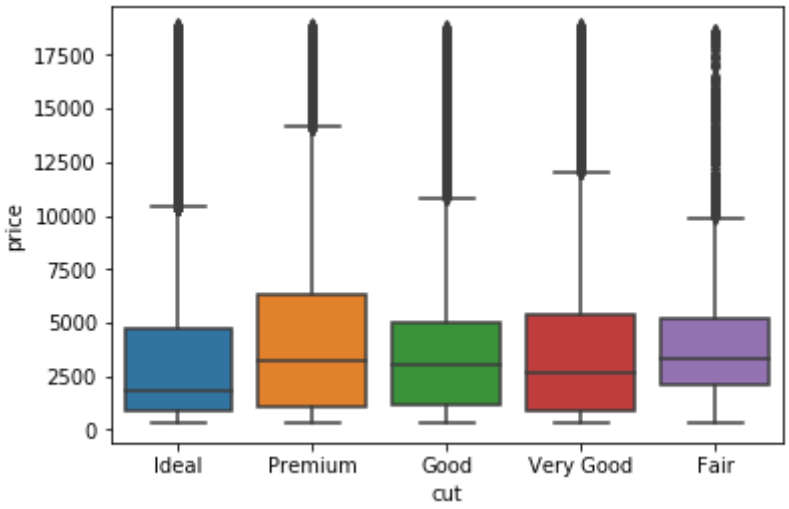
In [3]:
```
#Put the diamond.csv dataset on Root Directory
#Read the dataset
data1 = pd.read_csv('diamond.csv')
```

In [4]:
```
#Showing the 5 Data of given dataset
data1.head()
```

Out[4]:

	Unnamed: 0	carat	cut	color	clarity	depth	table	price	x	y	z
0	1	0.23	Ideal	E	SI2	61.5	55.0	326	3.95	3.98	2.43
1	2	0.21	Premium	E	SI1	59.8	61.0	326	3.89	3.84	2.31
2	3	0.23	Good	E	VS1	56.9	65.0	327	4.05	4.07	2.31
3	4	0.29	Premium	I	VS2	62.4	58.0	334	4.20	4.23	2.63
4	5	0.31	Good	J	SI2	63.3	58.0	335	4.34	4.35	2.75

In [6]:
```
#Plot a boxplot for “price” vs “cut” from the dataset “diamond.csv”
import matplotlib.pyplot as plt
import seaborn as sns
#=====================================
#     Box plot for two variables:
#=====================================
sns.boxplot(x=data1["cut"],y = data1["price"],data = data1)
plt.show()
```



In [7]:
```
#Which of the categories under “cut” have the highest median price?
data1.groupby('cut')['price'].median()
```

Out[7]:
```
cut
Fair         3282.0
Good         3050.5
Ideal        1810.0
Premium      3185.0
Very Good    2648.0
Name: price, dtype: float64
```

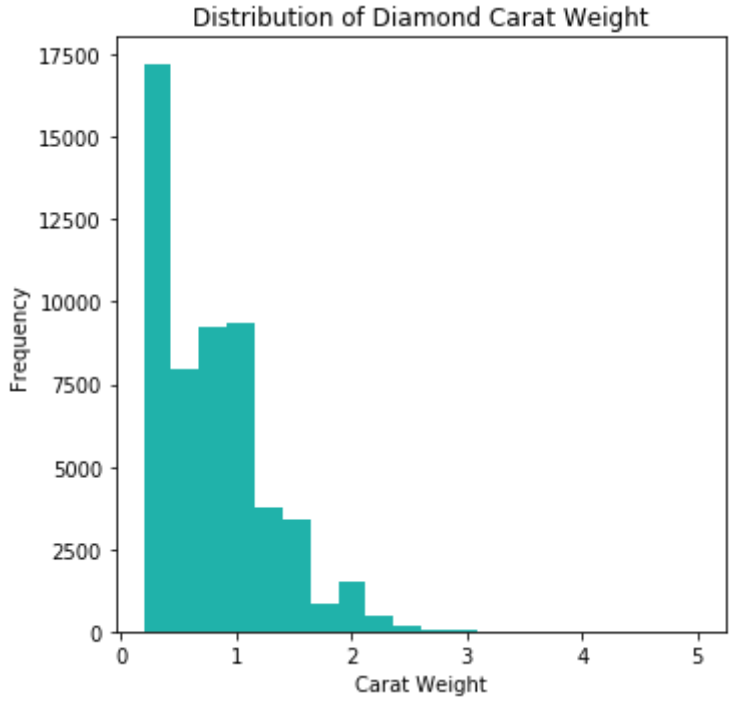
In [10]:
```
#Create a frequency table (one-way table) for the variable “cut” from the d
#What is the frequency for the cut type “Ideal”?
#====================================
#  Cross Tables: One Way Table
#====================================
pd.crosstab(index=data1['cut'],columns='count')
```

Out[10]:

col_0	count
cut	
Fair	1610
Good	4906
Ideal	21551
Premium	13791
Very Good	12082

In [11]:
```
#Show the subplot of the diamond carat weight distribution.
plt.figure(figsize=[12,12])

plt.subplot(221)
plt.hist(data1['carat'],bins=20,color='lightseagreen')
plt.xlabel('Carat Weight')
plt.ylabel('Frequency')
plt.title('Distribution of Diamond Carat Weight')
```

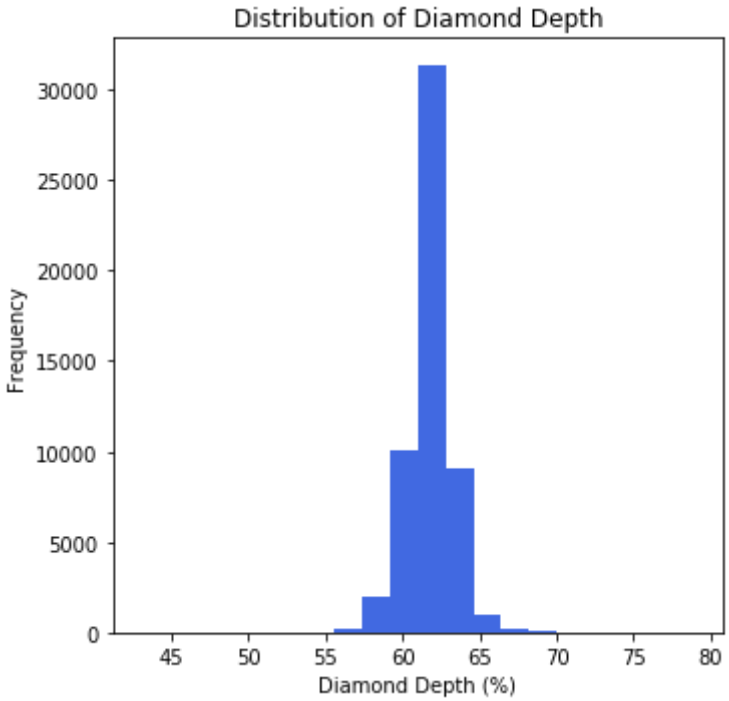
Out[11]:
```
Text(0.5, 1.0, 'Distribution of Diamond Carat Weight')
```



In [13]:

```
#Show the subplot of diamond depth distribution.
plt.figure(figsize=[12,12])

plt.subplot(222)
plt.hist(data1['depth'],bins=20,color='royalblue')
plt.xlabel('Diamond Depth (%)')
plt.ylabel('Frequency')
plt.title('Distribution of Diamond Depth')
```

Out[13]:
```
Text(0.5, 1.0, 'Distribution of Diamond Depth')
```



In [14]:
```
#Build the Model using linear regression and find the accuracy.
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn import datasets
from sklearn.linear_model import LinearRegression
from sklearn.metrics import accuracy_score
```

In [16]:
```
# Creating categorical variables for 'cut', 'color', and 'clarity'
df_final = pd.get_dummies(data1, columns=["cut", "color", "clarity"])
```

In [17]:
```
#Split the data into test and training datasets and explore the optimum pr
test_data = df_final.iloc[-round(len(df_final)*.1):].copy()
df_final.drop(df_final.index[-round(len(df_final)*.1):],inplace=True)
test_data.drop('price',1,inplace=True)
print(df_final.shape)
print(test_data.shape)
```

```
(48546, 28)
(5394, 27)
```

In [18]:
```
X = df_final.drop(['price'],1)
y = df_final['price']
```

In [19]:
```
#Spilting data with test size = 0.2
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.2)
```

In [20]:
```
model = LinearRegression()
model.fit(X_train,y_train)
model.score(X_test,y_test)
```

Out[20]:
```
0.9193301281260476
```

In [21]:
```
#Spilting data with test size = 0.3
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.3)
```

In [22]:
```
model = LinearRegression()
model.fit(X_train,y_train)
model.score(X_test,y_test)
```

Out[22]:
```
0.9257343915295821
```

In [23]:
```
#Spilting data with test size = 0.4
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.4)
```

In [24]:
```
model = LinearRegression()
model.fit(X_train,y_train)
model.score(X_test,y_test)
```

Out[24]:
```
0.9220422283679701
```

In [25]:
```
#Spilting data with test size = 0.5
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.5)
```

In [26]:
```
model = LinearRegression()
model.fit(X_train,y_train)
model.score(X_test,y_test)
```

Out[26]:
```
0.921202984506459
```

In [ ]:

In [1]:
```
#import the library Pandas
import pandas as pd
```

In [2]:
```
#Put the People Charm case.csv dataset on Root Directory
#Read the dataset
data = pd.read_csv('People Charm case.csv')
```

In [3]:
```
#Showing the 5 Data of given dataset
data.head()
```

Out[3]:

	satisfactoryLevel	lastEvaluation	numberOfProjects	avgMonthlyHours	timeSpent.company	w
0	0.38	0.53	2	157	3	
1	0.80	0.86	5	262	6	
2	0.11	0.88	7	272	4	
3	0.37	0.52	2	159	3	
4	0.41	0.50	2	153	3	

In [4]:
```
#Which of the variables have missing values?
#=====================================
#   Checking for Missing Value
#=====================================
print('Data Column with null Values:\n',data.isnull().sum())
```

```
Data Column with null Values:
 satisfactoryLevel        0
lastEvaluation            0
numberOfProjects          0
avgMonthlyHours           0
timeSpent.company         0
workAccident              0
left                      0
promotionInLast5years     0
dept                      0
salary                    0
dtype: int64
```

In [7]:
```
#What is the third quartile value for the variable “lastEvaluvation”?
#=====================================
#   Third Quartile Value
#=====================================
summary_num = data.describe()
print(summary_num)
```

```
       satisfactoryLevel  lastEvaluation  numberOfProjects  avgMonthlyHours
\
count       14999.000000    14999.000000      14999.000000     14999.000000
mean            0.612834        0.716102          3.803054       201.050337
std             0.248631        0.171169          1.232592        49.943099
min             0.090000        0.360000          2.000000        96.000000
25%             0.440000        0.560000          3.000000       156.000000
50%             0.640000        0.720000          4.000000       200.000000
75%             0.820000        0.870000          5.000000       245.000000
max             1.000000        1.000000          7.000000       310.000000

       timeSpent.company  workAccident          left  promotionInLast5years
count       14999.000000  14999.000000  14999.000000           14999.000000
mean            3.498233      0.144610      0.238083               0.021268
std             1.460136      0.351719      0.425924               0.144281
min             2.000000      0.000000      0.000000               0.000000
25%             3.000000      0.000000      0.000000               0.000000
50%             3.000000      0.000000      0.000000               0.000000
75%             4.000000      0.000000      0.000000               0.000000
max            10.000000      1.000000      1.000000               1.000000
```

In [8]:

```
#Construct a Crosstable for the variables ‘dept’ and “salary”.
#find out which department has highest frequency value in the category low
pd.crosstab(index = data['dept'],columns = data['salary'])
```

Out[8]:

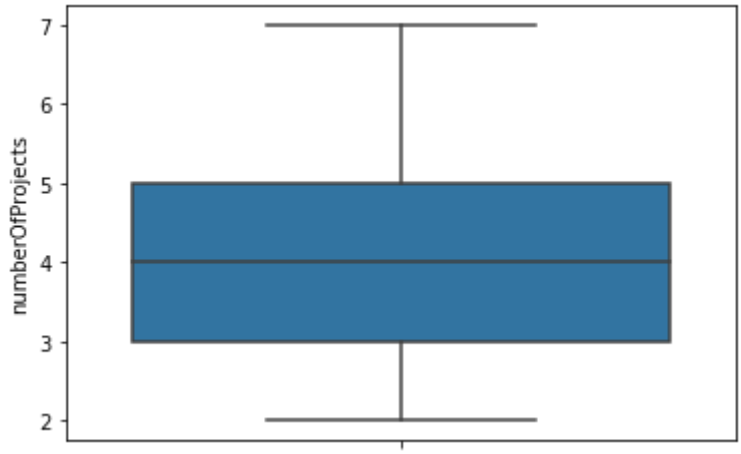
salary	high	low	medium
dept			
IT	83	609	535
RandD	51	364	372
accounting	74	358	335
hr	45	335	359
management	225	180	225
marketing	80	402	376
product_mng	68	451	383
sales	269	2099	1772
support	141	1146	942
technical	201	1372	1147

In [7]:

```
#Generate a boxplot for the variable “numberOfProjects”.
#get the median value for the number of projects where the employees have w
import matplotlib.pyplot as plt
import seaborn as sns

sns.boxplot(y=data["numberOfProjects"])
```

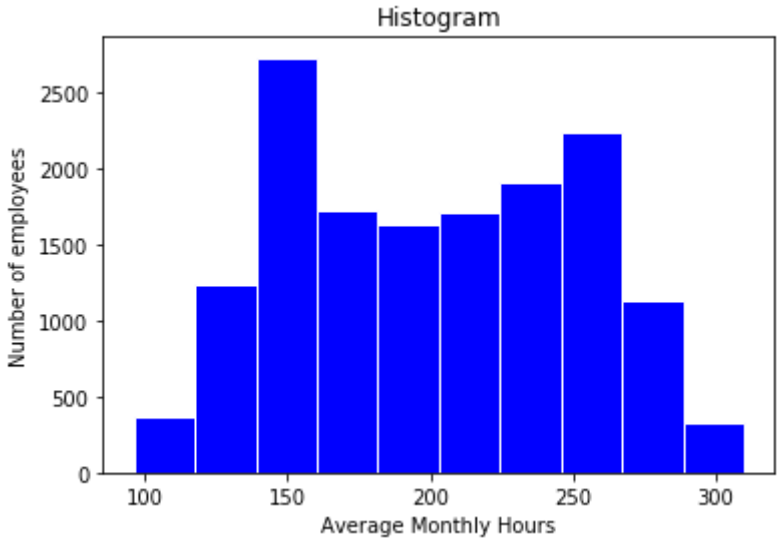
Out[7]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x241232ae550>
```



In [8]:

```
#Plot a histogram using the variable “avgMonthlyHours”.
#find the range in which the number of employees worked for 150 hours per m
plt.hist(data['avgMonthlyHours'],color ='blue',edgecolor='white',orientatio
plt.title('Histogram')
plt.xlabel('Average Monthly Hours')
plt.ylabel('Number of employees')
plt.show()
```

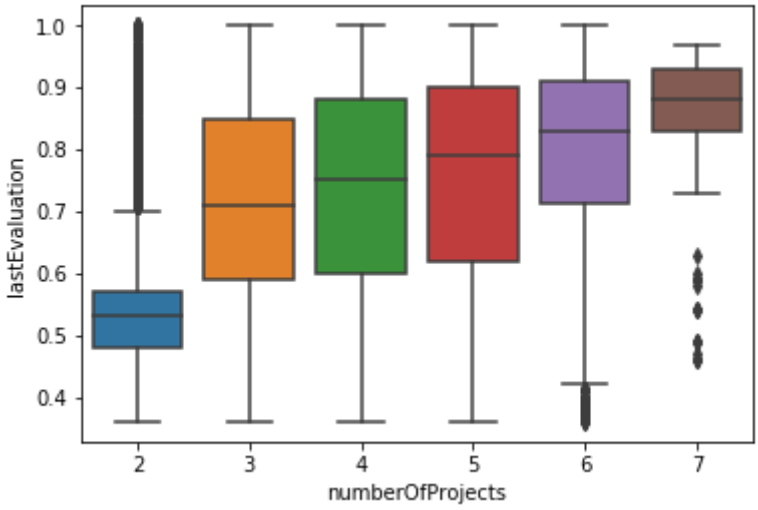


In [14]:

```
#Generate a boxplot for the variables “lastEvaluation” and “numberOfProject
sns.boxplot(x=data["numberOfProjects"], y = data["lastEvaluation"],data = d
```

Out[14]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1cacf647b38>
```


1.0
0.9
0.8
0.7
0.6
0.5
0.4
lastEvaluation
2
3
4
5
6
7
numberOfProjects

In [ ]:

In [2]:
```
#import the library Pandas
import pandas as pd
```

In [3]:
```
#Put the People Charm case.csv dataset on Root Directory
#Read the dataset
data = pd.read_csv('People Charm case.csv')
```

In [4]:
```
#Showing the 5 Data of given dataset
data
```

Out[4]:

	satisfactoryLevel	lastEvaluation	numberOfProjects	avgMonthlyHours	timeSpent.compan
0	0.38	0.53	2	157	
1	0.80	0.86	5	262	
2	0.11	0.88	7	272	
3	0.37	0.52	2	159	
4	0.41	0.50	2	153	
5	0.10	0.77	6	247	
6	0.92	0.85	5	259	
7	0.42	0.53	2	142	
8	0.45	0.54	2	135	
9	0.11	0.81	6	305	
10	0.36	0.56	2	137	
11	0.38	0.54	2	143	
12	0.45	0.47	2	160	
13	0.78	0.99	4	255	
14	0.76	0.89	5	262	
15	0.11	0.83	6	282	
16	0.09	0.95	6	304	
17	0.46	0.57	2	139	
18	0.40	0.53	2	158	
19	0.89	0.92	5	242	
20	0.82	0.87	4	239	
21	0.40	0.49	2	135	
22	0.38	0.50	2	132	
23	0.09	0.62	6	294	
24	0.45	0.57	2	134	
25	0.40	0.51	2	145	
26	0.84	0.87	4	246	
27	0.38	0.46	2	137	
28	0.45	0.50	2	126	
29	0.11	0.89	6	306	
...	...	...	...	...	.
14969	0.22	0.91	6	222	
14970	0.43	0.48	2	135	
14971	0.42	0.48	2	143	
14972	0.82	0.97	4	243	
14973	0.73	0.60	3	145	
14974	0.10	0.55	2	247	
14975	0.57	1.00	3	241	
14976	0.97	0.66	4	218	

	satisfactoryLevel	lastEvaluation	numberOfProjects	avgMonthlyHours	timeSpent.compan
14977	0.21	0.62	4	247	
14978	0.64	0.50	3	238	
14979	0.37	0.46	2	149	
14980	0.88	0.75	4	201	
14981	0.97	0.55	4	166	
14982	0.88	0.80	3	133	
14983	0.77	0.67	3	186	
14984	0.62	0.71	4	268	
14985	0.95	0.84	3	270	
14986	0.76	1.00	4	220	
14987	0.37	0.45	2	126	
14988	0.43	0.57	2	157	
14989	0.69	0.70	3	212	
14990	0.73	0.52	3	274	
14991	0.85	0.53	3	250	
14992	0.30	0.88	5	245	
14993	0.61	0.89	3	242	1
14994	0.11	0.85	7	275	
14995	0.99	0.83	4	274	
14996	0.72	0.72	4	175	
14997	0.24	0.91	5	177	
14998	0.77	0.83	6	271	

14999 rows × 10 columns

In [5]:
```
# To perform numerical operations
import numpy as np
# To Visualize data
import seaborn as sns
```

In [6]:
```
# To Partition the data
from sklearn.model_selection import train_test_split
# Importing library for logistic regression
from sklearn.linear_model import LogisticRegression
```

In [7]:
```
# Importing performance metrics - accuracy score & confusion matrix
from sklearn.metrics import accuracy_score,confusion_matrix
```

In [9]:
```
#=========================================
#   Logistic Regression Model
#=========================================
new_data = pd.get_dummies(data,drop_first = True)
columns_list = list(new_data.columns)
```

```
print(columns_list)

# Separating the input names from data
features=list(set(columns_list)-set(['left']))
print(features)

# Storing the output values in y
y=new_data['left'].values
print(y)

# Storing the values from input features
x = new_data[features].values
print(x)
# Spiliting the data into train and test
train_x,test_x,train_y,test_y = train_test_split(x,y,test_size=0.25,random_

# Make an instance of the Model
logistic = LogisticRegression()

# Fitting the Values for x and y
logistic.fit(train_x,train_y)

# Pridiction from test data
prediction = logistic.predict(test_x)

# Confusion matrix
confusion_matrix = confusion_matrix(test_y,prediction)
print(confusion_matrix)

# Calculating the accuracy
accuracy_score = accuracy_score(test_y,prediction)
print(accuracy_score)

# Printing the misclassified values from Prediction
print('Misclassified samples: %d' % (test_y != prediction).sum())
```

```
['satisfactoryLevel', 'lastEvaluation', 'numberOfProjects', 'avgMonthlyHour
s', 'timeSpent.company', 'workAccident', 'left', 'promotionInLast5years',
'dept_RandD', 'dept_accounting', 'dept_hr', 'dept_management', 'dept_market
ing', 'dept_product_mng', 'dept_sales', 'dept_support', 'dept_technical',
'salary_low', 'salary_medium']
['lastEvaluation', 'salary_low', 'dept_hr', 'satisfactoryLevel', 'promotion
InLast5years', 'avgMonthlyHours', 'dept_management', 'dept_RandD', 'dept_pr
oduct_mng', 'workAccident', 'dept_technical', 'dept_accounting', 'salary_me
dium', 'dept_marketing', 'numberOfProjects', 'timeSpent.company', 'dept_sup
port', 'dept_sales']
[1 1 1 ... 0 0 0]
[[0.53 1.   0.   ... 3.   0.   1.  ]
 [0.86 0.   0.   ... 6.   0.   1.  ]
 [0.88 0.   0.   ... 4.   0.   1.  ]
 ...
 [0.72 1.   0.   ... 4.   0.   0.  ]
 [0.91 1.   0.   ... 5.   0.   1.  ]
 [0.83 1.   0.   ... 3.   1.   0.  ]]
[[2687  181]
 [ 564  318]]
0.8013333333333333
Misclassified samples: 745
```

```
C:\Users\my pc\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.p
y:433: FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Sp
ecify a solver to silence this warning.
  FutureWarning)
```

In [10]:
```
# The accuracy score for the predicted model
print(accuracy_score)
```

```
0.8013333333333333
```

In [11]:
```
# Printing the misclassified values from Prediction
print('Misclassified samples: %d' % (test_y != prediction).sum())
```

```
Misclassified samples: 745
```

In [ ]:
```

```

In [1]:
```
#import the library Pandas
import pandas as pd
```

In [2]:
```
#Put the People Charm case.csv dataset on Root Directory
#Read the dataset
data = pd.read_csv('People Charm case.csv')
```

In [3]:
```
#Showing the 5 Data of given dataset
data.head()
```

Out[3]:

	satisfactoryLevel	lastEvaluation	numberOfProjects	avgMonthlyHours	timeSpent.company	w
0	0.38	0.53	2	157	3	
1	0.80	0.86	5	262	6	
2	0.11	0.88	7	272	4	
3	0.37	0.52	2	159	3	
4	0.41	0.50	2	153	3	

In [4]:
```
# To perform numerical operations
import numpy as np
# To Visualize data
import seaborn as sns
```

In [5]:
```
# To Partition the data
from sklearn.model_selection import train_test_split
# Importing the library of KNN
from sklearn.neighbors import KNeighborsClassifier
```

In [6]:
```
# Importing performance metrics - accuracy score & confusion matrix
from sklearn.metrics import accuracy_score,confusion_matrix
```

In [7]:
```
#=======================================
#         KNN Model
#=======================================
new_data = pd.get_dummies(data,drop_first = True)
columns_list = list(new_data.columns)
print(columns_list)

# Separating the input names from data
features=list(set(columns_list)-set(['left']))
print(features)

# Storing the output values in y
y=new_data['left'].values
print(y)

# Storing the values from input features
x = new_data[features].values
print(x)
# Spiliting the data into train and test
```

```
train_x,test_x,train_y,test_y = train_test_split(x,y,test_size=0.25,random_

# KNN Classification
# Stroing the K-nearest classifier
KNN_classifier = KNeighborsClassifier(n_neighbors = 2)

# Fitting the values for x and y
KNN_classifier.fit(train_x, train_y)

# Predicting the test values with Model
prediction = KNN_classifier.predict(test_x)

# Performance metric check
confusion_matrix = confusion_matrix(test_y,prediction)
print("\t","Predicted Values")
print("Original Values","\n",confusion_matrix)
accuracy_score = accuracy_score(test_y,prediction)
print(accuracy_score)

# Printing the misclassified values from Prediction
print('Misclassified samples: %d' % (test_y != prediction).sum())
```

```
['satisfactoryLevel', 'lastEvaluation', 'numberOfProjects', 'avgMonthlyHour
s', 'timeSpent.company', 'workAccident', 'left', 'promotionInLast5years',
'dept_RandD', 'dept_accounting', 'dept_hr', 'dept_management', 'dept_market
ing', 'dept_product_mng', 'dept_sales', 'dept_support', 'dept_technical',
'salary_low', 'salary_medium']
['salary_low', 'avgMonthlyHours', 'dept_management', 'numberOfProjects', 'w
orkAccident', 'dept_technical', 'promotionInLast5years', 'dept_support', 'd
ept_hr', 'dept_marketing', 'salary_medium', 'lastEvaluation', 'satisfactory
Level', 'dept_RandD', 'timeSpent.company', 'dept_accounting', 'dept_product
_mng', 'dept_sales']
[1 1 1 ... 0 0 0]
[[  1. 157.   0. ...   0.   0.   1.]
 [  0. 262.   0. ...   0.   0.   1.]
 [  0. 272.   0. ...   0.   0.   1.]
 ...
 [  1. 175.   0. ...   0.   0.   0.]
 [  1. 177.   0. ...   0.   0.   1.]
 [  1. 271.   0. ...   0.   0.   0.]]
        Predicted Values
Original Values
 [[2754   83]
 [  92  821]]
0.9533333333333334
Misclassified samples: 175
```

In [ ]:

In [1]:
```
#import the library Pandas
import pandas as pd
```

In [2]:
```
#Put the lendingdata.csv dataset on Root Directory
#Read the dataset
data = pd.read_csv('lendingdata.csv')
```

In [4]:
```
#Showing the 5 Data of given dataset
data
```

Out[4]:

	Unnamed: 0	activity	borrower_genders	country	country_code	currency_poli
0	0	Home Appliances	group	Cambodia	KH	shar
1	1	Cereals	female	Philippines	PH	shar
2	2	Clothing Sales	female	Peru	PE	shar
3	3	Clothing Sales	female	Tajikistan	TJ	not shar
4	4	Fish Selling	female	Uganda	UG	not shar
5	5	Personal Products Sales	female	Jordan	JO	shar
6	6	Transportation	male	Tajikistan	TJ	not shar
7	7	Home Appliances	group	Cambodia	KH	shar
8	8	Bakery	male	Nicaragua	NI	shar
9	9	Farming	male	Nigeria	NG	shar
10	10	Primary/secondary school costs	female	Colombia	CO	not shar
11	11	Construction Supplies	female	Nicaragua	NI	shar
12	12	Retail	female	Colombia	CO	not shar
13	13	Personal Products Sales	female	Philippines	PH	not shar
14	14	Cosmetics Sales	female	Ecuador	EC	not shar
15	15	Higher education costs	female	Colombia	CO	shar
16	16	General Store	male	Honduras	HN	shar
17	17	Retail	female	Benin	BJ	shar
18	18	Farming	group	Cambodia	KH	shar
19	19	Used Clothing	female	Nicaragua	NI	not shar
20	20	Personal Housing Expenses	group	Uganda	UG	not shar
21	21	Farming	group	Cambodia	KH	shar
22	22	Manufacturing	female	Philippines	PH	shar
23	23	Farming	male	Ecuador	EC	not shar
24	24	Cheese Making	female	Jordan	JO	shar
25	25	Cattle	NaN	Tajikistan	TJ	not shar
26	26	Higher education costs	female	Cambodia	KH	shar
27	27	Personal Housing Expenses	female	Indonesia	ID	shar
28	28	Farming	female	Philippines	PH	shar
29	29	Food	female	Kenya	KE	shar
...	...	...	...	...	...	
27506	27506	NaN	female	Philippines	PH	shar
27507	27507	Fruits & Vegetables	female	Philippines	PH	shar

	Unnamed: 0	activity	borrower_genders	country	country_code	currency_poli
27508	27508	Retail	female	Liberia	LR	shar
27509	27509	Grocery Store	male	Kenya	KE	shar
27510	27510	Pigs	group	Cambodia	KH	shar
27511	27511	Grocery Store	female	Kenya	KE	shar
27512	27512	Services	female	United States	US	not shar
27513	27513	Dairy	female	Kenya	KE	shar
27514	27514	Pigs	male	Nicaragua	NI	shar
27515	27515	Agriculture	female	El Salvador	SV	shar
27516	27516	Retail	group	NaN	PE	shar
27517	27517	Electronics Sales	female	Philippines	PH	shar
27518	27518	Food	group	Peru	PE	shar
27519	27519	Grocery Store	female	Indonesia	ID	not shar
27520	27520	Retail	female	Benin	BJ	shar
27521	27521	Motorcycle Transport	female	Philippines	PH	shar
27522	27522	General Store	female	Philippines	PH	shar
27523	27523	Crafts	male	El Salvador	SV	shar
27524	27524	Farming	group	Cambodia	KH	shar
27525	27525	Food Production/Sales	female	Cambodia	KH	not shar
27526	27526	Farm Supplies	male	El Salvador	SV	shar
27527	27527	Food	group	Ghana	GH	shar
27528	27528	Beverages	female	Nicaragua	NI	not shar
27529	27529	Grocery Store	female	Peru	PE	not shar
27530	27530	Transportation	male	Peru	PE	shar
27531	27531	Services	female	Zambia	ZM	shar
27532	27532	Pigs	female	Philippines	PH	not shar
27533	27533	General Store	female	Philippines	PH	shar
27534	27534	Food Market	female	Ghana	GH	shar
27535	27535	Farming	male	Tajikistan	TJ	not shar

27536 rows × 15 columns

In [4]:
```
# How many columns are of 'object' data type?
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 27536 entries, 0 to 27535
Data columns (total 15 columns):
Unnamed: 0            27536 non-null int64
activity              27535 non-null object
borrower_genders      27532 non-null object
country               27534 non-null object
country_code          27534 non-null object
currency_policy       27535 non-null object
distribution_model    27535 non-null object
lender_count          27534 non-null float64
loan_amount           27535 non-null float64
original_language     27533 non-null object
repayment_interval    27535 non-null object
sector                27536 non-null object
status                27536 non-null object
term_in_months        27536 non-null object
rMPI                  27536 non-null float64
dtypes: float64(3), int64(1), object(11)
memory usage: 3.2+ MB
```

In [6]:
```
len(data.select_dtypes('object').columns)
```

Out[6]:
```
11
```

In [6]:
```
data.get_dtype_counts()
```

Out[6]:
```
float64     3
int64       1
object     11
dtype: int64
```

In [7]:
```
# Find the total number of missing values in the data set?
data.isna().sum().sum()
```

Out[7]:
```
18
```

In [9]:
```
data.isnull().values.sum()
```

Out[9]:
```
18
```

In [11]:
```
# Identify which of the columns contain redundant information
# and can be dropped from the dataframe.
print(data.distribution_model.unique())
```

```
['field_partner' nan]
```

In [12]:
```
data.distribution_model.value_counts()
```

Out[12]:
```
field_partner    27535
Name: distribution_model, dtype: int64
```

In [14]:
```
data.distribution_model.duplicated().value_counts()
```

Out[14]:
```
True     27534
False        2
Name: distribution_model, dtype: int64
```

In [15]:
```
duplicate_percentage = (data.distribution_model.duplicated().sum()*100/data
print("Distribution_model duplicate rows in percentage:",duplicate_percenta
```

```
Distribution_model duplicate rows in percentage: 99.99273678094131
```

In [16]:
```
dup = data.activity.duplicated().value_counts()
print(dup)
```

```
True     27385
False      151
Name: activity, dtype: int64
```

In [24]:
```
duplicate_percentage1 = (dup[True]*100/data.shape[0])
print("Activity duplicate rows in percentage:",duplicate_percentage1)
```

```
Activity duplicate rows in percentage: 99.45162696106915
```

In [19]:
```
data.activity.value_counts()
```

Out[19]:

```
Farming                      2389
General Store                2353
Retail                       1506
Personal Housing Expenses    1262
Clothing Sales               1238
Food Production/Sales        1121
Agriculture                   907
Grocery Store                 815
Home Appliances               797
Pigs                          735
Fruits & Vegetables           687
Higher education costs        625
Food Market                   599
Food Stall                    446
Food                          437
Fish Selling                  424
Animal Sales                  372
Sewing                        347
Tailoring                     347
Fishing                       346
Services                      335
Personal Expenses             318
Beauty Salon                  312
Cattle                        308
Poultry                       300
Used Clothing                 299
Motorcycle Transport          294
Livestock                     292
Cosmetics Sales               289
Cereals                       251
                             ...
Blacksmith                     13
Cheese Making                  13
Electrician                    12
Hotel                          12
Movie Tapes & DVDs             12
Party Supplies                 11
Dental                         10
Bicycle Repair                 10
Call Center                     8
Religious Articles              7
Musical Performance             7
Child Care                      6
Music Discs & Tapes             6
Veterinary Sales                6
Musical Instruments             6
Waste Management                5
Phone Repair                    5
Machinery Rental                5
Well digging                    5
Upholstery                      5
Secretarial Services            4
Funerals                        3
Bicycle Sales                   3
Machine Shop                    3
Balut-Making                    3
Beekeeping                      2
Sporting Good Sales             2
Bookbinding                     2
Aquaculture                     1
Event Planning                  1
Name: activity, Length: 150, dtype: int64
```

In [20]:

```
print(data.country_code.unique())
```

```
['KH' 'PH' 'PE' 'TJ' 'UG' 'JO' 'NI' 'NG' 'CO' 'EC' 'HN' 'BJ' 'ID' 'KE'
 'GT' 'SL' 'SV' 'CD' 'BO' 'SS' nan 'PK' 'TL' 'MN' 'YE' 'CM' 'ML' 'GH' 'NP'
 'LR' 'LA' 'MZ' 'EG' 'VN' 'ZW' 'SN' 'AF' 'DO' 'BI' 'HT' 'LB' 'RW' 'MW'
 'US' 'TG' 'IN' 'MG' 'TZ' 'IQ' 'BR' 'CI' 'BF' 'BZ' 'SR' 'CG' 'MM' 'ZM'
 'GE' 'CR' 'LS' 'KG' 'WS']
```

In [22]:

```
dup1 = data.country_code.duplicated().value_counts()
print(dup1)
```

```
True     27474
False       62
Name: country_code, dtype: int64
```

In [25]:

```
duplicate_percentage2 = (dup1[True]*100/data.shape[0])
print("Country_code duplicate rows in percentage:",duplicate_percentage2)
```

```
Country_code duplicate rows in percentage: 99.77484020918071
```

In [26]:

```
# What is the third quartile value of the variable “loan_amount”?
data['loan_amount'].describe()
```

Out[26]:

```
count    27535.000000
mean       792.030143
std        902.925607
min         25.000000
25%        300.000000
50%        550.000000
75%       1000.000000
max      35000.000000
Name: loan_amount, dtype: float64
```

In [31]:

```
# What is the percentage split of the different categories in the column “
# after dropping the missing values?

data.repayment_interval.value_counts()
```

Out[31]:

```
monthly      15789
irregular     9264
bullet        2482
Name: repayment_interval, dtype: int64
```

In [32]:

```
round(data.repayment_interval.value_counts()*100/data.shape[0])
```

Out[32]:

```
monthly      57.0
irregular    34.0
bullet        9.0
Name: repayment_interval, dtype: float64
```

In [9]:

```
# What is the minimum loan amount disbursed in the Agriculture sector?
data.groupby('sector')['loan_amount'].max()['Agriculture']
```

Out[9]:

```
11025.0
```

In [ ]:

In [2]:
```
# Install nltk
# If you are using Windows or Linux or Mac, you can install NLTK using pip
# $ pip install nltk
import nltk
nltk.download()
```

```
showing info https://raw.githubusercontent.com/nltk/nltk_data/gh-pages/inde
x.xml
```

Out[2]:
```
True
```

In [3]:
```
# Here we will learn how to identify what the web page is about using NLTK
# First, we will grab a webpage and analyze the text to see what the page i
# urllib module will help us to crawl the webpage
import urllib.request
response =  urllib.request.urlopen('https://en.wikipedia.org/wiki/SpaceX')
html = response.read()
print(html)
```

```
<img src="/static/images/footer/poweredby_mediawiki_88x31.png" alt="Powered
by MediaWiki" srcset="/static/images/footer/poweredby_mediawiki_132x47.png
 1.5x, /static/images/footer/poweredby_mediawiki_176x62.png 2x" width="88"
 height="31" loading="lazy"/></a></li>\n</ul>\n\n</footer>\n\n\n<script>(RL
Q=window.RLQ||[]).push(function(){mw.config.set({"wgPageParseReport":{"limi
treport":{"cputime":"2.782","walltime":"3.233","ppvisitednodes":{"value":14
403,"limit":1000000},"postexpandincludesize":{"value":549890,"limit":209715
2},"templateargumentsize":{"value":14670,"limit":2097152},"expansiondepth":
{"value":21,"limit":40},"expensivefunctioncount":{"value":18,"limit":50
0},"unstrip-depth":{"value":1,"limit":20},"unstrip-size":{"value":718836,"l
imit":5000000},"entityaccesscount":{"value":1,"limit":400},"timingprofile":
["100.00% 2741.155      1 -total"," 51.64% 1415.516      2 Template:Reflis
t"," 30.06%  824.113    154 Template:Cite_web","  9.96%  273.093      1 Tem
plate:Infobox_company","  8.68%  238.028      1 Template:Infobox","  7.18%
196.900     38 Template:Cite_news","  4.43%  121.539      1 Template:Elon_M
usk_series","  4.26%  116.780      1 Template:Sidebar","  3.85%  105.525
10 Template:Navbox","  3.25%   88.961      1 Template:Short_descriptio
n"]},"scribunto":{"limitreport-timeusage":{"value":"1.468","limit":"10.00
0"},"limitreport-memusage":{"value":14849685,"limit":52428800},"limitreport
-profile":[["?","220","14.1"],["Scribunto_LuaSandboxCallback::callParserFun
ction","180","11.5"],["Scribunto_LuaSandboxCallback::getExpandedArgumen
t","160","10.3"],["Scribunto_LuaSandboxCallback::gsub","100","6.4"],["recur
siveClone \\u003CmwInit.lua:41\\u003E","100","6.4"],["Scribunto_LuaSandboxC
allback::plain","80","5.1"],["Scribunto_LuaSandboxCallback::match","80","5.
1"],["\\u003CModule:Citation/CS1:829\\u003E","80","5.1"],["dataWrapper \\u0
03Cmw.lua:668\\u003E","80","5.1"],["Scribunto_LuaSandboxCallback::getAllExp
andedArguments","60","3.8"],["[others]","420","26.9"]]},"cachereport":{"ori
gin":"mw1376","timestamp":"20210529151623","ttl":1814400,"transientconten
t":false}}});});</script>\n<script type="application/ld+json">{"@contex
t":"https:\\/\\/schema.org","@type":"Article","name":"SpaceX","url":"http
s:\\/\\/en.wikipedia.org\\/wiki\\/SpaceX","sameAs":"http:\\/\\/www.wikidat
a.org\\/entity\\/Q193701","mainEntity":"http:\\/\\/www.wikidata.org\\/entit
y\\/Q193701","author":{"@type":"Organization","name":"Contributors to Wikim
edia projects"},"publisher":{"@type":"Organization","name":"Wikimedia Found
ation, Inc.","logo":{"@type":"ImageObject","url":"https:\\/\\/www.wikimedi
a.org\\/static\\/images\\/wmf-hor-googpub.png"}},"datePublished":"2004-07-1
6T17:12:15Z","dateModified":"2021-05-29T15:16:13Z","image":"https:\\/\\/upl
oad.wikimedia.org\\/wikipedia\\/commons\\/e\\/ee\\/Iridium-4_Mission_%28255
57986177%29.jpg","headline":"American private aerospace company"}</script>
\n<script>(RLQ=window.RLQ||[]).push(function(){mw.config.set({"wgBackendRes
ponseTime":199,"wgHostname":"mw1325"});});</script>\n</body></html>'
```


In [4]:

```
# We will use Beautiful Soup which is a Python library for pulling data ou
# We will use beautiful soup to clean our webpage text of HTML tags.
from bs4 import BeautifulSoup
soup = BeautifulSoup(html,'html5lib')
text = soup.get_text(strip = True)
print(text)
```

بيةAragonésAsturianuAzərbaycancaتۆركجەবাংলাBân-lâm-gúБеларускаяभोजपुरीБългарски
CatalàЧăвашлаČeštinaCymraegDanskDeutschEestiΕλληνικάEspañolEsperantoEuskara
فارسیFrançais한국어Հայերենहिन्दीHrvatskiBahasa IndonesiaÍslenskaItalianoעבריתქარ
თულიҚазақшаKurdîКыргызчаLatviešuLëtzebuergeschLietuviųLimburgsMagyarമലയാളംम
राठीმარგალურიBahasa MelayuМонголNederlandsनेपाली日本語Norsk bokmålNorsk nynorskO
ccitanPolskiPortuguêsRomânăRuna SimiРусскийScotsShqipSimple EnglishSlovenči
naSlovenščinaСрпски / srpskiSrpskohrvatski / српскохрватскиSuomiSvenskaTaga
logதமிழ்ไทยTürkçeУкраїнськаئۇيغۇرچە / UyghurcheTiếng Việt吴语Yorùbá粵語Žemaitė
ška中文Edit linksThis page was last edited on 29 May 2021, at 15:16(UTC).Tex
t is available under theCreative Commons Attribution-ShareAlike License;
additional terms may apply. By using this site, you agree to theTerms of U
seandPrivacy Policy. Wikipedia® is a registered trademark of theWikimedia F
oundation, Inc., a non-profit organization.Privacy policyAbout WikipediaDis
claimersContact WikipediaMobile viewDevelopersStatisticsCookie statement(RL
Q=window.RLQ||[]).push(function(){mw.config.set({"wgPageParseReport":{"limi
treport":{"cputime":"2.782","walltime":"3.233","ppvisitednodes":{"value":14
403,"limit":1000000},"postexpandincludesize":{"value":549890,"limit":209715
2},"templateargumentsize":{"value":14670,"limit":2097152},"expansiondepth":
{"value":21,"limit":40},"expensivefunctioncount":{"value":18,"limit":50
0},"unstrip-depth":{"value":1,"limit":20},"unstrip-size":{"value":718836,"l
imit":5000000},"entityaccesscount":{"value":1,"limit":400},"timingprofile":
["100.00% 2741.155   1 -total"," 51.64% 1415.516   2 Template:Reflis
t"," 30.06% 824.113   154 Template:Cite_web"," 9.96% 273.093   1 Tem
plate:Infobox_company"," 8.68% 238.028   1 Template:Infobox"," 7.18%
196.900   38 Template:Cite_news"," 4.43% 121.539   1 Template:Elon_M
usk_series"," 4.26% 116.780   1 Template:Sidebar"," 3.85% 105.525
10 Template:Navbox"," 3.25%   88.961   1 Template:Short_descriptio
n"]},"scribunto":{"limitreport-timeusage":{"value":"1.468","limit":"10.00
0"},"limitreport-memusage":{"value":14849685,"limit":52428800},"limitreport
-profile":[["?","220","14.1"],["Scribunto_LuaSandboxCallback::callParserFun
ction","180","11.5"],["Scribunto_LuaSandboxCallback::getExpandedArgumen
t","160","10.3"],["Scribunto_LuaSandboxCallback::gsub","100","6.4"],["recur
siveClone \u003CmwInit.lua:41\u003E","100","6.4"],["Scribunto_LuaSandboxCal
lback::plain","80","5.1"],["Scribunto_LuaSandboxCallback::match","80","5.
1"],["\u003CModule:Citation/CS1:829\u003E","80","5.1"],["dataWrapper \u003C
mw.lua:668\u003E","80","5.1"],["Scribunto_LuaSandboxCallback::getAllExpande
dArguments","60","3.8"],["[others]","420","26.9"]]},"cachereport":{"origi
n":"mw1376","timestamp":"20210529151623","ttl":1814400,"transientcontent":f
alse}}});});{"@context":"https:\/\/schema.org","@type":"Article","name":"Sp
aceX","url":"https:\/\/en.wikipedia.org\/wiki\/SpaceX","sameAs":"http:\/\/w
ww.wikidata.org\/entity\/Q193701","mainEntity":"http:\/\/www.wikidata.org\/
entity\/Q193701","author":{"@type":"Organization","name":"Contributors to W
ikimedia projects"},"publisher":{"@type":"Organization","name":"Wikimedia F
oundation, Inc.","logo":{"@type":"ImageObject","url":"https:\/\/www.wikimed
ia.org\/static\/images\/wmf-hor-googpub.png"}},"datePublished":"2004-07-16T
17:12:15Z","dateModified":"2021-05-29T15:16:13Z","image":"https:\/\/upload.
wikimedia.org\/wikipedia\/commons\/e\/ee\/Iridium-4_Mission_%2825557986177%
29.jpg","headline":"American private aerospace company"}(RLQ=window.RLQ||
[]).push(function(){mw.config.set({"wgBackendResponseTime":199,"wgHostnam
e":"mw1325"});});});

In [5]:

```
# Now we have clean text from the crawled web page, let's convert the text
tokens = [t for t in text.split()]
print(tokens)
```

4670,"limit":2097152},"expansiondepth":{"value":21,"limit":40},"expensivefunctioncount":{"value":18,"limit":500},"unstrip-depth":{"value":1,"limit":20},"unstrip-size":{"value":718836,"limit":5000000},"entityaccesscount":{"value":1,"limit":400},"timingprofile":["100.00%', '2741.155', '1', '-total","', '51.64%', '1415.516', '2', 'Template:Reflist","', '30.06%', '824.113', '154', 'Template:Cite_web","', '9.96%', '273.093', '1', 'Template:Infobox_company","', '8.68%', '238.028', '1', 'Template:Infobox","', '7.18%', '196.900', '38', 'Template:Cite_news","', '4.43%', '121.539', '1', 'Template:Elon_Musk_series","', '4.26%', '116.780', '1', 'Template:Sidebar","', '3.85%', '105.525', '10', 'Template:Navbox","', '3.25%', '88.961', '1', 'Template:Short_description"]},"scribunto":{"limitreport-timeusage":{"value":"1.468","limit":"10.000"},"limitreport-memusage":{"value":14849685,"limit":52428800},"limitreport-profile":[["?","220","14.1"],["Scribunto_LuaSandboxCallback::callParserFunction","180","11.5"],["Scribunto_LuaSandboxCallback::getExpandedArgument","160","10.3"],["Scribunto_LuaSandboxCallback::gsub","100","6.4"],["recursiveClone', '\\u003CmwInit.lua:41\\u003E","100","6.4"],["Scribunto_LuaSandboxCallback::plain","80","5.1"],["Scribunto_LuaSandboxCallback::match","80","5.1"],["\\u003CModule:Citation/CS1:829\\u003E","80","5.1"],["dataWrapper', '\\u003Cmw.lua:668\\u003E","80","5.1"],["Scribunto_LuaSandboxCallback::getAllExpandedArguments","60","3.8"],["[others]","420","26.9"]]},"cachereport":{"origin":"mw1376","timestamp":"20210529151623","ttl":1814400,"transientcontent":false}}});});{"@context":"https:\\/\\/schema.org","@type":"Article","name":"SpaceX","url":"https:\\/\\/en.wikipedia.org\\/wiki\\/SpaceX","sameAs":"http:\\/\\/www.wikidata.org\\/entity\\/Q193701","mainEntity":"http:\\/\\/www.wikidata.org\\/entity\\/Q193701","author":{"@type":"Organization","name":"Contributors', 'to', 'Wikimedia', 'projects"},"publisher":{"@type":"Organization","name":"Wikimedia', 'Foundation,', 'Inc.","logo":{"@type":"ImageObject","url":"https:\\/\\/www.wikimedia.org\\/static\\/images\\/wmf-hor-googpub.png"}},"datePublished":"2004-07-16T17:12:15Z","dateModified":"2021-05-29T15:16:13Z","image":"https:\\/\\/upload.wikimedia.org\\/wikipedia\\/commons\\/e\\/ee\\/Iridium-4_Mission_%2825557986177%29.jpg","headline":"American', 'private', 'aerospace', 'company"}(RLQ=window.RLQ||[]).push(function(){mw.config.set({"wgBackendResponseTime":199,"wgHostname":"mw1325"});});']

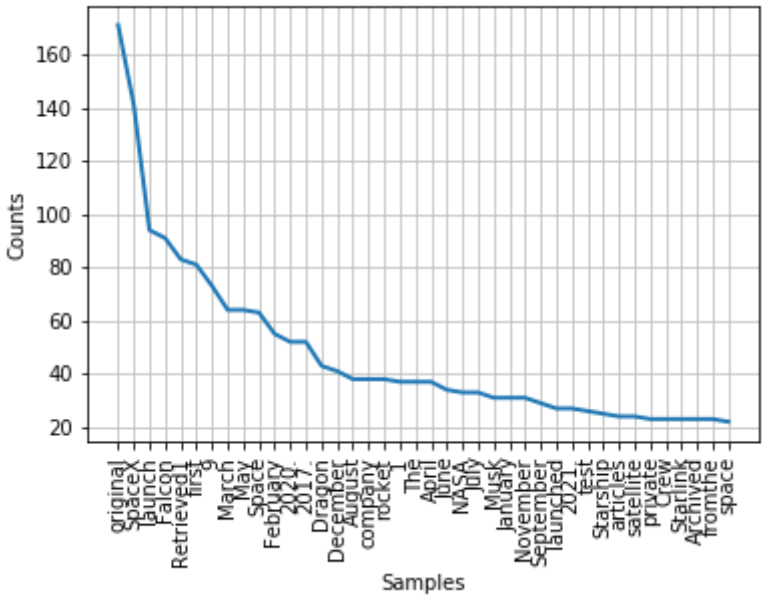
In [6]:

```
# Count word Frequency
# nltk offers a function FreqDist() which will do the job for us.
# Also, we will remove stop words (a, at, the, for etc) from our web page
# as we don't need them to hamper our word frequency count.

from nltk.corpus import stopwords
sr= stopwords.words('english')
clean_tokens = tokens[:]
for token in tokens:
    if token in stopwords.words('english'):

        clean_tokens.remove(token)
freq = nltk.FreqDist(clean_tokens)
for key,val in freq.items():
    print(str(key) + ':' + str(val))
```

a.org\/wikipedia\/commons\/e\/ee\/Iridium-4_Mission_%2825557986177%29.jpg","headline":"American:1 company"}(RLQ=window.RLQ||[]).push(function(){mw.config.set({"wgBackendResponseTime":199,"wgHostname":"mw1325"});});:1

In [8]:

```
# We will plot the graph for most frequently occurring words in the webpage
# to get the clear picture of the context of the web page

freq.plot(40, cumulative=False)
```



In [ ]:

In [1]:
```
#import the library Pandas
import pandas as pd
```

In [2]:
```
#Put the spam.csv dataset on Root Directory
#Read the dataset
data = pd.read_csv("spam.csv", encoding = "latin-1")
```

In [3]:
```
# Changed the column names to be more descriptive.
data = data[['v1', 'v2']]
data = data.rename(columns = {'v1': 'label', 'v2': 'text'})
```

In [4]:
```
#Showing the data
data
```

Out[4]:

	label	text
0	ham	Go until jurong point, crazy.. Available only ...
1	ham	Ok lar... Joking wif u oni...
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...
3	ham	U dun say so early hor... U c already then say...
4	ham	Nah I don't think he goes to usf, he lives aro...
5	spam	FreeMsg Hey there darling it's been 3 week's n...
6	ham	Even my brother is not like to speak with me. ...
7	ham	As per your request 'Melle Melle (Oru Minnamin...
8	spam	WINNER!! As a valued network customer you have...
9	spam	Had your mobile 11 months or more? U R entitle...
10	ham	I'm gonna be home soon and i don't want to tal...
11	spam	SIX chances to win CASH! From 100 to 20,000 po...
12	spam	URGENT! You have won a 1 week FREE membership ...
13	ham	I've been searching for the right words to tha...
14	ham	I HAVE A DATE ON SUNDAY WITH WILL!!
15	spam	XXXMobileMovieClub: To use your credit, click ...
16	ham	Oh k...i'm watching here:)
17	ham	Eh u remember how 2 spell his name... Yes i di...
18	ham	Fine if thatåÕs the way u feel. ThatåÕs the wa...
19	spam	England v Macedonia - dont miss the goals/team...
20	ham	Is that seriously how you spell his name?
21	ham	I◗Û÷m going to try for 2 months ha ha only joking
22	ham	So Ì_ pay first lar... Then when is da stock c...
23	ham	Aft i finish my lunch then i go str down lor. ...
24	ham	Fffffffff. Alright no way I can meet up with ...
25	ham	Just forced myself to eat a slice. I'm really ...
26	ham	Lol your always so convincing.
27	ham	Did you catch the bus ? Are you frying an egg ...
28	ham	I'm back & we're packing the car now, I'll...
29	ham	Ahhh. Work. I vaguely remember that! What does...
...	...	...
5542	ham	Armand says get your ass over to epsilon
5543	ham	U still havent got urself a jacket ah?
5544	ham	I'm taking derek & taylor to walmart, if I...
5545	ham	Hi its in durban are you still on this number
5546	ham	Ic. There are a lotta childporn cars then.
5547	spam	Had your contract mobile 11 Mnths? Latest Moto...
5548	ham	No, I was trying it all weekend ;V
5549	ham	You know, wot people wear. T shirts, jumpers, ...

	label	text
5550	ham	Cool, what time you think you can get here?
5551	ham	Wen did you get so spiritual and deep. That's ...
5552	ham	Have a safe trip to Nigeria. Wish you happines...
5553	ham	Hahaha..use your brain dear
5554	ham	Well keep in mind I've only got enough gas for...
5555	ham	Yeh. Indians was nice. Tho it did kane me off ...
5556	ham	Yes i have. So that's why u texted. Pshew...mi...
5557	ham	No. I meant the calculation is the same. That ...
5558	ham	Sorry, I'll call later
5559	ham	if you aren't here in the next <#> hou...
5560	ham	Anything lor. Juz both of us lor.
5561	ham	Get me out of this dump heap. My mom decided t...
5562	ham	Ok lor... Sony ericsson salesman... I ask shuh...
5563	ham	Ard 6 like dat lor.
5564	ham	Why don't you wait 'til at least wednesday to ...
5565	ham	Huh y lei...
5566	spam	REMINDER FROM O2: To get 2.50 pounds free call...
5567	spam	This is the 2nd time we have tried 2 contact u...
5568	ham	Will Ì_ b going to esplanade fr home?
5569	ham	Pity, * was in mood for that. So...any other s...
5570	ham	The guy did some bitching but I acted like i'd...
5571	ham	Rofl. Its true to its name

5572 rows × 2 columns

In [5]:

```
# For the classification problem, we will only use case normalisation.
# The rationale is that it will be hard to apply a stemmer or lemmatiser or
# and that since the text messages are so short, removing stop words might
# with much to work with.
def review_messages(msg):
    # converting messages to lowercase
    msg = msg.lower()
    return msg
```

In [6]:

```
# For reference, this function does case normalisation, removing stop words
from nltk import stem
from nltk.corpus import stopwords
stemmer = stem.SnowballStemmer('english')
stopwords = set(stopwords.words('english'))

def alternative_review_messages(msg):
    # converting messages to lowercase
    msg = msg.lower()
    # removing stopwords
    msg = [word for word in msg.split() if word not in stopwords]
    # using a stemmer
```

```
    msg = " ".join([stemmer.stem(word) for word in msg])
    return msg
```

In [7]:
```
# We apply the first function to normalise the text messages.
data['text'] = data['text'].apply(review_messages)
```

In [8]:
```
# Vectorizing the Text
# Before training the vectorizer, we split our data into a training set and
# 10% of our data is allocated for testing.
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(data['text'], data['lab
# training the vectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer()
X_train = vectorizer.fit_transform(X_train)
```

In [9]:
```
# Building and Testing the Classifier
from sklearn import svm
svm = svm.SVC(C=1000)
svm.fit(X_train, y_train)
```

```
C:\Users\my pc\Anaconda3\lib\site-packages\sklearn\svm\base.py:196: FutureW
arning: The default value of gamma will change from 'auto' to 'scale' in ve
rsion 0.22 to account better for unscaled features. Set gamma explicitly to
'auto' or 'scale' to avoid this warning.
  "avoid this warning.", FutureWarning)
```

Out[9]:
```
SVC(C=1000, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma='auto_deprecated',
  kernel='rbf', max_iter=-1, probability=False, random_state=None,
  shrinking=True, tol=0.001, verbose=False)
```

In [11]:
```
# Now, let's test it.
from sklearn.metrics import confusion_matrix
X_test = vectorizer.transform(X_test)
y_pred = svm.predict(X_test)
print(confusion_matrix(y_test, y_pred))
```

```
[[490   0]
 [ 10  58]]
```

In [11]:
```
# The results aren't bad at all! We have no false positives and around 15%
# Let's test it against a few new examples.
def pred(msg):
    msg = vectorizer.transform([msg])
    prediction = svm.predict(msg)
    return prediction[0]
```

In [13]:
```
data['text'] = data['text'].apply(pred)
```

In [ ]:
```
```