1.**Exploring Data**: Nowadays there are lots of interesting data sets publicly available. They range in size, quality and the type of features and have resulted in many new machine learning techniques being developed. Find a public, free, supervised (i.e. it must have features and labels), machine learning dataset. Once you've found the data set, provide the following information:

1. The name of the data set.
2. From where the data set is obtained.
3. A one or two brief sentences description of the data set including what the features are and what is being predicted.
4. The number of examples in the data set.
5. The number of features for each example. If this isn't concrete (i.e. it's text), then a short description of the features.

Extra credit will be given for particularly interesting data sets, e.g. the most unique, the data set with the largest number of examples and the data set with the largest number of features.

```
import pandas as pd
# by loading the csv file into a pandas dataframe we can analyse the dataset.
import numpy as np
```

1. **Name of the Dataset** : New York City Airbnb Open Data
2. Dataset is Obtained from Kaggle. Here is the link : [New York City Airbnb Open Data]
3. **About this dataset** Since 2008, guests and hosts have used Airbnb to expand on traveling possibilities and present more unique, personalized way of experiencing the world. This dataset describes the listing activity and metrics in NYC, NY for 2019. This data file includes all needed information to find out more about hosts, geographical availability, necessary metrics to make predictions and draw conclusions.
4. **Number of Examples** : 48895
5. **Number of Features** : 16. Details about the features are described in the below cells

We can set many of this features as Labels to train on, Like the price reviews etc.,

```
# Loading the csv file into AirBnbDF dataframe.
AirBnbDF = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/AB_NYC_2019.csv')

# To see the first 5 rows of the data frame
AirBnbDF.head()
```

	id	name	host_id	host_name	neighbourhood_group	neighbourhood	latitude	longitude	room_type	price	mi
0	2539	Clean & quiet apt home by the park	2787	John	Brooklyn	Kensington	40.64749	-73.97237	Private room	149	
1	2595	Skylit Midtown Castle	2845	Jennifer	Manhattan	Midtown	40.75362	-73.98377	Entire home/apt	225	
2	3647	THE VILLAGE OF HARLEM....NEW YORK !	4632	Elisabeth	Manhattan	Harlem	40.80902	-73.94190	Private room	150	
3	3831	Cozy Entire Floor of Brownstone	4869	LisaRoxanne	Brooklyn	Clinton Hill	40.68514	-73.95976	Entire home/apt	89	
4	5022	Entire Apt: Spacious Studio/Loft by central park	7192	Laura	Manhattan	East Harlem	40.79851	-73.94399	Entire home/apt	80	

```
print('Total Examples on this dataframe : ',len(AirBnbDF))
print('Total Features on this dataframe : ', len(AirBnbDF.columns))
```

```
Total Examples on this dataframe :  48895
Total Features on this dataframe :  16
```

```
# Feature types are listed using .info() of pandas.
AirBnbDF.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 48895 entries, 0 to 48894
Data columns (total 16 columns):
 #   Column                          Non-Null Count  Dtype  
---  ------                          --------------  -----  
 0   id                              48895 non-null  int64  
 1   name                            48879 non-null  object 
 2   host_id                         48895 non-null  int64  
 3   host_name                       48874 non-null  object 
 4   neighbourhood_group             48895 non-null  object 
 5   neighbourhood                   48895 non-null  object 
 6   latitude                        48895 non-null  float64
 7   longitude                       48895 non-null  float64
 8   room_type                       48895 non-null  object 
 9   price                           48895 non-null  int64  
 10  minimum_nights                  48895 non-null  int64  
 11  number_of_reviews               48895 non-null  int64  
 12  last_review                     38843 non-null  object 
 13  reviews_per_month               38843 non-null  float64
 14  calculated_host_listings_count  48895 non-null  int64  
 15  availability_365                48895 non-null  int64  
dtypes: float64(3), int64(7), object(6)
memory usage: 6.0+ MB
```

```
# This can give you an idea about the data that we are handling. Calulated only for Numerical values
AirBnbDF.describe()
```

	id	host_id	latitude	longitude	price	minimum_nights	number_of_reviews	reviews_per_mo
count	4.889500e+04	4.889500e+04	48895.000000	48895.000000	48895.000000	48895.000000	48895.000000	38843.000
mean	1.901714e+07	6.762001e+07	40.728949	-73.952170	152.720687	7.029962	23.274466	1.373
std	1.098311e+07	7.861097e+07	0.054530	0.046157	240.154170	20.510550	44.550582	1.680
min	2.539000e+03	2.438000e+03	40.499790	-74.244420	0.000000	1.000000	0.000000	0.010
25%	9.471945e+06	7.822033e+06	40.690100	-73.983070	69.000000	1.000000	1.000000	0.190
50%	1.967728e+07	3.079382e+07	40.723070	-73.955680	106.000000	3.000000	5.000000	0.720
75%	2.915218e+07	1.074344e+08	40.763115	-73.936275	175.000000	5.000000	24.000000	2.020
max	3.648724e+07	2.743213e+08	40.913060	-73.712990	10000.000000	1250.000000	629.000000	58.500

2.**Data analysis**: The rst thing to do before trying any machine learning technique is to look deep into the data. This can include looking for funny values in the data, looking for outliers, looking at the range of feature values, what features seem important, etc.

Consider a modied version of passenger survival data for the Titanic1(titanic-train.csv) that is uploaded in Google Classroom.

(a) For each of the features calculate the training error, Accuracy and error.

(b) Which feature would be the best to use?

```
# loading the csv file into a pandas dataframe
# since i am using google colab .. CSV file is loaded from mounted google drive
```

```
# storing the csv values into titanicDF dataframe
titanicDF = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/titanic-train.csv')

titanicDF.head() # to display the first five values of the dataframe
```

	First_class	Sex	Age	SibSp	ParCh	Embarked	Survived
0	0	0	0	0	1	0	1
1	0	1	0	1	1	0	1
2	0	1	0	1	1	0	1
3	0	1	0	0	1	0	1
4	0	0	0	0	1	0	1

```
# creating a function to find the accuracy and error

def AccuracyAndError(feature):

  """
    Used to Find the Accuracy and Error of Each Feature

    input: String, features of the dataset.
    return: Float, accuracy and error
  """

  # splitting the dataframe to two sets.

  # bin0 contains set of examples having the particular feature  = 0
  bin0 = titanicDF[titanicDF[feature]==0]

  # bin1 contains set of examples  having particular feature = 1
  bin1 = titanicDF[titanicDF[feature]==1]

  # calculating the majority count for the label in each bin

  majority_bin0 = max(len(bin0[bin0['Survived']==1]),len(bin0[bin0['Survived']==0]))
  majority_bin1 = max(len(bin1[bin1['Survived']==1]), len(bin1[bin1['Survived']==0]))

  # calculating the training accuracy and error of particluar feature

  accuracy = (majority_bin0 + majority_bin1)/len(titanicDF)
  error = 1 - accuracy
  return accuracy, 1-accuracy
```

```
# all the names of the features are stored in a numpy array
featureArray=np.array(['First_class', 'Sex', 'Age', 'SibSp', 'ParCh', 'Embarked'])

# a dictionary is used to store output
# key = feature name
# value = accuracy
resultDic={} # empty dictionary defined

# for storing and printing the output
for i in range(len(featureArray)):
  feature = featureArray[i]
  accuracy, error = AccuracyAndError(feature) # function call
  resultDic[featureArray[i]] = accuracy # updating the value into dictionary
  print('{} \n Accuracy: {} \n Error: {} \n \n'.format(feature, round(accuracy,3), round(error,3)))
```

```
    First_class
     Accuracy: 0.675
```

```
 Error: 0.325


Sex
 Accuracy: 0.78
 Error: 0.22


Age
 Accuracy: 0.594
 Error: 0.406


SibSp
 Accuracy: 0.594
 Error: 0.406


ParCh
 Accuracy: 0.615
 Error: 0.385


Embarked
 Accuracy: 0.616
 Error: 0.384
```

```
import operator # used to access particular values from dictonary

# MaxValueKey to store the key with highest value
MaxValueKey = max(resultDic.items(), key=operator.itemgetter(1))[0]

print('The Best Feature would be "{}" with an Accuracy of {}%'.format(MaxValueKey,
                                                                      round(resultDic[MaxValueKey]*100,3)))
```

```
The Best Feature would be "Sex" with an Accuracy of 78.011%
```

```

```