Data Science Project

August 5, 2022

Prepared by: Ahmad Feroz & Mohammad Omar

1 Business Understanding

For this project we are going use amazon.com books datasets that are taken from kaggle.com

Amazon.com, Inc. is an American multinational technology company which focuses on e-commerce, cloud computing, digital streaming, and artificial intelligence. It has been referred to as "one of the most influential economic and cultural forces in the world", and is one of the world's most valuable brands. And also, its one of the largest online book stores in most of countries.

In this project we are going to build a model to classify machine learing books from other books based on their title or name.

And also we use python and its different libraries

2 Data Understanding and Preparation

As we mention earlier for this project we are using datasets that are taken form kaggle.com and one of them is as follows:

[1]:
```
# importing necessary libraries
%matplotlib inline
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

2.1 Importing the dataset

we are using a pandas' method called read_csv for importing or reading a dataset

[2]:
```
#the dataset
df1 = pd.read_csv(r"Downloads\CSVFiles\ML_books.csv")
```

[3]:
```
# sample of data
df1.head()
```

```
[3]:                                                 Name                 Autor  \
0  Hands-On Machine Learning with Scikit-Learn, K…        Aurélien Géron
1  Machine Learning Design Patterns: Solutions to…  Valliappa Lakshmanan
2  AI and Machine Learning for Coders: A Programm…      Laurence Moroney
3                     Machine Learning Engineering         Andriy Burkov
4  Machine Learning: 4 Books in 1: Basic Concepts…         Ethem Mining

               Review Review qntd     Format   Price
0  4.8 out of 5 stars       1,808  Paperback  $17.50
1  4.6 out of 5 stars          66  Paperback  $35.49
2  4.8 out of 5 stars          29  Paperback  $45.59
3  4.7 out of 5 stars         113  Paperback  $35.49
4  4.3 out of 5 stars         106     Kindle   $0.00
```

2.2 Data information

```
[4]: df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 285 entries, 0 to 284
Data columns (total 6 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Name         285 non-null    object
 1   Autor        119 non-null    object
 2   Review       201 non-null    object
 3   Review qntd  192 non-null    object
 4   Format       256 non-null    object
 5   Price        261 non-null    object
dtypes: object(6)
memory usage: 13.5+ KB
```

2.3 Data Description

```
[5]: # decription of data
df1.describe()
```

```
[5]:                                                       Name Autor  \
count                                                      285   119
unique                                                     274    99
top     Funny Data Sciene Gift - AI Data Scientist Mac…  From
freq                                                         3     6

                    Review Review qntd     Format  Price
count                  201         192        256    261
unique                  21          99          8    162
top     4.5 out of 5 stars           2  Paperback  $0.00
```

```
freq                        31            11         112      52
```

[6]:
```
# we are going to see how many rows and columns do we have
df1.shape
```

[6]:
```
(285, 6)
```

[7]:
```
# 285 rows, and 6 columns
```

For every task that we want to perform later we need to ensure our self from quality of our data. Means we are going to check for short comming data and dirty data like: missing values, duplicated values and etc.

2.4 Missing Data And Data type conversion

Missing Data are those data or values that are missed in a dataset or we can simply say there places are empty in dataset, Missing data is a common problem before starting any task of data science, if they are not handle, will cause the result of analysis or final outcome to be incorrect. and missing values in pandas dataframe represented by (NaN).

[8]:
```
# checking for missing values
df1.isnull()
# it will show all values that missed with true otherwise false
```

[8]:
```
      Name  Autor  Review  Review qntd  Format  Price
0    False  False   False        False   False  False
1    False  False   False        False   False  False
2    False  False   False        False   False  False
3    False  False   False        False   False  False
4    False  False   False        False   False  False
..     …      …       …          …       …      …
280  False  False   False        False   False  False
281  False   True    True         True   False  False
282  False   True   False        False   False  False
283  False  False   False        False   False  False
284  False   True    True         True    True   True

[285 rows x 6 columns]
```

[9]:
```
# number of missing values in each feature or column
df1.isnull().sum()
```

[9]:
```
Name             0
Autor          166
Review          84
Review qntd     93
Format          29
Price           24
```

```
dtype: int64
```

[10]:
```python
# total number of missing values in our dataset
df1.isnull().sum().sum()
```

[10]:
```
396
```

For handling missing values we have two methods 1) Deleting missing values 2) Imputing missing values.

1) Deleting missing values: Simply delete the rows or columns that have missing values.
2) Imputing missing values: There are different ways of replacing the missing values. Replacing with Arbitrary values, with mean, with Mod, with Median , with previous value and with next value.

Note: mean, mod and median only use for numeric values.

[11]:
```python
# imputing values for (Autor) attribute using forwardfill technique
# forwardfill simply means replacing the missing value with its previous value␣
↪on that column

df1["Autor"] = df1["Autor"].fillna(method="ffill")
```

[12]:
```python
# checking for missing value on autor column
df1["Autor"].isnull().sum()
```

[12]:
```
0
```

[13]:
```python
# as we can see the (Autor) column name has spelling mistake we are going to␣
↪correct it
df1.rename(columns={"Autor": "Author"}, inplace=True)
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 285 entries, 0 to 284
Data columns (total 6 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Name         285 non-null    object
 1   Author       285 non-null    object
 2   Review       201 non-null    object
 3   Review qntd  192 non-null    object
 4   Format       256 non-null    object
 5   Price        261 non-null    object
dtypes: object(6)
memory usage: 13.5+ KB
```

[14]:
```
# And also we want to to change (Review) to (User Rating) and (Review qntd) to␣
↪(Reviews) for better undrestanding.
df1.rename(columns={"Review": "User Rating"}, inplace=True)
df1.rename(columns={"Review qntd": "Reviews"}, inplace=True)
```

[15]:
```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 285 entries, 0 to 284
Data columns (total 6 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Name         285 non-null    object
 1   Author       285 non-null    object
 2   User Rating  201 non-null    object
 3   Reviews      192 non-null    object
 4   Format       256 non-null    object
 5   Price        261 non-null    object
dtypes: object(6)
memory usage: 13.5+ KB
```

[16]:
```
df1.head()
```

[16]:
```
                                                Name               Author  \
0  Hands-On Machine Learning with Scikit-Learn, K…        Aurélien Géron
1  Machine Learning Design Patterns: Solutions to…  Valliappa Lakshmanan
2  AI and Machine Learning for Coders: A Programm…      Laurence Moroney
3                        Machine Learning Engineering          Andriy Burkov
4  Machine Learning: 4 Books in 1: Basic Concepts…          Ethem Mining

          User Rating Reviews     Format   Price
0  4.8 out of 5 stars   1,808  Paperback  $17.50
1  4.6 out of 5 stars      66  Paperback  $35.49
2  4.8 out of 5 stars      29  Paperback  $45.59
3  4.7 out of 5 stars     113  Paperback  $35.49
4  4.3 out of 5 stars     106     Kindle   $0.00
```

[17]:
```
# for (User Rating) column we want to have just the rating part as we can see␣
↪the data in this column is like
# (4.8 out of 5 stars) we want just the 4.8 not the whole sentence.

# for doing this we write a function
def formatUserRating(a):
    if type(a) == type(""):
        result = a.split(" ")
        return result[0]
    else:
```

```
        return np.nan

# apply function on that column
df1["User Rating"] = df1["User Rating"].apply(formatUserRating)
```

[18]:
```
df1.head()
```

[18]:
```
                                                Name                Author  \
0  Hands-On Machine Learning with Scikit-Learn, K…        Aurélien Géron
1  Machine Learning Design Patterns: Solutions to…  Valliappa Lakshmanan
2  AI and Machine Learning for Coders: A Programm…      Laurence Moroney
3                       Machine Learning Engineering        Andriy Burkov
4  Machine Learning: 4 Books in 1: Basic Concepts…         Ethem Mining

  User Rating Reviews     Format   Price
0         4.8   1,808  Paperback  $17.50
1         4.6      66  Paperback  $35.49
2         4.8      29  Paperback  $45.59
3         4.7     113  Paperback  $35.49
4         4.3     106     Kindle   $0.00
```

[19]:
```
# and we also convert the (User Rating) feature data type to numeric
# and then we can impute the missing values in this feature with mean of this␣
 ↪feature
df1["User Rating"] = pd.to_numeric(df1["User Rating"])
```

[20]:
```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 285 entries, 0 to 284
Data columns (total 6 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Name         285 non-null    object
 1   Author       285 non-null    object
 2   User Rating  201 non-null    float64
 3   Reviews      192 non-null    object
 4   Format       256 non-null    object
 5   Price        261 non-null    object
dtypes: float64(1), object(5)
memory usage: 13.5+ KB
```

[21]:
```
df1["User Rating"].isnull().sum()
```

[21]: 84

[22]:
```
# handling missing values of (User Rating) feature with mean
df1["User Rating"] = df1["User Rating"].fillna(np.mean(df1["User Rating"]))
```

[23]:
```
# checking for missing values in (User Rating)
df1["User Rating"].isnull().sum()
```

[23]:
```
0
```

[24]:
```
df1.head()
```

[24]:
```
                                                Name                Author  \
0  Hands-On Machine Learning with Scikit-Learn, K…        Aurélien Géron
1  Machine Learning Design Patterns: Solutions to…  Valliappa Lakshmanan
2  AI and Machine Learning for Coders: A Programm…      Laurence Moroney
3                         Machine Learning Engineering         Andriy Burkov
4  Machine Learning: 4 Books in 1: Basic Concepts…          Ethem Mining

   User Rating Reviews     Format   Price
0          4.8   1,808  Paperback  $17.50
1          4.6      66  Paperback  $35.49
2          4.8      29  Paperback  $45.59
3          4.7     113  Paperback  $35.49
4          4.3     106     Kindle   $0.00
```

[25]:
```
# chnage (Reviews) column data type to numeric
# to convert (Reviews) column data type to numeric first we need to remove␣
↪comma from its values ---> like (1,808)
# otherwise it we give errors

df1["Reviews"] = df1["Reviews"].replace(",", "", regex=True)
```

[26]:
```
# check for remov comma
df1["Reviews"].head()
```

[26]:
```
0    1808
1      66
2      29
3     113
4     106
Name: Reviews, dtype: object
```

[27]:
```
# and now we can change its data type to numeric
df1["Reviews"] = pd.to_numeric(df1["Reviews"])
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 285 entries, 0 to 284
```

```
Data columns (total 6 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Name         285 non-null    object
 1   Author       285 non-null    object
 2   User Rating  285 non-null    float64
 3   Reviews      192 non-null    float64
 4   Format       256 non-null    object
 5   Price        261 non-null    object
dtypes: float64(2), object(4)
memory usage: 13.5+ KB
```

[28]:
```
# missing values of (Reviews) feature
df1["Reviews"].isnull().sum()
```

[28]: 93

[29]:
```
# handling missing values of (Reviews) feature with mean
df1["Reviews"] = df1["Reviews"].fillna(np.mean(df1["Reviews"]))
df1["Reviews"].isnull().sum()
```

[29]: 0

[30]:
```
# missing values of (Format) feature
df1["Format"].isnull().sum()
```

[30]: 29

[33]:
```
# handling missing values of (Format) feature with backwardfill technique (next␣
 ↪Value)
df1["Format"] = df1["Format"].fillna(method="bfill")
df1["Format"].isnull().sum()
```

[33]: 0

[34]:
```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 285 entries, 0 to 284
Data columns (total 6 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Name         285 non-null    object
 1   Author       285 non-null    object
 2   User Rating  285 non-null    float64
 3   Reviews      285 non-null    float64
 4   Format       285 non-null    object
 5   Price        261 non-null    object
```

```
dtypes: float64(2), object(4)
memory usage: 13.5+ KB
```

[35]:
```python
# the (Price) attribute must also be numeric
df1["Price"].head()
```

[35]:
```
0    $17.50
1    $35.49
2    $45.59
3    $35.49
4     $0.00
Name: Price, dtype: object
```

[36]:
```python
# for converting or numeric first we need to remove the dollor sign
def formatPrice(a):
    if type(a) == type(""):
        result = a.split("$")
        return result[1]
    else:
        return np.nan
```

[37]:
```python
df1["Price"] = df1["Price"].apply(formatPrice)
df1["Price"].head()
```

[37]:
```
0    17.50
1    35.49
2    45.59
3    35.49
4     0.00
Name: Price, dtype: object
```

[38]:
```python
# now we can convert to numeric
df1["Price"] = pd.to_numeric(df1["Price"])
```

[39]:
```python
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 285 entries, 0 to 284
Data columns (total 6 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Name         285 non-null    object
 1   Author       285 non-null    object
 2   User Rating  285 non-null    float64
 3   Reviews      285 non-null    float64
 4   Format       285 non-null    object
 5   Price        261 non-null    float64
```

```
dtypes: float64(3), object(3)
memory usage: 13.5+ KB
```

[40]:
```
# (Price) feature missing values
df1["Price"].isnull().sum()
```

[40]:
```
24
```

[41]:
```
# imputing values for (User Rating) attribute useing mean of that attribute

df1["Price"] = df1["Price"].fillna(np.mean(df1["Price"]))
df1["Price"].isnull().sum()
```

[41]:
```
0
```

[42]:
```
# checking for missing values for all data set after handling
df1.isnull().sum()
```

[42]:
```
Name           0
Author         0
User Rating    0
Reviews        0
Format         0
Price          0
dtype: int64
```

[43]:
```
# checking for datatypes
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 285 entries, 0 to 284
Data columns (total 6 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Name         285 non-null    object
 1   Author       285 non-null    object
 2   User Rating  285 non-null    float64
 3   Reviews      285 non-null    float64
 4   Format       285 non-null    object
 5   Price        285 non-null    float64
dtypes: float64(3), object(3)
memory usage: 13.5+ KB
```

2.5 Duplicate Data

"Duplication" just means that you have repeated data in your dataset. This could be due to things like data entry errors or data collection methods

[44]:
```
# checking for duplicates
df1.duplicated().sum()
```

[44]: 6

[45]:
```
# we can see there is 6 books that are duplicated
# we are going to remove those duplicated rows

# First we want see those duplicated rows
df1.loc[df1.duplicated(), :]
```

[45]:
```
                                                  Name          Author  \
68         Machine Learning: New and Collected Stories               ,
89   Funny Data Sciene Gift - AI Data Scientist Mac…      Matt Taddy
117  Data Sciene Gift - Data Scientist Machine Lear…            From
199                                   Machine Learning            From
219  Artificial Intelligence: A Modern Approach (Pe…  Stuart Russell
220  Keras to Kubernetes: The Journey of a Machine …    Dattaraj Rao

     User Rating     Reviews            Format   Price
68       4.50000  167.000000  Audible Audiobook    0.00
89       4.39602  143.677083         Paperback   17.99
117      4.39602  143.677083            Kindle   17.99
199      5.00000    1.000000       Prime Video    1.99
219      4.60000  174.000000         Hardcover  159.99
220      3.80000    6.000000         Paperback   23.47
```

[46]:
```
# these are rows that are duplicated we are going ro remove them
df1 = df1.drop_duplicates()
```

[47]:
```
df1.duplicated().sum()
```

[47]: 0

[48]:
```
# all duplicated rows remove successfully.
```

2.6 Outliers

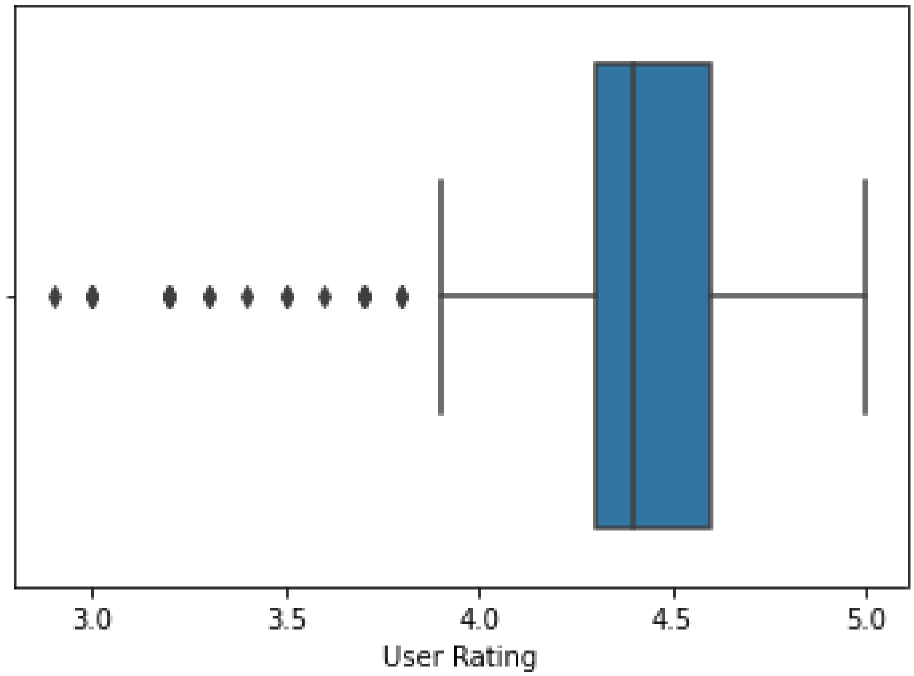
In simple terms, an outlier is an extremely high or extremely low data point relative to the nearest data point and the rest of the neighboring co-existing values in a data graph or dataset.

[49]:
```
# we are checking outliers for numeric attributes
```

[50]:
```
# if we want to see the outliers by graphs we can use seaborn library
```

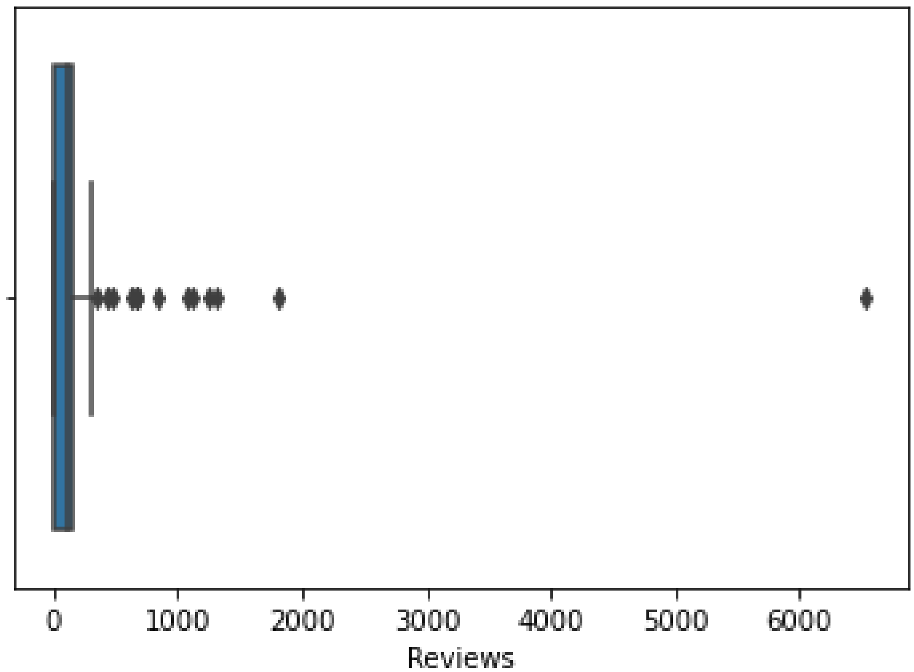
[51]:
```
# visualizing outliers for (User Rating) feature
sns.boxplot(x=df1["User Rating"])
```

[51]: <AxesSubplot:xlabel='User Rating'>



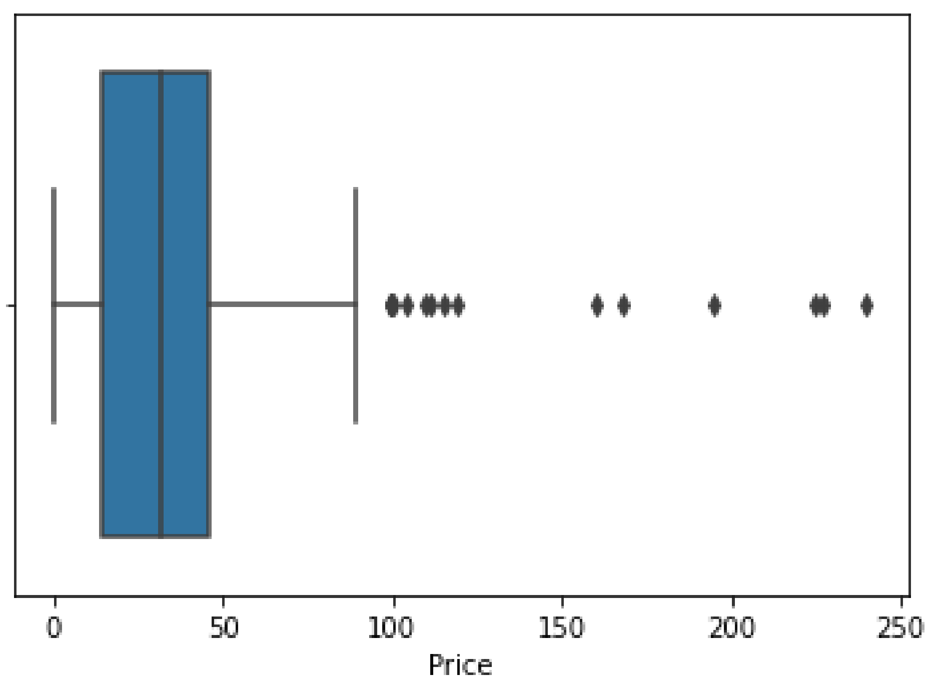
[52]:
```
# visualizing outliers for (Reviews) feature
sns.boxplot(x=df1["Reviews"])
```

[52]: <AxesSubplot:xlabel='Reviews'>



[53]:
```
# visualizing outliers for (Price) feature
sns.boxplot(x=df1["Price"])
```

[53]: <AxesSubplot:xlabel='Price'>



[54]:
```python
# as we can see there is outliers in our dataset
# for removing does outliers we are useing IQR

# function (outliers) returns a list of index of outliers
def outliers(df, ft):
    Q1 = np.percentile(df[ft], 25, interpolation = 'midpoint')
    Q3 = np.percentile(df[ft], 75, interpolation = 'midpoint')
    IQR = Q3 - Q1

    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR

    ls = df.index[ (df[ft] < lower_bound) | (df[ft] > upper_bound) ]

    return ls
```

[55]:
```python
# creating an empty list to store the output indices from multiple columns

index_list = []
for feature in ["User Rating", "Reviews", "Price"]:
    index_list.extend(outliers(df1, feature))
```

[56]:
```python
print(index_list)
```

```
[9, 19, 25, 47, 80, 131, 163, 191, 212, 213, 218, 228, 229, 230, 233, 235, 236,
253, 254, 262, 270, 280, 0, 5, 7, 8, 10, 18, 28, 36, 96, 104, 240, 246, 276,
282, 22, 45, 79, 107, 126, 131, 134, 135, 146, 198, 201, 212, 217, 248, 260]
```

[57]:
```python
# define a function called "remove" which returns a cleaned dataframe without␣
↪ouliers

def remove(df, ls):
    ls = sorted(set(ls))
    df = df.drop(ls)
    return df
```

[58]:
```python
df_cleaned = remove(df1, index_list)
```

[59]:
```python
# dataframe with outliers
df1.shape
```

[59]: (279, 6)

[60]:
```
# after removing outliers

df_cleaned.shape
```

[60]:
```
(230, 6)
```

2.7 Normalization

Data normalization is the method of organizing data to appear similar across all records and fields. Performing so always results in getting higher quality data.

[61]:
```
# normalizing (Reviews) feature -- MinMaxScaler method --
normalizeReviews = df1["Reviews"]
minF = normalizeReviews.min(axis=0)
maxF = normalizeReviews.max(axis=0)

normalizeReviews = normalizeReviews.apply(lambda v: (v-minF) / (maxF-minF))
normalizeReviews
```

[61]:
```
0      0.276553
1      0.009948
2      0.004285
3      0.017141
4      0.016070
         …
280    0.002296
281    0.021836
282    0.191001
283    0.000459
284    0.021836
Name: Reviews, Length: 279, dtype: float64
```

[62]:
```
# max value
normalizeReviews.max()
```

[62]:
```
1.0
```

[63]:
```
# min value
normalizeReviews.min()
```

[63]:
```
0.0
```

2.8 Standardization

Standardization entails scaling data to fit a standard normal distribution. A standard normal distribution is defined as a distribution with a mean of 0 and a standard deviation of 1.

[64]:
```
# standradizing (Reviews) attribute (z-score)
standardizeReviews = df1["Reviews"]
mean = np.mean(standardizeReviews)
SD = np.std(standardizeReviews)

standardizeReviews = standardizeReviews.apply(lambda x: (x-mean) / SD)
standardizeReviews.head()
```

[64]:
```
0    3.852376
1   -0.181767
2   -0.267452
3   -0.072924
4   -0.089135
Name: Reviews, dtype: float64
```

[65]:
```
# standardization using scipy module
```

[66]:
```
import scipy.stats as stats
values = df1["Reviews"]
zscores = stats.zscore(values)
print(zscores.head())
```

```
0    3.852376
1   -0.181767
2   -0.267452
3   -0.072924
4   -0.089135
Name: Reviews, dtype: float64
```

2.9 Cosine Similarity

Cosine Similarity is a measurement that quantifies the similarity between two or more vectors. The cosine similarity is the cosine of the angle between vectors. The vectors are typically non-zero and are within an inner product space.

The cosine similarity is described mathematically as the division between the dot product of vectors and the product of the euclidean norms or magnitude of each vector.

- Applications

1. Document Similarity
2. Pose Matching

[67]:
```
# In Here we use it for finding the documnet similarity
```

[68]:
```
from sklearn.feature_extraction.text import CountVectorizer

def cosineSimilarity(x, y):
```

```
    # Ensure length of x and y are the same
    if len(x) != len(y) :
        return None

    # Compute the dot product between x and y
    dotProduct = np.dot(x, y)

    # Compute the magnitudes of x and y
    magnitude_x = np.sqrt(np.sum(x**2))
    magnitude_y = np.sqrt(np.sum(y**2))

    consine_similarity = dotProduct / (magnitude_x * magnitude_y)

    return consine_similarity
```

[69]:
```
# cosine similarity between two first row
twoBookName = list((df1["Name"][0], df1["Name"][1]))
print(twoBookName)
```

```
['Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts,
Tools, and Techniques to Build Intelligent Systems', 'Machine Learning Design
Patterns: Solutions to Common Challenges in Data Preparation, Model Building,
and MLOps']
```

[70]:
```
# Create a matrix to represent the twoBookName
victorize_list = CountVectorizer().fit_transform(twoBookName).toarray()
print(victorize_list)
```

```
[[2 1 0 0 0 1 0 0 1 0 1 1 1 1 1 0 0 1 0 0 1 0 1 1 1 1 1 1]
 [1 0 1 1 1 0 1 1 0 1 0 0 0 1 1 1 1 0 1 1 0 1 0 0 0 1 0 0]]
```

[71]:
```
cos_sim_2BookName = cosineSimilarity(victorize_list[0], victorize_list[1])
```

[72]:
```
print('Cosine Similarity between: ')
print('Book Name 1 and Book Name 2: ', cos_sim_2BookName)
```

```
Cosine Similarity between:
Book Name 1 and Book Name 2:  0.28867513459481287
```

2.10 Euclidean distance

Euclidean distance calculates the distance between two real-valued vectors. You are most likely to use Euclidean distance when calculating the distance between two rows of data that have numerical values, such a floating point or integer values.

[73]:
```
def euclideanDistance(point1, point2):
```

```
    euclidean_distance = np.sqrt(np.sum(np.square(point1 - point2)))

    return euclidean_distance
```

[74]:
```
# euclidean distance between numeric value of the row 1 and 2 of the dataset
# (User Rating, Reviews)
point1 = np.array((df1["User Rating"][0], df1["Reviews"][0]))
point2 = np.array((df1["User Rating"][1], df1["Reviews"][1]))


euclidean_dis_1_2 = euclideanDistance(point1, point2)
print("Euclidean Distance Between:")
print("Book 1 and Book 2: ", euclidean_dis_1_2)
```

```
Euclidean Distance Between:
Book 1 and Book 2:  1742.0000114810562
```

2.11 City block or Manhattan distance

The Manhattan distance, often called Taxicab distance or City Block distance, calculates the distance between real-valued vectors. Imagine vectors that describe objects on a uniform grid such as a chessboard. Manhattan distance then refers to the distance between two vectors if they could only move right angles.

[75]:
```
def cityBlock(point1, point2):

    city_block = np.sum(np.abs(point1 - point2))

    return city_block
```

[76]:
```
# city block distance between numeric value of the row 1 and 2 of the dataset
# (User Rating, Reviews, Price)

point1 = np.array((df1["User Rating"][0], df1["Reviews"][0], df1["Price"][0]))
point2 = np.array((df1["User Rating"][1], df1["Reviews"][1], df1["Price"][1]))

cityBlock_dis_1_2 = cityBlock(point1, point2)
print("City block Distance Between:")
print("Book 1 and Book 2: ", cityBlock_dis_1_2)
```

```
City block Distance Between:
Book 1 and Book 2:  1760.19
```

2.12 Visualization

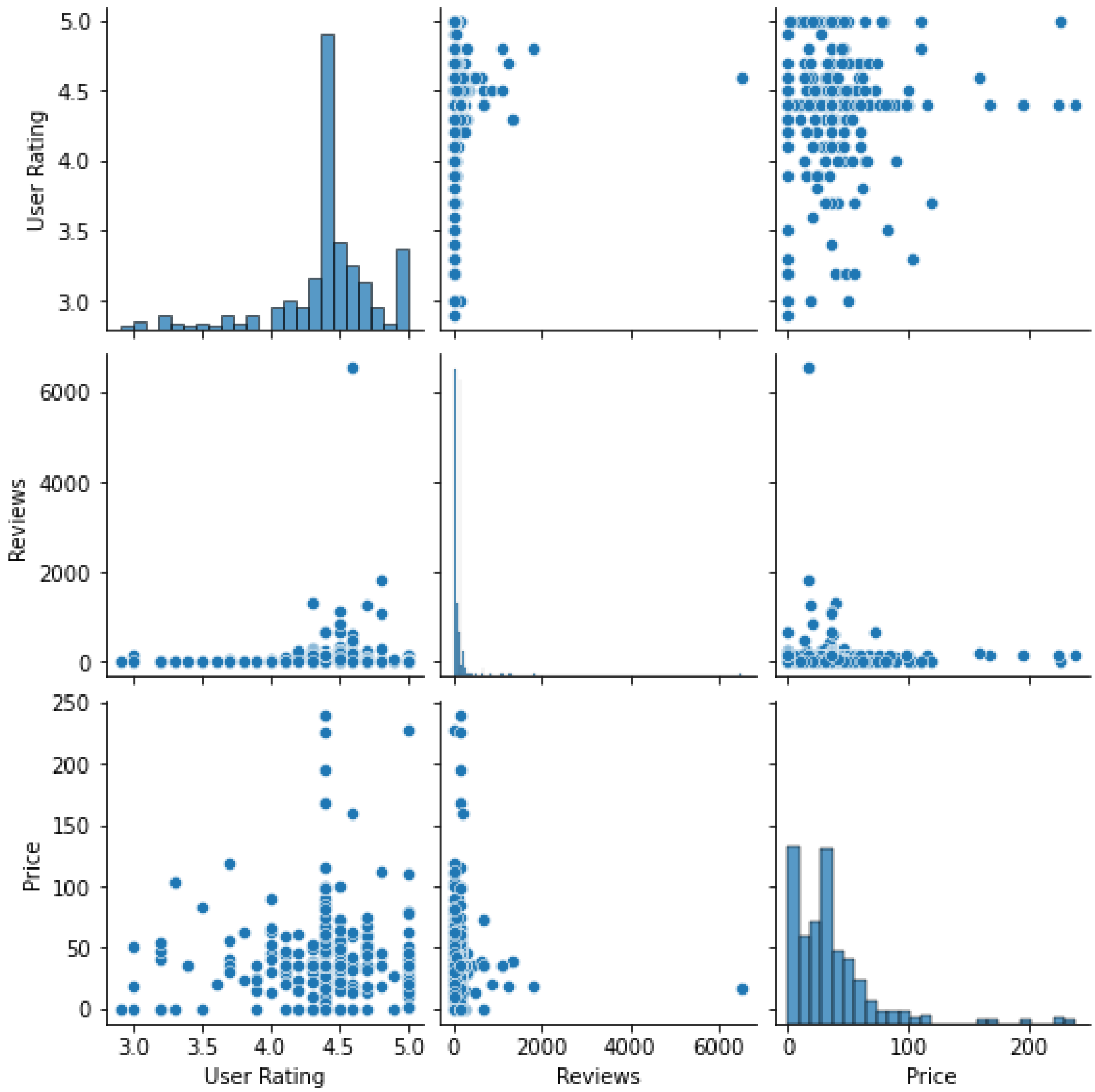
Data visualization is the graphical representation of information and data in a pictorial or graphical format(Example: charts, graphs, and maps). Data visualization tools provide an accessible way to see and understand trends, patterns in data, and outliers.

In applied Statistics and Machine Learning, Data Visualization is one of the most important skills. Data visualization provides an important suite of tools for identifying a qualitative understanding

[103]:
```
# in here we also visualize our data to deep undrestand our data
```

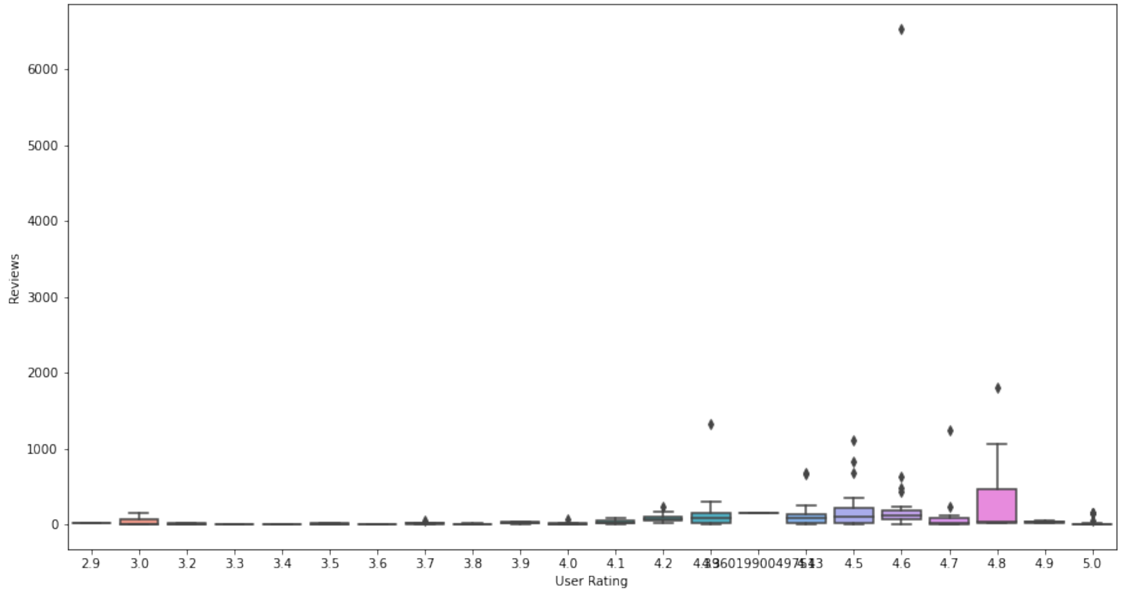
[104]:
```
sns.pairplot(df1)
```

[104]: <seaborn.axisgrid.PairGrid at 0x21ab4c2a850>



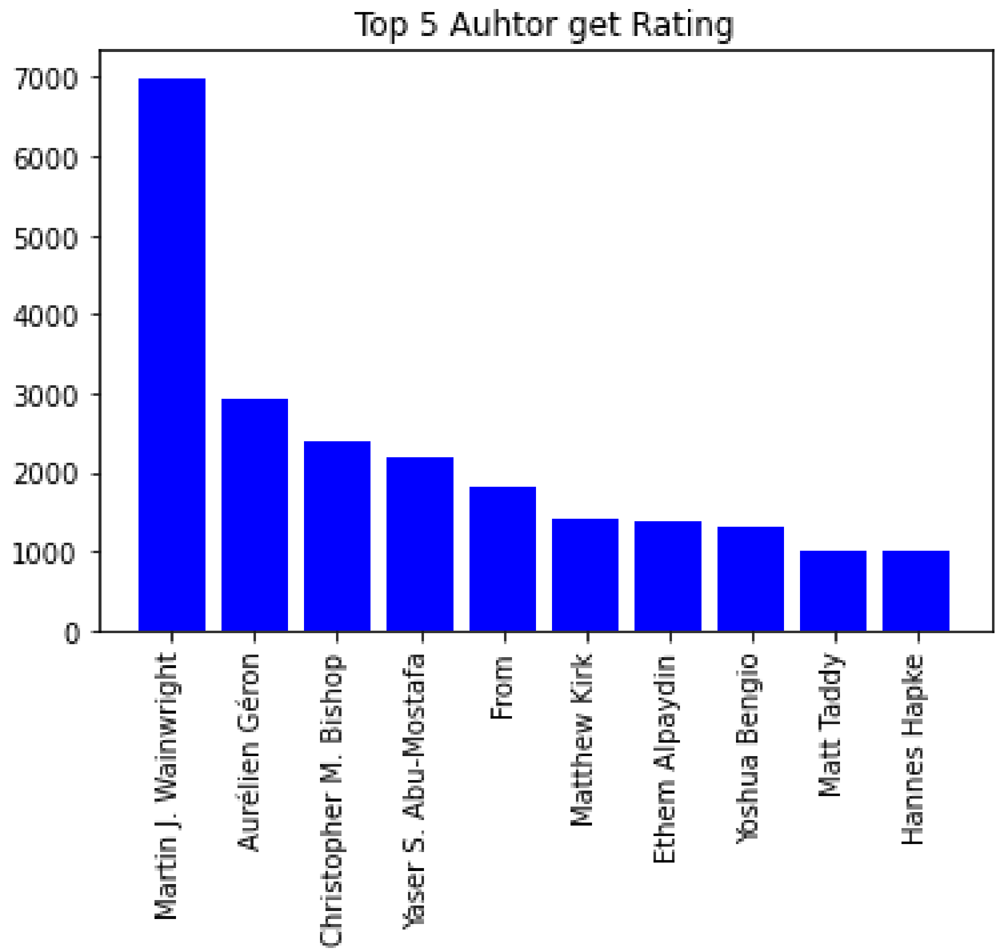
[105]:
```
plt.figure(figsize=(15,8))
sns.boxplot(x=df1['User Rating'],y=df1['Reviews'])
```

[105]: <AxesSubplot:xlabel='User Rating', ylabel='Reviews'>



[106]:
```
x=df1.groupby('Author')['Reviews'].agg([sum]).
 ↪sort_values(by=('sum'),ascending=False).head(10)
q3=pd.DataFrame(x)
q3.reset_index(inplace=True)
plt.bar(q3['Author'],q3['sum'],color="blue")
plt.title("Top 5 Auhtor get Rating")
plt.xticks(rotation=90)
plt.figure(figsize=(18,18))
```

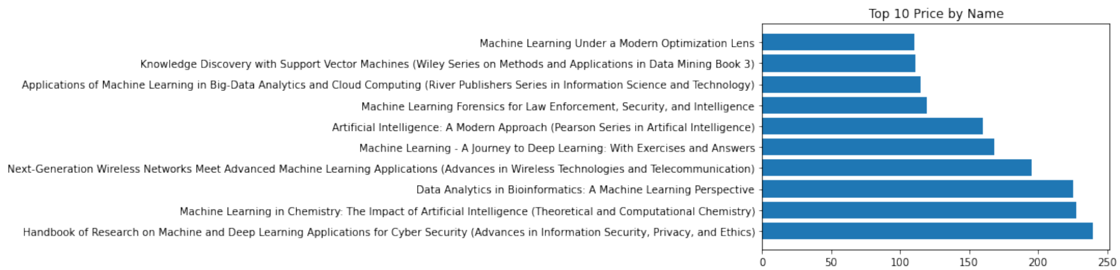
[106]: <Figure size 1296x1296 with 0 Axes>




```
<Figure size 1296x1296 with 0 Axes>
```

[107]:
```
x=df1.groupby('Name')['Price'].agg([sum]).
 ↪sort_values(by=('sum'),ascending=False).head(10)
q4=pd.DataFrame(x)
q4.reset_index(inplace=True)
plt.barh(q4['Name'],q4['sum'])
plt.title("Top 10 Price by Name")
# plt.xticks(rotation=90)
plt.figure(figsize=(18,18))
```

[107]:
```
<Figure size 1296x1296 with 0 Axes>
```




```
<Figure size 1296x1296 with 0 Axes>
```

2.13 Adding New Column

since all the books in the above dataset are Machine learning books we are going to add a new column with name of (Category) to show the category of the books because we need it when we want to predict whether a book is a machine learing book or not.

[108]:
```
df1["Category"] = "Machine Learning"
```

[109]:
```
df1.head()
```

[109]:
```
                                                Name                 Author  \
0  Hands-On Machine Learning with Scikit-Learn, K…         Aurélien Géron
1  Machine Learning Design Patterns: Solutions to…  Valliappa Lakshmanan
2  AI and Machine Learning for Coders: A Programm…      Laurence Moroney
3                       Machine Learning Engineering          Andriy Burkov
4  Machine Learning: 4 Books in 1: Basic Concepts…          Ethem Mining

   User Rating  Reviews     Format  Price          Category
0          4.8   1808.0  Paperback  17.50  Machine Learning
1          4.6     66.0  Paperback  35.49  Machine Learning
2          4.8     29.0  Paperback  45.59  Machine Learning
3          4.7    113.0  Paperback  35.49  Machine Learning
4          4.3    106.0     Kindle   0.00  Machine Learning
```

[110]:
```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 279 entries, 0 to 284
Data columns (total 7 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Name         279 non-null    object
 1   Author       279 non-null    object
 2   User Rating  279 non-null    float64
 3   Reviews      279 non-null    float64
```

```
 4   Format       279 non-null    object
 5   Price        279 non-null    float64
 6   Category     279 non-null    object
dtypes: float64(3), object(4)
memory usage: 25.5+ KB
```

2.14 Data Selection

Removing unnecessary columns: for our classification model we just need the (Name) column and (Category) column we are removing all other columns

[111]:
```python
# we first copy our dataframe
# and then remove the unnecessary cloumns from it
new_df1 = df1[:]

del new_df1["Author"]
del new_df1["User Rating"]
del new_df1["Reviews"]
del new_df1["Format"]
del new_df1["Price"]
```

[112]:
```python
new_df1.head()
```

[112]:

	Name	Category
0	Hands-On Machine Learning with Scikit-Learn, K…	Machine Learning
1	Machine Learning Design Patterns: Solutions to…	Machine Learning
2	AI and Machine Learning for Coders: A Programm…	Machine Learning
3	Machine Learning Engineering	Machine Learning
4	Machine Learning: 4 Books in 1: Basic Concepts…	Machine Learning

[113]:
```python
new_df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 279 entries, 0 to 284
Data columns (total 2 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Name      279 non-null    object
 1   Category  279 non-null    object
dtypes: object(2)
memory usage: 6.5+ KB
```

2.15 Data Integration

For our model to work well we are going to merge a new Amazon.com books dataset that contain books other then Machine learning books.

[129]:
```
df2 = pd.read_csv(r"Downloads\CSVFiles\amazonBooks.csv")
df2.head()
```

[129]:
```
                                                Name  \
0                       10-Day Green Smoothie Cleanse
1                                   11/22/63: A Novel
2             12 Rules for Life: An Antidote to Chaos
3                              1984 (Signet Classics)
4  5,000 Awesome Facts (About Everything!) (Natio…

                     Author  User Rating  Reviews  Price  Year        Genre
0                  JJ Smith          4.7    17350      8  2016  Non Fiction
1              Stephen King          4.6     2052     22  2011      Fiction
2        Jordan B. Peterson          4.7    18979     15  2018  Non Fiction
3             George Orwell          4.7    21424      6  2017      Fiction
4  National Geographic Kids          4.8     7665     12  2019  Non Fiction
```

[130]:
```
# first we remove the unnecessary features

del df2["Author"]
del df2["User Rating"]
del df2["Reviews"]
del df2["Price"]
del df2["Year"]
del df2["Genre"]
```

[131]:
```
df2.head()
```

[131]:
```
                                                Name
0                       10-Day Green Smoothie Cleanse
1                                   11/22/63: A Novel
2             12 Rules for Life: An Antidote to Chaos
3                              1984 (Signet Classics)
4  5,000 Awesome Facts (About Everything!) (Natio…
```

[132]:
```
# cleaning the second dataframe

# checking for missing value
df2.isnull().sum()
```

[132]:
```
Name    0
dtype: int64
```

[133]:
```
# checking for duplicates
df2["Name"] = df2["Name"].str.title().str.strip()
df2["Name"].duplicated().sum()
```

[133]: 200

[134]:
```
# droping duplicates
df2 = df2.drop_duplicates()
```

[135]:
```
df2["Name"].duplicated().sum()
```

[135]: 0

[136]:
```
# add the (Category) feature into the second dataframe
df2["Category"] = "Others"
```

[137]:
```
df2.shape
```

[137]: (350, 2)

[138]:
```
df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 350 entries, 0 to 546
Data columns (total 2 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Name      350 non-null    object
 1   Category  350 non-null    object
dtypes: object(2)
memory usage: 8.2+ KB
```

[139]:
```
# outer: use union of keys from both frames, similar to a SQL full outer join;␣
 ↪sort keys lexicographically.
final_df = pd.merge(new_df1, df2, how="outer")
final_df.head()
```

[139]:
```
                                                Name          Category
0  Hands-On Machine Learning with Scikit-Learn, K…  Machine Learning
1  Machine Learning Design Patterns: Solutions to…  Machine Learning
2  AI and Machine Learning for Coders: A Programm…  Machine Learning
3                      Machine Learning Engineering  Machine Learning
4  Machine Learning: 4 Books in 1: Basic Concepts…  Machine Learning
```

[140]:
```
final_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 629 entries, 0 to 628
Data columns (total 2 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Name      629 non-null    object
```

 1   Category 629 non-null   object
dtypes: object(2)
memory usage: 14.7+ KB

[141]:
```
# checking for missing values in merge dataframe
final_df.isnull().sum()
```

[141]:
```
Name        0
Category    0
dtype: int64
```

[142]:
```
# checking for duplicate values in merge dataframe
final_df.duplicated().sum()
```

[142]: 5

[143]:
```
final_df = final_df.drop_duplicates()
```

[144]:
```
final_df.duplicated().sum()
```

[144]: 0

[145]:
```
final_df.shape
```

[145]: (624, 2)

2.16 Shuffling dataframe

We are going to suffle our new dataframe because now all books from one category place togather, it mean all rows in the data frame are order based on category, its because when we merge the two dataframe the second data frame place on the end of the first dataframe.

[146]:
```
# the frac keyword argument specifies the fraction of rows to retrun in the␣
↪random sample,
# so frac=1 means retrun all rows in random order.
# specifying drop=True prevents .reset_index from creating a column cantaining␣
↪the old index entries.
final_df = final_df.sample(frac=1).reset_index(drop=True)
```

[147]:
```
final_df.shape
```

[147]: (624, 2)

[148]:
```
final_df.head()
```

[148]:

	Name	Category
0	Calm The F*Ck Down: An Irreverent Adult Colori…	Others
1	Toy Hammer w/ Lights, Learning Mode and Music …	Machine Learning

```
2                             The Secret Life of Chaos   Machine Learning
3                                          Born To Run             Others
4  An Introduction to Variational Autoencoders (F…  Machine Learning
```

[149]:
```python
final_df.tail()
```

[149]:
```
                                                     Name          Category
619                                    The 5000 Year Leap            Others
620  Machine Learning: 2 Books in 1: An Introductio…  Machine Learning
621                           Python Machine Learning  Machine Learning
622  Unbroken: A World War Ii Story Of Survival, Re…             Others
623  Machine Learning and AI for Healthcare: Big Da…  Machine Learning
```

3 Modeling

As we mention at the beginning we are going to build a model to classify a book whether it is a machine learning book or not by the use of book names, for this task to be done we are using a machine learning algorithm called Naive Bayes, if we want to be specific we are using Multinomial Naive Bayes.

3.1 Multinomial Naive Bayes

The Multinomial Naive Bayes algorithm is a Bayesian learning approach popular in Natural Language Processing (NLP). The program guesses the tag of a text, such as an email or a newspaper story, using the Bayes theorem. It calculates each tag's likelihood for a given sample and outputs the tag with the greatest chance.

3.2 Model Development

[150]:
```python
# importing libraries
import string
from sklearn.feature_extraction.text import CountVectorizer
from nltk.corpus import stopwords
```

[151]:
```python
# create transform
vectorizer = CountVectorizer()
```

[152]:
```python
def text_cleaning(a):

    # deleting or removing punctuations it will split to charecters
    remove_punctuation = [char for char in a if char not in string.punctuation]

    # join does charecters
    remove_punctuation = ''.join(remove_punctuation)

    # agin spliting or tokenizeing the words and removing the stop words and␣
↪return them as a list
```

```
    return [word for word in remove_punctuation.split() if word.lower() not in␣
↪stopwords.words("english")]
```

[153]:
```
final_df.head()
```

[153]:
```
                                                Name          Category
0  Calm The F*Ck Down: An Irreverent Adult Colori…             Others
1  Toy Hammer w/ Lights, Learning Mode and Music …  Machine Learning
2                          The Secret Life of Chaos  Machine Learning
3                                       Born To Run            Others
4  An Introduction to Variational Autoencoders (F…  Machine Learning
```

[154]:
```
# data after removing punctuations and stop words
print(final_df.iloc[:,0].apply(text_cleaning))
```

```
0       [Calm, FCk, Irreverent, Adult, Coloring, Book,…
1       [Toy, Hammer, w, Lights, Learning, Mode, Music…
2                              [Secret, Life, Chaos]
3                                         [Born, Run]
4       [Introduction, Variational, Autoencoders, Foun…
                              …
619                                [5000, Year, Leap]
620     [Machine, Learning, 2, Books, 1, Introduction,…
621                         [Python, Machine, Learning]
622     [Unbroken, World, War, Ii, Story, Survival, Re…
623     [Machine, Learning, AI, Healthcare, Big, Data,…
Name: Name, Length: 624, dtype: object
```

[155]:
```
bow_transformer = CountVectorizer(analyzer=text_cleaning).fit(final_df["Name"])

bow_transformer.vocabulary_
```

[155]:
```
{'Calm': 251,
 'FCk': 548,
 'Irreverent': 821,
 'Adult': 71,
 'Coloring': 312,
 'Book': 213,
 'Series': 1393,
 'Toy': 1597,
 'Hammer': 697,
 'w': 1852,
 'Lights': 921,
 'Learning': 890,
 'Mode': 1025,
 'Music': 1044,
 '-': 1854,
```

```
'Baby': 158,
'Plays': 1182,
'6': 43,
'Short': 1414,
'Kids': 853,
'Songs': 1447,
'Counts': 359,
'110': 5,
'Changes': 281,
'Funny': 621,
'Expressions': 545,
'12': 8,
'Months': 1032,
'Older': 1093,
'Secret': 1383,
'Life': 912,
'Chaos': 282,
'Born': 216,
'Run': 1348,
'Introduction': 815,
'Variational': 1663,
'Autoencoders': 144,
'Foundations': 606,
'Trendsr': 1610,
'Machine': 951,
'Towers': 1595,
'Midnight': 1005,
'Wheel': 1695,
'Time': 1575,
'Thirteen': 1564,
'Second': 1382,
'Bree': 229,
'Tanner': 1534,
'Eclipse': 488,
'Novella': 1080,
'Twilight': 1625,
'Saga': 1357,
'Robotics': 1338,
'AI': 52,
'Evolution': 530,
'Robot': 1336,
'TShirt': 1527,
'Cloud': 303,
'Computing': 334,
'Killing': 857,
'Kennedy': 844,
'End': 505,
```

'Camelot': 253,
'PoutPout': 1195,
'Fish': 588,
'Last': 879,
'Week': 1691,
'Tonight': 1589,
'John': 831,
'Oliver': 1095,
'Presents': 1212,
'Day': 408,
'Marlon': 971,
'Bundo': 241,
'Better': 194,
'Lgbt': 908,
'Children\x92S': 287,
'Homebody': 741,
'Guide': 684,
'Creating': 370,
'Spaces': 1451,
'Never': 1063,
'Want': 1680,
'Leave': 892,
'Designs': 430,
'Stress': 1495,
'Relief': 1306,
'Garden': 628,
'Mandalas': 964,
'Animals': 113,
'Paisley': 1128,
'Patterns': 1147,
'Data': 402,
'Management': 962,
'Model': 1026,
'Training': 1603,
'Neural': 1061,
'Networks': 1060,
'Algorithms': 92,
'Naive': 1049,
'Bayes': 172,
'Classifier': 299,
'Tutorial': 1622,
'House': 746,
'Hades': 692,
'Heroes': 728,
'Olympus': 1099,
'4': 33,
'Tinker': 1579,

```
'Toddlers': 1586,
'second': 1828,
'edition': 1768,
'Adaptive': 68,
'Computation': 329,
'series': 1832,
'Artificial': 128,
'Intelligence': 804,
'Practice': 1201,
'50': 38,
'Successful': 1502,
'Companies': 323,
'Used': 1653,
'Solve': 1442,
'Problems': 1227,
'R': 1272,
'5': 37,
'Love': 941,
'Languages': 876,
'Lasts': 880,
'Catching': 264,
'Fire': 584,
'Hunger': 752,
'Games': 627,
'Sycamore': 1522,
'Row': 1344,
'Jake': 824,
'Brigance': 230,
'Girl': 649,
'Dragon': 466,
'Tattoo': 1536,
'Millennium': 1010,
'Beginner’s': 187,
'Mining': 1015,
'Big': 196,
'Instant': 801,
'Pot': 1190,
'Pressure': 1216,
'Cooker': 352,
'Cookbook': 350,
'500': 39,
'Everyday': 526,
'Recipes': 1291,
'Beginners': 185,
'Advanced': 73,
'Users': 1655,
'Try': 1621,
```

'Easy': 484,
'Healthy…': 718,
'Frozen': 615,
'Little': 929,
'Golden': 664,
'Graph': 674,
'Representation': 1309,
'Synthesis': 1524,
'Lectures': 895,
'Complete': 326,
'Delphi': 425,
'programming': 1820,
'Developer': 434,
'2021': 24,
'Zero': 1733,
'Mastery': 976,
'become': 1750,
'get': 1782,
'hired': 1788,
'Build': 237,
'projects': 1821,
'learn': 1797,
'Light': 920,
'Cannot': 255,
'See': 1386,
'HighDimensional': 730,
'Statistics': 1470,
'NonAsymptotic': 1073,
'Viewpoint': 1668,
'Divergent': 452,
'Stop': 1484,
'Apologizing': 117,
'ShameFree': 1408,
'Plan': 1174,
'Embracing': 503,
'Achieving': 61,
'Goals': 658,
'Knowledge': 867,
'Discovery': 448,
'Support': 1510,
'Vector': 1664,
'Machines': 952,
'Wiley': 1701,
'Methods': 1002,
'Applications': 118,
'3': 26,
'Gaussian': 632,

'Processes': 1228,
'Four': 607,
'Agreements': 81,
'Practical': 1200,
'Personal': 1160,
'Freedom': 610,
'Toltec': 1588,
'Wisdom': 1707,
'Harry': 708,
'Potter': 1192,
'Goblet': 659,
'Illustrated': 772,
'Edition': 491,
'Rush': 1351,
'Revere': 1321,
'First': 586,
'Patriots': 1144,
'TimeTravel': 1576,
'Adventures': 75,
'Exceptional': 536,
'Americans': 105,
'2': 20,
'Forever': 602,
'Automated': 146,
'Systems': 1526,
'Challenges': 276,
'Springer': 1458,
'Computer': 331,
'Science': 1369,
'Nerd': 1057,
'Gift': 645,
'Raglan': 1278,
'Baseball': 166,
'Tee': 1545,
'Reagan': 1285,
'Violent': 1672,
'Assault': 131,
'Changed': 280,
'Presidency': 1213,
'Bill': 199,
'OReillyS': 1085,
'Deep': 418,
'Structured': 1496,
'Fine': 583,
'Childrens': 286,
'Story': 1487,
'SongPoem': 1446,

```
'Player': 1180,
'English': 509,
'Educational': 495,
'Babies': 157,
'GiftToddler': 646,
'Birthday': 205,
'Subtle': 1500,
'Art': 126,
'Giving': 653,
'Counterintuitive': 357,
'Approach': 120,
'Living': 932,
'Good': 668,
'Dome': 459,
'Novel': 1079,
'Game': 626,
'Thrones': 1571,
'Boxed': 220,
'Set': 1399,
'ThronesA': 1572,
'Clash': 294,
'KingsA': 859,
'Storm': 1486,
'SwordsA': 1521,
'Feast': 569,
'Crows': 380,
'ES6': 480,
'Aprende': 122,
'en': 1770,
'Español': 516,
'Teoría': 1551,
'Práctica': 1245,
'Python': 1259,
'Spanish': 1452,
'Beasts': 175,
'Terror': 1552,
'American': 104,
'Family': 560,
'HitlerS': 736,
'Berlin': 192,
'Rules': 1346,
'Antidote': 115,
'Pieces': 1167,
'Construction': 345,
'Vehicles': 1666,
'Trucks': 1615,
'Bulldozer': 239,
```

```
'Cement': 271,
'Mixer': 1021,
'Dumper': 475,
'Forklift': 603,
'Excavator': 535,
'Road': 1333,
'Roller': 1341,
'Contractor': 348,
'Push': 1255,
'Go': 657,
'Sliding': 1431,
'Toys': 1598,
'18m': 16,
'Nextjs': 1067,
'Crazy': 368,
'Rich': 1325,
'Asians': 129,
'Trilogy': 1612,
'Comparative': 324,
'Study': 1499,
'Visualization': 1676,
'Techniques': 1542,
'Guts': 686,
'Meltdown': 997,
'Diary': 438,
'Wimpy': 1702,
'Kid': 852,
'13': 9,
'Blueprints': 209,
'Finance': 579,
'Building': 238,
'Trading': 1599,
'Strategies': 1489,
'RoboAdvisors': 1335,
'Using': 1656,
'Amateur': 99,
'Sookie': 1448,
'Stackhouse': 1460,
'Legend': 898,
'Zelda': 1732,
'Hyrule': 761,
'Historia': 734,
'Retooling': 1317,
'Poverty': 1196,
'Targeting': 1535,
'OutofSample': 1115,
'Validation': 1659,
```

```
'Long': 934,
'Haul': 710,
'Diagnostic': 437,
'Statistical': 1469,
'Manual': 967,
'Mental': 1001,
'Disorders': 450,
'5Th': 42,
'Dsm5': 471,
'HandsOn': 701,
'C': 245,
'train': 1842,
'deploy': 1764,
'endtoend': 1771,
'machine': 1799,
'learning': 1798,
'deep': 1761,
'pipelines': 1813,
'Mathematics': 980,
'Technology': 1544,
'Alexander': 88,
'Hamilton': 696,
'Hbase': 712,
'Database': 405,
'Crawdads': 366,
'Sing': 1423,
'Programmer': 1234,
'Natural': 1051,
'Language': 875,
'Processing': 1229,
'Action': 64,
'Understanding': 1641,
'analyzing': 1743,
'generating': 1781,
'text': 1840,
'SQL': 1354,
'Generate': 634,
'Manipulate': 966,
'Retrieve': 1319,
'Happened': 702,
'National': 1050,
'Geographic': 638,
'1111': 6,
'Answers': 114,
'Everything': 528,
'Dog': 458,
'Man': 960,
```

'Brawl': 226,
'Wild': 1700,
'Creator': 373,
'Captain': 257,
'Underpants': 1639,
'Beginners’': 186,
'Scratch': 1376,
'Fantastic': 561,
'Find': 581,
'Original': 1107,
'Screenplay': 1377,
'Business': 242,
'Combining': 315,
'Economics': 489,
'Optimize': 1103,
'Automate': 145,
'Accelerate': 57,
'Decisions': 415,
'Milk': 1009,
'Vine': 1670,
'Inspirational': 799,
'Quotes': 1271,
'Classic': 296,
'Vines': 1671,
'Dead': 410,
'Reckoning': 1292,
'StackhouseTrue': 1461,
'Blood': 207,
'11': 4,
'Puppy': 1253,
'Paw': 1149,
'Patrol': 1145,
'Rust': 1352,
'Capital': 256,
'Twenty': 1624,
'Century': 273,
'Becoming': 180,
'Five': 590,
'Express': 544,
'Heartfelt': 719,
'Commitment': 320,
'Mate': 977,
'Interview': 811,
'Questions': 1265,
'Question': 1264,
'Bank': 164,
'Crack': 363,

```
'EmberJS': 502,
'Called': 248,
'Ove': 1116,
'Howard': 747,
'Stern': 1477,
'Comes': 316,
'Math': 978,
'Sagas': 1358,
'HandmaidS': 700,
'Tale': 1532,
'Breaking': 227,
'Dawn': 407,
'Happy': 704,
'Legacy': 897,
'Duck': 472,
'Commander': 318,
'Joyland': 837,
'Hard': 706,
'Case': 260,
'Crime': 376,
'Ship': 1412,
'Fools': 599,
'Selfish': 1387,
'Ruling': 1347,
'Class': 295,
'Bringing': 231,
'America': 102,
'Brink': 232,
'Revolution': 1323,
'Cravings': 365,
'Food': 596,
'Eat': 485,
'Lecture': 894,
'Mastering': 975,
'French': 612,
'Cooking': 353,
'Vol': 1677,
'Mikes': 1007,
'Peanuts': 1151,
'Linear': 924,
'Regression': 1300,
'Example': 532,
'intelligent': 1793,
'systems': 1838,
'using': 1846,
'TensorFlow': 1548,
'PyTorch': 1257,
```

```
'scikitlearn': 1826,
'3rd': 32,
'Wrinkle': 1724,
'Quintet': 1269,
'Shred': 1417,
'Revolutionary': 1324,
'Diet': 439,
'Weeks': 1692,
'Inches': 781,
'Sizes': 1427,
'Biomedical': 203,
'Health': 715,
'Informatics': 792,
'Studies': 1498,
'68': 45,
'Stolen': 1482,
'Memoir': 998,
'Lincoln': 923,
'Shocking': 1413,
'Assassination': 130,
'Programming': 1236,
'Crash': 364,
'Course': 361,
'Perfect': 1156,
'Skills': 1428,
'Coding': 306,
'Project': 1237,
'Even': 523,
'Absolute': 55,
'Beginner': 183,
'Learn': 889,
'Master': 973,
'Best': 193,
'Thug': 1573,
'Kitchen': 861,
'Official': 1089,
'Like': 922,
'Give': 652,
'Cookbooks': 351,
'Essentials': 518,
'Pattern': 1146,
'Recognition': 1293,
'Accessible': 58,
'Percy': 1155,
'Jackson': 823,
'Olympians': 1097,
'Paperback': 1132,
```

```
'Books': 214,
'Kings': 858,
'Swords': 1520,
'Dance': 396,
'Dragons': 467,
'Rigorous': 1328,
'Mathematical': 979,
'Analysis': 107,
'Models': 1028,
'Google': 670,
'Platform': 1177,
'Comprehensive': 328,
'Barefoot': 165,
'Contessa': 346,
'Fabulous': 552,
'Tips': 1582,
'George': 640,
'WashingtonS': 1684,
'Six': 1426,
'Spy': 1459,
'Ring': 1329,
'Saved': 1365,
'Applied': 119,
'Fitness': 589,
'Vision': 1673,
'Sensors': 1390,
'IoT': 820,
'1': 0,
'Swift': 1518,
'Fundamental': 619,
'Theory': 1558,
'Development': 435,
'AIDriven': 53,
'Apps': 121,
'Awesome': 151,
'Projects': 1238,
'20': 21,
'STEAM': 1355,
'Robots': 1339,
'Circuits': 293,
'Design': 428,
'Activities': 65,
'Humans': 750,
'New': 1064,
'York': 1729,
'Wash': 1683,
'Face': 553,
```

```
'Believing': 189,
'Lies': 911,
'Become': 178,
'Meant': 992,
'Network': 1059,
'Wonky': 1714,
'Donkey': 461,
'Looking': 935,
'Alaska': 85,
'Thief': 1559,
'Relieving': 1307,
'Dover': 465,
'Creative': 372,
'Nouveau': 1078,
'Animal': 112,
'AWS': 54,
'SageMaker': 1359,
'Apache': 116,
'Spark': 1453,
'Modeling': 1027,
'techniques': 1839,
'solve': 1837,
'RealWorld': 1288,
'problems': 1819,
'gain': 1780,
'valuable': 1847,
'insights': 1791,
'data': 1760,
'Quiet': 1268,
'Power': 1197,
'Introverts': 816,
'World': 1720,
'CanT': 254,
'Talking': 1533,
'Hour': 745,
'Body': 211,
'Uncommon': 1635,
'Rapid': 1282,
'Fat': 564,
'Loss': 939,
'Incredible': 784,
'Sex': 1403,
'Superhuman': 1508,
'Paris': 1135,
'Wife': 1699,
'Grokking': 681,
'Lost': 940,
```

```
'Symbol': 1523,
'Thousands': 1568,
'Simple': 1422,
'Swaps': 1517,
'Save': 1364,
'10': 1,
'30': 27,
'PoundsOr': 1194,
'Brave': 225,
'Pilgrims': 1168,
'Obama': 1086,
'Intimate': 813,
'Portrait': 1189,
'Red': 1294,
'Pyramid': 1258,
'Kane': 840,
'Chronicles': 291,
'112263': 7,
'10Day': 3,
'Green': 679,
'Smoothie': 1435,
'Cleanse': 300,
'Keras': 845,
'Kubernetes': 868,
'Journey': 835,
'Production': 1231,
'Prisoner': 1221,
'Azkaban': 152,
'Played': 1179,
'Scientists': 1373,
'Gentleman': 637,
'Moscow': 1036,
'Algebra': 89,
'Optimization': 1102,
'Textbook': 1555,
'David': 406,
'Goliath': 666,
'Underdogs': 1638,
'Misfits': 1017,
'Battling': 171,
'Giants': 644,
'Song': 1445,
'Ice': 764,
'Quantitative': 1261,
'Investment': 818,
'Moon': 1033,
'Implementing': 777,
```

```
'EndtoEnd': 506,
'RealTime': 1287,
'Pipelines': 1170,
'Ingest': 794,
'Mortal': 1035,
'Medicine': 994,
'Matters': 984,
'Fault': 566,
'Stars': 1462,
'Coded': 304,
'Mitigating': 1020,
'Bottlenecks': 218,
'Largescale': 878,
'Distributed': 451,
'Communications': 322,
'Information': 793,
'Liberty': 909,
'Tyranny': 1627,
'Conservative': 343,
'Manifesto': 965,
'Strengthsfinder': 1494,
'Forensics': 600,
'Law': 882,
'Enforcement': 507,
'Security': 1385,
'Publication': 1248,
'Psychological': 1246,
'Association': 133,
'6Th': 46,
'Hillbilly': 733,
'Elegy': 499,
'Culture': 384,
'Crisis': 378,
'Top': 1592,
'Discovering': 447,
'Analytics': 108,
'Cases': 261,
'Overview': 1117,
'Fires': 585,
'Everywhere': 529,
'Governance': 671,
'Definitive': 421,
'People': 1154,
'Tools': 1591,
'Operationalize': 1101,
'Trustworthiness': 1619,
'Make': 956,
```

```
'Bed': 181,
'Things': 1560,
'Change': 279,
'LifeAnd': 913,
'Maybe': 985,
'Probabilistic': 1224,
'Perspective': 1161,
'Basic': 167,
'Concepts': 336,
'Intelligent': 805,
'Libraries': 910,
'Two': 1626,
'Kitties': 863,
'Magnolia': 955,
'Table': 1528,
'Collection': 309,
'Gathering': 630,
'PythonBased': 1260,
'Adults': 72,
'Featuring': 571,
'Henna': 725,
'Inspired': 800,
'Flowers': 595,
'Paisley…': 1129,
'Owls': 1119,
'Going': 661,
'Rogue': 1340,
'Financial': 580,
'Signal': 1418,
'IEEE': 763,
'KnockKnock': 865,
'Jokes': 832,
'BigData': 197,
'River': 1332,
'Publishers': 1249,
'Bump': 240,
'n': 1806,
'Airplane': 84,
'Toddler': 1585,
'Device': 436,
'Develop': 433,
'Beginning': 188,
'Counting': 358,
'ABCs': 51,
'Ideal': 767,
'Mueller': 1041,
'Report': 1308,
```

```
'Serious': 1394,
'Scientific': 1371,
'Absurd': 56,
'Hypothetical': 760,
'Making': 958,
'Sense': 1389,
'Women': 1710,
'God': 660,
'Unexpected': 1642,
'Path': 1140,
'Almost': 97,
'Steve': 1478,
'Jobs': 830,
'Expert': 541,
'predictive': 1817,
'modeling': 1804,
'Research': 1311,
'Industry': 785,
'Mockingjay': 1024,
'Giraffes': 648,
'Cognitive': 307,
'MIT': 946,
'Press': 1215,
'Night': 1068,
'fourth': 1778,
'School': 1368,
'Zone': 1734,
'Preschool': 1210,
'Workbook': 1717,
'Ages': 80,
'Colors': 313,
'Shapes': 1409,
'Numbers': 1081,
'Alphabet': 98,
'PreWriting': 1205,
'PreReading…': 1204,
'Sat': 1363,
'Exploring': 543,
'Zynq': 1736,
'MPSoC': 950,
'PYNQ': 1122,
'Cyber': 388,
'Includes': 782,
'Linux': 925,
'Hacking': 691,
'Kali': 839,
'Ethical': 521,
```

'Cybersecurity': 389,
'Fundamentals': 620,
'PlayerS': 1181,
'Handbook': 699,
'Dungeons': 476,
'Explore': 542,
'power': 1816,
'cloud': 1756,
'services': 1833,
'artificial': 1747,
'intelligence': 1792,
'ScikitLearn': 1374,
'Tensorflow': 1549,
'Manuscript': 968,
'Wright': 1723,
'Brothers': 234,
'Blue': 208,
'Truck': 1614,
'Divine': 453,
'Soul': 1450,
'Mind': 1011,
'Healing': 714,
'Transmission': 1605,
'System': 1525,
'Way': 1689,
'Heal': 713,
'Humanity': 749,
'Mother': 1037,
'Earth': 483,
'All…': 96,
'Train': 1602,
'Watchmen': 1686,
'Reasons': 1290,
'Team': 1540,
'Hackers': 690,
'Powerful': 1199,
'ShT': 1404,
'Dad': 394,
'Says': 1366,
'Grey': 680,
'Fifty': 578,
'Shades': 1406,
'Told': 1587,
'Christian': 290,
'Gone': 667,
'Introducing': 814,
'MLOps': 947,

'Scale': 1367,
'Enterprise': 512,
'Womens': 1711,
'Scientist': 1372,
'VNeck': 1657,
'Hyperbole': 758,
'Half': 693,
'Unfortunate': 1643,
'Situations': 1425,
'Flawed': 591,
'Coping': 356,
'Mechanisms': 993,
'Mayhem': 986,
'Matrix': 982,
'Concentration': 335,
'Inequalities': 786,
'Basics': 168,
'Understand': 1640,
'Salt': 1361,
'Acid': 62,
'Heat': 720,
'Elements': 500,
'Stories': 1485,
'Pop': 1187,
'N': 1047,
'Play': 1178,
'Center': 272,
'Xylophone': 1726,
'Station': 1468,
'Piano': 1164,
'Keys': 849,
'Colorful': 311,
'Balls': 163,
'Exciting': 537,
'Fun': 618,
'Batteries': 170,
'Required': 1310,
'18': 15,
'Moving': 1040,
'Fear': 568,
'Trump': 1617,
'White': 1696,
'Olive': 1094,
'Kitteridge': 862,
'China': 288,
'Nutrition': 1084,
'Ever': 524,

```
'Conducted': 338,
'Startling': 1465,
'Implications': 778,
'Diet…': 440,
'Azure': 153,
'Perform': 1157,
'largescale': 1796,
'advanced': 1739,
'Microsoft': 1004,
'Delivering': 424,
'Happiness': 703,
'Profits': 1232,
'Passion': 1137,
'Purpose': 1254,
'Astronomy': 135,
'Survey': 1513,
'Updated': 1650,
'Princeton': 1218,
'Modern': 1029,
'Observational': 1087,
'Credit': 374,
'Risk': 1331,
'Powered': 1198,
'Idea': 766,
'Product': 1230,
'Engineering': 508,
'Racketeer': 1276,
'teytoy': 1841,
'Soft': 1438,
'Nontoxic': 1076,
'Fabric': 551,
'Cloth': 302,
'Early': 482,
'Education': 494,
'Activity': 66,
'Crinkle': 377,
'Infants': 787,
'Shower': 1416,
'Pack': 1124,
'Getaway': 643,
'Badass': 160,
'Doubting': 464,
'Greatness': 678,
'Start': 1463,
'Service': 1397,
'Serverless': 1396,
'Inference': 788,
```

```
'Prediction': 1206,
'Morgan': 1034,
'Kaufmann': 841,
'Confession': 339,
'Investor': 819,
'Step': 1473,
'Hive': 737,
'HiveMall': 738,
'Lambda': 873,
'Solr': 1439,
'Kibana': 850,
'ETL': 481,
'Concise': 337,
'handson': 1785,
'course': 1759,
'full': 1779,
'code': 1757,
'examples': 1773,
'excel': 1774,
'Difficult': 442,
'Riddles': 1326,
'Smart': 1434,
'300': 28,
'Brain': 223,
'Teasers': 1541,
'Families': 559,
'Bioinformatics': 202,
'Churn': 292,
'Mobile': 1022,
'Telecommunications': 1547,
'Predictive': 1207,
'Worked': 1718,
'Examples': 533,
'Lord': 936,
'Fleas': 592,
'Hyperparameter': 759,
'optimization': 1809,
'neural': 1808,
'architecture': 1746,
'search': 1827,
'algorithm': 1740,
'selection': 1830,
'platforms': 1814,
'Human': 748,
'Benefits': 191,
…}
```

[156]:
```python
# encode document
title_bow = bow_transformer.transform(final_df["Name"])

print(title_bow)
```

```
  (0, 71)	1
  (0, 213)	2
  (0, 251)	1
  (0, 312)	1
  (0, 548)	1
  (0, 821)	2
  (0, 1393)	1
  (1, 5)	1
  (1, 8)	1
  (1, 43)	1
  (1, 158)	2
  (1, 281)	1
  (1, 359)	1
  (1, 545)	1
  (1, 621)	1
  (1, 697)	2
  (1, 853)	2
  (1, 890)	1
  (1, 921)	2
  (1, 1025)	2
  (1, 1032)	1
  (1, 1044)	1
  (1, 1093)	1
  (1, 1182)	1
  (1, 1414)	1
  :	:
  (620, 890)	1
  (620, 951)	1
  (620, 978)	1
  (620, 1369)	1
  (620, 1640)	1
  (621, 890)	1
  (621, 951)	1
  (621, 1259)	1
  (622, 770)	1
  (622, 1295)	1
  (622, 1313)	1
  (622, 1487)	1
  (622, 1514)	1
  (622, 1634)	1
  (622, 1681)	1
  (622, 1720)	1
```

```
  (623, 52)     1
  (623, 196)    1
  (623, 402)    1
  (623, 715)    1
  (623, 716)    1
  (623, 780)    1
  (623, 890)    1
  (623, 951)    1
  (623, 1113)   1
```

[157]:
```python
X = title_bow.toarray()
print(X)

X.shape # 624 seperate words in our dataset and 1855 rows
```

```
[[0 0 0 … 0 0 0]
 [0 0 0 … 2 0 2]
 [0 0 0 … 0 0 0]
 …
 [0 0 0 … 0 0 0]
 [0 0 0 … 0 0 0]
 [0 0 0 … 0 0 0]]
```

[157]: (624, 1855)

[158]:
```python
#TF-IDF ALgo -term frequency-inverse document frequencey to know the most␣
 ↪significant words

from sklearn.feature_extraction.text import TfidfTransformer
tfidf_transformer = TfidfTransformer().fit(title_bow)
print(tfidf_transformer)

title_tfidf = tfidf_transformer.transform(title_bow)
print(title_tfidf)   # got tfidf values for whole vocabulary
print(title_tfidf.shape)
```

```
TfidfTransformer()
  (0, 1393)     0.20576310236163137
  (0, 821)      0.6987841077166791
  (0, 548)      0.3134848159841625
  (0, 312)      0.2485876928747754
  (0, 251)      0.34939205385833955
  (0, 213)      0.33191810200550953
  (0, 71)       0.27757757810998546
  (1, 1854)     0.2830278941530326
  (1, 1852)     0.32752375487077073
  (1, 1597)     0.27417420359965133
  (1, 1447)     0.16376187743538537
```

```
  (1, 1414)	0.14693196781706777
  (1, 1182)	0.16376187743538537
  (1, 1093)	0.16376187743538537
  (1, 1044)	0.15391701141814326
  (1, 1032)	0.14693196781706777
  (1, 1025)	0.32752375487077073
  (1, 921)	0.32752375487077073
  (1, 890)	0.04865869582045847
  (1, 853)	0.22360031373445619
  (1, 697)	0.3078340228362865
  (1, 621)	0.1415139470765163
  (1, 545)	0.16376187743538537
  (1, 359)	0.16376187743538537
  (1, 281)	0.16376187743538537
  :	:
  (620, 214)	0.28599682587693365
  (620, 185)	0.2717028073361186
  (620, 118)	0.2749721582317761
  (620, 20)	0.28209429367852884
  (620, 0)	0.27842842816930163
  (621, 1259)	0.7828709650651016
  (621, 951)	0.4476893209072197
  (621, 890)	0.43207328545475826
  (622, 1720)	0.29547951518017485
  (622, 1681)	0.33536347816311757
  (622, 1634)	0.38808720936571717
  (622, 1514)	0.36475658664054494
  (622, 1487)	0.3160027325573323
  (622, 1313)	0.36475658664054494
  (622, 1295)	0.38808720936571717
  (622, 770)	0.36475658664054494
  (623, 1113)	0.45563299255007567
  (623, 951)	0.14027560147373927
  (623, 890)	0.13538259048725823
  (623, 780)	0.45563299255007567
  (623, 716)	0.42824172276898487
  (623, 715)	0.3710023603313775
  (623, 402)	0.21261590267925534
  (623, 196)	0.3151559376243899
  (623, 52)	0.2850069305126477
(624, 1855)
```

[159]:
```python
# importing Multinomial naive bayes
from sklearn.naive_bayes import MultinomialNB
model = MultinomialNB().fit(title_tfidf, final_df["Category"])
```

```
[160]: # giving the data to the model
all_predictions = model.predict(title_tfidf)
print(all_predictions)
```

```
['Others' 'Machine Learning' 'Others' 'Others' 'Machine Learning' 'Others'
 'Others' 'Machine Learning' 'Machine Learning' 'Others' 'Others' 'Others'
 'Others' 'Others' 'Machine Learning' 'Others' 'Machine Learning'
 'Machine Learning' 'Machine Learning' 'Machine Learning' 'Others'
 'Others' 'Others' 'Others' 'Machine Learning' 'Others' 'Others'
 'Machine Learning' 'Machine Learning' 'Others' 'Machine Learning'
 'Others' 'Others' 'Machine Learning' 'Machine Learning' 'Others' 'Others'
 'Others' 'Others' 'Machine Learning' 'Others' 'Machine Learning' 'Others'
 'Machine Learning' 'Machine Learning' 'Others' 'Others' 'Others'
 'Machine Learning' 'Machine Learning' 'Others' 'Others' 'Others'
 'Machine Learning' 'Machine Learning' 'Others' 'Machine Learning'
 'Others' 'Others' 'Machine Learning' 'Others' 'Others' 'Others'
 'Machine Learning' 'Others' 'Others' 'Machine Learning'
 'Machine Learning' 'Others' 'Machine Learning' 'Others'
 'Machine Learning' 'Others' 'Machine Learning' 'Machine Learning'
 'Others' 'Others' 'Others' 'Machine Learning' 'Others' 'Machine Learning'
 'Others' 'Others' 'Others' 'Machine Learning' 'Others' 'Others' 'Others'
 'Machine Learning' 'Machine Learning' 'Others' 'Others'
 'Machine Learning' 'Others' 'Others' 'Others' 'Others' 'Others' 'Others'
 'Machine Learning' 'Others' 'Others' 'Others' 'Machine Learning'
 'Machine Learning' 'Others' 'Others' 'Machine Learning' 'Others' 'Others'
 'Machine Learning' 'Others' 'Machine Learning' 'Others' 'Others'
 'Machine Learning' 'Machine Learning' 'Others' 'Others'
 'Machine Learning' 'Others' 'Machine Learning' 'Machine Learning'
 'Others' 'Others' 'Machine Learning' 'Others' 'Others' 'Others' 'Others'
 'Others' 'Machine Learning' 'Machine Learning' 'Others' 'Others' 'Others'
 'Machine Learning' 'Machine Learning' 'Others' 'Machine Learning'
 'Others' 'Machine Learning' 'Others' 'Others' 'Others' 'Others' 'Others'
 'Machine Learning' 'Others' 'Others' 'Machine Learning' 'Others'
 'Machine Learning' 'Others' 'Others' 'Machine Learning'
 'Machine Learning' 'Others' 'Machine Learning' 'Machine Learning'
 'Others' 'Others' 'Machine Learning' 'Others' 'Others' 'Machine Learning'
 'Others' 'Others' 'Machine Learning' 'Machine Learning' 'Others'
 'Machine Learning' 'Others' 'Machine Learning' 'Machine Learning'
 'Others' 'Others' 'Machine Learning' 'Others' 'Others' 'Others'
 'Machine Learning' 'Others' 'Machine Learning' 'Machine Learning'
 'Others' 'Others' 'Machine Learning' 'Others' 'Others' 'Machine Learning'
 'Machine Learning' 'Others' 'Others' 'Machine Learning' 'Others'
 'Machine Learning' 'Others' 'Machine Learning' 'Others'
 'Machine Learning' 'Machine Learning' 'Others' 'Others'
 'Machine Learning' 'Machine Learning' 'Machine Learning' 'Others'
 'Machine Learning' 'Others' 'Others' 'Machine Learning' 'Others' 'Others'
 'Others' 'Machine Learning' 'Others' 'Others' 'Others' 'Others'
```

```
'Machine Learning' 'Machine Learning' 'Others' 'Machine Learning'
'Machine Learning' 'Others' 'Others' 'Machine Learning'
'Machine Learning' 'Others' 'Others' 'Others' 'Machine Learning' 'Others'
'Machine Learning' 'Machine Learning' 'Machine Learning'
'Machine Learning' 'Others' 'Machine Learning' 'Others' 'Others'
'Machine Learning' 'Machine Learning' 'Machine Learning'
'Machine Learning' 'Others' 'Machine Learning' 'Machine Learning'
'Others' 'Machine Learning' 'Machine Learning' 'Machine Learning'
'Machine Learning' 'Others' 'Machine Learning' 'Others'
'Machine Learning' 'Machine Learning' 'Others' 'Machine Learning'
'Others' 'Machine Learning' 'Machine Learning' 'Machine Learning'
'Others' 'Machine Learning' 'Machine Learning' 'Others' 'Others' 'Others'
'Others' 'Machine Learning' 'Others' 'Machine Learning'
'Machine Learning' 'Others' 'Machine Learning' 'Machine Learning'
'Others' 'Others' 'Machine Learning' 'Others' 'Others' 'Machine Learning'
'Others' 'Others' 'Others' 'Others' 'Others' 'Others' 'Machine Learning'
'Others' 'Others' 'Machine Learning' 'Machine Learning' 'Others'
'Machine Learning' 'Others' 'Others' 'Others' 'Others' 'Machine Learning'
'Others' 'Machine Learning' 'Others' 'Others' 'Machine Learning' 'Others'
'Machine Learning' 'Machine Learning' 'Machine Learning' 'Others'
'Machine Learning' 'Machine Learning' 'Others' 'Others'
'Machine Learning' 'Others' 'Others' 'Others' 'Machine Learning' 'Others'
'Others' 'Others' 'Machine Learning' 'Others' 'Others' 'Machine Learning'
'Others' 'Machine Learning' 'Others' 'Others' 'Machine Learning' 'Others'
'Others' 'Machine Learning' 'Machine Learning' 'Others'
'Machine Learning' 'Machine Learning' 'Others' 'Machine Learning'
'Others' 'Others' 'Others' 'Others' 'Machine Learning' 'Others'
'Machine Learning' 'Others' 'Machine Learning' 'Machine Learning'
'Others' 'Machine Learning' 'Others' 'Machine Learning' 'Others' 'Others'
'Others' 'Others' 'Others' 'Machine Learning' 'Machine Learning' 'Others'
'Others' 'Others' 'Others' 'Others' 'Others' 'Others' 'Machine Learning'
'Others' 'Others' 'Machine Learning' 'Others' 'Others' 'Machine Learning'
'Machine Learning' 'Others' 'Others' 'Machine Learning' 'Others'
'Machine Learning' 'Machine Learning' 'Others' 'Others' 'Others'
'Machine Learning' 'Others' 'Machine Learning' 'Machine Learning'
'Others' 'Machine Learning' 'Machine Learning' 'Others'
'Machine Learning' 'Machine Learning' 'Machine Learning'
'Machine Learning' 'Others' 'Others' 'Others' 'Others' 'Others'
'Machine Learning' 'Others' 'Others' 'Machine Learning'
'Machine Learning' 'Others' 'Machine Learning' 'Others'
'Machine Learning' 'Machine Learning' 'Others' 'Others' 'Others' 'Others'
'Machine Learning' 'Others' 'Others' 'Machine Learning' 'Others' 'Others'
'Machine Learning' 'Others' 'Others' 'Others' 'Machine Learning'
'Machine Learning' 'Machine Learning' 'Others' 'Others' 'Others' 'Others'
'Machine Learning' 'Machine Learning' 'Others' 'Others'
'Machine Learning' 'Machine Learning' 'Machine Learning' 'Others'
'Others' 'Others' 'Machine Learning' 'Machine Learning' 'Others'
'Machine Learning' 'Machine Learning' 'Machine Learning' 'Others'
```

```
 'Others' 'Machine Learning' 'Machine Learning' 'Others'
 'Machine Learning' 'Machine Learning' 'Machine Learning' 'Others'
 'Others' 'Others' 'Others' 'Machine Learning' 'Others' 'Machine Learning'
 'Machine Learning' 'Others' 'Others' 'Others' 'Others' 'Others'
 'Machine Learning' 'Machine Learning' 'Machine Learning' 'Others'
 'Machine Learning' 'Machine Learning' 'Machine Learning' 'Others'
 'Others' 'Machine Learning' 'Machine Learning' 'Others' 'Others'
 'Machine Learning' 'Machine Learning' 'Others' 'Others' 'Others'
 'Machine Learning' 'Machine Learning' 'Machine Learning'
 'Machine Learning' 'Machine Learning' 'Others' 'Machine Learning'
 'Machine Learning' 'Machine Learning' 'Others' 'Others' 'Others'
 'Machine Learning' 'Machine Learning' 'Others' 'Machine Learning'
 'Others' 'Others' 'Others' 'Machine Learning' 'Machine Learning'
 'Machine Learning' 'Machine Learning' 'Others' 'Machine Learning'
 'Others' 'Machine Learning' 'Others' 'Others' 'Machine Learning'
 'Machine Learning' 'Others' 'Others' 'Others' 'Machine Learning'
 'Machine Learning' 'Others' 'Others' 'Others' 'Machine Learning' 'Others'
 'Others' 'Machine Learning' 'Others' 'Machine Learning' 'Others' 'Others'
 'Others' 'Machine Learning' 'Machine Learning' 'Others' 'Others' 'Others'
 'Others' 'Others' 'Machine Learning' 'Others' 'Others' 'Others' 'Others'
 'Machine Learning' 'Machine Learning' 'Others' 'Machine Learning'
 'Machine Learning' 'Machine Learning' 'Machine Learning' 'Others'
 'Others' 'Others' 'Machine Learning' 'Machine Learning' 'Others' 'Others'
 'Others' 'Others' 'Others' 'Others' 'Machine Learning' 'Others' 'Others'
 'Others' 'Others' 'Others' 'Others' 'Others' 'Others' 'Machine Learning'
 'Others' 'Machine Learning' 'Others' 'Machine Learning' 'Others' 'Others'
 'Machine Learning' 'Others' 'Machine Learning' 'Machine Learning'
 'Others' 'Machine Learning' 'Others' 'Machine Learning' 'Others'
 'Machine Learning' 'Machine Learning' 'Machine Learning' 'Others'
 'Others' 'Others' 'Others' 'Machine Learning' 'Machine Learning'
 'Machine Learning' 'Machine Learning' 'Machine Learning'
 'Machine Learning' 'Others' 'Machine Learning' 'Others' 'Others' 'Others'
 'Machine Learning' 'Machine Learning' 'Others' 'Others'
 'Machine Learning' 'Others' 'Others' 'Others' 'Machine Learning'
 'Machine Learning' 'Others' 'Machine Learning']
```

[161]:
```python
# printing the confusion matrix of our prediction
from sklearn.metrics import confusion_matrix

confusion_matrix(final_df["Category"], all_predictions)
```

[161]:
```
array([[273,   1],
       [  0, 350]], dtype=int64)
```

[ ]:
```

```